Otimização de Controlador Fuzzy para Provisionamento de Recursos em ambiente DiffServ através de Algoritmo Genético

Marcial Porto Fernandez[1] — marcial@gta.ufrj.br
Aloysio de Castro P. Pedroza[1,2] — aloysio@gta.ufrj.br
José Ferreira de Rezende[1] — rezende@gta.ufrj.br

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

Resumo

A maioria dos trabalhos na área de Serviços Diferenciados (DiffServ) tratam a garantia de Qualidade de Serviço (QoS) em cada nó da rede, supondo-se que a garantia individual proporciona uma melhor qualidade de serviço em todo o domínio DiffServ(DS). No entanto, isso pode falhar quando há agregações de fluxos. Assim é necessário definir mecanismos de provisionamento de recursos para ajustar a rede de acordo com a variação de tráfego. Uma proposta para solucionar este problema é a adoção de um controlador baseado em lógica fuzzy que reconfigura os nós de um domínio de acordo com o tráfego entrante. Este trabalho propõe uma metodologia para construir um controlador fuzzy otimizado através da utilização de algoritmo de Wang-Mendel e algoritmo genético que possibilitaram obter melhoria da QoS. As funcionalidades do modelo são demonstradas através de simulação de um exemplo de aplicação voz sobre IP em um domínio DiffServ.

Abstract

The majority of work in the Differentiated Services (DiffServ) area handles Quality of Service (QoS) guarantees on a per node basis, assuming this provides QoS in the whole domain. But it could fail with data flow aggregation. Then, provisioning techniques should be used to reconfigure node parameters according traffic changes. We showed a proposal to use a fuzzy controller that reconfigures all DS nodes according to ingress traffic. This paper proposes a methodology to choose the best fuzzy controller parameters using the Wang-Mendel and a genetic algorithm to achieve the desired QoS. The functionality of a prototype are demonstrated by simulation of a voice over IP application in a DiffServ domain.

Palavras Chave: Qualidade de Serviço (QoS), Gerenciamento de Redes, DiffServ

1 Introdução

A Diferenciação de Serviços (DiffServ) [1] é uma proposta que visa oferecer garantias de qualidade de serviço (QoS) exigidas pelas aplicações multimídia, consistindo em prover serviços diferenciados para as agregações de fluxos de dados. 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. Pelo fato de não oferecer garantias de QoS para cada fluxo de dados individualmente, indica a utilização de um mecanismo para o provisionamento de recursos de rede no domínio.

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 a necessidade de crescimento, já que construir infra-estrutura de telecomunicações exige uma estimativa de tráfego futuro, que tende a ser imprecisa.

Uma proposta de provisionamento de recurso, RMD[2], utiliza um protocolo de sinalização que muda o controle de admissão do domínio conforme a entrada de tráfego. Outra proposta, o projeto Tequila[3], inclui análise estatísticas para dimensionar os recursos de rede. Nossa proposta trata da reconfiguração de cada nó do domínio, alterando os parâmetros de configuração do escalonador e condicionador DiffServ de acordo com a variação de tráfego. Esse sistema é orquestrado por um sistema de gerenciamento baseado em políticas.

Em trabalhos anteriores, propomos um controlador fuzzy que reconfigura dinamicamente os nós, fazendo que o provisionamento da rede se ajuste conforme o tráfego entrante. Esta proposta mostrou-se eficiente para melhorar a qualidade de serviço em um domínio DiffServ simples com 5 nós[4] e em um domínio DiffServ mais complexo em várias topologias aleatórias com 40 nós[5]. A lógica fuzzy foi utilizada em função das características de incerteza e imprecisão inerentes ao fluxo de dados, tendo em vista ser muito difícil estimar o tráfego em um nó da Internet. A utilização do controlador fuzzy justifica-se pela não linearidade e ausência de um modelo matemático preciso para tratar estimativa de tráfego[6]. Comparado a um controlador convencional (tipo PID), o controlador fuzzy apresenta vantagens significativas no tratamento de variáveis imprecisas.

O controlador fuzzy foi especificado através da definição das variáveis semânticas (função de pertinência) e do conjunto de regras. Pelo fato dessas variáveis serem escolhidas por um projetista, não podemos garantir a correção e a otimização do controlador fuzzy para o problema proposto[6]. Para melhorar o funcionamento do controlador fuzzy foram utilizadas técnicas baseadas em algoritmos genéticos que otimizam os parâmetros do controlador fuzzy[7, 8, 9]. Para validar o procedimento de otimização foi realizada a simulação de um protótipo, com avaliação de tempo de retardo, variação do retardo e descarte de fluxos de voz sobre IP, concorrendo com tráfegos não sensíveis ao retardo.

Este trabalho encontra-se organizado da seguinte forma: a seção 2 apresenta um resumo sobre mecanismos de provisionamento de recursos e a utilização de lógica fuzzy para melhoria de QoS; a seção 3 apresenta a arquitetura de um controlador fuzzy para um domínio DiffServ, conforme o proposto; a seção 4 apresenta a metodologia utilizada para otimização do controlador fuzzy; a seção 5 mostra a implementação do protótipo seguindo a metodologia proposta; a seção 6 apresenta os resultados obtidos na simulação; e finalmente, a seção 7 apresenta as conclusões do trabalho e sugere temas para trabalhos futuros.

