Um Controlador Robusto de Acordos de Nível de Serviço para Redes Virtuais Baseado em Lógica Nebulosa *

Hugo Eiji Tibana Carvalho, Natalia Castro Fernandes e Otto Carlos Muniz Bandeira Duarte

[1]Grupo de Teleinformática e Automação
Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro – RJ – Brasil

{hugo, natalia, otto}@gta.ufrj.br

Abstract. *The virtual network paradigm is an important proposal for the future Internet because it executes multiple protocols stacks in parallel over a single physical router to satisfy the different requirements of applications. This work proposes an efficient monitoring and controlling system for Service Level Agreements (SLAs) in multiple virtual networks. The system verifies the resource usage of the physical system; retrieves real-time profiles of virtual routers; and guarantees the SLA requirements. The control is based on nebulous logic and adequates the resource allocation according to the system overload and to the profile of routers. The control logic punishes virtual networks that exceed the contracted SLA. The punishment depends on the exceeding value and on the system charge. Results obtained from a developed prototype show the efficiency of the system under different conditions and strategic policies. It correctly allocates physical resources and meets the appropriate SLA values.*

Resumo. *O modelo pluralista de redes virtuais tem sido proposto como uma solução viável para atender os requisitos da Internet do Futuro. Neste paradigma os roteadores físicos executam em paralelo pilhas de protocolos distintas, que correspondem às redes virtuais, que permitem oferecer diversos serviços para atender aos requisitos das diferentes aplicações. Este trabalho propõe um sistema eficiente de monitoramento e controle de contratos de níveis de serviço (SLA) das múltiplas redes virtuais. O sistema analisa o uso de múltiplos recursos do roteador físico virtualizado; fornece, em tempo real, estatísticas de perfis de uso de cada elemento de rede virtualizado; e garante o cumprimento dos SLAs de cada rede virtual. O controle é baseado em lógica nebulosa e consiste em adequar a alocação dos recursos físicos disponíveis às redes virtuais segundo os perfis de uso avaliados. A lógica de controle pune as redes virtuais que apresentam perfis inadequados. A agressividade da punição depende do grau da violação e da carga total do sistema. Os resultados obtidos de um protótipo implementado comprovam a eficácia do controlador proposto, que aloca os recursos físicos de forma adequada para atender os contratos de níveis de serviços, sob diferentes estratégias de gerência.*

*Este trabalho foi realizado com recursos do FINEP, FUNTTEL, CNPq, CAPES e FAPERJ.

1. Introdução

A técnica de virtualização de recursos de um computador pessoal permite a execução de diversos sistemas operacionais sobre um mesmo *hardware* físico. Esta funcionalidade é exercida através de um domínio de gerência, que se responsabiliza por multiplexar o acesso ao *hardware* e prover fatias lógicas de recursos para os sistemas virtualizados. A virtualização oferece uma série de benefícios como o isolamento entre os sistemas virtualizados, o melhor aproveitamento dos recursos físicos e a facilidade de remapear os recursos físicos disponíveis entre os sistemas virtualizados. Uma das plataformas de virtualização mais utilizadas atualmente é o Xen [Barham et al., 2003], de código aberto. A arquitetura Xen permite a inovação em redes de computadores, pois cada máquina virtual pode ser um roteador virtual e assim um roteador físico pode suportar múltiplos roteadores virtuais com propriedades e pilhas de protocolos distintas, o que vai de acordo com a abordagem pluralista de redes [Fernandes et al., 2010].

No Xen, o operador do sistema pode utilizar algumas primitivas de controle como a criação e remoção de roteadores virtuais, a atribuição de diferentes prioridades e o mapeamento de processadores virtuais em processadores físicos. Atualmente, todas as propriedades citadas são realizadas de forma manual, o que implica em um grande problema de escalabilidade e de gerência. A alocação dinâmica de recursos em sistemas virtualizados é um desafio que tem recebido atenção especial em pesquisas recentes.

A maioria dos trabalhos de alocação dinâmica de recursos físicos em sistemas virtualizados concentra-se na aplicação de consolidação de servidores de *data centers*. O *Sandpiper* [Wood et al., 2007] é um sistema que monitora máquinas virtuais em um *data center* e migra de forma inteligente estas máquinas virtuais para outros servidores físicos com o objetivo de otimizar o desempenho de cada um deles e garantir o funcionamento das máquinas virtuais quando estas sofrem alguma mudança de comportamento, como, por exemplo, o aumento do número de acessos promovido por um *flash crowd*. O *Sandpiper* monitora os perfis das máquinas virtuais através de séries temporais e para estimar o volume de sobrecarga, os autores propõem a métrica de volume de uso, Vol, expressa por

$$Vol = \frac{1}{1 - cpu} * \frac{1}{1 - net} * \frac{1}{1 - mem}, \tag{1}$$

onde cpu é o uso de processamento, mem é o uso de memória e net o uso de rede. Meng *et al.* propõem algoritmos que fornecem a melhor distribuição de servidores virtualizados em uma malha de servidores físicos [Meng et al., 2010]. Os servidores virtuais são instanciados em servidores físicos de forma a diminuir a distância entre servidores que veiculam dados entre si, otimizando assim a escalabilidade da rede e o melhor aproveitamento da banda passante disponível nos enlaces de comunicação. Menascé *et al.* aplicam técnicas de computação autonômica para controlar o compartilhamento de processadores entre máquinas virtuais [Menasce e Bennani, 2006]. Propõe-se um algoritmo de alocação dinâmica de processamento para máquinas virtuais que é avaliado através de simulações. Xu *et al.* propõem mecanismos de controle baseados em lógica nebulosa para otimizar a alocação de recursos em *data centers* e realizam testes de desempenho para servidores web virtualizados com diferentes cargas de trabalho [Xu et al., 2008]. O controlador nebuloso é utilizado como mecanismo de aprendizado para entender o comportamento de servidores Web sob diferentes cargas.

