Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Modelo Baseado em Agentes com Uso de Confiança e Reputação para Seleção de Supernós em Fog Computacional

Thiago Milo Simões Ferreira

Dissertação apresentada como requisito parcial
para conclusão do Mestrado em Informática

Orientadora
Prof.ª Dr.ª Célia Ghedini Ralha

Brasília
2018

Universidade de Brasília — UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Mestrado em Informática

Coordenador: Prof. Dr. Bruno Luiggi Macchiavello Espinoza

Banca examinadora composta por:

Prof.ª Dr.ª Célia Ghedini Ralha (Orientadora) — CIC/UnB
Prof.ª Dr.ª Alba Cristina Magalhães Alves de Melo — CIC/UnB
Dr. Bruno Werneck Pinto Hoelz — DPF

CIP — Catalogação Internacional na Publicação

Ferreira, Thiago Milo Simões.

Modelo Baseado em Agentes com Uso de Confiança e Reputação para Seleção de Supernós em Fog Computacional / Thiago Milo Simões Ferreira. Brasília : UnB, 2018.
99 p. : il. ; 29,5 cm.

Dissertação (Mestrado) — Universidade de Brasília, Brasília, 2018.

1. Fog Computacional, 2. Agentes inteligentes, 3. Sistemas multiagentes, 4. Modelo de Confiança e Reputação

CDU 004

Endereço: Universidade de Brasília
Campus Universitário Darcy Ribeiro — Asa Norte
CEP 70910-900
Brasília–DF — Brasil

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Modelo Baseado em Agentes com Uso de Confiança e Reputação para Seleção de Supernós em Fog Computacional

Thiago Milo Simões Ferreira

Dissertação apresentada como requisito parcial
para conclusão do Mestrado em Informática

Prof.ª Dr.ª Célia Ghedini Ralha (Orientadora)
CIC/UnB

Prof.ª Dr.ª Alba Cristina Magalhães Alves de Melo
CIC/UnB

Dr. Bruno Werneck Pinto Hoelz
DPF

Prof. Dr. Bruno Luiggi Macchiavello Espinoza
Coordenador do Mestrado em Informática

Brasília, 20 de julho de 2018

Dedicatória

Dedico este trabalho aos meus pais Rubem e Rosângela, meus grandes heróis.

Agradecimentos

À minha família pelo apoio durante esta caminhada e pela compreensão nos momentos em que eu não pude estar presente. Aos meus pais Rubem e Rosângela, por todos os ensinamentos e pelo apoio constante. Aos meus sogros, Ivon e Vera pelo incentivo e suporte que sempre me deram.
À Cecília, minha esposa, companheira de todas as horas, por todo o apoio e paciência, e por estar ao meu lado em todos os momentos.
À Thalita, minha irmã, pela amizade e companheirismo de sempre, e por todo o suporte durante os anos em Brasília.
À minha orientadora Célia Ghedini Ralha, por todo o apoio, atenção e incentivo durante a condução deste projeto.

Resumo

O modelo de fog computacional coloca recursos de processamento e armazenamento mais próximos aos dispositivos usuários, na fronteira da rede. Algumas abordagens podem ser aplicadas para a implantação dos nós servidores que fornecem esses recursos. Um caminho é utilizar equipamentos pré-selecionados que já estão instalados na rede, como *switches*, roteadores e pontos de acesso *wi-fi*. Outra opção é implantar dispositivos específicos que exerceriam esse papel, e uma terceira via seria a utilização de dispositivos dos próprios usuários como nós servidores. Qualquer que seja a opção o problema de como selecionar esses nós servidores se coloca como um ponto importante para a manutenção da qualidade do serviço provido na fog computacional. Nesse contexto, esta dissertação apresenta um modelo baseado em agentes para a seleção dos nós servidores na fog computacional. Esse cenário complexo de relações entre diversos atores é propício para a utilização de agentes inteligentes. Um modelo de confiança e reputação é utilizado pelos agentes para realizar a seleção dos nós visando a manutenção da qualidade de serviço. O modelo proposto foi testado em ambiente simulado utilizando o Peersim e posteriormente implementado com base no *framework* JADE de maneira a permitir a avaliação da proposta. Os resultados coletados foram analisados de modo a verificar a validade do modelo proposto em relação à manutenção da qualidade de serviço no âmbito do modelo de fog computacional.

Palavras-chave: Fog Computacional, Agentes inteligentes, Sistemas multiagentes, Modelo de Confiança e Reputação

Abstract

The fog computaing model puts processing and storage resources closer to the end-user devices at the network edge. Some approaches can be applied to deploy server nodes that will act as providers for these resources. One way is to use pre-selected devices that are already installed on the network, such as switches, routers and wi-fi access points. Another option is to deploy specific devices that would play this role, and a third way would be to use the users' own devices as server nodes. Whatever the option, the problem of selecting these server nodes is a key point for maintaining the quality of the service provided on the fog. In this context, this Msc dissertation presents an agent-based model for the selection of these nodes in computational fog. This complex scenario, with multiple relations among different participants, can benefit from the use of intelligent agents. The agents use a trust and reputation model to perform the selection of the nodes in order to maintain the quality of service. The proposed model was implemented on the JADE framework to allow evaluation of the proposal. The results were analyzed in order to verify the validity of the proposed model and its contribution to the maintenance of service quality within the fog model.

Keywords: Fog computing, Intelligent agents, Multiagent systems, Trust and reputation models

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Todos os dias milhares de usuários ao redor do mundo se conectam à rede mundial de computadores (Internet) em busca dos mais variados tipos de serviços. Aplicações bancárias, portais de notícias, plataformas de *streaming* e redes sociais são somente alguns exemplos entre a infinidade de serviços que podem ser acessados pelos usuários da Internet. Segundo a agência especializada da Organização das Nações Unidas (ONU) para tecnologia da informação e comunicações - *International Telecommunication Union* (ITU), ao final de 2015, 3.2 bilhões de pessoas tinham acesso à Internet (Sanou, 2015). Com a difusão do uso de aplicações de Internet das Coisas (*Internet of Things* - IoT) o número de dispositivos tende a aumentar em uma proporção ainda maior, pois, além dos dispositivos utilizados diretamente pelos usuários, os sistemas de IoT empregam sensores e outros equipamentos que trafegam dados e executam ações de forma autônoma. A Ericsson, prevê que em 2023, 30 bilhões de dispositivos estarão conectados à Internet (Ericsson, 2017). Esse vasto conjunto de aplicações e dispositivos necessitam de capacidade de processamento e armazenamento para que possam gerar o valor esperado pelos usuários.

Neste cenário crescente de uso de aplicações, recursos e serviços de Internet existe a necessidade de baixar os custos de manutenção, manter a escalabilidade, e aumentar a facilidade de gerenciamento, aderente ao modelo *pay-as-you-use* da computação em nuvem (*cloud*). Durante os últimos anos a nuvem foi utilizada como base para a criação e disponibilização dos mais diversos tipos de serviços *on-line*, atendendo a milhares de usuários diariamente.

No entanto, apesar de ampliar o acesso a recursos computacionais, o modelo de *cloud* tende a centralizar o processamento das aplicações em grandes *data centers*. Dessa forma, serviços de processamento e de armazenamento são concentrados em algumas localidades que muitas vezes estão à milhares de quilômetros de distâncias dos dispositivos usuários, o que pode acarretar em um aumento considerável na latência de comunicação (*latency*). Ademais, essa concentração somada à distância entre o dispositivo usuário e o servidor pode acarretar em um uso ineficiente da rede, já que os dados devem ser movimentados por longas distâncias, o que pode causar congestionamentos de tráfego e problemas de escalabilidade.

Algumas aplicações são mais sensíveis à latência e a variações no tempo de resposta. Em alguns ambientes de IoT, por exemplo, os sensores podem capturar uma imensa quantidade de dados que devem ser tratados e processados de forma a controlar a ação de dispositivos atuadores. Nesse cenário, o tempo entre envio e retorno das ações de controle

pode afetar dramaticamente o funcionamento do sistema, em alguns casos tornando inviável sua utilização (Dastjerdi and Buyya, 2016). Sistemas de *healthcare*, de controle de tráfego e de segurança são exemplos de aplicações que podem ter seu valor comprometido quando alonga-se o tempo entre a coleta dos dados pelos sensores e o envio dos comandos para os dispositivos atuadores. Como os *datacenters* de nuvem estão centralizados geograficamente e muitas vezes distantes dos sensores e dispositivos finais, eles podem falhar no fornecimento de processamento e armazenamento que atendam aos requisitos de latência de resposta das aplicações de IoT (Mahmud et al., 2018). Em outros casos a latência pode afetar a experiência dos usuários e deteriorar a qualidade de serviço (*Quality of Service* - QoS). Plataformas de *streaming* e jogos *online* são exemplos com amplo uso atualmente.

O paradigma de fog computacional se coloca como uma solução para alguns desses problemas. A fog estende os serviços da nuvem para a borda da rede, atuando como uma interface entre os dispositivos que estão na fronteira da rede e os servidores na nuvem. Dessa forma, os serviços de computação, armazenamento e rede podem ser levados para a vizinhança dos usuários finais (Bonomi et al., 2012). O cenário de *fog computing* pode trazer benefícios como a redução da latência de resposta para diversos sistemas, redução do tráfego de rede, aumento da cobertura de usuários e redução de custos relativos à consumo de banda. Utilizando um modelo de fog, uma plataforma de *streaming* poderia, por exemplo, implantar nós de armazenamento mais próximos de seus usuários, permitindo que esses carreguem seus vídeos mais rapidamente e ainda reduzindo o tráfego de rede por não exigir que todo o conteúdo trafegue de um servidor de nuvem (mais centralizado) para o dispositivo do usuário. Esses dispositivos que atuam como provedores de serviço na fog, seja para armazenamento ou processamento, são denominados de supernós. Os nós, por sua vez, representam os dispositivos consumidores de serviço na fog.

O modelo de fog computacional coloca recursos de processamento e armazenamento na fronteira da rede. Algumas abordagens podem ser aplicadas para a implantação de nós servidores, ou supernós, na fronteira da rede. Um caminho é utilizar equipamentos que cumprem funções de roteamento e controle de tráfego de rede e já estão presentes nas proximidades do usuários, como roteadores, *switches*, e pontos de acesso *wi-fi*. Outra opção seria instalar equipamentos específicos para essa tarefa, como computadores *single-board*, que ficam ligados à rede de dados e são colocados em regiões pré-definidas para atender um determinado grupo ou demanda específica. Uma terceira via, seria a utilização de dispositivos dos próprios usuários que poderiam disponibilizar parte de seus recursos para serem utilizados no âmbito da fog computacional visando alguma recompensa.

Independente da abordagem para a implantação dos nós na fog computacional, a questão relacionada à QoS se fará presente. Para os usuários é indiferente se um determinado serviço está sendo provido em um servidor remoto na nuvem ou em um supernó na fronteira da rede desde que a QoS esperada seja atendida. Segundo esta visão, os usuários consideram o atendimento aos requisitos pré-estabelecidos de qualidade.

Dessa forma, no contexto de fog computacional estão presentes diferentes atores que compartilham alguns objetivos comuns e outros que podem ser conflitantes. Por exemplo, o usuário e o provedor de serviço compartilham o objetivo de obter a QoS esperada. O provedor, porém, associa esse objetivo ao custo de fornecimento do serviço, de modo a balancear o custo com os requisitos esperados de QoS. O usuário, ao seu turno, verifica se a QoS requisitada está sendo cumprida e caso não esteja qual o impacto que esse descumprimento gera para o eventual custo que lhe é cobrado para ter acesso ao serviço.

Somam-se à esse contexto os usuários que disponibilizam seus dispositivos para serem utilizados como pontos de provisão de serviço na fog. Nesse caso a relação custo x qualidade se inverte em relação ao usuário e provedor, pois o primeiro visa aumentar sua eventual recompensa, enquanto o segundo visa diminuir seus custos com recompensas mantendo e/ou aumentando o nível de QoS.

Esse contexto complexo de relações entre diversos atores é propício para a utilização de agentes inteligentes. Os agentes podem tomar decisões de maneira autônomas de modo a refletir de maneira mais fidedigna a relação entre esses diferentes atores envolvidos na provisão de serviço em um contexto de fog computacional. A interação entre os agentes pode propiciar um meio para a condução de processos de seleção de nós de processamento na fog e para a negociação de parâmetros de fornecimento dos serviços. Ademais, os agentes podem tomar decisões autonomamente para se adaptar às mudanças que podem ocorrer no ambiente da fog computacional.

1.1 Problema

Conforme exposto, existe uma grande capilaridade no modelo de fog computacional que resulta em um aumento de complexidade nos processos relacionados à garantia da QoS. Essa situação pode ser ainda agravada quando dispositivos de usuários são utilizados como supernós na fog. Pois, além de apresentarem os problemas adversos comuns aos demais tipos de equipamentos, como falhas em algum componente, no caso dos dispositivos de usuários há ainda a possibilidade de uma ação deliberada do usuário que detêm a posse do dispositivo de forma a afetar o serviço. Ou seja, um usuário que aloca parte da capacidade de seu dispositivo para hospedar um serviço na fronteira da rede pode, deliberadamente, retirar parte dessa capacidade sem prévio aviso, o que afetaria os usuários que estão utilizando o serviço provido por seu dispositivo e causaria um impacto na QoS.

Neste cenário, o processo de seleção dos dispositivos que irão atuar como supernós deve ser realizado de maneira criteriosa. Vários aspectos podem ser levados em consideração neste processo, como capacidade disponível de memória, espaço em disco, largura de banda, estabilidade e *delay* de comunicação. Um outro fator mais subjetivo que deve ser incluído neste processo de seleção é a disposição do nó usuário em manter os recursos que foram previamente alocados para a fog computacional. Dessa forma, o resultado dessa processo deve dirimir o risco relativo à utilização de dispositivos de usuários como nós servidores na fog em um cenário onde a QoS deve ser mantida a níveis desejados.

1.2 Objetivo

Considerando o cenário de fog computacional, onde centenas de nós coexistem em ambientes distribuídos, onde a oferta de serviços para os usuários finais devem ter QoS adequado as necessidades individuais e coletivas, a escolha de nós e supernós é um problema central. Nesse contexto, este trabalho apresenta a seguinte questão de pesquisa:

Um modelo baseado em agentes que utiliza confiança e reputação pode auxiliar na seleção de nós para manter a QoS em uma ambiente de fog computacional?

Dessa forma, o objetivo geral deste trabalho é definir e validar um modelo baseado em agentes com uso de confiança e reputação para a seleção de nós em um ambiente de fog computacional.

Partindo do objetivo principal, ficou definida como contribuição apresentada por este trabalho a definição de uma arquitetura flexível baseada em agentes, que seja adequada ao modelo proposto e que possa ser analisada em ambiente real e simulado.

1.3 Organização do Documento

O restante desse documento está estruturado em cinco capítulos: fundamentação teórica, trabalhos correlatos, proposta de trabalho, experimentos realizados, conclusões e trabalhos futuros, conforme segue:

- No Capítulo 2 são apresentados os principais elementos da fundamentação teórica incluindo os conceitos de computação em nuvem e fog computacional, agentes inteligentes, sistemas multiagentes e modelos de confiança e reputação.
- No Capítulo 3 é apresentado um resumo dos trabalhos correlatos em conjunto com uma comparação entre características desses trabalhos.
- No Capítulo 4 são apresentados os principais elementos do modelo proposto, assim como a abordagem utilizada para a avaliação deste modelo.
- No Capítulo 5 são apresentados os experimentos que foram conduzidos para avaliar o modelo proposto juntamente com a análise dos resultados.
- Por fim, no Capítulo 6 são apresentadas as conclusões acompanhadas de considerações sobre caminhos futuros para este trabalho.

Capítulo 2

Fundamentação teórica

Neste capítulo são apresentados os conceitos básicos que nortearam o desenvolvimento deste trabalho.

2.1 Computação em nuvem

Segundo Mell and Grance (2011), computação em nuvem é um modelo para habilitar o acesso compartilhado de forma ubíqua, conveniente e sob demanda, a um *pool* de recursos computacionais configuráveis que podem ser providos e disponibilizados com um esforço mínimo de gestão ou de interação com o provedor desse serviço. Para Buyya et al. (2009), a nuvem computacional é um tipo de sistema distribuído que consiste em uma coleção de máquinas virtualizadas interconectadas que são dinamicamente provisionadas. Essas máquinas são apresentadas aos usuários como um conjunto unificado de recursos, disponibilizados com base em um acordo de nível de serviço estabelecido entre provedores e consumidores.

Os recursos disponibilizados na nuvem computacional são configurados dinamicamente para se ajustar à uma demanda variável, permitindo uma utilização mais eficiente. Segundo Buyya et al. (2011), os serviços de Computação em Nuvem podem ser divididos em três classes, de acordo com o nível de abstração da capacidade provida e do modelo de provisionamento do serviço (Fig. 2.1): (1) Infraestrutura como serviço (IaaS), (2) Plataforma como serviço (PaaS), e Software como serviço (SaaS). Esses diferentes níveis de abstração também podem ser organizados em um arquitetura onde serviços de um nível superior podem ser constituídos a partir de serviços do nível inferior.

Segundo Mell and Grance (2011), o modelo de computação em nuvem possui cinco características básicas:

- Serviços sob demanda - os recursos da nuvem ficam disponíveis ao usuário que pode provisioná-los e liberá-los conforme sua necessidade. Um ponto importante desta característica é que esse processo de manutenção de recursos feita pelo usuário não envolve interação humana com o provedor do serviço, ou seja, não há necessidade de um contato entre o usuário e um representante do provedor para que recursos como capacidade de armazenamento ou largura de banda sejam alocadas, pois todo esse processo deve acontecer de maneira automatizada. Armbrust et al. (2009) acrescenta que esta característica da nuvem elimina a necessidade de um compromisso inicial

por parte dos usuários acerca do volume total de recursos que serão necessários no futuro. Assim, a nuvem torna-se uma boa opção para consumidores que estão em uma fase prematura de definição de seus serviços, pois podem alocar capacidade da nuvem de maneira gradativa à medida que sua necessidade aumenta.

- Acesso amplo por meio da rede - os recursos da nuvem são acessíveis por meio dos protocolos e mecanismos padrões utilizados na rede, permitindo, dessa forma, que diferentes tipos de equipamentos possam ter acesso à esses recursos.

- *Pool* de recursos - os provedores de serviço de nuvem gerenciam sua infraestrutura de maneira a formar um pool de recursos computacionais que podem ser alocados e liberados de acordo com a demanda dos usuários. Esses recursos podem ser virtuais ou físicos e, geralmente, seus usuários não possuem um domínio direto acerca de detalhes sobre a sua localização na infraestrutura do provedor. Em alguns casos o usuário pode definir uma região geográfica tal como um estado e/ou cidade onde os recursos devem ser alocados, respeitada, entretanto, a disponibilidade de recursos do provedor nesta localidade. Um exemplo prático são os serviços de nuvem computacional disponibilizados pela Amazon[1] e pela DigitalOcean[2]. Provedores desse tipo permitem que os usuários definam uma região geográfica para que seus recursos sejam alocados. Essas regiões são selecionadas com base na distribuição dos *datacenters* pertencentes a essas empresas. No caso da Amazon o usuário pode selecionar, por exemplo, uma localidade no norte do estado da Virgínia nos EUA, ou em São Paulo no Brasil. Na DigitalOcean outras localidades podem ser selecionadas, como Londres no Reino Unido ou Bangalore na Índia.

- Elasticidade rápida - a capacidade alocada na nuvem pode aumentar dinamicamente para suprir um aumento repentino na demanda ou pode ser diminuída para economizar recursos quando há uma queda na demanda. Neste sentido, Armbrust et al. (2009) explica que esta característica da nuvem cria a ilusão da existência de recursos infinitos que são disponibilizados conforme a necessidade dos usuários.

- Serviço mensurado - os recursos da nuvem podem ser gerenciados de maneira transparente entre usuário e provedor. A utilização de virtualização e as características relativas à alocação sob demanda exigem que diversas métricas de serviço sejam monitoradas como tempo de processamento, memória alocada, capacidade de armazenamento e largura de banda utilizada, entre outras. Isso facilita a implementação de um modelo *pay-per-use* onde os usuários pagam somente pelos recursos que efetivamente usaram de forma estratificada, ou seja, com valores definidos para cada tipo de recurso (largura de banda, memória, capacidade de armazenamento, tempo de processamento, etc.). Buyya et al. (2011) salienta a importância da utilização de um modelo de cobrança baseado em unidades mais granulares de tempo, como horas ou minutos, de maneira a permitir que os usuários liberem recursos alocados assim que estes não sejam mais necessários, resultando em uma economia para o usuário e para o provedor.

[1] https://aws.amazon.com/
[2] https://www.digitalocean.com/

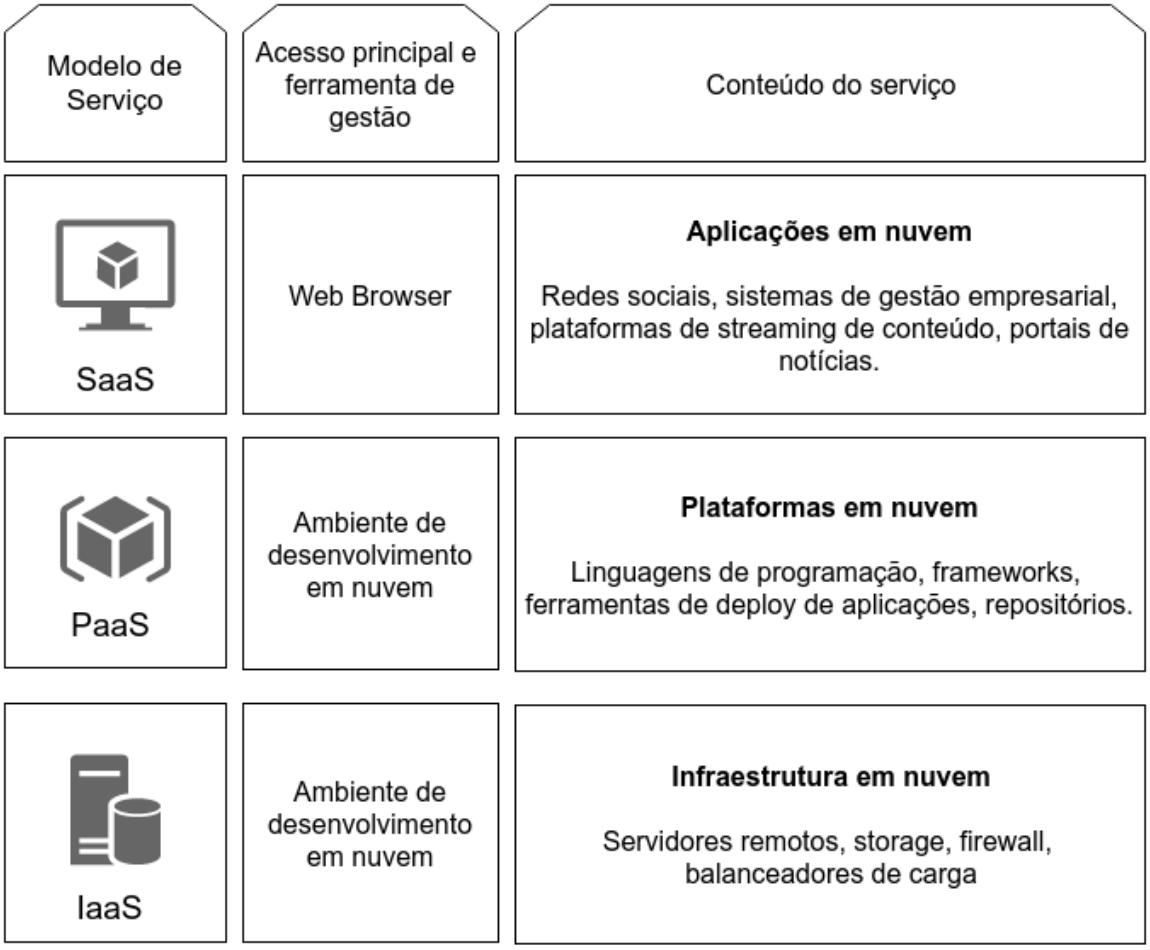

Figura 2.1: Classes de serviço do modelo de nuvem computacional (adaptada de Buyya et al., 2011).

Qualidade de serviço

A relação entre consumidor e provedor é balizada por métricas bem definidas dos serviços disponibilizados na nuvem. Por criar uma grande dependência em relação a esses serviços, o consumidor deve se atentar para o nível de qualidade percebido durante o uso da nuvem. Em muitos casos os serviços acessados na nuvem são críticos para os consumidores e problemas em seu fornecimento pode afetá-los dramaticamente. Uma empresa de comércio eletrônico, por exemplo, que utilize um serviço de IaaS para hospedar seu site de compras pode ter um prejuízo considerável com uma falha no serviço de nuvem que impeça os usuários de acessar o seu site.

Nesse sentido, a QoS é um parâmetro utilizado para medir o grau de qualidade dos diversos aspectos envolvidos na disponibilização de serviços em nuvem. A confiabilidade, a disponibilidade e a performance das aplicações e dos recursos disponibilizados na nuvem são características consideradas para aferir a QoS (Ardagna et al., 2014).

Os requisitos e métricas de QoS definidos para o serviço de nuvem são previamente acordados entre consumidor e provedor ficam registrados em um Acordo de Nível de Ser-

viço (*Service Level Agreement* - SLA) (Buyya et al., 2009). Na visão do consumidor os requisitos de QoS acordados no SLA são utilizados como base para verificar se as características dos serviços que foram divulgadas pelo provedor de nuvem estão sendo cumpridas e exigir que penalidades sejam aplicadas quando não forem atendidos os requisitos. A visão do provedor de serviços em relação aos níveis de QoS acordados envolve questões mercadológicas e decisões acerca dos custos relacionados aos recursos sendo providos para atender aos requisitos de QoS esperados pelos consumidores e o retorno financeiro obtido a partir do fornecimento desses recursos (Ardagna et al., 2014).

2.2 Fog computacional

O conceito de fog computacional foi proposto em 2012 pela CISCO como uma extensão do paradigma de computação em nuvem (Mahmud et al., 2018). A ideia principal era utilizar a capacidade computacional dos diversos dispositivos que se encontram espalhados na fronteira da rede. Assim, a fog computacional forneceria uma plataforma virtualizada que disponibiliza recursos computacionais, como capacidade de processamento e armazenamento, entre os dispositivos espalhados na fronteira da rede e os servidores na nuvem Bonomi et al. (2012). Com isso, busca-se diminuir a latência de comunicação inerente à distância entre cliente e servidor colocando nós com capacidade de processamento e armazenamento mais próximos dos dispositivos finais.

Vaquero and Rodero-Merino (2014) definem fog computacional como um cenário em que uma grande quantidade de dispositivos heterogêneos e descentralizados se comunicam e cooperam para executar tarefas sem a intervenção de terceiros. Na visão de Dastjerdi and Buyya (2016) a fog computacional é um paradigma distribuído que fornece serviços similares aos fornecidos pela nuvem na fronteira da rede. Conforme a visão apresentada por Dastjerdi et al. (2016a), a fog computacional engloba componentes que são executados tanto na nuvem quanto nos dispositivos localizados na fronteira da rede, provendo recursos computacionais, comunicação e integração de dados de forma a suportar aplicações sensíveis à latência e que estão distribuídas geograficamente.

O modelo de fog computacional (Figura 2.2) é composto por três elementos básicos: os *datacenters* de nuvem, os nós de fog e os dispositivos finais. Os *datacenters* de nuvem englobam os recursos providos pela nuvem como capacidade de processamento e armazenamento. Os dispositivos finais são os elementos que utilizam a capacidade da fog computacional para executar suas tarefas. Esse dispositivos variam desde *smartphones* e computadores *desktop* à sensores e atuadores de IoT. Por fim, os nós de fog são representados por dispositivos que atuam como provedores de serviço localizados nas proximidades dos dispositivos finais. Ou seja, são os nós de fog que utilizam parte ou a totalidade de seus recursos para atender às requisições de serviço dos dispositivos finais. Essas requisições seriam por padrão direcionadas aos *datacenters* de nuvem, mas no modelo de fog elas são atendidas pelos nós de fog.

A diminuição da latência de resposta é uma vantagem direta que pode ser obtida por meio da utilização da fog em determinados cenários. Porém, outras vantagens podem ser obtidas por meio do uso do modelo de fog computacional, como:

- Redução de tráfego de rede;
- Escalabilidade de serviços; e

- Redução de custos.

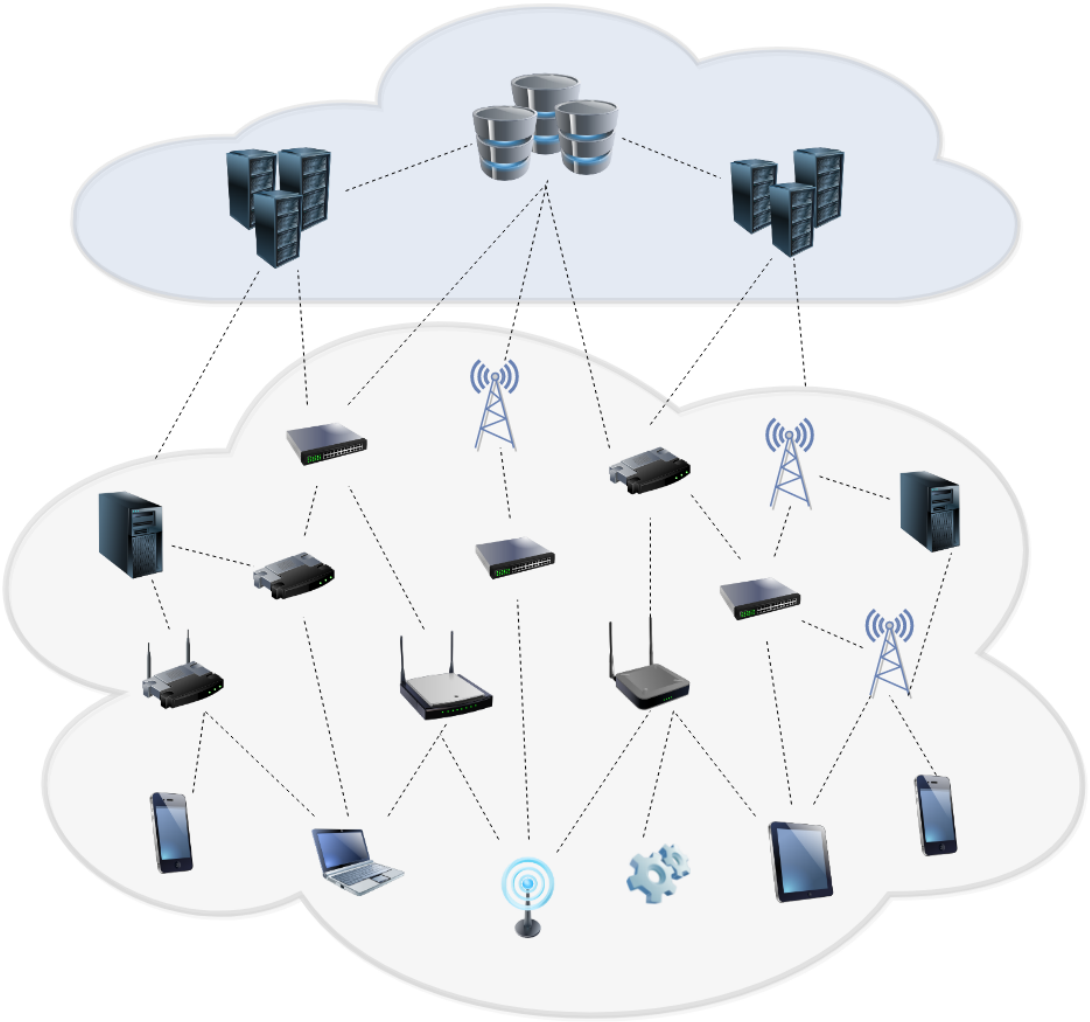

Figura 2.2: Visão geral do modelo de fog computacional

A Figura 2.3 apresenta uma arquitetura abstrata do modelo de fog computacional. Essa arquitetura está divida em três camadas. A primeira camada, no topo da figura, é composta pelos serviços de nuvem computacional. A segunda camada é composta pelos equipamentos de rede, responsáveis por conectar os usuários ao serviço de nuvem computacional. Por fim, a terceira camada é composta pelos dispositivos usuários que utilizam os elementos da segunda camada para se conectar uns aos outros e aos serviços de nuvem (Sarkar et al. (2018), Dastjerdi et al. (2016b)).

Os nós de fog podem ser implantados por quaisquer dos elementos da segunda e terceira camada da arquitetura da Figura 2.3. Em algumas propostas de uso da fog computacional pode-se encontrar uma tendência ao uso de dispositivos de rede como nós de fog. É o caso das propostas apresentadas em Varghese et al. (2017), Iotti et al. (2017), Bittencourt et al. (2017) e Alam et al. (2016), que utilizam roteadores, *gateways*, e controladoras de pontos de acesso como nós de fog. Nesse mesmo sentido outro trabalhos propõem

a utilização dos próprios dispositivos finais como nós de fog. Dessa forma, os usuários poderiam atuar tanto como consumidores quanto como provedores de serviço no âmbito da fog computacional. Esse tipo de abordagem pode ser encontrada em Lin and Shen (2017), He et al. (2017) e Skarlat et al. (2016).

