Dissertação de Mestrado

Giovani Prado Siqueira

# Modelagem e simulação de QoS em rede GPRS

2004

*Inatel*
*Instituto Nacional de Telecomunicações*

**MODELAGEM E SIMULAÇÃO**

**DE QOS EM REDE GPRS**

GIOVANI PRADO SIQUEIRA

Dissertação apresentada ao Instituto Nacional de Telecomunicações, como parte dos requisitos para obtenção do Título de Mestre em Engenharia Elétrica.

ORIENTADOR: Prof. Dr. Anilton Salles Garcia

Santa Rita do Sapucaí

2004

**FOLHA DE APROVAÇÃO**

Dissertação defendida e aprovada em 16/04/2004, pela comissão julgadora:

---

Prof. Dr. Anilton Salles Garcia / Instituto Nacional de Telecomunicações

---

Prof. Dr. José Marcos Câmara Brito / Instituto Nacional de Telecomunicações

---

Prof. Dr. Antônio Alfredo Loureiro / Universidade Federal de Minas Gerais

---

*Coordenador do Curso de Mestrado*

# DEDICATÓRIA

“Dedico este trabalho a todos aqueles que acreditam em seus ideais e fazem dos mesmos, instrumentos de contribuição efetiva para com a sociedade, tendo como meio os recursos tecnológicos e como fim o ser humano”

# AGRADECIMENTOS

Em primeiro lugar agradeço a Deus pela vida, capacidade e discernimento.

A minha eterna gratidão a meus pais Geraldo e Ivânia pelos sacrifícios realizados em prol de meus sonhos. Agradeço também a educação e valores os quais eu prezo.

A minhas irmãs Gislene, Gissélida, Giseli, Gisiane, meu irmão Geraldo, pelo apoio em todos os momentos da minha vida.

A minha noiva Renata que sempre acreditou no meu trabalho, agradeço todo amor, carinho e compreensão.

Ao Prof. Dr. Anilton Salles Garcia pela sua amizade e trabalho de orientação.

Ao Prof. Carlos Roberto dos Santos pelo apoio no Programa de Estágio Docente.

Faço um agradecimento especial a minha irmã Giseli e minha noiva Renata também pelo apoio financeiro, sem o qual não seria possível a realização deste trabalho.

A minha irmã Gissélida e meu amigo Humberto pelo suporte técnico.

Aos amigos Flávio, Wolney, André, Guilherme, Egídio, Cristian, Oscavo, Luciano, Sebastião, Ramon, por suas amizades e apoio.

Ao Prof. Dr. José Marcos Câmara Brito e Prof. Dr. Antônio Alfredo Loureiro por fazerem parte da banca examinadora dessa dissertação.

A CAPES pela bolsa de estudos.

Presto os meus sinceros agradecimentos a todos que direta ou indiretamente contribuíram para a realização deste trabalho.

# ÍNDICE

# LISTA DE FIGURAS

# LISTA DE TABELAS

# LISTA DE ABREVIATURAS E SIGLAS

**2G** - *Second Generation*
**3G** - *Third Generation*
**AF** - *Assured Forward*
**AGCH** - *Access Grant Channel*
**AM** - *Amplitude Modulation*
**AM-DSB** - *Double-Sideband Amplitude Modulation*
**AM-DSB-SC** - *Double-Sideband Suppressed Carrier Amplitude Modulation*
**AM-VSB** - *Vestigial-Sideband Amplitude Modulation*
**AP** - *Access Point*
**APN** - *Access Point Name*
**AQ** - *Assured Queue*
**ARPANET** - *Advanced Research Projects Agency Network*
**ASK** - *Amplitude Shift Keying*
**ATM** - *Asynchronous Transfer Mode*
**AUC** - *Authentication Center*
**BA** - *Behavior Aggregate*
**BB** - *Bandwidth Broker*
**BCCH** - *Broadcast Control Channel*
**BCS** - *Block Check Sequence*
**BE** - *Best Effort*
**BSC** - *Base Station Controller*
**BSS** - *Base Station Subsystem*
**BSSGP** - BSS GPRS *Protocol*
**BTS** - *Base Transceiver Station*
**CBQ** - *Class-based Queuing*
**CBR** - *Constraint-Based Routing*
**CDMA** - *Code Division Multiple Access*
**CGI+TA** - *Cell Global* ID *+ Time Advance*
**CoS** - *Class of Service*
**CPAGCH** - *Compact Packet Access Grant Channel*
**CPBCCH** - *Compact Packet Broadcast Control Channel*
**CPPCH** - *Compact Packet Paging Channel*
**CS** - *Coding Scheme*
**CSC** -*Class Selector Compliant*
**CSMA/CA** - *Carrier Sense Multiple Access with Collision Avoidance*
**CSMA/CD** - *Carrier Sense Multiple Access with Collision Detection*
**DARPA** - *Defense Advanced Research Projects Agency*
**DLCI** - *Data Link Connection Identifier*
**DNS** - *Domain Name Server*
**DS** - *Distribution System*
**DSCP** - *Diffserv Code-Point*
**DSSS** - *Direct Sequence Spread Spectrum*
**DWRR** - *Deficit Weighted Round Robin*
**EF** - *Express Forward*
**EGPRS** - EDGE-*based* GPRS
**EIR** - *Equipment Identity Register*
**EMS** - *Enhanced Message Service*
**ESS** - *Extended Service Set*
**ETSI** - *European Telecommunications Standards Institute*
**EV-DO** - *Evolution Data Only*
**EV-DV** - *Evolution Data and Voice*
**EY-NMPA** - *Elimination-Yield, Non Preemptive Multiple Access*
**FDD** - *Frequency Division Duplexing*
**FDMA** - *Frequency Division Multiple Access*
**FDP** - Função Distribuição de Probabilidade
**FEC** - *Forward Equivalence Class*
**FH** - *Frequency Hopping*
**FHSS** - *Frequency Hoped Spread Spectrum*
**FIFO** - *First In First Out*
**FM** - *Frequency Modulation*
**FQ** - *Fair Queuing*
**FSK** - *Frequency Shift Keying*
**FTP** - *File Transfer Protocol*
**GGSN** - *Gateway* GPRS *Support Node*
**GMM** - GPRS *Mobility Management*
**GMSC** - *Gateway* MSC
**GMSK** - *Gaussian Minimum Shift Keying*
**GPRS** - *General Packet Radio Service*
**GPRSS** - *General Packet Radio Service Simulator*
**GPS** - *Global Positioning System*
**GSM** - *Global System for Mobile Communication*
**GTP** - GPRS *Tunneling Protocol*
**HLR** - *Home Location Register*
**HSDPA** - *High Speed Downlink Packet Access*
**HTML** - *Hypertext Markup Language*
**HTTP** - *Hyper Text Trasnfer Protocol*
**IETF** - *Internet Engineering Task Force*
**ILR** - *Interworking Location Register*
**IMEI** - *International Mobile Equipment Identity*
**IMSI** - *International Mobile Subscriber Identity*
**IMT-2000** - *International Mobile Communication*-2000
**IS-2000** - *Interim Standard*
**ISM** - *Instrumentation, Scientific & Medical*
**ITU-T** - *International Telecommunication Union Telecommunication Standardization Sector*
**IWF** - *Interworking Function*
**JTACS** - *Japan* TACS
**KP** - *Kernel Procedures*

**LA** - *Location Area*
**LBS** - *Label Based Switching*
**LCS** - *Location Based Services*
**LLC** - *Logic Link Control*
**LMSI** - *Local Mobile Station Identity*
**LSP** - *Label Switching Path*
**LSR** - *Label Switching Routers*
**MF** - *Multi-field*
**MM** - *Mobility Management*
**MMS** - *Multimedia Message Service*
**MMSC** - *Multimedia Message Service Center*
**MPLS** - *Multi-Protocol Label Switching*
**MS** - *Mobile Station*
**MSC** - *Mobile-Services Switching Center*
**MSISDN** - *Mobile Subscriber Integrated Service Digital Network*
**MSRN** - *Mobile Station Roaming Number*
**MT** - *Mobile Terminal*
**NMT** - *Nordic Mobile Telephones System*
**NSAPI** - *Network layer Service Access Point Identifier*
**NSS** - *Network and Subsystem Switching*
**OFDM** - *Orthogonal Frequency Division Multiplexing*
**OSS** - *Operation and Support System*
**PACCH** - *Packet Associated Control Channel*
**PAGCH** - *Packet Access Grant Channel*
**PBCCH** - *Packet Broadcast Control Channel*
**PCCCH** - *Packet Common Control Channel*
**PCH** - *Paging Channel*
**PCU** - *Packet Control Unit*
**PCUSN** - *Packet Control Unit Support Node*
**PDA** - *Personal Digital Assistants*
**PDC** - *Personal Digital Cellular*
**PDCCH** - *Packet Dedicated Control Channels*
**PDCH** - *Packet Data Channel*
**PDN** - *Packet Data Network*
**PDP** - *Packet Data Protocol*
**PDTCH** - *Packet Data Traffic Channel*
**PDU** - *Packet Data Unit*
**PLL** - *Physical Link Layer*
**PLMN** - *Public Land Mobile Network*
**PM** - *Phase Modulation*
**PNCH** - *Packet Notification Channel*
**PPCH** - *Packet Paging Channel*
**PQ** - *Priority Queuing*
**PRACH** - *Packet Random Access Channel*
**PSK** - *Phase Shift Keying*
**PTCCH/D** - *Packet Timing advance Control Channel Downlink*
**PTCCH/U** - *Packet Timing advance Control Channel Uplink*
**PTM** - *Point-to-Multipoint*
**PTM-G** - *Point-to-Multipoint Group Call*
**PTM-M** - *Point-to-Multipoint Multicast*
**P-TMSI** - *Packet Temporary Mobile Subscriber Identity*
**PTP** - *Point-to-point*
**PTP-CONS** - *Point-to-point connections oriented network service*
**QAM** - *Quadrature Amplitude Modulation*
**QoS** - *Quality of Service*
**RA** - *Routing area*
**RACH** - *Random Access Channel*
**RED** - *Random Early Drop*
**RF** - *Radio Frequency*
**RFL** - *Radio Link Layer*
**RLC/MAC** - *Radio link Control/Medium Access Control*
**RSVP** - *Resource Reservation Protocol*
**S/R** - *Sinal/Ruído*
**SAP** - *Service Access Point*
**SDU** - *Service Data Unit*
**SGSN** - *Serving* GPRS *Supporting Node*
**SIM** - *Subscriber Identity Module*
**SLA** - *Service Level Agreement*
**SMS** - *Short Message Service*
**SMSC** - *Short Message Service Center*
**SMTP** - *Simple Mail Transfer Protocol*
**SNDCP** - *Sub-Network Dependent Convergence Protocol*
**TACS** - *Total Access Communications Systems*
**TBF** - *Temporary Block Flow*
**TCP** - *Transfer Control Protocol*
**TDD** - *Time Division Duplexing*
**TDMA** - *Time Division Multiple Access*
**TE** - *Terminal Equipment*
**TFI** - *Temporary Flow Identifier*
**TIA** - *Telecommunications Industry Association*
**TID** - *Tunnel Identifier*
**TLLI** - *Temporary Logical Link Identifier*
**TMSI** - *Temporary Mobile Station Identity*
**TOM** - *Type of Message*
**ToS** - *Type of Service*
**UDP** - *User Datagram Protocol*
**UMTS** -*Universal Mobile Telecommunications System*
**U-NII** - *Unlicensed National Information Infrastructure*
**URL** - *Universal Resource Locators*
**USF**- *Uplink State Flag*
**VLR** - *Visitor Location Register*
**VoIP** - *Voice Over IP*
**WAP** - *Wireless Application Protocol*
**WFQ** - *Weighted Fair Queuing*
**WI-FI** -*Wireless Fidelity*
**WLAN** - *Wireless Local Area Network*
**WPAN** - *Wireless Personal Area Network*
**WRR** - *Weighted Round Robin*
**WWAN** -*Wireless Wide Area Network*
**WML** - *Website Meta Language*
**XML** -*Extensible Markup Language*

# RESUMO

Siqueira, Giovani, P. "**Modelagem e Simulação de QoS em rede GPRS",** INATEL - Instituto Nacional de Telecomunicações Santa Rita do Sapucaí, Brasil, 2004.

Vislumbra-se num futuro próximo que diversas aplicações de tempo real baseadas em voz, vídeo e dados possam ser acessadas a partir de estações móveis suportadas por tecnologias de redes sem fio. Contudo ainda existem obstáculos para que tais aplicações sejam providas com qualidade de serviço satisfatória. O presente trabalho propõe um modelo de simulação que auxiliará na análise do desempenho de uma rede GPRS (*General Packet Radio Service*) a partir da implementação de diferentes mecanismos de provimento de QoS (*Quality of Service*). Tais mecanismos provêem um tratamento diferenciado ao tráfego de diversos tipos de mídia. Os resultados obtidos a partir de simulações do modelo proposto permitem uma avaliação comparativa entre os diversos mecanismos implementados (disciplinas de escalonamento: FIFO, PQ, WFQ, WRR e DWRR).

**Palavras-chave:** GPRS, QoS, FIFO, PQ, WFQ, WRR, DWRR.

# ABSTRACT

Siqueira, Giovani, P. "**Modelagem e Simulação de QoS em rede GPRS",** INATEL - Instituto Nacional de Telecomunicações, Santa Rita do Sapucaí, Brazil, 2004.

It is shimmered in a close future that several real-time applications based on voice, video and data can be accessed from mobile stations supported by wireless technologies. However they are obstacles to be overcame so that such applications are provided with satisfactory quality of service. The present work proposes the simulation model that will aid in the analysis of GPRS (General Packet Radio Service) network starting from the implementation of different mechanisms that provide QoS (Quality of Service). Such mechanisms provide the differentiated treatment to different types of traffic. The results obtained from simulations of the proposed model allow the comparative evaluation among the several implemented mechanisms (scheduling disciplines: FIFO, PQ, WFQ, WRR and DWRR).

**Keywords:** GPRS, QoS, FIFO, PQ, WFQ, WRR, DWRR.

# Capítulo I

## 1. Introdução

A possibilidade de se acessar informações e serviços a qualquer momento e em qualquer lugar está desenhando uma nova era nas comunicações. O acesso a informações de vídeo, voz e dados em tempo real com interatividade sobre uma plataforma móvel é uma realidade. Contudo, ainda existem alguns desafios, principalmente no que se refere à qualidade de serviço e otimização do desempenho.

A demanda por acesso sem fio tem crescido rapidamente e conquistado espaços que as redes convencionais levaram muitos anos para alcançar. Espera-se que num futuro próximo o acesso sem fio seja predominante e cobertura e *roaming* sejam globais.

A expansão das redes sem fio tem como principal propulsor a evolução da telefonia móvel celular. Em paralelo a esta evolução o número de aplicações do mundo IP (*Internet Protocol*) tem crescido de forma exponencial. Estas duas evoluções vislumbram uma convergência desde então chamada de Internet Móvel. A Internet Móvel combina a explosão do uso da Internet com a flexibilidade proporcionada pela mobilidade dos usuários das telecomunicações. Essa possibilidade de acessar informações e serviços a qualquer momento e em qualquer lugar está moldando uma nova sociedade da informação, que busca formas diferenciadas de acesso à Internet através de telefones celulares, PDAs (*Personal Digital Assistants*) e *laptops*. Entretanto, a Internet não foi concebida para lidar com as características inerentes ao ambiente onde vários usuários movimentam-se conectados à rede através de um enlace sem fio com baixa largura de banda. Pensando nisso, a infra-estrutura de comunicações móveis está evoluindo com o objetivo de dar suporte a aplicações de usuários com soluções fim-a-fim baseadas no protocolo IP. Mais capacidade está sendo proporcionada às redes móveis, com mais

largura de banda no enlace de rádio e a utilização de comutação por pacotes desde a interface aérea.

As aplicações de Internet Móvel abrangem todas as categorias de redes sem fio: WWAN (*Wireless Wide Area Network*), WLAN (*Wireless Local Area Network*) e WPAN (*Wireless Personal Area Network*).

O grande desafio para os próximos anos será o desenvolvimento de novas tecnologias e melhoramento das atuais a fim de viabilizar a comutação de pacotes sobre redes móveis sem fio para suportar novas aplicações.

## 1.1. Contextualização do trabalho

Embora exista uma diversidade de pesquisas relatadas na literatura sobre provimento de QoS em redes móveis, a implementação de mecanismos que a tornem realmente eficaz continua sendo um desafio. A implementação de tais mecanismos deve ocorrer em função das características da rede (tecnologia, dimensionamento, perfil do tráfego, etc.), o que requer um estudo prévio que seja consistente. Quais disciplinas de escalonamento de pacotes são mais adequadas para determinados perfis de tráfegos? Qual o dimensionamento ideal para recursos de rede? Tais perguntas ecoam no mundo da qualidade de serviço em redes sem fio. Assim, o desenvolvimento de um trabalho com o objetivo de enriquecer e ampliar as condições de análise desses mecanismos se torna uma contribuição valiosa para o meio científico.

Para tal desenvolvimento tem-se três alternativas possíveis: a implementação dos mecanismos de provimento de QoS em uma rede real, a construção de um modelo analítico ou a construção de um modelo de simulação. Por não dispor de uma rede real para tais experimentos, e por considerar que as avaliações ficam restritas a configuração implementada a primeira alternativa neste caso é considerada inviável. A construção de um modelo analítico baseado em desenvolvimentos matemáticos pode atingir resultados bastante precisos, contudo optou-se pela terceira alternativa

que se refere a construção de um modelo de simulação. Esta decisão foi tomada devido a flexibilidade na implementação de cenários proporcionada pelo uso de uma ferramenta de modelagem avançada.

O modelo de simulação proposto contribui com uma modelagem das principais funcionalidades dos protocolos do plano de transmissão da rede GPRS, tecnologia apontada como um dos principais agentes de transição entre a segunda e a terceira gerações da telefonia móvel celular. Contribui também com uma modelagem de mecanismos de provimento de QoS e com análises e conclusões obtidas a partir dos resultados das simulações.

Um modelo de simulação se torna uma ferramenta de análise eficiente quando utilizada em simulações cujos parâmetros de configuração representem as condições reais do ambiente de trabalho. No âmbito de provimento de QoS em redes móveis tal modelo contribui principalmente na análise comparativa entre os diversos mecanismos (disciplinas de escalonamento de pacotes).

Espera-se deste trabalho que, a partir dos resultados das simulações, seja possível identificar qual ou quais disciplina(s) de escalonamento de pacotes possibilita um melhor desempenho da rede quanto ao parâmetro atraso fim-a-fim. Espera-se, também, que o modelo de simulação sirva de ferramenta para o dimensionamento dos recursos da rede, o que tem influência direta na qualidade de serviço.

## 1.2 Descrição de trabalhos existentes

À medida que surgem novas aplicações e tecnologias, pesquisas para viabilizá-las quanto à QoS são realizadas.

Quanto ao provimento de QoS em redes GPRS existem vários trabalhos sobre reserva de recursos e otimização de desempenho, além de modelos com objetivos similares ao proposto neste trabalho.

Dentro da literatura relacionada é relevante mencionar trabalhos como o de Stuckmann [32] onde é apresentado o modelo GPRSim que permite um estudo sobre as tecnologias EGPRS e GPRS, bem como, uma avaliação do desempenho das mesmas. Outro trabalho interessante é apresentado por Halonen, Romero e Melero [31] onde é feita uma análise do desempenho das redes GRPS e EGPRS a partir de um modelo de simulação proposto. Há também o trabalho apresentado por Semeria [38] que se refere a um estudo de diversas disciplinas de escalonamento de pacotes. Contudo, estes trabalhos não focam a influencia das disciplinas de escalonamento na qualidade de serviço dentro de um ambiente de rede GPRS (proposta deste trabalho).

## 1.3 Organização do documento

O presente trabalho encontra-se estruturado em sete capítulos. O primeiro capítulo apresenta o panorama do ambiente onde o trabalho está contido e a importância de um trabalho desta natureza para o meio científico. O segundo capítulo apresenta uma visão geral das diversas tecnologias de redes sem fio existentes e os serviços providos pelas mesmas. O terceiro capítulo enfoca a descrição dos elementos e funcionalidades que compõem a tecnologia de rede GPRS. O quarto capítulo apresenta uma abordagem dos mecanismos utilizados para a reserva de recursos e otimização do desempenho de redes. As principais contribuições deste trabalho estão centradas no quinto e sexto capítulos. O quinto capítulo apresenta uma proposta de modelagem de simulação denominada GPRSS (*General Packet Radio Service Simulator*), assim como uma descrição de todos os níveis (rede, nó e processo) que constituem o mesmo. No sexto capítulo são realizados estudos de caso referentes a diversos cenários com o intuito de promover a validação do modelo de simulação proposto. No sétimo e último capítulo são apresentadas as conclusões finais obtidas a partir do desenvolvimento deste trabalho e sugestões para desenvolvimento de trabalhos futuros.

# Capítulo II

## 2. Redes Sem Fio

As redes sem fio utilizam a propagação de ondas eletromagnéticas para a transmissão de informações entre origem e destino. Esta utilização de ondas eletromagnéticas nas comunicações só foi possível após anos de estudos, experiências e descobertas.

No ano de 1805 Thomas Young demonstra que a luz é energia em forma de onda. Em 1831, Michael Faraday demonstra que a luz, eletricidade e magnetismo estão inter-relacionados. Em 1864, James Clerk Maxwell mostra a existência de ondas eletromagnéticas, velocidade, freqüência e comprimento de onda. Em 1887, Hertz prova que ondas eletromagnéticas podem ser transmitidas. Em 1897, Guglielmo Marconi transmite e recebe um sinal de rádio [1].

A onda eletromagnética é formada por um componente elétrico (campo elétrico E) e um componente magnético (campo magnético H) perpendiculares entre si (Figuras 2.1 e 2.2).

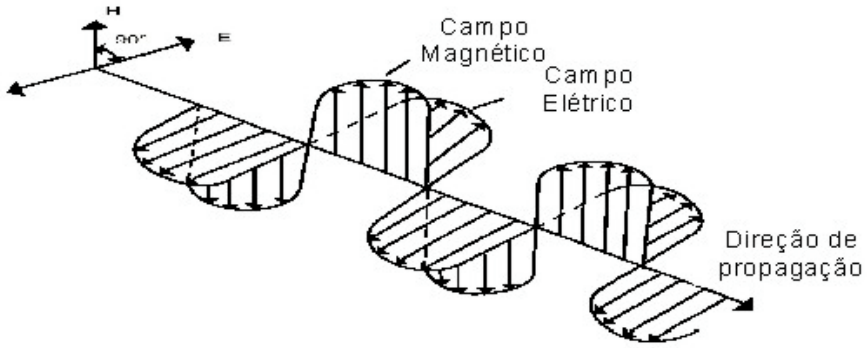


**Figura 2.1 –** *Onda Eletromagnética*

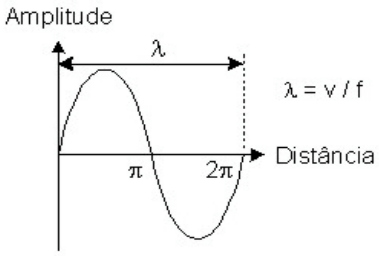


**Figura 2.2 -** *Parâmetros da onda*

A utilização das ondas eletromagnéticas para transmissão (voz ou dados) exige algum tipo de modulação, acesso e um planejamento de freqüências para que a transmissão seja bem sucedida e para que não haja interferência em transmissões simultâneas. Além disso, é necessário considerar os efeitos da propagação, tais como, atenuação por espaço livre, atenuação por obstáculo, efeitos da refração e reflexão.

## 2.1 Modulação

Existem dois tipos de modulação, a analógica e a digital. Pode-se citar como exemplo de modulação analógica a modulação AM (*Amplitude Modulation*) onde a variação da amplitude da portadora ocorre de acordo com a variação do sinal modulante. Este tipo de modulação tem algumas derivações, tais como, AM-DSB *(Double-Sideband Amplitude Modulation*), AM-DSB-SC (*Double-Sideband Suprisse Carrier Amplitude Modulatio*n) e AM-VSB (*Vestigial-Sideband Amplitude Modulation*). A modulação FM (*Frequency Modulation*) tem melhor qualidade que a AM por ser menos susceptível a ruídos. Neste tipo de modulação ocorre a variação de freqüência da portadora em função da variação do sinal modulante. A Modulação PM (*Phase Modulation)* tem como principal característica a variação da fase da portadora em função da variação do sinal modulante [2].

A modulação digital consiste em aplicar um sinal digital (binário) como sinal modulante da portadora ao invés de um sinal analógico como ocorre na modulação analógica. As modulações digitais mais usadas são ASK (*Amplitude Shift Keying*), FSK (*Frequency Shift Keying*), PSK (*Phase Shift Keying*) e QAM (*Quadrature*

A modulação é um mecanismo pelo qual um sinal de informação (sinal modulante) é deslocado para uma freqüência mais alta (portadora).

*Amplitude Modulation*). A modulação ASK consiste no chaveamento da amplitude da portadora como ilustrado na Figura 2.3.

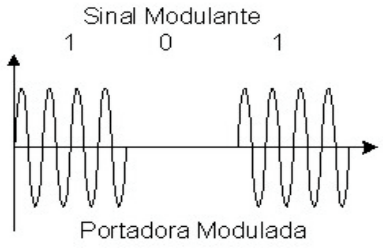


**Figura 2.3 -** *Modulação ASK*

A modulação FSK consiste no chaveamento de freqüências portadoras, como ilustrado na Figura 2.4. Esta é melhor que o ASK com relação à imunidade a ruídos, porém ocupa uma largura de faixa maior (analogamente ao AM e FM).

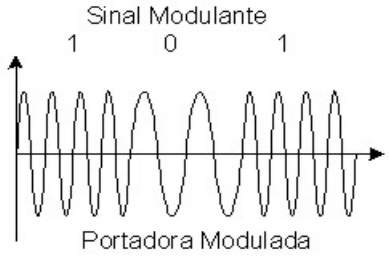


**Figura 2.4 -** *Modulação FSK*

A modulação PSK consiste no chaveamento da fase da portadora, como ilustrado na Figura 2.5. A largura de faixa deste tipo de modulação é igual a do ASK. A modulação PSK possui algumas derivações que são muito utilizadas, tais como QPSK, OQPSK, $\pi$/4DQPSK.

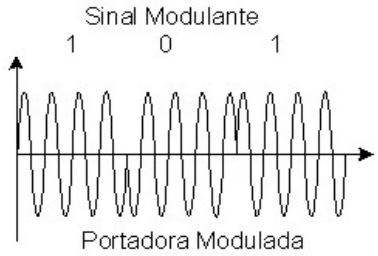


**Figura 2.5 -** *Modulação PSK*

Existe também uma combinação entre modulação ASK e PSK chamada de modulação em quadratura QAM.

O desempenho das técnicas de modulação está diretamente relacionado à largura de faixa (economia de espectro) e imunidade a ruídos (economia de potência). O ASK é pouco utilizado, pois apresenta problemas com ruídos e interferências. O FSK apresenta uma boa imunidade a ruído conseqüentemente uma baixa taxa de erro de bit, porém seu desempenho com relação à largura de faixa é muito baixo. O PSK e o QAM apresentam uma eficiência razoável em relação à largura de faixa e imunidade a ruído.

Algumas tecnologias de comunicação mais recentes fazem uso do espalhamento espectral como método de modulação do sinal a ser transmitido. Este método tem inúmeras vantagens, principalmente quanto à segurança das informações. As principais tecnologias são DSSS (*Direct Sequence Spread Spectrum*) e FHSS (*Frequency Hoped Spread Spectrum*). A técnica DS (*Direct Sequence*) utiliza códigos pseudo-aleatórios para modular a informação. Cada bit da informação é substituído por um código (seqüência de bits com a mesma duração de tempo). Então, se para transmitir o bit “1” em 1 segundo fosse necessário um sinal correspondente de 1 Hz, uma seqüência “10101110” durante um segundo necessitaria de uma banda de 8 Hz. Assim pode-se notar que ocorreu um espalhamento espectral. Cada bit da palavra código é denominado *chip* e a medida da taxa *chips*/segundo. Com a Técnica FH (*Frequency Hopping*) a portadora do sinal modulado muda de freqüência periodicamente. A informação é transmitida em um curto intervalo de tempo e então salta para uma outra freqüência onde é transmitida por um intervalo de tempo e depois salta novamente para outra freqüência [2].

## 2.2 Múltiplo Acesso

O acesso ao meio pode ser feito de várias formas através de algumas tecnologias comumente utilizadas, dentre elas pode-se citar o FDMA (*Frequency*

*Division Multiple Access)*, TDMA (*Time Division Multiple Access*), e CDMA (*Code Division Multiple Access*).

O FDMA disponibiliza um canal para cada usuário, ou seja, a banda de freqüências é dividida em sub-bandas menores que constituem os canais a serem alocados pelos usuários para a comunicação (Figura 2.6).

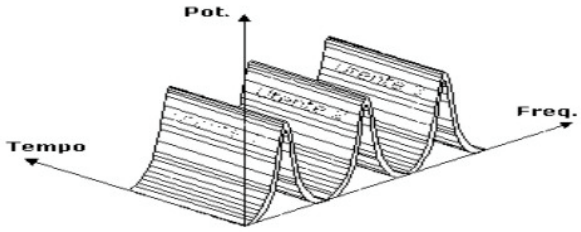


**Figura 2.6 -** *Acesso FDMA*

Apesar da simplicidade de implementação esta técnica não é viável atualmente devido à má utilização dos recursos se comparada a outras tecnologias de acesso.

O TDMA trabalha com o princípio do compartilhamento de canal por vários usuários, ou seja, os usuários têm a mesma banda de freqüência para transmissão, mas não a utilizam simultaneamente. O tempo de utilização do canal por usuário é chamado de *time slot* (Figura 2.7).

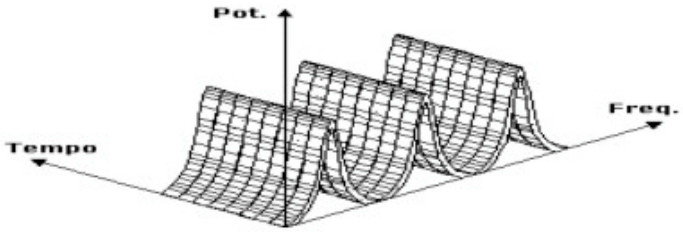


**Figura 2.7 -** *Acesso TDMA/FDMA*

O CDMA disponibiliza uma banda de freqüências para todos usuários e estes a utilizam simultaneamente (Figura 2.8). Nesta técnica, o espectro de freqüência do sinal de informação é espalhado (*spread*) através de códigos não correlacionados com o sinal. Como resultado a ocupação da banda é bem maior que a requerida. No receptor, através da correlação do código referente ao usuário com o sinal recebido é

possível obter a informação. Os sinais dos demais usuários são interpretados como ruídos pelo receptor uma vez que seus códigos não estão correlacionados com o sinal desejado. O espalhamento espectral pode ser feito usando DS ou FH [3].

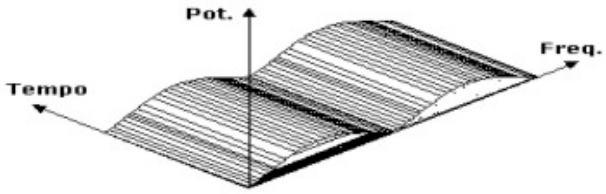


**Figura 2.8 -** *Acesso CDMA/FDMA*

A limitação de recursos também existe no CDMA, e é determinada pelo nível de interferência no sistema.

Na comunicação sem fio a separação do tráfego de transmissão do tráfego de recepção pode ser feita de duas formas, usando FDD (*Frequency Division Duplexing*) onde os sinais são separados na freqüência ou usando TDD (*Time Division Duplexing*) onde os sinais são separados no tempo.

## 2.3 Planejamento de Freqüência

O planejamento de freqüência tem as seguintes funções:

- Otimizar o uso do espectro da freqüência de rádio (Figura 2.9)

- Racionalizar a expansão

- Reduzir interferências entre sistemas.

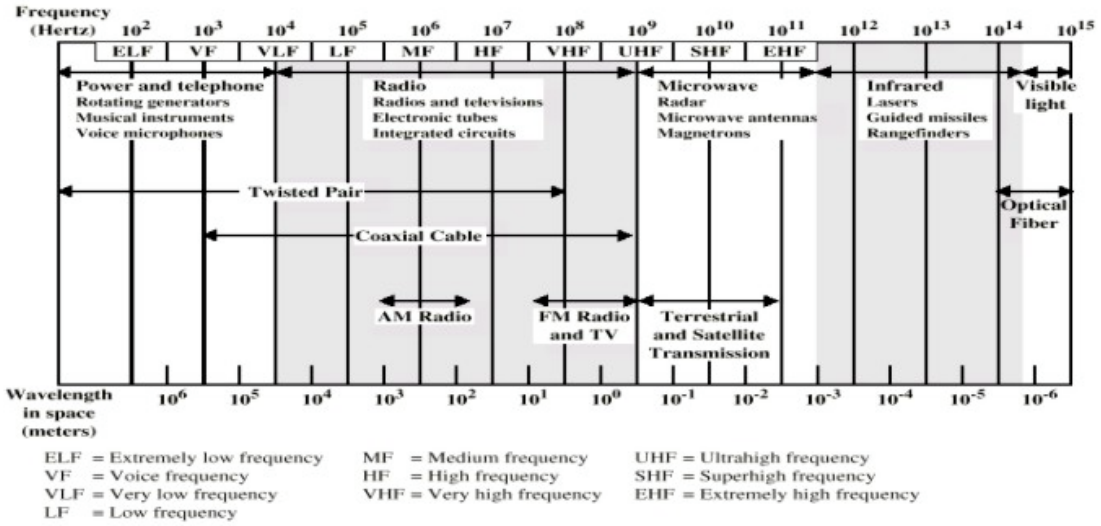


**Figura 2.9 -** *Espectro [4]*

O conceito de reuso de freqüência para aumentar a capacidade do sistema é muito utilizado, principalmente na telefonia móvel celular, o que requer um planejamento rigoroso da distribuição do espectro no *cluster*. Os padrões de reuso podem ser 4/12, 7/21 e 9/27 dentre outros (Figura 2.10) [5].

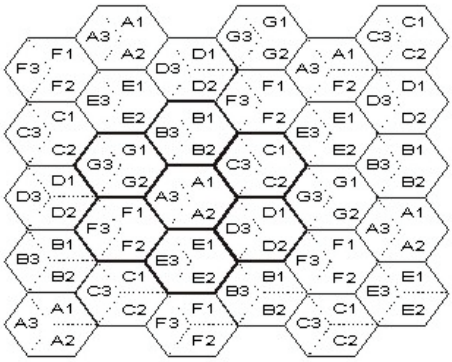

**Figura 2.10 -** *Reuso 7/21*

## 2.4 Características físicas

Os sistemas de comunicação sem fio podem ter restrições devido à interferência e ruídos diferentemente de um sistema fixo utilizando cabos ou fibras ópticas. As propriedades da onda de rádio são bastante dependentes da freqüência.

Nas freqüências baixas, as ondas podem passar através de obstáculos enquanto em freqüências mais altas o sinal pode sofrer reflexão ou ser absorvido por chuvas. A propagação por múltiplos percursos pode prover diferentes atrasos na recepção. Esses múltiplos percursos ocorrem devido à reflexão, difração ou refração ocasionados pelo espalhamento do sinal transmitido em estruturas no caminho até o receptor, tais como, prédios, montanhas, árvores, etc. A soma dos vários sinais recebidos pode resultar em uma interferência construtiva ou destrutiva do sinal recebido. Estes efeitos dos atrasos por múltiplos percursos também são conhecidos como espalhamento temporal ou "*Delay Spread*". O espalhamento temporal altera a amplitude das componentes do sinal transmitido. Esta alteração pode ocorrer em toda faixa de freqüência, caracterizando um desvanecimento plano, ou afetar somente uma faixa estreita caracterizando um desvanecimento seletivo. Um outro efeito importante é o efeito *Doppler* ocasionado pela percepção incorreta da freqüência por esta estar sendo recebido numa condição de movimento relativo entre transmissor e receptor. Quando no caminho entre transmissor e receptor existem grandes obstáculos ocorre um efeito denominado sombreamento, que afeta significativamente a recepção do sinal.

Em função de todas estas restrições, um ambiente de comunicações móveis apresenta características diferentes em relação a um sistema fixo, tais como, menor largura de banda, desconexões freqüentes (voluntária ou involuntária), taxa de erro do canal variável e dependente da localização [2].

## 2.5 Categorias de Redes Sem Fio

As tecnologias de rede sem fio podem ser agrupadas em três categorias principais denominadas WWAN, WLAN e WPAN. A classificação de uma tecnologia dentro de uma destas categorias ocorre em função de sua aplicação. Redes sem fio de pequena cobertura com aplicações de caráter pessoal são classificadas como WPAN. Redes usadas como extensão de redes locais em ambientes corporativos são classificadas como WLAN e redes com grandes áreas de cobertura e *roaming*, tais como na telefonia móvel celular são classificadas como WWAN.

### 2.5.1 WWAN

Dentre os sistemas classificados como WWAN o que mais tem conquistado espaço certamente é o de telefônica móvel celular. Os primeiros sistemas de telefonia móvel celular eram baseados no acesso FDMA. Vários padrões de telefonia móvel foram adotados em diferentes países e ficaram conhecidos como 1G (primeira geração), dentre eles pode-se citar o AMPS (*Advanced Mobile Phone System*) nos Estados Unidos, o TACS (*Total Access Communications Systems*) no Reino Unido, o JTACS (*Japan* TACS) no Japão e o NMT (*Nordic Mobile Telephones System*) nos países do norte Europeu (Dinamarca, Finlândia, Suécia e Noruega). Todos estes sistemas chamados de analógicos eram de baixa qualidade e baixa capacidade de canais, além de todas essas tecnologias serem incompatíveis entre si. As principais características destes sistemas eram a modulação analógica do sinal de voz em FM, sinalização feita com FSK, técnica de acesso FDMA, duplexação por divisão de freqüência FDD, tamanho das células de 0,5 km a 10 km e potência de transmissão do móvel de 1 a 8 Watts [3].

Apesar da baixa qualidade, a idéia de telefonia móvel celular foi bem aceita e impulsionou o desenvolvimento de novas tecnologias com intuito de aumentar a capacidade dos canais e melhorar a qualidade do sistema. Assim surgiu o 2G (segunda geração). O sistema AMPS evoluiu para D-AMPS (*Digital-AMPS* / IS-54, *Interim Standard-54*) que utiliza o TDMA como método de acesso, contudo este sistema não foi bem sucedido e então foi substituído pelo IS-136 que implementa algumas características novas. O IS-136 usa a mesma largura de banda de 30 kHz do AMPS, porém dividida em 6 *time slots*. Podem ser utilizados para tráfego de três usuários (*Full rate*) ou para tráfego de seis usuários (*Half rate*). A fim de desenvolver um padrão comum para atender os interesses dos países europeus, principalmente quanto à mobilidade entre países, foi desenvolvido o GSM (*Groupe Spéciale Mobile*) que mais tarde passou a se chamar GSM (*Global System for Mobile Communication*). O GSM também usa o TDMA, porém com características diferentes do IS-54 e do IS-136, portanto incompatíveis. O GSM usa uma largura de

faixa de 200KHz dividida em 8 *time slots*. Os *time slots* podem ser utilizados para tráfego de oito usuários (*Full rate*) ou para tráfego de dezesseis usuários (*Half rate*) [6]. O Japão desenvolveu neste mesmo período o PDC (*Personal Digital Cellular*). Um outro padrão ainda dentro do 2G foi o IS-95 que utiliza o método de acesso CDMA, porém sendo totalmente incompatível com os demais. A largura de banda do IS-95 é de 1,23 MHz utilizada por todos usuários [7] [11].

O momento atual representa uma transição entre o 2G e o 3G (terceira geração) que é denominada 2,5G. As principais tecnologias do 2,5G são evoluções das tecnologias usadas no 2G. O GPRS, por exemplo, consiste da utilização de comutação de pacotes sobre a estrutura de telefonia móvel celular GSM, onde ocorre a utilização dos *time slots* para tráfego de dados. O GPRS utiliza *time slots* que não estão sendo utilizados na transmissão de voz ou dados por comutação de circuitos para o tráfego de dados por comutação de pacotes (Figura 2.11) [8][9].

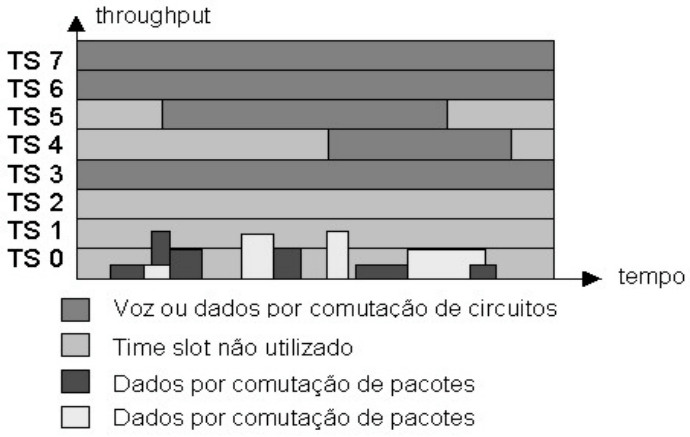


**Figura 2.11 –** *GPRS [8]*

O IS-2000 (CDMA 1x) também surgiu como uma evolução do IS-95 (Tabela2.1).

| CDMA One | | CDMA 2000 | | |
|---|---|---|---|---|
| IS-95-A<br>9.6 kbps | IS-95-B<br>64 kbps | IS-2000 1xRTT<br>144 kbps | IS-2000 1xEV-DO<br>600 kbps | IS-2000 1xEV-DV<br>2 Mbps |

2G → 3G

**Tabela 2.1 -** *Evolução CDMA [7]*

O CDMA 1xEV-DO (*Evolution Data Only*) utiliza uma portadora separada para a transmissão de dados (oferece uma taxa de pico em torno de 2,4 Mbps). O CDMA 1xEV-DV (*Evolution Data and Voice*) utiliza apenas uma portadora para o tráfego de dados e voz em taxa mais elevada. Existe também o padrão CDMA 3xRTT que utiliza três portadoras (3x 1xRTT), o que resulta em uma largura de banda três vezes maior [7].

Já no 3G, o objetivo principal é prover:

- Múltiplas taxas: 2 Mbps *indoor*, 384 kbps pedestre, 144 kbps em movimento.
- Múltiplos Serviços: Internet móvel, multimídia, serviços de pacotes e circuitos comutados.
- Múltiplas células: cobertura através de pico, micro e macro células.

Os principais desafios da terceira geração são:

- Convivência entre comutação de circuitos e comutação de pacotes.
- Tráfego assimétrico.
- Serviços de tempo real.
- Mobilidade de volume de tráfego para cada tipo de serviço.
- Qualidade de Serviço.

Apesar das redes 3G serem vistas, simplesmente, como uma evolução do 2G, na realidade constituem uma migração para uma nova arquitetura de redes. Os sistemas 3G são dotados de núcleo IP desenvolvido para transportar informações do tipo multimídia em altas velocidades e ainda prover QoS.

Apesar de inúmeras tecnologias terem sido propostas para a terceira geração apenas duas têm sido apontadas como principais tecnologias a serem utilizadas para este fim:

- UMTS - *Universal Mobile Telecommunications System* (WCDMA)
- IS-2000 (CDMA2000)

O 3G é denominado IMT-2000 (*International Mobile Communication-2000*), e se propõe a integrar voz, vídeo e dados em uma mesma plataforma móvel com QoS [10] [12].

Existe um movimento em direção à padronização e desenvolvimento de tecnologias 4G passando primeiramente pelo 3,5G. O HSDPA (*High Speed Downlink Packet Access*) é uma tecnologia 3,5G que permite taxas entre 8 e 10 Mbps por terminal de usuário. A quarta geração prove suporte a uma plataforma multimídia com alta taxa (≈ 100 Mbps).

### 2.5.2 WLAN

Uma WLAN é uma rede local sem fio, implementada como extensão ou alternativa para redes convencionais. Apresentam vantagens quanto à flexibilidade e custo se comparadas às redes fixas. As WLANs utilizam sinais de RF (*Radio Frequency*) ou infravermelho para a transmissão de dados, minimizando a necessidade de cabos de conexão dos usuários à rede. Desta forma, uma WLAN combina comunicação de dados com mobilidade dos usuários dentro da área de cobertura da rede. Além de redes locais, esta tecnologia pode ser utilizada para redes de acesso à Internet, que nestes casos são denominadas redes WI-FI (*Wireless Fidelity*) [13].

As arquiteturas utilizadas nas redes WLANs podem ser *Ad-Hoc* ou Infra-estruturada. Uma rede *Ad-Hoc* não possui uma estrutura pré-definida (Figura 2.12). Cada computador é capaz de se comunicar com qualquer outro. Este tipo de rede pode ser implementado através de técnicas de *broadcast* ou mestre escravo.

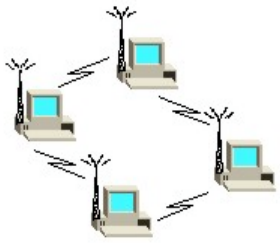

**Figura 2.12 -** *Ad-Hoc*

Numa rede do tipo Infra-estruturada os computadores se conectam a um elemento de rede central denominado AP (*Access Point*). Uma WLAN pode ter vários APs conectados entre si através de uma linha física, similar as redes celulares (Figura 2.13).

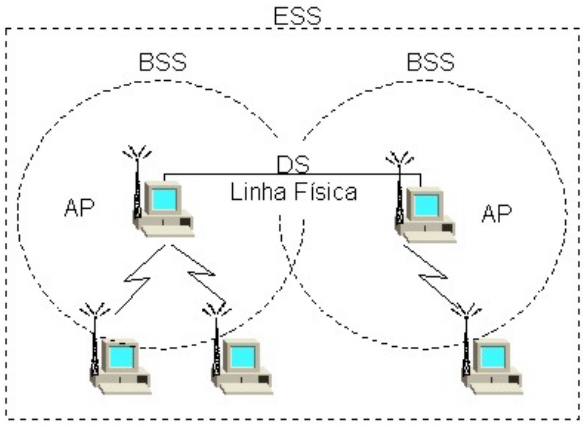


**Figura 2.13 -** *Infra-estruturada*

A área de cobertura de um AP onde se encontram as estações é denominada BSS (*Basic Service Set*) e corresponde a uma célula. O conjunto de BSS cujos APs estão conectados a uma mesma rede convencional é denominado ESS (*Extended Service Set*). Nestas condições, uma estação pode se movimentar de uma célula BSS para outra, permanecendo conectada à rede. Este processo é denominado *Roaming*. A estrutura que permite a comunicação entre os APs é chamada de DS (*Distribution System*) [14].

As principais tecnologias de redes WLAN são IEEE 802.11 e HiperLan. Dependendo da tecnologia utilizada, a transmissão de sinais de RF em redes WLANs pode ser realizada em duas categorias de bandas de freqüência:

- ISM – As Bandas ISM (*Instrumentation, Scientific & Medical*), compreendem três segmentos do espectro (902 a 928 MHz, 2.400 a 2.483,5 MHz e 5.725 a 5.850 MHz) reservados para uso sem a necessidade de licença.

- U-NII – *Unlicensed National Information Infrastructure*: Esta banda foi criada pelo FCC nos Estados Unidos, sem exigência de licença, para acesso à Internet, e compreende o segmento de freqüências entre 5.150 e 5.825 MHz.

O IEEE desenvolveu uma série de padrões para redes de transmissão de dados sem fio. O padrão IEEE 802.11 foi o primeiro a ser desenvolvido e permite taxas de transmissão brutas de 1 até 2 Mbps nas bandas ISM. O padrão evoluiu da seguinte forma:

- IEEE 802.11a: permite atingir taxas de transmissão de até 54 Mbps na banda de 5 GHz, utilizando a técnica OFDM (*Orthogonal Frequency Division Multiplexing*).

- IEEE 802.11b (*Wi-Fi*): permite taxas de transmissão brutas de até 11 Mbps nas bandas ISM.

- IEEE 802.11g: permite alcançar velocidades de transmissão de até 54 Mbps e exige uma regulamentação específica para seu funcionamento sem licença.

- IEEE 802.16: permite velocidades de até 54 Mbps na banda U-NII. Esta recomendação também exige o atendimento a uma regulamentação específica que limita a potência das estações transmissoras, mas não exige o uso de espalhamento de espectro.

Dentre os padrões indicados acima o IEEE 802.11b é o mais utilizado. Este padrão define as camadas Física e MAC (*Medium Access Control*) para redes sem fio. A camada Física especifica duas técnicas possíveis para transmissão, a transmissão por RF (2.4 - 2.4835 GHz) e por Infravermelho (300 - 428 GHz). A

modulação pode ser do tipo DSSS ou FHSS. A faixa de freqüências disponível, 2.4 - 2.4835 GHz (83,5 MHz) permite acomodar até 3 canais WLAN sem sobreposição (Figura 2.14). Ou seja, num mesmo espaço físico podem ser estabelecidos até três comunicações simultâneas sem interferência.

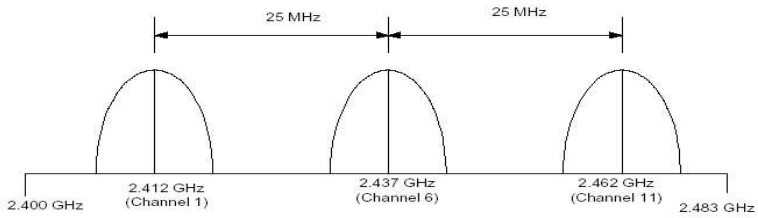


**Figura 2.14 -** *Canais IEEE 802.11b [14]*

O mecanismo de acesso básico é o CSMA/CA (*Carrier Sense Multiple Access with Collision Avoidance*) com intervalo *backoff* exponencial similar ao usado no IEEE 802.3 com CSMA/CD (*Carrier Sense Multiple Access with Collision Detection*).

O CSMA/CA pode ser resumido como segue:

1) Computador escuta o meio antes de transmitir.

2) Se o meio estiver ocupado um contador inicia uma contagem aleatória de espera.

3) A cada intervalo que ele verifica que o meio está livre ele decrementa o contador. Se o meio não estiver livre, então ele não decrementa.

4) Quando o contador atinge zero a estação transmite o pacote.

O HIPERLAN também é uma tecnologia WLAN e é um padrão que foi desenvolvido pelo ETSI (*European Telecommunications Standards Institute*). Existem várias versões para as redes HIPERLAN conforme mostrado na Tabela 2.2. Esta utiliza como o método de acesso o EY-NMPA (*Elimination-Yield, Non Premptive Multiple Access*), similar ao CSMA, porém com uma fase de priorização de pacotes. Os critérios são prioridade de usuário e tempo de vida dos pacotes [15][16].

| HIPERLAN | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| Aplicação | WLAN | ATM | WLL | ATM |
| Freqüência | 5.1 - 5.3 GHZ | | | 17.2 - 17.3 Ghz |
| Topologia | Ad-Hoc e Infra-estruturada | Celular | Ponto a multiponto | Ponto a ponto |
| Antena | Ominidirecional | | Direcional | |
| Cobertura | 50m | 50-100m | 5000m | 150m |
| QoS | Estatística | Classes de Tráfego ATM (VBR, CBR, ABR, UBR) | | |
| Mobilidade | <10m/s | | Estacionário | |
| Interface | LAN | ATM | | |
| Taxa (Mbps) | 23.5 | > 20 | | 155 |
| Controle de potência | Sim | | Não é necessário | |

**Tabela 2.2 -** *Tecnologias HIPERLAN*

## 3.5.3 WPAN

A categoria WPAN permite aos usuários estabelecer uma rede *Ad-Hoc* com dispositivos tais como PDAs, telefone celulares, *laptops* e eletroeletrônicos domésticos usados em um espaço de operação pessoal com uma cobertura aproximada de 10 m. As principais tecnologias WPANs são *Bluetooth* e *HomeRF*.

*Bluetooth* é um padrão para comunicação sem fio, de curto alcance (< 10 m) e baixo custo, estabelecido por meio de conexões de rádio e uma arquitetura *Ad-Hoc*. O *Bluetooth* opera na faixa de 2400 MHz a 2483,5 MHz. A modulação utilizada é 2FSK com espalhamento espectral por salto de freqüência FHSS. A largura de banda do canal é de 1 MHz e dentro desta faixa são definidas 79 portadoras nas quais instantaneamente um dispositivo pode estar transmitindo em uma seqüência pseudo-aleatória. A seqüência particular de freqüências de um canal é estabelecida pelo dispositivo mestre da *piconet* que é o responsável pelo controle do canal. Todos os outros dispositivos participantes da *piconet* são escravos e devem se sincronizar ao mestre (Figura 2.15). O dispositivo mestre muda sua freqüência de transmissão 1600 vezes por segundo com o objetivo de minimizar potenciais interferências. Um canal é dividido em *time slots* de 625 µs. O *Bluetooth* utiliza o método de *Polling* para evitar colisões devido a múltiplas transmissões de dispositivos escravos. Deste modo, somente o dispositivo indicado no *slot* mestre-para-escravo pode transmitir no *slot*

escravo-para-mestre seguinte. A Potência de transmissão é de 1 mW (0 dBm até 20 dBm com controle de potência) e a sensibilidade de recepção de –70 dBm [17].

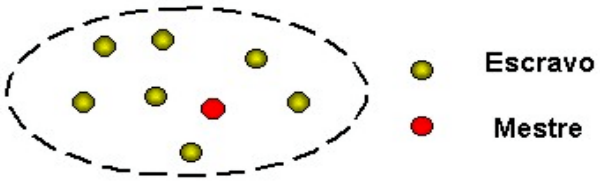


**Figura 2.15** - *PicoNet*

Tipicamente, nas aplicações *Bluetooth*, várias *piconets* independentes e não-sincronizadas podem se sobrepor ou existir na mesma área. Neste caso, forma-se um sistema *Ad-Hoc* disperso denominado *scatternet*, composto de múltiplas redes, cada uma contendo um número limitado de dispositivos (Figura 2.16) [18].

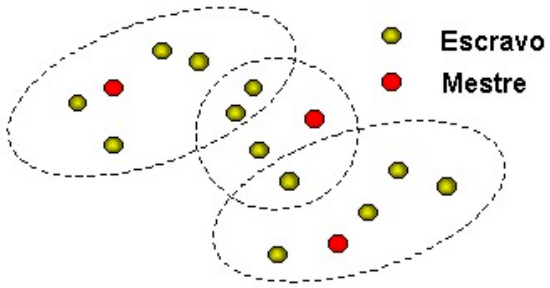


**Figura 2.16** - *Scatternet*

O *HomeRF* é mais um padrão de redes sem fio que utiliza a faixa dos 2.4 GHz. A idéia original era que o *HomeRF* fosse um padrão de redes sem fio de baixo custo, o que não se concretizou. O *HomeRF* utiliza um protocolo chamado *Shared Wireless Access Protocol*, onde as interfaces de rede se comunicam diretamente, sem o uso de um ponto de acesso. O *HomeRF* opera na faixa de 2400 MHz a 2483,5 MHz (Tabela 2.3). A modulação utilizada é 2FSK / 4FSK com espalhamento espectral por salto de freqüência FHSS (50 saltos/s). A largura de banda do canal é de 1 MHz (V1) e 5 MHz (V2). A Potência de transmissão é de 100 mW (20 dBm) e a sensibilidade de recepção de –80 dBm. A área de cobertura máxima é de 50 m. O *HomeRF* permite a conexão de até 127 dispositivos por rede [19].

| Atributos | HomeRF 1.0 | HomeRF 2.0 |
|---|---|---|
| Banda de Freqüência | 2,4 GHz ISM | 2,4 GHz ISM |
| Método de acesso | FHSS | FHSS |
| Máx. Taxa | 1,6 Mbps | 10 Mbps |
| *Throughput* (Usuário) | 650 kbps | 5 Mbps |
| Faixa *Indoor* | < 50m | < 50m |
| Potência de transmissão | 100mW | 100mW e 500mW |
| Priorização de fluxos | Não | < 8 fluxos simultâneos |
| Qualidade no suporte a voz | 4 handsets | 8 handsets |
| *Roaming* | Não | Sim |
| Criptografia da MAC | 40 bits | 128 bits |
| Consumo mínimo (*Standby*) | < 10 mW | < 10 mW |

**Tabela 2.3 -** *Tecnologias HomeRF*

O IEEE desenvolveu vários padrões para WPAN especificados como 802.15. Existem cinco grupos de trabalho responsáveis pelas padronizações:

- IEEE 802.15.1 (TG1-*Task Group 1*) – Este padrão é uma revisão e adaptação das especificações do *Bluetooth* V1.1.

- IEEE 802.15.2 TG2 (*Task Group 2*) – Este padrão recomenda práticas e desenvolve mecanismos para a coexistência de WPAN e WLAN.

- IEEE 802.15.3 TG3 (*Task Group 3*) – Este padrão define a utilização de WPAN com taxas de transmissão elevadas (11, 22, 33, 44 e 55 Mbps).

- IEEE 802.15.3a TG3a (*Task Group 3a*) – Este padrão define taxas elevadas para aplicações que envolvam imagem e multimídia.

- IEEE 802.15.4 TG4 (*Task Group 4*) – Este padrão define a utilização de baixas taxas de transmissão com vida útil de baterias longa e baixa complexidade.

Um outro padrão de comunicação sem fio é o IrDA (*Infrared Data Association*). O IrDA especifica um sistema de transmissão sem fio ponto a ponto utilizando luz infravermelha. Esta tecnologia é de baixo custo e pode operar com taxas de até 4 Mbps, porém o seu alcance é pequeno.

## 2.6 Serviços

As tecnologias só existem e o seu desenvolvimento só faz sentido se estas tiverem a capacidade de fornecer algum tipo de serviço aos usuários. Os serviços podem ser dos mais variados, porém o tráfego que o serviço gera pode ser:

- Tráfego contínuo com taxa constante (Figura 2.17)
- Tráfego contínuo com taxa variável (Figura 2.18)
- Tráfego em rajadas (Figura 2.19)

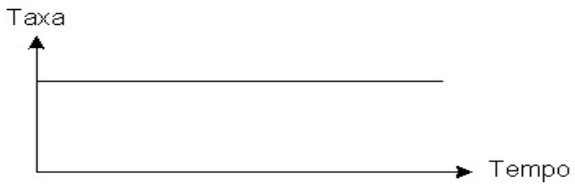


**Figura 2.17 -** *Tráfego contínuo com taxa constante*

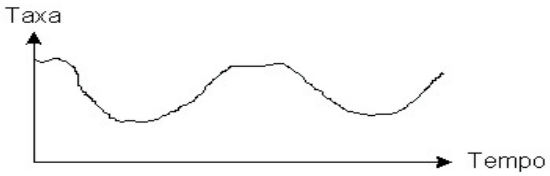


**Figura 2.18 -** *Tráfego contínuo com taxa variável*

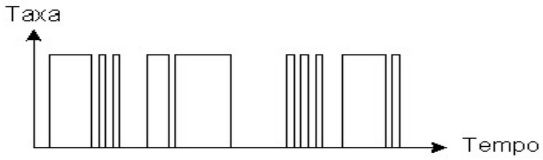


**Figura 2.19 -** *Tráfego em rajadas*

A informações são basicamente de quatro tipos:

- Texto – A informação do tipo Texto é caracterizada como tráfego de rajadas. Com exceção de aplicações de tempo real, o retardo máximo e a variação estatística não representam grandes preocupações, contudo não são permitidos erros ou perdas de pacotes.

- Imagem Gráfica – Assim como no texto esta é caracterizada como tráfego de rajadas. O atraso e variação estatística não representam um grande problema, porém também não suporta erros ou perdas de pacotes.

- Áudio – A informação do tipo Áudio provê um tráfego contínuo com taxas constantes. Em aplicações de tempo real o atraso e a variação estatística representam uma grande preocupação, porém o erro ou perda de pacotes nem tanto.

- Vídeo – A informação do tipo Vídeo (sem compressão) é caracterizada como tráfego contínuo com taxas constantes e assim como no áudio em aplicações de tempo real o atraso e a variação estatística representam uma grande preocupação.

Considerando que a maioria das aplicações de áudio são aplicações de tempo real, então qualquer atraso se torna crítico. Para amenizar este problema é necessário utilizar mecanismos de QoS para alocação de recursos e priorização de pacotes ou otimizar o desempenho da rede.

Existem várias classes de QoS que podem ser definidas:

- Classe de conversação – esta classe é ideal para aplicações de tempo real onde o atraso deve ser minimizado. Nessa classe o tráfego tem características simétricas e requer um intervalo constante entre pacotes.

- Classe de fluxo – nessa classe o fluxo de dados permanece constante, ou seja, a intervalo entre os pacotes é constante. O atraso não é tão essencial porque neste caso não há tantas interações caracterizando um fluxo assimétrico (áudio, vídeo e texto).

- Classe interativa – existe uma constante troca de informações entre usuário e a rede, mas o tempo não é tão critico quanto na classe de conversação. As aplicações deste tipo de serviço podem ser uma informação regular de busca com um *browser*, aplicações de *chat*, jogos

ou aplicações baseadas em localização. Basicamente o usuário requer uma informação e o servidor responde com a informação apropriada.

- Classe de *background* – Para as aplicações onde o tempo não é critico esta classe é apropriada. O atraso pode ser de segundos e até de minutos dependendo da carga da rede, contudo o custo é bem inferior às demais classes descritas. (*e-mail*, *download* de novas versões de aplicativos, etc).

A QoS em redes sem fio tem agravantes se comparadas às redes fixas:

- Enlaces sem fio provêem menos largura de banda que enlaces com fio.
- Enlaces sem fio são menos imunes a ruídos e sujeitos ao desvanecimento e interferências.
- À medida que os usuários se movimentam, a rede deve rastreá-los e descobrir suas novas localizações a fim de efetuar a entrega de dados.
- No caso de *handoff* a mesma largura de banda deve ser alocada na nova localização, o que nem sempre é possível.

O serviço mais utilizado mundialmente é o de telefonia que consiste em disponibilizar um canal de comunicação de voz entre dois pontos ou mais no caso de uma conferência. O serviço de telefonia surgiu no século XIX, mas evoluiu e ganhou dimensões extraordinárias de penetração na segunda metade do século XX. A transmissão de voz sempre foi utilizada com comutação de circuitos, o que permitia uma boa qualidade do sinal, porém desperdício de recursos da rede (Figura 2.20). Com o advento da comutação de pacotes, o sinal de voz é convertido em pacotes que compartilham o meio com pacotes de outros serviços, tais como, dados e imagens podendo gerar atrasos, erros e perdas.

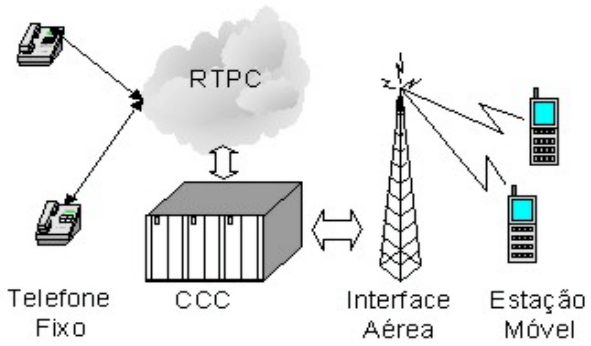


**Figura 2.20 -** *Arquitetura para serviço de voz*

O serviço SMS (*Short Message Service*), surgiu na Europa em 1991, sendo utilizado no padrão de telefonia móvel celular de segunda geração. Este serviço consiste no envio de mensagens de texto ponto a ponto entre estações móveis de até 160 bytes. Como as informações são transmitidas pelo canal de controle, então é possível enviar ou receber mensagens durante uma conversação. O serviço utiliza um centro de SMS (*SMSC – Short Message Service Center*) que recebe as mensagens da fonte e a encaminha para o destino (*store-and-foward*) (Figura 2.21). Uma característica importante deste serviço é que ele permite a entrega garantida da mensagem [20].

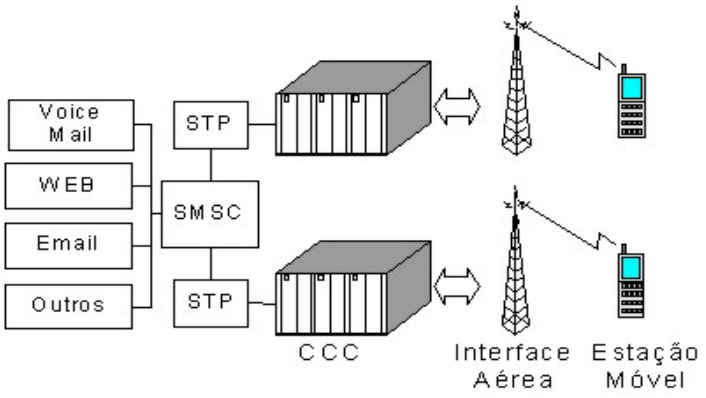


**Figura 2.21-** *Arquitetura para serviço de SMS*

As aplicações da tecnologia SMS são:

- Notificação: mensagem de voz ou fax que indica a presença de mensagens nas caixas postais de voz ou *e-mail*, serviços de calendários e lembretes.

- Informação: informações de tempo, tráfego, financeiras e entretenimento.

O EMS (*Enhanced Message Service*) adiciona ao serviço SMS de transmissão de texto, imagens em preto e branco e sons simples (ex: tons de campainha). O EMS é implementado bastando apenas um simples *upgrade* na infra-estrutura da rede. Um aparelho com funções SMS pode receber mensagens de texto de um outro com EMS, mas não imagens e sons.

O MMS (*Multimedia Message Service*) é uma evolução do EMS e representa um padrão universal que permite aos usuários de telefones móveis enviar e receber mensagens com texto, gráficos, fotografias, áudio e *video-clips* via canal de informação. Vídeo, áudio e imagens de alta qualidade podem ser baixados pelo telefone dos sites WAP (*Wireless Application Protocol*) ou serem transmitidos/recebidos pelo telefone através de um acessório acoplado (ex. câmera digital) utilizando uma mensagem de MMS. As mensagens de MMS podem ser enviadas para outros telefones móveis ou para um endereço de *e-mail*. Estes tipos de mídia também podem ser armazenados no aparelho móvel para uso posterior. MMS suporta formatos padrões de imagem tais como o GIF e o JPEG, formatos de vídeos tais como MPEG4 e formatos de áudio tais como MP3 e MIDI. As mensagens multimídia requerem elevadas taxas de transmissão, e o transporte é feito via WAP, assim este serviço se torna independente da tecnologia (2,5G ou 3G). Assim como no SMS, o MMS requer um elemento de rede que armazene a mensagem até que o receptor seja localizado, este elemento é o MMSC (*Multimedia Message Service Center*) (Figura 2.22) [21].

O WAP é um padrão para aplicações de ambientes sem fio desenvolvido pelo WAP Forum em 1997 e pode ser suportado por qualquer tecnologia. O principal objetivo do WAP é prover, sobre uma plataforma móvel (telefones celulares digitais, *PDAs-Personal Digital Assistants* e *Laptops*) aplicações de Internet (Páginas da WEB) e outros serviços de dados.

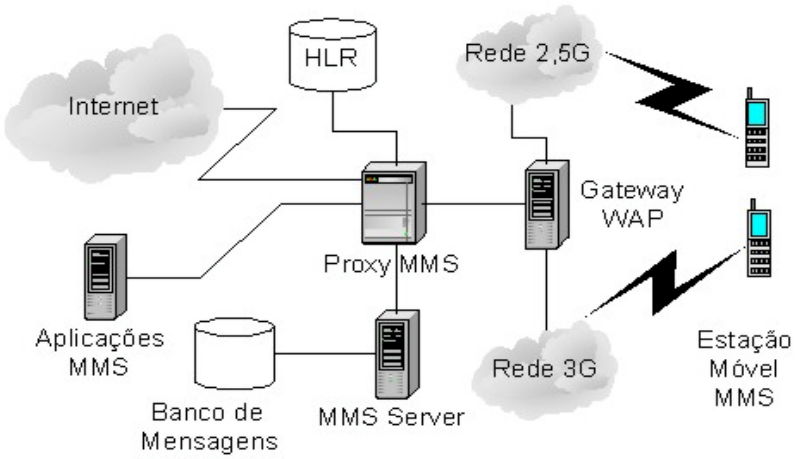


**Figura 2.22 -** *Arquitetura MMS*

O WAP baseia-se no modelo da Internet, e foi otimizado para levar em consideração as restrições de um ambiente sem fio. Todo conteúdo é especificado em formatos similares aos formatos padrões da Internet. O conteúdo é transportado usando protocolos padrões da Internet, domínio WWW e um protocolo otimizado similar ao HTTP (*Hiper Text Transfer Protocol)*, no domínio sem fio. Este modelo permite que todos os conteúdos e serviços sejam hospedados em servidores de aplicações tradicionais. Todo o conteúdo é localizado usando URLs (*Universal Resource Locators*) padrões. A arquitetura WAP utiliza um dispositivo *gateway* WAP entre o cliente e o servidor conforme ilustrado na Figura 2.23.

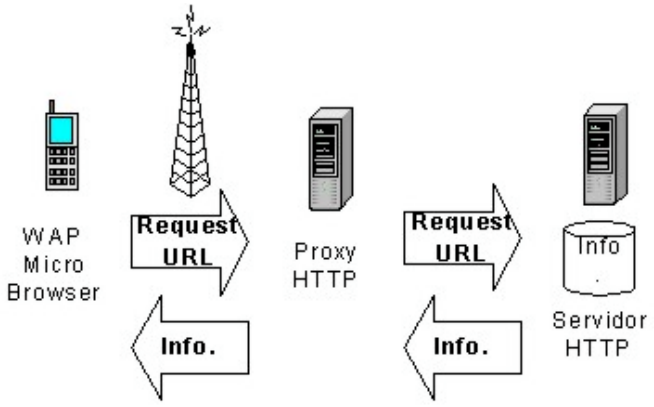


**Figura 2.23 -** *Modelo WAP*

Para viabilizar a navegação o WAP utiliza o WML (*Website Meta Language*) que é baseada no HTML (*Hypertext Markup Language*) padrão e é especificada como um tipo de documento XML (*eXtensible Markup Language*).

