UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA DE TELEINFORMÁTICA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGª DE TELEINFORMÁTICA - PPGETI

Controle PID Híbrido e Dinâmico de QoS Baseado em Políticas para Ambiente DiffServ

Aluno: Manoel Benedito da Cunha Morais

Orientador: Prof. Danielo G. Gomes, Dr.

Co-orientador: Prof. Pedro Klécius Farias Cardoso, Dr.

Fortaleza, CE – Brasil

Fevereiro de 2009

UNIVERSIDADE FEDERAL DO CEARÁ
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA DE TELEINFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGª DE TELEINFORMÁTICA - PPGETI

Controle PID Híbrido e Dinâmico de QoS Baseado em Políticas para Ambiente DiffServ

Manoel Benedito da Cunha Morais

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia de Teleinformática da Universidade Federal do Ceará (PPGETI/UFC), como parte dos requisitos para obtenção do título de mestre em Engenharia de Teleinformática.

Fortaleza, CE – Brasil
Fevereiro de 2009

Manoel Benedito da Cunha Morais

Controle PID Híbrido e Dinâmico de QoS Baseado em Políticas para Ambiente DiffServ

Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Engenharia de Teleinformática e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia de Teleinformática da Universidade Federal do Ceará.

Manoel Benedito da Cunha Morais

Manoel Benedito da Cunha Morais

Banca Examinadora:

Prof. Danielo Gonçalves Gomes, Dr.
Universidade Federal do Ceará – UFC

Prof. Paulo André da Silva Gonçalves, Dr.
Universidade Federal de Pernambuco – UFPE

Prof. José Tarcísio Costa Filho, Dr.
Universidade Federal do Ceará – UFC

Prof. Pedro Klécius Farias Cardoso, Dr.
Centro Federal de Educação Tecnológica – CEFET/CE

Fortaleza – CE, 9 de fevereiro de 2009.

Dedico este trabalho à minha esposa Ana Paula, meus filhos Emanoel Victor e Guilherme Vinícius e principalmente à minha mãe Maria Rita.

Agradecimentos

À minha esposa Ana Paula, pela compreensão e paciência.

Aos meus filhos Emanoel Victor e Guilherme Vinícius por me trazerem tanta felicidade.

Ao professor Dr. Pedro Klécius pelo apoio, co-orientação, amizade e por ceder um espaço em seu laboratório para o desenvolvimento deste trabalho.

Ao professor Dr. Danielo G. Gomes pelo apoio, orientação, motivação e entusiasmo.

Ao CEFET-CE pela oportunidade de realizar este curso de mestrado.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

AF	Encaminhamento Assegurado (***Assured Forwarding***)
AQM	***Active Queue Management***
BE	Melhor esforço (***Best Effort***)
CBR	***Constant Bit Rate***
CoS	Classe de Serviço
DiffServ	Serviços Diferenciados (***Differentiated Services***)
DS	Serviços Diferenciados (***Differentiated Services***)
DSCP	***Differentiated Services Code-Point***
EF	Encaminhamento Expresso (***Expedited Forwarding***)
FTP	Protocolo de transferência de arquivos (***File Transfer Protocol***)
IETF	***Internet Engineering Task Force***
IntServ	Serviços Integrados (***Integrated Services***)
IP	***Internet Protocol***
IPv4	***Internet Protocol version 4***
IPv6	***Internet Protocol version 6***
ITU	***International Telecommunication Union***
ITU-T	***Telecommunication Standardization Sector of ITU***
NS-2	***Network Simulator - 2***
OPNET	***OPtimised Network Engineering Tools***
P	Proporcional (controlador)
PI	Proporcional e Integrador (controlador)
PID	Proporcional, Integrador e Derivativo (controlador)
PHB	Comportamento por nó (***Per-Hop Behaviour***)

QoS	Qualidade de Serviço (***Quality of Service***)
RED	***Random Early Detection***
RFC	Recomendação do IETF (***Request For Comment***)
SLA	Contrato de nível de serviço (***Service Level Agreement***)
SLS	Especificação de Nível de Serviço
SP	Valor de referência (***set-point***)
TCA	***Traffic Conditioning Agreement***
TCP	***Transmission Control Protocol***
ToS	***Type of Service***
VoIP	Voz sobre IP
WFQ	***Weighted Fair Queuing***
WRR	***Weighted Round Robin***

Resumo

A convergência dos serviços e o aumento vertiginoso do tráfego na Internet dos últimos anos sugerem a busca por novos mecanismos para lidar com requisitos cada vez mais desafiadores de Qualidade de Serviço (QoS). Este trabalho apresenta a proposta de um controlador dinâmico, não-linear, que através do ajuste dos pesos do algoritmo WFQ em ambiente DiffServ, associado a políticas estabelecidas, proporciona a manutenção dos níveis de QoS das aplicações isócronas de multimídia, dentro dos limites aceitáveis, mesmo em condição de congestionamento severo e de grande variação de carga na rede. Os resultados alcançados pelo mecanismo dinâmico proposto são motivadores e, quando comparados aos do DiffServ clássico, apontam ganhos de eficiência na utilização dos recursos da rede.

Palavras-Chave — QoS, controle, WFQ, políticas**.**

Abstract

Converged IP-based networks and services, as well the huge increase in IP traffic along the last few years, have revealed the need for new dynamic mechanisms capable of handling all types of IP-based multimedia traffic. This work presents a dynamic, nonlinear controller, that adapt the weights of WFQ scheduling in a DiffServ architecture to maintain the QoS levels of the isochronous multimedia's applications, even under conditions of severe congestion and large load variation on the network. The results presented here show that our objective is accomplished with the correct fitting of the WFQ parameters without overestimating their weights.

Keywords — QoS control, WFQ, policies**.**

Capítulo 1 – Introdução

Na iminência de completar 40 anos de existência[1], a Internet tornou-se parte do cotidiano de aproximadamente 1,5 bilhão de usuários no mundo. Nascida de um projeto de pesquisa de uma agência norte-americana, a ARPA (***Advanced Research Project Agency***), permaneceu restrita ao mundo acadêmico até o final da década de 80. O protocolo IP (***Internet Protocol***) foi desenvolvido e implementado originalmente como um protocolo de comunicação com controle de tráfego utilizando a regra do melhor esforço (***best effort***), que não provê nenhum mecanismo de Qualidade de Serviço (QoS) e, conseqüentemente, nenhuma garantia de alocação de recursos da rede. Não se imaginava, na época que a Internet se tornaria a grande rede mundial que é atualmente.

Houve um crescimento intenso, sobretudo nos últimos 10 anos, das aplicações de tempo real e de tráfego interativo no mundo IP (***Internet Protocol***). A convergência dessas novas aplicações sobre as redes IP gera novos desafios sob o ponto de vista de rede, para a satisfação dos usuários finais, pois as mídias audiovisuais trazem consigo outros requisitos de tratamento bem distintos dos tradicionais.

Com o constante crescimento da Internet, a tendência atual é a integração de voz (telefonia), vídeo e dados numa única infra-estrutura de redes de pacotes, a rede IP. Essa emergente e crescente demanda pelos serviços multimídia, de videoconferência, VoIP, vídeo sob demanda, que necessitam cada vez mais de maior banda passante, sendo inclusive alguns sensíveis ao atraso de transmissão (***delay***) e à sua variação (***jitter***), como os serviços em tempo real, ocasionou intensas atividades de pesquisa por parte do IETF (***Internet Engineering Task Force***) e dos fabricantes de equipamentos de redes para desenvolverem protocolos, técnicas e mecanismos que garantissem QoS fim-a-fim às redes IP, propiciando a apresentação de diversas propostas nos últimos anos.

[1] A primeira RFC (***Request for Comments***) do IETF (***Internet Engineering Task Force***) data de abril de 1969.

1.1 O problema

O tráfego na Internet não só cresceu em volume, como também mudou em sua natureza. O crescimento vertiginoso do tráfego na Internet propicia a ocorrência de congestionamento, que degrada o desempenho das redes de computadores. Dentre as consequências negativas da sua ocorrência, cita-se a diminuição da vazão, a perda de pacotes e o aumento do atraso.

As aplicações tradicionais como o correio eletrônico e transferência de arquivos, chamadas de aplicações elásticas, são tolerantes a atrasos e intolerantes a perdas. Novas aplicações de tempo real, como o tráfego de voz e imagem, por serem interativas, são críticas e necessitam de requisitos especiais, como alta disponibilidade, alta largura de banda e baixa taxa de erro e de atraso. Essas aplicações multimídia, ditas isócronas, em oposição às elásticas, toleram um certo grau de perdas, mas não toleram atrasos.

O serviço tradicional e de fato provido no mundo IP, conhecido como o serviço de melhor esforço (***best effort***), no qual inexiste o provisionamento de qualidade de serviço, não é adequado para atender às demandas das aplicações avançadas (e.g., videoconferência, telemedicina) e das novas aplicações que vêm surgindo com a convergência dos serviços e das mídias audiovisuais. Qualidade de Serviço tem sido considerada um requisito desejável para diversos tipos de rede, em diferentes épocas, sendo tema de pesquisa e padronização de vasta quantidade de trabalhos, sobretudo nos últimos 20 anos. A base de dados Scopus[2] lista aproximadamente 7000 artigos, publicados desde 1990, trazendo o acrônimo QoS em seus títulos. Apesar dessa vigorosa atenção dada ao assunto, quase a totalidade dos mecanismos de QoS não é adotada ou implementada em larga escala nas redes reais, sejam redes do mundo IP, sejam redes do mundo Telecom. Portanto, constitui-se um grande desafio o desenvolvimento de mecanismos de QoS que possam ser integrados à estrutura existente da Internet e que consigam suprir as novas necessidades de comunicação e oferecer garantias de desempenho a determinados usuários e aplicações.

[2] http://www.scopus.com/scopus/home.url

1.2 Objetivos

Entre os objetivos gerais dessa dissertação, cita-se a análise de métodos de provisionamento dinâmico de Qualidade de Serviço em redes IP, o estudo investigativo de disciplinas de enfileiramento e de políticas de controle.

De um modo mais específico, o presente trabalho apresenta uma proposta de controle em malha fechada utilizando a arquitetura DiffServ em conjunto com o algoritmo PGPS (***Packet-by-Packet Generalized Processor Sharing***), também conhecido como WFQ (***Weighted Fair Queueing***), capaz de manter os níveis de QoS das aplicações multimídia, através do controle de seus atrasos fim a fim, mediante o ajuste ponderado correto dos parâmetros de peso do WFQ. Em caso de congestionamento da rede, o mecanismo proposto garante um limite máximo de atraso aceitável.

1.3 Trabalhos relacionados

A necessidade de oferecer garantias de QoS na Internet nos leva a mecanismos de provisionamento dinâmico, já que soluções estáticas não são adequadas à dinâmica da rede. A grande vantagem do provisionamento dinâmico é que há um aproveitamento maior da capacidade da rede, que pode oferecer um serviço de melhor qualidade mantendo a infra-estrutura dimensionada de acordo com a demanda. Assim pode-se dizer que o provisionamento dinâmico pode oferecer QoS com um custo menor (Fernandez, 2002).

Várias propostas têm surgido com a finalidade de ajustar dinamicamente os parâmetros de peso do algoritmo WFQ de acordo com as condições de tráfego da rede (PANZA ***et al***., 2005; PANZA ***et al***., 2006). Todavia, alguns algoritmos de predição do tráfego (DRUMMOND, 2005) ou de inteligência computacional têm apresentado limitações que dificultam sua aplicação no provimento de QoS em redes IP, seja devido ao alto custo computacional imposto aos nós da rede ou à necessidade de uma estação central de controle, o que implica em limitações de escalabilidade (FERNANDEZ, 2002; FERNANDEZ ***et al***., 2004).

Dos trabalhos relacionados com o ajuste dinâmico dos pesos do algoritmo WFQ, destaca-se uma proposta de extensão ao WFQ baseada em técnicas de predição de modelo de tráfego IP (Gallardo e Makrakis, 2001) e a proposta de um algoritmo WFQ adaptativo denominado AWFQ, que utiliza as estimativas das taxas de chegada dos fluxos para proceder ao ajuste dos pesos (Wang et al., 2006). Os resultados destes trabalhos mostram apenas uma melhoria nos níveis de QoS sem estabelecer um limite máximo de referência do atraso dos fluxos controlados em caso de congestionamento severo da rede.

Em (FERNANDEZ, 2002; FERNANDEZ ***et al***., 2004), é proposto um controlador baseado em lógica difusa (***fuzzy***) que reconfigura os pesos do algoritmo WRR nos nós de um domínio Diffserv de acordo com o tráfego entrante. Este controlador foi comparado a um controlador PD (Proporcional e Derivativo) digital simples executando um controle intuitivo. A idéia desse controlador é guardar as últimas três medidas de retardo da classe EF e ajustar uma reta a esses pontos. A inclinação dessa reta é aplicada ao peso do escalonador, aumentando ou reduzindo conforme a inclinação da reta. Dos resultados obtidos, vale destacar que ao se variar o intervalo de operação dos controladores com intervalo pequeno, o controlador ***fuzzy*** é apenas um pouco melhor que o controlador convencional, tornando-se mais eficiente com intervalos maiores. O controlador ***fuzzy*** do referido trabalho apresentou um desempenho superior, mas não tão marcante em relação ao controlador convencional.

Outro trabalho na linha da inteligência computacional, aplicada a um processo térmico de segunda ordem, compara os desempenhos de um controlador nebuloso e de um controlador PID (Proporcional-Integrador-Derivativo) otimizado através da técnica de Algoritmos Genéticos (AG) (ROMÃO ***et al***., 1999). Os resultados desse trabalho mostram que o controlador ***fuzzy*** apresentou menor tempo de estabilização, mas o controlador PID apresentou a menor variância do erro. Logo, os controladores PID, se bem projetados e sintonizados, constituem uma opção de solução.

As políticas de gerenciamento ativo de filas ou AQM (***Active Queue Management***), implementadas com o auxílio da teoria de controle, utilizam na sua grande maioria os controladores clássicos do tipo PID e suas derivações (AGUILAR E TORRES, 2007; LIMA, 2005; XU ***et al***., 2005). Por exemplo, um mecanismo de prevenção de congestionamento em redes TCP/IP, o algoritmo RED (***Random Early Detection***) e sua variante LRED (***Loss Ratio based*** RED) são controladores do tipo P; o ***Dynamic*** RED (DRED) é de fato um controlador do tipo I e o PI-AQM utiliza um controlador do tipo PI.

1.4 Contribuições

A principal contribuição desse trabalho é o desenvolvimento de um mecanismo de controle, associado a políticas estabelecidas, que proporciona a manutenção dos parâmetros de QoS dos fluxos mais prioritários dentro dos limites normatizados, mesmo em condição de congestionamento severo, elevando a eficiência na distribuição dos recursos da rede. Quando os fluxos mais prioritários tomam posse de uma quantidade minimizada de recursos, com a garantia de que seus níveis de QoS permaneçam em valores aceitáveis, sobram mais recursos para os demais fluxos.

1.5 Estrutura do trabalho

O trabalho está organizado da seguinte forma: No Capítulo 2 são apresentados sucintamente os conceitos básicos utilizados, notadamente os relativos à teoria de controle, Arquitetura dos Serviços Diferenciados (DiffServ) e algoritmo de escalonamento WFQ. Os cenários de estudo, bem como suas respectivas simulações preliminares à proposta, são descritos no Capítulo 3. No Capítulo 4 são apresentados a proposta do controlador associado às políticas definidas e os principais resultados obtidos na sua implementação. Finalmente, o Capítulo 5 traz as conclusões e algumas perspectivas futuras para esse trabalho.

Capítulo 2 - Fundamentos Conceituais

Neste capítulo, são introduzidos os conceitos básicos da arquitetura de serviços diferenciados, o algoritmo de enfileiramento WFQ e a teoria de controle necessários à compreensão do trabalho desenvolvido nessa dissertação.

2.1 A arquitetura de serviços diferenciados (DiffServ)

A recomendação do IETF 2475 estabeleceu uma arquitetura baseada em serviços diferenciados (RFC 2475, 1998). O modelo DiffServ trabalha com o conceito de classes de serviço e visa agrupar fluxos de tráfego semelhantes em uma mesma classe, permitindo a redução no número de estados a serem mantidos nos roteadores de um provedor. A arquitetura de serviços diferenciados tem por base um conjunto de mecanismos relativamente simples. À entrada de uma rede, o tráfego é classificado, condicionado e integrado em uma das diferentes classes de tráfego, sendo cada classe caracterizada pelo respectivo ***Differentiated Service Code Point*** (DSCP), que é um campo de 6 bits no cabeçalho dos pacotes IP.

As redes que implementam serviços diferenciados são chamadas de domínios DiffServ. Domínios DiffServ negociam entre si contratos que visam o provimento de garantias mínimas de QoS para as aplicações dos usuários. Os roteadores de borda ou ERs (***Edge Routers***) tem as tarefas mais complexas de classificação e policiamento dos fluxos que entram no provedor de serviços DiffServ. Nos roteadores centrais, chamados de TRs (***Transient Routers***), o processamento é simplificado devido à ausência de policiamento e à redução da complexidade da função de classificação, eles identificam as classes de serviço pela leitura direta do campo DSCP do cabeçalho IP. Para o IPV4, a classe é estabelecida no campo ToS (RFC 791, 1981), e no IPV6, o campo ***class of traffic*** foi criado para se fazer tal identificação (RFC 2460, 1998). Pacotes distintos podem ter tratamentos distintos nos roteadores, de acordo com seus requisitos de QoS. Esse tratamento específico de encaminhamento é chamado de ***Per-Hop-Behavior*** ou PHB. Os ***Per-Hop-Behaviors*** definem os procedimentos de encaminhamento a serem realizados nos pacotes e são usados para referir um agregado de fluxos que requerem um mesmo tratamento.

Dois PHBs foram padronizados pelo IETF: encaminhamento expresso (EF) (RFC 2598, 1999) e encaminhamento assegurado (AF) (RFC 2597, 1999). O PHB EF realiza uma alocação explícita de recursos para a sua agregação de fluxos. Tem como característica baixo retardo e baixa variação do retardo, taxa de erros controlada e largura de banda assegurada. O PHB-AF oferece garantias estatísticas de banda passante. Nesse PHB, não há uma garantia estrita na entrega dos pacotes, o que há é uma garantia de que pacotes marcados como conformes são entregues com alta probabilidade, enquanto que os que excederem a taxa especificada no contrato recebem uma probabilidade de descarte maior, podendo assim, serem mais facilmente descartados em momentos de congestionamento.

O encaminhamento expresso (EF) define garantias mais rígidas de QoS para aplicações muito sensíveis a variações de características temporais da rede. Ele pode ser utilizado para implementar um serviço com baixo retardo, baixa variação do atraso (***jitter***) e largura de banda garantida. Para os usuários, esse serviço, conhecido como ***Premium***, parece com uma "linha privada virtual". É ideal para aplicações que necessitam de rapidez, constância na sua transmissão, com pouco ou nenhum erro, como por exemplo, telemedicina, Voz sobre IP (VoIP), videoconferência, ou para emular uma linha dedicada virtual.

O encaminhamento assegurado (AF) define quatro classes de serviço, cada uma com três níveis de precedência para descarte. Tem por objetivo entregar os pacotes IP, com banda passante assegurada, mas não oferece garantias quanto ao atraso. Em cada classe AF os pacotes IP são marcados com um dos três valores que indicam a probabilidade de descarte. Em caso de congestionamento, o descarte será feito considerando-se a classe e a probabilidade de descarte escolhidos. O nó congestionado tenta proteger os pacotes com baixa prioridade de descarte. Desse modo, há uma flexibilidade maior entre os diversos fluxos estabelecidos, uma vez que pacotes de um fluxo de menor prioridade podem ser descartados se um outro fluxo de maior prioridade necessitar dos seus recursos.

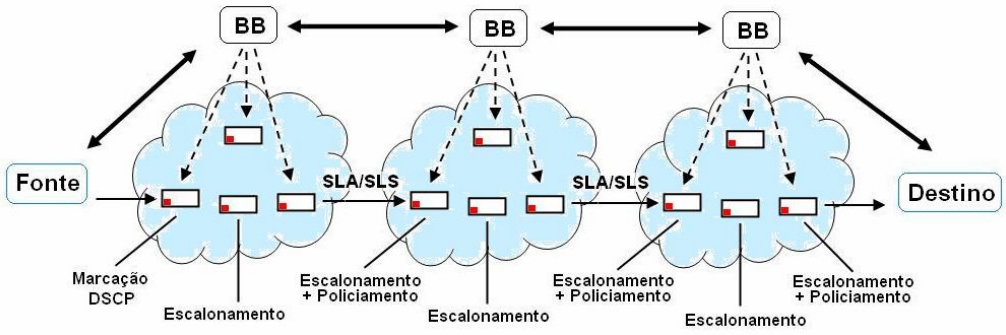

Figura 1: Arquitetura DiffServ

Na Figura 1, são mostrados os principais elementos da arquitetura de serviços diferenciados. O negociador de banda ou ***bandwidth broker*** (BB) é um agente responsável pela reserva de recursos para os tipos de serviços solicitados pelos usuários e configuração dos roteadores de borda com o PHB correto utilizando o DSCP. Podem existir mais de um BB atuando em um mesmo domínio. No entanto, apenas um dentre eles pode ser responsável pela comunicação com os BBs de domínios adjacentes e o responsável pela configuração dos roteadores de borda para que entre e saia do domínio a quantidade de tráfego desejada para cada classe.

O policiamento visa garantir que pacotes não ultrapassem o limiar contratado e requisitado no momento anterior ao estabelecimento do fluxo. Na Figura 1, apenas os roteadores de borda fazem o policiamento de tráfego. As setas contínuas mais espessas indicam a sinalização de controle entre aplicação e o controlador de banda e entre os controladores de banda de domínios diferentes. As setas tracejadas indicam a sinalização de controle exercida pelo controlador de banda nos roteadores de borda que compõem o domínio DiffServ. Por fim, as setas contínuas mais finas indicam o sentido do fluxo de pacotes que terão serviço diferenciado.