Um procedimento que permite a alocação dinâmica de recursos em sistemas que utilizam técnicas de virtualização é a migração de máquinas virtuais, que é utilizada no *Sandpiper* [Wood et al., 2007]. Através desta primitiva de virtualização é possível migrar máquinas virtuais para diferentes servidores físicos, permitindo assim a manutenção preventiva e a economia de energia gerada através da reorganização e desligamento de servidores físicos sub-utilizados. A funcionalidade de migração em roteadores virtuais é problemática por que na maioria dos procedimentos de migração existe um período no qual ocorrem perdas de pacotes. Wang *et al.* propõem um mecanismo de migração ao vivo de roteadores virtuais sem perda de pacotes [Wang et al., 2007] e Pisa *et al.* implementam esta proposta na arquitetura Xen [Pisa et al., 2010].

A alocação dinâmica e o controle dos recursos alocados em roteadores virtuais na arquitetura Xen é um desafio porque a tecnologia de virtualização de dispositivos de entrada e saída ainda é precária e, portanto, roteadores virtuais maliciosos podem influir no desempenho de outros roteadores virtuais por falta de isolamento. Para solucionar este problema foi proposto o XnetMon [Fernandes e Duarte, 2010] que é um sistema seguro de controle de tráfego de roteamento. O XNetMon tem como ideia-chave a separação dos planos de dados e de controle e, desta forma, limita o tráfego de roteadores virtuais que violam determinadas restrições de uso.

Este artigo propõe um controlador dinâmico de recursos baseado em acordos de níveis de serviço (SLA) de redes virtualizadas. O controle se baseia na geração e análise de perfis de uso de cada um dos roteadores virtuais e na detecção e na punição, em tempo real e de forma autonômica, de violações de contratos de níveis de serviço. O sistema de controle proposto neste artigo monitora os valores de carga de cada rede virtual e gera estimativas reais de perfis de uso de cada um dos roteadores. Estes perfis são usados para garantir a melhor organização dos roteadores de forma a diminuir a chance de possíveis sobrecargas futuras. O cálculo da carga é baseado no *Sandpiper* mas envolve outros parâmetros importantes que não são considerados no *Sandpiper*, como a robustez do sistema e a temperatura de operação, além de outras que podem ser adicionadas se desejado. O sistema de controle nebuloso proposto visa facilitar a tarefa de configuração de roteadores virtuais de forma a permitir um ajuste fácil e flexível dos pesos de cada um dos parâmetros. A lógica nebulosa é usada para mapear as estratégias dos administradores da rede e permitir que estas sejam aplicadas no cálculo das punições dos roteadores que violarem os contratos estabelecidos. Neste sentido, a proposta permite o controle de SLAs em diferentes níveis de política e com diferentes estratégias. Além disso, o administrador de rede pode facilmente inserir novas regras e estratégias de atuação. O sistema de controle nebuloso proposto suporta a separação dos planos de controle e dados e, portanto, é compatível com a implementação da migração de roteadores virtuais sem perdas de pacotes e da técnica de controle do XNetMon. Assim, a funcionalidade de alocação de recursos através da migração de roteadores virtuais funciona de forma complementar ao controle desenvolvido e é garantido o isolamento entre as diversas redes virtuais.

Os resultados obtidos de um protótipo desenvolvido demonstram o funcionamento do sistema, os perfis gerados para cada um dos roteadores virtualizados e os mecanismos de gerência de estratégias e políticas que permitem flexibilidade na gerência da rede. A sobrecarga de gerência induzida no sistema é pequena, de aproximadamente cinco porcento de um núcleo de um processador Intel Core i7 860 para o monitoramento e gerência

dos múltiplos parâmetros de um roteador virtual, o que permite o controle de diversos roteadores virtuais simultaneamente. Os controladores desenvolvidos atendem de forma adequada os níveis de serviço contratados e garantem uma distribuição justa de recursos, que varia de acordo com a sobrecarga do sistema.

O restante do artigo é estruturado da seguinte forma. A Seção 2 descreve os diversos elementos do sistema, desde a geração de perfis de uso até o sistema flexível de estratégias e políticas e os controladores de carga e punição. A Seção 3 apresenta os resultados obtidos com o sistema, que demonstram a baixa sobrecarga de processamento do sistema e o funcionamento do controlador nebuloso sob diferentes variáveis de ambiente. A Seção 4 apresenta as conclusões e direções futuras deste trabalho.

2. O Sistema de Controle Proposto

O sistema de controle proposto visa monitorar e atender acordos de níveis de serviço (SLA) de redes virtualizadas. A ideia-chave baseia-se na geração e análise de perfis de uso de cada um dos roteadores virtuais e na detecção e na punição, em tempo real e de forma autônoma, de violações de contratos de níveis de serviço. No ato da detecção da violação, são calculados parâmetros que ponderam o grau de punição aplicado a cada roteador virtual, de acordo com o estado atual do sistema e da gravidade da violação. O estado global dos roteadores virtuais e do domínio de controle é caracterizado também através de um controlador nebuloso, que pondera os parâmetros relacionados a processamento, memória, rede, temperatura e robustez (existência de mecanismos de redundância e tolerância a falhas) no cálculo da carga do sistema. De acordo com a saída do controlador, pode-se modificar dinamicamente as estratégias de punição e a tolerância do sistema a determinadas ações.

O sistema proposto segue um modelo de gerência distribuído composto por agentes controladores. Cada agente controlador está associado e controla um conjunto de roteadores físicos nos quais executam os roteadores virtuais, como pode ser visto na Fig. 1. Cada agente controlador pode ser associado a um determinado número de domínios, cabendo ao gerente de rede decidir de que forma será feita a associação. Por sua vez, todo roteador físico possui um domínio de controle no qual um *daemon* de monitoramento e controle é responsável por monitorar os recursos físicos alocados a cada roteador virtual, verificar em tempo real o cumprimento dos níveis de serviço contratados e gerar os perfis de uso de cada um dos roteadores virtuais. Os agentes controladores possuem cinco módulos. O módulo de estratégias e políticas (MEP) possui as estratégias de gerência que podem ser aplicadas sobre os roteadores físicos sob seu domínio e atualiza a estratégia vigente nos *daemons*de cada domínio de controle. O módulo níveis de serviço (MNS) mantém uma base de dados que associa os níveis de serviço acordados para cada máquina virtual. O módulo base de conhecimento (MBC) armazena o histórico, os perfis de uso dos roteadores virtuais e a descrição das violações realizadas por cada roteador virtual assim como a gravidade delas. Este módulo pode tanto servir para estimar futuras migrações quanto para re-negociar contratos, adaptando os contratos a realidade de cada roteador virtual. O módulo atuador age nos *daemons* e obtém os perfis e estatísticas de cada roteador. Os controladores possuem o módulo de comunicação (Comm), que permite a troca de informações de controle com outros controladores, através de canais seguros. Se necessário, os controladores podem utilizar estes canais para negociar trocas de domínios de atuações e negociar a migração de elementos virtuais.

