Dissertação de Mestrado

Inatel
Instituto Nacional de Telecomunicações

ALGUMAS ANÁLISES SOBRE MECANISMOS PARA PROVER QUALIDADE DE SERVIÇO EM REDES MULTIMÍDIA

JOSÉ CARLOS RUELA

SETEMBRO / 2006

Algumas Análises sobre Mecanismos para Prover Qualidade de Serviço em Redes Multimídia

JOSÉ CARLOS RUELA

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

Orientador: **Prof. Dr. JOSÉ MARCOS CÂMARA BRITO**

Santa Rita do Sapucaí

2006

FOLHA DE APROVAÇÃO

Dissertação defendida e aprovada em
13 / 09 / 2006, pela comissão julgadora:

_____

Prof. Dr. José Marcos Câmara Brito (Orientador) - INATEL

_____

Prof. Dr. Shusaburo Motoyama (membro externo) - FEEC-UNICAMP

_____

Prof. Dr. Carlos Roberto dos Santos (membro interno) - INATEL

_____

Coordenador do Curso de Mestrado

“Aos meus pais e irmãos que sempre estiveram ao meu lado. E a minha amada Raquel, que sempre me apoiou”.

Agradecimentos

Em especial ao meu orientador, Prof. Dr. José Marcos Câmara Brito, pela sua atenção, paciência e competência, que muito contribuíram para a minha motivação em concluir este trabalho.

Aos professores das demais disciplinas do Mestrado, Prof. Dr. Carlos Alberto Ynoguti, Prof. Dr. Anilton Salles Garcia, Prof. Dr. Sandro Adriano Fasolo e Prof. Dr. Geraldo Gil Raimundo Gomes, pelos conhecimentos transmitidos.

Ao Inatel, pela sua estrutura e aos seus funcionários, pelo imenso senso de profissionalismo e dedicação.

Índice

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

AQM	-	*Active Queue Management*
CQ	-	*Custom Queuing*
DSCP	-	*Differentiated Services Code Point*
ECN	-	*Explicit Congestion Notification*
FCFS	-	*First Come, First Served*
FIFO	-	*First In First Out*
FTP	-	*File Transfer Protocol*
GPS	-	*Generalized Processor Sharing*
HTML	-	*Hyper Text Markup Language*
HTTP	-	*HyperText Transfer Protocol*
IP	-	*Internet Protocol*
IPX	-	*Internetwork Packet Exchange*
ITU-T	-	*International Telecommunications Union – Telecommunication Standardization Sector*
LAN	-	*Local Area Network*
LLQ	-	*Low Latency Queueing*
OSI	-	*Open Systems Interconnection*
PGPS	-	*Packetized Generalized Prodessor Sharing*
PPP	-	Point to Point Protocol
PQ	-	*Priority Queuing*
QoS	-	*Quality of Service*
RED	-	*Random Early Detection*
REM	-	*Random Exponential Marking*

RFC	-	*Request for Comments*
RTP	-	*Real-time Transport Protocol*
SCQF	-	*Self Clocked Fair Queuing*
SFQ	-	*Start-time Fair Queuing*
SMSS	-	*Sender Maximum Segment Size*
TCP	-	*Transmission Control Protocol*
TCP/IP	-	*Transmission Control Protocol / Internet Protocol*
TOS	-	*Type of Service*
UDP	-	*User Datagram Protocol*
VoIP	-	*Voice over* IP
WFQ	-	*Weighted Fair Queuing*
WF^2Q	-	*Worst-case Fair Weighted Fair Queuing*
WRED	-	*Weighted Random Early Detection*

Resumo

Com a crescente demanda por utilização de aplicações multimídia, tais como voz e vídeo, e principalmente nas que exigem comunicações interativas, ocorrendo em tempo real, novos parâmetros (atraso, variação do atraso e perda) passam a ser importantes para o bom funcionamento destas aplicações. Sendo assim, torna-se necessário o provimento de QoS (*Quality of Service* – Qualidade de Serviço) pela rede para suportar estas novas aplicações. Nesta dissertação é verificada a eficiência da utilização dos mecanismos de QoS aqui estudados, mostrando a viabilidade de sua utilização e implementação em uma rede "*best effort*". Dentro deste propósito, este trabalho apresenta o resultado de inúmeras simulações, onde fica comprovado que, com a utilização adequada dos mecanismos QoS de priorização e controle de congestionamento, pode-se viabilizar a utilização dos aplicativos de VoIP (*Voice over IP*) dentro dos limites aceitáveis de atraso e perda, mesmo quando o tráfego de voz compartilha um enlace de comunicação com outros tráfegos em situações de extremo congestionamento.

Palavras-chave: QoS, qualidade de serviço, mecanismos de priorização, controle de congestionamento.

Abstract

With the crescent demand for the use of multimedia applications, like voice and video, and mainly in the ones that demand interactive communications, occurring in real time, new parameters (delay, variation of the delay and loss) start to be important for the good operation of these applications. So, it becomes necessary the provision of QoS (Quality of Service) by the network to support these new applications. This dissertation verifies the efficiency of the use of the mechanisms of QoS that were studied here, showing the viability of its use and implementation in a "best effort" network. Inside of this purpose, this work presents the result of several simulations, where it is proven that, with the appropriate utilization of the scheduling mechanisms and congestion control mechanisms, the use of the applications of VoIP (Voice over IP) can be made possible inside of the acceptable limits of delay and loss, even when the voice traffic shares a communication connection with other traffics in situations of intense congestion.

Keywords: QoS, quality of service, scheduling mechanisms, congestion control.

1. Introdução

A Internet tem apresentado crescimento intenso, tanto em seu número de usuários, quanto na sua infra-estrutura e capacidade de transmissão. Apesar deste crescimento, ela ainda mantém sua característica nativa, oferecendo um serviço *best effort* (melhor esforço), onde todos os tipos de tráfego são tratados de maneira igual e os pacotes são transmitidos sem nenhuma garantia de vazão, tempo máximo de atraso e variação de atraso, nem taxa máxima de perda de pacotes.

O serviço *best effort* tem sido adequado para as aplicações tradicionais, as quais podem necessitar de um certo nível de banda, porém não são muito sensíveis a atrasos.

Com a crescente demanda por utilização de aplicações multimídia, tais como voz e vídeo, e principalmente nas que exigem comunicações interativas, ocorrendo em tempo real, estes novos parâmetros (atraso, variação do atraso e perda) passam a ser importantes para o bom funcionamento destas aplicações. Sendo assim, torna-se necessária o provimento de QoS (*Quality of Service* – Qualidade de Serviço) pela rede para suportar estas novas aplicações.

Nos entroncamentos de uma rede, ou seja, em seus roteadores, quando vários fluxos compartilham a saída de um enlace, se fazem necessários critérios para priorização dos tráfegos e também, nos casos de congestionamento, critérios para o descarte de pacotes. Estes critérios implementados através de mecanismos de priorização e de controle de congestionamento, possibilitam estabelecer um tratamento diferenciado às necessidades dos diferentes tipos de tráfego.

O estabelecimento de um canal dedicado para a transmissão ou a utilização de protocolos que reservem recursos fim-a-fim para cada aplicação possibilitaria uma garantia de qualidade de serviço, porém sua implementação seria possível para redes corporativas ou privadas. Já sua implementação em um ambiente como a Internet seria, se possível, de enorme dificuldade.

Para possibilitar a adequação da Internet em termos de QoS, com um custo reduzido e certa facilidade e rapidez de implementação, tem sido estudados e propostos inúmeros mecanismos, os quais proporcionam capacidade de diferenciação entre os tráfegos, de forma a fornecer a qualidade de serviço necessária e adequada a cada aplicação que compartilhe a Internet ou qualquer outra rede IP.

Esta dissertação pretende verificar a eficiência da utilização dos mecanismos de QoS aqui estudados, mostrando a viabilidade de sua utilização e implementação em uma rede "*best effort*". Dentro deste propósito, este trabalho apresenta o resultado de inúmeras simulações, onde fica comprovado que, com a utilização adequada dos mecanismos QoS de priorização e controle de congestionamento, pode-se viabilizar a utilização dos aplicativos de VoIP (*Voice over IP*) dentro dos limites aceitáveis de atraso e perda, mesmo quando o tráfego de voz compartilha um enlace de comunicação com outros tráfegos em situações de extremo congestionamento.

Esta dissertação contém mais quatro capítulos. No próximo capítulo apresenta-se a definição de QoS e de seus principais parâmetros (atraso, variação do atraso, perdas e vazão). No capítulo 3, estudam-se os mecanismos de priorização e no capítulo 4, os

mecanismos de controle de congestionamento. O capítulo 5 traz as simulações realizadas, onde será possível verificar o comportamento dos tráfegos de voz, HTTP e FTP, quando submetidos aos mecanismos de priorização e congestionamento estudados, para atender as necessidades do ambiente montado.

2. QoS - Qualidade de Serviço - Definições e Parâmetros

O termo Qualidade de Serviço (*Quality of Service* – QoS) foi utilizado no modelo OSI (*Open Systems Interconnection*) há aproximadamente vinte anos atrás, e seu significado foi associado à habilidade do provedor de serviço (uma operadora de rede) de suportar os requisitos das aplicações dos usuários em relação à pelo menos quatro parâmetros: largura de banda, latência (atraso), variação do atraso (*jitter*) e taxa de perda de tráfego [1].

Qualidade de Serviço é definido na recomendação E.800 do ITU-T (*International Telecommunications Union – Telecommunication Standardization Sector*) como sendo o efeito conjunto do desempenho do serviço que determina o grau de satisfação de um usuário de um serviço [2].

O objetivo da QoS é fornecer um caminho de comunicação eficiente para transmissão de dados com valores aceitáveis de atraso, variação do atraso, banda e taxa de erro, além de garantir a disponibilidade destes recursos [3].

Uma forte tendência de mercado mundial atual aponta para a integração de comunicação de voz e dados em uma única rede. Existem duas razões principais para esta tendência: a eliminação do alto custo de operação e manutenção de duas redes distintas e o suporte a novas aplicações multimídias, as quais tipicamente permitiriam aos usuários falar e enviar dados e imagens em uma única sessão [4].

A Internet tem mudado significativamente tanto em suas características qualitativas

quanto quantitativas, estas principalmente na quantidade de usuários e na capacidade de transmissão. Em particular, serviços de comunicação multimídia em tempo real sob a Internet requerem pesquisas em seus aspectos qualitativos e novas arquiteturas para o provisionamento de QoS [5].

A telefonia via Internet é um serviço promissor, pois os serviços de comunicação de voz tradicionais são mundialmente difundidos e poderiam migrar para a Internet [5].

VoIP (*Voice over* IP), uma tecnologia de transmissão de dados de voz em uma rede IP , tem recebido atenção de um grande número de empresas e organizações devido, principalmente, a duas razões: primeiramente, sua eficiência, pois permite a transmissão de voz em conjunto com outros serviços; em segundo lugar, o barateamento das ligações telefônicas de longa distância. Contudo, VoIP enfrenta desafios, necessitando de padronização e garantia de QoS aplicado ao tráfego de voz. Desta maneira, o provisionamento de maior QoS para o tráfego de voz na rede IP é o problema a ser resolvido. Os clientes de voz esperam encontrar na rede IP o atendimento aos requisitos estritos de QoS em termos de atraso fim-a-fim, variação de atraso e perdas [6].

Nos itens a seguir são apresentadas as definições e principais características dos quatro principais parâmetros que influenciam a QoS das aplicações convergentes: atraso ou latência, variação do atraso, perdas e largura de banda ou vazão.

2.1. Atraso ou Latência

Atraso ou latência é o tempo necessário para um pacote ser transmitido de um ponto na rede a outro ponto na rede [7].

Em termos de voz, o atraso é definido como o tempo total entre o momento em que o som da voz é produzido em um ponto pelo locutor até o momento em que é ouvida no outro ponto pelo receptor [8] [9]. Esta latência inclui o atraso de codificação na fonte, o atraso através da rede e o atraso na decodificação no destino [9].

Vários fatores contribuem para o atraso na rede. Este atraso pode ser calculado pela soma individual dos seguintes fatores de atraso: encaminhamento, enfileiramento, propagação e transmissão [7]. Vale lembrar que além destes fatores, associados à contribuição da rede (roteadores e enlaces) ao atraso, também devem ser consideradas as contribuições dos fatores associados aos *hosts* de origem e destino dos pacotes: a atuação do sistema operacional (*scheduling* e encapsulamento), as características da aplicação e, no caso das aplicações de voz, o atraso devido a codificação, compressão e empacotamento associados ao CODEC em utilização. A qualidade das diferentes implementações de TCP/IP nos *hosts* também influencia no atraso.

- **Atraso de encaminhamento**

O atraso de encaminhamento é a quantidade de tempo que o roteador gasta para receber um pacote, fazer a decisão de encaminhamento e então começar a transmiti-lo através de uma saída livre.

- **Atraso de enfileiramento**

O atraso de enfileiramento é a quantidade de tempo que o pacote tem de aguardar em uma fila enquanto outros pacotes estão sendo servidos em uma porta de saída de um roteador. O atraso de enfileiramento em um determinado roteador pode variar enormemente, dependendo do nível de carga no enlace de saída. Durante os períodos de congestionamento, a utilização dos mecanismos de priorização e de controle de congestionamento permitem um controle da quantidade de atraso no enfileiramento experimentado por diferentes classes de tráfego localizados nas filas de um roteador.

- **Atraso de propagação**

O atraso de propagação é a quantidade de tempo que o sinal leva para atravessar um enlace. O atraso de propagação depende da distância entre as extremidades do enlace e da velocidade de propagação no meio. Em qualquer transmissão sempre existirá o atraso de propagação, porém este só será significativo para grandes distâncias [8].

- **Atraso de Transmissão**

O atraso de transmissão é a quantidade de tempo gasto para a transmissão dos bits de um pacote de um nó da rede (roteador) para outro. O atraso na transmissão é função do tamanho do pacote a da velocidade da porta [7]. Desta forma, o atraso na transmissão é inversamente proporcional à velocidade do enlace [8].

Efeitos do atraso em um sistema VoIP

Conforme visto, atrasos são introduzidos em todos os estágios de um sistema VoIP. Um usuário não se preocupa com o local onde a latência foi introduzida, sua exigência é ter baixa latência para experimentar um contato verdadeiramente interativo na comunicação interpessoal.

O atraso afeta a qualidade da conversação independente da fidelidade do sinal de voz recebido no destino. De acordo com o ITU, os usuários não perceberão atrasos abaixo de 100-150 ms. Atrasos entre 150 e 300 ms são percebidos como uma leve hesitação pelo parceiro na conversação. Atrasos acima de 300 ms são claramente percebidos pelos usuários, e a conversação pode se tornar até impossível. O problema de sobreposição de falas se torna significativo se o atraso em uma direção se torna maior que 250 ms. O ITU-TG.114 recomenda um atraso máximo de 150 ms para se obter uma conversação de alta qualidade [10].

2.2. Variação do Atraso (*Jitter*)

Jitter é a variação do atraso experimentado por pacotes consecutivos que são parte de um mesmo fluxo [7].

A medida do *jitter* está baseada na diferença do tempo de chegada esperado para um pacote e seu tempo real de chegada. Por exemplo, a Figura 2.1 mostra que os pacotes P1 e P3 chegaram no tempo previsto, porém os pacotes P2 e P4 apresentaram um atraso de 12 ms e 5 ms respectivamente [8].

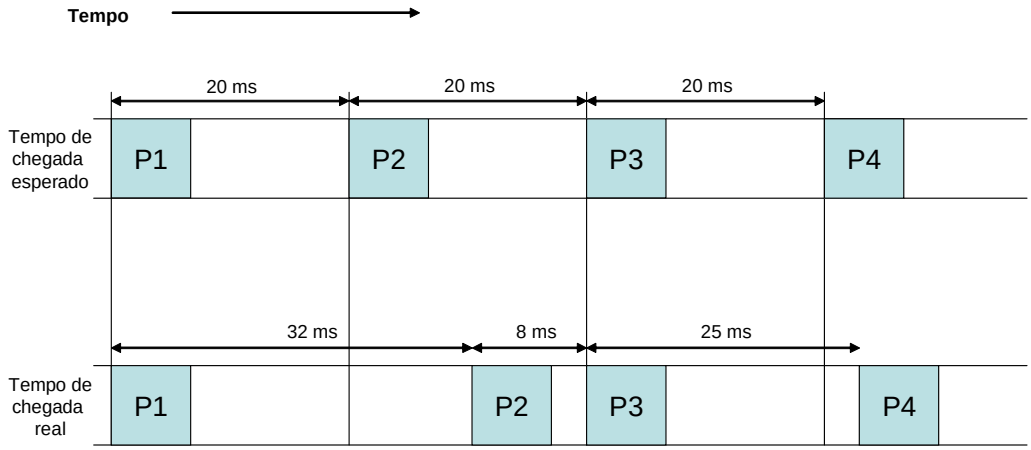


Figura 2.1 – Exemplo do Jitter.

A principal causa do *jitter* são as variações dos tempos de fila nos roteadores, ou seja, variação no atraso no enfileiramento, devido às mudanças dinâmicas do tráfego de rede. Outra possível causa é a utilização, por uma conexão, de enlaces de custo equivalente, porém com distância físicas ou elétricas diferentes [7].

Conforme a RFC 3550 [11], a qual obsoletou a RFC 1889 [12], para uma rede IP utilizando o protocolo RTP (*Real-time Transport Protocol*) para transporte de voz, o *jitter* é definido como uma estimativa da variação estatística do tempo entre chegadas dos pacotes de dados RTP, medido em unidades de *timestamp*. O *jitter J* é definido como sendo a variação média da diferença *D* entre o recebimento e envio de um pacote, entre um par de pacotes. Como apresentado na Equação 2.1, *D* é equivalente à diferença no tempo de trânsito relativo (*relativ transit time*) para os dois pacotes. O tempo de trânsito relativo é a diferença entre o *timestamp* do pacote RTP e o relógio do receptor.

Sendo S_i o *timestamp* RTP para o pacote *i*, e R_i o tempo de chegada em unidades de *timestamp* RTP para este pacote *i*, então para dois pacotes *i* e *j*, *D* pode ser expresso

como:

$$D(i,j) = (R_j - R_i) - (S_j - S_i) = (R_j - S_j) - (R_i - S_i) \quad (2.1)$$

Recomenda-se (SHOULD – conforme definido na RFC 2119 [13]) que o *jitter* seja calculado continuamente para cada pacote *i*, usando esta diferença *D* entre o pacote corrente (*i*) e o pacote anterior (*i-1*) em ordem de chegada (não necessariamente em seqüência), de acordo com a Equação 2.2.

$$J(i) = J(i-1) + (|D(i-1,i)| - J(i-1))/16 \quad (2.2)$$

O cálculo do *jitter* deve (MUST – conforme definido na RFC 2119) estar em conformidade com a Equação 2.2 para permitir seu monitoramento independente do perfil e tornar válida a interpretação de relatórios vindos de diferentes implementações.

A Equação 2.2 representa um estimador empírico com parâmetro de ganho 1/16, o que segundo a RFC dá uma boa redução da taxa de ruído (ocasionado por picos de *jitter*), enquanto mantém uma taxa razoável de convergência.

Algumas aplicações interativas, tais como voz e vídeo, são incapazes de lidar com o *jitter*, pois este resulta em "trancos" ou numa qualidade irregular para o som ou a imagem. A solução está no provisionamento adequado pela rede de mecanismos, tais como os mecanismos de priorização, que condicionem o *jitter* a níveis adequados. O *jitter* remanescente pode ser tratado com *dejitter buffers* no *host* de destino, que armazenam os pacotes brevemente antes de encaminhá-los de maneira adequada, sem *jitter* [7].

Dejitter Buffer

O *dejitter buffer* é utilizado para remover os efeitos do *jitter* do fluxo de voz, através do uso de um *buffer* de recepção, adiando com ele o "ponto de reprodução" do fluxo recebido [14]. Este mecanismo retira os efeitos do *jitter*, adicionando atraso e eventual perda de pacotes (descartando pacotes com atraso excessivo).

A Figura 2.2 ilustra um sistema onde o *dejitter buffer* introduz atrasos adicionais de modo que todo pacote recebido tenha um atraso total de 120 ms. O pacote 1, que sofreu atraso de 100 ms na rede, sofre um atraso adicional de 20 ms, e o pacote 2, que sofreu um atraso de 90 ms na rede, sofre um atraso adicional de 30 ms [15].

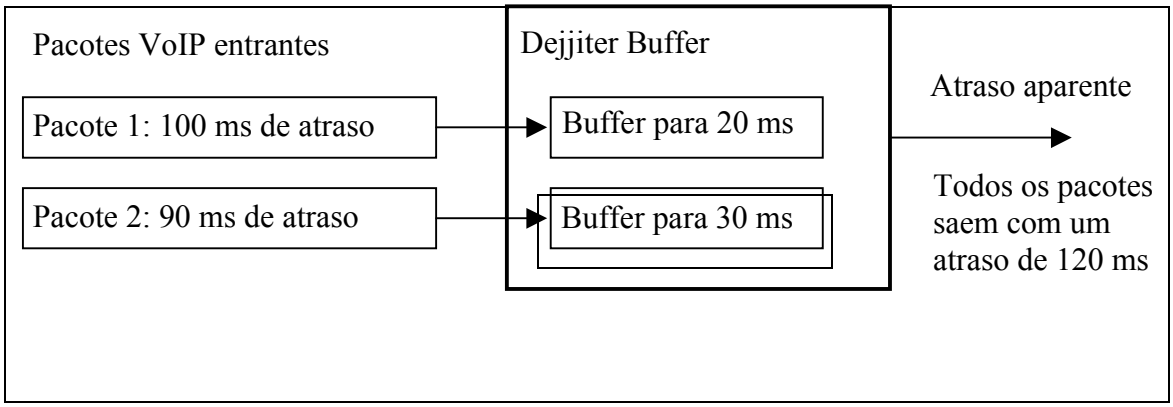


Figura 2.2 – Exemplo do comportamento estatístico do atraso na rede.

A escolha do atraso para o *dejitter buffer* é crítica. Para que o *dejitter buffer* seja capaz de eliminar completamente o efeito do *jitter*, sua janela de atraso deve ser igual à diferença entre o máximo e o mínimo atraso na rede. Por exemplo, se o mínimo atraso na rede é de 70 ms e o máximo atraso é de 130 ms, com um atraso nominal de 100 ms, um *dejitter buffer* com janela de atraso de 60 ms (os atrasos introduzidos pelo buffer são entre 0 e 60 ms) é capaz de eliminar completamente o *jitter*, fazendo com que todos os

pacotes recebidos tenham atrasos idênticos de 130 ms [15]. Este princípio está ilustrado na Figura 2.3, que mostra um exemplo de distribuição estatística dos atrasos na rede e o atraso total sofrido por todos os pacotes após o dejitter buffer [16].

O *dejitter buffer* pode ser fixo, mantendo tamanho constante, ou adaptativo, que tem a capacidade de ajustar seu tamanho dinamicamente com a finalidade de otimizar a relação atraso / descarte de pacotes.

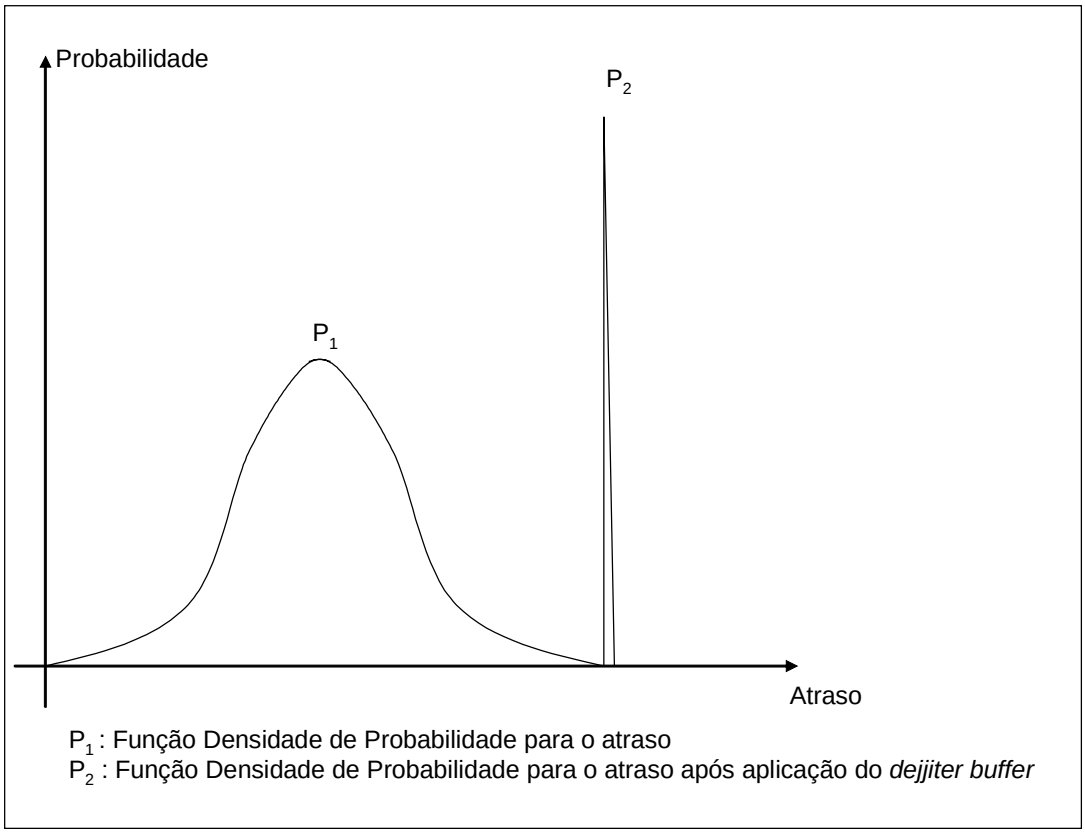


Figura 2.3 – Exemplo do comportamento estatístico do atraso na rede.

2.3. Perdas

Existem três causas associadas à perda de pacotes em redes IP [7]:

- Um problema no enlace físico que impeça a transmissão do pacote.
- Um pacote que é corrompido por ruído e é detectado por falha de *checksum* em um nó até seu destino.
- Um congestionamento na rede que conduz a um *buffer overflow*.

Problemas nos enlaces físicos podem ocorrer, mas são raros, e a combinação de autocorreção das camadas físicas e topologias redundantes respondem dinamicamente a esta causa de perda de pacotes. Com exceção das redes sem fio, a chance de corrupção de pacotes é estatisticamente insignificante. Desta forma, esta causa de perda de pacotes também pode ser ignorada. Conseqüentemente, a principal razão para a perda de pacotes em uma rede IP com fio é o *buffer overflow* resultante de um congestionamento [7].

Perda de pacotes para aplicação que não são de tempo real, tal como *web* e transferência de arquivos, é indesejável, porém os protocolos utilizados por estes tipos de aplicações, geralmente o TCP, são tolerantes a uma certa quantidade de perda de pacotes devido à sua capacidade de retransmissão.

Aplicações de tempo real são baseadas no UDP. O UDP não apresenta as facilidades de retransmissão e, além disto, as retransmissões quase nunca ajudariam. Em uma sessão RTP, o pacote de voz retransmitido poderia chegar muito tarde [8].

As perdas de pacotes de voz serão percebidas como "*gaps*" na conversação, degradando a qualidade de serviço. Contudo, uma certa percentagem de perda de pacotes, entre 3 e 5% [17], pode ser compensada por esquemas de recuperação dos Codecs. Por exemplo, o G.723.1 interpola um quadro perdido simulando as características vocais do quadro

anterior e atenuando suavemente a perda no sinal [18]. A taxa máxima tolerável de perda de pacotes é usualmente fixada em 10% [17] [18] [19]. A Figura 2.4 ilustra a relação do atraso e taxa de perda de pacotes com a qualidade de serviço na rede. Quatro níveis de QoS são definidos na figura: *Tool quality* (desejável), *Good quality* (boa), *Potentially useful quality* (útil) e *Poor quality* (inaceitável) [15] [18].

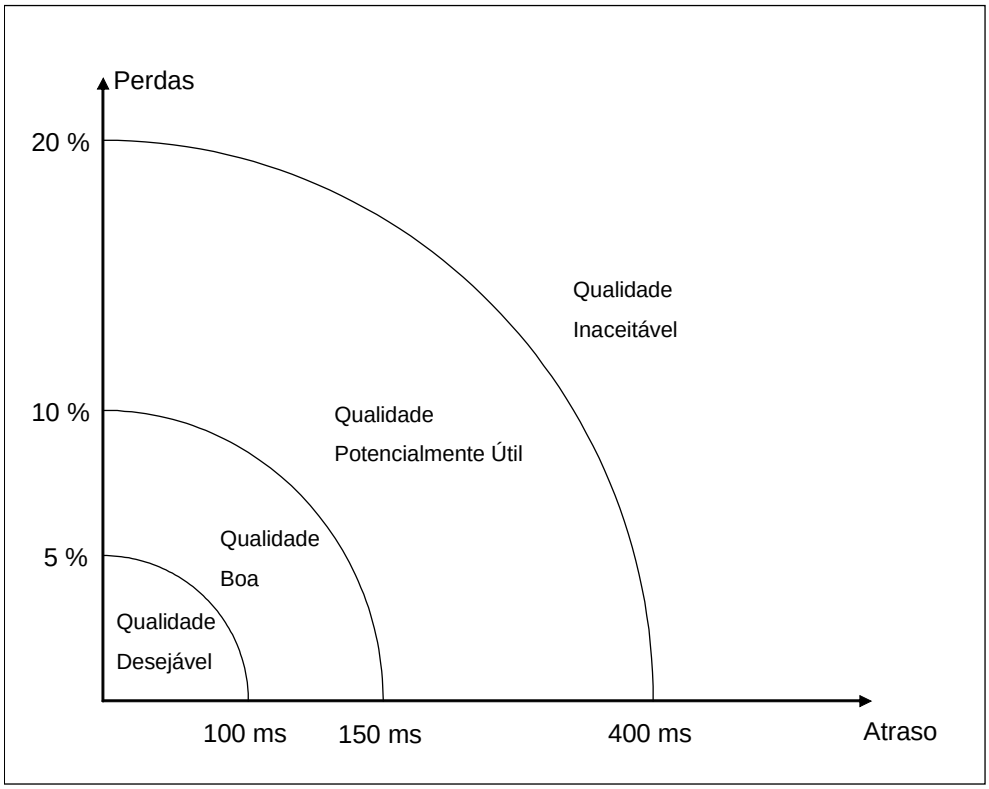


Figura 2.4 – Relação entre atraso, perda de pacotes e QoS.

2.4. Largura de Banda

Largura de banda (*bandwidth*) é o termo genérico utilizado para descrever a capacidade de um sistema para a transferência de dados.

Um dos principais atributos de um enlace de transmissão de dados é sua capacidade

nominal de transportar bits, ou banda nominal**.** A vazão (*throughput*) ou a banda de rede ocupada pela mídia corresponde à taxa de bits que é efetivamente transportada, tendo como valor máximo a banda nominal [14].

Um nível inadequado de vazão para uma determinada aplicação pode causar atraso e perda de pacotes. Uma vez que o tráfego na rede IP é estatístico, pacotes irão sempre sofrer atrasos se não houver algum tipo de priorização [20].

2.5. Características das Redes Multimídias e seus Parâmetros

A Internet comporta uma grande variedade de aplicações multimídias. Consideram-se três classes abrangentes de aplicações de multimídia: áudio e vídeo de fluxo contínuo armazenado, áudio e vídeo de fluxo contínuo ao vivo e áudio e vídeo interativos em tempo real [21].

A classe de áudio e vídeo interativos em tempo real permite que os usuários usem áudio / vídeo para se comunicar entre si em tempo real. Áudio interativo em tempo real pela Internet é denominado telefone por Internet ou telefonia IP, já que, da perspectiva do usuário, é semelhante ao tradicional serviço telefônico por comutação de circuitos.

Voz e vídeo interativos em tempo real impõem rígidas limitações ao atraso de pacote e à variação de atraso do pacote.

O projeto de aplicações de multimídia seria certamente menos complicado se houvesse algum tipo de classificação dos serviços na Internet em primeira e segunda classe, pela

qual pacotes de primeira classe teriam número limitado e receberiam serviço prioritário em filas de roteadores. Esse serviço de primeira classe poderia ser satisfatório para aplicações sensíveis ao atraso. Mas, até agora, a Internet adotou preferencialmente uma postura igualitária em relação ao escalonamento de pacotes em filas de roteadores. Todos os pacotes recebem serviço igual; nenhum pacote, incluindo os pacotes de áudio e vídeo, sensíveis ao atraso, recebe prioridade especial nas filas de roteadores [21].

Os roteadores têm inteligência para tratar o encaminhamento dos fluxos que passam por eles baseado em algum tipo de teoria das filas. Os pacotes chegam aos roteadores para serem servidos e os mecanismos de priorização e controle de congestionamento focam no processamento destas filas [22].

Segundo [23] [24] [25] e [26], as combinações de componentes como mecanismos de priorização, controle de congestionamento e regulação de tráfego, podem ser utilizadas como partes de uma arquitetura de serviços baseada em classes provendo forte diferenciação entre os tráfegos e implementados com baixa complexidade.

Nos capítulos seguintes, 3 e 4, são estudados os mecanismos de priorização e controle de congestionamento, os quais se apresentam como alternativas para proporcionar os níveis adequados de atraso e perda necessários para o tráfego das aplicações multimídia na Internet.

3. Mecanismos de Priorização

A função básica de um algoritmo de priorização é selecionar, entre os pacotes aptos a serem transmitidos, para cada enlace de saída de um roteador, o pacote a ser transmitido no próximo ciclo. Os algoritmos de priorização são implementados em um mecanismo no qual várias entradas são armazenadas e existe um único servidor. A arquitetura básica é ilustrada na Figura 3.1 [27].

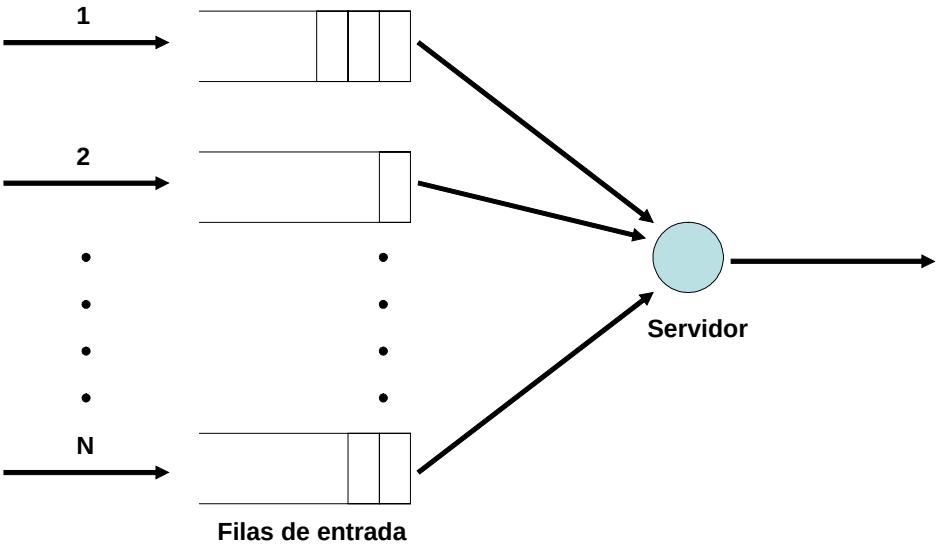


Figura 3.1 – Arquitetura de escalonamento

Devido às diferentes características e necessidades de QoS dos tráfegos em uma rede (principalmente com a coexistência de voz, vídeo e dados na rede), fica reforçada a importância dos mecanismos de priorização e compartilhamento de banda. Para que se possa fazer a priorização de determinados tráfegos, faz-se necessário prover mecanismos de classificação do tráfego.

Os tráfegos de dados, voz e vídeo possuem características distintas. O tráfego de dados,

geralmente, é gerado em rajadas, é relativamente pouco sensível a atraso, mas é sensível a perdas. Por outro lado, tráfegos em tempo real, como vídeo e voz, são sensíveis a atraso, mas são mais tolerantes a perdas. Em geral o tráfego de voz não é gerado em rajadas e necessita de pouca banda, ao contrário de vídeo, que consome muita banda e geralmente tem o tráfego se comportando em rajadas. Essa diversidade nas características dos tráfegos e conseqüentemente suas diferenças na necessidade de QoS indica que os tráfegos devem ser tratados de maneira diferenciada e divididos em classes que reflitam estes atributos [28].

3.1. Classificação do Tráfego

Para a atribuição de prioridade aos tráfegos, estes necessitam ser identificados e marcados. Estas duas tarefas são geralmente referenciadas como classificação do tráfego [29].

Utilizando a classificação de pacotes, pode-se particionar os tráfegos de uma rede em múltiplos níveis de prioridade ou classes de serviço [30]. Para realizar a classificação do tráfego, cada pacote que chega ao roteador de classificação é mapeado para uma classe específica. Todos os pacotes que forem classificados em uma mesma classe terão o mesmo tratamento, a ser dado pelo algoritmo de priorização.

Existem várias possibilidades de classificação de um determinado tráfego. A atribuição pelo classificador da classe para o tráfego pode estar associada a inúmeros aspectos, porém alguns dos métodos mais comuns consistem na identificação e classificação do tráfego por: [31]

- Protocolos da camada de rede ou transporte, tais como IP, TCP, UDP, IPX;
- Número da porta de origem;
- Número da porta de destino;
- Endereço IP de origem;
- Endereço IP de destino;
- Interface de chegada do tráfego;
- Fluxo, ou seja, uma combinação do endereço IP de origem e destino com o número da porta de origem e destino.

A utilização do campo TOS (*Type of Service* – Tipo de Serviço) do protocolo IPv4.

O octeto *Type of Service* (TOS) no cabeçalho do protocolo IPv4 foi designado para prover QoS através da identificação dos datagramas IP com as diferentes características de serviços. Estas características de serviços indicam como os nós da rede (roteadores), lendo o cabeçalho do datagrama, devem tratar o mesmo [32].

O octeto *Type of Service* consiste de 3 (três) campos, a saber: *Precedence*, TOS (*Type of Service*) e MBZ (*must be zero*), conforme ilustrado na Figura 3.2.

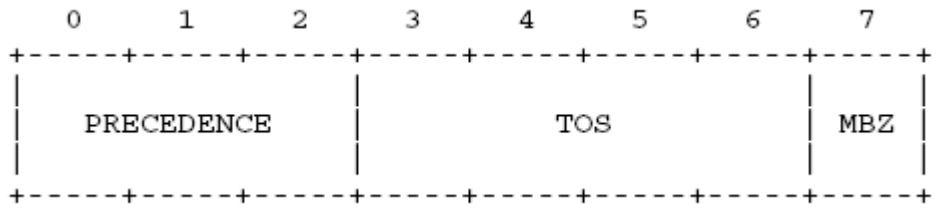


Figura 3.2 – O octeto *Type of Service*

As RFCs 791 [33], 1349 [34] e 1700 [35] especificam os campos do octeto *Type of*

Service do cabeçalho IP.