O escalonamento é o processo de decidir, no momento da transmissão, qual fila deve ser servida, dependendo da prioridade solicitada pelo pacote. Esse processo está associado a um algoritmo de enfileiramento de pacotes, sendo empregado no caso de otimização de recursos na saída do nó.

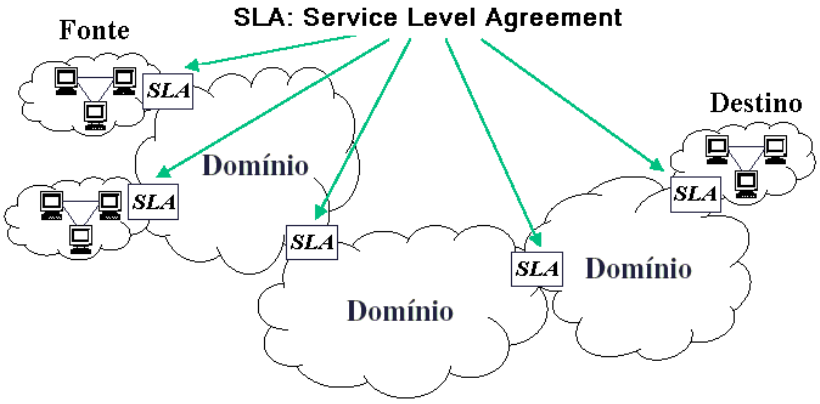

Figura 2: SLA: ***Service Level Agreement***

O contrato de nível de serviço (s***ervice level agreement - SLA***), descrito na Figura 2, é um contrato entre domínios DiffServ que estabelece alguns parâmetros de QoS que devem ser respeitados pelo domínio que está injetando pacotes em outro domínio. O SLA é traduzido para parâmetros de rede como banda contratada, vazão, atraso máximo, perda de pacote, etc., em um documento auxiliar conhecido por especificação do nível de serviço (s***ervice level specification –*** SLS***)***. SLA também pode ser estabelecido entre um cliente e um fornecedor de serviço na fronteira de um domínio DS, que especifica o serviço que o cliente deve receber, podendo incluir regras explícitas de condicionamento de tráfego, que fazem parte de um contrato de condicionamento de tráfego (***traffic conditioning agreement*** - TCA). TCA é um acordo que especifica as regras de classificação de pacotes e perfis de tráfego correspondentes e as regras de condicionamento de tráfego a aplicar aos fluxos de tráfego selecionados pelo classificador que inclui as regras explicitamente indicadas num SLA e regras implícitas derivadas dos requisitos do serviço e/ou das políticas de provisão de serviços no domínio. O SLA tem como objetivo especificar os níveis mínimos de desempenho que um provedor de serviços deverá manter a disposição do usuário e o não cumprimento desse acordo implica em penalidades, estipuladas contratualmente. O contrato SLA pode se estático ou dinâmico, porém um contrato dinâmico exige a utilização de um protocolo de sinalização e controle de forma a gerenciar a largura de banda disponível.

O modelo DiffServ, apesar de ser escalável, não oferece a garantia rígida de recursos para todos os fluxos, como o IntServ, pois tem como característica marcante o suporte de diferentes níveis de serviço a diferentes fluxos de informação agregados em classes de serviço (CoS). As reservas de recursos são feitas para grandes conjuntos de fluxos. Um fluxo individual pode não atingir as suas necessidades em termos dos parâmetros de QoS requeridos.

2.1.1 Classes de serviços

2.1.1.1 Serviço premium

O Serviço Premium tem como características a taxa de pico de transmissão e o retardo extremamente baixo. São permitidas duas ações com os pacotes contratados. O descarte de pacotes acima da taxa de pico, caracterizando o policiamento ou atrasá-los até que o tráfego novamente esteja de acordo com o contratado, caracterizando a suavização. Com essas características, fica óbvio que este serviço pode ser suportado pelo PHB-EF (e.g., redes privadas virtuais, telefonia via Internet e videoconferência).

2.1.1.2 Serviço assegurado

O serviço assegurado é caracterizado por um nível estatístico e menos rígido que o Serviço Premium. O nível de serviço geralmente é definido por uma largura de banda contratada. Para cada usuário, os pacotes dentro do perfil devem ser entregues com alta probabilidade. Já o restante é marcado de forma diferente e pode ser ou não entregue. Em momentos de congestionamento, os pacotes fora de perfil têm prioridade no descarte. Este grupo de PHBs pode ser usado para implementar um serviço que tem sido designado por Olímpico, com três classes (bronze, prata e ouro), enquanto a quarta classe (AF4) poderá ser usada para suportar um serviço idêntico ao melhor esforço (***best effort***).

2.1.1.3 Serviço olímpico

O serviço olímpico apresenta três categorias de serviços. As categorias ouro, prata e bronze. Em momentos de congestionamento, a categoria ouro irá obter uma maior parcela da banda passante, seguida pela categoria prata e bronze, nessa ordem.

A classe ouro deve garantir um determinado atraso, bem como a entrega dos pacotes e a conclusão do serviço. (e.g., vídeo sob demanda (VoD), operações bancárias e transações comerciais).

A classe prata não estabelece limites de atraso para o serviço, mas exige a garantia de sua entrega. O serviço deve ser utilizado em operações que não exijam brevidade nas transações, podendo ser concluídas inclusive, com usuários ***off-line***, mas garantindo a conclusão correta da operação.

A classe bronze apresenta descarte dos pacotes e elevado atraso quando comparado aos outros serviços.

2.2 Controle e prevenção de congestionamento

Os algoritmos de gestão e escalonamento em filas envolvem esquemas para medição de estado de utilização da fila, policiamento de tipos de tráfego e mecanismos de descarte de pacotes. Quando as filas são formadas, algoritmos de tratamento de filas determinam a ordem e a taxa em que o tráfego é retirado da fila para envio ao próximo dispositivo. Os roteadores devem suportar mecanismos de filas com priorização para permitir a implementação de mecanismos de QoS.

Nesse trabalho, o tipo de escalonamento escolhido para ser utilizado em conjunto com a arquitetura DiffServ foi o algoritmo WFQ, pois ele permite a implementação dinâmica da divisão da largura de banda para as filas dos nós roteadores de acordo com os seus pesos; esta disciplina de enfileiramento é capaz de prover QoS e de evitar a falta de recursos aos fluxos menos prioritários (condição de inanição).

2.2.1 WFQ - *Weighted Fair Queueing*

O algoritmo WFQ - ***Weighted Fair Queueing*** é uma implementação Cisco na qual é possível proporcionar, dinamicamente, a divisão da largura de banda para as filas, de acordo com os seus pesos. Cada fluxo, ou classe, é associado a um peso e a taxa do fluxo ou a classe do tráfego é servida proporcionalmente ao peso associado. O algoritmo escalona o tráfego prioritário para a frente da fila, reduzindo o tempo de resposta deste fluxo. Ao mesmo tempo, compartilha o restante da banda com os outros fluxos de menor prioridade, porém alocando uma largura de banda menor, já que os fluxos de menor prioridade têm também menor peso junto ao WFQ. O WFQ evita que as filas cheguem a uma situação de "inanição" (***starvation***) por falta de recursos, dando ao tráfego um serviço previsível.

Devido às características supracitadas, esse algoritmo de enfileiramento foi o escolhido para ser utilizado em conjunto com a arquitetura de serviços diferenciados, com o controlador proposto e com as políticas de controle especificadas na solução do provisionamento dinâmico de Qualidade de Serviço em redes IP apresentada neste trabalho.

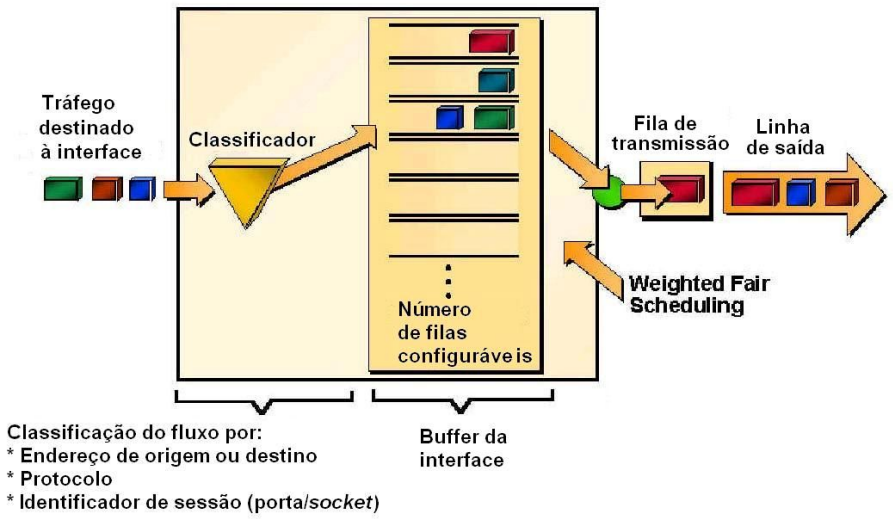

Figura 3 - Operação da Fila WFQ (SILVA, 2000)

A classificação dos fluxos de dados no algoritmo WFQ pode ser realizada de diversas formas, e.g. por endereço fonte ou destino, por protocolo, pelo campo precedência IP, pelo par porta/***socket***, etc. A quantidade de filas é configurável e a ponderação pode ser estabelecida por precedência IP, ou em conjunto com outros protocolos de QoS como o RSVP. Estas características do algoritmo WFQ podem ser observadas na Figura 3.

2.3 Parâmetros de avaliação de QoS

Disponibilizar QoS basicamente significa proporcionar garantias de transmissão para certos fluxos de dados. A garantia de transmissão pode ser expressa como a combinação de alguns dos parâmetros descritos a seguir.

2.3.1 Parâmetros quantitativos de QoS

Atraso fim-a-fim

O atraso fim-a-fim é o tempo necessário para um pacote percorrer a rede, medido do momento em que é transmitido pelo emissor até ser recebido pelo receptor. É composto por quatro tipos de atraso: de processamento, de enfileiramento, de transmissão e de propagação.

Variação do atraso (*jitter*)

É a variação no atraso fim-a-fim. Mesmo com níveis de retardo dentro dos limites aceitáveis, variações acentuadas do retardo podem ter efeitos negativos na qualidade do serviço oferecido a algumas aplicações (e.g. vídeo e voz).

Largura de banda

É a taxa de transmissão de dados máxima que pode ser sustentada entre dois pontos finais. Além dos limites físicos (tecnologia utilizada) a largura de banda é limitada também pela quantidade de fluxos que compartilham a utilização de determinados componentes da rede.

Taxa de perda de pacotes

A taxa de perda de pacotes é a razão entre o número de pacotes recebidos e o número de pacotes enviados em um certo intervalo de tempo, desta forma, esta taxa pode ser interpretada como o índice que mede o grau de sucesso na transmissão de pacotes IP entre a origem e o destino.

Taxa de erro

Porcentagem do número de pacotes enviados pela fonte, que são recebidos com erro no destino.

2.3.2 Utilização dos parâmetros quantitativos de QoS

Quando um cliente contrata um determinado serviço de uma operadora de telecomunicação, ou "***carrier***", um acordo entre as duas partes é firmado, sendo esse acordo conhecido como: SLA – ***Service Level Agreement***. De acordo com o ITU – ***International Telecommunication Union***, o SLA deve conter as seguintes informações (ITU-T Recommendation Y.1241, 2001):

a) Valores dos parâmetros que definem a classe de serviço contratada: vazão (***throughput***), atraso (***delay***), variação do atraso (***jitter***) e taxa de erro/perda de pacotes;

b) Disponibilidade e confiabilidade do serviço;

c) Método de monitoração dos parâmetros de classe de serviço e da disponibilidade do serviço;

d) Métodos de autenticação;

e) Compensações financeiras em caso de não cumprimento do SLA.

2.4 Técnicas de controle

As técnicas de controle podem ser classificadas como pertencentes a dois grandes grupos: as de controle clássico e as de controle moderno. Dentre os controladores clássicos mais conhecidos, podemos citar o ***on/off***, o tripartite Proporcional-Integrador-Derivativo (PID) e suas subdivisões P, PI e PD, os de avanço, atraso e avanço-atraso de fase. Alguns exemplos de técnicas modernas incluem os controles do tipo multivariável, adaptativo, ótimo, não-linear, preditivo, robusto e inteligente (OGATA, 1997).

A teoria de controle está cada vez mais integrada à área da computação. De forma crescente novos algoritmos e recursos computacionais estão sendo utilizados para realizar os mais diversos tipos de controle e podem ser encontrados em abundância em diversos setores da indústria (CARMO, 2006).

Apesar do grande avanço na teoria de controle, o controlador PID continua sendo largamente utilizado em malhas de controle industrial dados sua robustez e facilidade de implementação. Cerca de 95% dos controladores utilizados na industria possuem como estratégia algoritmos PID (CARMO, 2006).

A técnica de controle escolhida para ser utilizada neste trabalho será a dos controladores clássicos, mais especificamente os controladores PID.

2.4.1 Controladores PID

O controlador PID é um controle realimentado muito encontrado em sistemas de controle industriais. Ele compara um valor medido de um processo (PV, variável de processo) com um valor de referência (SP, ***set-point***). A diferença destes valores (erro) é usada para calcular um novo valor, desta vez para a variável manipulada, que levará o processo ao valor desejado, ou seja, para o ***set-point***.

O algoritmo do PID ajusta as saídas do processo baseada no histórico e na taxa de variação do erro do sinal, o que confere ao controlador mais precisão e estabilidade. Os controles PID podem ser mais facilmente ajustados, se comparados a outros algoritmos de controles mais sofisticados, como MPC (controladores preditivos multivariáveis).

2.4.2 Implementação digital de controladores PID

A implementação digital do controlador PID pode ser feita através de aproximações numéricas das derivadas e da integral que aparecem na lei de controle. Desta forma, é possível descrever cada uma das ações por uma equação de recorrência. As equações de recorrência descrevem as operações matemáticas a serem programadas em um microcontrolador ou em um microcomputador onde será implementado o PID digital.

Considere a equação do controlador PID ideal descrito no domínio do tempo (OGATA, 1997):

$$u(t) = K_c e(t) + \frac{K_c}{T_i}\int_0^t e(t)dt + K_c T_d \frac{de(t)}{dt} + u_0, \quad (1)$$

na qual K_c é o ganho crítico, T_i é o tempo integral, T_d é o tempo derivativo, T_s é o intervalo de amostragem e e(t) é o erro.

Para discretizar o controlador, precisa-se aproximar os termos de integral e derivada para uma forma que possa ser implementada em computadores digitais. De um ponto de vista puramente numérico pode-se usar:

$$\frac{d(e(t))}{dt} \approx \frac{e(t) - e(t-1)}{T} \quad \text{e} \quad \int_0^t e(t)dt \approx T_s \sum_{i=0}^{t} e(i). \quad (2)$$

Substituindo a Eq. (2) na Eq. (1), o algoritmo discretizado apresenta-se desta forma (COSTA e SOUZA, 2001):

$$u(t) = K_c e(t) + \frac{K_c}{T_i} T_s \sum_{i=0}^{t} e(i) + \frac{K_c T_d (e(t) - e(t-1))}{T_s} + u_0. \quad (3)$$

A Eq. (3) representa a resposta temporal do controlador PID. Esta forma particular é denominada de controlador PID posicional, porque o sinal de controle é calculado em referência a um nível base u_0.

Pode-se projetar um controlador PID discreto diretamente no domínio de Laplace. Temos então a equação

$$\frac{U(s)}{E(s)} = K_c\left[1 + \frac{1}{T_i s} + T_d s\right]. \quad (4)$$

Aplicando-se o método ***backward difference*** ou uma transformação bilinear para obter-se o controlador PID discreto equivalente, tem-se

$$\frac{U(s)}{E(s)} = K_c\left[1 + \frac{1}{T_i s} + T_d s\right] \longrightarrow u(t) = e(t)K_c\left[1 + \frac{T_s}{T_i(1-z^{-1})} + T_d\frac{(1-z^{-1})}{T_s}\right], \quad (5)$$

que, simplificando, obtém-se a seguinte equação de controle

$$u(t) = u(t-1) + K_c\left[e(t) - e(t-1)\right] + \frac{K_c T_s}{T_i} e(t) + \frac{K_c T_d}{T_s}\left[e(t) - 2e(t-1) + e(t-2)\right]. \quad (6)$$

Este controlador difere da estrutura do controlador obtido no domínio do tempo (Eq. (3)) e é conhecido como algoritmo PID de velocidade.

Ao contrário da referência fixa utilizada no algoritmo posicional, no algoritmo de velocidade o cálculo da variável de controle usa seus valores prévios como referência. Essencialmente o controle é calculado como uma mudança, daí o termo velocidade.

2.4.3 Algoritmos PID posicional e de velocidade

Embora a estrutura dos algoritmos PID posicional e de velocidade pareçam muito diferentes, elas estão, de fato, relacionadas. Considere a seguinte equação do algoritmo posicional

$$\boldsymbol{u(t) = K_c e(t) + \frac{K_c}{T_i} T_s \sum_{i=0}^{t} e(i) + \frac{K_c T_d (e(t) - e(t-1))}{T_s} + u_0,} \tag{3}$$

na qual K_c é o ganho crítico, T_i é o tempo integral, T_d é o tempo derivativo, T_s é o intervalo de amostragem e e(t) representa o erro. Aplicando um deslocamento regressivo no intervalo de amostragem da Eq. (3), vem:

$$\boldsymbol{u(t-1) = K_c e(t-1) + \frac{K_c}{T_i} T_s \sum_{i=0}^{t-1} e(i) + \frac{K_c T_d (e(t-1) - e(t-2))}{T_s} + u_0.} \tag{7}$$

Subtraindo (7) de (3), obtém-se a forma da velocidade, também conhecida como forma recursiva (COSTA e SOUZA, 2001):

$$\boldsymbol{u(t) = u(t-1) + K_c \left[e(t) - e(t-1)\right] + \frac{K_c T_s}{T_i} e(t) + \frac{K_c T_d}{T_s} \left[e(t) - 2e(t-1) + e(t-2)\right].} \tag{8}$$

2.4.4 Parametrização de controladores PID

Ziegler e Nichols propuseram métodos heurísticos para a determinação dos valores do ganho proporcional K_p, do tempo integral T_i e do tempo derivativo T_d, baseando-se nas características da resposta transitória de uma planta (OGATA, 1997).

As regras de Ziegler-Nichols tem sido amplamente utilizadas para determinar parâmetros de controladores PID em sistemas de controle de processos em que a dinâmica da planta não é precisamente conhecida. Estas regras podem, naturalmente, ser aplicadas a plantas cujas dinâmicas são conhecidas.

Geralmente, para plantas com dinâmica complicadas, mas sem integradores, as regras de determinação de parâmetros de Ziegler-Nichols podem ser aplicadas. No entanto, se a planta tiver um integrador, estas regras podem não ser aplicadas (OGATA, 1997).

Neste trabalho, foi utilizado o segundo método de Ziegler-Nichols, o qual é baseado no ganho crítico em malha fechada. De acordo com este método, estabeleceu-se inicialmente $T_i = \infty$ e $T_d = 0$. Usando a ação de controle proporcional, aumenta-se K_p desde zero até um valor crítico (K_c), no qual a saída exibe oscilações mantidas. Se a saída não exibir oscilações mantidas para quaisquer valores que K_p possa assumir, então este método não se aplica. Portanto, o ganho crítico K_c e o período correspondente P_c são experimentalmente determinados.

Ziegler-Nichols sugerem estabelecer os valores dos parâmetros K_p, T_i e T_d de acordo com a fórmula mostrada na Tabela 1 (AV221, 2006; OGATA, 1997).

Tabela I: Parâmetros K_p, T_i e T_d

Controlador	K_p	T_i	T_d
P	$0,5*K_c$	-	-
PD	$0,65*K_c$	-	$0,12* P_c$
PI	$0,45*K_c$	$(1/1,2)* P_c$	-
PID	$0,6* K_c$	$0,5* P_c$	$0,125* P_c$

2.5 Resumo

Foi apresentada, neste capítulo, uma introdução aos conceitos teóricos utilizados neste trabalho. Realizou-se inicialmente, uma introdução sobre serviços diferenciados (DiffServ). Mostrou-se a razão da escolha do algoritmo WFQ e foram expostas as teorias de controle necessárias ao desenvolvimento desta dissertação.

Capítulo 3 – Estratégia de Controle - Metodologia e Cenários

Nesse capítulo, apresenta-se a metodologia para a definição do controlador proposto, a ser utilizado no provisionamento de recursos de rede, em um domínio DiffServ. A metodologia tem como ponto de partida a análise dos resultados de controladores clássicos PID, P, PI e PD, utilizando as equações posicional e de velocidade, atuando no controle do atraso fim-a-fim do fluxo de maior prioridade na rede, em um cenário configurado de acordo com a topologia dumbbell (existe um único gargalo compartilhado pelos fluxos).

3.1 Escolha do simulador: NS-2 versus OPNET

O OPNET (OPNET, 2008) é um simulador comercial largamente utilizado no âmbito corporativo, principalmente em grandes empresas e operadoras de telecomunicações. Atualmente está disponibilizada uma versão acadêmica livre (IT ***Guru Academic Edition***), que apresenta todas as facilidades e potencialidades deste simulador. A edição acadêmica está limitada por um número máximo de eventos de simulação (50 milhões); também há um limite no número de roteadores na rede (máximo de vinte nodos com mais de duas ligações conectadas, mas nenhum limite em "dispositivos de borda" (clientes/servidores)). Há também um limite nos tipos de protocolos apoiados (por exemplo, não dá suporte a MPLS). O problema desta versão livre é que não se pode estender o código, i.e. não é possível desenvolver novas funcionalidades, adequando o ambiente simulado às necessidades específicas do problema analisado.

