Universidade Federal do Ceará

Centro de Ciências

Mestrado e Doutorado em Ciência da Computação

AESPmob - modelo autônomo e evolutivo para provisão de serviços essenciais em redes móveis

Dissertação de Mestrado

Marcio Espíndola Freire Maia

Orientadora: Rossana Maria de Castro Andrade, PhD.

Fortaleza

Julho - 2009

Marcio Espíndola Freire Maia

AESPmob - modelo autônomo e evolutivo para provisão de serviços essenciais em redes móveis

Dissertação de Mestrado submetida à Coordenação do Mestrado e Doutorado em Ciência da Computação da Universidade Federal do Ceará, como requisito parcial para obtenção do título de Mestre.

Orientadora: Rossana Maria de Castro Andrade, PhD.

Fortaleza

Julho - 2009

AESPmob - modelo autônomo e evolutivo para provisão de serviços essenciais em redes móveis

Dissertação de Mestrado submetida à Coordenação do Mestrado e Doutorado em Ciência da Computação da Universidade Federal do Ceará, como requisito parcial para obtenção do título de Mestre.

Aprovado em 6 de julho de 2009

Banca Examinadora

Prof.a Rossana Maria de Castro Andrade, PhD. (Orientadora)
Universidade Federal do Ceará - UFC

Prof. Wanderley Lopes de Souza, DSc.
Universidade Federal de São Carlos - UFSCar

Prof. Francisco Heron de Carvalho Júnior, DSc.
Universidade Federal do Ceará - UFC

Prof. Nabor das Changas Mendonça, PhD.
Universidade de Fortaleza - UNIFOR

Dedico esta dissertação

Aos meus pais Winston de Paulo Bastos Maia e Olga Lúcia Espíndola Freire Maia, por tudo que me ensinaram até hoje, pelas palavras de apoio e pelos exemplos de amor, caridade e honestidade.

Aos meus irmãos Renato e Diane, pelos momentos de alegria e companheirismo.

À Marina, companheira fiel e sempre presente, pelos momentos de amor e felicidade.

Agradecimentos

Gostaria de iniciar agradecendo a todos que de alguma forma contribuíram para que conseguisse chegar até aqui. Saibam que têm um papel importante nas escolhas que me fizeram quem eu sou.

Dito isso, agradeço à Deus por tudo que tenho, pelos momentos de felicidade, que me ensinaram a apreciar o dom da vida, e pelos momentos de dificuldade, que moldaram meu caráter. Agradecer com palavras, no entanto, é fácil. Espero que consiga dignificar e valorizar as oportunidades que tive.

Aos meus pais Winston de Paulo Bastos Maia e Olga Lúcia Espíndola Freire Maia. Tudo que tenho e sou é graças a vocês. Obrigado pelo que me ensinaram através da palavra e do exemplo. Obrigado pelo lar em paz e pelo amor em família.

Obrigado também aos meus avós paternos, Braz de Souza Maia e Neide Bastos Maia, e maternos, Aldenor Nunes Freire e Maria Anayde Espíndola Freire, pelo exemplo constante de honestidade e busca incessante de iluminação interior.

Aos meus irmãos Renato e Diane, obrigado. Vocês têm sido muito importantes durante toda essa caminhada.

Especialmente à Marina, obrigado. Que seja o início de uma vida juntos, repleta de amor e realizações.

Agradeço à Professora Rossana por acreditar e investir em mim e a todos os professores e servidores do MDCC. Agradeço ainda a todos no GREat, em especial à Christiane, à Darilu, à Liliane, ao seu Adir e ao Danilo.

Por último, mas não menos importante, obrigado a todos os amigos de pesquisa e projetos no GREat, em especial (ordem alfabética) Bosco, Bruno Góis, Carina, Cláudio, Davi, Diana, Flávio, João Borges, João Marcelo, Levi, Lincoln, Luana, Marcos Dantas, Mirko, Paulo Henrique, Reinaldo, Rute e Valéria.

Obrigado ainda à FUNCAP e à CAPES, que deram suporte financeiro indispensável à conclusão desse mestrado.

"... amaí-vos, eis o primeiro ensinamento; instruí-vos,

eis o segundo."

Espírito de Verdade

RESUMO

A popularização dos dispositivos móveis e das redes sem fio permitem a criação de um paradigma de interação, onde os dispositivos se reúnem e cooperam para a realização de tarefas de curta duração sem nenhuma pressuposição sobre os recursos disponíveis. Fatores como a heterogeneidade de hardware e software, mobilidade dos dispositivos e dinamicidade do ambiente, entretanto, podem dificultar ou inviabilizar a gerência e manutenção de aplicações móveis distribuídas. Dessa forma, essa dissertação propõe o AESPmob, um modelo autônomo e evolutivo, permitindo que serviços essenciais como Troca de Mensagens, Suporte à Mobilidade, Descrição, Descoberta e Coordenação de Serviços, além de Segurança, estejam sempre disponíveis e eficientes. O AESPmob permite a evolução dos serviços mediante a difusão das implementações (estratégias) mais eficientes. Para validar essa proposta, um modelo matemático descreve como varia a complexidade de tempo para difundir uma estratégia, à medida que o número dos dispositivos aumenta, e como a eficiência e o custo na rede variam quando as estratégias são difundidas. Finalmente, o mecanismo da difusão é implementado em um simulador para validar os resultados analíticos.

ABSTRACT

The popularization of mobile devices has allowed the development of a new interaction paradigm, where devices get together and cooperate to perform short-lasting tasks. However, the complexity to deal with hardware and software heterogeneity, lack and dynamicy of resources, along with device mobility, has nearly prohibited the decentralized execution of these tasks. To minimize these problems, this dissertation proposes AESPmob, a model to guarantee that essential services like service invocation, description, discovery and coordination are available and efficient. Acting autonomously, an implementation of AESPmob on each device monitors resource usage, adapts and evolves essential services. Service evolution occurs by difusing the most efficient implementation of each service (strategy). Consequently, by diffusing that strategy, the overall efficiency of the network is improved. To validate AESPmob, a mathematical model describes how the time complexity to diffuse one strategy varies as the number of devices increases, and how the efficiency and cost on the network is affected by the strategy diffusion process. Finally, AESPmob is implemented on a simulator to confirm the analytical results.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

P2P	Peer-to-Peer
SOA	Arquitetura orientada a serviço
QoS	Qualidade de serviço
EC	Computação evolutiva
EA	Algoritmos evolutivos
NFR	Non-functional requirement
SIG	Softgoal interdependency graph
RPC	Remote procedure call
IOEPs	Input, output, preconditions and effects
OWL	Ontology web language
RuleML	Rule markup language
SLA	Service level agreement
TTL	Time-to-live
AESPmob	Autonomous and evolutionary model for essential service provision in mobile networks
ES	Essential services

Lista de Algoritmos

Sumário

Capítulo 1

INTRODUÇÃO

Esta dissertação apresenta o AESPmob, um modelo autônomo e evolutivo para provisão de serviços essenciais em redes móveis. Este capítulo introduz, na Seção 1.1, aplicações móveis e a problemática relacionada ao tema. Em seguida, na Seção 1.2 expõe a motivação para o desenvolvimento deste trabalho. Na Seção 1.3 os principais objetivos desta dissertação são apresentados. Por fim, na Seção 1.4 traz a estrutura organizacional desta dissertação.

1.1 Contextualização e caracterização do problema

A popularização dos dispositivos móveis e da comunicação sem fio influenciaram uma mudança no *design* arquitetural das redes de computadores e das aplicações. As redes evoluíram de modelos centralizados, estáticos e baseados em cabos, para móveis e sem fio estruturados como as redes celulares e 802.11 a/b/g, mais recentemente, para modelos parcialmente desestruturados como redes de malha, e totalmente desestruturados como as redes *ad hoc* [Kiess e Mauve 2007]. Por outro lado, as aplicações têm evoluído do tradicional modelo cliente-servidor para modelos mais descentralizados, como o par-a-par ($P2P$) [Issarny et al. 2007].

Seguindo essa tendência, os dispositivos móveis utilizam simultaneamente várias interfaces de comunicação. Por exemplo, uma aplicação acessa uma página na

internet usando a conexão de dados da rede celular e, ao mesmo tempo, compartilha arquivos com dispositivos próximos de forma *P2P* via *Bluetooth*. Com isso, o desenvolvimento das aplicações móveis distribuídas ficou bem mais flexível. Essa flexibilidade, entretanto, também ensejou um aumento na complexidade de desenvolvimento e gerenciamento dessas aplicações.

Para minimizar a complexidade inerente dos sistemas móveis, normalmente são utilizados sistemas de *middleware* [Dario Bruneo e Scarpa 2006]. Um sistema de *middleware* é uma camada inserida entre as aplicações e os sistemas operacionais, e oferece abstrações que permitem aos desenvolvedores concentrar-se unicamente nos requisitos das aplicações. Dentre outras funcionalidades, os sistemas de *middleware* implementam protocolos na camada de aplicação responsáveis pelos requisitos de sistemas distribuídos como troca de mensagens ou invocação remota de métodos.

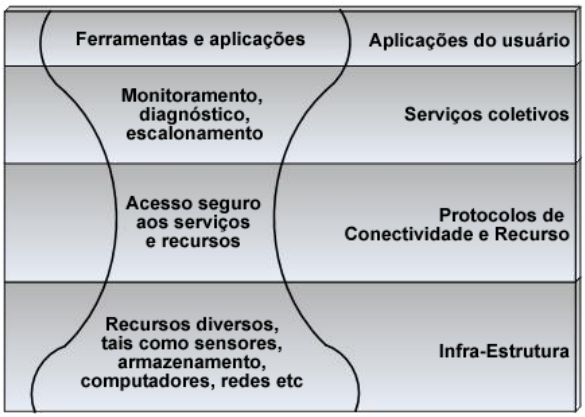


Figura 1.1: Sistema de *middleware* fazendo a interface entre as aplicações e os recursos (Adaptado de [Foster e Kesselman 2003]).

A Figura 1.1 mostra como um sistema de *middleware* é divido em camadas. A parte larga superior representa as aplicações acessando os serviços oferecidos pela parte mais estreita, que podem ser coletivos como monitoramento, diagnóstico, escalonamento e gerência; ou podem ser de conectividade e de acesso a recursos. Por fim, na camada larga inferior estão contidos os recursos do ambiente, tais como armazenamento, processamento ou informações de contexto.

Uma das vantagens em adotar esse modelo de camadas é o desacoplamento. Cada

camada é implementada de maneira independente e apenas as interfaces dos serviços são publicadas e dinamicamente descobertas. Dessa forma, recursos e serviços podem ser incluídos, atualizados e excluídos espontaneamente sem prejuízo para as camadas superiores.

Cada um desses serviços oferecidos pelo sistema de *middleware* tem características específicas, e sua aplicabilidade é definida pelo ambiente onde serão utilizados. Por exemplo, se a conexão é feita utilizando a rede celular, é factível o uso de um modelo síncrono de troca de mensagens. Por outro lado, formas de comunicação menos acopladas são úteis quando a conexão é feita por interfaces de menor alcance e, consequentemente, mais intermitente [Eugster et al. 2003]. Com isso, protocolos na camada de aplicação propostos para uma arquitetura de rede, muito raramente podem ser utilizados em uma outra arquitetura.

A eficiência (ou mesmo a aplicabilidade) de protocolos na camada de aplicação é muito dependente de informações na camada de rede. Uma forma de lidar com esse problema é utilizar *Cross-layering*, onde informações coletadas em uma camada ficam disponíveis às outras camadas [Conti et al. 2005]. Essa abordagem é útil para adaptar e otimizar o comportamento de todas as camadas às variações do ambiente.

Por outro lado, o uso de *Cross-layering* pode resultar em aumento na complexidade para manter e evoluir uma camada, já que uma modificação impacta em todas as outras, dificulta a padronização das camadas e ainda gera uma perda de eficiência se as entre camadas não forem cuidadosamente projetadas [Kawadia e Kumar 2005].

Consequentemente, o que deveria acontecer era a escolha de determinado protocolo de acordo com as características da rede. Porém, na prática, as aplicações estão muito acopladas ao sistema de *middleware* que implementa um determinando protocolo. Essa abordagem dificulta a manutenção e evolução do protocolo utilizado e restringe a interoperabilidade, já que os dispositivos precisam interagir utilizando o mesmo sistema de *middleware*.

1.2 Motivação

Em aplicações móveis distribuídas, é necessário que exista baixo acoplamento entre as diversas partes de um *software*. Isso acontece em razão de restrições no alcance da antena e no tempo de vida da bateria dos dispositivos, o que ocasiona uma incerteza sobre a disponibilidade dos pares interagindo.

Além disso, não é trivial garantir a interoperabilidade entre aplicações móveis distribuídas. Isso acontece por vários fatores, dentre os quais se pode destacar a diversidade de *hardware*, no que diz respeito à capacidade de processamento e armazenamento, além do uso simultâneo das diferentes interfaces de comunicação sem fio disponíveis. Outro fator que dificulta a interoperabilidade é a diversidade de *software*, em que as aplicações são desenvolvidas em linguagens diversas e sobre sistemas operacionais variados.

Uma alternativa viável para lidar com a necessidade de baixo acoplamento entre as aplicações e com a heterogeneidade de *software* é a utilização de arquiteturas orientadas a serviço (*SOA*) [Erl 2005]. *SOA* é um caminho natural para o desenvolvimento de aplicações móveis interoperáveis e com baixo acoplamento. No lugar de instalar todas as funcionalidades em tempo de desenvolvimento, ela permite que as aplicações encontrem no próprio ambiente as funcionalidades necessárias. Adicionalmente, padronizações amplamente aceitas permitem que aplicações desenvolvidas utilizando linguagens de programação diversas e sobre sistemas operacionais variados possam interagir.

Projetar aplicações de acordo com *SOA* permite que elas sejam decompostas em unidades lógicas menores e distintas. Essa decomposição possibilita a cada uma dessas unidades funcionar e evoluir de forma autônoma, mas não isolada. Isso acontece porque elas devem seguir uma série de padronizações que mantêm um nível mínimo de similaridade. Em *SOA*, essas unidades lógicas são conhecidas como serviços [Erl 2005].

Quatro ações bem definidas para a utilização de um serviço podem ser destaca-

das: descrição, descoberta, invocação e coordenação. Para cada uma delas existem protocolos que determinam como executá-las. O que acontece é que cada protocolo proposto está muito acoplado à arquitetura da rede para a qual ele foi projetado e ainda às condições momentâneas dessa rede.

Por exemplo, quando a conexão é feita por meio de uma rede celular, um protocolo de descoberta de serviços centralizado é adequado. Já quando a latência nessa rede estiver alta, ou quando o dispositivo estiver conectado apenas a uma rede *ad hoc* de pequeno porte e baixa mobilidade, um protocolo parcialmente distribuído é mais aconselhável. Quando a mobilidade ou o número dos dispositivos aumentar, um protocolo totalmente distribuído passa a ser a alternativa [Engelstad et al. 2006].

Essa diversidade e dinamicidade de cenários dificulta o desenvolvimento de um protocolo único que seja eficiente em todas as situações. Como cada protocolo é projetado para um contexto bem específico, é necessário um modelo autônomo que ofereça às aplicações um mecanismo de descoberta e instalação dos protocolos mais adequados a cada cenário.

1.3 Objetivos e contribuições

Este trabalho tem por objetivo apresentar um modelo autônomo e evolutivo para provisão de serviços essenciais em redes móveis. Esse modelo é chamado AESPmob. Nesta dissertação, serviços essenciais são aqueles utilizados para construção de aplicações móveis distribuídas baseados em *SOA*. Neste trabalho, são detalhados seis desses serviços: Troca de Mensagens, Suporte à Mobilidade, Descrição, Descoberta, e Coordenação de Serviços, além de Segurança.

O AESPmob constantemente monitora o ambiente para definir qual implementação existente, denominada estratégia, de um dado serviço essencial é mais adequada, ou eficiente, no contexto percebido. A medida de eficiência é descrita pela aplicação ao *AESPmob* e quantificada pelo histórico local de uso dessa estratégia, combinado

com informações recebidas dos dispositivos vizinhos.

Quando uma nova estratégia para um serviço essencial é encontrada em um dispositivo vizinho, o nível de eficiência dessa estratégia é comparado com o nível de eficiência da estratégia local equivalente. Caso o nível de eficiência remoto seja maior, essa estratégia é incorporada localmente e utilizada a partir desse momento.

À medida que os dispositivos interagem, as estratégias mais eficientes são difundidas, o que possibilita um aumento da eficiência do serviço essencial como um todo. Assim, nesta dissertação, a eficiência total é o valor que representa a soma da eficiência de todas as estratégias sendo utilizadas em um dado momento. Por isso que, ao difundir as estratégias mais eficientes e eliminar aquelas com menor eficiência, o valor da eficiência total tende a aumentar.

Este trabalho também apresenta um modelo analítico que descreve como a complexidade de tempo para difusão das estratégias mais eficientes varia com o aumento no número dos dispositivos, e como essa difusão impacta na eficiência total do serviço essencial e o custo na rede.

Para avaliar o AESPmob, quatro protocolos de descoberta de serviços são implementados em um simulador e atribuídos aleatoriamente aos nós. O objetivo é mostrar que a eficiência total da descoberta de serviço melhora à medida que as estratégias são difundidas. Na simulação, as estratégias são avaliadas de acordo com três métricas de eficiência: número de serviços encontrados, razão entre o número de serviços encontrados e o total de mensagens enviadas e a latência média para descoberta do serviço.

Outro fator analisado é como determinar, em tempo de execução, qual o protocolo mais eficiente. Nesse caso, três abordagens são utilizadas. A primeira das quais considera apenas o histórico do uso de uma estratégia, enquanto a segunda combina esse histórico com informações dos nós vizinhos e a terceira utiliza uma visão global compartilhada por todos os dispositivos. Essas três abordagens foram então comparadas entre si e com a abordagem onde não existe a difusão das estratégias.

No texto desta dissertação, três conceitos são bastante utilizados: sistemas móveis, redes móveis e aplicações móveis. Os sistemas móveis representam o conjunto formado pelas redes e aplicações móveis. Já as redes móveis são as redes de computadores que se comunicam utilizando tecnologia de transmissão sem fio e suportam a mobilidade de dispositivos. Por outro lado, as aplicações móveis são aplicações distribuídas que se comunicam utilizando as redes móveis.

1.4 Estrutura da dissertação

Esta dissertação está organizada em seis capítulos. Este capítulo fez uma ligeira introdução ao tema, contextualizando o assunto aqui abordado. Além disso, o capítulo 1 apresentou a motivação, o objetivo e a principal contribuição deste trabalho de dissertação.