2 Provisionamento de Recursos

A necessidade de garantias de qualidade de serviço na Internet indica a utilização de mecanismos de provisionamento de recursos. A característica aleatória da chegada de fluxos em diferentes classes de serviço obriga a utilização de alguma técnica de reconfiguração desses mecanismos. Mostramos a seguir duas propostas de mecanismos de provisionamento de recursos, no ambiente DiffServ, e a utilização de lógica fuzzy para melhoria de QoS.

2.1 Resource Management in DiffServ (RMD)

A arquitetura RMD, tem por objetivo oferecer reserva de recursos fim-a-fim no domínio DiffServ proposta por Westberg, Jacobson, et al[2]. Essa arquitetura inclui novos recursos de reserva à arquitetura DiffServ, como o protocolo Load Control, proposto por Westberg, Turanyi, et al [10]. A arquitetura RMD define dois conceitos estruturais: Reserva por Salto (PHR - Per Hop Reservation) e a Reserva por Domínio (PDR - Per Domain Reservation).

O protocolo PHR é usado no domínio DiffServ no âmbito do nó, como um argumento do PHB (Per Hop Behavior), para oferecer reserva de recursos. Este protocolo é implementado em todos os nós do domínio DiffServ. Por outro lado, o protocolo PDR gerencia todos os recursos de reserva no domínio, confiando no estado das reservas de recursos realizados pelo PHR em todos os nós. Esse protocolo somente é implementado nas bordas do domínio (nós de borda). Os autores citam como vantagens desta proposta a simplicidade e o baixo custo de implementação, além de sua boa escalabilidade.

2.2 Projeto Tequila

O projeto Tequila (Traffic Engineering for Quality of Service in Internet, at Large Scale) é uma arquitetura para oferecer QoS proposta por Trimintzios, Pavlou, et al [3]. O objetivo deste projeto é estudar, especificar, implementar e validar um conjunto de definições de serviço e ferramentas de engenharia de tráfego, de forma a obter garantias de QoS fim-a-fim, mediante dimensionamento cuidadoso, controle de admissão e gerenciamento dinâmico de recursos em uma rede de Serviços Diferenciados.

A arquitetura Tequila é constituída por dois grupos funcionais. O primeiro grupo inclui o Gerenciador de SLS (Service Level Specification), é responsável pela assinatura dos usuários e o controle de admissão, além da monitoração dos SLSs. O segundo grupo é responsável pelo gerenciamento de recursos de longo termo (meses ou anos), planejando os recursos físicos. Outra função importante desse grupo é o Gerenciador de Políticas, que interpreta as políticas definidas pelo operador da rede, implementa nos equipamentos e monitora seu comportamento.

2.3 Lógica Fuzzy para prover QoS

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

Guérin e Orda[11] apresentam um estudo sobre inacurácia e incerteza do estado de uma rede para garantir a QoS das conexões. Este trabalho demonstra que, quando a conexão exige apenas banda passante, a inacurácia não influencia no resultado final, porém, quando a conexão exige garantias de retardo fim-a-fim, a inacurácia da rede torna intratável o processo de seleção do caminho para garantir a QoS. Lorenz e Orda [12] mostram também que a incerteza do estado da rede dificulta a solução do problema de garantir o retardo fim-a-fim em um domínio. No entanto, eles mostram que, decompondo os requisitos de retardo em restrições locais e definindo uma classe de distribuição de probabilidade, é possível estabelecer uma solução eficiente e exata. A complexidade do algoritmo, entretanto, é muito grande para processamento no nó, obrigando então a usar uma aproximação polinomial, que é a proposta desse trabalho.

O problema de incerteza e imprecisão nos remetem à solução baseada em lógica fuzzy e realmente vários trabalhos apresentaram aplicações de controladores fuzzy, como é o caso de Li e Nahrstedt [13]. Os autores, no entanto, focaram o ambiente de configuração e não trataram profundamente o controle de recursos na rede. Cheng e Chang [14] mostram um controlador fuzzy para configurar os parâmetros de uma rede ATM. Este artigo, apesar de tratar dos parâmetros de rede, pressupõe a utilização de rede ATM, que já oferece intrinsecamente recursos de QoS. Vasilakos e Anagnostakis [15] apresentam um controlador fuzzy para determinar o melhor caminho que um pacote deve seguir, com o objetivo de oferecer garantias de QoS.

3 Controlador Fuzzy para Arquitetura DiffServ

A lógica fuzzy foi introduzida por Lofti Zedeh[16], 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, 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 [6]. 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.

Apresentamos, a seguir, o controlador fuzzy apropriado para controlar uma arquitetura DiffServ e 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].

3.1 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 parte: um controlador do escalonador existente em todos os nós do domínio e um controlador do condicionador existente apenas nos nós de borda.

