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 (GTA/PEE/COPPE)
Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro – RJ – Brasil

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

Resumo. *A ideia chave do paradigma de redes virtuais pluralistas é permitir que múltiplas pilhas de protocolos executem em paralelo sobre o mesmo roteador físico para satisfazer os diferentes requisitos das aplicações. Este trabalho propõe um sistema eficiente de controle de contratos de níveis de serviço (Service Level Agreements - SLAs) em ambientes de rede virtualizados. O sistema proposto monitora o uso de recursos do roteador físico, fornece 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 excedem os SLAs estabelecidos. A punição depende do grau da violação e da carga total do sistema. Os resultados obtidos com o protótipo implementado comprovam a eficiência do controlador proposto sob diferentes condições e políticas de estratégia.*

Abstract. *The key idea of the pluralist virtual network paradigm is to execute multiple protocol stacks in parallel over a single physical router to satisfy the different requirements of applications. This work proposes an efficient control system for Service Level Agreements (SLAs) in the virtual network environments. The proposed system verifies the physical resource usage, 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 established SLA. The punishment depends on the exceeding value and on the system charge. Results obtained from a developed prototype show the efficiency of the proposed system under different conditions and strategic policies.*

1. Introdução

A técnica de virtualização completa de recursos de um computador permite a execução de múltiplos sistemas operacionais sobre um mesmo *hardware* físico. Esta funcionalidade é exercida através de um monitor, que se responsabiliza por multiplexar

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

o acesso ao *hardware* e prover fatias lógicas de recursos para os sistemas virtualizados. A virtualização, que é baseada na premissa de que os ambientes virtuais são isolados entre si, oferece uma série de vantagens como 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. 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 para a Internet do futuro [Fernandes et al., 2010].

As plataformas de virtualização tais como o Xen, em geral, não apresentam nativamente um suporte para a divisão de recursos nos ambientes de redes virtuais. Assim, não existe uma forma direta de mapear os perfis contratados por cada rede em perfis de uso de recursos físicos por cada máquina virtual. Com isso, o gerenciamento de redes virtuais carece de duas primitivas básicas: a obtenção dos perfis de uso dos recursos físicos por cada rede virtual e a garantia de que, dado um perfil de uso contratado, a rede virtual não excederá o uso dos recursos.

Este artigo propõe um controlador dinâmico de recursos baseado em lógica nebulosa e 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. A partir destes dados gerados, é feita uma detecção de violações de contratos de níveis de serviço. As redes que não estiverem respeitando os contratos são, então, punidas em tempo real e de forma autonômica. Além disso, o sistema de controle proposto monitora os valores de carga de cada rede virtual e gera estimativas reais de perfis de uso de cada um dos roteadores. Estes perfis podem ser usados para garantir uma melhor distribuição dos roteadores sobre os roteadores físicos, de forma a diminuir a chance de possíveis sobrecargas dos roteadores físicos. O sistema de controle 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 que atendem a diferentes estratégias. Além disso, o administrador de rede pode facilmente inserir novas regras e estratégias de atuação.

Uma das vantagens do controlador proposto é que ele baseia o cálculo da carga do sistema não apenas com base nos parâmetros de uso de CPU, memória e rede como outros sistemas da literatura [Wood et al., 2007], mas leva em consideração também outros parâmetros como a a robustez do sistema a falhas e a temperatura de operação. Estes parâmetros são importantes em cenários onde é preciso mapear quais são as máquinas físicas menos propensas a falhas e sobrecargas para evitar que máquinas virtuais de operação crítica sejam migradas para roteadores físicos instáveis. O sistema proposto foi implementado e testado. Os resultados obtidos mostram que a proposta pune adequadamente as violações dos roteadores e aplica punições adaptativas, que dependem da disponibilidade de recursos dos sistemas físicos. No ambiente de teste proposto, o sistema adequa os roteadores violadores aos seus contratos em menos de cinco segundos em cenários de escassez de recursos e permite que estes roteadores consigam um consumo adicional de recursos em cenários com abundância de recursos.

O restante do artigo é estruturado da seguinte forma. A Seção 2 descreve alguns trabalhos relacionados e a Seção 3 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 4 apresenta os resultados obtidos com o sistema, que demonstram o funcionamento do controlador nebuloso sob diferentes variáveis de ambiente. A Seção 5 apresenta as conclusões e direções futuras deste trabalho.