O IP Móvel é uma proposta da IETF (*Internet Engineering Task Force*) como solução para prover mobilidade na camada de rede para usuários móveis da Internet, assim ampliando a gama de serviços oferecidos aos usuários móveis. O IP Móvel permite que os usuários móveis constituam suas comunicações enquanto se locomovem de um ponto de acesso a outro na Internet. O IP Móvel define duas entidades para prover suporte à mobilidade: um *Home Agent* e um *Foreign Agent*. O *Home Agent* é atribuído estaticamente à estação móvel e baseia-se no endereço IP *home* permanente da estação móvel. O *Foreign Agent* é atribuído à estação móvel, baseando-se na localização atual da estação móvel. O *Foreign Agent* tem associado consigo um endereço IP chamado *Care-of-address*. Pacotes destinados para a estação móvel são interceptados pelo *Home Agent*, encapsulados e enviados para o *Care-of-Address*. O *Foreing Agent* desencapsula os pacotes e encaminha-os diretamente para a estação móvel (Figura 2.24). Portanto o *Foreing Agent* é a estação mais próxima do IP Móvel [2].

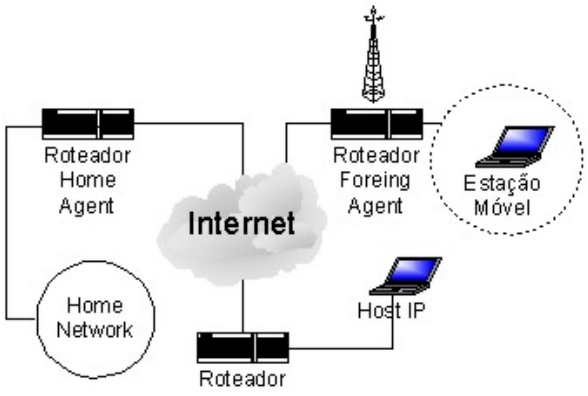


**Figura 2.24 -** *IP Móvel[2]*

Um outro tipo de serviço que tem evoluído é o serviço LCS *(Location Based Services)*, baseado em localização. Esse serviço compreende um conjunto de novas aplicações que utilizam a posição geográfica de um dispositivo móvel, tais como, serviços de emergência, localização de hospitais, restaurantes, etc.

As tecnologias utilizadas podem ser baseadas na rede ou no terminal. A tecnologia CGI+TA (*Cell Global ID + Time Advance*) é baseada na busca pela rede, e usa a identidade de cada célula para localizar a célula na qual o usuário se encontra mais uma estimativa de distância da estação móvel à estação base. A tecnologia baseada no terminal possui o GPS (*Global Positioning System*) que usa um conjunto de satélites para localizar a posição do usuário.

As aplicações são diversas:

- Serviços de informações: indica a localização de atrações e eventos próximos à localização do usuário.
- Serviços de rastreamento: usados para localizar usuários em caso de emergência.
- Gerenciamento de recursos: usados para gerenciar frotas de ônibus, caminhões, etc.
- Navegação: informa melhor trajeto entre dois locais.

## 2.7 Conclusão

A comunicação sem fio utilizando ondas eletromagnéticas foi um dos principais propulsores da evolução das telecomunicações devido à sua capacidade de transmissão de informações associada à mobilidade.

Em paralelo à evolução das telecomunicações também ocorreu um desenvolvimento significativo de aplicações e serviços. A crescente demanda por estes serviços tem apontado uma convergência das redes de comunicações existentes. Contudo restam alguns desafios no que se refere à qualidade destes serviços.

Num futuro próximo, espera-se a consolidação de algumas tecnologias. A tecnologia 2G de telefonia móvel celular GSM tem a maior infraestrutura implantada e em funcionamento no mundo, o que permite acreditar que a sua evolução para um

padrão de 3G é inevitável. A evolução natural desta tecnologia passa pelo GPRS, apontado como uma dos mais promissores agentes de transição do 2G para o 3G, onde serão providos serviços em banda larga.

No capítulo III é apresentada a tecnologia GPRS, que é a base deste trabalho.

# Capítulo III

## 3. GPRS

As tradicionais redes de comutação de circuitos são eficientes para prover serviços de voz. Entretanto, elas são ineficientes para oferecer serviços baseados em pacotes.

O GPRS é um padrão de comutação de pacotes para sistemas GSM, contudo também foi aceito pela TIA (*Telecommunications Industry Association*) como o padrão de comutação de pacotes para o sistema TDMA/IS-136.

O GPRS permite ao serviço de assinante transmitir ou receber dados no modo de transferência de pacotes fim-a-fim sem utilizar recursos da rede no modo de comutação de circuitos e tem funcionalidade de roteamento de pacotes para a infra-estrutura de rede. O tráfego sob demanda permite uma melhor utilização dos recursos da rede ao contrário dos serviços de comutação de circuitos onde os mesmos são alocados permanentemente para um usuário até o fim da transmissão, mesmo com períodos de ociosidade. Assim é possível suportar tráfego IP ou X.25 entre terminais GPRS e/ou entre terminais GPRS e redes de dados para diversas aplicações [22].

Com o GPRS as operadoras podem oferecer acessos com maiores taxas de transmissão de dados, diferenciação de usuários GPRS baseados num contrato de QoS e utilizar novos métodos de cobrança baseados em volume de dados e não na duração de uma conexão, o que permite reduzir a tarifação.

Para entender a arquitetura GPRS, é necessário, primeiramente, conhecer alguns fundamentos da tecnologia GSM.

## 3.1 GSM

O GSM é uma tecnologia desenvolvida na Europa e que tem o maior número de assinantes e a maior área de cobertura dentre os padrões de segunda geração de telefonia móvel no mundo.

### 3.1.1 Arquitetura da rede GSM

A tecnologia GSM está presente em muitos países, o que permite a facilidade do *roaming* do usuário, ou seja, o usuário pode utilizar o seu móvel fora da sua área de cobertura original. A cobertura de uma região é provida por uma ou mais PLMN (*Public Land Mobile Network*), que é uma rede operada ou licenciada por uma operadora e é subdividida em quatro subsistemas principais (Figura 3.1) [6]:

- NSS (*Network and Switching Subsystem*)
- OSS (*Operation and Support Subsystem*)
- BSS (*Base Station Subsystem*)
- MS (*Mobile Station*)

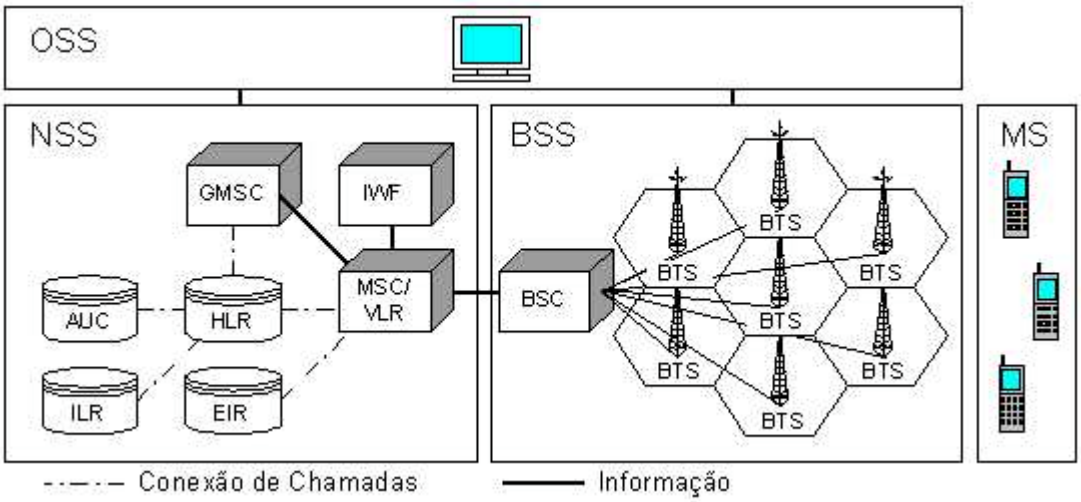


**Figura 3.1** – *Arquitetura GSM*

O OSS é conectado a diferentes componentes da NSS e da BSS, e tem a função de controlar e monitorar o sistema GSM de uma forma centralizada, diminuir o tempo de resposta à falhas e tornar mais fácil a monitoração do sistema como um todo. Dentre suas principais funções encontram-se o gerenciamento de configurações, gerenciamento de falhas e gerenciamento de desempenho.

O NSS inclui funções de comutação do GSM e possui uma base de dados necessária para os dados de assinante e gerenciamento da mobilidade. A principal função do NSS é gerenciar as comunicações entre os usuários GSM e outros usuários da rede de Telecomunicações. Os principais componentes do NSS são:

- HLR (*Home Location Register*) – O HLR é um banco de dados permanente que tem a função de gerenciamento das informações dos assinantes móveis. Uma PLMN pode conter um ou mais HLRs, dependendo do número de usuários, da capacidade do equipamento e da organização da rede. O HLR armazena informações do usuário e informações de localização para tarifação e roteamento de chamadas encaminhadas para a MSC onde a MS está registrada (Número de *roaming* da MS, número do VLR, número da MSC e a identidade local do móvel).

- VLR (*Visitor Location Register*) – O VLR é um banco de dados temporário que armazena informações dos usuários que estejam em sua área de responsabilidade. O VLR contém informações recebidas do HLR e informações de MS visitantes. Estas informações são necessárias para o controle e disponibilização dos serviços para cada assinante localizado dentro da área de serviço de uma MSC. O VLR armazena as seguintes informações: IMSI (*International Mobile Subscriber Identity);* MSISDN (*Mobile Subscriber Integrated Service Digital Network*); MSRN *(Mobile Station Roaming Number*); TMSI (*Temporary Mobile Station Identity*), se aplicável; LMSI (*Local Mobile Station Identity*); a área da posição onde a estação móvel foi registrada; a identidade do SGSN (*Serving GPRS*

Os padrões TMN (*Telecomunications Management Network*) podem ser utilizados no OSS.

*Support Node*) onde o MS foi registrado (este último, aplicável apenas se a PLMN suportar o GPRS).

- AUC (*Authentication Center*) – O AUC é um banco de dados que armazena a chave de identificação para cada assinante móvel registrado com o HLR associado. Esta chave é a cópia da existente em cada SIM (*Subscriber Identity Module*) e é usada para gerar dados usados na autenticação do IMSI e para comunicação codificada através da interface de rádio entre a MS e a BTS.

- EIR (*Equipment Identity Register*) - O EIR contém um ou vários bancos de dados que armazenam os IMEIs (*International Mobile Equipment Identity*) usados no sistema GSM. Uma MS pode ser armazenada em uma das seguintes listas: "*white list*" (equipamento OK); "*gray list*" (equipamentos com problemas de fabricação); "*black list*" (equipamentos roubados).

- ILR (*Interworking Location Register*) – O ILR é um registro que armazena e encaminha informações de *roaming* entre diferentes PLMN usando diferentes padrões de operação (utilizado apenas no GSM 1900).

- MSC (*Mobile-Services Switching Center*) – A MSC realiza funções de comutação e sinalização para as MSs localizadas dentro de sua área de responsabilidade. A principal diferença entre a MSC e uma central de rede fixa é que a MSC deve levar em conta o impacto para alocação de recursos de rádio assim como a natureza do assinante. Por controlar as chamadas a MSC é responsável por gerar a bilhetagem automática de cada chamada.

- GMSC (*Gateway MSC*) – O GMSC é o ponto de entrada e saída de uma rede de telecomunicações. Quando uma MSC está conectada à rede externa, esta também incorpora funcionalidades de *Gateway*. Uma chamada proveniente de uma rede PSTN para uma MS passa

primeiramente por uma GMSC que faz a busca da localização da MS consultando o HLR.

- IWF (*Interworking Funtion*) – O IWF é uma entidade funcional associada à MSC que tem a funcionalidade de interconexão entre a PLMN e redes de comunicação de dados externas.

O BSS é composto por equipamentos de estações bases vistos pela MSC através de uma interface “A” como as entidades responsáveis pela comunicação com as MSs em uma certa área. O BSS consiste de duas entidades:

- BSC (*Base Station Controller*) é o ponto central do BSS e faz o controle das conexões de estações móveis e *handover*; gerenciamento da rede de rádio, transcodificação e adaptação de taxa, concentração de tráfego, gerenciamento de transmissão das BTSs e controle a distância das BTSs.

- BTS (*Base Tranceiver Station*) inclui todos os rádios e interfaces de transmissão necessárias em uma célula. A BTS trabalha com pares de freqüência distintos para transmissão (canal direto) e para recepção (canal reverso).

A MS representa o terminal de acesso do usuário à rede. A MS possui um cartão de identificação chamado SIM que permite a mobilidade pessoal, de modo que um usuário pode ter acesso aos seus serviços independentemente do terminal. Basta inserir o cartão SIM em qualquer terminal GSM que o usuário será capaz de receber e executar chamadas ou qualquer outro tipo de serviço. O equipamento móvel é unicamente identificado pela IMEI, que é usada para procedimentos de segurança, como identificação de aparelho roubado, e também para evitar acesso não autorizado à rede.

A MS pode assumir três estados de operação: O estado livre, quando a estação móvel estiver ligada e em um canal de controle; O estado ativo, quando a MS está ligada e em um canal de conversação; E o estado desativado, quando a estação móvel está desligada.

### 3.1.2 Transmissão

Diferentes bandas de freqüência podem ser utilizadas, tais como, 900, 1800 e 1900 MHz. A banda de 900 ou banda primária inclui duas sub-bandas de 25 MHz cada, 890-915 MHz e 935-960 MHz. A banda de 1800 também possui duas sub-bandas de 75 MHz cada, 1710-1785 MHz e 1805-1880 MHz. Para a banda de 900 cada banda de 25 MHz é dividida em 125 canais de 200 kHz, dos quais 124 são usados, pois o canal 0 é utilizado como banda de guarda. Já na banda de 1800, as duas bandas de 75 MHz são divididas em 375 canais de 200 kHz, dos quais 374 são utilizados, pois o canal 0 é utilizado como banda de guarda [6].

O GSM utiliza acesso FDMA/TDMA, com os canais de rádio de 200 kHz divididos em quadros com 8 *time slots* com duração de 0.577ms cada. Os *time slots* entre os canais direto e reverso são relacionados de forma que o móvel não transmita e receba simultaneamente.

A técnica de modulação utilizada no GSM é o GMSK (*Gaussian Minimum Shift Keying*). Esta técnica de modulação digital de banda estreita é baseada no deslocamento de fase. Os bits de informação são representados por deslocamentos de fases positivos ou negativos. Mudando de fase continuamente as descontinuidades são evitadas, assim estreitando a banda de freqüência da portadora modulada. A modulação GMSK também envolve a filtragem do fluxo de bits que chegam com um filtro Gaussiano. De fato a largura de banda do canal se transforma em 162 kHz [6].

## 3.2 Rede GPRS

Adicionando as funcionalidades do GPRS, as operadoras podem proporcionar aos seus assinantes acesso com utilização eficiente de recursos a redes IP externas com taxas de dados de até 171,2 kbps.

### 3.2.1 Modelo de Referência Simplificado

O GPRS provê capacidade de transmissão entre uma entidade transmissora e uma ou várias entidades receptoras. Estas entidades podem ser MS ou TE (*Terminal Equipment*). A Figura 3.2 ilustra o modelo de referência simplificado GPRS [22].

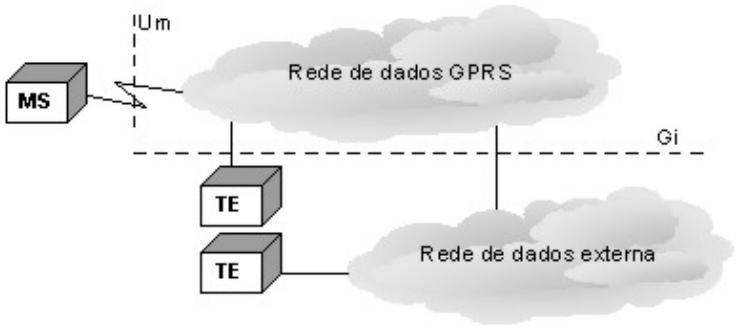


**Figura 3.2** - *Modelo de referência simplificado GPRS [22]*

Para os inúmeros serviços que podem ser disponibilizados existem duas configurações básicas:

- PTP (*Point-to-point*): Permite a comunicação entre dois pontos dedicados. Este tipo de comunicação é dividida em dois tipos de serviços:
  - PTP-CONS (*Point-to-point connections oriented network service*), estabelece a conexão antes da transferência de dados.
  - PTP-CLNS (*Point-to-point connections network service*), onde pacotes com informações são enviadas baseadas no endereço de destino. Não se garante a ordem e nem atrasos mínimos na rede.
- PTM (*Point-to-Multipoint*): Permite a transmissão da informação de um para vários usuários de três formas:
  - PTM-M (*Point-to-Multipoint Multicast*) permite ao usuário enviar mensagens dentro de uma área geográfica definida pelo próprio usuário.

– PTM-G (*Point-to-Multipoint Group Call*) permite ao usuário enviar mensagens para um grupo pré-definido.

– IP-M endereça um grupo no qual não existe limitação de área geográfica uma vez que a mensagem pode se propagar pela Internet.

## 3.2.2 Arquitetura da Rede GPRS

O GPRS é considerado como um serviço ou uma extensão do GSM e foi desenvolvido para ser implementado sobre a infra-estrutura existente sem interferir nos serviços já implantados. Isto é possível porque o GPRS utiliza a mesma banda de freqüência e técnicas de salto, a mesma estrutura do quadro TDMA e a mesma modulação que o GSM.

Para a introdução do GPRS são necessárias algumas modificações. Alguns dos nós já implementados nos sistemas atuais podem ser compartilhados entre o GPRS e o sistema GSM. Apenas dois novos tipos de nós, o SGSN (*Serving GPRS Support Node*) e o GGSN (*Gateway GPRS Support Node*) têm que ser adicionados (Figura 3.3) [30].

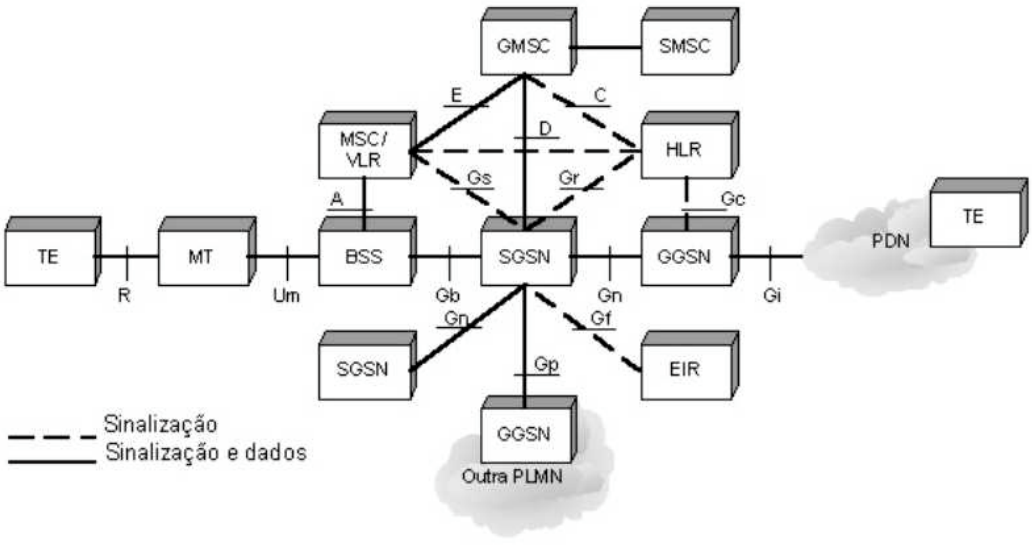


**Figura 3.3** – *Arquitetura GPRS [25]*

- SGSN é um roteador que tem a função de localizar uma MS dentro de sua área de responsabilidade e fazer o roteamento de pacotes dentro da PLMN. O SGSN serve todos os assinantes GPRS que estiverem fisicamente localizados dentro da área de serviço deste SGSN. O tráfego é roteado do SGSN para a BSC, BTS e então MS. O SGSN também é responsável por fazer a cifragem, autenticação, gerenciamento de sessão, gerenciamento de mobilidade, gerenciamento do enlace lógico para o MS e conexões para o HLR, MSC, BSC, GGSN.

- GGSN é o nó *gateway* entre uma rede de pacotes de dados externa e a rede de *backbone* do GPRS. No caso de uma rede IP externa, o GGSN pode ser visto como um roteador IP convencional servindo a todos os endereços IP das estações móveis. Este nó pode incluir *firewalls* e mecanismos de filtragem de pacotes. Além disso, sua tarefa é atribuir o SGSN correto para a estação móvel, dependendo da localização da mesma.

A rede GPRS utiliza o mesmo BSS que os serviços de voz GSM, com uma atualização de software na BTS e no BSC e a instalação de um novo componente de hardware chamado PCU (*Packet Control Unit*), geralmente localizado no PCUSN (*PCU Support Node*). O PCUSN reside tipicamente entre o BSC e o SGSN, sendo responsável pela transferência de pacotes entre o BSS e a rede GPRS.

Os bancos de dados VLR, HLR, EIR e AUC também são utilizados na rede GPRS. O HLR mantém informações atualizadas de cada assinante registrado na rede GPRS tais como: IMSI, informações de localização corrente, perfil de serviços contratados, endereço do SGSN corrente, um ou mais endereços PDPs (*Packet Data Protocol*), permissões de *roaming* e estado de atividade. O VLR armazena informações de usuários ativos dentro de uma área de serviço e usuários em *roaming*. O EIR e o AUC realizam o controle de segurança e autenticação dos assinantes móveis.

A entidade MSC é usada somente em GPRS quando serviços de dados comutados por pacotes e por circuitos são suportados. O tráfego de voz é enviado a

MSC pelo BSC como no sistema GSM, enquanto o tráfego de dados é enviado para o SGSN, através da PCU, em uma interface *Frame Relay*.

A BTS tem as mesmas funções que no sistema GSM, ou seja, modulação, demodulação, transmissão e recepção no enlace de rádio. A BTS suporta os protocolos GPRS relevantes para a comunicação na interface aérea, gerência de chamadas e executa a atribuição de recursos de rádio.

Os sistemas GPRS e GSM fornecem interoperabilidade e compartilhamento dos recursos dinamicamente entre os usuários. Por esta razão, esta nova tecnologia requer o desenvolvimento de novos terminais móveis. Três tipos de terminais foram definidos:

- Classe A: suporta uma conexão comutada por circuito e outra comutada por pacote simultaneamente, possibilitando ao usuário iniciar ou receber uma chamada de voz sem interromper uma comunicação de dados.

- Classe B: é capaz de conectar ambos, GSM e GPRS, mas receber uma chamada de voz requer que as transmissões de dados em andamento sejam suspensas durante o tempo da ligação.

- Classe C: permite aos assinantes acessarem de maneira exclusiva somente um tipo de serviço em um determinado tempo.

A MS GPRS possui dois componentes: o MT (*Mobile Terminal*, ex: *handset*), usado para acessar a interface de rádio, e o TE (ex: *laptop*). A MS também pode ser uma única unidade combinando as funcionalidades do MT e do TE.

Para definir as interações entre os elementos da rede GPRS, algumas interfaces foram introduzidas (Figura 4.3): Gb, entre PCU(BSS) e SGSN usando *Frame Relay*; Gr, entre o SGSN e o HLR; Gn, entre SGSN e o GGSN quando estes estão localizados na mesma PLMN; Gp, entre SGSNs e o GGSN caso estejam em PLMN diferentes; Ambas as interfaces Gn e Gp transmitem dados de usuários e sinalização; Gc, entre o GGSN e o HLR; Gf, entre o SGSN e o EIR; Gs, entre o SGSN e o VLR; Gi, entre o GGSN e PDNs externas [25].

As principais funções das entidades GPRS são apresentadas na Tabela 3.1.

| Controle de acesso à rede | MS | BSS | SGSN | GGSN | HLR |
|---|---|---|---|---|---|
| Registro | | | | | X |
| Autenticação e autorização | X | | X | | X |
| Controle de admissão | X | X | X | | |
| Exame de mensagem | | | | X | |
| Adaptação de terminal de Pacotes | X | | | | |
| Coleção de dados tarifados | | | X | X | |
| **Transferência e roteamento de pacotes** | | | | | |
| *Relay* | X | X | X | X | |
| Roteamento | X | X | X | X | |
| Mapeamento e translação de endereço | X | | X | X | |
| Encapsulamento | X | | X | X | |
| Tunelamento | | | X | X | |
| Compressão | X | | X | | |
| Cifragem | X | | X | | X |
| **Gerenciamento de mobilidade** | | | | | |
| Estabelecimento de enlace lógico | X | | X | | |
| Manutenção de enlace lógico | X | | X | | |
| Liberação de enlace lógico | X | | X | | |
| **Gerenciamento de recursos de rádio** | | | | | |
| Gerenciamento de Um | X | X | | | |
| Seleção de célula | X | X | | | |
| Um-Tranx (Transf. Dados BSS-MS) | X | X | | | |
| Gerenciamento de caminho | | X | X | | |

**Tabela 3.1 –** *Funções das entidades GPRS [25]*

## 3.2.3 Plano de Transmissão GPRS

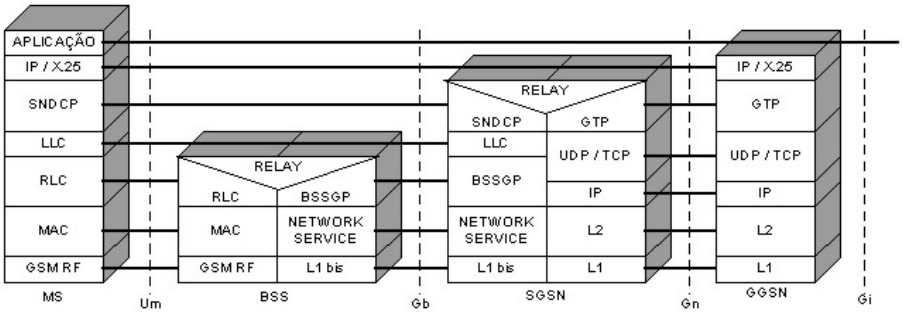


**Figura 3.4**- *Plano de transmissão GPRS [31]*

O plano de transmissão (Figura 3.4) consiste de protocolos em camadas que provêem transferência de informação de usuário, juntamente com procedimentos de controle de transferência de informação associado (ex: controle de fluxo, detecção e correção de erro).

A camada de aplicação representa aos serviços de usuários finais, tais como, HTTP, FTP (*File Transfer Protocol*), SMTP (*Simple Mail Transfer Protocol*) e Telnet.

A aplicação de usuário gera os dados que compõem a PDU (*Packet Data Unit*) que é encapsulada na camada SNDCP (*Sub-Network Dependent Convergence Protocol*). O SNDCP é responsável por fazer a multiplexação das várias conexões da camada de rede em uma única conexão lógica da camada LLC (*Logic Link Control*), e pelos processos de criptografia, segmentação, compressão e descompressão de dados do usuário. O SNDCP faz a compressão e segmenta a PDU em um ou mais quadros LLC dependendo do tamanho máximo definido para um quadro LLC, e os envia para a camada inferior. A transmissão e recepção de PDUs entre a MS e o SGSN pode ocorrer no modo com reconhecimento e sem reconhecimento. No modo com reconhecimento o número máximo de bytes de campo de informação de um quadro LLC é 1520 bytes e a recepção dos quadros LLC é confirmada na camada LLC. No modo sem reconhecimento, o número máximo de bytes contidos no campo de informação de um quadro LLC é de 500 bytes, sendo que nenhuma confirmação é retornada pela camada LLC. No lado do receptor, o SNDCP reagrupa os quadros LLC recebidos antes de efetuar a descompressão dos dados [23].

A camada LLC (MS-SGSN) e a camada RLC/MAC (*Radio link Control/Medium Access Control*) são subcamadas que compõem a camada de enlace de dados. A camada LLC opera acima da camada RLC, fornecendo um enlace lógico único e altamente confiável entre uma MS em particular e o seu SGSN corrente. Um DLCI (*Data Link Connection Identifier*) indica esse enlace lógico e é composto pelo TLLI (*Temporary Logical Link Identifier*) e pelo identificador SAP (*Service Access Point*). Os SAPs são os pontos onde a camada SNDCP pode acessar os serviços oferecidos pela camada LLC. A camada LLC faz o controle de seqüência, controle de fluxo, criptografia e tratamento de erros. A camada LLC também pode trabalhar

no modo de transferência com ou sem reconhecimento. O modo com reconhecimento permite a retransmissão de quadros corrompidos, perdidos e não confirmados, enquanto no modo sem reconhecimento, a recuperação de erro não é possível [24].

A transferência de dados entre a MS e o BSS (mais precisamente a PCU) sobre a camada física da interface aérea GPRS é sustentada pela camada RLC/MAC, que define quatros níveis de prioridade de rádio. Quando uma MS desejar acessar a rede GPRS, ela pode indicar um dos níveis de prioridade, indicando se o acesso é para transmissão de mensagem de sinalização ou mensagem de dados do usuário. Essa informação é utilizada pelo BSS para determinar a prioridade de acesso ao meio físico em caso de congestionamento. A principal função da camada RLC é fornecer um enlace confiável entre a MS e o BSS. Na camada RLC do transmissor, cada quadro recebido da camada LLC é segmentado em um ou mais blocos de dados RLC. Esses blocos são enviados para a camada MAC onde um cabeçalho MAC e um BCS (*Block Check Sequence*) são adicionados formando um bloco RLC/MAC ou bloco de rádio que possui tamanho fixo. Um quadro LLC pode ser espalhado em vários Blocos RLC/MAC, que por sua vez podem suportar vários LLCs. Na camada RLC do receptor, os blocos de dados RLC são reagrupados formando novamente um quadro LLC. A camada RLC também pode operar no modo de transmissão com ou sem reconhecimento [27].

A camada MAC presente na MS e na BTS faz o monitoramento e alocação dos canais físicos, controle de tentativa de acesso ao meio de transmissão (um ou mais canais físicos) compartilhado por várias MSs e multiplexação de blocos de rádio no meio físico. Além disso, a camada MAC pode ser responsável por colocar dados em uma fila e escaloná-los quando necessário podendo priorizar a transmissão de certos dados com base no contrato de QoS negociado entre o assinante móvel e a rede.

As subcamadas PLL (*Physical Link Layer*) e a RLL (*Radio Link Layer*) compõem a camada física. A subcamada PLL fornece um canal físico entre a MS e o BSS e suas funções incluem a transferência de blocos de rádio (unidade de transmissão da camada RLC/MAC ) no canal físico, codificação de canal, medidas de nível de qualidade de sinal recebido, medida de avanço de tempo, detecção de

congestionamento no meio físico, controle de potência, seleção de célula e recepção descontinua. A função da camada RLL inclui modulação e demodulação [26].

A camada BSSGP (*BSS GPRS Protocol*) fornece informações de QoS e roteamento para facilitar a transferência de dados entre o BSS e o SGSN. Quadros que chegam na PCU passam pelo BSSGP, onde as mensagens de dados do usuário e de sinalização são separadas em quadros LLC. Uma rede *Frame Relay* fornece um enlace confiável entre o BSS e o SGSN [28].

A transmissão de pacotes de dados através do *backbone* GPRS é feita utilizando um protocolo chamado GTP (*GPRS Tunneling Protocol*). Esse protocolo estabelece túneis de transmissões entre SGSN e GGSN [29]. Este opera acima do protocolo TCP (*Transfer Control Protocol*), usado principalmente para tráfego de pacotes X.25, ou do protocolo UDP (*User Datagram Protocol*), usado para tráfego de pacotes IP que exigem pequenos atrasos.

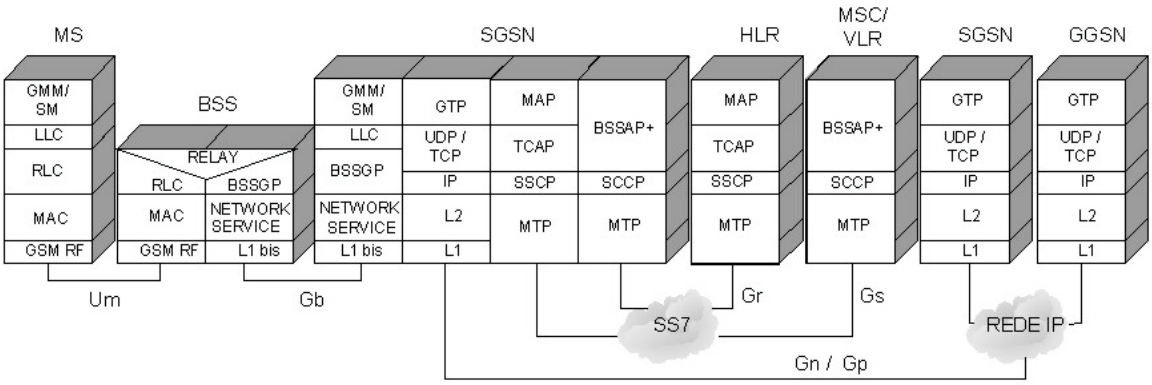


**Figura 3.5**- *Plano de controle GPRS [31]*

O plano de controle GPRS (Figura 3.5) consiste de um conjunto de protocolos com funções de sinalização e controle usados no estabelecimento, manutenção e término das transmissões. Além de todos elementos usados no plano de transmissão, o plano de controle ainda faz uso dos elementos HLR e MSC/VLR para o gerenciamento de mobilidade e autenticação.

## 3.2.4 Interface de Rádio GPRS

O método de acesso ao meio utilizado pelo GPRS é o FDMA/TDMA definido no GSM. O GPRS também utiliza a mesma estrutura de *time slot* do GSM. Cada *time slot* pode ser atribuído tanto ao GPRS para a transmissão de dados de comutação por pacote, quanto ao GSM tratando chamadas comutadas por circuito.

Na interface aérea os recursos são alocados apenas temporariamente a cada pacote, ao contrário, por exemplo, do esquema adotado pelo GSM que usa comutação por circuito onde *time slots* são atribuídos a um usuário por toda a duração de uma chamada. No GPRS os recursos de rádio são distribuídos apenas pela duração de um pacote IP. Os canais físicos utilizados pelo GPRS recebem a denominação de PDCH (*Packet Data Channel*). O número de PDCHs em uma célula pode ser fixo ou alocado dinamicamente para suportar a variação de demanda no tráfego. Mais de um *time slot* pode ser alocado para um usuário durante a transferência dos pacotes. Os recursos para conexões *uplink* e *downlink* são alocados separadamente de caso em caso, refletindo o comportamento assimétrico da comunicação de dados. Cada PDCH pode ser definido em uma estrutura de multiquadro de acordo com a Figura 3.6.

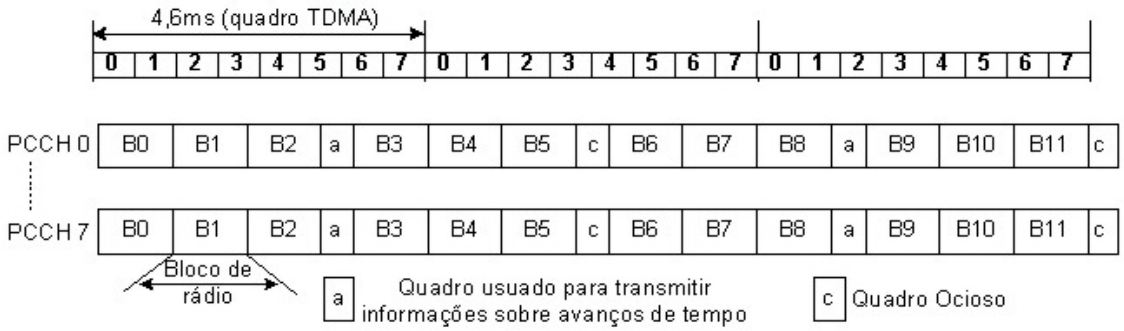


**Figura 3.6** – *Estrutura Multiquadros GPRS [26]*

O ciclo de 52 quadros TDMA sucessivos divididos em 12 blocos de 4 quadros TDMA consecutivos cada (totalizando 48 quadros TDMA para a alocação de PDCH), 2 quadros TDMA para transmitir informações sobre avanços de tempo e 2

quadros TDMA ociosos formam um multiquadro. Cada ciclo dura aproximadamente 240 ms (52 quadros vezes 8 períodos de *burst* → 52 x 8 x 0,577 ms) [32].

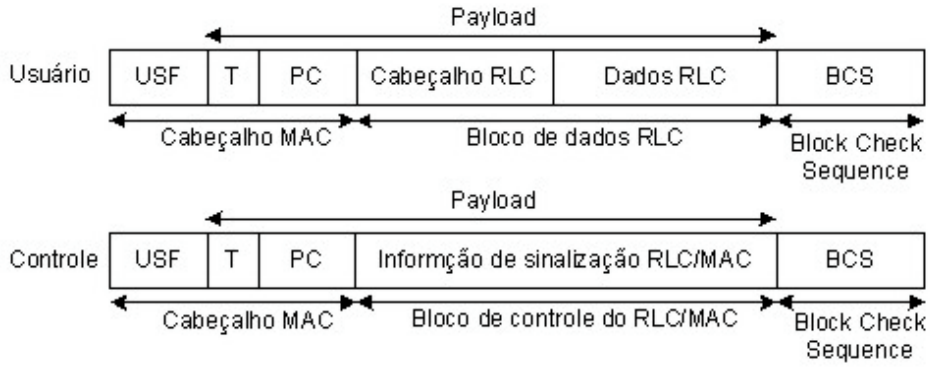


**Figura 3.7** – *Bloco de rádio [26]*

Um bloco de rádio (Figura 3.7) é composto por um bloco de dados RLC ou por um bloco de controle RLC/MAC, um cabeçalho MAC e um BCS (*Block Check Sequence*). Um bloco de rádio, resultando em 456 bits após a codificação de canal, é transmitido pela camada física a cada 20 ms aproximadamente, no mesmo *time slot* em 4 quadros TDMA consecutivos.

Um conceito importante em GPRS é sua capacidade *multislot*: dependendo do número de PDCHs disponíveis em uma célula, da capacidade *multislot* de uma MS e da carga do sistema, entre um e oito *time slots* por quadro TDMA podem ser alocados para uma única MS possibilitando a transmissão e recepção de pacotes em múltiplos PDCHs, variando assim a taxa de dados entre uma MS e a rede GPRS. Além disso, alocando um certo número de blocos de um PDCH para cada MS, é possível que até oito MSs compartilhem simultaneamente o mesmo PDCH.

Ao contrário do sistema GSM, onde uma MS utiliza o mesmo *time slot* para *uplink* e *downlink*, no GPRS os recursos de rádio *downlink* e *uplink* são alocados separadamente, oferecendo suporte a tráfego de dados assimétrico (como o tráfego WEB).

## 3.2.5 Canais Lógicos GPRS

Os canais lógicos são mapeados dentro dos canais físicos PDCH. São definidos três conjuntos similares de canais lógicos GPRS que são usados na fase de estabelecimento de transferência de pacotes para alocação de recursos e sinalização. O primeiro conjunto consiste de canais dos seguintes canais lógicos [27]:

- BCCH (*Broadcast Control Channel*): usado no *donwlink* para transmitir informações específicas do sistema a todos os usuários de uma célula.
- PCH (*Paging Channel*): usado no *downlink* para localizar um usuário antes de iniciar a transmissão de dados.
- RACH (*Random Access Channel*): usado no *uplink* para requisitar recursos de rádio antes de iniciar uma transmissão de dados ou sinalização ou em resposta a um *paging*.
- AGCH (*Access Grant Channel*): usado no *downlink* para alocar recursos de rádio ou um canal de controle dedicado.

O segundo conjunto (Figura 3.8) consiste dos seguintes canais lógicos [26]:

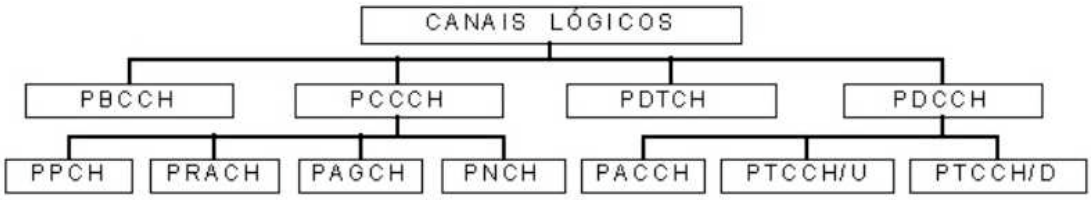


**Figura 3.8 –** *Canais Lógicos*

- PBCCH (*Packet Broadcast Control Channel*): usado no *donwlink* para transmitir informações específicas do sistema a todos os usuários da célula.
- PDTCH (*Packet Data Traffic Channel*): este canal é alocado temporariamente para um MS ou um grupo de MSs para a transferência de dados.

- PCCCH (*Packet Common Control Channel*): compreende quatro tipos de canais lógicos utilizados para controle comum e sinalização:

  - PPCH (*Packet Paging Channel*): usado no *downlink* para enviar requisições as MSs.

  - PRACH (*Packet Random Access Channel*): usado no *uplink* para requisitar recursos da rede GPRS.

  - PAGCH (*Packet Access Grant Channel*): usado no *downlink* para alocar recursos GPRS.

  - PNCH (*Packet Notification Channel*): usado para notificação (ponto-multiponto) de um grupo de MSs antes da transferência de pacotes.

- PDCCH (*Packet Dedicated Control Channels*): compreende três tipos de canais associados ao controle dos canais de dados:

  - PACCH (*Packet Associated Control Channel*): canal bidirecional associado a um TBF (*Temporary Block Flow*).

  - PTCCH/U (*Packet Timing advance Control Channel Uplink*): usado para transmitir rajadas de acesso aleatório para permitir estimar o avanço de tempo em uma MS em estado de transferência.

  - PTCCH/D (*Packet Timing advance Control Channel Downlink*): usado para transmitir o avanço de tempo para várias MS. Um PTCCH/D faz par com vários PTCCH/Us.

O terceiro conjunto consiste dos seguintes canais lógicos (canais de controle compactos):

- CPBCCH (*compact Packet Broadcast Control Channel*): usado no *downlink* para *broadcast* de informações específicas do sistema. O que diferencia este canal do PBCCH é a estrutura física.

- CPPCH (*Compact Packet Paging Channel*): usado no *downlink* para enviar requisições às MSs num canal de controle compacto.
- CPAGCH (*Compact Packet Access Grant Channel*): usado no *downlink* para alocar recursos GPRS em canal de controle compacto.
- PACCH: citado anteriormente.
- PTCCH/U: citado anteriormente.
- PTCCH/D: citado anteriormente.

### 3.2.6 Mapeamento de canais lógicos

O conceito de “mestre e escravo” é utilizado no mapeamento onde vários canais lógicos GPRS podem ser mapeados no tempo e na freqüência pelo mesmo canal físico PDCH. Através desse conceito, pelo menos um PDCH atua como um canal mestre, acomodando PCCCHs, PDTCH e PACCH, os outros 7 PDCHs atuam como escravos sendo usados somente para transmitir dados do usuário e sinalização dedicada [26].

### 3.2.7 Capacidade de transmissão

O GPRS possui dois mecanismos que determinam a capacidade de transmissão (ou taxa máxima de transmissão) e a qualidade de serviço:

- Esquemas de codificação (CS – *Coding Scheme*)
- Transmissão em múltiplos *time slots* (*Multislot Operation*)

Quatro esquemas de codificação de canal de CS-1 a CS-4 são definidos para os canais de tráfego de dados (PDTCHs). Para todos os outros canais de controle apenas o CS-1 é utilizado.

O esquema de codificação CS-1 (Figura 3.9) é o esquema de codificação usado no GSM, sendo obrigatória no móvel e na BTS. Esta codificação possui taxa 1/2 e vários bits são acrescentados na informação antes de serem transmitidos, garantindo uma maior proteção à interferência e uma cobertura maior. Na saída do codificador têm-se 456 bits, assim considerando que no GSM estão disponíveis dois blocos de 57 bits para a transferência de dados e sinalização, serão necessárias quatro rajadas para a transferência de todos os 456 bits [30].

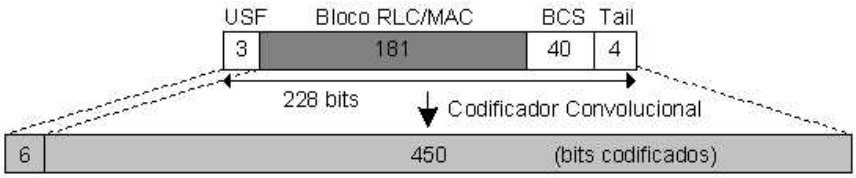


**Figura 3.9** – *CS-1*

O CS-2 (Figura 3.10) possui 268 bits de dados contra 181 bits do CS-1, o que significa uma maior taxa de dados, contudo uma menor proteção do canal. Os 268 bits somados com 23 bits de cabeçalho passarão pelo codificador 1/2 já existente na rede. Com isso, na saída do codificador têm-se 588 bits que não podem ser transmitidos em um bloco de rádio padrão que possui 456 bits. Para fazer esta adaptação é realizado um processo de *puncturing* (realizado por meio de *software*), onde 132 bits em posições pré-definidas são simplesmente descartados, diminuindo-se a proteção do canal. Esta é uma forma de se conseguir um codificador 2/3 a partir de um codificador 1/2 já implementado. Para os esquemas de codificação CS-2, CS-3 e CS-4 os bits de USF (*Uplink State Flag*) passam também por uma pré-codificação.

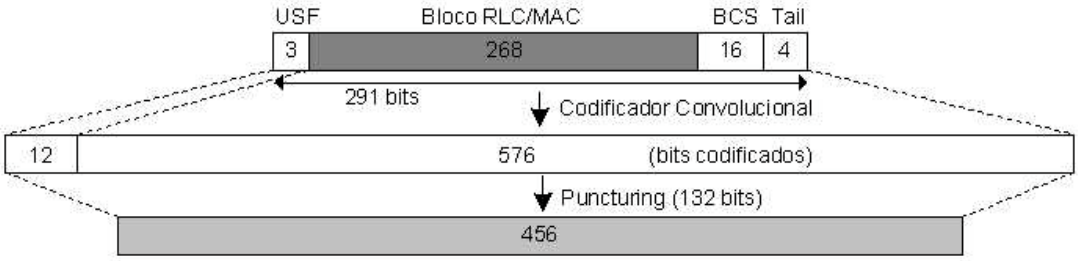


**Figura 3.10** – *CS-2*

O CS-3 (Figura 3.11) possui o mesmo principio do CS-2 para 312 bits de dados. Os 312 bits de dados somados a 23 bits de cabeçalho passam por um

codificador 1/2 gerando 676 bits na saída. Como no CS-2, é realizado o processo de *puncturing* com um descarte de 220 bits e conseqüentemente uma menor proteção do canal. Para CS-3 tem-se um codificador 3/4 e uma cobertura menor que CS-2.

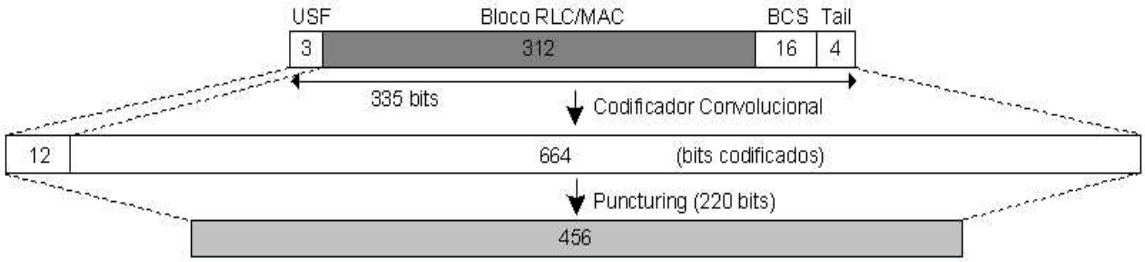


**Figura 3.11** – *CS-3*

O CS-4 (Figura 3.12) possui 428 bits de dados que não passam por um codificador sendo transmitidos direto num bloco de rádio. Apenas os 3 bits do USF são codificados.

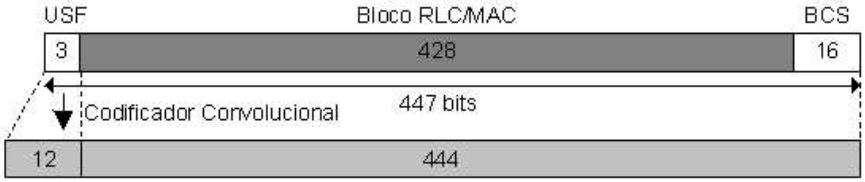


**Figura 3.12** – *CS-4*

A Tabela 3.2 abaixo mostra a relação entre o esquema de codificação e a relação sinal ruído requerida.

| Esquema | Taxa de codificação | Taxa de dados | S/R Requerido |
|---|---|---|---|
| CS-1 | 1 / 2 | 9.05 Kbps | 9 dB |
| CS-2 | 2 / 3 | 13.4 Kbps | 13 dB |
| CS-3 | 3 / 4 | 15.6 Kbps | 15 dB |
| CS-4 | 1 | 21.4 Kbps | 23 dB |

**Tabela 3.2** – *Taxa de dados x S/R requerido*

Dentro de uma célula têm-se diferentes relações S/R (Sinal/Ruído) o que requer o uso de diferentes esquemas de codificação (Figura 3.13). Quanto mais próximo da BTS um móvel estiver, maior será o nível de sinal recebido, tanto do ponto de vista da MS quanto da BTS.

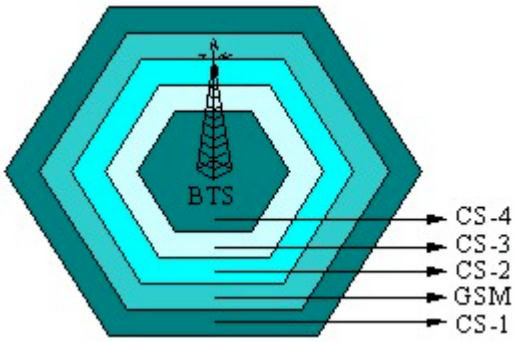


**Figura 3.13** – *Esquemas de codificação x Cobertura*

Na transmissão em múltiplos *time slots* o sistema disponibilizará ao usuário o volume de recursos necessários para a transferência de dados. Estes recursos serão tanto maiores quanto menor for a utilização da rede. Em momentos de baixo tráfego o usuário poderá receber a alocação de inúmeros *time slots*, e em momentos de congestionamentos vários usuários podem compartilhar um único *time slot*. Assim, a capacidade máxima teórica ocorre com a utilização de 8 PDCHs por usuário que representa 171,2 kbps (8 x 21,4 kbps).

## 3.2.8 Gerenciamento de mobilidade

O gerenciamento de mobilidade consiste em manter atualizada a informação sobre localização atual da MS a fim de facilitar o processo de *paging*. Um modelo dos estados assumidos por uma MS é descrito na Figura 3.14 [25].

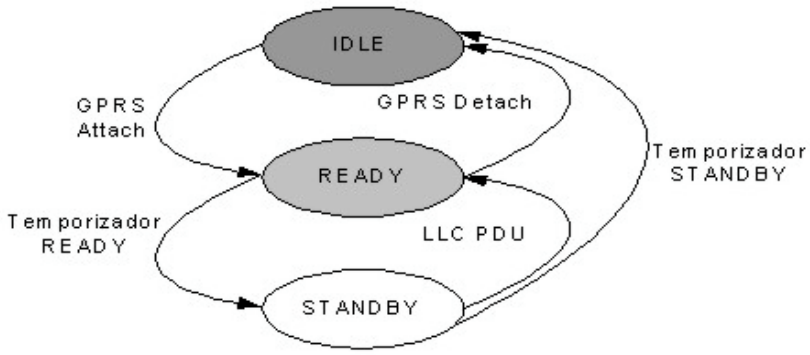


**Figura 3.14** – *Estados GMM [25]*

- IDLE – No estado IDLE são executados os serviços GSM. Neste estado o assinante não está conectado ao gerenciamento de mobilidade GPRS. Não existe nenhuma informação de roteamento ou localização válida para o assinante. A transmissão de dados e a busca de um assinante não são possíveis. A MS GPRS é marcada como indisponível neste caso. A fim de estabelecer uma conexão para gerenciamento de mobilidade GMM (GPRS *Mobility Management*) entre a MS e o SGSN, a MS deve executar o procedimento de GPRS *Attach*.

- READY – Ao executar o procedimento de *Attach* a MS entra no estado READY e pode dar início à transferência de dados. Neste estado a MS executa os procedimentos de gerenciamento de mobilidade para informar à rede qual é a célula selecionada atual. A seleção/reseleção de célula é executada localmente pela MS, ou pode ser opcionalmente executada pela rede. A MS pode também ativar ou desativar o contexto PDP. Independente se recursos de rádio estão ou não alocados para um assinante, o contexto MM (*Mobility Management*) permanece no estado READY mesmo não existindo dados para a transferência. Este estado possui um temporizador que marca o tempo de ociosidade da MS, se a MS ficar ociosa por um longo período a MS alcançará o estado STANDBY. A MS também pode retornar ao estado IDLE através de uma desconexão (GPRS *Detach*). Neste estado não é necessário o *paging* pois a cada mudança de célula a MS envia uma mensagem de atualização para o seu SGSN corrente.

- STANDBY – Neste estado o SGSN conhecerá apenas a RA (*Routing area*) de cada MS. Quando um pacote for destinado a uma MS que esteja nesse estado, será necessário enviar uma mensagem de *paging* para a área de roteamento na qual esta MS está localizada. A MS destino responde ao *paging* retornando a célula na qual está residindo atualmente. Caso a MS mude para uma nova área de roteamento, essa mudança de localização será informada ao seu SGSN. A partir desse estado uma MS pode

alcançar o estado IDLE, caso o temporizador STANDBY expire, ou o estado READY se PDUs começarem a ser transmitidas.

### 3.2.9 Procedimentos de Attach e Detach

Antes de iniciar uma transmissão de dados a MS precisa executar os procedimentos de *Attach* e PDP. No procedimento de *Attach* a MS requisita conexão com a rede fazendo seu registro no SGSN e estabelecendo um enlace lógico entre a MS e o SGSN conforme a Figura 3.15 [25].

- A MS envia ao SGSN uma requisição de *Attach* contendo sua identidade (P-TMSI – *Packet Temporary Mobile Subscriber Identity*), um NSAPI (*Network Layer Service Access Point Identifier*) que identifica uma aplicação de rede e um identificador de RA (*Routing Area*) onde ela está localizada.

- O SGSN verifica no HLR se o usuário está autorizado e autenticado.

- Após a autenticação, o SGSN envia uma resposta para a MS contendo um TLLI específico e temporário que será utilizado pela camada LLC para possibilitar a comunicação de dados no enlace lógico entre a MS e SGSN.

- O SGSN mantém uma tabela que contém entradas relacionadas a cada MS conectada a ele. Quando uma MS se registra com o SGSN, uma nova entrada é criada na tabela contendo informações que mapeiam a identidade móvel com o TLLI atribuído a ela e o NSAPI. As entradas da tabela do SGSN são atualizadas após a ativação de um contexto PDP.

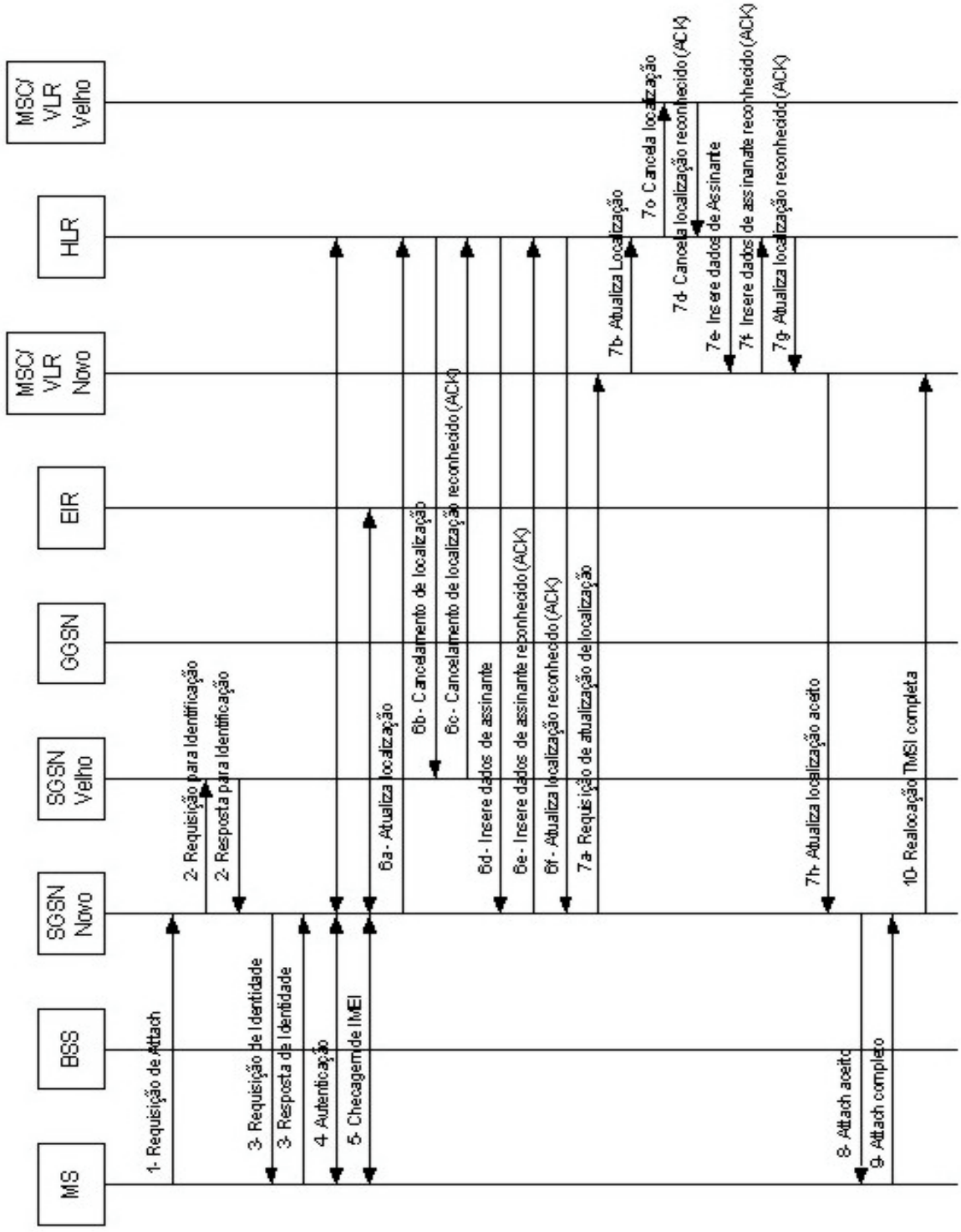


**Figura 3.15** – *Procedimento de Attach [25]*

A desconexão de uma MS da rede GPRS é conhecida por *Detach* e pode ser iniciada pela MS ou pela rede. Quando uma MS é desconectada da rede GPRS todos os seus contextos PDPs ativos são removidos pelo GGSN a pedido do SGSN. O

procedimento *Detach* pode ser explicito, quando a MS ou a rede GPRS requerer que a MS alcance o estado IDLE, ou implícito, quando expirar o tempo em STANDBY. Os procedimentos são ilustrados nas figuras 3.16 e 3.17.

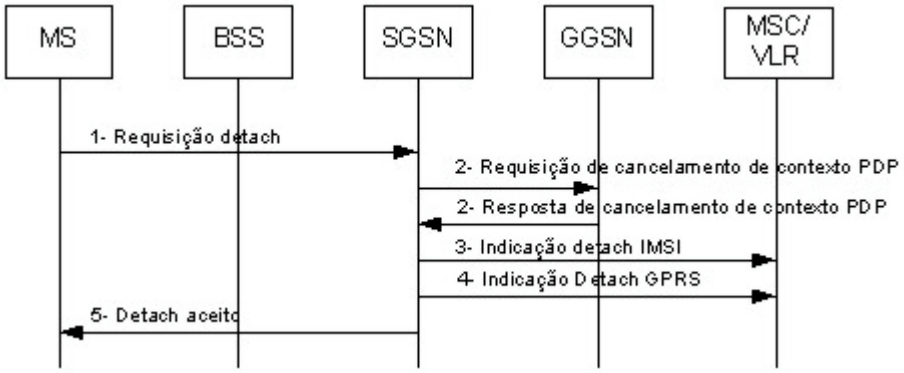


**Figura 3.16** - *Procedimento Detach iniciado pela MS [25]*

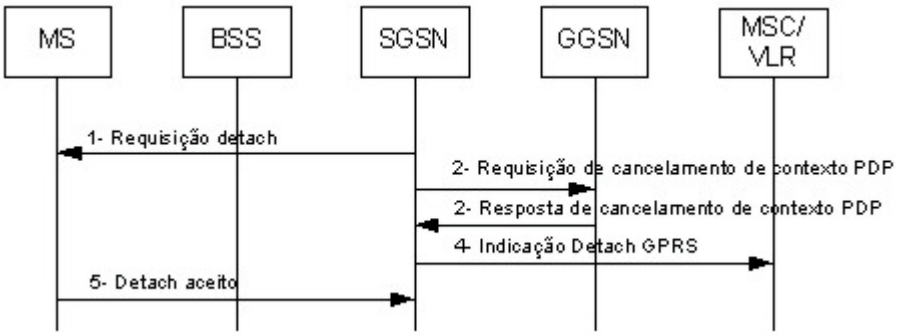


**Figura 3.17** – *Procedimento Detach iniciado pela rede GPRS [25]*

## 3.2.10 Ativação do Contexto PDP

Antes que o terminal móvel possa efetivar a comunicação com a rede de dados externa, o contexto PDP deve ser ativado. O contexto PDP descreve as características da conexão com a rede de dados externa, como: tipo de rede, endereço de rede, APN (*Acess Point Name*), QoS, etc. Para que isso ocorra a MS requisita uma ativação de contexto PDP. O SGSN valida a requisição baseando-se na informação de inscrição recebida do HLR durante o GPRS *Attach*. O APN é enviado para o DNS (*Domain Name Server*) no SGSN para que se encontre o endereço IP do GGSN relevante. Uma conexão lógica é criada entre o SGSN e o GGSN (GTP *tunnel*). O GGSN atribui ao terminal móvel um endereço IP dinâmico da faixa de endereços alocados à

PLMN ou a partir de um servidor RADIUS (*Remote Authentication Dial In User Service*).

Outro procedimento importante para o funcionamento do GPRS é o de localização. No GSM, a rede é dividida em várias áreas servidas pela MSC/VLR. Cada MSC/VLR se estende por um grupo de LAs (*Location Area*) que são compostas por um conjunto de células. No GPRS, um grupo de células é chamado RA. O SGSN controla uma área de serviço que contém várias RAs. As áreas de serviço do SGSN e da MSC/VLR não são necessariamente iguais. Algumas RAs estão dentro de uma mesma LA, assim, ocupando uma menor área geográfica há uma otimização dos recursos de rádio.

Quando uma estação móvel muda de RA, o GPRS precisa atualizar esta informação com os *gateways* do GPRS. A MS envia um pedido de atualização de roteamento contendo a identidade da célula e a identidade da RA anterior para o SGSN que está servindo a MS na sua nova RA. Se a RA é servida pelo mesmo SGSN, a informação de localização é atualizada e um reconhecimento é enviado de volta a MS. Esta atualização de roteamento é conhecida como Intra-SGSN. Caso a RA anterior seja servida por um outro SGSN, o GGSN deve ser informado. É requisitado que o SGSN anterior transmita os pacotes de dados não entregues ao novo SGSN. Além disso, a informação de contexto da MS deve ser removida da memória do SGSN anterior. Esta atualização de roteamento é chamada de Inter-SGSN.

Após a MS ter se registrado com um SGSN da sua rede GPRS, ela precisa alocar um ou mais endereços da camada de rede, chamados de endereços PDP, de modo a enviar/receber pacotes para/de uma MS residindo em uma PDN externa. Essa operação estabelece uma associação entre o SGSN com a qual a MS se registrou e o GGSN que mantém o(s) endereço(s) PDP atribuído(s) a MS, possibilitando a interoperabilidade com PDNs externas.

O registro da associação entre um endereço PDP individual que identifica a aplicação de um assinante móvel, um SGSN e um GGSN, é chamado de contexto PDP. Os seguintes parâmetros podem ser definidos em um contexto PDP:

- Endereço PDP (IP ou X.25) e tipo PDP (IPv4, IPv6, X.25) atribuído à MS.

- Endereço de um GGSN que serve como ponto de acesso para uma PDN externa.

- Perfil de QoS: define a QoS esperada por um assinante móvel em termos dos seguintes atributos: atraso, confiabilidade, precedência de serviço e *throughput*.

Uma MS pode possuir vários endereços PDP ativos simultaneamente e estes podem ser controlados por diferentes GGSNs. Para cada endereço PDP alocado, um contexto PDP com um perfil de QoS diferente pode ser criado e uma cópia de cada um desses contextos ativos é armazenada na MS, no SGSN e no GGSN. A disponibilidade de uma MS para ativação de um contexto PDP ocorre dentro dos estados MM (*Mobility Management*) READY ou STANDBY, onde se pode representar os dois sub-estados conforme ilustrado na Figura 3.18.

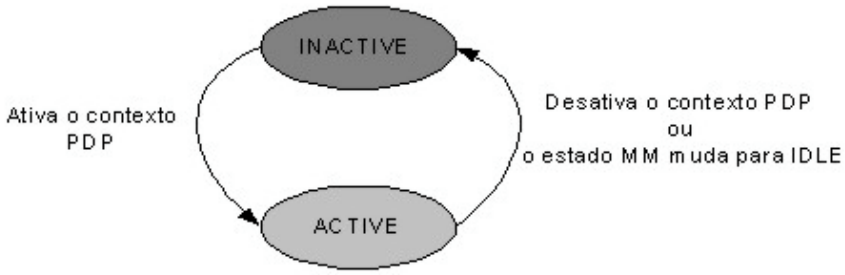


**Figura 3.18 –** *Transição de estados [25]*

No estado INACTIVE o endereço PDP não está ativo para a transferência de dados. O endereço PDP não possui nenhuma informação de roteamento ou mapeamento para o processo de PDP-PDUs. Uma mudança de localização do assinante não causa nenhuma atualização neste estado. A MS inicia a mudança de estado inativo para ativo iniciando o procedimento de ativação do contexto PDP.

No estado ACTIVE o contexto PDP para o endereço PDP é ativado na MS, SGSN e GGSN. Este contexto PDP possui informações de mapeamento e roteamento

para a transferência de PDP-PDUs para o endereço PDP particular entre a MS e o SGSN.

A ativação de um contexto PDP pode ser executada por uma MS ou pela rede (GGSN). Os procedimentos utilizados para a ativação de um contexto PDP iniciado por uma MS estão ilustrados na Figura 3.19.


**Figura 3.19** - *Ativação do contexto PDP pela MS [25]*

- Uma MS envia uma mensagem de "Requisição de ativação do Contexto PDP" para o SGSN contendo as seguintes informações: um endereço PDP, um tipo PDP, um APN opcional que identifica um GGSN através de um endereço IP ou um nome lógico, um NSAPI e um perfil de QoS.

- Para aceitar o perfil de QoS solicitado, o SGSN precisa autenticar a MS e verificar o contrato de serviço assinado com a operadora de rede em termos dos níveis de QoS.

- A qualquer momento, dependendo das condições de carga de uma PLMN GPRS e dos recursos disponíveis, um perfil de QoS pode ser modificado pelo SGSN.

- O SGSN envia o APN recebido a um DNS através de um pedido de busca. O DNS responde ao pedido com uma lista dos GGSNs disponíveis para uso. O SGSN seleciona o GGSN adequado e envia a este uma mensagem de "Requisição de Criação de Contexto PDP". Para enviar essa mensagem, um túnel GTP (ponto-a-ponto) é aberto entre o SGSN e o

GGSN, caso este ainda não exista. Esse túnel é identificado por um TID (*Tunnel Identifier*).

- O GGSN responde ao pedido retornando uma mensagem de confirmação "Resposta de Criação de Contexto PDP" para o SGSN contendo o endereço PDP para a MS e o perfil de QoS negociado. Além disso, o GGSN atualiza uma das entradas de sua tabela onde ele mapeia o TID e o endereço IP do SGSN com a MS associada a eles.

- O SGSN envia uma mensagem para a MS indicando que o contexto PDP solicitado foi ativado corretamente. Finalmente, o SGSN também atualiza uma entrada de sua tabela mapeando a identidade móvel com o TID e o endereço IP do GGSN com o qual ele estabeleceu o túnel.

O GGSN, ao receber um pacote (de uma PDN) destinado a um endereço PDP específico cujo contexto PDP não está ativo, tentará ativar um contexto PDP com a MS correspondente (Figura 3.20), em caso de fracasso o pacote é descartado.

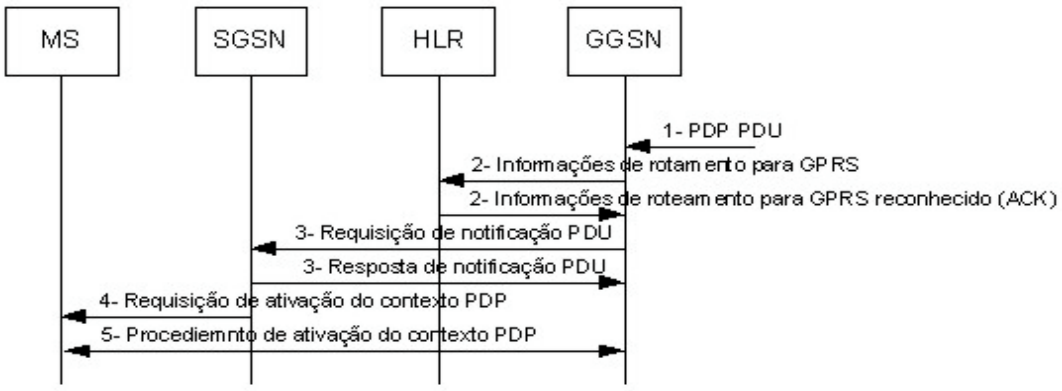


**Figura 3.20** - *Ativação do contexto PDP pela Rede [25]*

O GGSN pode consultar o HLR sobre informações de roteamento necessárias para o endereço PDP. Se o HLR possuir essas informações, ele enviará uma mensagem de confirmação contendo a IMSI e o endereço IP do SGSN servindo a MS. Caso contrário, o HLR informará a causa indicando a razão para a resposta negativa. Se uma resposta positiva for recebida, o GGSN pedirá ao SGSN para enviar uma mensagem à MS (contendo o endereço PDP e o tipo PDP) solicitando a ativação de um contexto PDP.

A desativação do contexto PDP pode ocorrer a partir da MS ou a partir da rede, como segue nas Figuras 3.21 e 3.22.