O ***network simulator*** (ns-2) teve origem no simulador REAL (NS-2, 2008) e, atualmente, apresenta-se como uma das mais poderosas ferramentas de análise, reconhecida internacionalmente e muito utilizada no meio acadêmico e nas maiores companhias de redes de informação. O ns-2 é utilizado principalmente por pesquisadores, por ter distribuição gratuita e código aberto. Tal fato o torna adequado a situações nas quais é necessário desenvolver novas funcionalidades, como em teses e projetos de pesquisa aplicada. No entanto, a sua interface (textual) não é amigável ao usuário. A execução de um experimento de simulação no ns-2 requer a elaboração de ***scripts*** em Tcl e grande trabalho adicional para obter e visualizar os resultados. Também é comum o usuário recorrer à programação em C++ para que as funcionalidades desejadas estejam disponíveis. Além disso, os protocolos e tecnologias no ns-2 em geral são desenvolvidos para uso isolado, para resolução de problemas específicos. Integrá-los normalmente é uma tarefa árdua, resultando na especificação complexa da rede a simular.

Por apresentar distribuição gratuita e ter grande aceitação na comunidade científica, o simulador escolhido para este trabalho foi o ns-2. Como a distribuição oficial do ns-2 não apresenta suporte a disciplinas de enfileiramento do tipo WFQ, utilizou-se adicionalmente uma extensão do simulador desenvolvida por Sérgio Andreozzi (ANDREOZZI, 2001).

3.2 Desenvolvimento dos cenários iniciais

Os cenários iniciais de simulação foram desenvolvidos com base na arquitetura de serviços diferenciados e a topologia escolhida para ser utilizada é a topologia dumbbell, i.e., existe um único gargalo compartilhado pelos fluxos (enlace ***core*** – ***e2***), conforme a Figura 4. Os fluxos são gerados por servidores de fluxos de voz, vídeo, FTP e de fluxo interferente. Os nós ***e1*** e ***e2*** são os roteadores de bordas da nuvem DiffServ e o nó ***core*** representa o roteador de núcleo. As capacidades de transmissão dos enlaces entre os nós da rede são de 10 Mbps, com exceção do enlace entre o roteador ***core*** e ***e2***, cuja capacidade foi estabelecida em 1,5 Mbps para provocar um ponto de congestionamento (gargalo) na rede. O atraso de cada enlace é de 5 ms.

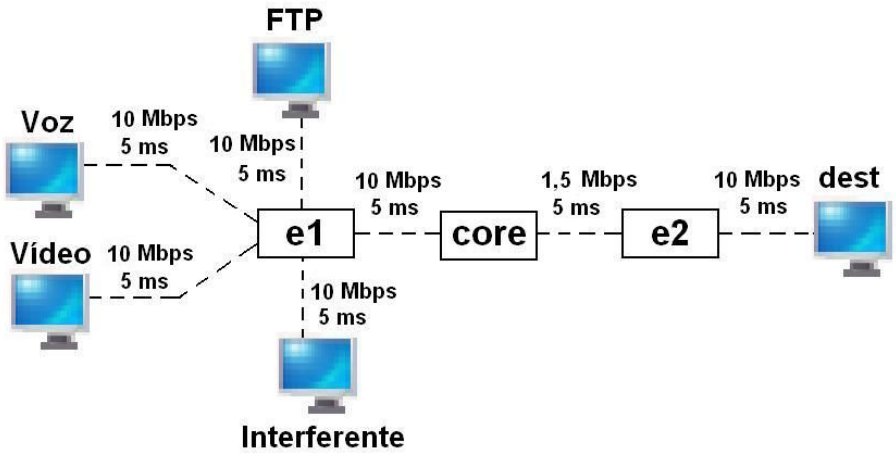

Figura 4: Cenário simples (topologia ***dumbbell***)

Os fluxos de voz, vídeo e interferente são do tipo CBR (***Constant Bit Rate***), com suas taxas variando de modo a criar situações distintas de congestionamento, para a análise da influência do congestionamento e/ou módulo de controle nos cenários desenvolvidos.

3.2.1 Sintonia dos controladores

Com a finalidade de se efetuar a sintonia dos controladores implementados, utilizou-se o segundo método de Ziegler-Nichols.

A rede foi configurada com taxa do tráfego interferente de 1 Mbps, taxa do fluxo de vídeo de 1 Mbps e um fluxo de FTP estabelecido com transferência de arquivos de 3 MB. A taxa do fluxo de voz iniciou com 200 Kbps e no tempo de simulação de 10 s elevou-se para 400 Kbps. A atuação do controlador P ***posicional*** iniciou-se em 20 s.

Os pesos dos fluxos foram assim estabelecidos: peso do fluxo de vídeo igual a 10, peso do fluxo de FTP igual a 20 e peso do fluxo interferente igual a 20. No código do controlador P ***posicional,*** ficou estabelecido um valor mínimo do peso referente ao fluxo de voz igual a 1 e um valor máximo do peso referente ao fluxo de voz de 200, de modo a acelerar a convergência das oscilações mantidas no sinal de saída (atraso-voz). O controlador atuou somente no fluxo de voz.

As oscilações mantidas foram obtidas para o ganho de 0,08, conforme a Figura 5. Sendo então determinado o ganho crítico (K_c = 0,08) e o período das oscilações (P_c = 1,62 segundos). A partir de K_c e P_c, foram obtidos os parâmetros K_p, T_i e T_d, de acordo com a Tabela I, necessários ao desenvolvimento dos controladores analisados na subseção seguinte.

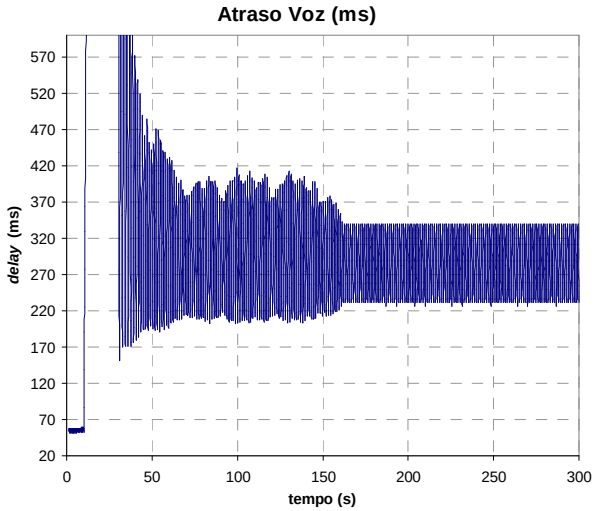

Figura 5: Oscilações mantidas

3.2.2 Análise dos controladores

Foram desenvolvidos controladores P, PI e PID a partir das Equações 3 e 8, respectivamente as equações posicional e de velocidade, sendo que cada controlador atua somente no fluxo de voz. No código dos controladores, ficou estabelecido um valor mínimo do peso referente ao fluxo de voz igual a 1(um) e um valor máximo do peso referente ao fluxo de voz de 40. Os cenários desenvolvidos apresentam as mesmas características básicas, mudando apenas o tipo de controle para o fluxo de voz.

A rede foi configurada com taxa do tráfego interferente de 1 Mbps, taxa do fluxo de vídeo de 1 Mbps, 10 fluxos FTP sendo estabelecidos pseudo-aleatoriamente segundo a distribuição uniforme no intervalo de 0 a 5 segundos. Os pesos dos fluxos foram assim estabelecidos: peso do fluxo de vídeo igual a 40, peso do fluxo de FTP igual a 20 e peso do fluxo interferente igual a 10.

As prioridades de descarte de pacotes nas filas dos roteadores foram configuradas de modo que o fluxo de voz tivesse a menor prioridade de descarte, seguido pelos fluxos de vídeo, FTP e interferente, de acordo com os valores listados abaixo. Estas são as configurações utilizadas em todas as simulações deste trabalho.

i) Voz: prioridade de descarte igual a 0,04;

ii) Vídeo: prioridade de descarte igual a 0,06;

iii) FTP: prioridade de descarte igual a 0,1;

iv) Fluxo interferente: prioridade de descarte igual a 0,5.

A taxa do fluxo de voz iniciou com 200 kbps e obedeceu à dinâmica descrita na Tabela II. A duração de todas as simulações foi de 280 s.

Tabela II: Dinâmica do fluxo de voz

Intervalo de simulação (s)	Taxa fluxo voz
0 – 50	200 kbps
50 – 100	250 kbps
100 – 150	200 kbps
150 – 200	80 kbps
200 – 280	250 kbps

O controle do fluxo de voz inicia-se sempre no tempo de simulação de 10 s. O período amostral utilizado pelos controladores simulados foi de 60 ms.

De acordo com a recomendação do ITU-T (G.114) atrasos até 150 ms são aceitáveis, proporcionando boa interatividade. Atrasos entre 150 ms e 400 ms são aceitáveis, com alguma perda de interatividade. O limite máximo de 400 ms deve ser respeitado. Quanto ao limite mínimo, em uma situação de congestionamento da rede um limite mínimo deve ser estabelecido. Como para o desenvolvimento deste trabalho, tem-se a necessidade de estabelecimento de limites mínimos para o atraso e ***jitter*** das aplicações de voz e vídeo, utilizou-se a Tabela III obtida de (Guido 2004).

O valor de referência (***set-point***) estabelecido para o atraso do fluxo de voz foi de 200 ms, valor condizente com a Tabela III, sendo maior que o limiar mínimo (100 ms), mas menor que o limiar máximo (400 ms).

Tabela III: Parâmetros de QoS (GUIDO, 2004)

Parâmetro		Serviço de Voz	Serviço de Dados	Serviço de Áudio e Imagem
Atraso de transmissão dos pacotes	Min.	100 ms	1.000 ms	100 ms
	Max.	400 ms	4.000 ms	400 ms
Variação do atraso dos pacotes	Min.	1 ms	10 ms	1 ms
	Max.	100 ms	100 ms	100 ms
Vazão	Min.	10 kbps	1 kbps	10 kbps
	Max.	64k bps	10 kbps	128 kbps
Taxa de perda de pacotes	Min.	10^{-4}	10^{-6}	10^{-4}
	Max.	10^{-3}	10^{-4}	10^{-3}

Os valores de vazão apresentados na Tabela III são valores referentes a fluxos de aplicações individuais. Os valores de vazão utilizados em todas as simulações deste trabalho são valores referentes a agregados de fluxos.

Os cenários desenvolvidos a seguir tem como objetivo a analise do comportamento dos controladores implementados, com a rede estando em condição de congestionamento, para verificar se eles conseguem manter o atraso do fluxo controlado (fluxo de voz) na referência estabelecida de 200 ms.

3.2.2.1 Simulação: controlador P *posicional*

O comportamento do parâmetro atraso-voz está exposto na Figura 6, onde se observa que para uma taxa de 80 kbps do fluxo de voz (intervalo de 150 a 200 s), o atraso mantém-se muito baixo, mesmo com peso de voz na faixa de 3,2, conforme ilustrado na Figura 7. Isto ocorre porque o fluxo de voz foi configurado com menor prioridade de descarte nos nós da arquitetura Diffserv. O controlador não consegue atingir e estabilizar o valor do atraso de acordo com a referência (200 ms). Nos intervalos de 0 a 50 s e de 100 a 150 s o valor do atraso do fluxo de voz se mantém em 120 ms e somente nos intervalos de 50 a 100 s e 220 a 280 s o controlador consegue manter o valor do atraso do fluxo de voz na referência.

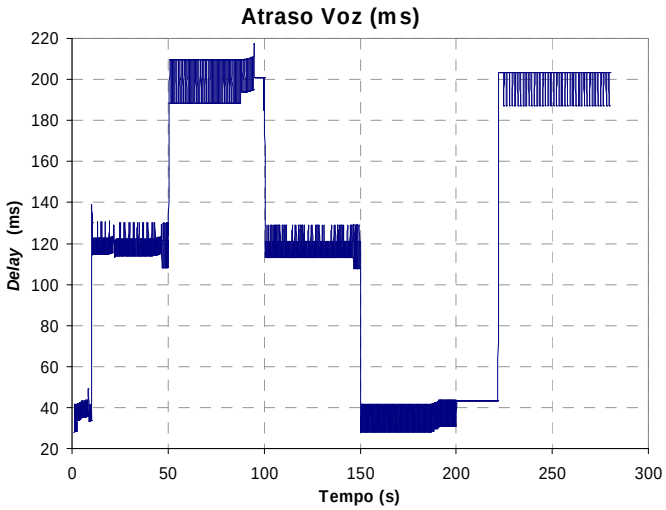

Figura 6: Controlador P ***posicional*** – atraso-voz

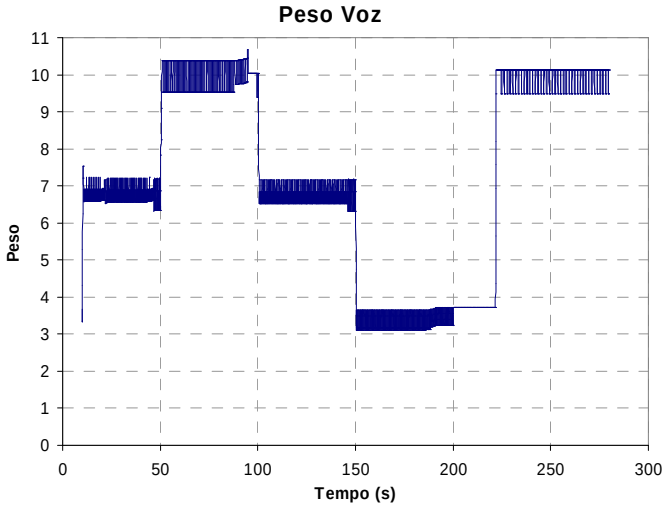

Figura 7: Controlador P ***posicional*** – peso-voz

3.2.2.2 Simulação: controlador P *velocidade*

O controlador P ***velocidade*** apresenta características semelhantes ao controlador P ***posicional***, não conseguindo atingir e manter o valor do atraso do fluxo de voz na referência (200 ms). Este controlador apresenta uma sobrelevação maior na busca pelo valor de referência, conforme se pode observar na Figura 8, nos instantes de 10 s e 200 s. A Figura 9 mostra a dinâmica da variável de controle peso-voz, ou seja, seu comportamento na tentativa de manter o atraso do fluxo de voz no valor de referência. Quando a taxa do fluxo de voz diminui, que acontece nos instantes de 100 s e 150 s, a ação do controlador é de diminuir o valor da variável de controle peso-voz. Com o aumento da taxa do fluxo de voz, que acontece nos instantes de 50 s e 200 s, a ação do controlador é de elevar o valor da variável de controle peso-voz, na tentativa de manter a variável controlada (atraso-voz) na referência.

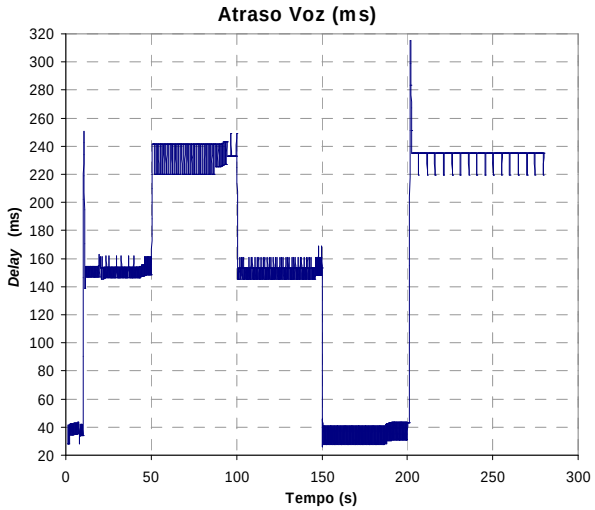

Figura 8: Controlador P ***velocidade*** – atraso-voz

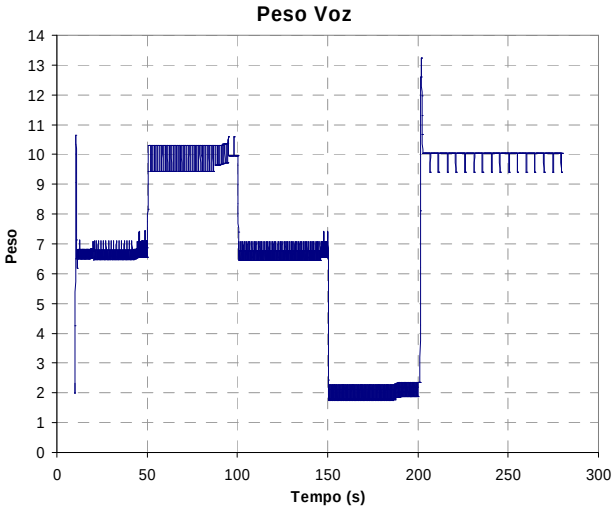

Figura 9: Controlador P ***velocidade*** – peso-voz

3.2.2.3 Simulação: controlador PI *velocidade*

Apesar do controlador PI ***velocidade*** conseguir manter e estabilizar o atraso do fluxo de voz na referência (200 ms), ele apresenta muita perturbação no intervalo de 150 a 200 s, em que o atraso tende a se manter muito abaixo da referência, como se pode observar na Figura 10. A ação integradora apresenta a característica de eliminar o erro em regime permanente, ou seja, deixar o valor da variável controlada (atraso-voz) na referência estabelecida. No intervalo de 150 a 200 s, o atraso do fluxo de voz tende a se manter abaixo da referência, devido a características da planta, menor prioridade de descarte de pacotes do fluxo de voz nas filas dos roteadores e maior prioridade no algoritmo WFQ em relação aos demais fluxos. Logo a ação integradora faz com que a variável de controle peso-voz apresente variações bruscas, na tentativa de corrigir o erro em regime permanente. Na Figura 11 observa-se, durante o intervalo de 150 a 200 s, as variações bruscas na variável peso-voz.

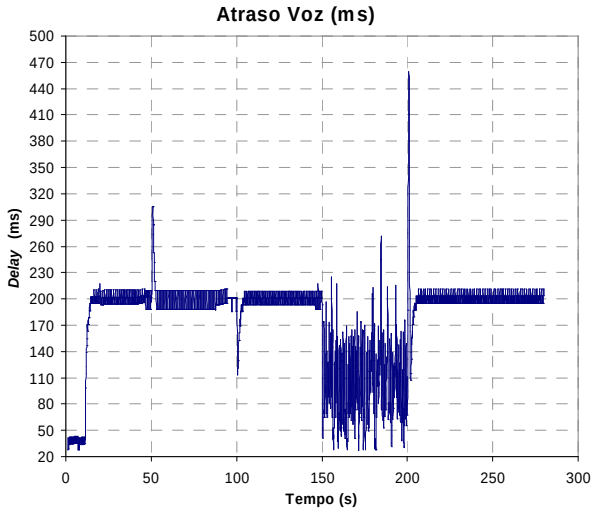

Figura 10: Controlador PI velocidade – atraso-voz

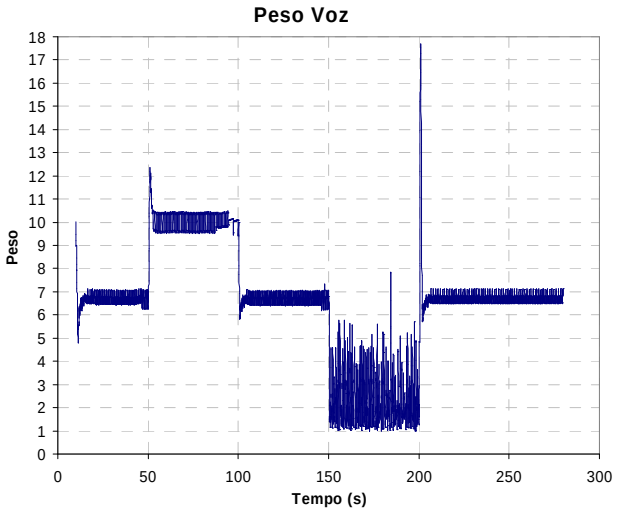

Figura 11: Controlador PI velocidade – peso-voz

3.2.3 Análise das simulações preliminares

De acordo com os resultados obtidos nas simulações anteriores, os melhores resultados obtidos nas simulações preliminares, as quais dão subsídios à proposta apresentada na Subseção 3.2.4, foram com os controladores P ***posicional*** e PI ***velocidade***.

O controlador P ***velocidade*** apresentou características semelhantes ao controlador P ***posicional***, com desnível no valor de referência estabelecido, mas com uma sobrelevação maior na busca pelo valor de referência. O controlador PI ***velocidade*** conseguiu manter e estabilizar o atraso na referência, mas apresentou muita perturbação na faixa em que o atraso do fluxo de voz tende a se manter muito abaixo da referência.

Os controladores PI ***posicional*** e PID ***posicional*** e ***velocidade*** não apresentaram resultados satisfatórios, gerando grande instabilidade e variações bruscas na variável controlada (atraso do fluxo de voz).

Nos controladores PI ***posicional*** e PID ***posicional***, tem-se o problema devido a ***wind-up*** da ação integral. Quando o valor da variável de controle atinge o limite máximo (ou mínimo) do atuador, ocorre a saturação do sinal de controle. Este fato faz com que a malha de realimentação seja de certa forma quebrada, pois o atuador permanecerá no seu limite máximo (ou mínimo) independentemente da saída do processo. Entretanto, se um controlador com ação integral é utilizado, o erro continuará a ser integrado e o termo integral tende a se tornar muito grande. Neste caso, para que o controlador volte a trabalhar na região linear (saia da saturação) é necessário que o termo integral se "descarregue". Para que isso aconteça, deve-se esperar que o sinal de erro troque de sinal, sendo necessário aplicar na entrada do controlador, um sinal de erro de sinal oposto, mas esta ação pode durar por um longo período de tempo. A consequência disto é que a resposta transitória do sistema tenderá a ficar lenta e oscilatória. Nos controladores PID ***posicional*** e ***velocidade***, a ação derivativa torna a resposta do sistema instável (BAZANELLA e JÚNIOR, 2000).

3.2.4 Resultados preliminares – proposta do controlador

A partir da análise dos resultados preliminares das simulações na Subseção 3.2.3, propõe-se um controlador, denominado de controlador PI_v - P_p, que, nos instantes iniciais da busca ao valor de referência se comporte como um controlador do tipo P ***posicional*** e, ao atingir uma certa estabilidade passe a se comportar como um controlador do tipo PI velocidade. Vale ressaltar que na faixa em que o atraso da variável controlada se mantém muito abaixo do valor de referência, por características da planta, seria desejável que o controlador se comportasse como um controlador P ***posicional***.