2. Trabalhos Relacionados

A maioria dos trabalhos de alocação dinâmica de recursos físicos em sistemas virtualizados concentra-se na aplicação de consolidação de servidores de *data centers* [Wood et al., 2007, Meng et al., 2010, Xu et al., 2008]. O *Sandpiper* [Wood et al., 2007] é um sistema que monitora máquinas virtuais em um *data center* e migra de forma automática 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 excessivo 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

$$Vol = \frac{1}{1 - cpu} . \frac{1}{1 - mem} . \frac{1}{1 - net}, \quad (1)$$

onde cpu é o percentual de uso de processamento, mem é o uso de memória e net o uso de rede. Estes três recursos são bons estimadores do volume, pois são os principais recursos que são compartilhados em ambientes virtualizados. Embora essa seja uma forma interessante de avaliar a sobrecarga do sistema, ela não considera todos os parâmetros que são importantes, além de não permitir que o administrador dê mais valor a um ou a outro parâmetro. Esse sistema também não tem objetivos de medir SLAs e garantir o seu cumprimento.

Xu *et al.* propõem mecanismos de controle em dois níveis para *data centers*, sendo um global, para remapear os recursos virtuais sobre os recursos físicos, e um local [Xu et al., 2008]. O controlador local tem como objetivos mapear a relação entre carga de trabalho e uso de recursos e predizer a carga de trabalho futura de cada rede virtual. Ambas as funções do controlador local são baseadas em lógica nebulosa. Para reduzir o número de regras geradas para a lógica nebulosa, é utilizado um mecanismo de clusterização para definir um conjunto de regras de decisão que represente a demanda de uso de uma dada aplicação, por exemplo, um serviço web, que executa dentro de um ambiente virtualizado. Os autores não apresentam nenhuma política para impedir o uso de recursos por uma rede, de tal forma que se a demanda da rede aumentar, os recursos devem ser disponibilizados de acordo com um nível de QoS contratado e o operador dos recursos virtualizados deve pagar pelo aumento do uso de recursos.

Existem propostas para a alocação inicial de recursos virtuais observando uma estimativa estática dos recursos com base em dados pré-determinados de carga do sistema. 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 virtuais que trocam muitas mensagens entre si, otimizando assim a escalabilidade da rede

e o aproveitamento da banda passante disponível nos enlaces de comunicação. Tal técnica poderia ser estendida com base no sistema proposto de cálculo de carga do sistema para permitir uma alocação dinâmica dos recursos, com base nos gastos reais de cada máquina, ao invés de se utilizar dados estáticos estimados.

O monitoramento e a alocação dinâmica de recursos físicos localmente em cada nó da rede também é uma abordagem muito utilizada. Menascé *et al.* aplicam técnicas de computação autonômica para controlar o compartilhamento de processadores entre máquinas virtuais em um *data center* [Menasce e Bennani, 2006]. Os autores utilizam técnicas de otimização para maximizar uma função de utilização global do nó físico e aplicam esses dados a um controlador do tipo caixa branca. O sistema é analisado por simulação e os resultados mostram que é possível dar prioridade a uma carga de trabalho específica.

Keller *et al.* realizam um estudo sobre os requisitos de QoS em ambientes virtualizados e propõem modelos de auditoria para contabilizar e garantir contratos de níveis de serviço localmente em roteadores virtuais [Keller et al., 2009]. Os autores sugerem dois modelos possíveis para garantir a auditoria da QoS dos serviços existentes: um modelo baseado no monitoramento de parâmetros de rede para verificação dos níveis de serviço e outro baseado na arquitetura confiável de computadores, que utiliza as plataformas seguras e mecanismos de chaves criptográficas para garantir a inviolabilidade do *firmware* das placas de rede e dos *containers* virtualizados que estão sujeitos a auditorias.

Outros sistemas propõem algoritmos de controle específicos para as plataformas de virtualização de rede, como o Xen [Barham et al., 2003]. 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 pouco eficiente, podendo causar problemas de isolamento. Além disso, essa plataforma não apresenta mecanismos de controle de recursos entre redes virtuais para prover QoS. Fernandes e Duarte propõem o *Xen Network Monitor*, que implementa um sistema seguro de controle para restringir o uso de recursos de entrada e saída de cada um dos roteadores virtuais [Fernandes e Duarte, 2010]. Esse sistema controla especificamente o uso dos recursos compartilhados em um domínio privilegiado que é utilizado por todos roteadores virtuais nas operações de rede, com base nas SLAs especificadas.