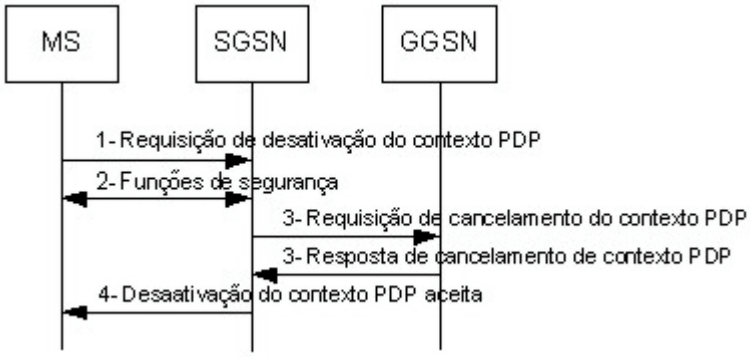


**Figura 3.21** - *Desativação do contexto PDP pela MS [25]*


**Figura 3.22** - *Desativação do contexto PDP pela rede [25]*

## 3.2.11 TBF

Com o contexto PDP ativo e o móvel tendo dados para serem transmitidos é necessário que um segundo contexto de comunicação seja aberto no acesso por rádio através do estabelecimento de um TBF. Um TBF é único para uma dada direção, (*downlink* ou *uplink*) sendo utilizado para identificar uma série de blocos RLC/MAC para/de uma MS específica. Cada TBF possui um TFI (*Temporary Flow Identifier*) que identifica unicamente uma transferência de dados completa e distingue TBFs pertencentes a diferentes MSs que compartilham um mesmo PDCH [27].

Devido à transferência de dados em uma rede GPRS ser tipicamente caracterizada por um tráfego intenso de dados intercalados por períodos de tempos ociosos, o TBF e seu TFI devem ser liberados quando a transmissão/recepção dos

pacotes estiver terminada e o seu recebimento confirmado. Durante o tempo ocioso a MS não transmite nenhum pacote, porém, o serviço GPRS ainda pode estar ativo. Assim, quando um novo pacote precisar ser enviado, um novo TBF deve ser estabelecido. Esse procedimento entre modos ociosos e modos de transferência de pacotes dura até que o serviço GPRS esteja completo.

Para transmitir pacotes *uplink*, um TBF é estabelecido seguindo um dos métodos de acesso randômico: acesso de uma fase e acesso de duas fases.

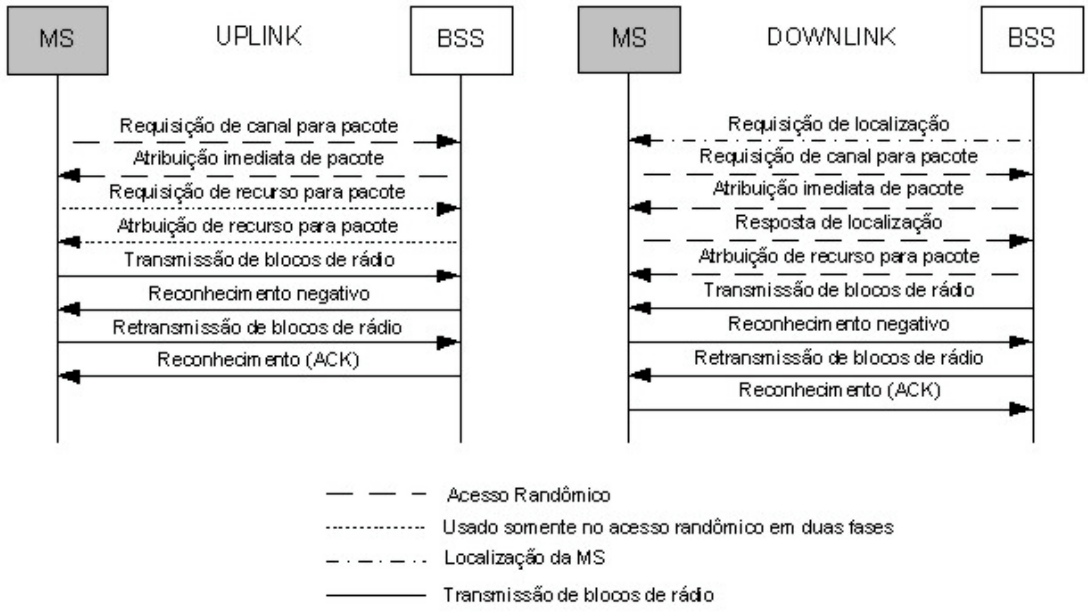


**Figura 3.23** – *Estabelecimento de TBF [25]*

Nos métodos ilustrados na Figura 3.23, a MS primeiramente envia uma mensagem de "requisição de canal para pacote" no PRACH. Cada mensagem desse tipo ocupa um único *time slot*, assim, um bloco PRACH pode acomodar até 4 mensagens de pedido de canal.

Para o método de uma fase, a mensagem de pedido de canal contém o nível de prioridade de rádio e o número de blocos solicitados no(s) PDCH(s). O BSS responde ao pedido enviando a mensagem "atribuição imediata de pacote" no PAGCH contendo a descrição dos PDCHs *uplink* reservados para a MS. Caso o BSS não possua PDCHs disponíveis para atender ao pedido da MS, ele pode escolher

colocar os pedidos em uma fila até que os recursos estejam disponíveis. Nesse caso, o BSS envia uma mensagem notificando a MS no PCCCH.

Para o método de duas fases, a mensagem “atribuição imediata de pacote” contém apenas um bloco PACCH alocado para a MS e utilizado por esta para enviar a mensagem “requisição de recurso para pacote” contendo sua identidade e uma descrição completa dos recursos solicitados para a transferência de dados. Esse pedido é respondido pelo BSS através da mensagem “atribuição de recurso para pacote” no PACCH. Nessa mensagem estão incluídos, entre outros, um TFI único para a MS, o CS utilizado, e a lista do(s) PDCH(s) *uplink* alocados para a MS.

No *downlink*, o BSS inicia a transferência de pacotes enviando uma mensagem de “requisição de localização” no PPCH para localizar a MS destino. A MS destino solicita um canal através da mensagem “requisição de canal para pacote” e o BSS envia a mensagem “atribuição imediata de pacote” reservando os recursos para a MS responder ao pedido de localização. Após a MS enviar a mensagem “resposta de localização”, o BSS envia a mensagem “atribuição de recurso para pacote” contendo a lista de PDCHs *downlink* alocados para a MS.

Para cada TBF atribuído a uma MS particular para uma transferência de pacotes *downlink* ou *uplink*, a PCU aloca blocos específicos de canais lógicos em um ou até oito PDCHs reservados para essa MS. Assim, várias MSs, possuindo diferentes blocos de canais lógicos, podem compartilhar o mesmo *time slot*. Se duas ou mais MSs estiverem compartilhando o mesmo PDCH *downlink* elas devem escutar todos os blocos de rádio *downlink* transmitidos nesse PDCH e comparando o TFI contido nesses blocos com o seu TFI recebido inicialmente durante o estabelecimento do TBF, a MS sabe para quem o bloco de rádio é destinado.

Após os blocos de rádio *downlink* ou *uplink* serem transmitidos nos *time slots* reservados, um pacote de confirmação de recebimento deve ser enviado no PACCH. Quando os blocos de rádio não são recebidos corretamente, gera-se uma confirmação negativa e os blocos listados como errados são retransmitidos.

## 3.2.12 Roteamento

Uma vez que a MS se registrou com o SGSN de sua área de serviço, ativou um contexto PDP e estabeleceu um TBF ela está pronta para transmitir pacotes de dados dentro da infraestrutura de rede GPRS (Figura 3.24).

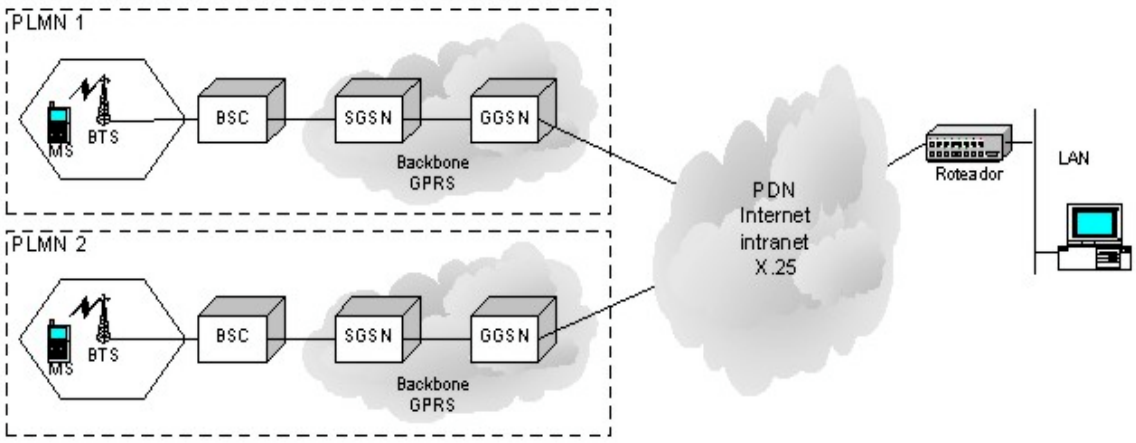


**Figura 3.24 –** *Roteamento de pacotes*

Os pacotes gerados pela aplicação da MS são enviados para a camada de rede. Em cada MS existe uma tabela que armazena informações de mapeamento da sua identidade móvel com o TLLI e o NSAPI correspondente.

A camada SNDCP da MS recebe um pacote de dados original da camada de rede, contendo como cabeçalho o endereço fonte e o endereço destino. Esse pacote de dados é encapsulado em um novo cabeçalho, contendo o TLLI e o NSAPI, e enviado para o SGSN da área de serviço da MS. O SGSN também possui uma tabela que mapeia o TLLI e o NSAPI ao TID correspondente e ao endereço IP de um GGSN. O pacote recebido pelo SGSN é enviado para o endereço do GGSN conectado à rede PDN externa.

A comunicação entre o SGSN e o GGSN utiliza uma técnica conhecida como tunelamento. No processo de tunelamento, o SGSN remove o cabeçalho (TLLI e NSAPI) do pacote recebido e adiciona um novo cabeçalho GTP contendo o seu endereço IP (endereço fonte), o endereço IP do GGSN (endereço destino) e o TID. O

pacote de dados original é então transmitido como o campo de carga útil de forma que ele possa ser roteado através da rede *backbone* GPRS.

O cabeçalho do pacote recebido do SGSN é removido pelo GGSN e o pacote de dados original é convertido para o formato PDP apropriado (endereço IP ou X.25), dependendo da PDN externa. Com base no endereço destino, o pacote é roteado, através de uma PDN até encontrar seu destino, que pode ser uma MS de uma outra PLMN. O GGSN, ao receber o pacote da PDN, procura em suas tabelas o endereço IP do SGSN que está servindo a MS destino e o TID correspondente. O GGSN encapsula o pacote original em um cabeçalho contendo o seu endereço IP, o endereço IP do SGSN destino e o TID.

Ao receber o pacote enviado através da rede *backbone* GPRS, o SGSN mapeia o TID e o endereço GGSN ao TLLI e NSAPI contidos em uma das entradas de sua tabela. Um cabeçalho contendo o TLLI e o NSAPI é adicionado ao pacote e enviado ao PCUSN. O PCUSN estabelece um canal de rádio *downlink* com a MS destino e inicia a transferência do pacote de dados. A camada SNDCP da MS remove todo o cabeçalho do pacote e apenas o pacote de dados original contido no campo de carga útil é enviado para a MS destino.

### 3.2.13 Limitações do GPRS

Apesar de seu grande potencial, o GPRS ainda tem as seguintes limitações:

- As taxas de transmissão são muito menores que as taxas teóricas. Para alcançar o máximo teórico de 172 kbps seria necessário à alocação de oito *time slots* para um único usuário, o que não costuma ser permitido pelos operadores de rede. Mesmo que essa máxima alocação fosse permitida, os terminais GPRS seriam limitados pelo número de *time slots* que podem assegurar.

- Utilizar comutação por pacote significa que os pacotes podem passar por diferentes rotas antes de serem remontados no destino final. Isto pode ocasionar potenciais variações no atraso, afetando a QoS.

- A retransmissão e os protocolos de integridade de dados usados para assegurar que os pacotes transmitidos pela interface aérea não sejam perdidos ou corrompidos também afetam a variação do atraso.

- Protocolo entre BSS e SGSN suporta principalmente aplicações com transferências assíncronas, tornando um desafio implementar interações em tempo real.

## 3.3 Conclusão

O GPRS é uma tecnologia que permite a comutação de pacotes sobre uma rede GSM proporcionando ao usuário GPRS conectividade com outras redes de dados externas.

A comutação de pacotes permite uma melhor utilização dos recursos da rede e acessos mais rápidos com relação à comutação de circuitos utilizada no GSM. Os recursos de rádio são alocados de forma temporária durante a transmissão/recepção dos dados ao invés de alocar os recursos permanentemente até o fim de toda transmissão (conexão).

Esta flexibilidade de atribuição de mais de um canal físico a um usuário reflete num aumento de taxa, assim suportando aplicações de dados, voz e vídeo. Contudo, ainda é um desafio obter QoS e um estudo criterioso neste sentido pode ser uma grande contribuição. No próximo capítulo verifica-se como é possível implementar mecanismos de QoS em redes GPRS para atender diversas aplicações.

# Capítulo IV

## 4. Qualidade de Serviço (QoS) em GPRS

A Internet tem crescido bastante nos últimos anos, o que tem aberto espaço para evolução de novas aplicações. Contudo, é um desafio tornar o ambiente de comutação e tráfego de pacotes do tipo datagrama utilizado pela Internet propício às aplicações multimídia. Assim, é necessário a utilização de mecanismos de QoS para que os recursos de rede provejam o conforto e a satisfação do usuário. O ITU-T define QoS como [39]:

"*Qualidade de Serviço (QoS) é o efeito coletivo de desempenho que determina o grau de satisfação do usuário deste serviço específico.*"

Também podemos encontrar o conceito de QoS expresso de outra forma [40]:

"*Qualidade de Serviço (QoS) é um requisito da(s) aplicação(ões) para a qual exige-se que determinados parâmetros (atrasos, vazão, perdas, ...) estejam dentro de limites bem definidos (valor mínimo, valor máximo).*"

A Qualidade de Serviço pode ser implementada de várias formas. Pode-se utilizar mecanismos de alocação de recursos, fazer uma otimização do desempenho ou ambos. A rede deve prover QoS baseado na solicitação da aplicação através do SLA (*Service Level Agreement*). Os parâmetros negociados servem como referencial para que seja fornecida a qualidade solicitada. Dentre os parâmetros mais comuns encontramos: vazão, atraso, *jitter*, perda de pacotes, taxa de erros e disponibilidade.

## 4.1 Visão Geral de Qualidade de Serviço

A cada dia que passa, a Internet se torna uma ferramenta inevitável para troca de informações. Além de impactar no dia a dia dos usuários residenciais, a Internet também provê uma revolução nas pesquisas e estratégias corporativas.

A Internet teve inicio com a ARPANET (*Advanced Research Projects Agency Network)*, uma rede de dados experimental fundada pela DARPA (*Defense Advanced Research Projects Agency*) nos Estados Unidos nos anos 60. O objetivo principal era construir uma rede robusta e flexível que fosse capaz de sobreviver a ataques militares tais como bombardeios. Para alcançar isso, foi desenvolvido o modelo de datagrama, no qual cada pacote individual é encaminhado de forma independente na direção do seu destino. A rede Datagrama é robusta, simples e tem habilidade de se adaptar automaticamente às mudanças da topologia da rede. Por muitos anos a Internet foi utilizada por cientistas como uma rede de pesquisas e troca de informações. O acesso remoto, a transferência de arquivos e *e-mails* foram às aplicações mais comuns. Atualmente, esta rede é uma gigantesca rede pública. Novas aplicações, tais como vídeo-conferência, pesquisa, mídia eletrônica e telefonia pela Internet são uma realidade cada vez mais próxima que vem ocorrendo com uma velocidade sem precedentes. Podemos citar com aplicação bem sucedida o *E-commerce* que revolucionou a maneira de se fazer negócios [35].

A expansão e o sucesso da Internet motivaram o desenvolvimento de inúmeras aplicações com diferentes exigências, para as quais não foi originalmente projetada e assim foi comprometido o seu desempenho. O modelo datagrama no qual a Internet é baseada tem pouca capacidade de gerenciamento de recursos dentro da rede e não pode prover garantia de recursos para o usuário. Algumas aplicações, tais como vídeo conferência, requerem recursos mínimos para uma operação efetiva.

Uma outra questão seria os serviços diferenciados. Como a Internet trata os pacotes todos da mesma maneira, então esta só pode oferecer um único nível de serviço. Aplicações como telefonia sobre Internet são sensíveis à latência e perdas de

pacotes. Em contraste a isso, a transferência de arquivos pode tolerar atrasos sem a degradação de seu desempenho.

O modelo de serviço tradicional oferece somente serviço de Melhor Esforço (*Best Effort*), que não dá nenhum tipo de garantia de que os pacotes enviados na rede chegarão ao seu destino. Este tipo de serviço não é adequado para tráfego sensível a atrasos, perdas de pacotes ou variações de atraso. Estas distorções podem diminuir a qualidade da transmissão em tempo real, podendo até inviabilizá-la.

A capacidade de prover garantia de recursos e serviços diferenciados é chamada de QoS. Com a introdução de QoS, podemos reservar banda para tipos diferentes de fluxo de dados, onde os pacotes não são descartados e a banda não excede valores pré-definidos. Isto pode ser conseguido através da inclusão de controle de tráfego, alocação e gerência de recursos, policiamento, controle de admissão, entre outros.

Existem duas arquiteturas de alocação de recursos dentro da Internet:

- Serviços Integrados
- Serviços Diferenciados

De um outro lado tem-se o MPLS (*Multi-Protocol Label Switching)* e a Engenharia de Tráfego, que provêem um conjunto de ferramentas de gerenciamento para aprovisionamento de banda e otimização de desempenho [34].

## 4.2 Parâmetros que influenciam na QoS

O nível de QoS em uma rede é um parâmetro subjetivo, sendo mensurado a partir da monitoração de parâmetros específicos definidos no contrato de prestação de serviço estabelecido entre o provedor e o usuário. Os principais parâmetros que influenciam no QoS são:

- Largura de Banda

- Transparência Temporal
- Transparência Semântica
- Disponibilidade

A Largura de Banda disponível na rede e a forma de compartilhamento desta largura para as diversas aplicações influenciam diretamente na QoS. A Largura de Banda é o parâmetro mais básico para o fornecimento da qualidade de serviço (somente toma-se vazão como parâmetro de QoS para aplicações de taxa constante). As aplicações geram vazões que devem ser atendidas pela rede. Vazões para diversas aplicações são apresentadas na Tabela 4.1. Deve-se tomar o cuidado para não se confundir a vazão (*throughput*) efetiva com a utilização do link. Uma rede com utilização alta não quer dizer que as informações estão sendo passadas com sucesso. Ela pode estar retransmitindo muito, pois a taxa de erro poder ser alta. O ideal é que uma rede com utilização alta, tenha uma vazão efetiva alta [33].

| **Aplicações** | **Vazão** |
|---|---|
| Aplicações Transacionais | 1 kbps a 50 kbps |
| Quadro Branco (*Whiteboard*) | 10 kbps a 100 kbps |
| Voz | 10 kbps a 120 kbps |
| Aplicações Web (WWW) | 10 kbps a 500 kbps |
| Transferência de Arquivos (Grandes) | 10 kbps a 1 Mbps |
| Vídeo (*Streaming*) | 100 kbps a 1 Mbps |
| Aplicação Conferência | 500 kbps a 1 Mbps |
| Vídeo MPEG | 1 Mbps a 10 Mbps |
| Aplicação Imagens Médicas | 10 Mbps a 100 Mbps |
| Aplicação Realidade Virtual | 80 Mbps a 150 Mbps |

**Tabela 4.1** – *Vazão para diversas aplicações [33]*

A Transparência Temporal corresponde ao atraso de pacotes e a variação do atraso, chamado de *Jitter*. O atraso de pacotes é basicamente o intervalo de tempo entre o envio do pacote pelo transmissor e a recepção. O atraso tem um comportamento aleatório devido à intensidade do tráfego da rede. O *Jitter* é a variação do atraso gerado principalmente pela variação da carga da rede. O ideal seria ter atrasos pequenos e constantes. Uma forma de contornar este problema é

utilizar um *buffer* na recepção que acrescente um atraso determinado, de forma a "nivelar" todos os atrasos.

Os componentes do atraso podem ser fixos e/ou variáveis:

- Atraso de propagação – este atraso é fixo e representa o tempo que o sinal leva para se propagar no meio de transmissão, o que depende do meio e da distância.

- Atraso de codificação e decodificação – este atraso é fixo e depende do algoritmo, tamanho do quadro e da capacidade de processamento.

- Atraso de empacotamento – é o tempo necessário para preencher o *payload* do pacote com os dados.

- Atraso nos nós da rede – o principal agente deste atraso é o enfileiramento nos roteadores, mais o tempo para decisão de roteamento e a transferência do *buffer* de entrada para o de saída.

- Atraso devido ao *Dejitter Buffer* – a variação do atraso da rede ocorre principalmente devido ao enfileiramento dos pacotes nos roteadores, que pode ser compensada por *buffers*. Entretanto, se a variação do atraso for grande, o atraso adicional para compensar a variação pode ser inaceitável.

A Transparência Semântica determina a possibilidade da rede transportar as informações livres de erro. O número de erros ponto-a-ponto introduzidos pela rede tem que ser aceitável para cada serviço (cada aplicação demanda sua própria QoS).

Normalmente, quando se quer medir a taxa de erro de bits, mede-se no nível da transmissão (camada 1 do modelo OSI) e quando se quer medir a taxa de erro de pacote usa-se o nível da multiplexagem / comutação (camada 2 e 3 do modelo OSI) [36].

Devido à intolerância ao atraso, protocolos como TCP para transmissão confiável não podem ser utilizados e assim a perda é inevitável. A perda de pacotes pode ocorrer devido a imperfeições na transmissão (problemas físicos, etc.), atraso

que exceda o "*Time-to-Live*" e congestionamento que resulte em estouro de capacidade do *Buffer*.

A disponibilidade seria uma medida do quanto a utilização da rede para um serviço está disponível. A disponibilidade é medida geralmente como uma porcentagem do dia, da semana, ou do mês onde o recurso poderia ser usado, como 99,99%. Este parâmetro é fundamental para o funcionamento de vários campos de negócio de empresas, principalmente se houver transações financeiras. Pode-se citar a disponibilidade das redes públicas e dos equipamentos presentes no caminho como fatores chave para uma QoS aceitável.

A sensibilidade para diversas aplicações é apresentada na tabela 4.2.

| Tráfego | Sensibilidade | | | |
|---|---|---|---|---|
| | Largura de Banda | Perda | Atraso | *Jitter* |
| Voz | muito baixo | médio | alto | alto |
| *E-commerce* | baixo | alto | alto | baixo |
| Transações | baixo | alto | alto | baixo |
| *E-mail* | baixo | alto | baixo | baixo |
| Telnet | baixo | alto | médio | baixo |
| Busca casual | baixo | médio | médio | baixo |
| Busca séria | médio | alto | alto | baixo |
| Transferência de arquivos | alto | médio | baixo | baixo |
| Vídeo conferência | alto | médio | alto | alto |
| *Multicasting* | alto | alto | alto | alto |

**Tabela 4.2** – *Sensibilidade para diversas aplicações [32]*

## 4.3 Alocação de Recursos

Fundamentalmente, muitos problemas que encontramos na Internet são resolvidos com alocação de recursos. O atraso e perda de pacotes decorrem da incapacidade de alocação de recursos para atender a demanda de tráfego.

Uma rede deve alocar recursos tais como largura de banda e *buffer* em função das características e requisitos da informação transportada. Uma rede com QoS tem condições de decidir quem deve utilizar os recursos e o quanto.

A Internet convencional trata cada pacote individualmente e exatamente da mesma maneira, utilizando a disciplina FIFO (*First In First Out*). Não existe um controle de admissão se o usuário desejar injetar pacotes tão rápido quanto for possível.

A Internet convencional trabalha com o protocolo TCP nos hospedeiros para detectar congestionamento na rede e reduzir a taxa de transmissão de acordo com a ocasião. O TCP usa um esquema baseado em janela para controle de congestionamento. A janela corresponde à quantidade de dados que pode ser transmitida do transmissor para o receptor. Se a fonte TCP detecta perda de pacotes, esta reduz a taxa de transmissão, reduzindo o tamanho da janela pela metade, e a aumenta gradativamente se houver largura de banda disponível. Este mecanismo é eficaz em um grupo pequeno. Contudo, para a Internet, que tem dimensões extremamente grandes, isso se torna impraticável.

O serviço de Melhor Esforço representa o mais simples que a rede pode oferecer e não oferece nenhuma forma de assegurar recursos para o tráfego. Quando um enlace está congestionado, os pacotes que chegam excedem a capacidade do *buffer*. Assim, este método é passível de congestionamento. Dessa forma, o serviço Melhor Esforço é tolerável para aplicações de transferência de arquivos e *e-mails* e não satisfaz as necessidades das novas aplicações multimídia.

Os serviços Integrados (*IntServ*) e os Serviços Diferenciados (*DiffServ*) representam duas diferentes soluções. O *IntServ* provê garantia de recursos através da reserva desses recursos para fluxos de aplicação individual, enquanto o *DiffServ* usa a combinação entre policiamento avançado, aprovisionamento e priorização de tráfego.

### 4.3.1 Serviços Integrados

A filosofia adotada pelo *IntServ* diz que "*para haver a garantia da qualidade de serviço, deve-se realizar uma reserva dos recursos da rede*" [33]. *IntServ* é

caracterizado pela reserva de recursos (largura de banda e *buffer*) por fluxo. O modelo propõe duas classes de serviço além do de Melhor Esforço. Um, é o Serviço Garantido, para aplicações que exigem limites fixos de atraso, e o outro é Serviço de Carga Controlada, para aplicações que necessitam de segurança e um limite de variação de atraso.

O Serviço de Carga Controlada é usado para aplicações de tempo-real tolerante, ou seja, a banda é reservada pelo receptor sem garantias de limites em relação ao atraso. As aplicações devem fornecer aos roteadores uma estimativa do tráfego de dados que irão gerar baseada em um “balde” de fichas. É um serviço de Melhor Esforço melhorado e confiável.

O Serviço Garantido oferece nível assegurado de largura de banda, limite rígido de atraso fim-a-fim e proteção contra perda de pacotes nas filas. Para ter acesso a esse serviço, as aplicações descrevem seus fluxos através de um “balde” de fichas.

A implementação do *IntServ* é feita por quatro componentes:

- Protocolo de sinalização (RSVP).
- Rotina de controle de admissão.
- Classificador.
- Escalonador de pacotes.

O RSVP (*Resource Reservation Protocol*) foi criado como um protocolo de sinalização para as aplicações realizarem reservas de recursos. O processo de sinalização se dá antes da transmissão dos dados e é renovado sempre que necessário. Para haver a requisição dos recursos, existem mensagens que são trocadas entre o receptor e o transmissor, são elas: PATH e RESV. O protocolo RSVP possui duas características básicas: modo de operação simplex e a orientação ao receptor. O processo de sinalização funciona da seguinte forma: a origem envia uma mensagem PATH para o receptor, especificando as características do tráfego.

Os roteadores intermediários enviam a mensagem PATH para o próximo *hop* determinado pelo protocolo de roteamento. Ao receber a mensagem PATH, o destino responde com uma mensagem RESV, que requisita os recursos do fluxo. Os nós intermediários podem aceitar ou rejeitar o pedido. Se a requisição for negada, o roteador enviará uma mensagem de erro para o receptor, e o processo de sinalização é encerrado.

As rotinas de controle de admissão decidirão se uma requisição por recursos de um novo fluxo pode ser garantida sem que comprometa as garantias feitas anteriormente para outros fluxos.

No classificador, os pacotes são marcados de modo que se possa reservar banda para determinado fluxo. São levados em consideração o endereço do destino, a porta e o número de protocolo.

O papel do escalonador é estabelecer políticas de enfileiramento e prevenção de congestionamento nas interfaces dos roteadores. O escalonador gerencia o encaminhamento dos diferentes fluxos de pacotes através de alguma política de filas e outros mecanismos como temporizadores. Deve haver uma comunicação com a interface da camada de enlace de dados para controlar a alocação da largura de banda entre os fluxos (Figura 4.1).

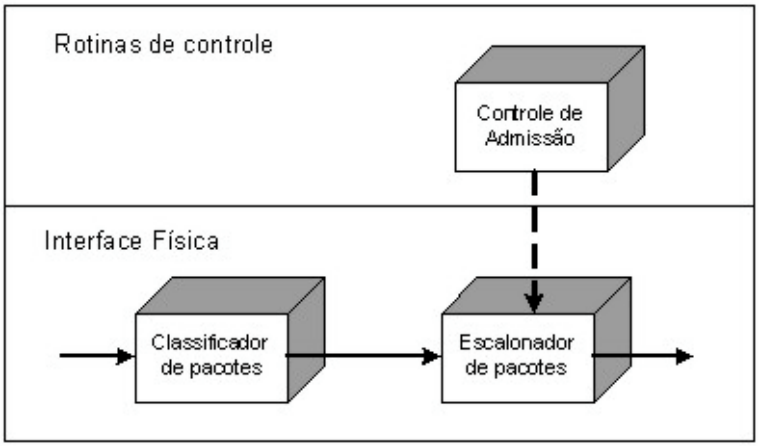


**Figura 4.1** – *Roteador com Intserv [33]*

Vários algoritmos podem ser usados para este fim, tais como, PQ (*Priority Queuing*), WFQ (*Weighted Fair Queuing*), etc.

Desvantagens do *IntServ*:

- A quantidade de informação de controle aumenta com o número de fluxos, exigindo muito espaço de armazenamento e gerando sobrecarga de processamento nos roteadores.
- Todos os elementos da rede devem implementar RSVP, controle de admissão, classificação e escalonamento de pacotes.

### 4.3.2 Serviços Diferenciados

A filosofia de funcionamento baseia-se na premissa de que "*a qualidade de serviço pode ser garantida através de um mecanismo de priorização de pacotes na rede*" [33].

O *DiffServ* (Figura 4.2) foi introduzido devido à grande dificuldade de implementação e desenvolvimento do *Intserv*. O *DiffServ* oferece QoS na Internet com escalabilidade, sem estado para cada fluxo e sinalização a cada nó. A escalabilidade pretende ser obtida através da agregação de fluxos em grandes conjuntos BA (*Behavior Aggregate*), aprovisionamento de recursos para essas agregações (sem protocolos de reserva) e separação das funções dos roteadores de núcleo e borda. Apesar de ser escalável, o *DiffServ* não oferece a garantia de recursos para todos os fluxos como o *IntServ*. As reservas de recursos são feitas para agregações, e um fluxo individual pode não atingir suas necessidades em termos dos parâmetros de QoS.

A priorização pode ser feita no IPv4, por exemplo, através do campo TOS (*Type Of Service*) de seu cabeçalho, e no IPv6, através do campo *Traffic Class*. Obtêm-se assim as Classes de Serviço.

Os roteadores ordenam os pacotes de entrada em diferentes classes de encaminhamento, de acordo com os valores correspondentes de DS *Field*.

Existem dois tipos de classificação dentro do *DiffServ*:

- BA (*Behavior Aggregate*), que é baseado somente no campo DSCP (*Diffserv Code-Point).*

- MF (*Multi-field*), baseado em endereço do destino, endereço da fonte, número da porta e a classificação do campo DS.

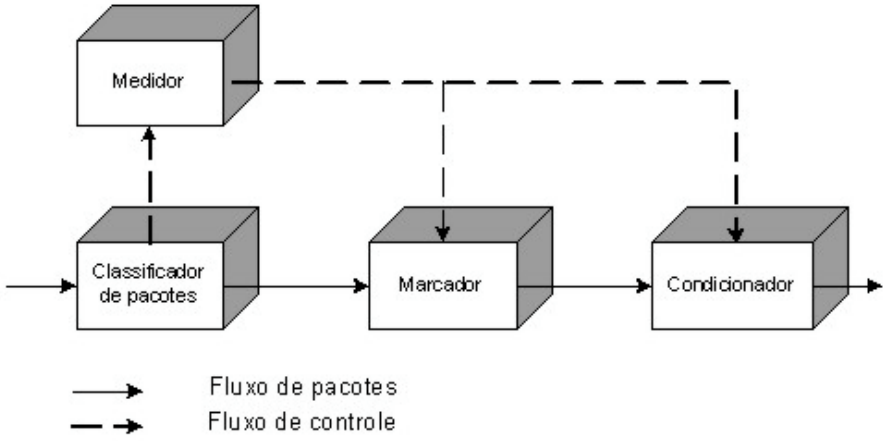


**Figura 4.2** – *Arquitetura funcional do DiffServ [33]*

O *DiffServ* parte do princípio de que domínios adjacentes tenham acordo sobre os serviços disponibilizados. As operações de classificação, policiamento e condicionamento são necessárias somente nos limites da rede (bordas). No centro da rede, os roteadores classificam e encaminham os pacotes. Os pacotes poderão receber tratamentos distintos em função da QoS que necessitam. Esse tratamento de encaminhamento é chamado de PHB (*Per-Hop-Behavior*). O PHB é o modo como um roteador central aloca recursos para os BAs, e é baseado nesse mecanismo local que serviços são construídos. A maneira mais simples de implementar um PHB é destinar a ele um determinado percentual de utilização da largura de banda de um enlace de saída [33].

Os grupos de PHBs que estão sendo padronizados são [33]:

- AF (*Assured Forward*): para aquelas aplicações que necessitam de uma confiabilidade maior do que o serviço de Melhor Esforço, embora não garanta qualquer limite para atraso e *jitter*. O nó deve alocar uma quantidade de *buffer* e banda para cada classe, e a fila para os pacotes, AQ

(*Assured Queue*), é gerenciada por um esquema chamado RED (*Random Early Drop*) para evitar congestionamento. Se uma AQ for substituída por outras três filas, gerenciadas pelo algoritmo WFQ, poderão ser oferecidos serviços de encaminhamento diferentes (serviços ouro, prata e bronze).

- EF (*Express Forward*): para aplicações que necessitem de pouco atraso, *jitter* e taxa de perda, além de garantia de banda. É chamado de "linha privada virtual" ou serviço *premium*. (Ex: telefonia via Internet, videoconferência, etc.). As filas na interface de saída devem permanecer descongestionadas, sendo assim, a taxa de saída dos pacotes do nó é maior que a taxa de entrada. Os pacotes entrarão em filas *premium* (PQ) para serem enviados, podendo-se utilizar políticas de escalonamento de filas como o WFQ, WRR (*Weighted Round Robin*) e CBQ (*Class-based Queuing*).

- CSC (*Class Selector Compliant*): grupo criado com o objetivo de manter a compatibilidade com as implementações existentes de IP *Precedence*, um padrão antigo para especificação de pacotes IP.

Além desses grupos, um PHB *default*, ou PHB BE (*Best Effort*) foi definido para encaminhamento de tráfego de Melhor Esforço.

As regras de classificação, policiamento, condicionamento e escalonamento usadas nos roteadores são determinadas pelo SLA, tanto quanto o espaço nos *buffers* de cada roteador. Um SLA determina que classes de serviços são suportadas e a quantidade de tráfego na banda entre os domínios. No *DiffServ*, o SLA pode ser estático ou dinâmico. Normalmente, o SLA estático é associado ao AF. Já para o EF, é desejável que o provedor suporte SLA estático e dinâmico. O esquema de controle de admissão para suportar SLAs dinâmicos é conhecido como BB (*Bandwidth Broker*). O BB autoriza ou não a entrada de determinado tráfego em seu domínio e sinaliza para outros BB's de outros domínios através dos quais o fluxo irá atravessar [33].

Desvantagens do *DiffServ*:

- Os serviços diferenciados dão segurança ao desempenho das aplicações somente em termos relativos.

- Garantem que uma aplicação gerando tráfego de determinada prioridade terá melhor desempenho que outra gerando tráfego de menor prioridade.

## 4.4 Otimização de Desempenho

Uma vez que a arquitetura de alocação de recursos e os modelos de serviços estão escolhidos, a segunda questão é a otimização de desempenho, isto é, como organizar os recursos numa rede de modo mais eficiente para maximizar a probabilidade de entrega dos pacotes e minimizar o custo de entrega.

A conexão entre a otimização de desempenho e o suporte a QoS pode parecer menos direta comparada com a alocação de recursos. A otimização de desempenho é, no entanto, um importante bloco de construção no desenvolvimento de QoS.

O roteamento baseado em datagramas da Internet não foi projetado para otimizar o desempenho da rede. Escalabilidade e manutenção da conectividade em face das falhas eram os objetivos primários de projeto. Os protocolos de roteamento tipicamente escolhem o menor caminho até o destino, baseados em algumas métricas simples, como contagem de saltos ou atraso. Mecanismos tão simples claramente não são adequados para o suporte a alocação de recursos. Por exemplo, para se fazer uma reserva, é preciso encontrar um caminho com certos requisitos de recursos, como largura de banda, mas o roteamento IP não tem a informação necessária para tomar tais decisões. Usar apenas o algoritmo de menor caminho para selecionar rotas provavelmente causará uma alta taxa de rejeição. O roteamento de menor caminho nem sempre usa as diversas conexões disponíveis na rede. Na verdade, o tráfego é freqüentemente distribuído desigualmente pela rede, o que pode criar pontos de congestionamento enquanto que outras partes da rede podem estar muito pouco carregadas [34].

A otimização de desempenho requer capacidades adicionais no roteamento IP e ferramentas de gerenciamento de desempenho. Para gerenciar o desempenho de uma rede, é necessário ter um controle explícito sobre os caminhos e o tráfego de modo que os fluxos possam ser organizados para maximizar os recursos e a utilização da rede.

O MPLS tem um mecanismo chamado roteamento explícito que é ideal para esse propósito. O MPLS usa a abordagem de comutação de rótulos para estabelecer circuitos virtuais em redes baseadas em IP. Esses circuitos virtuais podem seguir o roteamento IP baseados no destino, mas o roteamento explícito no MPLS também nos permite especificar salto a salto todo o caminho desses circuitos virtuais. Isso provê um modo de ignorar o roteamento baseado em destino e estabelecer troncos de tráfego baseados em objetivos de Engenharia de Tráfego.

O processo de otimização de desempenho das redes através de aprovisionamento eficiente e melhor controle dos fluxos é geralmente chamado de Engenharia de Tráfego. A Engenharia de Tráfego usa algoritmos avançados de seleção de rotas para abastecer os troncos de tráfego dentro de *backbones* e organiza os fluxos de tráfego de forma a maximizar a eficiência global da rede. A abordagem comum é calcular os troncos de tráfego baseado na distribuição do fluxo e então estabelecer os troncos como rotas explícitas com o protocolo MPLS. A combinação do MPLS com a Engenharia de Tráfego provê redes IP com um conjunto de ferramentas avançadas para que os provedores de serviço gerenciem o desempenho de suas redes e tenham capacidade para prover mais serviços a um menor custo.

### 4.4.1 MPLS

O MPLS é baseado em uma  técnica de comutação de rótulos denominada LBS (*Label Based Switching).* A tecnologia LBS usa rótulos pequenas e de tamanho fixo, análogas ao cabeçalho das células do ATM, que são adicionadas aos pacotes de dados que entram na rede MPLS. Os pacotes assim etiquetados podem ser agrupados em categorias. Os pacotes de uma mesma categoria (classificados a partir de algum

critério, como porta de origem e destino, tipo de protocolo, etc..) vão então seguir um mesmo caminho virtual, através da infra-estrutura MPBS.

Com esta tecnologia, o roteamento convencional (também chamado de roteamento implícito ou "*hop-by-hop*") só é feito para o primeiro pacote de uma dada classe (roteamento inicial). A partir do momento que outros pacotes (da mesma categoria) ingressarem na estrutura MPLS, um caminho virtual já estará marcado, e orientado a partir de um esquema de distribuição de rótulos. Com esta infra-estrutura, é possível agregar-se suporte a tecnologias emergentes, dentro do campo da QoS.

O roteamento explícito (todos os pacotes atribuídos a uma mesma classe de equivalência, encaminhados em um mesmo caminho virtual), por si só, já é um fator alavancador de QoS em um ambiente de rede. O roteamento implícito é um dos principais fatores de inserção de atrasos e, principalmente, da variação de atraso ("*jitter*") em uma transmissão de dados (uma vez que os cabeçalhos dos pacotes são analisados "*hop-by-hop*" e nem sempre o caminho que eles percorrem entre a origem e o destino é o mesmo).

Com a possibilidade de classificação dos pacotes em classes de equivalência, a integração da infra-estrutura MPLS com as tecnologias emergentes de aprovisionamento de CoS (*Class of Service*) possibilita a escolha de caminhos mais adequados a tráfego sensível a QoS, atribuídos a estas classes de equivalência.

Além disso, o roteamento explicito implementa o conceito de orientação à conexão, com emulação de circuitos virtuais, mesmo em redes tipicamente não orientadas a conexão, como é o caso das redes IP. Assim, se abre a possibilidade de suporte a serviços de priorização e reserva de recursos, como o *IntServ*.

Esta plataforma se propõe a atingir os seguintes objetivos gerais:

- Escalabilidade em preço e desempenho para toda a rede.
- Independência da camada de enlace (ATM, Ethernet, etc.).
- Independência da camada de rede. O foco principal dos trabalhos atuais é o IP, mas pode operar com redes IPX, Appletalk, DECnet, etc.

- Coexistência com redes ATM nativas e esquemas de roteamento convencionais.

- Facilidade de suporte a redes multiserviço (dados, voz e vídeo) e capacidade de tratar diferentemente tráfegos de Melhor Esforço e Tráfegos sensíveis a QoS.

Dentro deste enfoque, os principais benefícios da plataforma MPLS são:

- Roteamento simplificado: A tecnologia baseada em rótulos faz o encaminhamento de pacotes a partir da análise de rótulos de comprimento fixo e reduzido, que são anexadas ao pacote no ingresso da rede MPLS. Assim, os pacotes podem ser agrupados (de acordo com a informação contida nestes etiquetas) em FECs (*Forward Equivalence Class*), num processo denominado roteamento explícito, efetuado pelos nós principais da infra-estrutura MPLS, denominados LSR (*Label Switching Routers).*

- Roteamento explícito: o MPLS permite que os pacotes sejam classificados, a partir de rótulos atribuídas na admissão dos nós MPLS, e encaminhados, dentro de uma mesma classe, num caminho virtual, sem a necessidade de ser analisado nó a nó.

- Suporte à Engenharia de Tráfego: Entende-se por Engenharia de Tráfego o processo de seleção do caminho pelo qual o tráfego de dados deve ser encaminhado, a fim de propiciar o balanceamento de carga entre vários *links* que interligam roteadores e *switches* na rede. A Engenharia de Tráfego é muito útil em redes como a Internet, onde existem diversos caminhos alternativos. Nas redes baseadas em datagramas, como as redes IP, a Engenharia de Tráfego é dificultada, devido ao caráter de não orientação à conexão. Dentro deste enfoque, a arquitetura MPLS possibilita que fluxos de dados oriundos de um nó de entrada particular, direcionado para um nó de saída particular, possam ser identificados individualmente, como num circuito virtual. Além disso, dentro da estrutura de rótulos, com o roteamento explícito, pode ser habilitado o

encaminhamento destes fluxos a partir de um caminho preferencial denominado LSP (*Label Switching Path*).

- Suporte a QoS: A capacidade do MPLS em estabelecer uma orientação à conexão (no estabelecimento dos LSPs) entre os nós de entrada e de saída preenche uma das principais lacunas das redes IP para implementação de Serviços Integrados. Com o MPLS, o estabelecimento dos LSPs e a incorporação do protocolo RSVP ao plano de controle do roteamento MPLS facilitam a incorporação do *IntServ* nos domínios MPLS.

O advento das FECs e a possibilidade de associar estas classes de equivalência a classes de serviços também impulsionam a utilização desta arquitetura como suporte ao *DiffServ* do IETF [34].

### 4.4.2 Engenharia de Tráfego

Engenharia de Tráfego é o processo de organizar o fluxo de tráfego através da rede para que congestionamentos causados pela utilização desigual da rede possam ser evitados. A Engenharia de Tráfego é direcionada à otimização de desempenho de redes operacionais. Em geral, ela engloba a aplicação de princípios tecnológicos e científicos para medir, modelar, caracterizar e controlar o tráfego e a aplicação dessas técnicas e conhecimentos para atingir determinados objetivos de desempenho.

O objetivo básico da Engenharia de Tráfego pode ser maximizar a utilização dos recursos da rede ou minimizar o congestionamento. Tipicamente, o ponto ótimo de operação é alcançado quando o tráfego é igualmente distribuído pela rede. Com a distribuição balanceada de tráfego, tanto o atraso de enfileiramento quanto as taxas de perda ficam em seus pontos mais baixos.

Os objetivos de desempenho da Engenharia de Tráfego podem ser classificados como:

- Orientados a tráfego: incluem os aspectos relacionados à manutenção das garantias de QoS dos fluxos de dados (ou agregações de fluxos).

- Orientados a recursos: estão relacionados à otimização dos recursos da rede, como impedir que certas partes da rede se tornem congestionadas, enquanto outras permaneçam com recursos ociosos.

Obviamente esses objetivos não podem ser alcançados por meio do roteamento IP baseado em destino. Simplesmente há pouca informação disponível no roteamento IP para possibilitar tal otimização. Na Engenharia de Tráfego, técnicas avançadas de seleção de rotas, geralmente chamadas de roteamento baseado em restrições CBR (*Constraint-Based Routing*), são usadas para se calcular troncos de tráfego baseados nos objetivos de otimização. Para tanto, o sistema de Engenharia de Tráfego geralmente precisa de informações sobre a topologia e as demandas de tráfego de toda a rede. Assim, a Engenharia de Tráfego é tipicamente confinada a um único domínio administrativo.

As rotas produzidas pelo CBR são, provavelmente, muito diferentes daquelas no roteamento baseado em destino. Por essa razão, essas rotas baseadas em restrições não podem ser implementadas pelo encaminhamento baseado em destino. No passado, muitos provedores de serviço usavam ATM nos *backbones* para dar suporte ao CBR. A rede IP era sobreposta aos circuitos virtuais da rede ATM, num modelo conhecido como *overlay*. O MPLS oferece uma melhor alternativa, uma vez que oferece funções similares e ainda pode ser fortemente integrado às redes IP, e a um preço mais baixo. Igualmente importante, o MPLS oferece a possibilidade de se automatizar aspectos da função de Engenharia de Tráfego.

Para que grandes redes suportem a Engenharia de Tráfego sobre MPLS, estas devem dar suporte aos seguintes atributos:

a) Troncos de tráfego: É uma agregação de fluxos de tráfego da mesma classe que são colocados dentro de um LSP. Essencialmente, um tronco de tráfego é uma representação abstrata do tráfego à qual características específicas podem ser associadas. Isso é útil para se ver os troncos de

tráfego como objetos que podem ser roteados, ou seja, o caminho pelo qual o tronco trafega pode ser mudado. Um conjunto de atributos associado aos troncos de tráfego especifica coletivamente seu comportamento:

- Parâmetros de tráfego, como taxa de pico, taxa média, etc.
- Atributos de seleção de caminho e de manutenção (por exemplo, preferência de caminho em caso de múltiplos caminhos definidos administrativamente).
- Atributos de prioridade.
- Atributos de imunidade à falhas, que definem o que acontece numa situação de falha.

b) Atributos de recurso: São atributos associados aos recursos que restringem a colocação dos troncos de tráfego:

- Máximo multiplicador de alocação: é um atributo configurável administrativamente que determina a proporção do recurso que é disponível para a alocação dos troncos de tráfego, sendo mais aplicável à largura de banda do enlace.
- Atributos de classe de recurso: é usado para agrupar os recursos em conjuntos. Pode ser usado para implementar várias políticas que consideram tanto a otimização de desempenho orientada a tráfego quanto à orientada a recurso.

c) Roteamento baseado em restrições (CBR): É uma das ferramentas utilizadas para automatizar o processo de Engenharia de Trafégo. Dessa forma, é uma técnica que auxilia a otimizar a operação dos protocolos que habilitam QoS, onde a forma de operação visa, através de mecanismos de policiamento, computar rotas e distribuir cargas de forma a ir ao encontro dos requisitos de QoS de determinado fluxo ou agregado de fluxos.

Geralmente chamado de *QoS routing*, o CBR na verdade incorpora o roteamento baseado em política e as funcionalidades da Engenharia de Tráfego ao *QoS routing*. De uma forma geral, os objetivos do CBR são:

- Selecionar as rotas que melhor atendam certos requisitos de QoS.
- Maximizar, de forma racional e mais econômica, a utilização da rede.

Normalmente, pode-se usar como métrica na computação de rotas medidas como: latência, variação da latência (*jitter*), custo, contagem de nós e largura de banda. O CBR utiliza um algoritmo chamado NP-*complete*, que usa duas ou mais métricas para computar rotas ótimas de forma a atender a QoS solicitada por determinados fluxos [34].

## 4.5 QoS em GPRS

Uma rede móvel é definida como um sistema que provê aos usuários um acesso sem fio aos serviços de informação. Diferente das redes fixas, que assumem uma baixa taxa de erro e usuários estacionários, existem alguns fatores que dificultam a provisão de garantias de QoS para redes móveis sem fio.

Os recursos em redes sem fio são mais escassos que em redes com fio. Enlaces sem fio, em geral, provêem menos largura de banda que enlaces com fio. Em redes sem fio, os canais são inerentemente não confiáveis e sujeitos a erros devido a ruído, desvanecimento por múltiplos percursos (*multipath fading*), sombreamento (*shadowing*) e interferências. Usuários tendem a mover-se durante a sessão de comunicação causando *handoffs* entre células adjacentes. A tendência atual em redes celulares de reduzir o tamanho da célula de forma a acomodar mais usuários móveis em uma dada área, tornará ainda mais difícil tratar os problemas relacionados à mobilidade. A mobilidade do usuário proporciona vários novos desafios de projeto que não existem em redes tradicionais. A conexão deve ser roteada novamente para o novo ponto de acesso. Largura de banda deve ser alocada na nova localização, o que

pode causar o cancelamento da chamada se a nova localização não possuir largura de banda suficiente. À medida que os usuários móveis movimentam-se, a rede deve rastreá-los e descobrir suas novas localizações a fim de efetuar a entrega de dados.

### 4.5.1 Gerenciamento de QoS em GPRS

Para abordar o conceito de funcionalidade de gerenciamento de tráfego baseado em requisitos de QoS é necessário definir primeiramente quais são os perfis de QoS disponíveis.

#### 4.5.1.1 Perfil de QoS em GPRS

O GPRS associa um conjunto de parâmetros de QoS, denominado “perfil de QoS”, a cada contexto PDP. O perfil de QoS é considerado como sendo um único parâmetro com múltiplos atributos, definindo o QoS esperado em termos de:

- Classe de precedência: esta indica o nível de prioridade dos serviços oferecidos. São definidas três subclasses de precedência:
  - Alta: tem prioridade sobre todos as outras classes de precedência.
  - Normal: tem prioridade sobre a classe de baixa.
  - Baixa: é atendida somente após as classes de alta e normal.
- Classe de atraso: O GPRS não é um serviço "*Store and Forward*", embora os dados sejam armazenados temporariamente nos elementos da rede durante a transmissão. Assim, qualquer atraso ocorrido é devido a características técnicas de transmissão (ou limitações) do sistema e deve ser minimizado para uma classe de atraso específica. O parâmetro de atraso define o atraso médio e atraso máximo em 95 % das transferências fim-a-fim de SDUs pela rede GPRS (Tabela 4.3). Isto inclui o atraso de

acesso ao canal e rádio (*uplink*), o atraso no escalonamento de canal de rádio (*downlink*), o atraso de transmissão no canal de rádio (*uplink* e/ou *downlink*) e o atraso de transmissão na rede GPRS (múltiplos saltos). Este parâmetro não inclui atrasos em redes externas. O atraso é medido entre os pontos de referência R e a interface Gi (Figura 3.3).

| Classe | Tamanho dos pacotes SDU | | | |
|---|---|---|---|---|
| | 128 octetos | | 1024 octetos | |
| | Atraso médio | Atraso máximo 95% | Atraso médio | Atraso máximo 95% |
| 1 | < 0,5 s | < 1,5 s | < 2 s | < 7 s |
| 2 | < 5 s | < 25 s | < 15 s | < 75 s |
| 3 | < 50 s | < 250 s | < 75 s | < 375 s |
| 4 | Melhor Esforço | Melhor Esforço | Melhor Esforço | Melhor Esforço |

**Tabela 4.3** – *Atraso [22]*

- Classe de confiabilidade: O parâmetro de confiabilidade indica as características de transmissão que são requeridas por uma aplicação. A classe de confiabilidade define a probabilidade de perda, duplicação, erro de sequenciamento ou corrupção de SDUs (Tabela 4.4).

| Classe | Prob. de perda de SDU | Prob. de duplicação de SDU | Prob. de SDU fora de seqüência | Prob. de SDU corrompida | Exemplos de aplicações características |
|---|---|---|---|---|---|
| 1 | $10^{-9}$ | $10^{-9}$ | $10^{-9}$ | $10^{-9}$ | Sensível a erros, sem capacidade de correção de erros, capacidade de tolerância a erros limitada. |
| 2 | $10^{-4}$ | $10^{-5}$ | $10^{-5}$ | $10^{-6}$ | Sensível a erros, capacidade de correção de erros limitada, boa capacidade de tolerância a erros. |
| 3 | $10^{-2}$ | $10^{-5}$ | $10^{-5}$ | $10^{-2}$ | Não é sensível a erros, capacidade e correção de erros e muito boa capacidade de tolerância a erros. |

**Tabela 4.4** – *Confiabilidade [22]*

- Classe de vazão: O parâmetro de vazão indica o *throughput* de dados requisitado pelo usuário. É possível renegociar este parâmetro a qualquer hora durante a transmissão. A vazão é definida através de dois parâmetros negociáveis:

  - Taxa máxima de bit (9 classes de 8 a 2048 kbps).

- Taxa média de bit (31 classes de 0,22 a 111000 bps).

Durante condições normais de operação, a rede GPRS tentará alcançar os compromissos de serviço para todos os perfis de QoS. Vários perfis de QoS são possíveis através da combinação desses atributos. Durante a negociação do perfil de QoS, o MS poderá requisitar um valor para cada um dos atributos de QoS. Por exemplo, indicando quais pacotes devem ser descartados na ocorrência de problemas, tais como, congestionamento na rede e limitação dos recursos. A Tabela 4.5 apresenta o elemento de informação de perfil de QoS.

| | bit 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
|---|---|---|---|---|---|---|---|---|
| octeto 1 | QoS IEI (*Information Element Identifier)* | | | | | | | |
| 2 | Tamanho do IE (*Information Element*) | | | | | | | |
| 3 | Spare | | | Classe de confiabilidade | | | Classe de atraso | |
| 4 | Classe de Vazão (Máxima) | | | | Spare | | Classe de precedência | |
| 5 | Spare | | | | Classe de vazão (Média) | | | |

**Tabela 4.5** – *Elemento de informação de Perfil de QoS [32]*

A rede GPRS realiza o controle de admissão com base no perfil de QoS requisitado na mensagem de ativação do contexto PDP e na disponibilidade de recursos. Os algoritmos usados para o controle de admissão não são especificados e dessa forma, estes algoritmos podem ser específicos para cada operadora. Quando a ativação do contexto PDP é realizada com sucesso, o SGSN faz o mapeamento do perfil de QoS definido para um contexto PDP no nível apropriado de prioridade de rádio e indica para o MS que valor de prioridade ele deve usar no acesso *uplink*. Este valor de prioridade, juntamente com a informação do tipo de acesso de *uplink*, se dados ou sinalização, são usados pelo MS na requisição de acesso. No próximo passo o BSS determina a precedência de acesso de rádio por meio da informação provida pela MS. Para o aprovisionamento na rede núcleo do GPRS, o SGSN pode mapear o perfil de QoS no procedimento apropriado, por exemplo, marcando o campo DS (*Differentiated Services*) com o *code-point* apropriado [22].

### 4.5.1.2 GR (*GPRS Register*)

O GR representa o registro GPRS localizado no HLR. Este é responsável por armazenar os dados de perfil de QoS de usuário. Isto provê uma matriz de ponteiros para objetos do tipo perfil de QoS. O tamanho desta matriz é equivalente ao número de usuários de sua área de responsabilidade.

As tarefas do GR compreendem a iniciação da matriz de perfil de QoS de acordo com a especificação requerida pelo usuário, ou seja, *Premium, Standard, Best-Effort* e da distribuição de dados de perfil de QoS de usuário proveniente do SGSN na ativação do contexto PDP. O GR também provê uma interface para o gerador permitir a geração de tráfego em conformidade com o perfil de QoS validado para a transmissão desejada.

### 4.5.1.3 CAC (*Connection Admission Control*)

O CAC é realizado no SGSN e GGSN quando solicitada a ativação de um contexto PDP. O CAC tem a função de gerenciar o tráfego orientado a QoS no ambiente GPRS, ou seja, gerenciar os pedidos de ativação e desativação do contexto PDP, manter informações sobre a carga do tráfego na célula e o estado da utilização de recursos de rádio, controle de fluxo do BSSGP e renegociação de contexto PDP.

Com base na comparação do perfil de QoS solicitado e da disponibilidade de recursos e carga de uma célula em particular, um algoritmo de CAC é executado para decidir se uma determinada aplicação pode ou não ser admitida.

No controle de admissão para o tráfego *downlink*, o SGSN utiliza os parâmetros de controle de fluxo ajustados pelo BSS, tais como: a quantidade de espaço em *buffer* para um dado circuito virtual entre o BSS e o SGSN e a quantidade média de tráfego de dados que o BSS pode transmitir nesse circuito .

O GGSN participa do CAC determinando se ele tem capacidade disponível em sua interface com o SGSN e com outras redes de dados externas.

Os atributos de vazão máxima e média, especificadas no perfil de QoS, podem ser analisados para verificar se há largura de banda disponível na interface aérea, uma vez que esta representa o ponto de “gargalo” da rede. Com respeito aos recursos de rádio de uma rede sem fio, o objetivo de um controle de admissão é limitar o número de conexões de tal forma que, uma vez que uma conexão é admitida, a probabilidade dela encontrar um estado congestionado seja aceitavelmente baixa, provendo a qualidade de serviço exigida. Por essa razão, a probabilidade de bloqueio de uma conexão ativa em uma rede GPRS deve ser utilizada como uma métrica fundamental de QoS para todos os tipos de fontes de tráfego.

Após as exigências terem sido aceitas, o contexto PDP pode ser ativado e a rede GPRS deve ser capaz de fornecer os recursos requisitados durante o período de atividade de uma conexão. Caso contrário, se a rede não for capaz de suprir as exigências solicitadas, um novo perfil de QoS deverá ser renegociado.

#### 4.5.1.4 Alocação de recursos de rádio

A alocação de recursos de rádio é executada pelo BSS, durante o estabelecimento de um TBF, de acordo com o contrato de QoS negociado entre uma MS e a rede GPRS.

Com base na classe de *throughput*, o BSS pode determinar a porção da largura de banda que deve ser alocada, dinamicamente ou estaticamente, para uma determinada aplicação de modo que esta transmita seus pacotes de acordo com a taxa de dados negociada.

Em uma alocação estática (ou fixa), cada canal é atribuído durante o período inteiro de uma conexão. Conseqüentemente, esse tipo de alocação contribui para o desperdício da largura de banda total disponível no sistema quando o canal fica ocioso, isto é, não ocorre transmissão de dados durante o intervalo de tempo no qual a porção da largura de banda foi alocada para a conexão. Por esse motivo, a maioria

dos sistemas opta por alocar seus canais dinamicamente à medida que os mesmos são necessários.

### 4.5.1.5 Policiamento

Após a conexão ser aceita e os recursos serem alocados, o policiamento é executado durante a transmissão de dados para regular o tráfego e assegurar o perfil de QoS negociado. O policiamento pode ser aplicado no SGSN e no GGSN para policiar os pacotes de entrada e manter o tráfego abaixo da taxa máxima admitida. Assim, quando um fluxo mal comportado estiver acima de sua taxa máxima negociada, o comportamento do policiamento pode incluir um atraso na entrega dos pacotes, alterar o perfil de QoS negociado com a aplicação ou descarte. O principal objetivo do policiamento é fornecer justiça entre aplicações pertencentes a uma mesma classe de atraso e precedência de serviço, além de controlar o congestionamento em uma determinada célula.

### 4.5.1.6 Escalonamento

O grande problema da rede GPRS, e das redes sem fio de modo geral, está na alocação de recursos de rádio, uma vez que estes são limitados. Quando existe a contenção de recursos, é fundamental que os recursos disponíveis sejam alocados e escalonados de forma eficiente e justa. Então se faz necessário o uso de sistemas de filas. As filas são utilizadas para armazenar eventos, tais como chamadas telefônicas ou pacotes, ocorridos em tempos aleatórios, organizá-los em seqüência e servi-los de acordo com uma disciplina de escalonamento. No GPRS, uma disciplina de escalonamento de pacotes é implementada por um escalonador localizado na unidade de controle de pacotes (PCU) do BSS. A PCU pode ser instalada na BTS, BSC ou SGSN. Sua localização ideal depende da configuração da rede, contudo a sua localização mais comum é na BSC.

O escalonamento ocorre nos dois sentidos, *downlink* e *uplink*. No *downlink* o escalonamento é feito na chegada de um bloco de dados RLC na camada MAC da BSC enquanto no *uplink* o escalonamento ocorre quando chega um pedido de recurso, contido em um bloco de controle RLC/MAC, para dar início a uma transferência de dados. Um cabeçalho MAC é adicionado a esses blocos, formando um bloco de rádio. A camada MAC da BSC é, então, responsável por escalonar os blocos de rádio pela interface aérea em ambas as direções, com o objetivo de satisfazer os contratos de serviço de todos os assinantes móveis.

O escalonamento ocorre com base no perfil de QoS contido em um contexto PDP associado a um TBF. Os blocos de rádio que chegam no BSC podem ser distribuídos em uma ou mais filas distintas de acordo, por exemplo, com sua classe de precedência de serviço. Dentro das filas, esses blocos são escalonados seguindo uma disciplina de escalonamento.

Existem inúmeros algoritmos de escalonamento, sendo que estes podem ser os mesmos usados nos mecanismos *Intserv* e *Diffserv.*

## 4.5.1.6.1 FIFO

A disciplina de escalonamento do tipo FIFO é a mais básica de todas. Todos os pacotes são tratados igualmente, estes são colocados em uma única fila e servidos na mesma ordem que chegam na fila (Figura 4.3). Esta disciplina também é conhecida como FCFS (*First Come First Served*) [38].

A disciplina do tipo FIFO oferece as seguintes vantagens:

- Carga computacional extremamente baixa se comparada com outras disciplinas.
- Comportamento previsível. Atraso é determinado pelo tamanho da fila.

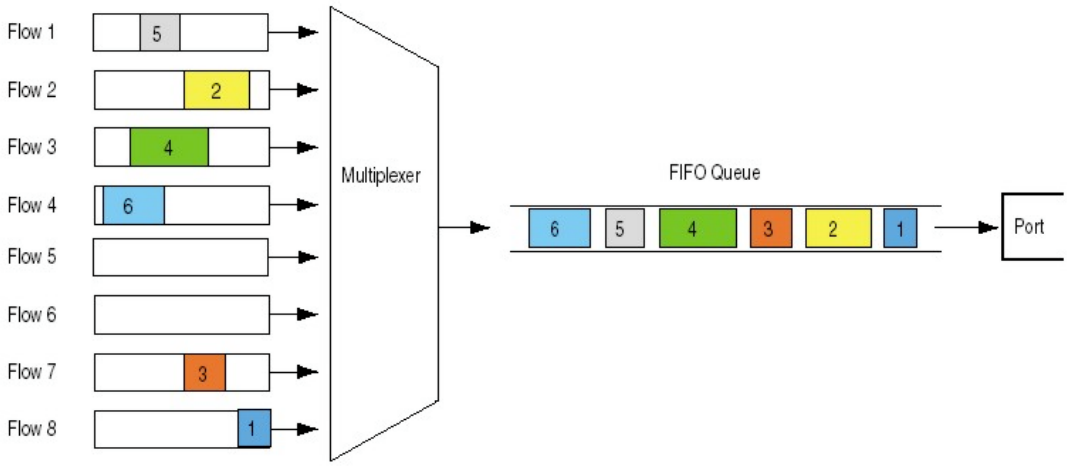


**Figura 4.3** – *FIFO [38]*

As principais limitações são:

- Uma única fila FIFO não permite reordenar/organizar os pacotes armazenados sendo incapaz de priorizar classes de tráfego.

- Uma única fila FIFO trata igualmente todos os fluxos. Assim, o atraso aumenta igualmente para todos os fluxos à medida que ocorre um congestionamento. Como resultado, FIFO pode significar atraso, *jitter* e perda para aplicações de tempo real.

- Durante um congestionamento, a FIFO beneficia fluxos de UDP sobre fluxos de TCP. Quando ocorre perda de pacote durante o congestionamento, as aplicações baseadas em TCP reduzem suas taxas de transmissão, mas aplicações baseadas em UDP permanecem constantes. Assim, o uso da FIFO pode resultar em aumento de atraso, *jitter* e redução da largura de banda consumida pelas aplicações TCP.

- Um fluxo de rajada pode consumir toda a fila e prejudicar os outros fluxos.

## 4.5.1.6.2 PQ

PQ é a base para uma disciplina de escalonamento que suporte classes de serviço diferenciadas. Na PQ clássica, os pacotes são classificados primeiro pelo classificador e então são colocados em filas (FIFO) de prioridade diferentes (Figura 4.4). Os pacotes apenas são retirados de uma determinada fila se as filas de prioridades mais altas estiverem vazias [38].

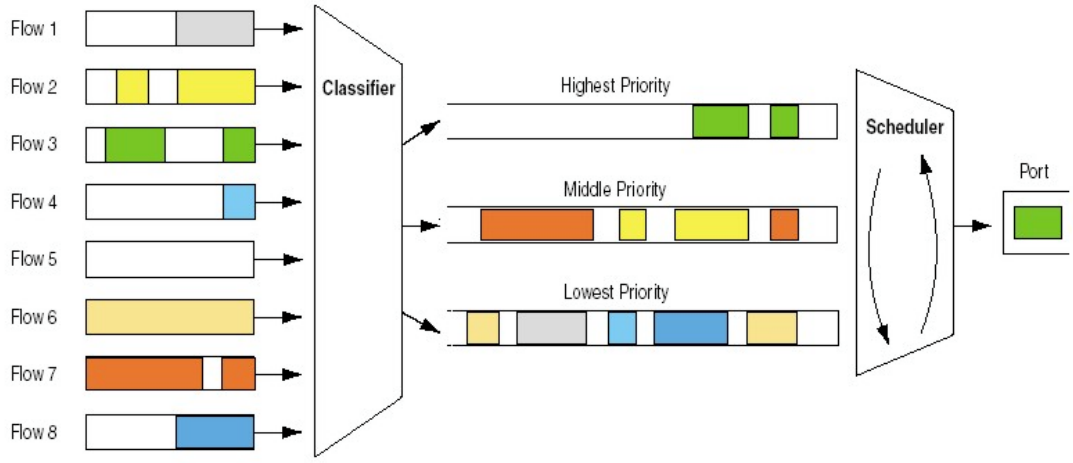


**Figura 4.4** – *PQ [38]*

A disciplina do tipo PQ oferece as seguintes vantagens:

- Carga computacional relativamente baixa se comparada com outras disciplinas.
- Permite organizar as filas e servir classes de tráfego diferentemente de outras classes de tráfego.

As principais limitações são:

- Se ocorrer uma grande quantidade de tráfego de alta prioridade, o tráfego de baixa prioridade pode sofrer atrasos excessivos.
- Pode ocorrer estouro de capacidade das filas de fluxo de baixa prioridade.

- Um fluxo de alta prioridade mal comportado pode causar atraso e *jitter* em outro fluxo dentro da mesma fila.

- PQ não é uma solução para atender a limitação da fila FIFO com relação a favorecer o tráfego UDP.

#### 4.5.1.6.3 WFQ

A disciplina de escalonamento WFQ foi desenvolvida para suprir limitações da disciplina FQ (*Fair Queuing* – não abordada neste trabalho).

A WFQ suporta fluxos com diferentes requisitos de largura de banda, atribuindo a cada fila um "peso" que assegura a diferença de banda por porta de saída. A WFQ também suporta pacotes de tamanhos variáveis, e assim fluxos com pacotes maiores não locam mais banda que os fluxos de pacotes menores, o que a torna bastante interessante.

A WFQ possui um classificador que classifica os fluxos de chegada e os insere em filas (FIFO) que podem ter larguras de bandas diferentes. O classificador também calcula e etiqueta cada o pacote com um número diferente denominado "*Finish Time*". O *Finish Time* representa a ordem com que os pacotes devem ser retirados das filas e encaminhados. Este é calculado a partir da taxa de bit de saída, do número de filas ativas, do peso relativo de cada fila e do tamanho do pacote. O escalonador verifica nas cabeças das filas qual delas possui o pacote com o menor *Finish Time*. O pacote que tiver o menor número é retirado da fila e encaminhado para a saída (Figura 4.5).

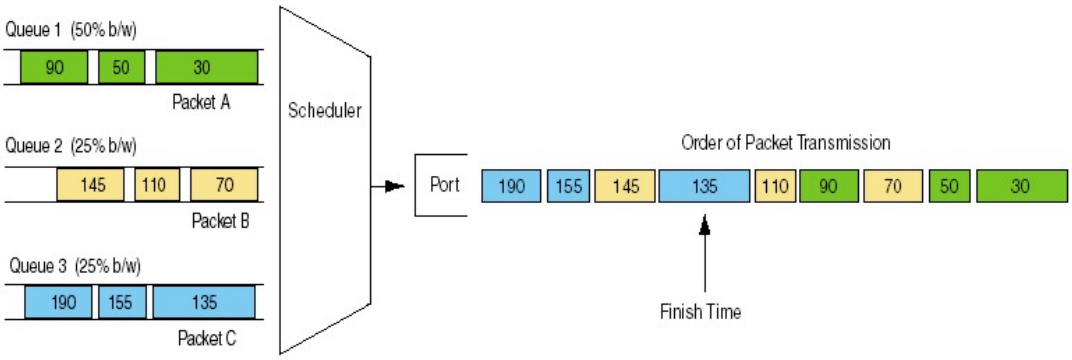


**Figura 4.5** – *WFQ [38]*

O cálculo do *Finish Time* é dado pela fórmula:

$$F_i(k, t) = max\{F_i(k-1, t), R(t)\} + P(k, t) / \phi(i) \quad \text{eq. 4.1}$$

Onde $F_i(k, t)$ é o *Finish Time* para o pacote *k*, da conexão *i* e no tempo *t*. O termo $F_i(k-1, t)$ é o *Finish Time* do pacote anterior, $R(t)$ representa o número do *round* no instante t, $P(k, t)$ é o tamanho do pacote *k* da conexão *i* no instante *t* e $\phi(i)$ é o peso da conexão *i* [37].

