Qualidade de Serviço em um Domínio DiffServ através de Gerenciamento Baseado em Políticas

Marcial Porto Fernandez[1]
marcial@gta.ufrj.br

Aloysio de Castro P. Pedroza[1,2]
aloysio@gta.ufrj.br

José Ferreira de Rezende[1]
rezende@gta.ufrj.br

[1]Grupo de Teleinformática e Automação (GTA)
Universidade Federal do Rio de Janeiro (UFRJ)
COPPE/PEE - Programa de Engenharia Elétrica
C.P. 68504 - CEP 21945-970 - Rio de Janeiro - RJ - Brasil
Tel: (021) 260-5010 Fax: (021) 290-6626
[2]Departamento de Eletrônica / EE-UFRJ

Resumo

A arquitetura de Serviços Diferenciados (Diffserv) tem por finalidade oferecer garantia de qualidade de serviço (QoS) na Internet. A maioria dos estudos sobre Diffserv (DS), no entanto, tem-se concentrado em garantir qualidade de serviço em cada nó da rede, supondo-se que a garantia individual proporciona a qualidade no domínio DS, o que nem sempre acontece. Este trabalho objetiva oferecer uma ferramenta para garantir QoS no âmbito do domínio, apresentando, para isso, um controlador baseado em lógica fuzzy, que reconfigura o nó Diffserv em função do tráfego passante e da política definida para o domínio. A técnica utilizada para permitir a qualidade no domínio é a de gerenciamento baseado em políticas, que oferece recursos para controlar uma rede heterogênea, com equipamentos de diversos modelos e fabricantes. A viabilidade e funcionalidade do modelo são demonstradas através de simulação de um exemplo de aplicação voz sobre IP.

Abstract

The Differentiated Services architecture has been proposed to offer quality of service in the Internet. Most of the works on Diffserv (DS) handles QoS garantees in a per node basis. This assumes that guaranteeing QoS in a a single node also leads to QoS in the entire DS domain. Nevertheless, this is not always true. This paper proposes a framework that offers QoS in a DS domain using a fuzzy logic controller. This QoS controller reconfigures all DS nodes according ingress traffic and domain policies. In this framework, Policy Based Management is used to provide QoS in DS domain, controling heterogeneous equipments. The prototype performance and functionalities are shown by simulation of a voice over IP application.

1 Introdução

A arquitetura atual da Internet não oferece as garantias de qualidade de serviço (QoS) exigidas pelas aplicações multimídia. A Diferenciação de Serviços (DiffServ) [18] é uma proposta de solução desse problema, consistindo em prover serviços diferenciados para uma agregação de fluxos de dados. Soluções alternativas para o problema são a Integração de Serviços (Intserv), que assegura qualidade de serviço para cada conexão, e a Engenharia de Tráfego (Trafic Engineering), onde a rede determina o melhor caminho que um determinado fluxo deve seguir para a obtenção das garantias solicitadas.

A arquitetura Diffserv, no entanto, pode apresentar problemas quando se trata de oferecer garantias de QoS em todo o domínio ou entre diversos domínios, o que pode comprometer a qualidade de serviço fim-a-fim. Nessa arquitetura, não há controle de QoS para cada fluxo de dados, como acontece no Intserv, e a qualidade do serviço podei, portanto, sofrer degradação quando ocorre congestionamento na agregação de vários fluxos. Além disso, o controle, exercido por nó, pode ser afetado pela heterogeneidade de equipamentos, configurações e topologias, dificultando a garantia de qualidade borda-a-borda de um domínio. Uma proposta de solução, nesse caso, é suprir o controle de QoS no nó Diffserv com uma ação de gerenciamento sobre todo o domínio, isto é, no conjunto de equipamentos que formam a rede de uma entidade provedora de meios de comunicação.

O gerenciamento baseado em políticas tem sido proposto como infra-estrutura para garantir qualidade de serviço nas arquiteturas Diffserv e Intserv[3]. Essa técnica não se aplica no caso de vários domínios, que normalmente estão sob a responsabilidade de entidades administrativas distintas. Entretanto, o gerenciamento é perfeitamente aplicável ao domínio, geralmente controlado por uma instituição apenas.

Este trabalho propõe o controle de QoS, dentro de um único domínio Diffserv, apresentando, para isso, um controlador fuzzy, que configura dinamicamente os nós do domínio, através da arquitetura de gerenciamento baseado em políticas. A lógica fuzzy foi utilizada em função das características de incerteza e imprecisão inerentes ao fluxo de dados, pois a previsão do tráfego em determinado nó da Internet é um problema complexo. A utilização de lógica fuzzy, em vista disso, possibilita solução interessante no ajuste dos parâmetros de controle. Comparado a um controlador convencional (digital simples), o controlador fuzzy apresenta vantagens significativas no tratamento de variáveis imprecisas, mantendo-se uma complexidade pequena.

O controlador fuzzy é especificado através da definição das variáveis semânticas (função de pertinência) e do conjunto de regras. A alteração nesses parâmetros possibilita alterar o comportamento do controlador, de maneira a tornar sua ação mais ou menos agressiva. Esta forma de atuação é definida por determinações administrativas, que faz parte do sistema de gerenciamento baseado em políticas. A utilização desse tipo de controlador para garantir qualidade de serviço em uma rede foi apresentada por Vasilakos e Anagnostakis[7]. O trabalho destes autores, no entanto, foi aplicado em ambiente de Engenharia de Tráfego, com concepção bem distinta do ambiente de gerenciamento baseado em políticas.

Para validar o modelo de controlador proposto foi realizada simulação de um protótipo. A validação do protótipo foi realizada através de uma situação prática, avaliação de tempo de retardo, variação do retardo e descarte de fluxo de voz sobre IP, concorrente com outros tráfegos não sensíveis ao retardo.

O presente trabalho encontra-se organizado da seguinte forma: a seção 2 apresenta um resumo sobre gerenciamento baseado em políticas e os problemas de controle de QoS na Internet; a seção 3 apresenta a arquitetura do controlador fuzzy para um nó DiffServ, conforme o proposto; enquanto a seção 4 mostra o ambiente de simulação e as ferramentas utilizadas e apresenta o modelo de simulação detalhando o controlador fuzzy implementado; a seção 5 apresenta os resultados obtidos na simulação; finalmente, a seção 6 apresenta as conclusões do trabalho e sugere temas para trabalhos futuros.

2 Qualidade de Serviço na Internet

Apresentamos, nesta seção, a arquitetura de um sistema de gerenciamento baseado em políticas e indicamos a posição de nosso controlador nessa arquitetura. A seguir, mostramos trabalhos que avaliam o problema de controle de QoS na Internet, bem como propostas para resolvê-los usando um controlador fuzzy.

2.1 Gerenciamento Baseado em Políticas

O gerenciamento baseado em políticas foi inicialmente proposto por Sloman [1]. O autor descreveu uma metodologia de especificar parâmetros de comportamento, mais abstratos, a serem cumpridos por cada elemento da rede, da melhor maneira possível, levando-se em conta as características de cada elemento. Em um sistema heterogêneo, constituído por diversos equipamentos de diversos fabricantes, essas políticas poderão assumir valores distintos em função das características particulares de cada elemento gerenciado.