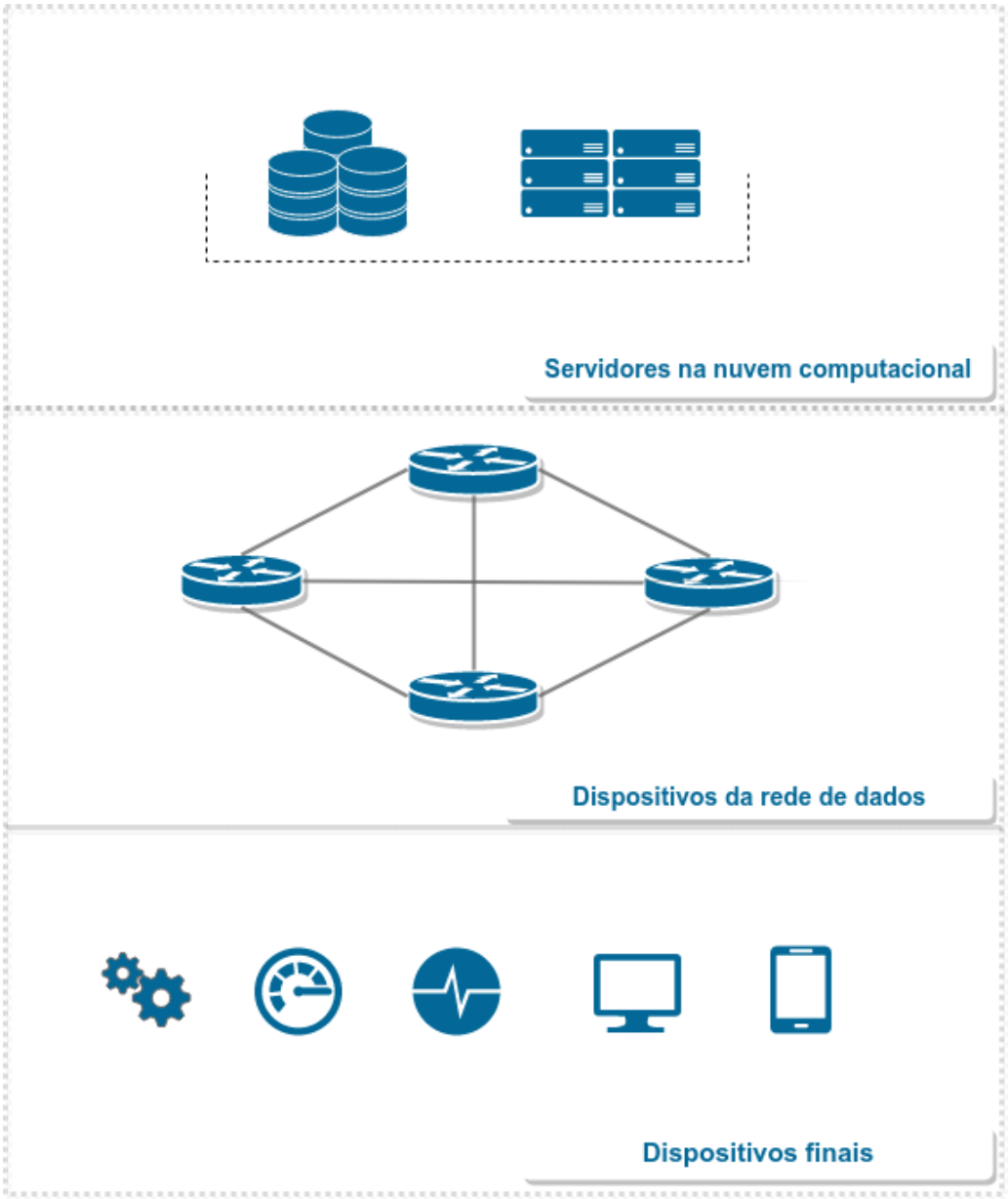

Figura 2.3: Arquitetura abstrata da fog computacional

2.2.1 Modelos de implantação

Partindo dos elementos básicos do modelo diversas formas de se implantar os nós de fog podem ser utilizadas. Os dispositivos representados na Figura 2.2 podem ser organizados de diversas maneiras de modo a compor uma estrutura para uma implementação específica de serviço que utilize o modelo de fog. Nesse sentido, pode-se aplicar à fog computacional

a mesma visão utilizada para a nuvem com dois atores principais: o provedor de serviço, que utiliza a estrutura da fog computacional para executar um serviço, e o consumidor de serviço que se conecta à fog utilizando um dispositivo final de modo a consumir o serviço ofertado pelo provedor. Essas figuras não são facilmente distinguíveis em todos os cenários de uso. Em algumas implementações, o provedor e o consumidor podem estar representados por uma mesma entidade, como, por exemplo no caso em que uma empresa de controle de tráfego urbano implementa um sistema de coleta de dados por sensores IoT, onde esses sensores serão os dispositivos que consumirão o serviço provido na fog. Nesse caso, os sensores e o serviço provido pertencem à mesma empresa. Mesmo nesse caso, apesar de se tratar de um só ente, pode-se empregar a visão dual de provedor/consumidor, mesmo que de maneira abstrata, de modo a manter a mesma abordagem utilizada no âmbito da nuvem computacional onde o provedor de serviço é o responsável final por manter os requisitos de QoS e os demais parâmetros requisitados para o serviço e o consumidor é responsável por cobrar o cumprimentos desses requisitos.

Apesar de não possuir um modelo de implantação fixo, é possível buscar uma compreensão sobre as topologias de implantação da fog computacional, abstraindo os detalhes de cada contexto específico. Como não foi identificado na literatura consultada uma topologia padrão de implantação para a fog computacional, e partindo da necessidade de se ter um entendimento acerca da estruturação dos elementos em um fog computacional segundo suas atribuições e papeis, foi definida neste trabalho quatro topologias de implantação para a fog computacional, conforme pode ser visualizado na Figura. 2.4. Essas topologias seguem descritas na sequencia do texto.

Fixa

Nesta categoria os nós de fog são implantados em dispositivos previamente selecionados pelo provedor do serviço. As propostas apresentadas em Iotti et al. (2017), Varghese et al. (2017) e Bittencourt et al. (2017) seriam classificadas nesta categoria. Os nós de fog podem ser selecionados a partir de dispositivos já existentes na infraestrutura de rede ou podem ser disponibilizados pelo provedor de serviço. Um exemplo é o caso da proposta de Varghese et al. (2017), onde computadores *single-board* são previamente configurados e ligados à rede de dados para serem utilizados como nós de fog. Em resumo, nessa categoria estão os modelos de implantação onde os nós de fog são estáticos e a implantação de novos nós ou alteração dos nós já existentes exige uma intervenção direta do provedor do serviço.

Controlada

Nesta categoria estão os modelos de implantação que utilizam um nó de fog para realizar o controle e coordenação dos demais nós de fog implantados. Esse nó de controle também é implantado na fronteira de rede e fica responsável por os demais nós que estão nas suas proximidades. Nesta categoria este nó de controle é previamente disponibilizado e/ou selecionado pelo provedor de serviço no mesmo sentido que os nós de fog na topologia Fixa. Ou seja, inclusão de novos nós de controle ou alterações nesses nós são realizadas pelo próprio provedor de serviço, e os dispositivos consumidores do serviço não precisam se atentar para questões relativas à confiabilidade e QoS provido por esse nó de controle. Outro ponto que diferencia esta da categoria de modelo Fixa é a dinamicidade dos nós de fog.

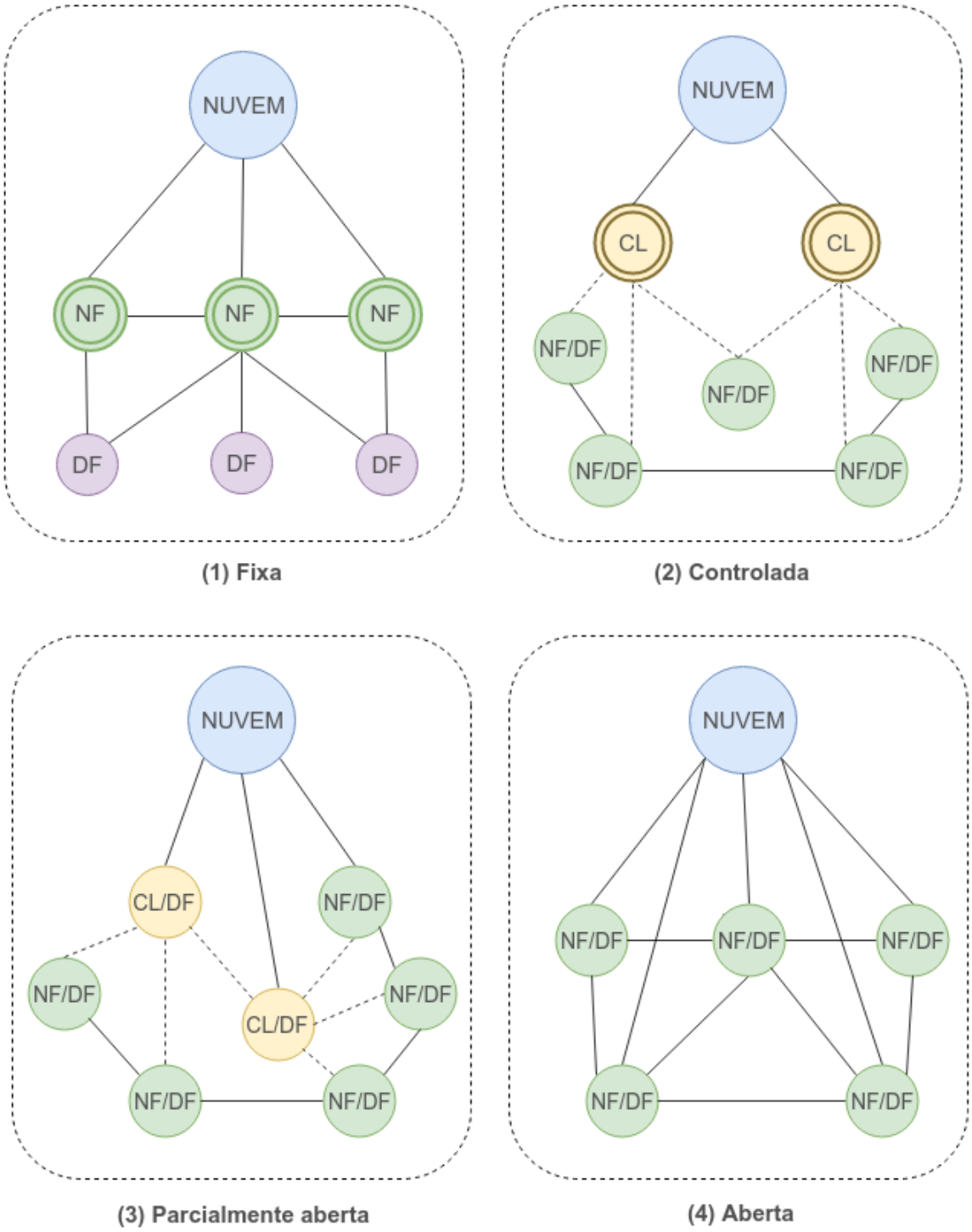

Figura 2.4: Topologias de implantação para a fog computacional, onde NUVEM é o servidor na nuvem computacional; NF o nó de fog; CL o nó de coordenação local; e DF o dispositivo final. As linhas tracejadas entre os elementos representam interações de controle e as linhas sólidas representam interações de consumo de serviço

Nesta categoria e nas demais, com exceção da Fixa, os nós de fog podem ser apontados de maneira dinâmica pelo provedor de serviço, aplicando a abordagem onde dispositivos

finais dos consumidores de serviço são utilizados como nós de fog. Dessa forma, os nós de fog podem ser implantados, modificados e removidos de maneira contínua de acordo com a demanda. O provedor de serviço deve implementar abordagens para selecionar os nós de fog e para garantir que os nós selecionados continuarão a prover o serviço na fog computacional com os requisitos esperados de QoS.

Parcialmente aberta

Assim como na topologia Controlada, é utilizada a figura de um nó de controle local na fronteira de rede. Porém, nesta categoria, esse nó de controle também tem uma característica de dinamicidade e pode ser apontado dentre os dispositivos finais dos consumidores do serviço. O ponto principal aqui é a ausência de controle direto do provedor de serviço, que deve implementar métodos para selecionar os dispositivos que serão utilizados como nós de controle, e deve conduzir avaliações contínuas para garantir que este nó está funcionamento corretamente e fornecendo um serviço com a QoS esperada. A proposta apresentada em Skarlat et al. (2016) segue este modelo de implantação e identifica como uma comunidade (ou *fog colony*, conforme denominado pelos autores) o conjunto formado pela nó de coordenação local e os nós de fog ligados a ele. Tarefas que são requisitadas no âmbito desta comunidade são distribuídas pelo nó de coordenação local para os demais nós da comunidade.

Aberta

Nesta categoria, como na Fixa, não existe a figura de um nó de coordenação local na fronteira de rede. Porém, ao contrário da categoria Fixa e na mesma linha das categorias Controlada e Parcialmente aberta, nesta categoria os nós de fog são apontados de maneira dinâmica podendo ser dispositivos finais dos consumidores de serviço. Por não possuir um controle local e devido à dinamicidade dos nós de fog, esta categoria apresenta complexidades para a garantia da QoS ofertado na fog computacional. O provedor de serviço deve implementar mecanismos para permitir a seleção de nós de fog aptos para fornecer um serviço de qualidade e, com vistas à garantir a QoS, deve executar processos de verificação para excluir nós de fog defeituosos ou que estejam deliberadamente desalocando capacidade de seus recursos de modo a impactar o serviço. A proposta apresentada em Lin and Shen (2017) se encaixa nesta topologia de implantação, com uma arquitetura de jogos online em fog computacional que utiliza os nós de fog para realizar a renderização de vídeos dos jogos na fronteira da rede visando reduzir a latência de resposta.

2.2.2 Cenários de aplicação

O modelo de fog computacional esteve inicialmente fortemente vinculado ao contexto de IoT. Apesar dessa vinculação ainda existir, diversos outros cenários de aplicação para a fog foram propostos. No trabalho seminal do modelo de fog (Bonomi et al., 2012) já é possível encontrar uma lista de cenários passíveis de benefícios por meio da implantação do modelo de fog. Listas similares de cenários de aplicação para a fog computacional podem ser encontradas em Yi et al. (2015), Dastjerdi et al. (2016b) e Dastjerdi and Buyya (2016). Alguns desses cenários são descritos a seguir.

Distribuição de conteúdo Os nós de fog podem ser utilizados para a criação de bases de cache de dados mais próximos aos dispositivos finais dos usuários. Os dados armazenados nesses caches são acessados com um menor número de saltos na rede gerando redução de tráfego e melhorando o tempo de resposta. Portais de conteúdo na web poderiam se beneficiar de uma infraestrutura de caches desse tipo, fazendo a distribuição de parte dos seus conteúdos de mídia, como imagens e vídeos, de modo a reduzir o tempo de carregamento de suas páginas. Esse formato de distribuição de mídias é análogo ao das CDN's (*Content Delivery Networks*), porém é mais flexível e dinâmico pois utiliza nós de processamento e armazenamento mais leves que podem ser implantados de maneira ágil conforme a demanda. Além dessa dinamicidade, os nós de fog apresentam uma maior capilaridade podendo regionalizar a distribuição de mídias com caches de dados em localidades mais próximas aos usuários. Outra vantagem obtida refere-se ao custo com banda de rede que pode ser economizado. Como a proposta apresentada em Iotti et al. (2017), que utiliza nós de fog para realizar *cache* dos conteúdos mais acessados por usuários conectados a uma rede *wi-fi* local. Nesse caso, a proposta visa diminuir o consumo da banda, evitando que um mesmo conteúdo seja retransmitido entre a origem (por exemplo o servidor de um portal de conteúdos) e o destino (os dispositivos dos usuários ligados à rede).

Realidade aumentada Aplicações de realidade aumentada necessitam de grande poder computacional para processar as imagens e sons captados por sensores e de uma capacidade de banda considerável para transmitir esses dados captados para o processamento remoto. Dispositivos como o *Google Glass*[3], *Sony SmartEyeglass*[4], *Microsoft HoloLens*[5], e o aplicativo móvel *Google Translator*, utilizam a realidade aumentada para auxiliar o usuário em diversas tarefas, como, por exemplo, identificar o nome e telefone de uma pessoa por meio de reconhecimento facial. Neste exemplo, um grande *delay* entre a captura da imagem da face de uma pessoa e o retorno da informação acerca de sua identificação pode impactar drasticamente a utilidade da aplicação para o usuário (reconhecer a pessoa somente depois que ela já não está mais presente pode invalidar toda a proposta). Nesse sentido, a aplicação de um modelo de fog computacional pode trazer ganhos para essas aplicações, reduzindo a latência envolvida no envio dos dados para processamento na nuvem e consequentemente também reduzindo o tráfego de dados na rede. O processamento poderia ser feito em um nó de fog mais próximo ao usuário que está utilizando o equipamento.

Assistência à saúde (*healthcare*) Sistemas de assistência à saúde são um campo promissor de aplicação para IoT. Sensores diversos são utilizados para monitoramento de sinais vitais e outros aspectos relativos à condição de saúde dos pacientes. Esses sistemas podem estar equipados em pequenos aparelhos ligados ao corpo dos pacientes (*wereables*) de modo a capturar dados para análise. Assim como no caso das aplicações de realidade aumentada, o requisito de tempo de resposta se impõem para os sistema de assistência à saúde. Em Dastjerdi and Buyya (2016) é dado o exemplo de um sistema para detectar e prever quedas em pacientes por derrames. Esse sistema utiliza o modelo

[3]https://x.company/glass/
[4]https://developer.sony.com/develop/smarteyeglass-sed-e1/
[5]https://www.microsoft.com/hololens

de fog para realizar o processamento dos dados em nós mais próximos aos sensores e reduzir a latência envolvida neste processo de coleta de dados, processamento e tomada de ação. Em Dastjerdi et al. (2016b) é descrita uma proposta de ambiente inteligente de monitoramento de saúde para pessoas idosas que utiliza o modelo de fog.

Jogos O nicho de jogos digitais está em contínuo crescimento. Atualmente há uma convergência entre as plataformas de jogos para consoles (como *XBox* e *Playstation*), computadores pessoais e dispositivos móveis, onde todos estão conectados em servidores remotos que proporcionam a interação entre os usuários e permite a disponibilização de jogos com múltiplos jogadores como os MMOG (*Massively Multiplayer Online Game*). Os servidores responsáveis por hospedar esses jogos recebem milhares de usuários por dia estando estes usuários espalhados geograficamente em diferentes continentes. Em um mesmo país jogadores situados em localidades afastadas do servidor do jogo podem ter sua experiência de uso dramaticamente afetada ou mesmo impossibilitada. Em outros casos, a variações de *delay* acarreta em problemas de comunicação e interação entre os jogadores e pode resultar em uma queda no número de usuários. Essa perda de usuários gera um impacto financeiro negativo para as empresas que produzem e disponibilizam esses jogos e para os provedores de serviços de comunicação.

Esse contexto é análogo ao de realidade aumenta, sendo até complementares em alguns casos, dado que alguns jogos se baseiam nessa técnica para proporcionar uma experiência mais interessante aos usuários. Sendo este o caso do jogo Pokemon GO[6], que conta com mais de 100.000.000 de instalações na plataforma Android[7]. Implantar os servidores dos jogos em nós de fog mais próximos aos usuários podem trazer os benefícios de redução de latência e até mesmo redução de variação de *delay* entre diferentes jogadores. Além disso, utilizando esquemas de *cache*, como explicado no contexto de distribuição de conteúdo, pode-se melhorar tanto o desempenho dos jogos quanto reduzir o consumo de banda e o tráfego de rede. Varghese et al. (2017) apresenta uma proposta de uso deste tipo de *cache* na fog para um jogo similar ao Pokemon GO.

2.2.3 Ambientes e ferramentas de experimentação

Não existe um ambiente ou ferramenta padrão para a realização de experimentos em fog computacional. Como os cenários de aplicação da fog são muito variados, o conjunto de ferramentas e ambientes que podem ser utilizados como base para implementação de estudos de caso é amplo. Na literatura disponível é possível encontrar a utilização das ferramentas CloudSim, iFogSim, OMNeT++ e Peersim, além dos ambientes do PlanetLab e de provedores de nuvem pública como Amazon AWS. Alguns dessas ferramentas e ambientes são descritos na sequência.

Cloudsim e iFogSim

CloudSim é um framework desenvolvido pelo CLOUDS *Laboratory* da Universidade de Melbourne, cujo objetivo é fornecer uma plataforma que permita a modelagem e simulação de infraestruturas de *Cloud Computing*. O framework oferece componentes básicos como

[6]https://www.pokemongo.com
[7]https://play.google.com/store/apps/details?id=com.nianticlabs.pokemongo

hosts, *virtual machines* e *schedulers* que permitem que o usuário realize a modelagem de serviços de *Cloud* e a simulação de diferentes cenários de IaaS, PaaS e SaaS (Calheiros et al., 2011).

O CloudSim está estruturado em quatro camadas. No contexto da ferramenta, um *datacenter* possui um conjunto de *host*, que por sua vez podem executar um conjunto de máquinas virtuais. O CloudLet representa um serviço a ser executado por uma máquina virtual. As máquinas virtuais são definidas a partir dos recursos como, memória, capacidade de armazenamento, largura de banda, número de unidades de processamento e capacidade de processamento (definida em milhões de instruções por segundo - *mips*). O CloudSim também possui componentes que implementam políticas de alocação de recursos e de escalonamento de tarefas que podem ser estendidos pelo usuário.

Mais recentemente, em 2016, foi disponibilizado o iFogSim, uma ferramenta que estende o CloudSim e implementa alguns componentes voltados para a simulação de ambientes de fog computacional. A ferramenta permite a simulação dos nós de fog, de sensores e atuadores de aplicações de IoT e dos *datacenters* de nuvem, e conta com uma coleção de classes que suportam a modelagem de uma aplicação de IoT permitindo a definição das conexões entre esses elementos e os demais dispositivos da fog, além de permitir a construção do padrão de comunicação entre esses elementos, como frequência de mensagens, número de componentes da aplicação e a dependência entre os componentes (Gupta et al., 2016).

O iFogSim pode ser utilizado para a simulação de modelos para a gestão de recursos da fog e para distribuição de módulos dos sistemas de IoT, permitindo que seja realizada a avaliação quanto ao impacto desses modelos na latência, no consumo de energia, nos custos operacionais e no uso da rede. O modelo básico de simulação da ferramenta é baseado no fluxo sensor-processamento-atuador. Esse modelo segue a visão básica dos sistemas de IoT, onde os sensores coletando dados que são enviados para processamento remoto gerando insumos para a definição de comandos que são enviados para os atuadores.

Peersim

O Peersim é um simulador implementado na linguagem JAVA (Gosling et al., 2014) voltado para a avaliação de protocolos *peer-to-peer* (Montresor and Jelasity, 2009a). O ambiente fornecido pelo Peersim é altamente escalável e suporta diferentes tipos de protocolos de simulação. A ferramenta fornece dois modelos de simulação: baseado em ciclos e baseado em eventos. O baseado em ciclos organiza a execução dos experimentos em ciclos periódicos com duração pré-definida. Com o baseado em eventos, a simulação é realizada com base nas trocas de mensagem entre os nós. Os dois modelos podem ser combinados em um mesmo experimento. Uma simulação é formada por uma rede, modelada como uma lista de nós, por um conjunto de protocolos, de inicializadores e de controles. Cada nó possui um conjunto de protocolos relacionados. Os inicializadores são executados no começo de cada simulação e são responsáveis por carregar parâmetros e preparar o ambiente para a execução do experimento. Os controles são invocados durante a simulação e podem ser utilizados para agregar dinamicidade à rede adicionando e removendo nós, e pode ser utilizados para monitoramento e coleta de dados da simulação. Os protocolos definem operações que serão realizadas pelos nós, no modelo baseado em ciclo essas opera-

ções são realizadas a cada novo ciclo e no baseado em eventos as operações são executadas pelo protocolo de acordo com a sequência de eventos gerados.

PlanetLab

PlanetLab (Chun et al., 2003) é uma rede mundial para pesquisa e desenvolvimento. A rede é constituída por nós de processamento espalhados por diversas localidades do planeta. Os nós são executados em máquinas dedicadas disponibilizadas por um grande grupo de instituições consorciadas do segmento acadêmico e da industria. Atualmente a rede do PlanetLab possui 1353 nós distribuídos em 717 localidades. Aos pesquisadores vinculados às instituições consorciadas é facultada a seleção de um ou mais nós em localidades diversas para a execução de seus experimentos.

A arquitetura do PlanetLab é dividida em um conjunto de *slices*. Cada *slice* é constituída por um conjunto de máquinas virtuais (*virtual machine* - VM) onde cada VM é executada em um nó ligado à rede do PlanetLab. A cada VM é alocada parte da capacidade de processamento, da memória e da capacidade de armazenamento do nó. Como mais de uma VM pode ser executada ao mesmo tempo, o nó conta com um *Virtual Machine Monitor* que coordena a alocação de recursos entre as diferentes VM's sendo executadas (Chun et al., 2003).

Os nós do PlanetLab são implantados em servidores alocados pelas instituições vinculadas. Desse modo, á uma distribuição desses nós permitindo que uma mesma *slice* possa conter VM's que estejam sendo executadas por nós em localidades diversas. Isso permite que os experimentos executados na infraestrutura do PlanetLab tenham acesso à um ambiente real de processamento e de comunicação de dados, já que o PlanetLab utiliza a infraestrutura de comunicação da Internet (Chun et al., 2003).

A Rede Nacional de Ensino e Pesquisa (RNP) firmou uma parceria com a rede do PlanetLab em 2004, realizando a implantação de três nós no Rio de Janeiro, no Ceará e no Rio Grande do Sul. Atualmente a RNP conta com mais um nó vinculado ao PlanetLab localizado no Pará. A cessão destes nós para a rede do PlanetLab permitiu que a RNP possa criar *slices* nos nós do PlanetLab. As instituições vinculadas à RNP podem solicitar a criação de *slices* para a condução de experimentos na rede do PlanetLab.

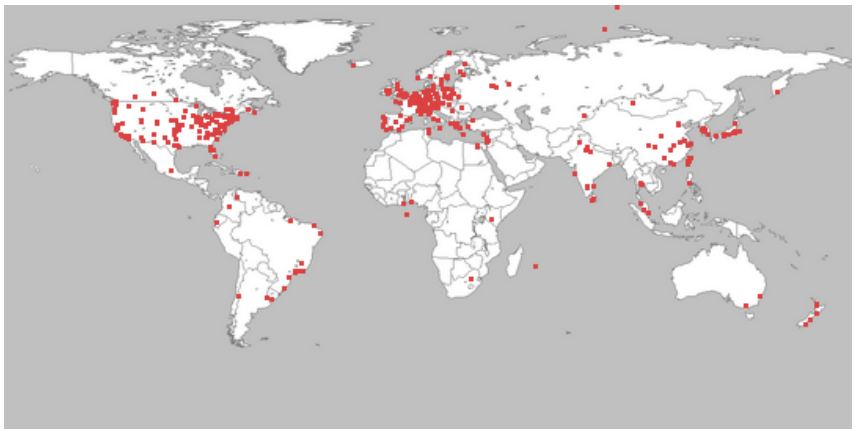

Figura 2.5: Mapa com distribuição dos nós da rede do PlanetLab (retirada do site oficial)

2.3 Agentes inteligentes

Os agentes são entidades computacionais situados em um ambiente, que realizam ações autônomas de forma a atingir seus objetivos. Os agentes percebem o ambiente em que estão inseridos por meio de seus sensores e atuam nesse ambiente por meio de seus atuadores (Russell and Norvig, 2010), conforme representado na Figura 2.6. No mesmo sentido, Wooldridge (2009) destaca que os agentes executam ações autônomas no ambiente em que estão situados de maneira a atingir seus objetivos.

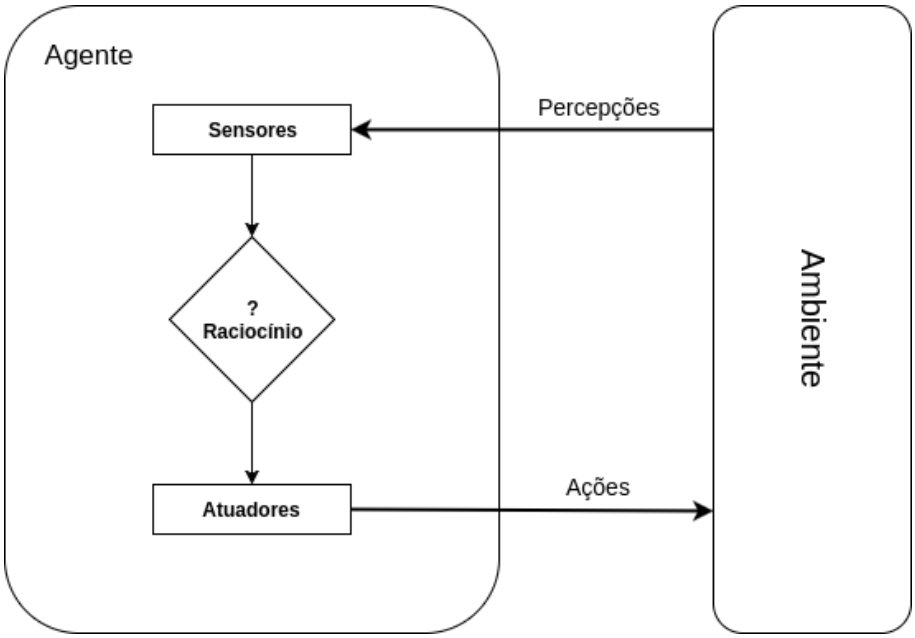

Figura 2.6: Arquitetura básica de um agente (adaptada de Russell and Norvig (2010)).

Para Wooldridge (2009), um agente inteligente é um sistema computacional capaz de ter ações autônomas flexíveis em determinado ambiente. Segundo o autor, a flexibilidade do sistema está relacionada a características básicas de agentes inteligentes que incluem: reatividade, pro-atividade e sociabilidade.

Na visão de Russell and Norvig (2010), os agentes podem ser classificados em quatro tipos:

- Agentes reativos simples - considera somente o que pode perceber do ambiente em determinado instante de tempo, ignorando quaisquer percepções anteriores. Essa característica torna mais simples a implementação do agente, porém limita sua inteligência, pois exige que o ambiente seja plenamente observável, fazendo com que falhas ou impedimentos na percepção afetem de forma dramática as decisões que o agente poderá tomar.
- Agentes reativos baseados em modelos - quando o ambiente é parcialmente observável, uma estratégia para a implementação do agente seria armazenar as percepções anteriores, constituindo um tipo de estado interno do agente. Para construir esse estado interno, é necessário que o agente possua um modelo sobre o ambiente em que está inserido, de forma a possuir algum conhecimento que permita identificar

como o ambiente evolui ao longo do tempo e como as ações do agente afetam o ambiente.

- Agentes baseados em objetivos - conhecer o estado atual do ambiente nem sempre é suficiente para o agente decidir o que deve fazer. Em determinadas situações, o agente necessita de objetivos definidos que descrevam estados desejados. Combinando essas informações com seu estado interno, o agente poderá responder questões mais complexas, de ações futuras que o levarão a alcançar seus objetivos. Agentes com essa característica tendem a ser mais flexíveis pois o conhecimento que apoia suas decisões é explícito e pode ser modificado.

- Agentes baseados em utilidade - objetivos sozinhos podem não bastar para basear as decisões de um agente, principalmente quando existem objetivos contraditórios ou vários objetivos que nem sempre podem ser atingidos. O agente pode aplicar uma medida que verifique a validade ou mesmo quão satisfatório é determinado resultado. Para tanto, o conceito de utilidade pode ser aplicado, de maneira a permitir que um determinado agente escolha, entre o conjunto de ações possíveis, aquela que leva a um resultado mais satisfatório. Para medir a satisfação ou quão "bom"é determinado estado resultante de uma ação, o agente aplica uma função de utilidade (Wooldridge, 2009). A função de utilidade ($U(s)$) mapeia um estado (ou uma sequência deles) em um valor numérico que descreve o grau de recompensa associado. O valor obtido através da função $U(s)$ permite avaliar e priorizar objetivos/ações visando obter maior segurança e desempenho. Um agente baseado em utilidade toma suas decisões de modo a maximizar sua utilidade esperada. Aqui o conceito de utilidade esperada reflete uma limitação própria ao agente que é encontrada na maior parte dos ambientes (principalmente aqueles que refletem uma situação real). Não há como o agente determinar todos os estados resultantes de uma ação e dessa forma calcular a utilidade de cada um deles. Assim, o agente trabalha com esse conceito de utilidade esperada, que é definida com base em uma soma das utilidades dos estados resultantes ponderados pela probabilidade daqueles estados serem atingidos. Conforme apresentado em Russell and Norvig (2010), a Equação 2.1 é utilizada para o cálculo da utilidade esperada:

$$EU(a|e) = \sum_{s'} P(Result(a) = s'|a, e)U(s')\,, \tag{2.1}$$

onde a função $P(Result(a) = s'|a, e)$ calcula a probabilidade de atingir o estado s' a partir da ação a e dadas as evidências e que podem ser observadas pelo agente no momento atual. Dessa forma, o agente seleciona dentre o conjunto de ações possíveis a que maximiza a utilidade esperada.

Segundo Wooldridge (2009), a arquitetura implementada por um agente pode ser caracterizada como:

- Deliberativa - o agente possui um modelo explícito de representação simbólica do mundo e toma suas decisões por meio de um raciocínio lógico com base em padrões e em manipulações simbólicas. Nessa classificação de arquitetura se incluem os agentes de raciocínio prático e orientado a objetivos.

- Reativa - agentes que implementam essa arquitetura não possuem nenhum tipo de modelo simbólico de representação do mundo e não fazem uso de um modelo de lógica formal para raciocinar e tomar decisões. Eles utilizam um paradigma de automata situacional, baseado em especificações declarativas e tomam decisões baseados no seu estado atual.
- Híbrida - combina a abordagem reativa e deliberativa, construindo um modelo em subsistemas ou camadas, onde comportamentos reativos e pró-ativos são implementados. As camadas são dispostas em uma organização horizontal ou vertical. Quando organizadas horizontalmente, as percepções enviadas pelos sensores e as ações são realizadas por somente uma das camadas por vez, de forma hierárquica. Na organização vertical, cada camada está diretamente conectada aos sensores e pode receber entradas e realizar ações paralelamente as demais camadas.

Ambiente

Um dos pontos principais a serem tratados na definição da arquitetura de um agente concerne à caracterização do ambiente onde o agente está inserido e com o qual irá interagir por meio de suas ações. O ambiente é parte da definição de agente dada por Russell and Norvig (2010). Como destaca Weiss (2013) são os atributos do ambiente que afetam a forma como é delineado o processo decisório conduzido pelo agente. Segundo Russell and Norvig (2010), o ambiente pode ser classificado com base em seis dimensões:

- Completamente observável x parcialmente observável - em um ambiente completamente observável o agente não precisa manter um estado interno para acompanhar as mudanças do ambiente pois ele pode detectar, por meio de seus sensores, todas os aspectos acerca do ambiente que são relevantes para sua tomada de decisão. Em um ambiente parcialmente observável o agente pode coletar dados imprecisos ou faltantes acerca do ambiente.
- Determinístico x estocástico - em um ambiente determinístico o estado seguinte é definido a partir do estado atual e das ações executadas pelo agente, ou seja, não há incertezas acerca dos estados futuros. Um ambiente estocástico é caracterizado por não possuir uma previsão antecipada em relação a seu estado quando da execução das ações pelo usuário. Os resultados dessas ações no ambiente estocástico são quantificadas probabilisticamente.
- Episódico x sequencial - em um ambiente episódico o tempo pode ser segmentado em episódios que se sucedem ao longo da atuação do agente. Um agente que testa defeitos em circuitos integrados em uma linha de montagem estaria em um ambiente episódico. Esse exemplo mostra outra característica importante de um ambiente episódico, as ações tomadas no episódio atual não influenciam o próximo episódio. A detecção de um defeito em uma placa não influencia na detecção de defeito na placa seguinte movimentada na linha de montagem. Por outro lado, no ambiente sequencial as ações e decisões atuais produzem efeitos nas ações e decisões futuras.
- Estático x dinâmico - enquanto o agente está deliberando para decidir qual ação tomar o ambiente pode mudar. Se esse for o caso, classifica-se o ambiente como dinâmico. Caso o ambiente não se altere enquanto o agente delibera, classifica-se

o ambiente como estático. Um agente solucionador de problemas matemáticos está em um ambiente estático, pois o problema dado não muda durante o período em que o agente delibera sobre o resultado. Um agente que controla o voo de um *Drone*, por exemplo, está em um ambiente dinâmico, pois durante a deliberação acerca de um movimento a ser executado pelo *Drone* outros objetos podem estar se aproximando e se movendo ao redor do *Drone*.

