IMPLEMENTAÇÃO DE POLÍTICAS DE GERENCIAMENTO COM LÓGICA FUZZY E ALGORITMO GENÉTICO VISANDO À MELHORIA DA QUALIDADE DE SERVIÇO (QOS)

Marcial Porto Fernandez, Aloysio de Castro P. Pedroza e José Ferreira de Rezende

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). Esses contratos indicam políticas abstratas difíceis de serem interpretadas e implementadas pelos equipamentos de rede, que requerem informações absolutas. Como a lógica fuzzy possibilita a representação de valores abstratos, um controlador fuzzy foi utilizado para implementar um mecanismo de provisionamento dinâmico, reconfigurando os nós de acordo como tráfego. Esse trabalho propõe uma metodologia de construção de um controlador para configurar o provisionamento de rede a partir de políticas de gerenciamento visando melhorar a QoS em um domínio DiffServ. As funcionalidades são demonstradas através de simulação de uma aplicação de Telefonia IP cruzando um domínio DiffServ.

Palavras-chave: Qualidade de Serviço, DiffServ, Provisionamento de Redes

Abstract - The policy based management is a technique to coordinate the configuration of several equipments in a network, from Service Level Agreements (SLAs). These agreements produce abstract policies difficult to be interpreted and implemented by network equipments, that requires absolute information. As the fuzzy logic has been used to represent abstract values, a fuzzy controller was used to implement a dynamic provisioning mechanism to reconfigure all nodes according ingress traffic. This work proposes a methodology to map management policies in fuzzy controller 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.

Keywords: Quality of Service, DiffServ, Network Provisioning

1. INTRODUÇÃO

A Diferenciação de Serviços (DiffServ)[1] 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. A arquitetura DiffServ é escalável, oferecendo garantias para as diferentes classes, porém a QoS pode sofrer violações quando ocorrer congestionamento na agregação de vários fluxos de uma mesma classe devido ao mau dimensionamento dos recursos, o que compromete a QoS borda-a-borda.

Vi-se, portanto, que essa escalabilidade tem seu preço: não se pode garantir a QoS para todos os fluxos de uma mesma classe. O tráfego de dados na Internet tem um comportamento aleatório, portanto violações da QoS são esperadas. 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 necessidade de oferecer garantias de QoS 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 super-dimensionar os recursos para obter a QoS desejada. 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, mostramos experiências que demonstraram que um controlador fuzzy reconfigurando dinamicamente os nós conforme o tráfego entrante pode melhorar a QoS em um domínio. Esse controlador mostrou-se eficiente para melhorar a qualidade de serviço em um domínio DiffServ simples[2] e em um domínio complexo com várias topologias aleatórias de 40 nós[3]. A utilização do controlador fuzzy justifica-se pela não linearidade e ausência de um modelo matemático preciso para tratar estimativa de tráfego[4]. Comparado a um controlador convencional, 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 (AG), que otimizam os parâmetros do controlador fuzzy[5, 6].

O Gerenciamento Baseado em Políticas[7] tem se mostrado uma técnica eficaz para coordenar a configuração de uma rede para obter a QoS desejada. A maioria dos trabalhos sobre Gerenciamento Baseado em Políticas foca na especificação de políticas e do modelo de informações das entidades onde as políticas serão aplicadas. Entretanto, pouco trabalho tem sido feito sobre como as entidades interpretam as políticas e definem comandos de configuração nos equipamentos reais[8, 9]. Além disso, ainda falta estudo sobre aplicação

Marcial Porto Fernandez é pesquisador do Instituto de Computação da Universidade Federal Fluminense. Aloysio de Castro P. Pedroza e José Ferreira de Rezende são professores do Programa de Engenharia Elétrica/COPPE e Escola Politécnica da Universidade Federal do Rio de Janeiro. E-mails: mfernandez@ic.uff.br, aloysio@gta.ufrj.br, rezende@gta.ufrj.br

de políticas em grandes domínios, onde interpretações diferentes de uma mesma política podem causar instabilidades e falhas na operação do sistema[10].

Apresentamos 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 de recursos nos nós do domínio. Mostramos então, o procedimento de otimização do controlador fuzzy para melhorar as métricas de QoS. Para validar a metodologia, foi construído um controlador fuzzy que implementa as políticas definidas inicialmente. Assim, foi realizada uma simulação desse protótipo, com avaliação de tempo de retardo, variação do retardo e descarte de fluxos de telefonia IP. Posteriormente, 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 apresenta a construção do controlador fuzzy; a seção 5 mostra a otimização do controlador fuzzy; a seção 6 mostra a implementação do protótipo, seguindo a metodologia proposta; a seção 7 apresenta os resultados obtidos na simulação; e, finalmente, a seção 8 apresenta as conclusões do trabalho e sugere temas para trabalhos futuros.

2. SISTEMA DE PROVISIONAMENTO DINÂMICO DE RECURSOS

Para coordenar a configuração de uma rede com requisitos de qualidade, precisamos de um sistema que mantenha o provisionamento da rede coerente e que, como a arquitetura DiffServ, também seja escalável. Definiu-se então um controlador que implementa o provisionamento dinâmico nos nós a partir das políticas especificadas pelo operador de redes. Abaixo, apresentamos a arquitetura do sistema e a metodologia para a construção desse controlador.

2.1 ARQUITETURA DO SISTEMA DE PROVISIONAMENTO DINÂMICO

A arquitetura do sistema de provisionamento dinâmico tem como objetivo propiciar o melhor desempenho mantendo a escalabilidade do sistema. Apresentamos, na figura 1, a arquitetura proposta.

Em cada nó do domínio, seja de borda ou de núcleo, o controlador do nó 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 relativamente leve comparada ao AG, 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 usar um mecanismo computacionalmente pesado pode impactar no desempenho

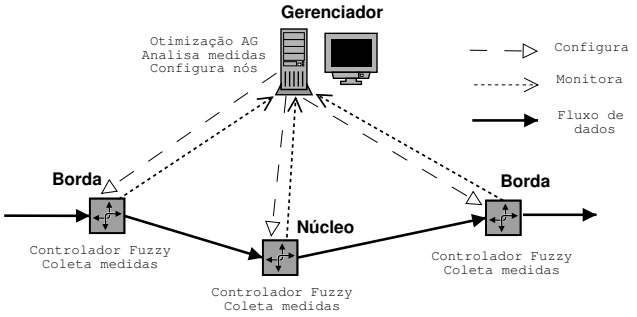


Figura 1. Arquitetura do sistema proposto.

do roteador. O nó também é responsável por coletar as informações do estado do equipamento e informá-las ao Gerenciador, utilizando um protocolo de gerenciamento de redes, por exemplo o 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 que sejam importantes para funcionamento do domínio. Realiza também 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. Outra observação importante é 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 METODOLOGIA PARA CONSTRUÇÃO DO CONTROLADOR DE PROVISIONAMENTO

A metodologia consiste na realização de definição, otimização e teste, efetuadas continuamente para manter o sistema ajustado à topologia usada. 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. Vamos supor que 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.