O capítulo 2 exibe alguns conceitos utilizados nesta dissertação. Inicialmente, ele trata dos sistemas autônomos e suas características como auto configuração, auto otimização, auto cura e auto proteção. A seguir, detalha os sistemas evolutivos, enfatizando como acontece um ciclo evolutivo. Para concluir, o capítulo descreve trabalhos relacionados ao tema tratado.

Já o capítulo 3 detalha serviços essenciais como Troca de mensagens, Suporte à Mobilidade, Descrição, Descoberta, e Coordenação de serviços, além de Segurança, exemplificando algumas abordagens utilizadas para implementar cada um deles e quais as vantagens e desvantagens dessas implementações. Para finalizar o capítulo, as arquiteturas de redes móveis são discutidas e como cada um dos serviços essenciais se adapta a essas arquiteturas.

O AESPmob é descrito no capítulo 4, sua arquitetura é apresentada, e o mecanismo de difusão das estratégias é discutido. Para finalizar o capítulo, um modelo analítico baseado em cadeias de Markov descreve como varia a complexidade de tempo para difusão de uma estratégia à medida que o número de dispositivos au-

menta, e como a eficiência e o custo na rede são influenciados por essa difusão.

Adicionalmente, o capítulo 5 mostra como a eficiência e o custo variam à proporção que as estratégias são difundidas. Para isso, quatro protocolos de descoberta de serviços são implementados em um simulador e atribuídos aleatoriamente aos nós da simulação. Em cada cenário, uma métrica de eficiência foi aplicada e a difusão é feita baseada nessa métrica. As métricas utilizadas foram número de serviços descobertos, razão entre número descoberto e número de mensagens enviadas e latência média de descoberta.

Já o capítulo 6 apresenta as conclusões e algumas questões que precisam ser analisadas antes de implementar o AESPmob em dispositivos reais.

Capítulo 2

SISTEMAS AUTÔNOMOS E EVOLUTIVOS

Este capítulo descreve os sistemas autônomos e evolutivos, lista algumas características que definem um comportamento autônomo evolutivo e as vantagens em utilizá-los. A Seção 2.1 define e caracteriza os sistemas autônomos, enquanto os algoritmos evolutivos são detalhados na Seção 2.2. Para finalizar o capítulo, a Seção 2.3 lista e descreve alguns sistemas que apresentam características autônomas evolutivas.

2.1 Sistemas autônomos

O constante crescimento no número dos dispositivos e da complexidade dos sistemas móveis distribuídos traz grande dificuldade na gerência e configuração das redes e aplicações que formam esse sistema. Essa dificuldade fez com que os pesquisadores buscassem soluções alternativas inspiradas em sistemas biológicos para lidar com o dinamismo, heterogeneidade e incerteza inerentes dos sistemas móveis [Sterritt et al. 2005].

A expressão Computação autônoma (*Autonomic Computing*)[1] foi criado em 2001

[1] Computação autônoma foi criada como uma solução para minimizar o custo e a complexidade

por Paul Horn, vice-presidente sênior de pequisa da IBM [IBM 2005], e é uma alternativa cada vez mais utilizada para lidar com o dinamismo inerente dos sistemas móveis distribuídos. Sistemas autônomos são capazes de se gerenciar, normalmente formados por, várias entidades autônomas que interagem e tomam decisões locais com o objetivo de atingir um comportamento satisfatório [Baduel e Matsuoka 2007]. Em ambientes altamente distribuídos, essas decisões devem ser tomadas com suporte na visão local de cada entidade e dirigidas por políticas de alto nível fornecidas pelos administradores.

Por exemplo, um algoritmo de roteamento em tempo real requer uma latência limite para a entrega dos pacotes. O valor máximo aceitável e algumas heurísticas para atingir esse valor são fornecidos pelo administrador ou pelas aplicações. Já a heurística utilizada para definir a rota que cada pacote percorrerá é definida de forma autônoma por parte de cada elemento que participa do roteamento, utilizando informações recebidas de dispositivos próximos e sua própria visão do sistema. Outra ação autônoma pode ser o monitoramento da carga na rede e a alocação de determinados serviços em locais estratégicos para melhorar o balanceamento de carga [Benda et al. 2006].

2.1.1 Atributos de uma entidade autônoma

Cada entidade autônoma é responsável por seu comportamento individual, gerenciando os recursos que estão sob o seu controle. Operações internas como autoconfiguração, auto-otimização, autocura e autoproteção [Tesauro et al. 2004] definem como o comportamento global desejado será alcançado.

Autoconfiguração

Autoconfiguração é a capacidade de um sistema automaticamente configurar o seu comportamento baseado em políticas de alto nível que descrevem o que deve

de manutenção e evolução dos grandes sistemas de TI. Segundo Paul Horn, sem a criação de sistemas auto gerenciáveis, as empresas não entrariam na "nova era da computação" [Ganek 2006].

ser realizado, mas não como fazê-lo. Em redes móveis, um dispositivo que se move para uma região onde existem novos serviços sendo oferecidos deve ser capaz de instalá-los dinamicamente. Por exemplo, um serviço de compartilhamento de arquivos está presente em uma região onde o usuário acaba de entrar. O sistema espontaneamente detecta a existência desse serviço, baixa os executáveis necessários e informa o usuário ou uma aplicação que um novo serviço foi instalado. Note-se que a política definida pelo administrador (usuário) informa apenas a descrição do serviço de compartilhamento e transferência de arquivos, possivelmente detalhando parâmetros de qualidade de serviço (QoS) como latência mínima da rede.

Auto-otimização

Auto-otimização permite que o sistema continuamente procure formas de otimizar e evoluir o comportamento para se tornar mais eficiente em termos de desempenho e custo. O sistema autônomo deve constantemente monitorar o ambiente, aplicando métricas para avaliar se existem implementações de serviços mais eficientes disponíveis. Por exemplo, um novo serviço de descoberta de recursos está presente no ambiente e fornece mais resultados para uma busca (maior eficiência), trocando um número menor de mensagens (menor custo). Entretanto, se mais mensagens são trocadas, mas os recursos encontrados são mais eficientes ou semanticamente mais próximos, o sistema deve avaliar se o aumento no custo em utilizar o serviço será compensado pela maior eficiência na execução do serviço.

No cenário descrito, a política de alto nível que guia o comportamento autônomo é definida pela métrica de eficiência necessária à aplicação, seja em termos do número de serviços descobertos ou da proximidade semântica desses serviços com o serviço procurado. Já o comportamento autônomo é o monitoramento das diferentes implementações existentes para um serviço de descoberta e a instalação de um serviço mais eficiente.

Autocura

Autocura é a capacidade de detectar, diagnosticar e reparar problemas resultantes de falhas de *software* e *hardware*, ou decorrentes da escassez de recursos como baixo nível de bateria ou alta latência da rede. Assim, o próprio sistema fica responsável por garantir as propriedades ACID (atomicidade, consistência, isolamento e durabilidade). Por exemplo, se uma aplicação está executando uma operação crítica que requer interação com outros dispositivos, o sistema deve garantir que essa operação seja executada corretamente, ou, em caso de falhas, permitir que a aplicação se recupere e volte a um estado consistente.

Autoproteção

Autoproteção consiste em defender o sistema contra ataques maliciosos ou falhas em cascata e ainda prever possíveis problemas utilizando informações fornecidas por sensores presentes no dispositivo e no ambiente. Se, durante a transmissão de um arquivo, o sistema percebe que a taxa de transmissão está diminuindo consideravelmente, ele pode já iniciar uma busca por novas rotas e redirecionar essa transferência para o caminho que possuir a menor latência.

2.1.2 Gerência autônoma

O mecanismo de gerência autônoma é responsável pela coordenação, manutenção e evolução de um sistema. Nele, existe um laço fechado de controle que determina as ações que devem realizadas para se atingir um nível máximo de autonomia em cada um dos quatro atributos de uma entidade autônoma [IBM 2005].

A Figura 2.1 mostra esse laço de controle. Ele consiste em quatro ações bem definidas que se relacionam e interferem no comportamento das outras, e são divididas em Monitorar, Analisar, Planejar e Executar (MAPE), além da função Conhecimento que determina o comportamento do laço MAPE.

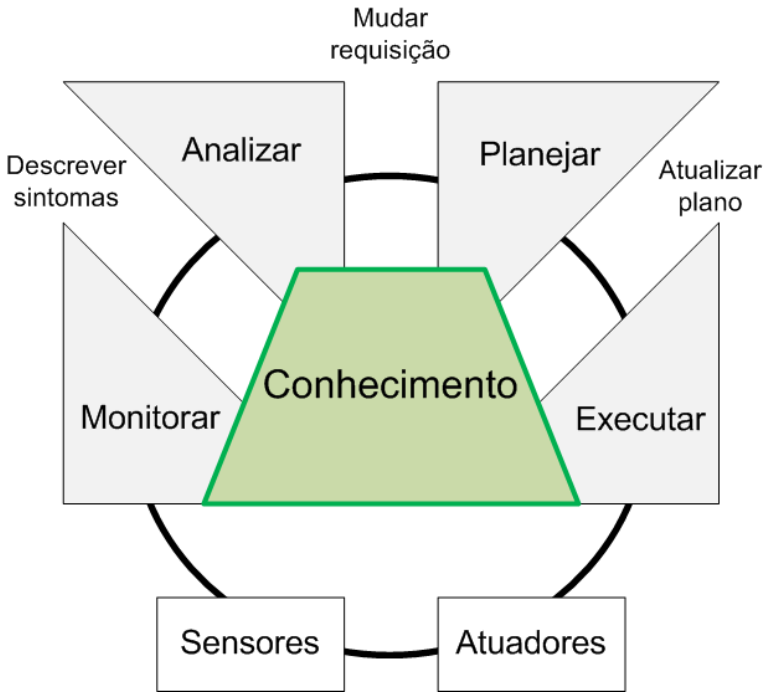


Figura 2.1: Laço de controle MAPE. Adaptada de [IBM 2005]

Monitorar

A ação de monitoramento oferece mecanismos para coletar, filtrar, agregar e descrever informações sobre os recursos disponíveis ao sistema. Essas informações podem ser estáticas, como topologia e taxa de transmissão nominal da rede, ou dinâmicas, como serviços disponíveis, localização ou nível de bateria.

Após essas informações serem capturadas, elas são correlacionadas e agregadas com outras informações, e descritas, sob a forma de sintomas, para a função seguinte chamada Analisar.

Analisar

Após receber informações descrevendo o ambiente, a função analisa se os níveis de *QoS* requeridos pelo sistema estão sendo respeitados. Depois, essa função utiliza modelos de predição baseados no conhecimento do sistema e do ambiente para entender como o sistema vai evoluir, e propõe uma requisição de mudança para a função Planejar, definindo as metas e objetivos a serem alcançados, mas não como

o fazer.

Planejar

Com a requisição de mudança definida e os objetivos e metas a serem alcançados detalhados, o sistema precisa planejar as ações a serem realizadas, que podem variar de simples comandos até um complexo *workflow*. Esse plano é, então, passado para a função Executar, e contém o conjunto de ações a serem executadas nos recursos e serviços, de forma a atender aos requisitos de QoS requeridos pelas aplicações.

Executar

Com as ações a serem realizadas definidas, a função Executar é responsável por colocar em prática as modificações no sistema, com o objetivo de oferecer os níveis de QoS requeridos. À medida que as ações que foram definidas são realizadas, o sistema monitora o ambiente para verificar se essas modificações tiveram o efeito desejado, e modifica o conhecimento do sistema, caso o efeito gerado seja negativo, ou ratifica esse conhecimento, se o efeito for positivo.

Conhecimento

A função conhecimento consiste em uma base de dados ou dicionário que auxilia as quatro funções anteriores, provendo informações sobre sintomas, políticas, requisições e planos de mudança. Essas informações são representadas sob a forma de uma linguagem de representação de conhecimento, e podem ser obtidas de três maneiras diferentes: passados diretamente para o sistema, inferidos de fontes externas ou gerados pelo próprio sistema.

O sistema pode obter diretamente do ambiente uma política de adaptação, contendo uma descrição das restrições e preferências que influenciam as decisões autônomas. Por outro lado, o sistema pode acessar um *log* de atividades e inferir quais as melhores políticas de adaptação. A terceira forma de produzir conhecimento é ba-

seada nas próprias ações do sistema, onde ações que trouxeram resultados positivos são valorizadas e as que resultaram em perdas são evitadas.

2.2 Computação evolutiva

Computação evolutiva (EC) é um ramo da computação dedicado ao estudo de uma classe de algoritmos chamados Algoritmos Evolutivos (EA), que se baseiam no princípio *darwiniano* da seleção natural e genética molecular [Eiben e Smith 2003b].

Tais algoritmos apresentam uma característica bem particular chamada tentativa e erro, caracterizada pela geração aleatoria de indivíduos diferentes, e a sobrevivência e reprodução de cada um está diretamente relacionada à sua capacidade de se adaptar às mudanças no ambiente. Quando os indivíduos mais eficientes (adaptados) reproduzem, mutações ocasionais geram novos indivíduos a serem testados. Com isso, à medida que o tempo passa, apenas as mutações que resultarem em vantagens para um indivíduo em relação ao resto da população é que são passadas adiante, através da reprodução.

Considerando que existe uma função de qualidade que mede a eficiência de determinada solução em um dado ambiente, as soluções que apresentarem os melhores valores para essa função de qualidade são escolhidas para servir de semente para as próximas gerações.

2.2.1 Características e ciclo de vida de um algoritmo evolutivo

Existem três características que definem os algoritmos evolutivos [Eiben e Smith 2003a]. Eles são **orientados à população**, onde as soluções candidatas são processadas e avaliadas simultaneamente, e são **adaptáveis**, já que mudam seu comportamento utilizando três técnicas distintas: 1) mutação, em que ocorre uma pequena variação no comportamento do algoritmo; 2) recombinação, juntando informações

de duas ou mais soluções candidatas; e 3) substituição, onde soluções mais eficientes substituem às menos adaptadas. Além disso, EA são **estocásticos**, pois são atribuídas probabilidades às soluções, e aquelas mais eficientes possuem maior probabilidade de serem difundidas. Por outro lado, soluções menos eficientes não são completamente excluídas.

Essas três características estão distribuídas por todo o ciclo de vida de um algoritmo evolutivo. A figura 2.2 mostra os estados que um EA pode assumir e as ações que fazem esse algoritmo mudar de estado.

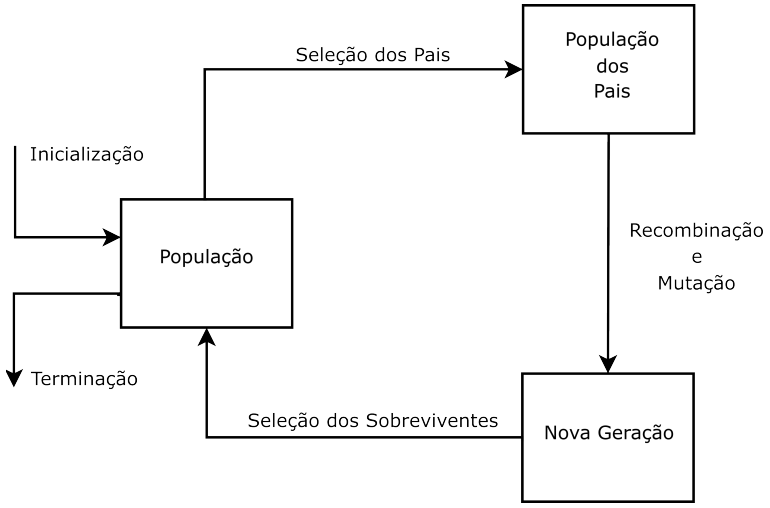


Figura 2.2: Estados de um algoritmo evolutivo. Adaptado de [Eiben e Smith 2003a]

Inicialização

A fase de inicialização define os indivíduos que formarão a primeira geração da população. A escolha do estado inicial pode influenciar no comportamento do EA, alterando o tempo de convergência para uma solução ótima, ou mesmo impossibilitando que o EA atinja essa solução.

Heurísticas específicas ao problema a que o EA está relacionado podem ser utilizadas para definir a população inicial, entretanto, depende do problema em questão, se o esforço computacional extra vai ser compensado pela diminuição no tempo de convergência.

População

A população é o multiconjunto de todas as soluções possíveis (indivíduos), onde cada solução pode aparecer mais de uma vez. Ela pode simplesmente ser definida como o número de vezes que cada indivíduo aparece, ou em algoritmos evolutivos mais complexos, podendo também conter a estrutura que define como os indivíduos se relacionam entre si e com o ambiente.

Assim, a evolução acontece no nível da população, onde um indivíduo reconhecidamente mais eficiente pode servir como semente para as próximas gerações, ou um indivíduo pouco eficiente pode ser substituído por um mais eficiente.

Uma medida para caracterizar uma população é a diversidade, que diz o número de solu¿ões diferentes existentes. A medida da diversidade pode considerar tanto o número de fenótipos ou de genótipos. Fenótipos são a representação abstrata de uma solução no contexto do problema a ser resolvido. Já os genótipos são a representação concreta do fenótipo dentro de um EA.

A busca evolutiva acontece no espaço de genótipos, onde as melhores soluções são identificadas utilizando uma **função de avaliação**. Essa função representa os requisitos que guiam a evolução da população, atribui medidas de qualidade a cada genótipo presente e permite a identificação dos indivíduos mais eficientes.

Seleção dos pais

A função de avaliação é utilizada para comparar a eficiência das diferentes soluções (genótipos), e definir quais serão utilizadas como semente para formar as próximas gerações. Soluções com alta eficiência possuem maior probabilidade de serem escolhidas. Soluções com um baixo fator de qualidade, entretanto, não são descartadas e recebem baixa probabilidade de serem utilizadas. Esse mecanismo permite uma evolução da população e evita que ela fique presa em um valor ótimo local.

Os indivíduos identificados como os mais eficientes somente são considerados pais

se servirem como base de variação. Dessa forma, aplica-se um operador de variação nesses indivíduos, e uma nova geração com um comportamento diferente é formada.

Operadores de variação

Existem dois tipos de operadores de variação: mutação e recombinação. A mutação é um operador unário e modifica o comportamento da solução pai. Esse operador deve ser sempre estocástico, e o valor final de uma solução filho é resultado de uma série de pequenas variações randômicas realizadas na solução pai.

O segundo tipo de operador de variação é a recombinação. Ela combina informações retiradas de dois ou mais pais para formar várias soluções filho. Assim como a mutação, a recombinação também é estocástica, pois a escolha das informações e dos pais de onde essas informações são retiradas é feita de forma randômica.