A utilização do campo *Precedence* (Preferência) permite a especificação da classe de serviço (*Class of Service* - CoS) para um pacote. São usados os 3 (três) bits do octeto *Type of Service* para este propósito. Os 8 (oito) possíveis níveis de preferência definidos pela RFC 791 [33] são apresentados na Tabela 3.1.

Nível	**Tipo de Tráfego**
111 (7)	*Network Control*
110 (6)	*Internetwork Control*
101 (5)	*Critical / ECP*
100 (4)	*Flash Override*
011 (3)	*Flash*
010 (2)	*Immediate*
001 (1)	*Priority*
000 (0)	*Routine*

Tabela 3.1 – Definição dos níveis de preferência do campo *Precedence*

Os níveis 110 (6) e 111 (7) são reservados para roteamento e algumas mensagens de ICMP (*Internet Control Message Protocol*). Então apenas seis níveis (de rotina a crítico) permaneceram para as aplicações de usuário [36].

Os 4 (quatro) bits seguintes do byte *Type of Service* formam o campo TOS e podem ser usados para marcar uma propriedade a ser priorizada entre custo, atraso, processamento, confiabilidade e segurança, conforme definido na RFC 1700 [35] e apresentados na

Tabela 3.2.

Valor TOS	Descrição	Referência
0000	Padrão	RFC 1349 [34]
0001	Minimiza custo	RFC 1349 [34]
0010	Maximiza confiabilidade	RFC 1349 [34]
0100	Maximiza processamento	RFC 1349 [34]
1000	Minimiza atraso	RFC 1349 [34]
1111	Maximiza segurança	RFC 1455 [37]

Tabela 3.2 – Definição dos valores TOS

Somente um valor TOS ou propriedade a ser priorizada pode ser requisitado em um datagrama IP [35].

O último campo, MBZ (*must be zero* – deve ser zero) não é utilizado. O emissor do datagrama seta este campo para zero. Os roteadores e os receptores do datagrama ignoram o valor deste campo.

Posteriormente a RFC 1812 [38] manteve o campo IP *Precedence* do octeto TOS e, devido a falta de uso, redefiniu os 5 demais bits como não utilizados [39].

Definição do Campo DSCP – *Differentiated Services Code Point*

O modelo Diffserv ou DS *(Differentiated Services)* define, através da RFC 2474 [40], o campo DSCP (*Differentiated Services Code Point*) no octeto *Type of Service* (TOS) no cabeçalho do protocolo IPv4 (Figura 3.3). O DSCP estende o número de priorizações

possíveis, provendo 3 bits extras. O total de 6 bits permite 64 diferentes classes de serviço.

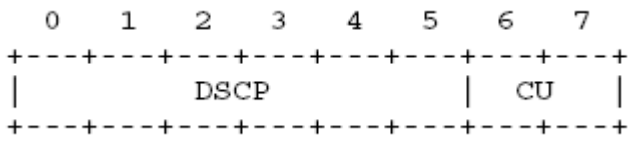


Figura 3.3 – O octeto *Type of Service*

O modelo Diffserv mantém a compatibilidade entre os mecanismos. Desta forma, os valores de 3 bits do *IP Precedence* e os 6 bits do DSCP podem coexistir na mesma rede. Lembrando que ambos os valores residem no octeto TOS, o valor do IP *Precedence* representa o seletor de classes dentro do DSCP [39].

O campo CU (*Currently Unused*) do octeto TOS, que foi definido como não utilizado é definido na RFC 3168 [41] para utilização do ECN (*Explicit Congestion Notification*), cuja utilização será detalhada na seção 4.6.

3.2. Algoritmos de Encaminhamento dos Pacotes

3.2.1. *First In First Out* (FIFO)

O algoritmo de enfileiramento FIFO, também chamado FCFS (*First Come, First Served* – primeiro a chegar, primeiro atendido), simplesmente transmite os pacotes na ordem em que foram recebidos [36].

O algoritmo FIFO não introduz nenhum conceito de priorização de classes de tráfego e conseqüentemente não faz o encaminhamento dos pacotes com diferentes prioridades [30]. Os pacotes são armazenados em uma única fila e são atendidos na mesma ordem de chegada à fila. Sob o mecanismo FIFO, todos os pacotes são tratados de forma igualitária [42].

O algoritmo FIFO é o mais simples algoritmo para o escalonamento de fluxos em uma rede. A simplicidade deste algoritmo traz, em contrapartida, grandes limitações, pois não existe a possibilidade de proteger ou privilegiar um fluxo em relação aos outros. A chegada de uma rajada de pacotes de um determinado fluxo afetará o desempenho de todos os demais fluxos que estejam compartilhando a rede. Desta forma, em uma rede com tráfego em rajadas, o algoritmo FIFO não será capaz de fornecer garantias de banda e atraso fim a fim para nenhum fluxo [43]. No algoritmo FIFO não existe nenhum mecanismo de QoS implementado, uma vez que inexiste neste algoritmo a possibilidade de um fluxo "furar" a fila. O tráfego de alta prioridade tem o mesmo tratamento que o de baixa prioridade [22].

3.2.2. Filas com Prioridades (*Priority Queuing* – PQ)

No algoritmo PQ existe a possibilidade de priorização de tráfegos. Um multiplexador no enlace de saída mantém uma fila distinta para cada prioridade de tráfego definida e somente encaminha um pacote de uma fila de menor prioridade se todas as filas de maior prioridade se encontrarem vazias. Cada fila que alimenta o multiplexador é tratada como uma fila FIFO [30] [44].

O algoritmo PQ provê tratamento absolutamente preferencial ao tráfego de maior prioridade. Desta forma, um tráfego de menor prioridade sempre terá a banda negada quando houver a presença de pacotes de um tráfego de maior prioridade. Esta característica do algoritmo PQ pode causar um fenômeno conhecido como *starvation*, ou seja, um tráfego de menor prioridade poderá nunca ser enviado em função da presença constante de tráfegos de maior prioridade [30].

A implementação do PQ envolve, em geral, a criação de um número pequeno de filas de prioridades. Em um exemplo de implementação, da CISCO, cada pacote é posicionado em uma das quatro filas criadas: alta, média, normal e baixa, de acordo com a classificação que lhe foi atribuída. Caso um pacote não tenha recebido uma classificação este será posicionado na fila normal [30].

Em uma rede que possua banda suficiente para a aplicação VoIP e que seja importante priorizar o tráfego de VoIP em relação às demais aplicações, pode-se classificar exclusivamente a aplicação VoIP como alta e as demais aplicações em prioridades inferiores. Esta é uma boa maneira de se garantir uma boa qualidade de transmissão de voz, mesmo sob uma carga pesada das demais aplicações [22]. Porém, considerando uma configuração mais global de QoS, onde não estaria envolvido somente o gerenciamento de QoS em relação a VoIP, este algoritmo se mostra pobre na possibilidade de se atribuir parcelas da banda para cada tráfego disponível, uma vez que o algoritmo provê tratamento preferencial absoluto às filas de alta prioridade em relação às filas de baixa prioridade.

3.2.3. Filas Customizadas (*Custom Queuing* – CQ)

O algoritmo CQ permite especificar uma percentagem da banda para uma determinada aplicação. A banda reservada é compartilhada proporcionalmente, no percentual pré-definido, entre as aplicações.

O algoritmo CQ controla o tráfego, alocando uma determinada parte da banda para cada fluxo classificado. As filas são atendidas ciclicamente num esquema *round-robin*, onde, para cada fila, é enviado a quantidade de pacotes referente à parte da banda alocada, antes de passar para a fila seguinte. Se uma fila está vazia, o algoritmo buscará pacotes na próxima fila que contenha pacotes prontos a serem enviados. Associado a cada fila, há um contador configurável que estabelece quantos bytes devem ser enviados antes de passar para a próxima fila. [45].

O roteador enviará os pacotes de uma determinada fila até que a quantidade de bytes configurada no contador seja atingida. Neste momento, o pacote que está sendo transmitido terá sua transmissão completada antes do algoritmo buscar pacotes na próxima fila.

O tamanho dos pacotes varia conforme a aplicação que está sendo transportada. Devido ao fato do pacote ser a unidade mínima tratada pelo rotetador e do contador estar configurado em quantidade de bytes, tem-se uma dificuldade para atender com precisão ao percentual definido para cada fila.

Por exemplo, considere-se que um protocolo apresente pacotes de tamanho igual a 500

bytes, outro protocolo pacotes de 300 bytes e um terceiro pacotes de 100 bytes. Se for configurado o contador para as três filas com o valor de 200 bytes, buscando dividir a banda em três partes iguais, onde cada aplicação utilize 1/3 da banda disponível, o algoritmo CQ não conseguirá atender esta configuração pois quando o roteador atender a primeira fila, este enviará um único pacote de 500 bytes; quando atender a segunda fila, enviará um único pacote de 300 bytes; e ao atender a terceira fila, enviará 2 pacotes de 100 bytes. Com esta configuração, a real divisão da banda entre as três aplicações será 50/30/20, ao invés de 33/33/33 conforme pretendido.

Pode-se concluir que configurar os contadores com valores baixos poderá resultar em uma grande discrepância em relação às bandas alocadas para cada aplicação. Se os contadores forem configurados com valores altos a discrepância diminuirá, no entanto, após uma fila ser atendida, esta demorará a ser atendida novamente, causando atraso às aplicações. Quanto maior o valor do tamanho do contador, menor será a discrepância em relação às bandas que se quer alocar para cada aplicação e maior será o atraso inserido para cada aplicação.

3.2.4. *Weighted Fair Queuing* (WFQ)

O mecanismo de *Fair Queueing* requer que cada conexão seja alocada a uma fila e que estas filas sejam servidas em um esquema *round robin*. Filas vazias são saltadas, e para um certo número de conexões ativas, a banda é igualmente dividida entre elas [46].

O conceito de justiça está associado à maneira pela qual a banda é compartilhada pelos fluxos que competem pelo enlace de saída [28].

A noção de justiça pode ter muitas interpretações. Pode-se considerar a idéia de que todos os fluxos devem ser processados (ou servidos) da mesma maneira ou de que o serviço deve ser proporcional à prioridade do fluxo [36].

O algoritmo WFQ – *Weighted Fair Queueing* (Enfileiramento Justo Ponderado) está baseado em um modelo referencial chamado GPS (*Generalized Processor Sharing)*, uma analogia com o compartilhamento de CPU entre os diversos processos em um sistema operacional multitarefa. O funcionamento do GPS pode ser definido por um modelo de fluido [44].

No GPS, considera-se que os pacotes podem ser fragmentados arbitrariamente, o que é chamado de aproximação de fluido. Nesta política, a capacidade de processamento do roteador e a largura de banda disponível são distribuídas entre os fluxos concorrentes. Cada um desses fluxos recebe pelo menos uma parte Φ_i desses recursos, sendo:

$$\sum_{i=1}^{N} \phi_i = 1 \qquad (3.1)$$

Esta disciplina pode ser visualizada imaginando um fluxo contínuo de dados, separado por classes de prioridades, sendo cada classe *i* servida de acordo com o peso Φ_i [47].

A Figura 3.4 representa o modelo GPS [36]. Nessa figura pode-se observar o recebimento de três pacotes de três fluxos distintos (A, B e C) com seus respectivos pesos (25%, 25% e 50%). Enquanto o fluxo A não está compartilhando o enlace de saída com nenhum outro fluxo, este utiliza todo o enlace (100%). Quando os fluxos A e B compartilham o enlace, eles utilizam todo o enlace na proporção de seus pesos, os

quais são iguais, e desta forma, cada um utiliza 50% do enlace. Quando o enlace é compartilhado pelos 3 fluxos, estes o compartilham na proporção de seus pesos. Dentro do modelo GPS, o pacote C, apesar de ser o último a chegar, seria o primeiro a ter seu envio finalizado. Isto acontece porque o modelo GPS utiliza a aproximação de fluido e desta forma consegue implementar justiça perfeita na alocação de banda.

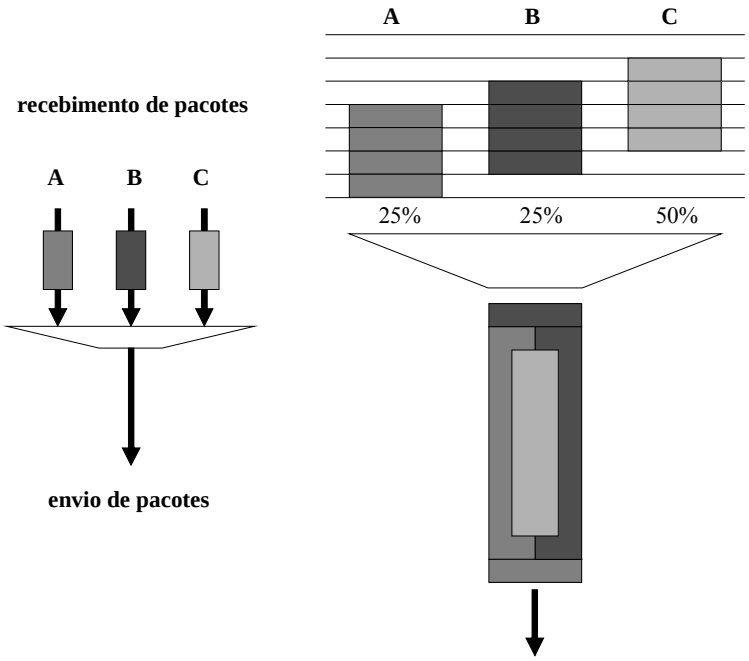


Figura 3.4 – GPS e a aproximação de fluido

Cada sessão *i* pode usar no mínimo a capacidade $\Phi_i * R$, onde *R* é a capacidade total da interface de saída considerada. As sessões ativas compartilham os recursos das sessões inativas proporcionalmente aos seus Φ_i.

Suponha-se que existam *N* fluxos sendo servidos por um roteador a uma taxa *R* e que ao i-ésimo fluxo *i* seja atribuído o peso Φ_i, e considere-se também que $W_i(\tau, t)$ seja a quantidade de dados servido ao fluxo *i* durante o intervalo (τ, t). No GPS, para qualquer fluxo *i* na fila do roteador (*backlogged*) e qualquer outro fluxo *j*, em (τ, t), tem-se que

$$\frac{W_i(\tau,t)}{W_j(\tau,t)} \geq \frac{\phi_i}{\phi_j} \tag{3.2}$$

No intervalo *(τ, t)*, o fluxo *i* recebe um percentual de *R* proporcional ao seu peso:

$$r_i = \frac{\phi_i}{\sum_{j=1}^{B} \phi_j} R \tag{3.3}$$

onde *B* representa o conjunto de fluxos que são armazenados na fila do roteador (*backlogged*) durante o referido intervalo.

Em um sistema GPS, todos os fluxos que utilizam uma porção da banda do enlace de saída sempre menor que a parte que lhes é atribuída são servidos sem fila, e a banda não utilizada por estes fluxos é compartilhada para os demais na proporção dos seus pesos. Dessa forma, no GPS, cada fluxo tem sua cota garantida, e a banda não utilizada é distribuída proporcionalmente pelos fluxos que estão alocados na fila.

O GPS fornece justiça perfeita na alocação de banda.

Na prática não se pode utilizar modelos de fluidos, pois os dados de cada fluxo não podem ser fragmentados arbitrariamente. O modelo de fluido por trás do GPS não é válido para redes no mundo real, nas quais os pacotes são recebidos como entidades inteiras e são inseridos na fila de saída como blocos de bits contíguos.

Para levar em consideração estas características da vida real, desenvolveu-se o GPS pacote a pacote, PGPS – *Packetized Generalized Prodessor Sharing* (Compartilhamento Generalizado do Processador por Pacotes) simulando o modelo referencial GPS. O

PGPS é também referenciado como WFQ – *Weighted Fair Queuing* [44].

O WFQ tenta emular o GPS, calculando o tempo de saída do pacote em um sistema GPS correspondente, e utilizando este tempo (*timestamp*) para ordenar o pacote. O tempo calculado para o sistema GPS não representará o tempo real de partida do pacote no WFQ, mas será associado à ordem de saída do pacote. Por exemplo, se o pacote A tem sua transmissão finalizada antes do pacote B em um sistema GPS, o pacote A será selecionado para ser transmitido antes do pacote B no WFQ.

Implementação do algoritmo *Weighed Fair Queueing*

Considere-se o enlace de saída de um roteador com capacidade total R (bits por segundo), sendo K o conjunto de fluxos que compartilham este enlace e por r_k, $k \in K$, a taxa de serviço (em bits por segundo) associado a cada fluxo k. Define-se $W_k(t_1, t_2)$, $t_2 > t_1$, como o serviço agregado (em bits) recebido pelo fluxo k durante o intervalo de tempo $[t_1, t_2]$. O valor de $W_k(t_1, t_2)$ é simplesmente igual ao tempo gasto pelo servidor com o fluxo k durante o intervalo de tempo $[t_1, t_2]$ vezes a capacidade total R.

Em qualquer momento t, um fluxo pode estar armazenado em uma fila do roteador (*backlogged*) ou ausente. Define-se $B(t)$ como o conjunto de fluxos que se encontram armazenados em t e por $B(t_1, t_2)$ o conjunto de fluxos que estejam armazenados durante todo o intervalo de tempo $[t_1, t_2]$. Para intervalos de tempo $[t_1, t_2]$, nos quais ambos os fluxos k e j estejam armazenados,

$$\frac{W_k(t_1,t_2)}{r_k} - \frac{W_j(t_1,t_2)}{r_j} = 0 \qquad (3.4)$$

Em uma situação real, onde um pacote inteiro de um certo fluxo deve ser transmitido antes do serviço ser passado para atendimento de outro fluxo, o algoritmo WFQ tem como objetivo manter a quantidade $| W_k(t_1, t_2) / r_k - W_j(t_1, t_2) / r_j |$ o mais próximo de zero possível.

No mecanismo WFQ define-se uma função de sistema chamada de *tempo virtual*. O *tempo virtual*, *v(t)*, está associado com o sistema de referência GPS e mede o progresso do trabalho neste sistema e é dependente da carga do sistema. O *tempo virtual* é utilizado no WFQ para definir a ordem na qual os pacotes são servidos no algoritmo. A função *tempo virtual* $v(t)$ tem uma taxa de crescimento no tempo igual ao serviço normalizado recebido por qualquer fluxo armazenado k no sistema de referência GPS. No GPS, todo fluxo armazenado recebe exatamente o mesmo serviço normalizado no tempo.

Matematicamente [27]

$$\frac{d}{dt}v(t) \equiv \lim \Delta t \to 0 \left[\frac{W_k(t, t+\Delta t) - W_k(t)}{\Delta t . r_k} \right] \tag{3.5}$$

Sendo assim, durante um determinado período de ocupação do mecanismo, pode-se dizer que:

$$v(t_2) - v(t_1) = \frac{W_k(t_1, t_2)}{r_k}, k \in B(t_1, t_2) \tag{3.6}$$

Onde $[t_1, t_2]$ é um subintervalo arbitrário de um período de ocupação, onde pode ser derivado:

$$v(t_2) - v(t_1) = \frac{R}{\sum_{k \in B(t_1, t_2)} r_k} . (t_2 - t_1) \tag{3.7}$$

Conseqüentemente, $v(t)$ é uma função crescente no tempo com declive que sofre mudança conforme o conjunto de fluxos armazenados, $B(t)$, muda e é inversamente proporcional a soma das sessões ativas que compartilham o enlace no tempo t:

$$\frac{d}{dt}v(t) = \frac{R}{\sum_{k \in B(t)} r_k} \tag{3.8}$$

Isto significa que quando existem sessões inativas o tempo virtual avança mais rápido. Na associação com o GPS, pode ser visto que as sessões que estão ativas obtêm mais banda.

Conforme visto, o *tempo virtual* $v(t)$ é uma função linear por partes, crescente no tempo, e com uma inclinação que muda sempre que o conjunto de fluxos em *backlog* $B(t)$ é alterado, e é inversamente proporcional à soma dos serviços compartilhados pelas sessões em *backlog*. A Figura 3.5 ilustra o comportamento da função *tempo virtual* $v(t)$.

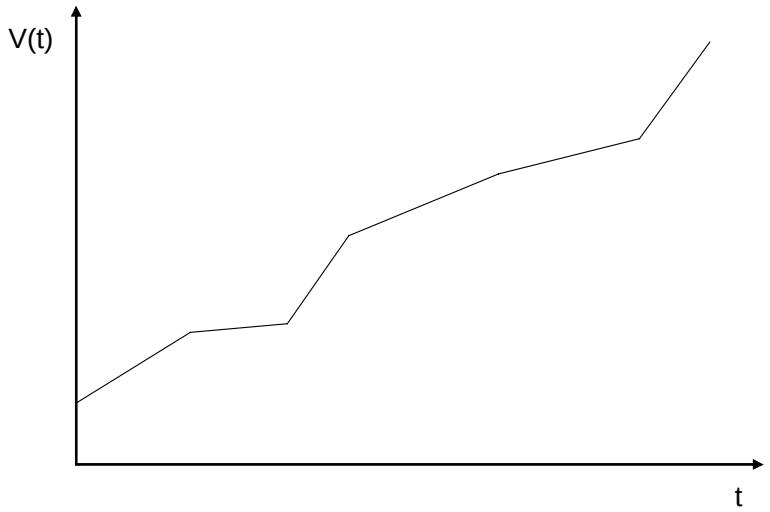


Figura 3.5 – Variação de v(t) no tempo

Seja F_p o instante no qual o pacote p seria enviado, ou seja, o tempo final de serviço se

este pacote estivesse sob o modelo referencial GPS. Dessa forma, uma boa aproximação do GPS seria o envio dos pacotes, pelo mecanismo WFQ, em uma ordem crescente de F_p.

Para obtenção do *tempo virtual final* (de serviço) F_k^i, define-se p_k^i como representando o *i-ésimo* pacote do fluxo *k*; a_k^i como seu tempo de chegada, L_k^i como seu tamanho e r_k como o fluxo proporcional, conforme eq. 3.3.

Quando um pacote chega ao roteador, o algoritmo WFQ lhe atribui o *tempo virtual final*, F_k^i, $i = 1, 2, 3, ...$, representado por:

$$F_k^i = \frac{1}{r_k} L_k^i + S_k^i \qquad i = 1,2,..., \qquad k \in K \tag{3.9}$$

e,

$$F_k^0 = 0, \qquad k \in K \tag{3.10}$$

onde,

$$S_k^i = \max(F_k^{i-1}, v(a_k^i)) \qquad i = 1,2,..., \qquad k \in K \tag{3.11}$$

S_k^i representa o *número de partida* (*start number*), que é o valor máximo entre o *tempo virtual final* do (*i*-1)*-ésimo* pacote do *k-ésimo* fluxo e $v(a_k^i)$, que representa o valor de $v(t)$ de chegada do pacote p_k^i.

Se o pacote chega em uma fila inativa, então o número de partida será o tempo virtual corrente $v(a_k^i)$. Se a chegada ocorre em uma fila ativa, o número de partida será o tempo final do último pacote na fila.

Sendo os pacotes servidos na ordem crescente do *tempo virtual final* F_k^i, cria-se a equivalência com o tempo virtual real que seriam tratados no sistema de referência GPS. Dessa forma, o mecanismo WFQ tem que identificar cada pacote que chega com seu *tempo virtual final* e servi-los na ordem crescente destas identificações. O cálculo do *tempo virtual final* para cada pacote é uma operação que requer o conhecimento do tamanho do pacote, tempo virtual final do pacote anterior, e o tempo virtual do sistema quando o pacote chegou [27].

O cálculo do *tempo virtual* requer que o mecanismo WFQ mantenha os parâmetros necessários: as coordenadas da última mudança de inclinação, a inclinação atual e o número de sessões atendidas. A partir do momento em que a chegada e partida de pacotes alteram estes valores, e que pacotes podem chegar simultaneamente, este é um grande desafio computacional.

Obs: A equação 3.9 também é apresentada [36] [44] [48] [49] como:

$$F_k^i = \frac{1}{\phi_k} L_k^i + S_k^i \qquad i = 1,2,..., \qquad k \in K \tag{3.12}$$

A equação 3.12 tem a mesma funcionalidade da equação 3.9, uma vez que Φ_k e r_k mantém relação diretamente proporcional conforme Equação 3.3.

Exemplo de atuação do mecanismo *Weighed Fair Queueing*

Para o exemplo a ser apresentado [50], considera-se o seguinte cenário:

- Para os 3 fluxos que serão tratados pelo mecanismo WFQ, assume-se que todos os

pacotes são do mesmo tamanho e que a taxa de transmissão R do enlace de saída é de um pacote por unidade de tempo.

- Os fluxos apresentam pesos iguais, $\Phi_1=\Phi_2=\Phi_3=1/3$. Desta forma, as taxas de transmissão proporcionais de cada fluxo também são iguais, $r_1=r_2=r_3=R/3$.
- Os tempos de chegada dos pacotes para os fluxos 1 e 2 são t=0,3,6,9,12, enquanto os tempos de chegada para o fluxo 3 são t=7,8,9,10,11,12.

A Figura 3.6 ilustra este cenário e traz o comportamento do GPS para esta situação.

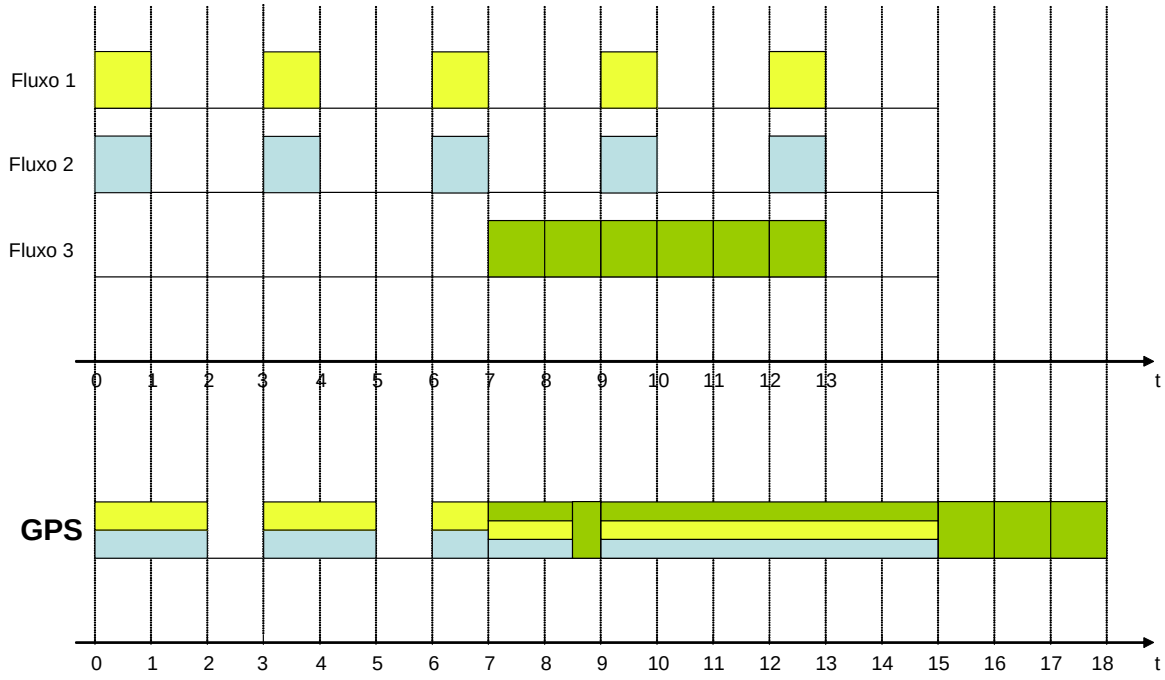


Figura 3.6 – Tempo de chegada dos fluxos e seu comportamento em um esquema GPS.

A partir do esquema GPS, obtém-se o conjunto de fluxos armazenados (*backlogged*) $B(t)$ e o tempo virtual $v(t)$, conforme apresentado na Figura 3.7.

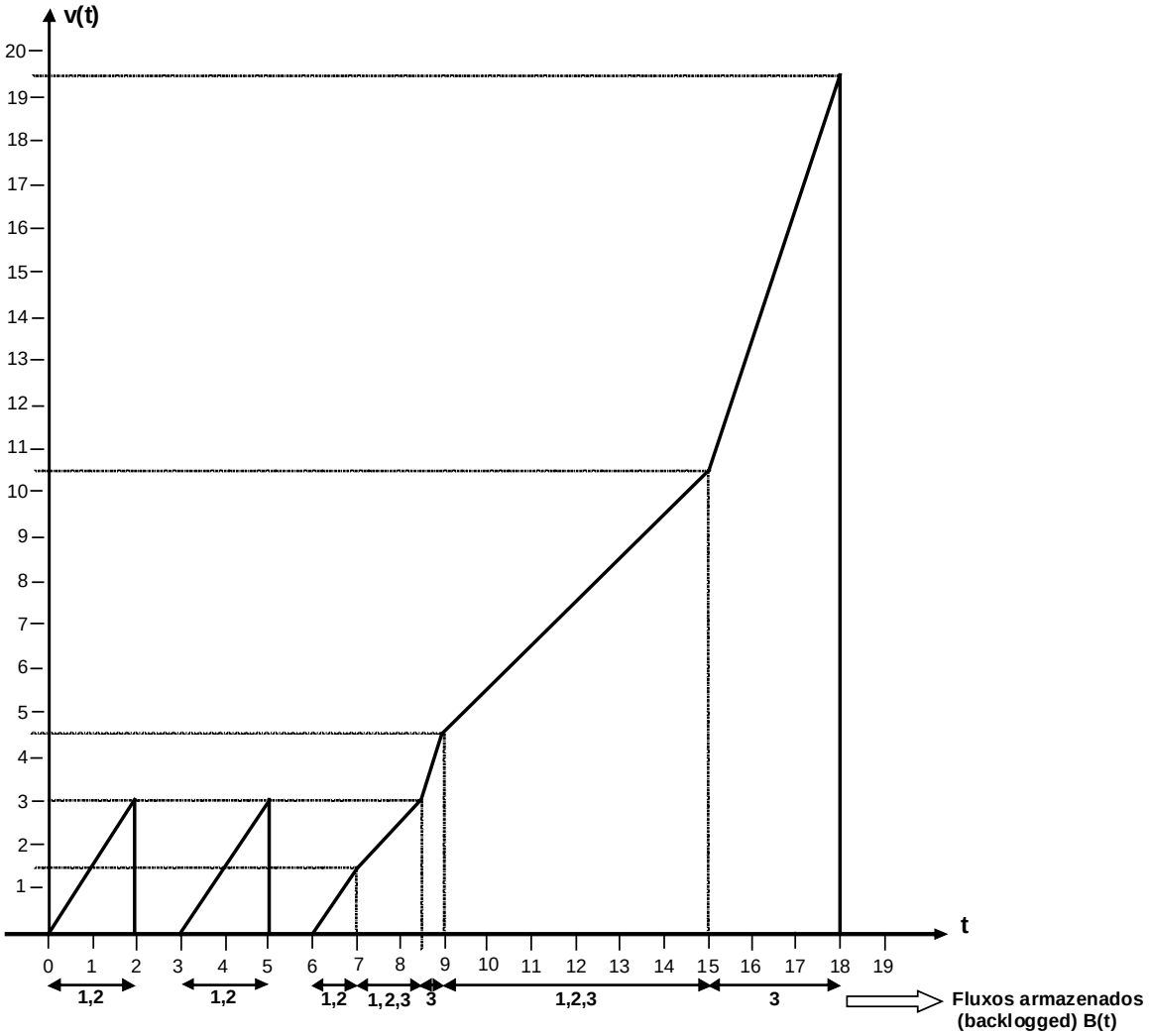


Figura 3.7 – Tempo virtual *v(t)* e os fluxos armazenados *B(t)*.

A Tabela 3.3 apresenta o tempo t, o tempo virtual v(t) e o tempo virtual final de serviço de cada pacote de cada fluxo.

O resultado apresentado na Tabela 3.3 resulta na seqüência implementada pelo mecanismo *Weighed Fair Queueing* conforme Figura 3.8. A ordem dos pacotes com o tempo virtual final de serviço F^i_k idênticos são arbitrários.

t	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
v(t)	0	1,5	3	0	1,5	3	0	1,5	2,5	4,5	5,5	6,5	7,5	8,5	9,5	10,5	13,5	16,5	19,5	0
F^i_1	3			3			3													
F^i_2	3			3			3													
F^i_3								4,5	7,5	10,5	13,5	16,5	19,5							

Tabela 3.3 – *t*, *v(t)* e F^i_k para o exemplo apresentado.

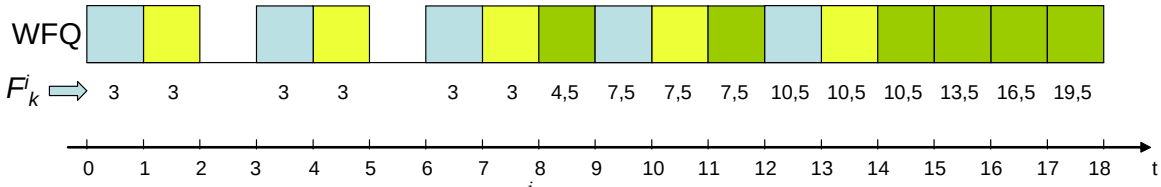


Figura 3.8 – Seqüência WFQ e valores F^i_k dos pacotes.

Avaliação do atraso no *Weighed Fair Queueing*

Uma vantagem associada ao mecanismo WFQ é a possibilidade de cálculo do atraso máximo fim-a-fim baseado no peso designado ao fluxo. Sendo R_n, n = 1, 2, …, N, a representação das velocidades dos enlaces percorridos pelo fluxo i e r_i a taxa mínima garantida para o fluxo i em cada enlace que este percorre, o atraso fim-a-fim garantido (menor ou igual) para o fluxo i é dado por:

$$D_i \leq \frac{b_i}{r_i} + \frac{(N-1)L\max_i}{r_i} + \sum_{n=1}^{N} \frac{L\max_n}{R_n} \qquad (3.13)$$

Onde b_i é a capacidade do armazenamento (*balde de fichas*) que caracteriza o fluxo, $Lmax_i$ representa o tamanho máximo do pacote para o fluxo i, e $Lmax_n$ representa o tamanho máximo do pacote no enlace n.

O atraso fim-a-fim é independente do número de outras conexões que estão compartilhando cada um dos enlaces percorridos pelo fluxo i. Essa propriedade faz o mecanismo WFQ um dos melhores em termos de provimento de garantia de atraso fim-a-fim confiável [44].

3.2.5. Variações do *Weighted Fair Queuing* (WFQ)

O mecanismo WFQ se mostra bastante competente na aproximação com o GPS, e no cumprimento do conceito de justiça associado à maneira pela qual a banda é compartilhada pelos fluxos. Fornece também considerável precisão na garantia do atraso e banda.

Cada uma das variações do WFQ apresenta alguma melhoria em relação ao WFQ, por exemplo: diminuição do custo computacional do *tempo virtual* $v(t)$, melhoria na garantia do atraso, melhoria na aproximação com o GPS e conseqüentemente no cumprimento do conceito de justiça, entre outras [51].

A seguir são apresentados sucintamente alguns destes algoritmos variantes do WFQ ou PGPS.

3.2.5.1. *Self Clocked Fair Queuing* (SCFQ)

Uma das maiores desvantagens do WFQ está na complexidade envolvida na implementação computacional da função *tempo virtual*. O *Self Clocked Fair Queuing* (SCFQ) foi proposto em [52] como um simplificador do algoritmo WFQ [44].

O SCFQ também é baseado na noção de *tempo virtual*, contudo é referenciado no enfileiramento atual do sistema, em lugar de um sistema hipotético. Alem disso, ao invés de utilizar uma definição abstrata analítica para o *tempo virtual*, adota-se como

tempo virtual uma quantidade que naturalmente surge no processo do algoritmo [52].

O SCFQ propõe uma aproximação simples para calcular o instante de partida (tempo final de serviço) no correspondente sistema GPS. Quando um pacote chega a uma fila de espera, o SCFQ usa como *v(t)* o *tempo virtual final* do pacote que está nesse momento em serviço (*Factual*). O instante de partida é então dado por:

$$F_k^i = \max\left(F_k^{i-1}, F_{actual}\right) + \frac{L_k^i}{\phi_k} \qquad i = 1,2,..., \qquad k \in K \tag{3.14}$$

Quando nenhum pacote é encontrado na fila, o algoritmo atribui o valor zero para *v(t)*.

3.2.5.2. *Start-time Fair Queuing* (SFQ)

O algoritmo *Start-time Fair Queuing* (SFQ) foi proposto em [49] como um outro mecanismo *Fair Queuing*. É bastante similar ao SCFQ com a principal diferença de escolher o pacote com o menor tempo virtual de início de serviço (em oposição ao menor tempo virtual de final de serviço) para transmissão no enlace de saída [44].

No SFQ, dois identificadores, um identificador de início e um de finalização, são associados a cada pacote. Contudo, diferente do WFQ e SCFQ, os pacotes são servidos pela ordem crescente do identificador de início dos pacotes. O algoritmo completo é definido a seguir [49]:

1) Na chegada, o pacote é marcado com o identificador de partida como

$$S_k^i = \max(F_k^{i-1}, v(a_k^i)) \qquad i \geq 1 \tag{3.15}$$

onde a identificação de finalização do pacote é definido como

$$F_k^i = S_k^i + \frac{L_k^i}{\phi_k} \qquad i \geq 1 \tag{3.16}$$

2) Inicialmente o *tempo virtual* do servidor é 0. Durante um período de ocupação, o *tempo virtual* no tempo *t*, *v(t)*, é definido como sendo igual ao identificador de partida do pacote sendo servido no tempo *t*. No final do período de ocupação, *v(t)* assume o valor máximo do identificador de finalização dos pacotes que já tenha sido servido.

3) Os pacotes são servidos na ordem crescente dos identificadores de partida.

Como evidenciado na definição, o custo computacional da implementação do SFQ é baixo e equivalente ao SCFQ, uma vez que envolve apenas a manipulação dos identificadores de partida dos pacotes.

3.2.5.3. *Worst-case Fair Weighted Fair Queuing* (WF^{2}Q)

O algoritmo *Worst-case Fair Weighted Fair Queuing* (WF^{2}Q) proposto em [53] utiliza tanto o tempo de partida quanto o tempo final de envio no sistema referencial GPS para obter uma emulação mais acurada do GPS. O WF^{2}Q seleciona o pacote com o menor tempo virtual final no sistema referencial GPS contanto que o tempo virtual de partida seja menor que o tempo corrente [44].

O funcionamento do mecanismo WF^{2}Q é bastante semelhante ao do PGPS (ou WFQ) com uma diferença na forma em que os pacotes a serem servidos são selecionados.