O ponto de partida do presente trabalho são as especificações de políticas administrativas com base nos requisitos de QoS, conforme 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

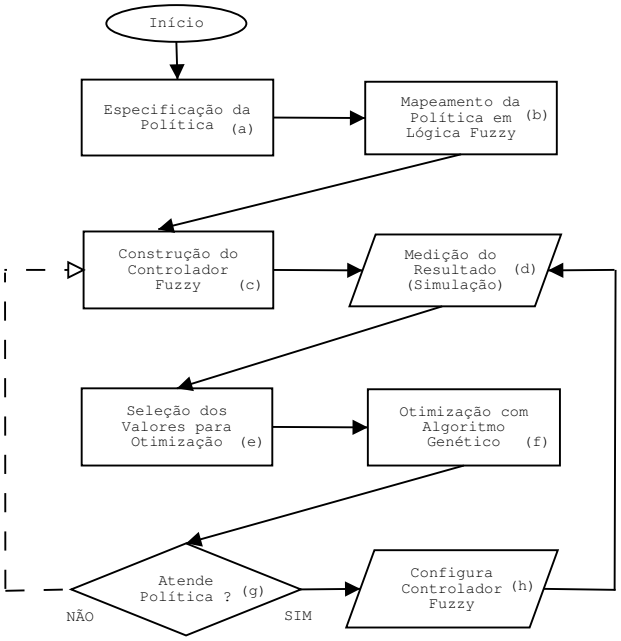


Figura 2. Fluxograma da metodologia proposta.

fuzzy, na seção 3.3.

A próxima etapa é a construção do controlador fuzzy, mostrado na figura 2(c). Essa etapa, apresentada na seção 4, 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 aplicá-lo aos equipamentos da rede ou, como em nosso experimento, realizar a simulação. Nessa 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 e descarte de pacotes.

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) e detalhada na seção 5.2. A etapa seguinte consiste na otimização dos parâmetros fuzzy através do algoritmo genético, utilizando como referência de função objetivo os valores selecionados no ítem anterior. Essa etapa é mostrada na figura 2(f) e detalhada na seção 5.3.

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; 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, pode então 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. 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[7] tem se mostrado uma técnica eficaz para coordenar a configuração de uma rede para obter a 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[11], apresentada a seguir.

3.1 LINGUAGEM DE ESPECIFICAÇÃO DE POLÍTICAS: PONDER

A linguagem *Ponder* foi proposta por Damianou *et al.*[12] para especificar textualmente políticas de gerenciamento, de acordo com as propostas de Sloman [7] e Lupu[13, 10]. 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 proíbe a execução de uma ação; e *política de delegação* (**deleg**), que permite a um elemento 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)*. A classe de melhor esforço, *(BE - Best Effort)* deverá ter a prioridade reduzida sempre que houver queda na qualidade da classe EF. Foram definidas duas especificações de política: uma para o escalonador, aplicável em todos os nós, e uma para o condicionador, aplicável apenas aos 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 e o retardo máximo é

de 100 ms. Da linha 10 à 14, são estabelecidas as restrições baseadas nos valores previamente definidos. Nas linhas 16 a 19, 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

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

A partir desse ponto, as políticas serão executadas pelo sujeito **s** e aplicadas ao alvo **t**. Nas linhas 22 a 27, 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 29 a 34, 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 praticamente 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.

Código 2. Mapeamento do comando constraint

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

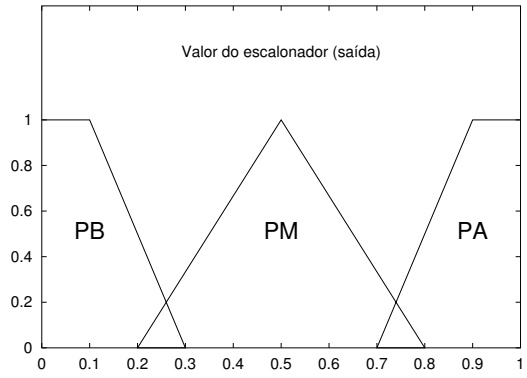


Figura 3. Função pertinência associada comando constraint

Apresentamos, no código 2, uma especificação usando os comandos **constraint** e **auth-**. Nesse caso, a restrição da banda mínima do escalonador seja 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, onde as variáveis "PB", "PM" e "PA" significam Prioridade Baixa, Prioridade Média e Prioridade Alta, respectivamente. Podemos observar que o centro de gravidade do polígono 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.

