Implementação de Políticas de Gerenciamento através de lógica Fuzzy visando melhoria da Qualidade de Serviço (QoS) *

Marcial Porto Fernandez[1] , **Aloysio de Castro P. Pedroza**[2,3] , **José Ferreira de Rezende**[2]

[1] Instituto de Computação - Coordenação de Pós-Graduação
Universidade Federal Fluminense (UFF)
Rua Passo da Pátria, 156 - Bloco E - 3º andar
CEP 24.210-240 - Niterói - RJ - Brasil

[2]Universidade Federal do Rio de Janeiro (UFRJ)
Grupo de Teleinformática e Automação (GTA)
COPPE/PEE - Programa de Engenharia Elétrica
C.P. 68504 - CEP 21945-970 - Rio de Janeiro, RJ
Tel: (21) 2260-5010 Fax: (21) 2290-6626

[3]Departamento de Eletrônica EE-UFRJ

mfernandez@ic.uff.br,aloysio@gta.ufrj.br,rezende@gta.ufrj.br

Resumo. *O Gerenciamento Baseado em Políticas é uma técnica para coordenar a configuração de diversos equipamentos de uma rede, a partir de contratos administrativos(SLAs). Essas políticas abstratas são difíceis de serem interpretadas e implementadas pelos equipamentos de rede, que requerem informações absolutas, e a lógica fuzzy tem se mostrado eficiente para representar valores abstratos. Um controlador fuzzy pode implementar um mecanismo de provisionamento dinâmico, reconfigurando os nós de acordo com o tráfego entrante. Esse trabalho propõe uma metodologia de mapeamento de políticas em parâmetros fuzzy para melhorar a qualidade de serviço em um domínio DiffServ. As funcionalidades são demonstradas através de simulação de uma aplicação de Telefonia IP cruzando um domínio DiffServ.*

Abstract. *The policy based management is a technique to coordinate the configuration of several equipments in a network, from Service Level Agreements(SLAs). These abstract policies are difficult to be interpreted and implemented by network equipments, that requires absolute values, and fuzzy logic has been showing to be efficient to represent abstract values. A fuzzy controller implement a dynamic provisioning mechanism used to reconfigure all nodes according ingress traffic. This work proposes a methodology to map policies in fuzzy parameters to achieve the desired QoS in a DiffServ domain. The functionalities are demonstrated by simulation of a IP Telephony application crossing a DiffServ domain.*

*Esse trabalho foi realizado com recursos da UFRJ, CNPq e CAPES

1 Introdução

A Diferenciação de Serviços (DiffServ) [Blake et al., 1998] é uma proposta que visa oferecer garantias de qualidade de serviço (QoS) na Internet, consistindo em prover serviços diferenciados para as agregações de fluxos de dados. Soluções como IntServ possibilitam o controle por fluxo de dados, garantindo QoS para cada fluxo individualmente, porém, apresentam problemas de escalabilidade nos roteadores de núcleo. A arquitetura DiffServ é uma arquitetura escalável, oferecendo garantias para as diferentes classes, porém a qualidade do serviço pode sofrer violações quando ocorre congestionamento na agregação de vários fluxos em uma mesma classe, comprometendo a qualidade de serviço borda-a-borda. Como a Internet tem um comportamento aleatório, devemos estar preparados para detectar violações da QoS.

A necessidade de oferecer garantias de qualidade de serviço na Internet nos leva a mecanismos de provisionamento dinâmico. A característica aleatória da chegada de fluxos em diferentes classes de serviço obriga a utilização de alguma técnica de reconfiguração dinâmica dos mecanismos de provisionamento. Em virtude da complexidade desses mecanismos, a maioria das empresas de telecomunicações tem preferido superdimensionar os recursos para obter a QoS desejado. Esse procedimento, no entanto, apresenta um custo muito alto, tanto pela capacidade não utilizada na maioria do tempo (deve-se provisionar pelo pico) como pela dificuldade do planejamento, pois a estimativa de tráfego futuro tende a ser imprecisa.

Em trabalhos anteriores, propomos um controlador fuzzy que reconfigura dinamicamente os nós, fazendo que o provisionamento da rede se ajuste conforme o tráfego entrante. Essa proposta mostrou-se eficiente para melhorar a qualidade de serviço em um domínio DiffServ simples[Fernandez et al., 2001b] e em um domínio mais complexo com várias topologias aleatórias de 40 nós[Fernandez et al., 2001a]. A utilização do controlador fuzzy justifica-se pela não linearidade e ausência de um modelo matemático preciso para tratar estimativa de tráfego[Lee, 1990]. Comparado a um controlador convencional PD (Proporcional Derivativo), o controlador fuzzy apresenta vantagens significativas no tratamento de variáveis imprecisas. Para melhorar a eficiência do controlador fuzzy foram utilizadas técnicas baseadas em algoritmos genéticos que otimizam os parâmetros do controlador fuzzy[Herrera et al., 1995, Velasco and Magdalena, 1995].

O Gerenciamento Baseado em Políticas[Sloman, 1994] tem se mostrado uma técnica eficaz para coordenar a configuração de uma rede para obter a qualidade de serviço (QoS) desejada. A maioria dos trabalhos sobre Gerenciamento Baseado em Políticas focam na especificação de políticas e do modelo de informações das entidades onde as políticas serão aplicadas[DMTF, 1999, Moore et al., 2000]. Entretanto, pouco trabalho tem sido feito sobre como as entidades interpretam as políticas e definem comandos de configuração nos equipamentos reais. Além disso, interpretações diferentes de uma mesma política por equipamentos distintos podem causar instabilidades e falhas na operação do sistema.

Mostramos nesse trabalho a arquitetura do sistema de gerenciamento baseado em políticas que coordena o provisionamento dinâmico dos nós de um domínio DiffServ. Apresentamos uma proposta de mapeamento de políticas de gerenciamento em parâmetros de controlador fuzzy, que será utilizado para reconfigurar o provisionamento nos nós do domínio. Para validar a metodologia foi construído um controlador a par-

tir das políticas mapeadas em parâmetros fuzzy. Assim, foi realizada uma simulação desse protótipo, com avaliação de tempo de retardo, variação do retardo e descarte de fluxos de telefonia IP. Posteriomente, apresentamos uma avaliação do controlador fuzzy mediante a variação de alguns parâmetros.