A disciplina do tipo WFQ oferece as seguintes vantagens:

- A WFQ provê proteção para cada classe de serviço assegurando um mínimo de largura de banda independente do comportamento de cada classe.

As principais limitações são:

- A implementação é feita apenas em software e não em hardware.
- Fluxo mal comportado pode impactar no desempenho de outros fluxos dentro da mesma classe.
- Algoritmo complexo.

## 4.5.1.6.4 WRR

A disciplina de escalonamento WRR também conhecida por CBQ foi desenvolvida para suprir limitações da disciplina FQ e PQ. O WRR supre as limitações do FQ suportando fluxos com diferentes requisitos de largura de banda. Com o WRR, cada fila pode ser atribuída com diferentes porcentagens de largura de banda de saída. O WRR supre as limitações do FQ assegurando que tráfego de baixa prioridade não é prejudicado. Com o WRR, no mínimo um pacote é removido de cada fila a cada *round* [38].

O WRR primeiro classifica os pacotes que chegam em várias classes de serviço (ex: tempo real, interativo, transferência de arquivos, etc) e então insere os pacotes nas filas dedicadas a cada classe de serviço. Cada fila é servida por um ordenador *Round Robin*.

O WRR suporta a alocação de diferentes quantidades de largura banda para diferentes classes de serviços, ou seja, permite que uma fila com largura de banda maior envie mais que um único pacote cada vez que é visitada durante um ciclo de serviço, ou permite que cada fila envie um pacote cada vez que é visitada, mas uma fila com maior largura de banda pode ser visitada múltiplas vezes em um único ciclo de serviço (Figura 4.6).

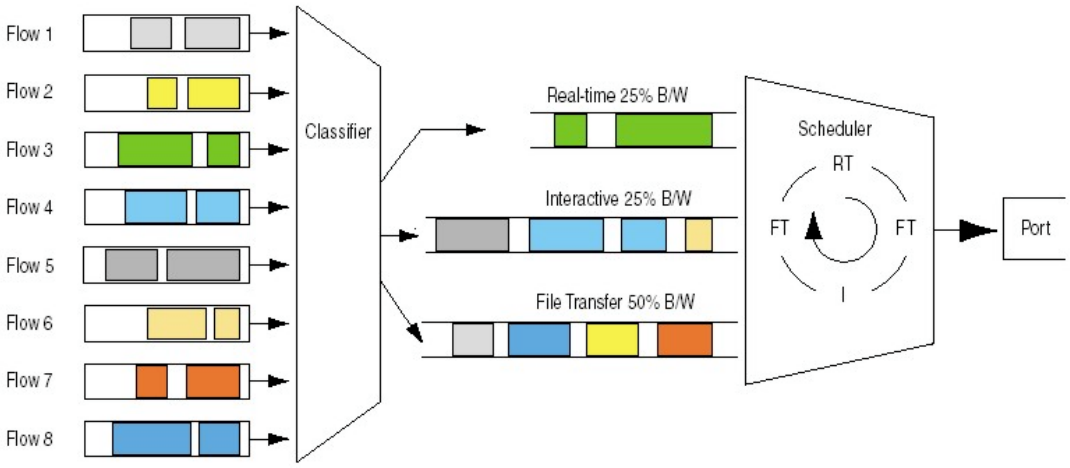


**Figura 4.6** – *WRR [38]*

A disciplina do tipo WRR oferece as seguintes vantagens:

- Pode ser implementado em hardware.
- Provê um controle sobre cada porcentagem de largura de banda de saída.
- Assegura que todas as classes de serviços tenham acesso a uma largura de banda de saída configurada.

A principal limitação é:

- Provê uma correta porcentagem de largura de banda de saída apenas se os pacotes inseridos em todas as filas forem do mesmo tamanho.

## 4.5.1.6.5 DWRR

O DWRR foi desenvolvido para suprir as limitações dos algoritmos de escalonamento WRR e WFQ. O DWRR supre as limitações do WRR através de uma distribuição de largura de banda de saída justa, para filas que contenham pacotes de tamanhos variáveis. O DWRR supre as limitações do WFQ reduzindo a complexidade computacional do algoritmo e também por permitir a sua implementação em hardware, o que possibilita o uso de interfaces de alta velocidade [38].

No DWRR cada fila é configurada com vários parâmetros:

- Um *Weight* define a porcentagem de largura de banda de saída alocada para cada fila.
- Um *Déficit Counter* especifica o número total de bytes ou bits que uma fila pode transmitir cada vez que é visitada pelo escalonador. Se o tamanho do pacote for maior do que o *Déficit Counter*, este último é creditado no saldo da fila para o próximo *round* onde é feita uma nova comparação.

- Um *Quantum* de serviço que é proporcional ao peso da fila e expresso em bytes. O *Déficit Counter* é incrementado de um *Quantum* cada vez que a fila é visitada pelo escalonador.

No algoritmo DWRR clássico, o escalonador visita cada fila não vazia e determina o número de bytes do pacote da cabeça da fila. A variável *Déficit Counter* é incrementada do valor *Quantum*. Se o valor do pacote da cabeça da fila é maior que a variável *Déficit Counter* então o escalonador se move para a próxima fila. Se o tamanho do pacote da cabeça da fila é menor ou igual a variável *Déficit Counter,* então esta é decrementada do tamanho do pacote e o pacote é transmitido pela porta de saída. O escalonador continua retirando os pacotes e decrementando a variável *Déficit Counter* do tamanho do pacote transmitido até que o tamanho do pacote da cabeça da fila seja maior que a variável *Déficit Counter* ou que a fila esteja vazia. Se a fila está vazia o valor *Déficit Counter* recebe zero. Quando isso ocorre o escalonador se move para a próxima fila não vazia (Figura 4.7).

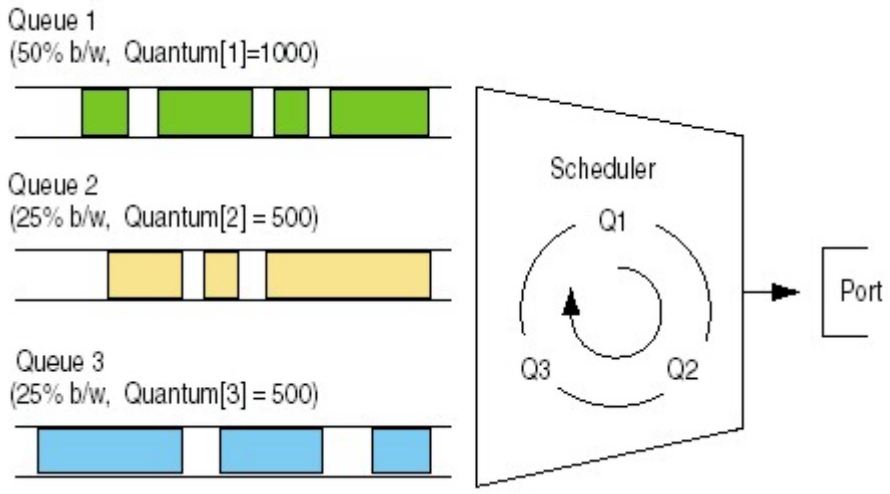


**Figura 4.7** – *DWRR [38]*

O DWRR tem as seguintes vantagens:

- Supera as limitações do WRR provendo um controle preciso da porcentagem de largura de banda de saída alocada para cada classe de serviços quando se encaminham pacotes de tamanhos variáveis.

- Supera as limitações do PQ assegurando que todas as classes de serviços terão acesso a uma quantidade de largura de banda de saída configurada.

- A implementação do algoritmo é relativamente simples.

A principal limitação é:

- Fluxos mal comportados podem impactar no desempenho de outros fluxos dentro da mesma classe de serviços.

## 4.6 Conclusão

A qualidade de serviço em redes móveis tem o intuito de atender os requisitos dos diversos tipos de aplicações num ambiente de espectro de freqüências limitado e com problemas inerentes de uma comunicação sem fio.

Mecanismos de provimento de QoS, tais como Reserva de Recursos e Otimização de Desempenho desenvolvidos e aplicados para as redes IP, podem ser uma solução eficiente para atender os requisitos de QoS no *Backbone* IP da rede GPRS.

Contudo, o “gargalo” da rede GPRS se encontra na interface aérea, sendo necessário um eficiente CAC, policiamento de tráfego, e algoritmos de escalonamentos de pacotes que melhor atendam aos requisitos e características de tráfego da rede.

Embora se encontre na literatura trabalhos sobre CAC e algoritmos de escalonamento, busca-se ainda soluções mais eficientes que atendam tal necessidade. Assim, o presente trabalho dá a sua contribuição através de um modelo computacional que será apresentado no próximo capítulo. Este modelo simula uma rede GPRS e permite fazer uma avaliação comparativa dos diversos algoritmos de escalonamento apresentados no presente capítulo sob condições críticas.

# Capítulo V

## 5. Modelo de simulação GPRSS

Os capítulos anteriores apresentaram a arquitetura e funcionamento da rede GPRS e, também os mecanismos que podem ser aplicados a seus elementos para prover um tratamento diferenciado aos diferentes tipos de mídia e atender aos respectivos requisitos de QoS.

Este capítulo descreve um modelo de simulação que foi desenvolvido com objetivo de fornecer condições para uma avaliação comparativa, consistente, do desempenho da rede GPRS a partir da implementação de mecanismos de provimento de QoS, tais como, algoritmos de escalonamento de pacotes.

Existem várias formas de se avaliar o comportamento da rede GPRS frente à implementação de diferentes algoritmos de escalonamento. Pode-se elaborar um modelo analítico, o qual se baseia em um desenvolvimento matemático, utilizando, por exemplo, equações de um sistema de filas de modo a gerar medidas de desempenho a partir de parâmetros fornecidos. Pode-se também fazer um experimento real utilizando uma rede implantada para a implementação dos algoritmos e coleta de medidas de desempenho. Optou-se por uma terceira alternativa, a qual se refere a um modelo de simulação que represente com a maior fidelidade possível uma rede real, e ainda permita uma avaliação de seu desempenho.

Para se desenvolver tal modelo denominado GPRSS, fez-se uso da ferramenta OPNET Modeler [41]. Sua escolha se deve ao fato da ferramenta possuir inúmeros recursos de modelagem e análise, bem como ser bastante conhecida e utilizada do meio científico/acadêmico. Esta provê um ambiente de desenvolvimento que suporta a modelagem e simulação de redes de comunicação e sistemas distribuídos baseada em eventos discretos.

## 5.1 GPRSS

O GPRSS é um conjunto de componentes que foi desenvolvido especialmente para representar os equipamentos (hardware e software) de uma rede GPRS real e para oferecer condições de análise de desempenho dentro de um ambiente de simulação. O desenvolvimento de todos os componentes que constituem o modelo fez parte do escopo deste trabalho, e só foi possível após um estudo aprofundado dos planos de transmissão e controle descritos nas recomendações do ETSI. Todos os componentes criados foram disponibilizados em uma biblioteca, o que permite a qualquer usuário do modelo construir cenários diversos. Com a disponibilidade destes componentes em uma biblioteca, uma nova contribuição deste trabalho fica mais evidente, a flexibilidade na implementação de cenários.

Uma rede GPRS real tem como ponto crítico (“gargalo”) a interface aérea, com tráfego no sentido da rede para estação móvel. Deste modo, os componentes utilizados no GPRSS foram desenvolvidos apenas para *downlink* e com alocação fixa de *time slots* (a capacidade de transmissão com múltiplos *time slots* pode ser configurada pelo usuário). Outra observação importante é que o GPRSS faz referência apenas ao plano de transmissão de dados da rede GPRS (Figura 3.4) e não ao plano de controle (Figura 3.5), ou seja, o modelo não dispõe de nenhum tipo de sinalização. Contudo, dispõe-se das tabelas de roteamento (configuradas pelo usuário) que representam as tabelas reais geradas pelo plano de controle após o processo de *Attach* e ativação do contexto PDP. Assim, o roteamento dos pacotes baseado em “túneis” dentro da rede GPRS é feito de acordo com os atributos da tabela de roteamento. É possível também configurar o atributo tempo de início da geração de pacotes simulando o tempo gasto na sinalização inicial.

Para que o comportamento do GPRSS se aproximasse de uma rede real foi necessário que o desenvolvimento passasse por algumas fases de aperfeiçoamento da modelagem e simulação, conforme ilustrado na Figura 5.1.

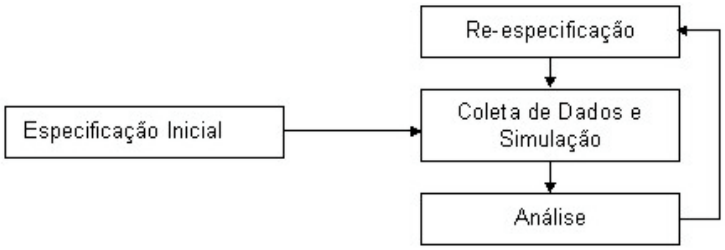


**Figura 5.1** – *Fases de modelagem e simulação*

Estas fases de desenvolvimento do GPRSS ocorreram em três domínios distintos conforme a Tabela 5.1.

| Domínio | Editor | Foco da modelagem |
|---|---|---|
| Rede | Project | Descreve a topologia de rede em termos de subredes, nós, enlaces e contexto geográfico. |
| Nó | Node | Descreve a arquitetura interna do nó em termos de módulos funcionais e fluxo de dados entre eles. |
| Processo | Process | Descreve o comportamento do processo especificado (Protocolos, algoritmos, aplicações) usando maquina de estado finita e linguagem de programação de alto nível. |

**Tabela 5.1** – *Domínios da modelagem*

O desenvolvimento teve inicio com um estudo das especificações da tecnologia a ser modelada. Com base, neste estudo que se refere principalmente as camadas (protocolos) contidas nos planos de transmissão e controle, pode-se fazer um esboço inicial de 25 processos. Cada processo simula o comportamento de uma camada e foi desenvolvido usando a linguagem Proto-C (combinação de diagrama de estados e C/C++) .

Com um esboço inicial dos processos, foi possível desenvolver os nós (9 no total). Cada nó representa um equipamento de rede (hardware e software), e este é constituído de vários processos interligados por *streams* de dados. Com a conclusão dos nós, foi criada uma biblioteca para disponibilizá-los. Cada nó possui atributos configuráveis pelo usuário, isso é feito através de uma interface bastante amigável e de acesso simples. Tais atributos determinam o comportamento dos processos contidos no nó. A definição de quais atributos seriam modelados para cada nó ocorreu de forma a permitir que o comportamento do nó reproduza o real e ainda forçar algumas situações, tal como a escolha do esquema de codificação e o número de *time slots* utilizados (tarefas executadas de forma automática pela rede).

O trabalho no domínio de rede consistiu em montar uma topologia com os nós disponíveis na biblioteca criada, interliga-los através de *links* de comunicação desenvolvidos a partir do estudo das interfaces do GPRS e configurá-los.

Com os resultados obtidos, após várias simulações, foi possível aperfeiçoar os processos, nós e rede de forma a torná-los mais próximos do real.

Apenas os ícones, as FDPs (Função Distribuição de Probabilidade) utilizadas na geração de pacotes, a modulação da interface aérea e os módulos G_rx, G_tx, G_rf_tx, G_rf_rx, G_ant_tx, G_ant_rx, foram obtidos nas bibliotecas padrões do OPNET Modeler. Com exceção da modulação, para todos outros elementos citados acima, fez-se necessária uma configuração especifica que permitisse operação dos mesmos nos cenários propostos.

Todos os demais elementos que constituem o modelo GPRSS, cujo desenvolvimento foi objeto principal deste trabalho, são discriminados de forma detalhada partir do item 5.2, incluindo os códigos dos processos (disponíveis no anexo A).

É importante ressaltar que o termo “usuário” muito utilizado no desenvolvimento deste capítulo se refere ao indivíduo que configura e opera o GPRSS e não o indivíduo que acessa a aplicação a partir de uma estação móvel.

## 5.2 Domínio de rede

Neste domínio foram usados todos os nós disponíveis na biblioteca criada a partir do desenvolvimento deste trabalho. Estes nós foram distribuídos em três ambientes: rede externa de pacotes de dados, rede GPRS e estações móveis.

No domínio de rede pode-se fazer uso de qualquer número de nós. Pode-se também definir o contexto geográfico através de mapas ou apenas dimensionar as distâncias dentro da área de cobertura da rede. Contudo, optou-se por criar uma rede com um número mínimo de nós e aplicações que fossem suficientes para a avaliação

dos mecanismos de QoS implementados. Com relação ao contexto geográfico, adotou-se um área de 190Km$^2$ (10Km x 19Km) onde a rede foi implementada conforme ilustrado na Figura 5.2.

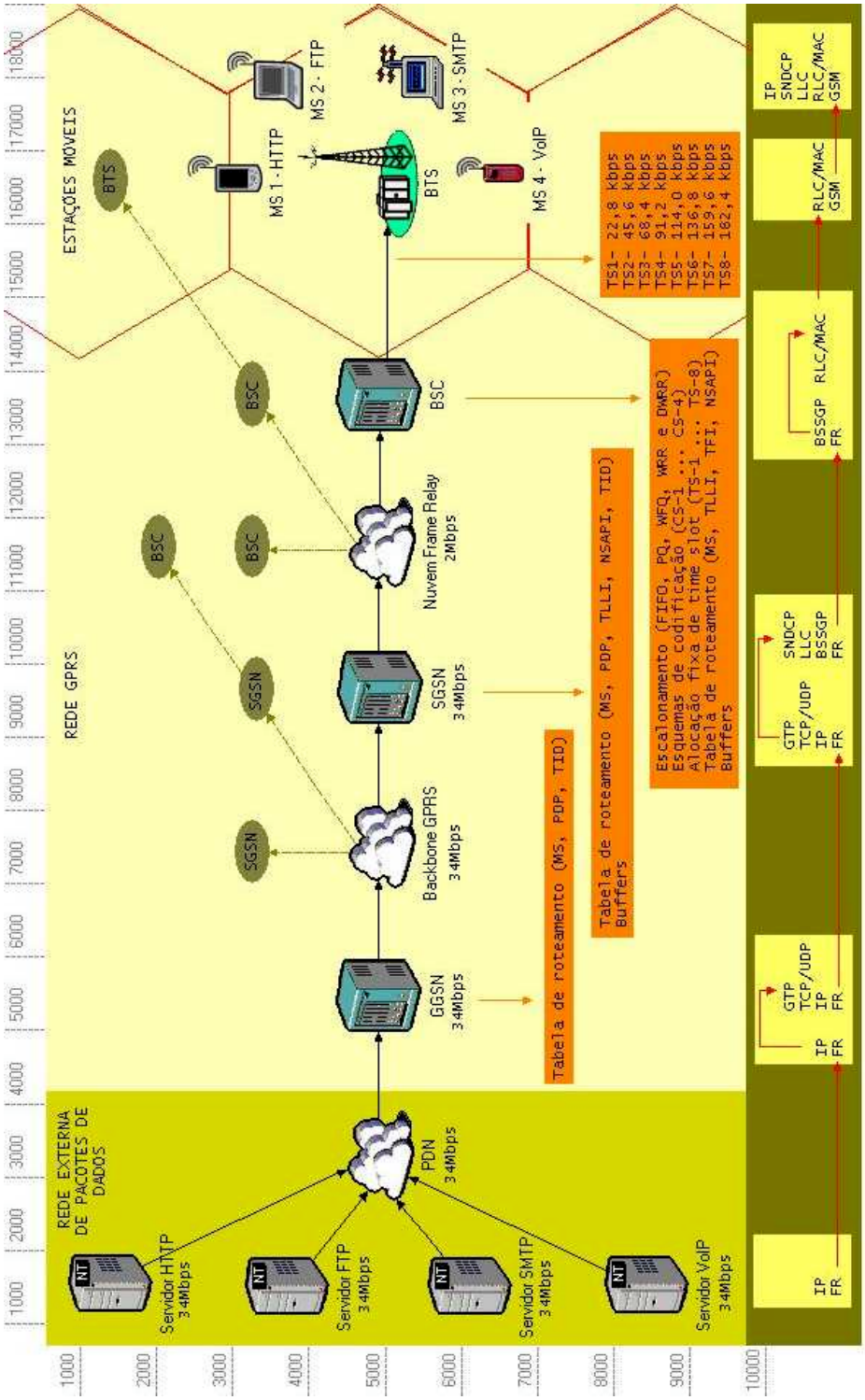


**Figura 5.2** – *Domínio de rede*

A rede externa de pacotes de dados contém servidores de aplicações, os quais geram e transmitem dados para as estações móveis. Esta contém também uma nuvem IP denominda PDN (*Packet Data Network*) que representa a Internet.

A rede GPRS é constituída dos nós GGSN, *Backbone* GPRS, SGSN, Nuvem *Frame Relay*, BSC e BTS, cujos funcionamentos estão descritos com detalhes no capítulo IV.

As estações móveis representam os pontos de acesso às aplicações dentro da área de cobertura de uma célula.

### 5.2.1 Servidor de Aplicação


**Figura 5.3** – *Servidor de Aplicação*

O Servidor de Aplicação (Figura 5.3) é o nó responsável por gerar dados de acordo com atributos definidos pelo usuário.

Os atributos (Figura 5.4) são específicos de cada nó e todos os nós deste modelo requerem uma configuração prévia à execução da simulação. Esta configuração consiste na atribuição de valores pelo usuário. Assim, configurando nó a nó é construída a configuração total da rede. Como o nó Servidor de Aplicação é a fonte de dados do GPRSS é importante definir o tipo de aplicação (HTTP, FTP, SMTP ou VoIP), o formato do pacote a ser gerado (IPv4, *Frame Relay*, TCP, UDP, etc), endereços de fonte e destino do pacote e tempo de início da geração de pacote depois de iniciada a simulação. Também é importante definir a FDP do intervalo entre os pacotes (Exponencial, Constante, Bernoulli, Pareto, etc.), a FDP do tamanho do pacote, o tempo do término da transmissão e atributos do canal de transmissão (taxa de transmissão e tipo de pacote).

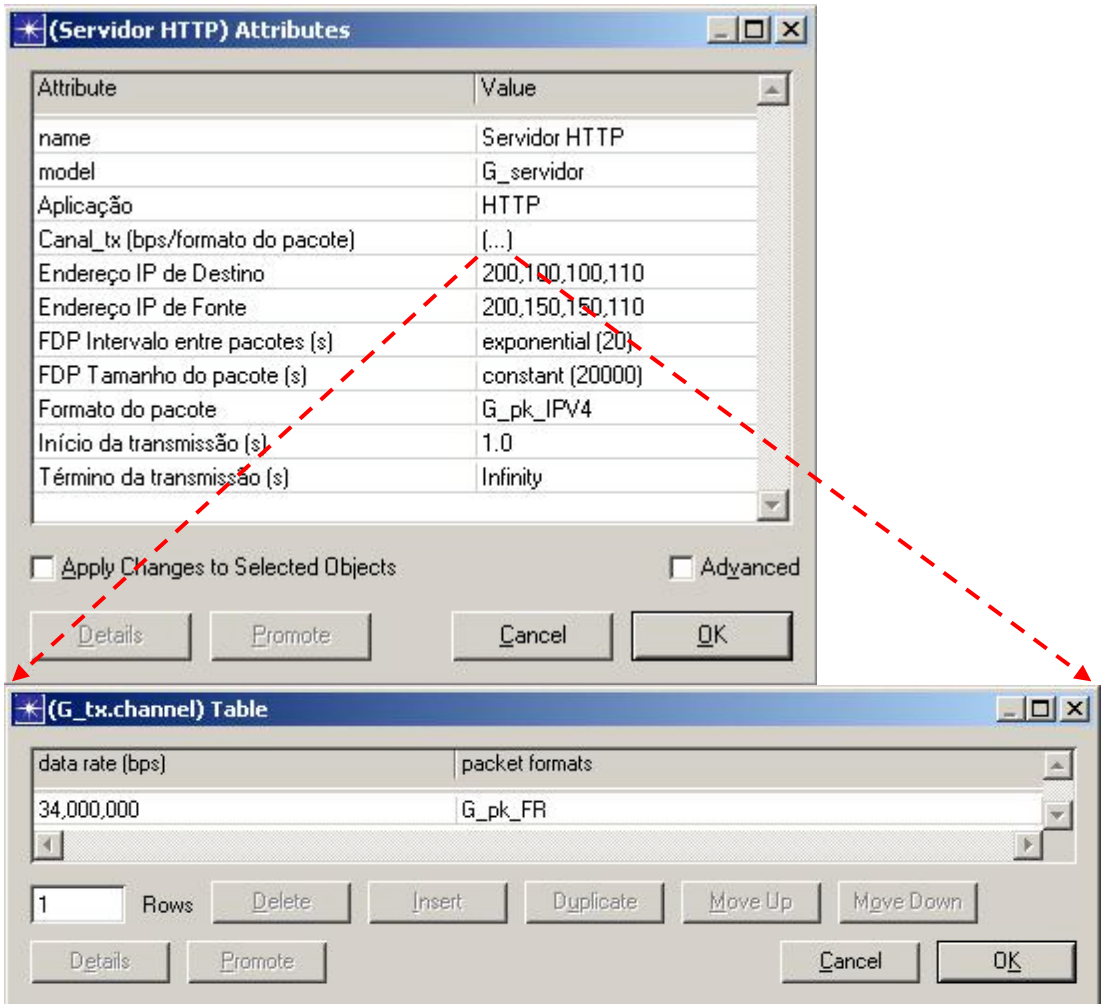


**Figura 5.4** – *Atributos de configuração – Servidor de aplicação*

## 5.2.2 PDN


**Figura 5.5** – *PDN*

A rede PDN (Figura 5.5) representa uma rede de dados externa. Alguns atributos (Figura 5.6) devem ser configurados de forma a aproximar o comportamento deste nó ao comportamento de uma rede de dados real. Assim temos,

a taxa de serviço (processamento) de pacotes da rede PDN como um todo (é usada à disciplina FIFO), o tamanho do *buffer* (capacidade em bits ou em pacotes), e atributos dos canais de transmissão.

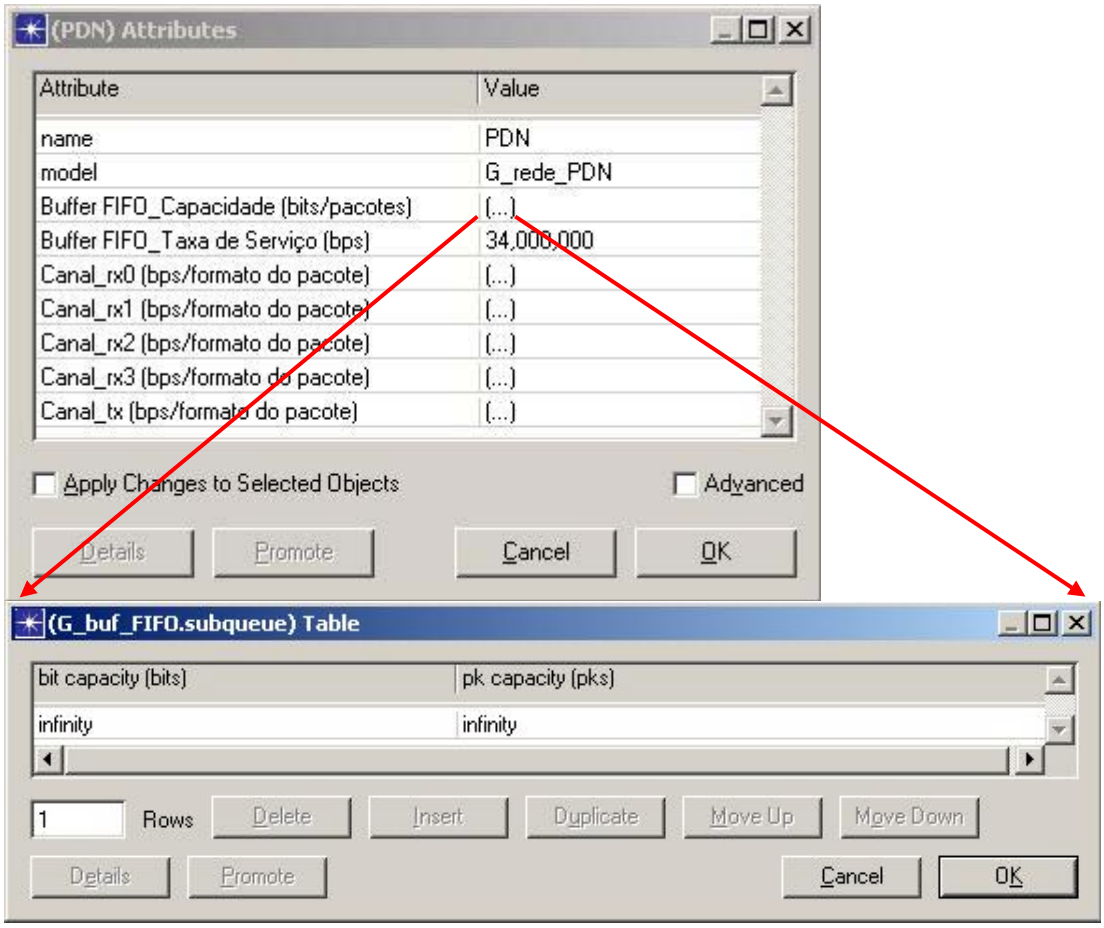


**Figura 5.6** – *Atributos de configuração – PDN*

## 5.2.3 GGSN


**Figura 5.7** – *GGSN*

O GGSN (Figura 5.7) é o nó de ligação/adaptação entre a rede externa e a rede GPRS. Com isso, deve-se configurar os atributos dos canais de transmissão e o endereço do nó GGSN. Além destes atributos, tem-se também a tabela de roteamento que representa a tabela criada de forma automática pelo plano de controle da rede GPRS real. Como o GPRSS se refere apenas a plano de transmissão, é necessário configurar os atributos desta tabela (Figura 5.8), tais como, a identificação da MS, o endereço PDP, o identificador do túnel entre GGSN e SGSN (TID), e o endereço do nó SGSN responsável pela área de roteamento (RA) onde a estação móvel se encontra.

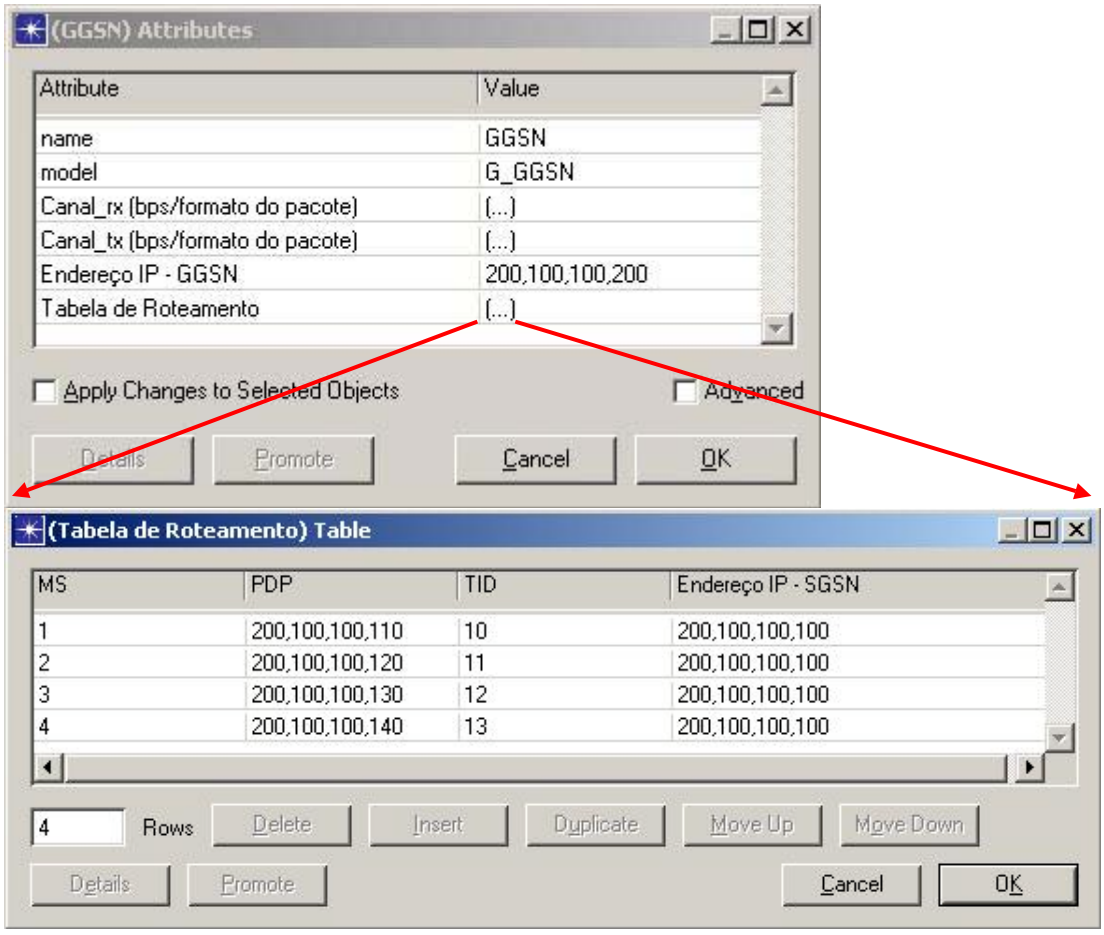


**Figura 5.8** – *Atributos de configuração – GGSN*

## 5.2.4 *Backbone* GPRS


**Figura 5.9** – *Backbone GPRS*

O *Backbone* GPRS (Figura 5.9) representa a nuvem IP da operadora. Assim como no item 5.2.2 deve-se configurar a taxa de serviço de pacotes da rede GPRS como um todo (é usada à disciplina FIFO), o tamanho do *buffer* e atributos dos canais de transmissão.

## 5.2.5 SGSN


**Figura 5.10** – *SGSN*

O SGSN (Figura 5.10) é o elemento responsável pelo gerenciamento de mobilidade da rede. Este nó também possui uma tabela de roteamento onde deve-se configurar a identificação da estação móvel, o endereço PDP, o identificador de enlace lógico temporário TLLI, o identificador de ponto de acesso de serviço da camada de rede NSAPI, o TID e o endereço IP do nó GGSN correspondente à extremidade do túnel.

O SGSN separa internamente os fluxos de dados provenientes do GGSN por estação móvel. Assim os pacotes são inseridos em *buffers* individuais denominados TLLIs responsáveis pelo policiamento do tráfego [30] (este trabalho não possui algoritmo de policiamento de tráfego implementado). Tais *buffers* devem ser configurados quanto a sua taxa de serviço e capacidade. Os fluxos de dados das

saídas dos *buffers* TLLIs são agregados por célula e inseridos em *buffers* denominados BVC (*BSSGP Virtual Channel*) que também requerem uma configuração quanto à taxa de processamento e capacidade. Além de todos esses atributos deve-se configurar os canais de transmissão e endereço do nó SGSN (Figura 5.11).

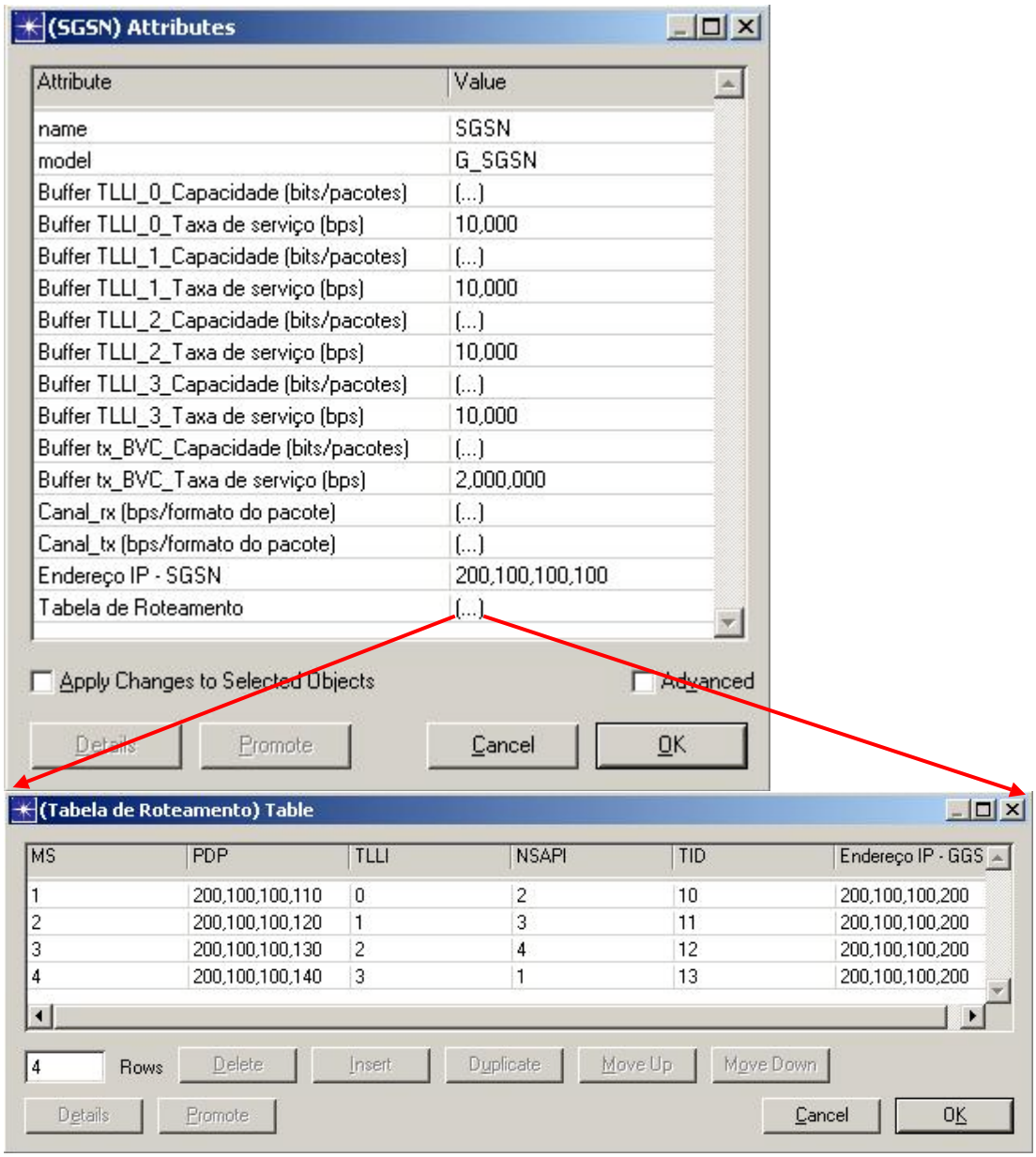


**Figura 5.11** – *Atributos de configuração – SGSN*

## 5.2.6 Nuvem *Frame Relay*


**Figura 5.12** – *Nuvem Frame Relay*

A nuvem *Frame Relay* (Figura 5.12) faz a interconexão entre vários nós BSCs e um nó SGSN. Os atributos de configuração são os mesmos apresentados no item 5.2.2.

## 5.2.7 BSC


**Figura 5.13** – *BSC*

O BSC (Figura 5.13) é o nó responsável pela reserva de recursos de rádio para transmissão. Os atributos de configuração do BSC variam em função da disciplina de escalonamento de pacotes implementada. O GPRSS possui cinco opções de disciplinas de escalonamento, FIFO, PQ, WFQ, WRR e DWRR (veja o item 5.5.1.6). Pode-se usar apenas uma dessas disciplinas dentro de um mesmo cenário de simulação.

Ao inserir um BSC no domínio de rede o mesmo virá com o modelo *default* G_BSC_FIFO. Para alterar a disciplina de escalonamento, basta mudar o valor do atributo "*model*" para a disciplina de escalonamento desejada e automaticamente todos os atributos referentes a esta nova disciplina são carregados.

Para o BSC com algoritmo do tipo FIFO deve-se configurar atributos (Figura 5.14) de *buffer* individual TLLI, *buffer* de célula BVC, *buffer* de escalonamento, esquema de codificação de canal CS-1, CS-2, CS-3 ou CS-4 (este modelo não inclui o codificador convolucional e consequentemente o processo de *puncturing*, apenas são utilizadas as estruturas de quadros dos esquemas de codificação) por MS e atributos dos canais de transmissão. Assim como o GGSN e SGSN, o BSC também possui uma tabela de roteamento que em uma rede real seria gerada automaticamente pelo plano de controle. Neste modelo, a configuração dos atributos desta tabela se faz necessária. Deve-se configurar a identificação da MS, TLLI, o identificador de fluxo temporário TFI que representa o rótulo do enlace entre BSC e MS, e o NSAPI que permite a diferenciação dos pacotes e conseqüentemente o escalonamento dos mesmos.

Para o BSC com algoritmo do tipo PQ deve-se configurar todos os atributos mencionados anteriormente para o algoritmo FIFO. Além disso deve-se também dimensionar a capacidade de três sub-filas de acordo com a classe de precedência padrão do GPRS, ou seja, sub-filas com prioridade baixa, média e alta respectivamente (atributo *Buffer* PQ_Capacidade(bits/pacotes)).

Para o BSC com algoritmo do tipo WFQ é necessário configurar todos os atributos mencionados anteriormente para o algoritmo PQ, inclusive as três sub-filas, e ainda os pesos de cada sub-fila expressos em porcentagem de alocação da banda de saída (%) conforme Figura 5.15.

Os atributos de configuração para o algoritmo do tipo WRR são os mesmos atributos mencionados anteriormente para o algoritmo WFQ.

Finalmente, para o algoritmo DWRR deve-se configurar todos os atributos (Figura 5.16) mencionados anteriormente, com a substituição dos pesos das sub-filas por quantidades de bits chamadas de "*quantum*". O termo *quantum* se refere à quantidade máxima de bits que podem ser retirados em cada *round* pelo escalonador.

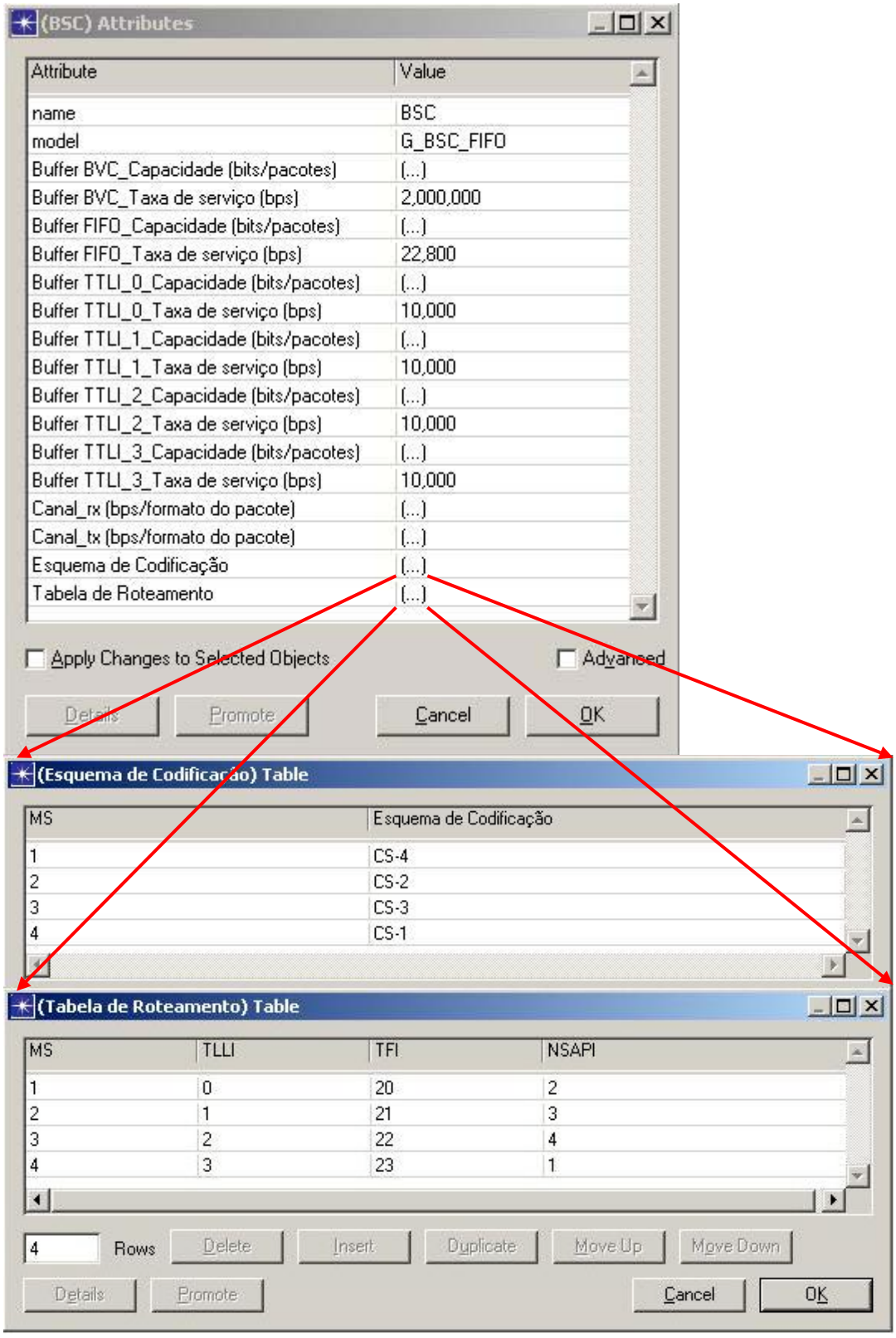


**Figura 5.14** – *Atributos de configuração – BSC com algoritmo FIFO*

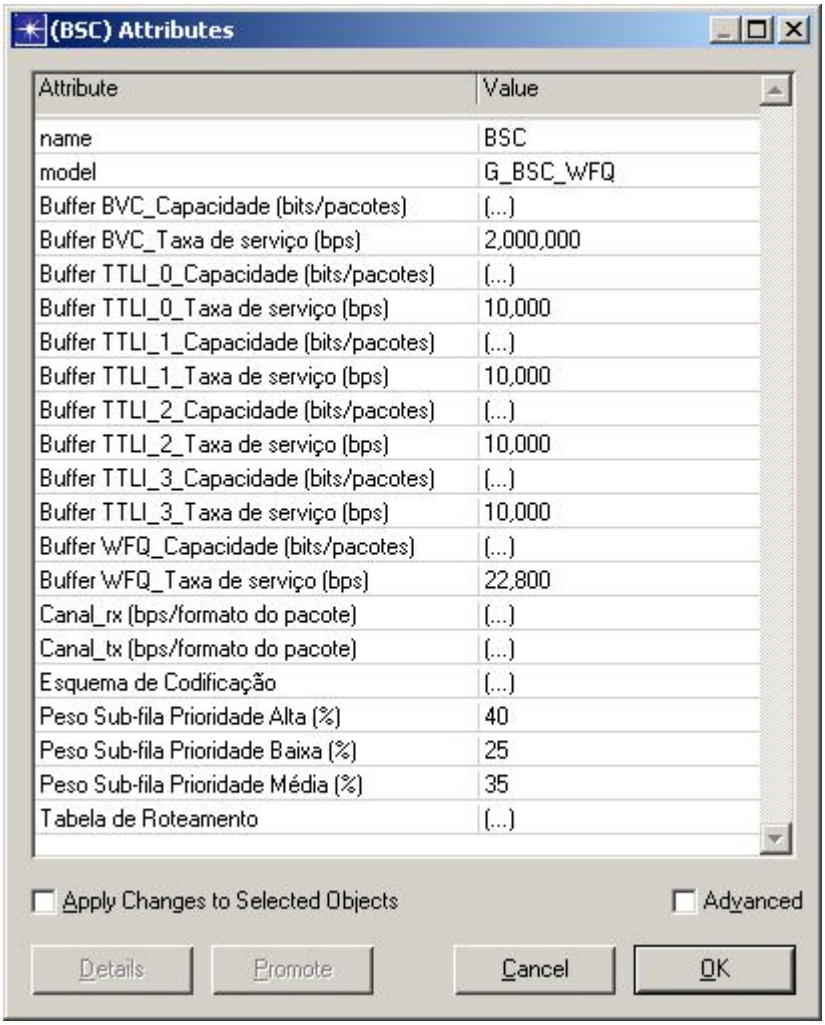


**Figura 5.15** – *Atributos de configuração – BSC com algoritmo WFQ / WRR*

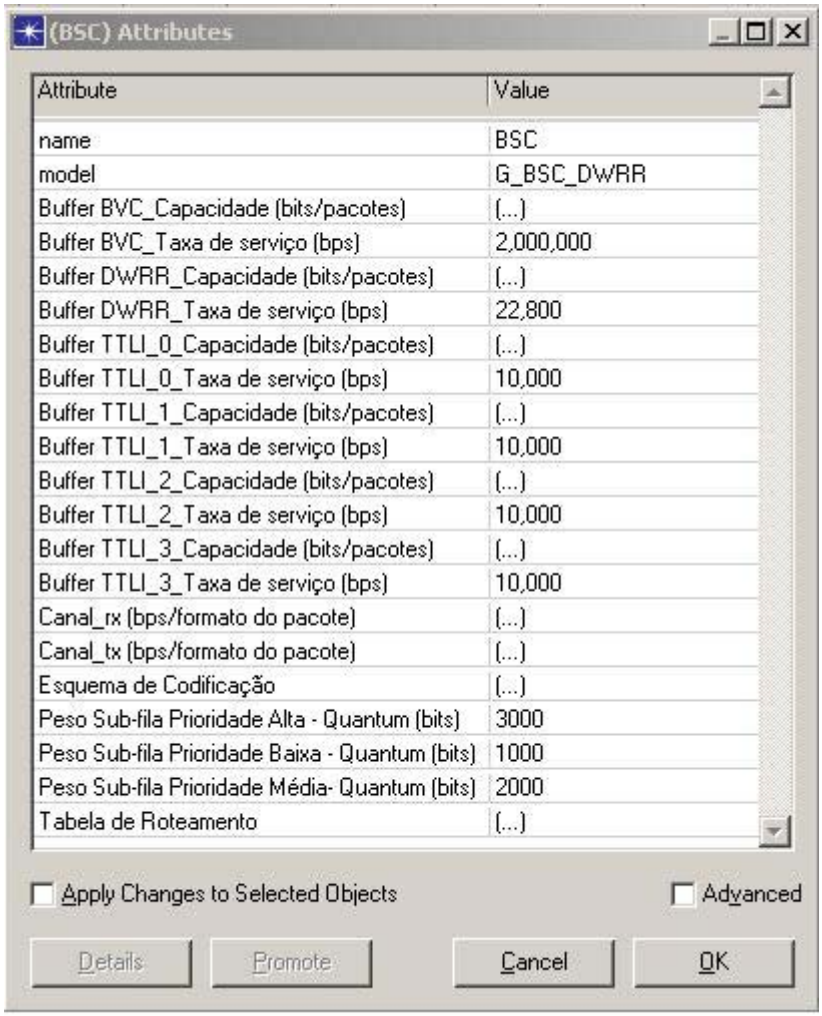


**Figura 5.16** – *Atributos de configuração – BSC com algoritmo DWRR*

## 5.2.8 BTS


**Figura 5.17**– *BTS*

A BTS (Figura 5.17) é responsável pela comunicação direta com a MS através de canais de RF. Além do atributo de modulação (GMSK, QPSK, FSK, etc.)

deve-se configurar os atributos dos canais de comunicação entre SGSN-BSC e entre BSC-MS (taxa de transmissão, formato dos pacotes, largura de banda do canal, freqüência mínima de transmissão, código de espalhamento e potência de transmissão) (Figura 5.18).

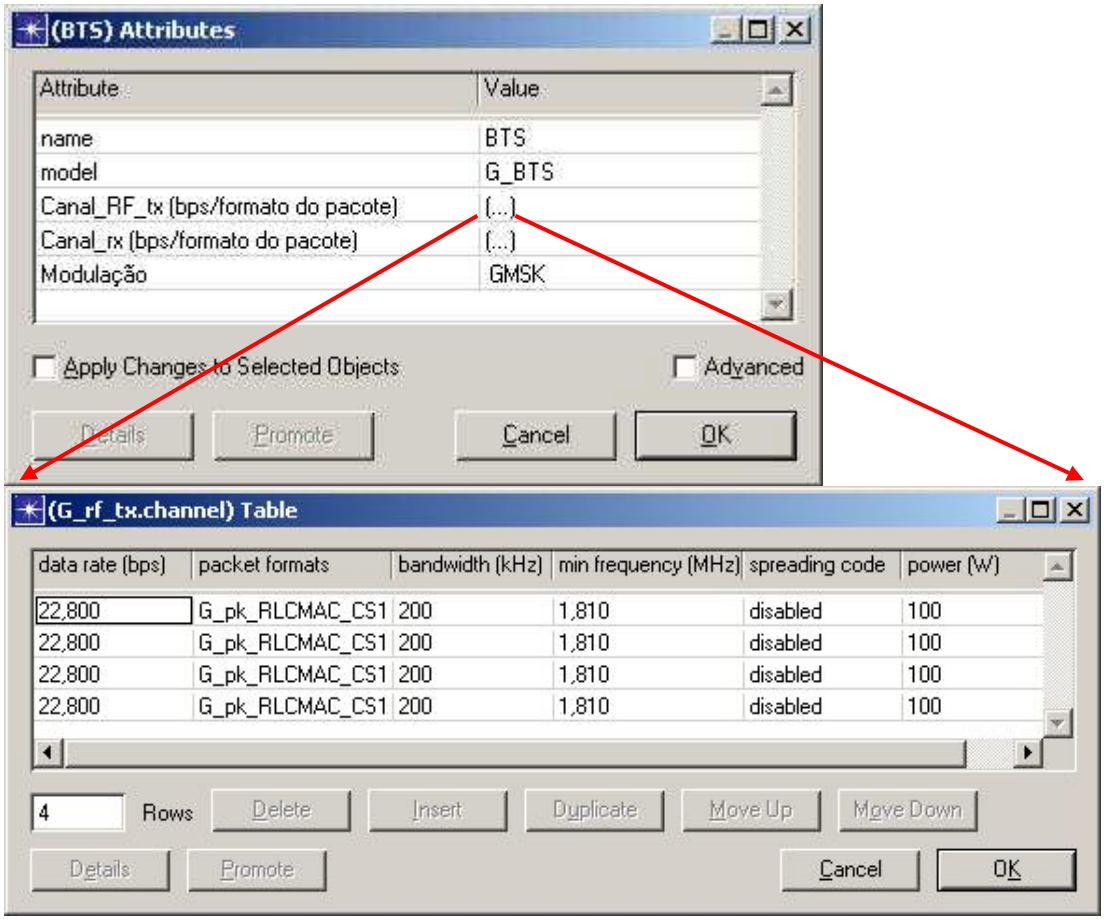


**Figura 5.18** – *Atributos de configuração – BTS*

## 5.2.9 MS


**Figura 5.19**– *MS*

A MS (Figura 5.19) é a interface entre o usuário e a aplicação/rede. Deve-se configurar o tipo de aplicação, atributos de *buffer*, atributos do canal de comunicação, e tipo de modulação. É possível configurar também a tabela da MS com a identificação da MS, TLLI, TFI, NSAPI e PDP (Figura 5.20).

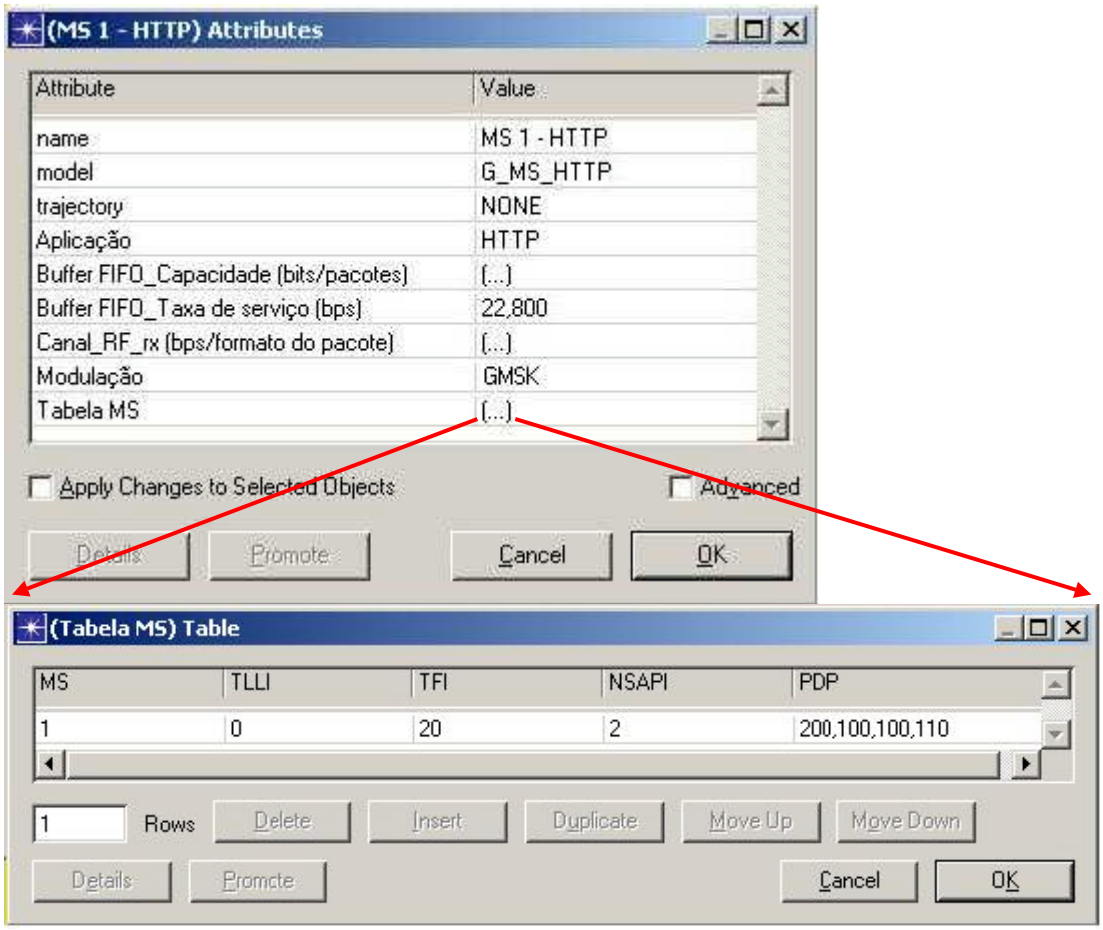


**Figura 5.20** – *Atributos de configuração – MS*

Outros atributos, tais como, tipo de antena, ganho de antena, altitude do nó, modelo de propagação e outros, devem ser configurados. É recomendável que tais atributos sejam configurados com modelos *"default"* da ferramenta OPNET Modeler.

## 5.3 Domínio de Nó

Através do domínio de nó é possível criar as entidades que são interconectadas no nível de rede. Modelos de nós são desenvolvidos através do *Node Editor* e são constituídos de pequenos blocos chamados módulos. No domínio de nó pode-se fazer uso de qualquer número de módulos e interligá-los através de *streams* (fluxos).

### 5.3.1 Servidor de Aplicação

O Servidor de Aplicação (G_servidor) representa a fonte de dados para o GPRSS (Figura 5.21) . Os pacotes (ex: IPv4) são gerados pelo módulo servidor (G_servidor) de acordo com os atributos apresentados no item 5.2.1. Os pacotes gerados são enviados ao módulo de encapsulamento *Frame Relay* (G_enc_FR). Uma vez encapsulados, os pacotes *Frame Relay* são enviados ao módulo transmissor (G_tx) que os transmite para o próximo nó.

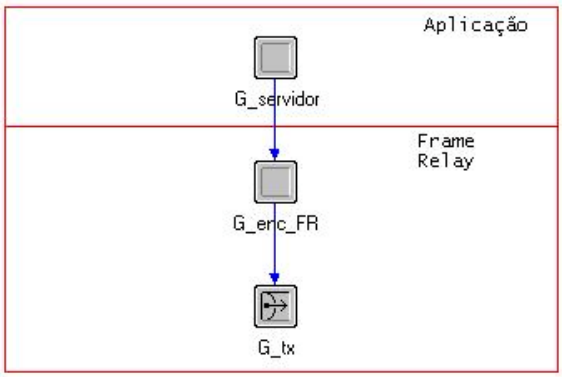


**Figura 5.21** – *Nó Servidor de Aplicação*

## 5.3.2 PDN

O nó PDN representa uma rede de dados externa e é constituído dos módulos receptores (G_rx0, G_rx1, G_rx2 e G_rx3), os quais recebem os pacotes *Frame Relay* (Figura 5.22). Estes pacotes são inseridos em um *Buffer* (G_buf_FIFO), cujos atributos de configuração permitem aproximar o comportamento da PDN deste modelo de uma rede de dados real. Após o processamento dos pacotes pelo *buffer* os mesmos são enviados ao módulo transmissor (G_tx) que os transmite para o próximo nó.

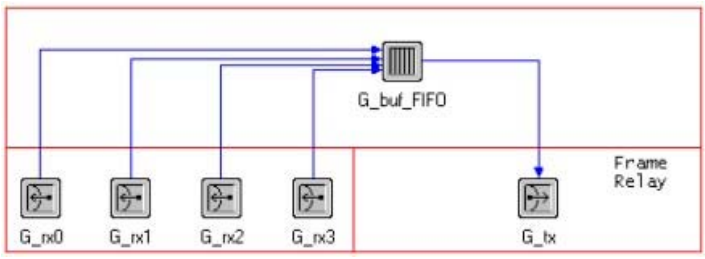


**Figura 5.22** – *Nó PDN*

## 5.3.3 GGSN

Os pacotes são recebidos pelo GGSN através do módulo receptor (G_rx) e enviados ao módulo de desencapsulamento *Frame Relay* (G_dec_FR), que envia os pacotes resultantes desta operação ao módulo de encapsulamento GTP (G_enc_GTP). Os pacotes GTP são classificados dentro do módulo de encapsulamento TCP/UDP (G_enc_TCP_UDP) e encapsulados em função do tipo de aplicação que transportam. Os pacotes TCP ou UDP são encapsulados no formato IPv4 pelo módulo de encapsulamento IP (G_enc_IP), e finalmente os pacotes IP são enviados ao módulo de encapsulamento *Frame Relay* (G_enc_FR) e posteriormente ao módulo transmissor (G_tx) (Figura 5.23).

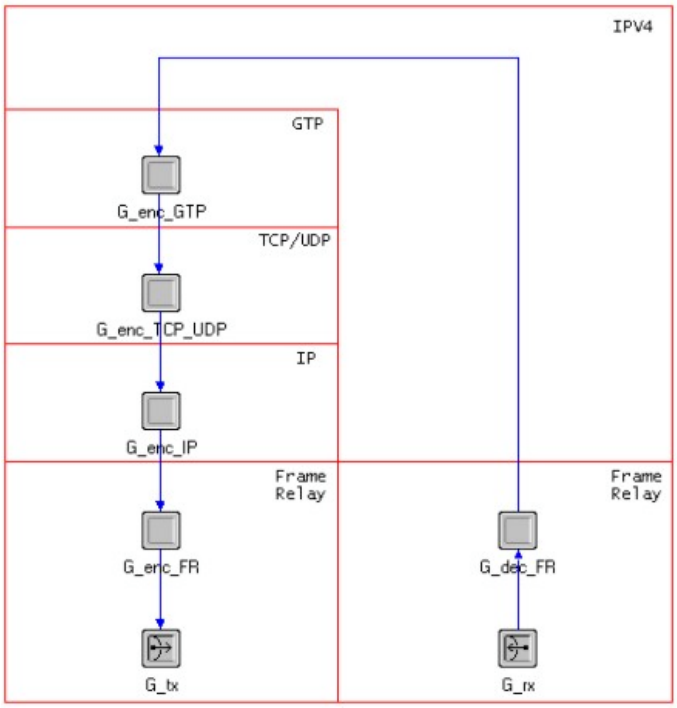


**Figura 5.23** – *Nó GGSN*

## 5.3.4 *Backbone* GPRS

O nó *Backbone* GPRS representa a rede de dados da operadora e, assim como o nó PDN, é constituído do módulo receptor (G_rx), *Buffer* com disciplina do tipo FIFO (G_buf_FIFO) e o módulo transmissor (G_tx) (Figura 5.24).

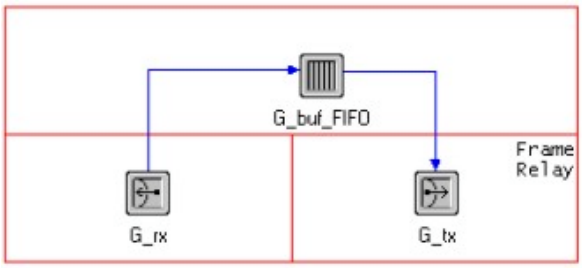


**Figura 5.24** – *Nó Backbone GPRS*

## 5.3.5 SGSN

O nó SGSN é constituído do módulo receptor (G_rx), módulo de desencapsulamento *Frame Relay* (G_dec_FR), módulo de desencapsulamento IP (G_dec_IP), módulo de desencapsulamento TCP/UDP (G_dec_TCP_UDP) e módulo de desencapsulamento GTP (G_dec_GTP). A partir deste ponto os pacotes são separados por estação móvel e inseridos no módulo de segmentação SNDCP (G_seg_SNDCP) que segmenta os pacotes recebidos em pacotes de 495 bytes de dados mais 5 bytes de cabeçalho (configuração de transmissão sem reconhecimento). Os pacotes SNCDP são enviados ao módulo de encapsulamento LLC (G_enc_LLC). Após encapsulamento LLC os pacotes são inseridos em *Buffers* (G_buf_FIFO) responsáveis pelo controle de fluxo e posteriormente multiplexados no módulo *Buffer* BVC (G_buf_tx_BVC). O fluxo de pacotes enviados pelo multiplexador é encapsulado novamente através do módulo (G_enc_FR) e enviado ao módulo transmissor (G_tx) (Figura 5.25).

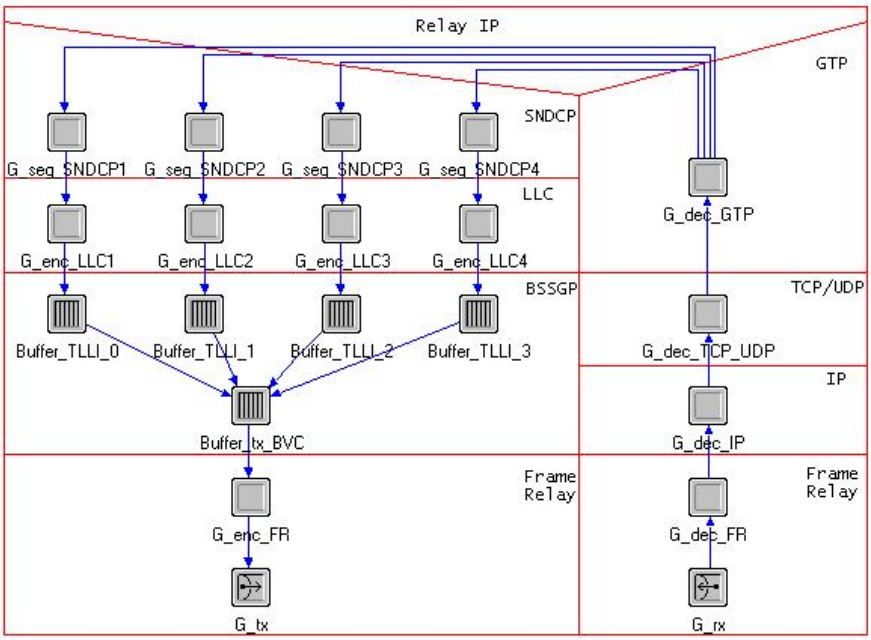


**Figura 5.25** – *Nó SGSN*

## 5.3.6 Nuvem *Frame Relay*

A nuvem *Frame Relay* representa uma rede de interconexão entre BSCs e SGSN. Sua composição é exatamente igual à demonstrada no item 5.3.4.

## 5.3.7 BSC

O módulo BSC é constituído do módulo receptor (G_rx), do módulo de desencapsulamento *Frame Relay* (G_dec_FR), do módulo demultiplexador (G_buf_rx_BVC) que separa novamente os fluxo de pacotes LLC para cada estação móvel. Os pacotes LLC são inseridos em *buffers* individuais (G_buf_FIFO) e posteriormente segmentados em pequenos pacotes de 456 bits pelo módulo de segmentação da camada RLC / MAC (G_seg_RLCMAC) e encapsulados em pacotes de acordo com o esquema de codificação (CS-1, CS-2, CS-3 ou CS-4). Estes pacotes seguem em direção ao módulo escalonador de pacotes (G_esc_FIFO, G_esc_PQ, G_esc_WFQ, G_esc_WRR ou G_esc_DWRR) onde são classificados e escalonados de acordo com a classe de precedência. Após o escalonamento os pacotes são transmitidos pelo módulo transmissor (G_tx) (Figura 5.26).

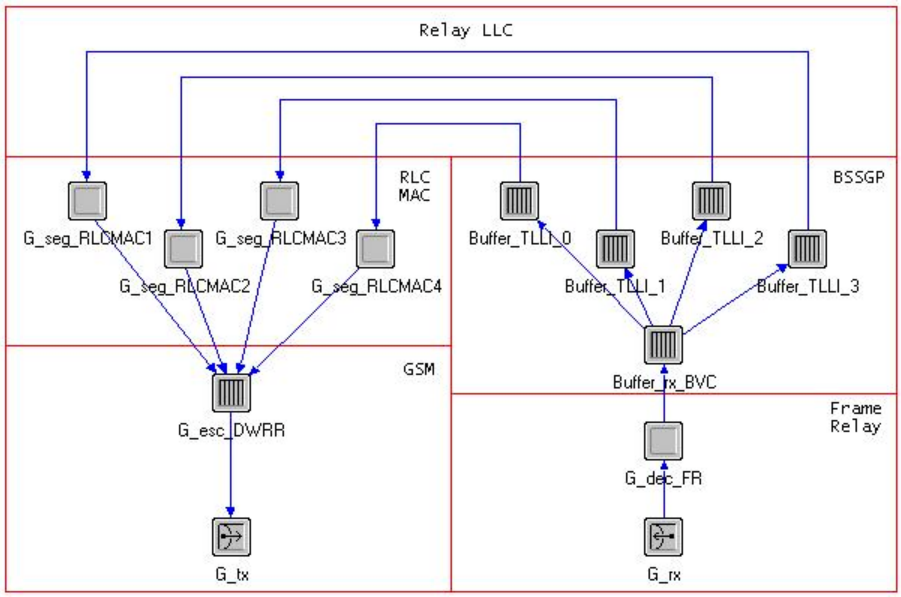


**Figura 5.26** – *Nó BSC*

## 5.3.8 BTS

O nó BTS é constituído pelo módulo receptor (G_rx), módulo transmissor de rádio (G_rf_tx) que faz a modulação e transmissão dos pacotes RLC/MAC recebidos em sinais de rádio via módulo da antena (G_ant_tx) (Figura 5.27).