Uma das aplicações sugeridas para gerenciamento baseado em políticas é o controle da qualidade de serviço (QoS) em uma rede IP. Rajan et al.[2] dá uma visão geral do uso de políticas em QoS, para serviços diferenciados e integrados. Estudo mais detalhado, focando a questão da escalabilidade alcançada com gerenciamento baseado em políticas para controlar QoS em redes públicas, encontra-se em Blight e Hamada [3].

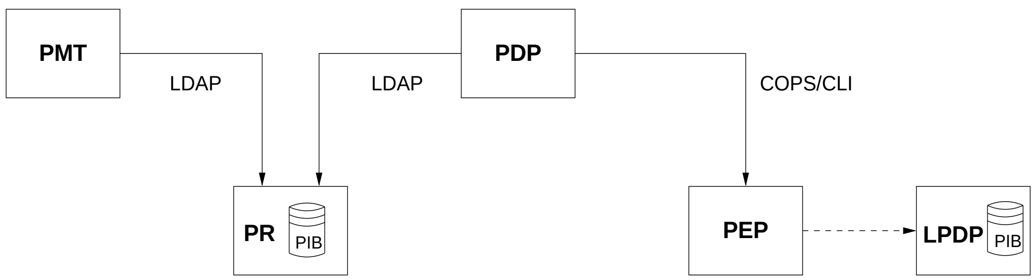


Figura 1: Arquitetura de Gerenciamento Baseado em Políticas

A arquitetura de um sistema de gerenciamento baseado em políticas foi proposto pelo IETF [8] [9] e é mostrada na Figura 1. O PMT (Policy Management Tool) é a interface com o administrador da rede, que especifica os requisitos de serviço. PR (Policy Repository) é o banco de dados de políticas. PDP (Policy Decision Point) é o componente responsável pela interpretação da política e pela decisão de configuração dos equipamentos. Sua função é adaptar a política para cada dispositivo que está sendo gerenciado. A comunicação com os diversos componentes é realizada através do protocolo LDAP. PIB (Policy Information Base) é a tabela que armazena todas as políticas do sistema. A comunicação com o PEP é realizada através do protocolo COPS [10]. PEP (Policy Enforcement Point) é o dispositivo que executa as ações determinadas pelo PDP. LPDP (Local Policy Decision Point) é um elemento opcional que contém seu próprio PDP e PIB, para a operação do sistema na falha do PDP.

O elemento chave desta arquitetura é o PDP que executa grande parte do processamento de controle do sistema. Este trabalho propõe um controlador fuzzy baseado em políticas, que ocupa a função de PDP na arquitetura de gerenciamento baseado em políticas.

2.2 Controle de QoS na Internet e Trabalhos Relacionados

É tarefa complexa controlar os parâmetros de qualidade de serviço (QoS), pois requer previsão do tráfego que entrará no sistema em um determinado período. Como tráfego é uma entidade com características aleatórias, as previsões tendem a ser pouco confiáveis.

Guérin e Orda [4] apresentam um estudo sobre inacurácia e incerteza do estado de uma rede para garantir o QoS das conexões. Este trabalho demonstra que, quando a conexão exige apenas banda passante, a inacurácia não influencia no resultado final, porém, quando a conexão exige garantias de retardo fim-a-fim, a inacurácia da rede torna intratável o processo de seleção do caminho para garantir o QoS. Lorenz e Orda [6] mostram também que a incerteza do estado da rede torna difícil de resolver o problema de garantir o retardo fim-a-fim em um domínio. No entanto, eles mostram que, decompondo os requisitos de retardo em restrições locais e definindo uma classe de distribuição de probabilidade, é possível estabelecer uma solução eficiente e exata. A complexidade do algoritmo, entretanto, é muito grande para processamento no nó, obrigando então a usar uma aproximação polinomial, que é proposta neste artigo.

O problema de incerteza e imprecisão nos remetem à solução baseada em lógica fuzzy e realmente vários trabalhos apresentaram soluções de controladores baseados em lógica fuzzy, como é o caso de Li e Nahrstedt [10][11]. Os autores, no entanto, focaram o ambiente de configuração e não trataram profundamente o controle de recursos na rede. Cheng e Chang [12] mostram um controlador fuzzy para configurar os parâmetros de uma rede ATM. Este artigo, apesar de tratar dos parâmetros de rede, pressupõe a utilização de rede ATM, que já oferece intrinsecamente recursos de QoS. Vasilakos e Anagnostakis [7] apresentam um controlador fuzzy para determinar o melhor caminho que um pacote deve seguir, com o objetivo de oferecer garantias de QoS. Esta proposta, aplicável à Engenharia de Tráfego, utiliza algoritmo genético aplicado sobre o histórico do tráfego no nó para definir os parâmetros fuzzy.

3 Arquitetura do Controlador Fuzzy

Nesta seção mostramos os parâmetros de medida de QoS em um nó DiffServ e a separação para os dois tipos de nós na arquitetura - nó de borda e nó de núcleo. Essa arquitetura, apesar de especificada para DiffServ, pode ser adaptada para qualquer mecanismo que utilize filas baseadas em classes.

3.1 Controle de um Nó DiffServ

O controlador em uma arquitetura DiffServ, mostrado na Figura 2, é constituído de duas partes, aplicadas conforme sua localização no domínio, se nos nós de borda ou nos nós de núcleo. Na arquitetura DiffServ, todos os nós têm uma fila distinta para cada classe, um classificador que coloca o pacote na respectiva fila e um escalonador que retira pacotes das filas. Os nós de borda, além das funções anteriores, contêm um marcador que marca ou remarca cada pacote e um condicionador que tem a função de manter o fluxo de entrada, conforme contratado. A arquitetura proposta implementa dois controladores: um para o controle de filas e escalonador, usado nos nós de núcleo e borda, e outro para o condicionador, utilizado apenas em nós de borda.

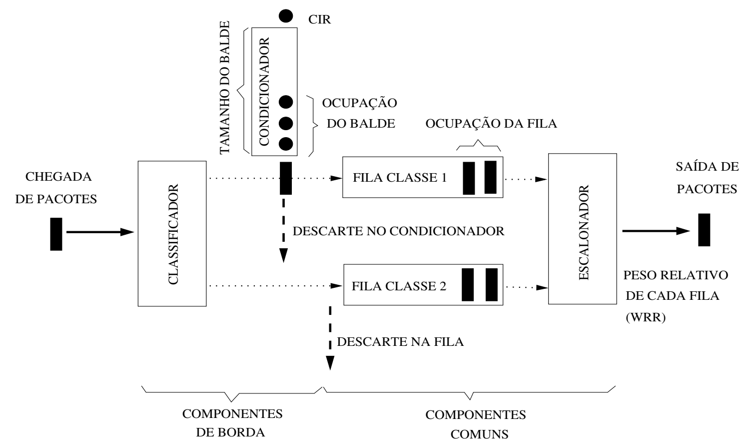


Figura 2: Arquitetura de um nó DiffServ

Controlador do Escalonador A variável de saída que possibilita o controle do escalonador, depende do tipo do mecanismo utilizado. Quando o escalonador for do tipo prioritário, não há controle a ser efetuado, pois aqui a regra é que as classes menos prioritárias somente serão servidas quando a classe mais prioritária não contiver mais nenhum pacote. O escalonador controlável deve ser do tipo WRR (Weighted Round-Robin) ou WFQ (Weighted Fair-Queueing), em que as filas são servidas de acordo com o peso definido na configuração. Alterando-se esse peso, podemos modificar o retardo dos pacotes em cada fila (classe).