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()
   }
```

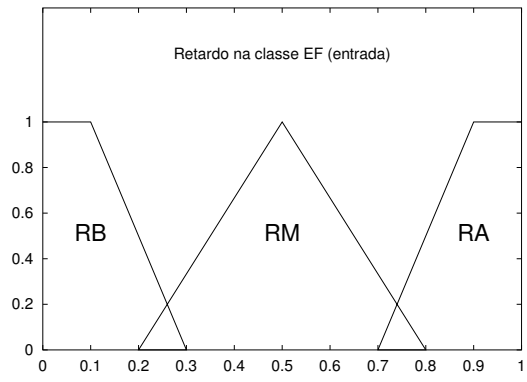


Figura 4. Função de pertinência associada ao comando event

A função de pertinência correspondente a essa especificação é mostrada na figura 4. Podemos definir que cada 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 e o formato do polígono 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[14] 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. CONSTRUÇÃO DO CONTROLADOR FUZZY

A lógica fuzzy foi introduzida por Lofti Zedeh[15], como uma generalização da teoria clássica. A extensão sugerida por Zadeh está na possibilidade de um determinado elemento poder pertencer a um conjunto com um valor chamado grau de pertinência. Assim, um elemento não simplesmente pertence ou não pertence a um conjunto, como na lógica clássica, mas poderá pertencer a um conjunto com grau de pertinência que varia no intervalo [0,1], onde o valor 0 indica uma completa exclusão, e o valor 1 representa completa inclusão.

A lógica fuzzy utiliza variáveis lingüísticas no lugar de variáveis numéricas. Variáveis lingüísticas admitem como valores apenas expressões lingüísticas, como "muito grande", "pouco frio", "mais ou menos jovem", que são representadas por conjuntos fuzzy. A teoria da construção de um controlador fuzzy foi mostrada em Lee[4]. A construção de um sistema de controle fuzzy é baseada na idéia de se incorporar

"experiência" ou "conhecimento especializado" de um operador humano para se obter a melhor estratégia de controle. Desse modo, a forma das regras empregadas depende do processo a ser controlado.

Os controladores fuzzy são a mais importante aplicação da Teoria Fuzzy. Seu funcionamento é diferente dos controladores convencionais, pois estes exigem o desenvolvimento das equações diferenciais que descrevem o sistema. Em um controlador fuzzy, o conhecimento pode ser expresso de forma intuitiva, usando as variáveis lingüísticas. As aplicações ideais para utilização de Controladores Fuzzy são as seguintes[16]:

1. Processos muito complexos, onde não há modelo matemático claro;
2. Processos altamente não lineares ou com comportamento probabilístico;
3. Aquelas em que o processamento de conhecimento especializado (formulados lingüisticamente) é o único possível.

Esses argumentos justificam a escolha do controlador fuzzy para nossa proposta. Apresentamos, a seguir, o controlador fuzzy apropriado para controlar o provisionamento dinâmico em arquitetura DiffServ obtendo como resultado uma melhor qualidade de serviço. A metodologia utilizada para construção do controlador fuzzy foi baseada na metodologia proposta por Cordón e Herrera[17].

4.1 FUNÇÕES DE PERTINÊNCIA

O controlador proposto utiliza variáveis lingüísticas triangulares e trapezoidais, pela possibilidade de serem implementadas com código mais simples e eficiente. Realizamos também experiência com uma função gaussiana, porém os resultados obtidos não justificaram a complexidade adicionada.

A arquitetura de nosso controlador foi dividida em duas partes: um controlador do escalonador existente em todos os nós do domínio e um controlador do condicionador existente apenas nos nós de borda.

4.1.1 CONTROLADOR DO ESCALONADOR

A variável de saída, que possibilita o controle do escalonador, depende do tipo do mecanismo utilizado. 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 retardo instantâneo do pacote na fila EF. Como os demais tempos do processamento do pacote são praticamente irrelevantes, consideramos o tempo de espera na fila como o tempo total de permanência 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 BE, indicando a exaustão do recurso. Vários mecanismos de gerenciamento ativo de fila podem ser utilizados neste caso, porém devemos considerar que, na ocorrência de descarte, há escassez de recurso de rede. As variáveis semânticas são:

Entrada:

- Peso relativo atual do escalonador (EF/BE).
- Retardo instantâneo na classe EF.
- Perda de pacotes na classe BE.

Saída:

- Peso relativo no escalonador da fila EF/BE.

4.1.2 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 de controle depende do tipo de condicionador utilizado. De forma geral, seguem a filosofia do balde de fichas (Token Bucket), que é um repositório onde se colocam fichas a uma taxa constante.

A primeira variável é o descarte de pacotes da classe EF no condicionador. Quando um pacote chega ao condicionador e encontra o balde vazio, o mesmo 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 outro lado, o PHB EF em um domínio DiffServ só deve descartar pacotes na borda[18], sendo indesejáveis os descartes no interior do domínio. 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. Assim, o valor de retardo máximo nos nós de núcleo é enviado ao gerenciador, que sinalizará a redução da taxa de entrada para os nós de borda.

A segunda variável de entrada é o retardo máximo da classe EF no interior do domínio, que indicará a necessidade de reduzir a taxa de enchimento do balde caso o retardo seja muito alto.

Esta foi a política escolhida para tratar congestionamento no núcleo, entretanto 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). As variáveis semânticas são:

Entrada:

- Taxa atual do Token Bucket do condicionador.
- Perda de pacotes na classe EF no condicionador.
- Retardo máximo na classe EF nos nós de núcleo.

Saída:

- Taxa do Token Bucket do condicionador.

4.2 BASE DE REGRAS

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 deve ser definida de acordo com a política administrativa. Para nosso exemplo, utilizamos um conjunto de regras priorizando a classe EF em qualquer situação e deixando para a classe BE um mínimo de 10% da banda.

A escolha dos operadores fuzzy seguiram as recomendações e metodologia de Cordón e Herrera[19], conforme o tipo de aplicação utilizada em nosso estudo. Os operadores selecionados foram **máximo** para união e interseção (envoltória externa de todas as variáveis semânticas válidas), enquanto a implicação utilizada foi do tipo **mínimo**, a disjunção (OU) do tipo **máximo** e conjunção (E), do tipo **mínimo**.

O valor de resposta semântico não tem valor prático, por isso deve ser convertido em um valor discreto (numérico), que será aplicado ao atuador do controlador. Esse processo é chamado de defuzificação. O método mais usual para aplicação em controladores é o do centro de gravidade que consiste em calcular o centro de gravidade da figura obtida através da combinação das funções de pertinência.