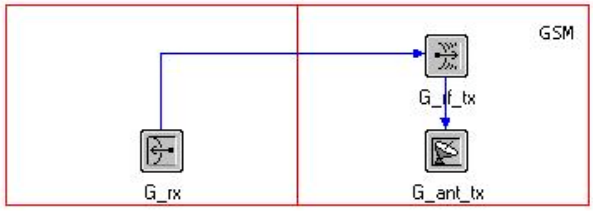


**Figura 5.27** – *Nó BTS*

## 5.3.9 MS

O nó MS é constituído do módulo de antena (G_ant_rx), módulo receptor de rádio (G_rf_rx), *buffer* de controle de fluxo (G_buf_ms_FIFO), módulo de montagem de pacotes LLC (G_mon_LLC) que executa o processo inverso à segmentação de pacotes, módulo de desencapsulamento SNDCP (G_dec_SNDCP), módulo de montagem do pacote IPv4 (G_mon_IPV4) e módulo consumidor de pacotes IPv4 (G_sink_IPV4) ou ponto final da recepção (Figura 5.28).

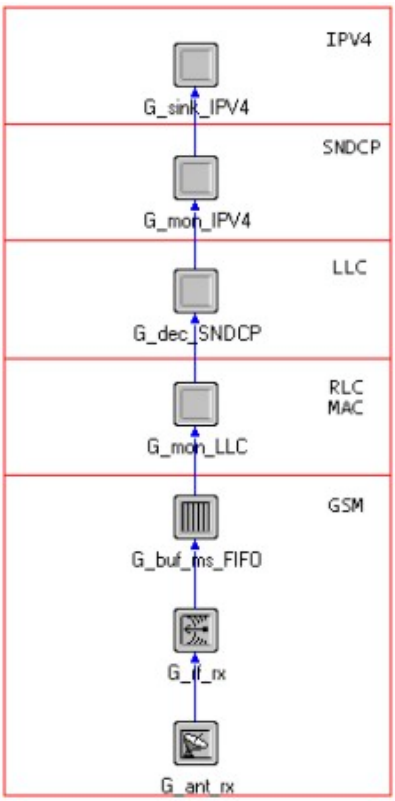


**Figura 5.28** – *Nó MS*

## 5.4 Domínio de Processo

Os módulos usados para compor o nó são elementos programáveis pelo usuário e fundamentais para a execução de suas tarefas que são chamadas de processos. Os processos são editados no *Process Editor*.

O *Process Editor* expressa modelos de processos em uma linguagem chamada Proto-C, a qual é especificamente desenvolvida para suportar o desenvolvimento de protocolos e algoritmos. O Proto-C é baseado em uma combinação de diagrama de transição de estados, uma biblioteca de comandos de alto nível conhecidos como KPs (*Kernel Procedures)*, e facilidades gerais da linguagem de programação C ou C++.

### 5.4.1 G_servidor

O processo G_servidor tem como função gerar pacotes de acordo com os atributos definidos pelo usuário.

O processo é constituído de três estados (INICIO, GERAÇÃO e TERMINO) (Figura 5.29). O estado INICIO é o ponto de partida do processo, este obtém os atributos definidos pelo usuário, verifica a validade dos mesmos e registra as estatísticas a serem coletadas dentro do processo para análise. No estado GERAÇÃO, uma sub-rotina de geração de pacotes é chamada de acordo com a FDP de intervalo entre pacotes. A sub-rotina de geração de pacotes (ss_gera_pacote) cria um pacote cujo formato está disponível em uma biblioteca criada especialmente para este modelo. A formatação de todos os pacotes utilizados neste modelo (anexo B) também fez parte do escopo deste trabalho. Os pacotes foram formatados usando o *Packet Editor*. O pacote também tem o seu tamanho em função de uma FDP definida pelo usuário. Se o tempo de término vencer, o processo passa para o estado TERMINO e encerra a geração de pacotes. O código deste processo está descrito no anexo A1.

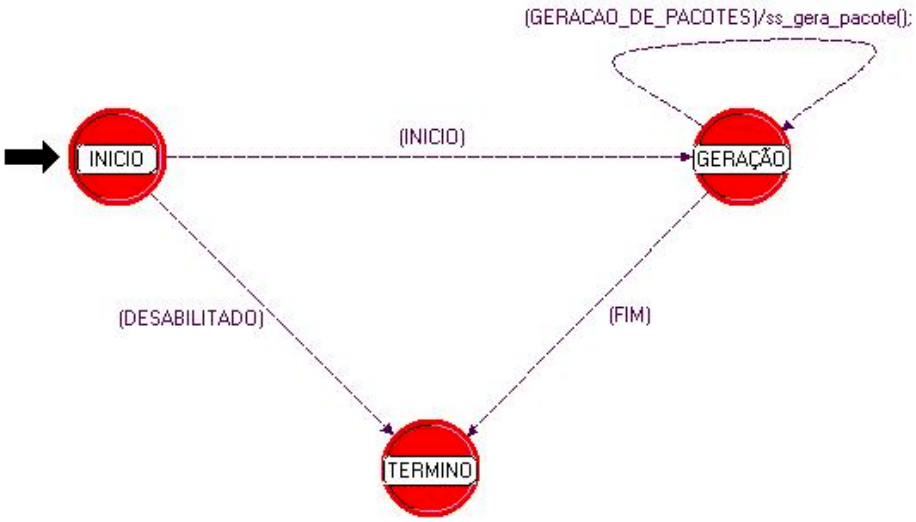


**Figura 5.29** – *G_servidor*

## 5.4.2 G_enc_FR

O processo G_enc_FR tem como função encapsular os pacotes recebidos da camada superior dentro de uma estrutura de quadro *Frame Relay*.

O processo é constituído de três estados (INICIO, LIVRE e ENC_FR) (Figura 5.30). O estado INICIO além de ser o ponto de partida, também registra as estatísticas a serem coletadas dentro do processo. As transições entre os estados são determinadas pela chegada ou saída de pacotes. O processo permanece no estado LIVRE quando não há pacotes para processar. O estado ENC_FR obtém o pacote da camada superior, gera um pacote no formato *Frame Relay*, faz a inserção do pacote recebido dentro do campo "Dados" do pacote *Frame Relay* e preenche o cabeçalho com endereço de destino. OBS: O processo G_enc_FR utilizado no nó GGSN preenche o endereço do cabeçalho DLCI com o TID e no nó SGSN com o TLLI. O código deste processo está descrito no anexo A2.

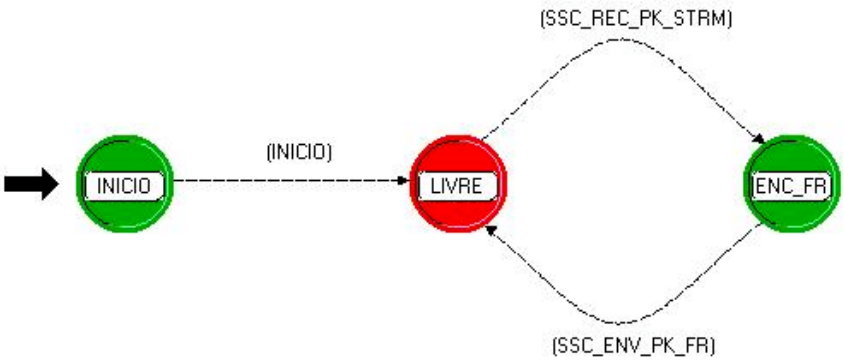


**Figura 5.30** – *G_enc_FR*

## 5.4.3 G_buf_FIFO

O processo G_buf_FIFO tem como função inserir os pacotes recebidos em uma fila que trabalha com a disciplina FIFO. O processamento dos pacotes segue uma taxa de serviço atribuída pelo usuário.

O processo é constituído de cinco estados (INICIO, CHEGADA, FILA, LIVRE e SAÍDA) (Figura 5.31). O estado INICIO é o ponto de partida do processo e onde se obtém os atributos definidos pelo usuário. No estado CHEGADA o pacote é recebido e inserido na cauda (*Tail*) da fila. O estado FILA é responsável por simular o tempo de processamento do pacote dentro da fila. Depois de transcorrido este tempo, o processo muda para o estado SAÍDA onde o pacote é retirado da cabeça da fila e enviado para a próxima camada. O processo assume o estado LIVRE em momentos de ociosidade. As transições são determinadas pela chegada e saída de pacotes, inserção de pacote na fila e término do processamento do pacote. O código deste processo está descrito no anexo A3.

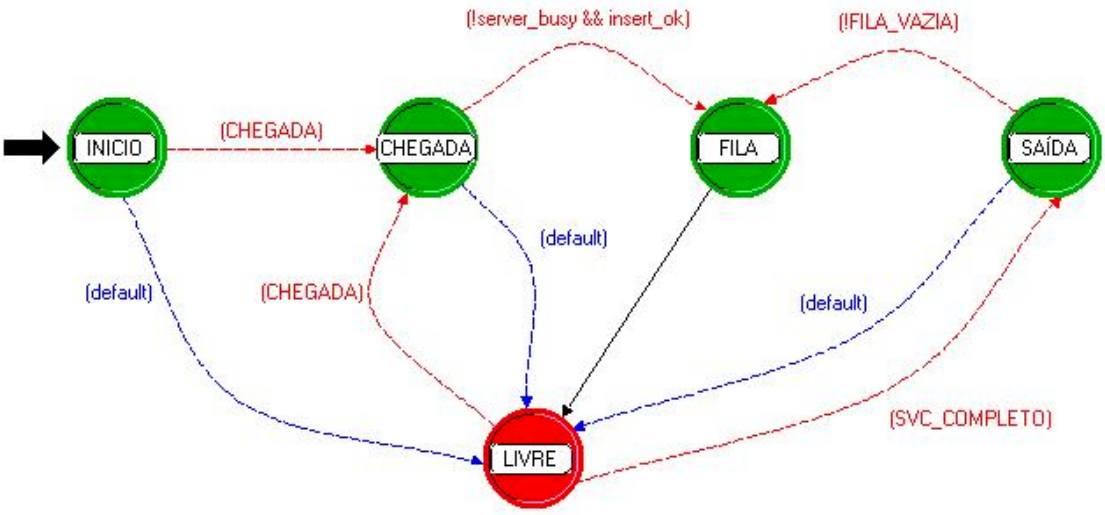


**Figura 5.31** – *G_buf_FIFO*

## 5.4.4 G_dec_FR

O processo G_dec_FR tem como função desencapsular o pacote original da estrutura de quadro *Frame Relay* e encaminhá-lo a próxima camada.

O processo é constituído de três estados (INICIO, LIVRE e DEC_FR), assim como item 5.4.2. Difere apenas no estado DEC_FR que obtém o pacote da camada inferior, verifica se o endereço do pacote corresponde ao nó, em caso negativo o pacote é destruído. Se o endereço do pacote corresponder ao nó, então o pacote

original é retirado do campo “Dados” do pacote *Frame Relay* e é transmitido para a próxima camada. O código deste processo está descrito no anexo A4.

### 5.4.5 G_enc_GTP

O processo G_enc_GTP tem como função encapsular os pacotes IPv4 recebidos dentro de uma estrutura de quadro GTP.

O processo é constituído de três estados (INICIO, LIVRE e ENC_GTP), como no item 5.4.2. Difere apenas no estado ENC_GTP que obtém o pacote da camada superior, gera um pacote no formato GTP, faz o encapsulamento do pacote recebido dentro do campo “Dados” e preenche o cabeçalho com o TID obtido na tabela de roteamento e o “TOM” (*Type of Message*) com o dado obtido no campo “ToS” (*Type of Service*) se o pacote for IPv4. O código deste processo está descrito no anexo A5.

### 5.4.6 G_enc_TCP_UDP

O processo G_enc_TCP_UDP tem como função encapsular os pacotes GTP recebidos dentro de uma estrutura de quadro TCP ou UDP, o que depende do tipo de informação. Neste modelo o pacote de aplicação HTTP, FTP e SMTP são encapsulados dentro de uma estrutura de quadro TCP e a aplicação VoIP é encapsulada dentro de uma estrutura de quadro UDP.

O processo é constituído de três estados (INICIO, LIVRE e ENC_TCP_UDP) assim como no item 5.4.2. Difere apenas no estado ENC_TCP_UDP que obtém o pacote da camada superior, verifica qual o tipo de aplicação que é transportada no pacote a partir do campo “TOM”, gera um pacote no formato TCP ou UDP dependendo da aplicação, faz o encapsulamento do pacote recebido dentro do campo “Dados” e preenche o cabeçalho com o TID do pacote. O código deste processo está descrito no anexo A6.

### 5.4.7 G_enc_IP

O processo G_enc_GTP tem como função encapsular os pacotes TCP ou UDP recebidos dentro de uma estrutura de quadro IPv4.

O processo é constituído de três estados (INICIO, LIVRE, ENC_IP) assim como no item 5.4.2. Difere apenas no estado ENC_IP que obtém o pacote da camada superior, gera um pacote no formato IPv4, faz o encapsulamento do pacote recebido dentro do campo “Dados” e preenche o cabeçalho com o endereço IP do SGSN destino. O código deste processo está descrito no anexo A7.

### 5.4.8 G_dec_IP

O processo G_dec_IP tem como função desencapsular o pacote original da estrutura de quadro IP e encaminhá-lo à próxima camada.

O processo é constituído de três estados (INICIO, LIVRE e DEC_IP) assim como no item 5.4.2. Difere apenas no estado DEC_IP que obtém o pacote da camada inferior e verifica se o mesmo está destinado a este nó. Se estiver, o pacote TCP ou UDP original é retirado do campo “Dados” do pacote IPv4 e transmitido para a próxima camada. O código deste processo está descrito no anexo A8.

### 5.4.9 G_dec_TCP_UDP

O processo G_dec_TCP_UDP tem como de função desencapsular o pacote original da estrutura de quadro TCP ou UDP e encaminhá-lo à próxima camada.

O processo é constituído de três estados (INICIO, LIVRE e DEC_TCP_UDP) assim como no item 5.4.2. Difere apenas no estado DEC_TCP_UDP que obtém o pacote da camada inferior, retira o pacote original do campo “Dados” do pacote TCP

ou UDP e o transmite para a próxima camada. O código deste processo está descrito no anexo A9.

### 5.4.10 G_dec_GTP

O processo G_dec_GTP tem como função desencapsular o pacote original da estrutura de quadro GTP e encaminhá-lo à próxima camada, contudo neste processo os pacotes são separados em fluxos por estação móvel.

O processo é constituído de três estados (INICIO, LIVRE e DEC_GTP) assim como no item 5.4.2. Difere apenas no estado DEC_GTP que obtém o pacote da camada inferior, retira o pacote original do campo “Dados” do pacote GTP e o transmite para a próxima camada de acordo com o fluxo TLLI da estação móvel obtida na tabela de roteamento. O código deste processo está descrito no anexo A10.

### 5.4.11 G_seg_SNDCP

O processo G_seg_SNDCP tem como função segmentar o pacote G_pk_IPV4 em pacotes menores de tamanho máximo de 500 bytes (495 bytes de dados e 5 bytes de cabeçalho). O resultado desta operação é um pacote G_pk_SNDCP que é enviado à próxima camada.

O processo é constituído de três estados (INICIO, LIVRE e SEG_SNDCP) assim como no item 5.4.2. Difere apenas no estado SEG_SNDCP que obtém o pacote G_pk_IPV4 e o insere em um *buffer* de segmentação. Os segmentos são retirados do *buffer* e inseridos no campo “Dados” do pacote G_pk_SNDCP, o cabeçalho é preenchido com o NSAPI obtido na tabela de roteamento. Em seguida o pacote é enviado para a próxima camada. O código deste processo está descrito no anexo A11.

### 5.4.12 G_enc_LLC

O processo G_enc_LLC tem como função encapsular os pacotes G_pk_SNDCP recebidos dentro de uma estrutura de quadro LLC.

O processo é constituído de três estados (INICIO, LIVRE e ENC_LLC) assim como no item 5.4.2 . Difere apenas no estado ENC_LLC que obtém o pacote da camada superior, gera um pacote G_pk_LLC, faz o encapsulamento do pacote recebido dentro do campo “Dados” do pacote LLC e preenche o cabeçalho com o SAPI obtido do pacote recebido (NSAPI). O código deste processo está descrito no anexo A12.

### 5.4.13 G_buf_tx_BVC

O processo G_buf_tx_BVC tem como função inserir os pacotes recebidos de fluxos de diferentes MS em uma fila única (FIFO) por célula.

O processo é constituído de cinco estados (INICIO, CHEGADA, FILA, LIVRE e SAÍDA) assim como no item 5.4.3. O código deste processo está descrito no anexo A13.

### 5.4.14 G_buf_rx_BVC

O processo G_buf_rx_BVC tem como função receber o fluxo de pacotes de uma célula e separar novamente este em vários fluxos por MS, ou seja um para cada estação móvel.

O processo é constituído de cinco estados (INICIO, CHEGADA, FILA, LIVRE e SAÍDA) assim como no item 5.4.3. Difere apenas no estado SAÍDA onde o pacote é retirado da cabeça da fila e enviado para a próxima camada já em fluxos

diferentes, um fluxo TLLI para cada MS. O código deste processo está descrito no anexo A14.

### 5.4.15 G_seg_RLC_MAC

O processo G_seg_RLC_MAC tem como função segmentar o pacote G_pk_LLC em pacotes menores de tamanho máximo de 456 bits. O resultado desta operação é um pacote G_pk_RLC_MAC_CS1 (*default*) que é enviado à próxima camada.

O processo é constituído de três estados (INICIO, LIVRE e SEG_RLC_MAC) assim como no item 5.4.2. Difere apenas no estado SEG_RLC_MAC que obtém o pacote G_pk_LLC e o insere em um *buffer* de segmentação. Os segmentos são retirados do *buffer* e inseridos no campo “Dados” de um pacote que será criado em função do esquema de codificação utilizado. Quatro tipos de pacotes podem ser usados: G_pk_RLC_MAC_CS1, G_pk_RLC_MAC_CS2, G_pk_RLC_MAC_CS3 ou G_pk_RLC_MAC_CS4. O cabeçalho é preenchido com o TFI obtido na tabela de roteamento e em seguida o pacote é enviado para a próxima camada. O código deste processo está descrito no anexo A15.

### 5.4.16 G_buf_ms_FIFO

O processo G_buf_ms_FIFO tem como função simular o policiamento do tráfego usando uma fila do tipo FIFO e descartar pacotes não destinados a esta estação móvel. O processamento dos pacotes segue uma taxa de processamento definida pelo usuário.

O processo é constituído de cinco estados (INICIO, CHEGADA, FILA, LIVRE e SAÍDA) assim com no item 5.4.3. Difere apenas no estado CHEGADA onde é feito o descarte de pacotes não destinados a esta estação móvel (baseado no

TFI), caso contrário o pacote é inserido na cauda (*Tail*) da fila. O código deste processo está descrito no anexo A16.

### 5.4.17 G_mon_LLC

O processo G_mon_LLC tem como função reagrupar os segmentos encapsulados dentro dos pacotes RLC_MAC formando o pacote LLC original.

O processo é constituído de três estados (INICIO, LIVRE e MON_LLC) assim como no item 5.4.2. Difere apenas no estado MON_LLC que obtém o pacote da camada inferior, retira o segmento encapsulado de dentro do pacote RLC_MAC e o insere dentro de um *buffer* de remontagem. Quando o número de segmentos dentro do *buffer* atingir o tamanho original do pacote LLC, este é retirado e enviado a próxima camada. O código deste processo está descrito no anexo A17.

### 5.4.18 G_dec_SNDCP

O processo G_dec_SNDCP tem como função desencapsular o pacote original SNDCP da estrutura de quadro LLC e encaminhá-lo à próxima camada.

O processo é constituído de três estados (INICIO, LIVRE e DEC_SNDCP) assim como no item 5.4.2. Difere apenas no estado DEC_SNDCP que obtém o pacote da camada inferior, retira o pacote original do campo “Dados” do pacote LLC e o transmite para a próxima camada de acordo com o fluxo da estação móvel. O código deste processo está descrito no anexo A18.

## 5.4.19 G_mon_IPV4

O processo G_mon_IPV4 tem como função reagrupar os segmentos encapsulados dentro dos pacotes SNDCP formando o pacote IPv4 original.

O processo é constituído de três estados (INICIO, LIVRE e MON_IPV4) assim como no item 5.4.2. Difere apenas no estado MON_IPV4 que obtém o pacote da camada inferior, retira o segmento encapsulado de dentro do pacote SNDCP e o insere dentro de um *buffer* de remontagem. Quando o número de segmentos dentro do *buffer* atingir o tamanho original do pacote IPv4, este é retirado e enviado à próxima camada. O código deste processo está descrito no anexo A19.

## 5.4.20 G_sink_IPV4

O processo G_sink_IPV4 tem como função coletar estatísticas e “consumir” (destruir) os pacotes recebidos.

O processo é constituído de dois estados (INICIO e DESCARTE) (Figura 5.32). O estado INICIO, além de ser o ponto de partida do processo, registra as estatísticas a serem coletadas. As transições entre os estados são determinadas pela chegada de pacotes. O estado DESCARTE obtém o pacote da camada inferior, gera as estatísticas, e em seguida destrói o pacote. O código deste processo está descrito no anexo A20.

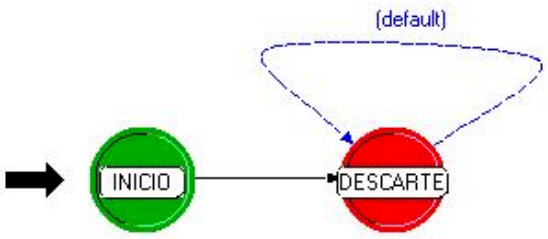


**Figura 5.32** – *G_sink_IPV4*

## 5.4.21 G_esc_FIFO

O processo G_esc_FIFO tem como função inserir os pacotes recebidos em uma fila que trabalha com a disciplina FIFO. O processamento dos pacotes segue uma taxa de serviço atribuída pelo usuário.

O processo é constituído de cinco estados (INICIO, CHEGADA, FILA, LIVRE e SAÍDA) assim como no item 5.4.3.

## 5.4.22 G_esc_PQ

O processo G_esc_PQ tem como função classificar os pacotes recebidos, inseri-los em três sub-filas de acordo com sua classe de precedência e escaloná-los. O processamento dos pacotes segue uma taxa de serviço atribuída pelo usuário.

O processo é constituído de cinco estados (INICIO, CLASSIFICADOR, SUB-FILAS, LIVRE e ESCALONADOR) (Figura 5.33). O estado INICIO é o ponto de partida do processo e onde se obtém os atributos definidos pelo usuário. No estado CLASSIFICADOR o pacote é recebido e inserido na cauda (*Tail*) de uma das três sub-filas (0, 1 ou 2) em função do NSAPI obtido na tabela de roteamento. O estado SUB-FILAS é responsável por simular o tempo de processamento do pacote dentro da sub-fila. Transcorrido este tempo, o processo muda para o estado ESCALONADOR onde o pacote é retirado da cabeça da fila e enviado para a próxima camada sempre obedecendo a uma ordem de prioridade. O escalonador só retirará e enviará um pacote contido na sub-fila 1 se a sub-fila 0 estiver vazia e só retirará e enviará um pacote da sub-fila 2 se as sub-filas 1 e 0 estiverem vazias. O processo assume o estado LIVRE em momentos de ociosidade. As transições são determinadas pela chegada e saída de pacotes, inserção de pacote na sub-fila e fim do processamento do pacote. O código do processo está descrito no anexo A21.

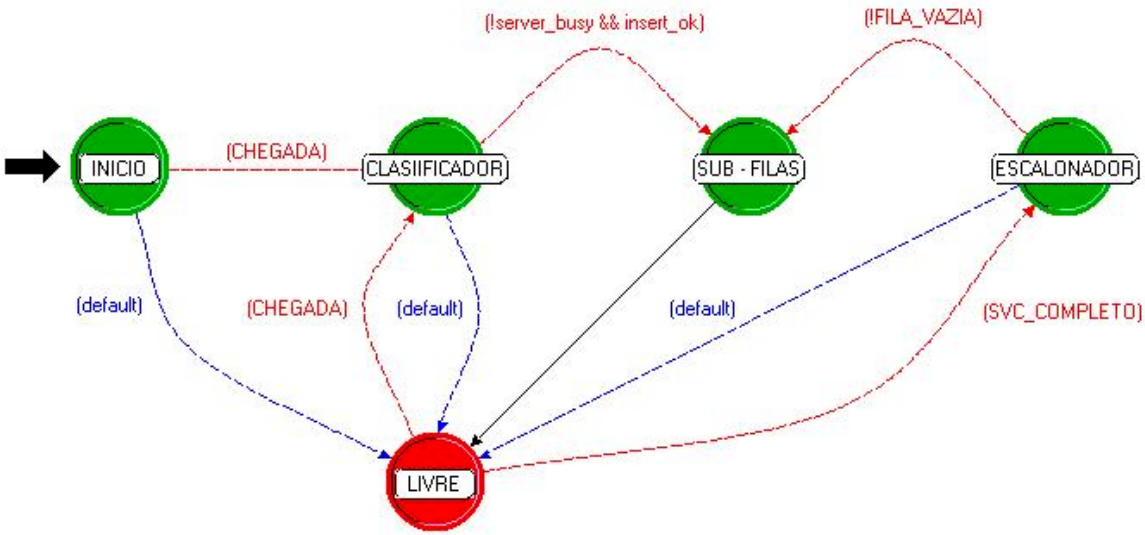


**Figura 5.33** – *G_esc_PQ*

## 5.4.23 G_esc_WFQ

O processo G_esc_WFQ tem como função classificar os pacotes recebidos, calcular o "*Finish Time*" (descrito no item 4.5.1.6.3), inseri-los em três sub-filas de acordo com sua classe de precedência e escaloná-los em função do menor "*Finish Time*". O processamento dos pacotes segue uma taxa de serviço atribuída pelo usuário.

O processo é constituído de cinco estados (INICIO, CLASSIFICADOR, SUB-FILAS, LIVRE e ESCALONADOR) assim como no item 5.4.2.2. Difere apenas nos estados CLASSIFICADOR e ESCALONADOR. O estado CLASSIFICADOR é responsável por verificar o número de conexões ativas, calcular o número do *round*, calcular o número do *Finish Time*, rotular o pacote com este último número e inseri-lo na sub-fila correspondente à sua classe de precedência. O estado ESCALONADOR verifica a cabeça das três sub-filas e retira o pacote rotulado com o menor número de *Finish Time* para ser enviado à próxima camada. O código deste processo está descrito no anexo A22.

## 5.4.24 G_esc_WRR

O processo G_esc_WRR tem como função classificar os pacotes recebidos, inseri-los em três sub-filas de acordo com sua classe de precedência e escaloná-los em função do peso de cada sub-fila. O processamento dos pacotes segue uma taxa de serviço atribuída pelo usuário.

O processo é constituído de cinco estados (INICIO, CLASSIFICADOR, SUB-FILAS, LIVRE e ESCALONADOR) assim como no item 5.4.2.2. Difere apenas nos estados CLASSIFICADOR e ESCALONADOR. O estado CLASSIFICADOR é responsável por classificar os pacotes recebidos e inseri-los na sub-fila correspondente a sua classe de precedência. O estado ESCALONADOR retira os pacotes da sub-filas em função do peso (atributo definido pelo usuário) de cada uma. Por exemplo, se tivermos a sub-fila 0 com peso 50 e as sub-filas 1 e 2 com pesos 25 cada uma, durante um ciclo de operação serão transmitidos dois pacotes da sub-fila 0, um pacote da sub-fila 1 e também um pacote da sub-fila 2. O código deste processo está descrito no anexo A23.

## 5.4.25 G_esc_DWRR

O processo G_esc_DWRR tem como função classificar os pacotes recebidos e inseri-los em três sub-filas de acordo com sua classe de precedência e escaloná-los em função de um saldo de bits de cada sub-fila. Inicialmente este saldo é atribuído a cada fila através das variáveis Quantum_0, Quantum_1 e Quantum_2. O processamento dos pacotes segue uma taxa de serviço atribuída pelo usuário.

O processo é constituído de cinco estados (INICIO, CLASSIFICADOR, SUB-FILAS, LIVRE e ESCALONADOR) assim como no item 5.4.2.2. Difere apenas nos estados CLASSIFICADOR e ESCALONADOR. O estado CLASSIFICADOR é responsável por classificar os pacotes recebidos e inseri-los na sub-fila correspondente às suas classes de precedência. O estado ESCALONADOR retira os pacotes da sub-filas em função do saldo (variável *deficit counter*) atual de

cada uma. Por exemplo, considerando que há vários pacotes de 456 bits nas três sub-filas e tem-se a sub-fila 0 com a variável *deficit_counter_0* igual a 1000, a sub-fila 1 com a variável *deficit_counter_1* igual a 500 e a sub-fila 2 com a variável *deficit_counter_2* igual a 300. Após um ciclo de operação serão transmitidos dois pacotes da sub-fila 0, um pacote da sub-fila 1 e nenhum pacote da sub-fila 2. O código deste processo está descrito no anexo A24.

## 5.5 Conclusão

O objetivo deste capítulo foi apresentar os aspectos mais importantes do modelo de simulação GPRSS proposto.

O GPRSS foi desenvolvido utilizando a ferramenta de modelagem OPNET Modeler, a qual dispõe três domínios/ambientes de desenvolvimento (rede, nó e processo) onde uma rede é constituída de nós e cada nó é constituído de processos.

O ambiente de rede descreve a topologia em termos de subredes, nós, enlaces, contexto geográfico e permite a configuração dos atributos dos nós. O ambiente de nó descreve a arquitetura interna do nó em termos de módulos funcionais e fluxo de dados entre eles. Finalmente, o processo descreve o comportamento do processo especificado (protocolos, algoritmos, aplicações) usando máquina de estado finita e linguagem de programação de alto nível.

O ambiente de rede do GPRSS possui uma rede de dados externa, a rede GPRS da operadora e estações móveis dispostas dentro da área de uma célula. Assim, é possível representar um ambiente real e fazer uma análise de seu desempenho a partir de parâmetros coletados durante a simulação.

O GPRSS é de funcionalidade razoavelmente simples, uma vez que este se refere apenas ao plano de transmissão da rede GPRS. Contudo a sua implementação é bastante complexa (envolvendo um estudo aprofundado das especificações do ETSI). Uma característica importante do GPRSS, não abordada neste capítulo, é a

riqueza de recursos disponíveis para análise, os quais serão demonstrados no próximo capítulo em diversos cenários e com métricas de desempenho diversas.

# Capítulo VI

## 6. Resultados e Análise da Simulação

O capítulo atual apresenta resultados e análises de simulações feitas utilizando o GPRSS. O objetivo principal dessas simulações é avaliar se o modelo proposto representa com uma certa fidelidade uma rede GPRS real. Uma vez obtidos e analisados os resultados da simulação, é possível avaliar também quais disciplinas de escalonamento são mais adequadas para diversos perfis de tráfego.

O GPRSS permite a coleta e avaliação de diversos parâmetros, tais como, atrasos fim-a-fim (s), atraso em *buffer* (s), espaço ocupado de *buffer* (pacotes, bits), espaço livre em *buffer* (pacotes, bits), *overflow* em *buffer*, taxa de perda de pacotes (pacotes), utilização de *links* (%), *throughtput* (bps), *throughtput* (pacotes/s), tamanho do pacote (bits), tráfego gerado (pacotes, pacotes/s, bits, bits/s), pacotes recebidos (pacotes, pacotes/s, bits, bits/s). Tais parâmetros coletados durante a simulação são denominados, dentro do ambiente OPNET de estatísticas.

A escolha de qual ou quais estatísticas devem ser coletadas para análise é feita através da opção "*Choose Individual Statistics"* disponível em todos os nós e *links* do GPRSS. Uma vez cumprida esta etapa, basta definir o tempo de simulação através da opção *"Configure Simulation"*, simular, e verificar o resultado com a opção *"View Results"*.

Para uma maior amplitude análise foram implementados sete cenários. O cenário 1 apresentada resultados para a validação do modelo. Os cenários 2, 3, 4, 5 e 6 apresentam resultados para uma avaliação comparativa do desempenho da rede GPRS quanto ao parâmetro atraso fim-a-fim, a partir da implementação das disciplinas de escalonamento de pacotes FIFO, PQ, WFQ, WRR e DWRR. O cenário 7 apresenta resultados para uma avaliação da taxa de utilização no enlace sem fio a partir de sessões de tráfegos distintos.

## 6.1 Cenário 1 : Verificação do GPRSS

O cenário 1 tem como objetivo principal fornecer resultados para a validação do GPRSS de acordo com o que este se propõe, ou seja, permitir simulações que obtenham resultados aproximados de uma rede GPRS real.

As Tabelas 6.1 e 6.2 apresentam os atributos configurados para este cenário.

| Atributo | Valor Configurado |
|---|---|
| *Link*: Servidor (HTTP, FP, SMTP, VoIP) → PDN | 34 Mbps |
| *Link*: PDN → GGSN | 34 Mbps |
| *Link*: GGSN → *Backbone* GPRS | 34 Mbps |
| *Link*: *Backbone* GPRS → SGSN | 34 Mbps |
| *Link*: SGSN → BSC | 2 Mbps |
| Taxa de Serviço de *Buffer*: PDN | 34 Mbps |
| Taxa de Serviço de *Buffer*: *Backbone* GPRS | 34 Mbps |
| Taxa de Serviço de *Buffer*: SGSN (BVC) | 2 Mbps |
| Taxa de Serviço de *Buffer*: BSC (BVC) | 2 Mbps |

**Tabela 6.1** – *Atributos - Cenário1*

Todos os *buffers* estão configurados com capacidade infinita e as taxas de serviço para *buffers* TLLI dependem da taxa efetiva de transmissão entre BSC e MS.

O cenário 1 também foi dividido em quatro sub-cenários: A,B,C e D. O objetivo desses sub-cenários é permitir a avaliação do GPRSS a partir de esquemas de codificação e números de *time slots* alocados (alocação fixa) distintos.

| Atributos | Servidor HTTP | Servidor FTP | Servidor SMTP | Servidor VoIP |
|---|---|---|---|---|
| Aplicação | HTTP | FTP | SMTP | VoIP |
| Canal_tx taxa (bps) | 34.000.000 | 34.000.000 | 34.000.000 | 34.000.000 |
| Canal_tx (formato) | G_pk_FR | G_pk_FR | G_pk_FR | G_pk_FR |
| FDP Intervalo entre pacotes (s) | Exp(5) | Exp(5) | Exp(5) | Exp(5) |
| FDP Tamanho do pacote (bits) | Const (8192) | Const (8192) | Const (8192) | Const (8192) |
| Formato do pacote | G_pk_IPV4 | G_pk_IPV4 | G_pk_IPV4 | G_pk_IPV4 |
| Início da transmissão (s) | 1 | 1 | 1 | 1 |
| Término da transmissão (s) | infinity | infinity | infinity | infinity |

**Tabela 6.2** – *Atributos: Servidores de Aplicação - Cenário1*

Os resultados dessa simulação (Tabela 6.3) mostram que o GPRSS atende ao comportamento esperado de uma rede real. Os pacotes gerados pelos servidores de

aplicações são recebidos de forma correta, mesmo passando por diversos processos que incluem encapsulamentos, segmentações, etc. A taxa e tamanho dos pacotes recebidos pela MS correspondem exatamente aos valores configurados no servidor. À medida que se aumenta o número de *time slots* alocados, assim como o esquema de codificação a capacidade de transmissão aumenta progressivamente, observa-se que a utilização do *link* BSC-BTS diminui significativamente. Além da utilização, o atraso fim-a-fim (entre GGSN e MS) e o espaço usado em *buffer* também sofrem um decréscimo em função do aumento da taxa efetiva de transmissão. O fato da utilização média apresentada no sub-cenário A ter sido alta é reflexo da FDP de intervalo entre pacotes que representa uma rede com um tráfego relativamente pesado.

| | **Cenário 1 (Tempo de simulação = 30min)** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | **Aplicação** | **CS / *TS*** | **Atraso médio fim-a-fim (s)** | **Atraso máx. fim-a-fim (s)** | **Taxa tx servidor (pacotes/s)** | **Taxa rx MS (pacotes/s)** | **Tamanho do pacote IPv4 - MS** | **Utilização do link BSC-BTS** | **Tamanho do buffer Esc_FIFO** |
| A | HTTP | CS-1 / TS1 | 5,667 | 17,743 | 7,020 | 7,000 | 8192 | - | - |
| | FTP | CS-1 / TS1 | 5,758 | 18,665 | 7,740 | 7,660 | 8192 | - | - |
| | SMTP | CS-1 / TS1 | 5,916 | 18,240 | 6,100 | 6,080 | 8192 | - | - |
| | VoIP | CS-1 / TS1 | 5,740 | 18,927 | 8,020 | 8,000 | 8192 | - | - |
| | - | CS-1 / TS1 | - | - | - | - | - | 88,620% | 88.623 bits |
| B | HTTP | CS-2 / TS3 | 0,506 | 0,782 | 7,020 | 7,020 | 8192 | - | - |
| | FTP | CS-2 / TS3 | 0,502 | 0,654 | 7,720 | 7,700 | 8192 | - | - |
| | SMTP | CS-2 / TS3 | 0,495 | 0,685 | 6,100 | 6,000 | 8192 | - | - |
| | VoIP | CS-2 / TS3 | 0,504 | 0,708 | 8,020 | 8,020 | 8192 | - | - |
| | - | CS-2 / TS3 | - | - | - | - | - | 19,094% | 1.192 bits |
| C | HTTP | CS-3 / TS6 | 0,215 | 0,257 | 7,020 | 7,020 | 8192 | - | - |
| | FTP | CS-3 / TS6 | 0,213 | 0,258 | 7,720 | 7,700 | 8192 | - | - |
| | SMTP | CS-3 / TS6 | 0,212 | 0,291 | 6,100 | 6,100 | 8192 | - | - |
| | VoIP | CS-3 / TS6 | 0,214 | 0,287 | 8,020 | 8,020 | 8192 | - | - |
| | - | CS-3 / TS6 | - | - | - | - | - | 8,140% | 362 bits |
| D | HTTP | CS-4 / TS8 | 0,118 | 0,133 | 7,020 | 7,020 | 8192 | - | - |
| | FTP | CS-4 / TS8 | 0,118 | 0,141 | 7,720 | 7,720 | 8192 | - | - |
| | SMTP | CS-4 / TS8 | 0,118 | 0,159 | 6,100 | 6,100 | 8192 | - | - |
| | VoIP | CS-4 / TS8 | 0,117 | 0,133 | 8,020 | 8,020 | 8192 | - | - |
| | - | CS-4 / TS8 | - | - | - | - | - | 4,300% | 128 bits |

**Tabela 6.3** – *Resultados - Cenário1*

## 6.2 Análise comparativa entre disciplinas de escalonamento

O cenário 1 apresentou resultados esperados e próximos do real. Contudo, observa-se que o atraso fim-a-fim com CS-1/TS1 não atende a classe de atraso determinada pelo perfil de QoS (Tabela 4.3). Esse atraso relativamente alto é devido ao uso da disciplina FIFO que trata todos os pacotes da mesma maneira.

As classes de precedência e atraso atribuídas às aplicações permitem que as mesmas sejam tratadas de forma diferenciada. O GPRS determina 3 classes de precedência e 4 classes de atraso para as SDUs (Tabela 6.4).

| Aplicações usadas na simulação | Classe de precedência | Classe de atraso | Atraso médio | Atraso máximo |
|---|---|---|---|---|
| HTTP | média | 2 | 15 s | 75 s |
| FTP | baixa | 3 | 75 s | 375 s |
| SMTP | baixa | 4 | ME | ME |
| VoIP | alta | 1 | 2 s | 7 s |

**Tabela 6.4** – *Classes de precedência e atraso*

O cenário 2 tem como objetivo principal obter resultados para uma análise comparativa entre as cinco disciplinas de escalonamento apresentadas no item 4.5.1.6 (FIFO, PQ, WFQ, WRR e DWRR) e verificar quais delas atendem aos requisitos de atraso das aplicações. Para uma avaliação mais consistente dessas disciplinas, as mesmas serão avaliadas sob condições de tráfego leve (cenário 2), tráfego médio (cenário 3) e tráfego médio-pesado (cenário 4), tráfego pesado (cenário 5) e tráfego extremamente pesado (cenário 6), conforme a Figura 6.1.

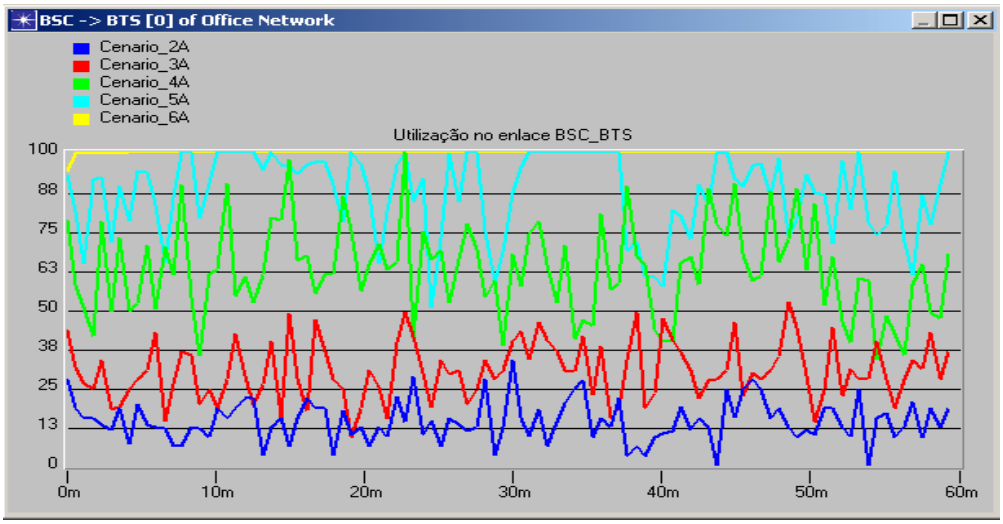


**Figura 6.1** – *Utilização do link BSC- BTS (Cenários 2,3,4,5 e 6)*

A configuração dos atributos de todos esses cenários permanece a mesma do cenário 1. Contudo, para representar o "pior caso", será utilizado apenas o esquema de codificação CS-1 e a alocação de *time slot* TS1.

O tempo de simulação utilizado para esses cenários é de uma hora.

### 6.2.1 Cenário 2 : Tráfego leve

Este cenário permite uma avaliação das disciplinas de escalonamento sob o aspecto de atraso fim-a-fim para um tráfego relativamente leve (utilização média do enlace entre BSC e BTS de 14 %).

| | Tráfego Leve / FDP Intervalo entre pacotes (s) = Exponencial (30) | | | | | | |
|---|---|---|---|---|---|---|---|
| | Aplicação | Disciplina | Atraso fim-a-fim (s) | | | Desvio padrão | Coeficiente de variação |
| | | | Mínimo | Médio | Máximo | | |
| A | HTTP | FIFO | 1,528 | 1,591 | 2,467 | 0,164 | 0,103 |
| | FTP | FIFO | 1,528 | 1,574 | 2,620 | 0,162 | 0,103 |
| | SMTP | FIFO | 1,528 | 1,598 | 2,870 | 0,213 | 0,133 |
| | VoIP | FIFO | 1,528 | 1,614 | 2,527 | 0,229 | 0,142 |
| B | HTTP | PQ | 1,517 | 2,018 | 18,055 | 2,213 | 1,097 |
| | FTP | PQ | 1,528 | 2,314 | 15,682 | 2,758 | 1,192 |
| | SMTP | PQ | 1,528 | 2,178 | 17,697 | 2,400 | 1,102 |
| | VoIP | PQ | 1,510 | 1,534 | 1,764 | 0,034 | 0,022 |
| C | HTTP | WFQ | 1,517 | 1,854 | 2,196 | 0,247 | 0,133 |
| | FTP | WFQ | 1,690 | 2,395 | 3,666 | 0,425 | 0,177 |
| | SMTP | WFQ | 1,744 | 2,289 | 3,210 | 0,286 | 0,125 |
| | VoIP | WFQ | 1,528 | 1,533 | 1,546 | 0,003 | 0,002 |
| D | HTTP | WRR | 1,528 | 1,666 | 7,774 | 0,751 | 0,451 |
| | FTP | WRR | 1,528 | 1,767 | 12,918 | 1,380 | 0,781 |
| | SMTP | WRR | 1,528 | 1,658 | 5,742 | 0,542 | 0,327 |
| | VoIP | WRR | 1,528 | 1,766 | 10,662 | 1,175 | 0,665 |
| E | HTTP | DWRR | 1,528 | 1,666 | 7,600 | 0,732 | 0,439 |
| | FTP | DWRR | 1,528 | 1,767 | 12,938 | 1,383 | 0,782 |
| | SMTP | DWRR | 1,528 | 1,666 | 5,702 | 0,555 | 0,333 |
| | VoIP | DWRR | 1,528 | 1,597 | 2,287 | 0,176 | 0,110 |

**Tabela 6.5** – *Resultados – Cenário2*

Observando os resultados da Tabela 6.5 verifica-se que todos os algoritmos atendem as classes de atraso. Observa-se também que a disciplina FIFO apresentou os menores atrasos exceto para VoIP, isso ocorre devido a sua baixa carga computacional e um intervalo entre pacotes relativamente alto. Contudo a disciplina PQ apresentou o menor atraso para a aplicação VoIP (mais sensível ao atraso), por atribuir prioridade absoluta a essa aplicação. Apesar dos atrasos serem baixos, pode ser que os mesmos não sejam suficientes para atender as exigências das aplicações, tais como VoIP. Assim, em uma transmissão real VoIP seria negociada uma maior capacidade de transmissão durante a ativação do contexto PDP para prover atrasos máximos inferiores a 200 ms. Contudo, como este trabalho tem objetivo de avaliar quais disciplinas de escalonamento atendem melhor as classes de atraso no pior caso (CS-1 / TS1), esta aplicação será mantida como referência de classe de precedência alta e classe de atraso 1.

## 6.2.2 Cenário 3 : Tráfego médio

Este cenário diferencia-se do anterior apenas quanto a FDP de intervalo entre pacotes, que tem como conseqüência uma utilização média do *link* BSC-BTS de 30 %.

| | Tráfego Médio / FDP Intervalo entre pacotes (s) = Exponencial (15) | | | | | | |
|---|---|---|---|---|---|---|---|
| | Aplicação | Disciplina | Atraso fim-a-fim (s) | | | Desvio padrão | Coeficiente de variação |
| | | | Mínimo | Médio | Máximo | | |
| A | HTTP | FIFO | 1,528 | 1,736 | 3,148 | 0,309 | 0,178 |
| | FTP | FIFO | 1,528 | 1,781 | 3,494 | 0,370 | 0,208 |
| | SMTP | FIFO | 1,528 | 1,733 | 3,528 | 0,328 | 0,189 |
| | VoIP | FIFO | 1,528 | 1,776 | 3,383 | 0,338 | 0,190 |
| B | HTTP | PQ | 1,513 | 2,276 | 12,814 | 1,818 | 0,799 |
| | FTP | PQ | 1,528 | 3,663 | 24,565 | 3,691 | 1,008 |
| | SMTP | PQ | 1,528 | 3,042 | 17,884 | 3,000 | 0,986 |
| | VoIP | PQ | 1,512 | 1,559 | 1,893 | 0,073 | 0,047 |
| C | HTTP | WFQ | 1,544 | 2,304 | 3,798 | 0,362 | 0,157 |
| | FTP | WFQ | 1,838 | 3,007 | 3,906 | 0,690 | 0,230 |
| | SMTP | WFQ | 2,359 | 3,413 | 5,652 | 0,572 | 0,168 |
| | VoIP | WFQ | 1,513 | 1,551 | 1,566 | 0,011 | 0,007 |
| D | HTTP | WRR | 1,528 | 3,213 | 66,263 | 7,372 | 2,294 |
| | FTP | WRR | 1,528 | 2,510 | 16,251 | 2,100 | 0,837 |
| | SMTP | WRR | 1,528 | 2,033 | 9,587 | 1,184 | 0,582 |
| | VoIP | WRR | 1,523 | 1,705 | 2,893 | 0,246 | 0,144 |
| E | HTTP | DWRR | 1,528 | 4,411 | 66,263 | 8,078 | 1,832 |
| | FTP | DWRR | 1,528 | 3,021 | 16,378 | 2,530 | 0,838 |
| | SMTP | DWRR | 1,528 | 2,552 | 14,515 | 2,210 | 0,866 |
| | VoIP | DWRR | 1,523 | 1,672 | 5,711 | 0,452 | 0,270 |

**Tabela 6.6** – *Resultados – Cenário3*

Observando os resultados da Tabela 6.6, verifica-se que apesar dos valores serem diferentes, os resultados foram os mesmos obtidos no cenário 2, ou seja todas as disciplinas atenderam as classes de atrasos.

## 6.2.3 Cenário 4: Tráfego médio-pesado

Este cenário oferece uma carga maior à rede e diferencia-se do anterior apenas quanto a FDP de intervalo entre pacotes, que tem como conseqüência uma utilização média do *link* BSC-BTS de 63 %.

| | Tráfego Médio-Pesado / FDP Intervalo entre pacotes (s) = Exponencial (7) | | | | | | |
|---|---|---|---|---|---|---|---|
| | Aplicação | Disciplina | Atraso fim-a-fim (s) | | | Desvio padrão | Coeficiente de variação |
| | | | Mínimo | Médio | Máximo | | |
| A | HTTP | FIFO | 1,528 | 2,343 | 5,681 | 0,751 | 0,320 |
| | FTP | FIFO | 1,528 | 2,394 | 5,775 | 0,736 | 0,307 |
| | SMTP | FIFO | 1,528 | 2,398 | 6,066 | 0,852 | 0,355 |
| | VoIP | FIFO | 1,528 | 2,371 | 5,647 | 0,744 | 0,314 |
| B | HTTP | PQ | 1,515 | 2,352 | 5,920 | 1,008 | 0,428 |
| | FTP | PQ | 1,528 | 9,200 | 38,055 | 8,372 | 0,910 |
| | SMTP | PQ | 1,528 | 9,346 | 40,075 | 8,350 | 0,893 |
| | VoIP | PQ | 1,513 | 1,619 | 2,035 | 0,120 | 0,074 |
| C | HTTP | WFQ | 1,519 | 2,522 | 5,920 | 1,125 | 0,446 |
| | FTP | WFQ | 1,528 | 7,457 | 38,055 | 6,765 | 0,907 |
| | SMTP | WFQ | 1,528 | 7,325 | 31,317 | 6,024 | 0,822 |
| | VoIP | WFQ | 1,513 | 1,887 | 6,911 | 0,995 | 0,527 |
| D | HTTP | WRR | 1,528 | 4,551 | 55,256 | 6,018 | 1,322 |
| | FTP | WRR | 1,528 | 4,622 | 13,687 | 2,501 | 0,541 |
| | SMTP | WRR | 1,528 | 5,048 | 15,230 | 3,259 | 0,646 |
| | VoIP | WRR | 1,525 | 1,944 | 3,677 | 0,462 | 0,237 |
| E | HTTP | DWRR | 1,876 | 4,414 | 5,667 | 0,717 | 0,162 |
| | FTP | DWRR | 2,711 | 3,943 | 4,712 | 0,501 | 0,127 |
| | SMTP | DWRR | 2,855 | 4,433 | 5,054 | 0,301 | 0,068 |
| | VoIP | DWRR | 1,680 | 1,937 | 2,034 | 0,104 | 0,054 |

**Tabela 6.7** – *Resultados – Cenário4*

Observando a Tabela 6.7 verifica-se que ocorreu um aumento generalizado dos atrasos. Observa-se também que apenas a disciplina FIFO não atendeu a classe de atraso 1. Este resultado já era esperado, uma vez que a disciplina FIFO não possui mecanismos de classificação e escalonamento de pacotes.

### 6.2.4 Cenário 5 : Tráfego pesado

Este cenário representa um esforço maior para a rede manter os atrasos dentro de níveis aceitáveis. Este se diferencia do anterior apenas quanto a FDP de intervalo entre pacotes, que tem como conseqüência uma utilização média do *link* BSC-BTS bastante alta sendo igual a 87 %.

| | Tráfego Pesado / FDP Intervalo entre pacotes (s) = Exponencial (5) | | | | | | |
|---|---|---|---|---|---|---|---|
| | Aplicação | Disciplina | Atraso fim-a-fim (s) | | | Desvio padrão | Coeficiente de variação |
| | | | Mínimo | Médio | Máximo | | |
| A | HTTP | FIFO | 1,701 | 5,755 | 17,936 | 4,512 | 0,784 |
| | FTP | FIFO | 1,716 | 5,739 | 19,809 | 4,432 | 0,772 |
| | SMTP | FIFO | 1,528 | 5,936 | 18,521 | 4,578 | 0,771 |
| | VoIP | FIFO | 1,578 | 5,767 | 19,685 | 4,521 | 0,784 |
| B | HTTP | PQ | 1,526 | 2,827 | 7,581 | 1,166 | 0,413 |
| | FTP | PQ | 7,202 | 279,525 | 506,572 | 160,085 | 0,573 |
| | SMTP | PQ | 6,131 | 279,889 | 506,767 | 159,839 | 0,571 |
| | VoIP | PQ | 1,510 | 1,662 | 2,167 | 0,144 | 0,087 |
| C | HTTP | WFQ | 3,491 | 5,491 | 6,654 | 0,789 | 0,144 |
| | FTP | WFQ | 3,806 | 6,183 | 7,504 | 0,999 | 0,162 |
| | SMTP | WFQ | 4,031 | 6,588 | 7,884 | 0,973 | 0,148 |
| | VoIP | WFQ | 3,175 | 5,609 | 7,904 | 0,906 | 0,162 |
| D | HTTP | WRR | 1,680 | 6,698 | 27,760 | 4,883 | 0,729 |
| | FTP | WRR | 2,167 | 14,288 | 47,067 | 12,200 | 0,854 |
| | SMTP | WRR | 1,924 | 14,360 | 47,985 | 12,357 | 0,860 |
| | VoIP | WRR | 1,581 | 3,787 | 14,255 | 2,555 | 0,675 |
| E | HTTP | DWRR | 1,607 | 8,435 | 48,607 | 6,986 | 0,828 |
| | FTP | DWRR | 2,218 | 9,793 | 29,691 | 7,131 | 0,728 |
| | SMTP | DWRR | 1,909 | 9,949 | 30,572 | 7,319 | 0,736 |
| | VoIP | DWRR | 1,630 | 4,531 | 17,459 | 3,446 | 0,761 |

**Tabela 6.8** – *Resultados – Cenário5*

Observa-se a partir da Tabela 6.8 que esta configuração representa o ponto limite para o restante das disciplinas, uma vez que todas deixaram de atender as classes de atraso. Uma observação importante é que disciplina PQ atende a classe 1 (mais crítica) e não atende apenas a classe 3, isso ocorreu porque a PQ atribui prioridade absoluta à classe 1. Observa-se também que a classe 4 do sub-cenário B apresentou um atraso relativamente alto, porém aceito por se tratar de uma classe de Melhor Esforço.

## 6.2.5 Cenário 5 : Tráfego extremamente pesado

Este cenário apresenta uma condição extremamente crítica para todas as aplicações. Esse se diferencia do anterior apenas quanto a FDP de intervalo entre pacotes, que tem como conseqüência uma utilização média do *link* BSC-BTS de 100 %, ou seja, congestionamento intenso.

| | Tráfego Extremamente pesado / FDP Intervalo entre pacotes (s) = Exponencial (3) | | | | | | |
|---|---|---|---|---|---|---|---|
| | Aplicação | Disciplina | Atraso fim-a-fim (s) | | | Desvio padrão | Coeficiente de variação |
| | | | Mínimo | Médio | Máximo | | |
| A | HTTP | FIFO | 11,279 | 296,323 | 579,754 | 169,182 | 0,571 |
| | FTP | FIFO | 11,526 | 295,942 | 579,439 | 169,183 | 0,572 |
| | SMTP | FIFO | 9,136 | 296,132 | 579,900 | 169,609 | 0,573 |
| | VoIP | FIFO | 12,139 | 296,232 | 579,891 | 169,300 | 0,572 |
| B | HTTP | PQ | 1,742 | 5,341 | 16,376 | 3,044 | 0,570 |
| | FTP | PQ | 14,668 | 1428,381 | 2614,410 | 787,821 | 0,552 |
| | SMTP | PQ | 20,856 | 1346,474 | 2614,540 | 784,597 | 0,583 |
| | VoIP | PQ | 1,516 | 1,790 | 3,087 | 0,235 | 0,131 |
| C | HTTP | WFQ | 10,085 | 578,416 | 1133,800 | 326,774 | 0,565 |
| | FTP | WFQ | 11,022 | 580,236 | 1136,467 | 327,268 | 0,564 |
| | SMTP | WFQ | 10,029 | 580,570 | 1134,713 | 327,099 | 0,563 |
| | VoIP | WFQ | 12,139 | 579,891 | 1135,221 | 327,263 | 0,564 |
| D | HTTP | WRR | 6,610 | 357,998 | 695,575 | 201,405 | 0,563 |
| | FTP | WRR | 16,294 | 818,021 | 1542,649 | 453,843 | 0,555 |
| | SMTP | WRR | 14,485 | 825,546 | 1544,616 | 450,020 | 0,545 |
| | VoIP | WRR | 5,453 | 352,653 | 693,483 | 200,166 | 0,568 |
| E | HTTP | DWRR | 7,927 | 415,301 | 836,222 | 235,231 | 0,566 |
| | FTP | DWRR | 15,794 | 761,022 | 1431,251 | 421,961 | 0,554 |
| | SMTP | DWRR | 13,558 | 761,114 | 1431,636 | 421,275 | 0,553 |
| | VoIP | DWRR | 5,941 | 403,714 | 798,994 | 229,808 | 0,569 |

**Tabela 6.9** – *Resultados – Cenário6*

Como já era esperado, todas as disciplinas deixaram de atender as classes de atraso (Tabela 6.9). A disciplina FIFO, por tratar todos pacotes da mesma maneira, obteve atrasos bastante próximos para as diferentes aplicações. A disciplina PQ apesar de atender três classes (melhor desempenho em termos de número de classes atendidas) obteve o pior resultado quanto aos atrasos de duas classes. Os atrasos com a disciplina WFQ se mantiveram na média. Os resultados obtidos pelas disciplinas WRR e DWRR foram bastante próximos, contudo apesar do bom desempenho obtido nos cenários anteriores não foi possível atender as classes de atraso sob esta condição de tráfego.

Observando gráficos (Figuras 6.2, 6.3, 6.4 e 6.5) dos cenários 2,3,4 e 5 onde as disciplinas de escalonamento de pacotes foram submetidas a uma avaliação sob o aspecto de atraso, é importante destacar os benefícios de se trabalhar com a configuração dos pesos das disciplinas WFQ, WRR e DWRR. Nestes cenários os pesos foram atribuídos diferentemente de forma a manter os atrasos dentro de

condições desejadas. Tais recursos podem ser configurados de forma personalizada ao tráfego e com uma boa precisão. De um modo geral nota-se que as disciplinas WFQ, WRR e DWRR também apresentaram os melhores desempenhos mantendo os atrasos dentro de condições desejadas. Avaliando a condição mais crítica (classe de atraso 1) é possível destacar as disciplinas WRR e DWRR demonstraram um ótimo desempenho com relação aos atrasos, uma boa precisão na alocação de largura de banda de saída a partir de uma carga computacional relativamente baixa.

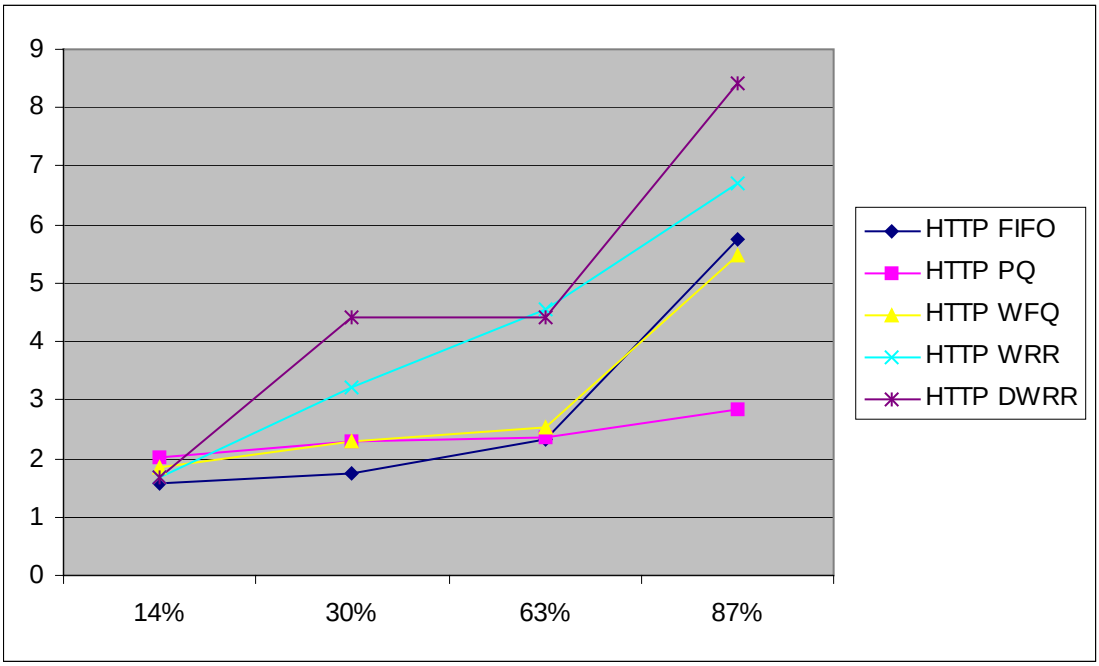


**Figura 6.2** – *Atraso x utilização (HTTP)*

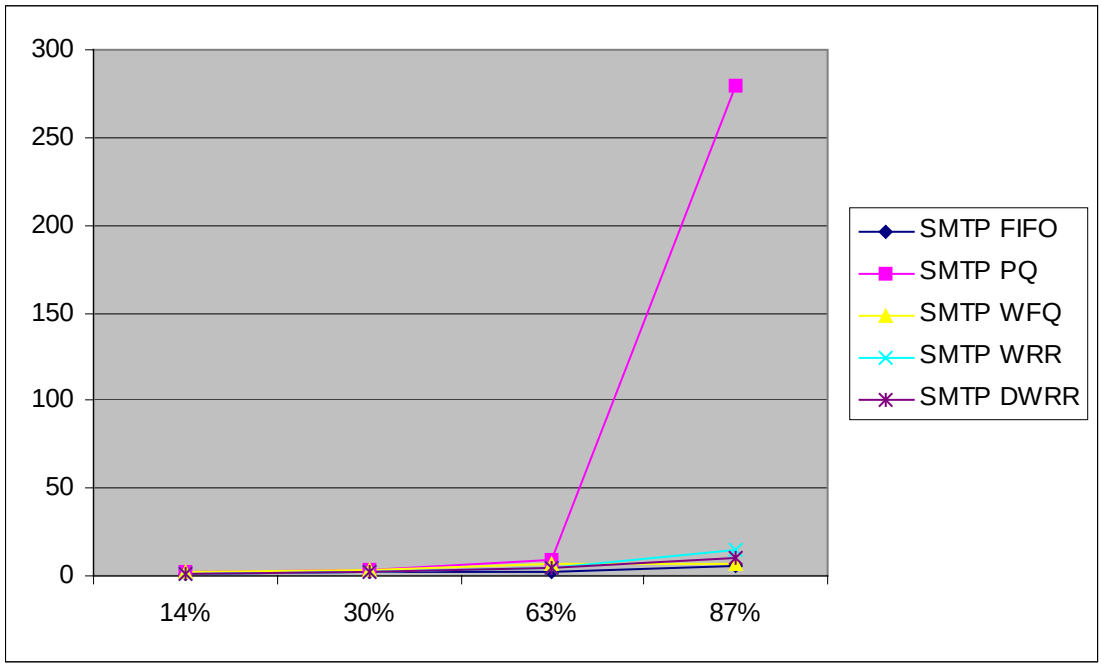


**Figura 6.3** – *Atraso x utilização (FTP)*

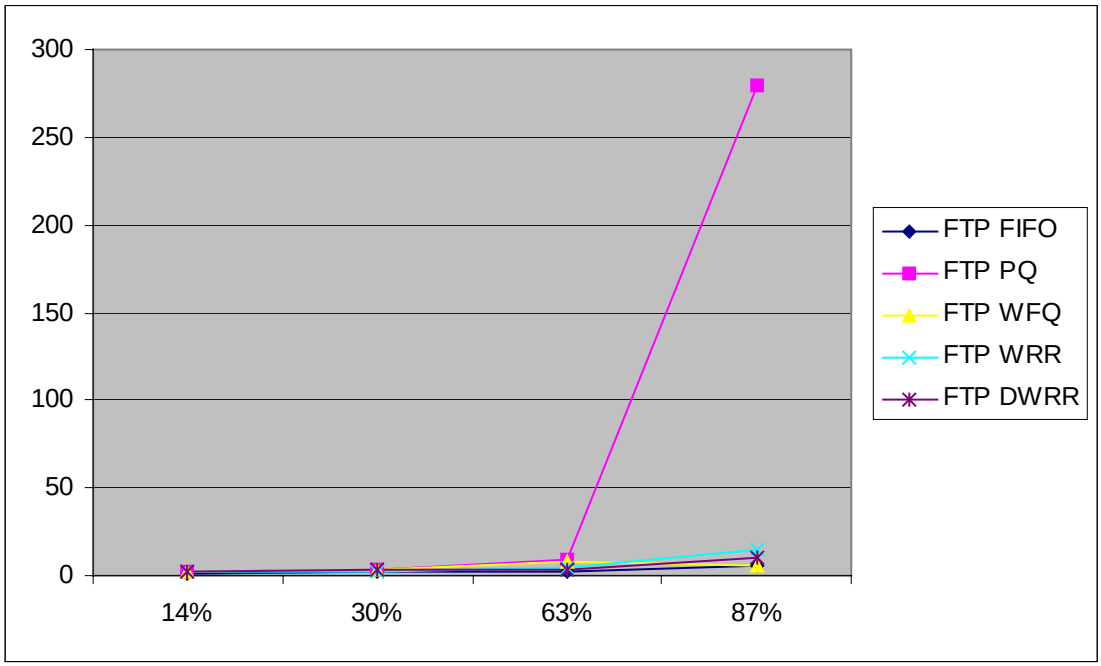


**Figura 6.4** – *Atraso x utilização (SMTP)*

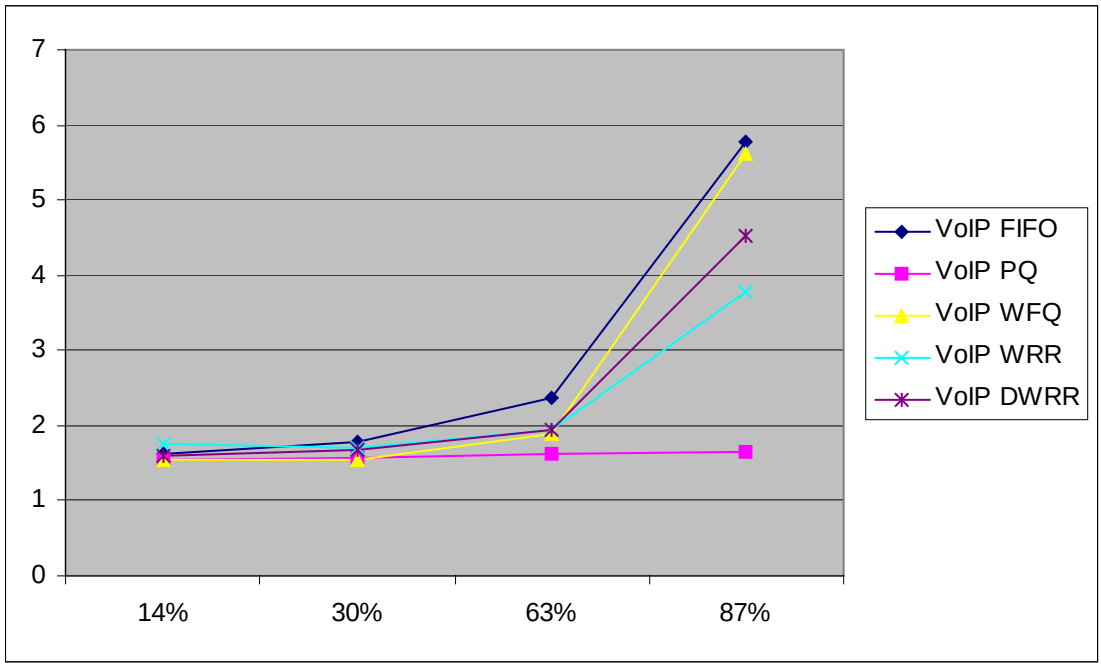


**Figura 6.5** – *Atraso x utilização (VoIP)*

## 6.3 Cenário 7 : Análise de tráfego

Os cenários anteriores permitiram uma análise das disciplinas de escalonamento a partir de tráfegos iguais para as diversas aplicações. O cenário atual permite avaliar a taxa de utilização no enlace sem fio e pacotes recebidos a partir de sessões de tráfegos distintos (Tabela 6.10).

| Atributos | Servidor HTTP | Servidor FTP | Servidor SMTP | Servidor VoIP |
|---|---|---|---|---|
| Aplicação | HTTP | FTP | SMTP | VoIP |
| Canal_tx taxa (bps) | 34.000.000 | 34.000.000 | 34.000.000 | 34.000.000 |
| Canal_tx (formato) | G_pk_FR | G_pk_FR | G_pk_FR | G_pk_FR |
| FDP Intervalo entre pacotes (s) | Exponencial (10) | Lognormal(4,3) | Exponencial (4) | Constante(1) |
| FDP Tamanho do pacote (bits) | Unif. (5000,30000) | Logn.(25000,10000) | Unif. (1000,5000) | Const. (500) |
| Formato do pacote | G_pk_IPV4 | G_pk_IPV4 | G_pk_IPV4 | G_pk_IPV4 |
| Início da transmissão (s) | 35 | 20 | 150 | 10 |
| Término da transmissão (s) | 280 | 80 | 200 | 290 |

**Tabela 6.10** – *Atributos– Cenário7*

Este cenário possui dois sub-cenários que se diferenciam no esquema de codificação de canal e alocação de *time slots*. Todos esses atributos têm como resultado gráficos como o da Figura 6.6, obtido com o uso da disciplina FIFO.