A primeira variável de entrada é o tempo que o pacote fica na fila. Como os demais tempos do processamento do pacote são praticamente irrelevantes, consideramos o tempo de espera na fila como tempo total no nó. Da mesma forma que o tempo de espera do pacote, variável equivalente seria o tamanho da fila, pois é diretamente proporcional ao retardo esperado. A segunda variável de entrada é a taxa de descarte por transbordamento da fila. A fila é um recurso limitado pelo tamanho, e que descarta pacotes quando atinge este limite. Vários mecanismos de gerenciamento ativo de fila podem ser utilizados neste caso, porém devemos considerar que, na ocorrência de descarte, há séria escassez de recurso disponível.

Controlador do Condicionador O condicionador está presente apenas nos nós de borda do domínio DiffServ. O objetivo principal deste controlador é policiar a entrada de fluxos de dados no domínio, de maneira que os fluxos bem comportados não sejam penalizados. A variável para controlar o condicionador depende do tipo utilizado. De forma geral, seguem a filosofia do balde furado, que é um repositório onde se colocam fichas a uma taxa constante.

A primeira variável de entrada é o nível do balde, isto é, a quantidade de fichas armazenadas no repositório. Se o balde está cheio, significa que o tráfego entrante no nó é menor que a taxa de enchimento; se o balde está vazio, significa que a taxa de enchimento do balde está compatível com o tráfego entrante ou é menor. A segunda variável é o descarte de pacotes no condicionador. Quando um pacote chega ao condicionador e encontra o balde vazio, este pacote é descartado, pelo que podemos concluir que o tráfego entrante é maior que a taxa contratada para sua entrada no domínio.

O condicionador, entretanto, não pode ter o valor da taxa de enchimento do balde alterada, pois seu objetivo é manter um tráfego suavizado e conforme ao contratado, não havendo sentido em aumentar ou reduzir sua taxa. Por definição, um domínio DiffServ só deve descartar pacotes na borda, assim, descartes no interior do domínio são indesejáveis. Quando não há mais recursos no núcleo do domínio, devemos sinalizar para os nós de borda uma redução na taxa de entrada, através da redução da taxa de enchimento do balde. A função do controlador do condicionador é ajustar a taxa conforme a variação da entrada. Quando houver alguma perda no núcleo, os nós de borda reduzirão a taxa de entrada, baseados na taxa média de entrada real. Por exemplo, se a taxa de referência é 100 e a taxa média real for 50, a redução indicada pelo sistema de gerenciamento, digamos de 10%, deve ser aplicada no valor de 50 e não no valor de 100, que não produziria o resultado esperado na redução da taxa de entrada.

Esta foi a política escolhida para tratar congestionamentos no núcleo, porém poderíamos usar, dependendo das conveniências do contratante, o critério de manter a taxa contratada e recusar a entrada de novas conexões ou cortar algumas conexões (atuando no marcador).

4 Modelo de Simulação

O modelo de simulação utilizou as classes EF *(Expedited Forwarding)* e BE *(Best Effort)*. Essa escolha deve-se ao fato de ser a classe EF ideal para aplicações com restrição de tempo, como é a voz sobre IP, pois oferece o menor retardo em cada nó. A classe BE representa uma rede IP (Internet) sem garantias de retardo, usada aqui para concorrer com o tráfego da classe EF. Além disso, Lorenz e Orda [7] demonstraram que a incerteza é fator agravante para garantia de qualidade (retardo e variação de retardo), o que possibilita verificar com mais rigor as vantagens do controlador fuzzy, proposto neste trabalho.

4.1 Ambiente de Simulação

A metodologia utilizada para validação do modelo proposto foi a de simulação. A plataforma de simulação utilizada foi o Network Simulator (NS), versão 2.1b6a[5]. Como a distribuição padrão do NS não inclui recursos DiffServ, foi necessário incluir uma implementação adicional. O modelo escolhido foi proposto por Pieda e Ethridge[6] e apresenta todas as funcionalidades necessárias para nossa experiência: PHBs, *Assured Forwarding* (AF), *Expedited Forwarding* (EF), *Class Selector* e *Default* (BE); condicionadores, *TSW 2 Color Marker, TSW 3 Color Marker, Single Rate 3CM, Two Rate 3CM* e *Token Bucket*; gerenciamento ativo de filas, RIO-C *(Coupled)*, RIO-D *(Decoupled)*, WRED e *DropTail*; escalonadores, *Weighted Round Robin* (WRR), *Weighted Interleaved Round Round* (WIRR), *Round Robin* (RR) e *Priority* (PRIO). Foi necessário acrescentar novas funções neste modelo DiffServ, para que permitisse a alteração de alguns parâmetros de configuração, não existentes na versão original.

O controlador fuzzy utilizado neste trabalho foi desenvolvido com a ferramenta Unfuzzy de Duarte[7]. Esta ferramenta oferece interface gráfica para desenvolvimento do protótipo (especificação das funções de pertinência, regras de inferência e o desfuzificador), além de permitir verificação inicial do modelo especificado. Após o desenvolvimento do modelo, é gerada biblioteca em código C ou C++, que implementa o controlador integrado ao simulador NS. O código gerado por esse programa atendeu nossas expectativas e possibilitou implementar as funções de um controlador fuzzy, com código eficiente e em curto espaço de tempo.

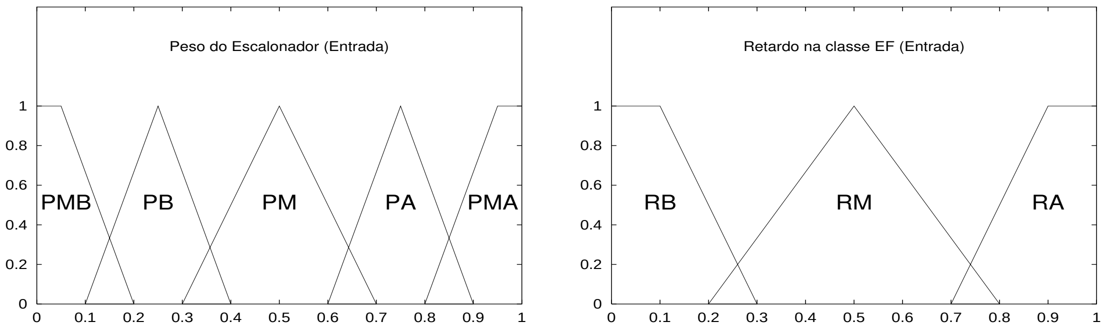
4.2 Funções de Pertinência e Fuzificação

Para o nosso exemplo, definimos a política de que toda prioridade deve ser dada à classe EF, isto é, a classe BE será reduzida sempre que houver queda na qualidade da classe EF. Apesar dessa regra, a banda passante dedicada à classe BE nunca deve ser zero, por isso a classe EF nunca deve ocupar mais de 90% da banda, aproximadamente. Essa política foi arbitrada para nosso exemplo, porém poderiam ser definidas várias outras, bastando para isso alterar as funções de pertinência e base de regras utilizadas.