5. OTIMIZAÇÃO DO CONTROLADOR FUZZY

Com o objetivo de especificar um controlador eficiente, utilizamos algumas ferramentas de otimização. Na criação de regras, usamos o algoritmo Wang-Mendel[20], que, a partir do comportamento desejado, cria um conjunto de regras coerentes. Para a otimização dos parâmetros das funções de pertinência, usamos um algoritmo genético que descobre a melhor combinação de parâmetros para obter o resultado ideal.

5.1 VERIFICAÇÃO DE REGRAS ATRAVÉS DO ALGORITMO DE WANG-MENDEL

A lógica fuzzy apresenta a vantagem de permitir a representação de conceitos ambíguos e produzir uma resposta eficaz mesmo com entradas duvidosas. Porém, um comportamento eficiente requer a definição de regras coerentes. Para que isso não dependa inteiramente do trabalho do projetista do sistema, é desejável a introdução de uma metodologia para produzir funções de pertinência e regras coerentes e eficientes. Utilizamos o algoritmo Wang-Mendel[20], que, a partir do comportamento desejado, cria um conjunto de regras coerentes. Seu funcionamento básico é apresentado no algoritmo 1 seguindo a implementação apresentada por Cox[16].

A maior utilidade dessa metodologia é verificar a consistência do conjunto de regras criadas pelo especialista identificando a ocorrência de regras contraditórias, que levariam a resultados errôneos.

Algoritmo 1 Algoritmo Wang-Mendel

Entrada: Base de regras nao verificada
$contador(i) \leftarrow 0$ {Inicia contador de posto}
$contradicao(i) \leftarrow 0$ {Marca todas as regras não contraditórias}
Normaliza as regras nao verificadas na forma IF <variável> IS <adjetivo> AND <variável> IS <adjetivo> AND ... THEN <variável> IS <adjetivo>.
enquanto Existe regra a ser verificada **faça**
  se Regra já existe na base de regras **então**
    Regra não é incluida e $contador(i)$ é incrementado
  senão se Regra ainda não existe e não contradiz nenhuma outra regra **então**
    Regra adicionada à base de regras e $contador(i)$ é incrementado
  senão se Regra contradiz alguma regra existente **então**
    Regra incluída na tabela de regras e marca $contradicao(i) = 1$ e $contador(i)$ é incrementado.
  fim se
fim enquanto
Regras com $contradicao(i) = 0$ sao incluídas na base de regras final.
Regras com $contradicao(i) = 1$ inclue a regra com $contador(i)$ maior e descarta a outra.

5.2 SELEÇÃO DOS VALORES PARA OTIMIZAÇÃO DO CONTROLADOR FUZZY

O processo de otimização por AG necessita de uma função objetivo, para onde a otimização deve convergir. A grande dificuldade, no entanto, é definir essa função objetivo[5]. No caso do controlador do escalonador, temos como entrada as variáveis peso inicial do escalonador, retardo na fila EF e descarte na fila BE e, como saída, o peso do escalonador. No entanto, a variável a ser otimizada é o retardo da fila EF, que somente pode ser avaliada em simulação.

Em vista disso, definimos uma metodologia para obter os valores utilizados na otimização. Durante a simulação com as funções de pertinência arbitradas pelo projetista, armazenamos todas as combinações de parâmetros de entrada do controlador fuzzy e as métricas de avaliação como, por exemplo, o valor de retardo obtido no período seguinte e a taxa de descarte de cada classe, ou seja, o resultado obtido pela aplicação de parâmetros na simulação.

De posse desses valores, escolhemos todas as combinações de parâmetros que produzem um retardo baixo, aquelas que produzem uma maior redução no retardo dos pacotes em medidas consecutivas e aquelas que produzem uma menor taxa de descarte agregado (EF + BE). Foi necessário estabelecer um critério para a ordenação das medidas, pois elas são independentes e algumas até contraditórias, como baixo retardo e baixo descarte. Ordenamos as medidas individualmente por ordem crescente, para a primeira e terceira métrica, e decrescente, para a segunda métrica. Calculamos um valor de posto

igual a média da posição de cada parâmetro, isto é, menor retardo absoluto, maior redução no retardo e menor descarte. Como as duas primeiras medidas estão relacionadas ao retardo, podemos dizer que a otimização utilizou uma ponderação de 66% para retardo e 33% para descarte. Assim, escolhemos as melhores combinações de parâmetros para nosso problema, segundo a ponderação escolhida.

5.3 OTIMIZAÇÃO DE CONTROLADOR FUZZY ATRAVÉS DE ALGORITMO GENÉTICO

O controlador definido com funções de pertinência arbitradas pelo projetista e com a base de regras verificado pelo algoritmo de Wang-Mendel, mostrado na seção 5.1, garante a definição de um controlador correto, porém ainda sem garantia de eficiência. Para otimizar o resultado do controlador, precisamos escolher os melhores parâmetros do controlador através da utilização do algoritmo genético.

5.4 FUNCIONAMENTO DOS ALGORITMOS GENÉTICOS

O algoritmo genético cria inicialmente um conjunto de soluções, a partir dos valores arbitrados inicialmente (indivíduos), para um determinado problema. Esse conjunto é chamado de população. O posto de cada indivíduo é calculado pelo teste da solução em relação ao valor desejado. Os indivíduos com posto baixo são substituídos por novos indivíduos criados a partir de indivíduos com posto alto. Esse processo continua até uma solução ser encontrada ou alguma condição de término seja alcançada (por exemplo, número de gerações). O indivíduo com maior posto é a solução encontrada para o problema.

A partir da tabela com valores de entrada e saída desejados, escolhidos na seção 5.2, o algoritmo genético cria um indivíduo, verifica se o resultado calculado se aproxima do valor desejado e pontua cada avaliação. A cada geração,os indivíduos mais aptos são selecionados e combinados (através de *crossover*), obtendo-se um conjunto de parâmetros otimizado para a tabela de valores desejados. Eventualmente, os parâmetros podem convergir para um bom valor local, mas que não é o melhor para o universo. Para evitar isso, aplica-se o processo de *mutação*, incluindo no conjunto de avaliação um valor aleatório totalmente novo, que pode indicar uma nova região de otimização.