O gráfico permite visualizar os diferentes comportamentos do tráfego em função das aplicações. No geral observa-se um comportamento de “rajadas” para as aplicações HTTP, FTP, SMTP e um comportamento “constante” para VoIP com amplitudes definidas pelos tamanhos dos pacotes e duração definida pelo inicio de transmissão, fim de transmissão e FDP de intervalo entre pacotes.

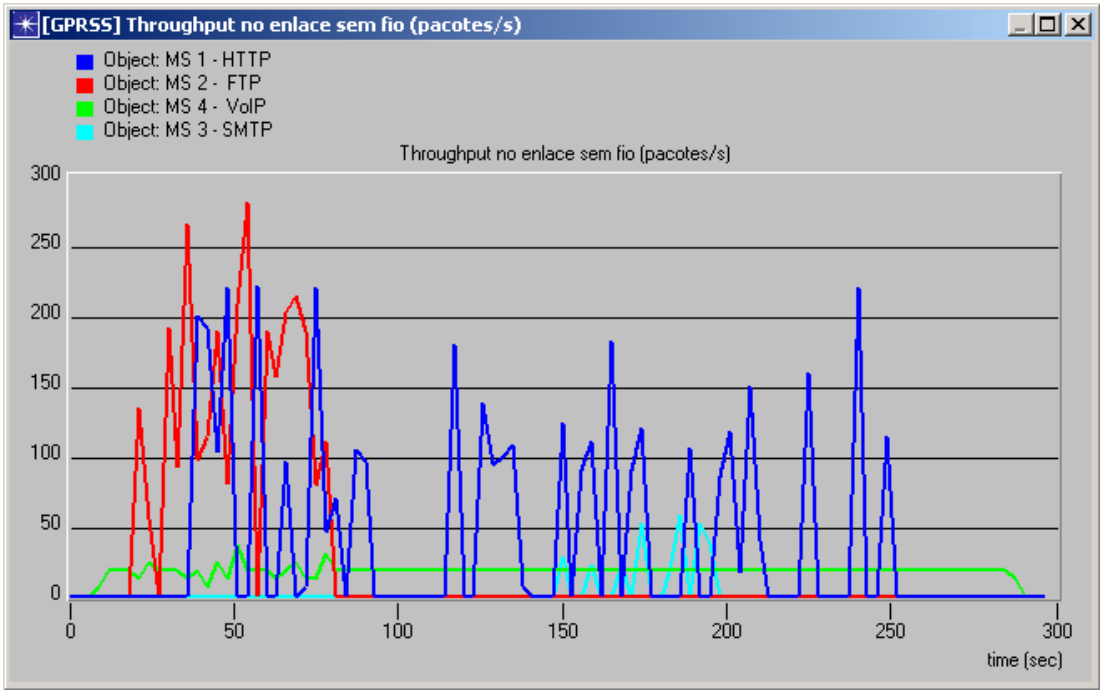


**Figura 6.6** – *Throughput no enlace sem fio*

Analisando os resultados da Tabela 6.11, constata-se que a disciplina PQ não “suportou” o tráfego e inviabilizou a transmissão dos pacotes da aplicação SMTP (menor prioridade). O mesmo poderia ter ocorrido com as disciplinas WFQ, WRR e DWRR, caso essas tivessem sido configuradas com um alto grau de prioridade para uma determinada classe de pacotes, contudo neste cenário as mesmas foram configuradas de forma equilibrada. (ex: WFQ - Classe de precedência 1 = 30 %, precedência 2 = 30 %, precedência 3 = 40 %).

Outra observação importante que se faz a partir dos resultados desse cenário é que tanto a utilização média quanto a máxima são maiores no sub-cenário com esquema de codificação CS-1. Isso ocorre porque o campo “Dados” do pacote G-pk_RLCMAC_CS1 tem tamanho máximo de 160 bits, enquanto no G-

pk_RLCMAC_CS3 o tamanho máximo é de 288 bits. Assim é necessário transmitir mais pacotes no CS-1 do que no CS-3 para atingir o mesmo volume de dados.

| | | | CS-3/TS3 | | | CS-1/TS1 | | |
|---|---|---|---|---|---|---|---|---|
| | **Aplicação** | **Duração da conexão (s)** | **Pacotes IP recebidos** | ***Throughput (pacotes/s)*** | | **Pacotes IP recebidos** | ***Throughput (pacotes/s)*** | |
| | | | | **Médio** | **Máximo** | | **Médio** | **Máximo** |
| **FIFO** | HTTP | 245,000 | 254,423 | 39,260 | 226,000 | 264,600 | 68,100 | 292,000 |
| | FTP | 60,000 | 60,000 | 28,540 | 192,000 | 60,000 | 49,500 | 292,000 |
| | SMTP | 50,000 | 75,000 | 2,900 | 58,000 | 66,667 | 5,000 | 92,000 |
| | VoIP | 280,000 | 831,064 | 16,740 | 18,000 | 879,140 | 22,320 | 110,000 |
| **PQ** | HTTP | 245,000 | 264,600 | 39,260 | 244,000 | 245,000 | 46,580 | 266,000 |
| | FTP | 60,000 | 75,000 | 28,540 | 280,000 | 60,000 | 5,260 | 200,000 |
| | SMTP | 50,000 | 100,000 | 2,900 | 72,000 | 0,000 | 0,000 | 0,000 |
| | VoIP | 280,000 | 831,064 | 16,740 | 18,000 | 831,064 | 22,320 | 24,000 |
| **WFQ** | HTTP | 245,000 | 254,423 | 39,260 | 226,000 | 264,600 | 68,100 | 292,000 |
| | FTP | 60,000 | 60,000 | 28,540 | 192,000 | 60,000 | 49,500 | 292,000 |
| | SMTP | 50,000 | 75,000 | 2,900 | 58,000 | 66,667 | 5,000 | 92,000 |
| | VoIP | 280,000 | 831,064 | 16,740 | 18,000 | 840,000 | 22,320 | 110,000 |
| **WRR** | HTTP | 245,000 | 275,625 | 39,260 | 226,000 | 254,800 | 67,680 | 292,000 |
| | FTP | 60,000 | 64,286 | 28,540 | 129,000 | 60,000 | 49,500 | 292,000 |
| | SMTP | 50,000 | 85,714 | 2,900 | 58,000 | 85,714 | 5,000 | 102,000 |
| | VoIP | 280,000 | 831,064 | 16,740 | 18,000 | 1107,867 | 22,320 | 150,000 |
| **DWRR** | HTTP | 245,000 | 254,423 | 39,260 | 222,000 | 255,208 | 67,940 | 300,000 |
| | FTP | 60,000 | 69,231 | 28,540 | 282,000 | 60,000 | 49,500 | 292,000 |
| | SMTP | 50,000 | 75,000 | 2,900 | 58,000 | 75,000 | 5,000 | 92,000 |
| | VoIP | 280,000 | 831,064 | 16,740 | 36,000 | 868,000 | 22,320 | 134,000 |

**Tabela 6.11** – *Resultados – Cenário7*

A variação do *throughput* de uma disciplina para outra dentro do mesmo sub-cenário ocorre em função do escalonamento e da intensidade do tráfego. Com estes resultados não se pode afirmar que o *throughput* é maior para com a utilização do CS-1 porque este modelo não possui o codificador convolucional implementado (possui apenas as estruturas de pacotes para os quatro esquemas de codificação) e também por se utilizar um modelo de canal com taxa de erro de bit igual a zero o que não força re-transmissões.

## 6.4 Avaliação do cenário 4

Como já foi demonstrado no item 6.2.3, o cenário 4 apresenta uma intensidade de tráfego intermediária dentre os cenários apresentados.

A partir desse cenário foram obtidos alguns gráficos que complementam o entendimento sobre as disciplinas de escalonamento de pacotes do ponto de vista de atraso.

Como se sabe a disciplina FIFO trata todos os pacotes da mesma maneira, independente de classes. Assim, o atraso gerado em fila é bem próximo para todos os pacotes e pode ser comprovado através da Figura 6.7. Em caso de congestionamento o atraso dos pacotes cresce na mesma proporção.

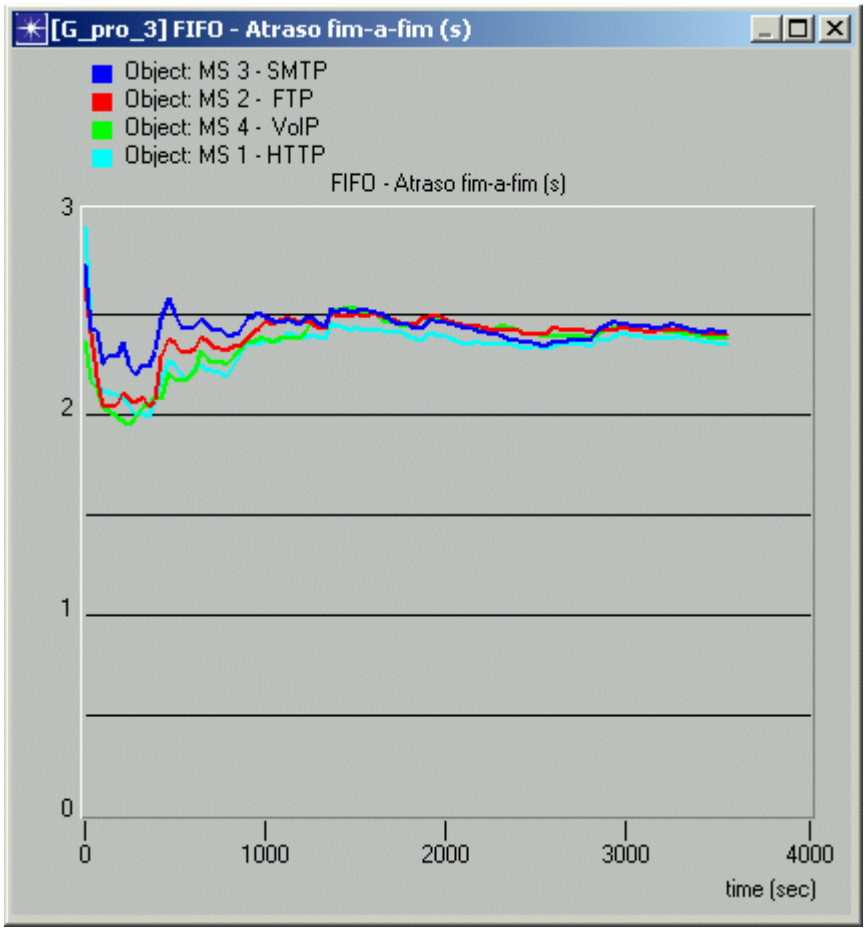


**Figura 6.7** – *FIFO – Atraso fim-a-fim (s)*

A disciplina PQ faz o escalonamento de pacotes de acordo com a classe de precedência, onde três sub-filas as determinam (alta, média e baixa prioridade). O pacote é inserido na sub-fila de acordo com sua classe de atraso. Assim, os pacotes com classes de atraso mais baixas (ex: VoIP) terão prioridade alta, e assim por diante.

Como conseqüência disso, tem-se para qualquer cenário um atraso muito pequeno e praticamente constante para VoIP (ideal para aplicações de tempo-real). Para HTTP que foi configurada com a classe 2 o atraso também é pequeno. Contudo, para as outras aplicações o atraso será tão alto quanto maior for a intensidade do tráfego. Tudo isso pode ser verificado através da Figura 6.8.

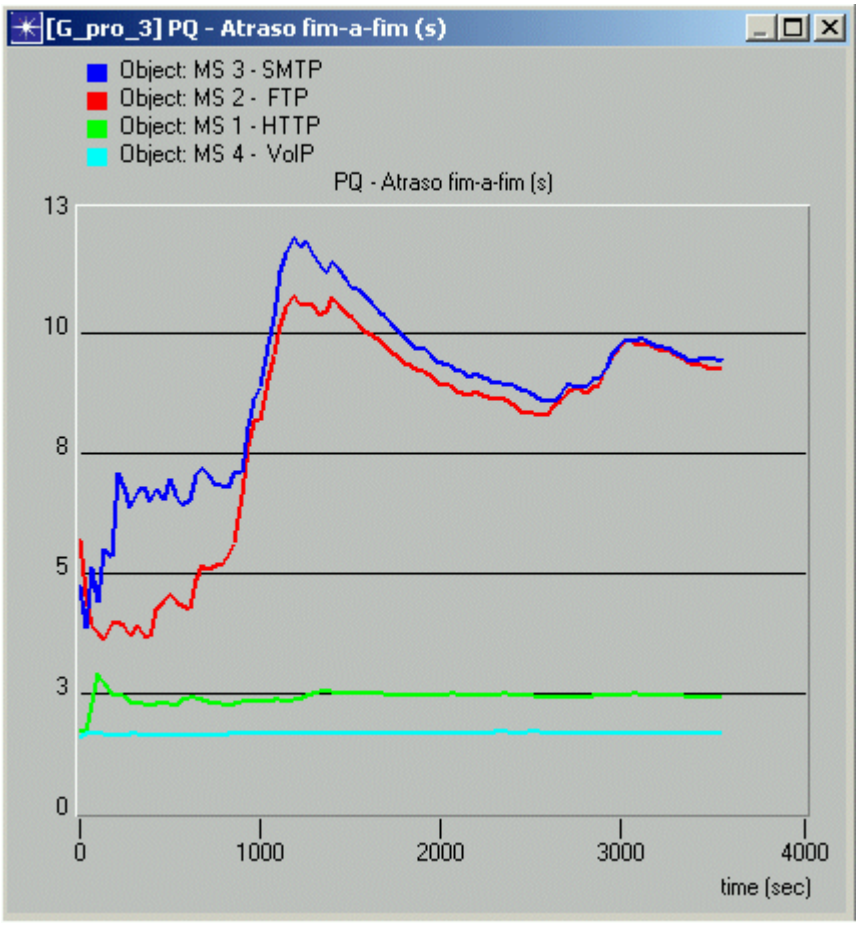


**Figura 6.8** – *PQ – Atraso fim-a-fim (s)*

A disciplina WFQ permite associar pesos a cada sub-fila, o que é muito interessante quando se deseja ter o controle preciso sobre a classificação e

conseqüentemente o escalonamento dos pacotes. Uma das vantagens é que se pode trabalhar como fosse uma FIFO, PQ e de forma personalizada.

Para este cenário foram atribuídos os seguintes pesos: Sub-fila 0 (prioridade alta) igual a 85 %, Sub-fila 1 (prioridade média) igual a 9 % Sub-fila 2 (prioridade baixa) igual a 6 %. Os pesos foram atribuídos desta forma apenas para se obter o menor atraso possível para VoIP. Os resultados podem ser observados na Figura 6.9.

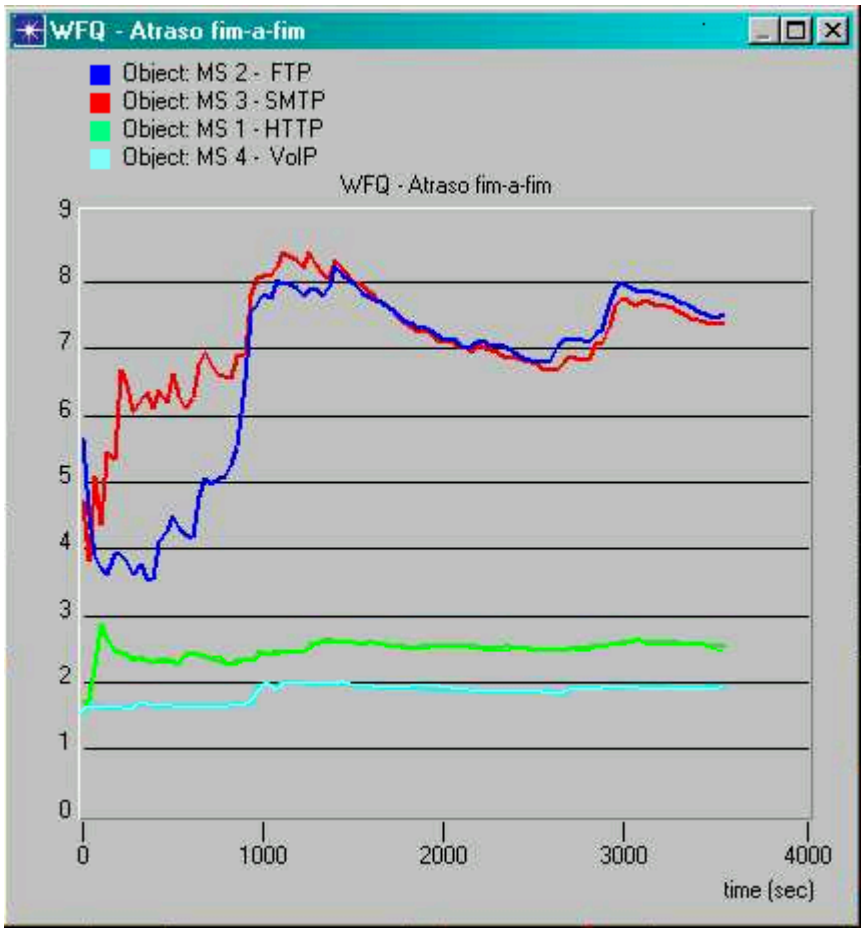


**Figura 6.9** – *WFQ – Atraso fim-a-fim (s)*

A disciplina WRR tem a vantagem de permitir a atribuição de pesos às sub-filas com um algoritmo relativamente simples.

Para esse cenário optou-se por atribuir o máximo peso à aplicação VoIP de forma a obter o menor atraso possível. Os pesos atribuídos foram: Sub-fila 0

(prioridade alta) igual a 98 %, Sub-fila 1 (prioridade média) igual a 1 % Sub-fila 2 (prioridade baixa) igual a 1 %. Os resultados podem ser observados na Figura 6.10.

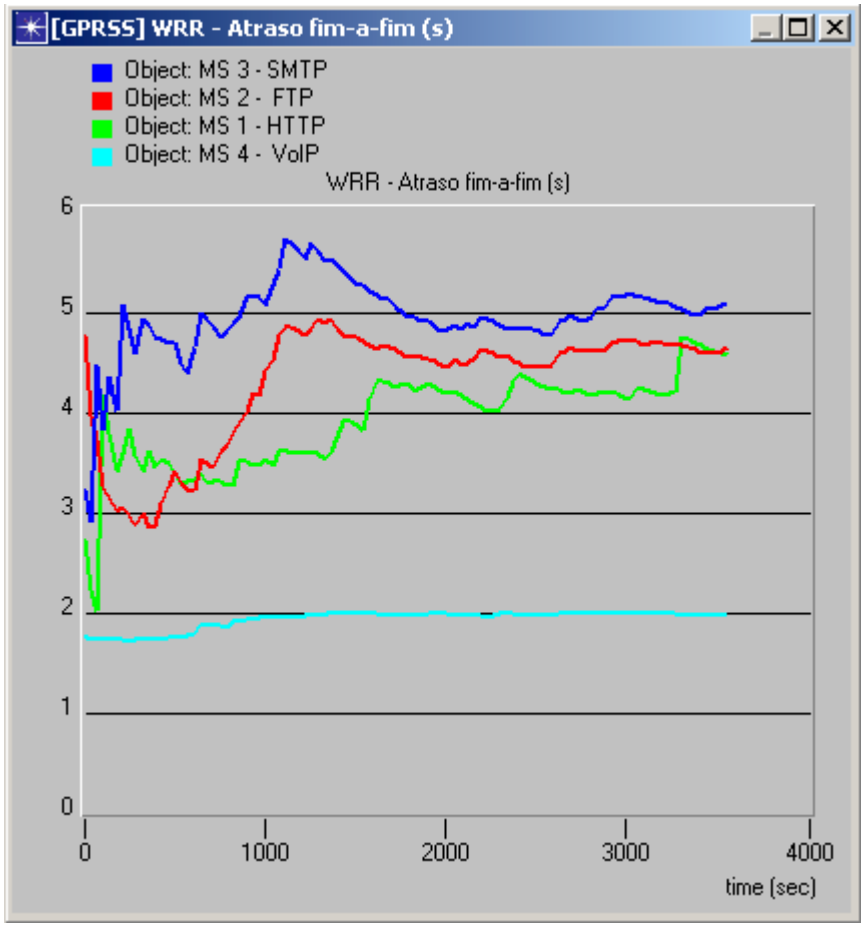


**Figura 6.10** – *WRR – Atraso fim-a-fim (s)*

A disciplina DWRR apresenta resultados muito próximos ao WRR (devido ao escalonamento de pacotes – blocos de rádio - com tamanhos fixos, se houvesse uma variação significativa no tamanho dos pacotes os benefícios do DWRR seriam mais evidentes), contudo com uma melhor precisão

Assim como nas disciplinas anteriores optou-se por atribuir o máximo peso à aplicação VoIP de forma a obter o menor atraso possível. Os pesos atribuídos foram: Sub-fila 0 (prioridade alta) igual a 48.700 bits Sub-fila 1 (prioridade média) igual a 800 bits Sub-fila 2 (prioridade baixa) igual a 500 bits. Os resultados podem ser observados na Figura 6.11.

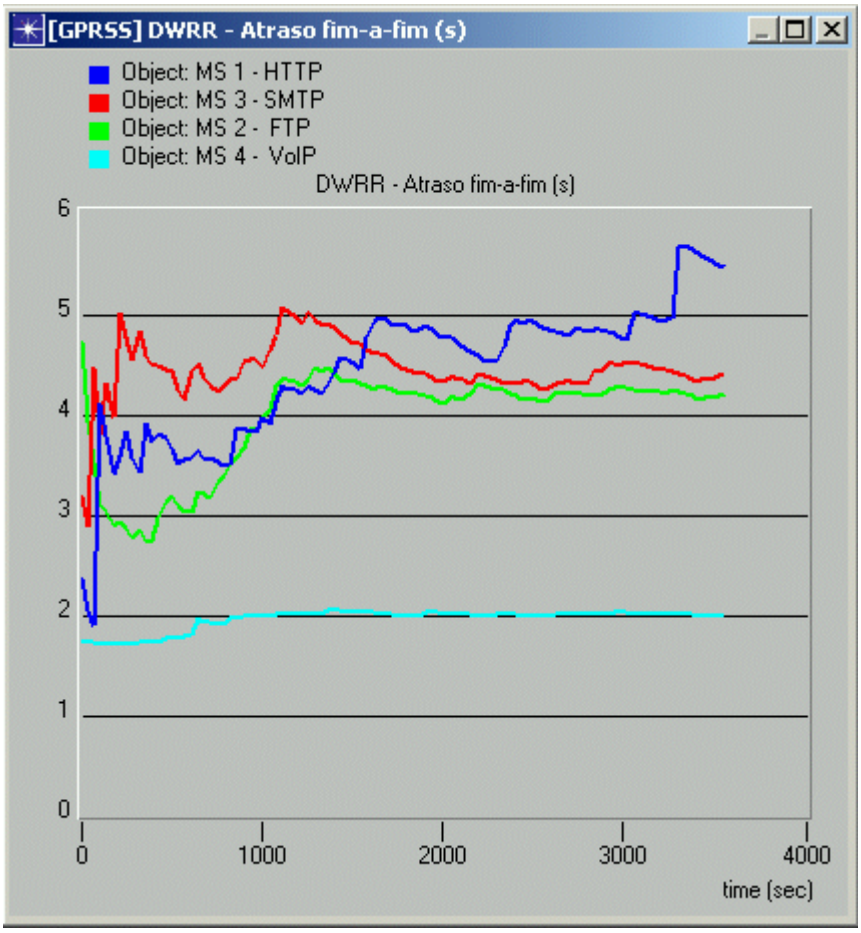


**Figura 6.11** – *DWRR – Atraso fim-a-fim (s)*

## 6.5 Avaliação dos *Buffers*

O GPRSS permite também que se obtenha dados para um dimensionamento adequado dos recursos da rede tais como *links* e *buffers*. No caso de *buffers* estes dados são importantes para dimensionar o tamanho (bits) e taxas de serviço (bps), e assim evitar o descarte de pacotes em caso de tráfego intenso.

Ainda referente ao cenário 4, foram coletados o atraso e tamanho do *Buffer* DWRR. Os resultados são apresentados na Figura 6.12.

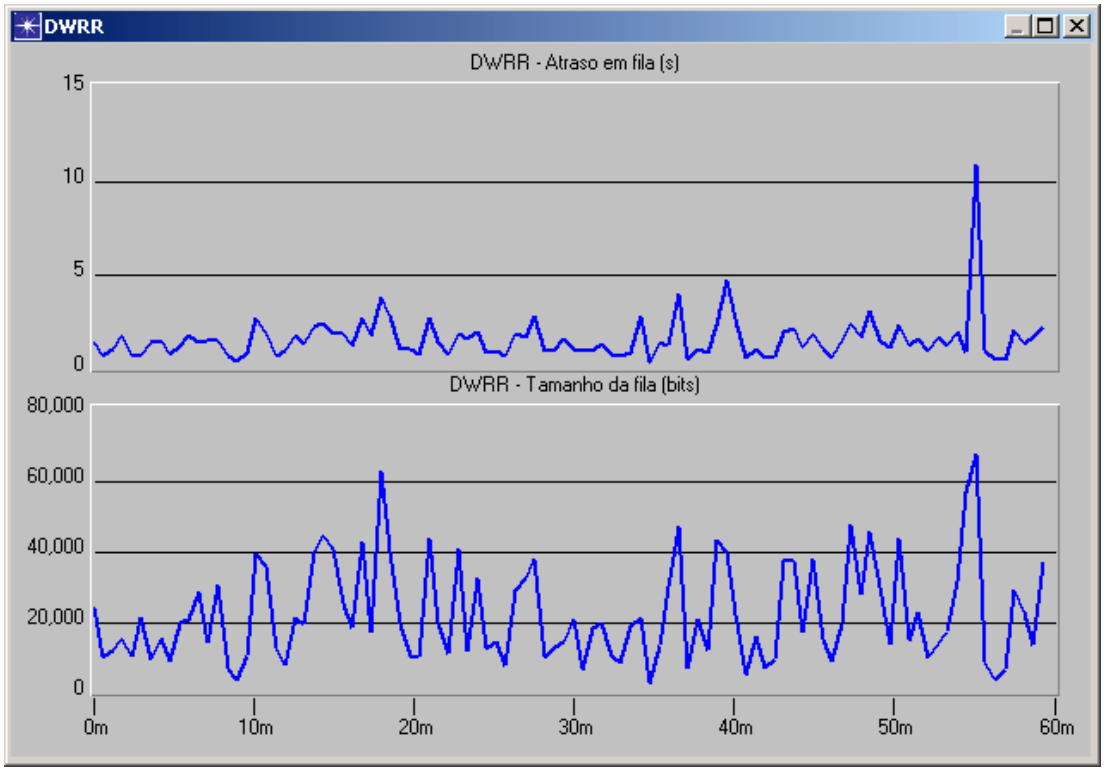


**Figura 6.12** – *DWRR – Atraso e tamanho da fila*

O primeiro gráfico ilustra o atraso em *buffer* e representa o tempo que o pacote fica na fila. Observa-se um atraso médio de 1,5 s e máximo de 10,8 s.

O segundo gráfico representa o tamanho do espaço ocupado pelos pacotes dentro da fila. Observa-se um tamanho médio de 21.810 bits e tamanho máximo de 67.285 bits.

## 6.6 Conclusão

O capítulo atual apresentou os resultados e análises obtidas a partir do modelo GPRSS proposto. Os objetivos principais eram demonstrar que o modelo representa uma rede GPRS real e avaliar a implementação/desempenho de várias disciplinas de escalonamento de pacotes para provisão de uma qualidade de serviço satisfatória.

Vários cenários foram “construídos” para oferecer uma maior amplitude de análise. Assim, foi possível analisar os tamanhos dos pacotes, taxas de transmissão, utilização de enlaces, etc.

Pode-se analisar os atrasos de pacotes fim-a-fim dentro de uma rede GPRS, e ainda, compara-los a partir da implementação de várias disciplinas de escalonamento de pacotes. A análise ocorreu tendo com referência principal as classes de atrasos determinadas pelo ETSI para provimento de QoS em redes GPRS.

Verificou-se também os atrasos e tamanhos dos *buffers*, o que serve de base para um dimensionamento adequado dos mesmos.

Após estas análises verifica-se a riqueza de recursos de coleta de dados para análise disponíveis no modelo/ferramenta de modelagem e conclui-se que o modelo atende ao comportamento esperado de uma rede GPRS real.

# Capítulo VII

## 7. Conclusões

A demanda por diferentes e novos serviços tem aumentado significativamente nos últimos anos, proporcionado um movimento em direção a convergência das redes de comunicação existentes e uma migração para a comutação de pacotes.

O grande desafio das operadoras de rede é manter um nível de qualidade de serviço satisfatório. Em se tratando de redes móveis, o desafio é maior, devido à escassez do espectro de freqüência, diversidade dos requisitos de desempenho dos serviços, mobilidade e qualidade dos enlaces susceptíveis a variações. Para atender a essas exigências, faz-se necessária a alocação de recursos de forma flexível associado à implementação de mecanismos de provimento de QoS.

A rede GPRS surgiu como uma extensão à rede de telefonia móvel celular de segunda geração GSM e possibilitou otimizar os recursos de rádio com o compartilhamento de um mesmo canal físico por vários usuários. Por se tratar de uma tecnologia de comutação de pacotes, os recursos são alocados somente durante o tempo necessário para transferência do pacote, o que é adequado para as aplicações de Internet, por exemplo.

A QoS fim-a-fim para as aplicações exigem a implementação de mecanismos nos elementos da rede GPRS. Contudo, estes mecanismos não estão contidos nas especificações da tecnologia, ficando a sua implementação de livre escolha para as operadoras. Quanto ao gerenciamento de QoS, tais especificações definem apenas as várias classes de serviços e parâmetros de desempenho ligados a essas classes.

Neste contexto, toda pesquisa que contribua para uma definição de quais mecanismos de provimento de QoS são adequados a determinado tipo de rede e perfil de tráfego é bem vinda. Assim, uma “ferramenta” de análise desenvolvida a

partir da modelagem da rede GPRS poderia ser uma contribuição significativa para a comunidade científica e para as operadoras de rede.

Tal ferramenta, denominado GPRSS foi desenvolvido a partir da ferramenta de modelagem OPNET Modeler e submetido a diversas simulações com cenários variados que representassem condições de funcionamento reais das redes GPRS. Os resultados obtidos validaram o GPRSS como um modelo bastante próximo do plano de transmissão de *downlink* de uma rede GPRS, o que representa o ponto critico (gargalo) e de maior atenção por parte das operadoras.

Com o GPRSS é possível simular a transmissão de pacotes entre servidores de aplicações e estações móveis através dos elementos da rede GPRS. Dentre os inúmeros recursos disponíveis no GPRSS, a geração dos pacotes pelos servidores admitem inúmeras distribuições de probabilidade, o que resulta em aplicações bastante aproximadas. Ainda pode-se implementar várias (uma por cenário) disciplinas de escalonamento de pacotes objetivando uma análise comparativa de desempenho baseado em parâmetros de QoS. Além disso, o GPRSS pode ser utilizado como uma ferramenta de dimensionamento de recursos de rede. No âmbito de análise, o GPRSS apresenta seus maiores benefícios e contribuições, uma vez que esse dispõe de inúmeros recursos de coleta de dados, tornando-se uma ferramenta "rica" para o meio científico.

Através da simulação de inúmeros cenários com condições de tráfego e configurações diferentes, obteve-se resultados consistentes sobre o desempenho da rede em função da implementação de disciplinas de escalonamento de pacotes, principalmente quanto a sua influência no atraso fim-a-fim. Pode-se também avaliar o comportamento de cada disciplina frente a classes de precedência distintas e, ainda, verificar qual a influência do esquema de codificação de canal na utilização do enlace sem fio e outras análises. Foram avaliadas cinco disciplinas de escalonamentos de pacotes, das quais, as disciplinas WFQ, WRR e DWRR obtiveram melhores resultados de desempenho com relação ao atraso fim-a-fim e destaque para a disciplina DWRR pela precisão na alocação de largura de banda e simplicidade do algoritmo.

Assim, conclui-se que o modelo proposto atendeu efetivamente o objetivo inicial do trabalho, o qual se refere ao desenvolvimento de um trabalho para ampliar e enriquecer as condições de análise dos mecanismos de provimento de QoS dentro do ambiente de rede GRPS.

## 7.1. Limitações e sugestões para trabalhos futuros

As maiores limitações encontradas durante a realização deste trabalho foram o entendimento do plano de controle da rede GPRS a partir das especificações do ETSI, devido à complexidade e volume de informações, e também o domínio das inúmeras KPs (*Kernel Procedures*) disponíveis na ferramenta de modelagem.

Como sugestão para trabalhos futuros propõe-se a ampliação do GPRSS para um modelo que suporte o plano de controle total da rede GPRS e com transmissões nos dois sentidos (*uplink* e *downlink*). Desta forma seria possível avaliar não só o escalonamento de pacotes mas também o Controle de Admissão de Chamadas.

Outra sugestão seria implementação de disciplinas de escalonamento de pacotes diferentes das cinco propostas neste trabalho no BSC e implementar *Diffserv* no núcleo da rede GPRS.

Poderia também ampliar o número de estações móveis em várias células e avaliar o desempenho da rede frente à mobilidade e procedimentos de *Handover*. O GPRSS utilizou taxas de erro de bit em seus enlaces igual zero, seria interessante avaliar também o desempenho da rede com taxas diferentes.

# Referências Bibliográficas


[1] d'Ávila, César Kyn, **"Telefonia Móvel Celular"**, Apostila volume 1, Cedetec, Inatel, 1998

[2] Dias, Kelvin Lopes e Djamel F. H. Sadok, **"Internet Móvel, Aplicações e QoS",** Centro de Informática, UFPE, 2002

[3] Clark, Martin P., **"Wireless Access Network – Fixed Wireless Access and WLL Networks",** John Wiley and Sons, New York -USA, 2000

[4] Stallings, William, "**Data & Computer Communications",** 6ª ed. New Jersey: Prentice Hall, 2000

[5] d'Ávila, César Kyn, **"Telefonia Móvel Celular"**, Apostila volume 2, Cedetec, Inatel, 1998

[6] Mouly, Michel, Marie B. Paulet, **"The GSM System for Mobile Communication"**, Cell & Sys, France, 1992

[7] Langer, Johan and Gwenn Larsson**, "CDMA2000 – A World view"**, Review, Ericsson, Março de 2001.

[8] Andersson, Christoffer, **"GPRS and 3G Wireless Application",** John Wiley, Canada, 2001

[9] Ekeroth, Lars and Per-Martin Hedstrom, **"GPRS Support Nodes"**, Review, Ericsson, Março de 2000.

[10] UMTS Fórum, em abril de 2003
www.umts-forum.org

[11] Grupo de Desenvolvimento em CDMA, em maio de 2003
http://www.cdg.org/

[12] Generation Partnership Project (3GPP), em maio de 2003
http://www.3gpp.org/ 3rd

[13] Geier, Jim. "**Wireless lans"** ISBN**,** USA, 2002.

[14] Ergen, Mustafa, "**802.11 Tutorial**", University of California Berkeley, USA, 2002

[15] "**Broadband Radio Access Networks (BRAN); HIgh PErformance Radio Local Area Network (HIPERLAN) Type 1; Functional specification**" EN 300 652 V1.2.1, 1998

[16] Jamshid Khun-Jush, Göran Malmgren, Peter Schramm and Johan Torsner. "**HIPERLAN type 2 for broadband wireless communication**", Review nº 2, Ericsson, 2000.

[17] Frodigh, Magnus, Per Johansson and Peter Larson, **"Wireless ad hoc networking – The art of networking without a network",** Review, Ericsson, Abril de 2000.

[18] BluetoothTM wireless technology, em maio de 2003
http://www.bluetooth.com/tech/works.asp

[19] "**HomeRF Specification, rev. 2.01",** The HomeRF Technical Committee, 2002

[20] "**Wireless Short Message Service – SMS"**, IEC – The International Engineering Consortium.

[21] "**MMS Architecture Overview",** WAP-205-MMSArchOverview-20010425-a, Wireless Application Protocol Forum

[22] ETSI EN 301 113 V6.3.1 (2000-11), **"Digital cellular telecommunications system (Phase 2+); General Packet Radio Service (GPRS); Service description; Stage 1"** (GSM 02.60 version 6.3.1 Release 1997)

[23] ETSI TS 144 065 V5.0.0 (2002-06), **"Digital cellular telecommunications system (Phase 2+); Mobile Station (MS) - Serving GPRS Support Node (SGSN); Subnetwork Dependent Convergence Protocol (SNDCP)"**, (3GPP TS 44.065 version 5.0.0 Release 5)

[24] ETSI TS 144 064 V5.1.0 (2002-03), **"Digital cellular telecommunications system (Phase 2+); Mobile Station - Serving GPRS Support Node (MS-SGSN), Logical Link Control (LLC) Layer Specification"**, (3GPP TS 44.064 version 5.1.0 Release 5)

[25] ETSI EN 301 344 V6.7.1 (2000-09), **"Digital cellular telecommunications system (Phase 2+); General Packet Radio Service (GPRS); Service description; Stage 2"** (GSM 03.60 version 6.7.1 Release 1997)

[26] ETSI TS 143 064 V5.1.1 (2003-05), **"Digital cellular telecommunications system (Phase 2+); Overall description of the GPRS radio interface; Stage 2"**, (3GPP TS 43.064 version 5.1.1 Release 5)

[27] ETSI TS 101 349 V8.19.0 (2003-07), **"Digital cellular telecommunications system (Phase 2+); General Packet Radio Service (GPRS); Mobile Station (MS)**

**- Base Station System (BSS) interface; Radio Link Control/Medium Access Control (RLC/MAC) protocol"**, (3GPP TS 04.60 version 8.19.0 Release 1999)

[28] ETSI TS 148 016 V5.2.0 (2003-07), **"Digital cellular telecommunications system (Phase 2+); General Packet Radio Service (GPRS); Base Station System (BSS) - Serving GPRS Support Node (SGSN) Interface; Network Service"**, (3GPP TS 48.016 version 5.2.0 Release 5)

[29] ETSI TS 101 347 V7.10.0 (2002-12), **"Digital cellular telecommunications system (Phase 2+); General Packet Radio Service (GPRS); GPRS Tunnelling Protocol (GTP) across the Gn and Gp Interface"**, (3GPP TS 09.60 version 7.10.0 Release 1998)

[30] Bates, Regis J. (Bud), **"GPRS General Packet Radio Service"**, McGraw-Hill, USA, 2002

[31] Halonen, Timo, Javier Romero and Juan Melero, **"GSM, GPRS and EDGE Performance",** John Wiley & Sons, England, 2002

[32] Stuckmann, Peter, **"The GSM Evolution – Mobile Packet Data Service"**, John Wiley & Sons, England, 2003

[33] Bermond, Antonio Augusto, **"Algoritmos de Roteamento para Suporte de Qualidade de Serviço em Aplicações Multimídia"**, UFES, Vitória, 2002.

[34] Wang, Zheng. "**Internet QoS**: **architecture and mechanisms for quality of service**", Morgan Kaufmann Publishers, New York, 2001.

[35] Tanenbaum, A. S. "**Redes de Computadores**", Ed.Campus, Brasil, 1997.

[36] Soares, Luiz F. Gomes. "**Redes de computadores das LANS, MANS e WANS às redes ATM",** 2.ed., Campus, Rio de Janeiro, 1995.

[37] S. Keshav, "**An Engineering Approach to Computer Networking**", 1997

[38] Semeria, Chuck, **"Supporting Differentiated Service Classes: Queue Scheduling Disciplines"**, Juniper Networks, USA, 2001

[39] 3G TR 22.925 3.1.1 (1999-04), "**3GPP; Technical Specification Group Services and System Aspects; Service aspects; Quality of Service and Network Performance",** (3G TR 22.925 version 3.1.1)

[40] Martins, J. "**QoS em Redes IP. Princípios Básicos, Parâmetros e Mecanismos**.", JSMNet *Networking Reviews*, 1999.

[41] OPNET Modeler (V9.1), em fevereiro de 2004.

http://www.opnet.com/products/modeler/home.html

# Glossário

***Ad-Hoc*** - arquitetura de rede sem fio que permite a associação de qualquer dispositivo
***Attach*** - procedimento de anexação onde a estação móvel se torna conhecida da rede
***Backbone*** - corpo central da rede responsável por ligar redes menores
***Backoff*** - tempo aleatório gerado por temporizadores em CSMA/CD e CDMA/AD
***Best-Effort*** - modo operação de entrega de dados utilizada pelo IP, onde a estrutura de transporte não assegura a entrega de serviço
***Broadcast*** - transmissão para todos nós sem distinção
***Browser*** - software que permite o acesso a sites da Internet
***Buffer*** - espaço de memória em dispositivos destinado a armazenar pacotes de dados, que esperam processamento
***Burst*** - rajada
***Cluster*** - grupo de células
***Default*** - valor usado por não ter outra opção
***Déficit Counter*** - contador de deficit que representa o saldo de cada sub-fila
***Dejitter Buffer*** - Buffer responsável por inserir atrasos aos pacotes de forma a diminuir a variação do mesmo.
***Detach*** - procedimento de desanexação onde a estação móvel se torna desconhecida da rede
***Diffserv*** - serviços diferenciados
***Downlink*** – sentido de descida da BTS para a estação móvel
***E-commerce*** - conceito de comércio eletrônico, onde através de uma estrutura é possível vender produtos ou serviços através da Internet
***E-mail*** - serviço de correspondência eletrônica
***Finish* T*ime*** - número atribuído ao pacote para que este possa ser escalonado.
***Firewalls*** - software ou dispositivo para prevenção de invasões em redes de dados
***Frame Relay*** - tecnologia de rede de pacotes.
***Gateway*** - elemento (hardware e/ou software) responsável pela conversão entre dois padrões diferentes
***Handoff*** - procedimento de controle usado quando uma estação móvel troca de célula
***Handset*** - aparelho de mão
***Hop*** - salto
***Hop-by-hop*** - salto por salto
***Indoor*** - dentro de ambiente fechado
***Intserv*** - serviços integrados
***Jitter*** - variação de atraso na transmissão de pacotes
***Laptop*** - computador portátil
***Link*** - enlace ponto a ponto
***Node Editor*** - editor de nós do software OPNET Modeler
***Overflow*** - estouro de capacidade
***Overlay*** - sobreposição
***Packet Editor*** - editor de pacotes do software OPNET Modeler
***Paging*** - procedimento de busca das estações móveis
***Payload*** - carga útil
***Piconet*** - rede com até 8 dispositivos Bluetooth
***Polling*** - método de acesso baseado em consulta dos nós
***Premium*** - classe de QoS com maior prioridade
***Process Editor*** - editor de processos do software OPNET Modeler
***Project Editor*** - editor de rede do software OPNET Modeler
***Pucturing*** - descarte de bits
***Quantum*** - quantidade ou saldo atribuído a uma sub-fila a cada round
***Roaming*** - procedimento que permite uma estação móvel operar fora da sua área de localização
***Round*** - representa uma rodada completa
***Routing*** - roteamento
***Scatternet*** - conjunto de piconets
***Standard*** - classe de QoS com média prioridade
***Standby*** - à espera
***Store and Forward*** - armazena e posteriormente encaminha a informação
***Stream*** - fluxo
***Switches*** - dispositivos de rede utilizados em comutação
***Tail*** - calda da fila
***Throughput*** - vazão efetiva de informação
***Time slot*** - intervalo de tempo para tráfego de informação
***Time-to-Live*** - campo do pacote Ipv4 que determina quando o pacote deve ser descartado
***Traffic Class*** - classe de tráfego
***Upgrade*** - melhorar
***Uplink*** - sentido de subida da estação móvel para a BTS
***Weight*** - peso

# Anexo A – Códigos dos Processos

## A1 – G_servidor

```
###############################################################################
                        Process Model Report:  G_servidor
###############################################################################

external file set:    oms_dist_support

===============================================================================
                                  Header Block
===============================================================================
#include       <oms_dist_support.h>
#define        SSC_INFINITE_TIME            -1.0
#define        SSC_INICIO                      0
#define        SSC_GERACAO                     1
#define        SSC_FIM                         2
#define        SSC_STRM_TO_LOW                 0
#define        INICIO                       (intrpt_code == SSC_INICIO)
#define        DESABILITADO                 (intrpt_code == SSC_FIM)
#define        FIM                          (intrpt_code == SSC_FIM)
#define        GERACAO_DE_PACOTES           (intrpt_code == SSC_GERACAO)
static void            ss_gera_pacote (void);

===============================================================================
                               State Variable Block
===============================================================================
Objid          \own_id;
char           \format_str [64];
double         \start_time;
double         \stop_time;
OmsT_Dist_Handle\interarrival_dist_ptr;
OmsT_Dist_Handle\pksize_dist_ptr;
Boolean        \generate_unformatted;
Evhandle       \next_pk_evh;
double         \next_intarr_time;
Stathandle     \bits_sent_hndl;
Stathandle     \packets_sent_hndl;
Stathandle     \packet_size_hndl;
Stathandle     \interarrivals_hndl;
int            \tos;
double         \end_destino;
double         \end_fonte;

===============================================================================
                             Temporary Variable Block
===============================================================================
char           interarrival_str [128];
char           size_str [128];
Prg_List*      pk_format_names_lptr;
char*          found_format_str;
int            low, high;
Boolean        format_found;
int            i;
int            intrpt_code;

===============================================================================
                                 Function Block
===============================================================================
static void
ss_gera_pacote (void)
       {
       Packet*        pkptr;
       SimT_Pk_Size   pksize;

       FIN (ss_gera_pacote ());
```

```
	pksize = (SimT_Pk_Size) ceil (oms_dist_outcome (pksize_dist_ptr));

	/* Gera um pacote de formato e tamanho especifico   */
	if (generate_unformatted == OPC_TRUE)
		{
		/* Gera um pacote não formatado      */
		pkptr = op_pk_create (pksize);
		}
	else
		{
		/* Gera um pacote com formato especifico    */
		pkptr = op_pk_create_fmt (format_str);
		/*Preenche o cabeçalho com o TOS e endereços*/
		op_pk_nfd_set (pkptr, "TOS", tos);
		op_pk_nfd_set (pkptr, "Endereço de Destino", end_destino);
		op_pk_nfd_set (pkptr, "Endereço de Fonte", end_fonte);
		op_pk_total_size_set (pkptr, pksize);
		}
	/* Atualiza as estatisticas  */
	op_stat_write (packets_sent_hndl, 1.0);
	op_stat_write (packets_sent_hndl, 0.0);
	op_stat_write (bits_sent_hndl, (double) pksize);
	op_stat_write (bits_sent_hndl, 0.0);
	op_stat_write (packet_size_hndl, (double) pksize);
	op_stat_write (interarrivals_hndl, next_intarr_time);
	/* Envia o pacote para a camada inferior    */
	op_pk_send (pkptr, SSC_STRM_TO_LOW);
	FOUT;
	}

================================================================================
                  Enter Execs for the unforced state "INICIO"
================================================================================
/* Obtem o id do módulo surrounding */
own_id = op_id_self ();

/* Leitura dos atributos     */
op_ima_obj_attr_get (own_id, "Intervalo entre pacotes",    interarrival_str);
op_ima_obj_attr_get (own_id, "Tamanho do pacote",          size_str);
op_ima_obj_attr_get (own_id, "Formato do pacote",          format_str);
op_ima_obj_attr_get (own_id, "Inicio (Start Time)",        &start_time);
op_ima_obj_attr_get (own_id, "Término (Stop Time)",        &stop_time);
op_ima_obj_attr_get (own_id, "Aplicação",                  &tos);
op_ima_obj_attr_get (own_id, "Endereço IP de Destino",     &end_destino);
op_ima_obj_attr_get (own_id, "Endereço IP de Fonte",       &end_fonte);

/* Carrega as PDFs de intervalo entre os pacotes                */
interarrival_dist_ptr = oms_dist_load_from_string (interarrival_str);
pksize_dist_ptr       = oms_dist_load_from_string (size_str);

/* Verifica se há algum pacote formatado para ser gerado */
if (strcmp (format_str, "NONE") == 0)
	{
	/* Gera pacotes não formatados       */
	generate_unformatted = OPC_TRUE;
	}
      else
	{
	/* Gera pacotes formatados    */
	generate_unformatted = OPC_FALSE;

	/* Obtem a lista de pacotes formatados      */
	pk_format_names_lptr = prg_tfile_name_list_get
(PrgC_Tfile_Type_Packet_Format);

	/* Busca o pacote formatado na lista */
	format_found = OPC_FALSE;
	for (i = prg_list_size (pk_format_names_lptr); ((format_found == OPC_FALSE) &&
(i > 0)); i--)
	    {
	    /* Acessa o nome do formato e compara com o requisitado */
	    found_format_str = (char *) prg_list_access (pk_format_names_lptr, i - 1);
		if (strcmp (found_format_str, format_str) == 0)
			format_found = OPC_TRUE;
```

```
            }

    if (format_found == OPC_FALSE)
        {
        /* Gera pacotes não formatados       */
        generate_unformatted = OPC_TRUE;

        /* Mostra o aviso     */
        op_prg_odb_print_major ("Warning from simple packet generator model
        (simple_source):",
                                "The specified packet format", format_str,
                                "is not found. Generating unformatted packets
                               instead.", OPC_NIL);
        }

        /* Destroi a lista dos elementos não necessários    */
        prg_list_free (pk_format_names_lptr);
        prg_mem_free  (pk_format_names_lptr);
        }


/* Verifica se o Start time e Stop Time são válidos */
if ((stop_time <= start_time) && (stop_time != SSC_INFINITE_TIME))
        {
        start_time = SSC_INFINITE_TIME;

        /* Mostra o aviso     */
        op_prg_odb_print_major ("Warning from simple packet generator model
        (simple_source):",
                "Although the generator is not disabled (start time is set to a finite
                value),",
                "a stop time that is not later than the start time is specified.",
                "Disabling the generator.", OPC_NIL);
        }
/*habilita a interrupção*/
if (start_time == SSC_INFINITE_TIME)
        {
        op_intrpt_schedule_self (op_sim_time (), SSC_FIM);
        }
else
        {
        op_intrpt_schedule_self (start_time, SSC_INICIO);

        if (stop_time != SSC_INFINITE_TIME)
                {
                op_intrpt_schedule_self (stop_time, SSC_FIM);
                }
        }

/* Registra as estatisticas utilizadas no processo  */
bits_sent_hndl        = op_stat_reg ("Generator.Traffic Sent (bits/sec)",
OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);
packets_sent_hndl   = op_stat_reg ("Generator.Traffic Sent (packets/sec)",
OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);
packet_size_hndl    = op_stat_reg ("Generator.Packet Size (bits)",
OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);
interarrivals_hndl  = op_stat_reg ("Generator.Packet Interarrival Time (secs)",
OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);
--------------------------------------------------------------------------------

================================================================================
                   Exit Execs for the unforced state "INICIO"
================================================================================
intrpt_code = op_intrpt_code ();
--------------------------------------------------------------------------------

================================================================================
                          transition   INICIO -> GERAÇÃO
================================================================================
name:  tr_0
condition:     INICIO
executive:
color:  RGB101
drawing style: spline
```

```
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                          transition   INICIO -> TERMINO
================================================================================
name:  tr_1
condition:     DESABILITADO
executive:
color: RGB101
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                   Enter Execs for the unforced state "GERAÇÃO"
================================================================================
/* Agenda a chegada dos pacotes      */
next_intarr_time = oms_dist_outcome (interarrival_dist_ptr);

/* Assegura que o intervalo não é negativo */
if (next_intarr_time <0)
       {
       next_intarr_time = 0;
       }
next_pk_evh      = op_intrpt_schedule_self (op_sim_time () + next_intarr_time,
SSC_GERACAO);
--------------------------------------------------------------------------------

================================================================================
                   Exit Execs for the unforced state "GERAÇÃO"
================================================================================
intrpt_code = op_intrpt_code ();
--------------------------------------------------------------------------------
================================================================================
                          transition   GERAÇÃO -> TERMINO
================================================================================
name:  tr_2
condition:     FIM
executive:
color: RGB101
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                          transition   GERAÇÃO -> GERAÇÃO
================================================================================
name:  tr_3
condition:     GERACAO_DE_PACOTES
executive:     ss_gera_pacote();
color: RGB101
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                   Enter Execs for the unforced state "TERMINO"
================================================================================
/* Cancela a geração de pacotes      */
if (op_ev_valid (next_pk_evh) == OPC_TRUE)
       {
       op_ev_cancel (next_pk_evh);
       }
--------------------------------------------------------------------------------

================================================================================
                   Exit Execs for the unforced state "TERMINO"
================================================================================
NONE
--------------------------------------------------------------------------------
```

# A2 – G_enc_FR

```
###############################################################################
                      Process Model Report:  G_enc_FR_GGSN
###############################################################################

===============================================================================
                                  Header Block
===============================================================================
#include <oms_dist_support.h>
#define        SSC_INICIO                  0
#define        SSC_REC_PK_IPV4             1
#define        SSC_ENV_PK_FR               2
#define        INICIO                      (intrpt_code == SSC_INICIO)
#define        REC_PK_IPV4                 (intrpt_code == SSC_REC_PK_IPV4)
#define        ENV_PK_FR                   (intrpt_code == SSC_ENV_PK_FR)
#define        SSC_STRM_TO_LOW             0

/*Variáveis Globais Externas*/
extern int TID_0, TID_1, TID_2, TID_3;
extern double SGSN_0, SGSN_1, SGSN_2, SGSN_3;

===============================================================================
                               State Variable Block
===============================================================================
Stathandle     \pktssec_rcvd_stathandle;
Stathandle     \packets_sent_hndl;
Stathandle     \packet_size_hndl;

===============================================================================
                             Temporary Variable Block
===============================================================================
int    pk_fr_size;
int    alloc_instrms;
int    i;
int    intrpt_code;
Packet* pkipv4;
Packet* pkfr;
int     TID;
double  end_destino;

===============================================================================
                   Enter Execs for the forced state "INICIO"
===============================================================================
/* Registra as estatisticas dos pacotes recebidos*/
pktssec_rcvd_stathandle = op_stat_reg ("Traffic Sink.Traffic Received (packets/sec)",
       OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);
packet_size_hndl = op_stat_reg ("Generator.Packet Size (bits)", OPC_STAT_INDEX_NONE,
OPC_STAT_LOCAL);

/* Registra as estatisticas dos pacotes enviados    */
packets_sent_hndl = op_stat_reg ("Generator.Traffic Sent (packets/sec)",
       OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);

/*Chama o estado IDLE */
op_intrpt_schedule_self (op_sim_time (), SSC_INICIO);
-------------------------------------------------------------------------------

===============================================================================
                    Exit Execs for the forced state "INICIO"
===============================================================================
intrpt_code = op_intrpt_code ();
-------------------------------------------------------------------------------

===============================================================================
                           transition   INICIO -> LIVRE
===============================================================================
name:  INICIO
condition:     INICIO
executive:
color:  RGB000
drawing style: spline
```

```
doc file:      pr_transition

--------------------------------------------------------------------------------

================================================================================
                   Enter Execs for the unforced state "LIVRE"
================================================================================
/*Identifica o tipo de interrupção*/
/*Se for uma interrupção de stream, o pacote sera obtido*/
if (op_intrpt_type () == OPC_INTRPT_STRM)
       {
       /*chama o estado SEG_SNDCP*/
       op_intrpt_schedule_self(op_sim_time (), SSC_REC_PK_IPV4);
       }
--------------------------------------------------------------------------------

================================================================================
                   Exit Execs for the unforced state "LIVRE"
================================================================================
intrpt_code = op_intrpt_code ();
--------------------------------------------------------------------------------

================================================================================
                           transition   LIVRE -> ENC_FR
================================================================================
name:  REC_PK_IPV4
condition:     SSC_REC_PK_IPV4
executive:
color:  RGB000
drawing style: spline
doc file:      pr_transition

--------------------------------------------------------------------------------

================================================================================
                   Enter Execs for the forced state "ENC_FR"
================================================================================
/* determina o número de streams alocadas */
alloc_instrms = op_strm_max_index_in ();

/* loop para cada stream alocado */
for (i = 0; i <= alloc_instrms; i++)
              {

               /* se um stream está conectado ... */
               if (op_strm_connected (OPC_STRM_IN, i) == OPC_TRUE)
                     {

                      /* Se o stream não está vazio obtem-se o pacote e seu tamanho
*/
                      if (op_strm_empty (i) == OPC_FALSE)
                            {
                             /* Obtem o pacote da camada superior*/
                             pkipv4 = op_pk_get(i);

                             /*Obtem o endereço de destino do pacote*/
                             op_pk_nfd_get (pkipv4, "Endereço de Destino",
&end_destino);

                             /* Atualiza estatisticas locais */
                             op_stat_write (pktssec_rcvd_stathandle,    1.0);
                             op_stat_write (pktssec_rcvd_stathandle,    0.0);

                             /* Cria um pacote Frame Realay */
                             pkfr = op_pk_create_fmt ("G_pk_FR");

                             /*Obtem o TID da tabela*/
                             if (end_destino == SGSN_0)
                                    TID = TID_0;
                             if (end_destino == SGSN_1)
                                    TID = TID_1;
                             if (end_destino == SGSN_2)
                                    TID = TID_2;
                             if (end_destino == SGSN_3)
```

```
                                TID = TID_3;

                               /*Encapsula os dados no campo Data*/
                               op_pk_nfd_set (pkfr, "Dados", pkipv4);

                               /*Preenhe o cabeçalho*/
                               op_pk_nfd_set (pkfr, "DLCI_0", TID);

                               /*Obtem o tamanho do pacote FR*/
                               pk_fr_size = op_pk_total_size_get (pkfr);

                               /*Atualiza estatisticas locais*/
                               op_stat_write (packets_sent_hndl, 1.0);
                               op_stat_write (packets_sent_hndl, 0.0);
                               op_stat_write (packet_size_hndl, pk_fr_size);

                               /* Envia o pacote para a camada inferior*/
                               op_pk_send (pkfr, SSC_STRM_TO_LOW);
                              }
                        }
               }
--------------------------------------------------------------------------------

================================================================================
                     Exit Execs for the forced state "ENC_FR"
================================================================================
intrpt_code = op_intrpt_code ();
--------------------------------------------------------------------------------

================================================================================
                           transition   ENC_FR -> LIVRE
================================================================================
name:  ENV_PK_FR
condition:     SSC_ENV_PK_FR
executive:
color:  RGB000
drawing style: spline
doc file:      pr_transition

--------------------------------------------------------------------------------
```

# A3 – G_buf_FIFO

```
###############################################################################
                        Process Model Report:  G_buf_FIFO
###############################################################################

external file set:    link_delay

===============================================================================
                                    Header Block
===============================================================================
#define FILA_VAZIA     (op_q_empty ())
#define SVC_COMPLETO   op_intrpt_type () == OPC_INTRPT_SELF
#define CHEGADA        op_intrpt_type () == OPC_INTRPT_STRM

===============================================================================
                                State Variable Block
===============================================================================
int      \server_busy;
double   \service_rate;
Objid    \own_id;

===============================================================================
                             Temporary Variable Block
===============================================================================
Packet*         pkptr;
int             pk_len;
double          pk_svc_time;
int             insert_ok;

===============================================================================
                    Enter Execs for the forced state "INICIO"
===============================================================================
/* Inicilializa o servidor com IDLE  */
server_busy = 0;

/* Obtem o id do módulo surround*/
own_id = op_id_self ();

/* Obtem o valo da taxa de serviço   */
op_ima_obj_attr_get (own_id, "Taxa de Serviço", &service_rate);
-------------------------------------------------------------------------------

===============================================================================
                     Exit Execs for the forced state "INICIO"
===============================================================================
NONE
-------------------------------------------------------------------------------

===============================================================================
                           transition   INICIO -> CHEGADA
===============================================================================
name:  tr_1
condition:     CHEGADA
executive:
color:  RGB300
drawing style: line
doc file:      pr_transition
-------------------------------------------------------------------------------

===============================================================================
                            transition   INICIO -> LIVRE
===============================================================================
name:  tr_2
condition:     default
executive:
color:  #0000FF
drawing style: spline
doc file:      pr_transition
-------------------------------------------------------------------------------

===============================================================================
```

```
                    Enter Execs for the forced state "CHEGADA"
================================================================================
/* Obtem o pacote do fluxo de entrada       */
pkptr = op_pk_get (op_intrpt_strm ());

/* Insere o pacote na fila    */
if (op_subq_pk_insert (0, pkptr, OPC_QPOS_TAIL) != OPC_QINS_OK)
    {
    op_pk_destroy (pkptr);

    insert_ok = 0;
    }
else{
    insert_ok = 1;
    }
--------------------------------------------------------------------------------

================================================================================
                    Exit Execs for the forced state "CHEGADA"
================================================================================
NONE
--------------------------------------------------------------------------------

================================================================================
                           transition   CHEGADA -> FILA
================================================================================
name:  tr_3
condition:     !server_busy && insert_ok
executive:
color: RGB300
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                          transition   CHEGADA -> LIVRE
================================================================================
name:  tr_4
condition:     default
executive:
color: #0000FF
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                    Enter Execs for the unforced state "LIVRE"
================================================================================
NONE
--------------------------------------------------------------------------------

================================================================================
                    Exit Execs for the unforced state "LIVRE"
================================================================================
NONE
--------------------------------------------------------------------------------

================================================================================
                           transition   LIVRE -> SAÍDA
================================================================================
name:  tr_6
condition:     SVC_COMPLETO
executive:
color: RGB300
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                           transition   LIVRE -> CHEGADA
================================================================================
name:  tr_5
condition:     CHEGADA
executive:
```

```
color:  RGB300
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                    Enter Execs for the forced state "FILA"
================================================================================
/* Acessa o pacote na cabeça da fila */
pkptr = op_subq_pk_access (0, OPC_QPOS_HEAD);

/* determina o tamanho do pacote     */
pk_len = op_pk_total_size_get (pkptr);

/* determina o tempo de processamento       */
pk_svc_time = pk_len / service_rate;

/* Agenda ainterrupção para o processo              */
op_intrpt_schedule_self (op_sim_time () + pk_svc_time, 0);

/* Muda o estado do servidor para busy      */
server_busy = 1;
--------------------------------------------------------------------------------

================================================================================
                     Exit Execs for the forced state "FILA"
================================================================================
NONE
--------------------------------------------------------------------------------

================================================================================
                            transition   FILA -> LIVRE
================================================================================
name:  tr_7
condition:
executive:
color:  #040004
drawing style: line
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                    Enter Execs for the forced state "SAÍDA"
================================================================================
/* Retira o pacote da cabeça da fila */
pkptr = op_subq_pk_remove (0, OPC_QPOS_HEAD);

/* Envia o pacote para camada inferior      */
op_pk_send_forced (pkptr, 0);

/* Muda o estado do servidor para IDLE      */
server_busy = 0;
--------------------------------------------------------------------------------

================================================================================
                    Exit Execs for the forced state "SAÍDA"
================================================================================
NONE
--------------------------------------------------------------------------------

================================================================================
                            transition   SAÍDA -> FILA
================================================================================
name:  tr_8
condition:     !FILA_VAZIA
executive:
color:  RGB300
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                           transition   SAÍDA -> LIVRE
================================================================================
```

```
name:  tr_9
condition:     default
executive:
color: #0000FF
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------
```

# A4 – G_dec_FR

```
###############################################################################
                     Process Model Report:  G_dec_FR_SGSN
###############################################################################

===============================================================================
                                  Header Block
===============================================================================
#include <oms_dist_support.h>
#define        SSC_INICIO                  0
#define        SSC_REC_PK_FR               1
#define        SSC_ENV_PK_IPV4             2
#define        INICIO                      (intrpt_code == SSC_INICIO)
#define        REC_PK_FR                   (intrpt_code == SSC_REC_PK_FR)
#define        ENV_PK_IPV4                 (intrpt_code == SSC_ENV_PK_IPV4)
#define        SSC_STRM_TO_HIGH            0

/* Variáveis Globais*/
int            TLLI_0, TLLI_1, TLLI_2, TLLI_3;
int            NSAPI_0, NSAPI_1, NSAPI_2, NSAPI_3;
int            TID_0, TID_1, TID_2, TID_3;
double         PDP_0, PDP_1, PDP_2, PDP_3;

===============================================================================
                              State Variable Block
===============================================================================
Stathandle     \pktssec_rcvd_stathandle;
Stathandle     \packets_sent_hndl;
Stathandle     \packet_size_hndl;
int    \own_id;
Compcode       \tabela;
Objid  \ith_tabela_objid;

===============================================================================
                           Temporary Variable Block
===============================================================================
double         pk_size;
int            alloc_instrms;
int            i;
int            intrpt_code;
Packet*        pkfr;
Packet*        pkipv4;
int            TID;
int            n;
int            n_atributos;

===============================================================================
                    Enter Execs for the forced state "INICIO"
===============================================================================
/* Registra as estatisticas dos pacotes recebidos*/
pktssec_rcvd_stathandle      = op_stat_reg ("Traffic Sink.Traffic Received
(packets/sec)",       OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);
packet_size_hndl                     = op_stat_reg ("Generator.Packet Size (bits)",
       OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);

/* Registra as estatisticas dos pacotes enviados    */
packets_sent_hndl            = op_stat_reg ("Generator.Traffic Sent (packets/sec)",
              OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);

/* Obtem o id do módulo surrounding */
own_id = op_id_self ();

/* Leitura dos atributos      */
op_ima_obj_attr_get (own_id, "Tabela de Roteamento", &tabela);
n_atributos = op_topo_child_count (tabela, OPC_OBJTYPE_GENERIC);

       for (n = 0; n < n_atributos ; n++)
              {
              ith_tabela_objid = op_topo_child (tabela, OPC_OBJTYPE_GENERIC, n);
```

```
               switch (n)
                      {
                      case 0: op_ima_obj_attr_get (ith_tabela_objid, "TLLI", &TLLI_0);
                             op_ima_obj_attr_get (ith_tabela_objid, "TID", &TID_0);
                             op_ima_obj_attr_get (ith_tabela_objid, "NSAPI",
&NSAPI_0);
                             op_ima_obj_attr_get (ith_tabela_objid, "PDP", &PDP_0);
                      break;
                      case 1: op_ima_obj_attr_get (ith_tabela_objid, "TLLI", &TLLI_1);
                             op_ima_obj_attr_get (ith_tabela_objid, "TID", &TID_1);
                             op_ima_obj_attr_get (ith_tabela_objid, "NSAPI",
&NSAPI_1);
                             op_ima_obj_attr_get (ith_tabela_objid, "PDP", &PDP_1);
                      break;
                      case 2: op_ima_obj_attr_get (ith_tabela_objid, "TLLI", &TLLI_2);
                             op_ima_obj_attr_get (ith_tabela_objid, "TID", &TID_2);
                             op_ima_obj_attr_get (ith_tabela_objid, "NSAPI",
&NSAPI_2);
                             op_ima_obj_attr_get (ith_tabela_objid, "PDP", &PDP_2);
                      break;
                      case 3: op_ima_obj_attr_get (ith_tabela_objid, "TLLI", &TLLI_3);
                             op_ima_obj_attr_get (ith_tabela_objid, "TID", &TID_3);
                             op_ima_obj_attr_get (ith_tabela_objid, "NSAPI",
&NSAPI_3);
                             op_ima_obj_attr_get (ith_tabela_objid, "PDP", &PDP_3);
                      break;
                      }
               }
/*Chama o estado Livre */
op_intrpt_schedule_self (op_sim_time (), SSC_INICIO);
--------------------------------------------------------------------------------

================================================================================
                     Exit Execs for the forced state "INICIO"
================================================================================
intrpt_code = op_intrpt_code ();
--------------------------------------------------------------------------------

================================================================================
                            transition   INICIO -> LIVRE
================================================================================
name:  INICIO
condition:     INICIO
executive:
color:  RGB000
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                     Enter Execs for the unforced state "LIVRE"
================================================================================
/*Identifica o tipo de interrupção*/
/*Se for uma interrupção de stream, o pacote sera obtido*/
if (op_intrpt_type () == OPC_INTRPT_STRM)
       {
       /*chama o estado Dec_FR*/
       op_intrpt_schedule_self(op_sim_time (), SSC_REC_PK_FR);
       }
--------------------------------------------------------------------------------

================================================================================
                     Exit Execs for the unforced state "LIVRE"
================================================================================
intrpt_code = op_intrpt_code ();
--------------------------------------------------------------------------------

================================================================================
                            transition   LIVRE -> DEC_FR
================================================================================
name:  REC_PK_FR
condition:     SSC_REC_PK_FR
executive:
color:  RGB000
```

```
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                   Enter Execs for the forced state "DEC_FR"
================================================================================
/* determina o número de streams alocadas */
alloc_instrms = op_strm_max_index_in ();

/* loop para cada stream alocado */
for (i = 0; i <= alloc_instrms; i++)
              {

               /* se um stream está conectado ... */
               if (op_strm_connected (OPC_STRM_IN, i) == OPC_TRUE)
                     {

                     /*Se o stream não está vazio obtem-se o pacote e seu tamanho */
                      if (op_strm_empty (i) == OPC_FALSE)
                             {
                              /* Obtem o pacote da camada superior*/
                              pkfr = op_pk_get(i);

                              /* Atualiza estatisticas locais */
                              op_stat_write (pktssec_rcvd_stathandle,    1.0);
                              op_stat_write (pktssec_rcvd_stathandle,    0.0);

                              /*Obtem dados o TID do cabeçalho*/
                              op_pk_nfd_get (pkfr, "DLCI_0", &TID);

                              /*Verifica se o pacote é destinado a este nó*/
                              if (TID == TID_0 || TID == TID_1 || TID == TID_2 || TID
== TID_3)
                                     {
                                     /*Desencapsula o pacote IPV4*/
                                     op_pk_nfd_get (pkfr, "Dados", &pkipv4);

                                     /*Destroi o pacote Frame Relay*/
                                     op_pk_destroy (pkfr);

                                     /*Obtem o tamanho do pacote IPV4*/
                                     pk_size = op_pk_total_size_get (pkipv4);

                                     /*Atualiza estatisticas locais*/
                                     op_stat_write (packets_sent_hndl, 1.0);
                                     op_stat_write (packets_sent_hndl, 0.0);
                                     op_stat_write (packet_size_hndl, pk_size);

                                     /* Envia o pacote para a camada superior*/
                                     op_pk_send (pkipv4, SSC_STRM_TO_HIGH);
                                     }
                                     else
                                            {
                                            /*Destroi o pacote Frame Relay*/
                                            op_pk_destroy (pkfr);
                                            }
                              }
                       }
                 }
--------------------------------------------------------------------------------

================================================================================
                    Exit Execs for the forced state "DEC_FR"
================================================================================
/* Determine the code of the interrupt, which is used in evaluating       */
/* state transition conditions.
                     */
intrpt_code = op_intrpt_code ();
--------------------------------------------------------------------------------

================================================================================
                          transition   DEC_FR -> LIVRE
================================================================================
```

```
name:   ENV_PK_IPV4
condition:     SSC_ENV_PK_IPV4
executive:
color:  RGB000
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------
```