O sistema proposto é um controlador local para ambientes de rede virtuais que independe de plataforma de virtualização. Embora o mecanismo proposto possa ser usado em *data centers*, a lógica desenvolvida nesse artigo visa controlar os desafios no cenário de roteadores virtuais. Diferentemente do SandPiper e da proposta de Xu *et al.*, que visam garantir que o provedor de infraestrutura não proveja um volume de recursos inferior ao SLA para a aplicação virtualizada, a proposta apresentada nesse artigo tem por objetivo auxiliar o provedor de infraestrutura a impedir que o operador de rede virtual exceda o uso de recursos que foi contratado nas SLAs. Assim, o sistema proposto monitora os recursos físicos utilizados por cada rede virtual, utiliza esses dados para alimentar um preditor e, com base nas SLAs contratadas e nos recursos estimados, pune as redes virtuais que violam as SLAs. Esse é um cenário diferente dos *data centers*, pois ao invés de ter como principal recurso controlado o uso de processamento, nesse caso é preciso controlar o uso de recursos causado pelas operações de encaminhamento de pacote, que dependem diretamente de uma variável não controlável que é o tráfego de entrada.

3. O Sistema de Controle Proposto

O sistema de controle proposto visa monitorar e atender acordos de níveis de serviço (SLA) de redes virtualizadas. Os SLAs são definidos como contratos estabelecidos entre os provedores de serviço e os clientes, que regulam diversos parâmetros como banda passante, atrasos mínimos, quantidade de recursos consumidos, garantias e multas. A ideia-chave é baseada na geração e análise de perfis de uso de cada um dos roteadores virtuais e na punição, utilizando lógica nebulosa em tempo real, 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, também chamado de carga do sistema, e da gravidade da violação. O controlador nebuloso pondera os parâmetros relacionados a processamento, memória, rede, temperatura do processador e robustez, que é a existência de mecanismos de redundância e tolerância a falhas, no cálculo da carga do sistema. A carga do sistema é utilizada ativamente no grau de punição aplicada aos roteadores que violam os acordos de nível de serviço e também pode ser usada para estimar possíveis migrações devido a sobrecargas do roteador físico.

O sistema proposto 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 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 que depende do estado em que se encontra a carga do sistema. Por exemplo, se o sistema está com uma baixa carga, uma violação de ordem média (por exemplo, ultrapassar em 20% o SLA estabelecido) 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. são valores obtidos pelo sistema, que são gerados após a aplicação de funções de pertinência aplicadas as entradas dos controladores.

3.1. Arquitetura do Sistema

O sistema proposto segue um modelo de gerência distribuído composto por agentes controladores dispersos pela rede física. Cada agente controlador está associado e controla um ou mais conjuntos de roteadores físicos, os 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, garantir 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. Assim, o MEP atualiza a estratégia vigente nos *daemons* de cada domínio de controle. O módulo de níveis de serviço (MNS) mantém uma

base de dados com os SLAs 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 renegociar contratos, adaptando os contratos à realidade de cada roteador virtual. O módulo atuador age nos *daemons* dos roteadores físicos e obtém os perfis reais de uso e estatísticas de cada roteador. O atuador é responsável, entre outros, por calcular e aplicar a punição nas redes virtuais que violam os SLAs contratados. Os agentes controladores também possuem o módulo de comunicação (Comm), que permite a troca de informações de controle com outros agentes, 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.