Quando o PGPS escolhe um pacote a ser servido no instante t_s, ele seleciona, entre todos os pacotes presentes no servidor, o pacote que terminaria o serviço primeiro no GPS. Já o WF^{2}Q, ao invés de escolher o pacote que terminaria primeiro o serviço no GPS dentre todos os pacotes presentes no servidor, escolhe somente entre os pacotes que já teriam iniciado o serviço no GPS (pacotes elegíveis). Ou seja, no mecanismo WF^{2}Q, o pacote escolhido para ser servido no instante t_s é o pacote que terminaria o serviço primeiro no GPS e que já teria começado a ser servido pelo GPS no instante t_s [48].

4. Mecanismos de Controle de Congestionamento

Em uma rede do tipo melhor esforço (*best-effort*), o controle de congestionamento está baseado em dois elementos fundamentais: um algoritmo de controle de fluxo na fonte (*source flow control*) e um algoritmo de gerenciamento de fila (AQM - *Active Queue Management*) [54].

O algoritmo de controle de fluxo na fonte reside no sistema operacional do *host* que está transmitindo a informação. Portanto, em uma arquitetura cliente-servidor, este algoritmo reside tanto no cliente quanto no servidor. O algoritmo de controle de fluxo na fonte deve decidir a quantidade de informação a ser transmitida, de acordo com o nível de congestionamento percebido na rede [54].

O algoritmo AQM reside nos dispositivos intermediários da conexão (roteadores), atuando nos *buffers* destes dispositivos, onde estão as filas de pacotes aguardando a transmissão em um enlace de saída. O algoritmo AQM estima o nível de congestionamento e gera uma informação de congestionamento para a fonte, alimentando o algoritmo de controle de fluxo na fonte.

Atualmente, o algoritmo de controle de fluxo na fonte predominante e praticamente único na Internet é o implementado pelo protocolo TCP (*Transmission Control Protocol*).

O desenvolvimento de algoritmos AQM tem recebido atenção especial dos

pesquisadores porque sua implementação tem proporcionado considerável melhoria no desempenho de QoS (*Quality of Service*) na Internet. A evolução dos algoritmos AQM tem relação crucial com a melhoria de desempenho da rede, por meio da diminuição do atraso, da variação de atraso (*jitter*) e das perdas de pacotes na rede.

O mecanismo predominante nos roteadores para controle de congestionamento é o *tail-drop*. O *tail-drop* não é considerado um algoritmo AQM, pois representa simplesmente uma fila que, quando preenchida ao máximo, estoura e descarta os pacotes que chegam a seguir.

Nas seções seguintes será visto o papel do TCP no controle do congestionamento, atuando como algoritmo de controle de fluxo na fonte. Será visto também a evolução dos algoritmos AQM, sua classificação entre "*backlog-based*" e "*rate-based*", e a introdução do ECN – *Explicit Congestion Notification*, o qual permitiu aos algoritmos AQM a possibilidade de marcação dos pacotes para sinalizar o congestionamento, como opção em relação ao descarte para realizar esta sinalização.

4.1. Controle de Fluxo e Congestionamento no Protocolo TCP

O protocolo TCP (*Transmission Control Protocol*) é considerado o protocolo mais importante da suíte TCP/IP. Este protocolo da camada de transporte, orientado à conexão, fornece entrega confiável entre *hosts* em um ambiente cliente/servidor e provê mecanismos de controle de fluxo e de congestionamento.

4.1.1. Mecanismo de Controle de Fluxo do TCP

O protocolo TCP utiliza um mecanismo de janela deslizante para controlar o fluxo de dados. Quando uma conexão é estabelecida, cada lado da conexão aloca um *buffer* para armazenar os dados que chegam e envia o tamanho desse *buffer* para o outro lado. À medida que os dados chegam, o receptor envia reconhecimentos (ACKs), reconhecendo o recebimento dos dados e informando o tamanho de *buffer* restante. Esta quantidade de espaço de *buffer* disponível é informada no campo janela (*window*) do segmento TCP. O espaço em *buffer* é liberado à medida que a aplicação "lê" os dados em espera no *buffer*. Desta forma, cada lado da conexão fica limitado a enviar, no máximo, a quantidade de dados especificada pela janela, evitando o *overflow* do *buffer* de destino. A Figura 4.1 ilustra este mecanismo [55]. Nesta figura o tamanho inicial da janela no *host* B é 2500. Após o recebimento do primeiro segmento, contendo 1000 bytes, o *host* B envia um reconhecimento do segmento recebido e comunica o novo valor da janela, que é igual a 1500; após o recebimento do segundo segmento, que contém 1000 bytes, o *host* B envia o reconhecimento deste segmento e informa o novo valor da janela de recepção, 500. As demais trocas de mensagens ilustradas são semelhantes e são auto-explicativas.

O mecanismo de controle de fluxo do TCP evita a perda de dados e onseqüentemente seu reenvio ao evitar o *overflow* do *buffer* de destino. Este controle se dá entre os *hosts* de origem e destino, porém este mecanismo não atua em problemas de congestionamento em dispositivos intermediários da rede (roteadores).

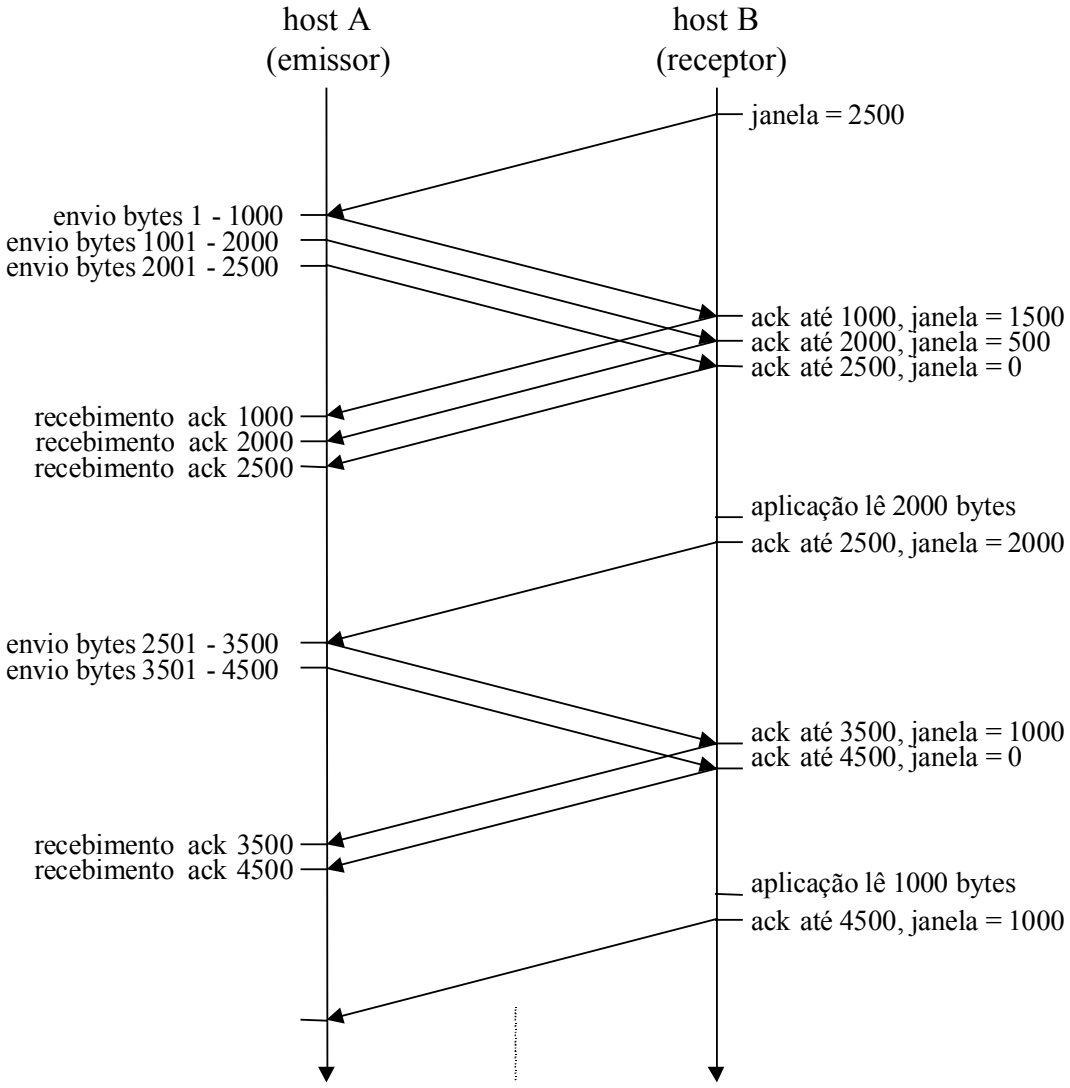


Figura 4.1 – Mecanismo de controle de fluxo do TCP

4.1.2. Mecanismo de Controle de Congestionamento do TCP

Uma das principais causas de congestionamento em uma rede é o excesso de tráfego em um dispositivo intermediário, ou seja, um roteador. Este excesso de tráfego pode acarretar um atraso excessivo ou perda de datagramas. O atraso excessivo ou a perda de um datagrama faz com que o protocolo TCP reenvie o segmento perdido, em função de não ter recebido o reconhecimento do *host* de destino.

Se o TCP do *host* de origem continuasse a enviar os dados, após uma perda de um segmento, apenas considerando o tamanho da janela do *host* de destino, haveria uma tendência de aumento do congestionamento. Para evitar isso, sempre que um segmento é perdido, o TCP inicia o mecanismo de controle de congestionamento. O TCP passa a enviar uma pequena quantidade de dados, em vez da quantidade estipulada pela janela anunciada pelo receptor. Se receber o reconhecimento do *host* de destino, enviará o dobro de dados no próximo RTT (*round-trip time*), e seguirá dobrando a quantidade de dados, se continuar recebendo os reconhecimentos, até atingir um limite (definido a frente), quando diminuirá a taxa de aumento do número de segmentos enviados. Este mecanismo implementado pelo TCP é a combinação dos algoritmos *slow start* (partida lenta) e *congestion avoidance* (prevenção de congestionamento) [36]. Estes algoritmos e outros implementados pelo TCP são detalhados nos itens a seguir.

Todas as conexões TCP que estiverem experimentando um congestionamento em um ponto da rede (roteador), e sofrerem perdas de dados, iniciarão o mecanismo de controle de congestionamento.

4.1.2.1. Detalhamento dos Algoritmos de Controle de Congestionamento do TCP

O protocolo TCP consegue adaptar a taxa de envio do emissor à capacidade da rede e desta forma evitar situações críticas de congestionamento. Várias implementações com relação ao controle de congestionamento tem sido adicionadas ao protocolo TCP ao longo dos últimos anos. O crescimento constante da Internet e de novas aplicações

sensíveis a QoS tem impulsionado as pesquisas nesta área.

As implementações modernas do TCP apresentam quatro algoritmos de controle de congestionamento, a seguir descritos: *slow start*, *congestion avoidance*, *fast retransmit* e *fast recovery* [56] [57].

- ***Slow Start***

Nas primeiras implementações do protocolo TCP havia somente mecanismos de controle de fluxo, ou seja, o emissor injetava segmentos na rede limitado somente pelo tamanho da janela do receptor. Geralmente este mecanismo se mostrava suficiente para as situações nas quais os dois *hosts* se encontravam na mesma LAN (*Local Area Network*). Caso os *hosts* estejam posicionados em LANs distintas interligadas por roteadores e enlaces de baixa velocidade, problemas de congestionamento podem ocorrer e se fazem necessários mecanismos de controle de congestionamento.

O algoritmo *slow start* (partida lenta) é um dos algoritmos implementados no protocolo TCP para compor os recursos do protocolo em relação ao controle de congestionamento.

O algoritmo *slow start* opera com base no fato de que a taxa de envio de pacotes injetados pelo emissor na rede está relacionada com os reconhecimentos (ACKs) retornados pelo receptor.

O algoritmo *slow start* adiciona uma nova janela aos emissores TCP, a janela de

congestionamento (*congestion window*), chamada *cwnd*. Quando uma nova conexão é estabelecida, a janela de congestionamento é inicializada com o valor *IW* (*Initial Window*) definido por: [57] [58]

$$IW = min\ (4*SMSS,\ max\ (2*SMSS,\ 4380\ bytes)) \qquad (4.1)$$

Onde *SMSS* (*Sender Maximum Segment Size*) é o tamanho do maior segmento que o emissor pode transmitir.

Inicialmente, a RFC 2001 definiu o valor de *IW* em um segmento. A RFC 2581 definiu que deveria ser menor ou igual a 2 segmentos (2*SMSS). A última atualização, RFC 3390, o definiu como na Equação 4.1.

A experimentação indica que um valor inicial de até 4 segmentos pode permitir um início de conexão mais eficiente, particularmente para aquelas conexões TCP de curta duração que prevalecem em uma busca *web*. Observações no tráfego *web* indicam uma média de transferência de dados de 17 segmentos. O *Slow Start* a partir de um segmento, levará cinco intervalos RTT para a transferência dos dados, enquanto o uso de *IW* de quatro segmentos reduz o tempo de transferência para três intervalos RTT. Contudo, quatro segmentos pode ser um número grande para enlaces de baixa velocidade e com limitação de *buffers*, nos quais uma aproximação mais robusta seria o uso de *IW* com valor não superior a dois segmentos para começar o *Slow Start* [59].

A janela de transmissão, que define a máxima quantidade de dados que o emissor pode

enviar, é o menor valor entre a janela de congestionamento e a janela de fluxo informada pelo receptor. A janela de congestionamento é o controle de fluxo imposto pelo emissor, enquanto a janela de fluxo é o controle de fluxo imposto pelo receptor. O valor da janela de congestionamento é definido com base na percepção do emissor sobre o congestionamento que a rede está enfrentando e a janela de fluxo é definida em função da quantidade de espaço em *buffer* disponível no receptor para esta conexão.

Durante o *Slow Start*, o TCP incrementa *cwnd* em até SMSS bytes para cada ACK recebido. Logo, este algoritmo provê um aumento exponencial na quantidade de dados enviados pelo emissor.

Em algum momento, a capacidade da rede IP, geralmente em um enlace de menor capacidade, pode ser atingida, e o roteador intermediário associado a este enlace inicia o descarte de pacotes. Isto é uma sinalização de que a janela de congestionamento está muito grande.

- ***Congestion Avoidance***

Existem duas indicações de perda de pacotes sentidas pelo emissor:

1. A ocorrência de *timeout* no recebimento do ACK.
2. Recebimentos de ACKs duplicados.

O algoritmo *slow start* atua até que o tamanho da janela de congestionamento (*cwnd*)

atinja o valor patamar *ssthresh* (definido a seguir), implementando um crescimento exponencial no valor da janela de congestionamento. Quando o tamanho da janela de congestionamento atinge o valor patamar, o algoritmo *congestion avoidance* começa a atuar, agora com um crescimento linear da janela de congestionamento. Os algoritmos *congestion avoidance* e *slow start* são algoritmos independentes, mas na prática, são implementados juntos.

Os algoritmos *congestion avoidance* e *slow start* requerem que duas variáveis sejam mantidas para cada conexão:

1. Uma janela de congestionamento (*congestion window*) – *cwnd*
2. Um tamanho de limiar (*slow start threshold size*) – *ssthresh*

Os algoritmos combinados atuam da seguinte forma:

1. Na inicialização de uma conexão, a variável *cwnd* assume o valor de *IW* e a variável *ssthresh* pode ser arbitrariamente alta (por exemplo, algumas implementações usam o tamanho da janela de fluxo do receptor) [57]. A RFC 2001 atribuía 65535 bytes para o valor inicial de *ssthresh*.

2. O TCP jamais envia mais segmentos do que o menor valor entre o especificado pela janela de congestionamento e o especificado pela janela de fluxo do receptor.

3. Quando o congestionamento ocorre (indicado por um *timeout* ou pela recepção de ACKs duplicados), a variável *ssthresh* é atualizada por:

$$ssthresh = max\ (FlightSize / 2,\ 2*SMSS) \quad (4.2)$$

onde *FlightSize* é a quantidade de dados enviada porém ainda não reconhecida (ACK). Adicionalmente, se o congestionamento está sendo indicado por um *timeout*, a variável *cwnd* assume o valor de *IW*.

4. A partir daí, quando um novo segmento é reconhecido pelo receptor, a variável *cwnd* é incrementada de acordo com o algoritmo que está atuando neste momento, o *slow start* ou o *congestion avoidance*. O algoritmo *slow start* é utilizado quando *cwnd* < *ssthresh*, enquanto o algoritmo *congestion avoidance* é utilizado quando *cwnd* > *ssthresh*. Quando *cwnd* = *ssthresh*, o emissor pode utilizar tanto o algoritmo *slow start* quanto o algoritmo *congestion avoidance* [57].

O algoritmo *slow start* irá atuar até que a janela de congestionamento atinja o valor salvo em *ssthresh* no passo 3. Neste ponto, o algoritmo *congestion avoidance* começa a atuar em lugar do *slow start*.

Conforme já mencionado, o algoritmo *slow start* inicia o valor de *cwnd* em *IW* e é incrementado em um segmento a cada ACK recebido. Este procedimento abre a janela exponencialmente.

O algoritmo *congestion avoidance* determina que *cwnd* seja incrementado por:

$$cwnd \leftarrow cwnd + (SMSS * SMSS / cwnd) \quad (4.3)$$

a cada ACK recebido.

O algoritmo *congestion avoidance* dita um crescimento linear do *cwnd*, enquanto no algoritmo *slow start* este crescimento é exponencial. Pode-se verificar que no *congestion avoidance*, o aumento de *cwnd* será de no máximo um segmento por cada RTT (independente de quantos ACKs são recebidos neste RTT).

- ***Fast Retransmit***

No algoritmo *fast retransmit* (retransmissão rápida) a retransmissão de um segmento é iniciada após o recebimento de 3 ACKs duplicados, caso ainda não tenha ocorrido o *timeout* do segmento. Logo, o algoritmo *fast retransmit* evita que o protocolo TCP tenha que aguardar um *timeout* para reenviar segmentos perdidos.

O protocolo TCP do receptor pode duplicar um ACK de maneira imediata quando um segmento fora de ordem é recebido. O objetivo desta duplicação do ACK é sinalizar para o emissor que um segmento foi recebido fora de ordem e indicar o número da seqüência que está sendo esperado.

A Figura 4.2 ilustra o algoritmo *fast retransmit*. Nesta figura observa-se que o receptor não recebe o segmento 3 e o emissor o retransmite após o recebimento de 3 ACKs do segmento 2 duplicados.

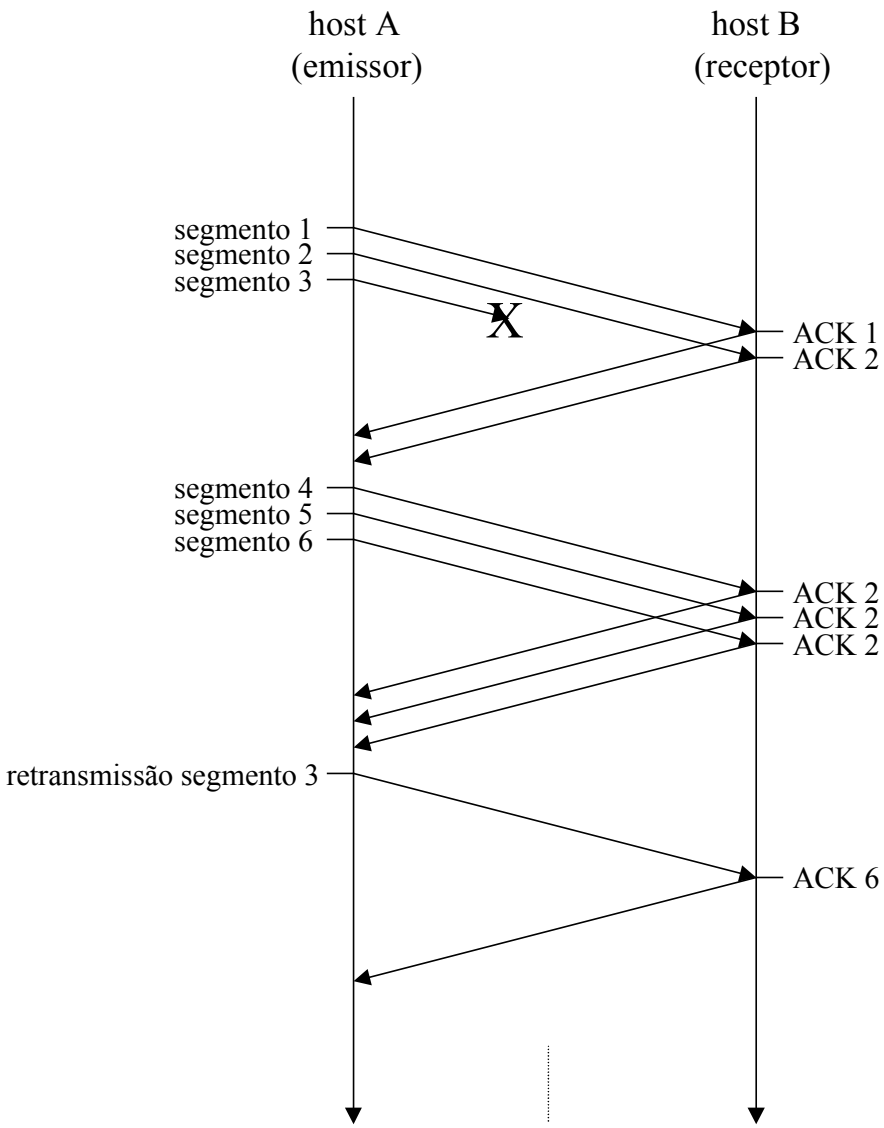


Figura 4.2 – TCP *fast retransmit* [55]

- ***Fast Recovery***

No algoritmo *fast recovery*, após o algoritmo *fast retransmit* ter enviado o que aparentava ter sido um segmento perdido, o algoritmo *congestion avoidance* e não o *slow start* é executado. Esta implementação é utilizada sob congestionamento moderado.

O motivo para a não utilização do *slow start* quando da recepção de ACKs duplicados está baseado no fato de que apenas um segmento foi perdido. Desde que o receptor só pode gerar ACKs duplicados quando segmentos posteriores forem recebidos, conclui-se que o tráfego entre os dois *hosts* ainda está fluindo e, desta forma, o protocolo TCP não deve reduzir abruptamente o fluxo de dados, o que aconteceria se o mesmo utilizasse o algoritmo *slow start*.

Os algoritmos *fast retransmit* e *fast recovery* são geralmente implementados juntos conforme segue:

1. Quando o terceiro ACK duplicado for recebido, fazer *ssthresh* assumir o valor dado pela Equação 4.2.

2. Retransmitir o segmento perdido. Atribuir para *cwnd* o valor de *ssthresh* acrescido de três vezes o tamanho do *SMSS* (3**SMSS*). Este procedimento acerta a janela de congestionamento com o número de segmentos que haviam sido enviados e que foram recebidos pelo receptor.

3. Cada vez que um outro ACK duplicado chegar, incrementar *cwnd* com *SMSS*. Este procedimento acerta a janela de congestionamento com o número de segmentos que foram enviados.

4. Transmitir um segmento, se permitido pelo novo valor de *cwnd* e pela janela de fluxo do receptor.

5. Quando receber o ACK deste novo segmento, atribuir para *cwnd* o valor de *ssthresh* que foi armazenado no passo 1.

4.2. *Tail-Drop*

O *tail-drop* caracteriza-se pela inexistência de um algoritmo AQM atuando no nó (roteador) da rede. O *tail-drop* consiste simplesmente em uma fila que, quando preenchida ao máximo, estoura (*overflow*) e, conseqüentemente, descarta os pacotes que chegam posteriormente [54]. O *tail-drop* trata todos os tráfegos do mesmo modo e não os diferencia entre classes de serviço [30].

O *tail-drop* é considerado ineficiente, pois nenhum congestionamento é detectado até que a fila esteja cheia, quando então todos os pacotes que chegam são descartados e, conseqüentemente, os transmissores com aplicações baseadas no protocolo TCP diminuirão suas taxas de envio. O descarte de pacotes é a sinalização para as fontes de tráfego da existência do congestionamento.

A diminuição das taxas de envio pelas aplicações que utilizam o nó em congestionamento causará a diminuição do tráfego e, conseqüentemente, o *buffer* sairá do estado de *overflow*.

No *tail-drop*, quando o *buffer* não está totalmente cheio, nenhum sinal de congestionamento é gerado, ou seja, nenhum pacote é descartado. Desta forma, as fontes de tráfego aumentam gradativamente suas taxas, podendo levar a uma situação de congestionamento.

Pode-se notar que, em uma situação de congestionamento, o *tail-drop* provoca um efeito cíclico de diminuição e aumento do tráfego, mantendo o *buffer* sempre beirando o *overflow*. Este efeito é conhecido como "sincronização global".

O *tail-drop* realimenta o protocolo TCP com uma sinalização inadequada, pois só é capaz de gerar um sinal de congestionamento quando este já ocorreu (com o *buffer* cheio e pacotes descartados).

4.3. *Random Early Detection* (RED)

O algoritmo RED (*Random Early Detection* – Detecção Antecipada Aleatória) foi proposto por Sally Floyd e Van Jacobson [60] no início dos anos 90 como um mecanismo de prevenção de congestionamento em fluxos de aplicações baseadas no TCP.

O RED é um algoritmo AQM de prevenção de congestionamento que mantém o tamanho médio da fila em um valor que proporciona baixo atraso e alto *throughput*, enquanto evita a situação na qual os fluxos TCP experimentam o congestionamento ao mesmo tempo, evitando conseqüentemente o efeito da sincronização global [61], comum no *tail-drop*.

O princípio básico no RED consiste na prevenção da ocorrência do congestionamento através da monitoração e controle do tamanho da fila [62]. Quando este tamanho ultrapassa um valor específico, o RED começa a selecionar aleatoriamente fluxos TCPs

individuais e descartar pacotes nestes fluxos, forçando as origens destes fluxos a diminuírem a taxa de envio de pacotes. A Figura 4.3 ilustra este mecanismo [31]. Nesta figura, pode-se observar que o algoritmo RED aleatoriamente descarta pacotes de fluxos individuais para sinalizar aos emissores a presença de congestionamento.

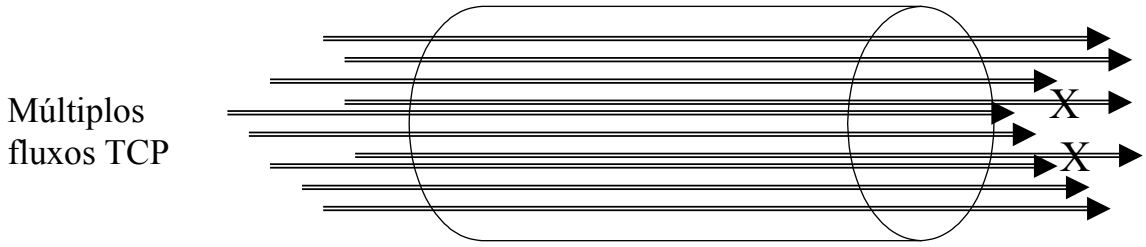


Figura 4.3 – O descarte aleatório do RED

A conexão TCP que sofrer a perda de pacotes iniciará a redução da sua taxa de transmissão de bits. Pouco a pouco, muitas conexões TCP sofrerão alguma perda de pacotes e reduzirão sua taxa de transmissão de bits, porém não simultaneamente.

O algoritmo RED descarta pacotes de maneira probabilística baseado na estimativa média do tamanho da fila. A probabilidade de descarte aumenta de maneira proporcional ao crescimento da fila [42]. Especificamente, o algoritmo RED consiste em duas partes principais: uma parte responsável pela estimativa do tamanho médio da fila (*average queue size*) e outra parte associada à tomada de decisão quanto ao descarte do pacote [63].

O tamanho médio da fila é comparado com dois valores de limiares definidos, limiar mínimo (th_{min}) e limiar máximo (th_{max}). Esta comparação define o tratamento do algoritmo quanto ao descarte do pacote. Caso o valor médio da fila esteja abaixo de th_{min}, o algoritmo estará na zona de operação normal e todos os pacotes são aceitos. Se o

valor médio da fila estiver acima de th_{max}, o algoritmo estará na zona de controle de congestionamento, e todos os pacotes serão descartados. Por fim, se o tamanho médio da fila se encontrar entre th_{min} e th_{max}, o algoritmo se encontrará na zona de prevenção de congestionamento, e os pacotes serão descartados com a probabilidade p_a. A Figura 4.4 ilustra as 3 zonas de atuação do algoritmo [64].

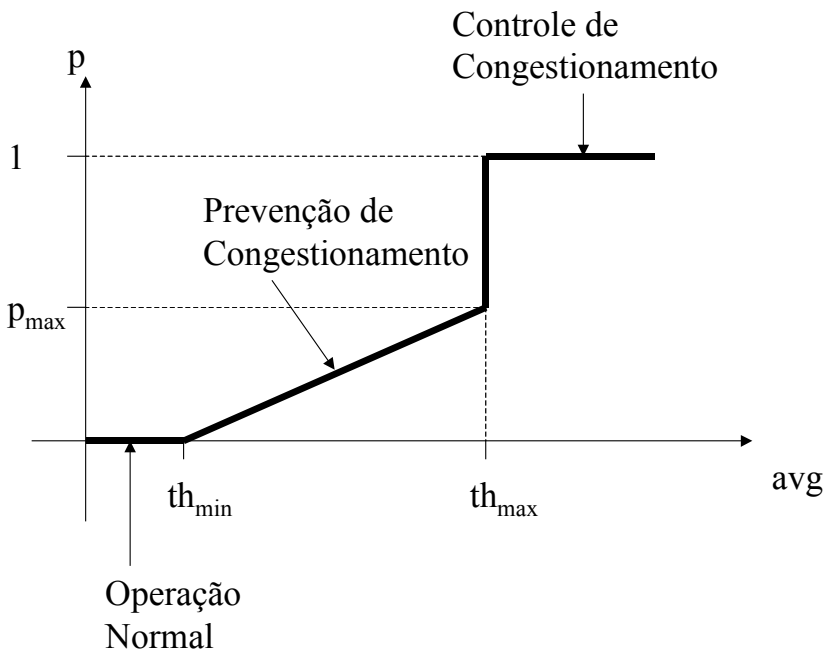


Figura 4.4 – Zonas de operação do RED

O RED estima o tamanho médio da fila para cada pacote recebido através de uma média ponderada móvel (*weighted moving average*) [60]:

$$avg_i = (1 - w_q)\, avg_{i-1} + w_q \, q \tag{4.4}$$

onde o peso w_q dimensiona a influência do tamanho atual da fila (q) no cálculo da nova média. O uso de w_q permite a acomodação de tráfegos em rajadas e congestionamentos transientes. A escolha do peso w_q determina a maneira como o algoritmo reage às mudanças no tamanho da fila. Caso w_q seja definido com um valor alto, o uso da média

não conseguirá filtrar os congestionamentos passageiros. Por outro lado, se w_q for definido com um valor baixo, o algoritmo reagirá de forma lenta ao congestionamento. O valor apropriado para w_q depende de th_{min} e da quantidade de pacotes em rajada (L) que se deseja suportar. Estes valores estão relacionados da seguinte forma [60]:

$$L+1+\frac{(1-w_q)^{(L+1)}-1}{w_q}<th_{\min} \tag{4.5}$$

Os valores ótimos para th_{min} e th_{max} dependem do valor desejado para o tamanho médio da fila (avg). Contudo, o valor desejado para o tamanho médio da fila depende do tipo de tráfego na rede [61]. Se o tráfego, em geral, for em rajadas, é desejável a utilização de um valor de avg alto o suficiente para manter uma alta utilização da rede, evitando a subutilização do enlace de saída do roteador. Por outro lado, um valor alto de avg, implicará em um maior atraso do datagrama neste roteador. A diferença entre os valores de th_{min} e th_{max} também influencia na prevenção do efeito indesejado da sincronização global.

A atribuição de valores ótimos para th_{min} e th_{max} depende do conhecimento das características do tráfego da rede e tem sido assunto de pesquisas. Um modelo apresentado em [65] demonstra o quanto a atribuição de diferentes valores para th_{min} e th_{max} resultam em diferentes resultados em relação a probabilidade de perdas e atrasos.

O cálculo da probabilidade p_a, que define a probabilidade de descarte do pacote na zona de prevenção de congestionamento é definido por [60]:

$$p_a=\frac{p_b}{(1-count\,p_b)} \tag{4.6}$$

$$p_b = p_{\max} \frac{(avg - th_{\min})}{(th_{\max} - th_{\min})} \quad (4.7)$$

onde p_{max} define a probabilidade máxima de descarte quando o tamanho médio da fila atinge th_{max} e *count* representa o número de pacotes desde o último pacote descartado. O parâmetro p_b varia linearmente entre 0 e p_{max}, enquanto $p_{a,}$ que define a real probabilidade de descarte do pacote aumenta com o aumento de *count*.

4.4. *Weighted Random Early Detection* (WRED)

O algoritmo RED pode ser considerado um mecanismo justo de descarte de pacotes, uma vez que seleciona pacotes a serem descartados de forma aleatória, sem privilégio do tráfego ao qual os pacotes descartados pertencem.

Nas implementações dos mecanismos de controle de congestionamento é sempre interessante a possibilidade de privilegiar certas categorias de tráfego. É importante que o administrador da rede tenha um mecanismo que lhe possibilite a utilização de ponderação dos tráfegos com relação ao descarte de seus pacotes. Por exemplo, em uma rede VoIP, o tráfego de voz é transportado sobre UDP, que não reage ao descarte de pacotes. Portanto, o descarte de pacotes de voz resultará em perda de qualidade para o sinal de voz, mas não resultará na diminuição do tráfego na rede. Assim, pacotes de voz não devem ser descartados.

O WRED (*Weighted Random Early Detection* - Detecção Antecipada Aleatória Ponderada) é a extensão ponderada do mecanismo RED. Os pesos atribuídos a cada

fluxo ou a um grupo de fluxos são baseados na política de QoS que se deseja implementar. Diferentes pesos podem ser associados a cada classe de tráfego, resultando em diferentes comportamentos de descartes para cada classe [46].

O WRED define múltiplos valores de limiares, mínimos e máximos, a partir dos quais a rejeição de pacotes é iniciada, cujos valores são diferentes de acordo com as prioridades definidas. O WRED fornece limiares e pesos distintos para diferentes tipos de tráfego.

Deve ser notado que é bastante difícil ajustar os parâmetros W(RED), ou seja, diferentes probabilidades de descarte para diferentes tamanhos de filas, de uma maneira científica. Diretrizes apontando para as melhores formas de realizar estes ajustes ainda estão sendo desenvolvidas [66]. O artigo [65] conclui que um baixo atraso para uma probabilidade específica de perda pode ser obtida através da utilização de uma alta probabilidade máxima de descarte, um baixo valor para o limiar th_{min} e uma diferença pequena entre os limiares. Enquanto uma baixa probabilidade de descarte de pacotes pode ser alcançada através da utilização de uma baixa probabilidade máxima de descarte, um alto valor para o limiar th_{min} e uma ampla diferença entre os limiares. A escolha adequada destes parâmetros poderá fornecer um melhor efeito no comportamento requerido por um tráfego específico. Por exemplo, serviços em tempo real requerem baixo atraso, enquanto certos serviços de dados requerem baixa perda de dados.

4.5. Algoritmos AQM - *Backlog Based* x *Rate Based*

O *tail-drop* e os algoritmos RED se enquadram na categoria de algoritmos AQM do tipo *backlog based* (baseados na taxa de ocupação do *buffer*). Esta categoria de algoritmos

baseia-se no fato de que a intensidade do sinal de realimentação (*feedback*) em relação ao congestionamento é proporcional à taxa de ocupação do *buffer*. A taxa de descarte de pacotes está relacionada com o tamanho da fila.

Nestes algoritmos há uma indesejada relação na qual o aumento da sinalização de realimentação de congestionamento é resultado do aumento da taxa de ocupação do *buffer* [54]. Isto significa que, se a carga no enlace aumenta, o sinal de congestionamento (número de pacotes descartados nas conexões TCP) deve aumentar para evitar o estouro do buffer. E este aumento é proporcional à taxa de ocupação do buffer.

Os algoritmos AQM *backlog based* tratam o congestionamento a partir do nível de ocupação do *buffer* (*backlog*), atraso e perda.

Existe uma nova categoria de algoritmos de controle de congestionamento que desassociam a sinalização de congestionamento do nível de ocupação do *buffer*. Esta categoria de algoritmos é denominada *rate based*, pois determinam o nível de congestionamento através da taxa de chegada dos fluxos.

Os algoritmos AQM *rate based* são capazes de gerar os sinais de realimentação de congestionamento independentemente da ocupação do *buffer* porque estes estimam a taxa de chegada de pacotes para determinar o nível de congestionamento.

Desta forma, os algoritmos AQMs *rate based* podem determinar a demanda de banda do fluxo antes da criação do *backlog*, enquanto os algoritmos AQM *backlog based*

necessitam do *backlog* estabelecido para determinar e tratar o congestionamento.

Existem vários estudos, analíticos e experimentais, em relação aos algoritmos AQM *rate based* que indicam suas vantagens de utilização, revelando seu potencial para diminuir os *backlogs* e fornecer menor atraso e perdas aos fluxos na rede [54].

Os algoritmos REM e GREEN são 2 algoritmos AQM do tipo *rate based* [67] [68].

4.6. ECN - *Explicit Congestion Notification*

Dependendo do nível de congestionamento em um enlace, os algoritmos AQM descartam pacotes a uma taxa que reflete a severidade do congestionamento. Quando um pacote é descartado de um fluxo de pacotes em uma conexão TCP, o *host* de destino detecta esta perda e a comunica ao *host* de origem. Através deste reconhecimento, o *host* de origem pode estimar o nível de congestionamento na rede.

A taxa de perda de pacotes é a sinalização de realimentação de congestionamento fornecida ao *host* de origem em relação à conexão com o *host* de destino.

Descarte de pacotes tem sido um mecanismo natural de sinalização de congestionamento em conexões TCP. Contudo, outras formas de sinalização de realimentação de congestionamento, em substituição ao descarte de pacotes, tem sido estudadas e desenvolvidas. O ECN – *Explicit Congestion Notification* é um exemplo.

Com a utilização do ECN, os pacotes são marcados, por meio de um código, para

sinalizar a ocorrência de congestionamento e evitar a perda dos mesmos.

O ECN, definido na RFC 3168 [41], possibilita aos algoritmos AQM uma alternativa de sinalização de congestionamento. Sem o ECN, os algoritmos AQM ficam restritos ao descarte de pacotes como indicativo para sinalização de congestionamento.