Para determinar a faixa de atuação dos controladores P e PI, utilizou-se a seguinte metodologia: quando o valor da diferenças entre os atrasos do fluxo de voz amostrados nos instantes ***t*** e ***t-1*** estiver na faixa de estabilização do sinal de ±10% da referência (φ) e atraso > 50% da referência (evitar faixa em que o atraso se mantém muito abaixo da referência - φ), de modo que o erro (atraso voz – referência) deverá ser maior que -50% da referência, o controlador atuante será o controlador ***PI velocidade***. Nos demais casos, o controlador atuante será o controlador ***P posicional***. Essas regras para a atuação do controlador híbrido PI_v - P_p proposto estão refletidas na Eq. (9):

$$\boldsymbol{peso(t)} = \begin{cases} \boldsymbol{peso}(t-1) + 0{,}45 * \boldsymbol{K_c}\left[\boldsymbol{e(t)} - \boldsymbol{e}(t-1)\right] + \dfrac{\boldsymbol{K_c T_s}}{(1/1{,}2) * \boldsymbol{P_c}} \boldsymbol{e(t)}, \\ \text{se } \left|\text{atraso(t)} - \text{atraso(t-1)}\right| < 10\%.\varphi \text{ e } \text{e(t)} > -50\%.\varphi, \\ 0{,}5 * \boldsymbol{K_c e(t)} + \boldsymbol{peso}(0), \text{caso contrário.} \end{cases} \tag{9}$$

O controlador PI_v - P_p foi desenvolvido no NS-2 e os resultados obtidos nas simulações com o controlador atuando no fluxo de voz são apresentados nas Figuras 12, 13 e 14.

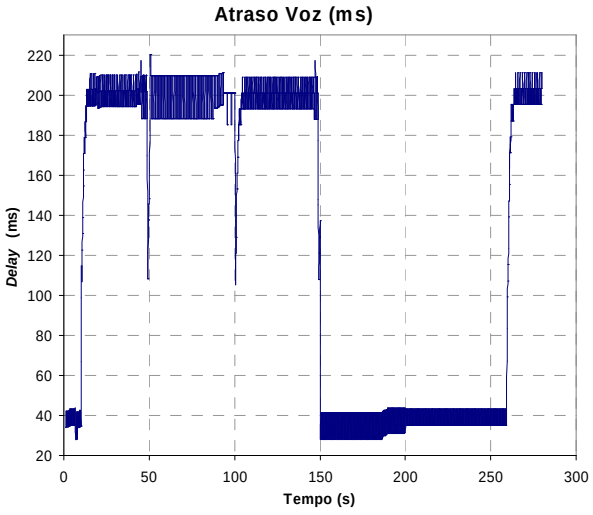

Figura 12: Controlador PI_v - P_p – atraso-voz

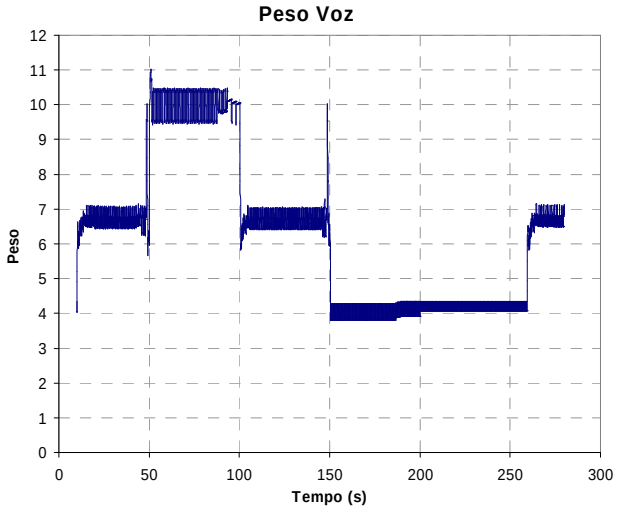

Figura 13: Controlador PI_v - P_p – peso-voz

O controlador PI_v - P_p consegue manter e estabilizar o atraso da aplicação de voz com o mínimo de sobrelevação, garantindo o nível de QoS desejado para este parâmetro, como se pode observar na Figura 12. Os valores da variação do atraso observados na Figura 14 apresentam-se dentro da faixa de 1 a 100 ms, que é faixa em que a QoS dos fluxos de voz para este parâmetro está assegurada, de acordo com os valores de referência da Tabela III. Logo o nível de QoS para a variação de atraso também foi garantido pela ação do controlador. Na Figura 13, tem-se o comportamento da variável de controle peso-voz.

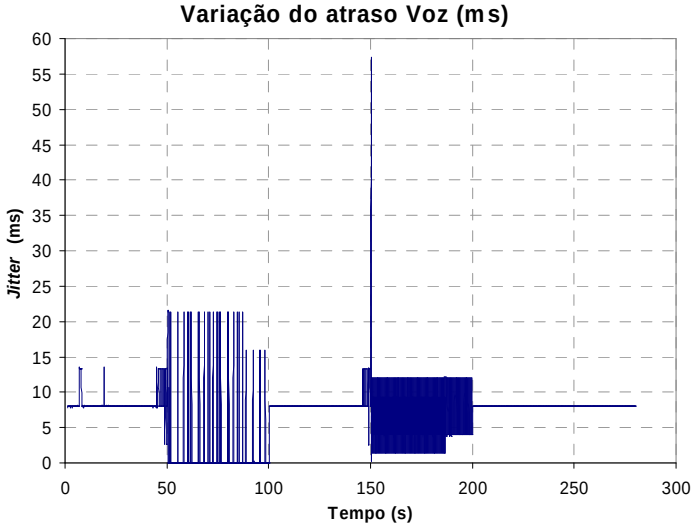

Figura 14: Controlador PI_v - P_p – ***jitter***-voz

3.3 Resumo

Apresentou-se, neste capítulo, a metodologia de construção de um controlador para o provisionamento de recursos de rede, com o objetivo de obter melhor QoS em um domínio DiffServ. A análise dos resultados da atuação dos controladores clássicos PID, P, PI e PD, utilizando as equações posicional e de velocidade, nos deram os subsídios necessários à proposta de um controlador híbrido, denominado de PI_v - $P_{p,}$ que nos testes preliminares conseguiu manter os níveis de QoS para os parâmetros de atraso e variação do atraso dentro dos limites normatizados, garantindo o nível de QoS para o fluxo de voz.

Capítulo 4 – Proposta - Políticas de Controle, Resultados e Discussões

A partir das simulações prévias e definições apresentadas no capítulo anterior, descreve-se nesse capítulo a construção do mecanismo para provisionamento de QoS em ambiente DiffServ. O objetivo é validar a metodologia proposta, desde a especificação das políticas definidas para o domínio até a definição do controlador apropriado para diversas topologias e padrões de tráfego.

A proposta desse trabalho consiste em um novo paradigma no controle de QoS em redes IP, com o controle rígido do atraso das aplicações mais prioritárias através do controlador desenvolvido. O objetivo não é somente minimizar este atraso, mas estabelecer um valor máximo aceitável em caso de congestionamento da rede. Esse valor máximo é configurado de acordo com as políticas estabelecidas.

4.1 Definição das políticas de controle

Com o objetivo de testar o desempenho do mecanismo proposto em condições diversas de congestionamento da rede, foram estabelecidas políticas a serem aplicadas aos cenários analisados. O fluxo de voz foi definido como o fluxo de aplicação crítica, ou seja, o fluxo mais prioritário. O valor de referência para o atraso desejado para este fluxo deve ser previamente determinado dentro da faixa de atraso mínimo e máximo, conforme a Tabela III (i.e. entre 100 ms e 400 ms).

A determinação do valor de referência do atraso escolhido para o fluxo de aplicação crítica (fluxo de voz) dependerá de decisões administrativas. Por exemplo, aplicações de telemedicina poderão ser agregadas nesta classe mais prioritária, na qual será estabelecido um controle rígido da QoS.

O fluxo de vídeo foi definido como o segundo fluxo mais prioritário. O valor de referência para o atraso deste fluxo foi relacionado com as características de descarte do fluxo menos prioritário (fluxo interferente) de acordo com as políticas estabelecidas para as simulações, que estão descritas na Tabela IV.

Tabela IV – Políticas dos valores de referência para o fluxo de vídeo

Percentual descarte (d) – fluxo interferente	Atraso – fluxo de vídeo
d ≤ 10%	100 ms
10% < d ≤ 20%	150 ms
20% < d ≤ 30%	200 ms
30% < d ≤ 40%	250 ms
40% < d ≤ 50%	300 ms
d > 50%	350 ms

4.2 Resultados – cenário simples

Os cenários de simulação foram desenvolvidos com base na arquitetura de serviços diferenciados e a topologia escolhida para ser utilizada é a topologia dumbbell, conforme a Figura 4 da seção 3.2. O desempenho do mecanismo proposto (controlador PI_v - P_p e políticas) é analisado nas subseções a seguir (simulações α.1 e α.2), através da atuação nos fluxos de voz e vídeo conforme as políticas estabelecidas na Tabela IV. O valor de referência escolhido para fluxo de voz foi de 200 ms, o qual está condizente com a Tabela III – maior que o limiar mínimo (100 ms) e abaixo do limiar máximo (400 ms).

4.2.1 Simulação α1

A rede foi configurada com a taxa de tráfego interferente de 500 kbps e 6 fluxos do tipo FTP com transferência de arquivos com 3 MB por fluxo, totalizando 18 MB. Os fluxos FTP foram iniciados pseudo-aleatoriamente com uma distribuição uniforme no intervalo de 45 a 50 segundos, sendo seu peso configurado com o valor de 20 e do fluxo interferente em 10.

A fim de se evitar problemas de sincronização, o controle do fluxo de voz inicia no instante de simulação igual a 2 segundos e o controle do fluxo de vídeo é disparado 8 segundos depois, i.e., em 10 segundos.

As taxas dos fluxos de voz e vídeo foram configuradas inicialmente com os valores 200 kbps e 1 Mbps, respectivamente, e foram modificadas segundo a dinâmica descrita na Tabela V.

Tabela V – Dinâmica dos fluxos de voz e vídeo.

Intervalo de simulação (s)	Taxa fluxo voz	Taxa fluxo vídeo
0 – 100	200 kbps	1 Mbps
100 – 150	200 kbps	900 kbps
150 – 200	200 kbps	1 Mbps
200 – 250	250 kbps	1 Mbps
250 – 300	200 kbps	1 Mbps
300 – 400	150 kbps	1 Mbps

Os resultados das simulações mostram que o mecanismo consegue manter o atraso do fluxo de voz no valor de referência de 200 ms, conforme se pode observar na Figura 15. No tempo de simulação $\boldsymbol{t}$ = 300 s, o atraso do fluxo de voz se mantém abaixo da referência estabelecida, devido à diminuição de sua vazão para 150 kbps (vide Tabela V) e a prioridade estabelecida tanto na arquitetura DiffServ quanto na ação do mecanismo. Os picos observados no atraso do fluxo de voz são devidos à ação do mecanismo respondendo às mudanças na dinâmica dos fluxos de voz e vídeo. Entretanto, estes picos estão dentro do limite máximo para fluxos de voz, conforme a Tabela III.

A dinâmica da variável de controle (peso-voz) variando de forma a manter o valor do atraso de acordo com a referência estabelecida (200 ms) pode ser observado na Figura 16. Quando o atraso do fluxo de voz tende a se manter abaixo da referência, a ação do mecanismo é de diminuir o valor do peso deste fluxo, de modo a permitir a transferência de recursos da rede para os demais fluxos menos prioritários.

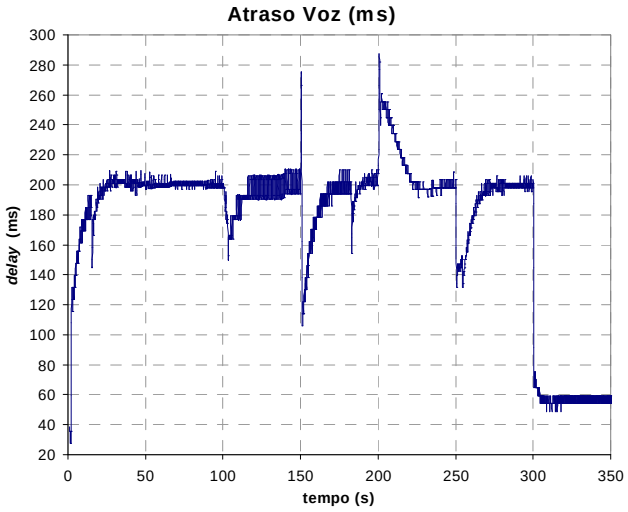

Figura 15. Simulação α.1 – atraso-voz.

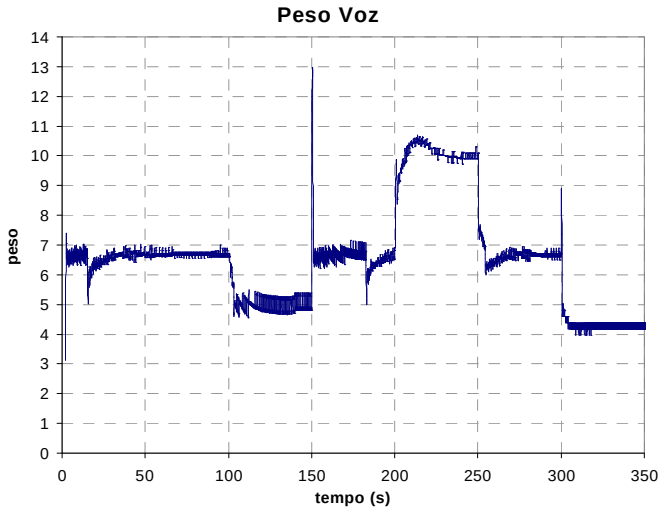

Figura 16. Simulação α.1 – peso-voz.

De acordo com a Figura 17, no instante t = 100 s, quando a taxa do fluxo de vídeo diminui para 900 kbps (vide Tabela V), o valor do atraso do fluxo de vídeo se mantém abaixo da referência. O percentual de descarte do fluxo interferente se mantém entre 30% e 40% fazendo com que o valor de referência do atraso do fluxo de vídeo tenda a se manter na maior parte da simulação em 250 ms, de acordo com as políticas definidas na Tabela IV.

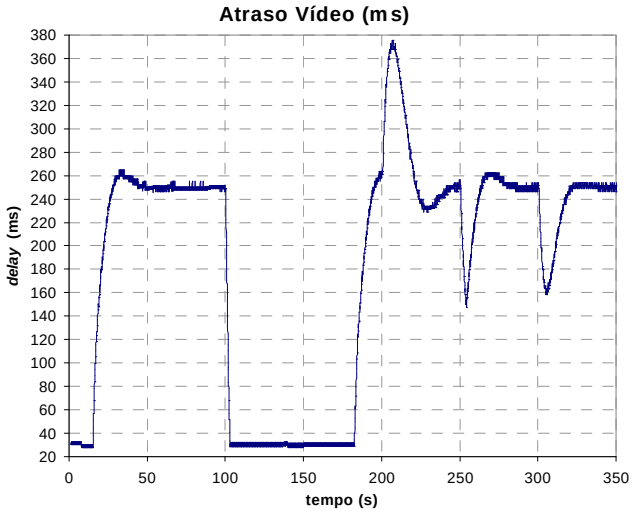

Figura 17. Simulação α.1 – atraso-vídeo.

Ocorre uma sobrelevação no atraso do fluxo de vídeo que chega a 370 ms, por volta dos 200 s de simulação (Figura 17), com uma duração longa, mais este valor está abaixo do limite máximo para fluxos de vídeo que é de 400 ms, conforme a Tabela III. Com a diminuição do fluxo de voz nos tempos de simulação de 250 s e 300 s, o atraso do fluxo de vídeo tende a diminuir e a ação do mecanismo é manter o atraso do fluxo de vídeo na referência estabelecida pelas políticas apresentadas na Tabela IV. Como é necessário menos recurso da rede para manter o atraso do fluxo de vídeo na referência com a diminuição do fluxo de voz, a variável de controle peso vídeo é decrementada, liberando os recursos excedentes para os outros fluxos.

Na Figura 18 observa-se em ***t*** = 100 s a ação do mecanismo minimizando o valor da variável de controle peso-vídeo, transferindo os recursos excedentes para os demais fluxos. Em ***t*** = 200 s, a taxa do fluxo de voz aumenta para 250 kbps e o atraso do fluxo de vídeo tende a aumentar muito; então, para estabilizar o atraso do fluxo de vídeo, o valor da variável de controle peso-vídeo apresenta um rápido e acentuado incremento.

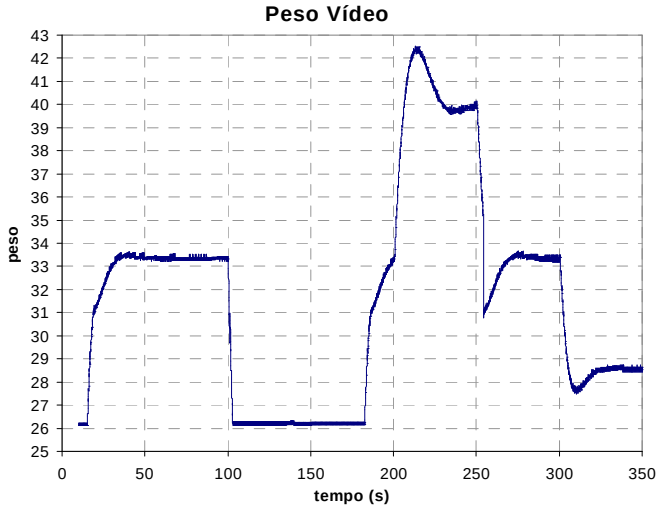

Figura 18. Simulação α.1 – peso-vídeo.

As funções de distribuição cumulativa de probabilidade do ***jitter*** para os fluxos de voz e vídeo são mostradas respectivamente nas Figuras 19 e 20. Verifica-se a eficiência do mecanismo proposto em manter a variação do atraso dos fluxos de voz e vídeo em valores muito abaixo do limite máximo de 100 ms, conforme a Tabela III. Em particular, para o fluxo de voz, praticamente 100% do ***jitter*** encontra-se abaixo de 10 ms (vide Figura 19). De modo semelhante, para o fluxo de vídeo, aproximadamente 100% do ***jitter*** encontra-se abaixo de 4 ms (vide Figura 20).

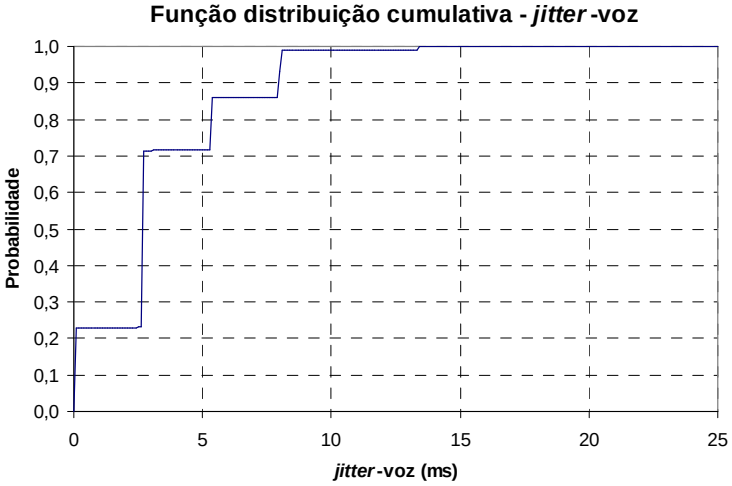

Figura 19. Simulação α.1 – ***jitter***-voz.

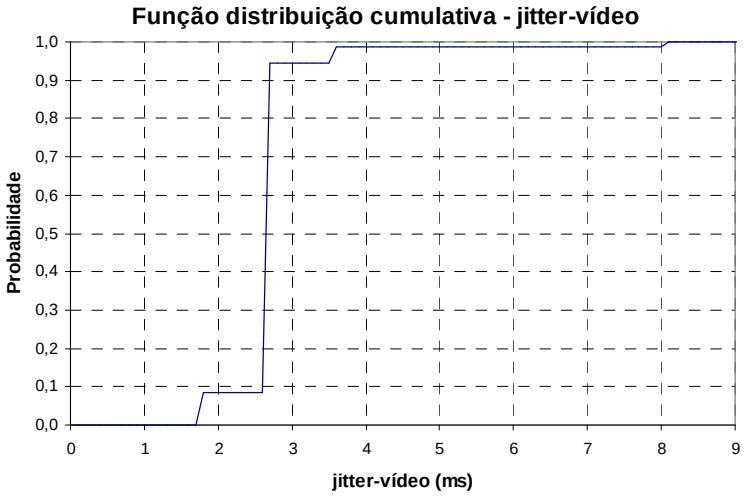

Figura 20. Simulação α.1 – ***jitter***-vídeo.

O fluxo FTP consegue atingir uma taxa de pico de vazão acima de 700 kbps. Como o fluxo FTP não apresentou perda de pacotes e não ocorreram retransmissões, as rajadas observadas na Figura 21, indicam a transferência efetiva de dados pelo fluxo FTP.

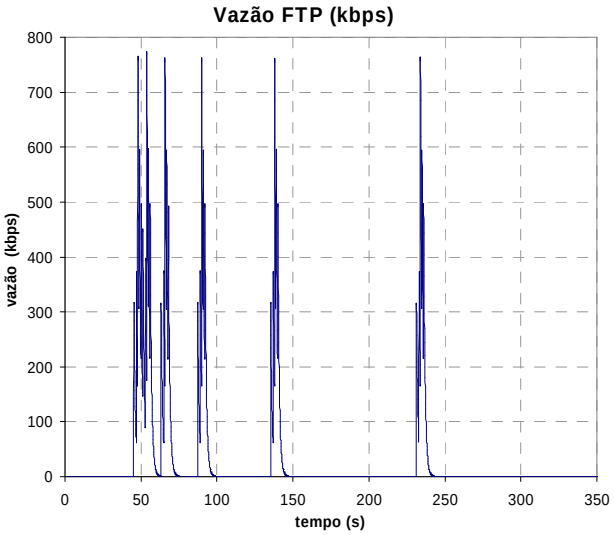

Figura 21. Simulação α.1 – vazão FTP.