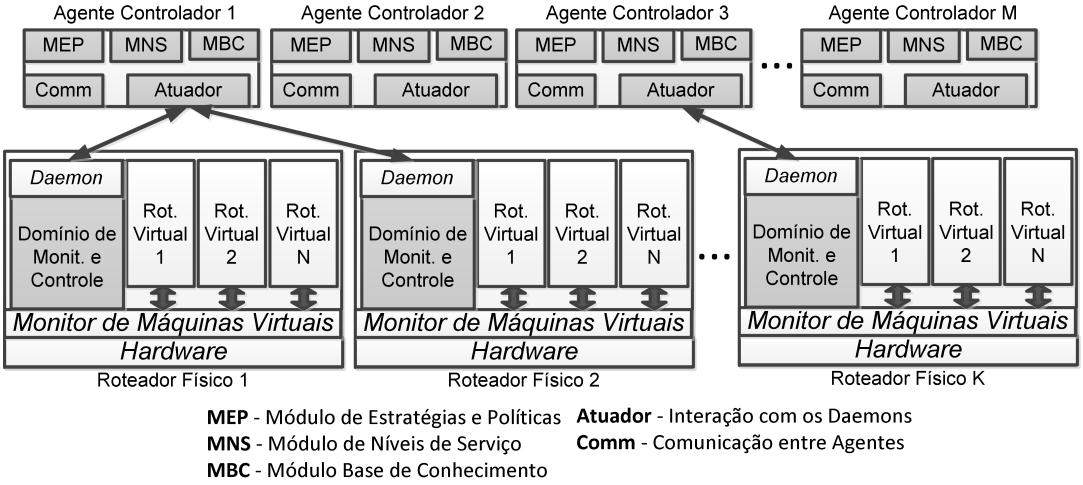
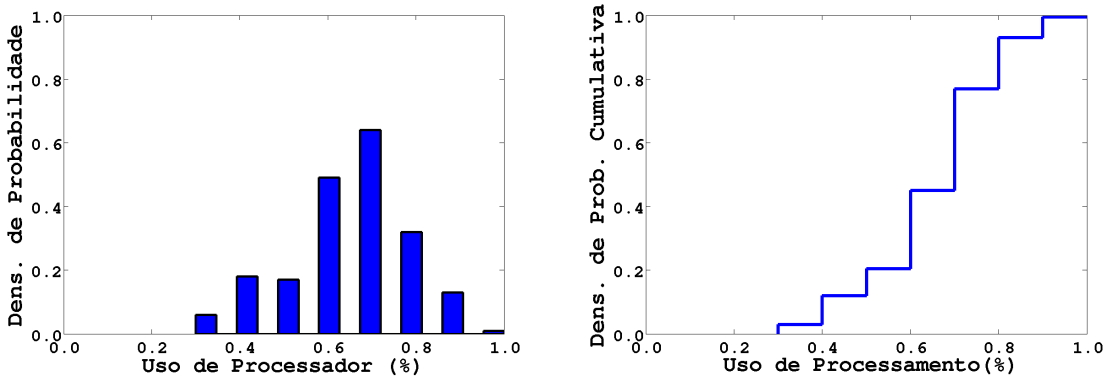


Figura 1. Arquitetura do sistema de controle. Módulos de agentes controladores distribuídos interagem com módulos de domínio de monitoramento e controle nos roteadores físicos.

O sistema descrito possui três mecanismos principais. O primeiro deles é o mecanismo de cálculo de perfis, que permite o cálculo de estatísticas de uso e a detecção de violações de SLA. Estas informações são armazenadas no módulo base de conhecimento (MBC) para, por exemplo, renegociar um SLA mal configurado. Se um determinado roteador sempre realiza uma dada violação em um dado período, é interessante que o sistema armazene esta informação para uma futura renegociação de SLAs. O segundo mecanismo é o estimador de carga do sistema, que fornece uma saída que combina múltiplos recursos em uma estimativa de carga no intervalo $[0, 1]$. O terceiro mecanismo do sistema é o mecanismo de punição adaptativa. Baseado na sobrecarga do sistema e nos perfis de uso, o mecanismo utiliza um controlador nebuloso que atribui um grau de punição proporcional ao estado do sistema. Por exemplo, se o sistema está com uma baixa carga, uma violação de ordem média (por exemplo, ultrapassar em 20% a SLA estabelecida) gera uma punição baixa (reduzir em 2% o percentual de uso de um determinado recurso). Por outro lado, se o sistema está sobrecarregado, com escassez de recursos, até uma pequena violação pode ser punida de forma severa.

2.1. Gerando Perfis de uso

Os perfis de uso de cada roteador virtual representam o padrão de consumo de recursos de cada um dos roteadores virtuais. Os perfis podem ser utilizados tanto para detectar violações de regras quanto para estimar o consumo e prever necessidades futuras de cada roteador. No protótipo, estes perfis são gerados através da captura da variação no tempo dos parâmetros de processamento, memória e rede. São utilizadas duas janelas deslizantes com tamanhos distintos, para armazenar tanto o passado recente das medidas quanto o passado longo. A geração de perfis de consumo baseados em funções de densidade de probabilidade é utilizada em [Wood et al., 2007]. No protótipo, o passado recente é utilizado na verificação dos níveis de serviço e o passado longo é utilizado para prever comportamentos futuros e estimar possíveis demandas. Estas janelas deslizantes auxiliam na detecção da correlação temporal entre as medidas. Para auxiliar na detecção destes padrões, algumas distribuições de probabilidade são geradas.



(a) Função densidade de probabilidade (PDF).

(b) Função densidade de probabilidade cumulativa (CDF).

Figura 2. Perfis de uso de processamento para um roteador virtual executando o protocolo RIPv2.