Esse artigo encontra-se organizado da seguinte forma: a seção 2 apresenta a arquitetura do sistema e metodologia para construção do controlador; a seção 3 mostra a metodologia de mapeamento de políticas; a seção 4 mostra a implementação do protótipo seguindo a metodologia proposta; a seção 5 apresenta os resultados obtidos na simulação; e finalmente, a seção 6 apresenta as conclusões do trabalho e sugere temas para trabalhos futuros.

2 Apresentação do sistema

Com o objetivo de coordenar a configuração de uma rede com requisitos de qualidade precisamos de um sistema que mantenha o sistema coerente e que seja escalável assim como a arquitetura DiffServ. Para isso foi definido um controlador que implementa o provisionamento dinâmico nos nós a partir das políticas especificadas pelo operador de redes. Assim, apresentamos a metodologia para construção desse controlador.

2.1 Apresentação da arquitetura

Apresentamos na figura 1 a arquitetura do sistema proposto. Essa arquitetura tem como objetivo propiciar o melhor desempenho e manter a escalabilidade do sistema.

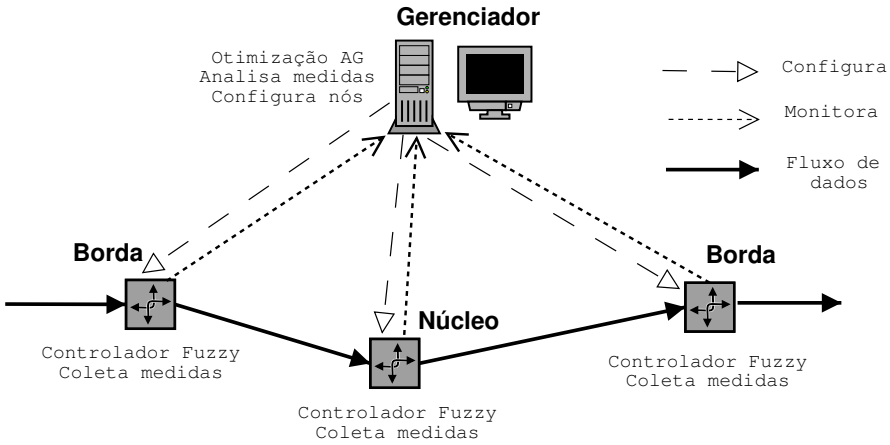


Figura 1: Arquitetura do sistema proposto.

Em cada nó do domínio, seja de borda ou núcleo, o controlador realiza medidas do estado atual, calcula o novo valor de configuração, utilizando um mecanismo de lógica fuzzy, e aplica o comando de controle no nó. Como a lógica fuzzy é computacionamente leve, pode ser executada em cada nó do domínio sem interferir muito no desempenho do roteador. O intervalo de operação desse controlador é da ordem de segundos, e portanto, um mecanismo computacionalmente pesado poderia impactar no desempenho global do sistema. Além disso, o nó é responsável por coletar as informações do estado do equipamento (retardo nas filas) e informá-las ao gerenciador,

utilizando um protocolo de gerenciamento de redes, por exemplo SNMP (*Simple Network Management Protocol*).

Algumas decisões exigem o conhecimento de um conjunto de nós (domínio), quando então o gerenciador de políticas realiza o cálculo e reconfigura todos os nós necessários. Um exemplo dessa situação é quando o retardo na classe mais prioritária no interior do núcleo aumenta e não existem mais recursos para alocar, obrigando a uma redução na taxa de entrada para não haver descarte no núcleo.

O gerenciador, único no domínio, é responsável por consolidar todas as informações colhidas pelos nós do domínio. Ele também realiza a otimização com algoritmo genético e reconfigura todos os nós regularmente, utilizando o protocolo COPS (*Common Open Policy Service*), por exemplo. Como o algoritmo genético exige uma quantidade maior de recursos computacionais, poderia interferir no desempenho dos roteadores, se fosse executado neles. Lembramos também que a otimização com algoritmo genético ocorre a intervalos maiores, da ordem de horas ou dias, portanto a escalabilidade pode ser mantida mesmo para grandes domínios.

2.2 Apresentação da metodologia

Apresentamos, na figura 2, o fluxograma da metodologia proposta. Nessa figura, os retângulos representam um procedimento da metodologia e os paralelogramos representam as interações com os usuários do sistema ou com os equipamentos de rede. Todos os procedimentos da metodologia são realizados no equipamento gerenciador pois são operações de configuração e otimização que não influem direatamente na comutação de pacotes, realizada pelos roteadores. Os operadores de rede de telecomunicações devem cumprir métricas de QoS especificadas em contrato, por exemplo, que o retardo máximo da rede seja de 100 ms.

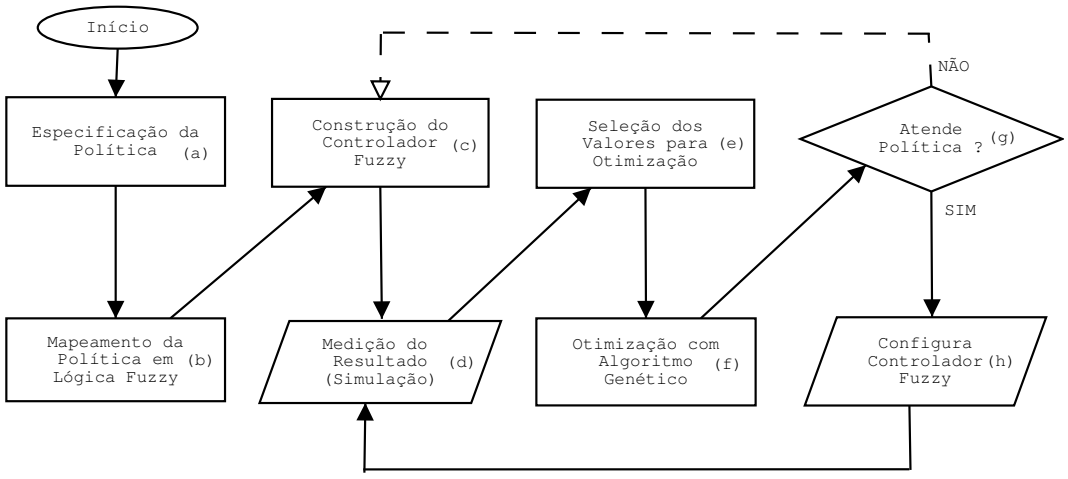


Figura 2: Fluxograma da metodologia proposta.