3.1.1 Controlador do Escalonador

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

A primeira variável de entrada é o tempo que o pacote fica na fila EF (Expedited Forwarding). Como os demais tempos do processamento do pacote são praticamente irrelevantes, consideramos o tempo de espera na fila como o tempo total no nó. Da mesma forma que o tempo de espera do pacote, variável equivalente seria o tamanho da fila, pois é diretamente proporcional ao retardo esperado. A segunda variável de entrada é a taxa de descarte por transbordamento da fila BE (Best Effort), 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:

- Retardo instantâneo na classe EF.
- Perda de pacotes na classe BE.

Saída:

- Peso relativo no escalonador da fila EF/BE.

3.1.2 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 furado, 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:

- Perda de pacotes na classe EF no condicionador.
- Retardo máximo na classe EF nos nós de núcleo.

Saída:

- Taxa do Token Bucket do condicionador.

3.2 Base de Regras

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 expressado de forma intuitiva, usando as variáveis lingüísticas. As aplicações ideais para utilização de Controladores Fuzzy são as seguintes:

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.

A base de regras é o conjunto de regras SE-ENTÃO dos valores lingüísticos, que representam o comportamento desejado do sistema fuzzy. Conforme dito anteriormente, a base de regras pode ser definida de acordo com a política administrativa. Para nosso exemplo, utilizamos um conjunto de regras 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 equação 1. O resultado discreto é obtido a partir do valor semântico através da equação 2. Apesar da necessidade de um cálculo integral, que pode ser computacionalmente pesado em caso de função complexa, o controlador proposto não apresenta o problema, devido ao uso de funções de pertinência triangulares e trapezoidais.

$$\tilde{Y} = \bigcup_{i=1}^{j} \tilde{Y}_i \tag{1}$$

$$\mu_{\tilde{Y}} = min \left[\mu_{\tilde{Y}_1},\ \mu_{\tilde{Y}_2},\ \mu_{\tilde{Y}_3},\ \ldots\ \mu_{\tilde{Y}_n} \right]$$

$$Y^* = \frac{\int_0^1 \ \mu_Y(x) \ x \ dx}{\int_0^1 \ \mu_Y(x) \ dx} \tag{2}$$

4 Otimização do Controlador Fuzzy

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, a serem definidas pelo projetista do sistema. É desejável a introdução de metodologia para produzir funções de pertinência e regras coerentes e eficientes, sem depender do talento do projetista.

Com o objetivo de especificar um controlador eficiente, utilizamos algumas ferramentas de otimização. Na criação de regras, usamos o algoritmo Wang-Mendel, 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.

4.1 Definição de Regras através do algoritmo de Wang-Mendel

O controlador fuzzy, que é baseado em uma base de regras, pode apresentar problemas de funcionamento quando uma regra errada é incluída em sua base. Para definir as regras do controlador evitando a ocorrência de erros utilizamos o algoritmo de Wang-Mendel [20]. Seu funcionamento básico é apresentado no algoritmo 1. Com essa metodologia conseguimos construir uma base de regras a partir de uma base de conhecimento do comportamento desejado, eliminando eventuais incoerências e erros. A contradição é detectada quando a base de conhecimento induz a criação de regras divergentes, por exemplo: 1) SE retardo É alto ENTÃO saída É *alta* e 2) SE retardo É alto ENTÃO saída É *baixa*.

Algoritimo 1 Algoritmo Wang-Mendel

Início: A partir dos valores de referência criar uma regra da forma IF <var> IS <adj> AND <var> IS <adj> AND . . . THEN <var> IS <adj>.
if Regra ainda não existe e não contradiz nenhuma outra regra **then**
  Regra adicionada à base de regras
end if
if Regra já existe na base de regras **then**
  Regra ignorada, porém o contador de posto é incrementado
end if
if Regra contradiz alguma regra existente **then**
  Regra incluída na tabela de regras, porém com marcação de contradição e o contador de posto é iniciado.
  Ao final do processamento a regra contraditória com maior pontuação é aproveitada e a de menor pontuação é descartada.
end if

A maior utilidade dessa metodologia é criar um conjunto de regras consistente, já que, se forem criadas por um especialista poderá haver ocorrência de regras contraditórias e conseqüentemente, resultados errôneos.

4.2 Otimização de parâmetros com Algoritmo Genético

O Algoritmo Genético é um mecanismo baseado no processo de evolução natural de Darwin e tem se mostrado uma ferramenta valiosa para otimização. O uso de algoritmo genético para otimizar controlador fuzzy foi mostrado por Kim, Moon e Zeigler[8], Velascos e Magdalena[9], Herrera e Lozano[7].

A otimização é um processo que busca encontrar uma melhor combinação de parâmetros de um controlador fuzzy para atingir o melhor resultado possível. Podemos considerar um controlador fuzzy como uma caixa com vários botões de ajuste dos parâmetros, e o processo de otimização consiste em escolher uma combinação de valores de ajuste dos botões (parâmetros) que resulta em melhores resultados.

Algoritmos Genéticos são algoritmos de otimização global, baseados nos mecanismos de de genética e seleção natural. Eles empregam uma estratégia de busca paralela e estruturada em direção da busca de pontos de "alta aptidão", ou seja, pontos nos quais a função a ser otimizada tem valores comparativamente melhores. Apesar de aleatórios, eles são direcionados tendo como referência as informações históricas para encontrar novos pontos de busca onde são esperados melhores resultados. Isto é feito através de processos iterativos, onde cada iteração é chamada de geração.