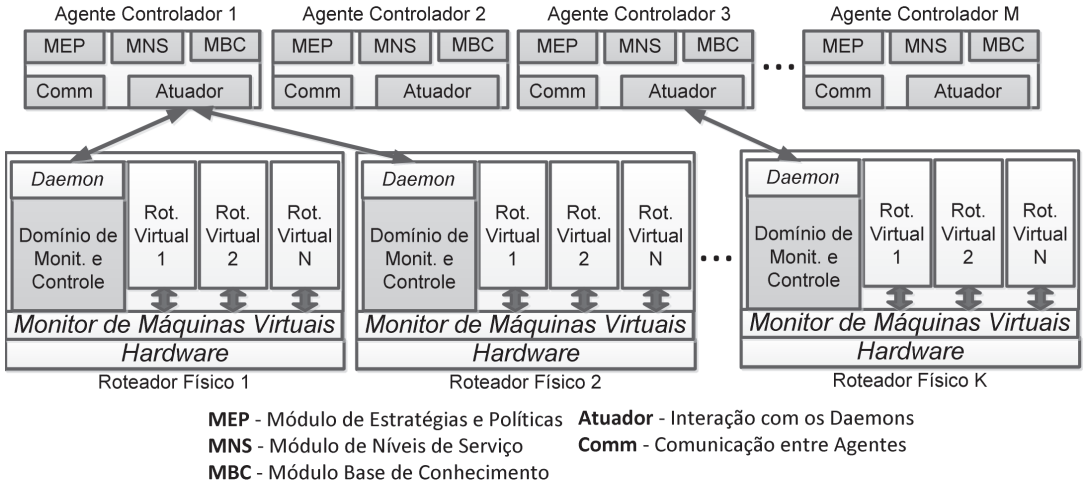


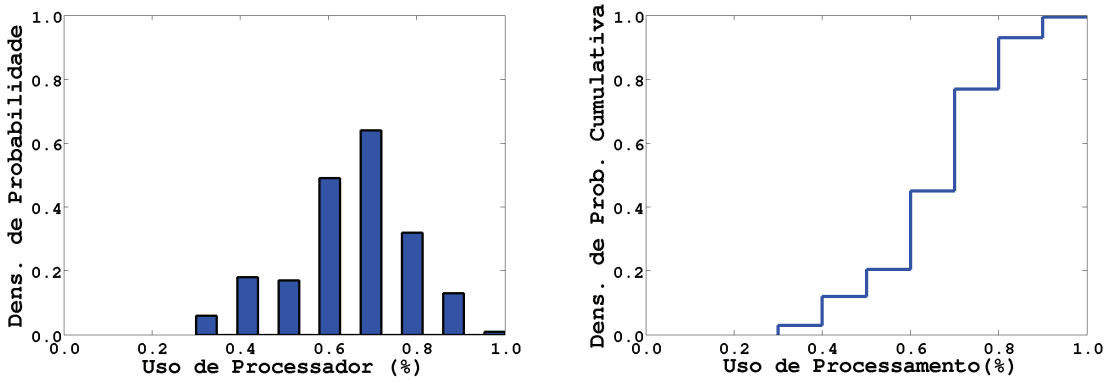
Figura 1. Arquitetura do sistema de controle, na qual os agentes controladores distribuídos interagem com os roteadores físicos para controlar o uso de recursos físicos.

Os roteadores físicos também participam do sistema de controle através do *daemon* que é executado no domínio de monitoramento e controle. Este *daemon*, além de receber os dados enviados pelo agente controlador, tais como a punição a ser aplicada em cada uma das redes virtuais, também realiza outras duas funções fundamentais do sistema. Primeiramente, o *daemon* calcula e atualiza as estatísticas de consumo de recurso para criar os perfis de cada roteador virtual. Além disso, o *daemon* também estima a carga do sistema com base nos dados que foram monitorados para cada um dos roteadores virtuais.

3.2. 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, que podem ser utilizados tanto para detectar violações de regras quanto para estimar o consumo e prever necessidades futuras de cada roteador. 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], através da captura de séries temporais que refletem o

consumo de recursos de memória, processador e rede. As funções geradas servem para detectar a mudança repentina de um perfil e estimar possíveis migrações. Neste artigo, 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. O passado recente reflete um estado mais próximo do estado atual e permite a detecção rápida em caso de violações de acordos. O passado longo serve como um histórico do comportamento da máquina e permite a detecção de um padrão de consumo de longo prazo da máquina. Para auxiliar na detecção destes padrões, nesse artigo propõe-se a utilização da função distribuição de probabilidades (PDF) e das funções cumulativas de distribuição de probabilidades (CDF). A PDF demonstra a probabilidade de um dado evento acontecer, dentro de um conjunto de medidas em uma janela longa. Assim, a PDF pode indicar, entre outros, a probabilidade de um determinado recurso ter sido consumido um certo número de vezes em um intervalo de tempo, por exemplo o consumo de processador em uma dada janela de amostragem. As CDFs podem ser utilizadas para verificar níveis de serviço em janelas de observação curtas. As CDFs refletem o quanto de um dado recurso foi utilizado em relação ao total de tempo decorrido e por essa razão são bons indicadores para a definição dos SLAs e para a verificação de possíveis violações de contrato. Através desta perspectiva, pode-se pensar em SLAs flexíveis.