Após uma série de iterações, obtemos um conjunto de parâmetros otimizado para o controlador. A grande vantagem desse procedimento é a otimização automática dos parâmetros das funções de pertinência, sem depender do critério do projetista. No método *simples*, que tem funcionamento intermediário aos métodos *geracional* e *em regime*, a avaliação de pais e filhos são realizadas em paralelo, possibilitando uma melhor escolha de indivíduos. Assim, obtemos uma convergência rápida para o caso da solução estar próxima ou distante do ponto de partida.

A vantagem da utilização de algoritmo genético para otimização é sua característica de poder ser executado continuamente, a partir de medidas de desempenho da rede durante o funcionamento normal. Esse procedimento permite o ajuste dos parâmetros quando ocorrem modificações na topologia ou no padrão de tráfego, durante a operação da rede. A execução do algoritmo genético, no entanto, não provoca queda no desempenho da rede, pois a otimização pode ser realizada em equipamentos dedicados (os roteadores da rede somente executam o controlador fuzzy). Além disso, a otimização não precisa ocorrer em períodos muito curtos, pois as mudanças que poderiam causar alterações ocorrem na ordem de dias ou semanas.

A codificação do cromossomo usado foi número real com quatro casas decimais, o mecanismo de funcionamento foi o método simples, o método de seleção foi da roleta e o cruzamento foi do tipo multi-ponto. Os mecanismos utilizados nesse trabalho foram detalhados por Michaelewicz[21].

5.5 PARÂMETROS DE CONFIGURAÇÃO DO ALGORITMO GENÉTICO

A configuração correta dos parâmetros do algoritmo genético é, sem dúvida, um dos aspectos mais importantes na estratégia dos AGs. Não existe uma metodologia genérica, uma vez que tais configurações dependem da aplicação a ser otimizada. É importante também levar em consideração os tempos de execução do problema e os recursos computacionais disponíveis.

Os parâmetros do algoritmo genético utilizado na otimização do controlador fuzzy estão relacionados na tabela 1. Foram usados valores dentro das faixas usuais,conforme comentados em seguida, porém com os ajustes para melhoria da eficiência.

Tabela 1. Parâmetros do algoritmo genético utilizados.

Parâmetro	Valor
Tamanho da população	500
Taxa de cruzamento	65%
Taxa de mutação	5%
Critério de parada	90% da população igual
Número de gerações	600

Apresentamos a seguir os parâmetros mais importantes e sua influência no desempenho do algoritmo.

5.5.1 TAMANHO DA POPULAÇÃO

O tamanho da população N afeta o desempenho global e a eficiência dos algoritmos genéticos, já que uma população pequena fornece uma cobertura pequena do espaço de busca do problema. Uma grande população fornece uma cobertura maior do espaço de busca, além de prevenir convergências prematuras para soluções locais. No entanto, para se trabalhar com grandes populações, são necessários maiores recursos computacionais e tempos muito longos para resolução de problemas simples.

Assim, a escolha correta do tamanho de população, que produz respostas corretas e rápidas, depende do problema e

da sua formulação. Geralmente, usa-se um tamanho de população proporcional ao tamanho do cromossomo, isto é, quanto maior for o cromossomo maior deverá ser o tamanho da população, para manter uma diversidade razoável. A literatura sugere populações entre 50 e 100 cromossomos como bom compromisso cobertura-desempenho[22].

O tamanho da população utilizada foi de 500 indivíduos, que possibilitou uma melhor cobertura do espaço, mas que se mostrou eficiente em nossa experiência, pela capacidade computacional disponível.

5.5.2 TAXA DE CRUZAMENTO

A taxa de cruzamento P_C define a probabilidade de ocorrer um cruzamento. Quanto maior for essa taxa, mais rapidamente novas estruturas serão introduzidas na população. Porém, estruturas com boas aptidões poderão ser substituídas mais rapidamente, perdendo-se boas oportunidades para achar uma solução ótima. Com um valor baixo, o algoritmo pode tornar-se muito lento, demorando a encontrar a solução ideal. A maior parte da literatura recomenda uma taxa de cruzamento entre 50% e 95%[22].

A taxa de cruzamento utilizada foi de 65%, valor usual que se mostrou adequado para nossa experiência.

5.5.3 TAXA DE MUTAÇÃO

A taxa de mutação P_M define a taxa de ocorrência de mutação. Uma taxa de mutação baixa diminui a inclusão de indivíduos novos na população, provocando o encontro de resultados baseados em máximos locais, pois restringe o espaço de busca. Uma taxa muito alta transforma a busca do algoritmo genético em busca essencialmente aleatória.

Alguns pesquisadores recomendam a escolha da taxa de mutação com base no tamanho do cromossomo e da população. De Jong [23] sugere uma taxa de mutação inversamente proporcional ao tamanho da população. A maior parte da literatura recomenda uma taxa de mutação entre 0,1% e 5% [22].

Utilizamos um valor de taxa de mutação alta (5%), porém dentro da faixa recomendada. Esse valor se justifica pelo fato de usarmos o método de seleção da roleta, que pode produzir uma baixa diversidade na população e não atingir o valor máximo global da função.

5.5.4 NÚMERO DE GERAÇÕES

O número de gerações define a quantidade de populações criadas até a resposta final. Com um valor baixo, podemos encontrar rapidamente uma solução ótima local, porém ainda longe da solução global. Com um valor alto, temos garantia de encontrar a solução ótima,porém o tempo para chegarmos à resposta pode ser longo, além de, no caso de problemas simples, criarmos novas gerações desnecessariamente, mesmo após a solução ótima ter sido encontrada. O valor ideal para este parâmetro depende do problema e da sua formulação. Em nossa experiência, estabelecemos um limite relativo válido para qualquer problema: encerramos o processamento do AG quando 90% da população for constituída de apenas um indivíduo, que é a solução desejada.

O programa JFS[14] oferece apenas duas formas de encerrar o processamento: pelo tempo decorrido ou pela quantidade de gerações. No entanto, para podermos garantir que encontraríamos o valor de ótimo global, estabelecemos que a população contivesse pelo menos 90% de um mesmo indivíduo. Assim fomos obrigados a executar uma grande quantidade de gerações e testar se o critério de fim tinha sido atingido. Em todas as otimizações realizadas, a quantidade de 600 gerações foi suficiente para atingir essa meta.