A primeira etapa da aplicação de algoritmo genético consiste em representar genotipicamente o conjunto de parâmetros. Cada gene representa um parâmetro e pode ser um valor binário (0 ou 1), um valor inteiro ou um valor real. O genótipo é um vetor de tamanho finito e, geralmente, com um alfabeto (valores possíveis) finito. Um genótipo produz um resultado, que chamamos fenótipo (características). Associando um genótipo a um fenótipo obtemos um indivíduo.

Durante cada iteração, os operadores de seleção e reprodução são aplicados a uma população (conjunto de indivíduos), na qual a quantidade de indivíduos pode variar em função da complexidade do problema e da capacidade de processamento. Por meio da seleção escolhemos os melhores indivíduos que irão se reproduzir, gerando descendentes para a próxima geração, com uma probabilidade determinada pelo índice de aptidão. Assim os indivíduos mais aptos têm mais chance de se reproduzir.

Após uma série de iterações, obtemos um valor de parâmetros otimizado para o controlador. A grande vantagem desse procedimento é que ele escolhe os parâmetros das funções de pertinência automaticamente, sem depender do critério do projetista.

4.2.1 Operações Genéticas

A reprodução de indivíduos de uma população de dá através de operadores genéticos: seleção, cruzamento e mutação. Essas operações simulam as operações biológicas para criar uma população de indivíduos cada vez mais aptos. Apresentamos a seguir esses operadores.

Seleção O principal operador do Algoritmo Genético é o critério de seleção para possibilitar que a reprodução dos indivíduos produzam indivíduos mais aptos. A maioria dos métodos de seleção escolhem preferencialmente indivíduos com maiores notas de aptidão, mas incluem indivíduos menos aptos para manter a diversidade da população. Um método de seleção muito utilizado é o Método da Roleta.

Nesse método, cada indivíduo da população é representado como um segmento de círculo no qual o tamanho do setor é proporcional ao seu índice de aptidão. O eixo da roleta é representada por uma quantidade de vetores igual ao tamanho da população desejada e ângulos equidistantes entre vetores. Girando aleatóriamente a roleta, os vetores apontam para os indivíduos selecionados para a próxima geração. Os indivíduos com melhor aptidão terão mais chances de serem escolhidos, dada a maior porção da roleta.

Cruzamento O cruzamento é o operador responsável pela recombinação de características dos pais durante a reprodução, permitindo que as próximas gerações herdem essas características. Ele é considerado o operador genético predominante, por isso a probabilidade dada pela taxa de cruzamento, deve ser maior que a taxa de mutação. Em nossos experimentos usamos a taxa de 65%.

Mutação O operador de mutação é importante para garantir a diversidade genética da população, alterando aleatoriamente um ou mais componentes (gene) de uma estrutura. O objetivo principal do operador mutação é a introdução de novos elementos (genótipos) à população. Assim, a mutação garante que a probabilidade de se chegar a qualquer ponto do espaço de busca é diferente de zero. Uma alteração drástica no genótipo propicia a alteração da direção da busca, permitindo a saída de pontos de máximo locais localizados pelo operador de seleção. O operador de mutação é aplicado aos indivíduos com uma probabilidade dada pela taxa de mutação, que por ser um operador genético secundário geralmente é um valor pequeno. Em nossos experimentos usamos a taxa de 2%.

5 Implementação do Protótipo

A partir da metodologia 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 apropriado para diversas topologia e tráfego.

5.1 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[21]. O controlador fuzzy utilizado nesse trabalho foi desenvolvido com a ferramenta JFS, de Mortensen[22]. 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. Ela dispõe de ferramentas de otimização baseada nos algoritmos de Wang-Mendel e algoritmo genético. Após o desenvolvimento do modelo, é gerada biblioteca em código C, que implementa o controlador, sendo, então, integrada ao simulador NS.

5.2 Otimização do Controlador Fuzzy

O controlador definido com funções de pertinência arbitradas pelo projetista, mostrado na seção 3.1, e com a base de regras estabelecido a partir da base de conhecimento semântico e através do algoritmo de Wang-Mendel, mostrado na seção 4.1, 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 4.2.

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, porém o resultado ótimo é sempre atingido.

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.

A grande vantagem do algoritmo genético é que a otimização pode ser um processo contínuo, 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. Porém, nossa experiência, utilizando diversas topologias e diferentes padrões de tráfego, mostrou que as melhorias são marginais em relação a primeira otimização realizada, a partir da sugestão do projetista.

5.3 Topologia de Simulação

A aplicação voz sobre IP foi implementada com tráfego CBR e On-Off exponencial sobre protocolo UDP. O tráfego CBR é o pior caso para o desempenho da rede, por outro lado, o tráfego On-Off é mais próximo de uma conversação normal, sendo sua taxa média sensivelmente menor que no caso CBR. O tráfego de voz foi classificado em classe EF (Expedited Forwarding)[18] e o tráfego concorrente, também CBR/UDP, foi classificado na classe BE (Best Effort).