O ponto de partida do presente trabalho são as especificações de políticas administrativas com base nos requisitos de QoS, mostrado na figura 2(a). O detalhamento dessa fase é apresentado na seção 3.2. A etapa seguinte, figura 2(b), mostra a definição do controlador a partir dessas políticas administrativas, por isso é apresentada uma

proposta de mapeamento de políticas em parâmetros do controlador fuzzy na seção 3.3.

A próxima etapa é a construção do controlador fuzzy, mostrado na figura 2(c). Essa etapa, apresentada em [Fernandez et al., 2001b], define os parâmetros do controlador fuzzy, como funções de pertinência e base de regras, a partir da especificação de política mapeada na etapa anterior. Uma vez construído o controlador, podemos realizar a configuração dos equipamentos da rede e avaliar o resultado. Essa etapa, mostrada na figura 2(d), pode-se coletar os valores de entrada e saída no controlador fuzzy e as medidas de desempenho desejadas como, por exemplo, retardo ou descarte de pacotes. Essa etapa foi detalhada em [Fernandez et al., 2002].

De posse desses valores, podemos escolher todas as combinações de valores de entrada e saída que maximizam as medidas de desempenho desejadas. Essa etapa é mostrada na figura 2(e). A etapa seguinte, mostrada na figura 2(f), consiste na otimização dos parâmetros fuzzy através do algoritmo genético, utilizando como referência de função objetivo os valores selecionados no item anterior. O detalhamento dessas etapas foram mostradas em [Fernandez et al., 2002].

A otimização por algoritmo genético pode distorcer as funções de pertinência e regras do controlador, fazendo com que o resultado do controlador deixe de cumprir as políticas originais. Na etapa 2(g), as novas funções de pertinência e regras são avaliadas com a especificação das políticas e, caso estejam em desacordo, o controlador fuzzy precisa ser redefinido na etapa 2(c), mostrada com uma linha tracejada. Caso o controlador produza um resultado coerente com as políticas originais, ele pode ser usado nos equipamentos para funcionamento normal, conforme indicado na etapa 2(h).

A metodologia admite que o processo de otimização seja realizado continuamente, adaptando os controladores fuzzy de acordo com as mudanças ocorridas na rede (ativação ou desativação de um canal, alteração no padrão de tráfego gerado pelos usuários etc.). Como normalmente essas mudanças são pequenas e eventuais, o algoritmo genético é muito eficiente na otimização, pois aproveita o conhecimento passado. O processo de adaptação é mostrado na figura 2, com uma linha cheia, considerando que as políticas administrativas se mantenham inalteradas. Caso contrário, deve-se iniciar a metodologia a partir da etapa inicial (figura 2(a)).

3 Implementação de políticas de gerenciamento

O Gerenciamento Baseado em Políticas, proposto por Sloman[Sloman, 1994], tem se mostrado uma técnica eficaz para coordenar a configuração de uma rede para obter a qualidade de serviço (QoS) desejada. Podemos definir uma **política** como uma *regra que direciona as opções de comportamento de um sistema de gerenciamento.* Para representar as políticas utilizamos a linguagem Ponder[Damianou et al.,], apresentada a seguir.

3.1 Linguagem de especificação de políticas: Ponder

A linguagem *Ponder* foi proposta por Damianou *et al.*[Damianou et al., 2001] para especificar textualmente políticas de gerenciamento, de acordo com as propostas de Sloman [Sloman, 1994] e Lupu[Lupu and Sloman, 1997, Lupu and Sloman, 1999]. É uma linguagem declarativa orientada a objetos e oferece ao usuário uma interface

simples para especificação de políticas, aproximando-se o máximo possível de regras de políticas abstratas. A linguagem Ponder foi escolhida para esse trabalho pois atendeu às necessidades requeridas e as ferramentas de auxílio se mostraram eficientes para implementar o protótipo.

Essa linguagem define quatro políticas básicas: *política de autorização*, que pode ser positiva (**auth+**), permitindo o acesso a um determinado recurso, ou negativa (**auth-**), proibindo o acesso; *política de obrigação* (**oblig**), que exige a execução de determinada ação (previamente autorizada);*política de proibição* (**refrain**), que proibe a execução de uma ação; e *política de delegação* (**deleg**), que permite a um sujeito delegar a outro o controle sobre determinado objeto.

3.2 Especificação das políticas de QoS

A partir dos requisitos administrativos, podemos especificar a política de gerenciamento. Em nosso exemplo, definimos a política de que toda prioridade deve ser dada à classe EF*(Expedited Forwarding)*, isto é, a classe BE*(BE - Best Effort)* será reduzida sempre que houver queda na qualidade da classe EF. Foram definidos dois controladores: um para o escalonador, existente em todos os nós, e um para o condicionador, existente apenas nos nós de borda.

O código 1 mostra um trecho da especificação da política do escalonador em *Ponder*. As linhas 4 a 8 definem os valores máximos dos parâmetros de QoS para uma determinada classe de serviço (EF). Podemos ver que o escalonador pode variar de 10% a 90% da banda de saída, o retardo máximo é de 100 ms e o descarte máximo é de 20%. Da linha 11 à 15, são estabelecidas as restrições baseadas nos valores previamente definidos. Nas linhas 17 a 20, são definidos os eventos de disparo das ações.