Uma função distribuição de probabilidades (PDF) do consumo de CPU de um roteador virtual que executa o protocolo de roteamento RIPv2, gerada pelo sistema implementado, pode ser vista na Fig. 2(a). Pode-se perceber que a troca de mensagens de controle e de dados deste roteador em específico gerou um consumo de aproximadamente 0.7% de CPU em 70% das medidas da janela longa de tempo observada, que foi de 200 medidas neste cenário hipotético. Devido a esta propriedade observada, pode-se concluir que seria viável agregar um grupo de roteadores RIP com propriedades semelhantes (por exemplo, 10 roteadores) e alocar um processador para ser compartilhado somente entre eles, sem que houvesse perdas de desempenho na troca de mensagens de controle e de dados. Outra proposta interessante da implementação é a utilização de funções cumulativas de distribuição de probabilidades (CDFs) para verificar níveis de serviço (*Service Level Agreements* - SLA). Através desta perspectiva, pode-se pensar em SLAs flexíveis. Por exemplo, pode-se definir que um roteador virtual pode utilizar 0.7% de um determinado recurso por até 80% do tempo. Através do cálculo da CDF da janela curta, é possível determinar por exemplo que o roteador virtual da Fig. 2(b) atenderia a SLA exemplificada anteriormente.

2.2. Módulo de Estratégias e Políticas

O módulo de estratégias e políticas (MEP) é responsável por armazenar as diversas estratégias de atuação possíveis, assim como mapear as decisões administrativas em ações e estratégias. Foi adotado o uso de controladores nebulosos [Kecman, 2001] por serem mais apropriados para problemas de tomada de decisão que envolvem incertezas, imprecisões ou valores qualitativos, como por exemplo, as estratégias de um administrador ou gerente de rede. Na lógica nebulosa um elemento pertence a um grupo de acordo com um grau de pertinência dentro do intervalo contínuo $[0, 1]$, onde $\mu_A(x) : X \rightarrow [0, 1]$ define uma função de pertinência Neste artigo, foi adotado o método de implicação Mamdani, com as operações de AND e OR de Zadeh [Zadeh, 1965] e o método centróide de *defuzzificação*. Os controladores nebulosos possuem baixa complexidade computacional e podem ter etapas de inferência paralelizáveis, aumentando o desempenho do sistema e diminuindo o tempo de reação.

O módulo de estratégias e políticas (MEP) suporta de diversas estratégias de atuação. Cada estratégia é composta de um conjunto de regras de inferência e um conjunto de funções de pertinência que associam os parâmetros de entrada com parâmetros da percepção do gerente de rede (uso de processador alto, carga de memória baixa, etc.) e um conjunto de funções de pertinência que regulam a saída do sistema. No sistema proposto, existem duas possíveis estratégias. As estratégias de estimação de sobrecarga do sistema e as estratégias de punição. Estas duas estratégias e os pacotes de estratégia são descritos na sub-seção 2.3. Estas estratégias formalizam um comportamento computacional que reflete a vontade e as estratégias do gerente de rede.

2.3. Estimando a Carga do Sistema nos *Daemons* de Controle

A carga do sistema é uma medida que determina o nível de carga do *hardware* gerenciado. Graças a ele, é possível detectar a saturação de recursos. O cálculo da carga do protótipo do sistema implementado envolve múltiplos parâmetros como o consumo de recursos de processamento, memória e rede assim como a temperatura de componentes do sistema e a robustez do sistema, que reflete a existência de mecanismos de redundância de disco, fontes de energia extras, etc. Foi definido o conjunto de funções de pertinência μ_{Proc}, μ_{Mem}, μ_{Net}, μ_{Temp} e μ_{Rob}, que associa cada um dos recursos ou parâmetros em variáveis *fuzzificadas*. A combinação destes parâmetros gera uma saída definida como carga do sistema, que possui valores no intervalo $[0, 1]$ e representa a carga dos recursos físicos do sistema. Este valor é utilizado como parâmetro de entrada de outro controlador, junto com o delta, que representa a diferença entre os recursos acordados e os recursos consumidos por um roteador, para estimar o grau de punição dos roteadores que violam os SLAs. Um diagrama em blocos do módulo de estratégias e políticas (MEP) para a carga do sistema é mostrado na Fig. 3.

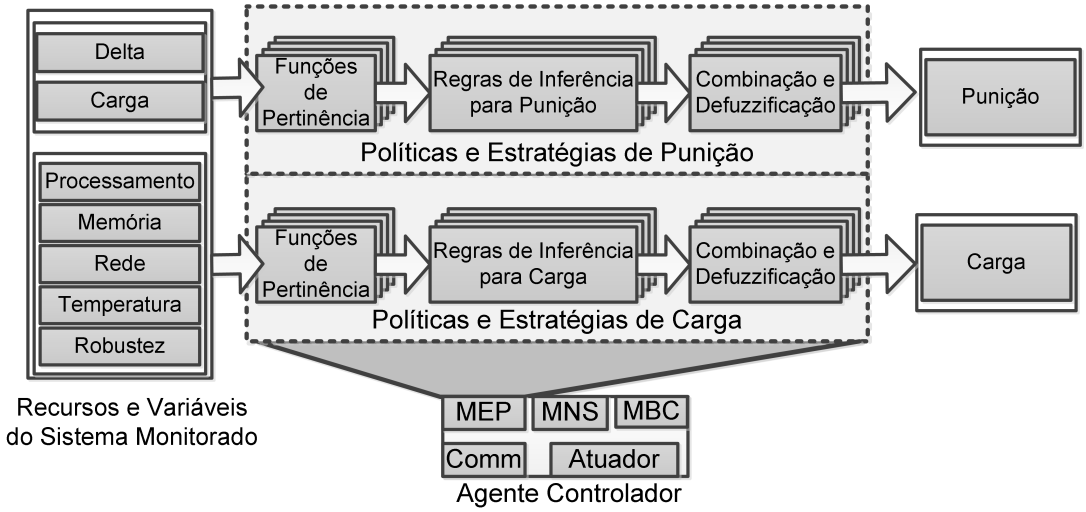
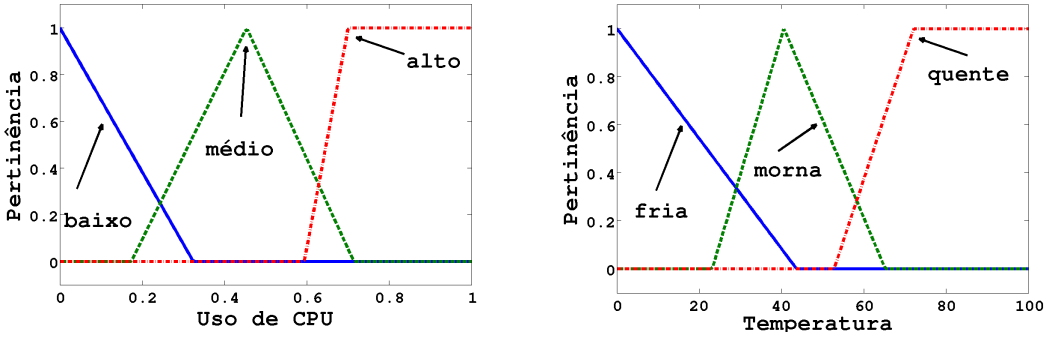