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

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

Para exemplificar o uso dos perfis com base nas duas distribuições de probabilidade, desenvolveu-se um protótipo para geração de perfis e realizou-se um teste, cujos resultados estão na Fig. 5[1].

Uma função distribuição de probabilidades (PDF) do consumo de processamento 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

[1] Neste cenário de testes, existem quatro roteadores físicos que executam o Xen e dentro de cada um destes existe um roteador virtual. Foram criados enlaces lógicos entre estes roteadores (nesta configuração, quatro roteadores virtuais) que executam o protocolo de roteamento RIPv2 através do XORP [Handley et al., 2003] e realizam a troca de mensagens de controle. Foi considerada uma janela longa de observação de 200 medidas para gerar a PDF e uma janela curta de observação de 20 medidas para gerar a CDF. Escolheu-se um intervalo de 1 s entre medidas

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 . Devido a esta propriedade observada, pode-se concluir que, nesse cenário, 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.

A CDF gerada para o mesmo cenário está na Fig. 2(b). Por essa, é possível observar que o roteador virtual utiliza, em média, apenas 0,7% dos recursos de processamento em 80% do tempo da janela longa.

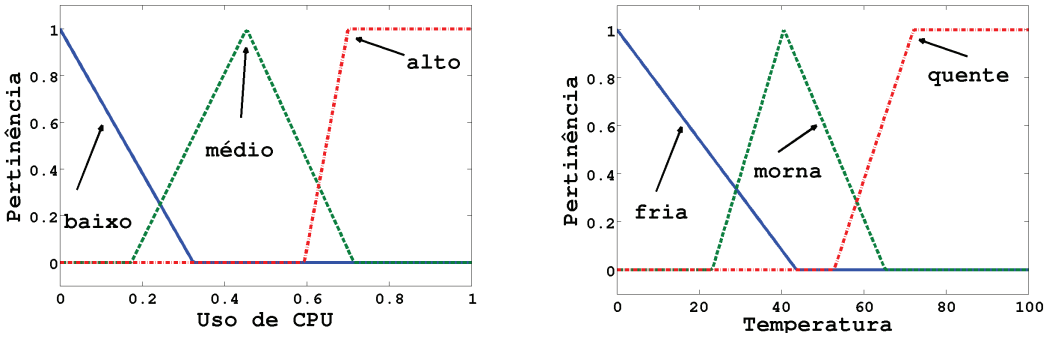
3.3. Estimando a Carga do Sistema nos *Daemons* de Controle

A carga do sistema é uma medida que indica o nível de carga do *hardware* gerenciado. Graças a ela é possível detectar a saturação de recursos. O cálculo da carga no sistema proposto 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. O monitoramento é feito através de medidas fornecidas pelo próprio sistema operacional do Domínio-0. Os dados monitorados são, então, processados por um controlador nebulosos para determinar a carga do sistema.

Foi adotado o uso de controladores nebulosos pela sua adequação 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. Assim, foi definido o conjunto de funções de pertinência μ_{Proc}, μ_{Mem}, μ_{Net}, μ_{Temp} e μ_{Rob}, correspondentes ao processador, memória, rede, temperatura do processador e robustez do sistema respectivamente, que associa cada um dos recursos ou parâmetros em variáveis *fuzzificadas*. A Fig. 3 mostra um exemplos de funções de pertinência para uso de CPU e para temperatura do processador. 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.

A combinação os graus de pertinência 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 pode ser utilizado para estimar futuras migrações. Além disso, a carga do sistema é utilizada como parâmetro de entrada de outro controlador (o controlador de punição). O controlador de punição também recebe como entrada a variável δ, que representa a diferença entre os recursos acordados e os recursos consumidos por um roteador. A variável δ é utilizada para estimar o grau de punição dos roteadores que violam os SLAs. Um diagrama em blocos do módulo de estratégias e políticas (MEP), que engloba os dois controladores, é mostrado na Fig. 4[2].

[2] Para ambos os controladores, foi adotado o método de implicação Mamdani, com as operações de AND e OR de Zadeh [Zadeh, 1965] e o método centroide de *defuzzificação*.



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

(b) Funções de pertinência para a temperatura.

Figura 3. Funções de pertinência para avaliar CPU e Temperatura.