Com a implementação do ECN, o algoritmo AQM pode modificar os bits CE (*Congestion Experienced*) no cabeçalho do datagrama IP ao invés de descartar o datagrama. Obviamente, este procedimento depende da disponibilidade do campo no protocolo IP e de sua compreensão na versão do protocolo TCP.

O uso do ECN permite que o receptor receba o datagrama com os bits CE marcados, evitando o atraso no aguardo da retransmissão, que seria necessário caso fosse utilizado o descarte de pacote como forma de sinalização.

4.6.1. Implementação do ECN

São utilizados dois bits no cabeçalho IP para designar o campo ECN. A RFC 3168, a qual obsoletou a RFC 2481, define dois bits que se encontravam reservados para uso futuro, no campo TOS (*Type of Service* – Tipo de Serviço) do Protocolo IP, versão 4 (IPv4).

Os dois bits ECN criam 4 códigos, definidos da seguinte forma:

00 – not-ECT -> ECT – ECN-*Capable Transport*

01 – ECT(0)

10 – ECT(1)

11 – CE -> CE - *Congestion Experienced*

O primeiro código, 00 – *not*-ECT, indica que o pacote não está utilizando o ECN, ou seja, nesta conexão a sinalização de congestionamento será através de descarte de pacotes.

Os códigos 01 e 10, ECT(0) e ECT(1) respectivamente, são setados pelo *host* emissor, indicando que os *hosts* fins da conexão TCP suportam o ECN. Os dois códigos, ECT(0) e ECT(1), são interpretados de maneira idêntica pelos roteadores da rede, e podem ser setados pelo emissor de maneira indistinta para indicar a capacidade ECN dos hosts da conexão TCP.

O código 11, CE – *Congestion Experienced*, é setado pelo roteador que está em estado de congestionamento.

O comportamento dos *hosts* em uma conexão TCP deve ser o mesmo para qualquer uma das formas de sinalização de congestionamento, seja por descarte de pacote, seja por habilitação do código CE. Deve-se enfatizar que o pacote IP com o código CE habilitado provoca no protocolo TCP do *host* emissor o mesmo efeito que o pacote descartado, em relação ao controle de congestionamento.

Da mesma forma, um roteador pode descartar um pacote *not*-ECT (não compatível com ECN) ou marcar o código CE em um pacote ECT (compatível com ECN), isto para um mesmo nível de congestionamento. Para um roteador, a marcação do código CE, para

um pacote compatível com ECN, deve ser realizada em uma situação na qual o roteador descartaria o pacote caso este fosse *not*-ECT.

Em uma situação na qual o roteador esteja recebendo pacotes e sua fila estiver cheia, este descartará pacotes, tal como faria em uma situação sem a implementação do ECN.

Caso o roteador receba um pacote com o código CE já marcado, este será encaminhado sem nenhuma alteração em relação ao campo ECN.

4.6.2. Suporte ECN no Protocolo TCP

Para o protocolo TCP, o ECN requer três novas características de funcionalidade:

- Negociação entre os *hosts* envolvidos durante o estabelecimento da conexão TCP, para determinar se ambos são ECT (ECN-*Capable Transport*).

- Um *flag* ECE (ECN-*Echo*) no cabeçalho TCP, para o receptor informar ao emissor, o recebimento de um pacote com o código CE habilitado.

- Um *flag* CWR (*Congestion Window Reduced*) no cabeçalho TCP, para o emissor informar ao receptor que a janela de congestionamento foi reduzida.

Os bits 8 e 9 do cabeçalho TCP, que eram bits reservados para uso futuro, são utilizados para representar os *flags* CWR e ECE respectivamente, dentro da implementação do ECN no protocolo TCP.

O ECN utiliza o ECT e o CE, no cabeçalho IP, para sinalização entre os roteadores e os *hosts* e utiliza o ECE e o CWR, no cabeçalho TCP, para sinalização entre os protocolos TCP dos *hosts* (transmissor e receptor).

Para uma conexão TCP, uma típica seqüência de eventos em uma reação ao congestionamento baseada no ECN é descrita a seguir:

- O código ECT é setado nos pacotes enviados pelo emissor para indicar que o ECN é suportado pelas camadas de transporte, ou seja, o protocolo TCP dos *hosts* envolvidos na conexão.

- Um roteador com capacidade ECN detecta congestionamento em uma de suas saídas e verifica que o código ECT está habilitado em um pacote que está para ser descartado. Em vez de descartar o pacote, o roteador escolhe marcar o código CE no cabeçalho IP do pacote e encaminha o mesmo.

- O receptor recebe o pacote com o código CE habilitado e seta o *flag* ECE (ECN-*echo*) no seu próximo TCP ACK a ser enviado ao emissor.

- O emissor recebe o TCP ACK com o ECE habilitado, e reage ao congestionamento como se o pacote tivesse sido perdido.

- O emissor seta o *flag* CWR no cabeçalho TCP do próximo pacote a ser enviado ao receptor para reconhecer que recebeu e reagiu ao *flag* ECE.

5. Simulações

Este capítulo apresenta os resultados das simulações que permitirão observar o comportamento dos algoritmos estudados. Utilizou-se o simulador "OPNET IT Guru Academic Edition", da OPNET Technologies, para preparar o ambiente e realizar as simulações.

5.1. Modelagem do tráfego

Para as simulações, realizou-se uma pesquisa para definir o comportamento típico dos tráfegos de voz, HTTP e FTP na *web*.

5.1.1. Tráfego HTTP

No tráfego HTTP uma sessão é composta de chamadas de pacotes (que representam o *download* das páginas *web*) e tempos de leitura. Em outras palavras, uma sessão HTTP é composta de períodos de ON e de OFF, resultantes da interação humana, onde as chamadas de pacotes representam as requisições dos usuários e os tempos de leitura representam o tempo necessário para o usuário analisar a informação recebida e decidir sobre uma nova requisição [69].

O HTTP é um protocolo do tipo requisição/resposta projetado para transferir arquivos da aplicação *web*. A página *web* pode ser vista como um objeto HTTP composto de um ou mais arquivos HTML (*Hyper Text Markup Language*), juntamente com outros elementos que possam vir a compor uma página *web* (imagens, sons, animações, etc...)

[70].

Foram pesquisados quatro parâmetros associados ao tráfego HTTP pertinentes às simulações realizadas, a saber:

- Tamanho do objeto principal (*main object size*): pesquisado nas referências [69], [71], [72], [73], [74], [75] e [76].
- Tamanho dos objetos embutidos (*embedded object size*): pesquisado nas referências [69], [71], [72], [73], [74], [75] e [76].
- Quantidade de objetos embutidos por página (*number of embedded objects per page*): pesquisado nas referências [69], [71], [73] e [74].
- Tempo entre requisições de leitura (*page interarrival time*): pesquisado nas referências [69], [73] e [74].

Embora a média e a mediana descrevam o tamanho de uma página típica, a distribuição de probabilidade oferece uma indicação melhor da variabilidade dos parâmetros avaliados. Trabalhos como [70], [76] e [77] relatam que a distribuição que melhor caracteriza os tamanhos do objeto principal e dos objetos embutidos é a distribuição de Pareto. Outros trabalhos, como [69] e [74] atribuem a distribuição Lognormal como a que melhor caracteriza estes parâmetros, enquanto outros trabalhos, como [72] e [78] fazem esta atribuição a uma distribuição híbrida (Pareto + Lognormal).

Para a quantidade de objetos embutidos, a maioria dos trabalhos aponta para a distribuição de Pareto como a melhor opção para caracterizá-la, enquanto para o tempo entre requisições de leitura a distribuição exponencial é considerada a melhor. Os

tempos entre requisições de leitura representam o tempo necessário para o usuário analisar a informação recebida e decidir sobre uma nova requisição.

O Anexo 2 traz as distribuições de probabilidades citadas neste trabalho, com as expressões e definições dos seus parâmetros.

O trabalho [71] apresenta um estudo referenciado em inúmeros trabalhos de estudo de modelagem de tráfego disponíveis na literatura, resultando em uma síntese dos valores utilizados pelos inúmeros trabalhos pesquisados.

Com base nas referências citadas, montou-se a Tabela 5.1, que representa a escolha da distribuição e dos respectivos parâmetros utilizados para a configuração da aplicação HTTP nas simulações realizadas.

Tráfego	Características	Distribuição	Parâmetros
HTTP	Tamanho do Objeto Principal (bytes)	Pareto	Média = 11000
			Localização (Location) k = 1833
			Curvatura (Shape) α = 1,2
	Tamanho dos Objetos Embutidos (bytes)	Pareto	Média = 8000
			Localização (Location) k = 1333
			Curvatura (Shape) α = 1,2
	Quantidade de Objetos Embutidos por Página	Pareto	Localização (Location) k = 1
			Curvatura (Shape) α = 2,43
	Tempo entre Requisições de Leitura (segundos)	Exponencial	Média (Mean Outcome) β = 30

Tabela 5.1 – Características do tráfego HTTP.

5.1.2. Tráfego FTP

Uma sessão FTP consiste de uma seqüência de transferência de arquivos, separadas por tempos de leitura. Desta forma há apenas dois parâmetros no modelamento do tráfego

FTP: o tamanho do arquivo que será transferido e o tempo de leitura do usuário (tempo entre requisições), que representa o tempo entre o término do *download* do arquivo anterior e a requisição do usuário por um novo arquivo [69].

Em relação ao tamanho dos arquivos transferidos, pesquisas mostram uma grande heterogeneidade. Ou seja, os trabalhos mostram que existem grandes variações nos tamanhos médios de arquivos transferidos [70].

Apesar do FTP representar um importante serviço, nos últimos anos esta importância vem diminuindo em contrapartida ao crescimento dos *downloads* HTTP. Por esta razão, as referências na literatura relativas à modelagem do FTP são bastante antigas [70].

A maioria dos trabalhos aponta a distribuição de Pareto ([70], [72] e [73]) para a representação das características da variação do tamanho do arquivo e a distribuição exponencial ([69], [70] e [73]) para o tempo entre requisições.

Com base nas referências citadas, e particularmente na referência [69] para o tamanho do arquivo e o tempo entre requisições, montou-se a Tabela 5.2, que representa a escolha da distribuição e dos seus respectivos parâmetros utilizados para a configuração da aplicação FTP nas simulações realizadas.

Tráfego	Características	Distribuição	Parâmetros
FTP	Tamanho do Arquivo (bytes)	Pareto	Média = 2000000
			Localização (Location) k = 181818
			Curvatura (Shape) $\alpha = 1,1$
	Tempo entre Requisições (segundos)	Exponencial	Média (Mean Outcome) $\beta = 180$

Tabela 5.2 – Características do tráfego FTP

5.1.3. Tráfego de Voz

O tráfego de voz foi configurado para representar a telefonia IP com codec G.729. O esquema de codificação G.729 produz quadros de 80 bits (10 bytes) codificando 10 ms de fala a uma taxa de 8 kbit/s [36]. Desta forma, o tráfego de voz gerado é de 100 pacotes/s ou 1000 bytes/s. Não foi configurada a opção de supressão de silêncio no codec G.729.

O *overhead* do protocolo UDP para o quadro de voz é de 8 bytes e o do protocolo IP é de 20 bytes. Logo o tamanho do pacote de voz, incluindo o *overhead*, é de 38 bytes. A taxa de bits efetiva é então igual a 30400 bps (38 bytes / pacote * 8 * 100 pacotes / segundo) [79].

5.2. Simulações

Nos subitens a seguir são apresentados alguns ambientes de simulação e os resultados obtidos. O detalhamento da configuração no simulador "OPNET IT Guru Academic Edition" é apresentado no Anexo 1.

5.2.1. Simulação 1

Esta simulação tem por objetivo analisar e comparar os mecanismos de priorização: FIFO, PQ, WFQ e CQ. A Figura 5.1 apresenta o ambiente da simulação 1.

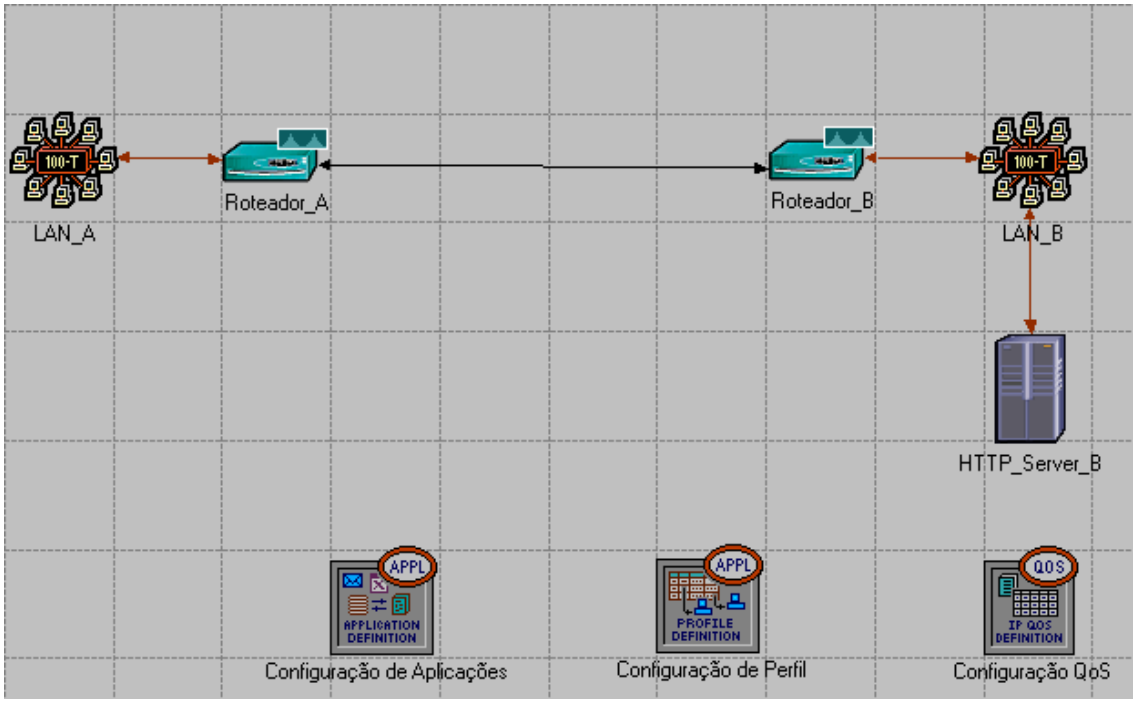


Figura 5.1 – Ambiente da Simulação 1

Nesta simulação as interfaces dos roteadores A e B, interligados por um enlace PPP de 512 Kbps, foram configuradas com os mecanismos de priorização (FIFO, PQ, WFQ e CQ) e avaliados quanto à proteção oferecida ao tráfego de voz em três situações: 5, 10 e 15 clientes de voz na LAN_A (acessando um servidor na LAN_B), ocupando aproximadamente 30%, 60% e 90% do enlace, respectivamente. Para cada uma destas 3 situações e para cada um dos 4 mecanismos de priorização, configurou-se de forma crescente (0, 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 150, 200, 300, 400 e 500) o número de clientes HTTP na LAN_A, os quais se conectam ao Servidor_HTTP_B localizado na LAN_B, de modo a variar a utilização do enlace de 30%, 60% e 90% até 100% (soma da utilização devido ao tráfego de voz e tráfego HTTP).

O mecanismo PQ foi configurado para priorizar voz. Configurou-se a aplicação de voz

para TOS igual a 6 (*Interative Voice*) e a aplicação HTTP para TOS igual a 0 (*Best Effort*). Para os mecanismos WFQ e CQ foram configurados 2% de folga para o tráfego de voz previsto para utilizar o enlace, ou seja, 32%, 62% e 92%. A Tabela 5.3 detalha as configurações dos mecanismos WFQ e CQ.

Configurações dos mecanismos CQ e WFQ	30%	60%	90%
Número de clientes de voz	5	10	15
Tráfego real de voz (bits/sec)	152000	304000	456000
Percentual real de utilização do enlace (%)	29,6875	59,375	89,0625
Configuração do contador de bytes (voz / HTTP) no mecanismo **CQ**	1600 / 3400	3100 / 1900	4600 / 400
Configuração do contador de bytes (voz / HTTP) no mecanismo **CQ** (%)	32 / 68	62 / 38	92 / 8
Configuração dos pesos (voz / HTTP) no mecanismo **WFQ**	32 / 68	62 / 38	92 / 8
Configuração dos pesos (voz / HTTP) no mecanismo **WFQ** (%)	32 / 68	62 / 38	92 / 8
Tráfego reservado para voz nos mecanismos **CQ** e **WFQ** (bits/sec)	163840	317440	471040

Tabela 5.3 – Configurações do mecanismo CQ e WFQ

Para todos os mecanismos de priorização verificados nesta simulação, cada fila foi configurada para armazenar no máximo 500 pacotes.

Nas Figuras 5.2, 5.3 e 5.4 são apresentados os resultados obtidos para o atraso médio fim-a-fim dos pacotes de voz (*voice packet end-to-end delay*) entre o servidor na LAN_B e os clientes na LAN_A. Deve-se notar nestas figuras e nas demais figuras deste capítulo, que o parâmetro foi observado em relação a utilização do enlace, tanto em percentual (%) quanto em quantidade de clientes, para possibilitar uma melhor visualização do comportamento do parâmetro em questão. Por exemplo, na Figura 5.2, para 60 clientes HTTP, a utilização do enlace já é superior a 90% e, neste caso, pode-se observar melhor a tendência de atraso máximo de aproximadamente 1,5 segundos para o mecanismo FIFO e aproximadamente 1,0 segundo para o mecanismo CQ no gráfico que apresenta a utilização do enlace em quantidade de clientes.

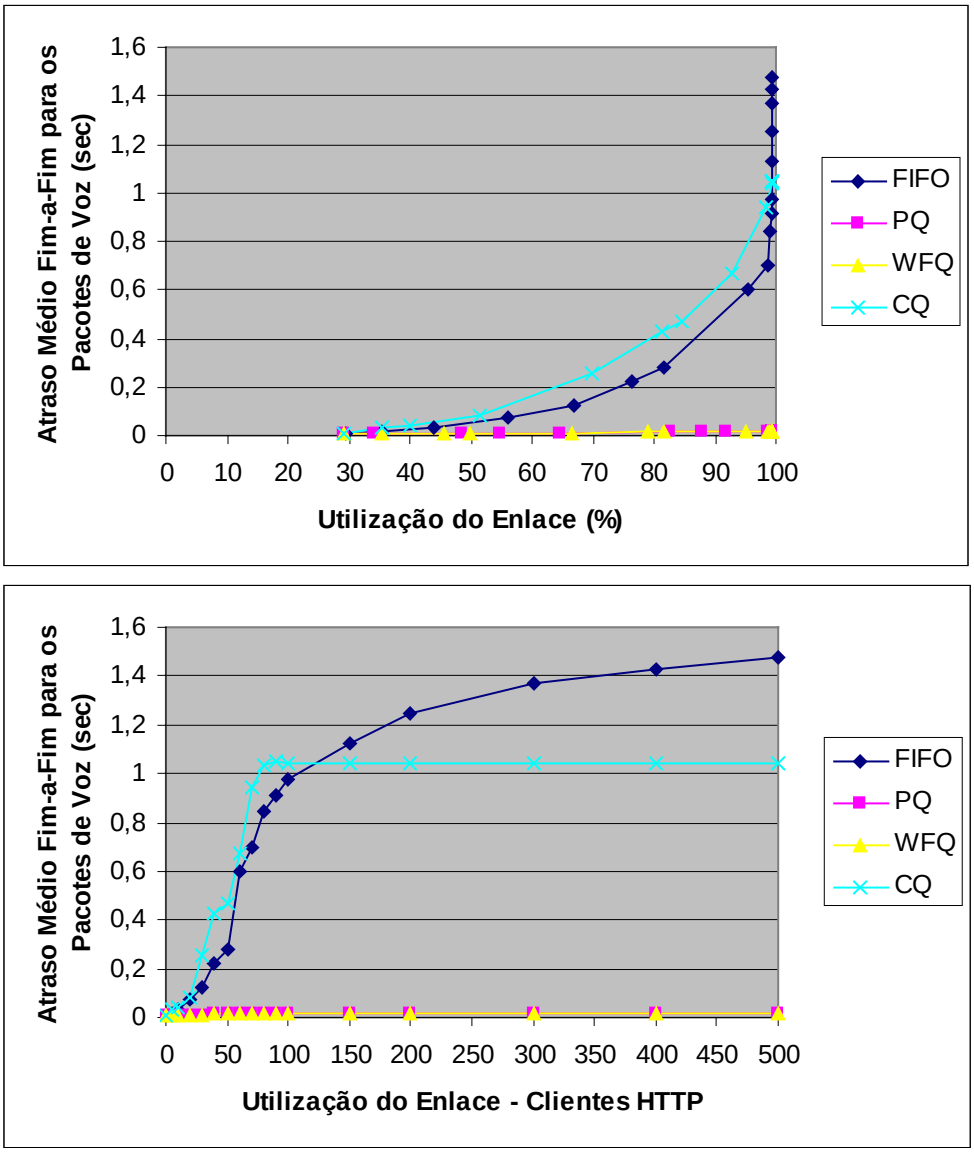


Figura 5.2 – Simulação com tráfego de voz ocupando 30% do enlace

Obs.: pode-se observar que nas Figuras 5.2, 5.3 e 5.4, com a utilização do mecanismo FIFO, o atraso máximo para o tráfego de voz se torna menor quanto maior a quantidade de usuários de voz. Apesar de parecer incoerente, este comportamento se explica pelo fato do tamanho máximo da fila configurada para este mecanismo ser de 500 pacotes. Uma vez que o tamanho do pacote de voz é fixo e igual a 38 bytes e o tamanho do pacote HTTP apresenta tamanho variável entre 516 e 1500 bytes, o tamanho máximo da

fila em *bytes* varia conforme a quantidade de pacotes de voz e HTTP que ocupam a fila. Sendo o tamanho dos pacotes de voz bem menor que os pacotes HTTP, quanto maior for a quantidade de usuários de voz, maior será o número de pacotes de voz povoando a fila, diminuindo seu tamanho em *bytes* e conseqüentemente diminuindo o atraso. A Figura 5.5 demonstra este comportamento. A Figura 5.6 apresenta a perda de pacotes IP (tráfego HTTP e voz) para o mecanismo FIFO.

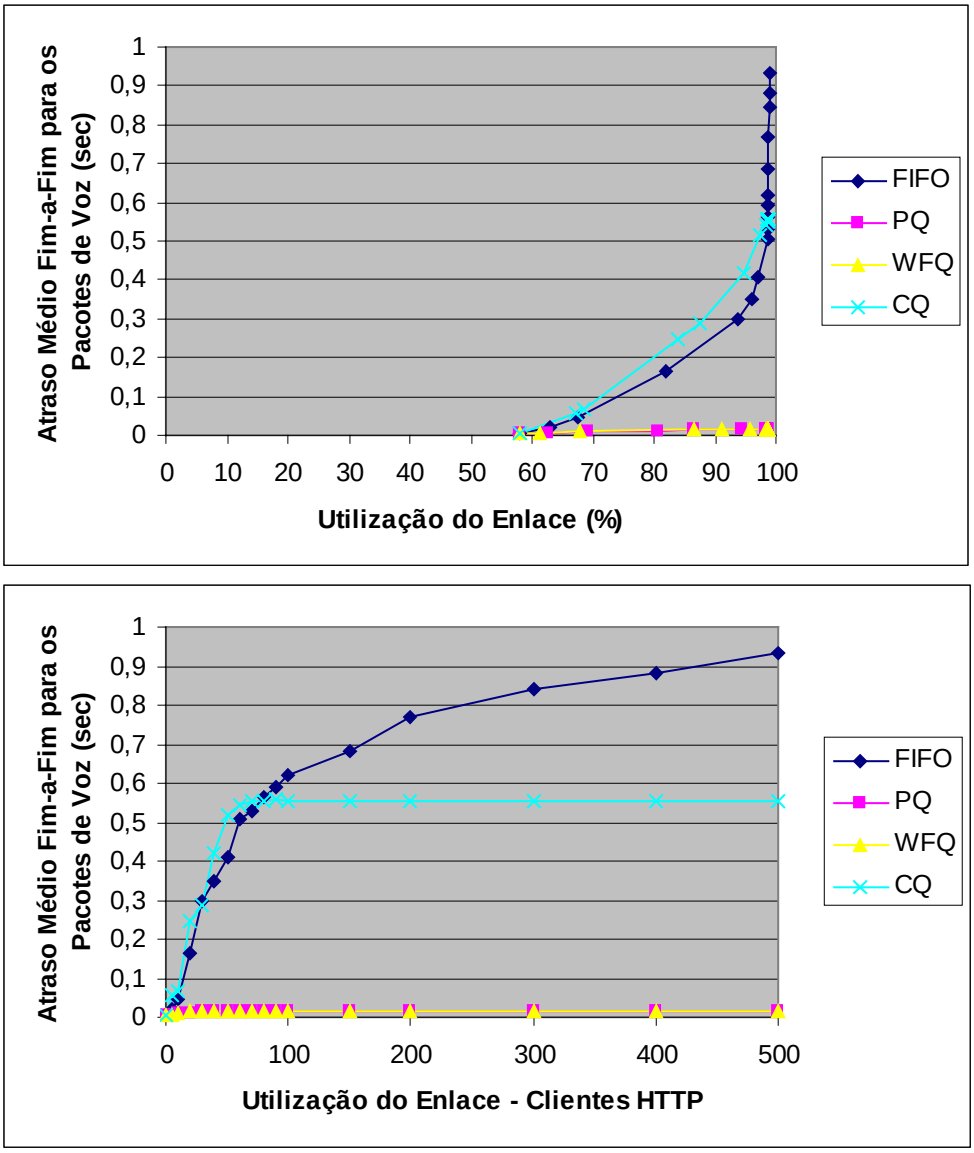


Figura 5.3 – Simulação com tráfego de voz ocupando 60% do enlace

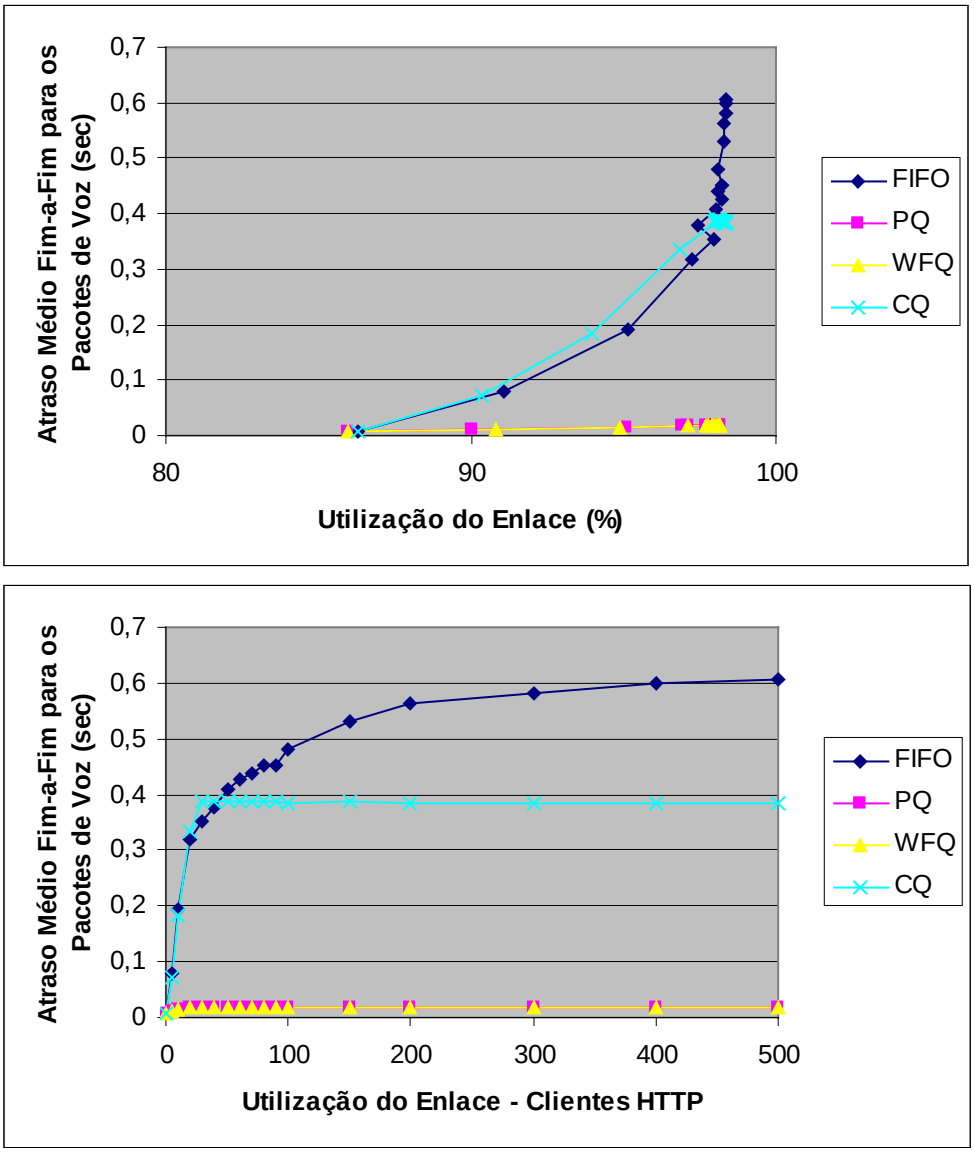


Figura 5.4 – Simulação com tráfego de voz ocupando 90% do enlace

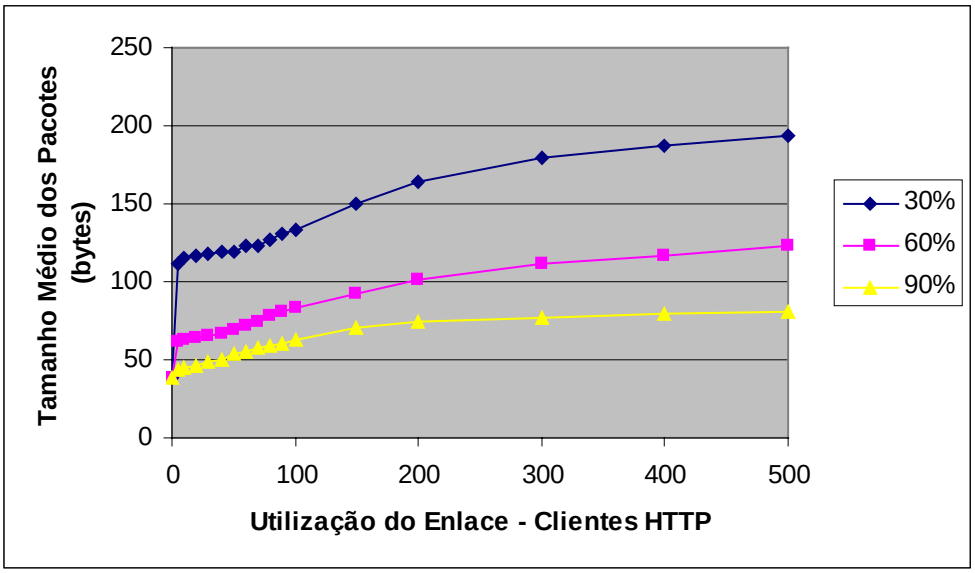


Figura 5.5 – Tamanho médio dos pacotes para o mecanismo FIFO para 30%, 60% e 90% de ocupação do tráfego de voz

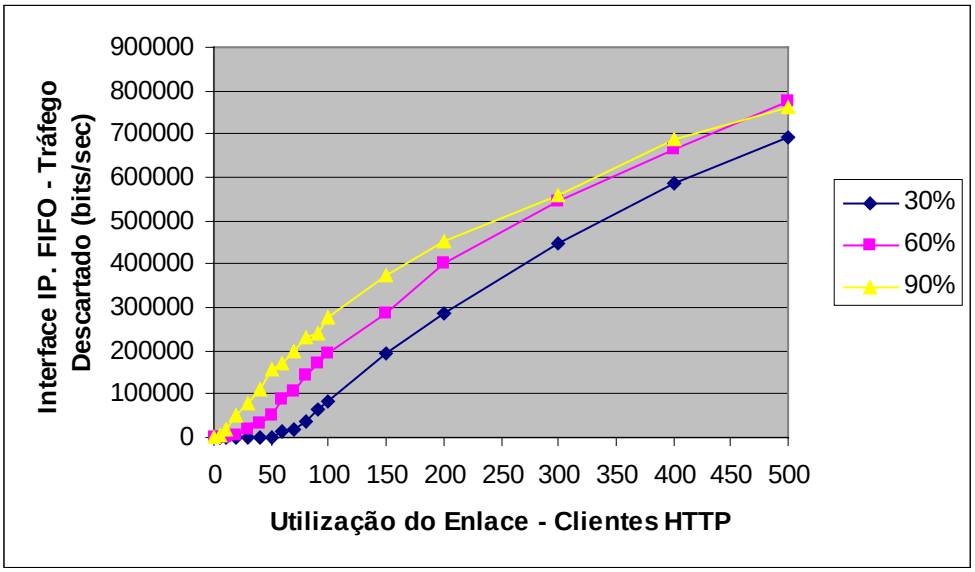


Figura 5.6 – Tráfego IP descartado no mecanismo FIFO para 30%, 60% e 90% de ocupação do tráfego de voz

Para o ambiente da simulação em questão, o mecanismo CQ não foi capaz de proteger o tráfego de voz com níveis adequados para o atraso, e também apresentou descarte de pacotes de voz (Figura 5.7), quando o tráfego de voz estava consumindo 30%, 60% e

90% do enlace.

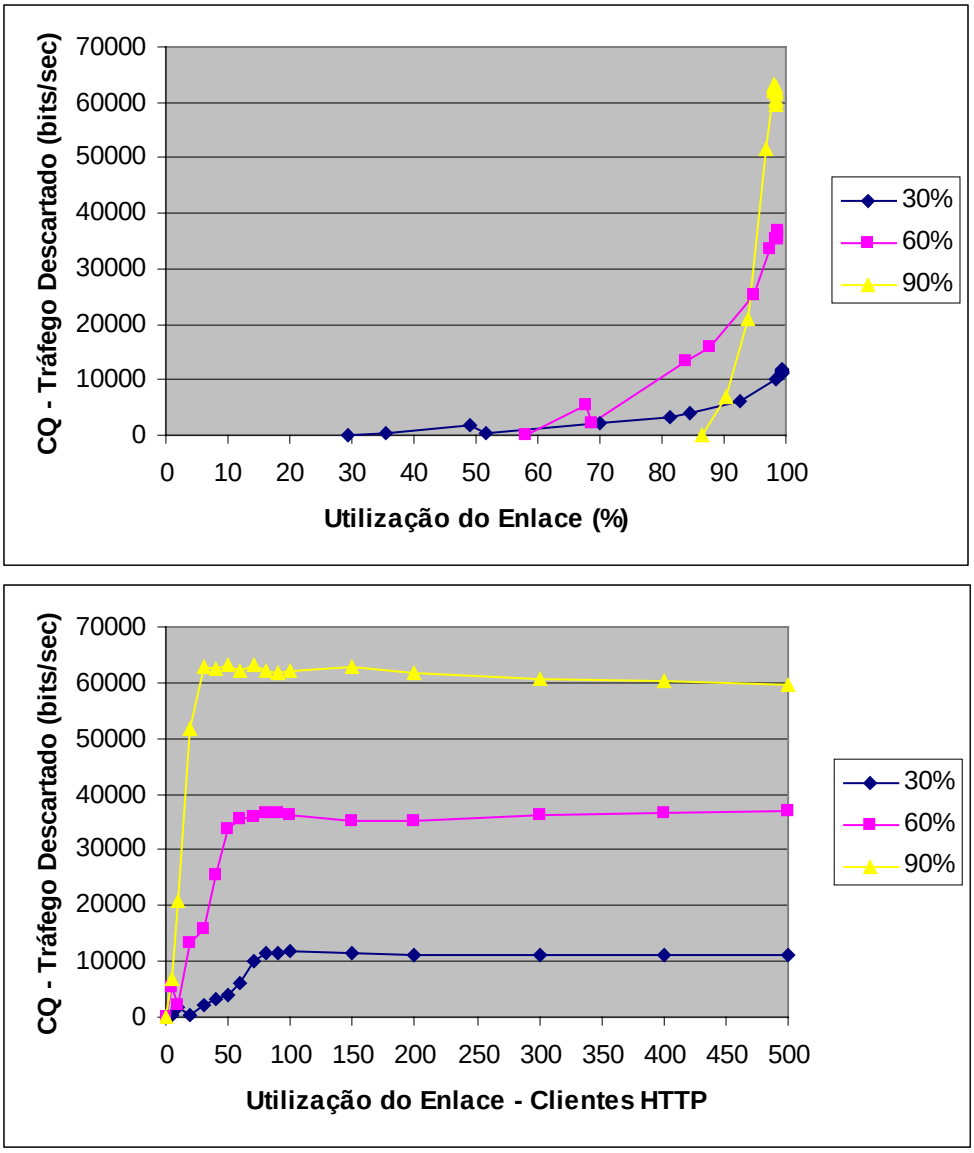


Figura 5.7 – Perdas para as simulações com tráfego de voz com o mecanismo CQ.

Existe uma grande dificuldade de configuração do mecanismo CQ em relação aos tráfegos que compartilham o enlace, uma vez que sua configuração, baseada em contador de *bytes* (*byte count*) para dimensionar o percentual do servidor para atender cada tipo de tráfego, se mostra bastante inadequado para este propósito. Nesta simulação, o tamanho do pacote de voz é fixo e igual a 38 bytes, enquanto o pacote

HTTP apresenta tamanho variável entre 516 e 1500 bytes. Desta forma, conforme exemplificado na Seção 3.2.3, a atribuição dos pesos a cada tráfego sendo dimensionada através do contador de bytes torna extremamente difícil uma configuração adequada do mecanismo CQ para os dois tráfegos (voz e HTTP) e se tornaria ainda pior para uma quantidade maior de tráfegos. Fica evidenciado que apesar de ter sido dimensionado o percentual suficiente para suportar o tráfego de voz através do contador de *bytes* do mecanismo CQ nas simulações realizadas (ver Tabela 5.3), o mecanismo não conseguiu implementar justiça e conseqüentemente proteger o tráfego de voz nas simulações referentes a 30%, 60% e 90% de ocupação do enlace para o tráfego de voz. Vale lembrar que para o mecanismo WFQ foram utilizados os mesmos percentuais e também o mesmo tamanho máximo de fila (500 pacotes por fila).