Dessa forma, quando uma nova geração é formada, mecanismos de seleção de sobreviventes são aplicados para definir quais indivíduos dessa nova geração serão incorporados na população.

Seleção dos sobreviventes

O objetivo do mecanismo de seleção dos sobreviventes é identificar, utilizando a função de avaliação, os indivíduos de uma nova geração que apresentam níveis de eficiência mais elevados do que os atuais membros da população. Dessa forma, com alta probabilidade, os indivíduos menos adaptados são substituídos por esses mais eficientes. O mecanismo de substituição tende a elevar o nível de eficiência da população, eliminando indivíduos menos eficientes.

Assim, uma vez que os sobreviventes são escolhidos, os menos eficientes identificados e substituídos, o ciclo evolutivo está concluído. A questão é definir se um novo ciclo será executado. Ele só será executado novamente se o nível de eficiência geral da população ainda não tenha atingido um patamar aceitável, definido no problema inicial.

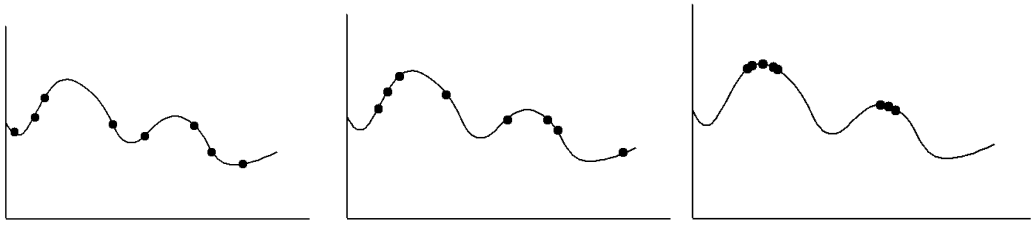
Terminação

A decisão de parar a execução do ciclo evolutivo pode ser feita de duas maneiras. A primeira é utilizar um valor ótimo esperado para a eficiência da população. Entretanto, como algoritmos evolutivos são estocásticos, não existem garantias de que esse patamar de eficiência será alcançado. Nesse caso, o algoritmo nunca terminará. Outro problema é a dificuldade em definir um valor mínimo para a eficiência da população que determina o critério de parada do algoritmo.

Dessa forma, a segunda op¿ão para a terminação do EA é o uso de informações como tempo de execução do algoritmo, número de ciclos evolucionários executados, taxa de aumento da eficiência da população, ou tamanho da diversidade da população.

2.2.2 Comportamento de um algoritmo evolutivo

Considerando que a população inicial de um algoritmo evolutivo é escolhida randomicamente, de inicio a eficiência de cada genótipo (indivíduo) fica igualmente distribuída no espaço de busca (todos os genótipos possíveis). À medida que o EA é executado, os indivíduos da população se deslocam das regiões de baixa eficiência em direção às regiões mais eficientes. No final da execução, a maioria dos indivíduos está concentrada no ótimo global, enquanto alguns podem ter ficado presos em algum ótimo local.



(a) Início (b) Durante (c) Final

Figura 2.3: Eficiência dos indivíduos de uma população em um EA. Adaptado de [Eiben e Smith 2003a]

A figura 2.3 mostra essa migração dos indivíduos em uma população das regiões

de menor eficiência em direção às regiões de alta eficiência. É possível que alguns indivíduos sigam na direção de uma região de boa eficiência, mas que representa um ótimo local, e não o ótimo global do espaço de busca. Essa migração pode ser dividida em dois momentos distintos: exploração, em que os indivíduos se movem para regiões ainda não testadas, e aproveitamento, onde os indivíduos se concentram nas redondezas de regiões sabidamente eficientes.

A migração dos indivíduos geralmente segue uma relação de custo/benefício entre a exploração e o aproveitamento. Muito tempo na exploração gera uma busca ineficiente, mas com maior probabilidade de encontrar o ótimo global. Por outro lado, se o *EA* passar muito tempo na fase de aproveitamento, ele pode ficar preso em uma região ótima local [Eiben e Schippers 1998].

2.3 Exemplos de sistemas autônomos e evolutivos

O uso da computação autônoma é discutido na literatura para lidar com as incertezas inerentes dos sistemas móveis, o que possibilita uma redução na complexidade de desenvolvimento e manutenção das aplicações. Já os sistemas evolutivos ainda estão em fase inicial de utilização, entretanto, é possível encontrar iniciativas para a utilização de algoritmos evolutivos em sistemas móveis. Dessa forma, o restante desse capítulo vai detalhar sistemas autônomos e evolutivos, que apresentam características semelhantes ao *AESPmob*, encontrados na literatura.

MeshMDL - *Coordination middleware for self-organizing applications in ad hoc networks*

MeshMDL [Herrmann et al. 2007] é um sistema de *middleware* que procura oferecer sensibilidade ao contexto e baixo acoplamento às aplicações, permitindo que elas se adaptem às condições das redes *ad hoc*.

A comunicação entre os dispositivos acontece segundo duas abordagens comple-

mentares: 1) *proxies* móveis que representam os serviços e podem ser transferidos para o cliente, com o objetivo de lidar com falhas de comunicação; e 2) comunicação baseada em tuplas, para aumentar o desacoplamento entre as partes.

Baseado nessas duas abordagens, um *deamon* funciona em cada dispositivo, com a responsabilidade de gerenciar quatro subsistemas a seguir: 1) Espaço de eventos é o núcleo da comunicação baseada em troca de mensagens e no espaço de tuplas. Adicionalmente, ele gerencia os espaços virtuais, que são os mecanismos usados para concatenar informações presentes em vários Espaços Virtuais; 2) *agent runtime* é responsável por executar e manter os agentes de cada dispositivo, os quais são entidades autônomas responsáveis por tarefas de gerência do *middleware*; 3) gerência de interação é responsável por qualquer comunicação entre os vizinhos; e 4) camada de conexão genérica é responsável pela descoberta de recursos e manutenção das conexões. Para isso, a camada de conexão genérica pode utilizar várias tecnologias de transmissão diferentes.

O espaço de eventos é a entidade mais importante do MeshMDL. Ele permite, além da troca de mensagens entre dispositivos, uma constante atualização da lista dos vizinhos, já que para cada novo vizinho encontrado é criada uma entrada nesse espaço com informações sobre esse novo dispositivo. Essas informações são utilizadas no processo de busca e adaptação da camada de conexão genérica.

O *MeshMDL* oferece um mecanismo autônomo que monitora os recursos e serviços presentes em dispositivos próximos. Dessa forma, mudanças na topologia e disponibilidade dos recursos podem ser notificadas às aplicações.

Toda a coordenção entre os agentes, entretanto, deve ser implementada pelos desenvolvedores das aplicações, já que o *MeshMDL* apenas oferece os mecanismos de troca de mensagens e notificação de contexto. Quais ações devem ser realizadas em determinado contexto, ou como reagir às mudanças na topologia devem ser tratadas pela própria aplicação.

Context-aware migratory services in ad hoc networks

Esse sistema de *middleware* [Riva et al. 2007] possibilita a construção de aplicações em tempo real para redes *ad hoc*. Ele é baseado na ideia de que redes *ad hoc* não devem ser usadas apenas como provedores de informações, mas também oferecendo uma nova classe de serviços específicos para ambientes ubíquos. Esses serviços capturam, processam e distribuem informação em tempo real de dispositivos presentes em determinadas regiões geográficas.

Dessa forma, o sistema de *middleware* é constituído com base nos seguintes requisitos: 1) sensibilidade ao contexto, onde o ambiente é monitorado e descrito; 2) adaptação voltada para os interesses e as necessidades do usuário, de acordo com o contexto capturado; 3) os serviços devem estar sempre disponíveis, mesmo ante situações imprevistas; e 4) distribuição de código sob demanda.

Baseado nesses requisitos, o sistema de *middleware* foi construído utilizando o Smart Messages [Ravi et al. 2004], e oferece ao desenvolvedor a possibilidade de monitorar e descrever o contexto. Dessa forma, aplicação e *middleware* adaptam-se de acordo com as condições do ambiente, baseando-se em regras de contexto fornecida pelos desenvolvedores. Essas regras são utilizadas para automaticamente avaliar se os serviços podem migrar de um dispositivo para outro.

Adicionalmente, o sistema de *middleware* apresenta características autônomas por meio de dois mecanismos: 1) são as mensagens que fazem o seu roteamento, removendo a necessidade de os dispositivos interagindo estarem disponíveis ao mesmo tempo, além de serem capazes de lidar com desconexões eventuais; e 2) utilizando o conceito de réplica inativa, onde o próprio sistema de *middleware* cria e monitora várias réplicas de um serviço e ativa ou desativa essas réplicas de acordo com a disponibilidade dos dispositivos, sobrecarga em um serviço, ou regras de contexto fornecidas pela aplicação.

QoS-aware service location in mobile ad Hoc networks

Utilizando o sistema de *middleware* AdHocWS [Liu e Issarny 2004], esse trabalho possibilita a alocação de serviços em uma rede *ad hoc* de acordo com políticas de *QoS* fornecidas pelo administrador. Os serviços são alocados e utilizados de acordo com funções de benefício que medem a utilidade do serviço consoante as características percebidas do ambiente.

Essas funções são calculadas por meio de valores de QoS obtidos do ambiente. Assim, o AdHocWS procura oferecer um arcabouço para classificação de parâmetros de QoS que seja: 1) expressivo o suficiente para capturar os detalhes de QoS; e 2) o mais simples possível, para minimizar o custo de processamento com a gerência de QoS.

Dessa forma, os parâmetros de QoS são divididos em dois grupos:

1) O grupo de *QoS* dos serviços procura descrever desempenho, lidando com restrições de latência, *jitter* e taxa de vazão; **confiabilidade**, que mede a probabilidade de um serviço estar disponível, tempo médio entre falhas ou o número de falhas em um determinado tempo; **segurança**, que mede o grau de confidencialidade das informações; e **corretude transacional**, que lida com as propriedades ACID - atomicidade, consistência, isolamento e durabilidade; e

2) o grupo de *QoS* dos Recursos lida com restrições de recursos do ambiente, como processamento, memória, bateria ou largura de banda.

Os parâmetros e valores de *QoS* são então agregados em uma única função de benefício que permite caracterizar e comparar os serviços e recursos, de acordo com níveis de *QoS* fornecidos pelos desenvolvedores. Assim, esses serviços e recursos são utilizados de acordo com o valor da função de benefício, além de dinamicamente alocados visando a maximizar esse valor e atender os níveis de *QoS* requeridos pelas aplicações.

Coordination middleware supporting rapid deployment of ad hoc mobile systems

Esse trabalho apresenta o sistema de *middleware* Lime [Handorean et al. 2003], que fornece mecanismos para lidar com a mobilidade de dispositivo e de código, utilizando três conceitos básicos: arquitetura orientada a serviços, notificação baseada em eventos e compartilhamento seguro do espaço de tuplas.

A primeira funcionalidade do Lime é a gerência transparente do contexto. Para isso, ele divide o espaço de tuplas entre os vários dispositivos presentes em um ambiente e cria regras para acessar e compartilhar esse espaço de acordo com o contexto.

Por outro lado, o Lime fornece mecanismos para especificar ações que devem ser realizadas quando determinada condição lógica é satisfeita. Uma tupla R(s,p) é inserida em um determinado espaço de tuplas. Quando as condições descritas por p são atendidas, o código s é executado. Entretanto, o comportamento dessas ações deve ser norteado pelos recursos existentes no dispositivo e pelas restrições de QoS.

O mecanismo de busca do Lime é baseado em mensagens de anúncio dos serviços difundidas pela rede. Essas mensagens contêm informações sobre a funcionalidade dos serviços e as restrições de recursos dos dispositivos. Adicionalmente, o Lime oferece também aos dispositivos a possibilidade de registrar interesse em determinados eventos, para serem notificados quando estes ocorrerem.

A segurança é inserida no modelo para garantir que apenas dispositivos autorizados acessem determinado espaço de tuplas. Isso é feito por mecanismos de autenticação utilizando senha. Sempre que um dispositivo insere uma tupla com restrições de segurança, ele também informa uma senha que deverá ser utilizada para acessar os dados presentes nessa tupla.

A flexible middleware for service provision over heterogeneous pervasive networks

[Loureiro et al. 2006] apresentam o middleware *Wings*, que permite a atualização sob demanda do sistema de *middleware* e a descoberta e execução de serviços baseados em protocolos e linguagens diversas. Mais ainda, o *Wings* oferece informações sobre o contexto para os serviços e as aplicações.

O objetivo do *Wings* é oferecer um sistema de *middleware* para computação pervasiva baseado em *SOA*, que seja flexível e extensível. Dessa forma, a sua arquitetura é baseada em quatro módulos principais. O primeiro é o módulo de componentes COMPOR, que permite atualização dos componentes da aplicação em tempo de execução. O segundo módulo é o *pervasive networking*, que possibilita a descoberta dinâmica de dispositivos e serviços. Já o módulo de sensibilidade ao contexto oferece mecanismos para coletar informações de contexto do ambiente e descrevê-las às aplicações. O último módulo lida com informações internas do *Wings* e está dividido em duas partes: o *middleware facade* oferece interfaces para as aplicações acessarem funcionalidades específicas do *Wings*, e o *context reasoner* é um espaço onde todos os executáveis das aplicações que façam uso de informações de contexto são armazenados e executados.

SLACER: A self-organizing protocol for coordination in peer-to-peer networks

Implementado visando redes P2P, o SLACER [Hales e Arteconi 2006] auto-organiza os nós em uma rede social artificial de alta cooperação e características *small world*, onde os dispositivos estão separados por no máximo cinco vizinhos [Meyer et al. 2002]. Os pares periodicamente trocam listas de seus vizinhos e verificam quem nessa lista possui o maior índice de cooperação. Se esse valor for maior do que o índice local, ele incorpora o comportamento desse nó recebido. Dessa forma, uma estratégia eficiente vai sendo difundida pela rede, evoluindo o comportamento de cada nó na rede P2P.

2.4 Conclusões

Neste capítulo, foi apresentada uma visão geral da computação autônoma e evolutiva, discutindo seus princípios e desafios, além de destacadas algumas soluções que utilizam conceitos de autonomia e evolução em aplicações móveis.

Dessa forma, a computação autônoma aparece como uma solução útil e bastante empregada para lidar com a incerteza e dinamicidade dos ambientes móveis. Normalmente formada por pequenos módulos autônomos e distribuídos, a dificuldade está no mapeamento dos requisitos de alto nível das aplicações para atividades locais em cada módulo.

Esses módulos cooperam e coordenam suas atividades para oferecer um comportamento global desejado. Com isso, buscando melhorar o desempenho de um sistema móvel, algoritmos evolutivos podem ser utilizados. O uso doa algoritmos evolutivos (EA) permite que o sistema evolua mediante a interação dos diversos módulos que o forma.

O próximo capítulo lista e detalha os principais requisitos necessários à construção de aplicações móveis distribuídas orientadas a serviço.

Capítulo 3

SERVIÇOS ESSENCIAIS EM APLICAÇÕES MÓVEIS

Este capítulo discute os serviços essenciais em aplicações móveis, e como esses serviços se adaptam às diferentes arquiteturas de redes móveis. Na seção 3.1 é feita uma breve introdução sobre a temática. Já na seção 3.2 os serviços essenciais são detalhados, e na seção 3.3 é discutido o papel de requisitos transversais como sensibilidade ao contexto e adaptação, autonomia e provisão de *QoS* na correta execução dos serviços essenciais. Já a serção 3.4 mostra como cada serviço essencial influencia e é influenciado pelos outros serviços essenciais. Ainda neste capítulo, na seção 3.5, é detalha como os serviços essencias se adaptam às diferentes tecnologias de redes sem fio. Para finalizar, a seção 3.6 apresenta algumas conclusões.

Mais ainda, a seção 3.4 analisa como os serviços essenciais se adéquam às diferentes arquiteturas de redes móveis, e qual o papel dos requisitos transversais nesse contexto.

3.1 Introdução

Aplicações móveis distribuídas, caracterizadas pela dinamicidade e incerteza dos recursos, requerem primitivas de programação adaptáveis e autônomas, que ofere-

çam ao desenvolvedor a possibilidade de se concentrar unicamente nos requisitos funcionais de suas aplicações. Normalmente, essas primitivas são oferecidas sob a forma de sistemas de *middleware*.

Os sistemas de middleware são implementados sob a forma de uma camada que fica entre as aplicações e os sistemas operacionais, e são responsáveis por implementar requisitos não funcionais como interação assíncrona, portabilidade, reconfiguração dinâmica e adaptação, provisão de QoS e segurança [Grigoras 2006].

Apesar de abstrair a complexidade dos ambientes móveis, tais sistemas ainda estão longe de tratar todos os aspectos requeridos por aplicações móveis. Um dos principais argumentos utilizados para justificar o uso de sistemas de *middleware* é que eles promovem a interoperabilidade. Geralmente, isso só é verdade quando todos os dispositivos utilizam o mesmo *middleware*. Em ambientes dinâmicos e heterogêneos, não é uma boa prática impor a todos os dispositivos a necessidade do uso de um único sistema de *middleware*.

Com o objetivo de promover interoperabilidade de troca de mensagens, dentre outros benefícios, arquiteturas orientadas a serviço são utilizadas [Erl 2005]. Muito embora SOA seja um passo à frente para melhorar interoperabilidade, outros aspectos precisam ser considerados. Por exemplo, as instâncias distribuídas de uma aplicação precisam concordar na linguagem de descrição de serviço, no protocolo de descoberta de serviço, em como a coordenação de serviços deverá se comportar, em como compor serviços complexos utilizando serviços mais simples, qual protocolo de troca de mensagens deve ser utilizado, e quais políticas de incentivo à cooperação são apropriadas. Mais ainda, aplicações sensíveis a contexto precisam concordar na linguagem de descrição de contexto ou quais ações devem ser tomadas quando uma notificação de mudança de contexto é recebida.

Para entender como esse nível de interoperabilidade pode ser alcançado em aplicações móveis distribuídas orientadas a serviço, este capítulo vai destacar alguns serviços essenciais (*essential services* - ES) presentes nessas aplicações e os desafios

em implementá-los.

Os ES são serviços utilizados por aplicações móveis distribuídas e foram identificados como Troca de Mensagens, Suporte a Mobilidade, Descrição, Descoberta e Coordenação de Serviços e Segurança. Adicionalmente, para lidar com a incerteza e dinamicidade inerentes aos ambientes móveis, esses serviços são implementados utilizando três requisitos transversais: sensibilidade ao contexto e adaptação, autonomia e provisão de QoS.