O diagrama da topologia complexa utilizada é mostrada na figura 1. Trata-se de um domínio DS constituído por 40 nós, sendo 30 nós de núcleo e 10 nós de borda (na figura indicado com números de 30 à 39). Definimos, na borda, 5 nós como entrada (30 à 34) e 5 nós como saída (35 à 39). A topologia foi criada com o pacote *gt-itm*, parte da distribuição do NS[21].

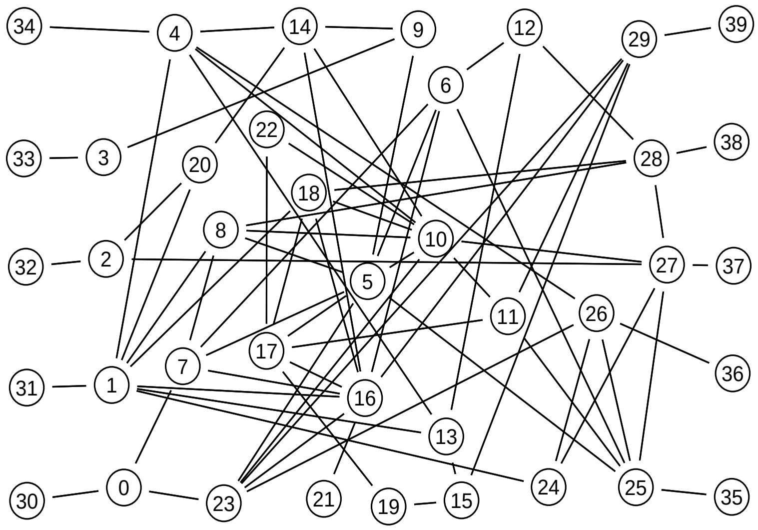


Figura 1: Diagrama da topologia de simulação

5.4 Modelo de Fonte de Tráfego

Para simular uma situação real de conexões de canais de voz, a quantidade de fontes de tráfego varia durante o tempo de simulação de 30 minutos. Com isso, verifica-se o funcionamento do controlador em uma situação realista. Essas variações seguem uma distribuição exponencial com os seguintes padrões de tempo de tráfego ativo e tráfego inativo em segundos: (120,60), (60,30), (180,60), (60,20), (120,60). Cada fonte CBR foi definida como um canal PCM, com taxa de 64 KBPS. Na fonte On-Off, foi usada taxa de 64 KBPS, com rajada de 400 ms e repouso de 600 ms, resultando em uma taxa média de 25,6 KBPS. Como essas fontes foram implementadas como Agentes do NS o valor efetivo de tráfego é de aproximadamente 80 KBPS, em razão do cabeçalho IP/UDP. O tamanho dos pacotes é de 576 bytes para ambos os casos.

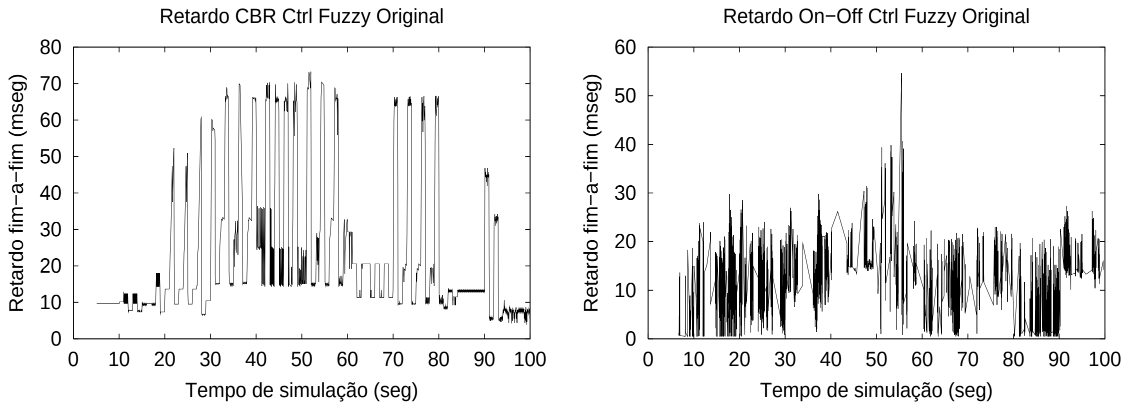
A quantidade de fontes CBR varia de 0 a 150, resultando em uma média de 93 fontes ativas durante a simulação. A quantidade de fontes On-Off varia de 0 a 350 resultando em uma média de 186 fontes ativas durante a simulação. Foram usadas, respectivamente, 250 e 450 fontes CBR 64 KBPS, como tráfego competitivo, na classe BE. Apesar de tipicamente o tráfego BE ser melhor caracterizado com TCP, utilizamos em nossos experimentos UDP para que o controle de fluxo do TCP não provocasse um bom resultado na classe EF com tráfego UDP.

O retardo de cada canal de 2 MBPS foi definido como 10ms. Todas as filas têm um tamanho máximo de 50 pacotes, produzindo um retardo máximo de 100 ms em cada nó. O modelo de simulação usou escalonador WRR, filas Drop Tail em ambas as classes e condicionador Token Bucket na classe EF, em todos os nós de borda (a classe BE não foi condicionada).