- Discreto x contínuo - os ambientes discretos são definidos por ter um número finito de estados que podem ser atingidos por meio de um conjunto limitado de ações e de possíveis percepções. Um "jogo da velha" seria um exemplo de ambiente discreto. Em um ambiente dinâmico as características e variáveis relativas ao ambiente variam segundo intervalo de valores contínuos. O agente controlador do Drone estaria em um ambiente contínuo, pois a velocidade e a angulação da aeronave durante o voo, por exemplo, são medidos em valores contínuos e mudam gradativamente durante o tempo. Em suma, esta dimensão de classificação está ligada à percepção que o agente tem em relação ao tempo e sua influência nos estados do ambiente.
- Multiagente x agente único - um ambiente pode conter somente um agente ou vários agentes. Um agente resolvendo uma equação está em um ambiente de agente único. Um ambiente multiagente, como o próprio nome já revela, é constituído por vários agentes.

2.4 Sistemas multiagentes

Sistemas multiagentes (SMA) são compostos por múltiplos agentes interativos que possuem a capacidade de tomar decisões autônomas para satisfazer seus objetivos (Wooldridge, 2009). Nesse mesmo sentido, Jennings et al. (1998) apresenta uma visão que trata os SMA como coleções de agentes autônomos ligados uns aos outros de modo a resolver um dado problema. Em uma visão mais ampla, Zambonelli et al. (2003) divide os SMA em: (i) sistemas solucionadores de problemas, onde os agentes são definidos de modo a cooperar para atingir um mesmo objetivo; e (ii) sistemas abertos, formados por agentes que não compartilham um mesmo objetivo e possuem uma característica de dinamicidade onde os agentes podem entrar e sair do sistema a qualquer momento.

Jennings et al. (1998) define as seguintes características para os SMA:

- os agentes nos SMA possuem uma visão limitada e informações incompletas acerca do problema em questão e não podem resolvê-lo sozinhos.
- não existe um controle global do sistema.
- os dados nos SMA estão descentralizados e a computação é feita de modo assíncrono.

A Figura 2.7 apresenta uma estrutura básica para SMA. Os agentes se situam em um ambiente onde cada um possui uma esfera de influência que abrange partes diferentes do ambiente. As sobreposições entre as esferas de influência explicitam as relações de dependência que podem existir entre os agentes. Em SMA, os agentes podem estar ligados por relações que explicitam alguma estrutura organizacional, com níveis de hierárquicos de poder, ou em grupos de influência onde os agentes se organizam com base na similaridade entre suas características (Wooldridge, 2009).

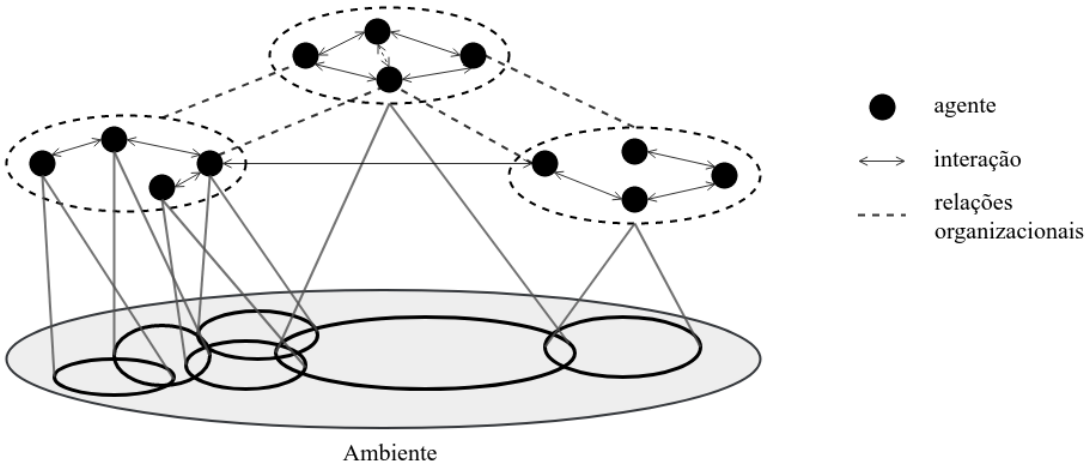

Figura 2.7: Estrutura de SMA (adaptada de Wooldridge, 2009).

2.4.1 Coordenação de agentes

A coordenação é a característica de SMA que permite que os agentes executem determinada ação de maneira compartilhada (Weiss, 2013). Como exposto em Jennings et al. (1998), a coordenação permite que os agentes explorem interações que possam lhes ajudar à atingir seus objetivos, e evitar aquelas interações que possam acarretar problemas e/ou dificuldades para a consecução desses objetivos. Wooldridge (2009) explica que, diferente de sistemas distribuídos tradicionais, os agentes em SMA tomam decisões de forma autônoma e dessa maneira podem não compartilhar os mesmos objetivos. Dessa forma, os agentes devem agir com base em alguma estratégia que os permita obter o resultado que lhes é mais favorável à partir da interação com os demais agentes.

A Figura 2.8 apresenta uma taxonomia parcial das formas de coordenação em SMA. Segundo Weiss (2013), a cooperação é um tipo de coordenação onde os agente não possuem objetivos conflitantes, enquanto a negociação refere-se ao contexto onde é realizada a coordenação entre agentes que possuem interesses próprios e podem ter objetivos conflitantes entre si.

2.4.2 Protocolo de comunicação

As linguagens de comunicação entre agentes em SMA tendem a ter como base a *speech act theory* (Wooldridge and Jennings, 1995). Por meio da *speech act theory* a linguagem natural humana é entendida a partir de ações comunicativas, como, por exemplo, requisições, sugestões, compromissos e respostas (Weiss, 2013). Do mesmo modo, Wooldridge (2009) explica que a partir da *speech act theory* a comunicação entre agentes pode ser estruturada por meio de ações comunicativas. Para o agente as ações comunicativas são similares às demais ações que ele executa para atingir seus objetivos, como, por exemplo, movimentar um braço mecânico.

As linguagens de comunicação entre agentes aplicam o conceito apresentado na *speech act theory* para permitir a interação entre diferentes agentes nos SMA. Essas linguagens de comunicação são constituídas como um conjunto de protocolos de comunicação que

suportam tipos diversos de mensagens e permitem tanto a troca de dados entre os agentes quanto a ação que representa a intenção do agente com aquela comunicação (Mayfield et al., 1996). São descritas a seguir duas importantes linguagens de comunicação entre agentes a KQML e a FIPA-ACL.

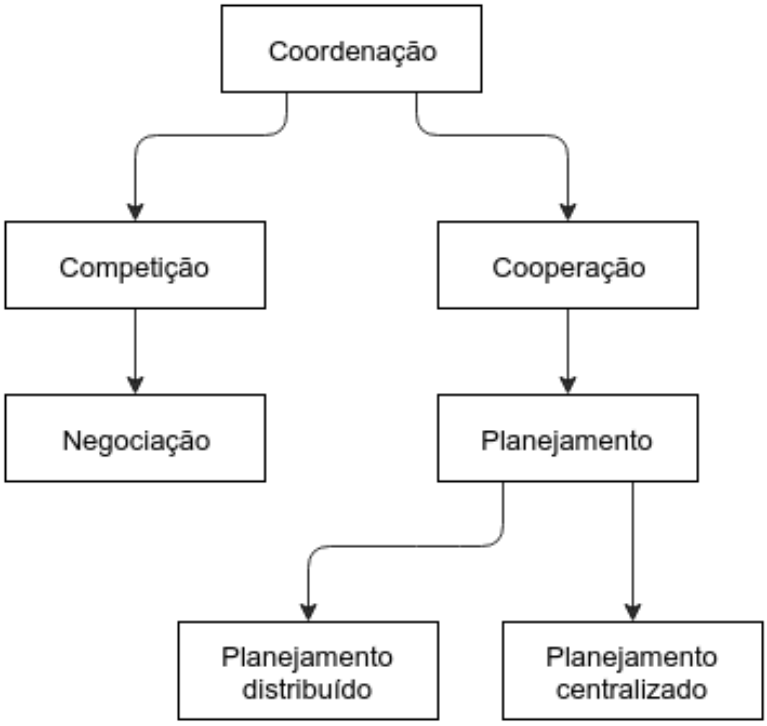

Figura 2.8: Taxonomia parcial das formas de coordenação em SMA (adaptada de Weiss, 2013).

KQML

Uma importante iniciativa para a definição de linguagens para sistemas autônomos foi conduzida no âmbito da *Knowledge Sharing Effort* (KSE) no início da década de 1990 (INTERCHANGE, 1998). Dessa iniciativa surgiu a linguagem de comunicação KQML (*Knowledge Query and Manipulation Language*). O KQML é uma linguagem de comunicação baseada em mensagens que não leva em consideração o conteúdo das mensagens trocadas pelos agentes, prescrevendo somente um formato padrão para essas mensagens. Assim como em outras linguagens de comunicação entre agentes, o KQML aplica o conceito de performativas, que são ações as ações comunicativas executadas pelo agente. Conforme apresentado em Finin et al. (1994), as mensagens KQML são constituídas por três camadas:

- Camada de conteúdo - abarca o conteúdo da mensagem sem pré-definir um formato que deve ser seguido. Esse formato é definido no âmbito de SMA que utilizam

a linguagem, podendo ser conteúdos do tipo texto ou objetos serializados em um formato binário.

- Camada de mensagem - contêm a performativa e o protocolo que serão utilizados na comunicação. Outros parâmetros podem ser incluídos como alguma descrição sobre o conteúdo da mensagem e uma definição de ontologia.
- Camada de comunicação - inclui um conjunto de parâmetros de comunicação como a identificação do emitente, do destinatário e um identificador único da mensagem.

A Figura 2.9 mostra um exemplo da estrutura de uma mensagem KQML e a Tabela 2.1 apresenta os possíveis parâmetros dessas mensagens.

```
(ask-one
  :sender joe
  :content (PRICE IBM ?price)
  :receiver stock-server
  :reply-with ibm-stock
  :language LPROLOG
  :ontology NYSE-TICKS)
```

Figura 2.9: Mensagem KQML - "joe" pergunta ao "stock-server" sobre o preço da ação da IBM. (retirado de Finin et al. (1994))

Tabela 2.1: Parâmetros de uma mensagem KQML

Parâmetro	Descrição
:content	conteúdo da mensagem.
:force	define se o remetente pode em algum tempo negar o conteúdo da mensagem.
:reply-with	define se o remetente espera uma resposta e estipula um identificador para ser utilizado nesta reposta.
:in-reply-to	referência ao parâmetro :reply-with
:sender	identificação do remetente
:receiver	identificação do destinatário

FIPA-ACL

A FIPA[8] é uma organização internacional vinculada à IEEE Computer Society[9] dedicada à promoção das tecnologias baseadas em agentes por meio do desenvolvimento de especificações para viabilizar a interoperabilidade entre agentes e aplicações baseadas em agentes (FIPA, 2002). A FIPA-ACL é a linguagem de comunicação entre agentes apresentada pela FIPA. Essa linguagem é similar ao KQML, e assim como o KQML define um formato para

[8]http://www.fipa.org/
[9]https://www.computer.org/

as mensagens contendo a identificação da performativa, do protocolo de interação e de outros parâmetros, como os apresentados na Tabela 2.2.

A Figura 2.10 mostra um exemplo da estrutura de uma mensagem FIPA-ACL.

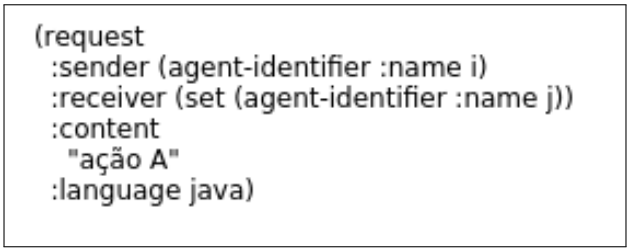

Figura 2.10: Mensagem FIPA-ACL - o agente i requisita que o agente j execute a ação A (adaptado de FIPA Communicative Act Library, 2002)

Tabela 2.2: Parâmetros de uma mensagem FIPA-ACL

Parâmetro	Descrição
performative	Tipo da ação comunicativa.
sender	Identificação do emissor da mensagem.
receiver	Identificação do destinatário da mensagem.
reply-to	Identificação do agente para o qual a resposta à mensagem deve ser direcionada.
content	Conteúdo da mensagem.
language	Definição da linguagem utilizada no conteúdo.
encoding	Definição da codificação utilizada para o conteúdo da mensagem,
ontology	Definição da ontologia a ser utilizada para interpetrar o conteúdo da mensagem.
protocol	Identifica o protocolo de interação sendo utilizado na conversação.
conversation-id	Expressão que identifica unicamente a sequência de ações comunicativas que formam uma conversação.
reply-with	Expressão que identifica a mensagem.
in-reply-to	Identificação de um ação comunicativa prévia que está sendo respondida pela presente mensagem.
reply-by	Define uma data/hora limite para que o agente emitente receba uma resposa à essa mensagem.

2.4.3 Protocolo de interação

Os agentes em um SMA interagem uns com os outros para atingir seus objetivos particulares ou para atingir um objetivo comum (Wooldridge and Jennings, 1995). A definição apresentada por Wooldridge (2009) e Wooldridge and Jennings (1995) ressalta que os agentes são entidades sociais capazes de se comunicar, visão também destacada por Weiss (2013), que coloca uma tendência para a interação entre agentes devido ao atual cenário onde existe uma crescente conexão entre dispositivos e sistemas.

Em sistemas distribuídos em geral a comunicação entre os componentes tem um viés imperativo. Em um sistema orientado à objetos que utilize RMI (*remote method invocation*), por exemplo, um objeto A invoca um método de um objeto B, fazendo com que B execute uma determinada ação. Nesse caso, a decisão de executar a ação não é tomada pelo ente executor (o objeto B) e sim pelo ente que requisita a ação (o objeto A). No caso de um SMA essa comunicação não costuma ser imperativa, ou seja, um agente A não pode simplesmente invocar um método do agente B que provocaria a execução de uma ação por este agente. Isso ocorre devido à autonomia inerente aos agentes que, partindo de seu estado atual e do conhecimento acerca do ambiente, decidem quais ações e comportamentos serão adotados (Wooldridge, 2009). Dessa forma, os agentes devem se engajar em formas de comunicação mais complexas de modo a permitir que um agente requisite informações a outros agentes ou que modifique o estado interno de outros agentes com vistas a solicitar (ou mesmo convencer) outros agentes a adotarem um determinado comportamento. Esse tipo de comunicação pode se da com base em protocolos de interação como os definidos pela FIPA em sua *Interaction Protocol Library* (FIPA-IPL). A FIPA-IPL é um conjunto de especificações de protocolos de interação para estruturar as comunicações entre agentes. Os protocolos são estruturados por meio de uma sequência de mensagens FIPA-ACL. As performativa da FIPA-ACL são apresentadas na Tabela 2.3 em conjunto com uma breve descrição sobre cada uma delas.

2.4.4 Ferramentas para o desenvolvimento de SMA

Os SMA podem ser implementados em qualquer linguagem de programação, mas existem algumas ferramentas que auxiliam no desenvolvimento de SMA por disponibilizarem os protocolos de comunicação e interação já implementados. Essas ferramentas aplicam os conceitos relacionados ao desenvolvimento de agentes e implementam arquiteturas criadas pela comunidade acadêmica e padrões definidos por organizações internacionais como a FIPA. A seguir é feita uma breve descrição acerca de algumas dessas ferramentas.

Jason (Bordini and Hübner, 2007)

O Jason é um interpretador para uma versão estendida da linguagem AgentSpeak (Rao, 1996). Ele possibilita o desenvolvimento de agentes cognitivos facilitando a implementação dos componentes do modelo *Belief-Desire-Intention* (BDI) (Bratman, 1987). O Jason permite uma fácil integração com as plataformas SACI(Hübner and Sichman, 2000) e JADE, garantindo, dessa forma, suporte ao desenvolvimento de SMA distribuídos(Bordini and Hübner, 2007).

JADE (Bellifemine et al., 2007)

O *Java Agente Development Framework* (JADE) é uma plataforma que oferece um conjunto de ferramentas de suporte ao desenvolvimento de aplicações distribuídas baseadas em agentes. O *framework* foi criado pela *Telecom Italia Labs* (TILAB) em 1998, com o objetivo inicial de validar as especificações definidas pela FIPA (Bellifemine et al., 2007). Desenvolvida completamente na linguagem JAVA e lançada como um software livre com o código aberto, o JADE evoluiu do conjunto inicial de implementações das especificações

Tabela 2.3: Performativas da FIPA-ACL

Performativas	**Descrição**
accept-proposal	O agente aceita uma proposta para executar uma ação.
agree	O agente aceita executar um determinada ação.
cancel	O agente emissor informa ao destinatário que não tem mais a intenção que o receptor execute uma determinada ação.
cfp	O agente emite uma chamada para que outros agentes apresentem propostas para realizar uma determianda ação .
confirm	O remetente informa ao destinatário que uma determinada proposição é verdadeira.
disconfirm	O remetente informa que uma determinada proposição é falsa.
failure	Um agente informa a outro que uma tentativa de realizar determinada ação falhou.
inform	O remetente enviar determinada informação ao destinatário. Essa informação é considerada verdadeira pelo emissor.
inform-if	O agente informa a outro agente se uma determinada proposição é verdadeira ou falsa.
inform-ref	É análoga ao inform-if porém o emissor envia o valor de uma expressão ou o descritor de um determinado objeto.
not-understood	Utilizado quando o agente não entendeu uma mensagem.
propagate	O remetente solicita que o receptor execute determinada ação e transmita a mensagem para outros agentes que são identificados a partir de uma descrição contida na mensagem.
propose	O remetente submete uma proposta para executar determinada ação.
proxy	O agente remetente solicita que o agente destinatário selecione outros possíveis destinatários a partir de uma descrição e transmita a mensagem a eles. Diferente do propagate, neste caso o destinatário não executa a ação solicitada na mensagem, servindo somente como um ponto de contato entre o emitente e os outros possíveis destinatários.
query-if	Remetente pergunta ao destinatário se determinada proposição é verdadeira ou falsa.
query-ref	Ação de pedir a outro agente um determinado objeto identificada a partir de uma expressão contida na mensagem.
refuse	O agente se recusa a executar uma ação e explica o motivo da recusa.
reject-proposal	O agente rejeita uma proposta para executar determinada ação.
request	O agente remetente solicita que o agente destinatário execute determinada ação.
request-when	O remetente solicita que o destinatário execute uma ação quando a expressão contida na mensagem se torne verdadeira.
request-whenever	O agente remetente solicita que o agente destinatário execute determinada ação todas as vezes em que a expressão contida na mensagem se torne verdadeira
subscribe	Ação de requisitar que o remetente seja informado pelo destinatário sempre que um determinado valor for alterado.

FIPA para um ambiente distribuído que permite que agentes sejam implantados remotamente e se comuniquem de forma transparente. Dessa forma, o desenvolvedor que utiliza o JADE não precisa se preocupar com detalhes de mais baixo nível - como a implementação de protocolos de invocação de métodos remotos - para permitir a interação entre seus agentes.

A Figura 2.11 apresenta a arquitetura da plataforma JADE. Os agentes e demais componentes do JADE são executados em *containers* distribuídos, sendo que esses *containers*, por sua vez, são executados no âmbito de uma *Platform*. Cada *Platform* possui um *Main Container* e pode possuir vários outros *containers* distribuídos. As *Platforms* já vem equipadas com um agente do tipo *Agent Management Service* (AMS), que implementa funcionalidades de configuração e manutenção da plataforma (serviço de Paginas Brancas), e um agente *Directory Facilitator* (DF)[10], que provê o serviço de Páginas Amarelas, onde os demais agentes podem se registrar e compartilhar seus serviços, e também podem encontrar outros agentes.

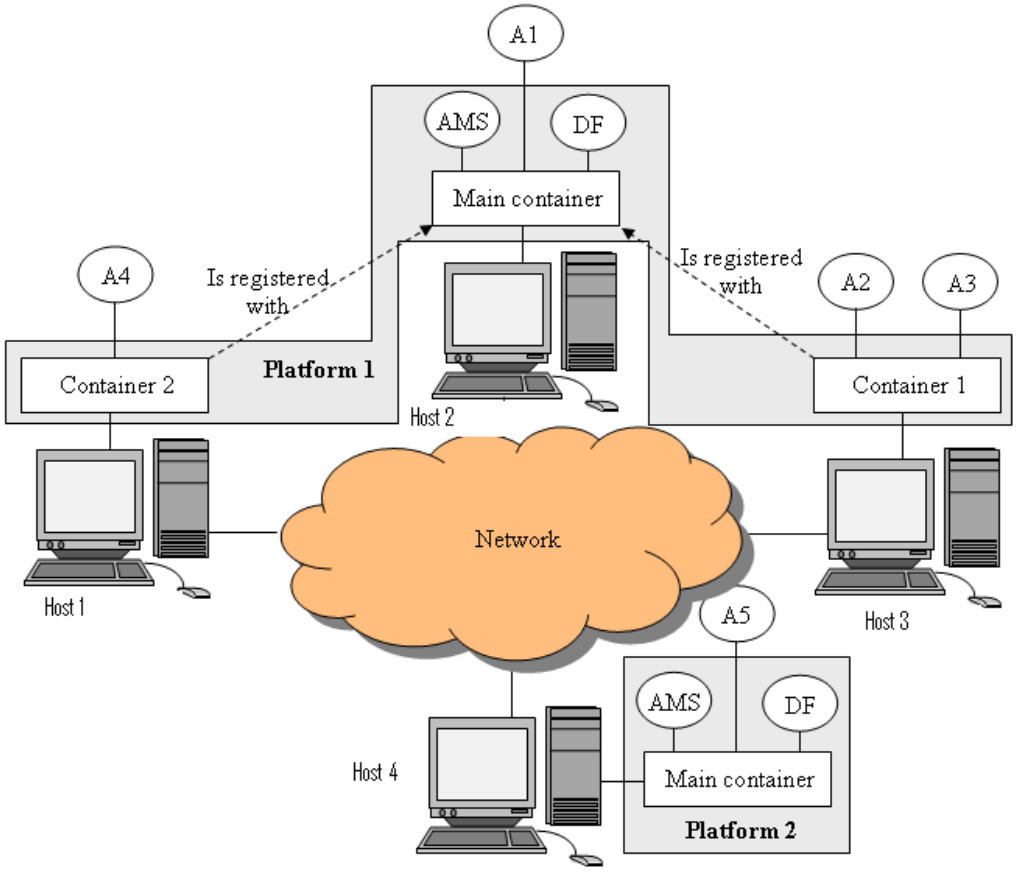

Figura 2.11: Arquitetura do JADE (Bellifemine et al., 2007).

Os agentes no JADE são implementados como uma extensão da classe *Agent*. Os comportamentos dos agentes são implementados através de várias ações relacionadas a classe *Behaviour*. O JADE conta com um conjunto de *Behaviours* pré-definidos que podem ser instanciados pelo agente, como o *SequentialBehaviour*, *ParallelBehaviour* e

[10]http://www.fipa.org/specs/fipa00023/SC00023K.html

FSMBehaviour (*Finite State Machine Behaviour*). Novos *Behaviours* podem ser criados à partir da extensão desse conjunto base fornecido pelo JADE para implementar os mais diversos tipos de ações.

JADEX (Pokahr et al., 2005)

O JADEX nasceu como uma extensão à plataforma JADE (JADE eXtension). O objetivo inicial desta ferramenta era integrar ao JADE funcionalidades que facilitassem a representação e o desenvolvimento de modelos mentais de agentes, utilizando, principalmente o modelo BDI (Braubach et al., 2003). Nesse sentido, o JADEX implementou um arquitetura abstrata e um conjunto de componentes que facilitam a implementação dos elementos da arquitetura JADEX como a base de crenças, o conjunto de planos e o conjunto de objetivos do agente.

As versões mais recents do JADEX apresentam um ambiente completo para o desenvolvimento de agentes. O *framework* aplica a conceito de *active componentes*, que podem ser tanto agentes BDI, como BPMN (*Business Process Model and Notation*) *workflows* (Braubach et al., 2004). Ademais, o JADEX conta com uma interface própria de comunicação que permite chamadas *Representational State Transfer* (REST) para o agentes, podendo ainda ser integrado ao JADE para aproveitar as funcionalidades de comunicação providas por esse.

2.5 Confiança e reputação

Confiança

O conceito de confiança permeia nosso dia-a-dia. Quando tomamos um ônibus para ir ao trabalho confiamos que o motorista é capaz de dirigi-lo. Quando depositamos dinheiro em um banco estamos confiando que o banco guardará essa quantia de maneira segura. As relações de confiança são feitas a todo momento, com maior ou menor esforço deliberativo (Marsh, 1994). Em alguns casos a decisão de confiar é tomada de maneira quase automática. Em outros casos essa decisão exige um esforço mental maior, demandando uma análise mais cuidadosa das informações disponíveis e dos resultados possíveis fruto do vínculo de confiança que será estabelecido. Como definido por Huynh (2006), "confiança é essencial para qualquer decisão que leva uma entidade a depender de outra".

Esse conceito de confiança pode ser aplicado também aos sistemas computacionais, principalmente aqueles que são compostos por um conjunto de agentes computacionais que interagem entre si. Em alguns casos, esses agentes compartilham um mesmo objetivo e atuam em cooperação para solucionar os problemas propostos, alcançando o objetivo esperado. Assim, os agentes confiam uns nos outros e acreditam que todos os agentes do sistema estão comprometidos em atingir o objetivo comum. Em outros casos, os agentes podem ter objetivos particulares diversos, algumas vezes conflitantes. Em situações desse tipo, um agente não costuma confiar à priori nos demais agentes do sistema e necessita de algum mecanismo para decidir quando um determinado parceiro é confiável e quando pode contar com a cooperação de outro agente (Huynh, 2006). Dessa forma, a confiança pode ser vista como uma expectativa acerca do resultado que será obtido da relação estabelecida entre o agente e seus parceiros.

Marsh (1994) apresenta uma visão da confiança através de três aspectos: confiança básica, confiança geral e confiança situacional. A confiança básica é construída à partir do conjunto de experiências passadas de um agente, e está relacionada à pré-disposição em confiar. Essa pré-disposição pode ter um viés otimista, quando o agente está inicialmente inclinado a confiar nos potenciais parceiros, ou um viés pessimista, quando o agente está inicialmente inclinado a não confiar. No caso da confiança geral, a pré-disposição de confiar está relacionada a um determinado parceiro, não considerando um cenário ou situação específica. Esse aspecto da confiança demonstra uma expectativa do agente em relação ao potencial parceiro, podendo essa expectativa ser positiva ou negativa. O terceiro aspecto da confiança, na visão de (Marsh, 1994), é a confiança situacional. Esse aspecto relaciona a confiança a uma situação ou cenário específico em que o agente pretende estabelecer o vinculo de confiança com os potenciais parceiros. Por exemplo, posso confiar em um programador experiente para desenvolver um programa simples de leitura de arquivos, mas não confiaria nesse mesmo programador para criar um projeto de construção de uma casa.

Sabater (2002) define confiança direta como um tipo de confiança baseado somente em interações vivenciadas diretamente pelo agente que pretende estabelecer o vínculo com os potenciais parceiros. A confiança direta, assim como a situacional de (Marsh, 1994), está ligada a um cenário ou situação específica. Porém, há a inclusão do aspecto relativo ao resultado esperado pelo agente à partir da interação com os potenciais parceiros. O resultado esperado é entendido como um contrato inicial entre os agentes, onde são fixados os termos como o resultado esperado, as condições e os aspectos que serão avaliados. Esses aspectos são definidos com base em fatores que influenciam o resultado da interação, considerando os objetivos e necessidades do agente. Por exemplo, em uma interação comercial de compra e venda, um agente poderia estabelecer o preço, a qualidade do produto e o tempo de entrega como aspectos que serão avaliados e farão parte do resultado esperado.

Reputação

A reputação é uma informação externa, composta coletivamente e formada à partir de comportamentos passados de determinado ente. Na ausência de experiências diretas costuma-se utilizar a experiência de terceiros como fonte de informação para decidir acerca da confiança em determinado ente (Sabater, 2002). As experiências de terceiros, ou mais especificamente, as avaliações que esses terceiros fazem em relação às experiências vivenciadas com um ente, são utilizadas para compor a reputação em relação a esse ente. Nesse sentido, a reputação, assim como a confiança, pode ser mensurada em um valor e pode se referir à reputação, seja de um ente específico ou de um grupo.

Jøsang et al. (2007) discute a dissociação entre confiança e reputação na medida em que um ente tende a dar mais peso para suas próprias experiências em detrimento de informações referentes às experiências de terceiros. Porém, na ausência das experiências próprias a decisão de confiar deve ser baseada em referências dadas por terceiros. Na visão do mesmo autor, a reputação é "uma medida coletiva de confiabilidade baseada nas referências ou avaliações dos membros em uma comunidade"(Hoelz, 2013).

Para Huynh (2006), um modelo de reputação define como coletar avaliações sobre um determinado agente, e como combiná-las de modo a representar, de maneira mais fiel

possível, a confiabilidade daquele agente. No modelo proposto por esse autor, a reputação é mensurada de duas formas: reputação certificada e reputação de testemunhas. Ambas lidam com as avaliações fornecidas por outros agentes em relação ao agente cuja confiabilidade está em análise. A diferença principal fica no método utilizado para coletar essas avaliações. No caso da reputação certificada, um agente B qualquer, cuja confiabilidade está sendo analisada, fornece um certificado sobre sua reputação. Esse certificado é formado por um conjunto de avaliações previamente armazenadas pelo próprio agente B. Por outro lado, na reputação de testemunhas, o agente A que está realizando a análise de confiabilidade de B faz uma busca pela rede de agentes conhecidos para angariar avaliações acerca de B.

Metamodelo de confiança e reputação

Hoelz (2013) propõe um metamodelo para adaptação de confiança e reputação (C&R) para SMA dinâmicos. A Figura 2.12 apresenta a visão geral desse metamodelo. Como pode ser observado, o metamodelo foi divido em três submodelos: confiança, reputação e exploração.

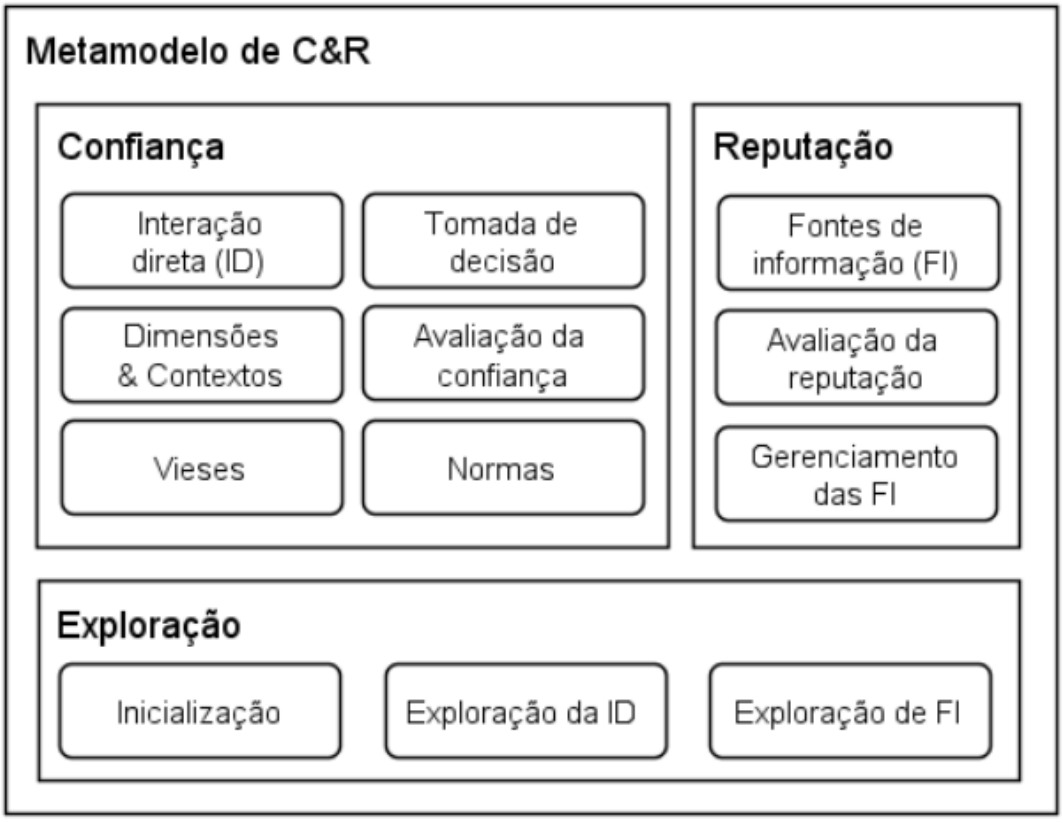

Figura 2.12: Metamodelo de C&R (Hoelz, 2013)

O modelo de confiança reúne os elementos relacionados ao processo de raciocínio utilizado como base para decisão de interagir com um potencial parceiro. Fazem parte desse modelo os elementos referentes às dimensões e contextos de confiança, à tomada de decisão de confiar, às abordagens de avaliação da confiança, às normas e vieses.

O modelo de reputação engloba os elementos relacionados à definição das fontes de informação que serão utilizadas na avaliação da reputação, a disponibilidade dessas fontes, o processo de aquisição, e as formas de compartilhamento.

O modelo de exploração contém os elementos relativos à exploração inicial do ambiente, à pesquisa por potenciais parcerias diretas, e à exploração de novas fontes de informação que podem ser utilizadas para a confiança e para a reputação.

No trabalho de Hoelz (2013) é apresentado um mapeamento dos modelos MARSH (Marsh, 1994), SPORAS (Zacharia and Maes, 2000) e FIRE (Huynh, 2006) para o metamodelo de C&R. A Figura 2.13 apresenta o mapeamento do modelo FIRE (Huynh, 2006) para o metamodelo.