3.2 Serviços essenciais

Orientação a serviço permite que um problema maior seja decomposto em partes atômicas menores [Erl 2005]. Essa abordagem facilita a implantação, gerência, manutenção e evolução dos sistemas móveis. Adicionalmente, como muitos dispositivos que formam esses sistemas apresentam restrições de recursos, e em muitos casos um servidor central dedicado não está disponível, essas partes atômicas são distribuídas por toda a rede. Consequentemente, o sistema de *middleware* precisa oferecer às aplicações mecanismos para encontrar, acessar e compor essas partes menores de forma segura, sensível ao contexto e tolerante a falhas.

Para decompor os serviços essenciais, esta dissertação utilizou um arcabouço de requisitos não funcionais (NFR) [Subramanian e Chung 2001]. Ele trata um requisito como um objetivo a ser alcançado na fase de desenvolvimento. A figura 3.1 mostra a notação do arcabouço NFR utilizada. Cada serviço essencial foi decomposto em um *Softgoal Interdependency Graph* (SIG). Um **softgoal** representa o requisito não funcional a ser modelado. Operacionalização é outro tipo de *softgoal* que representa uma técnica de desenvolvimento para um NFR específico, e pode ser decomposto em outros *softgoals* mais específicos. Para isso, operadores E e OU especificam se vários *softgoals* são necessários para formar um *softgoal* mais abrangente (E), ou apenas um desses *softgoals* deverá ser utilizado (OU).

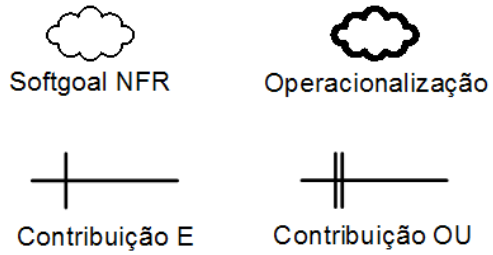


Figura 3.1: Notação do *SIG*

3.2.1 Troca de mensagens

Troca de mensagens é o mecanismo que permite processos distribuídos se comunicarem. A figura 3.2 mostra a Troca de Mensagens sendo dividida em três dimensões (*Softgoals*): garantia de entrega, ordenação e forma de troca [Coulouris e Dollimore 2005].

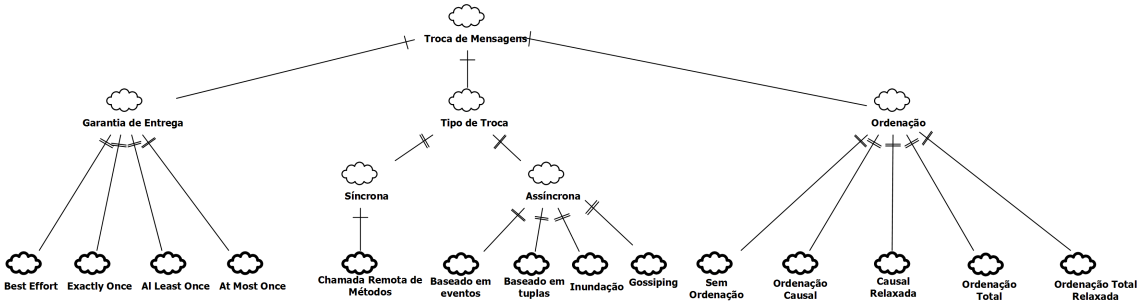


Figura 3.2: *SIG* da Troca de Mensagens

A primeira dimensão, chamada garantia de entrega, pode ser dividida posteriormente em quatro abordagens. *1) Best effort* não garante que a mensagem será entregue, não notifica a origem da mensagem sobre falhas na comunicação, nem retransmite uma mensagem para aumentar a chance dela ser entregue. Essa abordagem apresenta a menor confiabilidade dentre as cinco possíveis. Por outro lado, exibe também a menor latência de entrega. Ela é útil em aplicações que aceitam um certo nível de falha, mas requerem baixa latência, como transmissão em tempo

real de conteúdo multimídia.

2) At-least-once garante que a mensagem chega ao destino pelo menos uma vez, possivelmente retransmitindo a mesma mensagem várias vezes. Ela é útil em aplicações idempotentes, onde, executando uma operação várias vezes, tem o mesmo efeito que executá-la exatamente uma vez.

3) At-most-once, da mesma forma que a abordagem *Best Effort*, garante que a mensagem não é recebida mais de uma vez. A diferença aqui é que a fonte da mensagem é notificada em caso de falha na transmissão. Essa abordagem tem um custo menor na rede do que a abordagem *At-least-once* e permite que técnicas de tolerância a falhas sejam aplicadas.

4) Exactly-once é uma forma de entregar uma mensagem ao destinatário exatamente uma vez. Ela é útil em operações não idempotentes, onde executar uma operação mais de uma vez pode resultar em inconsistências.

A segunda dimensão que define a Troca de Mensagens é a ordenação. As mensagens que saem da origem podem chegar em uma ordem diferente no destino daquela na qual foram enviadas, o que pode acarretar inconsistências no estado da aplicação. Entretanto, em determinadas situações, restrições de ordenação podem ser relaxadas [Correa e Barbosa 2009].

A ordenação é dividida da seguinte maneira:

1) Total order restringe que as mensagens cheguem ao destino exatamente na mesma em que foram enviadas; *2) relaxed total order* não é tão restrita como a *total order*, já que permite que mensagens cheguem fora de ordem por um pequeno intervalo de tempo ou um dado número máximo de mensgens; *3) causal order* impõe ordenação total apenas às mensagens que possuem algum tipo de relação de causa e efeito. Mensagens que não apresentam essa relação não possuem restrições; na *4) relaxed causal order*, mensagens que representam uma causa podem chegar depois da mensagem que representa o efeito correspondente por um pequeno intervalo de tempo ou um pequeno número de mensagens; e a maneira *5) no order* relaxa

totalmente a ordem na qual as mensagens são entregues.

A terceira dimensão é a forma de troca. As mensagens podem ser trocadas de forma assíncrona ou síncrona. Na troca de mensagens síncrona, origem e destino bloqueiam até que a mensagem seja entregue. Ela é implementada utilizando chamada remota de procedimento (*remote procedure call - RPC*), ou métodos similares [Coulouris e Dollimore 2005]. *RPC* acopla as partes interagindo e é indicado apenas em ambientes móveis estáveis.

Por outro lado, a troca de mensagens assíncrona desacopla as partes que interagem e deve ser priorizada em ambientes móveis. Esse desacoplamento pode ser dividido em três níveis [Eugster et al. 2003]: 1) desacoplamento de tempo permite que as mensagens sejam trocadas entre processos que não estejam disponíveis simultaneamente; 2) desacoplamento de espaço elimina a necessidade de conhecer o endereço do destinatário; e 3) desacoplamento de sincronia oferece primitivas de envio e recebimento de mensagens que não bloqueiam o fluxo de execução do algoritmo até que o destino receba a mensagem.

A interação assíncrona pode ser implementada utilizado as quatro abordagens diferentes citadas a seguir [Rezende et al. 2008,Cabri et al. 2006,Khelil et al. 2007]:

1) Baseado em eventos. Nesse tipo de interação, uma aplicação notifica o interesse em receber mensagens (eventos) de um determinado tipo em um serviço de subscrição. Quando outras aplicações produzem um novo evento, elas o entregam ao serviço de subscrição, que fica responsável por encaminhar esse novo evento a todas as aplicações que subscreveram interesse nesse tipo de evento.

Se alguma aplicação interessada em um evento não estiver disponível, três estratégias podem ser utilizadas: armazenar o evento e tentar entregá-lo posteriormente até um número máximo de vezes; utilizar um terceiro dispositivo que fica responsável por encaminhar esse evento; ou simplesmente descartar o evento.

2) Baseado em tuplas. Uma mensagem a ser enviada (tupla) é inserida em um espaço compartilhado por todas as aplicações, chamado espaço de tuplas. As

aplicações proativamente acessam o espaço de tuplas e procuram tuplas com certas características.

As duas abordagens oferecem desacoplamento de sincronia e desacoplamento parcial de tempo e localização. O desacoplamento de tempo é parcial, já que as tuplas e eventos só ficam disponíveis por um intervalo máximo de tempo. Por outro lado, o desacoplamento de localização é parcial porque, ou as aplicações escolhem o espaço de tuplas e o serviço de subscrição correto, ou existe uma infraestrutura de compartilhamento de tuplas e eventos entre diferentes espaços de tuplas e serviços de subscrição.

A diferença principal é que enquanto o mecanismo baseado em eventos oferece uma infraestrutura que proativamente notifica as aplicações, o baseado em tuplas é mais espontâneo, e as próprias aplicações procuram por determinadas tuplas quando necessário. Consequentemente, o modelo baseado em eventos é mais indicado para sistemas com baixa dinamicidade, enquanto soluções baseadas em tuplas são mais indicadas a ambientes com alta incerteza. Ambas as abordagens são mais adequadas em ambientes infraestruturados. Entretanto, em ambientes desestruturados outras abordagens como *flooding* e *gossiping* são mais adequadas.

3) ***Flooding*** (inundação) consiste nos próprios dispositivos encaminhando as mensagens, até um número máximo de encaminhamentos (Time-to-Live - TTL), para todos os vizinhos conhecidos. *Flooding* é a abordagem mais fácil de ser implementada e tem uma elevada taxa de entrega. Por outro lado, introduz um custo muito alto na rede. Para minimizar o número de mensagens, o valor do TTL é mantido baixo. Entretanto, isso diminui a taxa e aumenta a latência de entrega de uma mensagem.

Quando o algoritmo de inundação é utilizado, dois extremos devem ser cuidadosamente analisados. Em uma rede com alta densidade, esse algoritmo pode gerar uma tempestade de mensagens. Por outro lado, em uma rede esparsa, pode acontecer uma partição na rede e alguns dispositivos ficarem inacessíveis.

Já o 4) ***gossiping*** é uma alternativa para minimizar o custo na rede, mantendo a taxa de entrega alta. Quando um dispositivo encontra um novo vizinho, ele envia parte das suas mensagens para esse vizinho, e também recebe uma parte das mensagens dele. Essa abordagem melhora o problema das redes densas, já que as mensagens são enviadas apenas para um subconjunto de dispositivos. O problema das redes esparsas também é parcialmente solucionado, pois ao encontrar um novo vizinho, ele pode reencaminhar uma mensagem já enviada.

As dimensões que formam um serviço essencial não são implementadas isoladamente, e a escolha por determinada estratégia em uma dimensão impacta no funcionamento das outras dimensões. Por isso, ao final de cada subseção dos serviços essenciais, esta dissertação analisa como as diferentes estratégias de implementação de cada dimensão impactam nas outras dimensões. A seguir são analisadas as dependências internas ao serviço essencial Troca de Mensagens.

Dependências internas da Troca de Mensagens

A garantia de entrega impacta diretamente na ordenação. Aplicações com restrições de ordenação não devem ser implementadas utilizando a abordagem *best effort*, e *at most once* deve ser utilizado apenas quando a própria aplicação lidar com perda de mensagens. Dessa forma, *at least once* e *exactly once* são mais indicados.

Interações síncronas são normalmente implementadas utilizando *total ordering* [Andrews 1991]; e, entretanto, relaxamento de ordenação pode ser utilizado se a camada de *middleware* lidar com a reordenação antes de entregar as mensagens às aplicações. Adicionalmente, operações de envio que bloqueiam devem ser implementadas utilizando *at least once* e *exactly once*.

3.2.2 Suporte à mobilidade

Suporte à mobilidade é um requisito importante em aplicações móveis e pode ser dividido em mobilidade de usuário, dispositivo e código [Popescu-Zeletin et al.

2006].

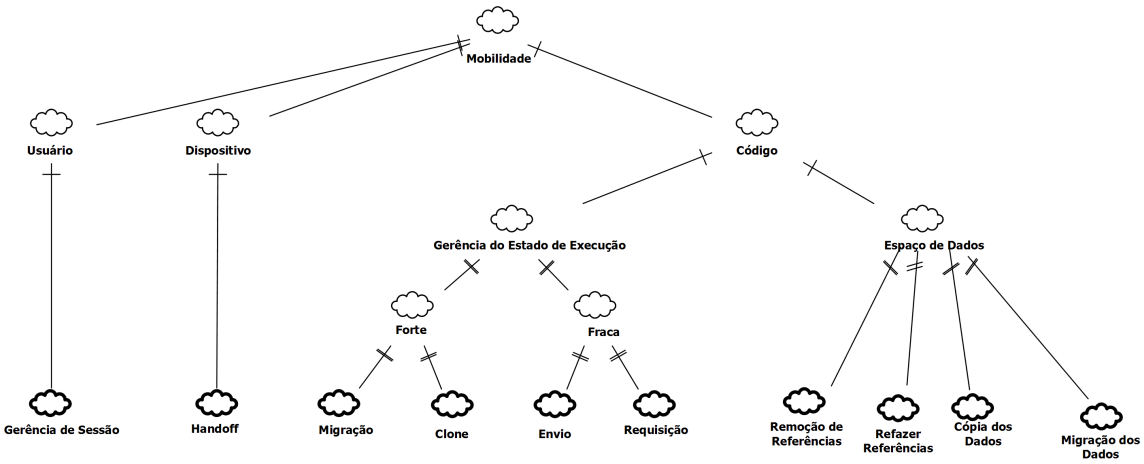


Figura 3.3: *SIG* do suporte à mobilidade

A figura 3.3 mostra o *SIG* do suporte à mobilidade. Mobilidade do usuário é a capacidade do usuário acessar serviços da aplicação a partir de vários dispositivos, mantendo seu identificador, interações passadas e sessões abertas. Para isso, deve ser disponibilizado às aplicações um mecanismo para salvar, compartilhar e transferir informações sobre um usuário a qualquer dispositivo.

Quando um dispositivo se distancia da rede a que está conectado, e novas redes são detectadas, uma decisão e execução de *handoff* é disparada [Andrade e Logrippo 2006]. *Handoff* é a capacidade de o dispositivo trocar a rede de acesso sem perder conectividade. Esse tipo de mobilidade é chamado de mobilidade de dispositivo.

A terceira dimensão é caracterizada por mobilidade de código [Fugetta et al. 1998], e pode ser definida por dois aspectos complementares:

1) Migração de binário e estado de execução é a habilidade de migrar um serviço, ou uma pequena parte dele, para um dispositivo remoto. Se essa migração é acompanhada do estado de execução, ela é chamada de migração forte. Caso seja apenas o código do serviço e alguns dados de inicialização, mas não o estado propriamente dito, ela é chamada migração fraca.

Adicionalmente, a mobilidade forte pode ser executada de duas formas. Na primeira, chamada migração, a execução local é suspensa, o executável e o estado

de execução são transferidos para uma máquina remota, e a execução é resumida nesse dispositivo remoto. A segunda forma é chamada de clone, onde o executável e o estado são copiados para a máquina remota, mas uma cópia local do serviço continua operacional.

Já a migração fraca pode ser iniciada proativamente pelo dispositivo que hospeda o serviço, ou requisitada por um dispositivo remoto.

2) Outro aspecto que caracteriza a mobilidade de código é a gerência do espaço de dados. Um serviço pode estar ligado a recursos locais, que podem ser recursos de *hardware* ou *software*. Dessa forma, antes de executar a migração, deve ser verificado se esses recursos locais poderão ser acessados a partir do dispositivo remoto. Caso os recursos não sejam acessíveis, eles podem também ser migrados.

Independentemente da forma como a mobilidade de código é executada, o suporte à mobilidade deve verificar se o dispositivo remoto conseguirá oferecer os níveis de *QoS* requeridos pelas aplicações e permitir o acesso aos recursos necessários.

Dependências internas do suporte à mobilidade

Quando um usuário troca de dispositivo (mobilidade de usuário), o novo dispositivo pode não ter disponível todas as interfaces de comunicação, e qualquer comunicação corrente deve ser redirecionada para outras interfaces (Mobilidade de Dispositivo). Adicionalmente, a Mobilidade de Código pode também ser disparada, já que serviços e recursos acessados por uma interface podem não estar disponíveis na outra.

3.2.3 Descrição de Serviços

Descrição de serviços é uma técnica semiformal ou formal utilizada para descrever um serviço. A ideia é permitir uma comparação entre os requisitos da aplicação e funcionalidades e capacidades de um determinado serviço.

A figura 3.4 mostra o *SIG* da descrição dos serviços. Ele é decomposto em duas

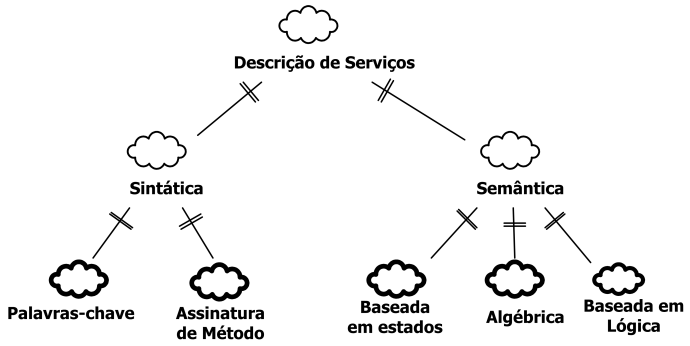


Figura 3.4: *SIG* da descrição de serviços

dimensões. A descrição sintática especifica as funcionalidades e capacidades de um serviço utilizando duas abordagens. A primeira abordagem é baseada em palavra-chave. Nela, as informações sobre um serviço são descritas associando palavras-chave a esse serviço. Já na segunda abordagem, baseada na assinatura dos métodos, é possível descrever um serviço informando o tipo de parâmetros de entrada, tipo de retorno e nome do método. O uso de descrição baseada em assinatura de método é mais expressivo do que palavra-chave. As duas abordagens, entretanto, tornam a aplicação muito inflexível e dificultam a sua manutenção e evolução.

Dessa forma, a descrição semântica melhora a expressividade na descrição de um serviço e permite que inferências possam ser feitas no estado dos serviços para derivar informações de mais alto nível. A descrição semântica pode ser dividida em três abordagens diferentes.

A abordagem baseada em estado especifica entrada, saída, précondições e efeitos (*input, output, preconditions and effects* - *IOEPs*) de um serviço utilizando lógica descritiva, como a *web ontology language (OWL)* [W3C 2008] e linguagem de marcação de regras *rule markup language* - *RuleML* [Boley et al. 2001]. Parâmetros de entrada e saída não estão relacionados apenas a tipos de dados, mas também a conceitos descritos em uma ontologia. Já as précondições e efeitos capturam a mudança de estado dos parâmetros durante a execução de um serviço. As précondições são restrições que precisam ser atendidas antes da invocação de um serviço,

enquanto os efeitos descrevem as condições que precisam atendidas após a execução desse serviço [Schade et al. 2004].