O controlador proposto utiliza variáveis lingüísticas triangulares e trapezoidais por serem implementadas com código mais simples e eficiente. Realizamos também experiência com uma função gaussiana, porém os resultados obtidos não justificaram a complexidade adicionada. O controlador do escalonador e do condicionador são apresentados a seguir.

4.2.1 Controlador do Escalonador

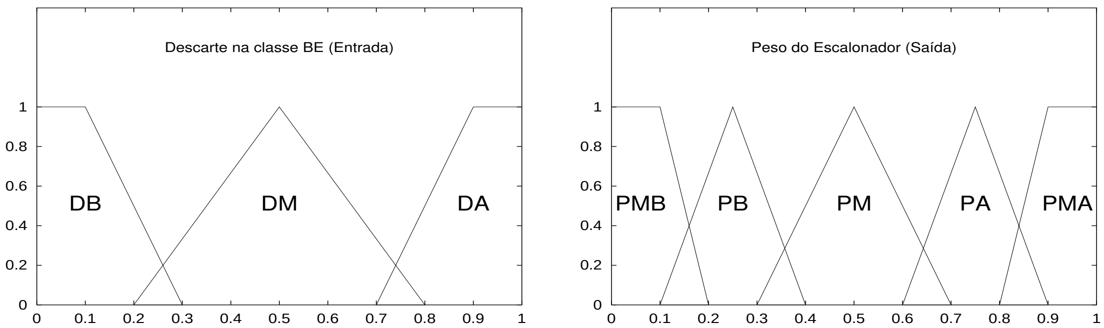
A primeira função de pertinência de entrada, Peso EF/BE, é mostrada na Figura 3(a). Ela representa o peso escalonador WRR da fila EF em relação ao peso total (EF+BE). As variáveis semânticas são: "PMB", prioridade muito baixa; "PB", prioridade baixa; "PM", prioridade média; "PA" , prioridade alta e "PMA" , prioridade muito alta. Como a relação peso da fila EF por peso total do WRR é sempre um número entre 0 e 1, não foi necessário efetuar a normalização dos valores.



(a) Entrada: peso EF/BE

(b) Entrada: Retardo fila EF

Figura 3: Funções de Pertinência do Controlador do Escalonador



(a) Entrada: Descarte na Classe BE

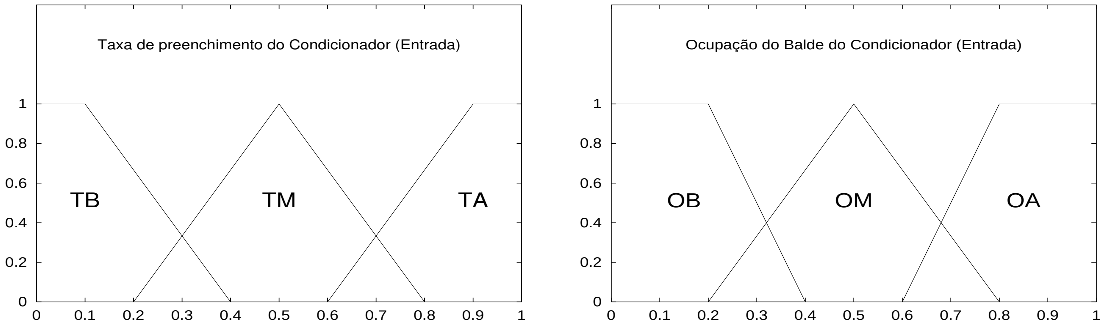
(b) Saida: peso do Escalonador

Figura 4: Funções de Pertinência do Controlador do Escalonador

A segunda função de entrada, retardo da fila EF, é mostrada na Figura 3(b). Este sensor lê o valor de retardo de um pacote durante sua passagem pelo nó. Utilizamos o valor de retardo porque esta é a métrica principal para avaliação de QoS. As variáveis semânticas são: "RB", retardo baixo; “RM”, retardo médio e “RA”, retardo alto. O valor lido é o tempo em segundos e a variável precisa ser normalizada antes de entrar no controlador.

A terceira função de pertinência de entrada, descarte de pacotes na fila BE, é mostrada na Figura 4. Este sensor lê a quantidade de pacotes da classe BE descartados em um determinado intervalo de tempo, com o objetivo de avaliar a possibilidade de redução na prioridade da fila EF. O valor é normalizado em relação ao percentual máximo de perdas. As variáveis semânticas são: “DB”, descarte baixo; “DM”, descarte médio e “DA”, descarte alto.

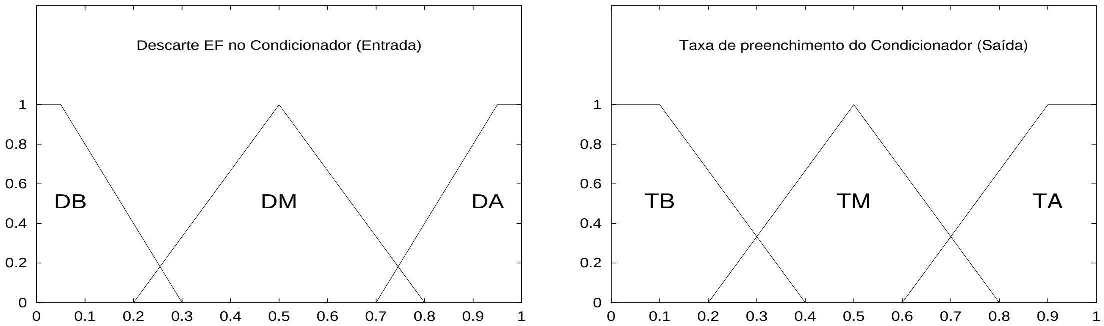
4.2.2 Controlador do Condicionador



(a) Entrada: Taxa enchimento do balde

(b) Entrada: Ocupaçao do balde

Figura 5: Funções de Pertinência do Controlador do Condicionador



(a) Entrada: Descarte EF Condicionador

(b) Saida: Taxa de enchimento do balde

Figura 6: Funções de Pertinência do Controlador do Condicionador

A primeira função de pertinência de entrada do controlador do condicionador, taxa de preenchimento do balde, é mostrada na Figura 6(a). Este sensor lê o valor da taxa atual de preenchimento do balde do condicionador. Este valor é normalizado em relação à taxa máxima de enchimento do balde. As variáveis semânticas são: “TB”, taxa baixa; “TM”, taxa média e “TA”, taxa alta.

A segunda função de pertinência de entrada do controlador do condicionador, ocupação do balde do condicionador, é mostrada na Figura 5(b). Este sensor lê o valor de ocupação do balde do condicionador da classe EF auxiliando na medida da taxa real do tráfego entrante. Quando a ocupação do balde é alta significa que a taxa de chegada de pacotes é menor do que a taxa de enchimento do balde. Este valor é normalizado em relação ao tamanho do balde. As variáveis semânticas são: “OB”, ocupação baixa; “OM”, ocupação média e “OA”, ocupação alta.