Código 1: Exemplo de especificação Ponder do escalonador

```
// Especificacao do Controlador do Escalonador
//
// Define limites minimos e maximos permitidos para o escalonador
const
    minsched = 0.10;  // Escalonador minimo 10%
    maxsched = 0.90;  // Escalonador maximo 90%
    MaxDelay = 100;   // Delay maximo 100 ms
    MaxDrop = 0.2;    // Descarte maximo 20%

// Define condicoes de restricao
constraint
    bwMin = bwShare < minsched;
    bwMax = bwShare > maxsched;
    bwIncrease = bwShare < maxsched;
    bwDecrease = bwShare > minsched;

event   // Definicao dos eventos
    lowdelay = 0.3 * MaxDelay
    mediumdelay = 0.5 * MaxDelay
    highdelay = 0.7 * MaxDelay

    // Autoriza aumentar escalonador se menor que maxsched
    auth+ increaseSchedulerAuth {
        subject s;
```

```
        target t;
26      action increaseBW();
        when bwIncrease;
28  }
    // Obriga aumentar 1 unidade escalonador quando retardo do EF e' ↳
        medio
30  oblig increaseScheduler1 {
        subject s;
32      target t;
        on EF.mediumdelay;
34      do increaseBW(level);
    }
```

A partir desse ponto, as políticas serão executadas pelo sujeito **s** e aplicadas ao alvo **t**. Nas linhas 23 a 28, é especificada uma política autorizando o controlador a executar o aumento da banda (*increaseBW()*), quando a condição particionamento da banda atual (*bwIncrease*) for menor que o máximo permitido (*maxsched*). Nas linhas 30 a 35, é definida uma política determinando que a ação de aumentar a banda (*increaseBW()*), previamente autorizada, seja executada quando o evento retardo na classe EF for médio.

3.3 Mapeamento de especificação de políticas em parâmetros de controlador fuzzy

A especificação de políticas traduz uma decisão administrativa em comando do sistema de gerenciamento. Por serem as regras de políticas abstratas e próximas da percepção humana, é muito difícil mapeá-las em regras computacionais, absolutas e exatas por natureza. A lógica fuzzy tem a característica de tratar variáveis semânticas com certo grau de imprecisão. Por isso, é quase intuitiva a aproximação das regras de especificação de políticas de gerenciamento dos atributos de um controlador fuzzy.

A linguagem *Ponder* permite especificar vários elementos de políticas de gerenciamento de forma textual. As funções restrição e evento podem ser representadas por funções de pertinência, enquanto as funções de comando podem ser representadas pela base de regras.

3.3.1 Representação do comando constraint e auth

Os comandos **constraint** e **auth-** indicam o limite de restrição de uma autorização, geralmente definindo um valor mínimo ou máximo para certa função. A associação em operação fuzzy consiste em estabelecer o valor defuzificado da variável de saída igual ao valor desejado.

Apresentamos, no código 2, uma especificação usando os comandos **constraint** e **auth-**. Nesse caso, a restrição é que a banda mínima do escalonador deve ser 0.10 (ou seja, 10% da banda total). O menor valor atingido por uma função de pertinência é o centro de gravidade (caso este seja o método de defuzificação utilizado) do menor valor semântico da função. O comando **auth-** indica a não autorização de executar o comando de reduzir a banda do escalonador, caso a banda atual seja menor ou igual ao valor mínimo.

A função de pertinência do controlador fuzzy correspondente a essa especificação

é mostrada na figura 3. Observe que o centro de gravidade do valor semântico ”PB” é 0.1. Assim, quando o resultado semântico for apenas ”PB” (que é o pior caso) o valor defuzificado será 0.1. Poderíamos fazer um mapeamento semelhante para a função peso máximo, que associaria o valor semântico ”PA” com o valor defuzificado 0.9.

Código 2: Mapeamento do comando constraint

```
constraint
    bwmin = bwShare <= 0.10 ;
inst
    auth- schedulerMin {
        subject s ;
        target sched ;
        action sched.reduceBW() ;
        when bwMin ;
    }
```

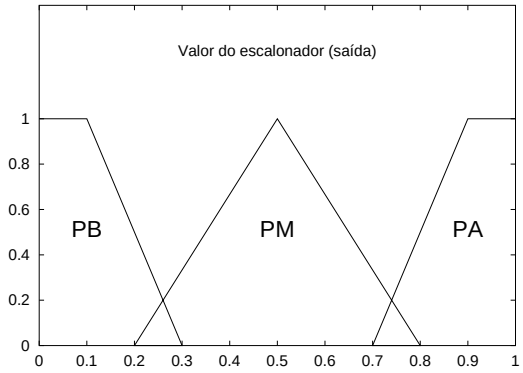


Figura 3: Função de pertinência associada a um comando constraint

3.3.2 Representação do comando event

O comando **event** lista os eventos que disparam comandos de obrigação (**oblig**) ou proibição (**refrain**). Uma ação pode ter vários eventos possíveis, conforme a política desejada. O mapeamento desse comando é uma função de pertinência de uma variável com seus valores semânticos. Assim, o comando **event** exprime a descrição de uma função de pertinência.

Apresentamos, no código 3, uma especificação usando um comando **event** e **oblig**. Atribuímos aos eventos RB, RM e RA, respectivamente, Retardo Baixo, Retardo Médio e Retardo Alto, um valor de disparo. Esses eventos serão utilizados no comando **oblig** como condição de disparo, na linha 10, e da ação *sched.increaseBW*, na linha 11.

Código 3: Mapeamento do comando event

```
event
    RB = 0.1 ;  // Retardo Baixo
    RM = 0.5 ;  // Retardo Médio
    RA = 0.9 ;  // Retardo Alto

inst
    oblig aumentaEscalonador {
        subject s ;
        target sched ;
        on classeEF.RA ;
        do sched.increaseBW()
    }
```

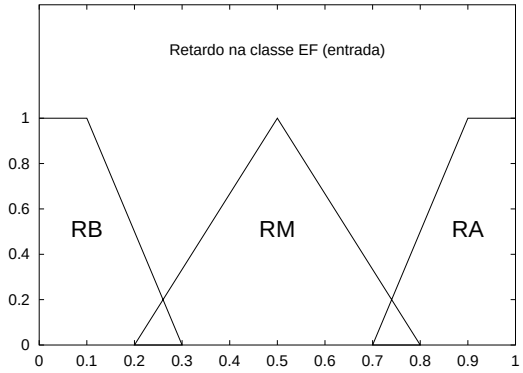


Figura 4: Função de pertinência associada a um comando event

A função de pertinência correspondente a essa especificação é mostrada na figura 4. Observe que cada valor de evento será associado a um valor semântico da função de pertinência. Podemos notar que a especificação *Ponder* atribui um valor absoluto,

sendo o critério de disparo atender ou não a esse valor. Para ser possível o mapeamento, consideramos o valor absoluto como o valor central dos valores semânticos e arbitramos a forma geométrica. O valor exato dessas variáveis não são importantes, pois poderão ser alterados no processo de otimização.