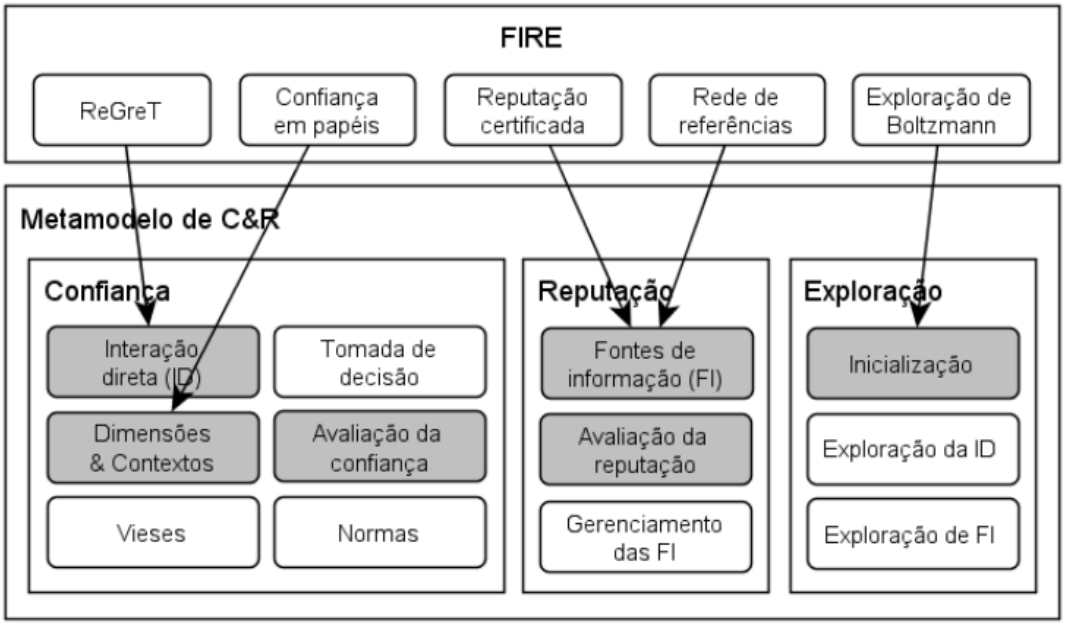

Figura 2.13: Mapeamento do modelo FIRE para o metamodelo de Hoelz (2013)

A seguir serão apresentados os pontos de mapeamento do modelo FIRE para o metamodelo de Hoelz (2013). Os pontos do modelo selecionados para serem apresentados são aqueles que serão utilizados durante a definição do modelo que é parte da proposta deste trabalho e que será apresentado na Capítulo 4. As equações e fórmulas apresentadas nestes itens foram retiradas de Huynh (2006), trabalho onde o modelo FIRE foi proposto. Alguns pontos do modelo ReGreT (Sabater, 2002) são discutidos nestes itens por esse modelo ter influenciado a definição de alguns dos componentes do modelo FIRE, existindo, desse modo, uma relação próxima entre conceitos apresentados em ambos os modelos.

Interação direta

O modelo FIRE adota o componente de interação direta do modelo ReGreT (Sabater, 2002). Esse componente utiliza as avaliações, fruto de interações passadas vivenciadas

pelo próprio agente, para realizar o cálculo da confiança direta em relação aos potenciais parceiros. O modelo FIRE utiliza a Equação 2.2 para a realização desse cálculo:

$$\mathcal{T}_I(a,b,c) = \frac{\sum_{r_i \in R_I(a,b,c)} \omega_I(r_i) * v_i}{\sum_{r_i \in R_I(a,b,c)} \omega_I(r_i)}, \tag{2.2}$$

onde $\mathcal{T}_I(a,b,c)$ representa o valor da confiança direta que o agente a tem em relação ao agente b em relação ao termo c (que representa um parâmetro relevante para medir o resultado esperado pela interação entre os agentes, como, por exemplo, um valor de QoS). $R_I(a,b,c)$ é o conjunto de avaliações fruto de interação direta entre o agente a e o agente b, $\omega_I(r_i)$ é uma função que define o peso de uma determinada avaliação para o valor de confiança direta, e v_i é o valor da avaliação r_i. A função de peso será melhor apresentada no item referente à fonte de informações desta seção.

Vieses

Os agentes tem crenças e preferências internas acerca de características de outros agentes. Essas crenças e preferências são constituídas a partir de outras experiências vivenciadas pelo agente ou de regras pré-existentes definidas para o ambiente. Essas preferências e regras, denominadas vieses em Hoelz (2013), podem influenciar o agente em relação à sua avaliação acerca da confiança de seus potenciais parceiros. O modelo ReGreT (Sabater, 2002) utiliza esse conceito de vieses para formular os componentes de reputação de grupo e de reputação do sistema que integram seu modelo de reputação.

No modelo FIRE (Huynh, 2006) os vieses são aplicados pelo componente de reputação baseada em papéis que integra as crenças e preferências do agente em papeis que são designados para os os potenciais parceiros que estão sendo avaliados. Os papeis podem refletir algum tipo de relação hierárquica entre os agentes (no contexto de uma empresa por exemplo), ou alguma preferência do agente avaliador em relação ao grupo ao qual o parceiro sendo avaliado pertence. Esse último caso reflete o mesmo conceito utilizado pelo componente de reputação de grupo do modelo ReGreT.

Para o cálculo da reputação baseada em papéis o agente utiliza sua base de regras $R_R(a,b,c)$ formada por tuplas com o formato: $rul = (role_a, role_b, c, e, v)$. As tuplas contém os valores que refletem os vieses do agente a em relação ao agente b. Os valores $role_a$ e $role_b$ referem-se aos papeis do agente a e b, respectivamente, enquanto os valores e e v representam, respectivamente, o nível de influência deste papel para a reputação e o valor esperado de performance para a interação com o agente b. Como na interação direta, o valor c representa um parâmetro para medir o resultado esperado pela interação entre os agentes. O cálculo de valor final para esse componente utiliza a Equação 2.2, com a função de peso sendo calculada como: $\omega_R(r_i) = e_i$.

Fontes de informação

Além da interação direta, o modelo FIRE utiliza a reputação certificada e a informação de testemunhas como fontes de informação. O componente de reputação de testemunha do FIRE é responsável por reunir as informações das testemunhas, coletadas por meio de uma rede de referências. Para buscar a informação nessa rede um agente a, que necessita de informações sobre um agente b, envia uma requisição para n_BF agentes conhecidos.

Os agentes conhecidos que receberam a requisição verificam se possuem as informações requisitadas. Em caso positivo a informação é enviada para o agente a, caso contrário é retornada uma lista contendo outros agentes que podem ser consultados. Para que a consulta não se estenda indefinidamente pela rede de referências o agente a define o valor n_{RL} que estipula um limite para a pesquisa.

Figura 2.14: Metamodelo de uma fonte de informação (Hoelz, 2013)

No componente de reputação certificada, por outro lado, o agente b armazenam em sua base local as avaliações acerca de sua *performance* quando da interação com outros agentes. Essas avaliações formam um tipo de certificado que pode atestar a capacidade que o agente tem em atingir determinada *performance* em uma interação futura. Para calcular o valor da reputação certificada a Equação 2.2 é empregada, porém é aplicada uma função de peso $\omega_W(r_i)$ diferente da utilizada para a confiança direta. A função $\omega_W(r_i)$ será apresentada no item de mapeamento referente a credibilidade e confiabilidade.

Credibilidade e confiabilidade

O número de interações entre um agente e um potencial parceiro pode afetar a confiabilidade que o agente tem em relação a esse potencial parceiro. Ou seja, se um agente a confia igualmente em dois outros agentes b e c ele pode optar por escolher aquele com o qual interagiu mais vezes no passado, por julgar que uma quantidade maior de interações resulta em uma avaliação mais acurada acerca do agente. Outro fator que pode influenciar a confiabilidade é o grau de variação entre os desempenhos obtidos em cada interação com um determinado parceiro. Um agente que apresenta um nível diferente de performance a cada interação pode ser visto como instável, dificultando a avaliação acerca da performance esperada em uma interação futura. O modelo FIRE emprega os dois conceitos no seu cálculo de confiabilidade, sendo a Equação 2.3 utilizada para calcular a influência do número de interações, e a Equação 2.4 utilizada para calcular a influência da variabilidade entre a *performance* obtida de cada interação.

$$\rho_{RK}(a,b,c) = 1 - e^{-\gamma_K \cdot (\sum_{r_i \in R_k(a,b,c)} \omega_k(r_i))} \tag{2.3}$$

$$\rho_{DK}(a,b,c) = 1 - \frac{1}{2} \cdot \frac{\sum_{r_i \in R_k(a,b,c)} \omega_k(r_i) \cdot |v_i - \mathcal{T}_K(a,b,c)|}{\sum_{r_i \in R_k(a,b,c)} \omega_k(r_i)} \tag{2.4}$$

Na Equação 2.3, γ_K é utilizado como um fator de ajuste definido de acordo com a função de peso de cada componente do modelo FIRE. A Equação 2.4 calcula a diferença entre os valores v_i das avaliações contidas no conjunto $R_k(a, b, c)$ e o valor de confiança esperado. A Equação 2.5 calcula o valor final da confiabilidade baseada nos valores calculados pelas Equações 2.3 e 2.4.

$$\rho_K(a, b, c) = \rho_{RK}(a, b, c) \cdot \rho_{DK}(a, b, c) \tag{2.5}$$

O modelo FIRE calcula a credibilidade das fontes de referência para o componente de reputação certificada e das testemunhas para o componente de reputação de testemunhas. O método utilizado para o cálculo em ambos os casos é similar, e, dessa forma, será referenciado aqui somente a credibilidade relativa à reputação certificada.

A credibilidade de cada fonte de referência é calculada com base na diferença entre os valores das avaliações enviadas por essa fonte, acerca do agente cuja reputação está sendo calculada, e os valores das avaliações da base local do agente que está realizando o cálculo. Assim, o agente a, que realiza o calculo de credibilidade, define um valor de avaliação para cada referência recebida e utiliza esse valor de avaliação para calcular a credibilidade da fonte de referência. A Equação 2.6 realiza o cálculo do valor de avaliação de credibilidade da fonte de referência:

$$v_w = \begin{cases} 1 - |v_k - v_a| & \text{se } |v_k - v_a| < \iota \\ -1 & \text{se } |v_k - v_a| \geq \iota \end{cases}, \tag{2.6}$$

onde ι representa um limiar de inacurácia definido por cada agente que realiza o calculo de credibilidade. Os valores v_k, v_a representam o valor da avaliação da referência e o valor da avaliação definida pelo próprio agente a. Como os valores v_k, v_a estão contidos no período $[-1, 1]$ o valor do limiar de inacurácia ficará compreendido entre 0 e 2 ($0 \leq \iota \leq 2$).

A credibilidade é definida como a confiança direta que o agente a tem em relação às fontes de referência com base nas avaliações calculadas pela Equação 2.6. Caso o agente a não possua nenhuma avaliação prévia para ser comparada com os valores das avaliações das referências, um valor padrão de credibilidade é definido. A Equação 2.7 calcula a credibilidade da fonte de referência com utilizando a confiança direta ($T_I(a, w, term_{RCr})$):

$$T_{WCr}(a, w) = \begin{cases} T_I(a, w, term_{WCr}) & \text{se } R_I(a, w, term_{WCr}) \neq \emptyset \\ T_{DWCr} & \text{se } R_I(a, w, term_{WCr}) = \emptyset \end{cases}, \tag{2.7}$$

onde T_{DRCr} é o valor padrão de credibilidade e $term_{WCr}$ denota que a avaliação é relativa à credibilidade da fonte de referência.

O FIRE utiliza a credibilidade da fonte de referência na definição da função de peso $\omega_W(r_i)$ utilizada no cálculo da reputação certificada. A função de peso $\omega_W(r_i)$ é definida conforme a Equação 2.8.

$$\omega_W(r_i) = \begin{cases} 0 & \text{se } T_{WCr}(a, w) \leq 0 \\ T_{WCr} \cdot \omega_I(r_i) & \text{se } T_{WCr}(a, w) > 0 \end{cases} \tag{2.8}$$

Dessa forma, no cálculo da reputação certificada, realizado no âmbito do modelo FIRE, a credibilidade da fonte de referência irá influenciar no peso que determinada avaliação tem para o valor final da reputação certificada. No caso de uma credibilidade negativa,

a avaliação obtida à partir da fonte de referência é ignorada, recebendo um peso igual a zero.

Memória e recência

Em um cenário ideal o agente poderia armazenar as avaliações sobre todas as suas interações passadas. Porém, na maioria dos casos, o agente não contará com capacidade disponível para armazenar uma grande quantidade de avaliações. Nesse caso, deve-se definir qual a extensão do histórico de avaliações do agente. O modelo FIRE não estabelece um método de definição da extensão do histórico do agente, mas aplica uma abordagem similar ao modelo ReGreT (Sabater, 2002) utilizando uma função que ajusta o peso das avaliações de acordo com a recência. Ou seja, avaliações mais recentes tem um maior peso que avaliações que foram colhidas em interações passadas. Essa abordagem é aplicada no modelo através da função de peso da confiança direta, que segue definida na Equação 2.9:

$$\omega_I(r_i) = e^{-\frac{\triangle t(r_i)}{\lambda}}\,, \tag{2.9}$$

onde o fator de escala de recência λ ajusta a granularidade temporal do ambiente em que o modelo está sendo aplicado.

Avaliação da reputação

No modelo FIRE a avaliação da reputação é feita com base na equação:

$$\mathcal{T}_K(a,b,c) = \frac{\sum_{r_i \in R_K(a,b,c)} \omega_W(r_i) * v_i}{\sum_{r_i \in R_K(a,b,c)} \omega_W(r_i)}\,, \tag{2.10}$$

onde $\mathcal{T}_K(a,b,c)$ representa o valor da reputação calculado, podendo ser a reputação certificada ou de testemunhas. A função de peso $\omega_W(r_i)$ é calculada com base na credibilidade das fontes de informações, conforme apresentado na Equação 2.8.

Avaliação da confiança

O modelo FIRE combina os valores dos seus componentes de C&R em um valor final de confiança calculado por meio da Equação 2.11:

$$\mathcal{T}(a,b,c) = \frac{\sum_{K \in \{I,R,W,C\}} \omega_K * \mathcal{T}_K(a,b,c)}{\sum_{K \in \{I,R,W,C\}} \omega_K}\,, \tag{2.11}$$

onde $\mathcal{T}_K(a,b,c)$ é o valor de cada componente do modelo.

Segundo Hoelz (2013) o modelo FIRE não apresenta uma definição clara de como um agente toma a decisão de confiar em outro. Entretanto, o modelo apresenta um cálculo para credibilidade geral do valor final de confiança, que pode ser utilizada para balizar a decisão do agente. O cálculo de credibilidade é realizado utilizando a Equação 2.12.

$$\rho_T(a,b,c) = \frac{\sum_{K \in \{I,R,W,C\}} \omega_K}{\sum_{K \in \{I,R,W,C\}} W_K} \tag{2.12}$$

Um agente orientado a utilidade pode aplicar o valor final de confiança calculado pela Equação 2.11 e o valor de credibilidade calculada pela Equação 2.12 para escolher ações de modo a maximizar a utilidade esperada. A confiança em um parceiro pode influenciar a visão do agente acerca da probabilidade relacionada à ação de interagir com o potencial parceiro.

No Capítulo 3 serão apresentados trabalhos que utilizam o modelo de fog computacional em diferentes cenários de aplicação.

Capítulo 3

Trabalhos correlatos

Nesta seção serão apresentados cinco trabalhos que baseiam suas propostas no modelo de fog computacional. Esses trabalhos foram selecionados a partir de pesquisas realizadas nas bases de artigos disponibilizadas no portal de periódicos da CAPES[1], e na pesquisa de artigos disponibilizada pelo *Google Scholar*[2]. Foram selecionados quinze artigos apresentados em conferências internacionais e publicados em periódicos da IEEE (*Institute of Electrical and Electronics Engineers*). A seleção dos artigos que serão apresentados neste capítulo levou em consideração: (i) o período de publicação, considerando os trabalhos publicados entre 2016 e 2017; (ii) a correlação com o contexto de fog computacional; e (iii) as ferramentas e ambientes utilizados para a avaliação das propostas apresentadas. A ideia com esse último ponto era a de selecionar artigos que utilizassem diferentes ferramentas aderentes ao modelo de fog para validar suas propostas. Um outro ponto considerado para a seleção destes trabalhos foi a diversificação das propostas baseadas no modelo de fog computacional, de modo a ressaltar a abrangência desse modelo e a diversidade de aplicações possíveis para a fog, refletindo os cenários de aplicação apresentados na Seção 2.2.2.

Lin and Shen (2017)

Lin and Shen (2017) apresentam o CloudFog, uma abordagem de fog computacional para o contexto de *cloud gaming*, mais especificamente, para os jogos do tipo MMOG. Os autores propõem uma infraestrutura que utiliza as máquinas dos usuários como nós locais de processamento. Esses nós, identificados como *supernodes*, são responsáveis por renderizar os vídeos do jogo online servindo como *hosts* locais para outros jogadores. Nessa infraestrutura, os usuários enviam os dados de interação com o jogo (como movimentação do jogador, ações de tiro, etc.) para o servidor na nuvem, que realiza o cálculo do estado geral do jogo. O servidor de nuvem, por sua vez, envia os parâmetros de renderização para os *supernodes*, que renderizam o vídeo do jogo e enviam como resposta para os usuários. Os principais objetivos com essa abordagem de fog são: (i) oferecer uma melhor QoS; (ii) aumentar a cobertura de usuário; (iii) reduzir a latência da resposta; e (iv) reduzir os custos com escalabilidade da infraestrutura.

[1] http://www.periodicos.capes.gov.br/
[2] https://scholar.google.com.br/

O CloudFog faz uso de um sistema de provisionamento dinâmico para prever o número de jogadores e provisionar *supernodes* para atender à demanda de serviço. Além disso, utiliza uma abordagem de rede social para atribuir grupos de jogadores a um mesmo servidor, evitando o *overhead* de comunicação entre os servidores. Os nós usuários selecionam os *supernodes* com base em um modelo de reputação. Após cada jogo, o jogador avalia seu *supernode* e atribui uma pontuação de desempenho baseada na QoS. Essas pontuações são usadas na avaliação da reputação. Cada jogador usa apenas suas próprias pontuações para fazer o cálculo da reputação, não levando em consideração a opinião de outros jogadores sobre os *supernodes*. Um sistema de recompensa é implementado para incentivar os jogadores a se tornarem *supernodes*.

O CloudFog foi avaliado utilizando o simulador PeerSim (Montresor and Jelasity, 2009b) e o ambiente real PlanetLab (Chun et al., 2003). Em ambas avaliações, a latência, a cobertura de usuário e o custo de implantação da proposta são testados. Em ambos os cenários são realizados testes utilizando de 10000 a 60000 nós no simulador Peersim e 750 nós no PlanetLab, variando o número de *supernodes* de 400 a 600 no Peersim e 300 no PlanetLab. Os nós interagem em ciclos que representam um dia de jogo. Em cada ciclo a dinâmica de entrada e saída de usuários no sistema segue um padrão geral de carga em servidores de jogos *online*, com mais usuários conectados em horários considerados de pico (20h às 24h). Os experimentos mostram que a latência é reduzida com a abordagem de fog computacional, atingindo 14% da latência do modelo tradicional centralizado na nuvem. Do ponto de vista da cobertura de usuários, os testes mostram que a implantação de 100 *supernodes* pode aumentar a cobertura de usuários de 25% para 65%, e aumentando esse número para 200 *supernodes* gera o mesmo resultado que a implantação de 25 *datacenters* de nuvem. Em relação aos custos e benefícios econômicos para o provedor de serviço, o trabalho apresenta uma economia de 30 dólares por dia quando comparado com o custo de se alugar um servidor EC2 na Amazon.

He et al. (2017)

Seguindo a abordagem de uso de máquinas usuárias como nós de fog computacional para processamento local, He et al. (2017) propõem uma estrutura para a realização de transcodificação de vídeo para plataformas de *crowdsourced livecast service* (CLS). Essas plataformas oferecem serviços de *streaming* ao vivo e são consumidos diariamente por milhões de usuários ao redor do mundo. A heterogeneidade dos dispositivos e da infraestrutura de rede utilizada pelos usuários impõe sérios desafios para os provedores desse tipo de serviço. Para atender aos requisitos dos diversos usuários, os provedores devem transcodificar os conteúdos de vídeo gerados pelos produtores de conteúdo em versões com diferentes níveis de qualidade. Outro problema enfrentado por provedores de CLS é a sincronização entre diferentes usuários. Durante uma transmissão ao vivo, os usuários podem interagir uns com os outros por mensagens de texto, fazendo comentários sobre o conteúdo que está sendo transmitido. Isso significa que um grupo de usuários que assiste ao mesmo *streaming* não pode ter uma grande diferença em seu *delay* ou não poderão interagir corretamente.

Os autores apresentam uma estrutura que distribui o esforço de transcodificação entre usuários selecionados. Esses usuários atuam como nós de fog computacional transcodificando conteúdo para o provedor de CLS. O *framework* proposto pelos autores apresenta

uma arquitetura dividida em regiões, sendo cada região mantida por um *data center* local. Esses *data centers* locais, ou regionais, são responsáveis por selecionar usuários para atuar como nós de transcodificação, por captar conteúdo produzido por usuários que estão localizados na sua área de atuação e por distribuir os conteúdos que já passaram pelo processo de transcodificação. O *framework* não aplica um modelo de reputação, selecionando os usuários com base em uma medida de estabilidade de cada nó. Essa medida considera o histórico do usuário na plataforma e o tempo de permanência em uma transmissão.

Para o estudo experimental, He et al. (2017) também faz uso do ambiente do PlanetLab, definindo 208 nós como usuários e 5 nós como servidores espalhados em 5 regiões da Europa, Ásia e América do Norte. No primeiro experimento, os nós usuários selecionados devem transcodificar um vídeo 1080p de 3.5 Mb/s em versões de menor qualidade. Os resultados dos testes realizados em um servidor regional localizado na América do Norte mostram um *delay* menor que 10 segundos, o que significa que a variância percebida pelos espectadores finais é mitigada. Os resultados desse teste também demonstram uma grande variação no *delay* em diferentes regiões. O segundo experimento é conduzido em um conjunto de dispositivos populares (Intel i7, i3, i5, Core 2 Duo e AMD a10). Esse experimento visa avaliar a capacidade de dispositivos populares modernos em transcodificar *streaming* de vídeo em tempo real. Os resultados mostram que quase todos os dispositivos puderam lidar com versões de qualidade iguais e inferiores a 480p. Para versões com qualidade 720p, apenas os i7 e i5 da Intel conseguiram manter um bom desempenho. Isso leva à conclusão de que as CPU's de alguns computadores pessoais modernos podem lidar com a tarefa de transcodificação sem prejudicar a experiência do usuário.

Skarlat et al. (2016)

Skarlat et al. (2016) apresenta uma proposta de *framework* para provisionamento e orquestração de recursos em um ambiente de fog computacional. O trabalho direciona o foco da proposta para o contexto de aplicações de IoT. Três elementos básicos compõem o *framework*: *cloud-fog control middleware*, *fog orchestration control node* e *fog cell*. O *cloud-fog control middleware* é responsável por manter a visão geral da estrutura da fog. Esse elemento provisiona os recursos na cloud para as tarefas que não são sensíveis à variação de delay ou que não podem ser executadas pelos nós na fog computacional. Ademais, o *cloud-fog control middleware* pode realizar alterações para otimizar a estrutura da fog. O segundo elemento do *framework*, *fog orchestration control node*, tem como atribuição principal suportar um conjunto de *fog cells* orquestrando as atividades desses nós e, em alguns casos, enviando tarefas para o *cloud-fog control middleware*. A *fog cell*, por sua vez, representam os nós locais de processamento e ficam responsáveis por controlar e monitorar os dispositivos de IoT conectados ao *framework*, recebendo e reenviando as tarefas enviadas por esses, além de analisar e alocar parte de seus recursos computacionais para o atendimento dessas mesmas tarefas. Cada *fog cell* disponibiliza um conjunto de serviços que podem ser utilizados para atender às tarefas enviadas pelos dispositivos. O conjunto formado pelo *fog orchestration control node* e as *fog cells* é identificado como uma *fog colony*.

O fluxo de execução do *framework* é concentrado nas fog colonies. O *fog orchestration control node* recebe as requisições de tarefas e produz um plano de provisionamento de recursos para a *fog colony*, definindo quais *fog cells* devem ser utilizadas para a execução

dessas tarefas. Para definir esse plano, o *fog orchestration control node* considera os recursos disponíveis em cada fog cell, como RAM, CPU e largura de banda. O *delay* de comunicação entre os elementos também é levado em consideração nesse cálculo. O objetivo desse processo de orquestração é balancear as tarefas de maneira a utilizar o máximo da capacidade disponibilizada pelas *fog cells*, além de garantir a redução do delay de resposta para as tarefas requisitadas. Quando não existem recursos suficientes para atender um plano de provisionamento para um conjunto de tarefas, o *fog orchestration control node* separa as tarefas que não podem ser atendidas e as redireciona para outras *fog colonies*.

Os autores utilizam o simulador CloudSim para realizar os experimentos. Uma *fog colony* é definida com 100 *fog cells* associadas. Desse conjunto, 10 *fog cells* emitem simultaneamente 1000 tarefas que são enviadas para o *fog orchestration control node*. No cenário base de avaliação os autores utilizam as políticas de provisionamento que fazem parte do pacote do simulador CloudSim. Em um segundo cenário todas as tarefas são enviadas para o *cloud-fog control middleware* para serem executadas na nuvem. Por fim, o cenário otimizado utiliza toda a capacidade de orquestração do *framework*. Nesse último caso, as políticas de provisionamento do CloudSim foram extendidas e adaptadas utilizando a abordagem proposta no âmbito do *framework*, resultando na definição de quatro políticas baseadas em divisão de tempo e espaço.

Os resultados apresentam uma redução de 39% de *delay* no cenário otimizado - que combina o provisionamento com divisão de tempo para as *fog cells* e de divisão de espaço para os serviços oferecidos para o atendimento de tarefas - em comparação com o cenário base. Quando comparado com o cenário onde as tarefas são delegadas para a nuvem, notou-se uma redução de 94% do delay.

Alam et al. (2016)

Alam et al. (2016) apresenta uma abordagem multiagente baseada no modelo de fog computacional para o contexto de *code offloading*. O *code offloading* é uma técnica que se baseia em servidores remotos para executar partes de código submetidas pelos dispositivos móveis. A aplicação dessa técnica envolve a identificação, particionamento e migração de partes do código que serão executados remotamente. A literatura conta com diversas propostas que aplicam a técnica de *code offloading* em dispositivos móveis. Essas propostas focam principalmente no uso da nuvem como base para a execução remota das porções de código que são migradas dos dispositivos móveis, utilizando, desse modo, um modelo de *mobile cloud computing*. A proposta de Alam et al. (2016) procura aplicar o modelo de fog computacional à esse contexto e dessa forma estender o modelo de *mobile cloud computing* para o uso de nós de fog como pontos de execução remota para o *code offloading*.

Seguindo o caminho utilizado por Varghese et al. (2017), Iotti et al. (2017), Bittencourt et al. (2017) e Alam et al. (2016), a abordagem proposta por Alam et al. (2016) utiliza como nós de fog os pontos de acesso e as controladoras *wi-fi*, e as estações de roteamento e de controle de tráfego da infraestrutura *Long Term Evolution* (LTE). A proposta dos autores apresenta um *framework* que fica embarcado em *smartphones* que é utilizado para realizar o *code offloading*. O *framework* é responsável por analisar o código das aplicações que serão executadas pelo *smartphone* e separar o código em blocos básicos formando um

fluxograma de execução de modo a identificar dependências entre esses blocos. Partindo dessa análise são selecionados blocos candidatos ao *code offloading* e é tomada a decisão de quais blocos devem ser executados remotamente.

A abordagem proposta divide as controladoras e pontos de acesso *wi-fi* em grupos denominados pelos autores de *mobile fogs*. Os agentes apresentados na proposta ficam responsáveis por selecionar quais nós de fog serão utilizados e por conduzir o *code offloading* (enviando o código para os nós de fog). As estações LTE executam nós de fog responsáveis por coordenar a comunicação entre os grupos. Para selecionar os nós de fog que serão utilizados para executar o *code offloading* os agentes realizam uma comparação entre as capacidades dos grupos de nós de fog. Essa capacidade é calculada com base no tempo de resposta para a execução dos blocos de código que é estimado com base em o modelo de fila M/M/1/K. O cálculo considera o número máximo de blocos que o nó pode processar (K), a taxa de envio de blocos, o tempo de atendimento e a probabilidade do nó estar ocupado processando um bloco. Este tempo é estimado tanto para um grupo específico, quanto para um grupo vizinho. Assim os agentes podem fazer uma comparação entre os valores e decidir qual a melhor opção entre enviar os blocos para a *mobile fog* mais próxima ao dispositivo final, dividir os blocos entre duas *mobile fogs* vizinhas ou enviar o bloco para ser processado na nuvem computacional. Os agentes podem formar grupos de nós em uma *mobile fog*. A formação desses grupos também utiliza a métrica de tempo de processamento, e busca reunir em um mesmo grupo os nós com tempo de processamento compatíveis. A proposta dos autores também aplica um método de recompensa para os nós de fog que irão executar o *code offloading*. Essa recompensa é calculada a partir do tempo de processamento estimado e pode ter seu valor acrescido se o SLA for cumprido e diminuída caso os requisitos do SLA sejam descumpridos.

A proposta é avaliada em um estudo de caso utilizando o simulador OMNeT++. Na topologia criada para o estudo de caso são definidas duas *mobile fogs* cada uma com 4 nós de fog (uma controladora e 3 pontos de acesso wi-fi). Uma implementação do problema N-queens é utilizada como base para o *code offloading*. Além dos nós de fog, são definidos na simulação 30 *smartphones* como dispositivos finais e 8 servidores na nuvem. São testados 3 cenários, o primeiro com a execução do N-queens localmente nos *smartphones*, um segundo com o uso exclusivo dos servidores na nuvem e um terceiro com o uso da fog computacional. Foram analisados o consumo de energia e o tempo de resposta para a execução dos 3 cenários. O primeiro cenário, com o a execução local no *smartphone*, é o que tem pior tempo de resposta em todos os ciclos de teste (com 5, 6, 7 e 8 rainhas no tabuleiro). A diferença de tempo supera 50% em relação aos cenários com o uso da nuvem e da fog. A diferença entre os outros dois, fog e nuvem, é pequena tornando-se menor em compasso com o aumento do número de rainhas a serem dispostas pelo N-queens. O mesmo ocorre para os dados referentes ao consumo de energia. Porém, apesar de pequena a diferença, a fog apresenta sempre valores menores para tempo de resposta e consumo energético (principalmente para os testes com 6 e 7 rainhas no N-queens).

Bittencourt et al. (2017)

No trabalho de Bittencourt et al. (2017) é apresentada uma visão hierarquizada da fog computacional que coloca o foco da discussão acerca do modelo de fog na demanda por processamento e na mobilidade geográfica dos dispositivos finais. Na visão apresentada

pelos autores, as características relativas à mobilidade e ao constante deslocamento de diversos dispositivos como *smartphones*, por exemplo, impõem um desafio para a alocação de recursos no ambiente de fog computacional. A implantação de nós de fog em uma determinada localidade deve levar em conta a demanda transiente e variações nesta em relação ao padrão de deslocamento dos dispositivos finais e a periodicidade deste deslocamento. O autores utilizam o conceito de *Cloudlets* para referenciar os nós de fog que realizam o processamento na fronteira da rede. Um *Cloudlet* é uma versão reduzida de um servidor de nuvem que apresentam capacidade de processamento e armazenamento condizentes com o dispositivo que o executa. Uma característica dos *Cloudlets* que pode ser destacada é sua mobilidade que permite que eles sejam implantados, atualizados e realocados em diferentes dispositivos de forma ágil.

A proposta dos autores é voltada para a alocação de módulos das aplicações móveis para serem executados por *Cloudlets* em nós de fog. Para tanto, são propostas três estratégias de escalonamento para o atendimento das requisições enviadas à esses módulos quando estão implantados nas *Cloudlets*, com vistas a otimizar a utilização dos recursos providos na fog e garantir a QoS. A primeira estratégia é denominada *Concurrent* e aloca as requisições em um *Cloudlet* sem verificar se há capacidade disponível. A segunda estratégia, *First Come-First Served* (FCFS), aloca os recursos de acordo com a ordem que as requisições chegam aos módulos na *Cloudlet* (para este fim os autores consideram somente a capacidade disponível de cpu). Por fim, a estratégia *Delay-priority*, faz o escalonamento das requisições de acordo com o *delay* requisitado pela aplicação dando-se prioridade àquelas que requisitam menores *delays*.

Essas estratégias implementam um mecanismo de fusão de módulos, onde os módulos de uma mesma aplicação são agrupados no mesmo dispositivo, e, dessa forma, quando um módulo de um determinado tipo em uma aplicação for movido para uma *Cloudlet*, os demais módulos do mesmo tipo também serão movidos para o mesmo destino. Além disso, esse mecanismo trata a fog por meio de uma visão hierarquizada, e movimenta os módulos dos nós de fog da fronteira (que estaria na base da hierarquia) em direção ao servidor de nuvem (que estaria no topo da hierarquia).

São identificadas dois tipos de aplicações: tempo-real e tolerantes à *delay*. Para cada um deles é proposto um cenário de avaliação diferente. Para avaliar as aplicações que executam em tempo real, os autores propõem um cenário de um jogo que utiliza sensores ligados à cabeça do usuário que coletam dados referentes à sua atividade cerebral (são sensores de *electroencephalography* (EEG), como o *headset* MINDO-4S[3]). O objetivo do jogo é mover objetos virtuais utilizando a mente. Para tanto, a partir dos dados coletados pelos sensores, é calculada a concentração empregada por um usuário para mover um determinado objeto. Aquele usuário que conseguir se concentrar mais em um determinado objeto conseguirá atraí-lo para mais perto de si. O fluxo de processamento dessa aplicação segue o fluxo básico de sistemas IoT: captura de dados por sensores -> processamento -> envio de controle para os atuadores. O atuador neste caso é a interface gráfica apresentada pelo *smartphone*. O processamento é realizado remotamente, e é realizado em dois módulos, um que calcula a concentração do usuário em relação ao objeto e o outro que coordena a interação entre os usuários no jogo definindo quem conseguirá atrair mais o objeto.

[3]http://mindo.com.tw

Para testar aplicações tolerantes a *delay* os autores propõem um sistema vigilância e rastreamento por vídeo (*video surveillance/object tracking application* - VSOT). Nesse sistema, as imagens capturadas por uma câmera de vigilância são tratadas por um módulo de reconhecimento de movimento, que fica integrado à própria câmera, e caso algum movimento seja detectado os dados de vídeo são enviados para um módulo de detecção de objetos, que identifica os objetos, e envia os dados para um terceiro módulo que gera os comandos para controlar a câmera de modo a permitir que ela acompanhe a movimentação dos objetos. Esses dois últimos módulos do sistema são externos à câmera e executados remotamente.

A avaliação das estratégias de escalonamento é feita utilizando o simulador iFogSim (Gupta et al., 2016). É analisada a performance das aplicações em relação ao *delay* e ao tráfego de rede. São definidas 3 *Cloudlets*, 4 aplicações VSOT e 12 usuários que utilizam o aplicativo EEG. Os usuários são alocados nas *Cloudlets* 1 e 3 e são gradualmente migrados para a *Cloudlet* 2, onde as estratégias de escalonamento são aplicadas. A primeira medida realizada leva em consideração o *delay* do fluxo de controle das aplicações que seria medido entre a captura dos dados pelos sensores e o retorno dos comandos para o módulo do jogo EEG no *smartphone* ou para a movimentação da câmera no sistema VSOT. As medições consideram o número de módulos dos usuários que são movidos no modelo hierarquizado da fog computacional. A estratégia *Concurrent* apresenta a pior performance com uma diferença de mais de 4000ms para o jogo EEG e de mais de 900ms para o VSOT em relação às outras duas estratégias quando mais de 12 usuários estão utilizando as aplicações. A estratégia FCFS apresenta um melhor desempenho para o VSOT, pois realoca alguns módulos para a nuvem evitando uma sobrecarga na *Cloudlet*.