A terceira função de pertinência de entrada, descarte de pacotes na fila EF, é mostrada na Figura 6(a). Este sensor mede a quantidade de pacotes descartados no condicionador. Quando um pacote é descartado, indica que o mesmo não encontrou ficha no balde, portanto a taxa de enchimento do balde é menor que a taxa de entrada. O valor é normalizado em relação ao percentual máximo de perdas admitido para a classe EF. As variáveis semânticas são: “DB”, descarte baixo; “DM”, descarte médio e “DA”, descarte alto.

4.2.3 Funções de Pertinência de Saída e Desfuzificador

As funções de saída também foram definidas como funções trapezoidais, pelos motivos expostos anteriormente. Utilizamos o método de desfuzificação de centro de gravidade, pois ofereceu um bom resultado para nossa aplicação, além de exigirem menos cálculos de ponto flutuante que outros métodos[17]. A primeira função de pertinência de saída, Peso EF/BE, é mostrada na Figura 4(b). Ela fornece o valor do peso do escalonador WRR da fila EF em relação ao peso total (EF+BE). As variáveis semânticas são: “PMB”, prioridade muito baixa; “PB”, prioridade baixa; “PM”, prioridade média; “PA” , prioridade alta e “PMA”, prioridade muito alta. Este valor é normalizado para um número inteiro que será aplicado ao simulador.

A segunda função de pertinência de saída, taxa de enchimento do balde, é mostrada na Figura 6(b). Ela fornece o valor da taxa de enchimento do balde do condicionador da classe EF. O resultado é normalizado em relação à taxa máxima. As variáveis semânticas são: “TB”, taxa baixa; “TM”, taxa média e “TA”, taxa alta.

4.3 Base de Regras e Sistema de Inferência

A base de regras é o conjunto de regras SE-ENTÃO dos valores lingüísticos que representam o comportamento desejado do sistema fuzzy. Conforme dito anteriormente, a base de regras pode ser definida de acordo com a política administrativa. Para nosso exemplo, utilizamos um conjunto de regras que priorizasse a classe EF em qualquer situação, deixando para a classe BE um mínimo de 10% da banda.

Os operadores fuzzy selecionados foram definidos experimentalmente. Para união e interseção foi escolhido o operador **máximo** (envoltória externa de todas as variáveis semânticas válidas). A implicação utilizada foi do tipo **produto** e a disjunção (OU) e conjunção (E), do tipo **mínimo**. Testados vários outros operadores, entretanto essa combinação ofereceu o melhor resultado, isto é, a reação às variações foi rápida e a oscilação não foi crítica.

4.3.1 Controlador do Escalonador

O controlador do escalonador foi definido através de 45 regras, das quais apresentamos uma síntese a seguir:

1. Se retardo da fila EF é médio (RM), então prioridade da fila é incrementado em 1, por exemplo, se era prioridade baixa (PB) passa a prioridade média (PM). Se retardo da fila EF é alto (RA), então prioridade da fila é incrementado em 2.

2. Se retardo da fila EF é baixo (RB) e descarte na fila BE é médio (DM), então prioridade da fila é reduzido em 1, por exemplo, se era prioridade média (PM) passa a prioridade

baixa (PB). Se retardo da fila EF é baixo (RB) e descarte na fila BE é alto (DA), então prioridade da fila é reduzido em 2.

4.3.2 Controlador do Condicionador

O controlador do condicionador foi definido através de 27 regras, das quais apresentamos uma síntese a seguir:

1. Se ocupação do balde do condicionador é baixa (OB) e o descarte na fila EF é médio (DM), então a taxa do condicionador é incrementada em 1, por exemplo, se era taxa baixa (TB) passa a taxa média (TM). Se ocupação do balde do condicionador é baixo (OB) e o descarte na fila EF é alto (DA), então a taxa do condicionador é incrementada em 2.

2. Se ocupação do balde do condicionador é média (OM) e o descarte na fila EF é baixo (DB), então a taxa do condicionador é reduzida em 1, por exemplo, se era taxa alta (TA) passa a taxa média (TM).Se ocupação do balde do condicionador é alta (OA) e o descarte na fila EF é baixo (DB), então a taxa do condicionador é reduzida em 2.

4.4 Controlador Convencional

Com o objetivo de validar nossa proposta, e tendo em vista que os resultados com a utilização do controlador fuzzy seriam melhores que a situação sem controlador, definimos, para comparação, um controlador digital convencional simples que executa um controle intuitivo, com as características mostradas a seguir:

- As variáveis amostradas são as mesmas utilizadas pelo controlador fuzzy.

- Se a média do retardo das três últimas amostras ultrapassar um determinado valor, calcula a inclinação da reta ajustada a esses três pontos e aplica a mesma inclinação à variável de saída peso da fila no escalonador.

- Se a média do retardo das três últimas amostras for menor que um determinado valor e o retardo da fila EF for pequeno, calcula a inclinação da reta ajustada a esses três pontos (que deve ser negativa) e aplica a mesma inclinação na variável de saída peso da fila no escalonador.

4.5 Topologia de Simulação

A aplicação voz sobre IP foi implementada com tráfego CBR e On-Off exponencial sobre protocolo UDP. O tráfego CBR é o pior caso para o desempenho da rede, por outro lado, o tráfego On-Off é mais próximo de uma conversação normal, sendo sua taxa média sensivelmente menor que no caso CBR. O tráfego de voz foi classificado em classe EF[19] e o tráfego concorrente, também CBR/UDP, foi classificado na classe BE. A topologia utilizada é apresentada na Figura 7, formando um domínio DiffServ com cinco nós, sendo dois de núcleo e três de borda (a quantidade real de fontes não é mostrada na figura).

A Figura 8 mostra a quantidade de fontes de tráfego de voz (classe EF) utilizadas durante o período de teste de 100 segundos. A variação de tráfego serviu para demonstrar o funcionamento do controlador nessa situação. Cada fonte de voz CBR foi definida com 64 Kbps, ou seja, um canal de voz PCM. No caso de fonte On-off, utilizamos uma taxa de 64 Kbps, com tempo de rajada de 600 ms e tempo de silêncio de 400 ms, representando uma taxa média de 25,6 Kbps.