3.3.3 Representação do comando oblig

O comando **oblig** indica a obrigação de execução de uma ação caso uma determinada condição seja atendida (evento). O mapeamento, portanto, é quase que imediato. O parâmetro **on** do comando **oblig** é mapeado na condição do comando **if** da base de regras, e o parâmetro **do** é mapeado na ação do comando **then**.

Apresentamos, no código 4, uma especificação usando o comando **oblig**. Consideramos as mesmas função de pertinência apresentada na figuras 4. Nesse comando, caso ocorra o evento retardo médio na classe EF (*classeEF.RM*), é disparada a ação de aumentar a banda do escalonador (*sched.increaseBW()*).

Código 4: Mapeamento do comando oblig

```
inst
  oblig aumentaEscalonador {
    subject s ;
    target sched ;
    on classeEF.RM ;
    do sched.increaseBW()
  }
```

O código 5 apresenta uma descrição JFS[Mortensen, 1998] da base de regras do controlador fuzzy, a partir da especificação de política apresentada no código 4. Podemos observar que o mapeamento de um comando increaseBW() foi desdobrado em várias regras *if* <*condição*> *then* <*ação*> para abranger todos os valores semânticos da função de pertinência. A partir da ação *increaseBW()* em *Ponder*, definimos três comandos JFS, por ser exigida a especificação de ação para cada valor semântico de entrada.

Código 5: Código JFS mapeado a partir com comando oblig

```
program
    if classeEF RM and sched PB then sched PM ;
    if classeEF RM and sched PM then sched PA ;
    if classeEF RM and sched PA then sched PA ;
```

4 Implementação do Protótipo

A partir da metodologia para construir o controlador de provisionamento apresentada na seção anterior, podemos descrever, nesta seção, o ambiente de simulação e a implementação do mecanismo de controle para nosso protótipo. O objetivo é validar a metodologia proposta, para definição de um controlador que implemente uma especificação de políticas.

4.1 Ambiente de Simulação

A metodologia utilizada para validação do modelo proposto foi a de simulação, com a plataforma do Network Simulator (*ns*), versão 2.1b8[McCanne and Floyd, 1998]. A ferramenta para especificação e verificação de políticas foi o*Ponder Toolkit* [Damianou et al.,], constituída por um editor de políticas e um compilador da linguagem. Foi utilizada a versão 1.0.1 do *Ponder Policy Editor* e a versão 0.2.1 do *Ponder Compiler*. O controlador fuzzy foi desenvolvido com a ferramenta JFS, de Mortensen[Mortensen, 1998]. Essa ferramenta oferece um ambiente para desenvolvimento do protótipo (especificação das funções de pertinência, regras de inferência e defuzificador), além de permitir verificação inicial do modelo especificado. Após o desenvolvimento do modelo, é gerada biblioteca em código C, que implementa o controlador, sendo, então, integrada ao simulador *ns*.

4.2 Topologia de simulação

A topologia utilizada para a simulação, é apresentada na figura 5. Essa topologia forma um domínio DiffServ com cinco nós, sendo dois de núcleo e três de borda. Podemos notar que existem dois pontos de congestionamento, o primeiro entre nós de borda e do núcleo e outro entre dois nós de núcleo.

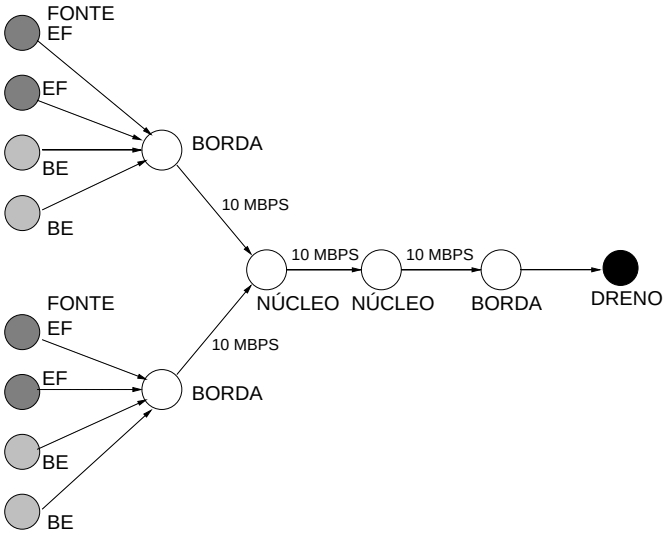


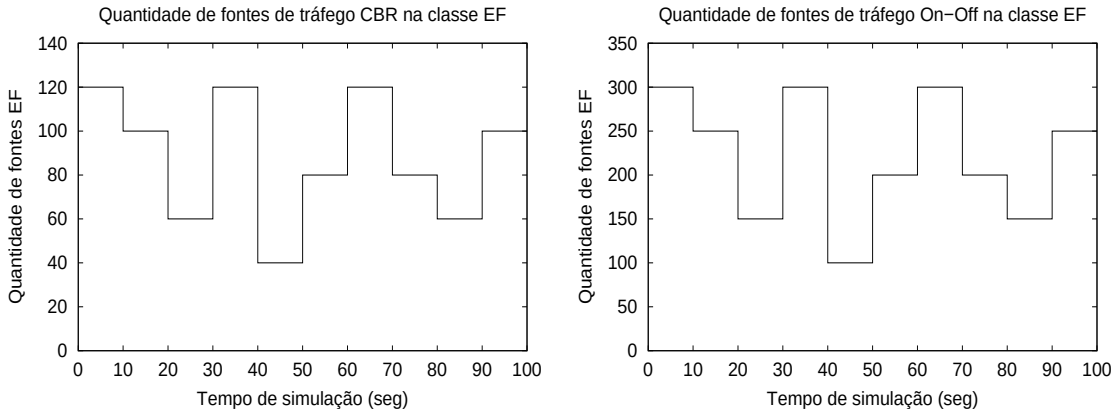
Figura 5: Topologia de simulação simples.

4.3 Modelo de tráfego de simulação

A aplicação de Telefonia IP foi implementada com tráfegos CBR e On-Off exponencial, sobre protocolo UDP. O tráfego CBR tem comportamento determinístico e exige mais banda da rede, enquanto o tráfego On-Off com distribuição exponencial é mais próximo de uma conversação normal. No entanto, a taxa média das fontes On-Off é sensivelmente menor que no caso CBR, por isso adicionamos mais fontes de tráfego On-Off para se obter o congestionamento desejado. O tráfego de voz foi direcionado para classe EF[Jacobson et al., 1999] e o tráfego concorrente para classe BE, sendo ambos CBR/UDP.