# A5 – G_enc_GTP

```
###############################################################################
                        Process Model Report:  G_enc_GTP
###############################################################################

===============================================================================
                                   Header Block
===============================================================================
#include <oms_dist_support.h>
#define        SSC_INICIO                  0
#define        SSC_REC_PK_IPV4             1
#define        SSC_ENV_PK_GTP              2
#define        INICIO                      (intrpt_code == SSC_INICIO)
#define        REC_PK_IPV4                 (intrpt_code == SSC_REC_PK_IPV4)
#define        ENV_PK_GTP                  (intrpt_code == SSC_ENV_PK_GTP)
#define        SSC_STRM_TO_LOW             0

/*Variáveis Globais*/
int     TID_0, TID_1, TID_2, TID_3;
double  PDP_0, PDP_1, PDP_2, PDP_3;
double  SGSN_0, SGSN_1, SGSN_2, SGSN_3;

===============================================================================
                                State Variable Block
===============================================================================
Stathandle     \pktssec_rcvd_stathandle;
Stathandle     \packets_sent_hndl;
Stathandle     \packet_size_hndl;
int            \own_id;
Compcode       \tabela;
Objid          \ith_tabela_objid;

===============================================================================
                             Temporary Variable Block
===============================================================================
double         pk_gtp_size;
double         pk_ipv4_size;
int            alloc_instrms;
int            n;
int            tos;
double         end_destino;
double         end_fonte;
int            intrpt_code;
Packet*        pkipv4;
Packet*        pkgtp;
Packet*        pkipv4_novo;
int            n_atributos;
int            i;
int            TID, TOM;

===============================================================================
                    Enter Execs for the forced state "INICIO"
===============================================================================
/* Registra as estatisticas dos pacotes recebidos*/
pktssec_rcvd_stathandle      = op_stat_reg ("Traffic Sink.Traffic Received
(packets/sec)",       OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);
packet_size_hndl                    = op_stat_reg ("Generator.Packet Size (bits)",
       OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);

/* Registra as estatisticas dos pacotes enviados    */
packets_sent_hndl            = op_stat_reg ("Generator.Traffic Sent (packets/sec)",
              OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);

/* Obtem o id do módulo surrounding */
own_id = op_id_self ();

/* Leitura dos atributos      */
op_ima_obj_attr_get (own_id, "Tabela de Roteamento", &tabela);

n_atributos = op_topo_child_count (tabela, OPC_OBJTYPE_GENERIC);
```

```
       for (n = 0; n < n_atributos ; n++)
              {
              /* Access the i_th dest specification.             */
              ith_tabela_objid = op_topo_child (tabela, OPC_OBJTYPE_GENERIC, n);

              switch (n)
                     {
                     case 0: op_ima_obj_attr_get (ith_tabela_objid, "PDP", &PDP_0);
                            op_ima_obj_attr_get (ith_tabela_objid, "TID", &TID_0);
                            op_ima_obj_attr_get (ith_tabela_objid, "Endereço IP -
SGSN ", &SGSN_0);
                     break;
                     case 1: op_ima_obj_attr_get (ith_tabela_objid, "PDP", &PDP_1);
                            op_ima_obj_attr_get (ith_tabela_objid, "TID", &TID_1);
                            op_ima_obj_attr_get (ith_tabela_objid, "Endereço IP -
SGSN ", &SGSN_1);
                     break;
                     case 2: op_ima_obj_attr_get (ith_tabela_objid, "PDP", &PDP_2);
                            op_ima_obj_attr_get (ith_tabela_objid, "TID", &TID_2);
                            op_ima_obj_attr_get (ith_tabela_objid, "Endereço IP -
SGSN ", &SGSN_2);
                     break;
                     case 3: op_ima_obj_attr_get (ith_tabela_objid, "PDP", &PDP_3);
                            op_ima_obj_attr_get (ith_tabela_objid, "TID", &TID_3);
                            op_ima_obj_attr_get (ith_tabela_objid, "Endereço IP -
SGSN ", &SGSN_3);
                     break;
                     }
              }
/*Chama o estado Livre */
op_intrpt_schedule_self (op_sim_time (), SSC_INICIO);
-------------------------------------------------------------------------------

===============================================================================
                     Exit Execs for the forced state "INICIO"
===============================================================================
/* Determine the code of the interrupt, which is used in evaluating      *//* state
transition conditions.
       */
intrpt_code = op_intrpt_code ();
-------------------------------------------------------------------------------

===============================================================================
                           transition   INICIO -> LIVRE
===============================================================================
name:  INICIO
condition:     INICIO
executive:
color: RGB000
drawing style: spline
doc file:      pr_transition
-------------------------------------------------------------------------------

===============================================================================
                     Enter Execs for the unforced state "LIVRE"
===============================================================================
/*Identifica o tipo de interrupção*/
/*Se for uma interrupção de stream, o pacote sera obtido*/
if (op_intrpt_type () == OPC_INTRPT_STRM)
       {
       /*chama o estado ENC_GTP*/
       op_intrpt_schedule_self(op_sim_time (), SSC_REC_PK_IPV4);
       }
-------------------------------------------------------------------------------

===============================================================================
                     Exit Execs for the unforced state "LIVRE"
===============================================================================
intrpt_code = op_intrpt_code ();
-------------------------------------------------------------------------------

===============================================================================
                           transition   LIVRE -> ENC_GTP
===============================================================================
```

```
name:  REC_PK_IPV4
condition:     SSC_REC_PK_IPV4
executive:
color:  RGB000
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                    Enter Execs for the forced state "ENC_GTP"
================================================================================
/* determina o número de streams alocadas */
alloc_instrms = op_strm_max_index_in ();

/* loop para cada stream alocado */
for (i = 0; i <= alloc_instrms; i++)
               {

                /* se um stream está conectado ... */
                if (op_strm_connected (OPC_STRM_IN, i) == OPC_TRUE)
                      {

                       /* Se o stream não está vazio obtem-se o pacote e seu tamanho
*/
                       if (op_strm_empty (i) == OPC_FALSE)
                             {
                              /* Obtem o pacote da camada superior*/
                              pkipv4 = op_pk_get(i);

                              /*Obtem o tamanho do pacote IPv4*/
                              pk_ipv4_size = op_pk_total_size_get (pkipv4);

                              /*Obtem o dados do cabeçalho*/
                              op_pk_nfd_get(pkipv4,"TOS", &tos);
                              op_pk_nfd_get(pkipv4,"Endereço de Destino",
&end_destino);
                              op_pk_nfd_get(pkipv4,"Endereço de Fonte", &end_fonte);

                              /*Destroy o pacote IPv4*/
                              op_pk_destroy(pkipv4);

                              /* Gera um novo pacote com formato IPv4*/
                              pkipv4_novo = op_pk_create_fmt ("G_pk_IPV4");

                              /*Preenche o cabeçalho */
                              op_pk_nfd_set (pkipv4_novo, "TOS", tos);
                              op_pk_nfd_set (pkipv4_novo, "Endereço de Destino",
end_destino);
                              op_pk_nfd_set (pkipv4_novo, "Endereço de Fonte",
end_fonte);


                              /*define o tamanho do pacote*/
                              op_pk_total_size_set (pkipv4_novo,  pk_ipv4_size);

                              /* Atualiza estatisticas locais */
                              op_stat_write (pktssec_rcvd_stathandle,    1.0);
                              op_stat_write (pktssec_rcvd_stathandle,    0.0);

                              /* Cria um pacote GTP */
                              pkgtp = op_pk_create_fmt ("G_pk_GTP");

                              /*Obtém o TID da tabela de roteamento*/
                              if (end_destino == PDP_0)
                                     TID = TID_0;
                              if (end_destino == PDP_1)
                                     TID = TID_1;
                              if (end_destino == PDP_2)
                                     TID = TID_2;
                              if (end_destino == PDP_3)
                                     TID = TID_3;

                              /*Encapsula os dados no campo Dados*/
                              op_pk_nfd_set (pkgtp, "Dados", pkipv4_novo);
```

```
                            /*Preenche o cabeçalho*/
                               TOM = tos;
                               op_pk_nfd_set (pkgtp, "TID", TID);
                               op_pk_nfd_set (pkgtp, "TOM", TOM);

                               /*Obtem o tamanho do pacote GTP*/
                               pk_gtp_size = op_pk_total_size_get (pkgtp);

                               /*Atualiza estatisticas locais*/
                               op_stat_write (packets_sent_hndl, 1.0);
                               op_stat_write (packets_sent_hndl, 0.0);
                               op_stat_write (packet_size_hndl,  pk_gtp_size);

                               /* Envia o pacote para a camada inferior*/
                               op_pk_send (pkgtp, SSC_STRM_TO_LOW);
                              }
                        }
                }
--------------------------------------------------------------------------------

================================================================================
                     Exit Execs for the forced state "ENC_GTP"
================================================================================
intrpt_code = op_intrpt_code ();
--------------------------------------------------------------------------------

================================================================================
                           transition   ENC_GTP -> LIVRE
================================================================================
name:  ENV_PK_GTP
condition:     SSC_ENV_PK_GTP
executive:
color:  RGB000
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------
```

# A6 – G_enc_TCP_UDP

```
###############################################################################
                    Process Model Report:  G_enc_TCP_UDP
###############################################################################

===============================================================================
                                 Header Block
===============================================================================
#include <oms_dist_support.h>
#define        SSC_INICIO                  0
#define        SSC_REC_PK_GTP              1
#define        SSC_ENV_PK_TCP_UDP          2
#define        INICIO                      (intrpt_code == SSC_INICIO)
#define        REC_PK_GTP                  (intrpt_code == SSC_REC_PK_GTP)
#define        ENV_PK_TCP_UDP              (intrpt_code == SSC_ENV_PK_TCP_UDP)
#define        SSC_STRM_TO_LOW             0

===============================================================================
                            State Variable Block
===============================================================================
Stathandle     \pktssec_rcvd_stathandle;
Stathandle     \packets_sent_hndl;
Stathandle     \packet_size_hndl;

===============================================================================
                          Temporary Variable Block
===============================================================================
int            pk_size;
int            alloc_instrms;
int            i;
int            intrpt_code;
int            TOM;
Packet*        pkgtp;
Packet*        pkstrm;
Packet*        pkipv4;
int            TID;

===============================================================================
                  Enter Execs for the forced state "INICIO"
===============================================================================
/* Registra as estatisticas dos pacotes recebidos*/
pktssec_rcvd_stathandle      = op_stat_reg ("Traffic Sink.Traffic Received
(packets/sec)",       OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);
packet_size_hndl                     = op_stat_reg ("Generator.Packet Size (bits)",
       OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);

/* Registra as estatisticas dos pacotes enviados    */
packets_sent_hndl            = op_stat_reg ("Generator.Traffic Sent (packets/sec)",
               OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);

/*Chama o estado LIVRE */
op_intrpt_schedule_self (op_sim_time (), SSC_INICIO);
-------------------------------------------------------------------------------

===============================================================================
                   Exit Execs for the forced state "INICIO"
===============================================================================
intrpt_code = op_intrpt_code ();
-------------------------------------------------------------------------------

===============================================================================
                           transition   INICIO -> LIVRE
===============================================================================
name:  INICIO
condition:     INICIO
executive:
color:  RGB000
drawing style: spline
doc file:      pr_transition
-------------------------------------------------------------------------------
```

```
================================================================================
                    Enter Execs for the unforced state "LIVRE"
================================================================================
/*Identifica o tipo de interrupção*/
/*Se for uma interrupção de stream, o pacote sera obtido*/
if (op_intrpt_type () == OPC_INTRPT_STRM)
       {
       /*chama o estado ENC_TCP_UDP*/
       op_intrpt_schedule_self(op_sim_time (), SSC_REC_PK_GTP);
       }
--------------------------------------------------------------------------------

================================================================================
                    Exit Execs for the unforced state "LIVRE"
================================================================================
intrpt_code = op_intrpt_code ();
--------------------------------------------------------------------------------

================================================================================
                       transition   LIVRE -> ENC_TCP_UDP
================================================================================
name:  REC_PK_GTP
condition:     SSC_REC_PK_GTP
executive:
color:  RGB000
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                  Enter Execs for the forced state "ENC_TCP_UDP"
================================================================================
/* determina o número de streams alocadas */
alloc_instrms = op_strm_max_index_in ();

/* loop para cada stream alocado */
for (i = 0; i <= alloc_instrms; i++)
              {

              /* se um stream está conectado ... */
              if (op_strm_connected (OPC_STRM_IN, i) == OPC_TRUE)
                     {
                     /* Se o stream não está vazio obtem-se o pacote e seu tamanho
*/

                      if (op_strm_empty (i) == OPC_FALSE)
                            {
                             /* Obtem o pacote da camada superior*/
                             pkgtp = op_pk_get(i);

                             /* Atualiza estatisticas locais */
                             op_stat_write (pktssec_rcvd_stathandle,    1.0);
                             op_stat_write (pktssec_rcvd_stathandle,    0.0);

                             /*Obtem dados do cabeçalho*/
                             op_pk_nfd_get (pkgtp, "TOM", &TOM);
                             op_pk_nfd_get (pkgtp, "TID", &TID);

                             /*Faz o encapsulamento de acordo com a aplicação*/
                             if(TOM == 2 || TOM == 3 || TOM == 4)
                                    {
                                    /* Cria um pacote TCP */
                                    pkstrm = op_pk_create_fmt ("G_pk_TCP");

                                    /*Encapsula o pacote GTP no campo Dados*/
                                    op_pk_nfd_set (pkstrm, "Dados", pkgtp);

                                    /*Preenche o cabeçalho*/
                                    op_pk_nfd_set (pkstrm, "Porta Destino", TID);
                                    }

                             if(TOM == 1)
                                    {
                                    /* Cria um pacote UDP */
```

```
                                        pkstrm = op_pk_create_fmt ("G_pk_UDP");

                                        /*Encapsula o pacote GTP no campo dados*/
                                        op_pk_nfd_set (pkstrm, "Dados", pkgtp);

                                        /*Preenche o cabeçalho*/
                                        op_pk_nfd_set (pkstrm, "Porta Destino", TID);
                                        }

                                    /*Obtem o tamanho do pacote TCP ou UDP*/
                                    pk_size = op_pk_total_size_get (pkstrm);

                                    /*Atualiza estatisticas locais*/
                                    op_stat_write (packets_sent_hndl, 1.0);
                                    op_stat_write (packets_sent_hndl, 0.0);
                                    op_stat_write (packet_size_hndl, pk_size);

                                    /* Envia o pacote para a camada inferior*/
                                    op_pk_send (pkstrm, SSC_STRM_TO_LOW);
                                   }
                            }
                    }
--------------------------------------------------------------------------------

================================================================================
                  Exit Execs for the forced state "ENC_TCP_UDP"
================================================================================
/* Determine the code of the interrupt, which is used in evaluating       */
/* state transition conditions.                                           */
intrpt_code = op_intrpt_code ();
--------------------------------------------------------------------------------

================================================================================
                        transition   ENC_TCP_UDP -> LIVRE
================================================================================
name:  ENV_PK_TCP_UDP
condition:     SSC_ENV_PK_TCP_UDP
executive:
color:  RGB000
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------
```

# A7 – G_enc_IP

```
###############################################################################
                        Process Model Report:  G_enc_IP
###############################################################################

===============================================================================
                                  Header Block
===============================================================================
#include <oms_dist_support.h>
#define        SSC_INICIO                          0
#define        SSC_REC_PK_STRM                     1
#define        SSC_ENV_PK_IPV4                     2
#define        INICIO                        (intrpt_code == SSC_INICIO)
#define        REC_PK_STRM                   (intrpt_code == SSC_REC_PK_STRM)
#define        SEND_PK_IPV4          (intrpt_code == SSC_ENV_PK_IPV4)
#define        SSC_STRM_TO_LOW               0

/*Variáveis Globais Externas*/
extern int TID_0, TID_1, TID_2, TID_3;
extern double SGSN_0, SGSN_1, SGSN_2, SGSN_3;
===============================================================================
                             State Variable Block
===============================================================================
Stathandle     \pktssec_rcvd_stathandle;
Stathandle     \packets_sent_hndl;
Stathandle     \packet_size_hndl;

===============================================================================
                           Temporary Variable Block
===============================================================================
int            pk_ipv4_size;
int            alloc_instrms;
int            i;
int            intrpt_code;
int            TID;
double         end_destino;
Packet*        pkipv4;
Packet*        pkstrm;

===============================================================================
                   Enter Execs for the forced state "INICIO"
===============================================================================
/* Registra as estatisticas dos pacotes recebidos*/
pktssec_rcvd_stathandle      = op_stat_reg ("Traffic Sink.Traffic Received
(packets/sec)",       OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);
packet_size_hndl                    = op_stat_reg ("Generator.Packet Size (bits)",
       OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);

/* Registra as estatisticas dos pacotes enviados    */
packets_sent_hndl            = op_stat_reg ("Generator.Traffic Sent (packets/sec)",
              OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);

/*Chama o estado LIVRE */
op_intrpt_schedule_self (op_sim_time (), SSC_INICIO);
-------------------------------------------------------------------------------

===============================================================================
                    Exit Execs for the forced state "INICIO"
===============================================================================
intrpt_code = op_intrpt_code ();
-------------------------------------------------------------------------------

===============================================================================
                           transition   INICIO -> LIVRE
===============================================================================
name:  INICIO
condition:     INICIO
executive:
color:  RGB000
drawing style: spline
doc file:      pr_transition
```

```
--------------------------------------------------------------------------------

================================================================================
                    Enter Execs for the unforced state "LIVRE"
================================================================================
/*Identifica o tipo de interrupção*/
/*Se for uma interrupção de stream, o pacote sera obtido*/
if (op_intrpt_type () == OPC_INTRPT_STRM)
       {
       /*chama o estado ENC_IP*/
       op_intrpt_schedule_self(op_sim_time (), SSC_REC_PK_STRM);
       }
--------------------------------------------------------------------------------

================================================================================
                    Exit Execs for the unforced state "LIVRE"
================================================================================
intrpt_code = op_intrpt_code ();
--------------------------------------------------------------------------------

================================================================================
                           transition   LIVRE -> ENC_IP
================================================================================
name:  REC_PK_STRM
condition:     SSC_REC_PK_STRM
executive:
color:  RGB000
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                    Enter Execs for the forced state "ENC_IP"
================================================================================
/* determina o número de streams alocadas */
alloc_instrms = op_strm_max_index_in ();

/* loop para cada stream alocado */
for (i = 0; i <= alloc_instrms; i++)
              {

               /* se um stream está conectado ... */
               if (op_strm_connected (OPC_STRM_IN, i) == OPC_TRUE)
                     {

                      /* Se o stream não está vazio obtem-se o pacote e seu tamanho
*/
                      if (op_strm_empty (i) == OPC_FALSE)
                            {
                             /* Obtem o pacote da camada superior*/
                             pkstrm = op_pk_get(i);

                             /*Obtem o TID do pacote*/
                             op_pk_nfd_get (pkipv4, "Porta Destino", &TID);

                             /* Atualiza estatisticas locais */
                             op_stat_write (pktssec_rcvd_stathandle,    1.0);
                             op_stat_write (pktssec_rcvd_stathandle,    0.0);

                             /* Cria um pacote IPV4 */
                             pkipv4 = op_pk_create_fmt ("G_pk_IPV4");

                             /*Obtem o endereço IP SGSN da tabela*/
                             if (TID == TID_0)
                                    end_destino = SGSN_0;
                             if (TID == TID_1)
                                    end_destino = SGSN_1;
                             if (TID == TID_2)
                                    end_destino = SGSN_2;
                             if (TID == TID_3)
                                    end_destino = SGSN_3;

                             /*Encapsula os dados no campo Dados*/
                             op_pk_nfd_set (pkipv4, "Dados", pkstrm);
```

```
                                /*Preenche o cabeçalho*/
                                op_pk_nfd_set (pkipv4, "Endereço de Destino",
end_destino);

                                /*Obtem o tamanho do pacote IPV4*/
                                pk_ipv4_size = op_pk_total_size_get (pkipv4);

                                /*Atualiza estatisticas locais*/
                                op_stat_write (packets_sent_hndl, 1.0);
                                op_stat_write (packets_sent_hndl, 0.0);
                                op_stat_write (packet_size_hndl, pk_ipv4_size);

                                /* Envia o pacote para a camada inferior*/
                                op_pk_send (pkipv4, SSC_STRM_TO_LOW);
                                }
                        }
                }
--------------------------------------------------------------------------------

================================================================================
                     Exit Execs for the forced state "ENC_IP"
================================================================================
intrpt_code = op_intrpt_code ();
--------------------------------------------------------------------------------

================================================================================
                            transition   ENC_IP -> LIVRE
================================================================================
name:  ENV_PK_IP
condition:     SSC_ENV_PK_IPV4
executive:
color:  RGB000
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------
```

# A8 – G_dec_IP

```
###############################################################################
                        Process Model Report:  G_dec_IP
###############################################################################

===============================================================================
                                  Header Block
===============================================================================
#include <oms_dist_support.h>
#define        SSC_INICIO                  0
#define        SSC_REC_PK_IP               1
#define        SSC_ENV_PK_STRM             2
#define        INICIO                      (intrpt_code == SSC_INICIO)
#define        REC_PK_IP                   (intrpt_code == SSC_REC_PK_IP)
#define        ENV_PK_STRM                 (intrpt_code == SSC_ENV_PK_STRM)
#define        SSC_STRM_TO_HIGH            0

===============================================================================
                             State Variable Block
===============================================================================
Stathandle     \pktssec_rcvd_stathandle;
Stathandle     \packets_sent_hndl;
Stathandle     \packet_size_hndl;
int    \own_id;
double  \end_SGSN;

===============================================================================
                            Temporary Variable Block
===============================================================================
int    pk_size;
int    alloc_instrms;
int    i;
int    intrpt_code;
double  end_destino;
Packet* pkipv4;
Packet* pkstrm;

===============================================================================
                   Enter Execs for the forced state "INICIO"
===============================================================================
/* Registra as estatisticas dos pacotes recebidos*/
pktssec_rcvd_stathandle      = op_stat_reg ("Traffic Sink.Traffic Received
(packets/sec)",       OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);
packet_size_hndl                     = op_stat_reg ("Generator.Packet Size (bits)",
       OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);

/* Registra as estatisticas dos pacotes enviados    */
packets_sent_hndl            = op_stat_reg ("Generator.Traffic Sent (packets/sec)",
              OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);

/* Obtem o id do módulo surrounding */
own_id = op_id_self ();

/* Leitura do atributo*/
op_ima_obj_attr_get (own_id, "Endereço IP - SGSN", &end_SGSN);

/*Chama o estado LIVRE */
op_intrpt_schedule_self (op_sim_time (), SSC_INICIO);
-------------------------------------------------------------------------------

===============================================================================
                    Exit Execs for the forced state "INICIO"
===============================================================================
intrpt_code = op_intrpt_code ();
-------------------------------------------------------------------------------

===============================================================================
                           transition   INICIO -> LIVRE
===============================================================================
name:  INICIO
condition:     INICIO
```

```
executive:
color:  RGB000
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                    Enter Execs for the unforced state "LIVRE"
================================================================================
/*Identifica o tipo de interrupção*/
/*Se for uma interrupção de stream, o pacote sera obtido*/
if (op_intrpt_type () == OPC_INTRPT_STRM)
       {
       /*chama o estado DEC_IP*/
       op_intrpt_schedule_self(op_sim_time (), SSC_REC_PK_IP);
       }
--------------------------------------------------------------------------------

================================================================================
                    Exit Execs for the unforced state "LIVRE"
================================================================================
intrpt_code = op_intrpt_code ();
--------------------------------------------------------------------------------

================================================================================
                          transition   LIVRE -> DEC_IP
================================================================================
name:  REC_PK_IP
condition:     SSC_REC_PK_IP
executive:
color:  RGB000
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                    Enter Execs for the forced state "DEC_IP"
================================================================================
/* determina o número de streams alocadas */
alloc_instrms = op_strm_max_index_in ();

/* loop para cada stream alocado */
for (i = 0; i <= alloc_instrms; i++)
              {
               /* se um stream está conectado ... */
               if (op_strm_connected (OPC_STRM_IN, i) == OPC_TRUE)
                     {
                      /* Se o stream não está vazio obtem-se o pacote e seu tamanho
*/
                      if (op_strm_empty (i) == OPC_FALSE)
                            {
                             /* Obtem o pacote da camada superior*/
                             pkipv4 = op_pk_get(i);

                             /* Atualiza estatisticas locais */
                             op_stat_write (pktssec_rcvd_stathandle,    1.0);
                             op_stat_write (pktssec_rcvd_stathandle,    0.0);

                             /*Obtem o endereço de destino*/
                             op_pk_nfd_get (pkipv4, "Endereço de Destino",
&end_destino);

                             /*Verifica se o pacote é destinado a este nó*/
                             if (end_destino == end_SGSN)
                                    {
                                    /*Desencapsula o pacote TCP OU UDP*/
                                    op_pk_nfd_get (pkipv4, "Dados", &pkstrm);

                                    /*Destroi o pacote IPV4*/
                                    op_pk_destroy (pkipv4);

                                    /*Obtem o tamanho do pacote SNDCP*/
                                    pk_size = op_pk_total_size_get (pkstrm);
```

```
                                        /*Atualiza estatisticas locais*/
                                        op_stat_write (packets_sent_hndl, 1.0);
                                        op_stat_write (packets_sent_hndl, 0.0);
                                        op_stat_write (packet_size_hndl, pk_size);

                                        /* Envia o pacote para a camada superior*/
                                        op_pk_send (pkstrm, SSC_STRM_TO_HIGH);
                                        }
                                        else
                                               {
                                               /*Destroi o pacote IPV4*/
                                                op_pk_destroy (pkipv4);
                                               }
                                 }
                          }
                 }
--------------------------------------------------------------------------------

================================================================================
                    Exit Execs for the forced state "DEC_IP"
================================================================================
intrpt_code = op_intrpt_code ();
--------------------------------------------------------------------------------

================================================================================
                           transition   DEC_IP -> LIVRE
================================================================================
name:  ENV_PK_STRM
condition:     SSC_ENV_PK_STRM
executive:
color:  RGB000
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------
```

# A9 – G_dec_TCP_UDP

```
###############################################################################
                    Process Model Report:  G_dec_TCP_UDP
###############################################################################

===============================================================================
                                Header Block
===============================================================================
#include <oms_dist_support.h>
#define        SSC_INICIO                  0
#define        SSC_REC_PK_IPV4             1
#define        SSC_ENV_PK_GTP              2
#define        INICIO                      (intrpt_code == SSC_INICIO)
#define        REC_PK_IPV4                 (intrpt_code == SSC_REC_PK_IPV4)
#define        SEND_PK_GTP                 (intrpt_code == SSC_ENV_PK_GTP)
#define        SSC_STRM_TO_HIGH            0

===============================================================================
                             State Variable Block
===============================================================================
Stathandle     \pktssec_rcvd_stathandle;
Stathandle     \packets_sent_hndl;
Stathandle     \packet_size_hndl;

===============================================================================
                           Temporary Variable Block
===============================================================================
int    pk_gtp_size;
int    alloc_instrms;
int    i;
int    intrpt_code;
Packet* pkgtp;
Packet* pkstrm;
===============================================================================
                  Enter Execs for the forced state "INICIO"
===============================================================================
/* Registra as estatisticas dos pacotes recebidos*/
pktssec_rcvd_stathandle      = op_stat_reg ("Traffic Sink.Traffic Received
(packets/sec)",       OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);
packet_size_hndl                    = op_stat_reg ("Generator.Packet Size (bits)",
       OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);

/* Registra as estatisticas dos pacotes enviados    */
packets_sent_hndl            = op_stat_reg ("Generator.Traffic Sent (packets/sec)",
              OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);

/*Chama o estado LIVRE */
op_intrpt_schedule_self (op_sim_time (), SSC_INICIO);
-------------------------------------------------------------------------------

===============================================================================
                   Exit Execs for the forced state "INICIO"
===============================================================================
intrpt_code = op_intrpt_code ();
-------------------------------------------------------------------------------

===============================================================================
                          transition   INICIO -> LIVRE
===============================================================================
name:  INICIO
condition:     INICIO
executive:
color:  RGB000
drawing style: spline
doc file:      pr_transition
-------------------------------------------------------------------------------


===============================================================================
                  Enter Execs for the unforced state "LIVRE"
===============================================================================
```

```
/*Identifica o tipo de interrupção*/
/*Se for uma interrupção de stream, o pacote sera obtido*/
if (op_intrpt_type () == OPC_INTRPT_STRM)
       {
       /*chama o estado DEC_TCP_UDP*/
       op_intrpt_schedule_self(op_sim_time (), SSC_REC_PK_IPV4);
       }
--------------------------------------------------------------------------------

================================================================================
                    Exit Execs for the unforced state "LIVRE"
================================================================================
intrpt_code = op_intrpt_code ();
--------------------------------------------------------------------------------

================================================================================
                        transition   LIVRE -> DEC_TCP_UDP
================================================================================
name:  REC_PK_IPV4
condition:     SSC_REC_PK_IPV4
executive:
color:  RGB000
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                  Enter Execs for the forced state "DEC_TCP_UDP"
================================================================================
/* determina o número de streams alocadas */
alloc_instrms = op_strm_max_index_in ();

/* loop para cada stream alocado */
for (i = 0; i <= alloc_instrms; i++)
              {

               /* se um stream está conectado ... */
               if (op_strm_connected (OPC_STRM_IN, i) == OPC_TRUE)
                     {

                     /*Se o stream não está vazio obtem-se o pacote e seu tamanho */
                      if (op_strm_empty (i) == OPC_FALSE)
                             {
                              /* Obtem o pacote da camada superior*/
                              pkstrm = op_pk_get(i);

                              /* Atualiza estatisticas locais */
                              op_stat_write (pktssec_rcvd_stathandle,    1.0);
                              op_stat_write (pktssec_rcvd_stathandle,    0.0);

                              /*Desencapsula o pacote GTP*/
                              op_pk_nfd_get (pkstrm, "Dados", &pkgtp);

                              /*Destroi o pacote TCP ou UDP*/
                              op_pk_destroy (pkstrm);

                              /*Obtem o tamanho do pacote GTP*/
                              pk_gtp_size = op_pk_total_size_get (pkgtp);

                              /*Atualiza estatisticas locais*/
                              op_stat_write (packets_sent_hndl, 1.0);
                              op_stat_write (packets_sent_hndl, 0.0);
                              op_stat_write (packet_size_hndl, pk_gtp_size);

                              /* Envia o pacote para a camada superior*/
                              op_pk_send (pkgtp, SSC_STRM_TO_HIGH);
                              }
                       }
                }
--------------------------------------------------------------------------------

================================================================================
                  Exit Execs for the forced state "DEC_TCP_UDP"
================================================================================
```

```
intrpt_code = op_intrpt_code ();
--------------------------------------------------------------------------------

================================================================================
                        transition   DEC_TCP_UDP -> LIVRE
================================================================================
name:  ENV_PK_GTP
condition:     SSC_ENV_PK_GTP
executive:
color:  RGB000
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------
```

# A10 – G_dec_GTP

```
###############################################################################
                        Process Model Report:  G_dec_GTP
###############################################################################

===============================================================================
                                  Header Block
===============================================================================
#include <oms_dist_support.h>
#define        SSC_INICIO                  0
#define        SSC_REC_PK_GTP              1
#define        SSC_ENV_PK_IPV4             2
#define        INICIO                      (intrpt_code == SSC_INICIO)
#define        REC_PK_GTP                  (intrpt_code == SSC_REC_PK_GTP)
#define        ENV_PK_IPV4                 (intrpt_code == SSC_ENV_PK_IPV4)
#define        SSC_STRM_TO_HIGH            0

/*Variáveis Globais Externas*/
extern int TID_0, TID_1, TID_2, TID_3;
extern int TLLI_0, TLLI_1, TLLI_2, TLLI_3;

===============================================================================
                             State Variable Block
===============================================================================
Stathandle     \pktssec_rcvd_stathandle;
Stathandle     \packets_sent_hndl;
Stathandle     \packet_size_hndl;

===============================================================================
                           Temporary Variable Block
===============================================================================
int     pk_ipv4_size;
int     alloc_instrms;
int     i;
int     intrpt_code;
int     QoS;
int     TID, TLLI;
Packet* pkgtp;
Packet* pkipv4;

===============================================================================
                   Enter Execs for the forced state "INICIO"
===============================================================================
/* Registra as estatisticas dos pacotes recebidos*/
pktssec_rcvd_stathandle      = op_stat_reg ("Traffic Sink.Traffic Received
(packets/sec)",       OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);
packet_size_hndl                    = op_stat_reg ("Generator.Packet Size (bits)",
       OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);

/* Registra as estatisticas dos pacotes enviados    */
packets_sent_hndl            = op_stat_reg ("Generator.Traffic Sent (packets/sec)",
              OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);

/*Chama o estado LIVRE */
op_intrpt_schedule_self (op_sim_time (), SSC_INICIO);
-------------------------------------------------------------------------------

===============================================================================
                    Exit Execs for the forced state "INICIO"
===============================================================================
intrpt_code = op_intrpt_code ();
-------------------------------------------------------------------------------

===============================================================================
                           transition   INICIO -> LIVRE
===============================================================================
name:  INICIO
condition:     INICIO
executive:
color:  RGB000
drawing style: spline
```

```
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                   Enter Execs for the unforced state "LIVRE"
================================================================================
/*Identifica o tipo de interrupção*/
/*Se for uma interrupção de stream, o pacote sera obtido*/
if (op_intrpt_type () == OPC_INTRPT_STRM)
       {
       /*chama o estado DEC_GTP*/
       op_intrpt_schedule_self(op_sim_time (), SSC_REC_PK_GTP);
       }
--------------------------------------------------------------------------------

================================================================================
                   Exit Execs for the unforced state "LIVRE"
================================================================================
intrpt_code = op_intrpt_code ();
--------------------------------------------------------------------------------

================================================================================
                         transition   LIVRE -> DEC_GTP
================================================================================
name:  REC_PK_GTP
condition:     SSC_REC_PK_GTP
executive:
color:  RGB000
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                   Enter Execs for the forced state "DEC_GTP"
================================================================================
/* determina o número de streams alocadas */
alloc_instrms = op_strm_max_index_in ();

/* loop para cada stream alocado */
for (i = 0; i <= alloc_instrms; i++)
              {

               /* se um stream está conectado ... */
               if (op_strm_connected (OPC_STRM_IN, i) == OPC_TRUE)
                     {

                      /* Se o stream não está vazio obtem-se o pacote e seu tamanho
*/
                      if (op_strm_empty (i) == OPC_FALSE)
                            {
                             /* Obtem o pacote da camada superior*/
                             pkgtp = op_pk_get(i);

                             /* Atualiza estatisticas locais */
                             op_stat_write (pktssec_rcvd_stathandle,    1.0);
                             op_stat_write (pktssec_rcvd_stathandle,    0.0);

                             /*Desencapsula o pacote IPV4*/
                             op_pk_nfd_get (pkgtp, "Dados", &pkipv4);
                             op_pk_nfd_get (pkgtp, "TID", &TID);

                             /*Destroi o pacote GTP*/
                             op_pk_destroy (pkgtp);

                             /*Obtem o tamanho do pacote IPV4*/
                             pk_ipv4_size = op_pk_total_size_get (pkipv4);

                             /*Obtem o TLLI da Tabela de roteamento*/
                             if (TID == TID_0)
                                    TLLI = TLLI_0;
                             if (TID == TID_1)
                                    TLLI = TLLI_1;
                             if (TID == TID_2)
                                    TLLI = TLLI_2;
```

```
                                if (TID == TID_3)
                                       TLLI = TLLI_3;

                                /*Atualiza estatisticas locais*/
                                op_stat_write (packets_sent_hndl, 1.0);
                                op_stat_write (packets_sent_hndl, 0.0);
                                op_stat_write (packet_size_hndl, pk_ipv4_size);

                                /*Envia o pacote para a camada inferior*/
                                op_pk_send (pkipv4, TLLI);
                                }
                        }
                }
--------------------------------------------------------------------------------

================================================================================
                    Exit Execs for the forced state "DEC_GTP"
================================================================================
intrpt_code = op_intrpt_code ();
--------------------------------------------------------------------------------

================================================================================
                           transition   DEC_GTP -> LIVRE
================================================================================
name:  ENV_PK_IPV4
condition:     SSC_ENV_PK_IPV4
executive:
color:  RGB000
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------
```

# A11 – G_seg_SNDCP

```
###############################################################################
                        Process Model Report:  G_seg_SNDCP
###############################################################################

===============================================================================
                                   Header Block
===============================================================================
#include <oms_dist_support.h>
#include <sar_prim_prototypes.h>
#define        SSC_INICIO                    0
#define        SSC_REC_PK_IPV4               1
#define        SSC_ENV_PK_SNDCP              2
#define        INICIO                        (intrpt_code == SSC_INICIO)
#define        REC_PK_IPV4                   (intrpt_code == SSC_REC_PK_IPV4)
#define        ENV_PK_SNDCP                  (intrpt_code == SSC_ENV_PK_SNDCP)
#define        SSC_STRM_TO_LOW               0
#define        DATA_SIZE_SNDCP               3960

/*Variáveis Globais Externas*/
extern int NSAPI_0, NSAPI_1, NSAPI_2, NSAPI_3;
extern double PDP_0, PDP_1, PDP_2, PDP_3;

===============================================================================
                                State Variable Block
===============================================================================
Sbhandle       \buf_seg;
Stathandle     \pktssec_rcvd_stathandle;
Stathandle     \packets_sent_hndl;
Stathandle     \packet_size_hndl;
int            \own_id;
Compcode       \tabela;
Objid          \ith_tabela_objid;

===============================================================================
                             Temporary Variable Block
===============================================================================
double         pk_sndcp_size;
int            pk_seq_num;
int            alloc_instrms;
int            i;
int            intrpt_code;
double         pk_ipv4_size;
double         pk_seg_sndcp_size;
Packet*        pkipv4;
Packet*        pksndcp;
Packet*        seg_sndcp;
double         end_destino;
int            NSAPI;
===============================================================================
                    Enter Execs for the forced state "INICIO"
===============================================================================
/* Registra as estatisticas dos pacotes recebidos*/
pktssec_rcvd_stathandle       = op_stat_reg ("Traffic Sink.Traffic Received
(packets/sec)",       OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);
packet_size_hndl                     = op_stat_reg ("Generator.Packet Size (bits)",
       OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);

/* Registra as estatisticas dos pacotes enviados    */
packets_sent_hndl             = op_stat_reg ("Generator.Traffic Sent (packets/sec)",
              OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);

/*Chama o estado LIVRE */
op_intrpt_schedule_self (op_sim_time (), SSC_INICIO);
-------------------------------------------------------------------------------

===============================================================================
                     Exit Execs for the forced state "INICIO"
===============================================================================
intrpt_code = op_intrpt_code ();
-------------------------------------------------------------------------------
```

```
================================================================================
                          transition   INICIO -> LIVRE
================================================================================
name:  INICIO
condition:     INICIO
executive:
color:  RGB000
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                   Enter Execs for the unforced state "LIVRE"
================================================================================
/*Identifica o tipo de interrupção*/
/*Se for uma interrupção de stream, o pacote sera obtido*/
if (op_intrpt_type () == OPC_INTRPT_STRM)
       {
       /*chama o estado SEG_SNDCP*/
       op_intrpt_schedule_self(op_sim_time (), SSC_REC_PK_IPV4);
       }
--------------------------------------------------------------------------------

================================================================================
                   Exit Execs for the unforced state "LIVRE"
================================================================================
intrpt_code = op_intrpt_code ();
--------------------------------------------------------------------------------

================================================================================
                        transition   LIVRE -> SEG_SNDCP
================================================================================
name:  REC_PK_IPV4
condition:     SSC_REC_PK_IPV4
executive:
color:  RGB000
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                  Enter Execs for the forced state "SEG_SNDCP"
================================================================================
/* determina o número de streams alocadas */
alloc_instrms = op_strm_max_index_in ();

/* loop para cada stream alocado */
for (i = 0; i <= alloc_instrms; i++)
              {
               /* se um stream está conectado ... */
               if (op_strm_connected (OPC_STRM_IN, i) == OPC_TRUE)
                     {

                     /*Se o stream não está vazio obtem-se o pacote e seu tamanho */
                      if (op_strm_empty (i) == OPC_FALSE)
                             {
                              /* Obtem o pacote da camada superior*/
                              pkipv4 = op_pk_get(i);

                              /* Atualiza estatisticas locais */
                              op_stat_write (pktssec_rcvd_stathandle,    1.0);
                              op_stat_write (pktssec_rcvd_stathandle,    0.0);

                              /*Obtem o endereço de destino do pacote*/
                              op_pk_nfd_get (pkipv4, "Endereço de Destino",
&end_destino);

                              /*Obtem o NSAPI da tabela de roteamento*/
                              if (end_destino == PDP_0)
                                     NSAPI = NSAPI_0;
                              if (end_destino == PDP_1)
                                     NSAPI = NSAPI_1;
                              if (end_destino == PDP_2)
```

```
                                    NSAPI = NSAPI_2;
                                if (end_destino == PDP_3)
                                    NSAPI = NSAPI_3;

                                /*Obtem o tamanho do pacote IPV4*/
                                pk_ipv4_size = op_pk_total_size_get (pkipv4);

                                /*Cria um buffer de segmentação*/
                                buf_seg = op_sar_buf_create (OPC_SAR_BUF_TYPE_SEGMENT,
OPC_SAR_BUF_OPT_DEFAULT);

                                /*Insere pacotes dentro do buffer de segmentação*/
                                op_sar_segbuf_pk_insert (buf_seg, pkipv4,
pk_seq_num++);

                                do{
                                    /* Cria um pacote SNDCP */
                                    pksndcp = op_pk_create_fmt ("G_pk_SNDCP");

                                    /*Remove o segmento do buffer de segmentação*/
                                    seg_sndcp = op_sar_srcbuf_seg_remove (buf_seg,
DATA_SIZE_SNDCP);

                                    pk_seg_sndcp_size = op_pk_total_size_get
(seg_sndcp);

                                    /*Encapsula os dados no campo Dados*/
                                    op_pk_nfd_set (pksndcp, "Dados", seg_sndcp);

                                    /*Preneche o cabeçalho*/
                                    op_pk_nfd_set (pksndcp, "NSAPI", NSAPI);

                                    /*Obtem o tamanho do pacote SNDCP*/
                                    pk_sndcp_size = op_pk_total_size_get (pksndcp);


                                    /*Atualiza estatisticas locais*/
                                    op_stat_write (packets_sent_hndl, 1.0);
                                    op_stat_write (packets_sent_hndl, 0.0);
                                    op_stat_write (packet_size_hndl,
pk_sndcp_size);

                                /*Envia o pacote para a camada inferior*/
                                    op_pk_send (pksndcp, SSC_STRM_TO_LOW);

                                    /*Verifica se ainda há algum segmento a ser
transmitido*/
                                    pk_ipv4_size = pk_ipv4_size -
pk_seg_sndcp_size;

                                }while (pk_ipv4_size > 0);
                            }
                        }
                    }
--------------------------------------------------------------------------------

================================================================================
                Exit Execs for the forced state "SEG_SNDCP"
================================================================================
intrpt_code = op_intrpt_code ();
--------------------------------------------------------------------------------

================================================================================
                        transition   SEG_SNDCP -> LIVRE
================================================================================
name:  ENV_PK_SNDCP
condition:     SSC_ENV_PK_SNDCP
executive:
color:  RGB000
drawing style: spline
doc file:      pr_transition

--------------------------------------------------------------------------------
```

# A12 – G_enc_LLC

```
###############################################################################
                        Process Model Report:  G_enc_LLC
###############################################################################


===============================================================================
                                  Header Block
===============================================================================
#include <oms_dist_support.h>
#define        SSC_INICIO                  0
#define        SSC_REC_PK_SNDCP            1
#define        SSC_ENV_PK_LLC              2
#define        INICIO                      (intrpt_code == SSC_INICIO)
#define        REC_PK_SNDCP                (intrpt_code == SSC_REC_PK_IPV4)
#define        ENV_PK_LLC                  (intrpt_code == SSC_ENV_PK_SNDCP)
#define        SSC_STRM_TO_LOW             0

===============================================================================
                               State Variable Block
===============================================================================
Stathandle     \pktssec_rcvd_stathandle;
Stathandle     \packets_sent_hndl;
Stathandle     \packet_size_hndl;

===============================================================================
                             Temporary Variable Block
===============================================================================
double         pk_llc_size;
int            alloc_instrms;
int            i;
int            intrpt_code;
int            SAPI;
Packet*        pksndcp;
Packet*        pkllc;

===============================================================================
                    Enter Execs for the forced state "INICIO"
===============================================================================
/* Registra as estatisticas dos pacotes recebidos*/
pktssec_rcvd_stathandle      = op_stat_reg ("Traffic Sink.Traffic Received
(packets/sec)",       OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);
packet_size_hndl                    = op_stat_reg ("Generator.Packet Size (bits)",
       OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);

/* Registra as estatisticas dos pacotes enviados    */
packets_sent_hndl            = op_stat_reg ("Generator.Traffic Sent (packets/sec)",
              OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);

/*Chama o estado LIVRE */
op_intrpt_schedule_self (op_sim_time (), SSC_INICIO);
-------------------------------------------------------------------------------

===============================================================================
                     Exit Execs for the forced state "INICIO"
===============================================================================
intrpt_code = op_intrpt_code ();
-------------------------------------------------------------------------------

===============================================================================
                           transition   INICIO -> LIVRE
===============================================================================
name:  INICIO
condition:     INICIO
executive:
color:  RGB000
drawing style: spline
doc file:      pr_transition
-------------------------------------------------------------------------------

===============================================================================
```

```
                    Enter Execs for the unforced state "LIVRE"
================================================================================
/*Identifica o tipo de interrupção*/
/*Se for uma interrupção de stream, o pacote sera obtido*/
if (op_intrpt_type () == OPC_INTRPT_STRM)
       {
       /*chama o estado ENC_LLC*/
       op_intrpt_schedule_self(op_sim_time (), SSC_REC_PK_SNDCP);
       }
--------------------------------------------------------------------------------

================================================================================
                    Exit Execs for the unforced state "LIVRE"
================================================================================
intrpt_code = op_intrpt_code ();
--------------------------------------------------------------------------------

================================================================================
                         transition   LIVRE -> ENC_LLC
================================================================================
name:  REC_PK_SNDCP
condition:     SSC_REC_PK_SNDCP
executive:
color:  RGB000
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                    Enter Execs for the forced state "ENC_LLC"
================================================================================
/* determina o número de streams alocadas */
alloc_instrms = op_strm_max_index_in ();

/* loop para cada stream alocado */
for (i = 0; i <= alloc_instrms; i++)
              {

               /* se um stream está conectado ... */
               if (op_strm_connected (OPC_STRM_IN, i) == OPC_TRUE)
                     {

                     /*Se o stream não está vazio obtem-se o pacote e seu tamanho */
                      if (op_strm_empty (i) == OPC_FALSE)
                            {
                             /* Obtem o pacote da camada superior*/
                             pksndcp = op_pk_get(i);

                             /* Atualiza estatisticas locais */
                             op_stat_write (pktssec_rcvd_stathandle,    1.0);
                             op_stat_write (pktssec_rcvd_stathandle,    0.0);

                             /*Obtém o NSAPI do pacote*/
                             op_pk_nfd_get (pksndcp, "NSAPI", &SAPI);

                             /* Cria um pacote LLC */
                             pkllc = op_pk_create_fmt ("G_pk_LLC");

                             /*Encapsula os dados no campo Dados*/
                             op_pk_nfd_set (pkllc, "Dados", pksndcp);

                             /*Preneche o cabeçalho*/
                             op_pk_nfd_set (pkllc, "SAPI", SAPI);

                             /*Obtem o tamanho do pacote LLC*/
                             pk_llc_size = op_pk_total_size_get (pkllc);

                             /*Atualiza estatisticas locais*/
                             op_stat_write (packets_sent_hndl, 1.0);
                             op_stat_write (packets_sent_hndl, 0.0);
                             op_stat_write (packet_size_hndl, pk_llc_size);

                             /* Envia o pacote para a camada inferior*/
                             op_pk_send_forced (pkllc, SSC_STRM_TO_LOW);
```

```
                        }
                    }
                }
--------------------------------------------------------------------------------

================================================================================
                    Exit Execs for the forced state "ENC_LLC"
================================================================================
intrpt_code = op_intrpt_code ();
--------------------------------------------------------------------------------

================================================================================
                           transition   ENC_LLC -> LIVRE
================================================================================
name:  ENV_PK_LLC
condition:     SSC_ENV_PK_LLC
executive:
color:  RGB000
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------
```

# A13 – G_buf_rx_BVC

```
###############################################################################
                        Process Model Report:  G_buf_rx_BVC
###############################################################################

===============================================================================
                                    Header Block
===============================================================================
#define FILA_VAZIA      (op_q_empty ())
#define SVC_COMPLETO  op_intrpt_type () == OPC_INTRPT_SELF
#define CHEGADA         op_intrpt_type () == OPC_INTRPT_STRM

/*Vaviáveis Globais Externas*/
extern int NSAPI_0, NSAPI_1, NSAPI_2, NSAPI_3;
extern int TLLI_0, TLLI_1, TLLI_2, TLLI_3;

===============================================================================
                                State Variable Block
===============================================================================
int    \server_busy;
double \service_rate;
Objid  \own_id;

===============================================================================
                              Temporary Variable Block
===============================================================================
Packet*         pkptr;
Packet*         pkllc;
int             pk_len;
double          pk_svc_time;
int             insert_ok;
int             TLLI, NSAPI;

===============================================================================
                   Enter Execs for the forced state "INICIO"
===============================================================================
/* Inicilializa o servidor com IDLE  */
server_busy = 0;

/* Obtem o id do módulo surround*/
own_id = op_id_self ();

/* Obtem o valo da taxa de serviço   */
op_ima_obj_attr_get (own_id, "service_rate", &service_rate);
-------------------------------------------------------------------------------

===============================================================================
                    Exit Execs for the forced state "INICIO"
===============================================================================
NONE
-------------------------------------------------------------------------------

===============================================================================
                          transition   INICIO -> CHEGADA
===============================================================================
name:  tr_1
condition:     CHEGADA
executive:
color:  RGB300
drawing style: line
doc file:      pr_transition
-------------------------------------------------------------------------------

===============================================================================
                           transition   INICIO -> LIVRE
===============================================================================
name:  tr_2
condition:     default
executive:
color:  #0000FF
drawing style: spline
```

```
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                    Enter Execs for the forced state "CHEGADA"
================================================================================
/* Obtem o pacote do fluxo de entrada       */
pkptr = op_pk_get (op_intrpt_strm ());

/* Insere o pacote na fila    */
if (op_subq_pk_insert (0, pkptr, OPC_QPOS_TAIL) != OPC_QINS_OK)
    {
    op_pk_destroy (pkptr);

    insert_ok = 0;
    }
else{
    insert_ok = 1;
    }
--------------------------------------------------------------------------------

================================================================================
                    Exit Execs for the forced state "CHEGADA"
================================================================================
NONE
--------------------------------------------------------------------------------

================================================================================
                           transition   CHEGADA -> LIVRE
================================================================================
name:  tr_4
condition:     default
executive:
color: #0000FF
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                            transition   CHEGADA -> FILA
================================================================================
name:  tr_3
condition:     !server_busy && insert_ok
executive:
color: RGB300
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                    Enter Execs for the unforced state "LIVRE"
================================================================================
NONE
--------------------------------------------------------------------------------

================================================================================
                    Exit Execs for the unforced state "LIVRE"
================================================================================
NONE
--------------------------------------------------------------------------------

================================================================================
                           transition   LIVRE -> CHEGADA
================================================================================
name:  tr_5
condition:     CHEGADA
executive:
color: RGB300
drawing style: spline
doc file:      pr_transition

--------------------------------------------------------------------------------

================================================================================
```

```
                          transition   LIVRE -> SAÍDA
================================================================================
name:  tr_6
condition:     SVC_COMPLETO
executive:
color: RGB300
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                    Enter Execs for the forced state "FILA"
================================================================================
/* Acessa o pacote na cabeça da fila */
pkptr = op_subq_pk_access (0, OPC_QPOS_HEAD);

/* determina o tamanho do pacote     */
pk_len = op_pk_total_size_get (pkptr);

/* determina o tempo de processamento       */
pk_svc_time = pk_len / service_rate;

/* Agenda ainterrupção para o processo              */
op_intrpt_schedule_self (op_sim_time () + pk_svc_time, 0);

/* Muda o estado do servidor para busy      */
server_busy = 1;
--------------------------------------------------------------------------------

================================================================================
                     Exit Execs for the forced state "FILA"
================================================================================
NONE
--------------------------------------------------------------------------------

================================================================================
                          transition   FILA -> LIVRE
================================================================================
name:  tr_7
condition:
executive:
color: RGB300
drawing style: line
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                    Enter Execs for the forced state "SAÍDA"
================================================================================
/*Retira o pacote da cabeça da fila  */
pkllc = op_subq_pk_remove (0, OPC_QPOS_HEAD);

/*Retira o endereço de destino*/
op_pk_nfd_get (pkllc, "SAPI", &NSAPI);

if (NSAPI == NSAPI_0)
       TLLI = TLLI_0;
if (NSAPI == NSAPI_1)
       TLLI = TLLI_1;
if (NSAPI == NSAPI_2)
       TLLI = TLLI_2;
if (NSAPI == NSAPI_3)
       TLLI = TLLI_3;

/* Envia o pacote para a camada inferior    */
op_pk_send_forced (pkllc, TLLI);

/* Muda o estado do servidor para IDLE      */
server_busy = 0;
--------------------------------------------------------------------------------

================================================================================
                    Exit Execs for the forced state "SAÍDA"
================================================================================
```

```
NONE
--------------------------------------------------------------------------------

================================================================================
                           transition   SAÍDA -> LIVRE
================================================================================
name:  tr_9
condition:     default
executive:
color:  #0000FF
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                            transition   SAÍDA -> FILA
================================================================================
name:  tr_8
condition:     !FILA_VAZIA
executive:
color:  RGB300
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------
```

# A14 – G_seg_RLC_MAC

```
###############################################################################
                    Process Model Report:  G_seg_RLC_MAC
###############################################################################

===============================================================================
                                  Header Block
===============================================================================
#include <oms_dist_support.h>
#define        SSC_INICIO                  0
#define        SSC_REC_PK_LLC              1
#define        SSC_ENV_PK_RLCMAC           2
#define        INICIO                      (intrpt_code == SSC_INICIO)
#define        REC_PK_LLC                  (intrpt_code == SSC_REC_PK_LLC)
#define        ENV_PK_RLCMAC               (intrpt_code == SSC_ENV_PK_RLCMAC)
#define        SSC_STRM_TO_LOW             0
#define        DATA_SIZE_RLCMAC_CS1        160
#define        DATA_SIZE_RLCMAC_CS2        240
#define        DATA_SIZE_RLCMAC_CS3        288
#define        DATA_SIZE_RLCMAC_CS4        400

/*Variáveis Globais */
int MS_0, MS_1, MS_2, MS_3;
int CS_0, CS_1, CS_2, CS_3;

/*Variáveis Globais Externas*/
extern int NSAPI_0, NSAPI_1, NSAPI_2, NSAPI_3;
extern int TFI_0, TFI_1, TFI_2, TFI_3;

===============================================================================
                             State Variable Block
===============================================================================
Sbhandle       \buf_reseg;
Stathandle     \pktssec_rcvd_stathandle;
Stathandle     \packets_sent_hndl;
Stathandle     \packet_size_hndl;
int            \tipo_cs;
Objid          \own_id;
Compcode       \esquema;
Objid          \ith_esquema_objid;

===============================================================================
                           Temporary Variable Block
===============================================================================
double         pk_llc_size;
int            DATA_SIZE_RLCMAC;
int            pk_seq_num;
int            alloc_instrms;
int            i;
int            intrpt_code;
double         pk_seg_llc_size;
Packet*        pkrlcmac;
Packet*        pkllc;
Packet*        seg_llc;
const char*    pkptr;
int            n;
int            n_atributos;
int            TFI;
int            TLLI;
int            NSAPI;
int            CS;

===============================================================================
                  Enter Execs for the forced state "INICIO"
===============================================================================
/* Registra as estatisticas dos pacotes recebidos*/
pktssec_rcvd_stathandle      = op_stat_reg ("Traffic Sink.Traffic Received
(packets/sec)",       OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);
packet_size_hndl                    = op_stat_reg ("Generator.Packet Size (bits)",
       OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);
```

```
/* Registra as estatisticas dos pacotes enviados    */
packets_sent_hndl            = op_stat_reg ("Generator.Traffic Sent (packets/sec)",
              OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);

/* Obtem o id do módulo surrounding */
own_id = op_id_self ();

/* Leitura dos atributos      */
op_ima_obj_attr_get (own_id, "Esquema de Codificação", &esquema);

n_atributos = op_topo_child_count (esquema, OPC_OBJTYPE_GENERIC);

       for (n = 0; n < n_atributos ; n++)
              {
              ith_esquema_objid = op_topo_child (esquema, OPC_OBJTYPE_GENERIC, n);

              switch (n)
                     {
                     case 0: op_ima_obj_attr_get (ith_esquema_objid, "MS", &MS_0);
                            op_ima_obj_attr_get (ith_esquema_objid, "Esquema de
Codificação", &CS_0);
                     break;
                     case 1: op_ima_obj_attr_get (ith_esquema_objid, "MS", &MS_1);
                            op_ima_obj_attr_get (ith_esquema_objid, "Esquema de
Codificação", &CS_1);
                     break;
                     case 2: op_ima_obj_attr_get (ith_esquema_objid, "MS", &MS_2);
                            op_ima_obj_attr_get (ith_esquema_objid, "Esquema de
Codificação", &CS_2);
                     break;
                     case 3: op_ima_obj_attr_get (ith_esquema_objid, "MS", &MS_3);
                            op_ima_obj_attr_get (ith_esquema_objid, "Esquema de
Codificação", &CS_3);
                     break;
                     }
              }
/*Chama o estado LIVRE */
op_intrpt_schedule_self (op_sim_time (), SSC_INICIO);
--------------------------------------------------------------------------------

================================================================================
                     Exit Execs for the forced state "INICIO"
================================================================================
intrpt_code = op_intrpt_code ();
--------------------------------------------------------------------------------

================================================================================
                           transition   INICIO -> LIVRE
================================================================================
name:  INICIO
condition:     INICIO
executive:
color:  RGB000
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                    Enter Execs for the unforced state "LIVRE"
================================================================================
/*Identifica o tipo de interrupção*/
/*Se for uma interrupção de stream, o pacote sera obtido*/
if (op_intrpt_type () == OPC_INTRPT_STRM)
       {
       /*chama o estado SEG_RLCMAC*/
       op_intrpt_schedule_self(op_sim_time (), SSC_REC_PK_LLC);
       }
--------------------------------------------------------------------------------

================================================================================
                    Exit Execs for the unforced state "LIVRE"
================================================================================
intrpt_code = op_intrpt_code ();
--------------------------------------------------------------------------------
```

```
================================================================================
                        transition   LIVRE -> SEG_RLC_MAC
================================================================================
name:  REC_PK_LLC
condition:     SSC_REC_PK_LLC
executive:
color:  RGB000
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                 Enter Execs for the forced state "SEG_RLC_MAC"
================================================================================
/* determina o número de streams alocadas */
alloc_instrms = op_strm_max_index_in ();

/* loop para cada stream alocado */
for (i = 0; i <= alloc_instrms; i++)
               {
                /* se um stream está conectado ... */
                if (op_strm_connected (OPC_STRM_IN, i) == OPC_TRUE)
                      {

                     /*Se o stream não está vazio obtem-se o pacote e seu tamanho */
                      if (op_strm_empty (i) == OPC_FALSE)
                             {
                             /* Obtem o pacote da camada superior*/
                              pkllc = op_pk_get(i);

                              /* Atualiza estatisticas locais */
                              op_stat_write (pktssec_rcvd_stathandle,    1.0);
                              op_stat_write (pktssec_rcvd_stathandle,    0.0);

                              /*Obtem o tamanho do pacote LLC*/
                              pk_llc_size = op_pk_total_size_get (pkllc);

                              /*Obtem o NSAPI do pacote*/
                              op_pk_nfd_get (pkllc, "SAPI", &NSAPI);

                              /*Obtem o TFI da tabela de roteamento* e o esquema de
codificação*/
                              if (NSAPI == NSAPI_0)
                                     {
                                     TFI = TFI_0;
                                     CS = CS_0;
                                     }
                              if (NSAPI == NSAPI_1)
                                     {
                                     TFI = TFI_1;
                                     CS = CS_1;
                                     }
                              if (NSAPI == NSAPI_2)
                                     {
                                     TFI = TFI_2;
                                     CS = CS_2;
                                     }
                              if (NSAPI == NSAPI_3)
                                     {
                                     TFI = TFI_3;
                                     CS = CS_3;
                                     }

                              switch (CS)
                                            {
                                            case 1 : DATA_SIZE_RLCMAC =
DATA_SIZE_RLCMAC_CS1;
                                                          pkptr =
"G_pk_RLCMAC_CS1";
                                            break;
                                            case 2 : DATA_SIZE_RLCMAC =
DATA_SIZE_RLCMAC_CS2;
```

```
                                                        pkptr =
"G_pk_RLCMAC_CS2";
                                        break;
                                        case 3 : DATA_SIZE_RLCMAC =
DATA_SIZE_RLCMAC_CS3;
                                                        pkptr =
"G_pk_RLCMAC_CS3";
                                        break;
                                        case 4 : DATA_SIZE_RLCMAC =
DATA_SIZE_RLCMAC_CS4;
                                                        pkptr =
"G_pk_RLCMAC_CS4";
                                        break;
                                        }

                                /*Cria um buffer de segmentação*/
                                buf_reseg = op_sar_buf_create
(OPC_SAR_BUF_TYPE_SEGMENT, OPC_SAR_BUF_OPT_DEFAULT);
                                /*Insere pacotes dentro do buffer de segmentação*/
                                op_sar_segbuf_pk_insert (buf_reseg, pkllc,
pk_seq_num++);

                                do{
                                        /* Cria um pacote RLCMAC */
                                        pkrlcmac = op_pk_create_fmt (pkptr);

                                        /*Remove o segmento do buffer de segmentação*/
                                        seg_llc = op_sar_srcbuf_seg_remove (buf_reseg,
DATA_SIZE_RLCMAC);

                                        /*Obtem o tamanho do pacote segmento llc*/
                                        pk_seg_llc_size = op_pk_total_size_get
(seg_llc);

                                        /*Encapsula os dados */
                                        op_pk_nfd_set (pkrlcmac, "Dados", seg_llc);
                                        op_pk_nfd_set (pkrlcmac, "TFI", TFI);

                                        /*Atualiza estatisticas locais*/
                                        op_stat_write (packets_sent_hndl, 1.0);
                                        op_stat_write (packets_sent_hndl, 0.0);
                                        op_stat_write (packet_size_hndl,
pk_seg_llc_size);
                                    /*Envia o pacote para a camada inferior*/
                                        op_pk_send (pkrlcmac, SSC_STRM_TO_LOW);

                                        /*Verifica se ainda há algum segmento a ser
transmitido*/
                                        pk_llc_size = pk_llc_size - pk_seg_llc_size;

                                  }while (pk_llc_size > 0);

                                }
                        }
                }
--------------------------------------------------------------------------------

================================================================================
                  Exit Execs for the forced state "SEG_RLC_MAC"
================================================================================
intrpt_code = op_intrpt_code ();
--------------------------------------------------------------------------------

================================================================================
                        transition   SEG_RLC_MAC -> LIVRE
================================================================================
name:  ENV_PK_RLCMAC
condition:     SSC_ENV_PK_RLCMAC
executive:
color:  RGB000
drawing style: spline
doc file:      pr_transition

--------------------------------------------------------------------------------
```

# A15 – G_buf_ms_FIFO

```
###############################################################################
                     Process Model Report:  G_buf_ms_FIFO
###############################################################################

===============================================================================
                                  Header Block
===============================================================================
#define FILA_VAZIA      (op_q_empty ())
#define SVC_COMPLETO    op_intrpt_type () == OPC_INTRPT_SELF
#define CHEGADA         op_intrpt_type () == OPC_INTRPT_STRM

===============================================================================
                               State Variable Block
===============================================================================
int            \server_busy;
Double         \service_rate;
Objid          \own_id;
Stathandle     \pktssec_rcvd_stathandle;
Stathandle     \packet_size_hndl;
Stathandle     \packets_sent_hndl;
int            \tipo;
Compcode       \tabela;
Objid          \ith_tabela_objid;
int            \TFI_MS;

===============================================================================
                            Temporary Variable Block
===============================================================================
Packet*         pkptr;
int             pk_len;
double          pk_svc_time;
int             insert_ok;
int                          TFI;

===============================================================================
                   Enter Execs for the forced state "INICIO"
===============================================================================
/* Registra as estatisticas dos pacotes recebidos*/
pktssec_rcvd_stathandle      = op_stat_reg ("Traffic Sink.Traffic Received
(packets/sec)",       OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);
packet_size_hndl                    = op_stat_reg ("Generator.Packet Size (bits)",
       OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);

/* Registra as estatisticas dos pacotes enviados    */
packets_sent_hndl            = op_stat_reg ("Generator.Traffic Sent (packets/sec)",
              OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);

/* Inicilializa o servidor com IDLE  */
server_busy = 0;

/* Obtem o id do modulo surround*/
own_id = op_id_self ();

/* Leitura dos atributos     */
op_ima_obj_attr_get (own_id, "service_rate", &service_rate);
op_ima_obj_attr_get (own_id, "Aplicação", &tipo);
op_ima_obj_attr_get (own_id, "Tabela MS", &tabela);

ith_tabela_objid = op_topo_child (tabela, OPC_OBJTYPE_GENERIC, 0);

op_ima_obj_attr_get (ith_tabela_objid, "TFI", &TFI_MS);
-------------------------------------------------------------------------------

===============================================================================
                    Exit Execs for the forced state "INICIO"
===============================================================================
NONE
-------------------------------------------------------------------------------
```

```
================================================================================
                          transition   INICIO -> CHEGADA
================================================================================
name:  tr_1
condition:     CHEGADA
executive:
color:  RGB300
drawing style: line
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                           transition   INICIO -> LIVRE
================================================================================
name:  tr_2
condition:     default
executive:
color:  #0000FF
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                    Enter Execs for the forced state "CHEGADA"
================================================================================
/* Obtem o pacote da camada inferior */
pkptr = op_pk_get (op_intrpt_strm());

/*Atualiza as estatisticas*/
op_stat_write (pktssec_rcvd_stathandle,     1.0);
op_stat_write (pktssec_rcvd_stathandle,     0.0);

/*Verifica se o pacote é destinado a este nó*/
op_pk_nfd_get (pkptr, "TFI", &TFI);

if (TFI == TFI_MS)
       {
       /*Insere o pacote na fila*/
       if (op_subq_pk_insert (0, pkptr, OPC_QPOS_TAIL) != OPC_QINS_OK)
              {
              op_pk_destroy (pkptr);

              insert_ok = 0;
              }
              else
              {
              insert_ok = 1;
              }
        }
       else
              {
              /*Destroi o pacote*/
              op_pk_destroy (pkptr);
              }
--------------------------------------------------------------------------------

================================================================================
                    Exit Execs for the forced state "CHEGADA"
================================================================================
NONE
--------------------------------------------------------------------------------

================================================================================
                          transition   CHEGADA -> LIVRE
================================================================================
name:  tr_4
condition:     default
executive:
color:  #0000FF
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
```

```
                          transition   CHEGADA -> FILA
================================================================================
name:  tr_3
condition:     !server_busy && insert_ok
executive:
color:  RGB300
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                    Enter Execs for the unforced state "LIVRE"
================================================================================
NONE
--------------------------------------------------------------------------------

================================================================================
                    Exit Execs for the unforced state "LIVRE"
================================================================================
NONE
--------------------------------------------------------------------------------

================================================================================
                          transition   LIVRE -> CHEGADA
================================================================================
name:  tr_5
condition:     CHEGADA
executive:
color:  RGB300
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                          transition   LIVRE -> SAÍDA
================================================================================
name:  tr_6
condition:     SVC_COMPLETO
executive:
color:  RGB300
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                     Enter Execs for the forced state "FILA"
================================================================================
if (op_subq_empty (0) == OPC_FALSE)
       {
       /* Acessa o pacote na cabeça da fila */
       pkptr = op_subq_pk_access (0, OPC_QPOS_HEAD);

       /* determina o tamanho do pacote     */
       pk_len = op_pk_total_size_get (pkptr);

       /* determina o tempo de processamento       */
       pk_svc_time = pk_len / service_rate;

       /* Agenda ainterrupção para o processo             */
       op_intrpt_schedule_self (op_sim_time () + pk_svc_time, 0);

       /* Muda o estado do servidor para busy      */
       server_busy = 1;
       }
--------------------------------------------------------------------------------

================================================================================
                      Exit Execs for the forced state "FILA"
================================================================================
NONE
--------------------------------------------------------------------------------

================================================================================
                          transition   FILA -> LIVRE
```

```
================================================================================
name:  tr_7
condition:
executive:
color: RGB300
drawing style: line
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                    Enter Execs for the forced state "SAÍDA"
================================================================================
/*Remove o pacote da fila*/
pkptr = op_subq_pk_remove (0, OPC_QPOS_HEAD);

/*Atualiza as estatisticas*/
op_stat_write (packets_sent_hndl, 1.0);
op_stat_write (packets_sent_hndl, 0.0);

/*Envia o pacote a camada superior*/
op_pk_send (pkptr, 0);

/* Muda o estado do servidor para IDLE */
server_busy = 0;
--------------------------------------------------------------------------------

================================================================================
                    Exit Execs for the forced state "SAÍDA"
================================================================================
NONE
--------------------------------------------------------------------------------

================================================================================
                           transition   SAÍDA -> LIVRE
================================================================================
name:  tr_9
condition:     default
executive:
color: #0000FF
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                            transition   SAÍDA -> FILA
================================================================================
name:  tr_8
condition:     !FILA_VAZIA
executive:
color: RGB300
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------
```