Os fluxos de voz e de vídeo não apresentam perda de pacotes e suas taxas de vazão não sofrem depreciação, apesar da rede estar congestionada. Deste modo, o mecanismo proposto assegura largura de banda para estes fluxos. A taxa de perda de pacotes do fluxo interferente é de 35,91%. A taxa de perda de pacotes referente ao enlace congestionado é de 10,67%. Essas informações foram obtidas através de funções desenvolvidas no NS-2, que facilitam a análise estatística do comportamento dos fluxos nos nós e enlaces da rede.

4.2.2 Simulação α2

Essa simulação apresenta as mesmas configurações da simulação α.1 (subseção precedente), sendo que a taxa do fluxo de voz se mantém em 200 kbps e a taxa do fluxo de vídeo se mantém em 1 Mbps. O fluxo interferente inicia com uma taxa de 500 kbps, aumentando para 900 kbps no instante de simulação igual a 200 s.

Nessa simulação, observa-se com maior clareza a influência das políticas adotadas relacionando os fluxos de vídeo e interferente. O valor de atraso do fluxo de vídeo de 250 ms está relacionado a uma taxa de descarte do fluxo interferente entre 20% e 30%, conforme estabelecido na Tabela IV. No instante t = 200 s, a taxa do fluxo interferente aumenta para 900 kbps, fazendo ocorrer um aumento na taxa de descarte do fluxo interferente. Quando o valor da taxa de descarte do fluxo interferente passa para a faixa entre 30% e 40%, o valor do atraso do fluxo de vídeo muda para 300 ms. Com o aumento da taxa de descarte do fluxo interferente para valores acima de 40%, o valor do atraso do fluxo de vídeo passa para 350 ms. A dinâmica da variação do valor do atraso do fluxo de vídeo pode ser observada na Figura 22.

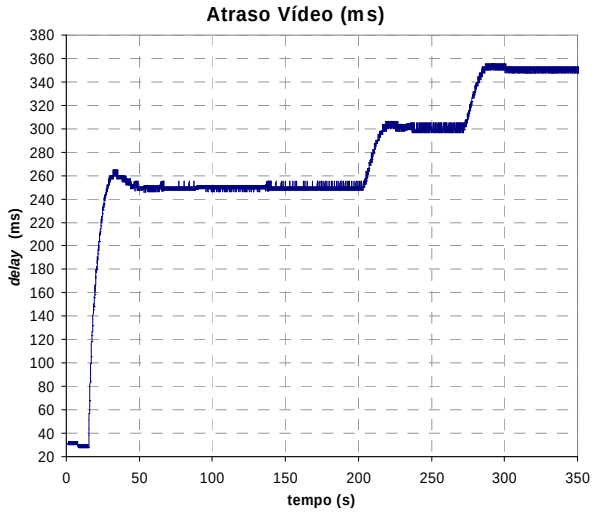

Figura 22. Simulação α.2 – atraso-vídeo.

4.3 Resultados – cenário complexo

Para avaliar o fator escalabilidade do mecanismo proposto, com vistas a um cenário real de rede, utilizou-se uma topologia com uma grande quantidade de nós. Preferiu-se usar topologias aleatórias para possibilitar a generalização da metodologia proposta. A topologia foi criada com o pacote `gt-itm`, incluído na distribuição do ns-2. O diagrama genérico da topologia complexa utilizada é mostrado na Figura 23. Trata-se de um domínio DiffServ constituído por 5 nós de borda e 100 nós no núcleo. Definiu-se, na borda, 4 nós como entrada, com fontes de tráfego de voz, vídeo FTP e interferente e 1 nó de saída.

Todos os enlaces da rede apresentam uma capacidade de 5 Mbps, exceto o enlace entre o nó de borda e a estação cliente, que é de 1,5 Mbps, com a finalidade de gerar uma condição de congestionamento na saída do domínio DiffServ. Em virtude da grande quantidade de nós, o atraso de cada enlace foi configurado em 1 ms.

Para todas as simulações realizadas, foram efetuados 20 (vinte) repetições e os resultados expostos nos gráficos apresentam um nível de confiança de 95%.

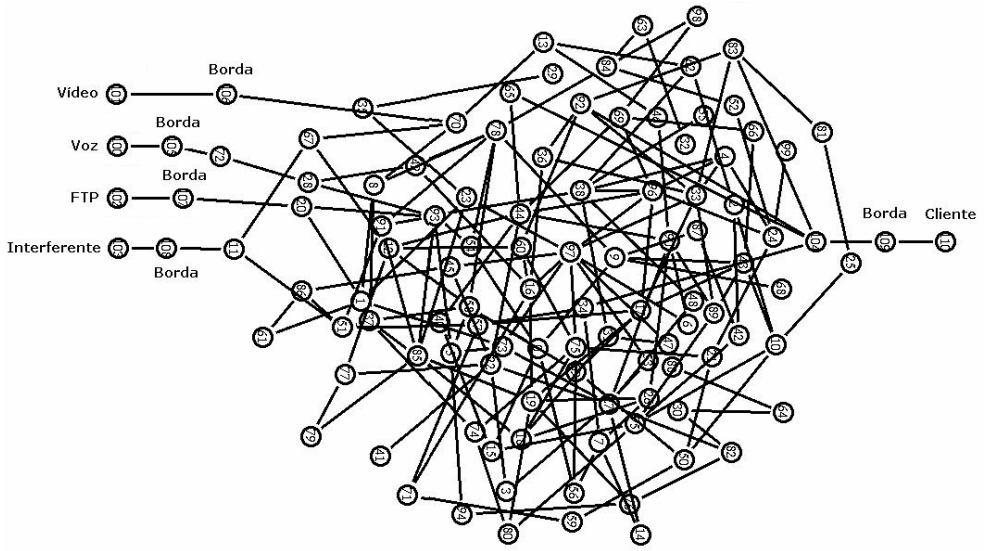

Figura 23. Cenário complexo.

4.3.1 Tipo de tráfego

A aplicação de voz foi implementada com tráfego CBR (C***onstant Bit Rate***), que tem comportamento determinístico e exige mais banda da rede. Embora o tráfego ***on-off*** com distribuição exponencial seja mais próximo de uma conversação normal, escolheu-se para a aplicação de voz o tráfego CBR, que exige mais banda passante, para tornar as condições da rede mais severas para a análise do comportamento do mecanismo. As aplicações de vídeo e interferente também são do tipo CBR/UDP.

O tráfego de fundo da rede, ou seja, entre os 100 (cem) nós que compõem o núcleo da rede, foi configurado de modo a ter 300 fontes de tráfego do tipo CBR/UDP, com os respectivos nós de saída e destino sendo escolhidos aleatoriamente segundo a distribuição uniforme. A taxa de cada fluxo é escolhida aleatoriamente no intervalo de 50 kbps a 100 kbps. A duração de cada fluxo é escolhida segundo a distribuição uniforme no intervalo de 10 s até o tempo final da simulação. Esta configuração de tráfego na rede não tem a pretensão de modelar um tráfego real, mas de proporcionar condição severas e aleatórias nos nós da rede, para os fluxos de voz, vídeo e FTP que irão atravessar o domínio DiffServ.

O tráfego interferente apresenta uma taxa de 1 Mbps, que é o dobro da taxa utilizada na simulação α_1 do cenário simples. A fonte de tráfego FTP apresenta 6 fluxos, com transferência de arquivos de 3 MB por fluxo, totalizando 18 MB. Os fluxos FTP foram iniciados pseudo-aleatoriamente com uma distribuição uniforme no intervalo de 45 a 50 segundos.

As taxas dos fluxos de voz e vídeo foram configuradas inicialmente com os valores 200 kbps e 1 Mbps, respectivamente, e obedeceram à dinâmica descrita na Tabela IV, para facilitar a análise comparativa entre os cenários simples e complexo.

O ponto de referência para os atrasos dos fluxos de voz e vídeo é de 200 ms, para verificar se será garantido um bom índice de QoS aos fluxos controlados, mesmo em condições extremas da rede (vide Tabela III).

Estas configurações de tráfego serão utilizadas em todas as simulações desenvolvidas a seguir.

4.3.2 Análise dos resultados – cenário complexo

Os resultados das simulações mostram que o mecanismo proposto consegue manter o atraso dos fluxos de voz e de vídeo em valores abaixo ou no nível de referência máxima estabelecida (200 ms), de acordo com as Figura 24 e 25, respectivamente. As sobrelevações observadas, não chegam a comprometer o nível de QoS dos fluxos de voz e vídeo, pois se encontram dentro do limite aceitável, ou seja abaixo de 400 ms (vide Tabela III).

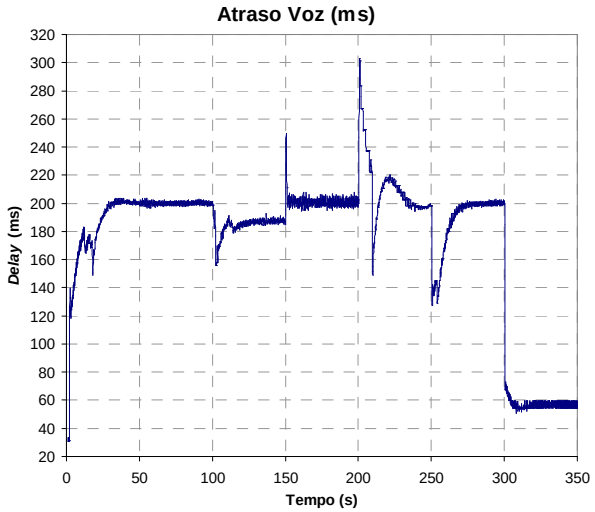

Figura 24. Cenário complexo – atraso-voz.

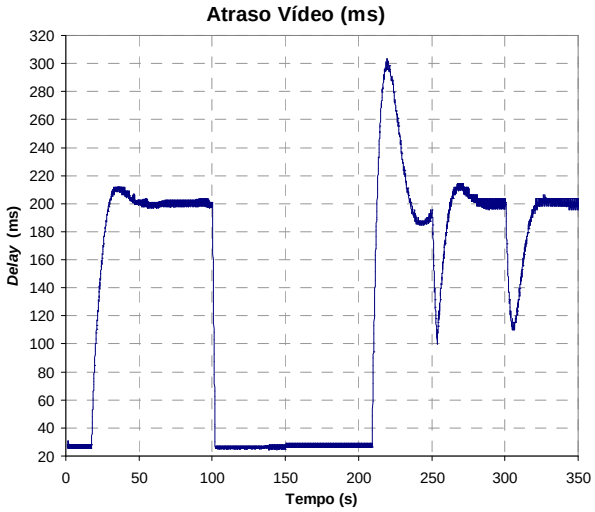

Figura 25. Cenário complexo – atraso-vídeo.

As funções de distribuição cumulativa de probabilidade do ***jitter*** para os fluxos de voz e vídeo são mostradas respectivamente nas Figuras 26 e 27. Para o fluxo de voz, praticamente 100% do ***jitter*** encontra-se abaixo de 10 ms (vide Figura 26) e aproximadamente 100% do ***jitter*** do fluxo de vídeo encontra-se abaixo de 8 ms (vide Figura 27). Logo, conclui-se que o mecanismo proposto consegue manter a variação do atraso dos fluxos de voz e vídeo em valores muito abaixo do limite máximo de 100 ms (vide Tabela III), garantido o nível de QoS dos fluxos controlados, para este parâmetro.

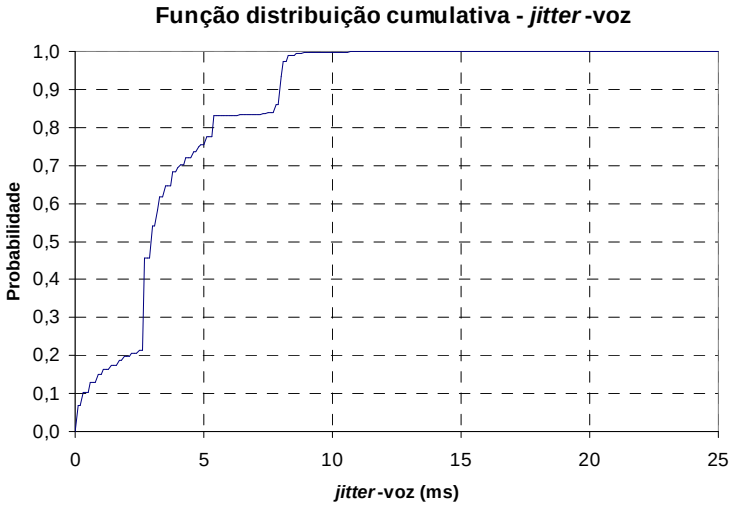

Figura 26. Cenário complexo – ***jitter***-voz.

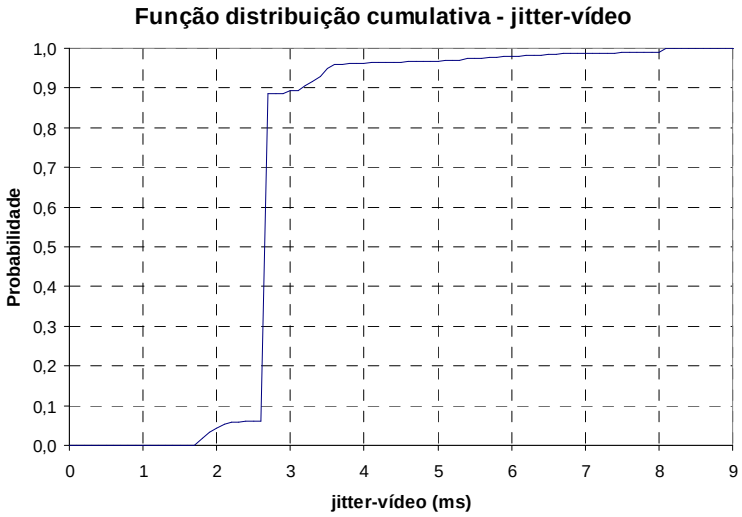

Figura 27. Cenário complexo – ***jitter***-vídeo.

Na Figura 28, observa-se que o fluxo FTP consegue atingir uma taxa de pico de vazão de 400 kbps. Com as informações referente à perda de pacotes, que são obtidas com o uso de funções desenvolvidas no NS-2, verifica-se que os fluxos de voz e de vídeo não apresentam perda de pacotes e suas taxas de vazão não sofrem depreciação no enlace congestionado. O mecanismo proposto assegura largura de banda para os fluxos controlados (voz e vídeo). A taxa média de perda de pacotes do fluxo interferente foi de 67,99%. A taxa média de perda de pacotes referente ao enlace congestionado foi de 31,16%.

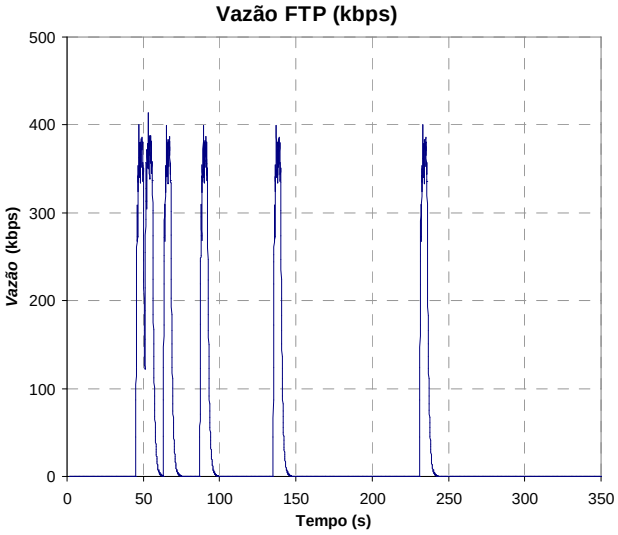

Figura 28. Cenário complexo – vazão-FTP.

4.3.3 Gargalos adicionais na rede - simulação γ_1

Com a finalidade de se analisar o comportamento do mecanismo em mais de um ponto de congestionamento severo da rede, foram realizadas simulações escolhendo pontos aleatórios no núcleo rede, na periferia e no centro, modificando as capacidades dos enlaces de 5 Mbps para 1,5 Mbps. Quando os pontos de congestionamento não faziam parte da rota dos fluxos controlados, ou seja, fluxo de voz e vídeo, os resultados obtidos foram similares aos apresentados na subseção anterior (4.4.2).

Os resultados mais expressivos ocorreram quando o gargalo adicional foi colocado em um trecho central que servia de passagem ao fluxo de vídeo (`n(12) - n(44)`), que é o segundo fluxo mais prioritário e apresenta uma taxa de vazão entre 1 Mbps e 900 kbps.

Os valores do ***jitter*** do fluxo de voz e vídeo apresentaram um pequeno incremento em relação aos resultados anteriores expostos nas Figuras 26 e 27. O ***jitter*** do fluxo de voz se manteve abaixo de 20 ms, enquanto que o ***jitter*** do fluxo de vídeo se manteve abaixo de 10 ms, conforme se pode constatar nas Figuras 29 e 30, respectivamente. Como os valores do ***jitter*** dos fluxos de voz e vídeo ainda se mantiveram muito abaixo da referência máxima aceitável de 100 ms, conforme a Tabela III, o mecanismo garantiu os níveis de QoS das aplicações analisadas. Os valores observados de atraso do fluxo de voz e vídeo foram similares aos apresentados na Subseção 4.4.2.

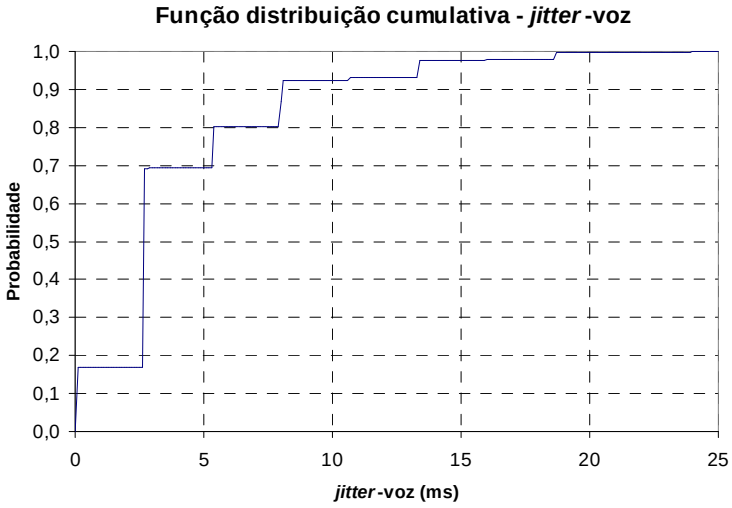

Figura 29. Simulação γ_1 – ***jitter***-voz.

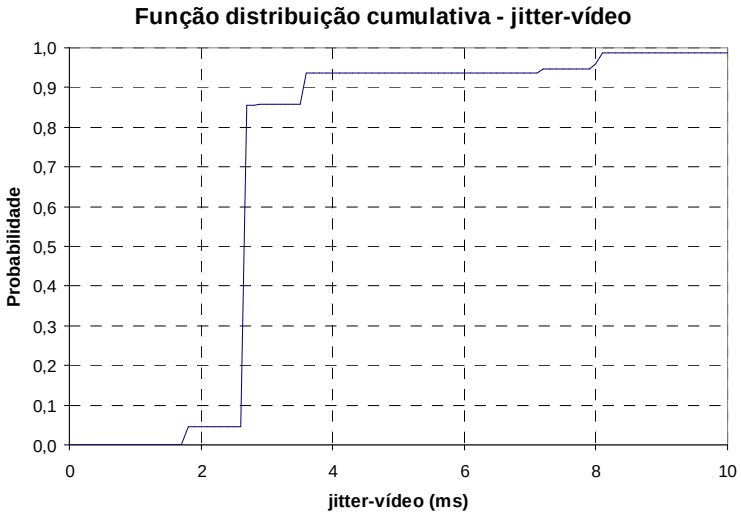

Figura 30. Simulação γ_1 – ***jitter***-vídeo.

4.3.3.1 Gargalos adicionais na rede - simulação γ_2

Com a finalidade de validar o mecanismo proposto (controlador PI_v - P_p e políticas) em condições mais adversas da rede e iniciar o estudo do impacto do fator escalabilidade, foi realizada uma análise do caminho fim-a-fim utilizado pelo fluxos de vídeo, através da ferramenta gráfica NAM do NS-2. Adicionou-se, sequencialmente, mais gargalos até que todos os enlaces localizados no núcleo da rede desse caminho tivessem sua capacidade reduzida para 1,5 Mbps.

Observa-se nas Figuras 31 e 32 que o sistema necessitou de aproximadamente 30 segundos para entrar em regime permanente e conseguir controlar o atraso dos fluxos de voz e vídeo. Este maior intervalo transitório de inicialização do sistema deve-se ao maior número de mecanismos presentes na rede.

Os resultados obtidos demonstram que o mecanismo conseguiu manter o nível de QoS das aplicações de voz e vídeo, apesar da saturação da rede e dos múltiplos pontos de congestionamento presentes. Os atrasos dos fluxos de voz e vídeo se mantiveram sempre abaixo ou na referência máxima estabelecida de 200 ms, como se pode constatar através dos resultados apresentados nas Figuras 31 e 32, respectivamente. Entre os instantes de 200 e 250 s, o atraso do fluxo de vídeo apresentou uma sobrelevação, o que pode ser averiguado na Figura 32, mas o nível de QoS desta aplicação não foi comprometido, pois os valores do atraso do fluxo de vídeo permaneceu abaixo do valor máximo normatizado de 400 ms (vide Tabela III).