Para o jogo EEG a estratégia *Delay-priority* apresenta uma melhor performance, porém para o cenário com 12 usuários sua performance se iguala ao da estratégia FCFS devido à ausência de mais capacidade de cpu na *Cloudlet*, o que exige que alguns módulos sejam movidos para nuvem o que gera um maior impacto para aplicações da classe de quase tempo-real (que é o caso do jogo EEG). No tocante ao tráfego de rede, a estratégia *Concurrent* apresenta a menor taxa, pois não realoca os módulos em outras *Cloudlets* ou na nuvem. A maior variação na taxa de tráfego é apresentada pela estratégia *Delay-priority* devido à realocação de mais módulos para a nuvem.

Comparação dos Trabalhos

A Tabela 3.1 mostra um comparativo entre as características dos trabalhos correlatos e a proposta contida neste documento.

A proposta apresentada em Lin and Shen (2017) utiliza um mecanismo simplificado de cálculo de confiança para selecionar os supernós. Assim, cada nó utiliza somente avaliações de sua própria base para realizar a seleção dos supernós, empregando, portanto, um modelo de confiança direta. Já na proposta de He et al. (2017) os nós de fog computacional são selecionados com base em uma medida de estabilidade dos usuários da plataforma de *livecast*. Essa medida considera o tempo de permanência do usuário em uma determinada transmissão e seu histórico geral de uso da plataforma. Os usuários com uma tendência maior de permanência em uma determinada transmissão serão selecionados como nós de fog computacional. Em Skarlat et al. (2016), os nós de fog, ou *fog cells*, conforme denominados pelos autores, são selecionados com base na disponibilidade de seus recursos

como memória, cpu, e banda. Além disso, o método de seleção considera a distribuição de carga entre os nós, visando garantir a máxima utilização do conjunto de *fog cells*, evitando o envio de requisições para a nuvem.

Tabela 3.1: Comparação das propostas

		Características				
		Cenário de uso	**Seleção de nós**	**Ambiente de experimentação**	**Parâmetros de avaliação**	**Topologia de implantação**
Propostas	**Lin and Shen (2017)**	*Cloud gaming*	Confiança direta	Peersim / Planetlab	Perda de pacotes (QoS), latência de resposta, custos de infraestrutura	Aberta
	Q. He et al. (2017)	*Crowdsourced livecast*	Estabilidade / Disponibilidade	Planetlab	Latência de resposta	Controlada
	Skarlat et al. (2016)	IoT	Balanceamento de carga	Cloudsim	Latência de resposta	Parcialmente aberta
	Alam et al. (2016)	*Code offloading*	Tempo esperado de resposta	OMNeT++	Tempo de execução	Parcialmente aberta
	Bittencourt et al. (2017)	IoT	Capacidade disponível	iFogSim	Latência de resposta / Tráfego de rede	Fixa
	Este trabalho	*Cloud gaming*	Confiança e reputação	Peersim / Planetlab	Perda de pacotes (QoS)	Fixa / Controlada / Parcialmente aberta / Aberta

O método de confiança utilizado em Lin and Shen (2017), segundo os autores, serve como base para que no processo de seleção de supernós os nós evitem selecionar aqueles que estejam deliberadamente diminuindo a capacidade alocada para a provisão de serviço na fog. Nesse sentido, o método de seleção utilizado naquele trabalho considera um aspecto qualitativo mais subjetivo que parâmetros de recursos alocados para a fog como cpu, memória e banda, utilizados por Skarlat et al. (2016), ou a estabilidade utilizada por He et al. (2017). Esse aspecto qualitativo considera a manutenção da QoS ao longo das interações experimentadas pelo nó e aplica essa medida para avaliar subjetivamente a confiança que um nó tem em relação ao comprometimento de um supernó em manter a capacidade alocada e, desse forma, não comprometer o serviço consumido.

A decisão acerca de quando confiar e, então, selecionar um supernó, poderia ser feita com base em outras fontes de informação que não somente as próprias experiências do nó que realiza a avaliação. Os modelos de confiança e reputação disponíveis na literatura (Marsh (1994), Sabater (2002) e Huynh (2006), por exemplo) podem servir como base para a construção de uma abordagem mais abrangente para o problema de seleção de supernós confiáveis. Soma-se a esses trabalhos o metamodelo para adaptação de confiança e reputação definido por Hoelz (2013), que oferece uma visão ampla dos elementos base relativos à confiança e reputação em sistemas multiagentes, e pode ser utilizado como fonte para a definição de um modelo customizado para arquiteturas de fog computacional.

O presente trabalho segue essa direção e aplica um método de seleção de supernós baseado em confiança e reputação. Dessa forma, na proposta apresentada neste trabalho

inclui-se a utilização de informações de terceiros como base para a seleção de um supernó (deste ponto em diante o termo supernó será utilizado como sinônimo para nó de fog, ou seja, um provedor local de serviço). Assim, no âmbito desta proposta amplia-se o método de seleção para o uso de um modelo mais abrangente de confiança e reputação que permite que um nó combine a sua própria base de avaliações com os dados enviados por terceiros acerca do serviço provido pelos supernós que são alvo do processo de seleção. Nesse sentido, a seleção baseada em C&R poderia extrapolar o aspecto qualitativo e incluir também as medidas quantitativas relativas provisão do serviço. Isso pode ser obtido por meio da utilização da QoS como parâmetro base para avaliação da C&R. A QoS é capaz de condensar ambos os aspectos, qualitativo e quantitativo, pois é afetada se há algum problema que afeta os recursos alocados pelo supernó como cpu e memória (aspecto quantitativo) e alguma diminuição deliberada da capacidade alocada para a fog (aspecto qualitativo). Dessa forma, a troca de informações entre os nós acerca dos supernós e o próprio processo de seleção conduzido pelos nós se dá considerando a QoS aferida. Esta abordagem é utilizada no âmbito da proposta apresentada pelo presente trabalho.

No tocante aos ambientes de experimentação, o uso da infraestrutura disponibilizada pelo Planetlab pode ser encontrado em Lin and Shen (2017) e He et al. (2017). O simulador CloudSim é utilizado em Skarlat et al. (2016), por oferecer opções para a simulação de recursos computacionais como memória e capacidade de processamento, além de incluir um pacote de implementações de políticas de provisionamento e distribuição de tarefas. A extensão desse simulador para o contexto de fog computacional, o iFogSim (Gupta et al., 2016), é utilizado por Bittencourt et al. (2017). Em Lin and Shen (2017), além do PlanetLab, o simulador Peersim também é utilizado como ambiente experimentação. Nesse mesmo sentido, o presente trabalho decidiu pela utilização do Peersim por oferecer grande flexibilidade para a construção dos elementos e protocolos a serem testados e facilitar a definição de configurações de rede. Além disso, o Peersim fornece um padrão de simulação orientado à eventos que permite a construção de protocolos mais fieis à um ambiente de teste real. Uma vantagem que pode-se destacar entre o Peersim e o IFogSim, e que influenciou a escolha do primeiro, é a capacidade de suportar um grande número de nós na simulação. A escolha do PlanetLab segue os mesmos motivos apresentados em Lin and Shen (2017) e He et al. (2017), ou seja, a possibilidade de se utilizar uma rede real de experimentação com máquinas distribuídas geograficamente.

A latência de resposta é utilizada como parâmetro de avaliação em todos os trabalhos aqui descritos por ser uma das principais melhorias que podem ser obtidas pela adoção do modelo de fog computacional. O trabalho de Alam et al. (2016) apresenta uma peculiaridade pois trata do tempo de execução para o algoritmo (N-Queens) por realizar a comparação entre o tempo de execução no dispositivo final, na nuvem e nos nós de fog. Entretanto, apesar de não ser explicitado no trabalho a latência de resposta está embutida no tempo de execução para os casos onde o N-Queens é executada na nuvem ou nos nós de fog. Além da latência de resposta, Bittencourt et al. (2017) avalia o impacto de sua proposta para o tráfego de rede, outro benefício propalado pela adoção do modelo de fog. Um parâmetro claro de QoS é utilizado em Lin and Shen (2017), que considera a perda de pacotes para o cálculo da QoS.

O presente trabalho também toma como base a perda de pacotes como parâmetro de QoS para ser utilizado nos experimentos. Porém, o intuito da proposta aqui apresentada é ser flexível e permitir o uso de um valor genérico de QoS que pode ser alterado conforme a

necessidade e as características do cenário de aplicação. Ficaria fixada somente o requisito de se definir a QoS por um valor numérico compreendido em um intervalo (por exemplo [0,1]). Dessa forma, diferentes parâmetros podem compor o cálculo da QoS resultando em um valor único numérico que seria utilizado na avaliação de C&R que faz parte da proposta deste trabalho.

Os trabalhos correlatos também foram classificas segundo a topologia de implantação que são utilizadas em suas propostas. A proposta apresentada em Lin and Shen (2017) utiliza uma topologia de implantação **Aberta** para seu modelo de jogos *online* em fog, pois não há a presença de um nó de controle na fronteira da rede ficando os nós de fog diretamente ligados à nuvem. Além disso o modelo de Lin and Shen (2017) utiliza os próprios dispositivos finais como nós de processamento local (supernós). He et al. (2017), por sua vez, utiliza *data centers* regionais para realizar a coordenação local dos nós de fog responsáveis pela transcodificação dos conteúdos de vídeo. Esses *data centers* regionais são fixados pelo provedor de serviço de CLS. Dessa forma, a proposta de He et al. (2017) aplica uma topologia de implantação **Controlada**.

As propostas de Skarlat et al. (2016) e Alam et al. (2016) utilizam topologias de implantação **Parcialmente aberta**. Skarlat et al. (2016) apresenta as *fog colonies*, onde um dos nós de fog (denominados *fog cells* no âmbito daquele trabalho) é selecionado dinamicamente para atuar como *fog orchestration control node* provendo um controle local para as *fog cells*. Em Alam et al. (2016) os nós de fog são organizados em grupos denominados *mobile fogs* que ficam responsáveis por executar o código enviado pelos dispositivos finais por meio do *code offloading*. No âmbito dessas *mobile fogs* é apontado um nó que atuará como líder e fará a interação com o agente responsável por selecionar os nós de fog que recepcionarão os códigos migrados. Além disso, no modelo apresentado em Alam et al. (2016) os módulos presentas nas estações LTE coordenam a interação entre as diferentes *mobile fogs*.

Na proposta apresentada por Bittencourt et al. (2017) as *Cloudlets* são executadas em dispositivos pré-selecionados que não são alterados dinamicamente. Ocorre somente a mudança de carga de requisições feitas pelos dispositivos finais de uma *Cloudlet* para outra em uma localidade diferente. Dessa forma, essa proposta aplica a topologia de implantação **Fixa**.

O modelo apresentado é flexível para ser implementado em diferentes topologias de implantação da fog computacional. O modelo de C&R pode ser adaptado para ser utilizado para a topologia Fixa de maneira a auxiliar o provedor de serviço à controlar os dispositivos pré-selecionados para atuarem como nós de fog. O provedor pode utilizar o modelo de C&R juntamente com as avaliações de QoS feitas pelos dispositivos finais para identificar nós de fog que devem ser substituídos. Nas demais topologias o modelo aqui proposto pode ser aplicado de maneira geral no processo de seleção dos nós, tanto dos provedores locais de serviço quanto dos coordenadores locais. Ademais, por estar baseado em agentes, o modelo aqui proposto permite que sejam realizadas adaptações no processo de decisão acerca de quais nós selecionar de acordo com a topologia de implantação. Os agente podem de forma autônoma decidir quais parâmetros e métodos serão aplicados para a seleção dos nós na fog computacional.

Capítulo 4

Modelo proposto

Esse capítulo apresenta o modelo proposto para seleção de nós e supernós o qual se baseia em agentes inteligentes com uso de confiança e reputação no ambiente de fog computacional. Na Seção 4.1 será apresentada uma visão geral do modelo proposto. A Seção 4.2 apresenta a descrição dos agentes que compõem o modelo. A Seção 4.3 contém a descrição do modelo de C&R utilizado no âmbito desta proposta. Por fim, na Seção 4.4 será apresentada a arquitetura da aplicação implementada a partir do modelo proposto com alguns detalhes sobre esta implementação.

4.1 Visão geral do modelo

A opção por um modelo baseado em agentes se dá principalmente pela característica de autonomia referente a esse tipo de elemento computacional. Conforme já discutido no capítulo introdutório deste documento, os agentes podem tomar decisões autônomas visando obter um determinado resultado. Essa característica pode ser aproveitada em um modelo de fog computacional permitindo que a gestão e controle dos elementos constituintes da fog seja feita de maneira autônoma prescindindo, ou pelo menos diminuindo, a atuação humana.

A Figura 4.1 apresenta uma visão geral do modelo que é composto por quatro agentes: Gerente de nuvem, Coordenador local, supernó e nó. Esses agentes representam os elementos da fog computacional e são definidos de maneira a permitir a interação entre os agentes e entre os serviços que são executados no âmbito da fog.

Assim, os agentes aqui propostos formam uma camada base que é consumida por algum serviço que será disponibilizado na fog computacional, conforme ilustrado na Figura 4.2. O objetivo dessa camada formada pelos agentes é garantir a seleção de supernós que proverão serviços na fog com a QoS requisitada.

O modelo pode ser utilizado em diferentes topologias de implantação da fog computacional. No entanto, essa topologia deve ser pré-selecionada previamente à implantação dos agentes do modelo. Os agentes receberão a definição da topologia selecionada como um parâmetro de inicialização. Os agentes do modelo não são capazes de se adaptar de forma autônoma à mudanças no ambiente de modo a alterar a topologia de implantação.

As interações entre os agentes são modificadas à depender da topologia de implantação da fog. A distinção principal ocorre entre a topologia Fixa, a Aberta e as demais. Isso

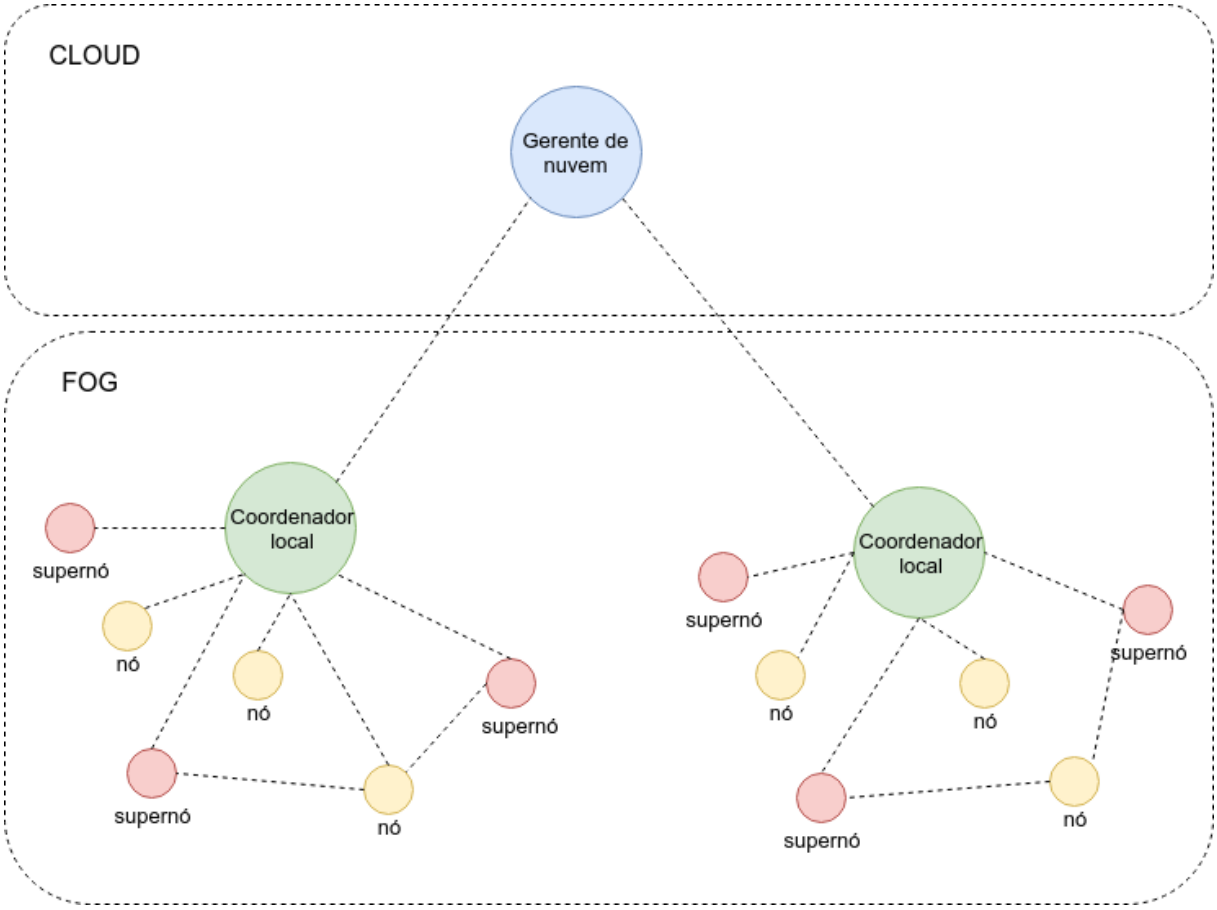

Figura 4.1: Visão geral do modelo proposto

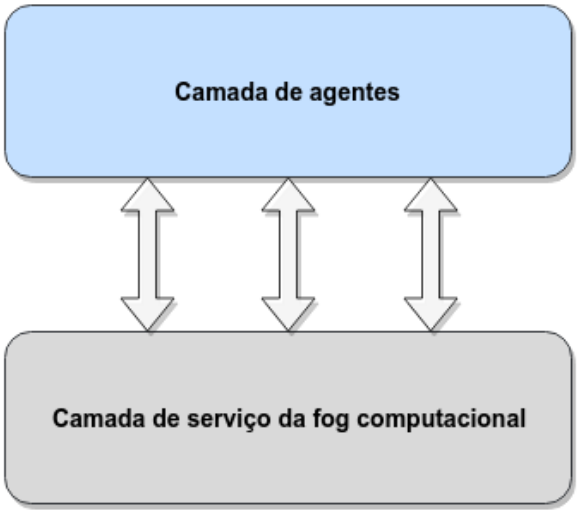

Figura 4.2: Diferentes camadas do modelo

ocorre porque na topologia Fixa e Aberta não há a presença do nó de coordenação local. A seguir serão apresentados os fluxos de comunicação ajustados para essas topologias.

Fixa

Nesta topologia os supernós não são escolhidos dinamicamente. Desse modo, a lista desses supernós é pré-definida e fornecida para o Gerente de nuvem. A Figura 4.3 apresenta o fluxo de comunicação utilizado pelo agentes nesta topologia. Nesse fluxo, ao ser iniciado o nó se registra com o Gerente de nuvem (*Passo 1*). O Gerente de nuvem busca na sua base local qual o supernó mais próximo do nó e retorna como resposta à solicitação enviada (*Passo 2*). Caso o Gerente de nuvem não encontre nenhum supernó disponível o nó irá consumir o serviço diretamente dos servidores na nuvem. Caso contrário, o nó se conecta ao supernó selecionado e inicia a interação para consumo do serviço. Ao fim da interação, o nó avalia a sua satisfação com o serviço prestado pelo supernó definindo um valor de QoS e envia essa avaliação para o Gerente de nuvem (*Passo 3*).

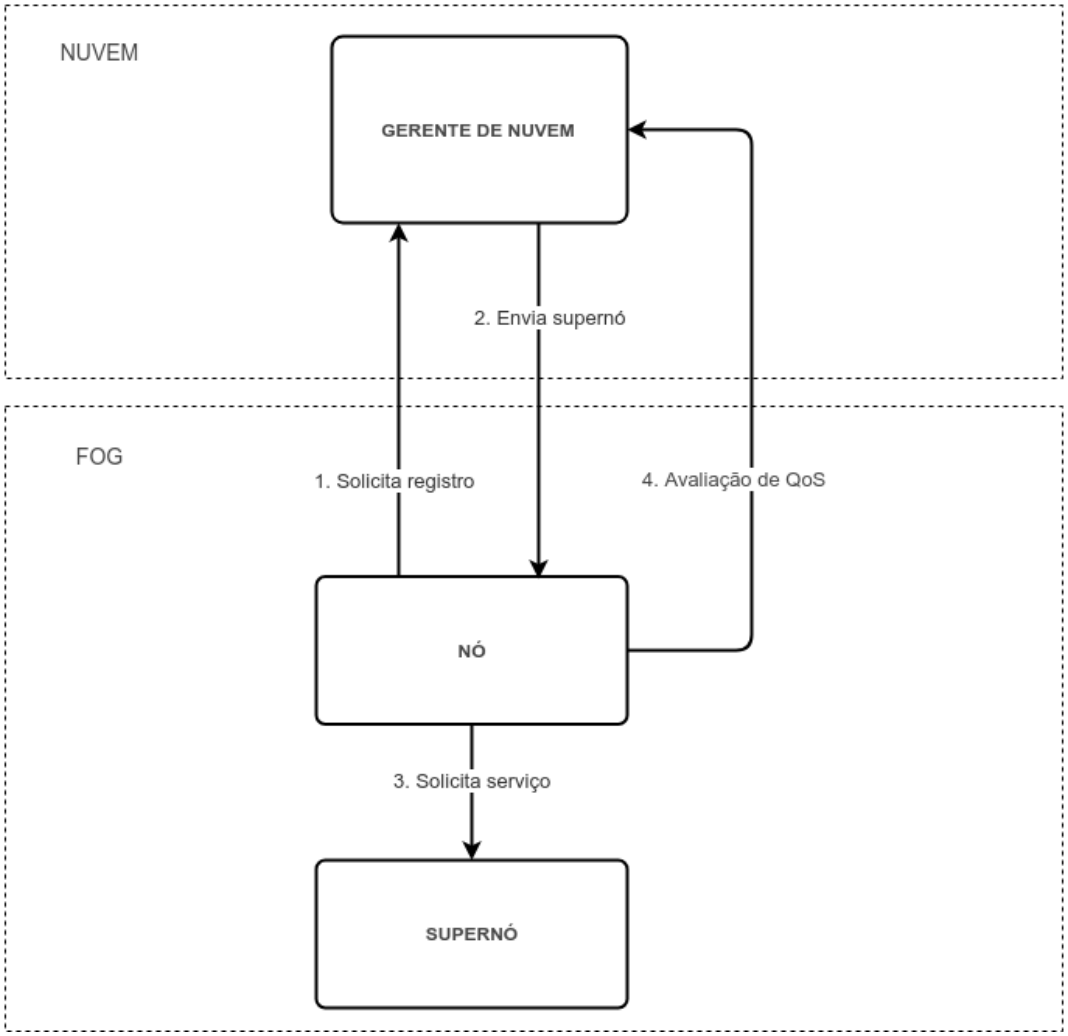

Figura 4.3: Fluxo de comunicação para a topologia Fixa

Controlada e Parcialmente aberta

Conforme apresentado na Figura 4.1 nas topologias Controlada e Parcialmente aberta o modelo apresentado neste trabalho apresenta o agente Coordenador local como o nó

de coordenação na fronteira de rede. A Figura 4.4 apresenta o fluxo de comunicação utilizado pelo nó para selecionar os supernós e utilizar o serviço provido na fog. Nesse fluxo, ao ser iniciado o nó se registra com o Gerente de nuvem (*Passo 1*). O Gerente de nuvem busca na sua base local qual o Coordenador Local mais próximo do Nó e retorna como resposta à solicitação enviada (*Passo 2*). Na sequência destes passos iniciais, o Nó solicita ao Coordenador local a lista de supernós próximos que estão disponíveis (*Passo 3*). O Coordenador Local busca os supernós disponíveis e envia como resposta ao Nó, juntamente com a tabela contendo o registro das interações realizadas entre os supernós e os demais nós ligados ao Coordenador Local e a confiança local (ξ) em relação aos supernós e nós (*Passo 4*).

Caso o nó não encontre um supernó adequado ele irá consumir o serviço diretamente dos servidores na nuvem. Caso contrário, o Nó se conecta ao supernó selecionado e inicia a interação para consumo do serviço. O supernó, ao seu turno, envia ao Coordenador Local a tupla (a, b, i) contendo o registro da interação (*Passo 6*). Ao fim da interação, o Nó avalia a sua satisfação com o serviço prestado definindo um valor de QoS e envia essa avaliação para o supernó (*Passo 7*).

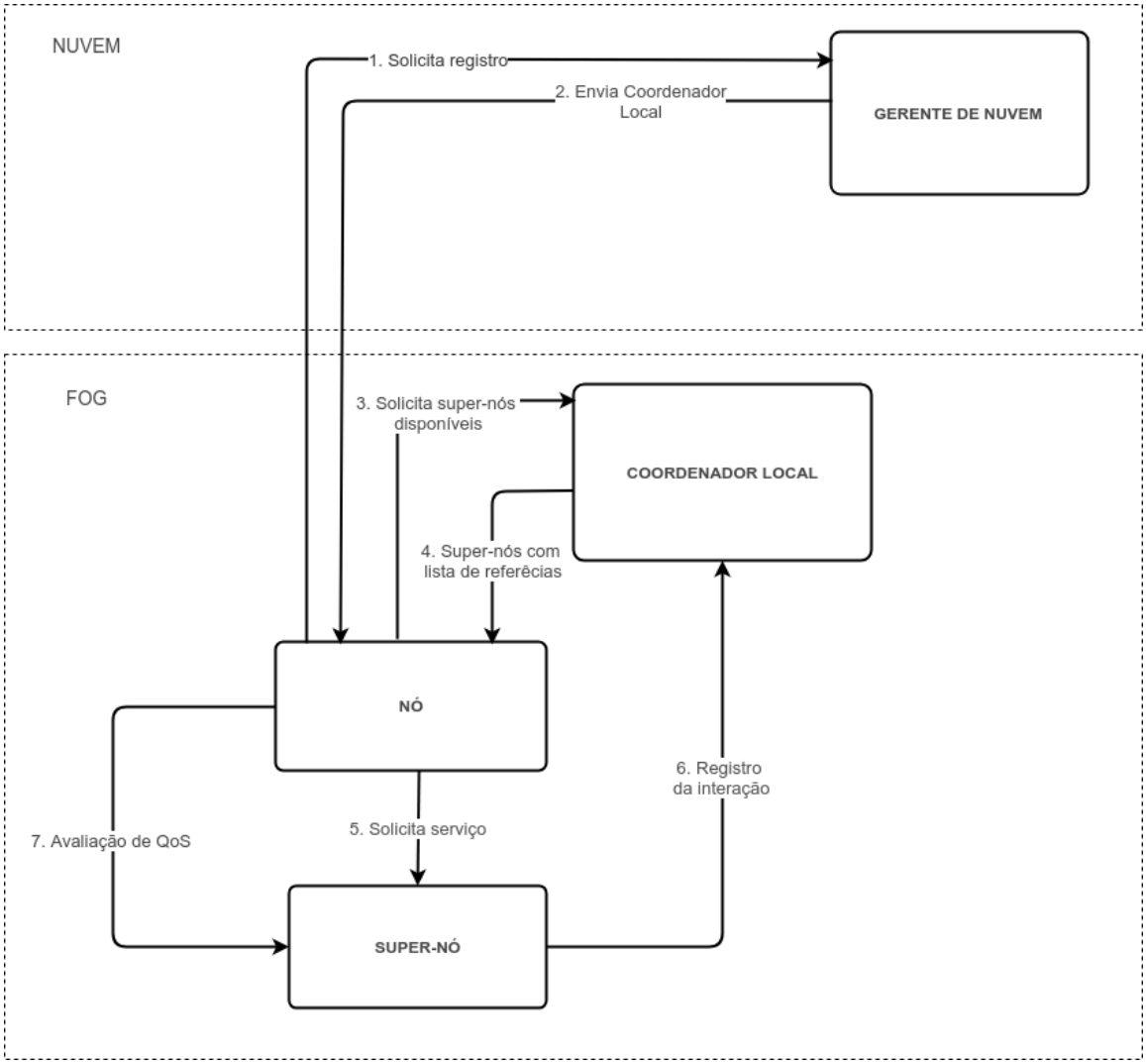

Figura 4.4: Fluxo de comunicação para as topologias Controlada e Parcialmente aberta

Aberta

Na topologia Aberta não há a presença do Coordenador local. A Figura 4.5 apresenta o fluxo de comunicação utilizado pelo nó para selecionar os supernós e utilizar o serviço provido na fog nesta topologia. Nesse fluxo, ao ser iniciado o nó se registra com o Gerente de nuvem (*Passo 1*). O nó então solicita ao Gerente de nuvem a lista de supernós próximos que estão disponíveis (*Passo 2*). O Gerente de nuvem busca os supernós disponíveis e envia como resposta ao nó, juntamente com a tabela contendo o registro das interações realizadas entre os nós e supernós (*Passo 3*). Caso o nó não encontre um supernó adequado ele irá consumir o serviço diretamente dos servidores na nuvem. Caso contrário, o nó se conecta ao supernó selecionado e inicia a interação para consumo do serviço. O supernó, ao seu turno, envia ao Gerente de nuvem a tupla (a, b, i) contendo o registro da interação (*Passo 5*). Ao fim da interação, o Nó avalia a sua satisfação com o serviço prestado definindo um valor de QoS e envia essa avaliação para o supernó (*Passo 6*).

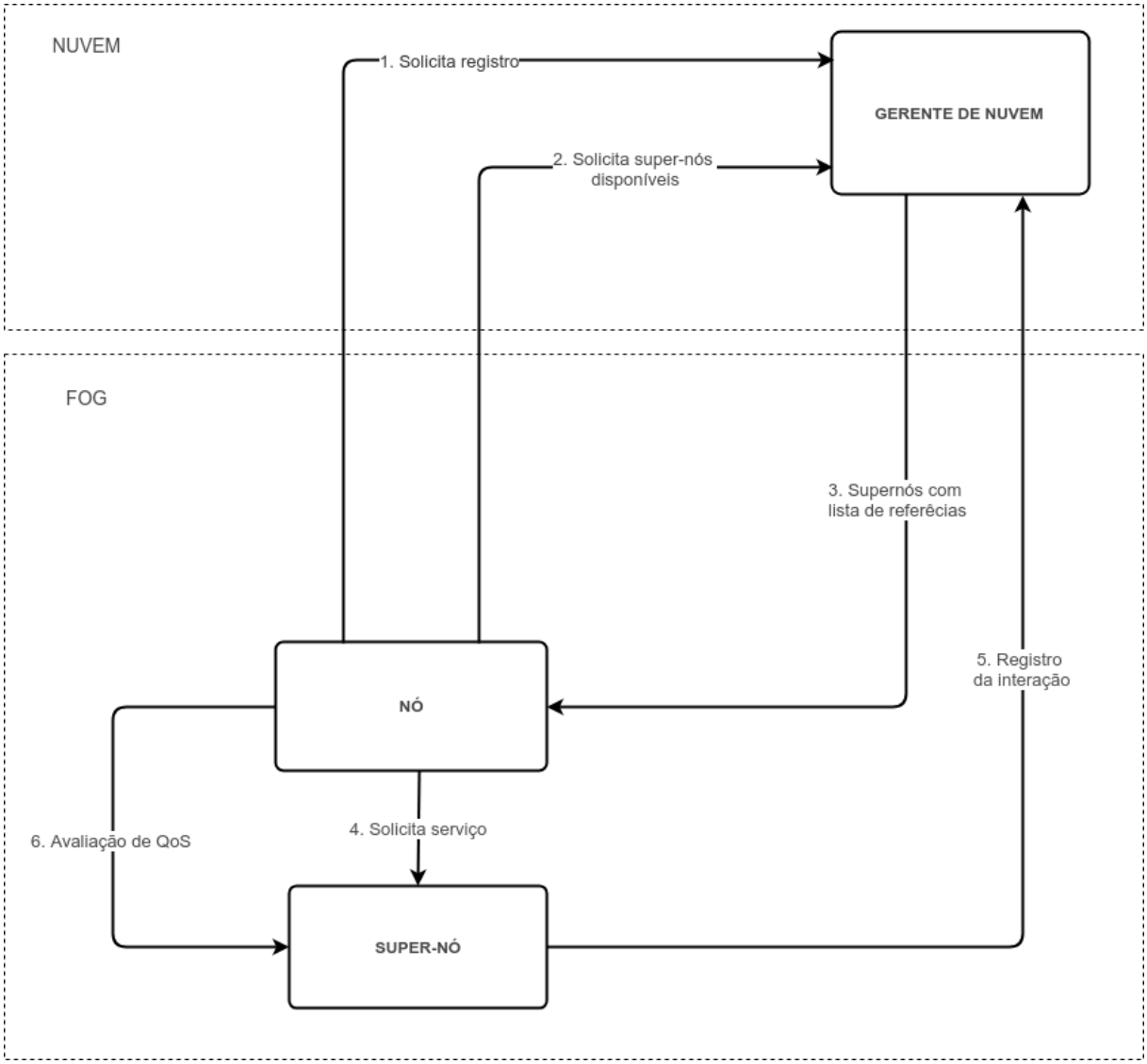

Figura 4.5: Fluxo de comunicação para a topologia Aberta

4.2 Descrição dos agentes

Gerente de nuvem

O Gerente de nuvem é o agente que representa o provedor de serviço e fica localizado no servidor de nuvem. Trata-se de um agente orientado a objetivos que tem uma visão geral sobre os demais elementos do modelo, mantendo as bases de registro que armazenam a identificação dos coordenadores locais, supernós, e nós. O objetivo principal deste agente é garantir a QoS esperada para o serviço sendo executado na fog. Apesar dos nós interagirem com outros elementos do modelo, a responsabilidade final sobre a QoS é do Gerente de nuvem por representar o provedor do serviço.

É comum que os serviços providos na nuvem utilizem mais de um servidor localizado em *datacenters* diferentes. O Gerente de *cloud*, entretanto, abstrai essa estrutura e é visto como um elemento único, apesar de sua implementação poder estar distribuída entre vários servidores.

A identificação da topologia de implantação da fog é pré-estabelecida pelo provedor de serviço e é fornecida para este agente. Cabendo a ele compartilhar essa informação com os demais agentes do modelo. As atribuições, fluxos de interação e processos decisórios conduzidos pelo Gerente de nuvem dependem da topologia de implantação da fog.

Nos casos em que a topologia de implantação for Fixa, cabe ao Gerente de nuvem controlar a QoS dos supernós. Para tanto o Gerente de nuvem utiliza as avaliações acerca da QoS enviadas pelos nós conforme apresentado no *Passo 4* do fluxo da Figura 4.3. Periodicamente o Gerente de nuvem utiliza essas avaliações enviadas pelos nós para calcular a confiança em relação aos supernós. A partir desse cálculo o Gerente de nuvem pode decidir se o dispositivo no qual o supernó está implantado deve ser substituído ou deve passar por algum processo de manutenção, conforme pode ser visualizado no Algoritmo 1.