Figura 3. Módulo de Estratégias e Políticas

O consumo de recursos de cada um dos roteadores virtuais é agregado para gerar as entradas do controlador. Um exemplo de possíveis configurações para avaliar o uso de processamento e a influência da temperatura do sistema pode ser visto na Fig. 4. É importante lembrar que as curvas apresentadas refletem a configuração de um gerente de rede em particular. Elas podem ser modificadas de acordo com as necessidades e premissas

de cada gerente. As curvas baixo, médio, alto, fria, morna e quente são funções de pertinência. Nesta configuração, utilizaram-se três funções de pertinência para mapear cada um dos recursos. Percebe-se que o estabelecimento das funções de pertinência representa o mapeamento de parâmetros qualitativos do gerente. Pode-se pensar em uma tomada de decisão baseada nestes parâmetros. Por exemplo, se o uso de processador é alto e o sistema está quente, então a sobrecarga do sistema é alta.



(a) Funções de Pertinência para o uso de processamento.

(b) Funções de pertinência para a temperatura.

Figura 4. Funções de pertinência para avaliar CPU e Temperatura.

2.3.1. Estratégias Baseadas em Regras de Inferência

As estratégias dos controladores *fuzzy* são baseadas no sistema padrão de regras nebulosas. Estas regras nebulosas seguem o padrão SE $\rightarrow$ ENTÃO que representam a estratégia atual de atuação. Este conjunto de regras que definem uma estratégia é definido como um pacote de estratégia. Um pacote de estratégias exemplo que calcula o grau de punição de acordo com o delta (diferença entre o SLA acordado e o consumo real) e a carga do sistema pode ser visto na Tabela. 1.

Pacote de Estratégia
Se **Delta** (baixo) e **Sobrecarga** (baixa) Então **Punição** (baixa)
Se **Delta** (médio) e **Sobrecarga** (baixa) Então **Punição** (baixa)
Se **Delta** (alto) e **Sobrecarga** (baixa) Então **Punição** (média)
Se **Delta** (baixo) e **Sobrecarga** (alta) Então **Punição** (média)
Se **Delta** (médio) e **Sobrecarga** (alta) Então **Punição** (alta)
Se **Delta** (alto) e **Sobrecarga** (alta) Então **Punição** (alta)

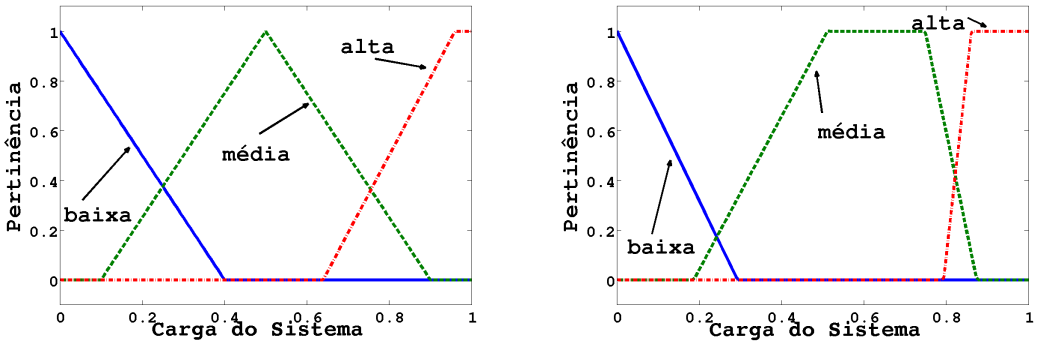
Tabela 1. Exemplo de um pedaço de um pacote de estratégias.

No pacote apresentado, o gerente de rede estabeleceu que quando o sistema está com uma carga baixa, mesmo violações grandes de SLA não serão punidas de forma severa, pois o sistema possui abundância de recursos e neste momento, disponibilizar uma quantidade de recursos adicionais ao roteador não atrapalharia o funcionamento dos demais roteadores virtuais. Quando o sistema está sobrecarregado, o gerente é menos

tolerante e até violações leves são tratadas com rigor. Estas estratégias funcionam em conjunto com as funções de pertinência, que também podem ser modeladas pelo gerente de rede. Assim é possível inserir novas estratégias e políticas de forma simples, cabendo ao agente de controle responsável exportar o pacote de estratégia alvo para o *daemon* que deve adotar a estratégia. Pode-se estabelecer diferentes estratégias para cada um dos recursos controlados, aumentando assim a flexibilidade do controlador.

2.3.2. Políticas de Carga

Após a aplicação das regras de inferência, são gerados valores *fuzzy* que representam o grau de pertinência de cada uma das regras de inferência e é feito um mapeamento entre estas regras e a saída do controlador, que é um valor no intervalo $[0, 1]$ que representa a carga do sistema. Na Fig. 5, pode-se observar duas políticas de carga possíveis, uma conservadora e uma agressiva. Dependendo do perfil do gerente de rede, ele pode escalonar as políticas atuais de cada domínio.