6 Resultados

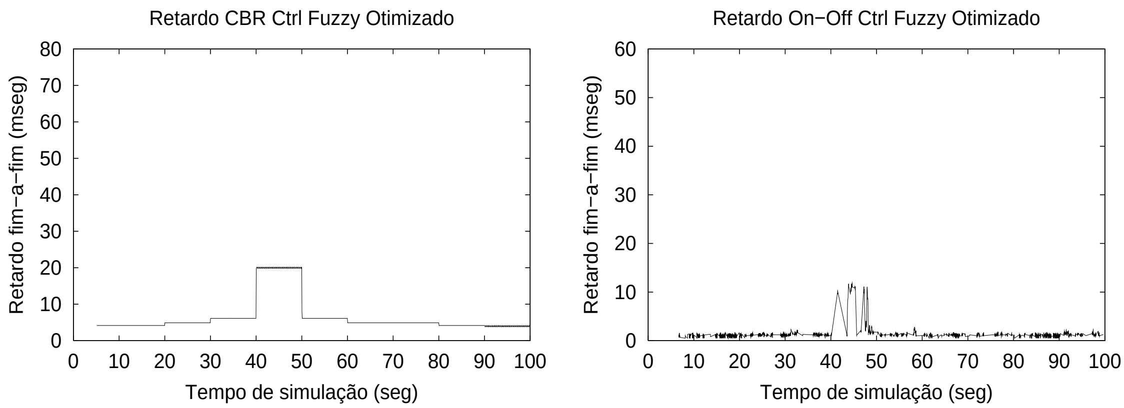
Inicialmente verificamos o processo de otimização com algoritmo genético, mostrado em 6.1. A seguir, mostramos os resultados do retardo fim-a-fim da classe EF em um domínio na seção 6.2. Nesta seção mostramos os gráficos de retardo de um fluxo e a tabela de percentil de retardo e variação do retardo 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.

6.1 Resultado da Otimização



(a) Tráfego CBR (b) Tráfego On-Off

Figura 2: Retardo fim-a-fim de um fluxo da classe EF com controlador fuzzy original



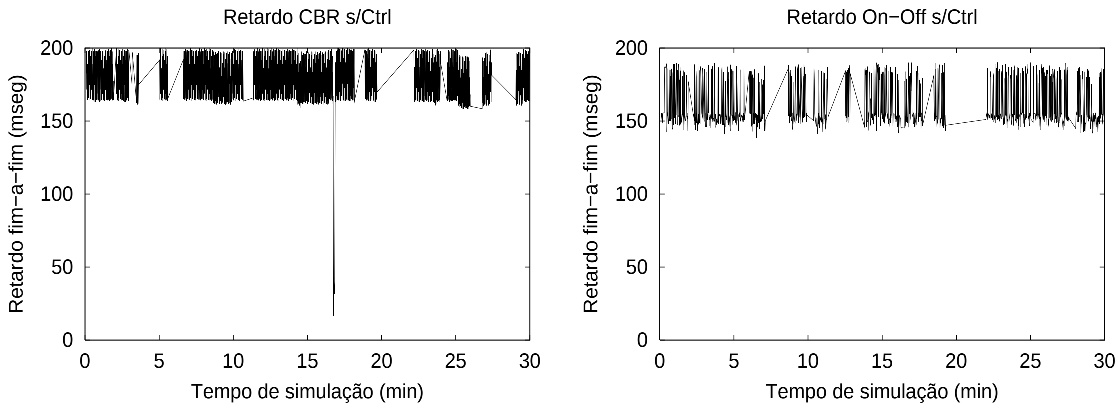
(a) Tráfego CBR (b) Tráfego On-Off

Figura 3: Retardo fim-a-fim de um fluxo da classe EF com controlador fuzzy otimizado

A verificação do processo de otimização foi verificada comparando-se o retardo do tráfego EF, CBR e On-Off, em uma topologia simples, apresentada em [4]. O gráfico da figura 2 mostra o retardo fim-a-fim de um fluxo EF com controlador fuzzy original, isto é, sem otimização. Este exemplo apresenta uma base de regras corretas, verificadas pelo algoritmo de Wang-Mendel. O gráfico da figura 3 mostra o retardo fim-a-fim de um fluxo EF com controlador fuzzy otimizado, isto é, após a otimização com uso do algoritmo genético.