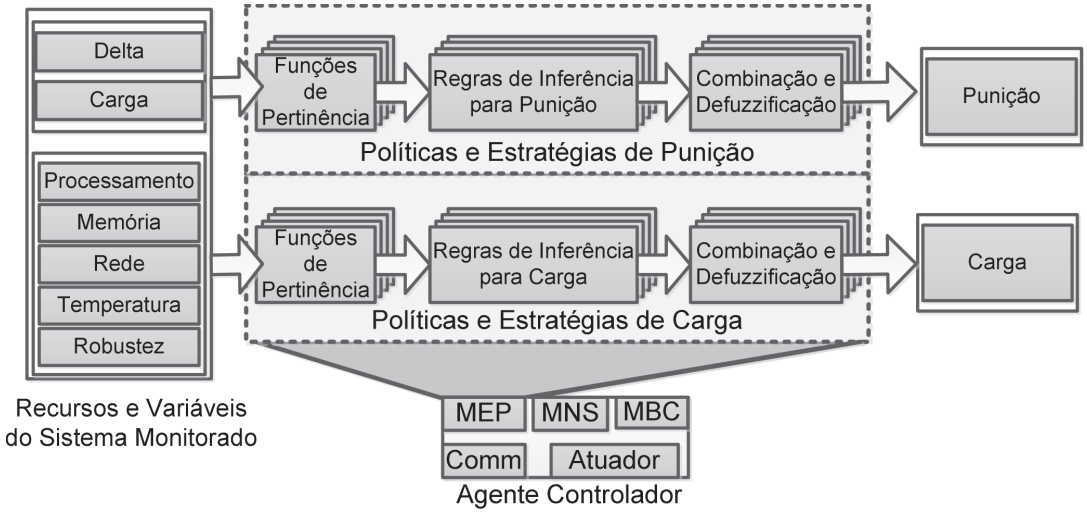


Figura 4. Módulo de Estratégias e Políticas

3.3.1. Estratégias Baseadas em Regras de Inferência

As estratégias dos controladores nebulosos 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 estratégias. Um exemplo de pacote de estratégias que calcula o grau de punição de acordo com a diferença entre o SLA acordado e o consumo real, chamada de δ, e a carga do sistema pode ser visto na Tabela. 1.

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*

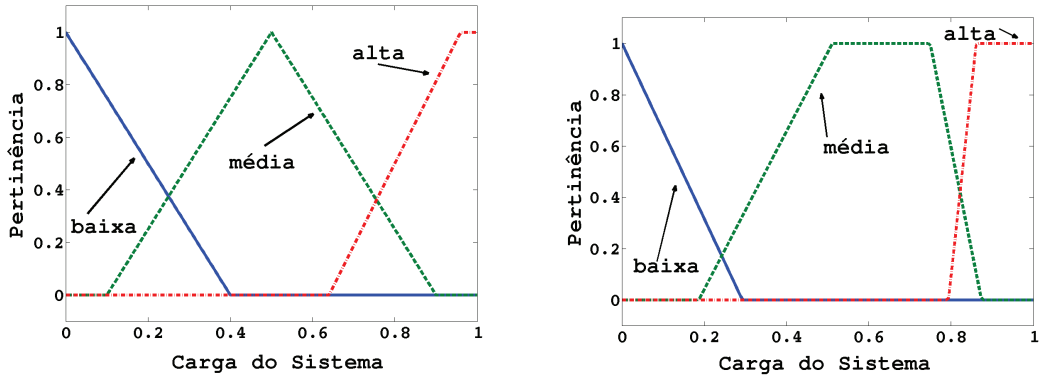
Tabela 1. Exemplo de um pedaço de um pacote de estratégias.

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)

que deve adotar a estratégia. Pode-se estabelecer diferentes estratégias para cada um dos recursos controlados, aumentando assim a flexibilidade do controlador.

3.3.2. Políticas de Carga

Após a aplicação das regras de inferência, são gerados valores nebulosos 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 contínuo entre $[0, 1]$ que representa a carga do sistema. Na Fig. 5, pode-se observar duas políticas de carga possíveis, uma conservadora e uma agressiva.