A segunda abordagem é a algébrica [Schade et al. 2004], que descreve um serviço utilizando um subconjunto de primitivas presentes em linguagens funcionais. As principais características dessa abordagem são a utilização de uma linguagem de programação para descrever e inferir sobre as ontologias, e o uso de sistemas de tipos para especificar o comportamento do sistema.

Já a abordagem baseada em lógica descreve um serviço utilizando lógica formal, e permite a utilização de regras e axiomas para inferir na descrição de um serviço, além de conseguir provar características de um serviço [Chang et al. 2007].

3.2.4 Descoberta de serviços

A descoberta de serviços possibilita que os serviços possam ser descobertos e acessados quando se tornarem disponíveis, e permitem a criação de aplicações móveis robustas e flexíveis. A eficiência do mecanismo de busca, entretanto, está fortemente acoplada a informações na camada de rede, sejam estáticas, como arquitetura, ou dinâmicas, como latência e topologia. A figura 3.5 mostra as diferentes estratégias para implementar mecanismos de descoberta de serviços.

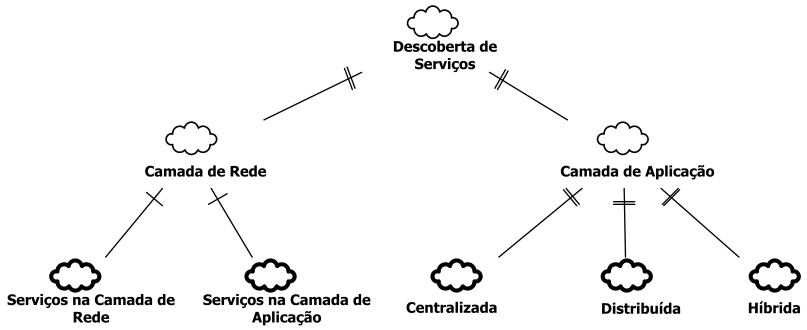


Figura 3.5: *SIG* da descoberta de serviços

A abordagem baseada na camada de rede usa pacotes de descoberta de rotas

para levar mensagens de descoberta de serviços. Dessa forma, quando um serviço é encontrado, a mensagem de resposta sobre como invocar um serviço é enviada no mesmo pacote que a resposta de descoberta de rota. Essa abordagem minimiza o custo na rede e o consumo de energia; entretanto, a expressividade na descoberta é restringida, já que é utilizada uma abordagem de descrição baseada em palavras-chave. Outro problema é garantir interoperabilidade, pois os algoritmos de roteamento precisam implementar o mecanismo de descoberta.

Mecanismos baseados na camada de rede são divididos em descoberta de serviços de rede e descoberta de serviços de aplicação. Exemplos de serviços na camada de rede são decisão e execução de *handoff*. Por outro lado, serviços de aplicação podem ser qualquer serviço que implemente um requisito funcional da aplicação.

Utilizar mecanismos na camada de rede para descobrir serviços na camada de aplicação quebra a separação tradicional de camadas. Essa abordagem é chamada de *Cross-layering*, e informações coletadas em uma camada ficam disponíveis às outras camadas, possibilitando a adaptação e otimização do comportamento de uma camada às condições momentâneas das outras camadas. Por outro lado, o uso de *cross-layering* pode aumentar a complexidade para manter e evoluir uma camada, já que ele pode acoplar e enrijecer a interação das camadas, dificultar a padronização de cada camada, além de resultar em perda de desempenho se todas as interações entre as camadas não forem cuidadosamente analisadas [Kawadia e Kumar 2005].

Já a descoberta de serviços na camada de aplicação respeita a separação de camadas, e é mais fácil de implementar e manter. Ela pode ser dividida em duas dimensões [Seyed Amin Hosseini e Seno 2007]: 1) quando publicar informação sobre o serviço; e 2) como armazenar informação sobre o serviço.

O momento de publicar um serviço é dividido em *pull-based* e *push-based*. O primeiro publica as informações sobre um serviço no momento em que elas ficam disponíveis, e são indicadas para sistemas com baixa dinamicidade. Já os modelos *push-based* descobrem as informações apenas quando requisitadas. Eles devem ser

utilizados em situações de maior dinamicidade.

A segunda abordagem diz como armazenar as informações sobre um serviço, e pode ser implementada de três formas diferentes. A primeira é totalmente centralizada, e usa um servidor dedicado para armazenar todos os registros (informações sobre serviços). Em geral, a abordagem centralizada apresenta uma baixa latência e é fácil de implementar. Por outro lado, esse servidor representa um ponto único de falha. A abordagem centralizada deve ser utilizada em ambientes pouco dinâmicos ou infraestruturados, onde exista um dispositivo com abundância de recursos.

Abordagens totalmente distribuídas são mais robustas e menos susceptíveis a falhas. Por outro lado, elas têm uma latência de descoberta e um custo na rede maior e são indicadas para redes desestruturadas e muito dinâmicas, ou em situações em que o servidor central esteja indisponível ou sobrecarregado.

A terceira solução é chamada de híbrida. Ela procura utilizar os pontos positivos das abordagens centralizada e distribuída, como a alta disponibilidade e baixa latência dos mecanismos centralizados, além da robustez dos mecanismos distribuídos. Em ambientes muito dinâmicos, entretanto, a abordagem híbrida pode apresentar custo e latência maiores do que a totalmente distribuída, e, em topologias estáticas, a abordagem centralizada é mais eficiente.

3.2.5 Coordenação de serviços

A coordenação de serviços reflete a habilidade de coordenar a execução distribuída de determinada tarefa, criando aplicações escaláveis e flexíveis baseadas em agentes autônomos e componentes que se organizam e coordenam suas ações de forma adaptativa [Mamei e Zambonelli 2005].

A figura 3.6 mostra o *SIG* da coordenação de serviços. A primeira dimensão descreve a forma de coordenação, e pode ser implementada de duas maneiras [Mamei e Zambonelli 2005]. Na coordenação baseada em infraestrutura, existe uma infraestrutura que conhece os agentes e componentes disponíveis, e sabe como en-

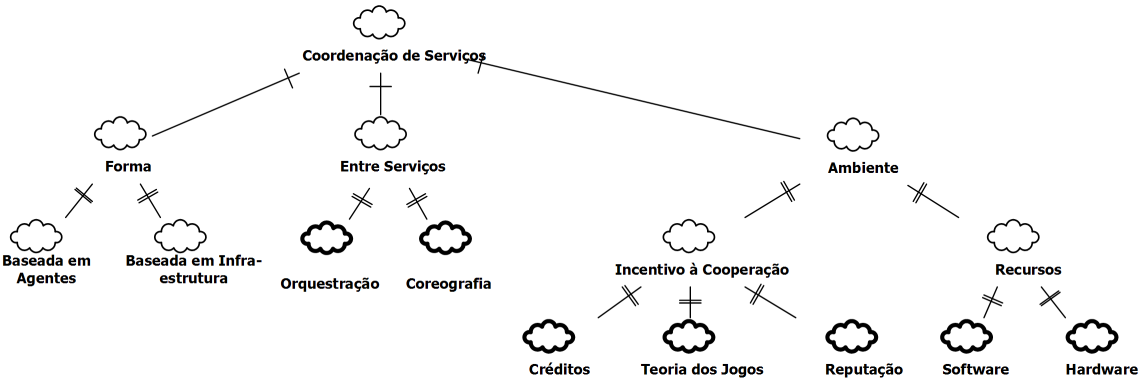


Figura 3.6: *SIG* da coordenação de serviços

contrar, invocar e compor os serviços. A segunda forma de coordenar as atividades é baseada em agentes, que são unidades autônomas do sistema capazes de sentir o ambiente, inferir quais ações devem ser tomadas com base em políticas locais e atuar nos recursos com o objetivo de atingir um comportamento global desejável.

A diferença principal entre essas duas abordagens é que, enquanto na primeira o conhecimento está na infra-estrutura, responsável por coordenar a execução das tarefas distribuídas, na baseada em agentes, cada entidade possui sua própria lógica e conhecimento sobre como executar uma dada tarefa.

A segunda dimensão trata da interação dos serviços, e está dividida em orquestração e coreografia [Erl 2005]. A orquestração descreve e gerencia todas as ações necessárias para a execução de serviços complexos, baseados na composição de serviços mais simples. Já a coreografia é um acordo feito entre as partes do sistema para permitir a interoperabilidade. Ela descreve como invocar um serviço, especificando quais mensagens precisam ser trocadas, quais as précondições e os efeitos gerados por essa invocação.

Já a terceira dimensão que caracteriza a coordenação de serviços especifica quais são as políticas de incentivo à cooperação e como acessar e descrever os recursos disponíveis. Em aplicações móveis distribuídas, as tarefas geralmente são realizadas por vários dispositivos, em muitos casos com fortes restrições de recursos. Dessa forma, são úteis as políticas de incentivo à cooperação para aumentar o nível geral

de cooperação do ambiente. Elas podem ser divididas em três abordagens distintas [Conti et al. 2004, Jong et al. 2008, Yu 2007].

Na solução baseada em reputação, um dispositivo monitora o comportamento dos seus vizinhos e aumenta ou diminui o valor dessa reputação de acordo com o quanto cada vizinho cooperou. Ela é posteriormente difundida pela rede e agregada com informações disponíveis em outros dispositivos. Dessa forma, nós com maior valor para a reputação são considerados mais confiáveis e são priorizados quando necessitam de recursos remotos.

A solução baseada em reputação é feita de forma descentralizada, e é útil em ambientes mais dinâmicos, onde a mobilidade dos dispositivos é alta e não existe um dispositivo centralizador. Por outro lado, ela é custosa, já que muitas mensagens de difusão das reputações são trocadas.

Outra abordagem para melhorar o nível de cooperação é baseada em créditos. Cada requisição de serviço possui os créditos necessários para a realização dessa requisição. Dispositivos que cooperam mais possuem maior número de créditos. Esses créditos podem ser usados posteriormente quando esse dispositivo necessitar executar um serviço remoto.

Essa abordagem é fácil de implementar se existe um dispositivo centralizador responsável por gerenciar o uso dos créditos; entretanto, se esse dispositivo ficar inacessível, toda o mecanismo fica comprometida.

A terceira abordagem, baseada na teoria dos jogos, é utilizada para modelar e analisar as interações estratégicas entre agentes. Dessa forma, para cada ação possível de um agente, uma função de utilidade descreve o lucro e o custo associado a essa ação. O objetivo da Teoria dos Jogos é maximizar a eficiência geral da rede, mediante de uma recompensa aos nós que cooperam e uma punição àqueles que não cooperam.

Outro aspecto importante da coordenação de serviços é como descrever e acessar os recursos disponíveis no ambiente. Esses recursos podem ser tanto de *hardware*

como *software*. Em sistemas móveis, a disponibilidade e eficiência de cada recurso pode oscilar bastante. Com isso, deve ser oferecido às aplicações um mecanismo de gerência autônoma desses recursos. Por exemplo, o mecanismo de invocação de um serviço pode ser dinamicamente adaptado ou falhas em um dispositivo de armazenamento devem ser detectadas e resolvidas.

Dependências internas da Coordenação de Serviços

A natureza descentralizada e autônoma da coordenação baseada em agentes requer uma coordenação com baixo acoplamento. Mais ainda, como esses agentes normalmente estão executando em dispositivos com restrição de recursos, os protocolos de orquestração e coordenação devem ser leves, descentralizados e gerenciados localmente por parte de cada agente. Por isso, as soluções devem ser autoinstaláveis e configuráveis para permitir que sejam encontradas no próprio ambiente e utilizadas automaticamente.

Adicionalmente, como a composição de serviços é mais intermitente em sistemas baseados em agentes, as políticas de incentivo à cooperação que devem ser utilizadas são baseadas em reputação ou teoria dos jogos. Políticas baseadas em reputação trocam mais mensagens e estão sujeitas à manipulação das informações, mas são mais fáceis de implementar e são descentralizadas. O problema com a teoria dos jogos é a dificuldade em modelar todas as interações possíveis e definir uma função de utilidade que produza um estado global desejável.

Por outro lado, abordagens que utilizam infraestrutura são mais fáceis de implementar, manter e evoluir, além de mais confiáveis. O uso de servidores dedicados pode facilitar o desenvolvimento de políticas de incentivo à cooperação baseadas em créditos, tornar mais efetivo o monitoramento e adaptação do ambiente e acesso aos recursos, e aumentar a disponibilidade dos serviços. Em muitas situações, entretanto, não é possível prover às aplicações uma infraestrutura, e mecanismos mais dinâmicos e flexíveis, como abordagens baseadas em agentes, são necessários.

3.2.6 Segurança

O uso e o compartilhamento de serviços e recursos distribuídos somente serão viáveis se mecanismos de segurança forem oferecidos aos usuários. A figura 3.7 mostra o *SIG* da segurança. Ele divide a segurança em três dimensões: código de terceiros, estratégia de reputação e garantia à privacidade.

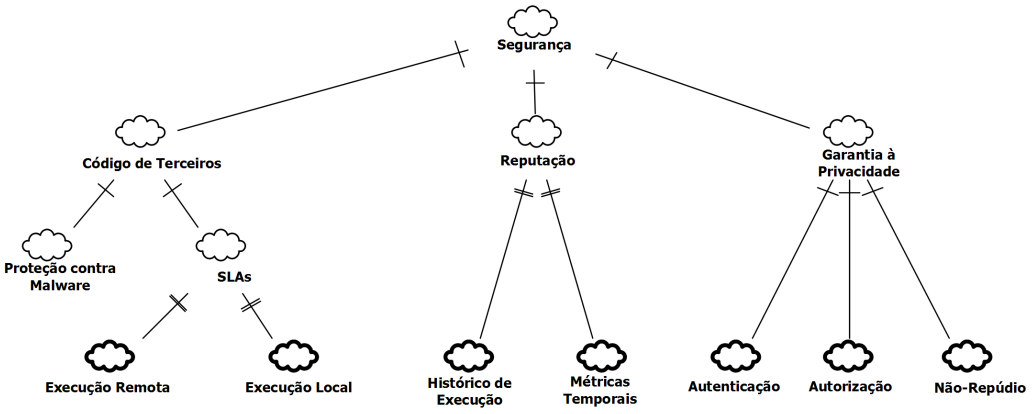


Figura 3.7: SIG da segurança

A primeira dimensão, chamada código de terceiros, lida com aspectos relativos à execução de códigos e serviços de origem desconhecida. Ela trata primeiro da proteção contra código malicioso, por exemplo, vírus de computadores. Já *service level agreement (SLA)* é uma obrigação contratual acertada entre os dispositivos, e define as possibilidades e expectativas na execução de um serviço [Yan et al. 2007]. SLAs especificam os requisitos funcionais e não funcionais oferecidos por esse serviço, garantindo a sua execução de acordo com o que está na descrição, e descrevem níveis de *QoS* que podem ser oferecidos pelos serviços. SLAs podem ser utilizados para garantir execução de códigos e serviços remotos, de forma correta e eficiente, e permitir a integridade dos recursos locais quando hospedando código remoto.

A segunda dimensão define os mecanismos de reputação. Dispositivos com níveis de reputação mais elevados são supostamente mais confiáveis e devem ser priorizados. Esses níveis de reputação são calculados baseados em interações passadas, e mecanismos de *spot checking* [Martins et al. 2006] podem ser utilizados para minimizar manipulação de execução remota.

Geralmente, os mecanismos de reputação são implementados utilizando a manutenção dos identificadores; entretanto, dispositivos e usuários reconhecidamente pouco confiáveis podem utilizar um novo identificador, sempre que quiserem burlar o mecanismo de reputação. Para solucionar esse problema, métricas temporais podem ser utilizadas para valorizar dispositivos com identificadores mais antigos.

A terceira dimensão é a garantia à Privacidade, a qual busca restringir o acesso às informações apenas a dispositivos confiáveis, e pode ser derivada em três requisitos: autorização, autenticação e não-repúdio. Dessa forma, a autorização e a autenticação permitem que apenas dispositivos com acesso às informações podem utilizar o sistema. A autenticação pode ainda ser utilizada para implementar o não-repúdio, onde toda ação dentro do sistema é assinada inequivocamente pelo dispositivo que a realizou.

Esses mecanismos são implementados utilizando um par de chaves público/privado disponibilizados ou por uma entidade certificadora, ou por um algoritmo distribuído executado pelos próprios dispositivos [Merwe et al. 2007]. Aplicações que utilizam uma entidade certificadora já possuem um entendimento preestabelecido, e normalmente estão vinculadas a um único domínio administrativo. Por outro lado, a distribuição desse par de chaves de forma descentralizada pelos próprios dispositivos é mais flexível, entretanto, mais complexo de implementar e insere um maior na rede.

Dependências internas da segurança

SLAs são contratos entre duas entidades que mutuamente se respeitam. Dessa forma, o uso de reputação é uma técnica valiosa para garantir que esses contratos são respeitados, onde interações passadas e métricas temporais ajudam a decidir se uma entidade é confiável ou não. Adicionalmente, SLAs trocam informações privadas sobre os dispositivos e mecanismos de privacidade devem ser utilizados.

3.3 Requisitos transversais

A correta implementação dos serviços essenciais para atender às necessidades das aplicações móveis é fortemente influenciada por três requisitos transversais: sensibilidade ao contexto e adaptação, autonomia e provisão de QoS. Esses requisitos são considerados transversais por estarem presentes em todos os serviços essenciais e influenciarem de forma decisiva na correta execução desses serviços.

3.3.1 Sensibilidade ao contexto e adaptação

Sensibilidade ao contexto é a habilidade de prover informação relevante sobre o ambiente para as aplicações e serviços [Coutaz et al. 2005]. Para isso, aplicações móveis precisam capturar, inferir e descrever informações de contexto em um formato que dispositivos computacionais sejam capazes de processar. Nesse momento, informações capturadas do contexto disparam um mecanismo de adaptação que lida com mudanças no ambiente e otimiza o comportamento da aplicação.

Adaptação é um importante requisito que melhora o reúso, a portabilidade e a confiabilidade de sistemas de *software*, e suporta redefinição de requisitos, manutenção, evolução e mudanças no ambiente de execução [Oreizy et al. 1998, Oreizy et al. 1999]. Esses dois mecanismos são considerados requisitos transversais porque impactam diretamente no comportamento de cada serviço essencial.

Por exemplo, a correta escolha dos mecanismos de troca de mensagens para implementar garantia de entrega, ordenação e tipo de entrega deve ser feita baseada nos requisitos da aplicação e condições do ambiente. Dessa forma, sensibilidade ao contexto e adaptação podem ser utilizados para adaptar a garantia de entrega de *at least once* para *at most once*, se a confiabilidade na rede é alta, relaxar restrições de ordenação, se a latência estiver alta, ou utilizar um método síncrono, se a disponibilidade for alta.