Para melhor verificar a atuação deste mecanismo (CQ), as Figura 5.8, 5.9 e 5.10 apresentam o comportamento deste mecanismo para a situação de ocupação do enlace em 30% para o tráfego de voz e 150 clientes HTTP, variando os valores do contador de *bytes* (*byte count*) mantendo a mesma proporção da relação entre tráfego de voz e tráfego HTTP, ou seja, 32% e 68% respectivamente. Nota-se que a atribuição de valores muito pequenos para os contadores resulta em uma grande discrepância para as bandas alocadas devido à grande diferença de tamanho entre os pacotes de voz e HTTP, gerando grande atraso para o tráfego de voz. Para valores grandes, a discrepância diminui, porém aumenta o atraso associado ao crescimento da fila HTTP. Esta característica também influencia nas perdas do tráfego de voz (Figura 5.9) e verificamos na Figura 5.10 que o mecanismo CQ só consegue prover justiça nas atribuições de cada tráfego (32% e 68%) para alguns valores atribuídos ao contador de *bytes*.

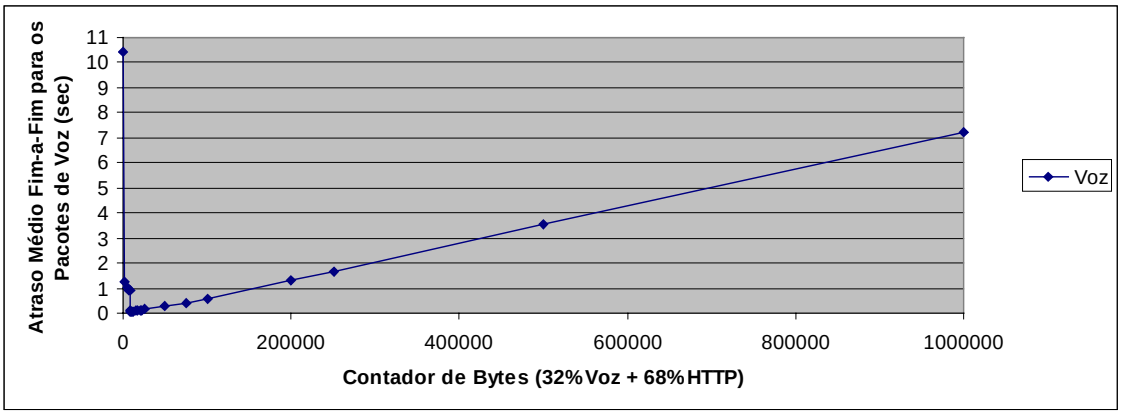


Figura 5.8 – Atraso em relação à variação do contador de bytes no mecanismo CQ.

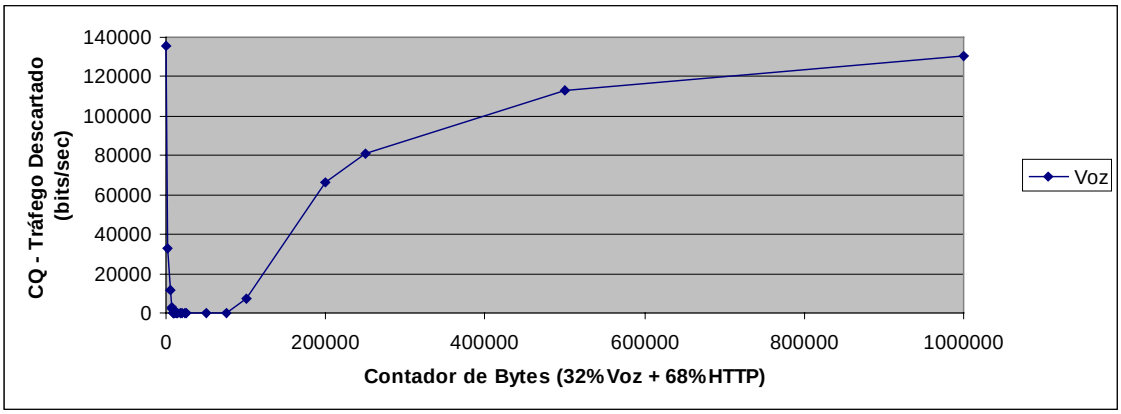


Figura 5.9 – Perdas em relação à variação do contador de bytes no mecanismo CQ.

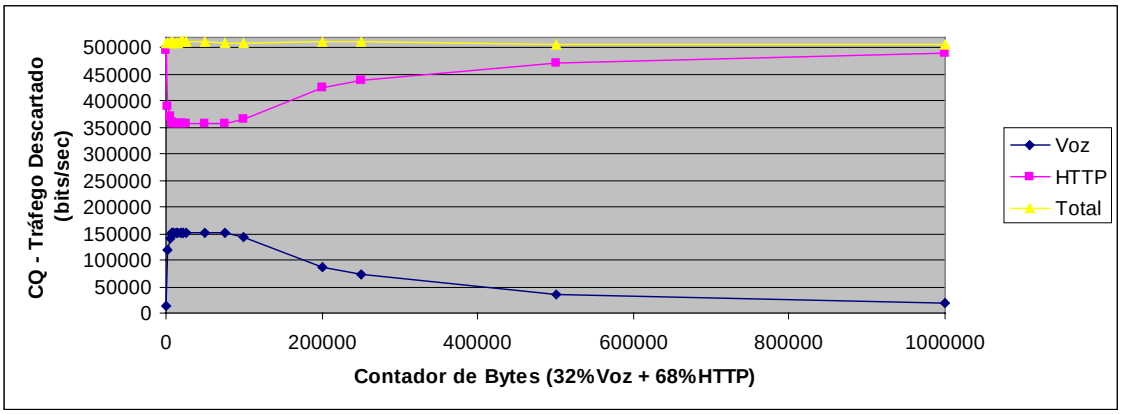


Figura 5.10 – Tráfego transmitido (Roteador_B para Roteador_A) em relação à variação do contador de bytes no mecanismo CQ

Avaliando os resultados, pode-se verificar que os mecanismos PQ e WFQ mantiveram o atraso sempre menor que 20 ms e sem nenhum descarte de pacotes. Para uma melhor visualização, a Figura 5.11 apresenta o comportamento do atraso fim-a-fim para estes dois mecanismos. A Figura 5.11 foi extraída das Figuras 5.2, 5.3 e 5.4.

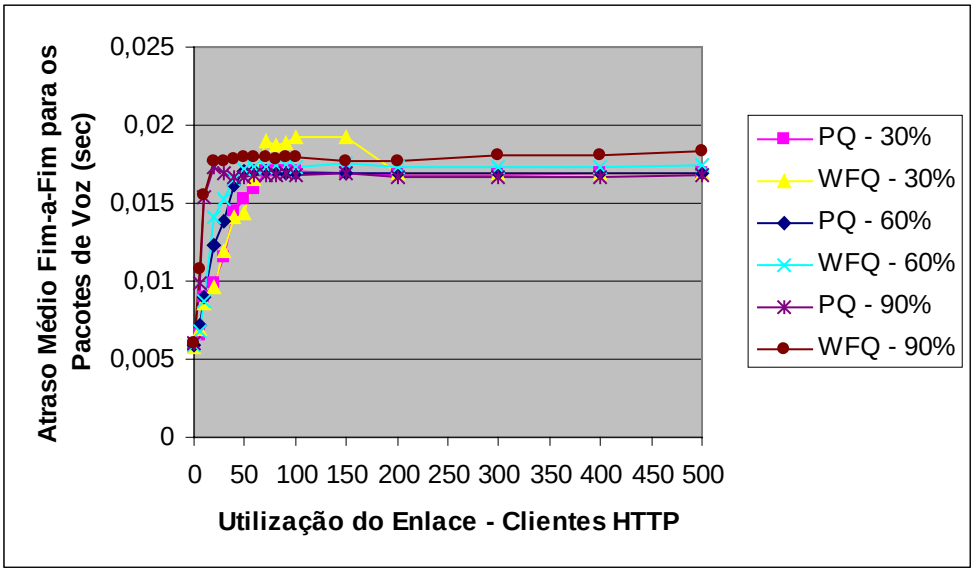


Figura 5.11 – Atraso para o tráfego de voz para os mecanismos PQ e WFQ

O mecanismo PQ se apresenta como uma boa solução para priorização do tráfego de voz em relação a outros tráfegos, porém para uma situação onde se pretende priorizar voz e compartilhar o restante da banda de forma balanceada para os demais tráfegos, o mecanismo PQ não é adequado. Uma adequação para esta situação seria sua utilização casada com o WFQ, tendo como exemplo a implementação LLQ (*Low Latency Queueing*) da Cisco [29] [30].

Para uma situação com inúmeros tráfegos, o mecanismo WFQ apresenta os recursos adequados para distribuição da banda entre os tráfegos, com a atribuição ponderada

desejada. Na simulação 2 seguinte, veremos o comportamento deste mecanismo e também sua utilização com o WRED e o ECN.

O comportamento do atraso percebido pelo tráfego HTTP para as simulações realizadas pode ser observado na Figura 5.12 para o tráfego de voz ocupando 30% do enlace e na Figura 5.13 para o tráfego de voz ocupando 60% do enlace. Uma vez que o tráfego priorizado é o tráfego de voz, é natural que o atraso irá aumentar para o tráfego HTTP quando se utiliza qualquer mecanismo de priorização em lugar do mecanismo FIFO. Porém vale observar que para níveis de utilização do enlace em até aproximadamente 95%, que representa aproximadamente 60 usuários HTTP para o tráfego de voz ocupando 30% do enlace, aproximadamente 40 usuários HTTP para o tráfego de voz ocupando 60% do enlace ou aproximadamente 10 usuários HTTP para o tráfego de voz ocupando 90% do enlace, estes aumentos de atraso que o tráfego HTTP sofre são pequenos e perfeitamente suportáveis. Em contrapartida, o atraso para o tráfego de voz para este mesmo nível de ocupação do enlace, quando utilizados os mecanismos PQ e WFQ são ínfimos e a comunicação de voz se torna viável graças à utilização destes dois mecanismos. As Figuras 5.14, 5.15 e 5.16, e as Figuras 5.17, 5.18 e 5.19 mostram respectivamente, o comportamento do atraso para o tráfego HTTP e voz, dentro da faixa de ocupação do link que se deseja observar (até 95% de ocupação do enlace).

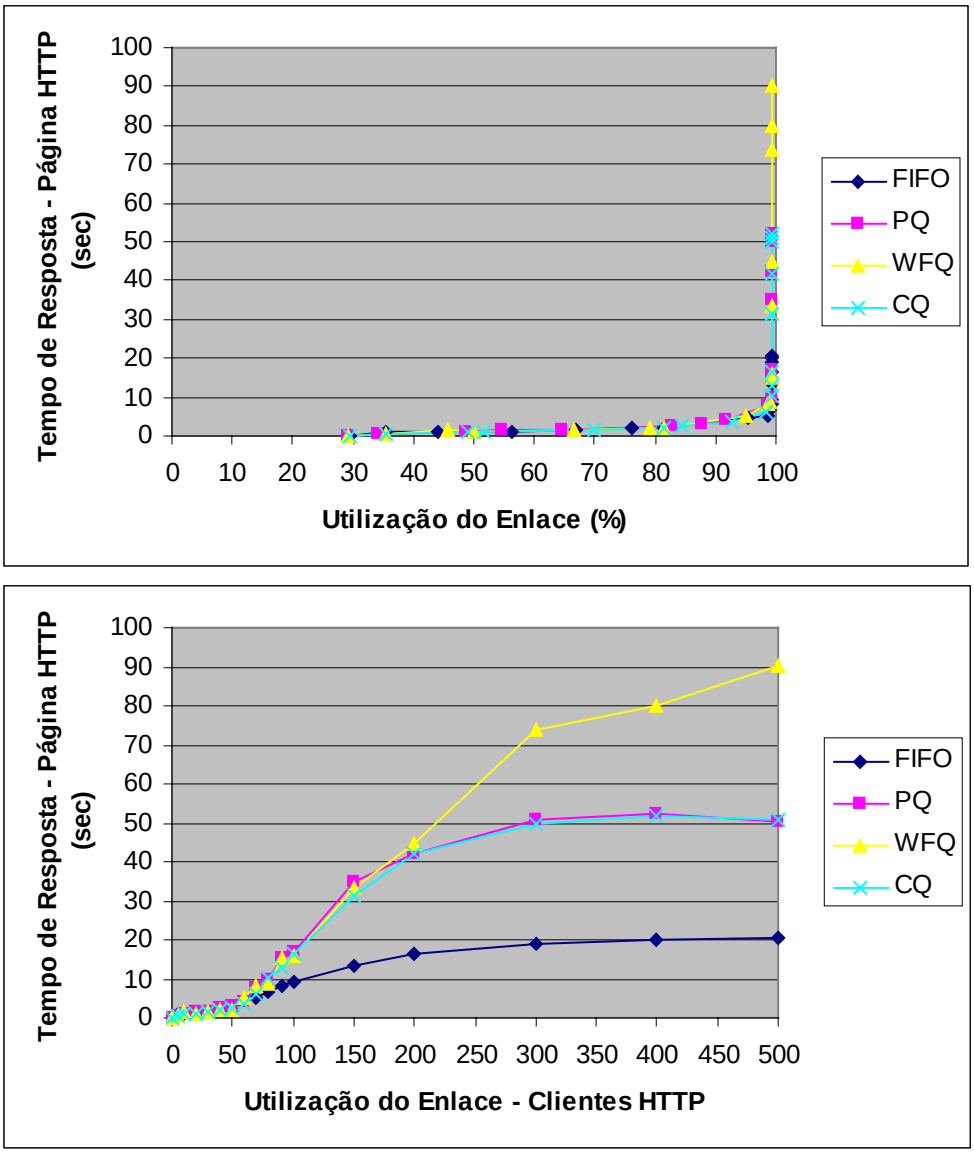


Figura 5.12 – Tempo Médio de Resposta da Página HTTP (p/ tráfego de voz ocupando 30% do enlace)

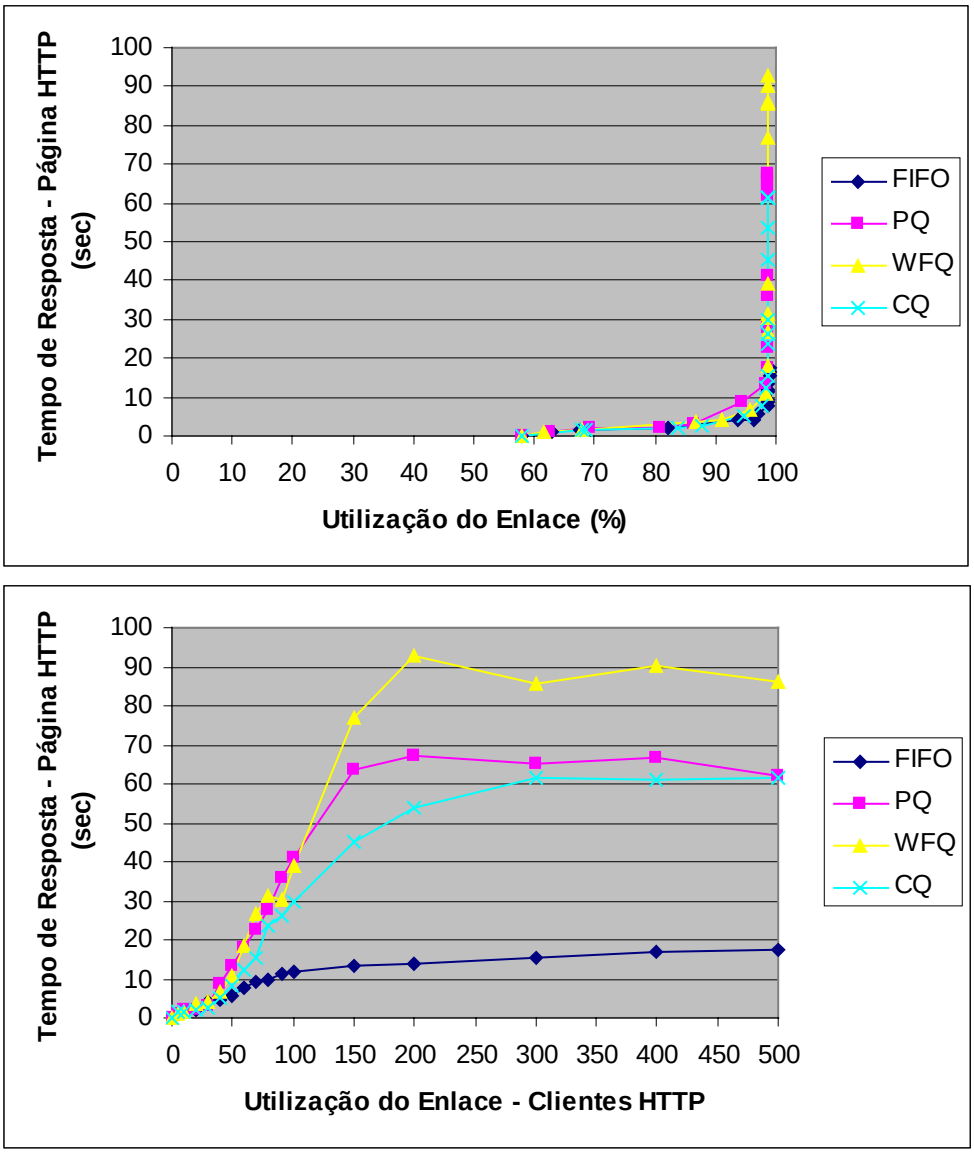


Figura 5.13 – Tempo Médio de Resposta da Página HTTP (p/ tráfego de voz ocupando 60% do enlace)

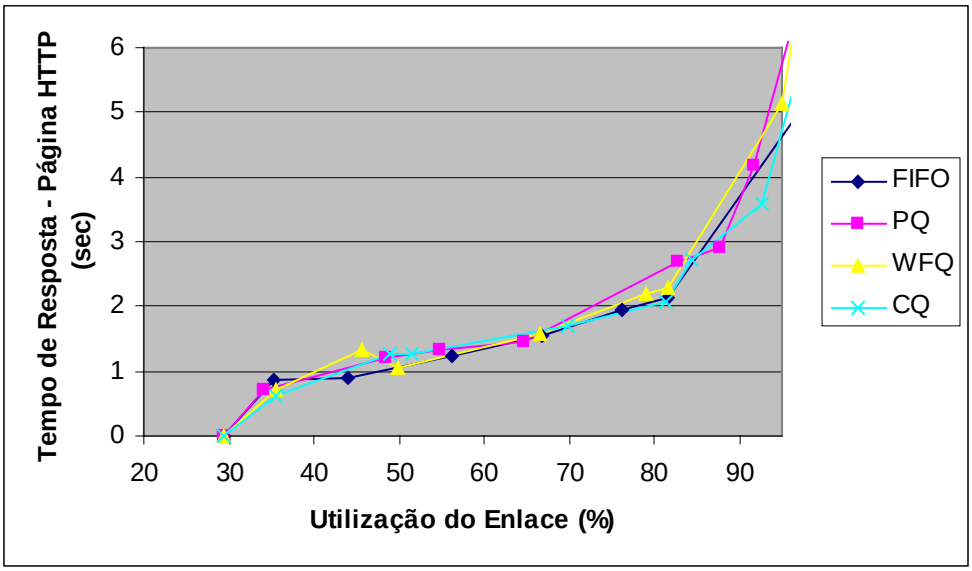


Figura 5.14 – Tempo Médio de Resposta da Página HTTP (p/ até 95% de ocupação total do enlace e tráfego de voz ocupando 30% do enlace)

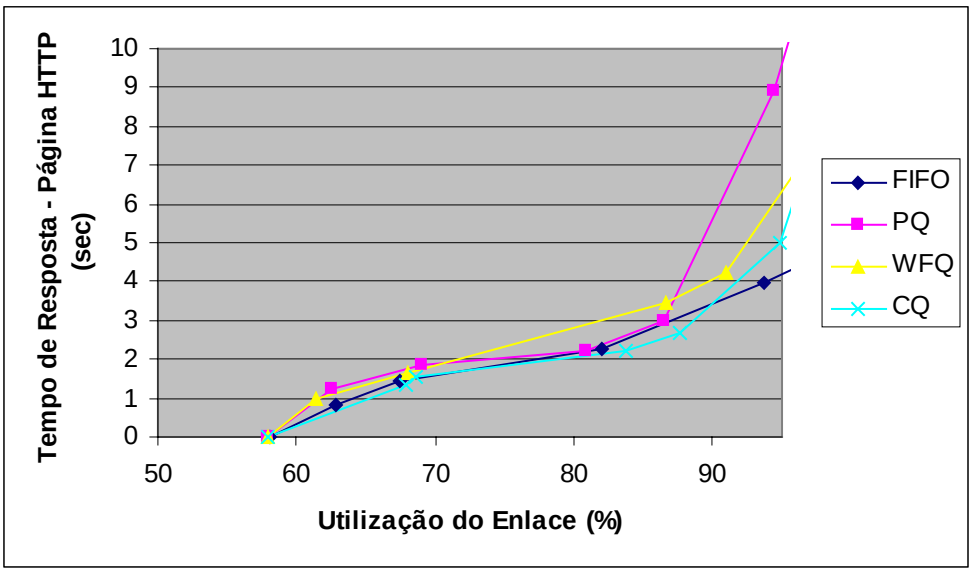


Figura 5.15 – Tempo Médio de Resposta da Página HTTP (p/ até 95% de ocupação total do enlace e tráfego de voz ocupando 60% do enlace)

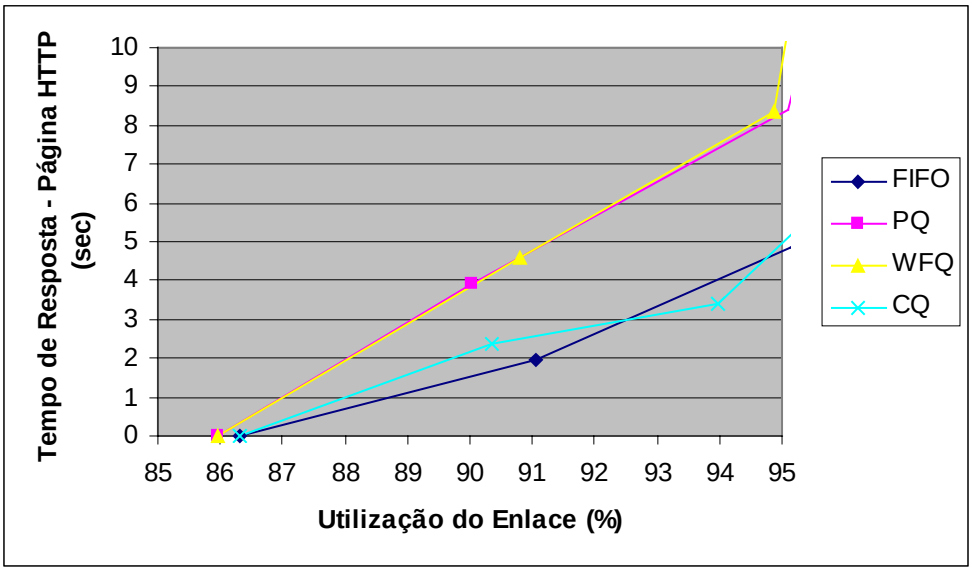


Figura 5.16 – Tempo Médio de Resposta da Página HTTP (p/ até 95% de ocupação total do enlace e tráfego de voz ocupando 90% do enlace)

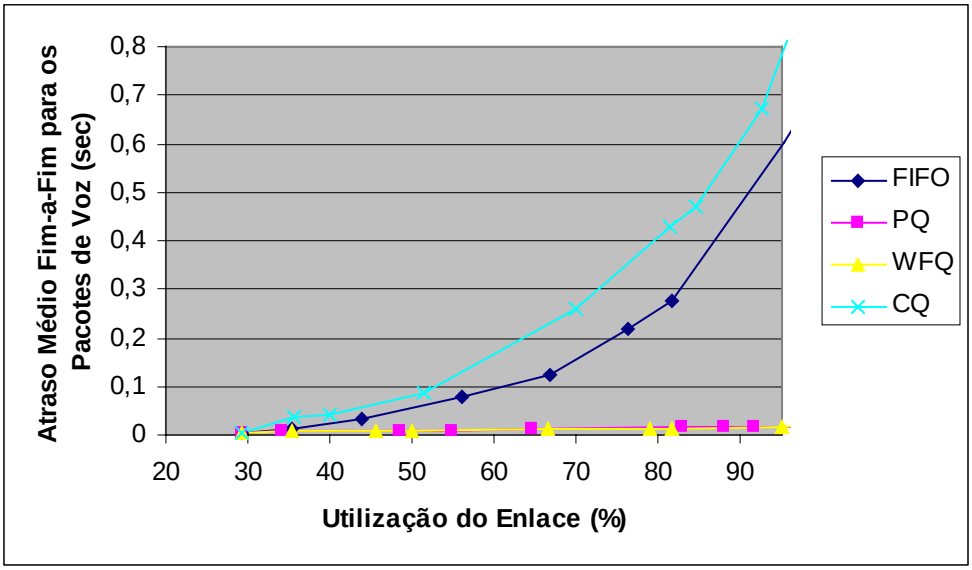


Figura 5.17 – Atraso Médio Fim-a-Fim para os Pacotes de Voz (p/ até 95% de ocupação total do enlace e tráfego de voz ocupando 30% do enlace)

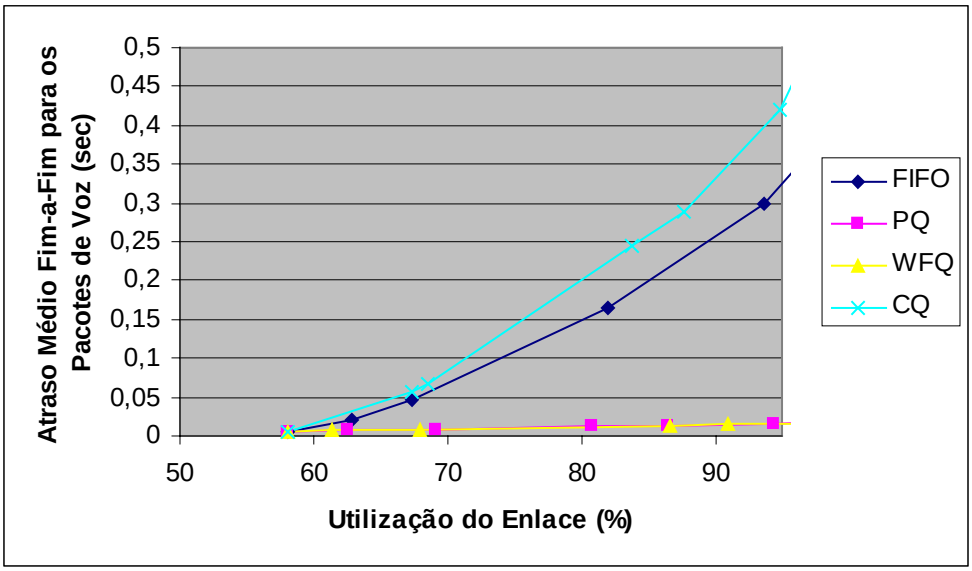


Figura 5.18 – Atraso Médio Fim-a-Fim para os Pacotes de Voz (p/ até 95% de ocupação total do enlace e tráfego de voz ocupando 60% do enlace)

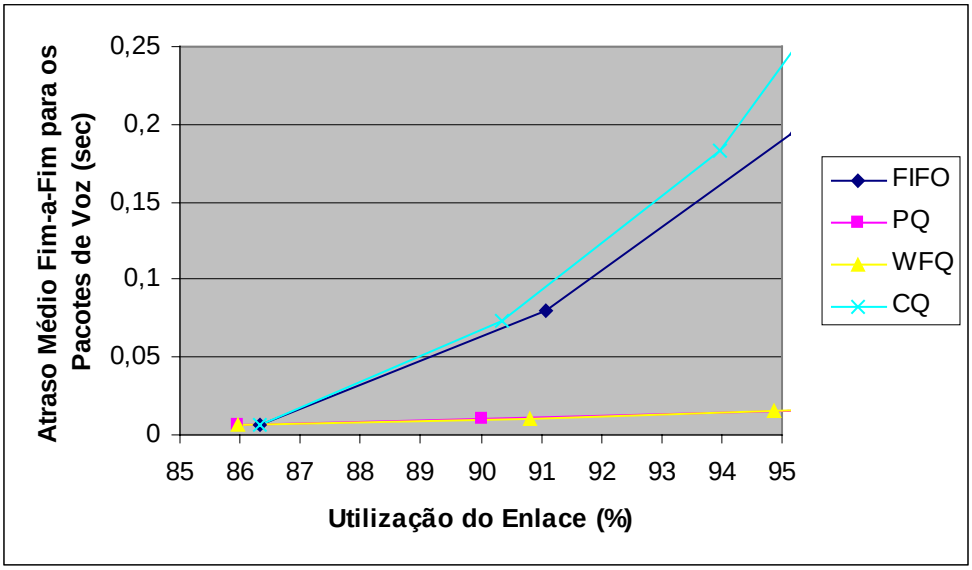


Figura 5.19 – Atraso Médio Fim-a-Fim para os Pacotes de Voz (p/ até 95% de ocupação total do enlace e tráfego de voz ocupando 90% do enlace)

Utilizando o mesmo ambiente da simulação 1 (Figura 1), com os mecanismos CQ e WFQ configurados para 30% de reserva para o tráfego de voz (Tabela 5.3), foi realizada nova simulação, desta vez variando o número de clientes de voz na LAN_A (1, 2, 3, 4,

5, 6, 7, 8, 9, 10, 12, 14, 16, 18, 20 e 25) acessando um servidor na LAN_B, mantendo fixo em 200 o número de clientes HTTP na LAN_A, os quais se conectam ao Servidor_HTTP_B localizado na LAN_B. A Figura 5.20 apresenta o atraso médio fim-a-fim dos pacotes de voz entre o servidor na LAN_B e os clientes na LAN_A. A Figura 5.21 apresenta o tráfego de voz descartado.

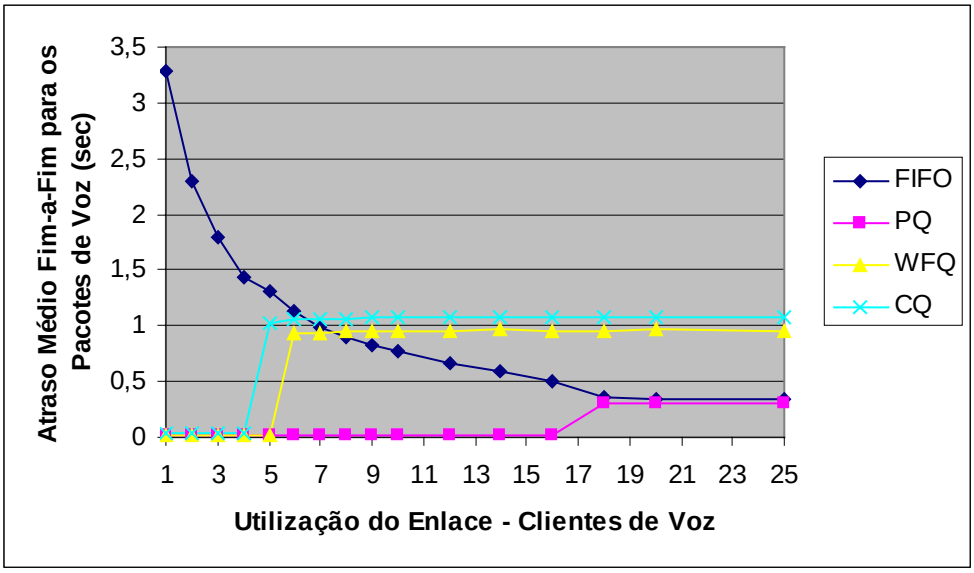


Figura 5.20 – Simulação com variação dos clientes de voz mantendo em 200 os clientes HTTP

Nas Figuras 5.20 e 5.21 pode-se verificar que o mecanismo PQ manteve um nível adequado de atraso (menor que 20 ms) e nenhuma perda enquanto o número de clientes de voz exigia menos que 512 Kbps, ou seja, 16 clientes de voz (512 Kbps / 30,4 Kbps = 16,84). Desta forma, verificou-se que o mecanismo PQ protegeu o tráfego de voz até o estouro da capacidade do enlace (512 Kbps). O mecanismo WFQ manteve o nível adequado de atraso (menor que 20 ms) e nenhuma perda enquanto o número de clientes de voz consumia menos que a banda reservada (163840 bps – Tabela 5.3), ou seja, 5 clientes.

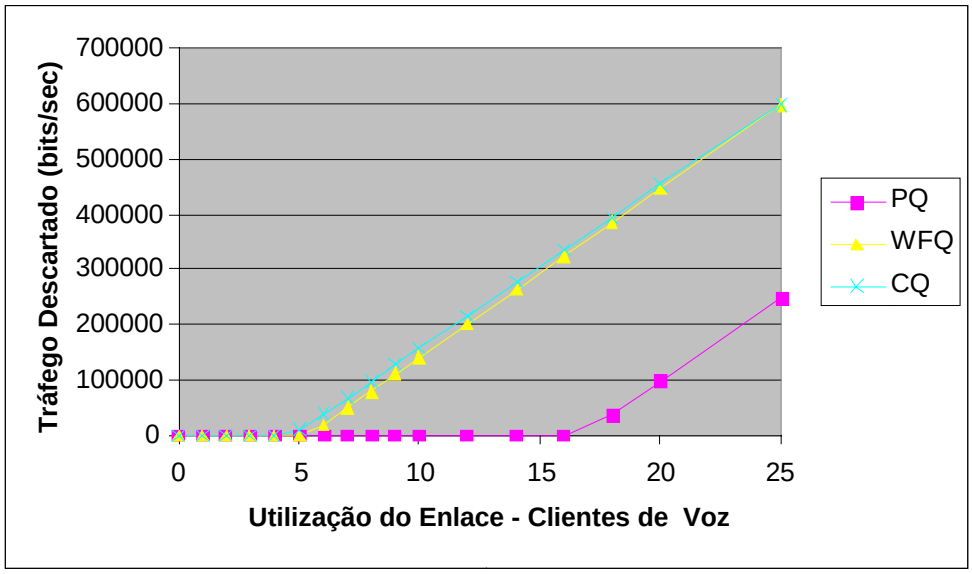


Figura 5.21 – Simulação com variação dos clientes de voz mantendo em 200 os clientes HTTP

Nesta simulação, onde foi variado o número de clientes de voz, fica evidenciada a eficiência da utilização do mecanismo PQ em uma situação onde o tráfego de voz não é totalmente conhecido, e a banda necessária a ser reservada seria imprecisa. Sua utilização em conjunto com o mecanismo WFQ, conforme já mencionado, pode fornecer flexibilidade e eficiência nas configurações de tráfegos onde se deseja priorizar voz.

O comportamento do atraso para o mecanismo FIFO observado na Figura 5.20, onde o atraso diminui com o aumento do número de usuários de voz, conforme já explicado anteriormente (página 85 e Figura 5.5), se deve ao fato de quanto maior for a quantidade de usuários de voz, maior será o número de pacotes de voz povoando a fila, diminuindo seu tamanho em *bytes* e conseqüentemente diminuindo o atraso.

5.2.2. Simulação 2

Esta simulação tem por objetivo analisar o comportamento do mecanismo WFQ, tal como sua utilização com o WRED e o ECN. A Figura 5.22 apresenta o ambiente da simulação 2.

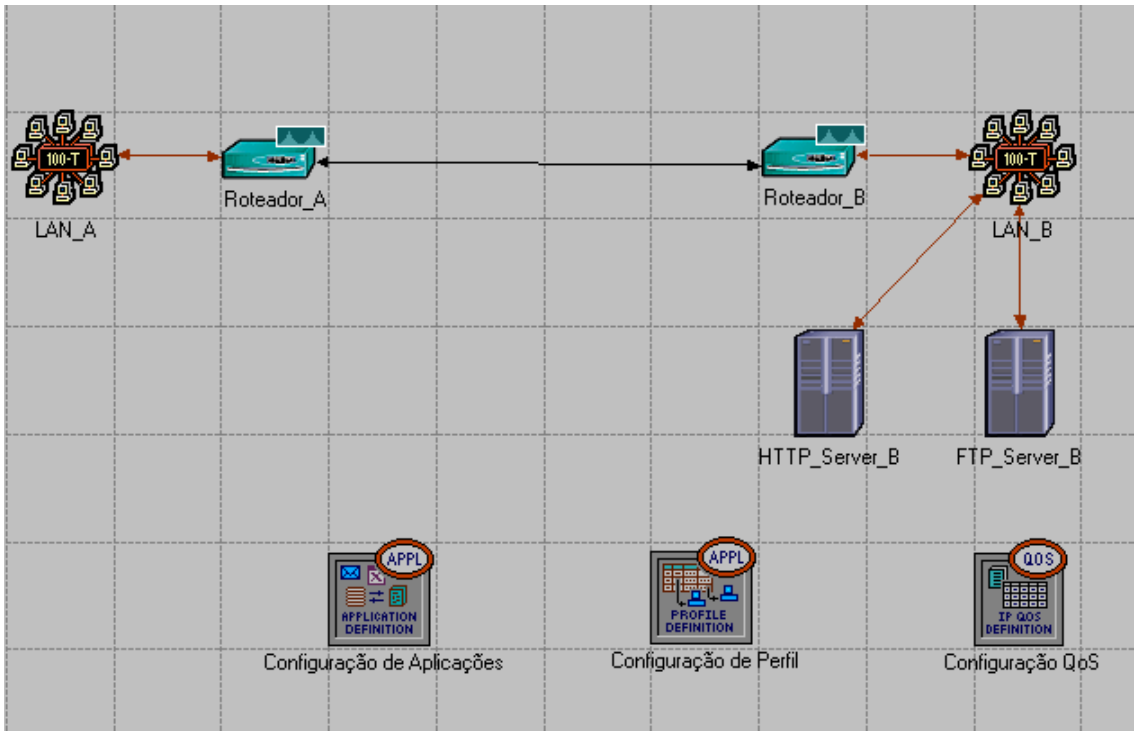


Figura 5.22 – Ambiente da Simulação 2

Nesta simulação, as interfaces dos roteadores A e B, interligados por um enlace PPP de 512 Kbps, foram configuradas com os mecanismos de priorização WFQ com *tail-drop*. Depois foram configuradas com WFQ e o mecanismo de controle de congestionamento WRED e posteriormente com a opção ECN habilitada. O mecanismo WFQ foi configurado com peso 20 e TOS 6 para o tráfego de voz, peso 40 e TOS 3 para o tráfego HTTP, e peso 20 e TOS 2 para o tráfego FTP.

Para as simulações realizadas foram configurados de forma crescente (1, 3, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 60, 70, 80, 90, 100, 150, 200 e 300) os clientes HTTP e FTP na LAN_A, os quais se conectam respectivamente aos servidores Servidor_HTTP_B e Servidor_FTP_B localizado na LAN_B. Não foram configurados clientes de voz, uma vez que na simulação 1 foi demonstrada a capacidade do mecanismo WFQ na proteção deste tráfego. Além disso, a utilização do WRED e ECN só se aplica para tráfegos baseados no protocolo TCP, como o HTTP e o FTP.