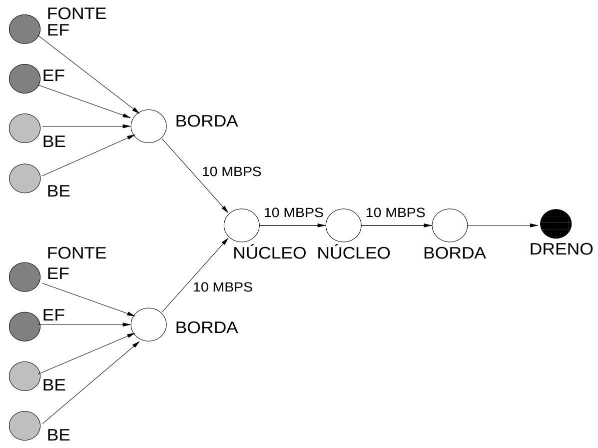
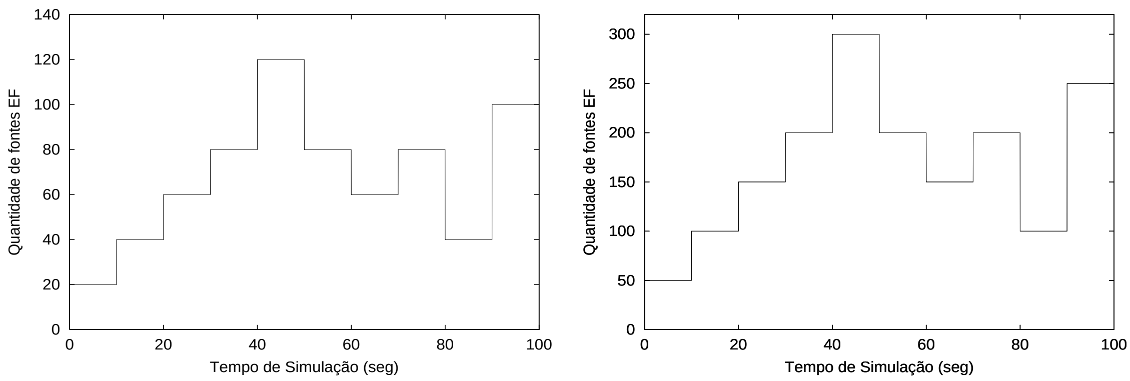


Figura 7: Diagrama de Simulação

O tamanho do pacote é de 576 bytes. Enquanto o tráfego CBR variou de 20 a 120 fontes, o tráfego On-Off variou de 50 a 300 fontes, exatamente porque a taxa média do tráfego On-Off é aproximadamente 2,5 vezes menor. Utilizamos 300 fontes BE concorrentes com taxa também de 64 Kbps. O retardo de cada enlace de 10 Mbps é de 5 ms. Todas as filas têm o tamanho de 400 pacotes representando um retardo máximo de aproximadamente 23 ms por nó.



(a) Fontes CBR

(b) Fontes On-Off Exponencial

Figura 8: Quantidade de fontes de tráfego EF durante a simulação

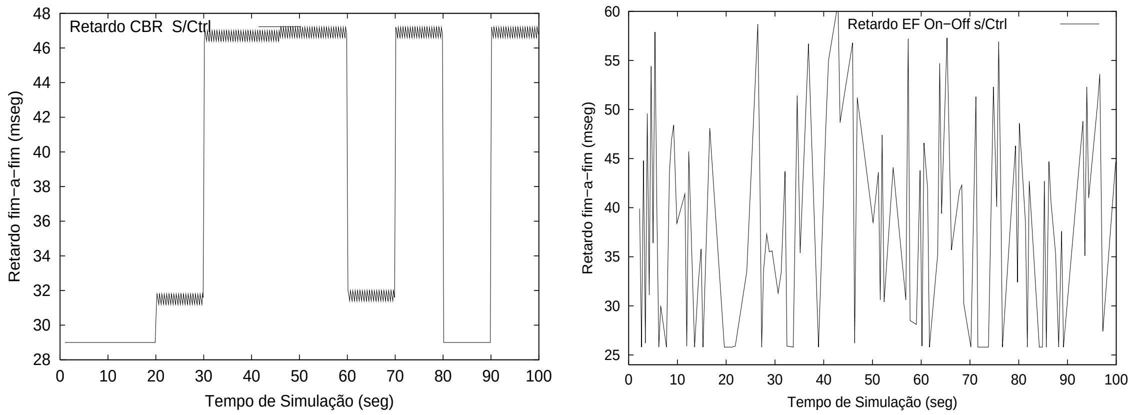
O modelo DiffServ utilizado nas simulações usou escalonador WRR, filas *Drop Tail* em ambas as classes e condicionador *Token Bucket* para os fluxos da classe EF (a classe BE não foi condicionada). A utilização de WRR estática para a aplicação sugerida, a transmissão de voz, já foi mostrada por Ziviani et. al.[13] e aqui o utilizamos com configuração dinâmica. O balde furado *(Token Bucke)*t é um condicionador simples, mas apresenta muito bom resultado para a aplicação sugerida.

5 Resultados Obtidos

As medidas de avaliação de desempenho avaliadas foram o retardo fim-a-fim e o descarte de um fluxo pertencente à classe EF. As medidas de variação de retardo *(jitter)* poderão ser deduzidas a partir do gráfico de retardo. Para cada medida, apresentamos os gráficos relativos ao tráfego CBR e On-off exponencial. Mostramos as curvas dos resultados em um domínio DiffServ sem controlador, com controlador convencional e com o controlador fuzzy. Todas as simulações iniciaram de uma configuração no escalonador com 50% da banda de saída para cada uma das classes EF e BE. Para eliminar medidas de desempenho com a rede vazia, a seqüência de medidas iniciou-se sempre 5 segundos após o inicio do tráfego.

O período de amostragem das variáveis de controle influencia diretamente a instabilidade do controlador, isto é, quando o período é longo, pode haver oscilações. Os resultados apresentados neste trabalho usaram o intervalo de 1,0 s entre as amostragens e a atuação do controlador mostrou-se eficiente. Caso seja reduzido o intervalo de atuação do controlador, digamos 0,1 s, há uma melhora significativa das medidas de retardo e perda quando ocorre uma variação de tráfego.