A periodicidade em que o processo descrito nesse algoritmo é executado depende do cenário de aplicação da fog computacional. A discussão de granularidade temporal colocada por Huynh (2006) e apresentada na Seção 2.5 (na parte que trata de Memória e recência), pode ser aplicada aqui para estabelecer qual a periodicidade que será utilizada pelo Gerente de nuvem. As funções utilizadas na linha 6 e 7 do algoritmo serão descritas na Seção 4.3 deste capítulo.

Algoritmo 1 Avalia supernos

```
1:  AvaliarSupernos( )
2:      supernos ← RecuperarSupernos()
3:      baseDeAvaliacoes ← CarregarLimiteDistancia()
4:      para cada superno ← supernos faça
5:          avaliacoes ← CarregarAvaliacoes(superno)
6:          confianca ← CalcularConfianca(avaliacoes)
7:          avaliacaoConfianca ← AvaliarConfianca(confiancas)
8:          se avaliacaoConfianca < limiteDeConfiana então
9:              SubstituirSuperno(superno)
10:         fim
11:     fim
12: fim
```

A introdução do Coordenador local como o nó de coordenação na fronteira de rede, nas topologias Controlada e Parcialmente aberta, resultam na inclusão de alguns objetivos ao Gerente de nuvem que se referem à gestão dos Coordenadores locais. Esses objetivos seguem listados a seguir:

- Manter as bases de registro dos coordenadores locais, supernós e nós - os agentes integrantes da fog se registram com o Gerente de nuvem, que mantém um registro contendo a identificação e localização desses agentes.
- Selecionar Coordenadores Locais - em uma topologia de implantação **Controlada** a seleção dos Coordenadores Locais é feita de maneira estática, dessa forma, os dispositivos na fronteira onde serão implantados os Coordenadores Locais são apontados pelo provedor de serviço. Em uma topologia de implantação **Parcialmente aberta** os Coordenadores Locais são selecionados pelo Gerente de nuvem a partir de dispositivos finais que se candidatam. O processo de seleção segue o protocolo de interação FIPA Contract Net[1]. O Gerente de nuvem estabelece um valor de recompensa para os dispositivos que irão executar o agente Coordenador Local e seleciona aqueles que enviarem as melhores propostas. Essa recompensa pode ser baseada em um valor "pago"para cada nó que for suportado pelo Coordenador local, por exemplo. O Gerente de nuvem envia o pedido de propostas colocando um valor máximo a ser recompensado para cada nó suportado pelos Coordenadores locais. Um outro método que pode ser utilizado é calcular o valor de recompensa baseado em horas, como é feito comumente em provedores de serviços de nuvem como a Amazon AWS. Nesse caso, o Gerente de nuvem estabelece um valor máximo a ser pago por hora para o Coordenador local. Detalhes da definição desta recompensa foge ao escopo deste trabalho e será tratada no Capítulo 6 como trabalhos futuros relativos ao modelo aqui proposto.
- Indicar os Coordenadores Locais aos supernós e nós - cabe ao Gerente de nuvem identificar qual o Coordenador Local mais próximo a um nó ou supernó e indicar esse Coordenador ao agente em questão, conforme o Algoritmo 2.

  Um limiar de distância é pré-estabelecido de modo a evitar que um Coordenador Local muito distante dos nós/supernós requisitantes seja indicado, resultando em um impacto elevado na latência das requisições de controle realizadas por esse agente e invalidando o seu uso mesmo. Caso nenhum Coordenador Local seja encontrado, os nós e supernós direcionam suas requisições para o Gerente de nuvem.
- Controlar a QoS provida pelos Coordenadores Locais - o Gerente de nuvem deve garantir que os Coordenadores Locais estejam cumprindo com suas atribuições. Para isso, o Gerente de nuvem conta com os nós para que avaliem o Coordenador Local ao qual estão ligados. Essas avaliações feita pelos nós se referem à disponibilidade do Coordenador local. Para tanto os nós consideram eventos de indisponibilidade do Coordenador local, ou seja, interações iniciadas com esse agente que não foram completadas. Esse valor é comparando com o quantitativo total de interações realizadas com esse Coordenador para compor um valor de avaliação de QoS que será enviado ao Gerente de nuvem.

[1] http://www.fipa.org/specs/fipa00029/SC00029H.html

Algoritmo 2 Indicar Coordenador Local para nós e supernós

```
1:  IndicarCoordenadorLocal(localizacao)
2:      coordenadoresLocais ← RecuperarCoordenadoresLocais()
3:      limiteDistancia ← CarregarLimiteDistancia()
4:      coordenadorIndicado ← NULL
5:      menorDistancia ← limiteDistancia
6:      para cada coordenadorLocal ← coordenadoresLocais faça
7:          distancia ← CalcularDistancia(coordenadorLocal.localizacao,localizacao)
8:          se distancia < limiteDistancia e distancia < menorDistancia então
9:              coordenadorIndicado ← coordenadorLocal
10:             menorDistancia ← distancia
11:         fim
12:     fim
13:     retorna coordenadorIndicado
14: fim
```

Dessa forma, seguindo os passos do Algoritmo 3, o Gerente de nuvem avalia se é necessário mudar o Coordenador local. Periodicamente o Gerente de nuvem seleciona um grupo de Coordenadores Locais para serem avaliados (passo 3). Essa seleção é feita com o intuito de evitar uma sobrecarga de processamento no Gerente de nuvem quando existir um número elevado de Coordenadores Locais. São solicitadas aos nós as avaliações acerca dos Coordenadores selecionados (passo 7). Os nós avaliam o Coordenador ao qual estão ligados considerando a QoS medida com base na disponibilidade do mesmo, conforme a Equação 4.1:

$$V_{cqos} = \frac{N_{fal}}{N_{req}} \tag{4.1}$$

onde V_{cqos} representa a QoS, N_{fal} o número de requisições não atendidas e N_{req} o número de requisições enviadas pelo nó ao Coordenador Local.

Algoritmo 3 Controla a QoS dos Coordenadores Locais

```
1:  ControlarQoSCoordenadoresLocais( )
2:      limiarDeConfianca ← CarregarLimiarDeConfiança()
3:      coordenadoresLocais ← RecuperarCoordenadoresLocais()
4:      coordenadoresSelecionados ← SelecionarCoordenadoresLocais(coordenadoresLocais)
5:      para cada coordenadorLocal ← coordenadoresSelecionados faça
6:          nos ← RecuperarNosParaCoordenador(coordenadorLocal)
7:          avaliacoes ← RecuperarAvaliacoesParaCoordenador(nos)
8:          confianca ← CalcularConfiança(avaliacoes)
9:          se confianca < limiarDeConfianca então
10:             DefinirNovoCoordenadorLocal()
11:         fim
12:     fim
13: fim
```

Com base nessas avaliações o Gerente de nuvem realiza um cálculo de confiança em relação ao Coordenador local. Os detalhes desse cálculo serão apresentados na Seção 4.3. Aqueles cuja confiança for menor que um limiar que é pré-definido serão substituídos (passo 9). Quando se tratar de um cenário de topologia de implantação Controlada, a implantação de um novo Coordenador Local é feita diretamente pelo Gerente de nuvem. Quando se tratar de uma topologia de implantação Parcialmente Aberta, a indicação de um novo Coordenador Local segue um processo interativo que busca dispositivos finais que estejam interessados em hospedar um Coordenador Local em troca de uma recompensa.

Na topologia **Aberta** não há a presença do Coordenador local, dessa forma os objetivos referentes à gestão desses agentes não são utilizados pelo Gerente de nuvem, restando somente o objetivo relativo à manutenção das bases de registros dos nós e supernós.

Coordenador local

O Coordenador Local é um agente orientado a objetivos responsável por coordenar a seleção de um conjunto de supernós que estão geograficamente próximos. No mesmo sentido da *fog colony* de Skarlat et al. (2016), apresentada na Seção 3, o Coordenador local forma um tipo de comunidade que agrupa supernós e nós que estão localizados próximos uns dos outros. De modo análogo ao Gerente de nuvem, o Coordenador Local mantém uma base local de registro desses nós e supernós.

Para facilitar o processo de seleção dos supernós o Coordenador local mantém uma base com a identificação dos supernós próximos e uma tabela de interações ocorridas entre supernós e nós. Essa tabela de interações é constituída por tuplas com o seguinte formato: $d = (a, b, i)$, onde a é a identificação do nó, b a identificação do supernó e i a interação. O Coordenador local define um período de validade para as tuplas (ex.: 5 interações) e, após o vencimento, a tupla é descartada. Somente a tupla mais recente (com maior valor de i) é mantida para cada par (a, b).

Com base nessa tabela de interações o Coordenador local calcula o valor de confiança local para a comunidade de nós e supernós vinculados a ele. Esse valor de confiança é utilizado pelos nós daquela comunidade para embasar suas decisões acerca da seleção dos supernós. Os detalhes do cálculo desse valor serão apresentados Seção 4.3 (Inicialização), mas em suma ele verifica a variabilidade das avaliações de QoS dadas pelos nós e analisa esse dado em conjunto com a variabilidade dos valores obtidos à partir da diferença entre os valores de QoS esperados e os valores de QoS das avaliações da base de dados de cada nó.

Em uma topologia de implantação **Controlada** este agente é implantado pelo Gerente de nuvem em um dispositivo na fronteira da rede. Para uma topologia **Parcialmente aberta** o Coordenador Local é implantado em um dispositivo final que se candidata para hospedá-lo em troca de alguma recompensa. Hospedar o Coordenador local não impede que o dispositivo também hospede um agente nó e supernó, já que cada um desses agentes possui objetivos próprios e atuam de forma diversa no âmbito do modelo. Nos casos em que a topologia de implantação é **Aberta**, e não há um Coordenador local, as atribuições relativas a este elemento são executadas pelo Gerente de *cloud*.

Nó e supernó

O Nó é o agente que representa o dispositivo usuário consumidor de serviço. Trata-se de um agente orientado à utilidade que aplica o modelo de confiança e reputação apresentado na Seção 4.3 para selecionar os supernós aos quais irá se conectar para consumir o serviço provido na fog computacional.

No mesmo sentido que as propostas apresentadas em (Lin and Shen, 2017) e (He et al., 2017) o supernó representa o dispositivo usuário que disponibiliza parte de seus recursos para atuar como um provedor local de serviço na fog computacional. Todo supernó também é um nó, podendo atuar hora como consumidor de serviço e hora como provedor.

Para selecionar o supernó mais adequado para ser utilizado o nó conduz um processo de seleção que utiliza as avaliações de QoS acerca dos supernós e aplica o modelo de C&R, que será apresentado na Seção 4.3 deste capítulo.

Baseando-se no valor da confiança local, o nó pode escolher entre duas opções para realizar a seleção do supernó, conforme ilustrado na Figura 4.6. Se o valor da confiança local (ξ) for menor que o limiar de confiança local, o nó solicita aos supernós disponíveis os valores das avaliações de QoS que eles receberam dos demais Nós com os quais interagiram de modo a utilizar a reputação certificada para o cálculo de confiança. Se o valor de ξ for maior que o limiar de confiança local, o nó, com base na tabela de registros de interações, solicitará aos demais nós as avaliações de QoS acerca dos supernós disponíveis que constam na lista enviada pelo Coordenador local. Com a posse das informações requisitadas o Nó pode realizar o processo de seleção baseado em C&R que será detalhado na Seção 4.3.

Algoritmo 4 Avalia a QoS do Coordenador Local

```
1:  AvaliarCoordenadorLocal(coordenadorLocal, tamanhoHistorico)
2:      {numeroDeInteracoes, numeroDeFalhas} ← DadosCoordenadorLocal()
3:      se numeroDeInteracoes < limiarDeInteracoes então
4:          numeroDeFalhas ← ContarInteracoesFalhasComCoordenadorLocal()
5:          pecentualDeFalhas ← numeroDeFalhas/numeroInteracoes
6:          valorAvaliacao ← 1
7:          se percentualDeFalhas > requisitoQoSCoodenadorLocal então
8:              valorAvaliacao ← percentualDeFalhas − requisitoQoSCoodenadorLocal
9:          fim
10:         avaliacao ← CriarAvaliacao(valorAvaliacao)
11:         EnviarAvaliação(avaliacao)
12:     fim
13: fim
```

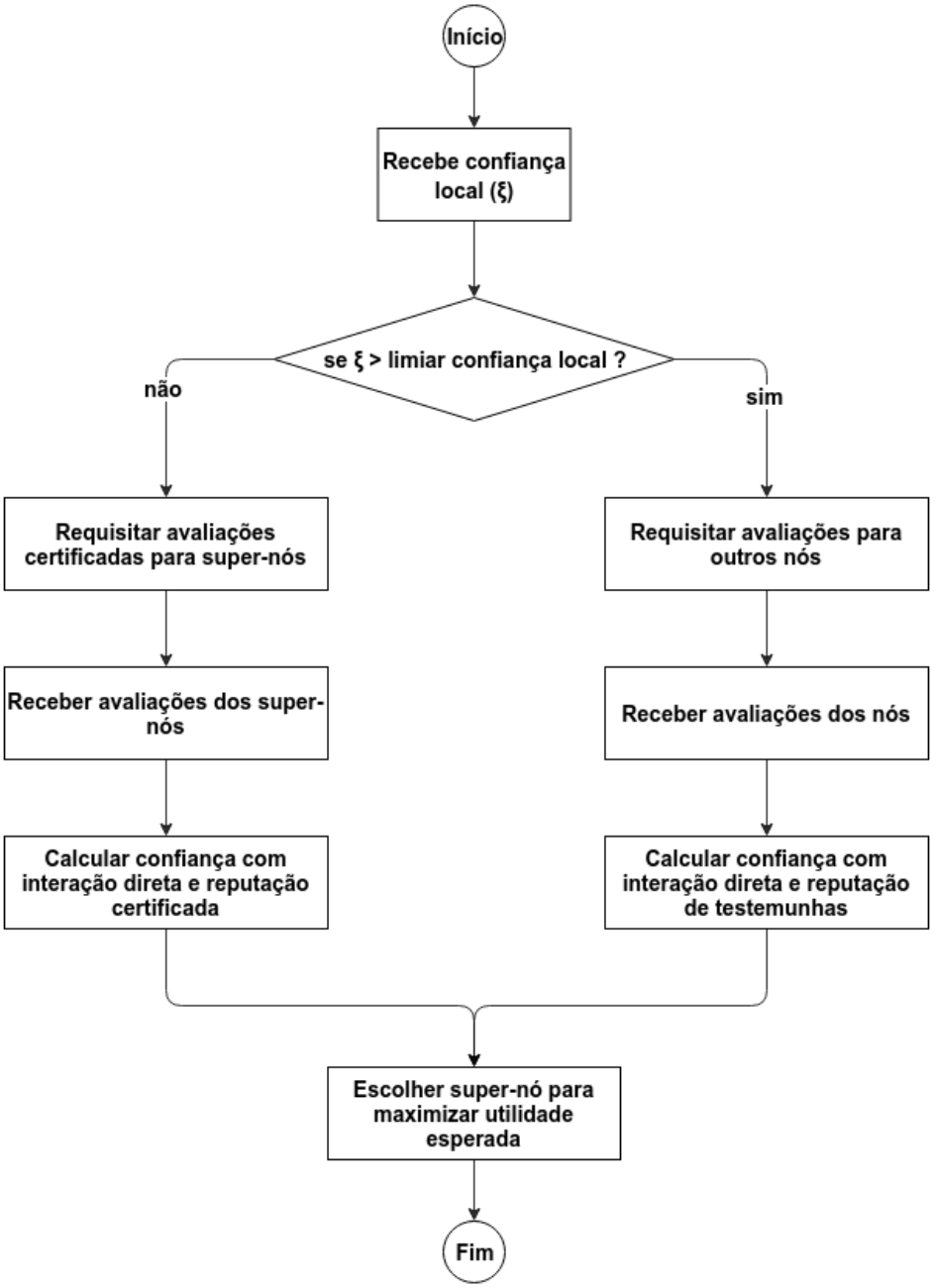

Figura 4.6: Fluxograma de decisão para seleção dos supernós

4.3 Modelo de confiança e reputação

O processo de seleção dos supernós na estrutura de fog computacional proposta aplica um modelo de confiança e reputação. Este modelo é estruturado com base no mapeamento feito do modelo FIRE Huynh (2006) para a o metamodelo de Hoelz and Ralha (2015), conforme apresentado na Seção 2.5. O modelo FIRE foi selecionado por apresentar os ambos os componentes de confiança direta e de reputação. Além disso, o FIRE define um componente de reputação certificada que permite a aplicação da reputação por meio

de um fluxo que gera uma sobrecarga menor para o agente que realiza a avaliação. Em resumo a seleção do modelo FIRE para ser utilizado neste trabalho se deu com base principalmente nos seguintes pontos:

- A presença do componente de reputação certificada que causa uma sobrecarga menor para o agente que realiza a avaliação de C&R. Para um cenário de uso de fog computacional isso é interessante pois os nós podem ser executados por dispositivos com pouco poder de processamento e armazenamento.
- A possibilidade de conjugar a avaliação de confiança direta com a de reputação para o cálculo de um valor final de confiança.
- Um método de avaliação de credibilidade das testemunhas que permite lidar com a inacurácia e/ou vieses no cálculo de reputação.
- Os mecanismos de adaptação que podem ser utilizados no âmbito do modelo.

Nesta seção os aspectos apresentados Seção 2.5 sobre o mapeamento do modelo FIRE para o meta-modelo de Hoelz (2013) serão instanciados para a estrutura de fog computacional. A instanciação destes elementos segue descrita a seguir.

Fontes de informação

O nó pode utilizar como fontes de informação as avaliações de QoS armazenadas no seu histórico local (criadas a partir de suas interações passadas com os supernós), a reputação certificada dos supernós candidatos e a informação de testemunhas. As avaliações são armazenadas como tuplas no mesmo formato que o utilizado no modelo FIRE(Huynh, 2006): $r = (a, b, c, i, v)$, onde a identifica o nó, b o supernó, i é a interação e v é o valor da avaliação. O valor v é calculado com base na QoS aferida pelo nó. Os parâmetros para esse calculo podem variar dependendo do contexto em que o modelo de fog computacional está sendo aplicado. No caso do uso da fog para jogos online ou distribuição de *streaming*, a perda de pacotes poderia ser utilizada como parâmetro para o cálculo do valor de QoS conforme apresentado na Equação 4.2:

$$v = 1 - \frac{n_p}{n_t}, \tag{4.2}$$

onde n_p representa o número de pacotes perdidos e n_t o número total de pacotes.

Assim, o valor 0 representa uma avaliação completamente negativa, enquanto +1 significa uma avaliação absolutamente positiva. Se um nó receber um serviço abaixo da qualidade esperada, o valor da avaliação dada ao supernó fornecedor do serviço terá seu valor diminuindo de acordo com a proporção entre a qualidade esperada e a percebida pelo nó. Ou seja, quão pior a qualidade menor será o valor da avaliação. O mesmo acontece para o caso em que a qualidade percebida pelo nó supera a qualidade requisitada. Nesse caso, o valor da avaliação parte do limiar 0 e é acrescido de acordo com a proporção entre qualidade esperada e percebida.

Duas abordagens podem ser utilizadas pelo nó para selecionar as fontes de informação acerca dos supernós: a reputação certificada e a reputação de testemunhas. A primeira abordagem é utilizada quando o nível de confiança local é maior que o limiar de confiança

local (L_{cl}) definido pelo nó. Nesse caso, utiliza-se a mesma abordagem presente em Huynh (2006), onde os supernós armazenam as avaliações recebidas dos nós com os quais interagiu e as apresentam quando solicitado de modo a formar sua reputação certificada. A segunda abordagem é utilizada quando a confiança geral é menor que L_{cl}. Nesse caso, o nó utiliza a lista de referências enviada pelo Coordenador Local para requisitar aos nós dessa lista as avaliações acerca dos supernós sendo avaliados. Assim como o a confiança local, o valor de L_{cl} é definido e enviado pelo Coordenador local para os nós no momento em que esses são inicializados. Esse limiar pode ser periodicamente atualizado pelo Coordenador local com base no estado do conjunto de nós e supernós que estão ligados.

A escolha entre reputação certificada e informação de testemunhas baseada na confiança local segue a caracterização de previsibilidade dada à reputação obtida à partir dessas duas fontes de informação. Como explica Huynh (2006), a reputação certificada, apesar de demandar um custo menor, tem um poder de previsibilidade reduzido pois os supernós tendem a selecionar somente as melhores avaliações que lhes foram dadas para que sejam enviadas para os nós durante o processo de seleção. Assim, o parâmetro da confiança local é utilizada para dirimir esse risco, utilizando a reputação certificada somente quando há uma possibilidade maior que o supernós estejam utilizando avaliações condizentes com a sua *performance*. Caso contrário, a informação de testemunhas é aplicada por aumentar esse grau de previsibilidade por utilizar informações colhidas diretamente com os demais Nós.

Aquisição e compartilhamento de informações

Como apresentado nos fluxos as Figuras 4.3 e 4.4 da Seção 4.2, a aquisição de informação para a avaliação de C&R é feita por meio de consultas aos supernós e diretamente aos nós.

Quando utilizada a reputação certificada as solicitações por informações para a avaliação da reputação são enviadas pelos Nós diretamente aos supernós. Quando a informação de testemunhas é utilizada para avaliar a reputação do supernó, o nó realiza o processo de aquisição da informação com base na tabela de interações que é disponibilizada pelo Coordenador local ou pelo Gerente de *cloud*. A utilização desta tabela facilita o processo de aquisição diminuindo seu custo ao nó, em comparação com o método de pesquisa em rede de referências, que tende a ser mais demorado e custoso dado a dinamicidade do ambiente de fog computacional. Ademais, em alguns casos, por restrições de recursos computacionais nos Nós a pesquisa em rede de referência poderia inviabilizar a utilização da informação de testemunhas.

Memória e recência

O ambiente de fog computacional é dinâmico e tende a apresentar uma alta frequência de interações entre Nós e supernós. Assim, os Nós tendem a ter à disposição um grande conjunto de interações para compor o histórico de avaliações sobre os supernós. Esse histórico, porém, não pode ser muito extenso pois acarretaria em um consumo de recursos excessivos por parte do Nó. Dessa forma, opta-se por utilizar a abordagem do modelo FIRE (Huynh, 2006), onde é dado um peso maior para as avaliações mais recentes por refletirem mais precisamente o comportamento atual do supernó. Essa abordagem permite que o histórico local armazenado pelos Nós e supernós não seja muito extenso. Para

calcular o peso de cada avaliação segundo sua recência é utilizada a Equação 2.9 da Seção 2.5.

A periodicidade de cada avaliação pode ser estabelecida de acordo com o cenário de aplicação da fog computacional. Em um sistema de *cloud gaming* ou de *streaming* de vídeo, por exemplo, os nós poderiam avaliar os supernós a cada uma hora ou a cada um dia de jogo, pois a interação entre esses elementos se dá por meio de um fluxo contínuo de mensagens, e definir uma interação com granularidade muito baixa poderia acarretar em uma demanda excessiva por espaço de armazenamento nos Nós. Em outros cenários, onde as interações não são baseadas em um fluxo contínuo, como, por exemplo, um sistema de IoT para monitoramento de atividades físicas, os nós poderiam armazenar uma avaliação para cada conjunto de mensagens que formam o fluxo de coleta de dados pelos sensores e recebimento de comandos pelos atuadores. A escala de recência, λ, da Equação 2.9 (Seção 2.5) será atribuída de acordo com essa definição da periodicidade temporal das avaliações.

Credibilidade e confiabilidade

A confiabilidade da avaliação de confiança do modelo aqui proposto utiliza dois fatores em seu cálculo: o número de interações e o grau de variação entre as diferentes avaliações acerca de um supernó. O primeiro fator considera a quantidade de avaliações sobre o supernó que está sendo analisado, visto que quanto mais avaliações um Nó possui acerca de um supernó, maior será a confiabilidade. O cálculo desse fator é feito conforme a Equação 2.3 do modelo FIRE.

O segundo fator considera a variabilidade do conjunto de avaliações. Ou seja, avaliações muito discrepantes sobre um determinado supernó levam a uma incerteza acerca da QoS que este supernó poderá oferecer em interações futuras. A Equação 2.4 do modelo FIRE é utilizada no cálculo deste segundo fator.

A credibilidade é aplicada pelos nós para calcular o peso das avaliações que são enviadas por nós terceiros, tanto para o a reputação por testemunhas, quanto para a reputação certificada. O cálculo dessa reputação é realizado seguindo o método proposto no modelo FIRE (Huynh, 2006), e utilizando as Equações 2.6 e 2.7 (apresentadas na Seção 2.5 do Cap. 2 deste documento). Como visto naquela seção, esse valor de credibilidade é utilizado para calcular o peso das avaliações enviadas pelos agentes terceiros.

Neste mesmo sentido, o Gerente de nuvem aplica uma versão adaptada do método de cálculo de credibilidade das testemunhas do modelo FIRE para calcular a confiança em relação aos Coordenadores Locais. Como o Gerente de nuvem não interage diretamente com os Coordenadores locais ele não possui uma base própria de avaliações acerca daqueles agentes. Dessa forma, o cálculo da base de avaliações de credibilidade utilizada na Equação 2.7 (Seção 2.5, Cap. 2) não seria viável e somente o valor padrão de credibilidade seria utilizado (T_{DWCr}) para todas as referências. Assim, é proposto aqui um método para calcular a credibilidade dos nós que enviam avaliações acerca do Coordenador Local. Esse método é ilustrado na Figura 4.7 e descrito a seguir:

1. Calcula-se o coeficiente de variação a partir do conjunto de avaliações recebidas pelo Gerente de nuvem conforme a Equação 4.3:

$$C_v = \frac{\sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(\upsilon_i - \overline{\upsilon})^2}}{\overline{\upsilon}}, \tag{4.3}$$

onde $\overline{\upsilon}$ representa a média dos valores υ_i das avaliações. Esse coeficiente é utilizado para verificar se o serviço provido pelo Coordenador local possui uma QoS estável. Instabilidades no serviço podem reduzir, pelo menos na média, a QoS percebida pelos nós. De toda forma, não é interessante do ponto de vista do fornecimento de serviço atender a alguns nós com a QoS esperada e a outros não. Esse coeficiente serve para atestar em que grau esta variabilidade está ocorrendo.

2. Para cada nó testemunha é calculado o desvio entre o valor de sua avaliação em relação a cada um dos valores das avaliações dos demais nós testemunhas conforme a Equação 4.4:

$$\psi_a = \frac{\sum_{i=1}^{N}|\upsilon_a - \upsilon_i|}{N} \tag{4.4}$$

A ideia aqui empregada é comparar a diferença entre as avaliações de QoS colhidas por um nó em relação aos demais nós ligados ao mesmo Coordenador local. O cálculo desse desvio evita que uma avaliação muito negativa de poucos nós tenham grande impacto no resultado final. Por exemplo, um nó pode deliberadamente enviar ao Gerente de nuvem uma avaliação negativa acerca do Coordenador local. Dessa forma, calculando o valor de desvio é possível apontar que trata-se de uma situação onde um nó está fornecendo valores de avaliações discrepantes dos demais nós ligados aos mesmo Coordenador local.

3. A credibilidade de cada nó testemunha é calculada conforme a Equação 4.5:

$$T_{WCr}(a,w) = \begin{cases} e^{-0.8\cdot(\psi_a \cdot C_v)} & \text{se } C_v > 0 \\ T_{DWCr} & \text{se } C_v = 0 \end{cases} \tag{4.5}$$

A credibilidade do nó para o cálculo da confiança em relação ao Coordenador local é feito pela conjugação do coeficiente de variação e do desvio dos valore das avaliações fornecidas pelo nó. Assim, quanto maior a instabilidade em relação à QoS do Coordenador local menor será a credibilidade dos nós vinculados à ele. Isso se dá pois em um ambiente instável é difícil definir se o valor de uma determinada avaliação de um nó representa uma situação que ocorre com frequência ou é somente um caso fortuito. Por outro lado, se o Coordenador local prover um serviço com QoS estável o valor do coeficiente de variação será baixo e desse modo o que produzirá maior impacto na credibilidade do nó será o desvio dos valores das avaliações deste nó em relação aos demais. Isso significa que em um ambiente estável, um nó que apresente valores de avaliação muito discrepantes dos demais nós ligados ao Coordenador local terá uma credibilidade menor. Pois, em se tratando de um ambiente de QoS estável, há uma maior chance dessa avaliação estar enviesada sendo deliberadamente aumentada ou diminuída pelo nó.

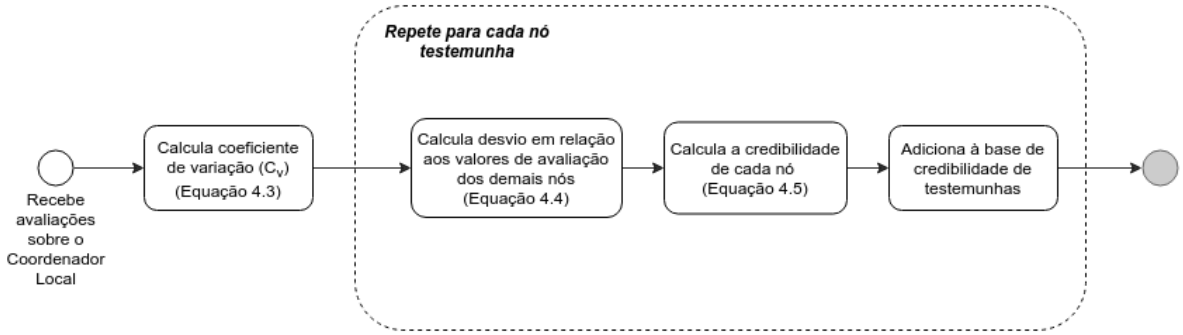

Figura 4.7: Fluxograma de decisão para seleção dos supernós

O valor de credibilidade $T_{WCr}(a, w)$ é utilizado na Equação 2.8, apresentada na Seção 2.5 do Cap. 2, para calcular o peso das avaliações de determinado nó para o cálculo de reputação de testemunhas (lembrando que o cálculo dessa reputação é realizado por meio da Equação 2.10).

Dimensões e contexto

Conforme exposto no início do Capítulo 4, a dimensão utilizada para caracterização da interação entre nós e supernós é a QoS. Apesar de poder englobar diversos parâmetros em seu cálculo, no âmbito desta proposta, a QoS é baseada somente na latência percebida pelos nós usuários durante o consumo do serviço fornecido na fog computacional. Os detalhes do cálculo de QoS para contexto de aplicação da fog computacional fogem do escopo deste proposta. Porém, apesar de basear seu cálculo em diferentes métodos a QoS pode ser abstraída em uma medida de percentual de satisfação do usuário em uma escala de 0 a 1, por exemplo. Assim, no âmbito deste modelo a dimensão utilizada que fica materializada no valor das avaliações definidas pelos nós para o serviço prestado pelos supernós é um valor de satisfação baseado na QoS. O cálculo deste valor foi apresentado na Seção 4.3.

Vieses

Os vieses representam preferências, configurações pré-estabelecidas, pré-conceitos ou condições de hierarquia entre elementos em um modelo de CR. Para o contexto aqui tratado, os nós confiam a priori no Gerente de *cloud*.

Interação direta

A interação direta é materializada pelas avaliações definidas pelos Nós para a QoS provido pelo supernó. A Equação 2.2 é utilizada para o cálculo da confiança baseada nas interações diretas do nó em relação aos supernós.

Inicialização

O Coordenador local calcula um valor de confiança local (ξ) que é utilizado pelos nós como um parâmetro de inicialização para o modelo de confiança e reputação. O conceito por trás desse valor está ligado ao de confiança geral apresentado em Marsh (1994). Ou seja, esse valor de confiança local reflete um grau ou um nível de confiança inicial que um nó pode ter nos demais nós e supernós ligados à um determinado Coordenador local. O nó utiliza esse valor para ajustar a credibilidade padrão (T_{DRCr}) no cálculo da reputação e para decidir qual componente de reputação irá utilizar, certificada ou de testemunhas.

Como visto na Seção 4.2, o Coordenador local conta com a ajuda dos nós para realizar esse cálculo. Os nós realizam o cálculo de desvio nos seu conjunto de avaliações. O cálculo é realizado por meio da Equação 2.4 do modelo FIRE (apresentada no Cap. 2), utilizando o componente de confiança direta ($\mathcal{T}_I(a,b,c)$). Para este caso, porém, o cálculo da confiança direta não é endereçado a um supernó b específico, utilizando, dessa modo, uma base de avaliações $R_I(a,,c)$. Essa base é formada pelas avaliações que foram agrupadas pelos nós dentro do período de histórico de confiança local (η) estipulado pelo Coordenador local, independente de qual o supernó b foi objeto da avaliação. Em posse desses valores de desvio do conjunto de avaliações dos nós, o Coordenador local calcula a variabilidade dentre este conjunto utilizando a Equação 4.6:

$$C_{cl} = \frac{\sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(\rho_R(a,,c)_i - \overline{\rho_R(a,,c)})^2}}{\overline{\rho_R(a,,c)}}, \tag{4.6}$$

onde $\overline{\rho_R(a,,c)}$ representa a média dos valores $\rho_R(a,,c)_i$ dos desvios enviados pelos nós e N o tamanho do conjunto.

O valor final da confiança local é então calculado conforme a Equação 4.7 proposta no presente trabalho.

$$\xi = 1 - C_{cl} \tag{4.7}$$

Avaliação da confiança

A avaliação de confiança é realizada pelo agente Nó, para decidir qual supernó será selecionado e pelo Gerente de nuvem.

Avaliação feita pelo Agente Nó

Depois que todas as informações foram coletadas, sejam de testemunhas, reputação de testemunhas ou interação direta, e feita a avaliação de reputação, o Nó deve ser capaz de obter um valor de confiança para embasar sua decisão sobre qual supernó deve ser selecionado. Para isso é utilizada a Equação 2.11 do modelo FIRE, descrita na Seção 2.5 do Capítulo 2. A partir dela, o nó obtém um valor final para a confiança de cada supernó candidato. A Figura 4.9 apresenta os principais passos utilizados para a realização deste cálculo.

Por ser um agente orientado a utilidade, a decisão de qual supernó será selecionado tem por objetivo maximizar a utilidade esperada. Conforme explicado na Seção 2.3 do Capítulo 2, o agente seleciona a ação que maximiza a Equação 2.1. Neste caso, a ação refere-se à qual supernó será selecionado. A seguinte função é utilizada para o cálculo da

utilidade:

$$U(s) = 1 - e^{-x \cdot \mu},$$

onde x ($x \in [0, 1]$) representa a QoS percebida pelo Nó em uma interação. O parâmetro μ é utilizada para ajustar a função de utilizada de forma a refletir a nível de QoS requisitado pelo nó. Isso significa que ao configurar esse parâmetro, o nó busca refletir o impacto de uma QoS baixa. Em outra visão, esse valor reflete o perfil de risco do nó. Por exemplo, quando um serviço consumido necessita de um nível de perda de pacotes de 5%, o parâmetro μ poderia ser ajustado para $\mu \leq 10$ resultando em gráfico com uma inclinação mais pronunciada (Figure 4.8).