Assim que o mecanismo de suporte à mobilidade percebe que o nível de segurança

em uma nova rede é maior do que a rede atual, ele dispara a mobilidade de dispositivo e passa a se comunicar utilizando essa nova rede. A seguir, adapta o mecanismo de privacidade, desabilitando a criptografia em nível de aplicação para economizar bateria.

A coordenação de serviços pode utilizar a adaptação para implementar composição de serviços dinâmica e atender redefinição de requisitos em tempo de execução, mudança de preferências ou manutenção e evolução do sistema.

3.3.2 Autonomia

Sistemas autônomos utilizam a sensibilidade ao contexto para perceber e descrever o ambiente, e mecanismos de adaptação para configurar o comportamento do sistema e atender objetivos globais. As técnicas de autonomia possuem o conhecimento de como as mudanças percebidas no ambiente impactam no comportamento do sistema, e quais ações devem ser tomadas para minimizar o impacto dessas mudanças.

Esse conhecimento está constantemente sendo atualizado pelo resultado positivo ou negativo das mudanças realizadas. Essa atualização é guiada pelos requisitos de *QoS* descritos pelas aplicações. Eles são as políticas globais que os sistemas autônomos buscam atingir.

3.3.3 Provisão de *QoS*

Um mecanismo de descrição de *QoS* é útil para as aplicações descreverem seus requisitos de *QoS* e para os mecanismos de autonomia descreverem para as aplicações os níveis de *QoS* que podem ser oferecidos em um determinado contexto.

Existem medidas de adaptação que podem ser realizadas pelo mecanismo de autonomia de forma transparente às aplicações, como há situações nas quais são as políticas de adaptação específicas de cada aplicação e não cabe ao mecanismo de autonomia decidir quais são as políticas de adaptação. Sendo assim, o mecanismo

de autonomia descreve às aplicações os níveis de *QoS* em um determinado contexto, e deixa a cargo das aplicações realizarem as medidas de adaptação necessárias.

3.4 Dependências externas

Determinado serviço essencial não é influenciado apenas pelos requisitos de *QoS* das aplicações, mas também pelas diferentes implementações dos outros serviços essenciais em um determinado momento. A figura 3.8 mostra como cada requisito é influenciado por outros requisitos primários e transversais.

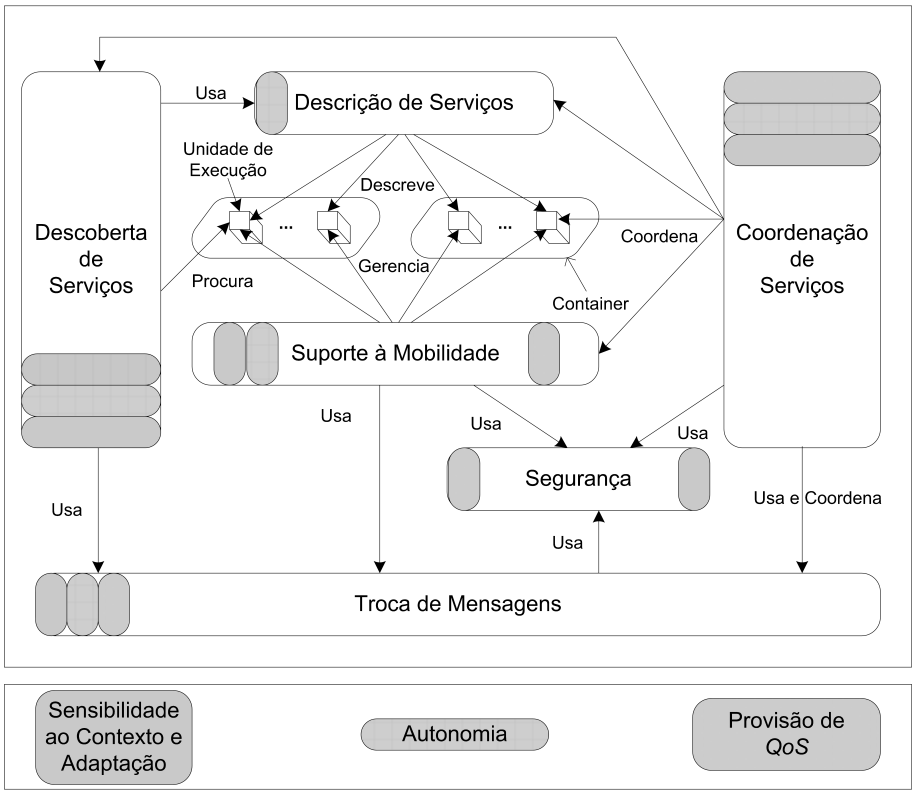


Figura 3.8: Arquitetura dos requisitos primários e dependências externas

3.4.1 Troca de mensagens e descoberta de serviços

A eficiência do mecanismo de descoberta de serviços é diretamente influenciada pelas diferentes implementações da troca de mensagens utilizada. Mecanismos de

descoberta centralizados podem utilizar troca de mensagens síncrona baseada em *RPC*, ou assíncrona baseada em eventos ou tuplas. Adicionalmente, descoberta de serviços descentralizadas devem utilizar troca de mensagens assíncrona. Se for totalmente descentralizada, deve utilizar *flooding* ou *gossiping*. Por último, caso seja híbrida, mecanismos baseados em eventos e tuplas são mais adequados.

3.4.2 Troca de mensagens e coordenação de serviços

Mecanismos de troca de mensagens baseados em espaço de tuplas distribuídos são de particular importância para a coordenação dos agentes distribuídos [Mamei e Zambonelli 2005], e úteis para implementar provisão de serviços e adaptação baseados na localização do dispositivo.

Por outro lado, a coordenação baseada em infraestrutura permite a troca de mensagens baseada em eventos de forma independente da localização do dispositivo, onde a própria infraestrutura de comunicação fica responsável por gerenciar a mobilidade e as desconexões temporárias.

Orquestração e coreografia, normalmente, seguem uma regra bem rígida de troca de mensagens e requerem um nível considerável de garantia de entrega. Com isso, *best effort* não deve ser utilizado para implementar composição de serviço. Seguindo a mesma linha, *total order* ou *causal order* são adequadas para serem utilizadas na composição de serviços. Relaxamento de ordenação, entretanto, pode ser tratado pela própria infraestrutura de coordenação.

3.4.3 Troca de mensagens e segurança

A troca de mensagens possui restrições importantes de segurança já que está suscetível ao acesso às informações por pessoas não autorizadas, ou ainda à manipulação do conteúdo das mensagens. Por isso, é necessário um mecanismo robusto de autorização e validação do conteúdo implementado em conjunto com a troca de mensagens.

A centralização das informações melhora a garantia à privacidade, e qualquer forma de descentralização dificulta o controle de acesso às informações por indivíduos não autorizados. Mecanismos de troca de mensagens totalmente distribuídos, como *Flooding* e *Gossiping*, tornam essa tarefa ainda mais complicada.

3.4.4 Suporte à mobilidade e coordenação de serviços

O suporte à mobilidade, no que diz respeito aos mecanismos de migração de código, estão relacionados com a coordenação de serviços, pois precisam verificar se o ambiente de destino é capaz de prover os mesmos recursos que o ambiente de origem, como acessibilidade aos serviços, infra-estrutura de coordenação e políticas de incentivo à cooperação. Adicionalmente, a mobilidade de dispositivo precisa garantir que o protocolo de troca de mensagens utilizado é adequado à arquitetura da rede de destino, possivelmente adaptando ou modificando o protocolo utilizado.

3.4.5 Suporte à mobilidade e segurança

A mobilidade de código também possui restrições de segurança, já deve ser implementada utilizando a negociação e verificação autônoma de SLAs. Além disso, as aplicações requerem garantias referentes à corretude na execução remota de código e serviços.

A mobilidade de dispositivo pode fazer com que informações privadas passem a ser transmitidas utilizando uma rede com níveis de segurança abaixo do necessário às aplicações. Com isso, mecanismos autônomos e adaptáveis precisam identificar variações no nível de segurança, possivelmente inserindo uma camada de segurança em nível de aplicação .

3.4.6 Descrição e descoberta de serviços

A descrição semântica de serviços é mais adequada em protocolos de descoberta de serviços centralizados e híbridos. A falta de padronização de ontologias introduz uma dificuldade extra em utilizar descrição semântica em protocolos de descoberta totalmente descentralizados.

3.4.7 Descrição e coordenação de serviços

A descrição semântica de serviços permite a criação de aplicações móveis autônomas, onde informações de contexto de alto nível podem ser inferidas a partir de dados recebidos de sensores. Por exemplo, os sensores em um ambiente podem informar que as luzes em uma sala de aula repleta de alunos estão apagadas e o projetor está ligado. Nesse caso, o contexto de alto nível inferido é que está acontecendo uma apresentação.

Informações de contexto de alto nível podem ser utilizadas para coordenar a execução de uma tarefa, além de permitir adaptação para atender objetivos globais. Além disso, esse tipo de informação semântica pode ser utilizada também para descrever recursos disponíveis, restrições de *QoS* e níveis de reputação.

3.4.8 Coordenação de serviços e segurança

A execução correta dos SLAs é influenciada por informações de cooperação fornecidas às aplicações. Assim, dispositivos maliciosos com um alto nível de reputação podem comprometer toda a execução e coordenação distribuída de uma tarefa.

3.5 Aplicabilidade dos serviços essenciais às diferentes tecnologias de redes móveis

As aplicações devem descrever os níveis de *QoS* necessários, e cabe ao mecanismo de autonomia perceber o estado do ambiente e utilizar as implementações mais adequadas ao contexto percebido e aos requisitos de *QoS* das aplicações.

Cada implementação diferente dos serviços essenciais é altamente influenciada pela arquitetura das camadas de rede e inferiores, que possuem características estáticas como área de cobertura ou velocidade de conexão nominal, além de informações dinâmicas como número de dispositivos e padrão de mobilidade. Outra informação relevante é a topologia da rede, que pode ser estática em redes estruturadas como as celulares e 802.11, ou dinâmica em redes desestruturadas como as redes *ad hoc*.

3.5.1 Redes celulares

As redes celulares têm a maior área de cobertura entre as redes sem fio móveis, apresentam uma velocidade de conexão nominal de alguns *megabits* por segundo, e possuem uma infraestrutura estática que dá suporte à mobilidade de dispositivo [Andrade e Lograppo 2006]. Variações de carga nas redes celulares têm pouco impacto nos níveis de *QoS* oferecidos. O problema em utilizar as redes celulares é o custo cobrado para transmissão de dados.

Dessa forma, os protocolos de troca de mensagens utilizados nas redes celulares devem minimizar o número de mensagens sendo trocadas. Modelos síncronos, ou assíncronos baseados em tuplas e eventos, podem ser utilizados. Modelos baseados em *flooding* ou *gossiping* devem ser evitados, pois trocam muitas mensagens.

A descrição de serviços pode ser tanto sintática quanto semântica. Descrição sintática é mais leve e deve ser utilizada quando os dispositivos interagindo possuírem restrições de processamento ou estiverem sobrecarregados. Já a descrição semântica pode ser utilizada quando existir uma ontologia comum e os dispositivos possuírem

uma capacidade razoável de processamento ou existir um servidor dedicado para realizar as inferências sobre as ontologias [Liu et al. 2006].

A descoberta de serviços deve ser implementada utilizando uma abordagem centralizada ou híbrida, visto que um servidor dedicado pode ser acessado mais facilmente. Já a descoberta totalmente distribuída deve ser evitada. Quando o servidor centralizado estiver indisponível, a abordagem híbrida pode ser utilizada.

A coordenação de serviços pode ser baseada em infraestrutura, e as políticas de incentivo à cooperação baseadas em créditos, já que um servidor dedicado pode ficar responsável por gerenciar a composição de serviços e a troca de créditos. Abordagens baseadas em reputação trocam muitas mensagens e devem ser evitadas.

Serviços de autenticação e autorização podem ser utilizados nas redes celulares, onde um servidor confiável é responsável por distribuir as chaves aos dispositivos, assim como os SLAs podem ser negociados e gerenciados também de forma centralizada.

3.5.2 Redes locais sem fio

Redes locais sem fio baseadas em interfaces 802.11b/g apresentam um raio de cobertura em ambientes fechados de até cinquenta metros, a melhor taxa de conexão dentre as redes sem fio móveis. Por outro lado, essas redes não permitem o mesmo nível de mobilidade que as redes celulares.

Como as redes locais sem fio são menos resilientes à mobilidade do que as redes celulares, mecanismos de troca de mensagens devem ser baseados em eventos ou tuplas, pois apresentam menor acoplamento que *RPC* e geralmente são mais confiáveis do que *gossiping* e *flooding*. Além de diminuir o acoplamento, essas duas abordagens possibilitam a adaptação e provisão de serviços baseadas na localização.

Por outro lado, a largura de banda nas redes locais sem fio é maior do que nas outras redes sem fio, o que diminui as restrições de sobrecarga na rede. Por isso, serviços essenciais utilizando as redes locais podem priorizar a eficiência na execução

do serviço, no lugar do custo inserido na rede.

3.5.3 Redes *ad hoc* móveis

As redes *ad hoc* móveis são redes formadas dinamicamente, sem nenhuma infra-estrutura fixa, onde os próprios dispositivos encaminham pacotes endereçados a outros dispositivos de forma multisalto [Imrich Chlamtac e Liu 2003].

Essa falta de uma infraestrutura fixa causa uma relação de dependência entre o desempenho dos protocolos de aplicação e as características momentâneas da rede. Algumas dessas características são o número e a densidade dos dispositivos, recursos de *hardware* disponíveis, como processamento, armazenamento e bateria, congestionamento da rede, além do padrão de mobilidade.

Essa dependência acontece por dois motivos principais: imprevisibilidade no canal de comunicação e sobrecarga nos dispositivos. A primeira ocorre em virtude da mobilidade dos dispositivos, que muda constantemente a topologia da rede, e os protocolos devem constantemente se adaptar a uma nova topologia. Adicionalmente, como esses dispositivos já estão sobrecarregados com funções de roteamento, os protocolos precisam ser leves e descentralizados.

Protocolos de troca de mensagens a serem utilizados devem ser descentralizados, e *flooding* e *gossiping* são uma alternativa natural; mecanismos baseados em eventos e tuplas só quando o nível de mobilidade for baixo.

A descrição de serviços deve ser leve e a descrição sintática é preferível. O uso de uma descrição semântica é ainda mais difícil nas redes *ad hoc*, já que não existe uma ontologia comum aos dispositivos.

O uso de mecanismos de descoberta totalmente distribuídos é encorajado. Essa abordagem é bem adaptada à natureza dinâmica das redes *ad hoc* móveis. Se o nível de mobilidade diminuir, protocolos híbridos projetados especificamente para essas redes podem ser utilizados.

A coordenação deve ser baseada em agentes, já que não existe uma infra-estrutura

para coordenar as atividades. Adicionalmente, políticas de incentivo à cooperação devem ser baseadas em reputação, já que são mecanismos descentralizados e gerenciados pelos próprios dispositivos.

Em redes *ad hoc* móveis é muito difícil garantir privacidade das informações, já que os próprios dispositivos são responsáveis por encaminhar e rotear os pacotes. Para isso, ou os dispositivos pertencem a um mesmo domínio administrativo e as chaves são implantadas em tempo de desenvolvimento, ou os próprios dispositivos são responsáveis por atribuir essas chaves de forma distribuída. A primeira abordagem restringe o acesso aos dispositivos, e a segunda minimiza as garantias de segurança.

SLAs são baseados em informações de reputação informadas pelos vizinhos, e muito suscetíveis à manipulação dessas informações.

3.6 Conclusões

Este capítulo detalhou os serviços essenciais para construção de aplicações móveis. Esses serviços são definidos como funcionalidades essenciais às aplicações e classificados como Troca de Mensagens, Suporte à Mobilidade, Descrição, Descoberta e Coordenação de Serviços, e Segurança. Cada um deles foi decomposto em requisitos implementacionais, o que permitiu identificar as dependências internas e externas entre os diferentes serviços essenciais.

Esse capítulo detalhou ainda os requisitos transversais, que são sensibilidade ao contexto e adaptação, autonomia e restrições de QoS. Cada um dos requisitos transversais influencia de maneira decisiva no funcionamento eficiente dos serviços essenciais.

Os requisitos transversais influenciam no comportamento dos serviços essenciais de acordo com restrições de QoS e políticas de adaptação das aplicações, além das características momentâneas da rede sem fio. Dessa forma, o capítulo identificou

como as diferentes arquiteturas de redes móveis influenciam na eficiência dos serviços essenciais.

O levantamento e detalhamento desses serviços, o relacionamento entre eles, e como a implementação desses requisitos pode ser mapeada para as diferentes arquiteturas de redes sem fio são as primeiras contribuições desta dissertação.

O capítulo seguinte vai detalhar o AESPmob, um modelo autônomo e evolutivo para melhorar a eficiência e adaptabilidade dos serviços essenciais às restrições de *QoS* das aplicações e às diferentes arquiteturas de redes sem fio.

Capítulo 4

AESPmob

Este capítulo apresenta o *AESPmob*, um modelo autônomo e evolutivo para provisão de serviços essenciais em redes móveis. A seção 4.1 faz uma introdução ao *AESPmob* e a seção 4.2 mostra a arquitetura do *AESPmob*, detalha os diferentes módulos que o formam. Já na seção 4.3 estão as atividades realizadas pelo *AESPmob*, que estão divididas em invocação dos serviços essenciais e difusão das estratégias. Por fim, a seção 4.4 mostra um modelo que descreve como o tempo de difusão das estratégias varia com o número de dispositivos, e como o custo e a eficiência dos serviços essenciais variam com a difusão das estratégias.

4.1 Introdução

Esta dissertação apresenta o *AESPmob*, um modelo autônomo e evolutivo para aumentar a eficiência e disponibilidade dos serviços essenciais em redes móveis. O *AESPmob*, quando implantado nos dispositivos móveis, constantemente analisa a eficiência desses serviços, enquanto simultaneamente procura por diferentes implementações (estratégias) mais eficientes.

As diferentes formas de implementar cada serviço essencial detalhado no capítulo 3 representam as estratégias monitoradas em tempo de execução pelo *AESPmob*. Com isso, sempre que uma nova estratégia de um serviço essencial é encontrada

em um vizinho, o nível de eficiência dessa estratégia remota é comparado com o nível de eficiência das estratégias locais equivalentes. Quando a estratégia remota for mais eficiente, ela é instalada localmente. Na próxima vez que uma aplicação requisitar um serviço essencial, o *AESPmob* garante que será utilizada a estratégia mais eficiente.