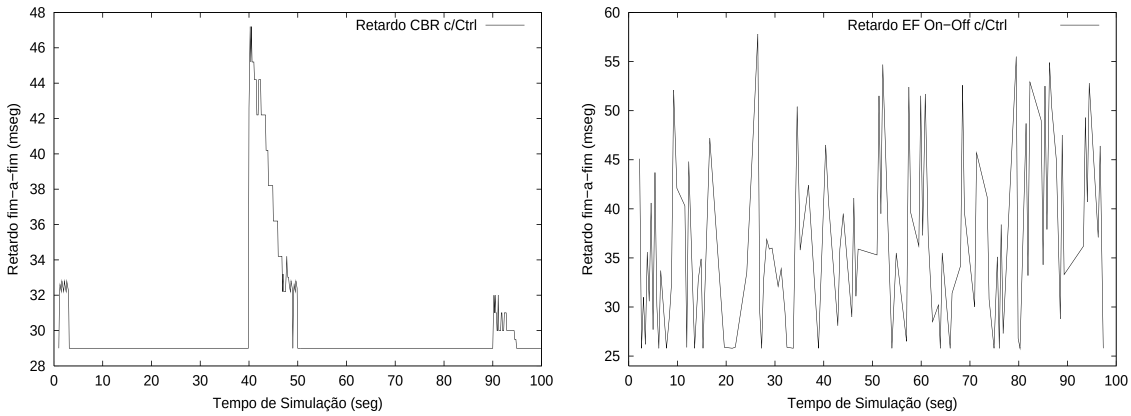
5.1 Retardo fim-a-fim da classe EF



(a) Tráfego CBR

(b) Tráfego On-Off

Figura 9: Retardo fim-a-fim da classe EF sem Controlador



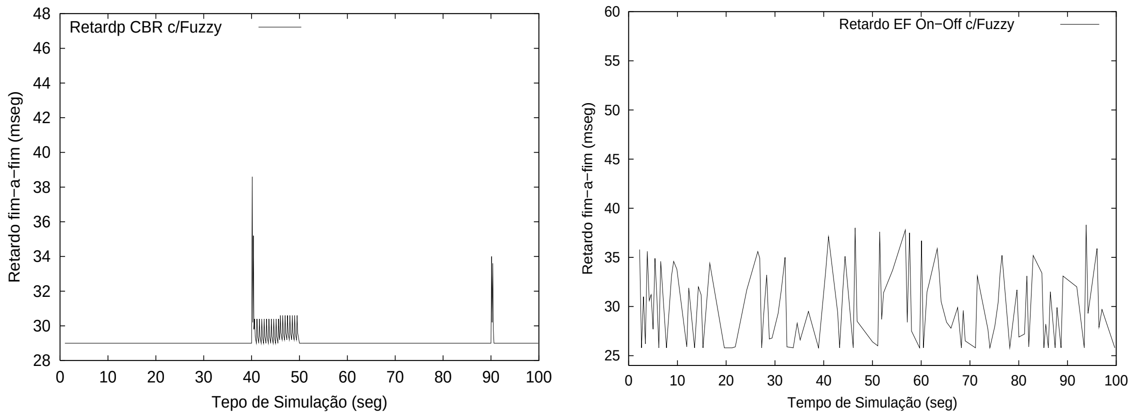
(a) Tráfego CBR

(b) Tráfego On-Off

Figura 10: Retardo fim-a-fim da classe EF com Controlador Convencional

O gráfico da Figura 9 mostra o retardo fim-a-fim, isto é, da fonte de origem até o destino, do tráfego de voz classificado na classe EF. Este gráfico exibe o resultado de um domínio DiffServ sem controlador. A Figura 9(a) representa o tráfego CBR e a Figura 9(b) o tráfego On-Off exponencial. Notamos, em 9(a), um patamar durante maior parte da simulação, indicando que a fila atingiu a capacidade máxima e os pacotes adicionais são descartados. No tráfego On-off, o retardo varia bastante ao longo da simulação em função da distribuição exponencial das fontes de tráfego que entram no domínio.

O gráfico da Figura 10 mostra o retardo fim-a-fim em um domínio DiffServ, com um controlador convencional. Notamos em 10(a) uma melhoria do retardo em relação ao caso sem controlador, o controlador corrige as variações quando o sistema atinge o limite, porém muito lentamente. No tráfego On-off, Figura 10(b), o retardo já apresenta uma melhoria no retardo médio em relação ao caso sem controlador.



(a) Tráfego CBR

(b) Tráfego On-Off

Figura 11: Retardo fim-a-fim da classe EF com Controlador fuzzy

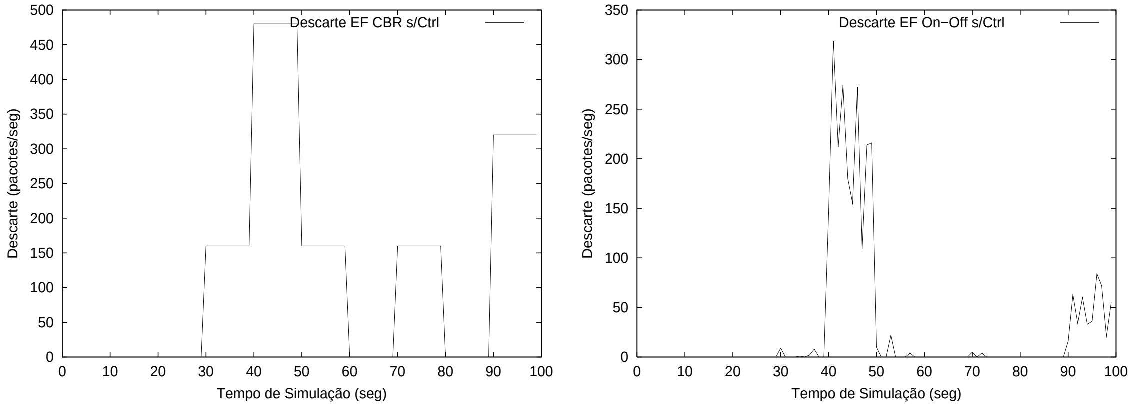
O gráfico da Figura 11 mostra o retardo fim-a-fim em um domínio DiffServ, com o controlador fuzzy. Notamos, em 11(a), uma melhoria do retardo em relação ao caso sem controlador e com controlador convencional. O controlador fuzzy corrige as variações rapidamente mantendo os valores de retardo próximos ao do retardo de propagação da rede (25 ms). O pico que surge nos tempos 40 s e 90 s são justificados pelo período de amostragem do controlador, que somente executa controle de correção dos parâmetros a cada 1 segundo. No tráfego On-off, Figura 11(b), o retardo já é bem inferior aos casos anteriores.

5.2 Descarte na classe EF

O gráfico da Figura 12 mostra a taxa de descarte a cada segundo do tráfego de voz da classe EF. Sem controlador, o descarte com tráfego CBR ocorre em todo o período em que a fila atinge o limite, conforme mostrado em 5.1. No caso do On-off, 12(b), o descarte só ocorre quando os recursos de rede se esgotam.

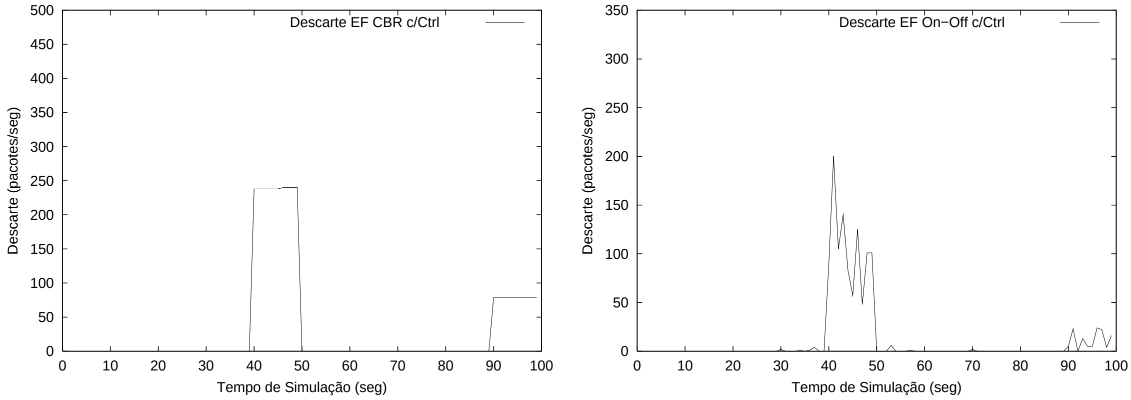
O gráfico da Figura 13 mostra o descarte com o controlador convencional. Notamos uma diminuição do descarte da classe EF para ambos os tráfegos, em relação ao caso anterior.

O gráfico da Figura 14 mostra o descarte quando usamos o controlador fuzzy. No caso de tráfego CBR, temos uma melhoria sensível na taxa de descarte e, no caso de tráfego On-off, as perdas caem a zero.