6. 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.

Os experimentos consistiram em realizar diversas simulações avaliando a QoS dos fluxos de telefonia IP concorrendo com outros fluxos não prioritários. Comparamos os resultados sem a utilização de controlador, utilizando um controlador convencional e o controlador proposto.

6.1 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[24]. A ferramenta para especificação e verificação de políticas foi a *Ponder Toolkit*[11], constituída por um editor de políticas e um compilador da linguagem. Foi utilizada aversão 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[14]. 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*.

6.2 OTIMIZAÇÃO DO CONTROLADOR FUZZY

O controlador definido com funções de pertinência arbitradas pelo projetista, mostrado na seção 4.1, e com a base de regras estabelecido a partir da base de conhecimento semântico e através do algoritmo de Wang-Mendel, mostrado na seção 5.1, garante a definição de um controlador correto porém sem garantia de eficiência. Para otimizar o resultado do controlador precisamos escolher os melhores parâmetros do controlador através da utilização do algoritmo genético, detalhado na seção 5.3.

A grande dificuldade para uso do algoritmo genético é definir a função objetivo para otimização. Em nossa experiência, a definição da função objetivo é agravado pelo fato de que o valor a ser otimizado não é obtido diretamente da saída do controlador. Enquanto o controlador do escalonador fornece o peso de configuração do escalonador, a variável que deve ser otimizada é o retardo dos pacotes na classe EF, obtidos apenas após a simulação.

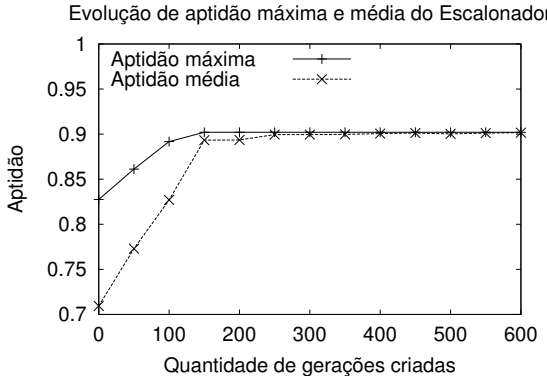
Sendo assim foi definida uma metodologia para obter os valores da função objetivo utilizados na otimização. A partir de funções de pertinência arbitradas pelo projetista,executamos uma simulação com esses valores. Os parâmetros definidos a priori não são críticos, pois o algoritmo genético converge para resultado ótimo a partir de qualquer ponto de partida. Obviamente a convergência pode ser mais rápida ou mais lenta de acordo com a escolha dos parâmetros iniciais.

Durante a simulação armazenamos todas as combinações de valores de entrada do controlador fuzzy e o valor de retardo obtido no período seguinte, isto é, o resultado obtido pela aplicação dos valores de entrada do controlador. De posse desses valores, podemos escolher as combinações de parâmetros que produzem um retardo baixo e aquelas que produzem uma maior redução no retardo dos pacotes entre duas medidas consecutivas. Finalmente, escolhemos as melhores combinações de valores para nosso problema.

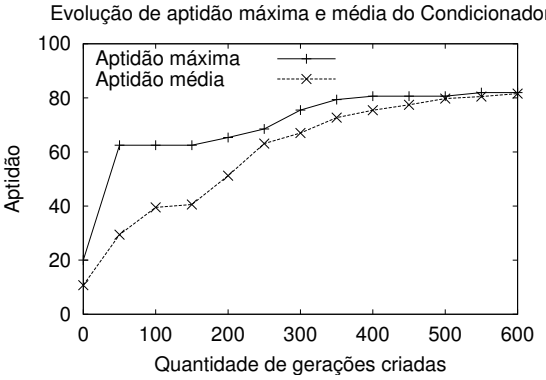
Os valores selecionados são utilizados como base de conhecimento para a aplicação do algoritmo genético, que produzirá o melhor conjunto de parâmetros do controlador fuzzy para atingir a otimização desejada. Utilizamos como referência para escolha os 20% melhores valores de retardo e redução do retardo. Fizemos experiência escolhendo também 10% e 30% dos melhores valores, porém os resultados obtidos foram semelhantes,mostrando que a quantidade de valores escolhida não é importante para a otimização, pelo menos para os percentuais testados.

Apresentamos na figura 5 o gráfico da evolução da aptidão média e aptidão máxima ao longo do tempo. Podemos observar na figura 5(a) que a convergência do controlador do escalonador é bastante rápida, atingindo valores semelhantes após 150 gerações. Na figura 5(b) notamos que a convergência é um pouco mais lenta, atingindo valores semelhantes após 500 gerações.

Após a realização do procedimento de otimização obtivemos como resultado novas funções de pertinência para o controlador fuzzy. Apresentamos na figura 6 a função de pertinência da variável **retardo na classe EF**, utilizada pelo controlador do escalonador. A figura 6(a) mostra a função de pertinência original, especificada pelo projetista, onde notamos uma distribuição equilibrada dos valores semânticos ao longo do intervalo de operação. A figura 6(b) mostra a função de pertinência otimizada, obtida após a aplicação da otimização através de algoritmo genético. Podemos notar que o valor semântico "RB" (Retardo Baixo) foi reduzido ao mínimo possível e que o valor semântico "RM" (Retardo Médio) e "RA" (Retardo Alto) sofreram um leve deslocamento para aproximá-los ao valor "RB". Deslocar todos os valores semânticos para a esquerda determina que a base de regra executará mais cedo ações de aumento da banda de saída do escalonador, fazendo que se obtenha um valor de retardo menor, comparando-se com o controlador original não otimizado.



(a) Controlador do Escalonador



(b) Cntrolador do Condicionador

Figura 5. Evolução da aptidão máxima e média

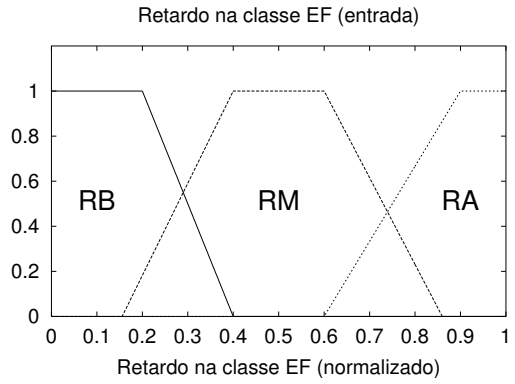
A grande vantagem do algoritmo genético é o processo contínuo de otimização, utilizando valores de várias simulações ou até mesmo de dados reais de uma rede em funcionamento. Esse procedimento permite a atualização contínua dos parâmetros de acordo com a mudança de topologias e padrões de tráfego, comuns em uma situação real.

6.3 TOPOLOGIA DE SIMULAÇÃO

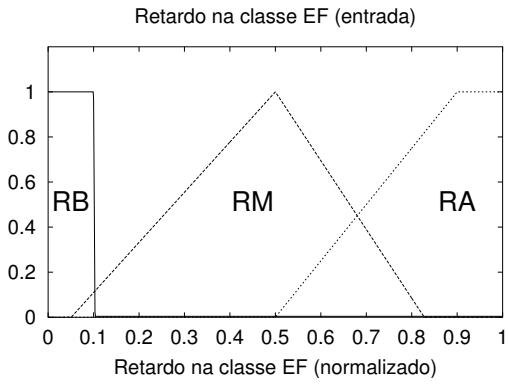
A topologia utilizada para a simulação, é apresentada na figura 7. Essa topologia forma um domínio DiffServ com cinco nós, sendo dois de núcleo e três de borda. Podemos notar que existem dois pontos de congestionamento, o primeiro entre nós de borda e do núcleo e outro entre dois nós de núcleo.

6.4 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



(a) Controlador não otimizado



(b) Controlador otimizado

Figura 6. Função de pertinência de entrada retardo na classe EF

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[18] e o tráfego concorrente para classe BE. O tráfego concorrente foi definido como CBR/UDP para provocar uma concorrência mais severa com o tráfego de voz na classe EF.

A figura 8 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[25].

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 ve-

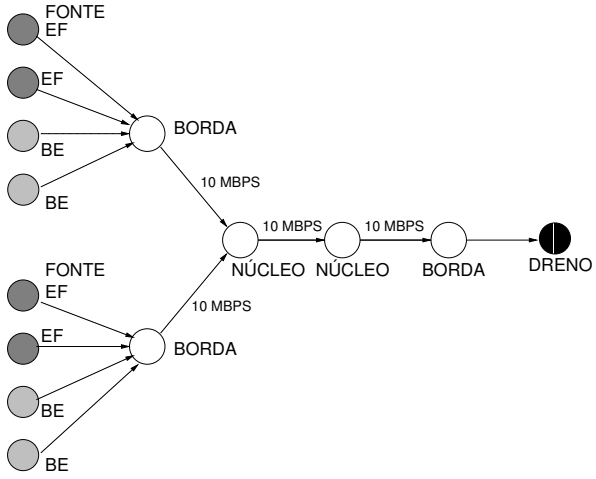


Figura 7. Topologia de simulação.

zes 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.

6.5 CONTROLADOR CONVENCIONAL

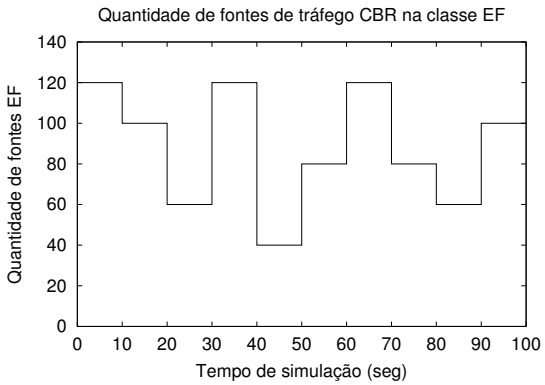