É importante ressaltar que a definição de eficiência (métrica e nível de *QoS*) de um determinado serviço está intrinsecamente relacionada à aplicação utilizando esse serviço. Para isso, essa aplicação precisa notificar ao *AESPmob* a métrica de *QoS* que deve ser empregada para quantificar a eficiência de um serviço, além de definir níveis mínimos de *QoS* que o *AESPmob* deve buscar atender. Esse par formado pela métrica de *QoS* usada e o nível dessa métrica é definido como o descritor de *QoS*. Por exemplo, um mecanismo de descoberta de serviço pode ser avaliado segundo o número de respostas positivas a uma requisição. Dessa forma, a métrica de *QoS* é o número de respostas positivas, e o nível de *QoS* é valor para essa métrica requerido por uma determinada aplicação. Outras formas de avaliar um mecanismo de descoberta de serviço podem ser o custo inserido na rede por esse mecanismo ou a qualidade dos serviços descobertos.

No *AESPmob*, o monitoramento e a substituição das estratégias é feito de forma totalmente transparente às aplicações. Quando determinada aplicação precisa invocar um serviço essencial, ela acessa uma interface fornecida pelo *AESPmob* para esse serviço específico. A implementação que será utilizada desse serviço, entretanto, é definida pelo *AESPmob*, baseada nos parâmetros de *QoS* descritos pela aplicação, além da medida de eficiência das estratégias encontradas no ambiente.

4.2 Arquitetura do *AESPmob*

O *AESPmob* é implementado como um sistema de *middleware* e utilizado pelas aplicações para invocar cada um dos serviços essenciais descritos no capítulo 3. A

figura 4.1 mostra a arquitetura do *AESPmob* proposta nesta dissertação.

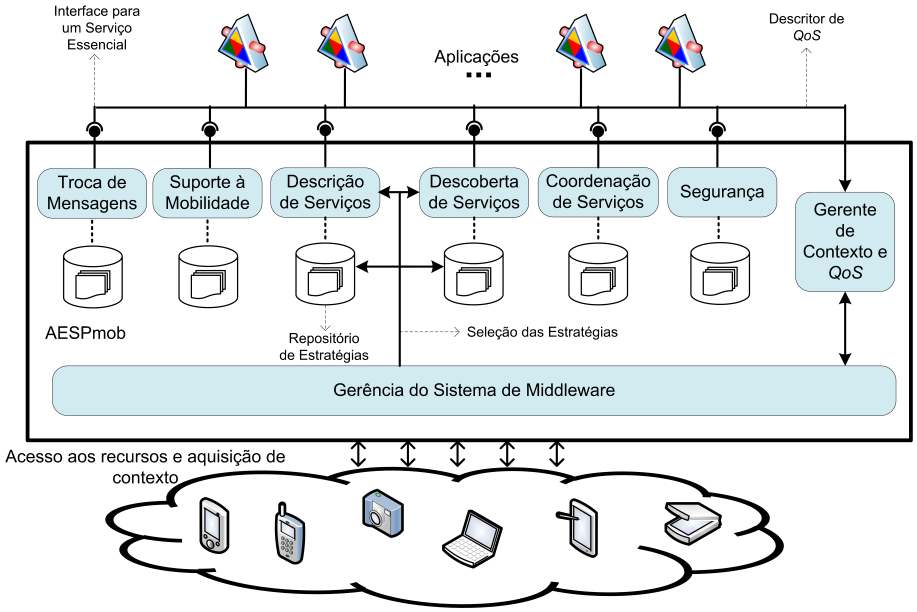


Figura 4.1: Arquitetura do *AESPmob*

Ainda na figura 4.1, três estruturas podem ser destacadas: 1) as interfaces de invocação dos serviços essenciais, detalhadas na seção 4.2.1; 2) o módulo responsável pela gerência de contexto e *QoS*, que está na seção 4.2.2; e 3) o módulo de gerência do middleware, especificado na seção 4.2.3.

4.2.1 Interfaces de invocação

Na camada superior, as aplicações invocam os serviços essenciais utilizando uma interface específica para cada serviço. A escolha da estratégia que será utilizada quando uma interface é invocada é feita pelo *AESPmob* utilizando os requisitos de *QoS* informados pela aplicação e informações de contexto coletadas pelo *AESPmob*.

Isso permite que os desenvolvedores se concentem nos requisitos funcionais das aplicações, já que a implementação dos serviços essenciais, além da adaptação e evolução desses serviços, fica sob responsabilidade do *AESPmob*.

Essa separação entre interface de invoção de um serviço essencial e execução da estratégia escolhida só é possível se as diferentes estratégias que implementam

determinado serviço essencial forem acessadas de forma semelhante. Por exemplo, dentre as estratégias de descoberta de serviços pesquisadas [Engelstad et al. 2006], independentemente de serem totalmente centralizadas, distribuídas ou híbridas, elas se assemelham pela mensagem que é trocada. A Tabela 4.1 mostra alguns campos presentes nessa mensagem e uma breve descrição de cada um deles.

Tabela 4.1: Campos em uma mensagem de descoberta de serviço

Informação	**Descrição**
ID Dispositivo	Dispositivo que está iniciando a busca ou publicando um serviço.
Informações do Dispositivo	Características e recursos disponíveis do dispositivo.
Descrição do Serviço	Informações sobre o serviço oferecido (modelo *Pull*) ou parâmetros da consulta (modelo *Push*).
TTL (*Time to Live*)	Número de vezes que uma mensagem pode ser encaminhada
Validade	Tempo que a mensagem pode ser armazenada em *cache*

Dentre todos esses campos, a descrição do serviço é o mais difícil de ser padronizado. Isso acontece porque a forma como um serviço é descrito está muito acoplado ao tipo de aplicação e aos dispositivos que o acessam. Dessa forma, no lugar de padronizar essa descrição para garantir a interoperabilidade, é preferível que esse campo seja abstrato para aumentar a flexibilidade.

Por outro lado, os campos restantes podem ser padronizados. Assim, se as diferentes estratégias de descoberta enviarem para a rede um mesmo tipo de mensagem de busca, cada uma pode utilizar uma abordagem local para difusão das mensagens e armazenamento das informações, e ainda assim participar do processo de descoberta em conjunto com outras estratégias.

Dessa forma, o papel do *AESPmob* é definir, em tempo de execução e de forma autônoma, qual estratégia deve ser utilizada quando um serviço essencial é invocado.

4.2.2 Gerente de contexto e *QoS*

O Gerente de contexto e *QoS* é responsável por gerenciar os requisitos de *QoS* informados pelas aplicações, além de constantemente monitorar o contexto ao qual as aplicações estão sujeitas e também as diferentes estratégias para os serviços essenciais.

Dessa forma, quando um serviço essencial é invocado por uma aplicação, ela informa também seu descritor de *QoS*. Esse descritor será armazenado pelo gerente de contexto e *QoS* e utilizado sempre que essa aplicação invocar um serviço essencial, para definir qual estratégia deve ser utilizada, além de servir para comparar o nível de eficiência das diferentes estratégias que implementam um serviço essencial.

O gerente de contexto e *QoS* armazena informações sobre a eficiência das estratégias em diferentes contextos. Sempre que uma mudança de contexto é detectada, ele descreve para a gerência do sistema de middleware o novo contexto percebido.

Para o *AESPmob*, os contextos relevantes são as informações sobre as diferentes tecnologias de redes móveis descritas na seção 3.5, além dos recursos de hardware e software disponíveis.

4.2.3 Gerência do sistema de middleware

O módulo de gerência de *middleware* é responsável por duas atividades principais: 1) garantir o uso das estratégias mais eficientes; e 2) lidar com a invocação dos serviços essenciais.

A primeira atividade, responsável por garantir que as estratégias mais eficientes são utilizadas, consiste no monitoramento das estratégias que estão sendo utilizadas, detecção de novas estratégias e difusão das estratégias eficientes.

Já a segunda atividade, que lida com o uso dos serviços essenciais, recebe a invocação desses serviços feita pelas aplicações, utiliza informações recebidas pelo módulo de contexto e *QoS* para definir qual a estratégia mais adequada e executa essa estratégia.

Os conceitos de autonomia e evolução, citados no capítulo 2, também são implementados no módulo de gerência. Esse módulo monitora, atualiza e difunde as estratégias de forma autônoma, sem que aconteça intervenção do usuário, e transparente às aplicações, que apenas invocam uma interface para os serviços essenciais, sem conhecer qual estratégia é de fato utilizada.

A evolução dos serviços essenciais acontece através da difusão das estratégias mais eficientes. Isso permite que estratégias mais adaptadas sejam sempre utilizadas, além de minizar a presença daquelas menos eficientes.

4.3 Funcionamento do *AESPmob*

O funcionamento do *AESPmob* está dividido em duas atividades principais: 1) invocação dos serviços essenciais e difusão das estratégias. A atividade relacionada à invocação dos serviços essencias está detalhada na seção 4.3.1 e a difusão das estratégias é analisada na seção 4.3.2.

4.3.1 Invocação dos serviços essenciais

Para invocar um serviço essencial, a aplicação informa o seu identificador ao *AESPmob*, a descrição do serviço a ser executado, os parâmetros que esse serviço requer, além do descritor de *QoS* requerido pela aplicação.

Ao receber uma invocação para determinado serviço essencial, o *AESPmob* executa uma série de ações descritas pelo algoritmo 1.

Algoritmo 1 Invocando serviço essencial

```
1: procedure Invocando_Servico(id_aplicacao, descricao_servico, parametros_servico, descritor_qos)
2:     resultado ← subscreve_invocacao(id_aplicacao, descritor_qos);
3:     estrategia ← define_estrategia(descricao_servico, descritor_qos, contexto);
4:     resultado ← executa_estrategia(parametros_servico, descritor_qos);
           return resultado;
5: end procedure
```

Essa invocação não bloqueia a execução da aplicação. O **resultado** é acessado posteriormente, utilizando um objeto retornado pelo *AESPmob* para a aplicação

(algoritmo 1, linha 2).

Por outro lado, uma invocação a uma interface dispara um evento ao módulo de gerência, que define a estratégia mais adequada naquele momento, utilizando o descritor de *QoS* fornecido pelas aplicações e o contexto percebido (algoritmo 1, linha 3). Nesse momento, o módulo de gerência acessa o repositório de estratégias e executa a estratégia escolhida (algoritmo 1, linha 4).

O emprego do descritor de *QoS* é útil quando a execução de uma determinada estratégia acontece de forma distribuída, as mensagens que são trocadas levam o descritor de *QoS* fornecido pelas aplicações. Isso permite que cada instância distribuída do *AESPmob* utilize a estratégia local mais eficiente de acordo com esse descritor, e coopere para entregar o nível de *QoS* requerido pelas aplicações.

Com isso, quando uma instância distribuída do *AESPmob* recebe uma mensagem remota, ela dispara um evento para o módulo de gerência. Esse módulo define o serviço essencial que essa mensagem implementa e o descritor de *QoS* permite escolher a estratégia mais eficiente para lidar com essa mensagem.

A seguir, a mensagem recebida é encaminhada para essa estratégia, que define a ação correta a ser realizada. Algumas ações possíveis podem ser responder ao dispositivo que enviou a mensagem, difundir essa mensagem para outros dispositivos vizinhos, ou notificar uma determinada aplicação de que existe uma resposta a uma invocação anterior. Essa aplicação é notificada por meio do objeto **resultado** recebido pela execução do algoritmo 1, na linha 4.

4.3.2 Difusão das Estratégias

Simultaneamente ao mecanismo de invocação dos serviços, o módulo de gerência do *AESPmob*, periodicamente, contacta um subconjunto dos dispositivos vizinhos para definir qual a estratégia mais eficiente para cada serviço essencial, dados o contexto percebido e os requisitos de *QoS* das aplicações.

O módulo de gerência informa a alguns de seus vizinhos qual a estratégia local

mais eficiente para cada serviço essencial e Descritor de *QoS* existente, e recebe desses vizinhos a mesma informação. Quando uma nova estratégia mais eficiente é encontrada, o *AESPmob* faz um teste de compatibilidade para verificar se essa estratégia pode ser utilizada localmente.

O teste de compatibilidade deve considerar se a linguagem de programação utilizada para implementar a estratégia remota é compatível com o dispositivo local, e se existem recursos locais disponíveis para executar essa estratégia.

O número de vizinhos escolhidos para comparar as estratégias é chamado *fator de difusão*. Esse fator tem impacto significativo no tempo de difusão das estratégias. Um fator de difusão menor significa que as estratégias serão difundidas mais lentamente, porém introduzindo um custo menor na rede. Por outro lado, um fator de difusão maior introduz um custo maior, mas diminui o tempo de difusão das estratégias.

Dessa forma, é preciso entender como a variação do fator de difusão impacta na difusão das estratégias e no custo introduzido na rede. Isso vai ajudar a definir valores adequados para o fator de difusão, dadas as características momentâneas das redes sem fio.

4.4 Modelo analítico para difusão das estratégias

Esta dissertação apresenta um modelo analítico que descreve como o tempo de difusão das estratégias varia à medida que o número de dispositivos aumenta, e como o fator de difusão impacta nesse tempo de difusão e no custo inserido na rede.

Uma rápida difusão das estratégias é importante devido a alta dinamicidade e a incerteza das redes móveis, já que elas requerem uma rápida adaptação dos serviços essenciais às condições dessas redes.

O modelo analítico apresentado nesta dissertação foi baseado em um artigo publicado por [Groenevelt et al. 2005], que analisa o tempo médio necessário para

enviar uma mensagem a um determinado dispositivo. Nesta dissertação o objetivo do modelo é encontrar o tempo que leva para difundir uma estratégia a todos os dispositivos. Outra diferença é que o modelo de Groenevelt considera um cenário onde os dispositivos não deixam a rede até que a mensagem seja entregue, enquanto nesta dissertação também é analisado um cenário onde dispositivos entram e saem da rede segundo o modelo de nascimento e morte [Bhat e Miller 2002].

O modelo proposto nesta dissertação analisa ainda o custo e o benefício em utilizar cada estratégia, e fornece um mecanismo para auxiliar o *AESPmob* a definir qual estratégia deve ser difundida, seja a com menor custo ou a mais eficiente.

4.4.1 Suposições

Assim como no artigo publicado por Gronevelt, esta dissertação faz algumas suposições que são utilizadas para desenvolver o modelo analítico.

Considerando que existem n dispositivos em uma rede totalmente desestruturada se movendo de forma independente de acordo com o modelo de mobilidade *random way point* [Tracy Camp 2002] e com um raio de transmissão limitado, cada dispositivo utiliza em um dado momento uma estratégia $s_i, i \in E$, onde E representa o conjunto de todas as estratégias para um dado serviço essencial.

Inicialmente, apenas um dispositivo possui a estratégia reconhecidamente mais eficiente, chamada s_e.

Sempre que dois dispositivos se encontram, i.e., cada um está dentro do raio de cobertura do outro, e apenas um deles possui a estratégia s_e, ela é difundida para o outro dispositivo. Essa abordagem de contato e infecção é conhecida como difusão epidêmica [Bailey 1975].

Sejam $0 \leq t_{i,j}(1) < t_{i,j}(2) < ...$ os tempos de encontros sucessivos entre os dispositivos i e j. Mais ainda, se $\tau_{i,j}(k) := t_{i,j}(k+1) - t_{i,j}(k)$ representa o tempo entre os encontros $k+1$ e k. As seguintes suposições são consideradas válidas:

1) as transmissões acontecem apenas durante os tempos de encontro $t_{i,j}(k)$ e

são instantâneas. Isso acontece quando o tempo de transmissão de uma estratégia é descartável se comparado com o tempo entre encontros $\tau_{i,j}(k)$. A simulação realizada no capítulo 5 mostra que essa suposição é válida.

2) os processos $\{t_{i,j}(k), k > 1\}, 1 < i, j < n, i \neq j$ são processos de Poisson mutuamente independentes que seguem uma distribuição exponencial com média $1/\lambda_{const}$. Groenevelt mostra, utilizando funções de autocorrelação [Bailey 1975], que essa suposição também é válida. A4.2 taxa λ_{const} é função do raio de transmissão e da velocidade média dos dispositivos, além da área de mobilidade dos dispostivos [Groenevelt et al. 2005].

4.4.2 Tempo de difusão das estratégias

Considerando que existem n dispositivos na rede, e apenas um com a estratégia mais eficiente s_e, esse modelo vai analisar o tempo de difusão dessa estratégia a todos os dispositivos.

A difusão das estratégias é modelada utilizando uma cadeia de Markov com n estados, onde cada estado $i, 1 \leq i \leq n$ representa o momento no qual i dispositivos estão utilizando a estratégia s_e que está sendo difundida. Assim, o modelo inicia no estado 1, pois inicialmente só existe um dispositivo com a estratégia s_e. O modelo passa por todos os estados até atingir o estado permanente n.

A figura 4.2 mostra a cadeia de Markov representando todos os estados possíveis que o sistema pode assumir. Ainda na figura 4.2, quando o sistema atinge o estado i (i dispositivos utilizando a estratégia s_e), ele nunca retorna ao estado $i-1$, pois não existe uma estratégia mais eficiente que s_e, e os dispositivos não saem da rede. Posteriormente será analisado o tempo de difusão quando os dispositivos entram e saem da rede.

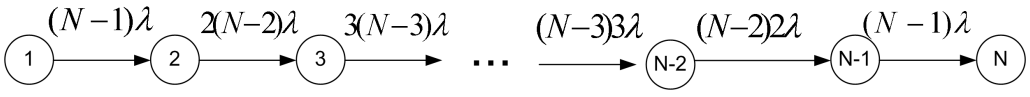


Figura 4.2: Difusão das estratégias sem a saída dos dispositivos

Na cadeia de Markov da figura 4.2, o sistema sai do estado i para o estado $i+1$ de acordo com a taxa da equação (I). Nela, existem i dispositivos com a estratégia s_e, $(n-i)$ maneiras do sistema evoluir para o estado $i+1$, e dois dispositivos entram em contato a uma taxa λ. Ainda na equação (I), o fator de difusão v diminui o número de vizinhos escolhidos para comparar as estratégias.

$$\lambda(i) = i(n-i)v\lambda_{const} \text{ (I)}$$

Adicionalmente, o tempo total que o sistema passa no estado i até deixar o estado permanentemente é representado por S_i. Com isso, T_S é o somatório de todos os tempos que o sistema passa em cada estado, ou seja, é o tempo para difundir uma estratégia.