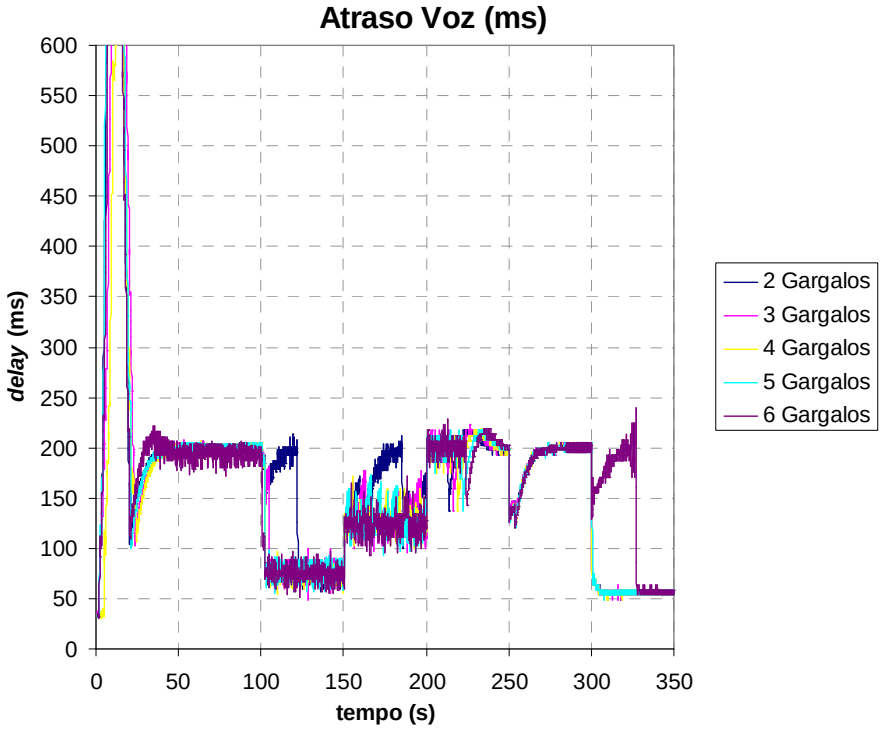

Figura 31. Simulação γ_2 – atraso-voz.

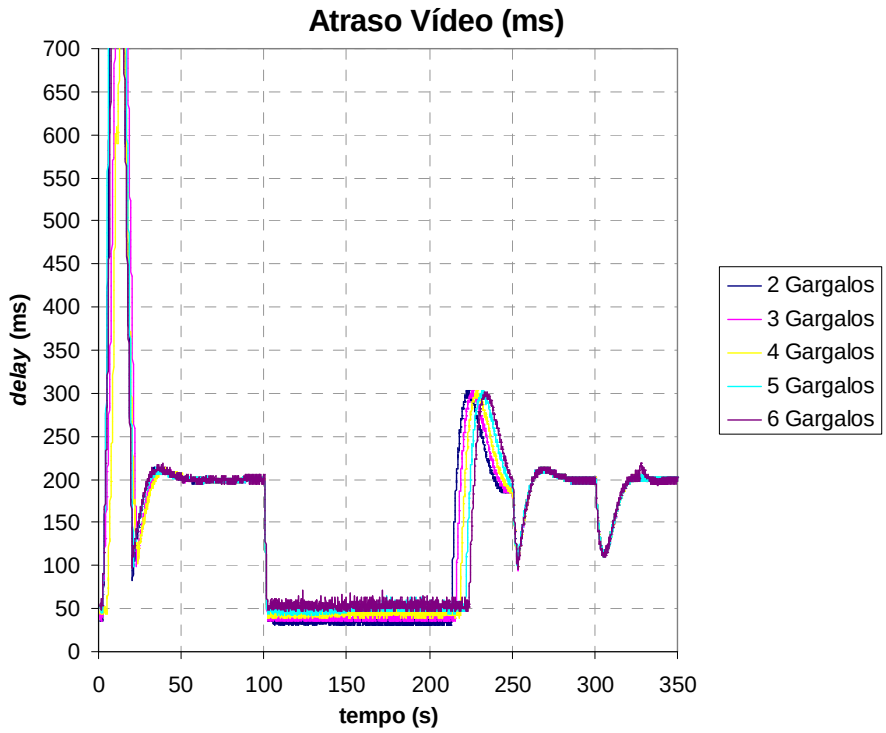

Figura 32. Simulação γ_2 – atraso-vídeo.

O mecanismo conseguiu manter os valores da variação do atraso do fluxo de voz sempre abaixo de 25 ms e para o fluxo de vídeo os valores da variação do atraso do fluxo de vídeo se mantiveram sempre abaixo de 10 ms. Estes valores de ***jitter*** estão bem abaixo do limite máximo normatizado de 100 ms (vide Tabela III) e foram obtidos em condições severas com múltiplos pontos de congestionamentos na rede.

Ocorreu uma pequena depreciação deste parâmetro de QoS, em relação à seção anterior quando a rede apresentava somente 2 (dois) pontos de gargalo, mas estes valores estão muito abaixo do valor máximo normatizado de 100 ms. Analisando os resultados apresentados nas Figuras 33 e 34, que ilustram, respectivamente, as funções de distribuição cumulativa da variação do atraso dos fluxos de voz e vídeo, verifica-se que ocorre uma leve depreciação desse parâmetro de QoS, na medida em que mais gargalos são adicionados na rede. Estes resultados demonstram a eficiência do mecanismo proposto no controle do atraso e variação do atraso em situações distintas de congestionamento da rede, mesmo com vários pontos de congestionamento simultâneos.

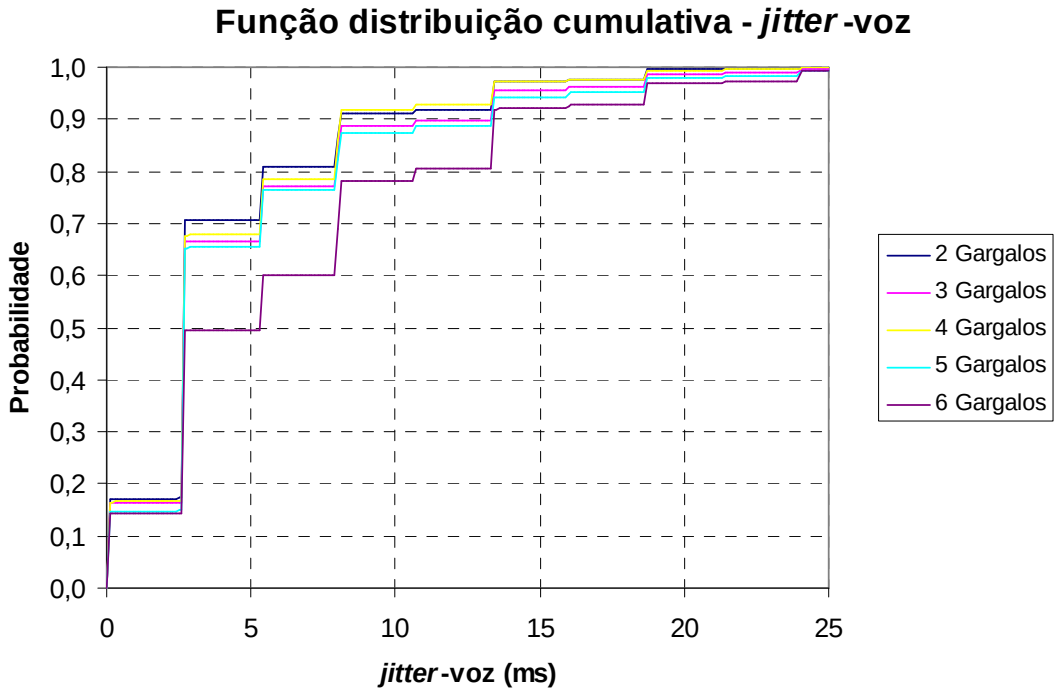

Figura 33. Simulação γ_2 – ***jitter***-voz.

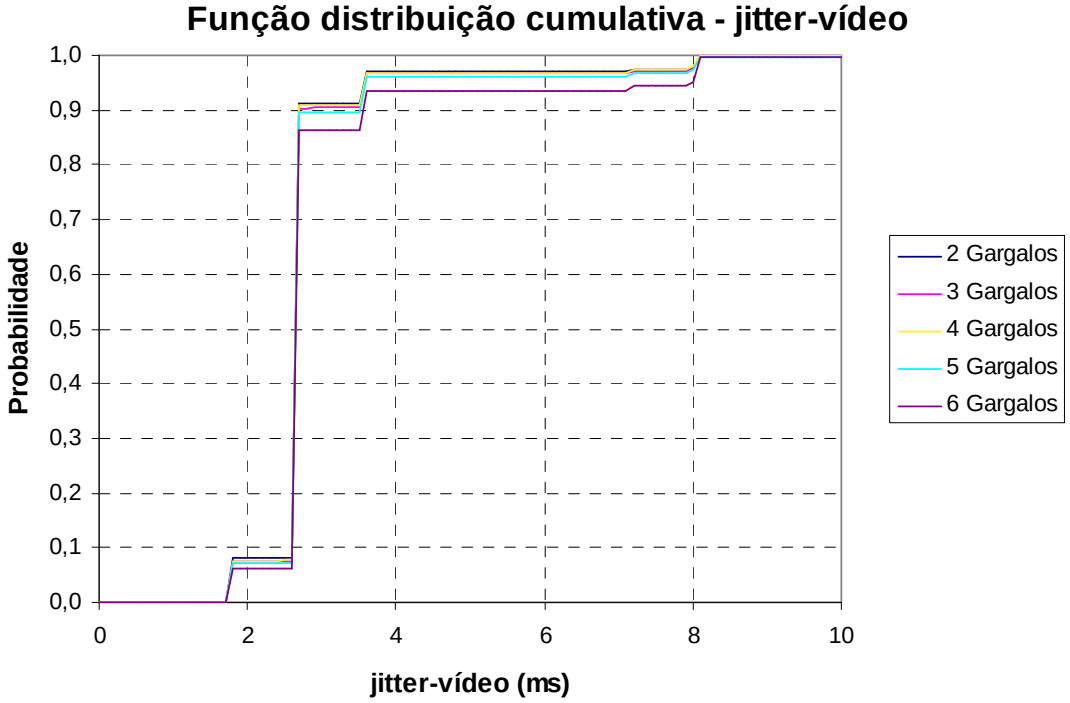

Figura 34. Simulação γ_2 – ***jitter***-vídeo.

4.4 Resumo

Foram apresentadas, neste capítulo, as topologias de simulação, os tipos de tráfego utilizados e os resultados das medidas de avaliação de desempenho, quais sejam, o atraso fim-a-fim e a variação do atraso fim-a-fim dos fluxos controlados (voz e vídeo), o descarte de pacotes dos fluxos controlados, do fluxo FTP e interferente, e a vazão do fluxo FTP. Mostrou-se, também, os resultados das simulações em uma topologia simples com apenas um ponto de congestionamento e em uma topologia complexa que apresentava um ou mais pontos de congestionamento.

A análise comparativa entre os resultados obtidos no cenário simples (simulação α.1) e no cenário complexo, que apresentam a mesma dinâmica dos fluxos de voz e vídeo, nos permite concluir que os níveis de QoS obtidos são mais sensíveis à variação dos fluxos mais prioritários do que à variação do tráfego interferente e do tráfego de fundo da rede.

O mecanismo (controlador PI_V - P_p e políticas) conseguiu garantir os níveis de QoS das aplicações de voz e vídeo, referente aos parâmetros de atraso e variação de atraso em todas as condições impostas nas diversas simulações realizadas, desde um cenário simples até em um cenário complexo apresentando múltiplos pontos de congestionamento simultâneos.

A seguir, serão expostas as considerações finais, bem como as sugestões para trabalhos futuros.

Capítulo 5 – Considerações Finais

Neste trabalho, foi proposto um mecanismo de controle clássico, não-linear, que melhora a provisão de QoS através de ações nas filas dos nós roteadores de um ambiente DiffServ bem definido. A ação do mecanismo de controle pode ser refinada com a adoção de políticas bem estabelecidas. A simplicidade analítica da Eq. (9), sua facilidade de implementação em código e sua baixa demanda de poder de processamento de máquina são pontos vantajosos. Como nas simulações realizadas, o mecanismo proposto (controlador PI_v - P_p e políticas) atua somente em pontos de congestionamento da rede, não é necessário que este esteja presente em todos os nós do núcleo da rede. Isto se constitui em uma vantagem, pois o mecanismo pode ser colocado somente nos pontos mais susceptíveis a congestionamento, ou seja, nos gargalos da rede, garantindo uma fácil integração a uma estrutura já existente que esteja fazendo uso da arquitetura de serviços diferenciados.

O diferencial da proposta consiste em manter o atraso dos fluxos controlados em um certo nível máximo, caso a rede se encontre congestionada, de modo a estabelecer um limite rígido de atraso fim-a-fim. Mantendo o atraso dos fluxos controlados em um certo nível máximo, é possível prover aos fluxos menos prioritários os recursos excedentes, melhorando assim a distribuição dos recursos da rede entre os fluxos IP, proporcionado mais eficiência no uso da rede. Mediante o ajuste do peso da disciplina de enfileiramento WFQ, evitou-se a posse excessiva de recursos da rede pelos fluxos mais prioritários, melhorando a QoS ofertada aos fluxos menos prioritários, diminuindo assim suas perdas de pacotes e melhorando suas métricas temporais (atraso e variação do atraso).

A ação do mecanismo consegue garantir os níveis de QoS dos fluxos controlados para os parâmetros de atraso, ***jitter***, perda de pacote e largura de banda, tanto em um cenário simples como em um cenário complexo. Além disso, os fluxos menos prioritários (FTP e interferente) não entram em condição de inanição, pois foi garantida largura de banda para estes fluxos, apesar do congestionamento da rede.

O mecanismo mostrou-se bastante versátil, adequando-se tanto às políticas estabelecidas na Tabela IV, quanto ao uso de políticas mais simples, como a adotada no cenário complexo, com o estabelecimento de valores fixos de referência para os fluxos controlados.

Os métodos de inteligência computacional podem ser empregados com sucesso no ajuste dos parâmetros dos controladores PID, sendo, portanto uma área bastante promissora para estudos futuros (CASTRO, 2001; COELHO e COELHO, 1999; KO et al., 2006; MOEDINGER e COELHO, 2002; RISSO, 2002). O uso de algoritmos adaptativos, e.g., controle ***gain-scheduling*** ou técnicas de aprendizado de máquina (***machine learning***) podem ser considerados na perspectiva de se efetuar o ajuste dinâmico e a sintonia fina dos parâmetros dos controladores PID, diminuindo as sobrelevações da variável controlada (atraso). Pode-se até mesmo vislumbrá-los como uma solução completa de gerência autonômica de uma rede IP. Em todo caso, pode-se afirmar que os resultados obtidos até aqui são bastante motivadores na medida em que apontam para a viabilidade do uso deste tipo de controladores para provimento de QoS de uma forma diferente e partitiva.

Na sequência, pretende-se aprofundar a análise do fator escalabilidade, analisar o comportamento do mecanismo com algumas das técnicas supracitadas, para efetuar a análise de desempenho, visando seu refinamento e otimização. Também será objeto de estudos futuros, a viabilidade de utilização do mecanismo em redes móveis. Os resultados obtidos até aqui revelaram-se bastante promissores, servindo de incentivo à sequência do trabalho ora apresentado.

Referências Bibliográficas

AGUILAR, Daniel Melchor; TORRES, Víctor Castillo. ***Stability Analysis of Proportional-Integral AQM controllers supporting TCP flows***. Revista Computación y Sistemas, 2007. http://www.cic.ipn.mx/revistas/pages/vol10-04/v10no4_Art05.pdf

ANDREOZZI, Sérgio. ***Diffserv Simulations Using the Network Simulator: Requirements, Issues And Solutions***. Università Degli Studi di Pisa, Facoltà di Ingegneria, Corso di Laurea In Ingegneria Informatica, 2001.

AVR221: ***Discrete PID controller***, 2006.
http://www.atmel.com/dyn/resources/prod_documents/doc2558.pdf.

BAZANELLA, Alexandre Sanfelice e JÚNIOR, João Manoel Gomes da Silva. **Ajuste de Controladores PID**. Apostila de Curso de Extensão – Universidade Federal do Rio Grande do Sul, Departamento de Engenharia Elétrica, 2000.
http://www.ece.ufrgs.br/~jmgomes/pid/Apostila/apostila/apostila.html.

CASTRO, Rodrigo Evangelista de. **Otimização de Estruturas com Multi-Objetivso via Algoritmos Genéticos**. Rio de Janeiro, R.J. – Brasil, 2001.

COELHO, Leandro dos Santos; COELHO, Antônio Augusto Rodrigues. **Algoritmos Evolutivos em Identificação e Controle de Processos: uma Visão Integrada e Perspectivas**. Departamento de Automação e Sistemas - Universidade Federal de Santa Catarina. Florianópolis - SC – Brasil. SBA Controles Automação Vol. 10 no. 01 / Jan., Fev. Mar, Abril de 1999.

COSTA Filho, J. T.; SOUZA, C. P. **Controle por Computador - Desenvolvendo Sistemas de Aquisição de Dados para PC**. 1. ed. São Luís: EDUFMA, 2001.

DRUMMOND, André Costa. **Alocação de banda passante em redes auto-ajustáveis**. Dissertação de Mestrado, IC/Unicamp, 2005.

FERNANDEZ, Marcial Porto. **Provisionamento de Recursos em Arquitetura Diffserv para Melhoria da Qualidade de Serviço (QoS)**. Tese de Doutorado. COPPE/UFRJ, D.Sc., Engenharia Elétrica, 2002.

FERNANDEZ, Marcial Porto; PEDROZA, Aloysio de Castro P.; REZENDE, José Ferreira de. ***Dynamic QoS Provisioning in DiffServ Domains Using Fuzzy Logic Controllers***. Telecommunication Systems 26:1, 9–32, 2004.

GALLARDO, José R.; MAKRAKIS, Dimitrios. ***Dynamic Predictive Weighted Fair Queueing for Differentiated Services***. Communications. 2001. ICC 2001. IEEE International Conference, 2001.

GUIDO, Marcelo. **Propostas para Estender as Funcionalidades do RSVP-TE**. Dissertação de Mestrado em Engenharia de Computação - Instituto de Pesquisas Tecnológicas do Estado de São Paulo, 2004.

ITU-T Recommendation Y.1241. ***Support of IP-Based Services Using IP Transfer Capabilities***, Março de 2001.

KO, Chia-Nan; LEE, Tsong-Li; FAN, Han-Tai e WU, Chia-Ju. ***Genetic Auto-Tuning and Rule Reduction of Fuzzy PID Controllers***. Systems. Man and Cybernetics. 2006. SMC '06. IEEE International Conference on Volume 2, 8-11 Oct. 2006 Page(s):1096 -1101.

LIMA, Michele Mara de Araújo Espindula. **Projeto de Controladores Ótimos para Gerenciamento Ativo de Filas**. Tese de Doutorado. Instituto de Computação. Universidade Estadual de Campinas, 2005.

CARMO, Marlon José do. **Ambiente Educacional Multifuncional Integrado para Sintonia e Avaliação do Desempenho de Malhas Industriais de Controle**. Dissertação de Mestrado em Engenharia Elétrica da Faculdade de Engenharia da Universidade Federal de Juiz de Fora – MG, Agosto de 2006.

MOEDINGER, Luis Henrique e COELHO, Leandro dos Santos. **Sintonia de Controlador com Escalonamento de Ganhos Baseada em Algoritmos Meméticos**. 1º Simpósio Sul-Brasileiro de Matemática e Informática. PUCPR / CCET / LAS / PPGEPS - Laboratório de Automação e Sistemas Centro de Ciências Exatas e de Tecnologia Pontifícia Universidade Católica do Paraná, 2002.

NS-2 - ***The Network Simulator***. http://www.isi.edu/nsnam/ns/.

OGATA, Katsuhiko. **Engenharia de Controle Moderno**. Prentice Hall do Brasil. 4ª Edição. Rio de Janeiro – RJ, 2003.

OPNET (*Optimised Network Engeneering Tools*). http://www.opnet.com.

PANZA, Gianmarco; GRAZIOLI, Matteo; SIDOTI, Filippo. ***Design and analysis of a dynamic Weighted Fair Queuing (WFQ) scheduler***. IST Mobile and Wireless Communication Summit '05, 2005.

PANZA, G., FRANCO, C.; LAMY-BERGOT, C., SIDOTI, F.. ***Weight updating methods for a dynamic Weighted Fair Queuing (WFQ) scheduler***. IST Mobile and Wireless Communication Summit ’06, 2006.

RISSO, Gerson. **Sintonia de Controladores PID por Algoritmos Genéticos**. São Paulo: EPUSP, 2002. 13 p. (Boletim Técnico da Escola Politécnica da USP, Departamento de Engenharia Mecânica, BT/PME/0219), 2002.

RFC 791. University of Southern Califórnia, Information Sciences Institute. ***Internet Protocol Darpa Internet Program Protocol Specification***, Setembro de 1981.

RFC 2460. Deering S., Hinden R. ***Internet Protocol, Version 6 (IPv6) Specification***, Dezembro de 1998. Acesso em: 10/02/2006.

RFC 2475. Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., Weiss, W. ***Architecture for Differentiated Services***, Dezembro de 1998.

RFC 2597. Heinanen, J., Baker, F., Weiss, W., Wroclawski, J. ***Assured Forwarding PHB Group***, Junho de 1999.

RFC 2598. Jacobson, V., Nichols, K., Poduri, K. ***An Expedited Forwarding PHB*** , Junho de 1999.

ROMÃO, Wesley; NIEDERAUER, Carlos A. P.; MARTINS, Alejandro; TCHOLAKIAN, Aran; PACHECO, Roberto C. S. **Algoritmos Genéticos e Conjuntos Difusos Aplicados ao Controle de um Processo Térmico**. Revista Tecnológica, n. 8, p. 7-21, 1999.

SILVA, Adailton J. S. **Qualidade de Serviço em VoIP - Parte I**. Rede Nacional de Ensino e Pesquisa (RNP). Boletim bimestral sobre tecnologia de redes. Maio de 2000. http://www.rnp.br/newsgen/0005/qos_voip1.html.

WANG, Lijun; FAYEK, Dalia; SIVANANTHAN, Thushyanth. ***A Bandwidth Bargain Model based on Adaptive Weighted Fair Queueing***. Network Qperations and Managemant Symposium. 2006. NOMS 2006. 10th IEEE/IFIP, 2006.