A figura 6 mostra a quantidade de fontes de tráfego de voz (classe EF), utilizada 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 400 ms e tempo de silêncio de 600 ms, representando uma taxa média de 25,6 Kbps. O tamanho do pacote escolhido foi de 576 bytes, que corresponde a 91,7% dos pacotes de um codificador G.711 (PCM) a 64 Kbps [Hsiung et al., 1999].



(a) Fontes CBR (b) Fontes On-Off Exponencial

Figura 6: Quantidade de fontes de tráfego EF durante a simulação

Enquanto o tráfego CBR variou de 40 a 120 fontes, o tráfego On-Off variou de 100 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 tempo de propagação de cada enlace de 10 Mbps é de 5 ms. Todas as filas têm o tamanho de 50 pacotes, representando um retardo máximo de aproximadamente 23 ms por nó.

4.4 Controlador convencional

Com o objetivo de validar nossa proposta, e tendo em vista que os resultados com a utilização de qualquer controlador seriam melhores que a situação sem controlador, definimos, para comparação, um controlador PD (Proporcional e Derivativo). A idéia desse controlador é guardar as últimas três medidas de retardo da classe EF, ajustar uma reta a esses pontos. A inclinação dessa reta é aplicada ao peso do escalonador, aumentando ou reduzindo conforme a inclinação da reta.

5 Resultados

Nesta seção mostramos as tabela de percentil de retardo, percentil da variação do retardo e taxa de descarte em três situações: sem a utilização de controlador, com controlador convencional e com o controlador fuzzy. Todas as simulações iniciam com alocação de 50% da banda de saída para cada classe. Para eliminar medidas com a rede sem tráfego, iniciamos as medidas após 5 segundos do início da simulação.

Os resultados apresentados aqui usaram o intervalo de 1 s entre as amostragens e, conseqüentemente, o intervalo de atuação do controlador. Apresentamos, nas seções 5.1 e 5.2, uma discussão sobre o comportamento dos diversos mecanismos de controle com a variação do intervalo de operação do controlador e do tamanho do pacote.

A primeira medida avaliada é o retardo fim-a-fim de pacotes pertencentes à classe EF, desde a fonte até o destino do tráfego. A tabela 1 apresenta os valores de medidas de retardo e variação de retardo médio e percentil 50, 90 e 95. Mostramos os resultados de simulação de tráfego CBR e On-Off sem o uso de controlador, usando o controlador convencional e usando o controlador fuzzy.

Tabela 1: Retardo e variação do retardo da classe EF (ms)

Tráfego	Média		Percentil 50		Percentil 90		Percentil 95	
	Retardo	Variação	Retardo	Variação	Retardo	Variação	Retardo	Variação
CBR S/Ctrl	43.4	1.7	43.0	0.4	65.8	0.4	75.3	7.3
CBR Conven.	33.6	1.5	30.3	0.4	58.8	0.4	71.3	6.0
CBR Fuzzy	19.3	1.0	17.3	0.4	34.9	0.4	41.1	0.4
OO S/Ctrl	17.9	1.2	16.6	0.3	33.2	3.6	36.9	5.7
OO Conven.	14.9	1.1	14.7	0.2	27.1	3.3	32.3	4.6
OO Fuzzy	5.6	0.5	3.7	0.1	11.7	1.3	12.4	1.9

A tabela 2 apresenta o descarte de pacotes nas classes EF e BE, na topologia simples. Mostramos os resultados de simulação de tráfego CBR e On-Off sem o uso de controlador, usando o controlador convencional e usando o controlador fuzzy.

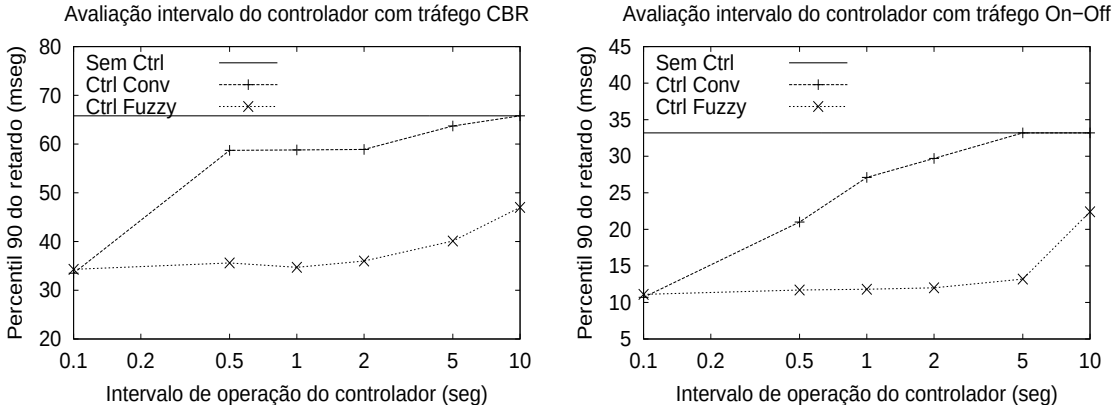
Tabela 2: Descarte

Controlador	EF (CBR)	BE (CBR)	EF(On-Off)	BE (On-Off)
Sem Ctrl	0.0198	0.0766	0.0379	0.1388
Convencional	0.0125	0.0843	0.0272	0.1439
Fuzzy	0.0068	0.0795	0.0027	0.1486

Podemos observar que há uma melhora na QoS e redução no descarte da classe EF com o controlador fuzzy comparado às situações sem controlador e com controlador convencional. Obviamente, quando se reduz a taxa de descarte da classe EF, provocamos um aumento na taxa da classe BE, pois a rede está com sua capacidade esgotada. No entanto, podemos observar que o uso do controlador fuzzy diminui a taxa de descarte agregada, isto é, a soma das taxas das classes EF e BE é menor que as demais taxas agregadas, produzindo um melhor desempenho global.