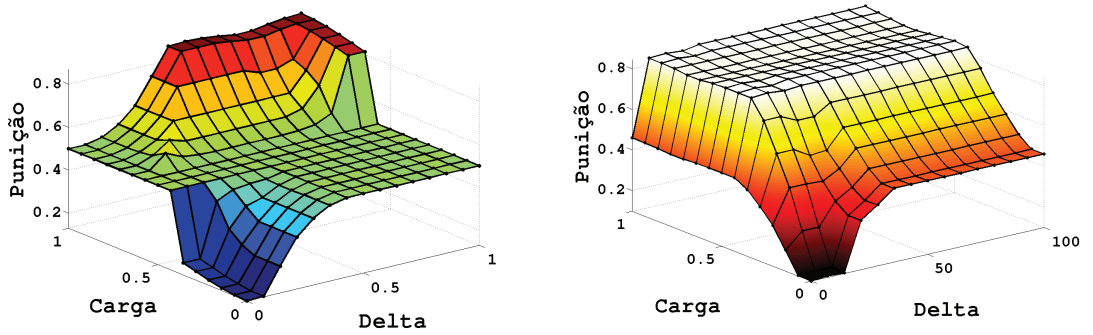


(a) Política conservadora. (b) Política agressiva.

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 δ, que corresponde ao grau de violação do SLA . Esta relação pode ser analisada na Fig. 6. Nestas superfícies, pode-se perceber que a política agressiva ou conservadora é refletida no grau de variação da punição.

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)). Percebe-se que variações no delta (δ) se refletem em punições altas somente quando o valor de carga também é alto. Além disso, neste caso particular, forma-se um patamar próximo da punição média de 0.5. Na política agressiva, pequenos aumentos na sobrecarga e no delta geram grandes punições (Fig. 6(b)). Pode-se perceber que se o sistema tiver muitos recursos ociosos (carga próxima de zero), o delta pode variar livremente sem que o resultado



(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.

final seja uma punição muito severa. Porém, ao fixar o delta em um valor baixo, variações na carga aumentam rapidamente o grau de punição, levando o sistema a aplicar punições graves. A aplicação de uma superfície conservadora permitiria um maior aproveitamento dos recursos ociosos e permitiria que roteadores que violam os acordos consigam utilizar mais recursos por mais tempo, enquanto que uma superfície agressiva manteria uma quantidade significativa de recursos ociosos e permitiria que as violações ocorressem por um período menor de tempo.

3.3.3. Controle da Sobrecarga do Sistema dos Níveis de Contrato

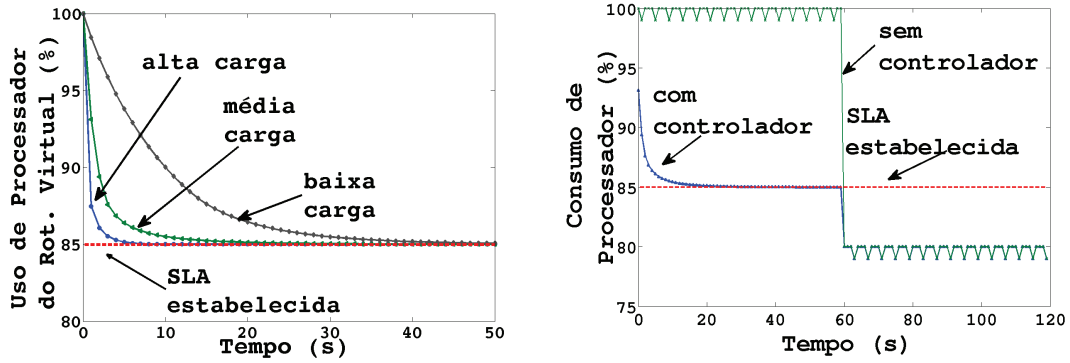
O sistema desenvolvido é capaz de gerar perfis de uso, avaliar se os perfis correspondem às SLAs negociadas, gerar estimativas de sobrecarga do sistema e punir de forma adaptativa os roteadores virtuais que desrespeitarem as regras propostas. Na implementação, o *daemon* que executa em cada domínio de controle realiza a coleta dos parâmetros dos recursos a cada intervalo de tempo, que pode ser definido pelo gerente de rede. 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 do cumprimento das SLAs. Estas informações são enviadas ao agente controlador. O *daemon* do agente controlador 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. Com base nesses dados, calcula-se o δ. O sistema então utiliza este δ 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.