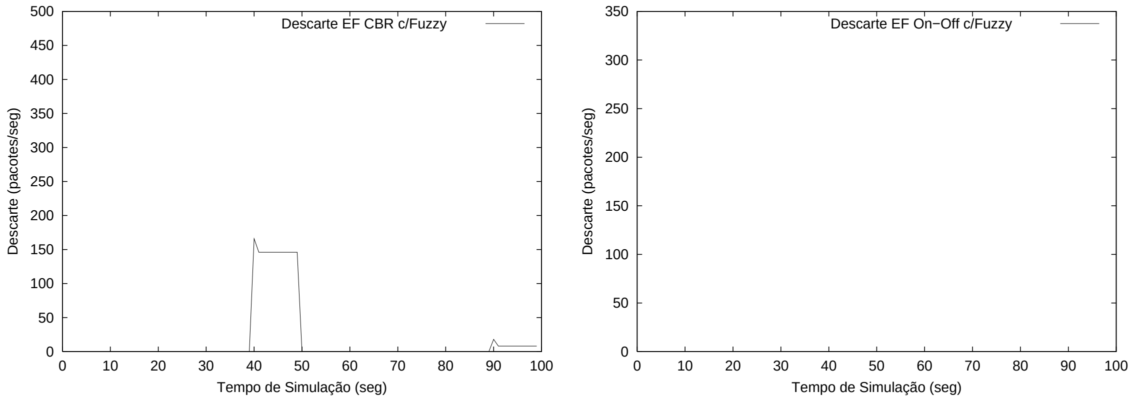
(a) Tráfego CBR (b) Tráfego On-Off

Figura 12: Descarte na classe EF sem Controlador



(a) Tráfego CBR (b) Tráfego On-Off

Figura 13: Descarte na classe EF com Controlador Convencional



(a) Tráfego CBR (b) Tráfego On-Off

Figura 14: Descarte na classe EF com Controlador fuzzy

6 Conclusão e Trabalhos Futuros

Apresentamos, neste artigo, um controlador fuzzy que reconfigura os parâmetros de roteadores, de modo a oferecer melhor qualidade de serviço dentro de um domínio DiffServ. A utilização de lógica fuzzy possibilitou tratamento eficaz das imprecisões e incertezas do tráfego de entrada no domínio. O controlador, por outro lado, tem complexidade baixa, não comprometendo a característica de escalabilidade do DiffServ .

O gerenciamento baseado em políticas permite que o controlador seja genérico, podendo definir o comportamento do sistema de acordo com as decisões administrativas do gerenciador do domínio. Além disso, o comportamento pode ser facilmente alterado em todo o domínio, durante o funcionamento normal do sistema.

As simulações realizadas para validar o modelo utilizaram exemplo com classes EF e BE. Apesar de conceitualmente simples, a avaliação de retardo e variação de retardo são dificultadas pela imprecisão e incerteza do tráfego que entra no domínio[8]. Os resultados obtidos através de simulação demonstram a viabilidade da proposta pela sensível melhoria nas medidas da qualidade de serviço, comparadas àquelas das situações sem controlador ou com controlador convencional simples. Os resultados tendem, ainda, a melhorar no caso de um dominio maior, com aumento da quantidade de nós e enlaces com gargalo, tendo em vista que o tráfego pode ser dividido por mais enlaces.

Os resultados obtidos com nós DiffServ simples, utilizando filas *Drop Tail*, escalonador WRR e condicionador balde furado *(Token Bucket)*, já mostraram grande melhoria nas métricas de QoS. Se forem utilizados mecanismos mais sofisticados, como, por exemplo, fila WRED e escalonador WFQ, os resultados, certamente, serão ainda melhores.

Como continuação do trabalho, será definido controlador que inclua suporte a outras classes DiffServ, como AF *(Assured Forwarding)*. Esta classe segue filosofia diferente, devendo o controlador tratar variáveis distintas das consideradas neste artigo. Assim, teremos um controlador completo, com capacidade de ajustar dinamicamente os parâmetros de todas as classes DiffServ, de acordo com as variações do tráfego e políticas especificadas.

Referências

[1] SLOMAN, M., “Policy Driven Management for Distributed Systems”, Journal of Network and Systems Management, Vol. 2, No. 4, pp. 333-360, 1994.

[2] RAJAN, R., VERMA, D., KAMAT, S., at al. “A Policy Framework for Integrated and Differentiated Services in the Internet”, IEEE Network Magazine. USA, pp. 36-41, September/October 1999.

[3] BLIGHT, D., HAMADA, T., “Policy-Based Networking Architecture for QoS Interworking in IP Management - Scalable Architecture for Large-Scale Enterprise Public Interoperation”,

[4] GUÉRIN, R., ORDA, A., “QoS-based Routing in Networks with Inacurate Information: Theory and Algorithms”, IEEE Infocom 97. Kobe, Japan, 1997

[5] Network Simulator - NS Version 2, http://www.isi.edu/nsnam/ns/

[6] PIEDA, P., ETHRIDGE, J., BAINES, M., et al.,”A Network Simulator Differentiated Services Implementation - Open IP”, Nortel Networks, http://www7.nortel.com:8080/CTL/

[7] DUARTE, O., Unfuzzy, http://ohm.ingsala.unal.edu.co/ogduarte/

[8] LORENZ, D., ORDA, A., “QoS Routing in Networks with Uncertain Parameters”, IEEE Infocom 98

[9] VASILAKOS, A., ANAGNOSTAKIS, K., “Evolutionary-Fuzzy Prediction for Strategic Inter-Domain Routing: Architecture and Mechanisms”, WCCI 98. Anchorage, USA, May 4-9 1998.

[10] LI, B., NAHRSTEDT, K., “A Control-Based Middleware Framework for Quality of Service Adaptions”, IEEE Journal on Select Areas in Communication, September 1997.

[11] LI, B., NAHRSTEDT, K., “Dinamic Reconfiguration for Complex Multimidia Application”, IEEE International Conference on Multimedia Computing and Systems 99, Vol 1, pp 165-170, July 1999.

[12] CHENG, R., CHANG, C., “Design of a Fuzzy Traffic Controler for ATM Networks”, IEEE/ACM Transaction on Networking, v.4 nº 3, pp 460-469, June 1996.

[13] STEVENS,M., WEISS, W., MAHON, H., at al., "Policy Framework", Internet Draft draft-ietf-policy-framework-00.txt, September 1999.

[14] MAHON, H., BERNET, Y., HERZOG, S., at al., "Requirements for a Policy Management System", Internet Draft draft-ietf-policy-req-02.txt, November 2000.

[15] DURHAM, D., BOYLE, J., COHEN, R., et al, ”The COPS (Common Open Policy Service) Protocol”, Request for Comments 2748, January 2000.

[16] ZIVIANI, A., REZENDE, J., DUARTE, O., “Tráfego de Voz em um Ambiente de Diferenciação de Serviços na Internet” , 17º Simpósio Brasileiro de Redes de Computadores. Salvador, Brasil, 25-28 de maio de 1999.

[17] ROSS, T., “Fuzzy Logic with Engineering Applications”, McGraw-Hill, New York, 1995.

[18] BLAKE,S., BLACK, D., ,CARLSON, M., et al, ”An Architecture for Differentiated Services”, Request for Comments 2475, December 1998.

[19] JACOBSON, V., NICHOLS, K., PODURI, K., ”An Expedited Forwarding PHB”, Request for Comments 2598, June 1999.