Com o objetivo de validar nossa proposta, e tendo em vista que os resultados com a utilização de qualquer controlador seriam melhores que a situação sem controlador, definimos, para comparação, um controlador PD (Proporcional e Derivativo). A idéia desse controlador é guardar as últimas três medidas 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.

7. RESULTADOS

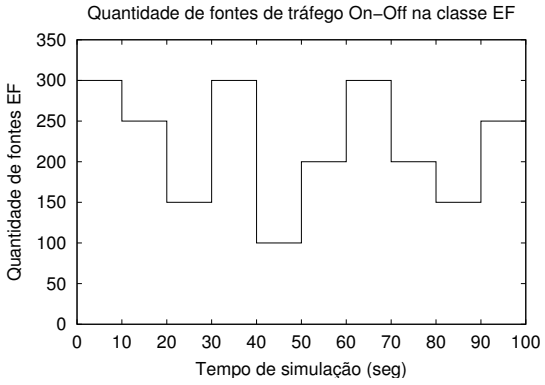
Nesta seção mostramos as tabela de percentil de retardo, percentil da variação do retardo e taxa de descarte em três situações: sem a utilização de controlador, com controlador convencional e com o controlador fuzzy. Todas as simulações iniciam com alocação de 50% da banda de saída para cada classe. 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. O tamanho do pacote escolhido foi de 576 bytes. Apresentamos, nas seções 7.1 e 7.2, 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, mostrada na tabela 2. Essa avaliação apresenta o percentil 50, 90 e 95 para tráfegos CBR e On-Off comparando



(a) Fontes CBR



(b) Fontes On-Off Exponencial

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

as situações sem controlador, com controlador convencional e com controlador fuzzy. A tabela 3 apresenta os valores de percentil 50, 90 e 95 para medidas de variação de retardo para tráfego CBR e On-Off sem o uso de controlador, usando o controlador convencional e o controlador fuzzy.

Tabela 2. Retardo na classe EF (ms)

Tráfego	Médio	Perc 50	Perc 90	Perc 95
CBR S/Ctrl	43.4	43.0	65.8	75.3
CBR Conven.	33.6	30.3	58.8	71.3
CBR Fuzzy	19.3	17.3	34.9	41.1
OO S/Ctrl	17.9	16.6	33.2	36.9
OO Conven.	14.9	14.7	27.1	32.3
OO Fuzzy	5.6	3.7	11.7	12.4

A tabela 4 apresenta a taxa de descarte de pacotes (pacotes descartados/pacotes transmitidos) para o tráfego CBR e a tabela 5 mostra a taxa de descarte para o tráfego On-Off. Mostramos as taxas de descarte nas classes EF e BE sem o uso de controlador, usando o controlador convencional e o controlador fuzzy.

Tabela 3. Variação do retardo na classe EF (ms)

Tráfego	Médio	Perc 50	Perc 90	Perc 95
CBR S/Ctrl	1.7	0.4	0.4	7.3
CBR Conven.	1.5	0.4	0.4	6.0
CBR Fuzzy	1.0	0.4	0.4	0.4
OO S/Ctrl	1.2	0.3	3.6	5.7
OO Conven.	1.1	0.2	3.3	4.6
OO Fuzzy	0.5	0.1	1.3	1.9

Tabela 4. Taxa de descarte com tráfego CBR

Controlador	EF	BE
Sem Ctrl	0.0198	0.0766
Convencional	0.0125	0.0843
Fuzzy	0.0068	0.0795

Tabela 5. Taxa de descarte com tráfego On-Off

Controlador	EF	BE
Sem Ctrl	0.0379	0.1388
Convencional	0.0272	0.1439
Fuzzy	0.0027	0.1486

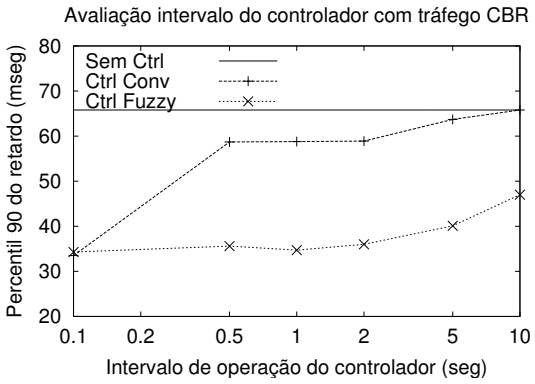
Podemos observar que há uma melhora na QoS e redução no descarte da classe EF como controlador fuzzy comparado às situações sem controlador e com controlador convencional. Obviamente, quando se reduz a taxa de descarte da classe EF, provocamos um aumento na taxa da classe BE, pois a rede está com sua 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.

7.1 AVALIAÇÃO DO INTERVALO DE OPERAÇÃO DO CONTROLADOR

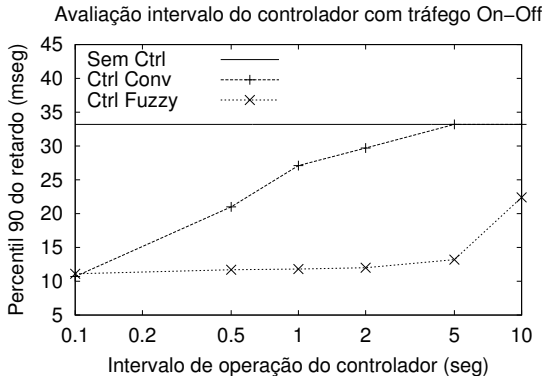
O primeiro parâmetro avaliado foi o intervalo de operação do controlador que é mostrado na figura 9. A figura 9(a) mostra a avaliação do retardo na classe EF para tráfego CBR e a figura 9(b) mostra o retardo para tráfego On-Off. Medimos o resultado de percentil 90 do retardo variando o intervalo de operação do controlador em 0,1 s, 0,5 s, 1 s, 2 s, 5 s e 10 s. Utilizamos o tamanho de pacote 576 bytes. Para efeito de comparação, indicamos com uma linha contínua o percentil 90 do retardo da situação sem 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 8). 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



(a) Tráfego CBR



(b) Tráfego On-Off

Figura 9. Avaliação do intervalo de atuação do controlador

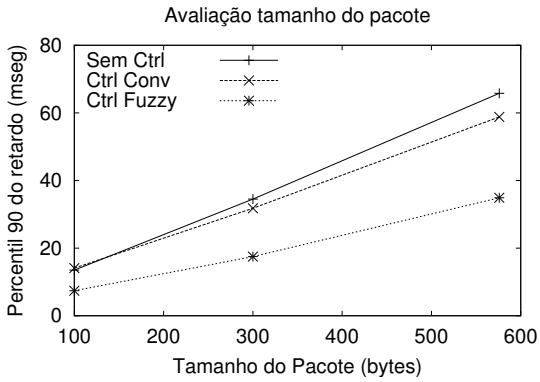
CBR,mostrando a maior eficiência do controlador fuzzy para tratar tráfego On-Off. O critério para a escolha do valor 1 segundos para intervalo de operação do controlador foi o equilíbrio entre resultado da QoS obtida e a sobrecarga computacional para ambos os controladores.

Outra conclusão desses resultados é a demonstração 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.

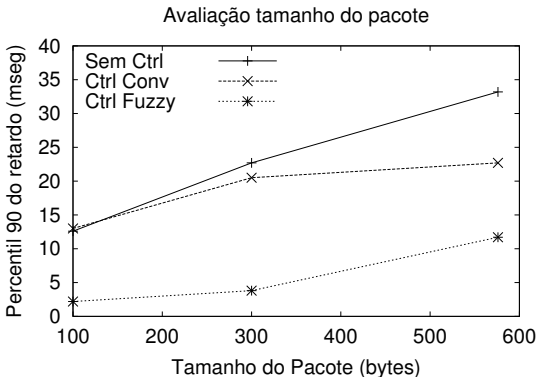
7.2 AVALIAÇÃO DA VARIAÇÃO DO TAMANHO DO PACOTE IP

O segundo parâmetro avaliado foi o tamanho do pacote, mostrado na figura 10. 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.

Na avaliação com tráfego CBR, mostrada na figura 10(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 qualquer tamanho de pacote avaliado, 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 10. Avaliação da influência do tamanho dos pacotes

Na avaliação com tráfego On-Off, mostrada na figura 10(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.

8. 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.

O gerenciamento baseado em políticas permite que o controlador seja genérico, podendo definir o comportamento do sistema de acordo com as decisões administrativas do operador de rede do domínio. Além disso, o comportamento pode ser facilmente alterado em todo o domínio, durante o funcionamento normal do sistema. A proposta de mapeamento de políticas *Ponder* em parâmetros de controlador fuzzy demonstra a capacidade da lógica fuzzy de representar requisitos de QoS abstratos.

As simulações realizadas para validar o modelo utilizaram exemplo com classes EF e BE. Apesar de conceitualmente simples, as avaliações de retardo e variação de retardo são dificultadas pela imprecisão e incerteza do tráfego que entra no domínio[26]. 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 um controlador convencional.

Foram realizadas simulações variando-se o intervalo de operação do controlador e o tamanho dos pacotes transmitidos. 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 importante é o intervalo ideal que 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. Como 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 super-provisionamento possibilita a manutenção da qualidade, porém o custo para manter essa infra-estrutura extra é alto. Mostramos também que a lógica fuzzy foi adequada para tratar tráfegos com alta variação, comuns nas situações reais. A arquitetura proposta permitiu a construção de um sistema escalável e eficiente. A utilização de gerenciamento baseado em políticas facilitou a especificação dos requisitos de QoS desejados.