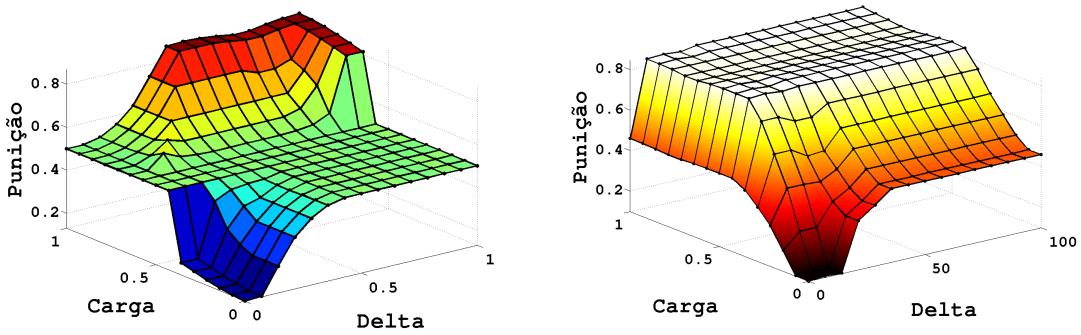
(a) Política conservadora. (b) Política agressiva.

Figura 5. Políticas de carga do sistema

Dado que as configurações e funções de pertinência foram estabelecidas, é possível verificar a relação gerada entre o grau de punição, a sobrecarga do sistema e o delta, que corresponde ao grau de violação da SLA, ou a diferença entre a quantidade de recursos utilizada e a permitida. Esta relação pode ser analisada na Fig. 6. Nestas superfícies, pode-se perceber que a política é refletida no grau de variação da punição. Além disso, a combinação de diferentes regras de inferência e funções de pertinência geram superfícies diferentes. No caso da política conservadora, observa-se que as punições só são severas quando o delta é muito elevado e o sistema se encontra saturado (Fig. 6(a)). Na política agressiva, pequenos aumentos na sobrecarga e no delta geram grandes punições (Fig. 6(b)).

2.3.3. Controle da Sobrecarga do Sistema dos Níveis de Contrato

O sistema desenvolvido é capaz de gerar perfis de uso, avaliar se os perfis correspondem às SLAs negociadas, gerar estimativas de sobrecarga do sistema e punir de forma adaptativa os roteadores virtuais que desrespeitarem as regras propostas. Na



(a) Superfície de decisão gerada para uma política conservadora.

(b) Superfície de decisão gerada para uma política agressiva.

Figura 6. Superfícies de decisão para diferentes estratégias de gerência.

implementação, o *daemon* que executa em cada domínio de controle realiza a coleta dos parâmetros dos recursos a cada intervalo de tempo, que pode ser definido pelo gerente. A partir dos parâmetros, são geradas as séries temporais que representam o consumo de cada recurso, assim como as distribuições que permitem a verificação dos perfis e o cumprimento das SLAs. Estas informações são enviadas ao agente controlador responsável. O *daemon* verifica se os perfis de uso de cada roteador virtual condizem com as SLAs negociadas. Além disso, ele agrega os recursos consumidos por cada roteador virtual para estimar a sobrecarga total de cada sistema físico. Caso algum roteador viole os contratos, é calculado um valor que representa o delta, que é a diferença entre os recursos consumidos e os recursos que de fato podiam ser consumidos. O sistema então utiliza este delta e o valor de carga do sistema obtido para tomar a decisão de qual é o grau adequado de punição que deve ser aplicado no roteador. No caso da arquitetura do Xen, um parâmetro de controle utilizado é o *caps*. O *caps* regula a quantidade de processamento que cada elemento virtual pode utilizar. Desta forma, ao manipular o *caps* de forma inteligente, é possível controlar o uso de recursos de processamento de cada um dos roteadores virtuais. Outra ferramenta de controle utilizada é o *Traffic Control* (TC), que permite o controle de filas, gerenciando assim a vazão de cada um dos roteadores virtuais, caso estes ultrapassem as SLAs definidas.

3. Resultados

O sistema de controle desenvolvido é eficiente e flexível. No decorrer do artigo, foram demonstradas as diversas saídas do sistema, como os pacotes de estratégias testados, as superfícies de decisão fornecidas pelo sistema e os perfis de uso de múltiplos parâmetros dos roteadores virtuais e dos servidores reais. Para validar o funcionamento do sistema, foram desenvolvidos alguns experimentos que tiveram o objetivo de comprovar a baixa sobrecarga de gerência do sistema e o funcionamento eficiente do controle nebuloso adaptativo de SLAs. Os testes foram realizados em uma máquina física com um processador core i7 860 com 4 núcleos reais e 8GB de memória RAM DDR3. Esta máquina foi configurada com o *hypervisor* Xen 4.0. Os roteadores virtuais foram configurados com 128MB de memória RAM e acesso a um processador virtual. Os roteadores virtuais e o domínio de controle foram configurados com o Debian Lenny e o kernel 2.6.32-5-amd64 com os *patches* de suporte ao Xen.

Um foco do desenvolvimento foi a minimização da sobrecarga de processamento induzida pela gerência das máquinas virtuais. Para avaliar a sobrecarga de gerência no domínio de controle, foram criados diversos roteadores virtuais. Em seguida, foi medido o consumo de processamento no domínio de controle quando este gerencia um número variável de roteadores virtuais. Os resultados obtidos podem ser visualizados na Fig. 7, que representa o consumo de processamento no domínio de controle em função da quantidade de roteadores monitorados. Nesta configuração, as medidas e as decisões foram coletadas e avaliadas a cada segundo. Os valores do gráfico representam o consumo médio de processamento para cada uma das configurações, com um intervalo de confiança de 95%. Observa-se que a relação entre o consumo e o número de roteadores monitorados é linear, e que mesmo em situações onde o *daemon* gerencia simultaneamente oito roteadores virtuais, a carga gerada é pequena e se aproxima de 40% de uso de um núcleo de um processador.