5.1 Avaliação do intervalo de operação do controlador

O primeiro parâmetro avaliado foi o intervalo de operação do controlador que é mostrado na figura 7. A figura 7(a) mostra a avaliação do retardo na classe EF para tráfego CBR e a figura 7(b) mostra o retardo para tráfego On-Off. Medimos o resultado de percentil 90 do retardo variando o intervalo de operação do controlador em 0,1 s, 0,5 s, 1 s, 2 s, 5 s e 10 s. O tamanho do pacote foi mantido fixo em 576 bytes. Para efeito de comparação, indicamos com uma linha contínua o percentil 90 do retardo da situação sem controlador.



(a) Tráfego CBR (b) Tráfego On-Off

Figura 7: Avaliação do intervalo de atuação do controlador

Podemos observar que a resposta do controlador fuzzy é melhor para qualquer intervalo de operação, sendo praticamente constante para intervalos até 2 segundos e aproximadamente linear para intervalos a partir de 5 s. Esse comportamento é justificado pela variação do perfil do tráfego a cada 10 segundos (vide figura 6). Somente a partir desse intervalo (10 s), podemos notar uma degradação na resposta do controlador fuzzy.

Podemos também notar que a diferença do retardo do tráfego On-Off obtido pelo controlador fuzzy, comparado com o caso sem controlador, é maior do que no caso com tráfego CBR, mostrando a maior eficiência do controlador fuzzy para tratar tráfego On-Off. O critério para a escolha do valor 1 segundos para intervalo de operação do controlador foi o equilíbrio entre resultado da QoS obtida e a sobrecarga computacional para ambos os controladores.

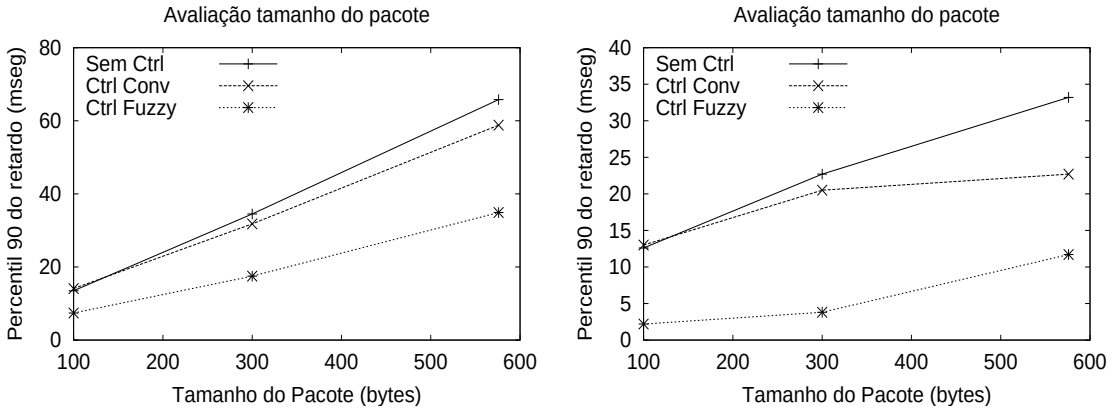
Outra conclusão desses resultados é que o controlador fuzzy implementa um controle mais eficiente que o convencional. Mesmo reconhecendo a maior necessidade de recursos computacionais exigidos pelo controlador fuzzy, ele pode ser aplicado em intervalos maiores do que o controlador convencional. Assim, podemos indicar, pelo menos, um equilíbrio na necessidade de recursos computacionais em ambos os controladores.

5.2 Avaliação da variação do tamanho do pacote IP

O segundo parâmetro avaliado foi o tamanho do pacote, mostrado na figura 8. Variamos o tamanho do pacote desde 100 bytes, correspondente ao tráfego produzido por um codificador H.323, até 576 bytes, correspondente ao tráfego produzido por um codificador G.711. Os valores medidos foram o percentil 90 do retardo para pacotes com 100, 300 e 576 bytes. O tempo de operação do controlador foi de 1 segundo. Como a aplicação em análise é telefonia IP não avaliamos pacotes maiores.

Na avaliação com tráfego CBR, mostrada na figura 8(a), podemos notar que com pacotes pequenos (100 bytes) os retardos estão próximos, apesar do controlador fuzzy já mostrar uma pequena superioridade. A justificativa para o retardo ser baixo

com pacotes pequenos em qualquer controlador é o melhor intercalamento dos pacotes no escalonador. Com o aumento do tamanho dos pacotes o retardo aumenta quase linearmente. Para os tamanhos de pacote avaliados, o percentil 90 do retardo obtido com controlador fuzzy sempre foi inferior aos retardos obtidos pelo controlador convencional e no caso sem controlador, nesta ordem.



(a) Tráfego CBR (b) Tráfego On-Off

Figura 8: Avaliação da influência do tamanho dos pacotes

Na avaliação com tráfego On-Off, mostrada na figura 8(b), podemos notar que o retardo obtido com o controlador fuzzy foi sempre inferior aos retardos obtidos com controlador convencional e sem controlador. Observamos também que o retardo, nesses dois últimos casos, é semelhante para pacotes pequenos, devido ao melhor intercalamento dos pacotes, conforme explicado anteriormente. Com pacotes maiores que 300 bytes, os retardos com controlador convencional são inferiores aos obtidos sem controlador.

Notamos também que a diferença do retardo com tráfego On-Off obtido pelo controlador fuzzy comparado com o controlador convencional e sem controlador, é maior que no caso com tráfego CBR, mais uma vez mostrando a melhor eficiência do controlador fuzzy para tratar tráfego On-Off. Finalmente, concluímos que o resultado obtido com o controlador fuzzy é melhor que o controlador convencional e sem controlador para qualquer tamanho de pacote.

6 Conclusão e Trabalhos Futuros

Apresentamos nesse trabalho, uma arquitetura de sistema de gerenciamento para manter a QoS em uma arquitetura DiffServ. Foi proposta, também, uma metodologia para realizar o mapeamento de políticas de gerenciamento em parâmetros de controlador fuzzy. Esse controlador implementa um mecanismo de provisionamento dinâmico, que coordenadas pelo sistema de gerenciamento, melhora a QoS no âmbito do domínio.