XU, Yue-Dong; YANG, Jie; DU, Qing. ***Nonlinear PI active queue management based on hyperbolic secant functions***. Machine Learning and Cybernetics. Proceedinqs of 2005 International Conference on Volume 2, 18-21. Aug. 2005. Page(s):715 - 720 Vol. 2, 2005.

Apêndice A

Produção científica

Artigo Publicado:

MORAIS, Manoel B. C. , CARDOSO, Pedro Klécius Farias e GOMES, Danielo G. **Controle dinâmico de QoS baseado em políticas**. ***XXVI Simpósio brasileiro de telecomunicações – SBrT'08***, Rio de Janeiro, set. 2008.

Artigo Submetido:

MORAIS, Manoel B. C. , CARDOSO, Pedro Klécius Farias e GOMES, Danielo G. **Controle de QoS dinâmico, não-linear, baseado em políticas, utilizando o controlador PI-P em Ambiente DiffServ**. ***XXVII Simpósio brasileiro de redes de computadores e sistemas distribuídos – SBRC 2009***, Recife, maio 2009.

Apêndice B

Código: Topologia complexa (6 gargalos) e configuração das filas WFQ no NS-2

```
# Manoel Morais - 2009
# GRAPH (#nodes #edges id uu vv ww xx yy zz):
# 100 354 geo(0,{100,100,3,0.033,0.000,100.000}) 100

# Criação da topologia do núcleo da rede

proc create-topology {nsns node linkBW} {
      upvar $node n
      upvar $nsns ns

      global n

      set verbose 1

      if {$verbose} {
            puts "creating nodes..."
}
            for {set i 0} {$i < 100} {incr i} {
                  set n($i) [$ns node]
      }

      # EDGES (from-node to-node length a b):
      if {$verbose} {
            puts -nonewline "Creating links 0..."
            flush stdout
      }
      $ns duplex-link $n(0) $n(58) $linkBW 1ms DropTail
      $ns duplex-link $n(0) $n(56) $linkBW 1ms DropTail
      $ns duplex-link $n(0) $n(54) $linkBW 1ms DropTail
      $ns duplex-link $n(0) $n(15) $linkBW 1ms DropTail
      $ns duplex-link $n(0) $n(9) $linkBW 1ms DropTail
      $ns duplex-link $n(1) $n(73) $linkBW 1ms DropTail
      $ns duplex-link $n(1) $n(61) $linkBW 1ms DropTail
      $ns duplex-link $n(1) $n(23) $linkBW 1ms DropTail
      $ns duplex-link $n(1) $n(20) $linkBW 1ms DropTail
      $ns duplex-link $n(1) $n(8) $linkBW 1ms DropTail
      if {$verbose} { puts -nonewline "10..."; flush stdout }
      $ns duplex-link $n(2) $n(62) $linkBW 1ms DropTail
```

```
$ns duplex-link $n(2) $n(42) $linkBW 1ms DropTail
$ns duplex-link $n(2) $n(10) $linkBW 1ms DropTail
$ns duplex-link $n(3) $n(27) $linkBW 1ms DropTail
$ns duplex-link $n(3) $n(22) $linkBW 1ms DropTail
$ns duplex-link $n(4) $n(96) $linkBW 1ms DropTail
$ns duplex-link $n(4) $n(38) $linkBW 1ms DropTail
$ns duplex-link $n(4) $n(33) $linkBW 1ms DropTail
$ns duplex-link $n(4) $n(24) $linkBW 1ms DropTail
$ns duplex-link $n(5) $n(94) $linkBW 1ms DropTail
if {$verbose} { puts -nonewline "20..."; flush stdout }
$ns duplex-link $n(5) $n(93) $linkBW 1ms DropTail
$ns duplex-link $n(5) $n(73) $linkBW 1ms DropTail
$ns duplex-link $n(5) $n(58) $linkBW 1ms DropTail
$ns duplex-link $n(6) $n(97) $linkBW 1ms DropTail
$ns duplex-link $n(6) $n(90) $linkBW 1ms DropTail
$ns duplex-link $n(7) $n(56) $linkBW 1ms DropTail
$ns duplex-link $n(7) $n(34) $linkBW 1ms DropTail
$ns duplex-link $n(7) $n(26) $linkBW 1ms DropTail
$ns duplex-link $n(7) $n(14) $linkBW 1ms DropTail
$ns duplex-link $n(8) $n(78) $linkBW 1ms DropTail
if {$verbose} { puts -nonewline "30..."; flush stdout }
$ns duplex-link $n(8) $n(70) $linkBW 1ms DropTail
$ns duplex-link $n(8) $n(37) $linkBW 1ms DropTail
$ns duplex-link $n(9) $n(87) $linkBW 1ms DropTail
$ns duplex-link $n(9) $n(78) $linkBW 1ms DropTail
$ns duplex-link $n(9) $n(68) $linkBW 1ms DropTail
$ns duplex-link $n(9) $n(43) $linkBW 1ms DropTail
$ns duplex-link $n(10) $n(95) $linkBW 1ms DropTail
$ns duplex-link $n(10) $n(50) $linkBW 1ms DropTail
$ns duplex-link $n(10) $n(43) $linkBW 1ms DropTail
$ns duplex-link $n(10) $n(25) $linkBW 1ms DropTail
if {$verbose} { puts -nonewline "40..."; flush stdout }
$ns duplex-link $n(11) $n(67) $linkBW 1ms DropTail
$ns duplex-link $n(11) $n(51) $linkBW 1ms DropTail
$ns duplex-link $n(12) $n(77) $linkBW 1ms DropTail
# Enlace congestionado - Atuação do controlador -----
$ns simplex-link $n(67) $n(12) 1.5Mb 1ms dsRED/core
# ------------------------------------------------
$ns simplex-link $n(12) $n(67) $linkBW 1ms DropTail
$ns duplex-link $n(12) $n(60) $linkBW 1ms DropTail
$ns duplex-link $n(12) $n(58) $linkBW 1ms DropTail
# Enlace congestionado - Atuação do controlador -----
$ns simplex-link $n(12) $n(44) 1.5Mb 1ms dsRED/core
# ------------------------------------------------
$ns simplex-link $n(44) $n(12) $linkBW 1ms DropTail
$ns duplex-link $n(13) $n(70) $linkBW 1ms DropTail
$ns duplex-link $n(13) $n(62) $linkBW 1ms DropTail
$ns duplex-link $n(13) $n(46) $linkBW 1ms DropTail
if {$verbose} { puts -nonewline "50..."; flush stdout }
$ns duplex-link $n(14) $n(27) $linkBW 1ms DropTail
$ns duplex-link $n(15) $n(95) $linkBW 1ms DropTail
$ns duplex-link $n(15) $n(40) $linkBW 1ms DropTail
$ns duplex-link $n(16) $n(97) $linkBW 1ms DropTail
$ns duplex-link $n(16) $n(71) $linkBW 1ms DropTail
$ns duplex-link $n(16) $n(60) $linkBW 1ms DropTail
$ns simplex-link $n(49) $n(16) $linkBW 1ms DropTail
$ns simplex-link $n(16) $n(49) $linkBW 1ms DropTail
$ns simplex-link $n(16) $n(36) $linkBW 1ms DropTail
$ns simplex-link $n(36) $n(16) $linkBW 1ms DropTail
$ns duplex-link $n(16) $n(19) $linkBW 1ms DropTail
$ns duplex-link $n(17) $n(96) $linkBW 1ms DropTail
```

```
if {$verbose} { puts -nonewline "60..."; flush stdout }
$ns duplex-link $n(17) $n(57) $linkBW 1ms DropTail
$ns duplex-link $n(17) $n(43) $linkBW 1ms DropTail
$ns duplex-link $n(17) $n(39) $linkBW 1ms DropTail
$ns duplex-link $n(17) $n(33) $linkBW 1ms DropTail
$ns duplex-link $n(17) $n(19) $linkBW 1ms DropTail
$ns duplex-link $n(18) $n(85) $linkBW 1ms DropTail
$ns duplex-link $n(18) $n(75) $linkBW 1ms DropTail
$ns duplex-link $n(18) $n(31) $linkBW 1ms DropTail
$ns duplex-link $n(18) $n(26) $linkBW 1ms DropTail
$ns duplex-link $n(19) $n(85) $linkBW 1ms DropTail
if {$verbose} { puts -nonewline "70..."; flush stdout }
$ns duplex-link $n(19) $n(80) $linkBW 1ms DropTail
$ns duplex-link $n(19) $n(76) $linkBW 1ms DropTail
$ns duplex-link $n(19) $n(26) $linkBW 1ms DropTail
$ns duplex-link $n(20) $n(93) $linkBW 1ms DropTail
$ns duplex-link $n(21) $n(75) $linkBW 1ms DropTail
$ns duplex-link $n(21) $n(50) $linkBW 1ms DropTail
$ns duplex-link $n(21) $n(24) $linkBW 1ms DropTail
$ns duplex-link $n(22) $n(77) $linkBW 1ms DropTail
$ns duplex-link $n(22) $n(45) $linkBW 1ms DropTail
$ns duplex-link $n(22) $n(44) $linkBW 1ms DropTail
if {$verbose} { puts -nonewline "80..."; flush stdout }
$ns duplex-link $n(22) $n(41) $linkBW 1ms DropTail
$ns duplex-link $n(22) $n(27) $linkBW 1ms DropTail
$ns duplex-link $n(23) $n(97) $linkBW 1ms DropTail
$ns duplex-link $n(23) $n(85) $linkBW 1ms DropTail
$ns duplex-link $n(23) $n(35) $linkBW 1ms DropTail
$ns duplex-link $n(24) $n(99) $linkBW 1ms DropTail
$ns duplex-link $n(24) $n(96) $linkBW 1ms DropTail
$ns duplex-link $n(24) $n(52) $linkBW 1ms DropTail
$ns duplex-link $n(25) $n(81) $linkBW 1ms DropTail
$ns duplex-link $n(26) $n(90) $linkBW 1ms DropTail
if {$verbose} { puts -nonewline "90..."; flush stdout }
$ns duplex-link $n(27) $n(90) $linkBW 1ms DropTail
$ns duplex-link $n(27) $n(82) $linkBW 1ms DropTail
$ns duplex-link $n(27) $n(73) $linkBW 1ms DropTail
$ns duplex-link $n(27) $n(53) $linkBW 1ms DropTail
$ns duplex-link $n(27) $n(48) $linkBW 1ms DropTail
$ns duplex-link $n(28) $n(93) $linkBW 1ms DropTail
$ns simplex-link $n(72) $n(28) $linkBW 1ms DropTail
$ns simplex-link $n(28) $n(72) $linkBW 1ms DropTail
$ns simplex-link $n(28) $n(49) $linkBW 1ms DropTail
$ns simplex-link $n(49) $n(28) $linkBW 1ms DropTail
$ns duplex-link $n(29) $n(35) $linkBW 1ms DropTail
$ns duplex-link $n(30) $n(82) $linkBW 1ms DropTail
if {$verbose} { puts -nonewline "100..."; flush stdout }
$ns duplex-link $n(30) $n(64) $linkBW 1ms DropTail
$ns duplex-link $n(31) $n(47) $linkBW 1ms DropTail
$ns duplex-link $n(31) $n(38) $linkBW 1ms DropTail
$ns duplex-link $n(32) $n(63) $linkBW 1ms DropTail
$ns duplex-link $n(33) $n(97) $linkBW 1ms DropTail
$ns duplex-link $n(33) $n(92) $linkBW 1ms DropTail
$ns duplex-link $n(33) $n(89) $linkBW 1ms DropTail
$ns duplex-link $n(33) $n(83) $linkBW 1ms DropTail
$ns duplex-link $n(33) $n(36) $linkBW 1ms DropTail
$ns duplex-link $n(34) $n(57) $linkBW 1ms DropTail
if {$verbose} { puts -nonewline "110..."; flush stdout }
$ns simplex-link $n(36) $n(92) $linkBW 1ms DropTail
$ns simplex-link $n(92) $n(36) $linkBW 1ms DropTail
$ns duplex-link $n(37) $n(85) $linkBW 1ms DropTail
```

```
$ns duplex-link $n(37) $n(74) $linkBW 1ms DropTail
$ns duplex-link $n(37) $n(58) $linkBW 1ms DropTail
$ns duplex-link $n(37) $n(51) $linkBW 1ms DropTail
$ns duplex-link $n(38) $n(96) $linkBW 1ms DropTail
$ns duplex-link $n(38) $n(93) $linkBW 1ms DropTail
$ns duplex-link $n(38) $n(62) $linkBW 1ms DropTail
$ns duplex-link $n(38) $n(55) $linkBW 1ms DropTail
$ns duplex-link $n(38) $n(48) $linkBW 1ms DropTail
if {$verbose} { puts -nonewline "120..."; flush stdout }
$ns duplex-link $n(39) $n(58) $linkBW 1ms DropTail
$ns duplex-link $n(39) $n(50) $linkBW 1ms DropTail
$ns duplex-link $n(40) $n(44) $linkBW 1ms DropTail
$ns duplex-link $n(42) $n(95) $linkBW 1ms DropTail
$ns duplex-link $n(42) $n(87) $linkBW 1ms DropTail
$ns duplex-link $n(43) $n(89) $linkBW 1ms DropTail
$ns duplex-link $n(43) $n(66) $linkBW 1ms DropTail
$ns duplex-link $n(44) $n(96) $linkBW 1ms DropTail
$ns duplex-link $n(44) $n(90) $linkBW 1ms DropTail
$ns duplex-link $n(44) $n(69) $linkBW 1ms DropTail
if {$verbose} { puts -nonewline "130..."; flush stdout }
# Enlace congestionado - Atuação do controlador -----
$ns simplex-link $n(44) $n(65) 1.5Mb 1ms dsRED/core
# ------------------------------------------------
$ns simplex-link $n(65) $n(44) $linkBW 1ms DropTail
$ns duplex-link $n(45) $n(97) $linkBW 1ms DropTail
$ns duplex-link $n(45) $n(86) $linkBW 1ms DropTail
$ns duplex-link $n(46) $n(90) $linkBW 1ms DropTail
$ns duplex-link $n(47) $n(97) $linkBW 1ms DropTail
$ns duplex-link $n(49) $n(78) $linkBW 1ms DropTail
$ns duplex-link $n(50) $n(59) $linkBW 1ms DropTail
$ns duplex-link $n(51) $n(91) $linkBW 1ms DropTail
$ns duplex-link $n(51) $n(86) $linkBW 1ms DropTail
$ns duplex-link $n(51) $n(57) $linkBW 1ms DropTail
if {$verbose} { puts -nonewline "140..."; flush stdout }
$ns duplex-link $n(52) $n(84) $linkBW 1ms DropTail
$ns duplex-link $n(53) $n(97) $linkBW 1ms DropTail
$ns duplex-link $n(54) $n(93) $linkBW 1ms DropTail
$ns duplex-link $n(54) $n(78) $linkBW 1ms DropTail
$ns duplex-link $n(56) $n(75) $linkBW 1ms DropTail
$ns duplex-link $n(57) $n(71) $linkBW 1ms DropTail
$ns duplex-link $n(58) $n(78) $linkBW 1ms DropTail
$ns duplex-link $n(59) $n(71) $linkBW 1ms DropTail
$ns duplex-link $n(60) $n(92) $linkBW 1ms DropTail
$ns duplex-link $n(60) $n(75) $linkBW 1ms DropTail
if {$verbose} { puts -nonewline "150..."; flush stdout }
$ns duplex-link $n(61) $n(86) $linkBW 1ms DropTail
$ns duplex-link $n(63) $n(84) $linkBW 1ms DropTail
$ns duplex-link $n(64) $n(88) $linkBW 1ms DropTail
$ns duplex-link $n(66) $n(69) $linkBW 1ms DropTail
# Enlace congestionado - Atuação do controlador -----
$ns simplex-link $n(70) $n(67) 1.5Mb 1ms dsRED/core
# ------------------------------------------------
$ns simplex-link $n(67) $n(70) $linkBW 1ms DropTail
$ns duplex-link $n(69) $n(98) $linkBW 1ms DropTail
$ns duplex-link $n(69) $n(96) $linkBW 1ms DropTail
$ns duplex-link $n(73) $n(97) $linkBW 1ms DropTail
$ns duplex-link $n(74) $n(80) $linkBW 1ms DropTail
$ns duplex-link $n(74) $n(75) $linkBW 1ms DropTail
if {$verbose} { puts -nonewline "160..."; flush stdout }
$ns duplex-link $n(75) $n(97) $linkBW 1ms DropTail
$ns duplex-link $n(76) $n(94) $linkBW 1ms DropTail
```

```
      $ns duplex-link $n(76) $n(82) $linkBW 1ms DropTail
      $ns duplex-link $n(77) $n(79) $linkBW 1ms DropTail
      $ns duplex-link $n(78) $n(91) $linkBW 1ms DropTail
      $ns duplex-link $n(78) $n(84) $linkBW 1ms DropTail
      $ns duplex-link $n(79) $n(85) $linkBW 1ms DropTail
      $ns duplex-link $n(80) $n(95) $linkBW 1ms DropTail
      $ns duplex-link $n(81) $n(83) $linkBW 1ms DropTail
      $ns duplex-link $n(83) $n(92) $linkBW 1ms DropTail
      if {$verbose} { puts -nonewline "170..."; flush stdout }
      $ns duplex-link $n(84) $n(98) $linkBW 1ms DropTail
      $ns duplex-link $n(85) $n(93) $linkBW 1ms DropTail
      $ns duplex-link $n(85) $n(91) $linkBW 1ms DropTail
      $ns duplex-link $n(89) $n(95) $linkBW 1ms DropTail
      $ns duplex-link $n(89) $n(90) $linkBW 1ms DropTail
      $ns duplex-link $n(91) $n(93) $linkBW 1ms DropTail
      $ns duplex-link $n(96) $n(97) $linkBW 1ms DropTail

      if {$verbose} {
            puts -nonewline "177..."
            flush stdout
            puts "starting"
      }
      return 100}

#
```

```
set ns [new Simulator]

# Criar os arquivos de saida
set f [open out.tr w]
$ns trace-all $f

global ns
set linkBW 5Mb

# Código para utilização da ferramenta gráfica NAM
  set nam [open out.nam w]
  $ns namtrace-all $nam

# inicializacao das variaveis dos fluxo distintos
set TxVoz    500000
set PcVoz    2000
set TxVideo  20000000
set PcVideo  2000

set rndStartTime [new RNG]
$rndStartTime seed 0


# inicializacao das variaveis de controle
set sumVozQueueLen   0
set sumVideoQueueLen 0
set samplesNum       0
set quiet            false
set erro_ant         0
set cont             0
set contX             0
set contX1            0
set contX2            0
```

```
set contX3            0
set contX4            0
set contX5            0
set contX6            0
set contX7            0
set contX8            0
set contX9            0
set contX10           0

set contv             0
set contvY            0
set contvY1           0
set contvY2           0
set contvY3           0
set contvY4           0
set contvY5           0
set contvY6           0
set contvY7           0
set contvY8           0
set contvY9           0
set contvY10          0

set time_voz 0.06
set time_video 0.06
set Kc 0.08  ;# Ganho crítico medido
set Kp [expr 0.45*$Kc] ; # Ganho proporcional
set Pc 8.2 ;# 1.62
set Ti [expr (1/1.2)*$Pc] ;
set Ki [expr ($Kp*($time_voz))/$Ti] ; # Ganho integrador

set peso_voz         10
set peso_vozX        10
set peso_vozX1       10
set peso_vozX2       10
set peso_vozX3       10
set peso_vozX4       10
set peso_vozX5       10
set peso_vozX6       10
set peso_vozX7       10
set peso_vozX8       10
set peso_vozX9       10
set peso_vozX10      10
set peso_max_voz     60
set atraso_max_voz   200
set peso_video       30
set peso_videoY      30
set peso_videoY1     30
set peso_videoY2     30
set peso_videoY3     30
set peso_videoY4     30
set peso_videoY5     30
set peso_videoY6     30
set peso_videoY7     30
set peso_videoY8     30
set peso_videoY9     30
set peso_videoY10    30
set peso_max_vid     200
set atraso_max_vid   200
set tempo_final      350
```

```
# diferenciacão dos fluxos por cor
$ns color 0  Black
$ns color 1  Yellow
$ns color 2  Blue
$ns color 3  Green
$ns color 4  Red

# Geração dos arquivos de dados
set geral           [open geral.tr           w]
set taxas           [open taxas.tr           w]
set filas           [open filas.tr           w]
set descarte        [open descarte.tr        w]
set controle_voz    [open controle_voz.tr    w]
set controle_video  [open controle_video.tr  w]


create-topology ns node $linkBW

# ajuste da topologia da rede
set s(0) [$ns node]
set s(1) [$ns node]
set s(2) [$ns node]
set s(3) [$ns node]
set nx [$ns node]

set e1a [$ns node]
set e1b [$ns node]
set e1c [$ns node]
set e1d [$ns node]

set e2 [$ns node]
set dest [$ns node]

$e1a label "Edge_1a"
$e1b label "Edge_1b"