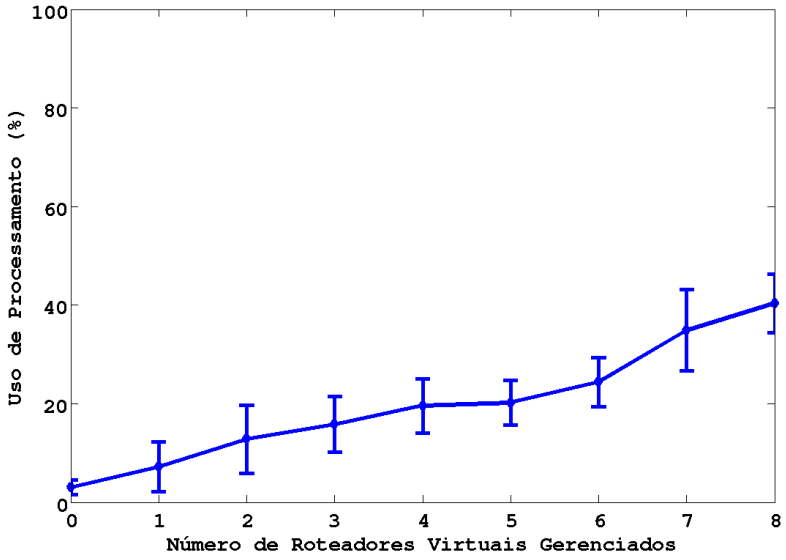


Figura 7. Uso de processamento no domínio de controle em função do número de roteadores virtuais gerenciados.

Este resultado é bom, visto que o sistema está monitorando múltiplas variáveis de múltiplos roteadores simultaneamente e, além disso, está gerenciando as SLAs de cada um destes roteadores. Pode-se estimar que um domínio de controle, de configurações semelhantes às utilizadas no teste, que reserve um de seus processadores para a tarefa de gerência e monitoramento poderá gerenciar até vinte roteadores virtuais ao mesmo tempo através do sistema proposto.

O segundo experimento avalia o funcionamento do controlador proposto e do mecanismo de punição nebuloso. Para realizar este teste, um dos roteadores virtuais foi selecionado. O contrato deste roteador define, dentre outras regras, que o roteador pode utilizar até 85% de processamento para efetuar o encaminhamento de pacotes. Em seguida é criado um fluxo de pacotes que é encaminhado pelo roteador. Ao encaminhar este tráfego, ocorre a violação dos níveis de serviço contratados. O roteador extrapola a sua SLA e o sistema de controle proposto regula através da primitiva do *caps* a quantidade de

processamento que pode ser utilizado pelo processador. Para este experimento, o intervalo entre as verificações de SLAs foi definido como um segundo. Além disso, o sistema de punição foi habilitado quando o roteador já estava consumindo uma quantidade de processamento que extrapolava o seu limite. É importante lembrar que o intervalo entre verificações é regulável no sistema.

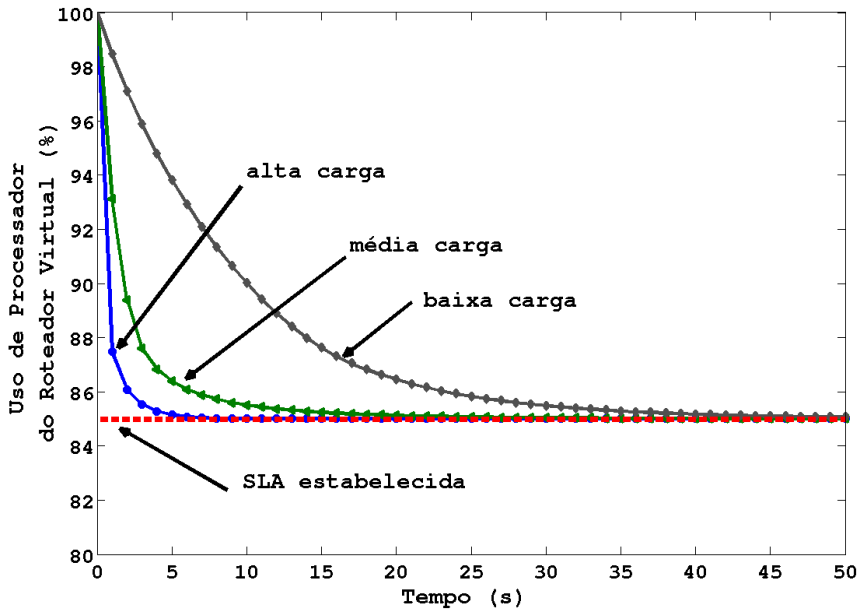


Figura 8. Estabilidade do Sistema proposto sob diferentes cargas do sistema. Dependendo da carga do sistema, o controle de punição pode ser mais rigoroso ou menos rigoroso.

Para este cenário desenvolvido, definiu-se três ambientes de fundo. No primeiro deles, só existe o próprio roteador monitorado e outro roteador virtual que quase não usa recursos, e portanto, o sistema mantém uma carga baixa. No segundo ambiente, existe o roteador que está sendo monitorado e mais um conjunto de cinco roteadores virtuais, que estão realizando um consumo moderado de recursos. Neste caso a carga foi estabelecida como média. No terceiro caso, existem sete roteadores virtuais além do roteador monitorado, e todos estes sete estão utilizando os recursos de forma próxima dos seus SLAs. A carga do sistema nesta configuração é alta. Os resultados verificados na Fig. 8 demonstram que o sistema converge para garantir a SLA estabelecida. Dependendo do nível de carga do sistema, para cada um dos ambientes definidos, o grau de punição é variável. Percebe-se que no ambiente de carga baixa, a punição é baixa e o sistema demora cerca de 40 segundos até que, de fato, o SLA passe a ser respeitado. Como a quantidade de recursos de processamento ociosos é grande, esta violação não prejudicaria outros roteadores. Ao utilizar o sistema em um ambiente de carga média, percebe-se que a punição ocorre de forma mais intensa e, em menos de 15 segundos, o roteador virtual violador tem o uso excessivo de recursos contido. Por fim, o ambiente de carga alta demonstra que o mecanismo de punição atuou de forma severa e agressiva e limitou o uso de recursos em menos de cinco segundos. Percebe-se, portanto, que a proposta atende aos requisitos estabelecidos, atuando de forma conservadora em situações de abundância de recursos ociosos e de forma agressiva em situações críticas.