# A16 – G_mon_LLC

```
###############################################################################
                        Process Model Report:  G_mon_LLC
###############################################################################

===============================================================================
                                  Header Block
===============================================================================
#include <oms_dist_support.h>
#include <sar_prim_prototypes.h>
#define        SSC_INICIO                  0
#define        SSC_REC_PK_RLCMAC           1
#define        SSC_ENV_PK_LLC              2
#define        INICIO                      (intrpt_code == SSC_INICIO)
#define        REC_PK_RLCMAC               (intrpt_code == SSC_REC_PK_RLCMAC)
#define        ENV_PK_LLC                  (intrpt_code == SSC_ENV_PK_LLC)
#define        SSC_STRM_TO_HIGH            0
Sbhandle       buf_mon;

===============================================================================
                               State Variable Block
===============================================================================
Stathandle     \pktssec_rcvd_stathandle;
Stathandle     \packets_sent_hndl;
Stathandle     \packet_size_hndl;

===============================================================================
                             Temporary Variable Block
===============================================================================
int            alloc_instrms;
int            i;
int            intrpt_code;
packet*        pkllc;
packet*        seg_llc;
packet*        pkrlcmac;
double         pk_llc_size;

===============================================================================
                    Enter Execs for the forced state "INICIO"
===============================================================================
/* Registra as estatisticas dos pacotes recebidos*/
pktssec_rcvd_stathandle      = op_stat_reg ("Traffic Sink.Traffic Received
(packets/sec)",       OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);
packet_size_hndl                      = op_stat_reg ("Generator.Packet Size (bits)",
       OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);

/* Registra as estatisticas dos pacotes enviados    */
packets_sent_hndl             = op_stat_reg ("Generator.Traffic Sent (packets/sec)",
               OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);

/*Chama o estado IDLE */
op_intrpt_schedule_self (op_sim_time (), SSC_INICIO);

/*Cria um buffer de monatem de pacotes*/
buf_mon = op_sar_buf_create (OPC_SAR_BUF_TYPE_REASSEMBLY, OPC_SAR_BUF_OPT_DEFAULT);

-------------------------------------------------------------------------------

===============================================================================
                     Exit Execs for the forced state "INICIO"
===============================================================================
intrpt_code = op_intrpt_code ();
-------------------------------------------------------------------------------

===============================================================================
                           transition   INICIO -> LIVRE
===============================================================================
name:  INICIO
condition:     INICIO
executive:
color:  RGB000
```

```
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                   Enter Execs for the unforced state "LIVRE"
================================================================================
/*Identifica o tipo de interrupção*/
/*Se for uma interrupção de stream, o pacote sera obtido*/
if (op_intrpt_type () == OPC_INTRPT_STRM)
       {
       /*chama o estado SEG_SNDCP*/
       op_intrpt_schedule_self(op_sim_time (), SSC_REC_PK_RLCMAC);
       }
--------------------------------------------------------------------------------

================================================================================
                   Exit Execs for the unforced state "LIVRE"
================================================================================
intrpt_code = op_intrpt_code ();
--------------------------------------------------------------------------------

================================================================================
                         transition   LIVRE -> MON_LLC
================================================================================
name:  REC_PK_RLCMAC
condition:     SSC_REC_PK_RLCMAC
executive:
color:  RGB000
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                   Enter Execs for the forced state "MON_LLC"
================================================================================
/* determina o número de streams alocadas */
alloc_instrms = op_strm_max_index_in ();

/* loop para cada stream alocado */
for (i = 0; i <= alloc_instrms; i++)
              {

               /* se um stream está conectado ... */
               if (op_strm_connected (OPC_STRM_IN, i) == OPC_TRUE)
                     {

                     /*Se o stream não está vazio obtem-se o pacote e seu tamanho */
                      if (op_strm_empty (i) == OPC_FALSE)
                             {
                             /* Obtem o pacote da camada superior*/
                             pkrlcmac = op_pk_get (i);

                             /* Atualiza estatisticas locais */
                             op_stat_write (pktssec_rcvd_stathandle,     1.0);
                             op_stat_write (pktssec_rcvd_stathandle,     0.0);

                             /*Desencapsula o segmento LLC*/
                             op_pk_nfd_get (pkrlcmac, "Dados", &seg_llc);

                             /*Monta pacote LLC original*/
                             if (op_sar_pk_is_segment (seg_llc) == OPC_TRUE)
                                    {
                                    op_sar_rsmbuf_seg_insert (buf_mon, seg_llc);

                                    if ((pkllc = op_sar_rsmbuf_pk_remove (buf_mon))
!= OPC_NIL)
                                           {
                                           /*Obtem op tamnhao do pacote*/
                                           pk_llc_size = op_pk_total_size_get
(pkllc);

                                           /*Atualiza as estatisticas*/
                                           op_stat_write (packets_sent_hndl, 1.0);
```

```
                                        op_stat_write (packets_sent_hndl, 0.0);

                                        op_stat_write (packet_size_hndl,
pk_llc_size);


                                        /* Envia o pacote para a camada
superior*/
                                        op_pk_send (pkllc, SSC_STRM_TO_HIGH);
                                        }
                                    }
                                /*Destroi o pacote RLC_MAC*/
                                op_pk_destroy (pkrlcmac);
                                }
                            }
                    }
--------------------------------------------------------------------------------

================================================================================
                    Exit Execs for the forced state "MON_LLC"
================================================================================
intrpt_code = op_intrpt_code ();
--------------------------------------------------------------------------------

================================================================================
                           transition   MON_LLC -> LIVRE
================================================================================
name:  ENV_PK_LLC
condition:     SSC_ENV_PK_LLC
executive:
color:  RGB000
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------
```

# A17 – G_dec_SNDCP

```
###############################################################################
                        Process Model Report:  G_dec_SNDCP
###############################################################################

===============================================================================
                                  Header Block
===============================================================================
#include <oms_dist_support.h>
#define        SSC_INICIO                  0
#define        SSC_REC_PK_LLC              1
#define        SSC_ENV_PK_SNDCP            2
#define        INICIO                      (intrpt_code == SSC_INICIO)
#define        REC_PK_LLC                  (intrpt_code == SSC_REC_PK_LLC)
#define        ENV_PK_SNDCP                (intrpt_code == SSC_ENV_PK_SNDCP)
#define        SSC_STRM_TO_HIGH            0

===============================================================================
                               State Variable Block
===============================================================================
Stathandle     \pktssec_rcvd_stathandle;
Stathandle     \packets_sent_hndl;
Stathandle     \packet_size_hndl;

===============================================================================
                             Temporary Variable Block
===============================================================================
int pk_sndcp_size;
int alloc_instrms;
int i;
int intrpt_code;
packet* pkllc;
packet* pksndcp;

===============================================================================
                    Enter Execs for the forced state "INICIO"
===============================================================================
/* Registra as estatisticas dos pacotes recebidos*/
pktssec_rcvd_stathandle      = op_stat_reg ("Traffic Sink.Traffic Received
(packets/sec)",       OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);
packet_size_hndl                     = op_stat_reg ("Generator.Packet Size (bits)",
       OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);

/* Registra as estatisticas dos pacotes enviados    */
packets_sent_hndl            = op_stat_reg ("Generator.Traffic Sent (packets/sec)",
              OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);

/*Chama o estado IDLE */
op_intrpt_schedule_self (op_sim_time (), SSC_INICIO);
-------------------------------------------------------------------------------

===============================================================================
                     Exit Execs for the forced state "INICIO"
===============================================================================
/* Determine the code of the interrupt, which is used in evaluating      *//* state
transition conditions.
       */
intrpt_code = op_intrpt_code ();
-------------------------------------------------------------------------------

===============================================================================
                           transition   INICIO -> LIVRE
===============================================================================
name:  INICIO
condition:     INICIO
executive:
color: RGB000
drawing style: spline
doc file:      pr_transition
-------------------------------------------------------------------------------
```

```
================================================================================
                   Enter Execs for the unforced state "LIVRE"
================================================================================
/*Identifica o tipo de interrupção*/
/*Se for uma interrupção de stream, o pacote sera obtido*/
if (op_intrpt_type () == OPC_INTRPT_STRM)
       {
       /*chama o estado SEG_SNDCP*/
       op_intrpt_schedule_self(op_sim_time (), SSC_REC_PK_LLC);
       }
--------------------------------------------------------------------------------

================================================================================
                   Exit Execs for the unforced state "LIVRE"
================================================================================
/* Determine the code of the interrupt, which is used in evaluating       */
/* state transition conditions.
                     */
intrpt_code = op_intrpt_code ();
--------------------------------------------------------------------------------

================================================================================
                         transition   LIVRE -> DEC_SNDCP
================================================================================
name:  REC_PK_LLC
condition:     SSC_REC_PK_LLC
executive:
color:  RGB000
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                  Enter Execs for the forced state "DEC_SNDCP"
================================================================================
/* determina o número de streams alocadas */
alloc_instrms = op_strm_max_index_in ();

/* loop para cada stream alocado */
for (i = 0; i <= alloc_instrms; i++)
              {

               /* se um stream está conectado ... */
               if (op_strm_connected (OPC_STRM_IN, i) == OPC_TRUE)
                     {

                     /*Se o stream não está vazio obtem-se o pacote e seu tamanho */
                      if (op_strm_empty (i) == OPC_FALSE)
                             {
                              /* Obtem o pacote da camada superior*/
                              pkllc = op_pk_get(i);

                              /* Atualiza estatisticas locais */
                              op_stat_write (pktssec_rcvd_stathandle,    1.0);
                              op_stat_write (pktssec_rcvd_stathandle,    0.0);

                              /*Desencapsula o pacote SNDCP*/
                              op_pk_nfd_get (pkllc, "Dados", &pksndcp);

                              /*Destroi o pacote LLC*/
                              op_pk_destroy (pkllc);

                              /*Obtem o tamanho do pacote SNDCP*/
                              pk_sndcp_size = op_pk_total_size_get (pksndcp);

                              /*Atualiza estatisticas locais*/
                              op_stat_write (packets_sent_hndl, 1.0);
                              op_stat_write (packets_sent_hndl, 0.0);
                              op_stat_write (packet_size_hndl, pk_sndcp_size);

                              /* Envia o pacote para a camada superior*/
                              op_pk_send (pksndcp, SSC_STRM_TO_HIGH);
                              }
                      }
```

```
                }
--------------------------------------------------------------------------------

================================================================================
                  Exit Execs for the forced state "DEC_SNDCP"
================================================================================
/* Determine the code of the interrupt, which is used in evaluating       */
/* state transition conditions.
                      */
intrpt_code = op_intrpt_code ();
--------------------------------------------------------------------------------

================================================================================
                         transition   DEC_SNDCP -> LIVRE
================================================================================
name:  ENV_PK_SNDCP
condition:     SSC_ENV_PK_SNDCP
executive:
color:  RGB000
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------
```

# A18 – G_mon_IPV4

```
###############################################################################
                        Process Model Report:  G_mon_IPV4
###############################################################################

===============================================================================
                                   Header Block
===============================================================================
#include <oms_dist_support.h>
#include <sar_prim_prototypes.h>
#define        SSC_INICIO                  0
#define        SSC_REC_PK_SNDCP            1
#define        SSC_ENV_PK_IPV4             2
#define        INICIO                      (intrpt_code == SSC_INICIO)
#define        REC_PK_SNDCP                (intrpt_code == SSC_REC_PK_SNDCP)
#define        ENV_PK_IPV4                 (intrpt_code == SSC_ENV_PK_IPV4)
#define        SSC_STRM_TO_HIGH            0
Sbhandle       buf_mon;

===============================================================================
                                State Variable Block
===============================================================================
Stathandle     \pktssec_rcvd_stathandle;
Stathandle     \packets_sent_hndl;
Stathandle     \packet_size_hndl;

===============================================================================
                              Temporary Variable Block
===============================================================================
int            alloc_instrms;
int            i;
int            intrpt_code;
packet*        pkipv4;
packet*        seg_ipv4;
packet*        pksndcp;
double         pk_ipv4_size;

===============================================================================
                    Enter Execs for the forced state "INICIO"
===============================================================================
/* Registra as estatisticas dos pacotes recebidos*/
pktssec_rcvd_stathandle      = op_stat_reg ("Traffic Sink.Traffic Received
(packets/sec)",       OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);
packet_size_hndl                     = op_stat_reg ("Generator.Packet Size (bits)",
       OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);

/* Registra as estatisticas dos pacotes enviados    */
packets_sent_hndl            = op_stat_reg ("Generator.Traffic Sent (packets/sec)",
               OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);

/*Chama o estado IDLE */
op_intrpt_schedule_self (op_sim_time (), SSC_INICIO);

/*Cria um buffer de montagem de pacotes*/
buf_mon = op_sar_buf_create (OPC_SAR_BUF_TYPE_REASSEMBLY, OPC_SAR_BUF_OPT_DEFAULT);
-------------------------------------------------------------------------------

===============================================================================
                     Exit Execs for the forced state "INICIO"
===============================================================================
intrpt_code = op_intrpt_code ();
-------------------------------------------------------------------------------

===============================================================================
                            transition   INICIO -> LIVRE
===============================================================================
name:  INICIO
condition:     INICIO
executive:
color:  RGB000
drawing style: spline
```

```
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                   Enter Execs for the unforced state "LIVRE"
================================================================================
/*Identifica o tipo de interrupção*/
/*Se for uma interrupção de stream, o pacote sera obtido*/
if (op_intrpt_type () == OPC_INTRPT_STRM)
       {
       /*chama o estado SEG_SNDCP*/
       op_intrpt_schedule_self(op_sim_time (), SSC_REC_PK_SNDCP);
       }
--------------------------------------------------------------------------------

================================================================================
                   Exit Execs for the unforced state "LIVRE"
================================================================================
intrpt_code = op_intrpt_code ();
--------------------------------------------------------------------------------

================================================================================
                           transition   LIVRE -> MON_IP
================================================================================
name:  REC_PK_SNDCP
condition:     SSC_REC_PK_SNDCP
executive:
color:  RGB000
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                   Enter Execs for the forced state "MON_IP"
================================================================================
/* determina o número de streams alocadas */
alloc_instrms = op_strm_max_index_in ();

/* loop para cada stream alocado */
for (i = 0; i <= alloc_instrms; i++)
              {

               /* se um stream está conectado ... */
               if (op_strm_connected (OPC_STRM_IN, i) == OPC_TRUE)
                     {
                     /*Se o stream não está vazio obtem-se o pacote e seu tamanho */
                      if (op_strm_empty (i) == OPC_FALSE)
                            {
                            /* Obtem o pacote da camada superior*/
                            pksndcp = op_pk_get (i);

                            /* Atualiza estatisticas locais */
                            op_stat_write (pktssec_rcvd_stathandle,     1.0);
                            op_stat_write (pktssec_rcvd_stathandle,     0.0);

                            /*Desencapsula o pacote IPV4*/
                            op_pk_nfd_get (pksndcp, "Dados", &seg_ipv4);

                            /*MOnta o pacote IPV4 original*/
                            if (op_sar_pk_is_segment (seg_ipv4) == OPC_TRUE)
                                  {
                                  op_sar_rsmbuf_seg_insert (buf_mon, seg_ipv4);

                                  if ((pkipv4 = op_sar_rsmbuf_pk_remove (buf_mon))
!= OPC_NIL)
                                        {
                                        /*Obtem o tamnho do pacote*/
                                        pk_ipv4_size = op_pk_total_size_get
(pkipv4);

                                        /*Atualiza as estatisticas*/
                                        op_stat_write (packets_sent_hndl, 1.0);
                                        op_stat_write (packets_sent_hndl, 0.0);
```

```
                                                op_stat_write (packet_size_hndl,
pk_ipv4_size);

                                                /* Envia o pacote para a camada
superior*/
                                                op_pk_send (pkipv4, SSC_STRM_TO_HIGH);
                                                }
                                        }
                                /*Destroi o pacote SNDCP*/
                                op_pk_destroy (pksndcp);
                                }
                        }
                }
--------------------------------------------------------------------------------

================================================================================
                     Exit Execs for the forced state "MON_IP"
================================================================================
intrpt_code = op_intrpt_code ();
--------------------------------------------------------------------------------

================================================================================
                            transition   MON_IP -> LIVRE
================================================================================
name:  ENV_PK_IPV4
condition:     SSC_ENV_PK_IPV4
executive:
color:  RGB000
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------
```

# A19 – G_sink_IPV4

```
################################################################################
                        Process Model Report:  G_sink_IPV4
################################################################################

================================================================================
                              State Variable Block
================================================================================
Stathandle     \bits_rcvd_stathandle;
Stathandle     \bitssec_rcvd_stathandle;
Stathandle     \pkts_rcvd_stathandle;
Stathandle     \pktssec_rcvd_stathandle;
Stathandle     \ete_delay_stathandle;
Stathandle     \bits_rcvd_gstathandle;
Stathandle     \bitssec_rcvd_gstathandle;
Stathandle     \pkts_rcvd_gstathandle;
Stathandle     \pktssec_rcvd_gstathandle;
Stathandle     \ete_delay_gstathandle;
Stathandle     \packet_size_hndl;

================================================================================
                            Temporary Variable Block
================================================================================
Packet*        pkptr;
double         pk_size;
double         ete_delay;

================================================================================
                  Enter Execs for the unforced state "DESCARTE"
================================================================================
NONE
--------------------------------------------------------------------------------

================================================================================
                   Exit Execs for the unforced state "DESCARTE"
================================================================================
/* Obtem o pacote da camada inferior */
pkptr = op_pk_get (op_intrpt_strm ());

/* Atualiza as estatisticas*/
pk_size = (double) op_pk_total_size_get (pkptr);
ete_delay = op_sim_time () - op_pk_creation_time_get (pkptr);

op_stat_write (bits_rcvd_stathandle,               pk_size);
op_stat_write (pkts_rcvd_stathandle,               1.0);
op_stat_write (ete_delay_stathandle,               ete_delay);
op_stat_write (bitssec_rcvd_stathandle,            pk_size);
op_stat_write (bitssec_rcvd_stathandle,            0.0);
op_stat_write (pktssec_rcvd_stathandle,            1.0);
op_stat_write (pktssec_rcvd_stathandle,            0.0);
op_stat_write (packet_size_hndl, pk_size);
op_stat_write (bits_rcvd_gstathandle,              pk_size);
op_stat_write (pkts_rcvd_gstathandle,              1.0);
op_stat_write (ete_delay_gstathandle,              ete_delay);
op_stat_write (bitssec_rcvd_gstathandle,           pk_size);
op_stat_write (bitssec_rcvd_gstathandle,           0.0);
op_stat_write (pktssec_rcvd_gstathandle,           1.0);
op_stat_write (pktssec_rcvd_gstathandle,           0.0);

/* Destroi o pacote recebido  */
op_pk_destroy (pkptr);
--------------------------------------------------------------------------------

================================================================================
                         transition   DESCARTE -> DESCARTE
================================================================================
name:  default
condition:     default
executive:
color:  #0000FF
drawing style: spline
```

```
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                    Enter Execs for the forced state "INICIO"
================================================================================
/* Registra as estatisitcas   */
bits_rcvd_stathandle          = op_stat_reg ("Traffic Sink.Traffic Received (bits)",
                        OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);
bitssec_rcvd_stathandle       = op_stat_reg ("Traffic Sink.Traffic Received
(bits/sec)",           OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);
pkts_rcvd_stathandle          = op_stat_reg ("Traffic Sink.Traffic Received
(packets)",            OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);
pktssec_rcvd_stathandle       = op_stat_reg ("Traffic Sink.Traffic Received
(packets/sec)",        OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);
ete_delay_stathandle          = op_stat_reg ("Traffic Sink.End-to-End Delay
(seconds)",            OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);
bits_rcvd_gstathandle         = op_stat_reg ("Traffic Sink.Traffic Received (bits)",
                        OPC_STAT_INDEX_NONE, OPC_STAT_GLOBAL);
bitssec_rcvd_gstathandle      = op_stat_reg ("Traffic Sink.Traffic Received
(bits/sec)",           OPC_STAT_INDEX_NONE, OPC_STAT_GLOBAL);
pkts_rcvd_gstathandle         = op_stat_reg ("Traffic Sink.Traffic Received
(packets)",            OPC_STAT_INDEX_NONE, OPC_STAT_GLOBAL);
pktssec_rcvd_gstathandle      = op_stat_reg ("Traffic Sink.Traffic Received
(packets/sec)",        OPC_STAT_INDEX_NONE, OPC_STAT_GLOBAL);
ete_delay_gstathandle         = op_stat_reg ("Traffic Sink.End-to-End Delay
(seconds)",            OPC_STAT_INDEX_NONE, OPC_STAT_GLOBAL);
packet_size_hndl                      = op_stat_reg ("Traffic Sink.Packet Size
(bits)",               OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);
--------------------------------------------------------------------------------

================================================================================
                     Exit Execs for the forced state "INICIO"
================================================================================
NONE
--------------------------------------------------------------------------------

================================================================================
                          transition   INICIO -> DESCARTE
================================================================================
name:  tr_12
condition:
executive:
color:  #080808
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------
```

# A20 – G_esc_FIFO

```
###############################################################################
                        Process Model Report:  G_esc_FIFO
###############################################################################

===============================================================================
                                  Header Block
===============================================================================
#define FILA_VAZIA      (op_q_empty ())
#define SVC_COMPLETO    op_intrpt_type () == OPC_INTRPT_SELF
#define CHEGADA         op_intrpt_type () == OPC_INTRPT_STRM

===============================================================================
                               State Variable Block
===============================================================================
int     \server_busy;
double  \service_rate;
Objid   \own_id;
Stathandle     \pktssec_rcvd_stathandle;
Stathandle     \packet_size_hndl;
Stathandle     \packets_sent_hndl;

===============================================================================
                             Temporary Variable Block
===============================================================================
Packet*         pkptr;
int             pk_len;
double          pk_svc_time;
int             insert_ok;
int             strm;

===============================================================================
                    Enter Execs for the forced state "INICIO"
===============================================================================
/* Registra as estatisticas */
pktssec_rcvd_stathandle      = op_stat_reg ("Traffic Sink.Traffic Received
(packets/sec)",       OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);
packet_size_hndl                    = op_stat_reg ("Generator.Packet Size (bits)",
       OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);
packets_sent_hndl            = op_stat_reg ("Generator.Traffic Sent (packets/sec)",
              OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);

/* Inicilializa o servidor com IDLE  */
server_busy = 0;

/* Obtem o id do módulo surround*/
own_id = op_id_self ();

/* Obtem o valo da taxa de serviço   */
op_ima_obj_attr_get (own_id, "service_rate", &service_rate);
-------------------------------------------------------------------------------

===============================================================================
                     Exit Execs for the forced state "INICIO"
===============================================================================
NONE
-------------------------------------------------------------------------------

===============================================================================
                           transition   INICIO -> INSERÇÃO
===============================================================================
name:  tr_1
condition:     CHEGADA
executive:
color:  RGB300
drawing style: line
doc file:      pr_transition
-------------------------------------------------------------------------------

===============================================================================
                             transition   INICIO -> LIVRE
```

```
================================================================================
name:  tr_2
condition:    default
executive:
color: #0000FF
drawing style: spline
doc file:     pr_transition
--------------------------------------------------------------------------------

================================================================================
                  Enter Execs for the forced state "INSERÇÃO"
================================================================================
/* Obtem o pacote do fluxo de entrada       */
pkptr = op_pk_get (op_intrpt_strm ());

/* Insere o pacote na fila    */
if (op_subq_pk_insert (0, pkptr, OPC_QPOS_TAIL) != OPC_QINS_OK)
    {
    op_pk_destroy (pkptr);
    insert_ok = 0;
    }
else{
    insert_ok = 1;
    }
--------------------------------------------------------------------------------

================================================================================
                   Exit Execs for the forced state "INSERÇÃO"
================================================================================
NONE
--------------------------------------------------------------------------------

================================================================================
                          transition   INSERÇÃO -> FILA
================================================================================
name:  tr_3
condition:    !server_busy && insert_ok
executive:
color: RGB300
drawing style: spline
doc file:     pr_transition
--------------------------------------------------------------------------------

================================================================================
                          transition   INSERÇÃO -> LIVRE
================================================================================
name:  tr_4
condition:    default
executive:
color: #0000FF
drawing style: spline
doc file:     pr_transition
--------------------------------------------------------------------------------

================================================================================
                   Enter Execs for the unforced state "LIVRE"
================================================================================
NONE
--------------------------------------------------------------------------------

================================================================================
                   Exit Execs for the unforced state "LIVRE"
================================================================================
NONE
--------------------------------------------------------------------------------

================================================================================
                          transition   LIVRE -> INSERÇÃO
================================================================================
name:  tr_5
condition:    CHEGADA
executive:
color: RGB300
drawing style: spline
```

```
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                          transition   LIVRE -> REMOÇÃO
================================================================================
name:  tr_6
condition:     SVC_COMPLETO
executive:
color:  RGB300
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                     Enter Execs for the forced state "FILA"
================================================================================
/* Acessa o pacote na cabeça da fila */
pkptr = op_subq_pk_access (0, OPC_QPOS_HEAD);

/* determina o tamanho do pacote     */
pk_len = op_pk_total_size_get (pkptr);

/* determina o tempo de processamento       */
pk_svc_time = pk_len / service_rate;

/* Agenda ainterrupção para o processo              */
op_intrpt_schedule_self (op_sim_time () + pk_svc_time, 0);

/* Muda o estado do servidor para busy      */
server_busy = 1;
--------------------------------------------------------------------------------

================================================================================
                      Exit Execs for the forced state "FILA"
================================================================================
NONE
--------------------------------------------------------------------------------

================================================================================
                           transition   FILA -> LIVRE
================================================================================
name:  tr_7
condition:
executive:
color:  RGB300
drawing style: line
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                    Enter Execs for the forced state "REMOÇÃO"
================================================================================
/* Retira o pacote da cabeça da fila */
pkptr = op_subq_pk_remove (0, OPC_QPOS_HEAD);

/* Envia o pacote para camada inferior      */
op_pk_send_forced (pkptr, 0);

/* Muda o estado do servidor para IDLE      */
server_busy = 0;
--------------------------------------------------------------------------------
================================================================================
                    Exit Execs for the forced state "REMOÇÃO"
================================================================================
NONE
--------------------------------------------------------------------------------

================================================================================
                           transition   REMOÇÃO -> FILA
================================================================================
name:  tr_8
condition:     !FILA_VAZIA
executive:
```

```
color:  RGB300
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                          transition   REMOÇÃO -> LIVRE
================================================================================
name:   tr_9
condition:     default
executive:
color:  #0000FF
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------
```

# A21 – G_esc_PQ

```
###############################################################################
                        Process Model Report:  G_esc_PQ
###############################################################################

===============================================================================
                                   Header Block
===============================================================================
#define FILA_VAZIA     (op_subq_empty (n))
#define SVC_COMPLETO    op_intrpt_type () == OPC_INTRPT_SELF
#define CHEGADA         op_intrpt_type () == OPC_INTRPT_STRM

/*Variáveis Globais Externas*/

extern int TFI_0, TFI_1, TFI_2, TFI_3;
extern int NSAPI_0, NSAPI_1, NSAPI_2, NSAPI_3;

===============================================================================
                               State Variable Block
===============================================================================
int    \server_busy;
double \service_rate;
Objid  \own_id;
Stathandle     \pktssec_rcvd_stathandle;
Stathandle     \packet_size_hndl;
Stathandle     \packets_sent_hndl;
int    \n;

===============================================================================
                            Temporary Variable Block
===============================================================================
Packet*         pkptr;
int             pk_len;
double          pk_svc_time;
int             insert_ok;
int                   QoS;
int                          fluxo;
int                          TFI;

===============================================================================
                    Enter Execs for the forced state "INICIO"
===============================================================================
/* Registra as estatisticas */
pktssec_rcvd_stathandle      = op_stat_reg ("Traffic Sink.Traffic Received
(packets/sec)",       OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);

/* Inicilializa o servidor com IDLE  */
server_busy = 0;

/* Obtem o id do módulo surround*/
own_id = op_id_self ();

/* Obtem o valo da taxa de serviço   */
op_ima_obj_attr_get (own_id, "service_rate", &service_rate);
-------------------------------------------------------------------------------

===============================================================================
                     Exit Execs for the forced state "INICIO"
===============================================================================
NONE
-------------------------------------------------------------------------------

===============================================================================
                        transition   INICIO -> CLASIIFICADOR
===============================================================================
name:  tr_1
condition:     CHEGADA
executive:
color:  RGB300
drawing style: line
doc file:      pr_transition
```

```
--------------------------------------------------------------------------------

================================================================================
                          transition   INICIO -> LIVRE
================================================================================
name:  tr_2
condition:     default
executive:
color:  #0000FF
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                 Enter Execs for the forced state "CLASIIFICADOR"
================================================================================
fluxo = op_intrpt_strm();
if (op_strm_empty (fluxo) == OPC_FALSE)
       {
/* Obtem os pacotes de múltiplos fluxos de entrada*/
pkptr = op_pk_get (fluxo);

/*Obtem a prioridade do pacote*/
op_pk_nfd_get (pkptr, "TFI", &TFI);

if(TFI == TFI_0)
       QoS = NSAPI_0;
if(TFI == TFI_1)
       QoS = NSAPI_1;
if(TFI == TFI_2)
       QoS = NSAPI_2;
if(TFI == TFI_3)
       QoS = NSAPI_3;

switch (QoS)
                {
                case 1: n=0;               break;
                case 2: n=1;               break;
                case 3: n=2;               break;
                case 4: n=2;               break;
                }

/* Insere o pacote na fila*/
if (op_subq_pk_insert (n, pkptr, OPC_QPOS_TAIL) != OPC_QINS_OK)
   {
       op_pk_destroy (pkptr);
       insert_ok = 0;
       }
       else{
              insert_ok = 1;
              }
}
--------------------------------------------------------------------------------

================================================================================
                 Exit Execs for the forced state "CLASIIFICADOR"
================================================================================
NONE
--------------------------------------------------------------------------------

================================================================================
                    transition   CLASIIFICADOR -> SUB - FILAS
================================================================================
name:  tr_3
condition:     !server_busy && insert_ok
executive:
color:  RGB300
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                       transition   CLASIIFICADOR -> LIVRE
================================================================================
```

```
name:  tr_4
condition:     default
executive:
color: #0000FF
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                    Enter Execs for the unforced state "LIVRE"
================================================================================
NONE
--------------------------------------------------------------------------------

================================================================================
                    Exit Execs for the unforced state "LIVRE"
================================================================================
NONE
--------------------------------------------------------------------------------

================================================================================
                       transition   LIVRE -> CLASIIFICADOR
================================================================================
name:  tr_5
condition:     CHEGADA
executive:
color: RGB300
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                        transition   LIVRE -> ESCALONADOR
================================================================================
name:  tr_6
condition:     SVC_COMPLETO
executive:
color: RGB300
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                  Enter Execs for the forced state "SUB - FILAS"
================================================================================
/* Acessa o pacote na cabeça da fila */
pkptr = op_subq_pk_access (n, OPC_QPOS_HEAD);
/* determina o tamanho do pacote     */
pk_len = op_pk_total_size_get (pkptr);
/* determina o tempo de processamento       */
pk_svc_time = pk_len / service_rate;
/* Agenda ainterrupção para o processo              */
op_intrpt_schedule_self (op_sim_time () + pk_svc_time, 0);
/* Muda o estado do servidor para busy      */
server_busy = 1;
--------------------------------------------------------------------------------

================================================================================
                  Exit Execs for the forced state "SUB - FILAS"
================================================================================
NONE
--------------------------------------------------------------------------------
================================================================================
                        transition   SUB - FILAS -> LIVRE
================================================================================
name:  tr_7
condition:
executive:
color: #040004
drawing style: line
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
```

```
                Enter Execs for the forced state "ESCALONADOR"
================================================================================
/*Remove o pacote da fila de maior prioridade*/
if (op_subq_empty (0) == OPC_FALSE)
    {
       pkptr = op_subq_pk_remove (0, OPC_QPOS_HEAD);
       op_pk_send (pkptr, 0);
       }
       else
              {
              if (op_subq_empty (1) == OPC_FALSE)
                     {
                     pkptr = op_subq_pk_remove (1, OPC_QPOS_HEAD);
                     op_pk_send (pkptr, 0);
                     }
                     else
                            {
                            if (op_subq_empty (2) == OPC_FALSE)
                                   {
                                   pkptr = op_subq_pk_remove (2, OPC_QPOS_HEAD);
                                   op_pk_send (pkptr, 0);
                                   }
                            }
               }
/* Muda o estado do servidor para IDLE      */
server_busy = 0;
--------------------------------------------------------------------------------

================================================================================
                 Exit Execs for the forced state "ESCALONADOR"
================================================================================
NONE
--------------------------------------------------------------------------------

================================================================================
                    transition   ESCALONADOR -> SUB - FILAS
================================================================================
name:  tr_8
condition:     !FILA_VAZIA
executive:
color: RGB300
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                       transition   ESCALONADOR -> LIVRE
================================================================================
name:  tr_9
condition:     default
executive:
color: #0000FF
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------
```

# A22 – G_esc_WFQ

```
###############################################################################
                        Process Model Report:  G_esc_WFQ
###############################################################################

===============================================================================
                                   Header Block
===============================================================================
#define FILA_VAZIA           (op_subq_empty (n))
#define SVC_COMPLETO         op_intrpt_type () == OPC_INTRPT_SELF
#define CHEGADA              op_intrpt_type () == OPC_INTRPT_STRM

/*Variáveis Globais Externas*/
extern int TFI_0, TFI_1, TFI_2, TFI_3;
extern int NSAPI_0, NSAPI_1, NSAPI_2, NSAPI_3;
===============================================================================
                                State Variable Block
===============================================================================
int            \server_busy;
double         \service_rate;
Objid          \own_id;
Stathandle     \pktssec_rcvd_stathandle;
Stathandle     \packet_size_hndl;
int            \peso_0;
int            \peso_1;
int            \peso_2;
int            \n;
int            \round_anterior;
int            \tempo_anterior;
int            \FN_anterior;

===============================================================================
                             Temporary Variable Block
===============================================================================
Packet*                      pkptr;
int                          prio;
int                          prio_anterior;
double                       pk_svc_time;
double                       pk_size;
int                          pk_len;
int                          insert_ok;
int                          QoS;
int                          m;
int                          i;
int                          FN;
int                          round;
int                          tempo_atual;
int                          max;
int                          peso;
int                          peso_total;
int                          con_ativas;
int                          fluxo;
int                          TFI;

===============================================================================
                    Enter Execs for the forced state "INICIO"
===============================================================================
/* Inicilializa o servidor com IDLE  */
server_busy = 0;

/* Obtem o id do módulo surround*/
own_id = op_id_self ();

/* Obtem os atributos do módulo surround    */
op_ima_obj_attr_get (own_id, "service_rate", &service_rate);
op_ima_obj_attr_get (own_id, "Peso_0", &peso_0);
op_ima_obj_attr_get (own_id, "Peso_1", &peso_1);
op_ima_obj_attr_get (own_id, "Peso_2", &peso_2);

/*Registra as estatisticas*/
```

```
pktssec_rcvd_stathandle      = op_stat_reg ("Traffic Sink.Traffic Received
(packets/sec)",       OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);
packet_size_hndl                    = op_stat_reg ("Generator.Packet Size (bits)",
       OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);

/*Inicializa algumas variáveis*/
tempo_anterior = 0;
round_anterior = 0;
FN_anterior = 0;
--------------------------------------------------------------------------------
================================================================================
                    Exit Execs for the forced state "INICIO"
================================================================================
--------------------------------------------------------------------------------
================================================================================
                       transition   INICIO -> CLASSIFICADOR
================================================================================
name:  tr_1
condition:     CHEGADA
executive:
color:  RGB300
drawing style: line
doc file:      pr_transition
--------------------------------------------------------------------------------
================================================================================
                           transition   INICIO -> LIVRE
================================================================================
name:  tr_2
condition:     default
executive:
color:  #0000FF
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------
================================================================================
                Enter Execs for the forced state "CLASSIFICADOR"
================================================================================
/*Obtem-se o pacote e seu tamanho*/
pkptr = op_pk_get (op_intrpt_strm());
pk_size = op_pk_total_size_get (pkptr);
peso_total = 0;
/*Obtem-se a prioridade do pacote*/
op_pk_nfd_get (pkptr, "TFI", &TFI);
if(TFI == TFI_0)
       QoS = NSAPI_0;
if(TFI == TFI_1)
       QoS = NSAPI_1;
if(TFI == TFI_2)
       QoS = NSAPI_2;
if(TFI == TFI_3)
       QoS = NSAPI_3;
/*Atribui-se o peso*/
switch (QoS)
       {
  case 1:      peso = peso_0; n = 0;         break;
  case 2:      peso = peso_1; n = 1;     break;
  case 3:      peso = peso_2; n = 2;         break;
  case 4:      peso = peso_2; n = 2;         break;
    }
if (tempo_anterior == 0)
       {
       /*tempo_anterior = op_sim_time();*/
       FN_anterior = 0;
       round_anterior =0;
       }
/*Verifica quantas conexões estão ativas*/
con_ativas = 0;
for (i=0;i<3;i++)
       {
       if (op_subq_empty (i) == OPC_FALSE)
              {
              con_ativas++ ;
              switch (i)
                     {
```

```
                        case 0 : peso_total = peso_total + peso_0;          break;
                        case 1 : peso_total = peso_total + peso_1;          break;
                        case 2 : peso_total = peso_total + peso_2;          break;
                        }
                }
        }
/*Obtem o tempo atual*/
tempo_atual = op_sim_time();
/*Cálculo do Round*/
if (con_ativas != 0)
        {
        round = ((tempo_atual - tempo_anterior)/peso_total) + round_anterior;
        }
        else
                {
                round = 0;
                }

 if (round == 0)
        {
        FN_anterior = 0;
        }

/* Cálculo do Finish Time (FN)*/
if (round > FN_anterior)
        {
        max = round;
        }
        else
                {
                max = FN_anterior;
                }
FN = max + (pk_size/peso);
op_stat_write (packet_size_hndl, FN);
/*Atribui-se a prioridade de tempo ao pacote*/
op_pk_priority_set (pkptr, FN);
/*Insere-se o pacote na cauda da lista*/
if (op_subq_pk_insert (n, pkptr, OPC_QPOS_TAIL) != OPC_QINS_OK)
        {
        op_pk_destroy (pkptr);
        insert_ok = 0;
        }
        else
                {
                insert_ok = 1;
                }
/* Atualização de variáveis*/
tempo_anterior = tempo_atual;
round_anterior = round;
FN_anterior = FN;
--------------------------------------------------------------------------------
================================================================================
                Exit Execs for the forced state "CLASSIFICADOR"
================================================================================
NONE
--------------------------------------------------------------------------------
================================================================================
                   transition   CLASSIFICADOR -> SUB - FILAS
================================================================================
name:  tr_3
condition:     !server_busy && insert_ok
executive:
color: RGB300
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------
================================================================================
                        transition   CLASSIFICADOR -> LIVRE
================================================================================
name:  tr_4
condition:     default
executive:
color: #0000FF
drawing style: spline
```

```
doc file:      pr_transition
--------------------------------------------------------------------------------
================================================================================
                    Enter Execs for the unforced state "LIVRE"
================================================================================
NONE
--------------------------------------------------------------------------------
================================================================================
                    Exit Execs for the unforced state "LIVRE"
================================================================================
NONE
--------------------------------------------------------------------------------
================================================================================
                       transition   LIVRE -> CLASSIFICADOR
================================================================================
name:  tr_5
condition:     CHEGADA
executive:
color: RGB300
drawing style: spline
doc file:      pr_transition

--------------------------------------------------------------------------------
================================================================================
                        transition   LIVRE -> ESCALONADOR
================================================================================
name:  tr_6
condition:     SVC_COMPLETO
executive:
color: RGB300
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------
================================================================================
                  Enter Execs for the forced state "SUB - FILAS"
================================================================================
/* Acessa o pacote na cabeça da fila */
pkptr = op_subq_pk_access (n, OPC_QPOS_HEAD);

/* determina o tamanho do pacote     */
pk_len = op_pk_total_size_get (pkptr);

/* determina o tempo de processamento       */
pk_svc_time = pk_len / service_rate;

/* Agenda ainterrupção para o processo             */
op_intrpt_schedule_self (op_sim_time () + pk_svc_time, 0);

/* Muda o estado do servidor para busy      */
server_busy = 1;
--------------------------------------------------------------------------------
================================================================================
                  Exit Execs for the forced state "SUB - FILAS"
================================================================================
NONE
--------------------------------------------------------------------------------
================================================================================
                        transition   SUB - FILAS -> LIVRE
================================================================================
name:  tr_7
condition:
executive:
color: #040004
drawing style: line
doc file:      pr_transition
--------------------------------------------------------------------------------
================================================================================
                  Enter Execs for the forced state "ESCALONADOR"
================================================================================
/*Remove o pacote que tiver o menor FN dentre as cabeças das sub-filas*/
m=3;
for(i=0;i<3;i++)
       {
       if (op_subq_empty (i) == OPC_FALSE)
```

```
                {
                 pkptr = op_subq_pk_access (i, OPC_QPOS_HEAD);
                 prio =  op_pk_priority_get (pkptr);
                 if (i == 0)
                        {
                        prio_0 = prio;
                        m=0;
                        }
                 if (i == 1)
                        {
                        if (m != 0)
                                {
                                prio_1 = prio;
                                m=1;
                                }
                        else
                                {
                                if (prio < prio_0)
                                      m=1;
                                }
                        }
                 if (i == 2)
                        {
                        if (m == 0)
                                {
                                if (prio < prio_0)
                                        m=2;
                                }
                        if (m == 1)
                                {
                                if (prio < prio_1)
                                        m=2;
                                }
                        if (m == 3)
                                m=2;
                        }
                 }
        }
if (m < 3)
        {
        if (op_subq_empty (m) == OPC_FALSE)
                {
                pkptr = op_subq_pk_remove (m, OPC_QPOS_HEAD);
                op_pk_send_forced (pkptr, 0);
                }
        server_busy = 0;
        }
--------------------------------------------------------------------------------
================================================================================
                  Exit Execs for the forced state "ESCALONADOR"
================================================================================
NONE
--------------------------------------------------------------------------------
================================================================================
                       transition   ESCALONADOR -> LIVRE
================================================================================
name:  tr_9
condition:     default
executive:
color: #0000FF
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------
================================================================================
                     transition   ESCALONADOR -> SUB - FILAS
================================================================================
name:  tr_8
condition:     !FILA_VAZIA
executive:
color: RGB300
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------
```

# A23 – G_esc_WRR

```
###############################################################################
                        Process Model Report:  G_esc_WRR
###############################################################################

===============================================================================
                                  Header Block
===============================================================================
#define FILA_VAZIA            (op_subq_empty (n))
#define SVC_COMPLETO          op_intrpt_type () == OPC_INTRPT_SELF
#define CHEGADA               op_intrpt_type () == OPC_INTRPT_STRM

/*Variáveis Globais Externas*/
extern int TFI_0, TFI_1, TFI_2, TFI_3;
extern int NSAPI_0, NSAPI_1, NSAPI_2, NSAPI_3;
===============================================================================
                                State Variable Block
===============================================================================
int            \server_busy;
double         \service_rate;
Objid          \own_id;
Stathandle     \pktssec_rcvd_stathandle;
int            \n;
int            \peso_0;
int            \peso_1;
int            \peso_2;
int            \base;
Stathandle     \packet_size_hndl;

===============================================================================
                             Temporary Variable Block
===============================================================================
Packet*         pkptr;
int             pk_len;
double          pk_svc_time;
int             Insert_ok;
int             QoS;
int             a,m;
double          pkptr_size;
int             TFI;

===============================================================================
                    Enter Execs for the forced state "INICIO"
===============================================================================
/* Inicilializa o servidor com IDLE  */
server_busy = 0;

/* Obtem o id do módulo Surround     */
own_id = op_id_self ();

/* Obtem os tributos do módulo       */
op_ima_obj_attr_get (own_id, "service_rate", &service_rate);
op_ima_obj_attr_get (own_id, "Peso_0", &peso_0);
op_ima_obj_attr_get (own_id, "Peso_1", &peso_1);
op_ima_obj_attr_get (own_id, "Peso_2", &peso_2);
op_ima_obj_attr_get (own_id, "Base", &base);

/*Registra as estatisticas*/
pktssec_rcvd_stathandle      = op_stat_reg ("Traffic Sink.Traffic Received
(packets/sec)",       OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);
packet_size_hndl                    = op_stat_reg ("Generator.Packet Size (bits)",
       OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);
-------------------------------------------------------------------------------

===============================================================================
                     Exit Execs for the forced state "INICIO"
===============================================================================
-------------------------------------------------------------------------------

===============================================================================
                       transition   INICIO -> CLASSIFICADOR
```

```
================================================================================
name:  tr_1
condition:     CHEGADA
executive:
color: RGB300
drawing style: line
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                          transition   INICIO -> LIVRE
================================================================================
name:  tr_2
condition:     default
executive:
color: #0000FF
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                Enter Execs for the forced state "CLASSIFICADOR"
================================================================================
          /*Otem-se o pacote*/
          pkptr = op_pk_get (op_intrpt_strm());

          /*Obtem a prioridade do pacote*/
          op_pk_nfd_get (pkptr, "TFI", &TFI);

          if(TFI == TFI_0)
                 QoS = NSAPI_0;
          if(TFI == TFI_1)
                 QoS = NSAPI_1;
          if(TFI == TFI_2)
                 QoS = NSAPI_2;
          if(TFI == TFI_3)
                 QoS = NSAPI_3;

          switch (QoS)
                 {
                 case 1:      n=0;   break;
                 case 2:      n=1;   break;
                 case 3:   n=2;      break;
                 case 4:      n=2;   break;
                  }
          /*Insere-se o pacote na cauda da lista*/
          if (op_subq_pk_insert (n, pkptr, OPC_QPOS_TAIL) != OPC_QINS_OK)
                       {
                       op_pk_destroy (pkptr);
                       insert_ok = 0;
                       }
               else
                       {
                       insert_ok = 1;
                       }
--------------------------------------------------------------------------------

================================================================================
                Exit Execs for the forced state "CLASSIFICADOR"
================================================================================
NONE
--------------------------------------------------------------------------------

================================================================================
                    transition   CLASSIFICADOR -> SUB - FILAS
================================================================================
name:  tr_10
condition:     !server_busy && insert_ok
executive:
color: RGB300
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------
```

```
================================================================================
                        transition   CLASSIFICADOR -> LIVRE
================================================================================
name:  tr_4
condition:     default
executive:
color: #0000FF
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                    Enter Execs for the unforced state "LIVRE"
================================================================================
NONE
--------------------------------------------------------------------------------

================================================================================
                    Exit Execs for the unforced state "LIVRE"
================================================================================
NONE
--------------------------------------------------------------------------------

================================================================================
                        transition   LIVRE -> CLASSIFICADOR
================================================================================
name:  tr_5
condition:     CHEGADA
executive:
color: RGB300
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                         transition   LIVRE -> ESCALONADOR
================================================================================
name:  tr_6
condition:     SVC_COMPLETO
executive:
color: RGB300
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                  Enter Execs for the forced state "ESCALONADOR"
================================================================================
/*Remove o pacote da sub-fila de acordo com o seu peso*/
switch (n)
       {
       case 0 : m = peso_0/base; break;
       case 1 : m = peso_1/base; break;
       case 2 : m = peso_2/base; break;
       }
a=0;
do{
       if (op_subq_empty (n) == OPC_FALSE)
              {
              pkptr = op_subq_pk_remove (n, OPC_QPOS_HEAD);
              op_pk_send (pkptr, 0);
              }
       a=a+1;

       }while (a != m);

server_busy = 0;
       op_stat_write (packet_size_hndl, a);
--------------------------------------------------------------------------------
================================================================================
                  Exit Execs for the forced state "ESCALONADOR"
================================================================================
NONE
--------------------------------------------------------------------------------
```

```
================================================================================
                        transition   ESCALONADOR -> LIVRE
================================================================================
name:  tr_9
condition:     default
executive:
color: #0000FF
drawing style: spline
doc file:      pr_transition

--------------------------------------------------------------------------------

================================================================================
                     transition   ESCALONADOR -> SUB - FILAS
================================================================================
name:  tr_88
condition:     !FILA_VAZIA
executive:
color: #FF0000
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                 Enter Execs for the forced state "SUB - FILAS "
================================================================================
/* Acessa o pacote na cabeça da fila */
pkptr = op_subq_pk_access (n, OPC_QPOS_HEAD);

/* determina o tamanho do pacote     */
pk_len = op_pk_total_size_get (pkptr);

/* determina o tempo de processamento       */
pk_svc_time = pk_len / service_rate;

/* Agenda ainterrupção para o processo              */
op_intrpt_schedule_self (op_sim_time () + pk_svc_time, 0);

/* Muda o estado do servidor para busy      */
server_busy = 1;
--------------------------------------------------------------------------------

================================================================================
                  Exit Execs for the forced state "SUB - FILAS "
================================================================================
NONE
--------------------------------------------------------------------------------

================================================================================
                        transition   SUB - FILAS  -> LIVRE
================================================================================
name:  tr_12
condition:
executive:
color: #040004
drawing style: line
doc file:      pr_transition
--------------------------------------------------------------------------------
```

# A24 – G_esc_DWRR

```
###############################################################################
                        Process Model Report:  G_esc_DWRR
###############################################################################

===============================================================================
                                  Header Block
===============================================================================
#define FILA_VAZIA      (op_subq_empty (n))
#define SVC_COMPLETO    op_intrpt_type () == OPC_INTRPT_SELF
#define CHEGADA         op_intrpt_type () == OPC_INTRPT_STRM

/*Variáveis Globais Externas*/
extern int TFI_0, TFI_1, TFI_2, TFI_3;
extern int NSAPI_0, NSAPI_1, NSAPI_2, NSAPI_3;
===============================================================================
                               State Variable Block
===============================================================================
int            \server_busy;
double         \service_rate;
Objid          \own_id;
Stathandle     \pktssec_rcvd_stathandle;
int            \n;
int            \quantum_0;
int            \quantum_1;
int            \quantum_2;
Stathandle     \packet_size_hndl;
int            \deficit_counter_0;
int            \deficit_counter_1;
int            \deficit_counter_2;

===============================================================================
                            Temporary Variable Block
===============================================================================
Packet*         pkptr;
int             pk_len;
double          pk_svc_time;
int             insert_ok;
int             QoS,a;
int             quantum;
int             deficit_counter;
double          pkptr_size;
int             TFI;
===============================================================================
                   Enter Execs for the forced state "INICIO"
===============================================================================
/* Inicilializa o servidor com IDLE  */
server_busy = 0;

/* Obtem o id do módulo Surround     */
own_id = op_id_self ();

/* Obtem os atributos do módulo      */
op_ima_obj_attr_get (own_id, "service_rate", &service_rate);
op_ima_obj_attr_get (own_id, "Quantum_0", &quantum_0);
op_ima_obj_attr_get (own_id, "Quantum_1", &quantum_1);
op_ima_obj_attr_get (own_id, "Quantum_2", &quantum_2);

/*Registra as estatisticas*/
pktssec_rcvd_stathandle      = op_stat_reg ("Traffic Sink.Traffic Received
(packets/sec)",       OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);
packet_size_hndl                    = op_stat_reg ("Generator.Packet Size (bits)",
       OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);

/*Inicializa as variáveis*/
deficit_counter_0 = 0;
deficit_counter_1 = 0;
deficit_counter_2 = 0;
-------------------------------------------------------------------------------

===============================================================================
```

```
                    Exit Execs for the forced state "INICIO"
================================================================================
--------------------------------------------------------------------------------

================================================================================
                       transition   INICIO -> CLASSIFICADOR
================================================================================
name:  tr_1
condition:     CHEGADA
executive:
color:  RGB300
drawing style: line
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                            transition   INICIO -> LIVRE
================================================================================
name:  tr_2
condition:     default
executive:
color:  #0000FF
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                 Enter Execs for the forced state "CLASSIFICADOR"
================================================================================
/*Otem-se o pacote*/
pkptr = op_pk_get (op_intrpt_strm());

/*Obtem a prioridade do pacote*/
op_pk_nfd_get (pkptr, "TFI", &TFI);

if(TFI == TFI_0)
       QoS = NSAPI_0;
if(TFI == TFI_1)
       QoS = NSAPI_1;
if(TFI == TFI_2)
       QoS = NSAPI_2;
if(TFI == TFI_3)
       QoS = NSAPI_3;

switch (QoS)
         {
         case 1:      n=0;   break;
         case 2:      n=1;   break;
         case 3:      n=2;   break;
         case 4:      n=2;   break;
          }

  /*Insere-se o pacote na cauda da lista*/
if (op_subq_pk_insert (n, pkptr, OPC_QPOS_TAIL) != OPC_QINS_OK)
       {
       op_pk_destroy (pkptr);
       op_stat_write (pktssec_rcvd_stathandle,     1.0);
       op_stat_write (pktssec_rcvd_stathandle,     0.0);
       insert_ok = 0;
       }
       else
              {
              insert_ok = 1;
              }
--------------------------------------------------------------------------------

================================================================================
                 Exit Execs for the forced state "CLASSIFICADOR"
================================================================================
NONE
--------------------------------------------------------------------------------

================================================================================
                     transition   CLASSIFICADOR -> SUB - FILAS
```

```
================================================================================
name:   tr_10
condition:     !server_busy && insert_ok
executive:
color:  RGB300
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                       transition   CLASSIFICADOR -> LIVRE
================================================================================
name:   tr_4
condition:     default
executive:
color:  #0000FF
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                     Enter Execs for the unforced state "LIVRE"
================================================================================
NONE
--------------------------------------------------------------------------------

================================================================================
                     Exit Execs for the unforced state "LIVRE"
================================================================================
NONE
--------------------------------------------------------------------------------

================================================================================
                        transition   LIVRE -> ESCALONADOR
================================================================================
name:   tr_6
condition:     SVC_COMPLETO
executive:
color:  RGB300
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                       transition   LIVRE -> CLASSIFICADOR
================================================================================
name:   tr_5
condition:     CHEGADA
executive:
color:  RGB300
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                   Enter Execs for the forced state "ESCALONADOR"
================================================================================
/*Atualiza as variáveis deficit_counter e quantum*/
switch (n)
       {
       case 0: deficit_counter = deficit_counter_0;
                      quantum = quantum_0;
                      break;
       case 1: deficit_counter = deficit_counter_1;
                      quantum = quantum_1;
                      break;
       case 2: deficit_counter = deficit_counter_2;
                      quantum = quantum_2;
                      break;
       }

deficit_counter = quantum + deficit_counter;
a=0;
```

```
/*Remove pacotes da sub-fila de acordo com o saldo*/
do{
       if (op_subq_empty (n) == OPC_FALSE)
              {
              pkptr_size = op_pk_total_size_get(op_subq_pk_access (n,
OPC_QPOS_HEAD));
              if(deficit_counter > pkptr_size)
                     {
                     pkptr = op_subq_pk_remove (n, OPC_QPOS_HEAD);
                     op_pk_send (pkptr, 0);
                     deficit_counter = deficit_counter - pkptr_size;
                     }
                     else
                            a=2;
              }
              else
                     a=2;

       }while (a != 2);

/*Atualiza as variáveis*/
switch (n)
          {
          case 0:     if (op_subq_empty (0) == OPC_FALSE)
                                deficit_counter_0 = deficit_counter;
                                else
                                       deficit_counter_0 = 0;
          break;
          case 1:     if (op_subq_empty (1) == OPC_FALSE)
                                deficit_counter_1 = deficit_counter;
                                else
                                       deficit_counter_1 = 0;
          break;
          case 2:     if (op_subq_empty (2) == OPC_FALSE)
                                deficit_counter_2 = deficit_counter;
                                else
                                       deficit_counter_2 = 0;
          break;
          }
server_busy = 0;
--------------------------------------------------------------------------------

================================================================================
                  Exit Execs for the forced state "ESCALONADOR"
================================================================================
NONE
--------------------------------------------------------------------------------

================================================================================
                        transition   ESCALONADOR -> LIVRE
================================================================================
name:  tr_9
condition:     default
executive:
color: #0000FF
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------


================================================================================
                     transition   ESCALONADOR -> SUB - FILAS
================================================================================
name:  tr_88
condition:     !FILA_VAZIA
executive:
color: #FF0000
drawing style: spline
doc file:      pr_transition
--------------------------------------------------------------------------------

================================================================================
                  Enter Execs for the forced state "SUB - FILAS"
================================================================================
```

```
/* Acessa o pacote na cabeça da fila */
pkptr = op_subq_pk_access (n, OPC_QPOS_HEAD);

/* determina o tamanho do pacote     */
pk_len = op_pk_total_size_get (pkptr);

/* determina o tempo de processamento       */
pk_svc_time = pk_len / service_rate;

/* Agenda ainterrupção para o processo              */
op_intrpt_schedule_self (op_sim_time () + pk_svc_time, 0);

/* Muda o estado do servidor para busy      */
server_busy = 1;
--------------------------------------------------------------------------------

================================================================================
                 Exit Execs for the forced state "SUB - FILAS"
================================================================================
NONE
--------------------------------------------------------------------------------

================================================================================
                        transition   SUB - FILAS -> LIVRE
================================================================================
name:  tr_12
condition:
executive:
color:  RGB300
drawing style: line
doc file:      pr_transition
--------------------------------------------------------------------------------
```

# Anexo B – Formatos de pacotes

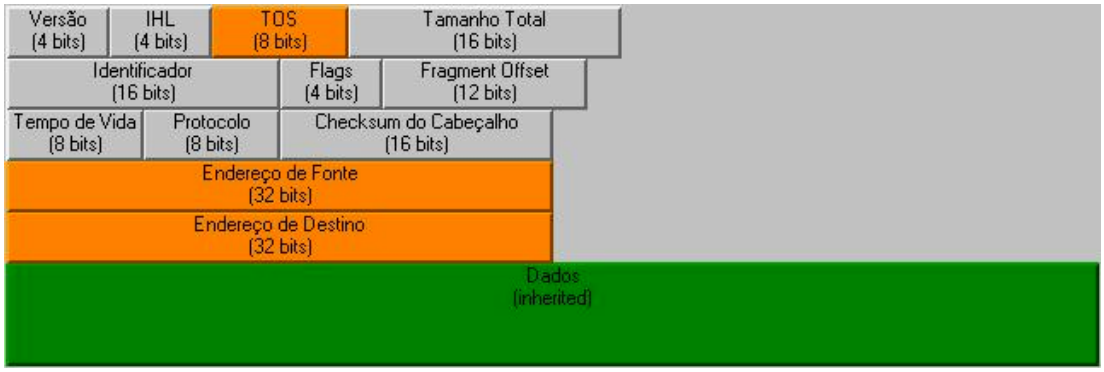


**Figura B1 –** *G_pk_IPV4*

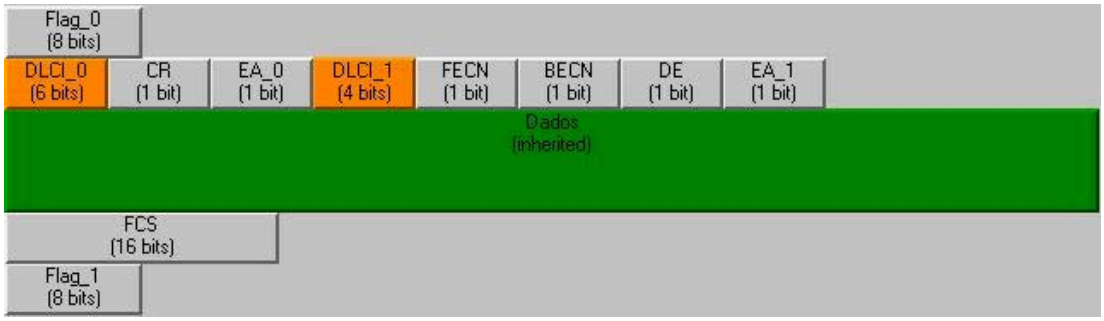


**Figura B2 –** *G_pk_FR*

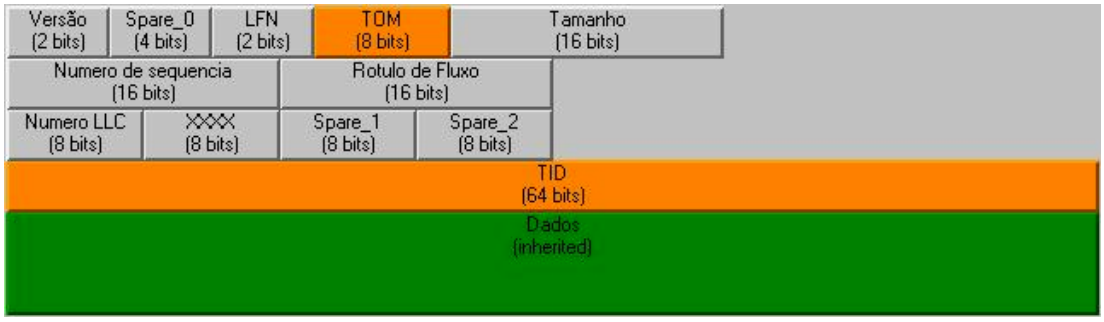


**Figura B3 –** *G_pk_GTP*

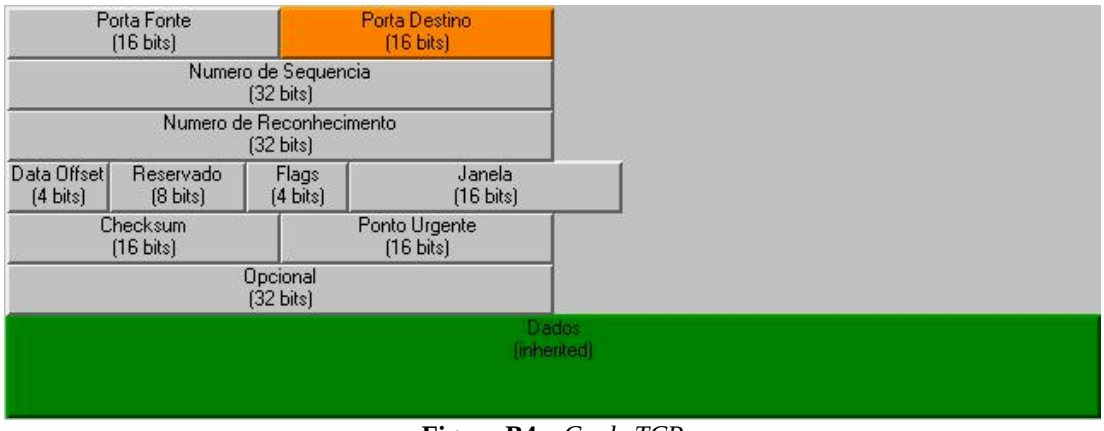


**Figura B4 –** *G_pk_TCP*


**Figura B5 –** *G_pk_UDP*

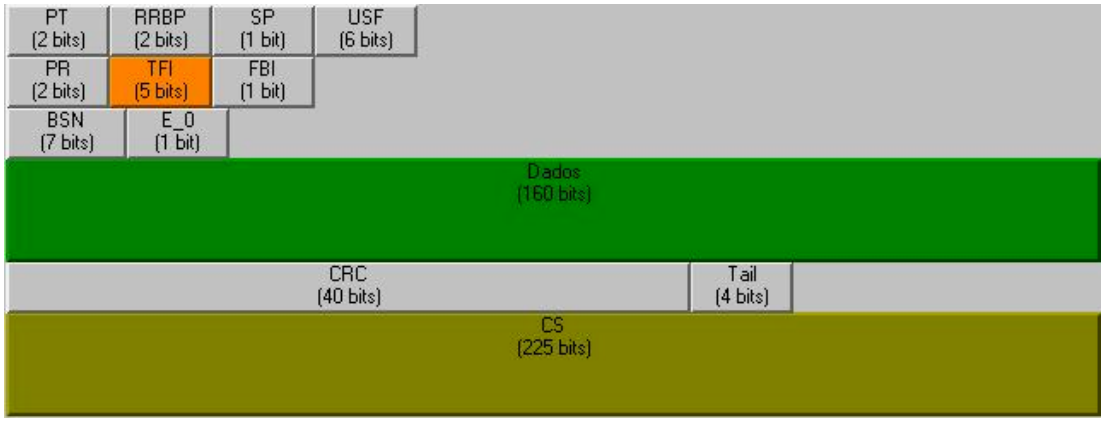


**Figura B6 –** *G_pk_RLCMAC_CS1*

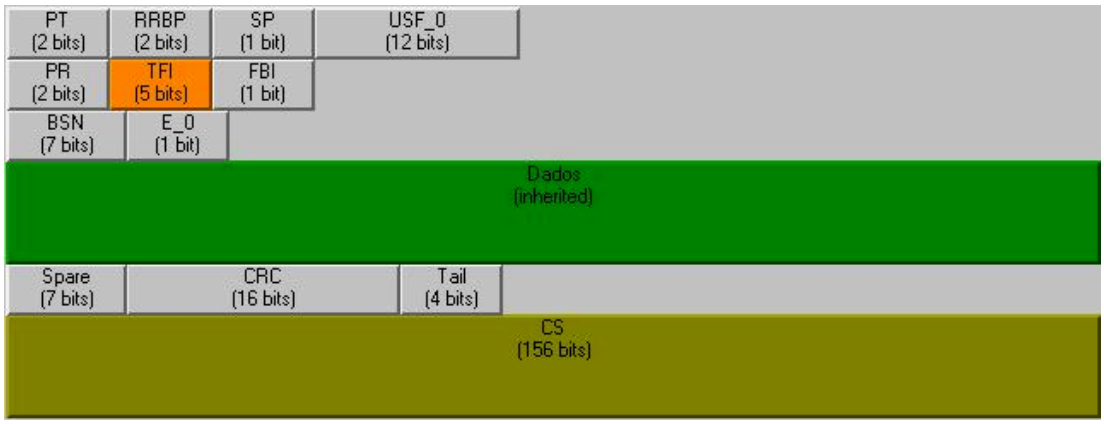


**Figura B7 –** *G_pk_RLCMAC_CS2*

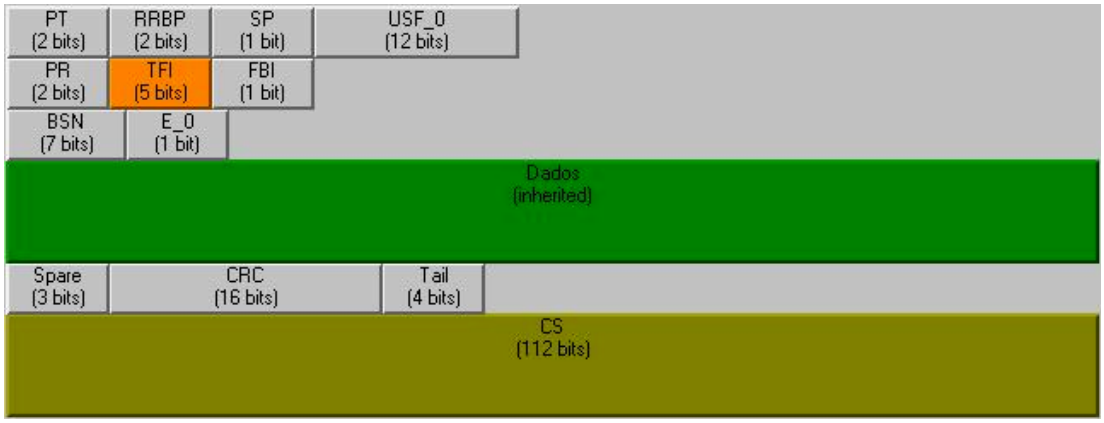


**Figura B8 –** *G_pk_RLCMAC_CS3*

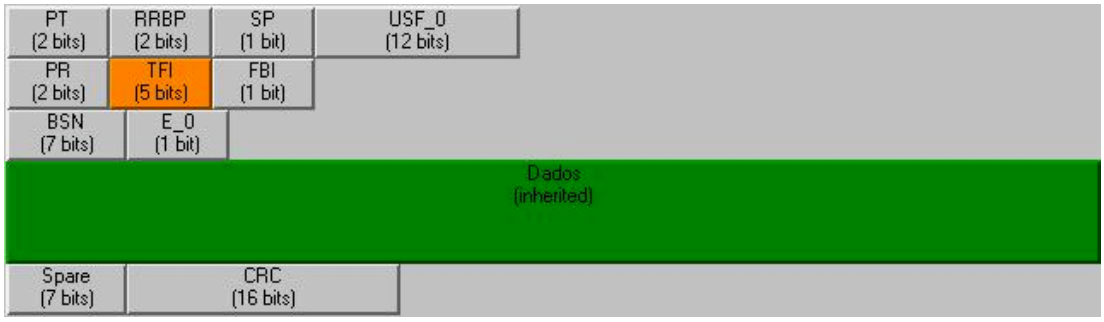


**Figura B9 –** *G_pk_RLCMAC_CS4*

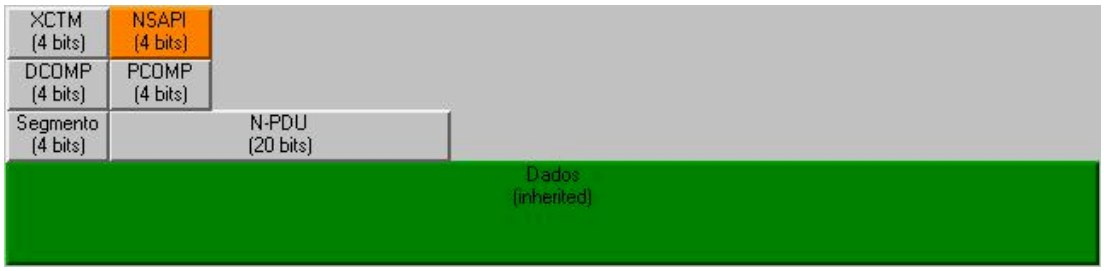


**Figura B10 –** *G_pk_SNDCP*


**Figura B11 –** *G_pk_LLC*

# Apêndice : Taxas de transmissão

| | | Taxa de transmissão (bps) | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | **(bits)** | **TS1** | **TS2** | **TS3** | **TS4** | **TS5** | **TS6** | **TS7** | **TS8** |
| **Bloco RLC/MAC CS-1** | 181 | 9050 | 18100 | 27150 | 36200 | 45250 | 54300 | 63350 | 72400 |
| **Bloco RLC/MAC CS-2** | 268 | 13400 | 26800 | 40200 | 53600 | 67000 | 80400 | 93800 | 107200 |
| **Bloco RLC/MAC CS-3** | 312 | 15600 | 31200 | 46800 | 62400 | 78000 | 93600 | 109200 | 124800 |
| **Bloco RLC/MAC CS-4** | 428 | 21400 | 42800 | 64200 | 85600 | 107000 | 128400 | 149800 | 171200 |
| **Bloco de rádio** | 456 | 22800 | 45600 | 68400 | 91200 | 114000 | 136800 | 159600 | 182400 |

**Tabela do Apêndice -** *Taxas de Transmissão*