Para cada fila do WFQ configurou-se o tamanho máximo de fila (*maximum queue size*) para 100 pacotes, limitado a um total (*buffer capacity*) de 200 pacotes. Desta forma, enquanto este valor (200 pacotes) não é alcançado, uma fila pode armazenar pacotes, independente do tamanho associado à sua fila (100 pacotes). Quando o total de 200 pacotes é atingido, a interface está em um estado de congestionamento total e conseqüentemente as filas individuais passam a ser consideradas. Este comportamento pode ser observado na Figura 5.23, que representa o resultado da simulação (WFQ e *tail-drop*) quanto à utilização do buffer de saída do Roteador_B para ocupação do enlace entre os roteadores A e B. Verifica-se a utilização acima de 100 pacotes do *buffer* pelo tráfego FTP, enquanto o tráfego HTTP não o utiliza. Quando os 2 tráfegos estão utilizando toda a extensão de suas respectivas filas, o mecanismo atribui 100 pacotes do buffer total a cada tráfego, conforme configurado.

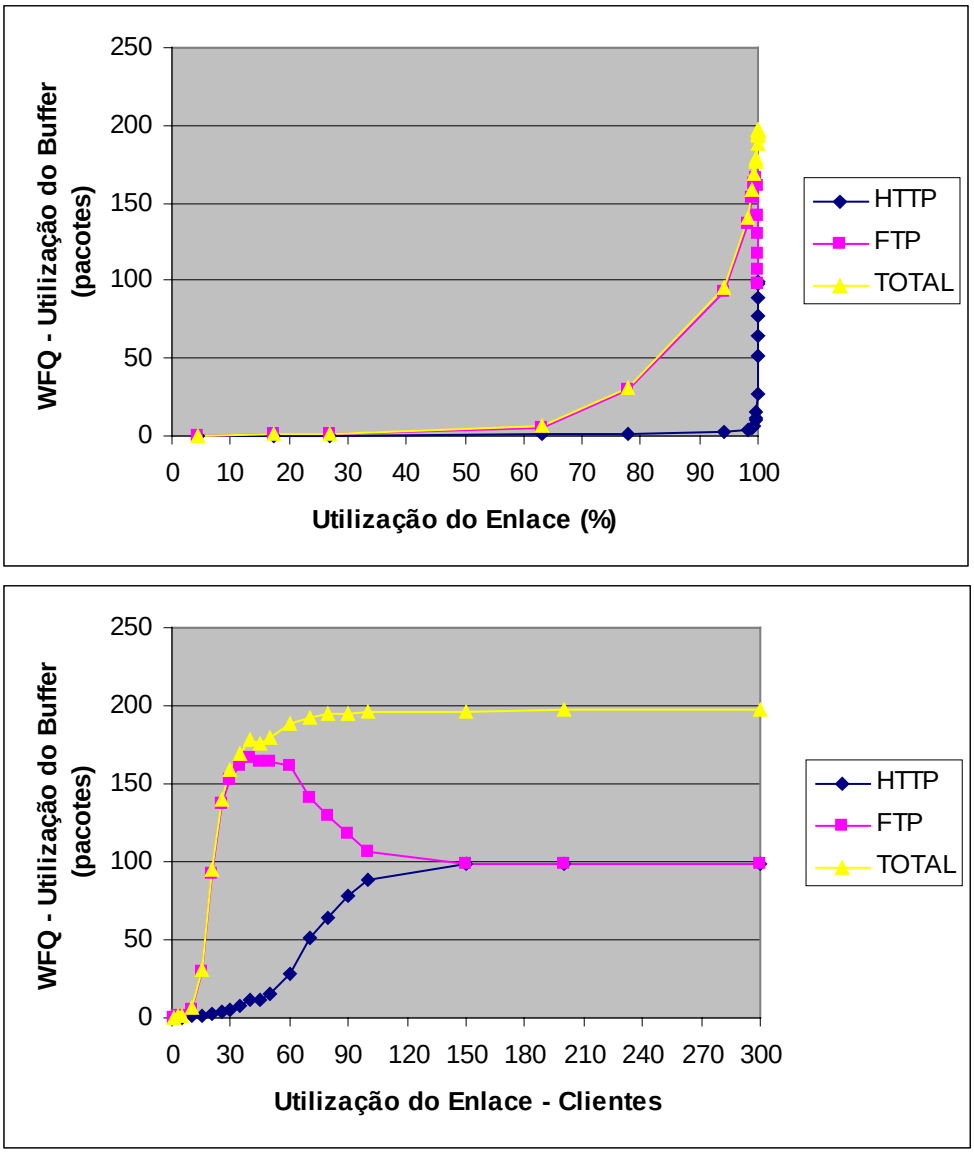


Figura 5.23 – WFQ – Utilização do *Buffer*

A Figura 5.24 apresenta o tráfego recebido (*traffic received*) da LAN_B pelo Roteador_B, a Figura 5.25 apresenta o tráfego descartado no Roteador_B e a Figura 5.26 apresenta o tráfego enviado pelo Roteador_B ao Roteador_A através do enlace de 512 Kbps que os interliga.

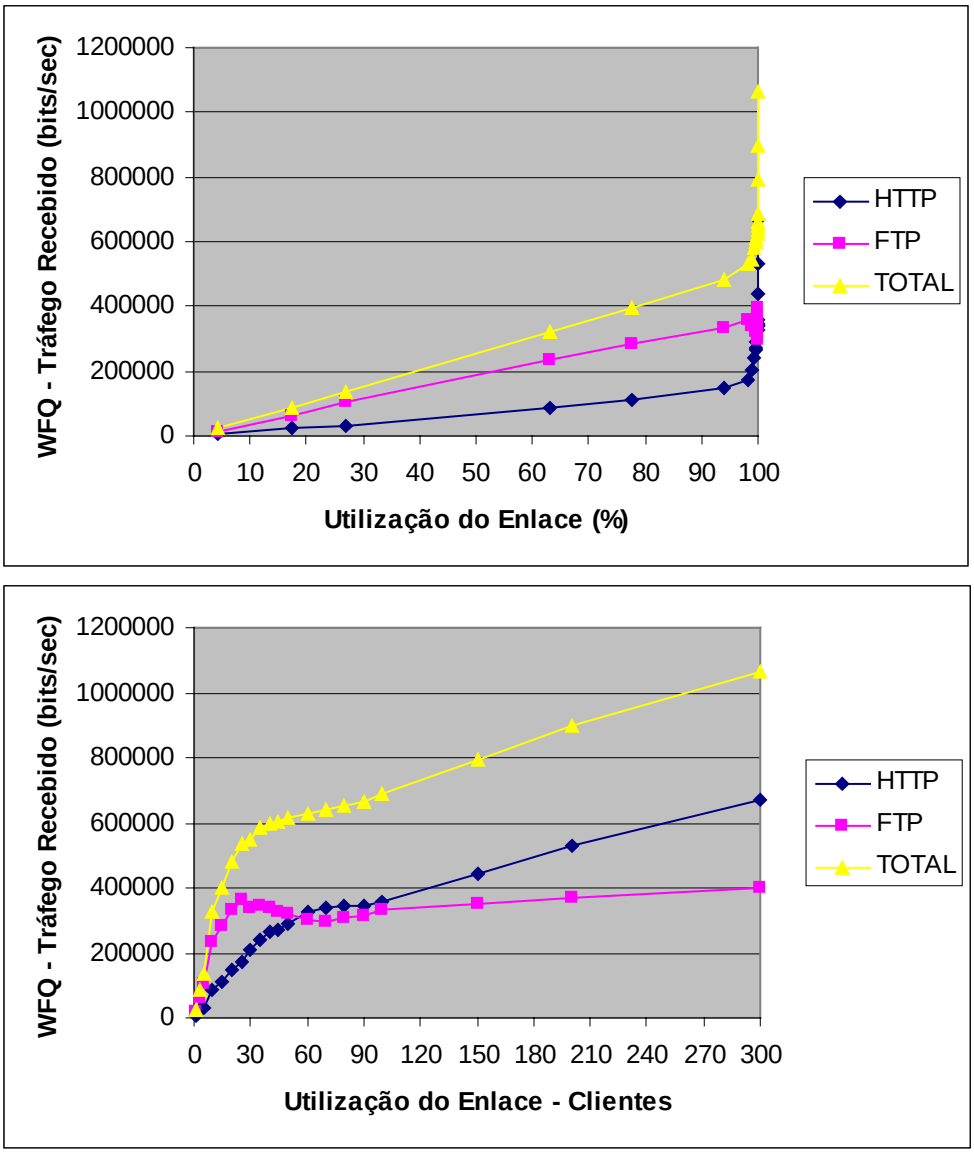


Figura 5.24 – WFQ – Tráfego Recebido

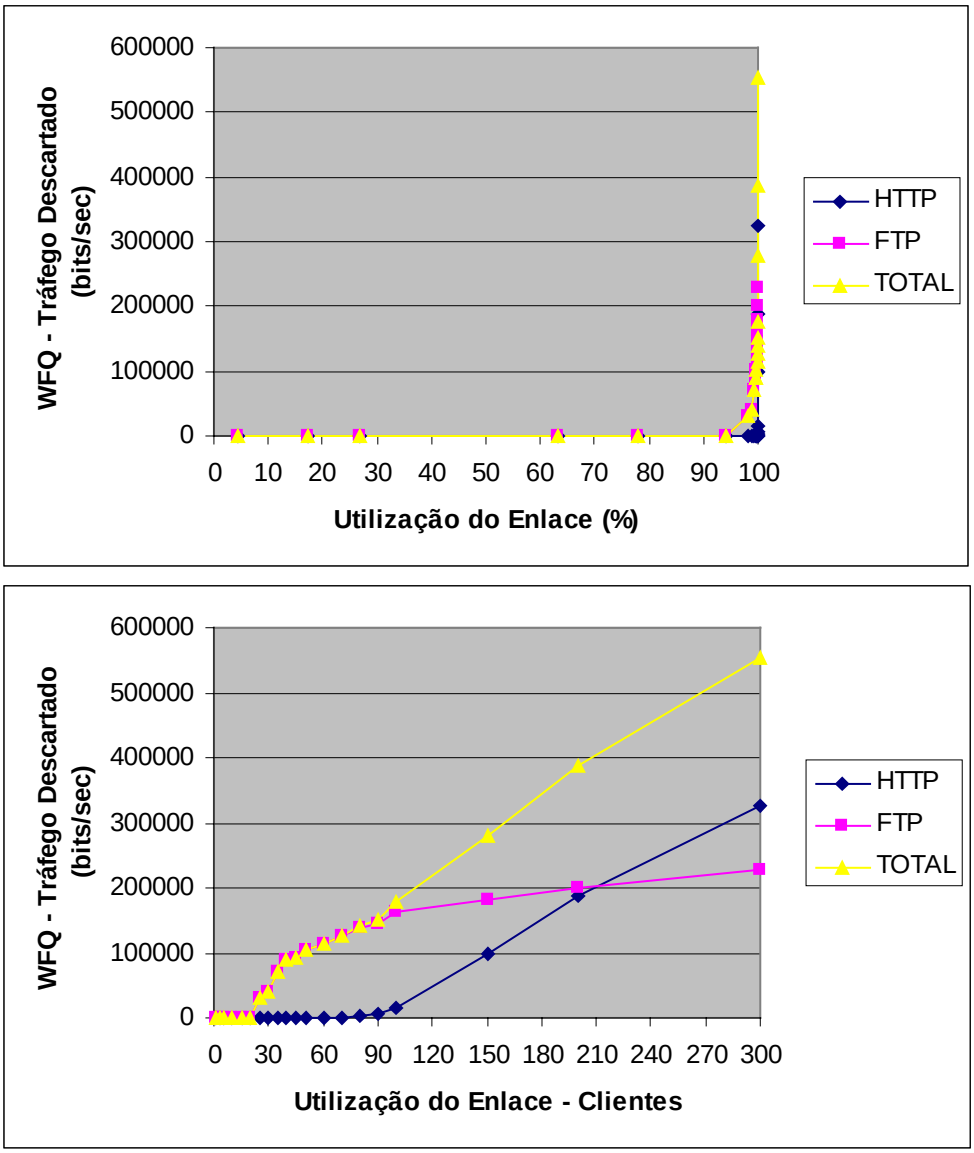


Figura 5.25 – WFQ – Tráfego Descartado

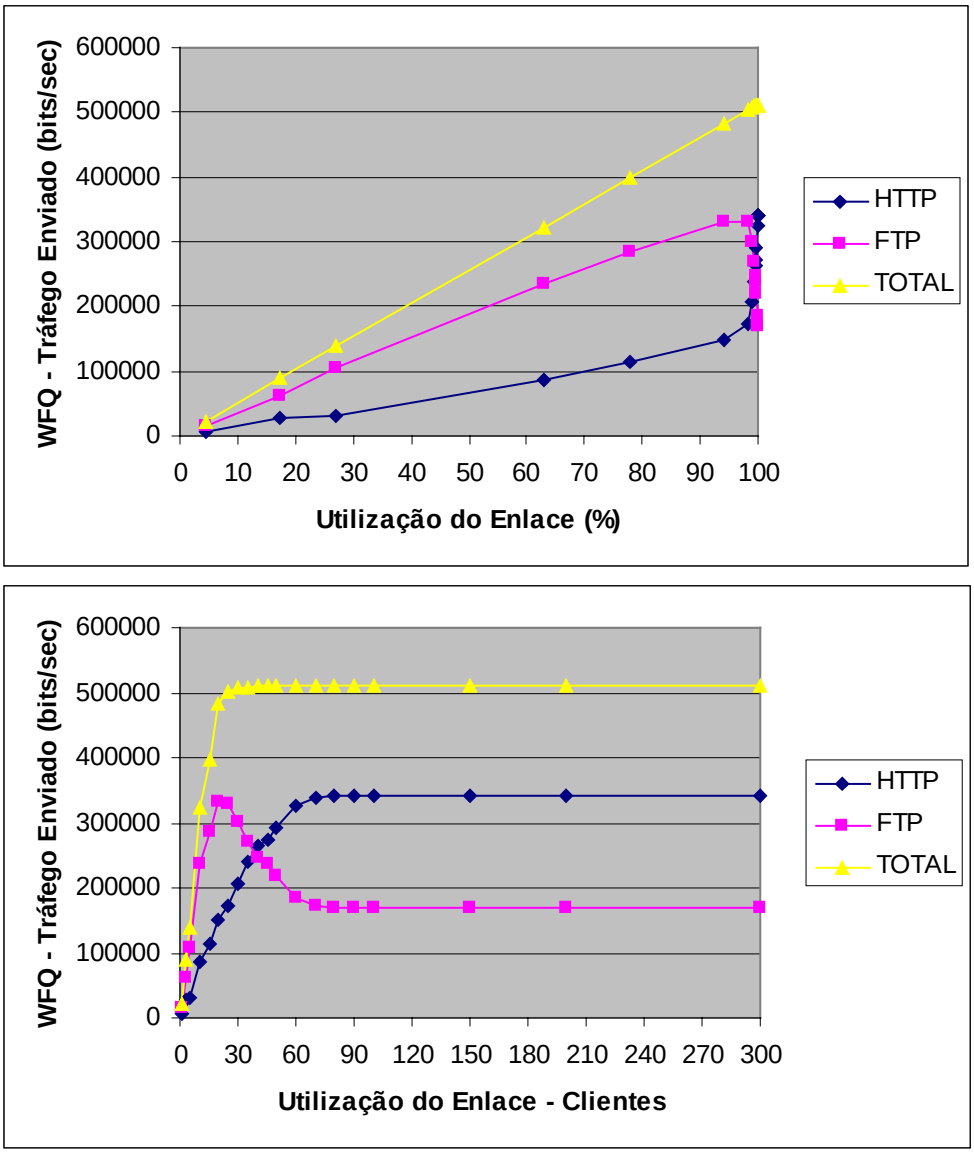


Figura 5.26 – WFQ Tráfego Enviado (tráfego enviado pelo Roteador_B ao Roteador_A através do enlace de 512 Kbps)

Na Figura 5.26 fica evidenciada a justiça na distribuição da capacidade do enlace de 512 kbps, pois uma vez que o peso atribuído ao tráfego HTTP foi igual a 40 e ao tráfego FTP igual a 20, e sendo no mecanismo WFQ o restante da banda compartilhado pelos fluxos na proporção dos seus pesos, temos que no estado de congestionamento, o tráfego HTTP ocupa 2/3 do enlace (341,3 Kbps) e o tráfego FTP ocupa 1/3 do enlace (170,7 Kbps), conforme esperado. Nota-se também que enquanto o tráfego HTTP não

utiliza sua banda reservada, esta é aproveitada pelo tráfego FTP, demonstrando a característica conservativa do mecanismo na utilização dos recursos.

Para este mesmo ambiente, foi realizada nova simulação, habilitando o mecanismo de controle de congestionamento WRED.

O artigo [65], referenciado no Capítulo 5 quanto à escolha dos valores para th_{min} e th_{max} no mecanismo WRED, conclui o seguinte: um baixo atraso para uma probabilidade específica de perda pode ser obtida através da utilização de uma alta probabilidade máxima de descarte, um baixo valor para o limiar th_{min} e uma diferença pequena entre os limiares. Utilizando estas premissas, escolheu-se a seguinte configuração do WRED para a fila Q3 (*Queue* 3 e TOS 3 atribuídos ao tráfego HTTP), buscando melhoria do atraso para o tráfego HTTP:

	Q3 (HTTP)
Probabilidade máxima de descarte (*1)	2
limiar th_{min} (minimum threshold)	50
limiar th_{max} (maximum threshold)	75

(*1) – um em cada 2 pacotes em Q3 será descartado quando o tamanho médio da fila (*avg*) atingir o limiar máximo.

As Figuras 5.27, 5.28, 5.29, 5.30 e 5.31 mostram a melhoria no atraso de enfileiramento (*queueing delay*), associado à diminuição da utilização do buffer (*buffer usage*) causado pela antecipação de descarte de pacotes implementado pelo mecanismo WRED. Pode-se observar que, com a utilização do WRED, houve diminuição no atraso do enfileiramento, com manutenção da quantidade de tráfego enviado.

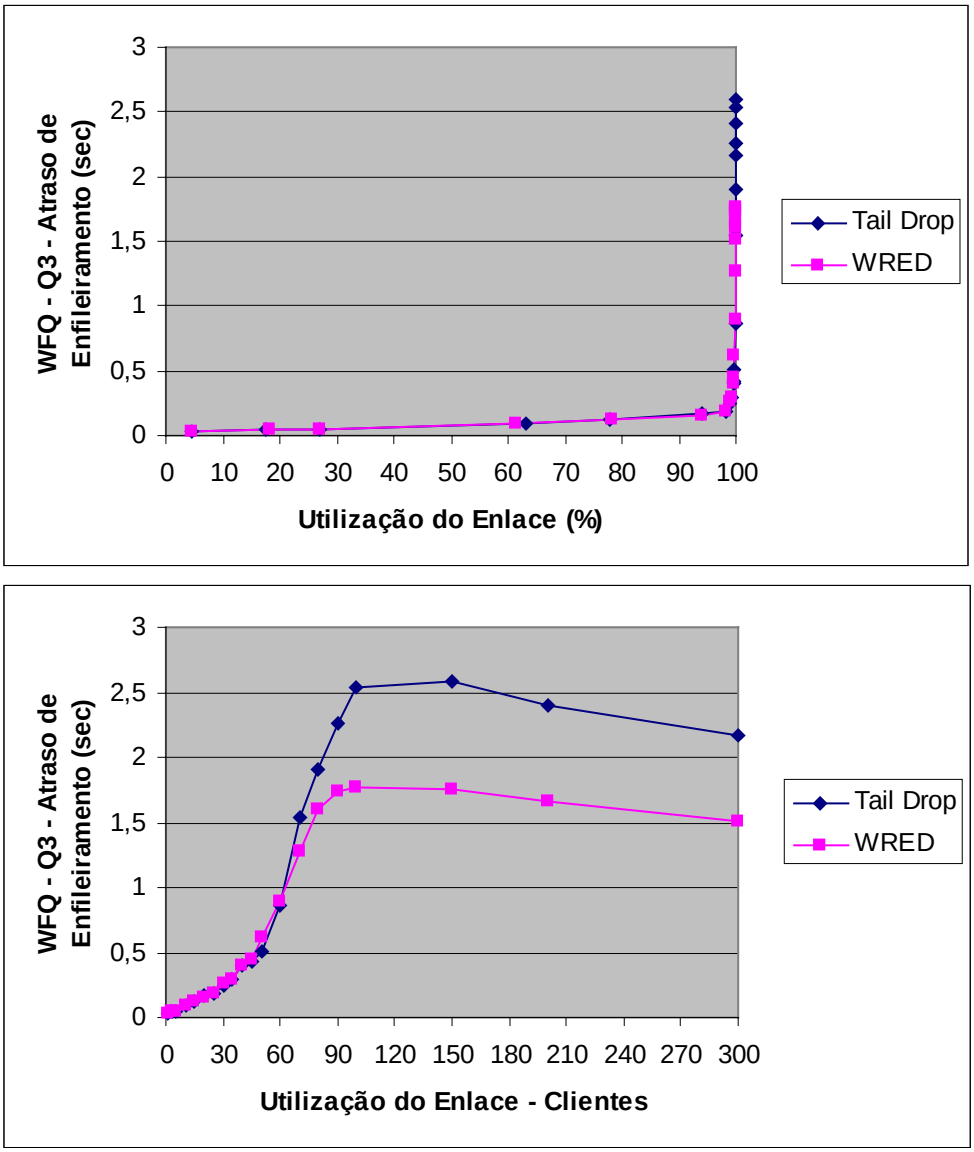


Figura 5.27 – WFQ – Atraso de Enfileiramento (Q3)

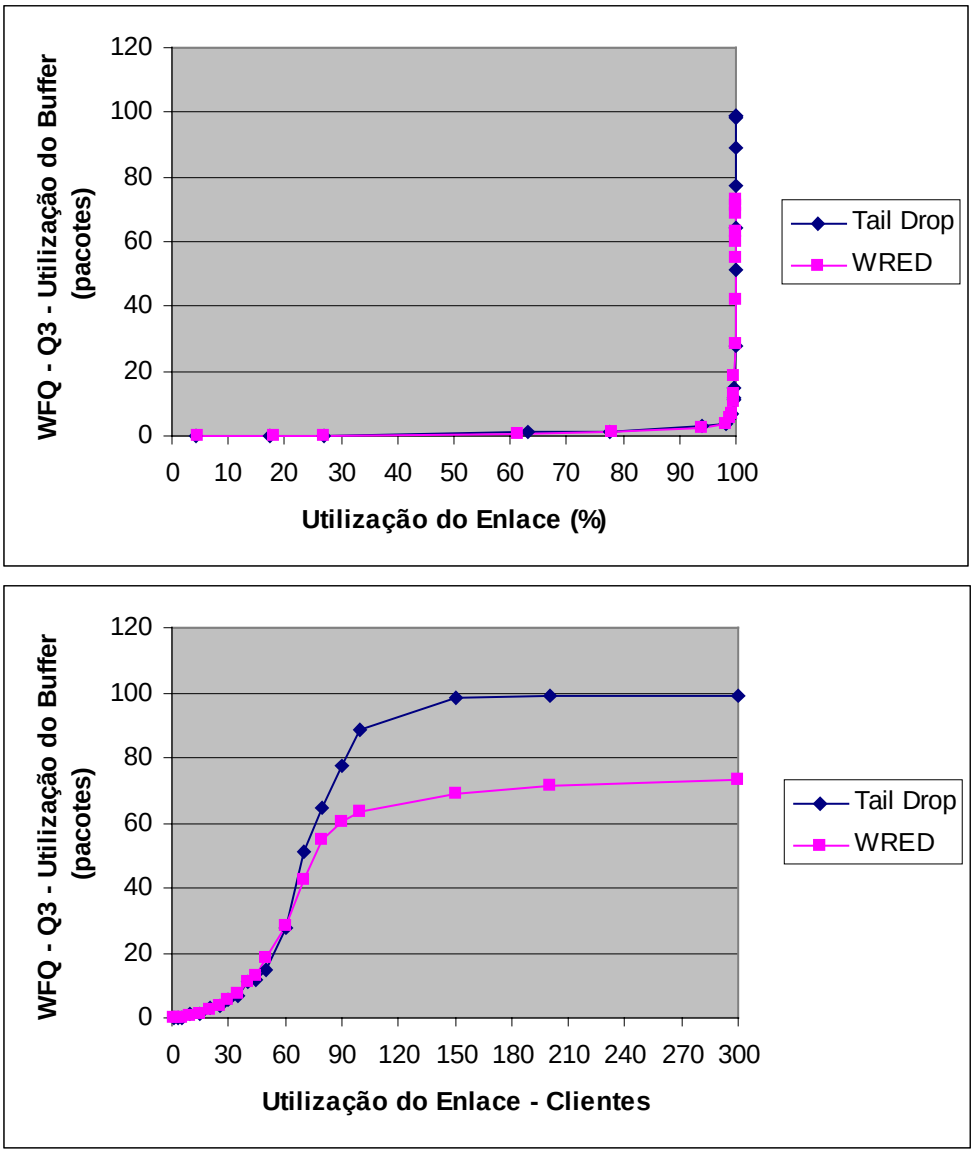


Figura 5.28 – WFQ – Utilização do *Buffer* (Q3)

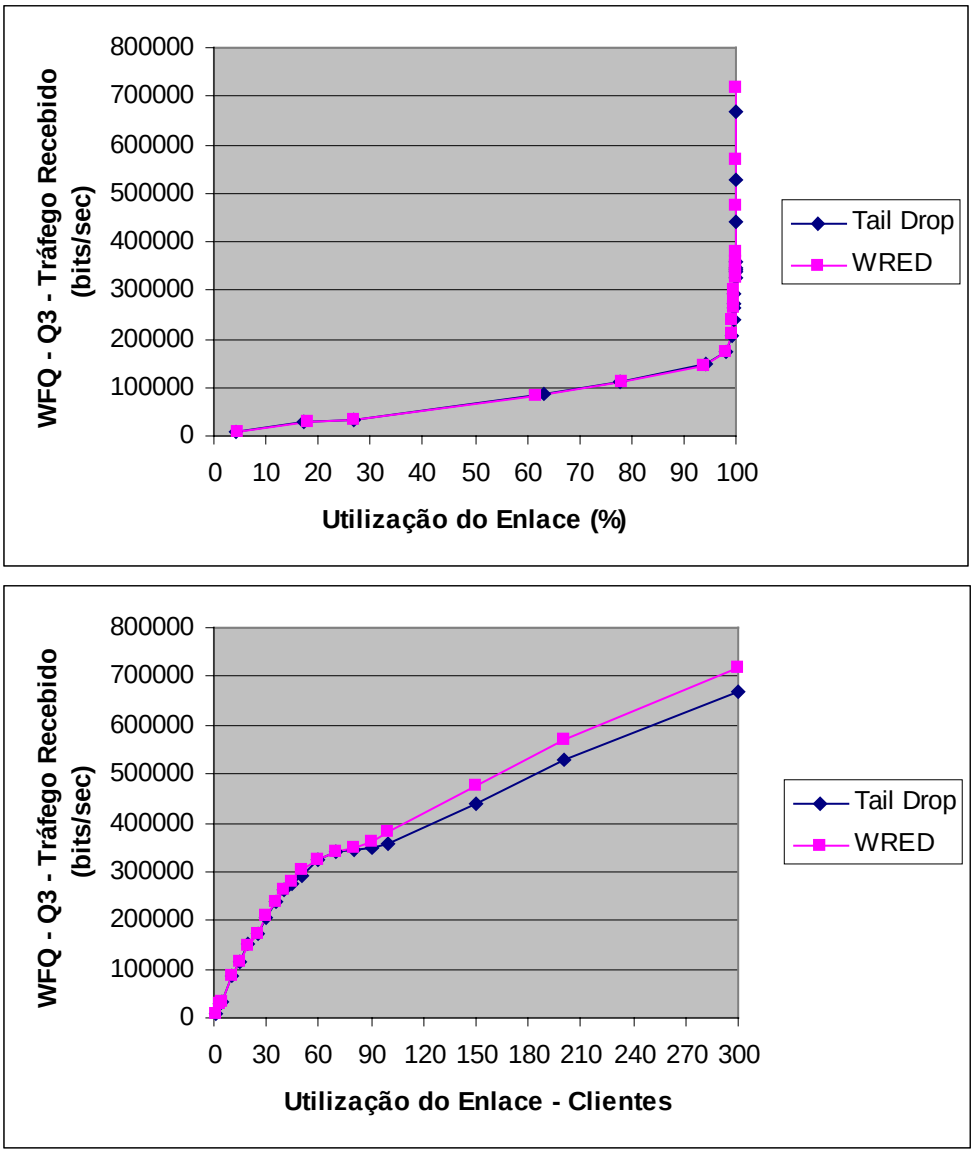


Figura 5.29 – WFQ – Tráfego Recebido (Q3)

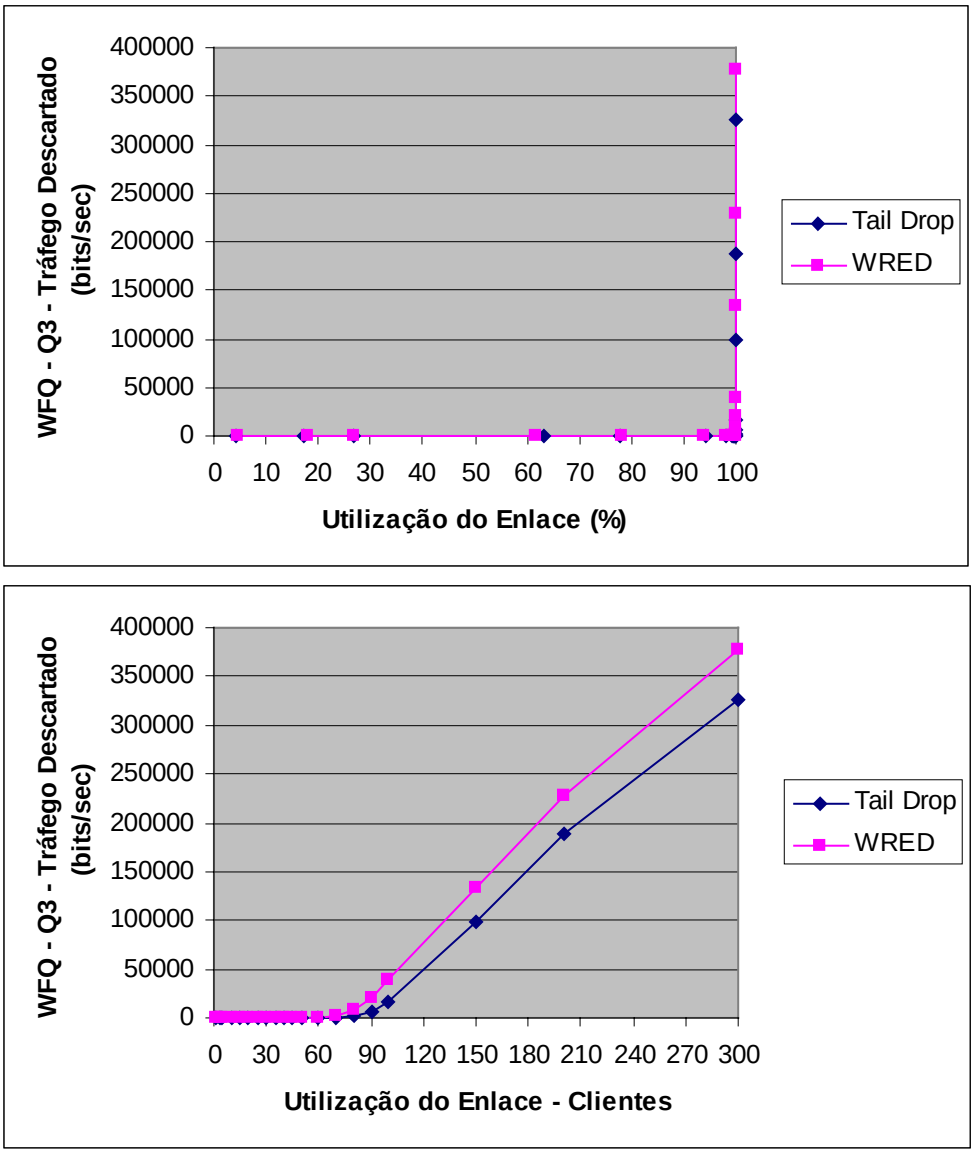


Figura 5.30 – WFQ – Tráfego Descartado (Q3)

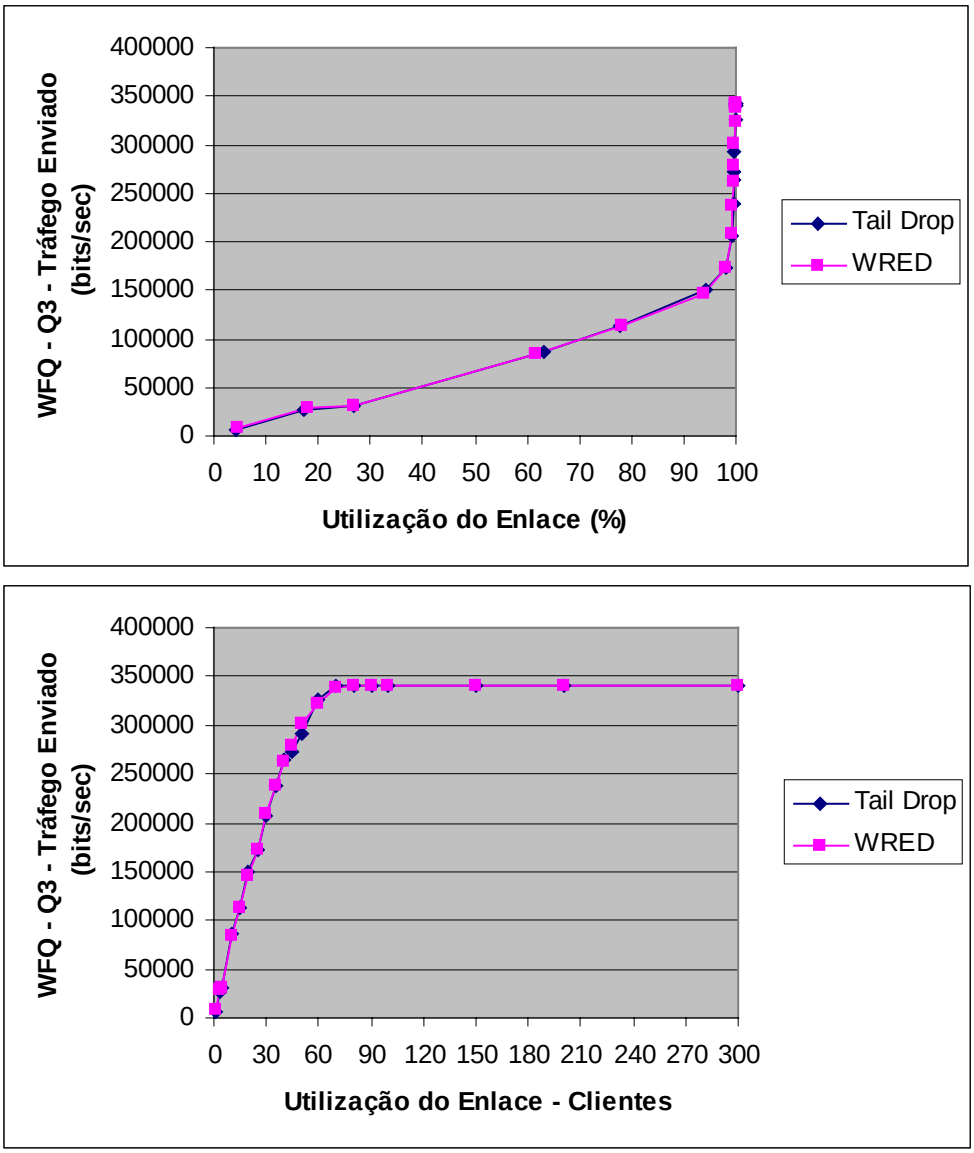


Figura 5.31 – WFQ – Tráfego Enviado (Q3)

Na Figura 5.27 observa-se um decréscimo no atraso de enfileiramento após a ocupação total do *buffer*, tanto com a utilização do *tail-drop* quanto do WRED, em aproximadamente 150 clientes. Este comportamento se explica pela diminuição contínua do tamanho médio dos pacotes à medida que o congestionamento cresce. Esta diminuição dos pacotes em *bytes* após a capacidade total do *buffer* ser atingida (dimensionado em quantidade de pacotes) gera uma diminuição da capacidade total do buffer em bytes conseqüentemente diminuindo o atraso. A Figura 5.32 ilustra a variação

do tamanho médio dos pacotes em relação ao número de clientes tanto para a utilização do *tail-drop* e WRED quanto do WRED c/ ECN que será abordado mais à frente.

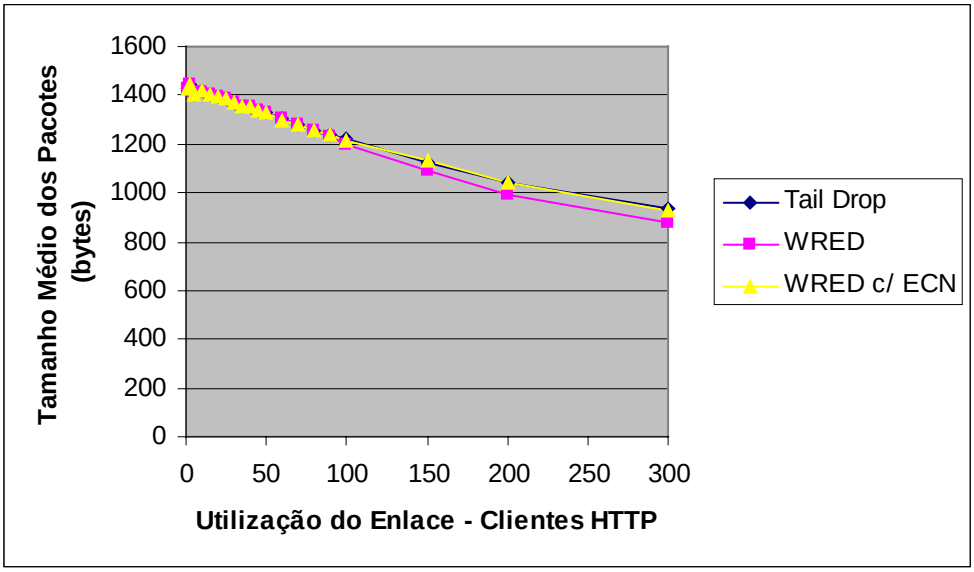


Figura 5.32 – Tamanho médio dos pacotes

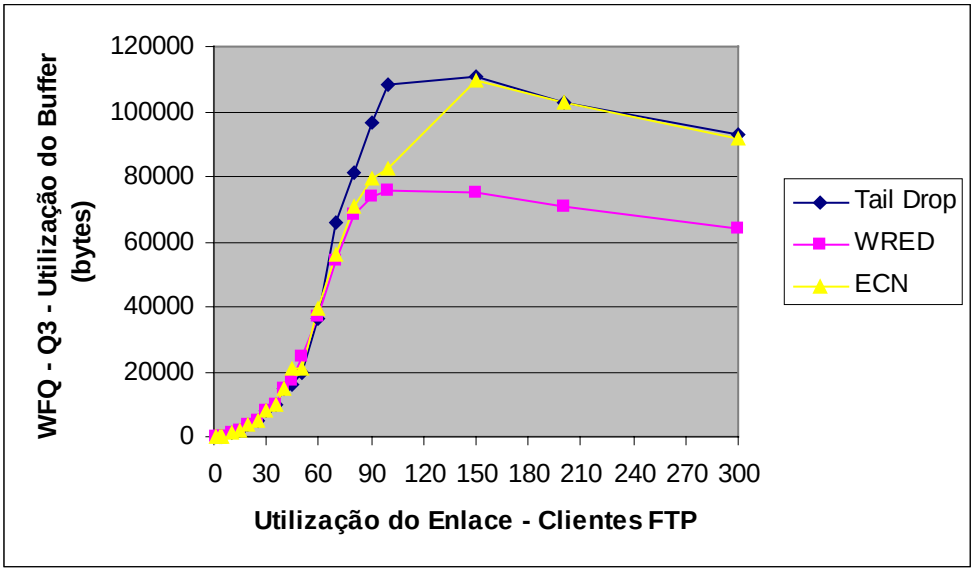


Figura 5.33 – WFQ – Utilização do *Buffer* (Q3) em *bytes*

Este comportamento para o tamanho médio dos pacotes é refletido na Figura 5.33 (WFQ *Buffer Usage (Q3)* em *bytes*), o qual apresenta comportamento compatível com o

atraso de enfileiramento apresentado na Figura 5.27 e na Figura 5.38 que será mostrada à frente.