Como continuação do trabalho, poderá ser definido um controlador que inclua suporte a outras classes DiffServ, como AF *(Assured Forwarding)*. Essa classe segue filosofia diferente, devendo o controlador tratar variáveis distintas das consideradas neste trabalho. 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.

Outro desafio é o tratamento de redes móveis. Como é previsto que qualquer infra-estrutura de redes em um futuro próximo tenha segmentos móveis e aéreos, seria interessante avaliara aplicação da metodologia aqui proposta nesse caso. O tratamento de QoS em ambiente móvel ainda é um problema em aberto.

Outro trabalho futuro deve ser 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.

AGRADECIMENTOS

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

REFERÊNCIAS

[1] S. Blake, D. Black e M. Carlson, “An architecture for differentiated services”. RFC 2475, dezembro de 1998.

[2] M. P. Fernandez, A. de Castro P. Pedroza e J. F. de Rezende, “Quality of service in a DiffServ domain using policy-based management”, in *XVII International Teletraffic Congress (ITC’17)*, Salvador, Brazil, dezembro de 2001.

[3] M. P. Fernandez, A. de Castro P. Pedroza e J. F. de Rezende, “QoS provisioning across a DiffServ domain using policy-based management”, in *(Globecom 2001)*, San Antonio, USA, novembro de 2001.

[4] C. C. Lee, “Fuzzy Logic in control systems: Fuzzy logic controller, Part II”, *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 20, no. 2, pp. 419–435, março de 1990.

[5] F. Herrera, M. Lozano e J. Verdegay, “Tuning fuzzy logic controllers by genetic algorithms”, *International Journal of Approximate Reasoning*, vol. 12, no. 3, pp. 299–315, junho de 1995.

[6] J. Velasco e L. Magdalena, “Genetic algorithms in fuzzy control systems”, in *Genetic Algorithms in Engineering and Computer Science* (G. Winter, J. Periaux, M. Galan e P. Cuesta, eds.), pp. 141–165, John Wiley & Sons, 1995.

[7] M. Sloman, "Policy driven management for distributed systems", *Journal of Networking and Systems Management*, vol. 2, no. 4, no. 4, pp. 333–360, 1994. Plenum Press.

[8] DMTF, "Common Information Model (CIM) specification - version 2.2". Distributed Management Task Force. http://www.dtmf.org/spec/cims.html, junho de 1999.

[9] B. Moore, J. Strassner e E. Elleson, "Policy core information model". Internet Draft draft-ietf-policy-core-info-model-08.txt, outubro de 2000.

[10] E. Lupu e M. Sloman, "Conflicts in policy-based distributed systems management", *IEEE Transactions on Software Engineering, Special Issue on Inconsistency Management*, vol. 25, no. 6, no. 6, pp. 852–869, 1999. IEEE.

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

[12] N. Damianou, N. Dulay, E. Lupu e M. Sloman, "The Ponder policy specification language", in *Policy 2001: Workshop on Policies for Distributed Systems and Networks*, Bristol, UK, pp. 18–39, janeiro de 2001.

[13] E. Lupu e M. Sloman, "Towards a role based framework for distributed systems management", *Journal of Network and Systems Management*, vol. 5, no. 1, pp. 5–30, janeiro de 1997. Plenum Press Publishing.

[14] J. E. Mortensen, "JFS Fuzzy System". http://www.inet.uni2.dk/ jemor/jfs.htm, 1998.

[15] L. A. Zadeh, "Fuzzy sets", *Information and Control*, vol. 8, pp. 338–353, 1965.

[16] E. Cox, *Fuzzy logic for business and industry*. Charles River Media, outubro de 1995.

[17] O. Cordón, F. Herrera e A. Peregrín, "A practical study on the implementation of fuzzy logic controllers", *The International Journal of Intelligent Control and Systems*, vol. 3, no. 3, pp. 49–91, junho de 1999.

[18] V. Jacobson, K. Nichols e K. Poduri, "An expedited forwarding PHB". RFC 2598, junho de 1999.

[19] O. Cordón, F. Herrera e A. Peregrín, "Looking for the best defuzzification method features for each implication operator to design accurate fuzzy models", tech. rep., University of Granada, abril de 1999. Technical Report DECSAI-99108, Dept. of Computer Science and A.I., University of Granada.

[20] L. X. Wang e J. Mendel, "Generating fuzzy rules by learning from examples", *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 22, no. 2, pp. 1414–1427, julho de 1992.

[21] Z. Michaelewicz, *Genetic algorithms + data structure = evolution programs*. Springer-Verlag, março de 1996.

[22] M. Mitchell, *An Introduction to Genetic Algorithms*. MIT Press, março de 1996.

[23] K. De Jong, *An Analysis of the Behavior of a class of Genetic Adaptive System*. Tese de Doutorado, University of Michigan, 1975.

[24] S. McCanne e S. Floyd, "ns Network Simulator - Version 2". http://www.isi.edu/nsnam/ns/, 1998.

[25] H. Hsiung, M. J. Fischer, D. M. Masi, D. Cuffie e S. Scheurich, "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, julho de 1999.

[26] D. Lorenz e G. Orda, "QoS routing in networks with uncertain parameters", in *IEEE Infocom 98*, 1998.

Marcial Porto Fernandez é graduado em Engenharia Eletrônica pela Universidade Federal do Rio de Janeiro (UFRJ) em 1988, recebeu o título de mestre e doutor em Engenharia Elétrica no Grupo de Teleinformática e Automação (GTA) na COPPE/UFRJ em 1998 e 2002, respectivamente. Atualmente é pesquisador do Instituto de Computação da Universidade Federal Fluminense, Niterói. Suas áreas de interesse são Gerenciamento de Redes e Qualidade de Serviço na Internet.

Aloysio de Castro P. Pedroza é graduado em Engenharia Eletrônica pela Universidade Federal do Rio de Janeiro (UFRJ) em 1975, recebeu título de mestre em Engenharia Elétrica na COPPE/UFRJ em 1980 e doutor em Engenharia Elétrica e Ciência da Computação pela Universite Paul-Sabatier/LAAS, França, em 1985. Atualmente é professor adjunto da UFRJ, Rio de Janeiro. Suas áreas de interesse são linguagens de especificação, verificação e implementação de software e hardware de sistemas distribuídos.

José Ferreira de Rezende recebeu o diploma de Engenheiro Eletrônico e o título de Mestre em Engenharia Elétrica pela Universidade Federal do Rio de Janeiro (UFRJ) em 1988 e 1992, respectivamente, e o título de Dr. em Ciência da Computação pela Université Pierre et Marie-Curie (Paris 6), em 1997. Desde 1998 ele é professor adjunto do Programa de Engenharia Elétrica da COPPE/UFRJ. Suas atividades de pesquisa concentram-se nas seguintes áreas: aplicações multimídia distribuídas, comunicação de grupo, redes de alta velocidade, redes móveis e QoS na Internet.