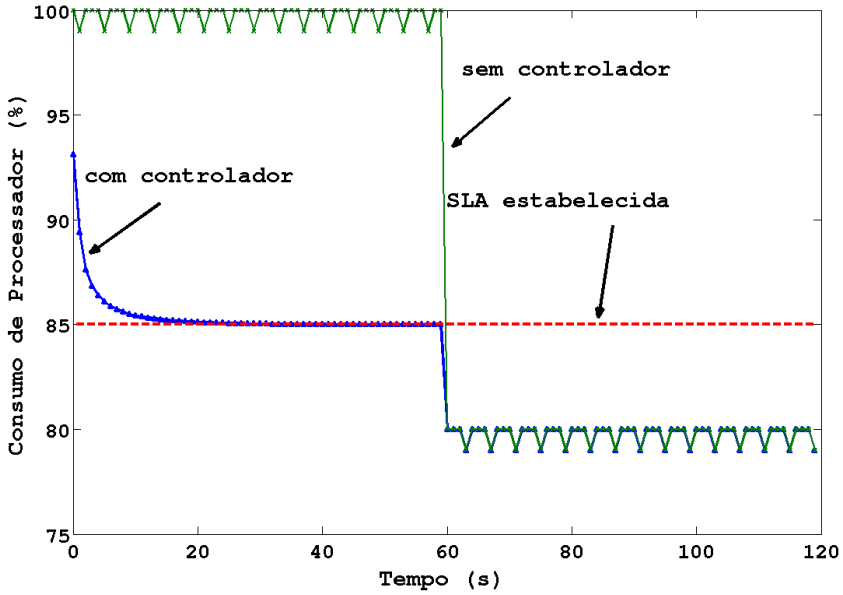


Figura 9. Uso de processamento de um roteador virtual que viola a SLA por um período de tempo, com média carga do sistema.

O terceiro resultado avalia o funcionamento do controlador quando um dado roteador virtual extrapola as SLAs contratadas por um dado período, e após este período, passa a respeitar os contratos. Neste experimento, o roteador virtual encaminha uma quantidade grande de tráfego, o que ocupa 100% do seu processador. Após 100 segundos, o roteador passa a encaminhar uma quantidade menor de tráfego, que consome em média 80% do processador, respeitando a SLA. Neste resultado, a carga do sistema manteve-se média. Os resultados são apresentados na Fig. 9, na qual observa-se que sem o controlador, o roteador virtual consegue consumir o que ele deseja, podendo assim prejudicar o funcionamento dos demais roteadores. Ao utilizar o controlador proposto, o roteador virtual tem o seu valor de *caps* limitado gradualmente até que a máquina passe a respeitar o contrato estabelecido.

4. Conclusão

Neste trabalho, desenvolveu-se um protótipo de controle nebuloso para controlar níveis de serviço em ambientes de redes virtualizados, nos quais a falta de isolamento representa um grande desafio de gerência. O sistema desenvolvido funciona de forma eficiente e é compatível com outras soluções de controle de recursos. Os gerentes de rede podem inserir regras que refletem experiências particulares na tomada de decisão em redes. Esta inserção pode ser feita através de pacotes de estratégias que podem ser facilmente gerados. Os resultados obtidos demonstram que o sistema consegue controlar de forma eficiente os SLAs estabelecidos, punindo os roteadores que violam as regras de acordo com a carga do sistema e do nível de violação. Nas configurações do experimento realizado, o sistema consegue limitar de forma adaptativa o SLA estabelecido para um dado roteador. Em momentos em que existem recursos ociosos o sistema aplica punições leves e, em momentos críticos o sistema aplica punições severas. Na condição severa testada, o controlador nebuloso conseguiu adequar o uso de recursos do roteador virtual em

menos de cinco segundos. Em um momento de carga baixa, o sistema efetuou o mesmo controle de forma mais branda, convergindo em 40 segundos para o nível de SLA acordado. Além disso, o monitoramento e gerência de múltiplos roteadores e recursos gera uma pequena sobrecarga de processamento no domínio de controle (aproximadamente 5% de um processador para cada novo roteador gerenciado). Futuramente, o sistema integrará o mecanismo de migração sem perda de pacotes e agregará algoritmos de decisão relacionados a migração de roteadores virtuais para garantir uma alocação de recursos mais eficiente.

5. Referências

[Barham et al., 2003] Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I. e Warfield, A. (2003). Xen and the art of virtualization. Em *Proceedings of the nineteenth ACM symposium on Operating systems principles*, páginas 164–177. ACM.

[Fernandes et al., 2010] Fernandes, N., Moreira, M., Moraes, I., Ferraz, L., Couto, R., Carvalho, H., Campista, M., Costa, L. e Duarte, O. (2010). Virtual networks: Isolation, performance, and trends. *Annals of Telecommunications*, páginas 1–17.

[Fernandes e Duarte, 2010] Fernandes, N. C. e Duarte, O. C. M. B. (2010). Xnetmon: Uma arquitetura com segurança para redes virtuais. Em *Anais do X Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais - SBSeg'10*, páginas 339–352, Fortaleza, CE, Brazil.

[Kecman, 2001] Kecman, V. (2001). *Learning and soft computing: support vector machines, neural networks, and fuzzy logic models*. The MIT press.

[Menasce e Bennani, 2006] Menasce, D. e Bennani, M. (2006). Autonomic virtualized environments. Em *Autonomic and Autonomous Systems, 2006. ICAS'06. 2006 International Conference on*, página 28. IEEE.

[Meng et al., 2010] Meng, X., Pappas, V. e Zhang, L. (2010). Improving the scalability of data center networks with traffic-aware virtual machine placement. Em *INFOCOM, 2010 Proceedings IEEE*, páginas 1–9. IEEE.

[Pisa et al., 2010] Pisa, P., Fernandes, N., Carvalho, H., Moreira, M., Campista, M., Costa, L. e Duarte, O. (2010). OpenFlow and Xen-Based Virtual Network Migration. *The World Computer Congress 2010 - Network of the Future Conference*, páginas 170–181.

[Wang et al., 2007] Wang, Y., van der Merwe, J. e Rexford, J. (2007). VROOM: Virtual routers on the move. Em *Proc. ACM SIGCOMM Workshop on Hot Topics in Networking*. Citeseer.

[Wood et al., 2007] Wood, T., Shenoy, P., Venkataramani, A. e Yousif, M. (2007). Black-box and gray-box strategies for virtual machine migration. Em *Proc. Networked Systems Design and Implementation*.

[Xu et al., 2008] Xu, J., Zhao, M., Fortes, J., Carpenter, R. e Yousif, M. (2008). Autonomic resource management in virtualized data centers using fuzzy logic-based approaches. *Cluster Computing*, 11(3):213–227.

[Zadeh, 1965] Zadeh, L. (1965). Fuzzy sets*. *Information and control*, 8(3):338–353.