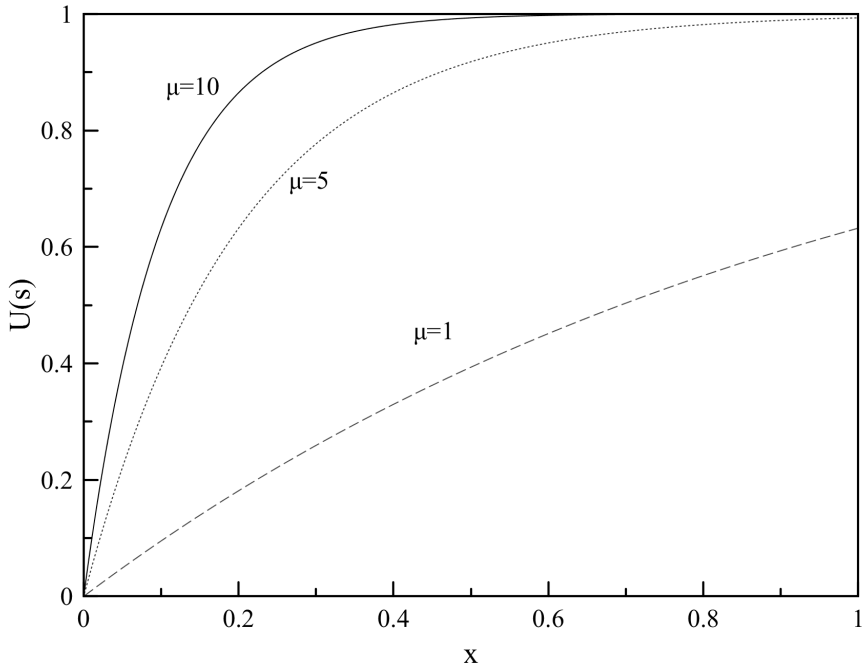

Figura 4.8: Função de utilidade para diferentes valores de μ

A probabilidade $P(Result(a) = s'|a, e)$ da Equação 2.1 é calculada com base na confiança obtida pelo nó em relação ao supernó.

Avaliação feita pelo Gerente de nuvem

O Gerente de nuvem utiliza o componente de reputação de testemunhas do modelo FIRE para calcular a confiança em relação aos coordenadores locais na topologia de implantação **Controlada** e **Parcialmente aberta** e em relação aos supernós na topologia de implantação **Fixa**. Em ambos os casos a Equação 2.10 é aplicada para calcular o valor da reputação. Esse valor de reputação calculado é aplicado na Equação 2.11 do modelo FIRE que calcula o valor geral de confiança.

Conforme explicado no tópico sobre **Credibilidade e confiabilidade** desta seção, a função de peso $\omega_W(r_i)$ é ajustada pois o Gerente de nuvem não interage diretamente com o Coordenadores locais e com os supernós, portanto não tem uma base de avaliações próprias que podem ser comparadas com as avaliações enviadas pelas testemunhas.

O valor obtido da Equação 2.11 é comparada com um limiar de confiança pré-estabelecido pelo Gerente de nuvem (como pode ser visto na linha 9 do Algoritmo 3 e na linha 8 do Algoritmo 1). Esse limiar reflete a qualidade esperada para o Coordenador local ou para o supernó.

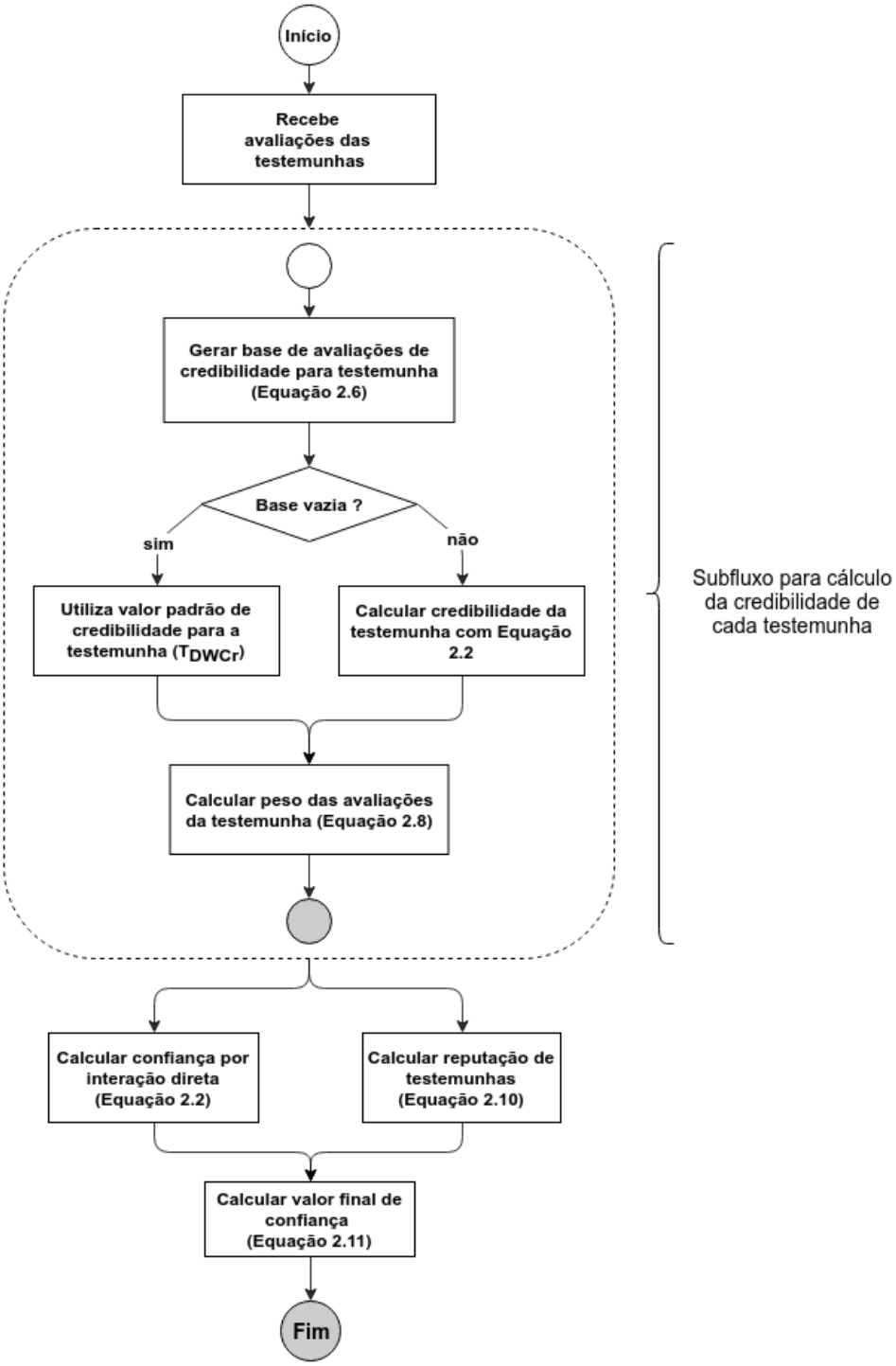

Figura 4.9: Fluxograma para cálculo da confiança

4.4 Modelo implementacional

Os agentes do modelo definido neste trabalho foram implementados em um SMA utilizando o *framework* JADE (conforme apresentado na Seção 2.4.4). A Figura 4.10 apresenta uma visão geral da arquitetura com foco na comunicação entre os elementos do modelo implementacional. Essa arquitetura está dividida em uma camada composta pelos agentes do SMA e outra composta pelos componentes do serviço que é provido na fog computacional. Os componentes da camada de serviço da fog interagem com os agentes na camada do SMA. O Cliente do serviço é o componente executada no dispositivo final do usuário do serviço que está sendo provido na fog. Esse componente interage com o agente Nó com vistas a obter uma opção viável de *Host* de serviço local. Para tanto, o Cliente de serviço fornece ao Nó os dados relativos ao consumo do serviço. É a partir desses dados que o Nó gera as avaliações relativas aos supernós utilizadas no processo de seleção (conforme descrito na Seção 4.3). O *Host* de serviço é o componente que atua como servidor local na fronteira da fog e é instanciado pelo supernó para atender aos Clientes de serviço. De maneira análoga o Servidor de nuvem é o componente que atua como servidor para o serviço provido na fog, porém, como o nome revela, este componente fica implantado em um *datacenter* de nuvem.

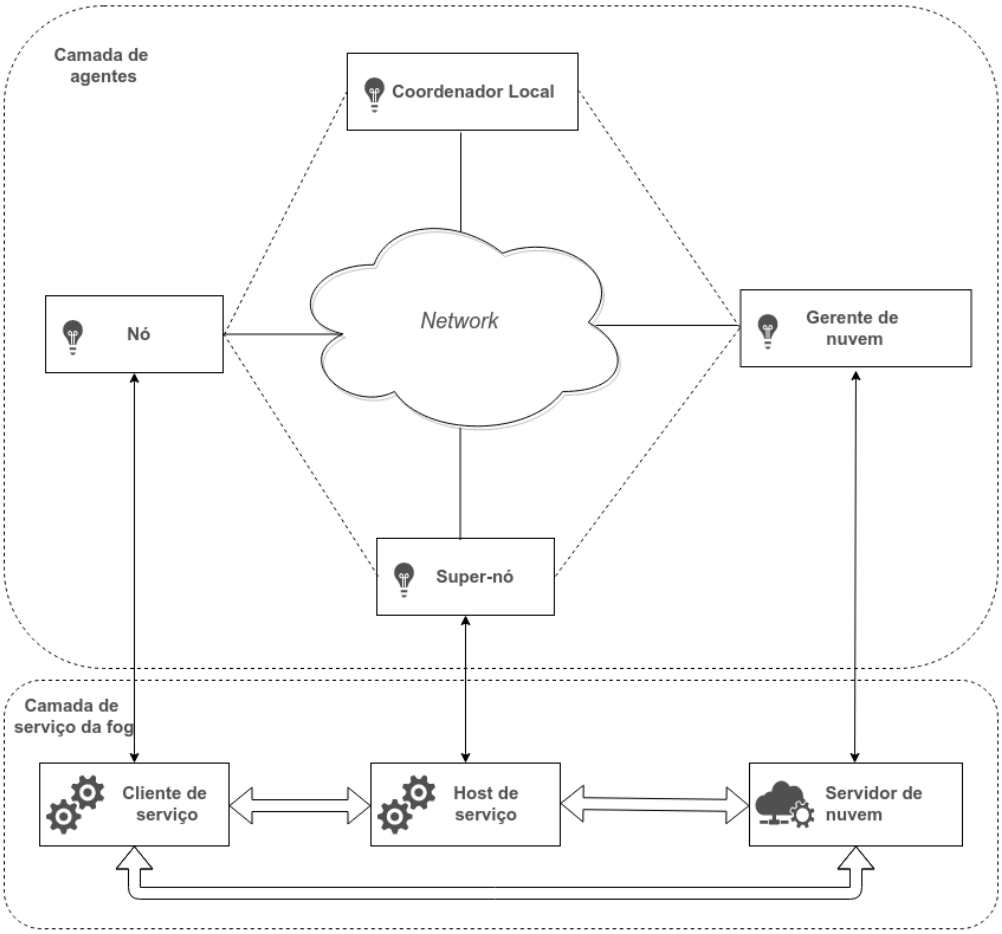

Figura 4.10: Comunicação entre os elementos do modelo implementacional

A Figura 4.11 apresenta o diagrama contendo as classes implementadas no JADE. O Gerente de nuvem é implementado pela classe *CloudManagerAgent*, o Coordenador local pela *LocalCoordinatorAgent*, o supernó pela classe *SupernodeAgent* e o nó pela classe *NodeAgent*. A classe *LocalCoordinatorAgent* extende a classe *CloudManagerAgent* herdando assim os atributos referentes aos registros de nós e supernos (*nodeRegistry* e *supernodeRegistry*, respectivamente). Porém, o *LocalCoordinatorAgent* não tem acesso ao método *calculateLocalCoordinatorTrust(ratings)* que realiza o cálculo de confiança em relação aos Coordenador locais, pois essa atribuição é exclusiva do Gerente de nuvem. No mesmo sentido, a classe *SupernodeAgent* extende a classe *NodeAgent*, pois, conforme explicado na Seção 4.2, o supernó pode ser também um nó, dessa forma, todas os atributos do *NodeAgent* também são acessados pelo *SupernodeAgent*.

Os relacionamentos representados na Figura 4.11 são referentes à associação entre os agentes no âmbito da plataforma JADE. Essa associação é feita somente pelo armazenamento da identificação única de cada agente no JADE. A associação entre *SupernodeAgent* e *NodeAgent* com o *LocalCoordinatorAgent* é de um para muitos, demostrando que para cada *SupernodeAgent* e *NodeAgent* estão vinculados a um *LocalCoordinatorAgent*. A associação entre *SupernodeAgent*, *NodeAgent* e *LocalCoordinatorAgent* com o *CloudManagerAgent* também possui essa mesma cardinalidade, um para muitos. A associação entre *SuperAgent* também é de um para muitos em relação aos *NodeAgent*, dado que a um supernó podem estar ligados vários nós e cada nó está ligado somente a um supernó por vez.

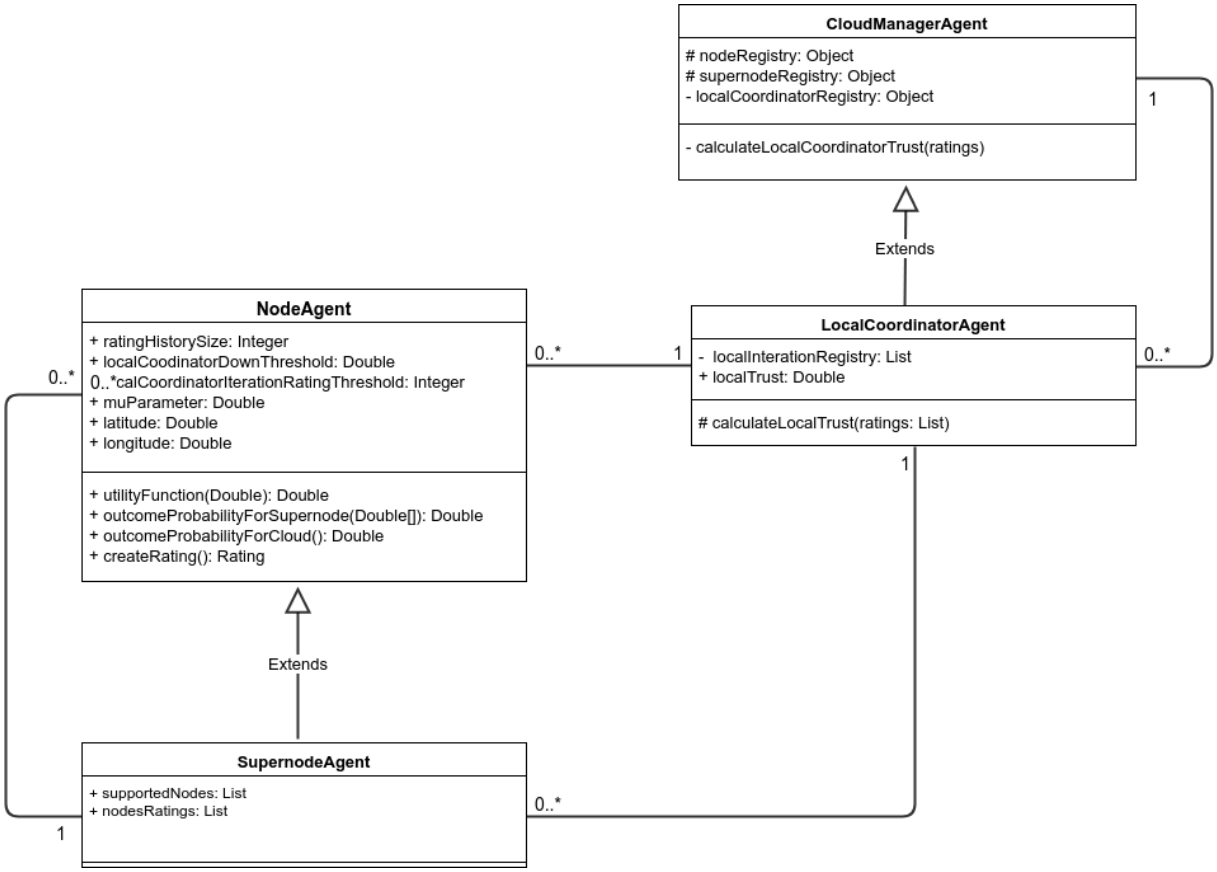

Figura 4.11: Diagrama de classes do modelo implementacional

A Figura 4.12 apresenta a implantação dos componentes do SMA. É utilizada a plataforma distribuída do JADE para realizar a comunicação entre os elementos do sistema. O *Main Container* do JADE é implantado no servidor de nuvem e nele é executado o Gerente de nuvem (representado na Fig. 4.12 pelo *CloudManagerAgent*). Os demais agentes são executados nos *Containers* distribuídos do JADE, ficando cada *Container* implantado em um dispositivo integrante da fog computacional. Esses *Containers* distribuídos estão conectados ao *Main Container* na nuvem. Assim, durante a execução do sistema a comunicação entre os agentes é transparente. Para um agente nó enviar uma mensagem para outro, ele deve possuir somente a identificação desse na plataforma de comunicação do JADE. Essa identificação é estipulada para cada agente no momento em que a classe é instanciada no *Container* do JADE, sendo única para cada um no contexto da plataforma.

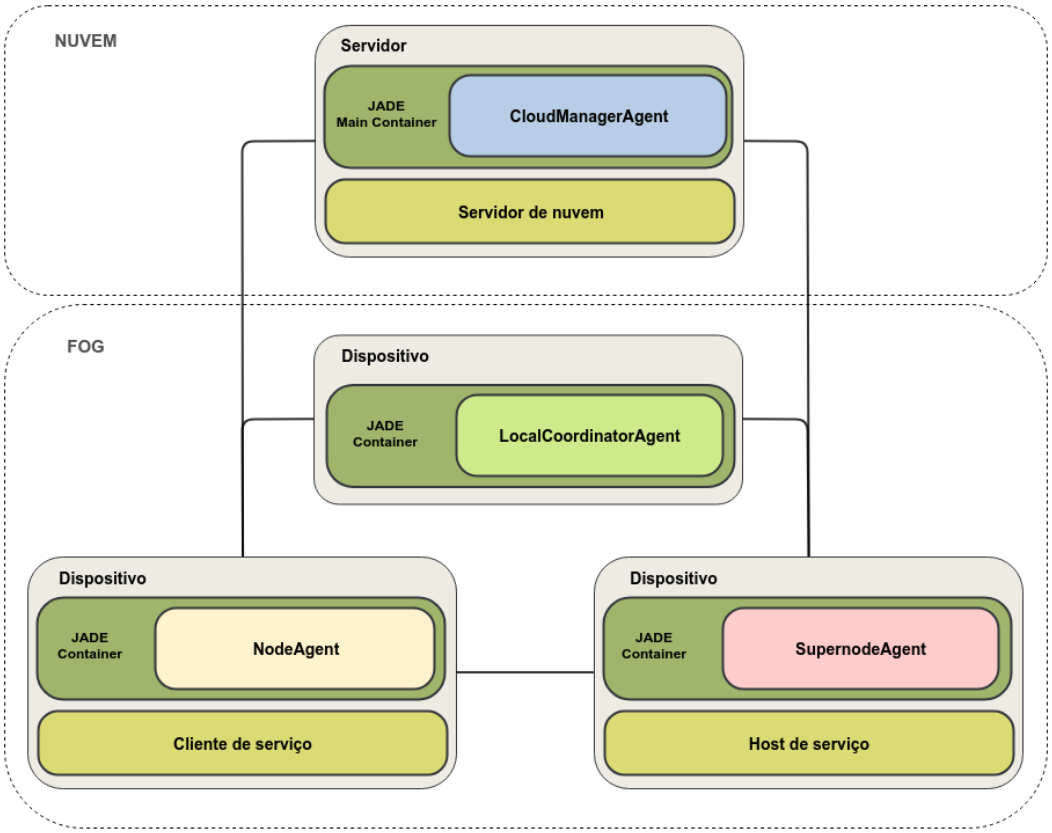

Figura 4.12: Diagrama de implantação do modelo implementacional. Os agentes são executados na plataforma JADE. Os componentes de serviço da fog computacional são executados no mesmo dispositivo que o *Container* JADE.

O *CloudManagerAgent*, ao ser iniciado, envia uma mensagem para se registrar junto ao agente DF que provê o serviço de Páginas Amarelas no JADE. A classe *ServiceDescription* é um descritor do serviço provido pelo *CloudManagerAgent* para ser registrado nas Páginas Amarelas. São atribuídos o tipo (*"cloud-manager"*), o endereço IP e a porta do servidor de nuvem, e a latitude e longitude desse servidor. Quando os demais agentes do sistema são iniciados eles buscam junto ao DF a identificação do *CloudManagerAgent* utilizando o tipo *"cloud-manager"*.

A Figura 4.13 apresenta as performativas de comunicação utilizadas pelos agentes *NodeAgent*, *SupernodeAgent* e *LocalCoordinatorAgent* no fluxo de consumo de serviço ilustrado na Figura 4.4 da Seção 4.2. O *NodeAgent* utiliza a performativa *subscribe* para se registrar junto ao *LocalCoordinatorAgent* e a performativa *request* para solicitar os supernós disponíveis (Passo 3 da Fig. 4.4). A performativa *request* também é utilizada pelo *NodeAgent* para solicitar o serviço para o *SupernodeAgent* (Passo 5 da Fig. 4.4). Da mesma forma, o *SupernodeAgent* utiliza a performativa *subscribe* para se registrar junto ao *LocalCoordinatorAgent* como um supernó. Essa interação do *SupernodeAgent* envolve o envio dos dados de registro na mensagem identificada pela performativa *subscribe*. Esses dados estão apresentados na Tabela 4.1. São esses mesmos dados que são enviados pelo *LocalCoordinatorAgent* para o *NodeAgent* juntamente com a performativa *inform* como resposta à performativa *request*.

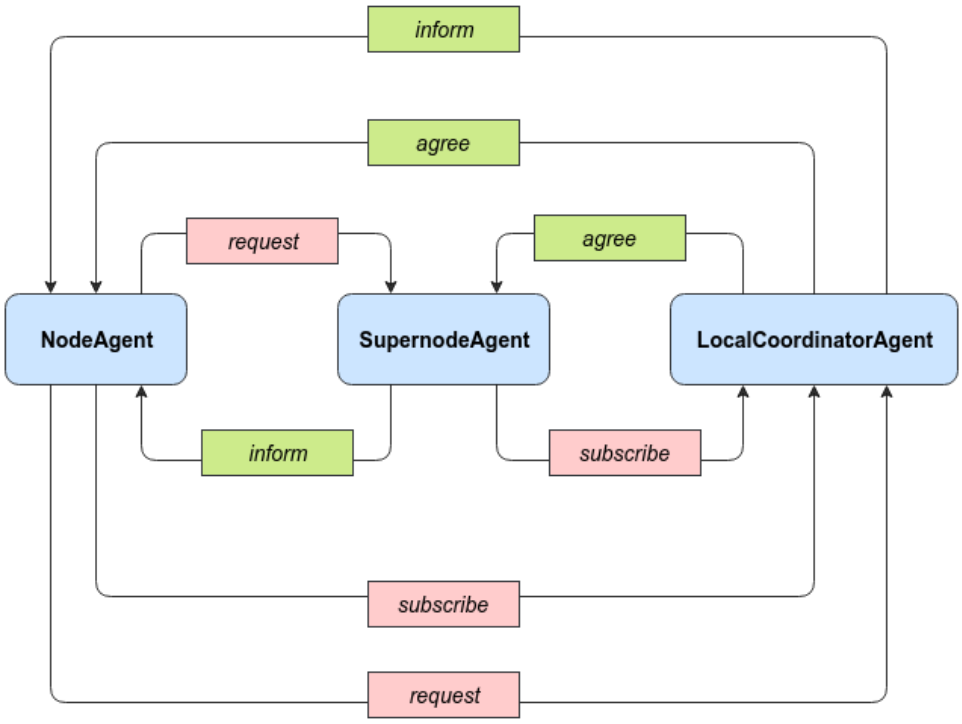

Figura 4.13: Performativas utilizadas pelos agentes no fluxo de consumo de serviço.

Tabela 4.1: Informações de registro para os supernós

capacity	A quantidade de nós que são suportados por esse supernó.
latitude	A latitude da geolocalização do dispositivo onde o supernó é executado.
longitude	A longitude da geolocalização do dispositivo onde o supernó é executado.
hostPort	Porta utilizada pelo host de serviço em que o cliente de serviço pode se conectar.
hostIP	IP utilizado pelo host de serviço.

No Capítulo 5 serão apresentados os experimentos realizados no ambiente simulado no Peersim e no ambiente real do PlanetLab, juntamente com a análise dos resultados obtidos.

Capítulo 5

Experimentos

Os experimentos para a avaliação do modelo proposto no Capítulo 4 são definidos no contexto de jogos on-line, mais especificamente de *Cloud Gaming*. Assim, no cenário proposto para os experimentos os nós representam os usuários que se conectam ao um servidor na nuvem para jogar um jogo do tipo MMOG. Os experimentos foram realizados utilizando o ambiente simulado do Peersim (Montresor and Jelasity, 2009b) e o ambiente real do Planetlab(Chun et al., 2003). De maneira geral, o cenário de experimentação é similar ao apresentado em Lin and Shen (2017) com a diferença que para o estudo de caso aqui apresentado foram utilizadas os modelos de implantação **Controlado** e **Parcialmente aberto**, introduzindo, dessa forma, a figura do Coordenador Local. Para fins de comparação foram utilizados no ambiente simulado os parâmetros apresentados nos experimentos do trabalho de Lin and Shen (2017).

Dessa forma, o experimento aqui proposto é dividido em 28 ciclos, onde cada ciclo representa um dia de jogo e é subdividido em 24 subciclos, representando as horas de jogo. Os 28 ciclos representam a simulação de um mês de jogo (quatro semanas). A cada ciclo os nós, que representam nesse caso os usuários, se conectam ao servidor e jogam por um período variável de subciclos. Os nós são ativados entre os subciclos 1 e 19, com probabilidade de 30%, e entre os subciclos 20 e 24, com probabilidade de 70%, refletindo os horários de pico de consumo de serviços *online* conforme apresentado em Lin and Shen (2017). Em cada ciclo, 50% dos nós ficam ativos por 2 subciclos, 30% ficam ativos por 3 subciclos e os 20% restantes ficam ativos por um período de até 19 subciclos.

O parâmetro de QoS é definido com base na perda de pacotes. Um pacote é considerado perdido quando chega ao dispositivo usuário com uma latência superior à requisitada. A quantidade de pacotes perdidos afeta diretamente a QoS afetando, desse modo, a satisfação do usuário (Lin and Shen, 2017). Neste experimento o percentual de usuário satisfeitos com o serviço é utilizado para efeitos de avaliação. Nesse sentido, um usuário é considerado satisfeito se 95% dos seus pacotes são recebidos na latência requisitada. Esse percentual é medido à partir da proporção entre o total de pacotes recebidos na latência esperada e o total de pacotes enviados. O percentual de usuário satisfeitos é medido a cada subciclo e depois agrupado em um valor por ciclo.

5.1 Ambiente simulado

Esta primeira etapa do estudo de caso utiliza o ambiente simulado do Peersim. O objetivo principal é avaliar a capacidade do modelo de confiança e reputação em ajudar os nós a selecionar supernós que forneçam um serviço com a QoS esperada. Assim, não são utilizados os agentes implementados na plataforma JADE, mas somente a parte de suas funcionalidades que estão ligadas à aplicação do modelo de confiança e reputação, como o cálculo da utilidade esperada.

Foi definida uma estrutura no Peersim contendo um servidor de nuvem (representados por um Gerente de *Cloud*), um Coordenador local, 1000 nós e 99 supernós. A capacidade dos supernós segue um distribuição de pareto com parâmetro $\alpha = 2$, conforme apresentado em Lin and Shen (2017). Os nós utilizam uma escala de recência $\lambda = -5/ln(0.5)$, assim, uma diferença de cinco ciclos entre as avaliações gera uma diferença de peso de 0.5.

A latência entre cada um dos elementos foi escolhida aleatoriamente de uma base de *logs* de latência coletados para o jogo *League of Legends* (2018), ponderada pela quantidade de ocorrências registradas para cada valor de latência. Dessa forma, é feita uma distribuição das latências proporcional à quantidade de nós presentes na simulação (Lin and Shen, 2017). Em cada ciclo os nós escolhem a latência requisitada de forma aleatório a partir do conjunto: $\{110, 90, 70, 50, 30\}$ (Lin and Shen, 2017). Ademais, 1/10 e 1/5 dos supernós são escolhidos aleatoriamente para reduzirem sua capacidade em 80% e 50%, respectivamente, com 50% de probabilidade a cada ciclo (Lin and Shen, 2017). A Tabela 5.1 apresenta um resumo dos parâmetros do experimento no ambiente simulado do Peersim.

São testados três cenários: (1) os nós selecionam os supernós aleatoriamente; (2) os nós utilizam um cálculo de confiança direta, conforme utilizado em Lin and Shen (2017); e (3) o modelo de confiança e reputação proposto neste trabalho é utilizado. Nos três cenários, os 13 primeiros ciclos são utilizados para inicialização e acumulação de avaliações. Dessa forma, somente à partir do 14 ciclo que o cálculo de confiança direta (cenário 2) e o modelo proposto (cenário 3) começam a ser aplicados pelos nós. Após o 20º ciclo 300 novos nós são introduzidos na rede.

O Gráfico 5.1 apresenta os resultados coletados para o experimento no Peersim. Após o 13º a utilização da confiança direta e do modelo proposto resulta em um aumento no percentual do nós satisfeitos, pois os nós começam a identificar quais os supernós não estão atendendo à QoS esperada. O modelo proposto, por utilizar o componente de reputação, apresenta um percentual de melhoria mais acentuado. Ele permite que os nós

Tabela 5.1: Parâmetros para o ambiente simulado

Parâmetro	**Valor**
Número de Gerentes de nuvem	1
Número de Coordenadores locais	1
Número de nós	1000
Número de supernós	99
Parâmetro distrib. de pareto (α)	2
Latência requisitada (ms)	{110, 90, 70, 50, 30}
Escala de recência (λ)	5

compartilhem suas avaliações e possam tanto se ligar a supernós com os quais ainda não interagiram, quanto ter mais informação para evitar os supernós que não estão mantendo a QoS esperada. Após a introdução dos novos nós na rede, no 20º ciclo, há um queda no percentual de QoS, pois esses nós não possuem informações acerca dos supernós. Porém, essa queda percentual é menos acentuada no cenário em que o modelo proposto é utilizado, pois a reputação de testemunhas auxilia esses novos nós à selecionar supernós que estão mantendo a QoS, o que não ocorre no cenário 2, pois utilizando a confiança direta o nó utiliza somente suas próprias avaliações para realizar a seleção.

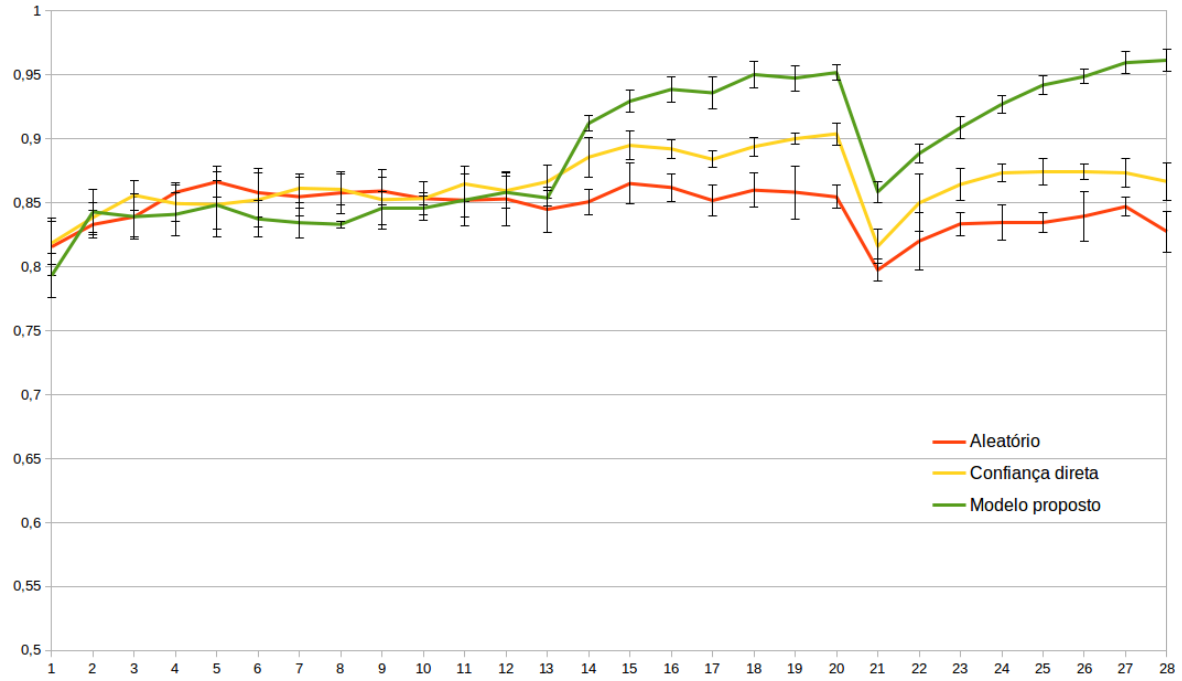

Figura 5.1: Resultados do experimento no Peersim

5.2 Ambiente real

Esta etapa dos experimentos utiliza o ambiente real de avaliação fornecido pelo PlanetLab. O objetivo desta etapa do estudo de caso é avaliar a implementação dos agentes feita na plataforma JADE conforme apresentado na Seção 4.4.

Para o primeiro teste foi definida uma estrutura da fog computacional contendo 46 nós dos quais 10 são selecionados aleatoriamente para serem utilizados como supernós. Ademais, foram definidos um Gerente de nuvem e três Coordenadores locais. Os agentes foram implantados utilizando 28 nós do PlanetLab.

Assim como no cenário simulado a capacidade dos supernós segue um distribuição de pareto com parâmetro $\alpha = 2$. O requisito de latência estipulado pelos nós é escolhido aleatoriamente do conjunto $\{30, 40, 50, 60, 70\}$. Esse conjunto foi definido à partir das latências reais medidas entre os nós de processamento do PlanetLab. De maneira análoga ao experimento no ambiente simulado, neste experimento seis supernós são selecionados aleatoriamente para diminuírem suas capacidades com 70% de probabilidade a cada ciclo.

Os supernós diminuem suas capacidades para um percentual que varia no intervalo de 10% a 30%. Esses parâmetros do experimento no ambiente real são resumidos na Tabela 5.2.