Quando variamos o intervalo de operação do controlador mostramos que, com intervalo pequeno, o resultado do controlador fuzzy é apenas um pouco melhor que o do controlador convencional, tornando-se mais significativo em intervalos maiores. O

controlador fuzzy mostrou-se mais eficiente que o convencional para qualquer intervalo de operação. Outra observação é que o intervalo ideal é proporcional ao tempo médio das conexões dos fluxos que cruzam o domínio. Comparando os resultados de tráfego CBR e On-Off notamos que o controlador fuzzy apresenta-se mais eficiente com tráfego On-Off, ou seja, tráfegos encontrados em uma rede real.

Quando variamos o tamanho do pacote, observamos que, com pacotes pequenos, as medidas de QoS são semelhantes para todas as situações. Esse fato, já conhecido nas redes ATM, demonstra que reduzir os tamanhos dos pacotes é uma medida eficaz para melhorar a QoS. Com o aumento do tamanho do pacote, observamos uma degradação na QoS em todas as situações, porém o controlador fuzzy continua apresentando o melhor resultado. Novamente podemos observar que o controlador fuzzy produz um melhor resultado comparativo quando trata tráfego On-Off em relação ao tráfego CBR, confirmando sua melhor eficiência para tratar tráfegos reais.

Finalmente, podemos concluir que mecanismos de provisionamento dinâmico apresentam vantagens em manter QoS, quando ocorre variações normais nos fluxos de tráfego. A atitude usual de superprovisionamento possibilita a manutenção da qualidade, porém o custo para manter essa infra-estrutura extra é alto. Mostramos também que a lógica fuzzy se mostrou adequada para tratar tráfegos com alta variação, mais comuns nas situações reais. A arquitetura proposta permitiu a construção de um sistema escalável e eficiente. A utilização de gerenciamento baseado em políticas possibilitou a facilidade da especificação dos requisitos de QoS desejados.

Como continuação do trabalho, deve-se realizar a avaliação de desempenho, tanto nos equipamentos de rede como no gerenciador central. Como todas as avaliações foram realizadas em simulação, não foi possível avaliar o impacto dos novos mecanismos adicionados à rede. No caso dos equipamentos de rede, a única função nova é o controlador fuzzy, considerando que o equipamento já disponha de protocolo de gerenciamento SNMP/COPS. Como esse controle é realizado com frequência alta (da ordem de segundos), é necessário verificar se não é prejudicial ao desempenho do equipamento.

Uma proposta para melhorar o desempenho de controlador fuzzy é implementá-lo em hardware (silício), no lugar da forma tradicional em software. Essa proposta, feita por Catania *et al.*[Catania et al., 1996] demonstra que a implementação de controlador fuzzy em silício torna praticamente imperceptível o impacto no desempenho do equipamento. No entanto, essa metodologia não pode ser implementada diretamente em nossa aplicação, porque o controlador fuzzy deve ser reconfigurado durante a operação da rede. Assim, a utilização dessa metodologia em nossa proposta exigiria a utilização de algum mecanismo de ”reconfiguração de hardware”.

Referências

Blake, S., Black, D., and Carlson, M. (1998). An architecture for differentiated services. RFC 2475.

Catania, V., Ficili, G., Palazzo, S., and Panno, D. (1996). A comparative analysis of fuzzy versus conventional policing mechanisms for atm networks. *IEEE/ACM Transactions on Networking*, 4(3):449–459.

Damianou, N., Dulay, N., Lupu, E., and Sloman, M. Ponder Policy Specification Language. http://www-dse.doc.ic.ac.uk/policies/ponder.shtml.

Damianou, N., Dulay, N., Lupu, E., and Sloman, M. (2001). The Ponder policy specification language. In *Policy 2001: Workshop on Policies for Distributed Systems ans Networks*, pages 18–39, Bristol, UK.

DMTF (1999). Common Information Model (CIM) specification - version 2.2. Distributed Management Task Force. http://www.dtmf.org/spec/cims.html.

Fernandez, M. P., de Castro P. Pedroza, A., and de Rezende, J. F. (2001a). Qos provisioning across a diffserv domain using policy-based management. In *(Globecom 2001)*, San Antonio, USA.

Fernandez, M. P., de Castro P. Pedroza, A., and de Rezende, J. F. (2001b). Quality of service in a diffserv domain using policy-based management. In *XVII International Teletraffic Congress (ITC'17)*, Salvador, Brazil.

Fernandez, M. P., de Castro P. Pedroza, A., and de Rezende, J. F. (2002). Optimizing fuzzy controllers with genetic algorithms for qos improvement. In *International Telecommunication Symposium (ITS'2002)*, Natal, Brazil.

Herrera, F., Lozano, M., and Verdegay, J. (1995). Tuning fuzzy logic controllers by genetic algorithms. *International Journal of Approximate Reasoning*, 12(3):299–315.

Hsiung, H., Fischer, M. J., Masi, D. M., Cuffie, D., and Scheurich, S. (1999). An approach to IP telephony performance measurement and modeling in government environments. In *9th Annual Conference of the Internet Society, INET'99*, San Jose, USA.

Jacobson, V., Nichols, K., and Poduri, K. (1999). An expedited forwarding PHB. RFC 2598.

Lee, C. C. (1990). Fuzzy Logic in control systems: Fuzzy logic controller, Part II. *IEEE Transactions on Systems, Man, and Cybernetics*, 20(2):419–435.

Lupu, E. and Sloman, M. (1997). Towards a role based framework for distributed systems management. *Journal of Network and Systems Management*, 5(1):5–30. Plenum Press Publishing.

Lupu, E. and Sloman, M. (1999). Conflicts in policy-based distributed systems management. *IEEE Transactions on Software Engineering, Special Issue on Inconsistency Management*, 25(6):852–869. IEEE.

McCanne, S. and Floyd, S. (1998). ns Network Simulator - Version 2. http://www.isi.edu/nsnam/ns/.

Moore, B., Strassner, J., and Elleson, E. (2000). Policy core information model. Internet Draft draft-ietf-policy-core-info-model-08.txt.

Mortensen, J. E. (1998). JFS Fuzzy System. http://www.inet.uni2.dk/ jemor/jfs.htm.

Sloman, M. (1994). Policy driven management for distributed systems. *Journal of Networking and Systems Management*, 2(4):333–360. Plenum Press.

Velasco, J. and Magdalena, L. (1995). Genetic algorithms in fuzzy control systems. In Winter, G., Periaux, J., Galan, M., and Cuesta, P., editors, *Genetic Algorithms in Engineering and Computer Science*, pages 141–165. John Wiley & Sons.