Na Figura 5.34 pode-se observar que houve melhoria no tempo de resposta da página sob o ponto de vista do cliente, ou seja, das estações da LAN_A que estão acessando o servidor Server_HTTP_B localizado na LAN_B.

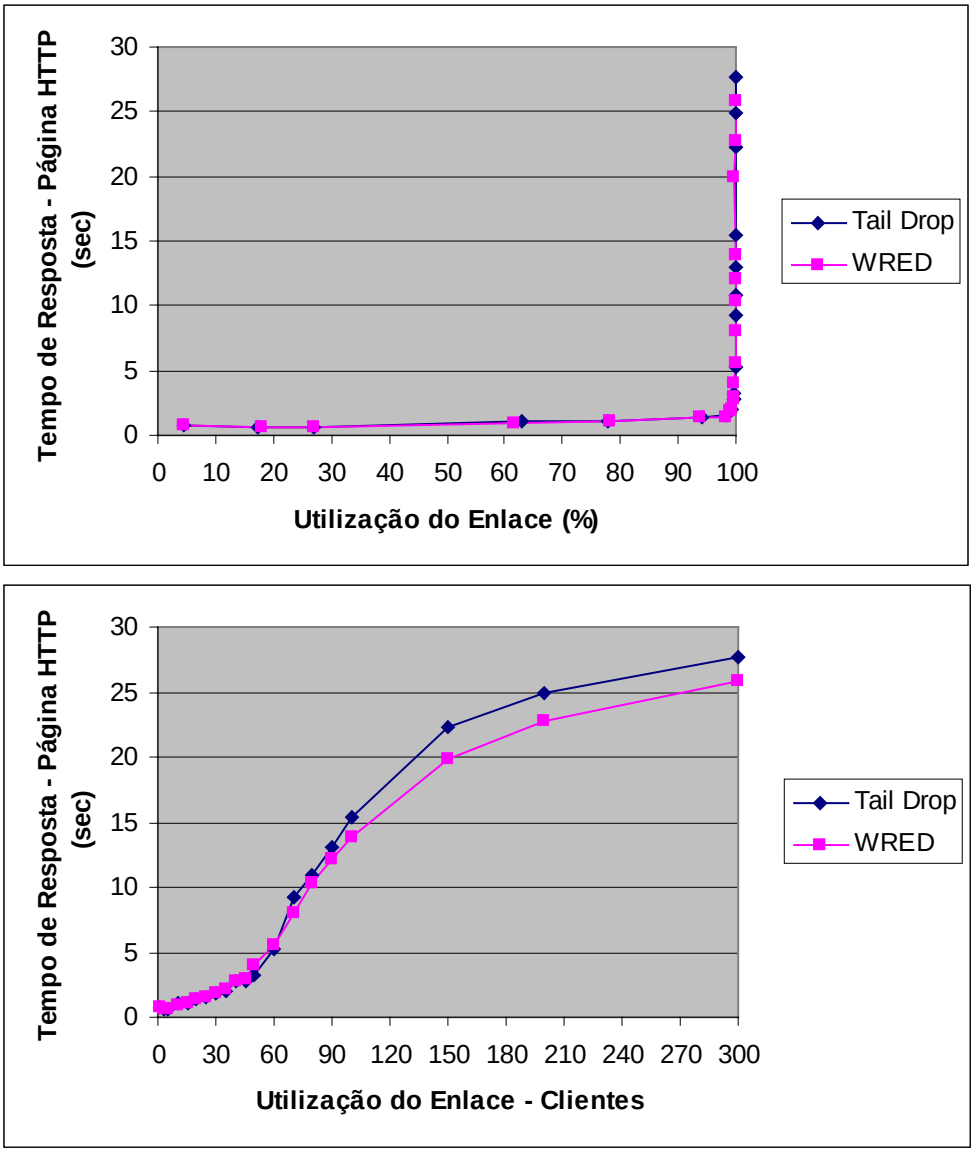


Figura 5.34 – Tempo de Resposta – Página HTTP

Observa-se que esta melhoria (tempo de resposta da página) não representa a mesma proporção alcançada na melhoria do atraso de enfileiramento (*queueing delay*). O tempo de resposta não é dependente somente do atraso de enfileiramento, mas também do nível de descartes imposto pelo algoritmo, o qual implicará na quantidade de retransmissão e conseqüentemente no tempo de resposta como um todo, ou seja, o tempo total de transmissão de todos os segmentos que compõem a página HTTP. Em alguns dos trabalhos pesquisados ([61], [63], [65]), a atuação dos algoritmos RED e WRED sobre o atraso é avaliada de maneira simplista, levando em consideração somente o atraso de enfileiramento.

Neste mesmo ambiente, mantendo a mesma configuração WFQ, habilitou-se a opção ECN para a configuração WRED do tráfego HTTP (fila Q3 – TOS 3).

As Figuras 5.35, 5.36 e 5.37 apresentam o comportamento do tráfego recebido, descartado e enviado para o tráfego HTTP na fila Q3 (TOS 3) do mecanismo WFQ, configurado com *tail-drop*, WRED e WRED com ECN.

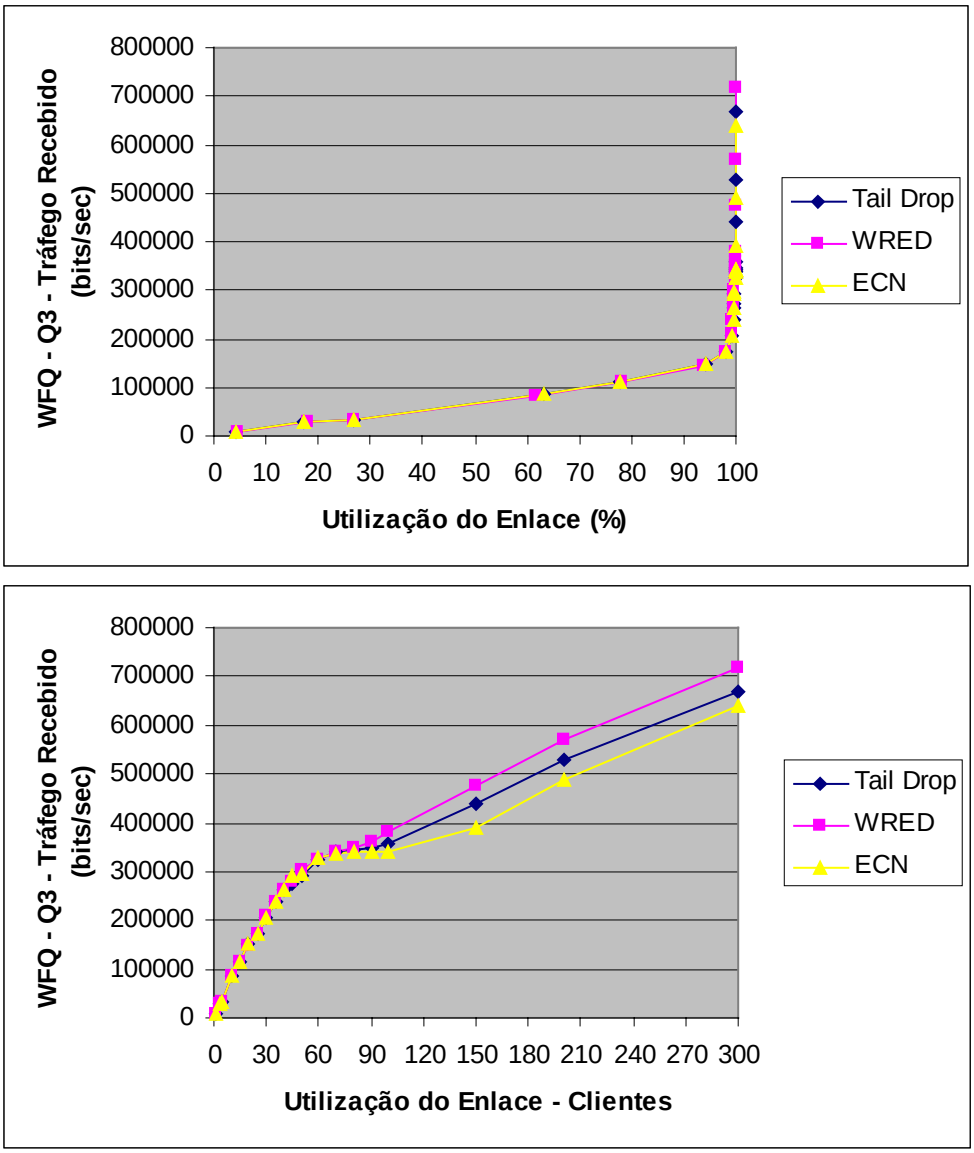


Figura 5.35 – WFQ – Tráfego Recebido (Q3)

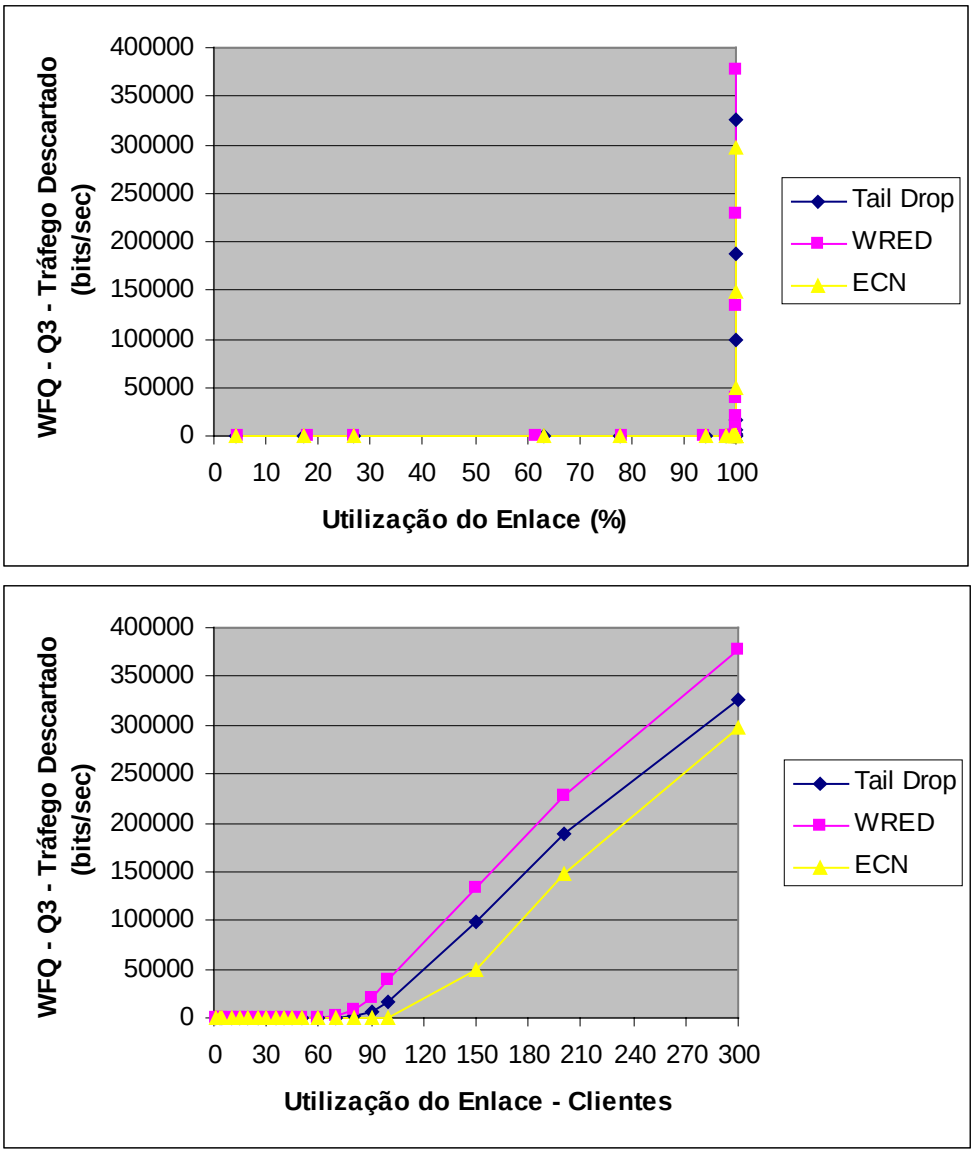


Figura 5.36 – WFQ – Tráfego Descartado (Q3)

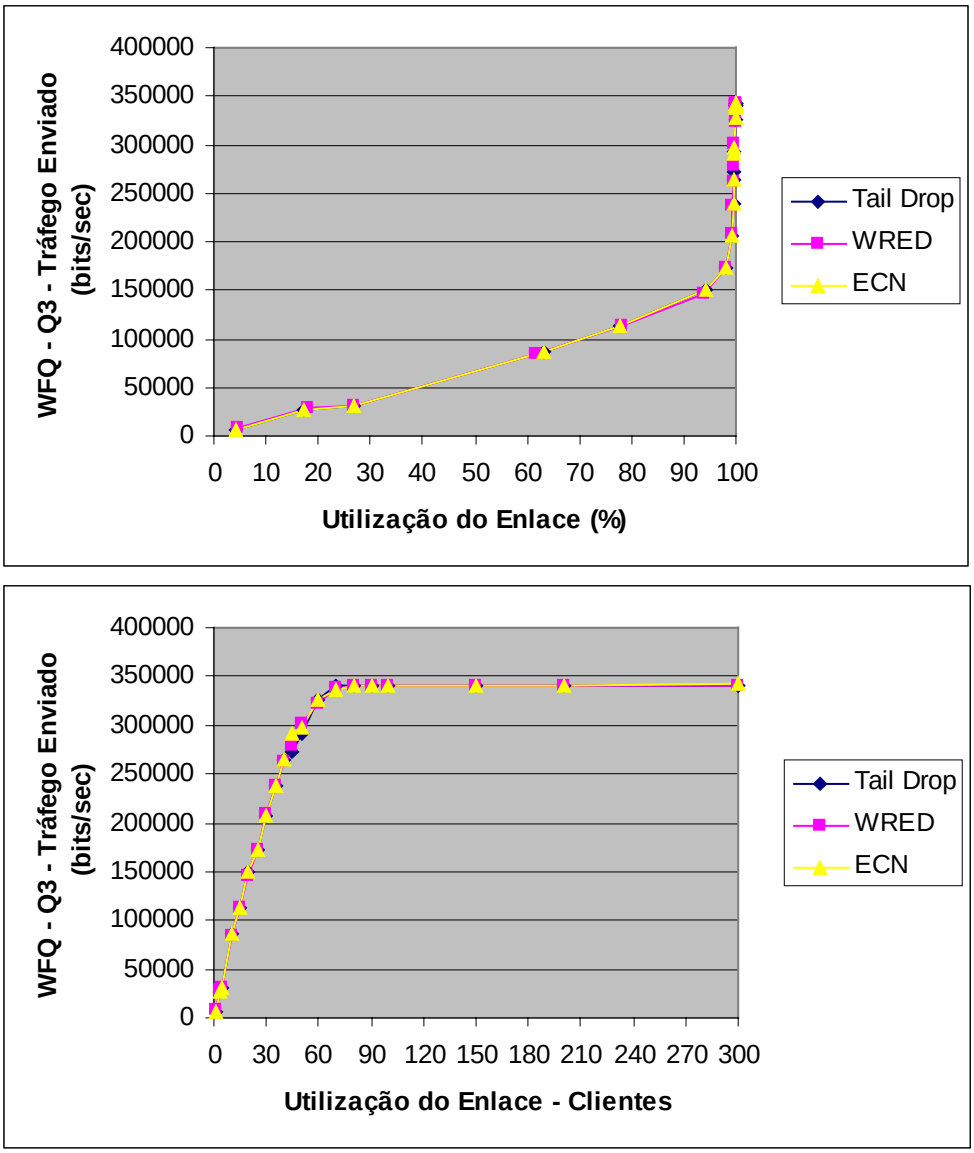


Figura 5.37 – WFQ – Tráfego Enviado (Q3)

Observa-se que a habilitação do ECN resultou em significativa redução de descarte do tráfego HTTP, sem nenhuma alteração do tráfego enviado. Por outro lado, ao se observar as Figuras 5.38, 5.39 e 5.40, as quais representam o tempo de atraso de enfileiramento, a utilização do *buffer* e o tempo de resposta da página, respectivamente, verifica-se que o atraso de enfileiramento migra de comportamento entre o WRED e o *tail-drop*, uma vez que a identificação de congestionamento pelo emissor através do ECN somente é realizada após um ciclo de ida e volta (RTT), e nesta simulação existe

congestionamento nos dois sentidos do enlace, em função da configuração do tráfego FTP, conforme Anexo 1. Sendo assim, com o aumento do número de usuários, o nível de congestionamento aumenta e ocorre estouro do *buffer* em aproximadamente 150 usuários (Figura 5.39), quando o atraso se iguala ao apresentado pelo mecanismo FIFO. A partir deste número de usuários, o *buffer* se mantém constantemente no limite do estado de congestionamento total. Neste estado, o mecanismo ora descarta, ora marca o pacote. A marcação de um pacote enviado, porém de uma conexão já perdida, causa desperdício do enlace e conseqüentemente aumento de retransmissões e maior atraso fim-a-fim.

Aponta-se em [23] e [26] a utilização do ECN como importante para a diminuição do descarte de pacotes e principalmente para um controle mais efetivo do tráfego quanto aos parâmetros de atraso e perda. Para uma verificação mais pontual, mostra-se nas Figuras 5.41 e 5.42 o comportamento do tráfego (atraso de enfileiramento e tráfego descartado) para a quantidade de 100 clientes (anterior à situação de congestionamento total – ver Figura 5.39) quanto ao atraso de enfileiramento e quanto às perdas, com e sem a utilização do ECN.

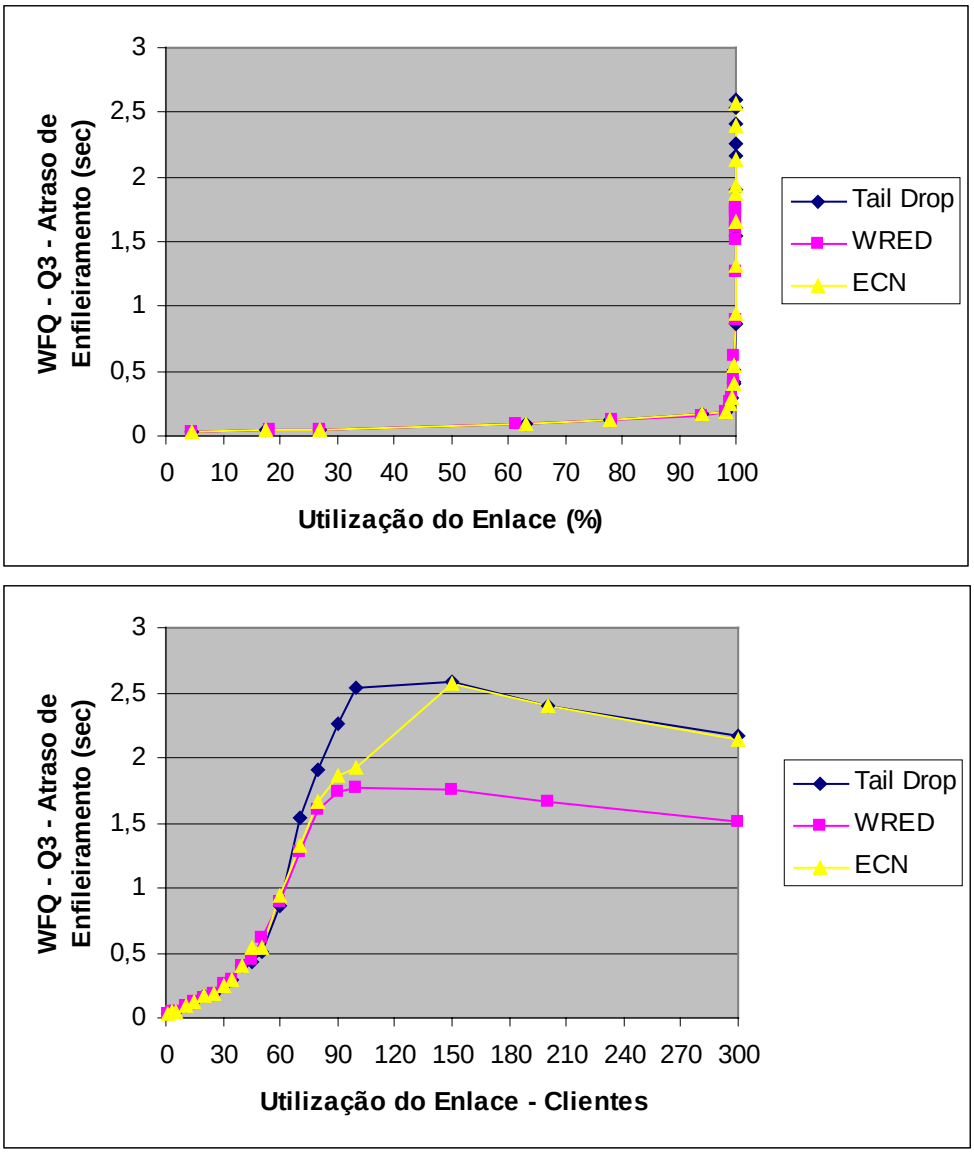


Figura 5.38 – WFQ – Atraso de Enfileiramento (Q3)

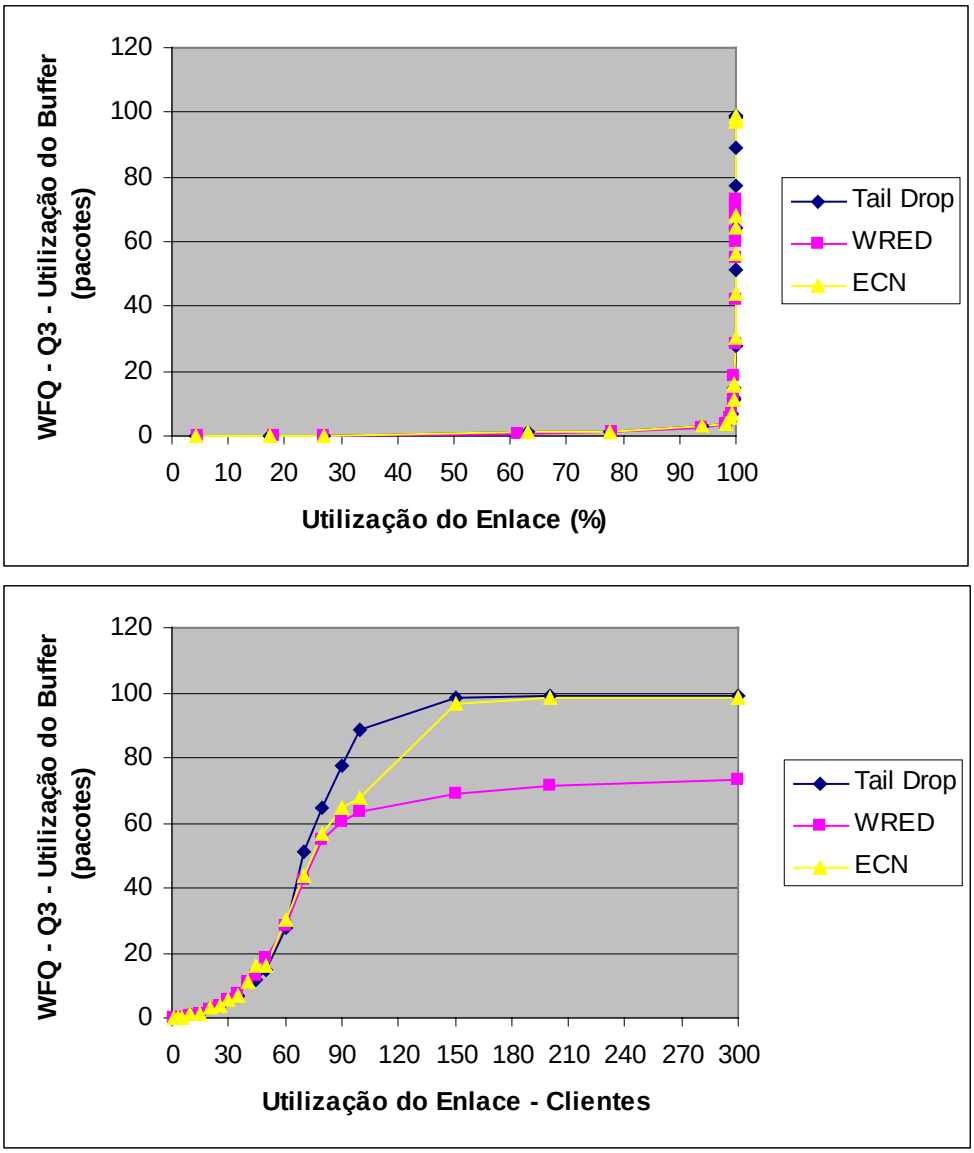


Figura 5.39 – WFQ – Utilização do *Buffer* (Q3)

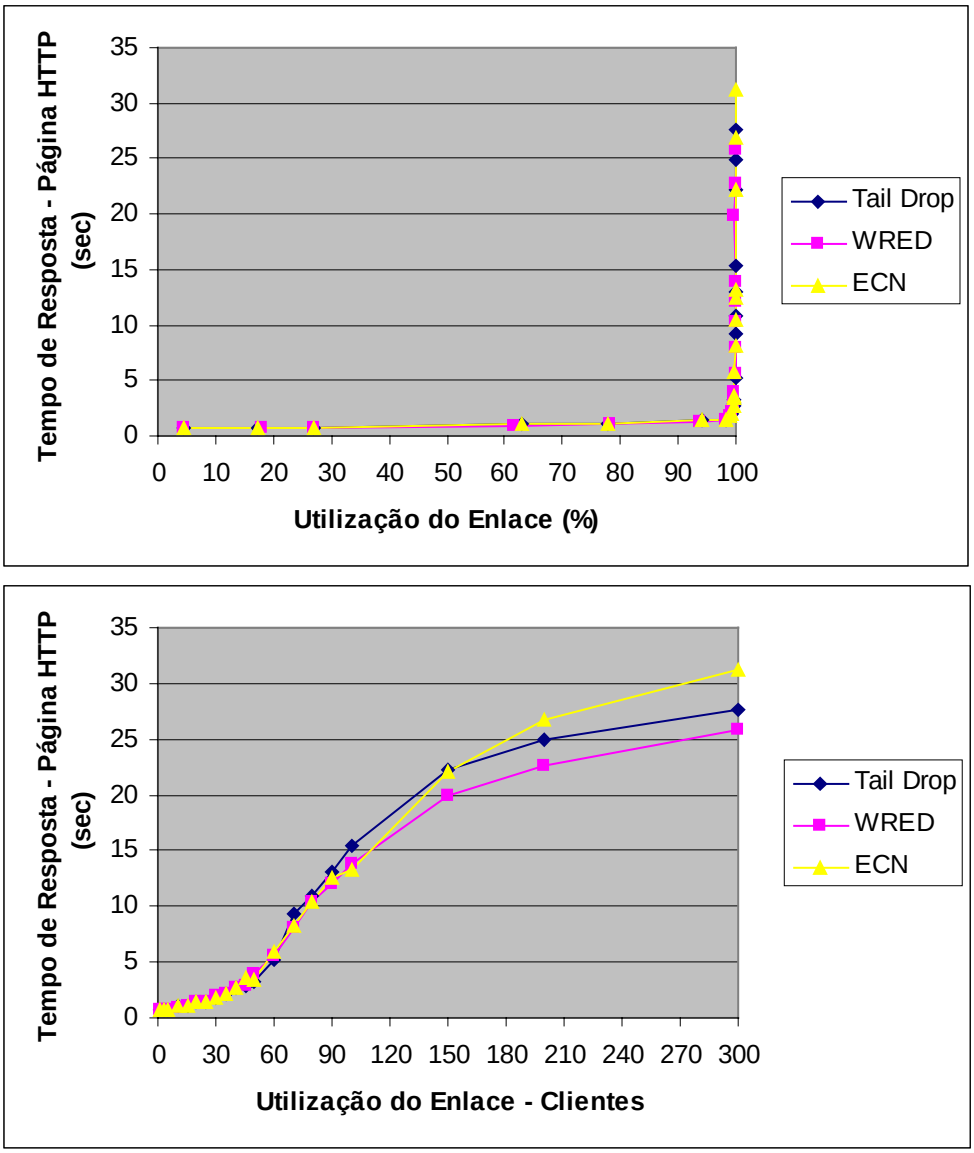


Figura 5.40 – Tempo de Resposta – Página HTTP

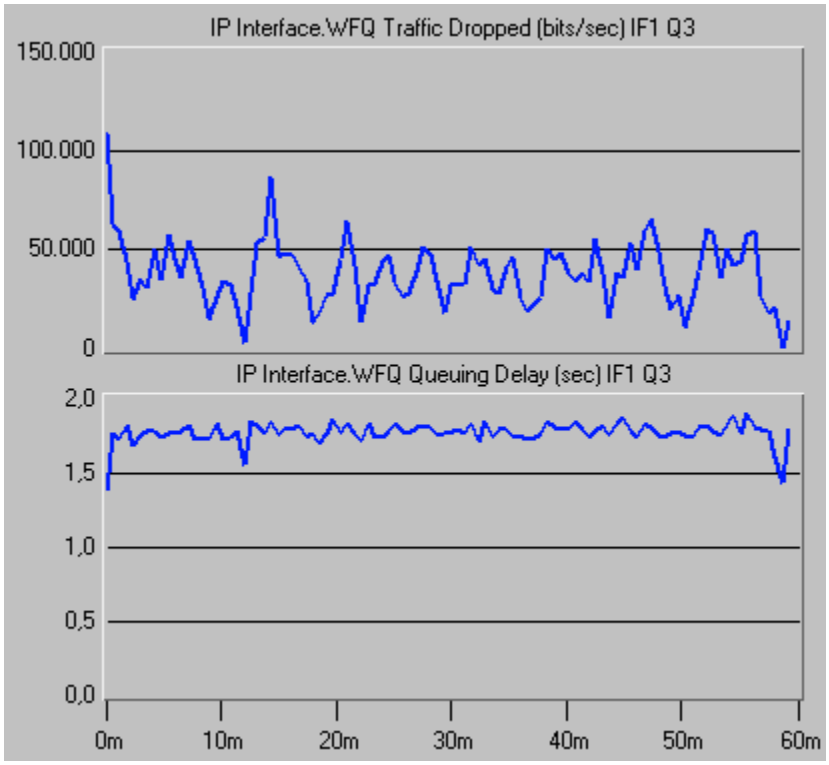


Figura 5.41 – WFQ *Queuing Delay (Q3)* e WFQ *Traffic Dropped (Q3)* – WRED s/ ECN

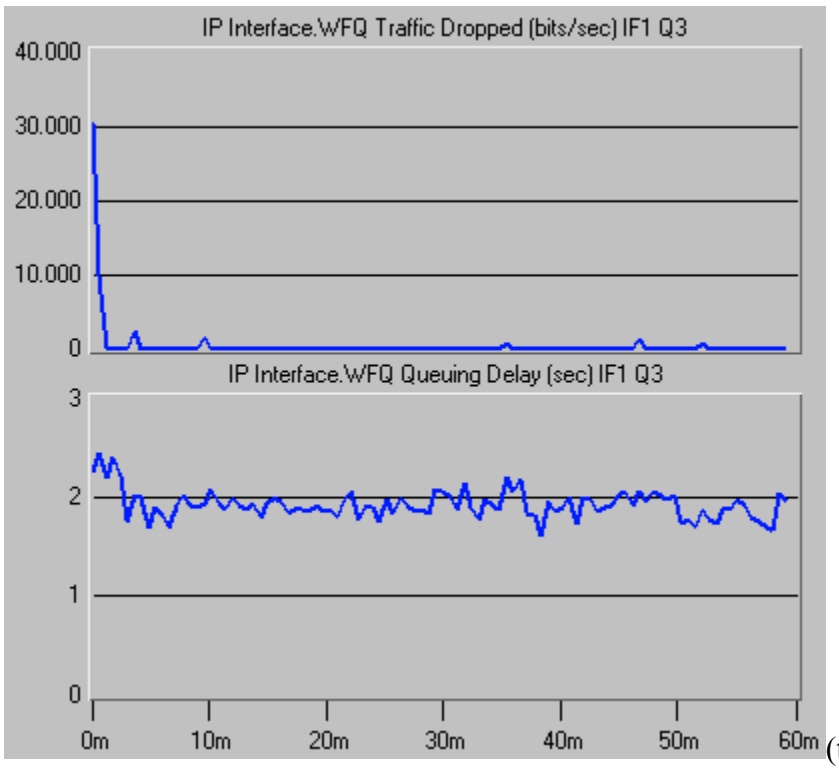


Figura 5.42 – WFQ *Queuing Delay (Q3)* e WFQ *Traffic Dropped (Q3)* – WRED c/ ECN

Com a utilização do ECN, foi possível diminuir a taxa de descarte médio de 38790 bps para 494 bps enquanto o atraso de enfileiramento médio sofreu acréscimo de 1,77 ms para 1,93 ms.

Conforme verificado nestas simulações, fica evidenciado que a utilização adequada do ECN, dentro de certos limites, poderá permitir adequação dos parâmetros de perda e atraso de enfileiramento para atender restrições de um tráfego quanto a estes parâmetros.

5.2.3. Simulação 3

Esta simulação tem como objetivo analisar o comportamento da aplicação HTTP quando se varia o valor de IW (*Initial Window*). Utilizou-se o mesmo ambiente da simulação 2, com a configuração WFQ e *tail-drop*.

Na Figura 5.43 observa-se que ocorreu variação significativa no tempo de resposta da aplicação HTTP quando o valor de IW é configurado para 1, 2, e 4 vezes o valor de MSS (*Maximum Segment Size*) e também configurado com o valor definido na RFC 3390, conforme a Equação 4.1.

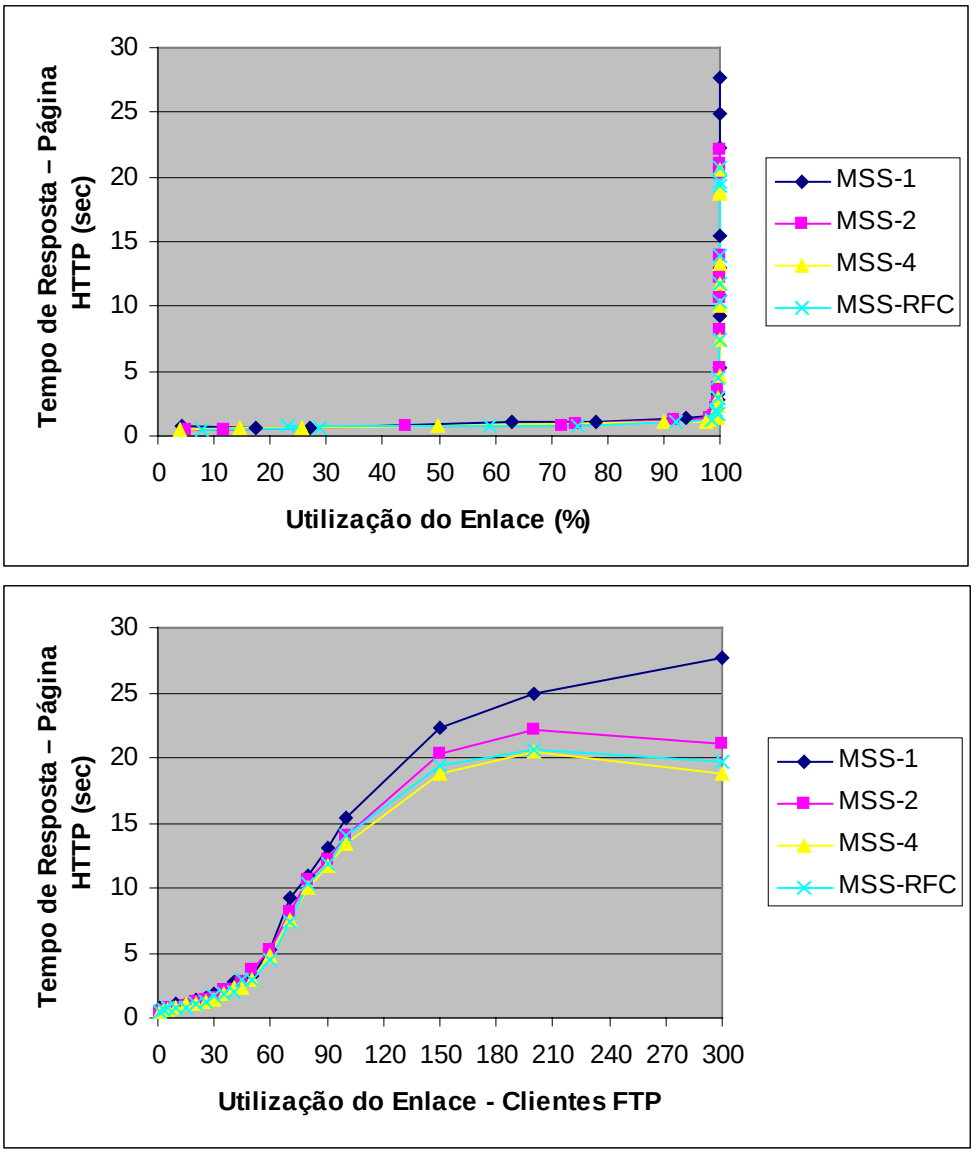


Figura 5.43 – Tempo de Resposta – Página HTTP

6. Conclusão

Os mecanismos aqui estudados e avaliados mostram a sua importância no fornecimento de QoS.