$e1c label "Edge_1c"
$e1d label "Edge_1d"
$e2 label "Edge_2"
$s(0) label "Voz"
$s(1) label "Video"
$s(2) label "FTP"
$s(3) label "Interferente"
$dest label "Cliente"

$ns duplex-link $s(0) $e1a 5Mb 1ms DropTail
$ns duplex-link $s(1) $e1b 5Mb 1ms DropTail
$ns duplex-link $s(2) $e1c 5Mb 1ms DropTail
$ns duplex-link $s(3) $e1d 5Mb 1ms DropTail

$ns simplex-link $e1a $n(72) 5Mb 1ms dsRED/edge
$ns simplex-link $n(72) $e1a 5Mb 1ms dsRED/core

$ns simplex-link $e1b $n(70) 5Mb 1ms dsRED/edge
$ns simplex-link $n(70) $e1b 5Mb 1ms dsRED/core

$ns simplex-link $e1c $n(20) 5Mb 1ms dsRED/edge
$ns simplex-link $n(20) $e1c 5Mb 1ms dsRED/core

$ns simplex-link $e1d $n(11) 5Mb 1ms dsRED/edge
$ns simplex-link $n(11) $e1d 5Mb 1ms dsRED/core
```

```
# Enlace congestionado - Atuação do controlador -----
$ns simplex-link $nx $e2 1.5Mb 1ms dsRED/core
$ns simplex-link $e2 $nx 1.5Mb 1ms DropTail
# ------------------------------------------------

$ns simplex-link $e2 $dest 5Mb 1ms dsRED/edge
$ns simplex-link $dest $e2 5Mb 1ms DropTail

$ns simplex-link $nx $n(83) 5Mb 1ms DropTail
$ns simplex-link $n(83) $nx 5Mb 1ms DropTail

$ns simplex-link $n(92) $nx $linkBW 1ms DropTail
$ns simplex-link $nx $n(92) $linkBW 1ms DropTail

$ns simplex-link $nx $n(34) 5Mb 1ms DropTail
$ns simplex-link $n(34) $nx 5Mb 1ms DropTail

# Enlace congestionado - Atuação do controlador -----
$ns simplex-link $n(65) $nx 1.5Mb 1ms dsRED/core
# ------------------------------------------------
$ns simplex-link $nx $n(65) $linkBW 1ms DropTail

# Criação das filas que utilizarão o algoritmo WFQ
# (Nós de borda – configuração estática e nós do núcleo - com controle)

# Nós de borda - configuração estática
set qE1Ca    [[$ns link $e1a $n(72)] queue]
set qE1Cb    [[$ns link $e1b $n(70)] queue]
set qE1Cc    [[$ns link $e1c $n(20)] queue]
set qE1Cd    [[$ns link $e1d $n(11)] queue]
set qCE1a    [[$ns link $n(72) $e1a] queue]
set qCE1b    [[$ns link $n(70) $e1b] queue]
set qCE1c    [[$ns link $n(20) $e1c] queue]
set qCE1d    [[$ns link $n(11) $e1d] queue]
set qE2Dest [[$ns link $e2 $dest] queue]

# Nós do núcleo - com controle
set qCE2    [[$ns link $nx $e2] queue]
set qXX [[$ns link $n(12) $n(44)] queue]
set qXX1 [[$ns link $n(44) $n(65)] queue]
set qXX2 [[$ns link $n(67) $n(12)] queue]
set qXX3 [[$ns link $n(70) $n(67)] queue]
set qXX4 [[$ns link $n(65) $nx ] queue]
#---------------------------------------------------------------------
# Configuração das filas que utilizarão o algoritmo WFQ
# ---------------------------------------------------------------------

$qE1Ca set numQueues_ 4
$qE1Ca setNumPrec 0 1; # queue 0, one level of precedence
$qE1Ca setNumPrec 1 1; # queue 1, one level of precedence
$qE1Ca setNumPrec 2 1; # queue 2, one level of precedence
$qE1Ca setNumPrec 3 2; # queue 3, one level of precedence

$qE1Ca setSchedularMode WFQ
$qE1Ca addQueueWeight 0 30
$qE1Ca addQueueWeight 1 40
$qE1Ca addQueueWeight 2 20
$qE1Ca addQueueWeight 3 10

$qE1Ca setQSize 0 300
$qE1Ca setQSize 1 300
```

```
$qE1Ca setQSize 2 100
$qE1Ca setQSize 3 100

$qE1Ca setMREDMode DROP

$qE1Ca addMarkRule 14 [$s(0) id] -1 any any
$qE1Ca addMarkRule 22 [$s(1) id] -1 any any
$qE1Ca addMarkRule 30 [$s(2) id] -1 any any
$qE1Ca addMarkRule 38 [$s(3) id] -1 any any

$qE1Ca addPolicyEntry 14 TokenBucket $TxVoz $PcVoz
$qE1Ca addPolicyEntry 22 TokenBucket $TxVideo $PcVideo
$qE1Ca addPolicyEntry 30 TokenBucket $TxVideo $PcVideo
$qE1Ca addPolicyEntry 38 Dumb

$qE1Ca addPolicerEntry TokenBucket 14 0
$qE1Ca addPolicerEntry TokenBucket 22 0
$qE1Ca addPolicerEntry TokenBucket 30 0
$qE1Ca addPolicerEntry Dumb 38 0

$qE1Ca addPHBEntry 14 0 0
$qE1Ca addPHBEntry 22 1 0
$qE1Ca addPHBEntry 30 2 0
$qE1Ca addPHBEntry 38 3 0
$qE1Ca addPHBEntry 0  3 1

$qE1Ca configQ 0 0 240 300 0.04
$qE1Ca configQ 1 0 240 300 0.06
$qE1Ca configQ 2 0 60  100 0.10
$qE1Ca configQ 3 0 60  100 0.50
$qE1Ca configQ 3 1 60  100 0.80

#---------------------------------------------------------------------------
$qE1Cb set numQueues_ 4
$qE1Cb setNumPrec 0 1; # queue 0, one level of precedence
$qE1Cb setNumPrec 1 1; # queue 1, one level of precedence
$qE1Cb setNumPrec 2 1; # queue 2, one level of precedence
$qE1Cb setNumPrec 3 2; # queue 3, one level of precedence

$qE1Cb setSchedularMode WFQ
$qE1Cb addQueueWeight 0 30
$qE1Cb addQueueWeight 1 40
$qE1Cb addQueueWeight 2 20
$qE1Cb addQueueWeight 3 10

$qE1Cb setQSize 0 300
$qE1Cb setQSize 1 300
$qE1Cb setQSize 2 100
$qE1Cb setQSize 3 100

$qE1Cb setMREDMode DROP

$qE1Cb addMarkRule 14 [$s(0) id] -1 any any
$qE1Cb addMarkRule 22 [$s(1) id] -1 any any
$qE1Cb addMarkRule 30 [$s(2) id] -1 any any
$qE1Cb addMarkRule 38 [$s(3) id] -1 any any

$qE1Cb addPolicyEntry 14 TokenBucket $TxVoz $PcVoz
$qE1Cb addPolicyEntry 22 TokenBucket $TxVideo $PcVideo
$qE1Cb addPolicyEntry 30 TokenBucket $TxVideo $PcVideo
$qE1Cb addPolicyEntry 38 Dumb
```

```
$qE1Cb addPolicerEntry TokenBucket 14 0
$qE1Cb addPolicerEntry TokenBucket 22 0
$qE1Cb addPolicerEntry TokenBucket 30 0
$qE1Cb addPolicerEntry Dumb 38 0

$qE1Cb addPHBEntry 14 0 0
$qE1Cb addPHBEntry 22 1 0
$qE1Cb addPHBEntry 30 2 0
$qE1Cb addPHBEntry 38 3 0
$qE1Cb addPHBEntry 0  3 1

$qE1Cb configQ 0 0 240 300 0.04
$qE1Cb configQ 1 0 240 300 0.06
$qE1Cb configQ 2 0 60  100 0.10
$qE1Cb configQ 3 0 60  100 0.50
$qE1Cb configQ 3 1 60  100 0.80

#-------------------------------------------------------------------------
$qE1Cc set numQueues_ 4
$qE1Cc setNumPrec 0 1; # queue 0, one level of precedence
$qE1Cc setNumPrec 1 1; # queue 1, one level of precedence
$qE1Cc setNumPrec 2 1; # queue 2, one level of precedence
$qE1Cc setNumPrec 3 2; # queue 3, one level of precedence

$qE1Cc setSchedularMode WFQ
$qE1Cc addQueueWeight 0 30
$qE1Cc addQueueWeight 1 40
$qE1Cc addQueueWeight 2 20
$qE1Cc addQueueWeight 3 10

$qE1Cc setQSize 0 300
$qE1Cc setQSize 1 300
$qE1Cc setQSize 2 100
$qE1Cc setQSize 3 100

$qE1Cc setMREDMode DROP

$qE1Cc addMarkRule 14 [$s(0) id] -1 any any
$qE1Cc addMarkRule 22 [$s(1) id] -1 any any
$qE1Cc addMarkRule 30 [$s(2) id] -1 any any
$qE1Cc addMarkRule 38 [$s(3) id] -1 any any

$qE1Cc addPolicyEntry 14 TokenBucket $TxVoz $PcVoz
$qE1Cc addPolicyEntry 22 TokenBucket $TxVideo $PcVideo
$qE1Cc addPolicyEntry 30 TokenBucket $TxVideo $PcVideo
$qE1Cc addPolicyEntry 38 Dumb

$qE1Cc addPolicerEntry TokenBucket 14 0
$qE1Cc addPolicerEntry TokenBucket 22 0
$qE1Cc addPolicerEntry TokenBucket 30 0
$qE1Cc addPolicerEntry Dumb 38 0

$qE1Cc addPHBEntry 14 0 0
$qE1Cc addPHBEntry 22 1 0
$qE1Cc addPHBEntry 30 2 0
$qE1Cc addPHBEntry 38 3 0
$qE1Cc addPHBEntry 0  3 1

$qE1Cc configQ 0 0 240 300 0.04
$qE1Cc configQ 1 0 240 300 0.06
```

```
$qE1Cc configQ 2 0 60  100 0.10
$qE1Cc configQ 3 0 60  100 0.50
$qE1Cc configQ 3 1 60  100 0.80

#---------------------------------------------------------------------------
$qXX set numQueues_ 4
$qXX setNumPrec 0 1; # queue 0, one level of precedence
$qXX setNumPrec 1 1; # queue 1, one level of precedence
$qXX setNumPrec 2 1; # queue 2, one level of precedence
$qXX setNumPrec 3 2; # queue 3, two levels of precedence

$qXX setSchedularMode WFQ
$qXX addQueueWeight 0 30
$qXX addQueueWeight 1 40
$qXX addQueueWeight 2 20
$qXX addQueueWeight 3 10

$qXX setQSize 0 300
$qXX setQSize 1 300
$qXX setQSize 2 100
$qXX setQSize 3 100

$qXX setMREDMode DROP

$qXX addPHBEntry 14 0 0
$qXX addPHBEntry 22 1 0
$qXX addPHBEntry 30 2 0
$qXX addPHBEntry 38 3 0
$qXX addPHBEntry 00 3 1

$qXX configQ 0 0 240 300 0.04
$qXX configQ 1 0 240 300 0.06
$qXX configQ 2 0 60  100 0.10
$qXX configQ 3 0 60  100 0.50

#---------------------------------------------------------------------------
$qXX1 set numQueues_ 4
$qXX1 setNumPrec 0 1; # queue 0, one level of precedence
$qXX1 setNumPrec 1 1; # queue 1, one level of precedence
$qXX1 setNumPrec 2 1; # queue 2, one level of precedence
$qXX1 setNumPrec 3 2; # queue 3, two levels of precedence

$qXX1 setSchedularMode WFQ
$qXX1 addQueueWeight 0 30
$qXX1 addQueueWeight 1 40
$qXX1 addQueueWeight 2 20
$qXX1 addQueueWeight 3 10

$qXX1 setQSize 0 300
$qXX1 setQSize 1 300
$qXX1 setQSize 2 100
$qXX1 setQSize 3 100

$qXX1 setMREDMode DROP

$qXX1 addPHBEntry 14 0 0
$qXX1 addPHBEntry 22 1 0
$qXX1 addPHBEntry 30 2 0
$qXX1 addPHBEntry 38 3 0
$qXX1 addPHBEntry 00 3 1
```

```
$qXX1 configQ 0 0 240 300 0.04
$qXX1 configQ 1 0 240 300 0.06
$qXX1 configQ 2 0 60  100 0.10
$qXX1 configQ 3 0 60  100 0.50

#---------------------------------------------------------------------------
$qXX2 set numQueues_ 4
$qXX2 setNumPrec 0 1; # queue 0, one level of precedence
$qXX2 setNumPrec 1 1; # queue 1, one level of precedence
$qXX2 setNumPrec 2 1; # queue 2, one level of precedence
$qXX2 setNumPrec 3 2; # queue 3, two levels of precedence

$qXX2 setSchedularMode WFQ
$qXX2 addQueueWeight 0 30
$qXX2 addQueueWeight 1 40
$qXX2 addQueueWeight 2 20
$qXX2 addQueueWeight 3 10

$qXX2 setQSize 0 300
$qXX2 setQSize 1 300
$qXX2 setQSize 2 100
$qXX2 setQSize 3 100

$qXX2 setMREDMode DROP

$qXX2 addPHBEntry 14 0 0
$qXX2 addPHBEntry 22 1 0
$qXX2 addPHBEntry 30 2 0
$qXX2 addPHBEntry 38 3 0
$qXX2 addPHBEntry 00 3 1


$qXX2 configQ 0 0 240 300 0.04
$qXX2 configQ 1 0 240 300 0.06
$qXX2 configQ 2 0 60  100 0.10
$qXX2 configQ 3 0 60  100 0.50

#---------------------------------------------------------------------------
$qXX3 set numQueues_ 4
$qXX3 setNumPrec 0 1; # queue 0, one level of precedence
$qXX3 setNumPrec 1 1; # queue 1, one level of precedence
$qXX3 setNumPrec 2 1; # queue 2, one level of precedence
$qXX3 setNumPrec 3 2; # queue 3, two levels of precedence

$qXX3 setSchedularMode WFQ
$qXX3 addQueueWeight 0 30
$qXX3 addQueueWeight 1 40
$qXX3 addQueueWeight 2 20
$qXX3 addQueueWeight 3 10

$qXX3 setQSize 0 300
$qXX3 setQSize 1 300
$qXX3 setQSize 2 100
$qXX3 setQSize 3 100

$qXX3 setMREDMode DROP

$qXX3 addPHBEntry 14 0 0
$qXX3 addPHBEntry 22 1 0
$qXX3 addPHBEntry 30 2 0
$qXX3 addPHBEntry 38 3 0
```

```
$qXX3 addPHBEntry 00 3 1

$qXX3 configQ 0 0 240 300 0.04
$qXX3 configQ 1 0 240 300 0.06
$qXX3 configQ 2 0 60  100 0.10
$qXX3 configQ 3 0 60  100 0.50

#---------------------------------------------------------------------------
$qXX4 set numQueues_ 4
$qXX4 setNumPrec 0 1; # queue 0, one level of precedence
$qXX4 setNumPrec 1 1; # queue 1, one level of precedence
$qXX4 setNumPrec 2 1; # queue 2, one level of precedence
$qXX4 setNumPrec 3 2; # queue 3, two levels of precedence

$qXX4 setSchedularMode WFQ
$qXX4 addQueueWeight 0 30
$qXX4 addQueueWeight 1 40
$qXX4 addQueueWeight 2 20
$qXX4 addQueueWeight 3 10

$qXX4 setQSize 0 300
$qXX4 setQSize 1 300
$qXX4 setQSize 2 100
$qXX4 setQSize 3 100

$qXX4 setMREDMode DROP

$qXX4 addPHBEntry 14 0 0
$qXX4 addPHBEntry 22 1 0
$qXX4 addPHBEntry 30 2 0
$qXX4 addPHBEntry 38 3 0
$qXX4 addPHBEntry 00 3 1

$qXX4 configQ 0 0 240 300 0.04
$qXX4 configQ 1 0 240 300 0.06
$qXX4 configQ 2 0 60  100 0.10
$qXX4 configQ 3 0 60  100 0.50

#---------------------------------------------------------------------------

# Edge1d to Core:
$qE1Cd set numQueues_ 4
$qE1Cd setNumPrec 0 1; # queue 0, one level of precedence
$qE1Cd setNumPrec 1 1; # queue 1, one level of precedence
$qE1Cd setNumPrec 2 1; # queue 2, one level of precedence
$qE1Cd setNumPrec 3 2; # queue 3, one level of precedence

$qE1Cd setSchedularMode WFQ
$qE1Cd addQueueWeight 0 30
$qE1Cd addQueueWeight 1 40
$qE1Cd addQueueWeight 2 20
$qE1Cd addQueueWeight 3 10

$qE1Cd setQSize 0 300
$qE1Cd setQSize 1 300
$qE1Cd setQSize 2 100
$qE1Cd setQSize 3 100

$qE1Cd setMREDMode DROP

$qE1Cd addMarkRule 14 [$s(0) id] -1 any any
```

```
$qE1Cd addMarkRule 22 [$s(1) id] -1 any any
$qE1Cd addMarkRule 30 [$s(2) id] -1 any any
$qE1Cd addMarkRule 38 [$s(3) id] -1 any any

$qE1Cd addPolicyEntry 14 TokenBucket $TxVoz $PcVoz
$qE1Cd addPolicyEntry 22 TokenBucket $TxVideo $PcVideo
$qE1Cd addPolicyEntry 30 TokenBucket $TxVideo $PcVideo
$qE1Cd addPolicyEntry 38 Dumb

$qE1Cd addPolicerEntry TokenBucket 14 0
$qE1Cd addPolicerEntry TokenBucket 22 0
$qE1Cd addPolicerEntry TokenBucket 30 0
$qE1Cd addPolicerEntry Dumb 38 0

$qE1Cd addPHBEntry 14 0 0
$qE1Cd addPHBEntry 22 1 0
$qE1Cd addPHBEntry 30 2 0
$qE1Cd addPHBEntry 38 3 0
$qE1Cd addPHBEntry 0  3 1

$qE1Cd configQ 0 0 240 300 0.04
$qE1Cd configQ 1 0 240 300 0.06
$qE1Cd configQ 2 0 60  100 0.10
$qE1Cd configQ 3 0 60  100 0.50
$qE1Cd configQ 3 1 60  100 0.80

#-------------------------------------------------------------------------
# Ajuste dos parametros Core para o Edge1a:
$qCE1a setMREDMode DROP
$qCE1a set numQueues_ 1
$qCE1a setNumPrec       1
$qCE1a addPHBEntry  00 3 1
$qCE1a configQ      3 1 60 100 0.02

#-------------------------------------------------------------------------
# Ajuste dos parametros Core para o Edge1b:
$qCE1b setMREDMode DROP
$qCE1b set numQueues_ 1
$qCE1b setNumPrec       1
$qCE1b addPHBEntry  00 3 1
$qCE1b configQ      3 1 60 100 0.02

#-------------------------------------------------------------------------
# Ajuste dos parametros Core para o Edge1c:
$qCE1c setMREDMode DROP
$qCE1c set numQueues_ 1
$qCE1c setNumPrec       1
$qCE1c addPHBEntry  00 3 1
$qCE1c configQ      3 1 60 100 0.02

#-------------------------------------------------------------------------

# Ajuste dos parametros Core para o Edge1d:
$qCE1d setMREDMode DROP
$qCE1d set numQueues_ 1
$qCE1d setNumPrec       1
$qCE1d addPHBEntry  00 3 1
$qCE1d configQ      3 1 60 100 0.02

#-------------------------------------------------------------------------
```

```
# Ajuste dos parametros Core para o Edge2:
$qCE2 set numQueues_ 4
$qCE2 setNumPrec 0 1; # queue 0, one level of precedence
$qCE2 setNumPrec 1 1; # queue 1, one level of precedence
$qCE2 setNumPrec 2 1; # queue 2, one level of precedence
$qCE2 setNumPrec 3 2; # queue 3, two levels of precedence

$qCE2 setSchedularMode WFQ
$qCE2 addQueueWeight 0 30
$qCE2 addQueueWeight 1 40
$qCE2 addQueueWeight 2 20
$qCE2 addQueueWeight 3 10

$qCE2 setQSize 0 300
$qCE2 setQSize 1 300
$qCE2 setQSize 2 100
$qCE2 setQSize 3 100

$qCE2 setMREDMode DROP

$qCE2 addPHBEntry 14 0 0
$qCE2 addPHBEntry 22 1 0
$qCE2 addPHBEntry 30 2 0
$qCE2 addPHBEntry 38 3 0
$qCE2 addPHBEntry 00 3 1

$qCE2 configQ 0 0 240 300 0.04
$qCE2 configQ 1 0 240 300 0.06
$qCE2 configQ 2 0 60  100 0.10
$qCE2 configQ 3 0 60  100 0.50

#------------------------------------------------------------------------
$qE2Dest set numQueues_ 4
$qE2Dest setNumPrec 0 1; # queue 0, one level of precedence
$qE2Dest setNumPrec 1 1; # queue 1, one level of precedence
$qE2Dest setNumPrec 2 1; # queue 2, one level of precedence
$qE2Dest setNumPrec 3 2; # queue 3, one level of precedence

$qE2Dest setSchedularMode WFQ
$qE2Dest addQueueWeight 0 30
$qE2Dest addQueueWeight 1 40
$qE2Dest addQueueWeight 2 20
$qE2Dest addQueueWeight 3 10

$qE2Dest setQSize 0 300
$qE2Dest setQSize 1 300
$qE2Dest setQSize 2 100
$qE2Dest setQSize 3 100

$qE2Dest setMREDMode DROP

$qE2Dest addMarkRule 14 [$s(0) id] -1 any any
$qE2Dest addMarkRule 22 [$s(1) id] -1 any any
$qE2Dest addMarkRule 30 [$s(2) id] -1 any any
$qE2Dest addMarkRule 38 [$s(3) id] -1 any any

$qE2Dest addPolicyEntry 14 Dumb
$qE2Dest addPolicyEntry 22 Dumb
$qE2Dest addPolicyEntry 30 Dumb
$qE2Dest addPolicyEntry 38 Dumb
```

```
$qE2Dest addPolicerEntry Dumb 14 0
$qE2Dest addPolicerEntry Dumb 22 0
$qE2Dest addPolicerEntry Dumb 30 0
$qE2Dest addPolicerEntry Dumb 38 0

$qE2Dest addPHBEntry 14 0 0
$qE2Dest addPHBEntry 22 1 0
$qE2Dest addPHBEntry 30 2 0
$qE2Dest addPHBEntry 38 3 0
$qE2Dest addPHBEntry 0  3 1

$qE2Dest configQ 0 0 240 300 0.04
$qE2Dest configQ 1 0 240 300 0.06
$qE2Dest configQ 2 0 60  100 0.10
$qE2Dest configQ 3 0 60  100 0.50
$qE2Dest configQ 3 1 60  100 0.80

#--------------------------------------------------------------------------
```