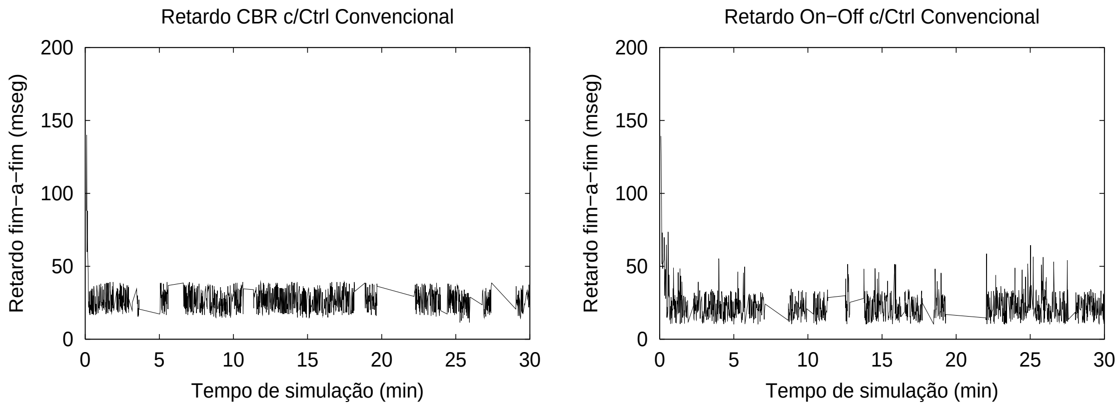
6.2 Retardo fim-a-fim da classe EF

O gráfico da figura 4 mostra o retardo fim-a-fim de um tráfego de voz da classe EF, sem controlador. O gráfico mostra apenas o tempo das filas pois o tempo de transmissão dos links, por ser constante, foi subtraído. A figura 4(a) foi obtida com tráfego CBR e a figura 4(b), com tráfego On-Off exponencial.



(a) Tráfego CBR (b) Tráfego On-Off

Figura 4: Retardo fim-a-fim de um fluxo da classe EF, sem controlador

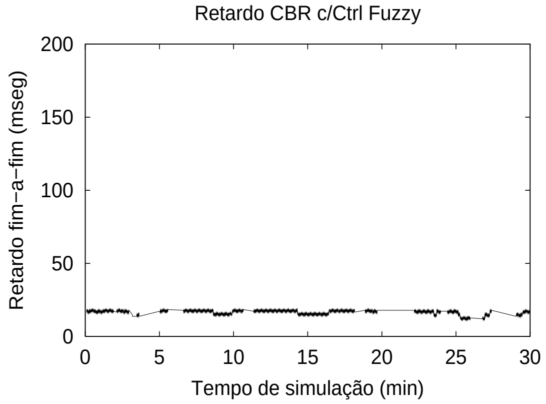


(a) Tráfego CBR (b) Tráfego On-Off

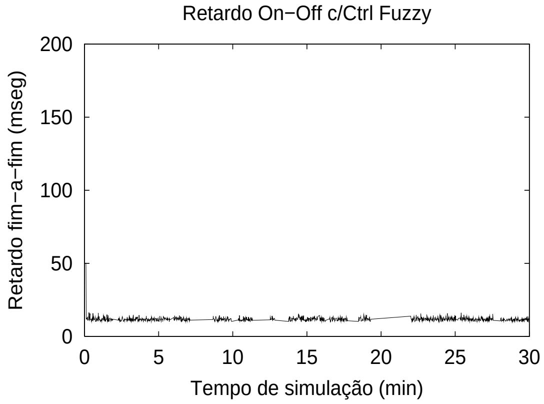
Figura 5: Retardo fim-a-fim de um fluxo da classe EF, com controlador convencional

O gráfico da figura 5 mostra o retardo fim-a-fim com um controlador convencional. Observamos na figura a melhoria do retardo comparado com o caso sem controlador. O gráfico da figura 6 mostra o retardo fim-a-fim com um controlador fuzzy. Observamos uma pequena melhoria no retardo dos pacotes, mas notamos uma sensível melhoria na variação do retardo.

A tabela 1 mostra numericamente o retardo e a variação de retardo do tráfego CBR e On-Off no domínio. A tabela confirma os resultados obtidos para um único fluxo, mostrado nas figuras.



(a) Tráfego CBR



(b) Tráfego On-Off

Figura 6: Retardo fim-a-fim de um fluxo da classe EF, com controlador fuzzy

Tabela 1: Retardo e variação do retardo da classe EF no domínio (ms)

	Média		Percentil 50		Percentil 90		Percentil 95	
Tráfego	Retar.	Varia.	Retar.	Varia.	Retar.	Varia.	Retar.	Varia.
CBR S/Ctrl	200.6	20.3	203.3	0.0	240.7	71.9	252.2	72.1
CBR Conven.	45.8	1.9	44.9	0.0	66.8	0.0	72.0	3.1
CBR Fuzzy	35.1	1.1	33.9	0.0	54.7	0.0	58.7	0.0
OO S/Ctrl	161.1	15.4	153.7	8.7	184.2	35.0	186.5	37.8
OO Conven.	25.3	12.2	23.4	9.1	40.3	27.1	50.1	35.6
OO Fuzzy	11.0	3.6	11.0	3.1	14.3	6.6	15.0	7.7

6.3 Descarte da classe EF

A tabela 2 mostra a taxa de descarte de pacotes pertencentes ao tráfego de voz, na classe EF, e o tráfego concorrente da classe BE no domínio. Observamos uma melhora sensível no descarte da classe EF, demonstrando que o controlador fuzzy não só reduz o retardo dos pacotes como reduz a taxa de descarte. Lembramos que os descartes na classe EF, mostrado nesta tabela, ocorrem somente nas bordas do domínio. Observamos também que o descarte na classe BE aumenta quando melhoramos a QoS da classe EF, o que era esperado pois os recursos de rede são limitados. Porém podemos notar que a taxa total de descarte (EF+BE) é menor com o uso do controlador fuzzy, demonstrando uma melhoria na taxa de descarte global da rede.