Por meio da Simulação 1, verificou-se que a utilização de um mecanismo eficiente de priorização, tal como o PQ ou o WFQ, viabiliza a transmissão de tráfego de voz, com qualidade de serviço, em uma rede IP, mesmo em situações de carga elevada.

Com a utilização do mecanismo WFQ foi possível reservar com eficiência a banda desejada para o tráfego de voz. Constatou-se também a capacidade do mecanismo em ser justo na distribuição da capacidade do enlace e sua característica conservativa na distribuição da banda reservada e não utilizada por determinado tráfego.

Verificou-se também que através do mecanismo PQ foi possível proteger o tráfego de voz quando se desconhece a banda necessária a ser reservada para este tráfego. Desta forma, sua utilização em conjunto com o mecanismo WFQ, pode fornecer flexibilidade e eficiência nas configurações de tráfegos onde se deseja priorizar voz.

Na Simulação 2 verificou-se a capacidade do mecanismo WFQ em prover justiça na distribuição dos recursos de rede. Verificou-se também que com a utilização do WRED e ECN podemos refinar a QoS, permitindo ajustes para melhorias no atraso e perdas.

Apesar deste trabalho ter avaliado os mecanismos de QoS em uma pequena rede montada para este fim, fica a perspectiva de sua utilização em uma rede maior como a

Internet, que apesar de exigir maior complexidade de implementação, pode viabilizar a disseminação de aplicações multimídias em tempo real como a telefonia IP.

Como sugestão para novos trabalhos, poderia ser realizada a análise dos mecanismos aqui apresentados com tráfego de vídeo em tempo real.

Anexo 1 – Detalhamento da Configuração do Simulador OPNET

Este anexo apresenta o detalhamento da configuração do simulador "OPNET IT Guru Academic Edition" nas simulações apresentadas no capítulo 5.

Simulação 1

- Configuração das Aplicações
  - HTTP

(Http) Table

Attribute	Value
HTTP Specification	HTTP 1.1
Page Interarrival Time (seconds)	exponential (30)
Page Properties	(...)
Server Selection	(...)
RSVP Parameters	None
Type of Service	Best Effort (0)

(Page Properties) Table

Object Size (bytes)	Number of Objects...	Location	Back-End Custom ...	Object Group Name
pareto (1833, 1.2)	constant (1)	HTTP Server	Not Used	Not Used
pareto (1333, 1.2)	pareto (1, 2.43)	HTTP Server	Not Used	Not Used

(Server Selection) Table

Attribute	Value
Initial Repeat Probability	Browse
Pages Per Server	exponential (10)

- Voz

(Voice) Table

Attribute	Value
Silence Length (seconds)	default
Talk Spurt Length (seconds)	default
Symbolic Destination Name	Voice Destination
Encoder Scheme	G.729
Voice Frames per Packet	1
Type of Service	Interactive Voice (6)
RSVP Parameters	None
Traffic Mix (%)	All Discrete
Signaling	None

- Configuração dos Perfis
  - HTTP

row 1	
Profile Name	Cliente HTTP
Applications	(...)
rows	1
row 0	
Name	Web Browsing (Heavy HTTP1.1)
Start Time Offset (seconds)	constant (0)
Duration (seconds)	End of Profile
Repeatability	(...)
Inter-repetition Time (sec...	exponential (30)
Number of Repetitions	Unlimited
Repetition Pattern	Serial
Operation Mode	Simultaneous
Start Time (seconds)	constant (0)
Duration (seconds)	End of Simulation
Repeatability	(...)
Inter-repetition Time (seconds)	exponential (30)
Number of Repetitions	Unlimited
Repetition Pattern	Serial

- Voz

row 0	
Profile Name	Cliente LAN
Applications	(...)
rows	1
row 0	
Name	Voice over IP Call (PCM Quality)
Start Time Offset (seconds)	constant (10)
Duration (seconds)	End of Profile
Repeatability	(...)
Inter-repetition Time (sec...	exponential (300)
Number of Repetitions	Unlimited
Repetition Pattern	Serial
Operation Mode	Simultaneous
Start Time (seconds)	constant (0)
Duration (seconds)	End of Simulation
Repeatability	(...)
Inter-repetition Time (seconds)	constant (300)
Number of Repetitions	None
Repetition Pattern	Serial

- Configuração de QoS
  - FIFO

FIFO Profiles	(...)
rows	1
row 0	
Profile Name	FIFO Profile
Details	(...)
Maximum Queue Size (pkts)	500
RED Parameters	(...)
RED Status	Not used
Exponential Weight Factor	9
Minimum Threshold	100
Maximum Threshold	200
Mark Probability Denominator	10
CE Marking	Disabled

- PQ

row 0	
Priority Label	0 (Low)
Maximum Queue Size (pkts)	500
Classification Scheme	(...)
rows	2
row 0	Best Effort (0),Unassigned,Unassigned,Unassign...
row 1	Background (1),Unassigned,Unassigned,Unassi...
RED Parameters	(...)
Queue Category	Default Queue
row 1	1 (Normal),60,(...),Disabled,None
row 2	2 (Medium),40,(...),Disabled,None
row 3	
Priority Label	3 (High)
Maximum Queue Size (pkts)	500
Classification Scheme	(...)
rows	2
row 0	Interactive Voice (6),Unassigned,Unassigned,Un...
row 1	Reserved (7),Unassigned,Unassigned,Unassign...
RED Parameters	(...)
Queue Category	None

- WFQ
  - p/ 30%

WFQ Profiles	(...)
rows	5
row 0	
Profile Name	ToS Based
Queues Configuration	(...)
rows	8
row 0	68,500,(...),(...),Default Queue
row 1	10,500,(...),Disabled,None
row 2	20,500,(...),Disabled,None
row 3	30,500,(...),Disabled,None
row 4	40,500,(...),Disabled,None
row 5	50,500,(...),Disabled,None
row 6	32,500,(...),(...),None
row 7	70,500,(...),Disabled,None
Buffer Capacity	1000

- p/ 60%

WFQ Profiles	(...)
rows	5
row 0	
Profile Name	ToS Based
Queues Configuration	(...)
rows	8
row 0	38,500,(...),(...),Default Queue
row 1	10,500,(...),Disabled,None
row 2	20,500,(...),Disabled,None
row 3	30,500,(...),Disabled,None
row 4	40,500,(...),Disabled,None
row 5	50,500,(...),Disabled,None
row 6	62,500,(...),(...),None
row 7	70,500,(...),Disabled,None
Buffer Capacity	1000

- p/ 90%

WFQ Profiles	(...)
rows	5
row 0	
Profile Name	ToS Based
Queues Configuration	(...)
rows	8
row 0	8,0,500,(...),(...),Default Queue
row 1	10,500,(...),Disabled,None
row 2	20,500,(...),Disabled,None
row 3	30,500,(...),Disabled,None
row 4	40,500,(...),Disabled,None
row 5	50,500,(...),Disabled,None
row 6	92,500,(...),(...),None
row 7	70,500,(...),Disabled,None
Buffer Capacity	1000

- CQ
  - p/ 30%

Custom Queuing Profiles	(...)
rows	4
row 0	
Profile Name	ToS Based
Details	(...)
rows	8
row 0	3400,500,(...),Disabled,Default Queue
row 1	4000,20,(...),Disabled,None
row 2	6000,20,(...),Disabled,None
row 3	8000,20,(...),Disabled,None
row 4	10000,20,(...),Disabled,None
row 5	12000,20,(...),Disabled,None
row 6	1600,500,(...),Disabled,None
row 7	16000,20,(...),Disabled,None

  - p/ 60%

Custom Queuing Profiles	(...)
rows	4
row 0	
Profile Name	ToS Based
Details	(...)
rows	8
row 0	1900,500,(...),Disabled,Default Queue
row 1	4000,20,(...),Disabled,None
row 2	6000,20,(...),Disabled,None
row 3	8000,20,(...),Disabled,None
row 4	10000,20,(...),Disabled,None
row 5	12000,20,(...),Disabled,None
row 6	3100,500,(...),Disabled,None
row 7	16000,20,(...),Disabled,None

- p/ 90%

Custom Queuing Profiles	(...)
rows	4
row 0	
Profile Name	ToS Based
Details	(...)
rows	8
row 0	400,500,(...),Disabled,Default Queue
row 1	4000,20,(...),Disabled,None
row 2	6000,20,(...),Disabled,None
row 3	8000,20,(...),Disabled,None
row 4	10000,20,(...),Disabled,None
row 5	12000,20,(...),Disabled,None
row 6	4600,500,(...),Disabled,None
row 7	16000,20,(...),Disabled,None

Simulação 2

- Configuração das Aplicações
  - HTTP

(Http) Table

Attribute	Value
HTTP Specification	HTTP 1.1
Page Interarrival Time (seconds)	exponential (30)
Page Properties	(...)
Server Selection	(...)
RSVP Parameters	None
Type of Service	Excellent Effort (3)

(Page Properties) Table

Object Size (bytes)	Number of Objects...	Location	Back-End Custom ...	Object Group Name
pareto (1833, 1.2)	constant (1)	HTTP Server	Not Used	Not Used
pareto (1333, 1.2)	pareto (1, 2.43)	HTTP Server	Not Used	Not Used

(Server Selection) Table

Attribute	Value
Initial Repeat Probability	Browse
Pages Per Server	exponential (10)

- FTP

(Ftp) Table

Attribute	Value
Command Mix (Get/Total)	50%
Inter-Request Time (seconds)	exponential (180)
File Size (bytes)	pareto (181818, 1.1)
Symbolic Server Name	FTP Server
Type of Service	Standard (2)
RSVP Parameters	None
Back-End Custom Application	Not Used

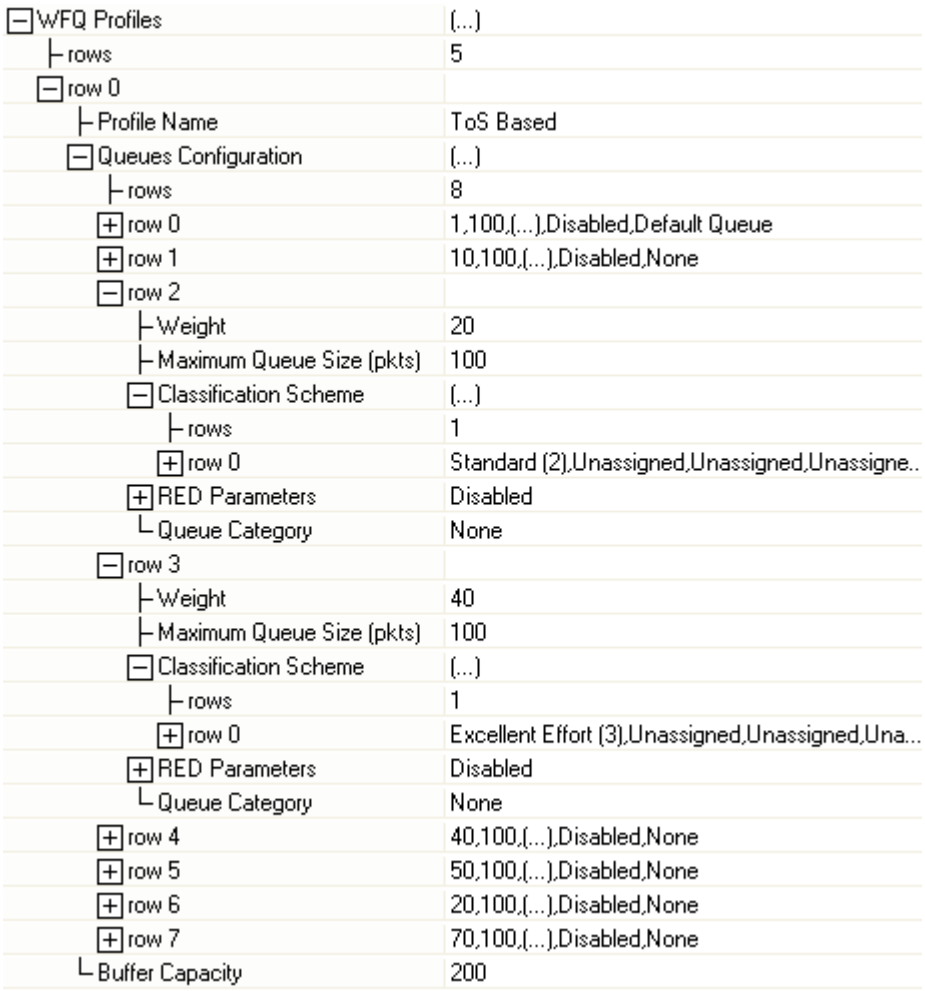
- Configuração dos Perfis
  - HTTP

row 2	
Profile Name	Cliente HTTP
Applications	(...)
rows	1
row 0	
Name	Web Browsing (Heavy HTTP1.1)
Start Time Offset (seconds)	constant (0)
Duration (seconds)	End of Profile
Repeatability	(...)
Inter-repetition Time (sec...	exponential (30)
Number of Repetitions	Unlimited
Repetition Pattern	Serial
Operation Mode	Simultaneous
Start Time (seconds)	constant (0)
Duration (seconds)	End of Simulation
Repeatability	(...)
Inter-repetition Time (seconds)	exponential (30)
Number of Repetitions	constant (0)
Repetition Pattern	Serial

- FTP

row 1	
Profile Name	Cliente FTP
Applications	(...)
rows	1
row 0	
Name	File Transfer (Heavy)
Start Time Offset (seconds)	constant (0)
Duration (seconds)	End of Profile
Repeatability	(...)
Inter-repetition Time (sec...	exponential (180)
Number of Repetitions	Unlimited
Repetition Pattern	Serial
Operation Mode	Simultaneous
Start Time (seconds)	constant (0)
Duration (seconds)	End of Simulation
Repeatability	(...)
Inter-repetition Time (seconds)	exponential (180)
Number of Repetitions	constant (0)
Repetition Pattern	Serial

- Configuração de QoS
    - WFQ c/ *tail drop*



WFQ Profiles	(...)
rows	5
row 0	
Profile Name	ToS Based
Queues Configuration	(...)
rows	8
row 0	1,100,(...),Disabled,Default Queue
row 1	10,100,(...),Disabled,None
row 2	
Weight	20
Maximum Queue Size (pkts)	100
Classification Scheme	(...)
rows	1
row 0	Standard (2),Unassigned,Unassigned,Unassigne..
RED Parameters	Disabled
Queue Category	None
row 3	
Weight	40
Maximum Queue Size (pkts)	100
Classification Scheme	(...)
rows	1
row 0	Excellent Effort (3),Unassigned,Unassigned,Una...
RED Parameters	Disabled
Queue Category	None
row 4	40,100,(...),Disabled,None
row 5	50,100,(...),Disabled,None
row 6	20,100,(...),Disabled,None
row 7	70,100,(...),Disabled,None
Buffer Capacity	200

- WFQ c/ WRED

⊟ WFQ Profiles	(...)
├ rows	5
⊟ row 0	
├ Profile Name	ToS Based
⊟ Queues Configuration	(...)
├ rows	8
⊞ row 0	1,100,(...),Disabled,Default Queue
⊞ row 1	10,100,(...),Disabled,None
⊟ row 2	
├ Weight	20
├ Maximum Queue Size (pkts)	100
⊞ Classification Scheme	(...)
⊞ RED Parameters	Disabled
└ Queue Category	None
⊟ row 3	
├ Weight	40
├ Maximum Queue Size (pkts)	100
⊞ Classification Scheme	(...)
⊟ RED Parameters	(...)
├ RED Status	WRED Enabled
├ Exponential Weight Factor	9
├ Minimum Threshold	50
├ Maximum Threshold	75
├ Mark Probability Denomi...	2
└ CE Marking	Disabled
└ Queue Category	None
⊞ row 4	40,100,(...),Disabled,None
⊞ row 5	50,100,(...),Disabled,None
⊞ row 6	20,100,(...),Disabled,None
⊞ row 7	70,100,(...),Disabled,None
└ Buffer Capacity	200

- WFQ c/ WRED e ECN

WFQ Profiles	(...)
rows	5
row 0	
Profile Name	ToS Based
Queues Configuration	(...)
rows	8
row 0	1,100,(...),Disabled,Default Queue
row 1	10,100,(...),Disabled,None
row 2	
Weight	20
Maximum Queue Size (pkts)	100
Classification Scheme	(...)
RED Parameters	Disabled
Queue Category	None
row 3	
Weight	40
Maximum Queue Size (pkts)	100
Classification Scheme	(...)
RED Parameters	(...)
RED Status	WRED Enabled
Exponential Weight Factor	9
Minimum Threshold	50
Maximum Threshold	75
Mark Probability Denomi...	2
CE Marking	Enabled
Queue Category	None
row 4	40,100,(...),Disabled,None
row 5	50,100,(...),Disabled,None
row 6	20,100,(...),Disabled,None
row 7	70,100,(...),Disabled,None
Buffer Capacity	200

Anexo 2 – Distribuições de Probabilidades

Este anexo apresenta as distribuições de probabilidades utilizadas neste trabalho, com as expressões e definições dos seus parâmetros.

Distribuição de Pareto

A distribuição de Pareto apresenta a seguinte função densidade de probabilidade:

$$f(x) = \frac{\alpha\ k^{\alpha}}{x^{\alpha+1}} \qquad \text{(A2.1)}$$

onde α é o parâmetro de curvatura, $k > 0$ é o parâmetro de localização, e $x \geq k$. A Figura A3.1 apresenta uma exemplificação da função densidade de probabilidade da distribuição de Pareto [73].

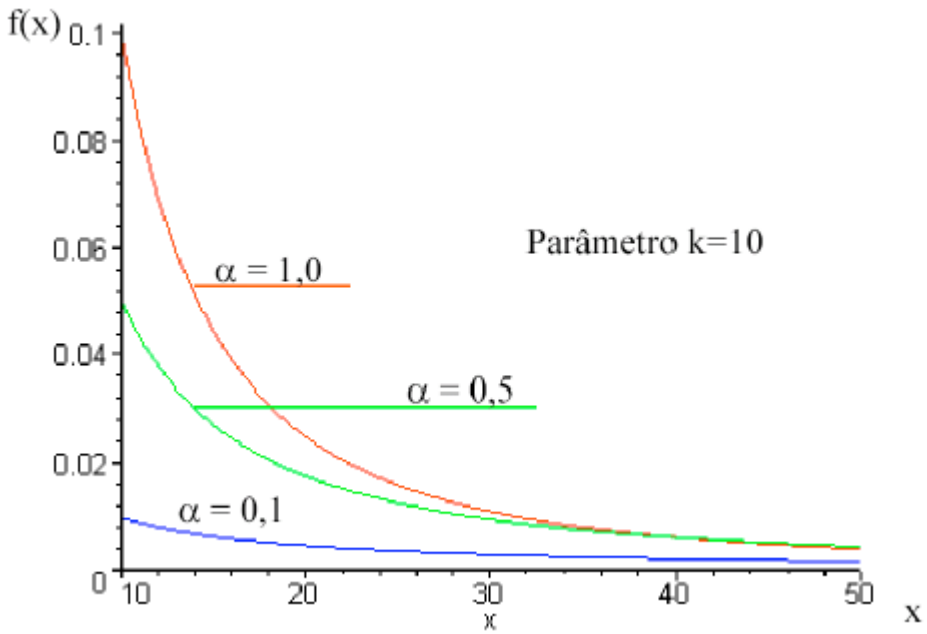


Figura A2.1 – Exemplo da função densidade de probabilidade – Pareto (k, α)

- Média:

$$E(x) = \frac{\alpha\ k}{\alpha - 1} \tag{A2.2}$$

Distribuição Exponencial

A distribuição de Pareto apresenta a seguinte função densidade de probabilidade:

$$f(x) = \frac{e^{-x/\beta}}{\beta}, x \geq 0$$
$$f(x) = 0\ , x < 0 \tag{A2.3}$$

A Figura A3.2 apresenta uma exemplificação da função densidade de probabilidade da distribuição Exponencial [73].

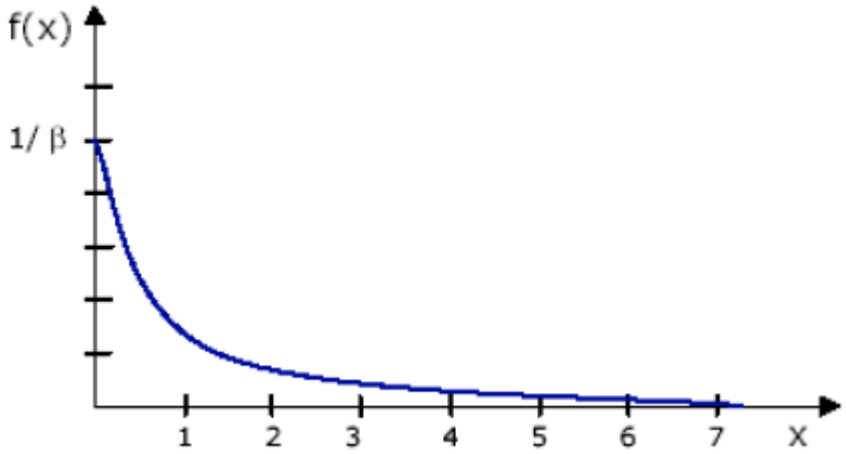


Figura A2.2 – Exemplo da função densidade de probabilidade – Expo (β)

- Média:

$$E(x) = \beta \tag{A2.4}$$

Referências Bibliográficas

[1] Uyless Black, "QOS in Wide Area Networks" – Livro – Publicação Prentice Hall - 2000.

[2] UMTS – "Quality of Service and Network Performance" - Universal Mobile Telecommunication System (UMTS) – Technical Report TR 22.25 V3.1.0 – 1998 – 03.

[3] Shin-Jer Yang and Hung-Cheng Chou, "Adaptive QoS parameters approach to modeling Internet performance" - International Journal of Network Management - January/February 2003. Vol. 13, No 1, PP. 69-82.

[4] Thuan Nguyen, Ferit Yegenoglu, Agatino Sciuto and Ravi Subbarayan, "Voice over IP Service and Performance in Satellite Networks" – IEEE Communications Magazine – March 2001. Vol. 39, No 3, PP. 164-171.

[5] Katsuyoshi Lida, Kenji Kawahara, Tetsuya Takine and Yuji Oie, "Performance Evaluation of the Architecture for End-to-End Quality-of-Service Provisioning" – IEEE Communications Magazine – April 2000. Vol. 38, No 4, PP. 76-81.

[6] Jeong-Soo Han, Seong-Jin Ahn and Jin-Wook Chung, "Study of delay patterns of weighted voice traffic of end-to-end users on the VoIP network" - International Journal of Network Management - September/October 2002. Vol. 11, No 5, PP. 271-280.

[7] Chuck Semeria and John W. Stewart III, “Supporting Differentiated Service Classes in Large IP Networks” – Juniper Networks White Paper (Part Number: 200019-001) – December 2001. (http://www.juniper.net/solutions/literature/white_papers/200019.pdf).

[8] Stefan Brunner and Akhlaq A. Ali, “Voice Over IP 101 – Understanding VoIP Networks” - Juniper Networks White Paper (Part Number: 200087-002) – August 2004. (http://www.juniper.net/solutions/literature/white_papers/200087.pdf).

[9] Johnny M. Matta and Atsushi Takeshita, “End-to-End Voice Over IP Quality of Service Estimation Throught Router Queuing Delay Monitoring” – IEEE Global Telecommunications Conference - November 2002 (GLOBECOM ‘02). Vol. 3, PP. 2458-2462.

[10] Dimitrios Miras, “A Survey of Network QoS Needs of Advanced Internet Applications” - November 2002 - Internet2 QoS Working Group. (http://qos.internet2.edu/wg/apps/fellowship/Docs/Internet2AppsQoSNeeds.pdf)

[11] RFC 3550 – H. Schulzrinne, S. Casner, R. Frederick and V. Jacobson, “RTP: A Transport Protocol for Real-Time Applications” - July 2003.

[12] RFC 1889 – H. Schulzrinne, S. Casner, R. Frederick and V. Jacobson, “RTP: A Transport Protocol for Real-Time Applications” - January 1996.

[13] RCF 2119 - S. Bradner, “Key words for use in RFCs to Indicate Requirement Levels” - March 1997.

[14] José Luiz A. da Fonseca e Michael A. Stanton, “Estudo experimental de videoconferência pessoal em inter-redes IP com QoS” – RNP – Boletim bimestral sobre tecnologia de redes - Dezembro 2001. Vol. 5, No 6.

[15] José Marcos Câmara Brito, “Voz sobre IP – Tecnologias e Aplicações”. (http://www.inatel.br/docentes/brito/Voz_sobre_IP.pdf)

[16] Martin de Prycker, “Asynchronous Transfer Mode, Solution for Broadband ISDN” – Third Edition – Prentice Hall – 1995.

[17] Lucent Technologies, “Impact and Performance of Lucent’s Internet Telephony Server (ITS) over IP Networks” – Copyright 1997 - Lucent Technologies, Inc - November 1997. (http://engr.smu.edu/ee/8392/its_perform.pdf).

[18] Thomas J. Kostas, Michael S. Borella, Ikhlaq Sidhu, Guido M. Schuster, Jacek Grabiec, and Jerry Mahler, “Real-Time Voice Over Packet-Switched Networks” - IEEE Network - January/February 1998.

[19] Applied Technologies, “Voice over IP” Tutorial, Copyright © 1998 by The Applied Technologies Group, Inc. Edited by Jerry Ryan. (http://www.iskon.hr/FAQ/white_papers/VoiceoverIP.pdf)

[20] Intel – Whitepaper: "Overcoming Barriers to High-Quality Voice over IP Deployments" – March 2003. (http://www.intel.com/network/csp/pdf/8539.pdf)

[21] James F. Kurose e Keith W. Ross, "Redes de Computadores e a Internet" – Livro – Publicação Addison Wesley – 3.a Edição - 2006.

[22] Martin W. Murhammer, Kok-Keong Lee, Payam Motallebi, Paolo Borghi and Karl Wozabal, "IP Network Design Guide" - IBM Publication – IBM Redbooks – Document Number SG24-2580-01 - June 1999. (http://www.redbooks.ibm.com/redbooks/pdfs/sg242580.pdf)

[23] Nicolas Christin and Jörg Liebeherr, "A QoS Architecture for Quantitative Service Differentiation" - IEEE Communications Magazine – June 2003. Vol. 4, No 6, PP. 38-45.

[24] Nicolas Christin and Jörg Liebeherr, "Rate Allocation and Buffer Management for Differentiated Services" – Computer Networks, Special Issue on the New Internet Architecture – 2002. Vol. 40, No 1, PP. 89-110.

[25] Nicolas Christin and Jörg Liebeherr, "A Quantitative Assured Forwarding Service" – IEEE INFOCOM - July 2002.

[26] Nicolas Christin and Jörg Liebeherr, "Marking Algorithms for Service

Differentiation of TCP Traffic" – Computer Communications, 2004.

[27] Gayathri Chandrasekaran, "Performance Evaluation of Scheduling Mechanisms for Broadband Networks" – 2001 - Master's Thesis Defense - University of Kansas - Defense Date: July 31, 2003. (http://www.ittc.ku.edu/research/thesis/documents/gayathri_chandrasekaran_thesis.pdf)

[28] Towela Nyirenda-Jere and Victor S. Frost, "Impact of Traffic Aggregation on Network Capacity and Quality of Service"- Technical Report ITTC-FY2002-TR-22730-01 – The University of Kansas Center for Research - Nov. 2001. (http://www.ittc-ku.net/publications/documents/Nyirenda-Jere2001_SPRINT_22730_01_Impact%20of%20Traffic%20Aggregation.pdf)

[29] Cisco, "Internetworking Technology Handbook" – Chapter 49 – February 2003. (http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/qos.pdf)

[30] Cisco – "Cisco IOS Quality of Service Solutions Configuration Guide" – Release 12.2 - Copyright © 2001, Cisco Systems, Inc. (http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122cgcr/fqos_c/qcfbook.pdf)

[31] Paul Ferguson and Geoff Huston, "Quality of Service - Delivering QoS on the Internet and in Corporate Networks" – Book – Wiley Computer Publishing – February 1998.

[32] Agilent Technologies, “IP Quality of Service” – 2000. (http://www.securitytechnet.com/resource/rsc-center/vendor-wp/agilent/5968-9722E.pdf)

[33] RFC 791 – J. Postel, “Internet Protocol” – September 1981.

[34] RFC 1349 – P. Almquist, “Type of Service in the Internet Protocol Suite” – July 1992.

[35] RFC 1700 – J. Reynolds and J. Postel, “Assigned Numbers” – October 1994.

[36] Oliver Hersent, David Guide & Jean-Pierre Petit, “Telefonia IP - Comunicação Multimídia Baseada em Pacotes” – Livro – Publicação Addison Wesley - 2002. (Tradução da 1.a edição – IP Telephony).

[37] RFC 1455 - D. Eastlake, “Physical Link Security Type of Service” – May 1993.

[38] RFC 1812 – F. Baker, “Requirements for IP Version 4 Routers” – Jun 1995.

[39] Dennis Hartmann, “Introduction to Quality of Service” – Global Knowledge – June 2004. (http://searchcio.bitpipe.com/data/document.do?res_id=1088099513_494)

[40] RFC 2474 - K. Nichols, S. Blake, F. Baker and D. Black, “Definition of the

Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers" - December 1998.

[41] RFC 3168 - K. Ramakrishnan, S. Floyd and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP" – September 2001.

[42] Hui-Lan Lu and Igor Faynberg, "An Architectural Framework for Support of Quality of Service in Packet Networks" – IEEE Communications Magazine – June 2003. Vol. 41, No 6, PP. 98-105.

[43] Jasleen Sahni, Pawan Goyal and Harrick M. Vin, "Scheduling CBR Flows: FIFO or Per-flow Queuing?" - 9th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV99), June 1999. PP 13-27.

[44] R. Guérin and V. Peris, "Quality-of-service in packet networks: basic mechanisms and directions" – Computer Networks, 31:3, PP. 169-189 – February – 1999.

[45] Adailton J. S. Silva, "Qualidade de Serviço em VoIP – Parte I" – RNP – Boletim bimestral sobre tecnologia de redes - Maio 2000. Vol. 4, No 3.

[46] Syed Ljlal Ali Shah, "Bringing Comprehensive Quality of Service Capabilities to Next-Generation Networks" – Motorola QOSM-WP/D - October 2001.

[47] Daniel Cavas Otero, “Diferenciação e QoS em Redes Sem Fio” – Junho 2003 – Publicação GARF (Grupo de Atuação em Redes sem Fio) – COPPE – UFRJ. (http://www.garf.coppe.ufrj.br/publicacoes.htm)

[48] Antônio M. Alberti, Daniel Zaccarias, Samuel Penha e Lenardo de Souza Mendes, “Modelamento e Simulação de um Escalonador WF^2Q Aplicado à Redes ATM” – Inatel – Revista Científica Periódica – Telecomunicações – Dezembro de 2001. Vol. 04, No 02, PP. 22-36.

[49] Pawan Goyal, Harrick M. Vin, and Haichen Cheng, “Start-Time Fair Queueing: A Scheduling Algorithm for Integrated Services Packet Switching Networks” – IEEE / ACM Transactions on Networking – October 1997. Vol. 5, No 5, PP. 690-704.

[50] Jörg Liebeherr, Notas de aula da disciplina ECE1545F – Bridges and Routers - Department of Electrical and Computer Engineering - University of Toronto – November 2005. (http://www.comm.utoronto.ca/%7Ejorg/ece1545/exercises/Ex6.pdf)

[51] David Harrison, "Edge-to-edge Control: A Congestion Control and Service Differentiation Architecture for the Internet" - Ph.D. Dissertation, Computer Science Department, Rensselaer Polytechnic Institute, May 2002. (http://poisson.ecse.rpi.edu/~harrisod/thesis.pdf)

[52] S. Jamaloddin Golestani, “A Self-Clocked Fair Queueing Scheme for

Broadband Applications" - IEEE INFOCOM'94 - Toronto, Canadá - June 1994. PP. 636-646.

[53] J.C.R. Bennett and H. Zhang, "WF^2Q: Worst-case Fair Weighted Fair Queuing" - IEEE INFOCOM'96 – San Francisco, USA – March 1996. PP. 120-127.

[54] Bartek Wydrowski and Moshe Zukerman, "Qos in Best-Effort Networks" – IEEE Communications Magazine – December 2002. Vol. 40, No 12, PP. 44-49.

[55] Douglas E. Comer, "Redes de Computadores e Internet" - Livro – Publicação Bookman – 2.a Edição - 2001.

[56] RFC 2001 - W. Stevens, "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms" - January 1997.

[57] RFC 2581 - M. Allman, V. Paxson and W. Stevens, "TCP Congestion Control" – April 1999.

[58] RFC 3390 - M. Allman, S. Floyd and C. Partridge, "Increasing TCP's Initial Window" - October 2002.

[59] Geoff Huston, "TCP Performance" – CISCO – The Internet Protocol Journal – June 2000. Vol. 3, No 2, PP. 2-24.

[60] Sally Floyd and Van Jacobson, "Random Early Detection Gateways for

Congestion Avoidance" – IEEE/ACM Transactions on Networking – August 1993. Vol. 1, No 4, PP. 397-413.

[61] Miguel A. Labrador and Sujata Banerjee, "Packet Dropping Policies for ATM and IP Networks" – IEEE Communications Surveys – Third Quarter 1999. Vol. 2, No 3, PP 2-14.

[62] Subrat Kar, Peyman Farjami and Rajiv Chakravorty, "Issues and Architectures for Better Quality of Service (QoS) from the Internet" - Proc. NCC-2000 - 6th National Conference on Communications – New Delhi, India - January 2000. Vol. 0, PP. 181-187.
http://www.comnets.rwth-aachen.de/typo3conf/ext/cn_download/pi1/passdownload.php?downloaddata=207|/var/www/cnhome/downloads/publications/)

[63] RFC 2309 - B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, L. Zhang, "Recommendations on Queue Management and Congestion Avoidance in the Internet" – April 1998.

[64] José Ferreira de Rezende, "Tópicos Especiais em Integração de Serviços na Internet" – 2003 - Grupo de Teleinformática e Automação (GTA) – COPEE – UFRJ. (http://www.gta.ufrj.br/~rezende/cursos/coe889/)

[65] L. Guan, M. E. Woodward and I. U. Awan, "A Discrete-Time Performance

Model for Congestion Control Based on Random Early Detection Using Queue Thresholds" – UKSim 2004 - United Kingdom Simulation Society Conference – Oxford - 29-31 March 2004. No 04-07. (http://ducati.doc.ntu.ac.uk/uksim/uksim'04/Papers/Glamorgan-Bradford%20papers/Lin%20Guan-%2004-07/paper04-07%20CR.pdf)

[66] XiPeng Xiao, Thomas Telkamp, Victoria Fineberg, Cheng Chen and Lionel M. Ni, "A Practical Approach for Providing QoS in the Internet Backbone" - IEEE Communications Magazine – December 2002. Vol. 3, No 12, PP. 56-64.

[67] Bartek Wydrowski and Moshe Zukerman, "High Performance DiffServ Mechanism for Routers and Switches: Packet Arrival Rate Based Queue Management for Class Based Scheduling" – Networking 2002, Pisa, Italy – May 2002. PP. 62-73.

[68] Bartek Wydrowski and Moshe Zukerman, "Implementation of Active Queue Management in a Combined Input and Output Queued Switch" – ICC – IEEE International Conference on Communications - May 2003. Vol. 1, PP. 168-172.

[69] Tania Aida Garcia Bassi, "Análise do Desempenho do Sistema 3G 1xEV-DV no Enlace Direto" - Tese de Mestrado apresentada à Faculdade de Engenharia Elétrica e de Computação (UNICAMP) – Campinas, SP, Brasil – Fevereiro 2006.

[70] Fernando Hwang, Gabriel Rocon Bianchi e Luan Ling Lee, "Impacto Gerado

pelo Comportamento das Aplicações (Web, FTP e E-mail) e pelo Perfil das Redes na Característica Auto-Similar" - Revista IEEE América Latina – Outubro 2005. Vol. 3, No 4, PP. 30-35.

[71] Dirk Staehle, Kenji Leibnitz, and Phuoc Tran-Gia, "Source Traffic Modeling of Wireless Applications" - International Journal of Electronics and Communications – 2001 - Vol. 55, No 1.

[72] Alejandro Quintero, Yacine Elalamy and Samuel Pierre, "Performance evaluation of a broadband wireless access system subjected to heavy load" – Computer Communications 27 – June 2004. Vol. 27, No 9, PP. 781-791.

[73] Fernando Hwang, "Análise dos Efeitos Gerados pelo Comportamento das Aplicações e pelo Perfil das Redes na Característica Auto-Similar do Tráfego Internet" - Tese de Mestrado apresentada à Faculdade de Engenharia Elétrica e de Computação (UNICAMP) – Campinas, SP, Brasil – Fevereiro 2004.

[74] Hyoung-Kee Choi and John O. Limb, "A Behavioral Model of Web Traffic" - Annual International Conference on Network Protocols, Toronto, Canada, 1999. PP. 327-334.

[75] E. Casilari, A. Reyes-Lecuona, F.J. González, A. Díaz-Estrella and F. Sandoval, "Characterisation of Web Traffic" - Global Telecommunications Conference, 2001. GLOBECOM '01. IEEE , Vol. 3 , PP. 1862-1866.

[76] Bruce A. Mah, “An Empirical Model of HTTP Network Traffic” – IEEE INFOCOM Kobe, Japan, April 7 -12, 1997 – Vol. 2, No 5C, PP. 592-600.

[77] A. Reyes-Lecuona, E. González-Parada, E. Casilari, J. C. Casasola and A. Díaz-Estrella, “A page-oriented WWW traffic model for wireless system simulations” - Proceedings of the 16th International Teletraffic Congress (ITC'16), Edinburgh, United Kingdom, June 1999. PP. 1271-1280.

[78] Ajit K Jena, Adrian Popescu and Parag Pruthi, “Modeling and Analysis of HTTP Traffic” - The ITC Specialist Seminar on IP Traffic Measurement, Modeling and Management, Monterey, California, USA, 2000. (http://www.bth.se/fou/forskinfo.nsf/7172434ef4f6e8bcc1256f5f00488045/7e6beb9743ce33a8c1256c75006f4b6f/$FILE/final_paper.pdf).

[79] OPNET Modeler Product Documentation – Release 10.5 – Methodologies and Case Studies – Network Analysis and Design - “Case Study: Managing Server Traffic”.