4. Resultados

Para validar o funcionamento do sistema, foram desenvolvidos alguns experimentos com o protótipo desenvolvido que tiveram o objetivo de comprovar a baixa sobrecarga de gerência do sistema e o funcionamento eficiente do controle nebuloso adaptativo de SLAs. Para o ambiente do teste, foi selecionada a plataforma de virtualização do Xen para criar os roteadores virtuais. O parâmetro de controle utilizado para a aplicação das punições é o *caps*. O *caps* regula o limiar superior 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 suportada é o *Traffic Control* (TC) [Almesberger et al., 1999], que permite o controle de filas, gerenciando assim a vazão de cada um dos roteadores virtuais, caso estes ultrapassem as SLAs definidas. Os testes foram realizados em uma máquina física com um processador core i7 860 com 4 núcleos reais e 8 GB de memória RAM DDR3. Esta máquina foi configurada com o *hypervisor* Xen 4.0. Os roteadores virtuais foram configurados com 128 MB 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.

O primeiro experimento avalia o funcionamento do controlador proposto e do mecanismo de punição nebuloso. Para realizar este teste, um dos roteadores virtuais foi selecionado. O acordo de nível de serviço deste roteador define, dentre outras regras, que o roteador pode utilizar até 85% de processamento da máquina virtual para efetuar o encaminhamento de pacotes e a troca de mensagens de controle. 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 o seu SLA e o sistema de controle proposto regula através da primitiva do *caps* a quantidade de processamento que pode ser utilizada pelo processador. Para este experimento, o intervalo entre as verificações de SLAs foi definido como um segundo. O sistema de punição foi habilitado quando o roteador já estava consumindo uma quantidade de processamento que extrapolava o seu limite, em 60 s.



(a) Estabilidade do sistema proposto sob diferentes cargas do sistema.

(b) Uso de processamento de um roteador virtual que viola o SLA até 60 s..

Figura 7. Eficiência do controlador proposto em diferentes cenários.

Para estes testes, definiram-se três cenários. No primeiro deles, só existe o próprio roteador monitorado e outro roteador virtual que apenas troca de mensagens de controle no protocolo RIPv2 com três outros roteadores. Portanto, o sistema mantém uma baixa carga. No segundo cenário, existe o roteador que está sendo monitorado e mais um conjunto de cinco roteadores virtuais, que estão realizando um consumo moderado de recursos, que gerou um valor de carga de aproximadamente 0,5. 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, com um valor de aproximadamente 0,6. Para cada uma das situações foi executada apenas uma rodada de testes. Os resultados verificados na Fig. 7(a) demonstram que o sistema converge para garantir o SLA

estabelecido. Nestes testes, utilizou-se a política conservadora. 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 s 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 s, 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 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. Caso ocorressem mudanças significativas no valor de carga do sistema, as curvas observadas sofreriam mudanças de inclinação, que refletiriam a variação da carga durante o período de teste.

O segundo teste avalia o funcionamento do controlador quando um dado roteador virtual extrapola os SLAs contratados 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 60 segundos, o roteador passa a encaminhar uma quantidade menor de tráfego, que consome em média 80% do processador, respeitando o SLA. Neste resultado, a carga do sistema manteve-se média. Os resultados são apresentados na Fig. 7(b), na qual observa-se que sem o controlador, o roteador virtual consegue consumir todos os recursos, podendo, assim, prejudicar o funcionamento dos demais roteadores. Pode se perceber que neste resultado, diferente dos resultados anteriores, a medida de processamento apresenta algumas variações. Isto acontece devido a pequenas oscilações nas medidas realizadas pelas ferramentas de captura e oscilações inerentes ao consumo de processador variável de cada máquina. Quando o parâmetro *caps* limita que um determinado roteador utilize menos do que ele tenta utilizar, este corte ocorre pelo valor do *caps* e não ocorrem variações. Quando o roteador virtual utiliza um percentual de processamento menor que o *caps* no entanto, podem ocorrer estas variações. 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.

5. 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, de 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.

6. Referências

[Almesberger et al., 1999] Almesberger, W. et al. (1999). Linux network traffic control: implementation overview. Em *Sixth IEEE Symposium on*, volume 2001, páginas 296–301.

[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.

[Handley et al., 2003] Handley, M., Hodson, O. e Kohler, E. (2003). XORP: An open platform for network research. *ACM SIGCOMM Computer Communication Review*, 33(1):53–57.

[Keller et al., 2009] Keller, E., Lee, R. e Rexford, J. (2009). Accountability in hosted virtual networks. Em *Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures*, páginas 29–36. ACM.

[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.

[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.