Tabela 5.2: Parâmetros para o ambiente real

Parâmetro	**Valor**
Número de Gerentes de nuvem	1
Número de Coordenadores locais	3
Número de nós	36
Número de supernós	10
Parâmetro distrib. de pareto (α)	2
Latência requisitada (ms)	[30, 40, 50, 60, 70]
Número de supernós que diminuem a capacidade	6
% que a capacidade dos supernós é diminuída	[10, 30]

São utilizados dois cenários de avaliação neste ambiente, um aleatório e outro que utiliza o modelo proposto. Da mesma forma que no ambiente simulado, os 14 primeiros ciclos são utilizados para a inicialização e acumulação de avaliações. A Figura 5.2 apresenta os resultados coletados para o percentual de usuário satisfeitos a cada ciclo no PlanetLab.

A partir do ciclo 14, quando os nós começam a selecionar os supernós utilizando o modelo de confiança e reputação, pode-se notar que há uma modificação no percentual de nós satisfeitos, elevando-se esse número e mantendo-o mais estável. Isso mostra que com o modelo de confiança e reputação os nós são capazes de evitar os supernós que apresentam falhas mais frequentemente mantendo os requisitos de QoS.

A utilização do modelo proposto mostra uma diferença de mais de 20% no percentual de nós satisfeitos. Pensando no contexto de um jogo que conta com 100.000 usuários, por exemplo, isso representaria um grupo de 20.000 usuários que poderiam ser afetados.

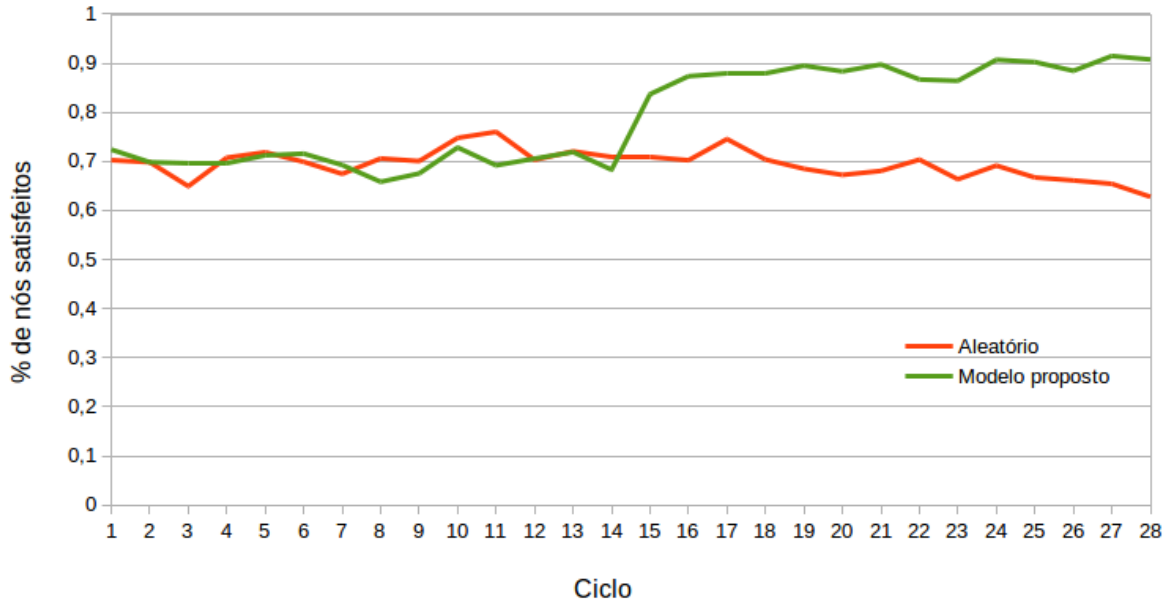

Figura 5.2: Comparação de percentual de satisfação de nós para cenário aleatório e com modelo de confiança e reputação - experimento no PlanetLab

A Figura 5.3 apresenta uma comparação entre o tempo de execução dos nós para o cenário aleatório e com o modelo proposto. Esse tempo foi medido no intervalo entre a solicitação da lista de supernós disponíveis feita pelo nó (passo 3 da Fig 4.4) e a inicialização do cliente de serviço de fog feita após o supernó ter aceitado a solicitação de serviço. Os valores apresentados refletem a média do tempo de execução dos nós. Essa média é colhida em milissegundos e depois é feita a comparação percentual em proporção a 1 minuto (60000ms). O gráfico mostra os percentuais para cada cenário em escala logarítmica. A diferença entre a utilização do modelo e o cenário aleatório em relação à 1 minuto é muito pequena mostrando que o *overhead* imposto pela utilização do modelo pode ser absorvido pelos nós sem ocasionar em um grande impacto no desempenho.

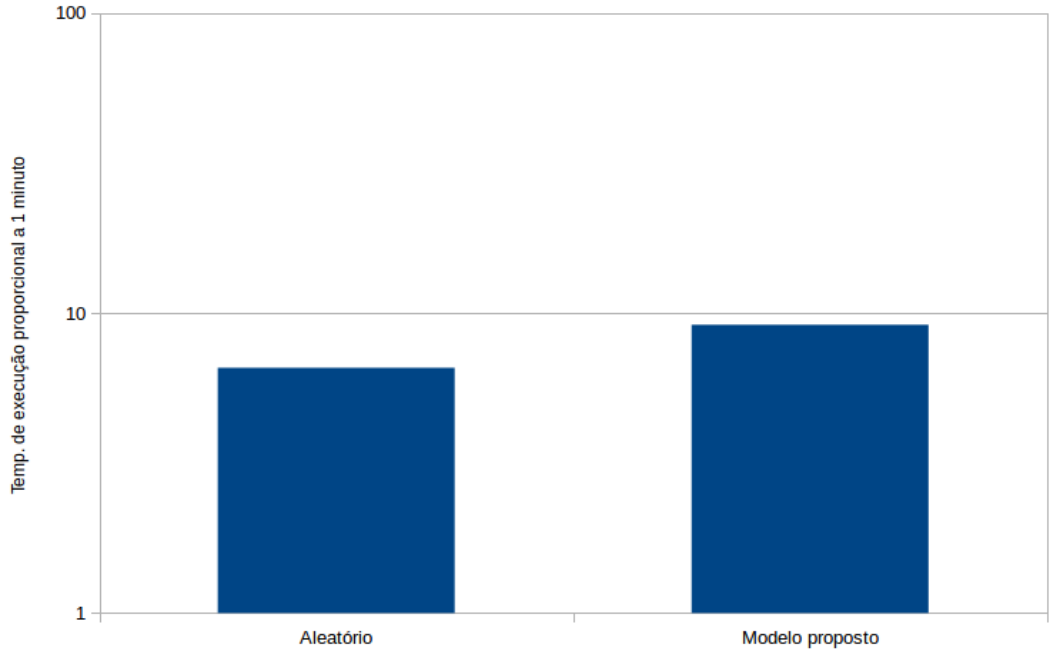

Figura 5.3: Percentual do tempo de execução em relação a um minuto para os cenários aleatório e com o modelo de confiança e reputação (em escala logarítmica)

Capítulo 6

Conclusões

Este trabalho propõe um modelo baseado em agentes para a escolha de supernós com uso de confiança e reputação em um ambiente de fog computacional. Foi apresentado um modelo composto por quatro agentes que interagem para facilitar a seleção desses supernós. O modelo proposto é flexível e pode ser utilizado em diferentes topologias de implantação da fog computacional. Porém, essa flexibilidade ainda é estática, ou seja, o modelo é configurado para cada topologia específica previamente à sua execução. Ou seja, ainda não há um mecanismo que permita aos agentes compreender o ambiente e se adaptar em uma topologia que seja mais ajustada ao cenário de fog em que estão inseridos.

O estudo de caso realizado mostrou que a utilização da confiança e reputação para a seleção de supernós garante um melhor nível de satisfação dos nós do serviço que está sendo provido na fog computacional. Esse estudo utilizou um contexto específico que simulava um padrão de utilização de serviço análogo à um jogo online.

A utilização do modelo proposto apresentou um *overhead* no tempo de execução, como seria esperado dado o aumento no volume de comunicações e de processamento principalmente para o cálculo da confiança e reputação dos supernós. Esse *overhead*, entretanto, não se mostrou muito elevado podendo ser absorvido pelo tempo de inicialização em aplicações do tipo da que foi avaliada no estudo de caso. Outras aplicações, como as que tratam com aparelhos móveis com pouca capacidade de processamento e comunicação limitada poderiam ser mais afetadas por esse aumento no volume de comunicações e de processamento. Isso poderia ser contornado por meio da utilização de *offloading* de parte deste processamento para outros elementos do modelo. Esse *offloading* poderia ser feito do nó para o Coordenador Local, por exemplo. Além disso, há espaço para melhorias no código com vistas a aumento de performance na execução dos agentes, principalmente com a inclusão de algum tipo de banco de dados de cache que seria utilizados pelos serviços de registro de nós, supernós e Coordenadores Locais. Apesar desse *overhead*, a utilização do modelo proposto mostra-se válida, dada a diferença de mais de 20% no percentual de nós satisfeitos.

Assim, pode-se considerar que o objetivo geral proposto para este trabalho foi alcançado e em consequência a questão originária à esse objetivo foi respondido. Ou seja, a proposição do modelo baseado em agentes com uso de confiança e reputação auxiliou na seleção de supernós na fog computacional de modo a garantir a QoS. Os experimentos realizados mostram que a arquitetura definida e implementada deste modelo consegue apresentar resultados com melhoria da QoS percebida pelos usuários, tanto nas avaliações

no ambiente simulado, quanto no ambiente real.

Outros experimentos são necessários para avaliar mais profundamente o modelo, principalmente quanto a sua flexibilidade em diferentes cenários de utilização da fog computacional.

Diversos pontos tratados na proposta de modelo podem ser aprofundados. Pode-se tomar como base os desafios e questões propostas por Yi et al. (2015) e Dastjerdi et al. (2016b), principalmente no tocante à segurança no contexto de fog computacional. Porém, para o modelo aqui apresentado, os principais avanços que poderiam ser feitos são:

- execução de testes utilizando outros contextos de aplicação da fog computacional.
- aplicação do aspecto de adaptação do meta-modelo de C&R de Hoelz (2013) para ajustar parâmetros do modelo de confiança e reputação de acordo com o as variações do ambiente.
- inclusão de outros modelos de racionalidade. O modelo BDI (*Belief-Desire-Intention*) poderia ser aplicado no Gerente de Nuvem de modo a equipá-lo com um modelo racional mais completo.
- inclusão de mecanismos e abordagens de negociação em SMA para incentivar a ativação de supernós e Coordenadores Locais mas mantendo a eficiência do ponto de vista de custo para o provedor de serviço que utiliza o modelo de fog.

Referências

(2018). Latency (lag) vs win rate in League of Legends. https://www.reddit.com/r/dataisbeautiful/comments/1t23a0/latency_lag_vs_win_rate_in_league_of_legends_oc/. Online; acessado em Junho de 2018. 73

Alam, M. G. R., Tun, Y. K., and Hong, C. S. (2016). Multi-agent and reinforcement learning based code offloading in mobile fog. In *2016 International Conference on Information Networking (ICOIN)*, pages 285–290. 9, 41, 46, 47

Ardagna, D., Casale, G., Ciavotta, M., Pérez, J. F., and Wang, W. (2014). Quality-of-service in cloud computing: modeling techniques and their applications. *Journal of Internet Services and Applications*, 5(1):11. 7, 8

Armbrust, M., Fox, A., Griffith, R., D. Joseph, A., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., and Zaharia, M. (2009). Above the clouds: A berkeley view of cloud computing. 28. 5, 6

Bellifemine, F. L., Caire, G., and Greenwood, D. (2007). *Developing Multi-Agent Systems with JADE (Wiley Series in Agent Technology)*. John Wiley & Sons. x, 26, 28

Bittencourt, L. F., Diaz-Montes, J., Buyya, R., Rana, O. F., and Parashar, M. (2017). Mobility-aware application scheduling in fog computing. *IEEE Cloud Computing*, 4(2):26–35. 9, 11, 41, 42, 46, 47

Bonomi, F., Milito, R., Zhu, J., and Addepalli, S. (2012). Fog computing and its role in the internet of things. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, MCC '12, pages 13–16, New York, NY, USA. ACM. 2, 8, 13

Bordini, R. H. and Hübner, J. F. (2007). Jason – a java-based interpreter for an extended version of agentspeak. 26

Bratman, M. (1987). *Intention, Plans, and Practical Reason*. Center for the Study of Language and Information. 26

Braubach, L., Lamersdorf, W., and Pokahr, A. (2003). Jadex: Implementing a bdi-infrastructure for jade agents. 29

Braubach, L., Pokahr, A., and Lamersdorf, W. (2004). Jadex: A short overview. In *In Main Conference Net.ObjectDays 2004*. 29

Buyya, R., Broberg, J., and Goscinski, A. M. (2011). *Cloud Computing Principles and Paradigms*. Wiley Publishing. x, 5, 6, 7

Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., and Brandic, I. (2009). Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Gener. Comput. Syst.*, 25(6):599–616. 5, 8

Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A. F., and Buyya, R. (2011). Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw. Pract. Exper.*, 41(1):23–50. 16

Chun, B., Culler, D., Roscoe, T., Bavier, A., Peterson, L., Wawrzoniak, M., and Bowman, M. (2003). Planetlab: An overlay testbed for broad-coverage services. *SIGCOMM Comput. Commun. Rev.*, 33(3):3–12. 17, 39, 72

Dastjerdi, A. V. and Buyya, R. (2016). Fog computing: Helping the internet of things realize its potential. *Computer*, 49(8):112–116. 2, 8, 13, 14

Dastjerdi, A. V., Gupta, H., Calheiros, R. N., Ghosh, S. K., and Buyya, R. (2016a). Fog computing: Principals, architectures, and applications. 8

Dastjerdi, A. V., Gupta, H., Calheiros, R. N., Ghosh, S. K., and Buyya, R. (2016b). Fog computing: Principles, architectures, and applications. *CoRR*, abs/1601.02752. 9, 13, 15, 78

Ericsson, A. (2017). Ericsson mobility report. *Ericsson, Sweden, Tech. Rep. EAB-17*, 1. 1

Finin, T., Fritzson, R., McKay, D., and McEntire, R. (1994). Kqml as an agent communication language. In *Proceedings of the Third International Conference on Information and Knowledge Management*, CIKM '94, pages 456–463, New York, NY, USA. ACM. x, 23, 24

FIPA (2002). FIPA ACL message structure specification. FIPA agent communication language specifications. 24

Foundation for Intelligent Physical Agents (2002). FIPA Communicative Act Library Specification. http://www.fipa.org/specs/fipa00037/SC00037J.html. Online; acessado em 26 de Junho de 2018. x, 25

Gosling, J., Joy, B., Steele, G. L., Bracha, G., and Buckley, A. (2014). *The Java Language Specification, Java SE 8 Edition*. Addison-Wesley Professional, 1st edition. 16

Gupta, H., Dastjerdi, A. V., Ghosh, S. K., and Buyya, R. (2016). ifogsim: A toolkit for modeling and simulation of resource management techniques in internet of things, edge and fog computing environments. *CoRR*, abs/1606.02007. 16, 44, 46

He, Q., Zhang, C., Ma, X., and Liu, J. (2017). Fog-based transcoding for crowdsourced video livecast. *IEEE Communications Magazine*, 55(4):28–33. 10, 39, 40, 44, 45, 46, 47, 57

Hoelz, B. W. P. (2013). *Metamodelo para adaptação de confiança e reputação em sistemas multiagente dinâmicos*. PhD thesis, Departamento de Engenharia Elétrica, Faculdade de Tecnologia, Universidade de Brasília. x, 30, 31, 32, 33, 34, 36, 45, 59, 78

Hoelz, B. W. P. and Ralha, C. G. (2015). Towards a cognitive meta-model for adaptive trust and reputation in open multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 29(6):1125–1156. 58

Huynh, T. D. (2006). *Trust and reputation in open multi-agent systems*. PhD thesis, University of Southampton. 29, 30, 32, 33, 45, 53, 58, 59, 60, 61

Hübner, J. and Sichman, J. (2000). Saci: Uma ferramenta para implementação e monitaração da comunicação entre agentes. 26

INTERCHANGE, K. (1998). The darpa knowledge sharing effort: Progress report. *Readings in agents*, page 243. 23

Iotti, N., Picone, M., Cirani, S., and Ferrari, G. (2017). Improving quality of experience in future wireless access networks through fog computing. *IEEE Internet Computing*, 21(2):26–33. 9, 11, 14, 41

Jennings, N. R., Sycara, K., and Wooldridge, M. (1998). A roadmap of agent research and development. *Autonomous Agents and Multi-Agent Systems*, 1(1):7–38. 21, 22

Jøsang, A., Ismail, R., and Boyd, C. (2007). A survey of trust and reputation systems for online service provision. *Decis. Support Syst.*, 43(2):618–644. 30

Lin, Y. and Shen, H. (2017). Cloudfog: Leveraging fog to extend cloud gaming for thin-client mmog with high quality of service. *IEEE Transactions on Parallel and Distributed Systems*, 28(2):431–445. 10, 13, 38, 44, 45, 46, 47, 57, 72, 73

Mahmud, R., Kotagiri, R., and Buyya, R. (2018). *Fog Computing: A Taxonomy, Survey and Future Directions*, pages 103–130. Springer Singapore, Singapore. 2, 8

Marsh, S. P. (1994). *Formalising trust as a computational concept*. PhD thesis. 29, 30, 32, 45, 64

Mayfield, J., Labrou, Y., and Finin, T. (1996). Evaluation of kqml as an agent communication language. In Wooldridge, M., Müller, J. P., and Tambe, M., editors, *Intelligent Agents II Agent Theories, Architectures, and Languages*, pages 347–360, Berlin, Heidelberg. Springer Berlin Heidelberg. 23

Mell, P. M. and Grance, T. (2011). The nist definition of cloud computing. Technical report, Gaithersburg, MD, United States. 5

Montresor, A. and Jelasity, M. (2009a). Peersim: A scalable p2p simulator. In *2009 IEEE Ninth International Conference on Peer-to-Peer Computing*, pages 99–100. 16

Montresor, A. and Jelasity, M. (2009b). PeerSim: A scalable P2P simulator. In *Proc. of the 9th Int. Conference on Peer-to-Peer (P2P'09)*, pages 99–100, Seattle, WA. 39, 72

Pokahr, A., Braubach, L., and Lamersdorf, W. (2005). Jadex: A bdi reasoning engine. In *Multi-agent programming*, pages 149–174. Springer. 29

Rao, A. S. (1996). Agentspeak(l): Bdi agents speak out in a logical computable language. In Van de Velde, W. and Perram, J. W., editors, *Agents Breaking Away*, pages 42–55, Berlin, Heidelberg. Springer Berlin Heidelberg. 26

Russell, S. and Norvig, P. (2010). *Artificial intelligence : a modern approach*. Prentice Hall, 3 edition. x, 18, 19, 20

Sabater, J. (2002). *Trust and reputation for agent societies*. Tesi doctoral, Universitat Autònoma de Barcelona, España. 30, 32, 33, 36, 45

Sanou, B. (2015). Ict facts and figures 2015. *International Telecommunication Union (ITU) Fact Sheet*. 1

Sarkar, S., Chatterjee, S., and Misra, S. (2018). Assessment of the suitability of fog computing in the context of internet of things. *IEEE Transactions on Cloud Computing*, 6(1):46–59. 9

Skarlat, O., Schulte, S., Borkowski, M., and Leitner, P. (2016). Resource provisioning for iot services in the fog. In *2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA)*, pages 32–39. 10, 13, 40, 44, 45, 46, 47, 56

Vaquero, L. M. and Rodero-Merino, L. (2014). Finding your way in the fog: Towards a comprehensive definition of fog computing. *SIGCOMM Comput. Commun. Rev.*, 44(5):27–32. 8

Varghese, B., Wang, N., Nikolopoulos, D. S., and Buyya, R. (2017). Feasibility of fog computing. *CoRR*, abs/1701.05451. 9, 11, 15, 41

Weiss, G. (2013). *Multiagent Systems*. The MIT Press. x, 20, 22, 23, 25

Wooldridge, M. (2009). *An Introduction to MultiAgent Systems*. Wiley Publishing, 2nd edition. x, 18, 19, 21, 22, 25, 26

Wooldridge, M. and Jennings, N. R. (1995). Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 10:115–152. 22, 25

Yi, S., Li, C., and Li, Q. (2015). A survey of fog computing: Concepts, applications and issues. In *Proceedings of the 2015 Workshop on Mobile Big Data*, Mobidata '15, pages 37–42, New York, NY, USA. ACM. 13, 78

Zacharia, G. and Maes, P. (2000). Trust management through reputation mechanisms. *Applied Artificial Intelligence*, 14(9):881–907. 32

Zambonelli, F., Jennings, N. R., and Wooldridge, M. (2003). Developing multiagent systems: The gaia methodology. *ACM Trans. Softw. Eng. Methodol.*, 12(3):317–370. 21

Apêndice A

Detalhes de implementação do modelo

A.1 Comunicação entre os elementos

Para se comunicar com o cliente de serviço implantado no dispositivo o agente *NodeAgent* implementa um objeto da classe *Socket* do Java. Assim, o agente inicia uma *thread* Java onde esse *socket* é executado e fica aguardando por mensagens provindas do cliente de serviço. O fluxo de mensagens a partir da *thread* onde o *socket* é executado para o *NodeAgent*, que reside no *Container* do JADE, é feito por meio da implementação de uma fila de mensagens. O Código A.1 apresenta a classe *NodeCommunicationInterfaceBehaviour* que estende um dos Behaviours pré-definidos pelo JADE. Esse *Behaviour* instanciado pelo *NodeAgent* que consome a fila de mensagens provindas do cliente de serviço de fog. Essa fila é implementada pela classe *BlockingQueue* disponibilizada pelo Java, de forma que toda mensagem que é recebida pelo *NodeAgent* por meio do *socket*, é incluída na *BlockingQueue* (Figura A.1). As mensagens colocadas na fila podem ser *strings* que indicam a ativação (*"started"*) ou encerramento (*"stopped"*) do cliente de serviço, ou podem ser objetos da classe *MessageStats*, definida no âmbito desse modelo implementacional, que agrupam informações acerca do consumo do serviço e da QoS. Este modelo de comunicação também é implementado pelo *SupernodeAgent* e pelo *CloudManagerAgent* para realizar a comunicação com os elementos da camada de serviço de fog.

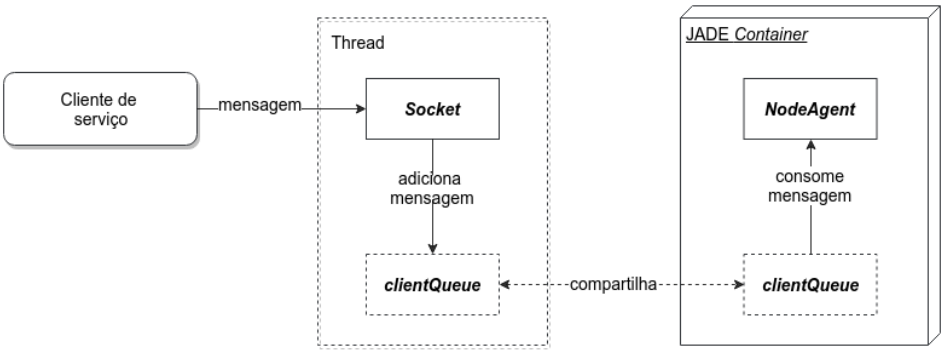

Figura A.1: Comunicação entre cliente de serviço e *NodeAgent*, onde *clientQueue* é uma instância da classe *BlockingQueue* disponibilizada pelo Java

```
public class NodeInterfaceBehaviour extends CyclicBehaviour {
  private BlockingQueue<Object> clientQueue;

  public NodeInterfaceBehaviour(BlockingQueue<Object> clientQueue) {
    this.clientQueue = clientQueue;
  }

  @Override
  public void action() {
    consume(this.clientQueue.poll());
  }

  private void consume(Object queueValue) {
    if(queueValue != null) {
      if(queueValue instanceof String) {
        String value = (String) queueValue;
        if(value.toLowerCase().contains("started")) {
          getAgent().nodeStatus = NodeStatus.STARTED;
        } else if(value.toLowerCase().contains("stopped")) {
          getAgent().nodeStatus = NodeStatus.INACTIVE;
          getAgent().addBehaviour(new TearDownBehaviour(true));
        }
      } else if(queueValue instanceof MessageStats) {
        MessageStats msg = (MessageStats) queueValue;
        if(getAgent().serviceStats.containsKey(msg.server)) {
          getAgent().serviceStats.get(msg.server).add(msg);
        } else {
          List<MessageStats> lista = new ArrayList<>();
          lista.add(msg);
          getAgent().serviceStats.put(msg.server, lista);
        }

      }
    } else {
      block(100);
    }
  }

  @Override
  public NodeAgent getAgent() {
    return (NodeAgent) myAgent;
  }
}
```

Código A.1: Parte da comunicação entre *NodeAgent* e cliente de serviço de fog

A.2 Relacionamento entre as classes

A Figura A.2 apresenta o relacionamento entre as classes que implementam os agentes Gerente de Nuvem e Coordenador Local juntamente com seus *Behaviours*. Na sequência seguem descritos cada um desses *Behaviours*.

- *CloudFogServiceBehaviour*: Implementa a interação com o *host* do serviço que fica localizado na nuvem.
- *LocalCoordinatorRegistrationServiceBehaviour*: Serviço de registro de Coordenadores Locais. Implementa os protocolos de interação para adição e remoção de Coordenadores Locais na estrutura da fog computacional.

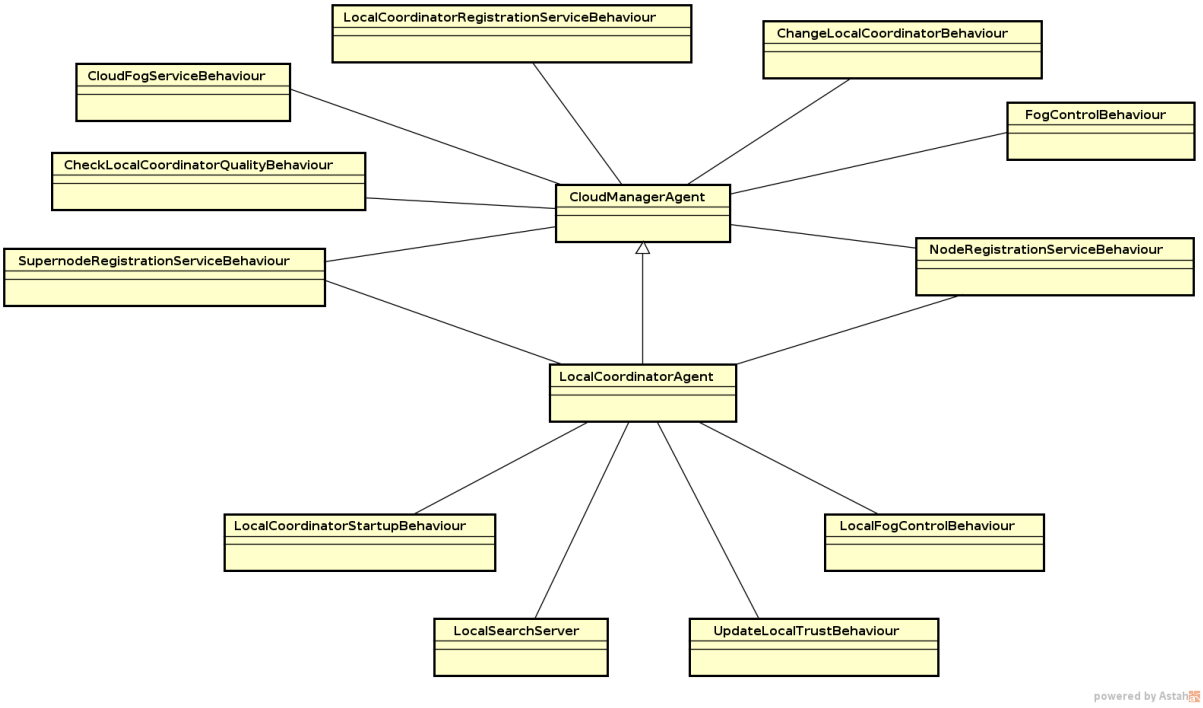

Figura A.2: Diagrama com as classes que implementam o Gerente de Nuvem e o Coordenador Local juntamente com seus *Behaviours*

- *SupernodeRegistrationServiceBehaviour*: Serviço de registro de supernós. Implementa os protocolos de interação para adição e remoção de supernós na estrutura da fog computacional.
- *NodeRegistrationServiceBehaviour*: Serviço de registro de nós. Implementa os protocolos de interação para adição e remoção de nós na estrutura da fog computacional.
- *FogControlBehaviour*: Implementa as ações de controle da fog computacional e serve como base para a inicialização do *CheckLocalCoordinatorQualityBehaviour*.
- *CheckLocalCoordinatorQualityBehaviour*: Implementa o processo de verificação da QoS dos Coordenadores Locais utilizando os protocolos de interação para a obtenção de avaliações dos nós acerca dos Coordenadores Locais. Caso seja necessário inicializa o *ChangeLocalCoordinatorBehaviour*.
- *ChangeLocalCoordinatorBehaviour*: Realiza a alteração dos Coordenadores Locais a partir do protocolo de interação FIPA ContractNet.
- *LocalCoordinatorStartupBehaviour*: Realiza a inicialização do Coordenador Local implementando os protocolos de interação para fazer o registro do Coordenador Local com o Gerente de Nuvem e invocar o *NodeRegistrationServiceBehaviour* e o *SupernodeRegistrationServiceBehaviour*.
- *LocalSearchServer*: Serviço que busca os supernós ligados ao Coordenador Local que estão mais próximos ao nó requisitante.
- *UpdateLocalTrustBehaviour*: Implementa o processo de cálculo da confiança local.

- *LocalFogControlBehaviour*: Implementa as ações de controle da comunidade local da fog computacional e serve como base para a inicialização do *UpdateLocalTrustBehaviour*.

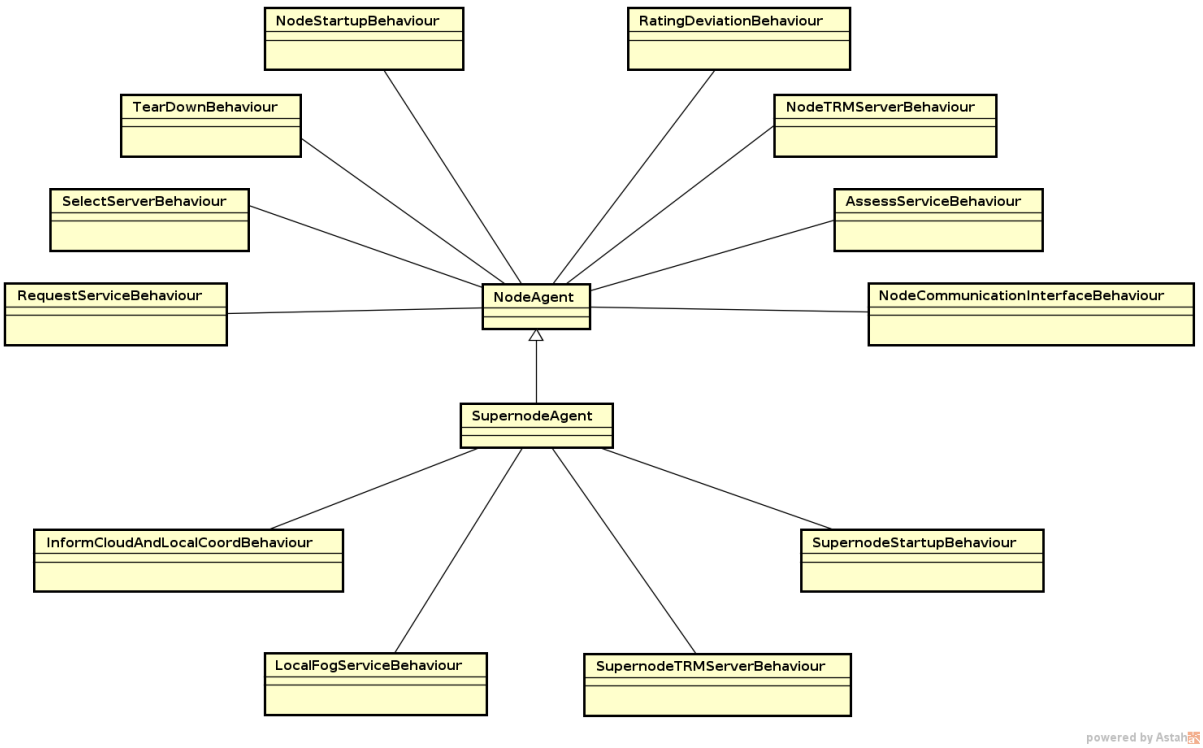

Figura A.3: Diagrama com as classes que implementam os agentes nó e supernó juntamente com seus *Behaviours*

Nesse sentido, a Figura A.3 apresenta o relacionamento entre as classes que implementam os agentes Nó e supernó juntamente com seus *Behaviours*. Na sequência seguem descritos cada um dos *Behaviours* que estão ligados à esses agentes:

- *NodeStartupBehaviour*: Realiza a inicialização do nó, solicitando o registro ao Gerente de nuvem e posteriormente ao Coordenador Local. Implementa os protocolos de interação para realizar essas solicitações.
- *TearDownBehaviour*: Quando o cliente de serviço de fog fica inativo esse *Behaviour* é lançado para se desconectar do supernó, do Coordenador Local e do Gerente de Nuvem.
- *RequestServiceBehaviour*: Implementa o fluxo de requisição de serviço enviada ao Coordenador Local de forma a buscar a lista de supernós disponíveis.
- *SelectServerBehaviour*: Implementa o processo de seleção dos supernós baseado na confiança e reputação.
- *RatingDeviationBehaviour*: Responde ao Coordenador Local para realizar o cálculo do desvio do conjunto de avaliações como parte do processo realizado por aquela agente para calcular a confiança local.

- *NodeTRMServerBehaviour*: Implementa um serviço que responde à solicitações de envio de avaliações feitas pelos supernós e por outros nós. Este *Behaviour* que é responsável por utilizar os protocolos de interação que permitem a troca de avaliações que serão aplicadas no cálculo da reputação (certificada ou de testemunhas).
- *AssessServiceBehaviour*: Realiza a avaliação do supernó com base nas informações de consumo de serviço enviadas pelo cliente de serviço da fog.
- *NodeCommunicationInterfaceBehaviour*: Implementa a comunicação com o cliente de serviço da fog que utiliza o agente Nó para selecionar um provedor de serviços.
- *InformCloudAndLocalCoordBehaviour*: Implementa a comunicação ao Coordenador Local e ao Gerente de Nuvem quando um nó é incluído na lista de nós suportados pelo supernó. Partindo desta interação que o Coordenador Local irá manter sua tabela de interações entre nós e supernós.
- *LocalFogServiceBehaviour*: Implementa a interação com o *host* do serviço localizado no dispositivo onde reside o supernó.
- *SupernodeTRMServerBehaviour*: Serviço que responde às solicitações dos nós acerca das avaliações para formação da reputação certificada.
- *SupernodeStartupBehaviour*: De maneira análoga ao *NodeStartupBehaviour*, este *Behaviour* realiza a inicializaçao do supernó, solicitando o registro ao Gerente de nuvem e ao Coordenador local.