Tabela 2: Descarte no domínio

Controlador	EF (CBR)	BE (CBR)	EF(On-Off)	BE (On-Off)
Sem Ctrl	669897	2446937	3899233	5040731
Convencional	2138	3114725	54741	6442433
Fuzzy	15	3116753	415	6462221

7 Conclusão e Trabalhos Futuros

Apresentamos nesse trabalho, uma proposta de metodologia de construção de um mecanismo de provisionamento dinâmico, para a melhoria da qualidade de serviço em uma arquitetura DiffServ. Essa metodologia propõe um controlador que reconfigura dinamicamente os parâmetros dos equipamentos de redes, melhorando a QoS no âmbito de um domínio.

A utilização de lógica fuzzy no controlador possibilitou tratamento eficaz das imprecisões e incertezas do tráfego de entrada no domínio. O uso dos algoritmos Wang-Mendel e Genético propiciaram uma otimização nos parâmetros do controlador e a obtenção de um resultado ótimo, sem depender dos critérios de escolha do projetista. A metodologia de utilizar dados de medidas de tráfego em simulação como base de conhecimento para o algoritmo genético se mostrou eficaz, mesmo quando se utilizou topologias e modelos de tráfegos diferentes dos usados para otimização. O controlador, ainda assim, mantém sua complexidade baixa, não comprometendo a característica de escalabilidade da arquitetura DiffServ.

Como continuação do trabalho, será 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, além de utilizar outras medidas para otimização. 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 conforme a política definida.

Referências

[1] S. Blake, D. Black, and M. Carlson, "An architecture for differentiated services." RFC 2475, Dec. 1998.

[2] L. Westberg, M. Jacobsson, G. Karagiannis, and S. Oosthoek, "Resource management in diffserv (RMD) framework." Internet Draft draft-westberg-rmd-framework-00.txt, Apr. 2001.

[3] P. Trimintzios, G. Pavlou, I. Andrikopoulos, D. Griffin, C. Jacquenet, P. Georgatsos, Y. T'joens, L. Georgiadis, R. Egan, and G. Memenios, "An architectural framework for providing qos in ip differential service networks," in *VII IFIP/IEEE International Symposium on Integrated Network Management (IM 2001)*, 2001.

[4] M. P. Fernandez, A. de Castro P. Pedroza, and J. F. de Rezende, "Qualidade de serviço em um domínio diffserv através de gerenciamento baseado em políticas," in *XIX Simpósio Brasileiro de Redes de Computadores (SBRC'2001)*, (Florianópolis, Brasil), May 2001.

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

[6] C. C. Lee, "Fuzzy Logic in Control Systems: Fuzzy Logic Controller, Part II," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 20, pp. 419–435, Mar. 1990.

[7] F. Herrera, M. Lozano, and J. Verdegay, "Tuning fuzzy logic controllers by genetic algorithms," *International Journal of Approximate Reasoning*, vol. 12, pp. 299–315, June 1995.

[8] J. Kim, Y. Moon, and B. P. Zeigler, "Designing fuzzy net controllers using GA optimization," in *Proceedings IEEE/IFAC Joint Symposium on Computer-Aided Control System Design*, (Tucson (AZ), USA), pp. 83–88, Mar. 1994.

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

[10] L. Westberg, Z. Turanyi, and D. Partain, "Load control of real-time traffic." Internet Draft draft-westberg-rmd-framework-00.txt, Oct. 2000.

[11] R. Guérin and G. Orda, "Qos-based routing in networks with inacurate information: Theory and algorithms," in *IEEE Infocom 97*, (Kobe, Japan), 1997.

[12] D. Lorenz and G. Orda, "Qos routing in networks with uncertain parameters," in *IEEE Infocom 98*, 1998.

[13] B. Li and K. Nahrstedt, "A control-based middleware framework for quality of service adaptions," *IEEE Journal on Selected Areas in Communications*, Sept. 1997.

[14] R. Cheng and C. Chang, "Design of a fuzzy traffic controler for atm networks," *IEEE/ACM Transactions on Networking*, vol. 4, pp. 460–469, June 1996.

[15] A. Vasilakos and K. Anagnostakis, "Evolutionary-fuzzy prediction for strategic interdomain routing: Architecture and mechanisms," in *WCCI 98*, (Anchorage, EUA), May 1998.

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

[17] O. Cordón, F. Herrera, and A. Peregrín, "A practical study on the implementation of fuzzy logic controllers," *The International Journal of Intelligent Control and Systems*, vol. 3, pp. 49–91, June 1999.

[18] V. Jacobson, K. Nichols, and K. Poduri, "An expedited forwarding PHB." RFC 2598, June 1999.

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

[20] L. X. Wang and J. Mendel, "Generating fuzzy rules by learning from examples," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 22, pp. 1414–1427, July 1992.

[21] S. McCanne and S. Floyd, "ns Network Simulator - version 2." http://www.isi.edu/nsnam/ns/, 1998.

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