Se $T_S^*(\theta)$ é a transformada Laplace-Stieljs (LTS) do tempo de difusão das estratégias, então $T_S^*(\theta) = E[e^{-\theta T_S}]$, onde $T_S = \sum_{i=1}^{n-1} S_i$

$$T_S^*(\theta) = E[e^{-\theta \sum_{i=1}^{n-1} S_i}] = \prod_{i=1}^{n-1} E[e^{-\theta S_i}] \text{ (II)}$$

Como consequência da distribuição exponencial e da função geradora de momento que mostram [Feller 1968]

$$E[x] = \int_0^\infty e^{xt}\lambda e^{-\lambda x}dx = \frac{\lambda}{\lambda - t} \text{ (III)}$$

Colocando as equações (I), (II) e (III) juntas, tem-se

$$T_S^*(\theta) = \prod_{j=1}^{n-1} \frac{\lambda(i)}{\lambda(i)+\theta} \text{ (IV)}$$

Da derivada da LST [Feller 1968]

$E[T_S] = -\frac{d}{d\theta}T_S^*(\theta) \mid_{\theta=0}$ (V)

Substituindo (IV) em (V), tem-se

$E[T_S] = \frac{d}{d\theta}[\prod_{i=1}^{n-1} \frac{\lambda(i)}{\lambda(i)+\theta}]$ (VI)

Considerando que $\frac{d}{dx}(\frac{c}{c+x}) = -\frac{c}{x^2+2cx+c^2}$,

e fazendo $P_i = \frac{\lambda(i)}{\lambda(i)+\theta}$ (VII)

tem-se que

$\frac{d}{d\theta}(P_i) = \frac{\lambda(i)}{\lambda^2(i)+2\lambda(i)\theta+\theta^2}$ (VIII)

Portanto,

$$E[T_S] = -\frac{d}{d\theta}T_S^*(\theta) = -\{\frac{d}{d\theta}[\prod_{i=1}^{n-1} P_i]\} = -\{\frac{d}{d\theta}P_1 \prod_{i=2}^{n-1} P_i + P_1\frac{d}{d\theta}[\prod_{i=2}^{n-1} P_i]\} =$$

$$-\{\frac{d}{d\theta}P_1 \prod_{i=2}^{n-1} P_i + P_1(\frac{d}{d\theta}P_2 \prod_{i=3}^{n-1} P_i + P_2\frac{d}{d\theta}\prod_{i=3}^{n-1} P_i)\} =$$

$$-\{\frac{d}{d\theta}P_1 \prod_{i=2}^{n-1} P_i + P_1\frac{d}{d\theta}P_2 \prod_{i=3}^{n-1} P_i + P_1P_2(\frac{d}{d\theta}P_3 \prod_{i=4}^{n-1} P_i + P_3\frac{d}{d\theta}\prod_{i=4}^{n-1} P_i)\} \text{ (IX)}$$

Continuando a derivar o produtório, fazendo $\theta = 0$ de acordo com (V) e substituindo esse valor de θ em (VII) e (VIII), tem-se que

$P_i = 1, \forall i = 1, ..., i = n-1$ (X) e

$\frac{d}{d\theta}P_i = -\frac{1}{\lambda(i)}$ (XI)

Dessa forma, substituindo (X) e (XI) em (IX) temos

$$E[T_S] = \sum_{i=1}^{n-1} \frac{1}{\lambda(i)} \text{ (XII)}$$

$$E[T_S] = \sum_{i=1}^{n-1} \frac{1}{i(n-j)v\lambda_{const}}$$

$$E[T_S] = \frac{1}{v\lambda_{const}} \sum_{i=1}^{n-1} \frac{1}{i(n-i)}$$

$$E[T_S] = \frac{1}{vn\lambda_{const}} \sum_{i=1}^{n-1} (\frac{1}{i} + \frac{1}{n-j}) \text{ (XIII)}$$

Considerando que $H_n = \sum_{i=1}^{n} \frac{1}{i}$ são os números harmônicos, e $H_{n-1} = \ln(n-1) + \gamma + o(\frac{1}{n-1})$, onde $\gamma \approx 0,57721$ é a constante de Euller.

Dessa forma, o tempo de difusão das estratégias dado por $E[T_S]$ é

$$E[T_S] = \frac{2}{vn\lambda_{const}} [\ln(n-1) + \gamma + o(\frac{1}{n-1})] \text{ (XIV)}$$

Modelo de nascimento e morte

O modelo anterior assume a idéia de que os dispositivos não entram ou saem da rede enquanto as estratégias estão sendo difundidas. Um cenário mais realístico, todavia, deve permitir a mobilidade dos dispositivos. Para isso, esta dissertação utilizou um modelo de nascimento e morte que descreve o impacto da entrada e saída dos dispositivos no tempo de difusão das estratégias [Feller 1968].

O modelo apresentado aqui considera que um dispositivo utilizando a estratégia s_e deixa a rede a uma taxa μ, e um outro dispositivo entra segundo a mesma taxa. Essa abordagem permite que o número de dispositivos permaneça constante, mas o número das estratégias s_e seja variável.

A figura 4.3 mostra a cadeia de Markov, considerando a entrada e saída de dis-

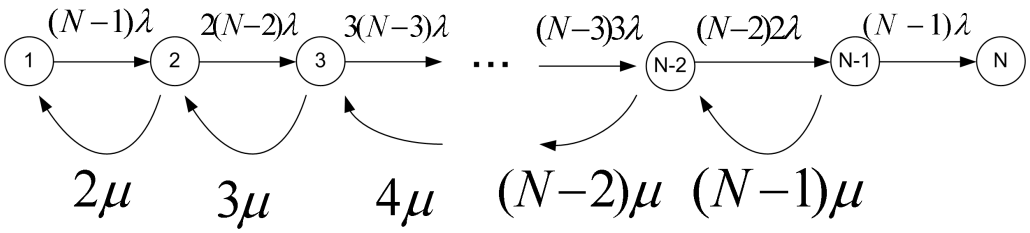


Figura 4.3: Difusão das estratégias com a saída dos dispositivos

positivos. Com isso, a cadeia pode sair do estado i para o estado $i+1$ a uma taxa $i(n-i)\lambda v$ quando uma estratégia eficiente é difundida, ou sair do estado i para $i-1$ a uma taxa $i\mu$ quando um dispositivo com uma estratégia eficiente deixa a rede. É importante ressaltar que esse modelo de nascimento e morte assume a noção de que a taxa na qual as estratégias são difundidas é maior que a taxa na qual os dispositivos deixam a rede, caso contrário as estratégias não seriam difundidas. Por isso,

$$v\lambda_{const} > \mu \text{ (XV)}$$

Considerando a entrada e a saída dos dispositivos, a taxa que o sistema sai do estado i é

$$\lambda(i) = i[(n-i)v\lambda_{const} - \mu] \text{ (XVI)}$$

Seguindo o mesmo raciocínio do modelo anterior, onde a saída dos dispositivos não era considerada, é fácil mostrar que a esperança de T_S para difusão das estratégias é dada pela equação (XII), a seguir:

$$E[T_S] = \sum_{i=1}^{n-1} \frac{1}{\lambda(i)}$$

$$E[T_S] = \sum_{i=1}^{n-1} \frac{1}{i[(n-i)v\lambda_{const} - \mu]}$$

$$E[T_S] = \sum_{i=1}^{n-1} \frac{1}{iv\lambda_{const}[n - i - \frac{\mu}{v\lambda_{const}}]}$$

$$E[T_S] = \frac{1}{v\lambda_{const}} \sum_{i=1}^{n-1} \frac{1}{i(n - i - \frac{\mu}{v\lambda_{const}})}$$

$$E[T_S] = \frac{1}{v\lambda_{const}} \sum_{i=1}^{n-1} [\frac{1}{i} + \frac{1}{n - i - \frac{\mu}{v\lambda_{const}}}] \frac{1}{n - \frac{\mu}{v\lambda_{const}}}$$

$$E[T_S] = \frac{1}{v\lambda_{const}(n - \frac{\mu}{v\lambda_{const}})} \sum_{i=1}^{n-1} [\frac{1}{i} + \frac{1}{n - i - \frac{\mu}{v\lambda_{const}}}]$$

$$E[T_S] = \frac{1}{nv\lambda_{const} - \mu} \sum_{i=1}^{n-1} [\frac{1}{i} + \frac{1}{n - i - \frac{\mu}{v\lambda_{const}}}]$$

De acordo com a equação (XV), onde $v\lambda_{const} > \mu$, o termo $\frac{\mu}{v\lambda}$ pode ser descartado. Com isso, a esperança para o tempo de difusão das estratégias com saída dos dispositivos é dada por

$$E[T_S] = \frac{2}{nv\lambda_{const} - \mu}[\ln(n-1) + \gamma + o(\frac{1}{n-1})] \text{ (XVII)}$$

Perceba que à medida que $\mu \to \lambda vN$, $T_S \to \infty$.

Gráficos

O objetivo principal desse modelo matemático é entender como o fator de difusão influencia o tempo de difusão das estratégias. Para entender como outros parâmetros, como padrão de mobilidade, velocidade média dos dispositivos ou alcance da antena impactam na taxa de encontro dos nós, o leitor pode consultar [Groenevelt et al. 2005, Sharma et al. 2007].

A figura 4.4 mostra o tempo de difusão das estratégias para o cenário onde não existe a saída dos dispositivos, para diferentes taxas de difusão v, variando o número de dispositivos. Nesta figura, quando não existem mais do que 20 dispositivos, um

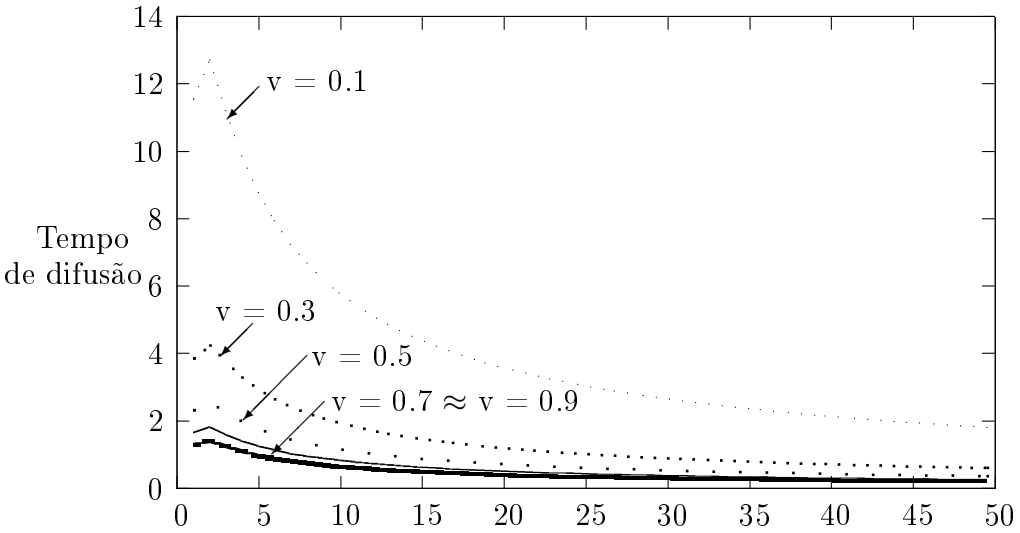


Figura 4.4: Tempo de difusão sem a saída dos dispositivos, para diferentes taxas de difusão v, variando o número de dispositivos

fator de difusão $v = 0.5$ apresenta um tempo de difusão semelhante ao caso onde todos os vizinhos são escolhidos (fator igual a um). Por outro lado, quando o número de dispositivos aumenta, um fator $v = 0.3$ é suficiente para que o tempo de difusão seja próximo ao caso onde todos os vizinhos são escolhidos.

Dessa forma, o *AESPmob* utiliza uma abordagem adaptativa para o fator de difusão de acordo com o número de vizinhos detectados. Quando o número de vizinhos aumenta, o fator de difusão pode ser diminuído, minimizando o custo na rede, sem impactar significativamente no tempo de difusão. Por outro lado, se a rede ficar mais esparsa, o fator de difusão pode ser aumentado, para melhorar o tempo de difusão.

Já a figura 4.5 mostra o tempo de difusão das estratégias, onde a saída dos dispositivos é considerada, para um fator de difusão $v = 0.3$. Essa figura compara os cenários onde a taxa de saída μ é 90%, 50% e 30% da taxa de encontro dos dispositivos λ, com o caso onde não existe saída dos dispositivos.

A escolha da taxa de difusão $v = 0.3$ foi feita baseada em dois motivos. Primeiro, de acordo com a equação (XVII), um fator de difusão baixo permite analisar como a taxa de saída dos dispositivos μ impacta na difusão das estratégias. Segundo, esse

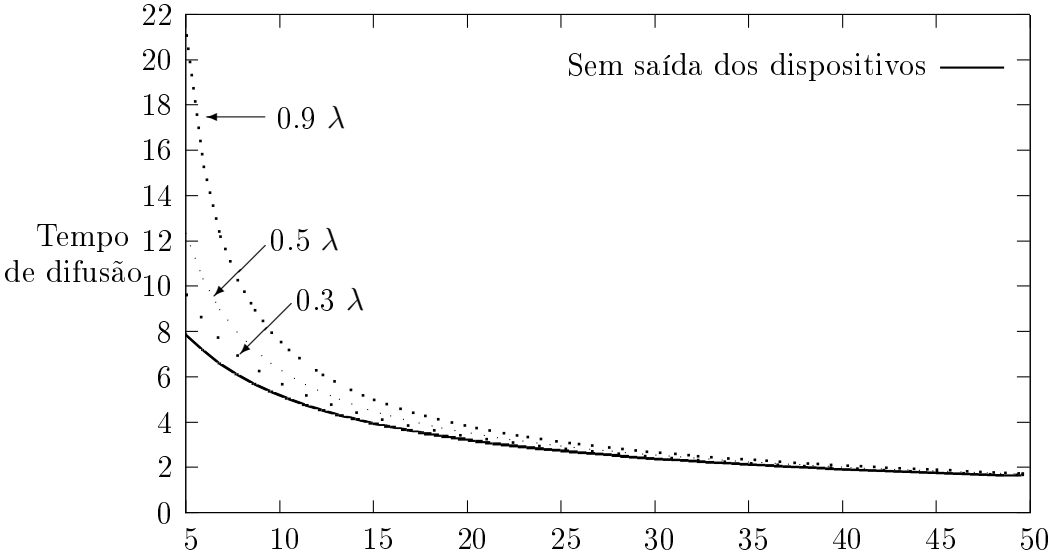


Figura 4.5: Tempo de difusão com a saída dos dispositivos, para diferentes taxas de saída e um fator de difusão $v = 0.3$, variando o número de dispositivos

é um fator que apresenta uma boa relação entre custo na rede e tempo de difusão para redes com densidades médias e altas.

Com o fator de difusão $v = 0.3$, quando o número de dispositivos é pequeno (menor que 10 dispositivos), o tempo de difusão é afetado de forma significativa pela saída dos dispositivos. Quando, porém, o número de dispositivos aumenta, a taxa com a qual os dispositivos saem da rede perde importância.

A dificuldade em adaptar o fator de difusão está na necessidade de conhecer o número total de dispositivos existentes. Por outro lado, em redes móveis, as decisões normalmente devem ser baseadas em parâmetros locais, e decisões baseadas em informações globais devem ser evitadas [Mamei e Zambonelli 2005]. Por isso, a adaptação do fator de difusão é feita baseada em uma estimativa do número total de dispositivos, e considerando apenas o número de vizinhos percebidos.

Assim, admitindo que o raio r de transmissão é pequeno em relação à área onde os dispositivos estão dispersos, o número médio de vizinhos é dado por $d_{avg} \sim \pi r^2(n-1)$ [Bar-Yossef et al. 2008]. O valor para o alcance da antena do dispositivo, que garante com alta probabilidade a conectividade da rede, é $r = \sqrt{\frac{C \ln n}{n}}$, onde C é uma constante, e $C > \frac{1}{\pi}$ [Gupta e Kumar 1998]. Dessa forma, fazendo $C = 1$, o número

médio de vizinhos é

$$d_{med} = \pi \ln n \text{ (XVIII)}$$

De acordo com a figura 4.5, é possível perceber que a taxa de saída dos dispositivos começa a perder importância para uma rede com número superior a 20 dispositivos. Para adaptar o fator de difusão de acordo com o número de dispositivos, é possível utilizar a equação (XVIII), que relaciona o número de vizinhos de acordo com o número total de dispostivos.

Utilizando o número de 20 dispositivos como base para adaptar o fator de difusão v, e de acordo com a equação (XVIII), onde o número médio de vizinhos é aproximadamente 9 para uma rede com 20 dispositivos, o fator de difusão pode ser adaptado da seguinte maneira:

$$v = \begin{cases} 50\%, & \text{se } d_{med} < 9, ou \\ 30\%, & \text{caso contrário} \end{cases}$$

É importante ressaltar que a dedução da equação (XVIII) faz algumas suposições sobre os dispositivos e a densidade da rede. Dessa forma, os valores previstos pela equação podem não ser refletidos na prática.

Como o objetivo dessa modelagem, porém, é dar uma indicação de como o fator de difusão deve ser adaptado utilizando apenas parâmetros locais, e não uma previsão precisa do número de dispositivos, os erros introduzidos por essas suposições são aceitáveis. Mais ainda, o número mínimo e máximo de vizinhos é da mesma ordem de grandeza do número médio [Bar-Yossef et al. 2008], apontando que a estimativa do número total de dispositivos utilizando o número de vizinhos pode dar uma indicativa razoável sobre o número total de dispositivos.

4.4.3 Relação entre custo de difusão e ganho em eficiência

A cada momento i, a estratégia $s_j, j \in E$, onde E é o conjunto das estratégias existentes, insere um custo sobre a rede. Por exemplo, um mecanismo de descoberta de serviços introduz um custo referente ao número de mensagens de descoberta enviadas. Dessa forma, c_j representa o custo por instante de tempo referente à estratégia s_j.

Com isso, quando o sistema está no estado i, existem i dispositivos utilizando a estratégia mais eficiente s_e. Essa estratégia tem um custo c_e associado a ela. Adicionalmente, existem $(n-i)$ dispositivos utilizando todas as outras estratégias. Para cada uma dessas estratégias não-eficientes, será atribuído um mesmo custo c_s. Esse é o custo associado à estratégia com menor custo ($c_s < c_e$). Essa abordagem permite comparar o ganho de eficiência em difundir a estratégia mais eficiente com o custo introduzido por essa estratégia.

Nessa comparação entre o ganho de eficiência e o custo introduzido será considerado o modelo sem a saída dos dispositivos, de acordo com a figura 4.2. O sistema sai do estado i para o $i+1$ a uma taxa $i(n-i)v\lambda_{const}$, e fica no estado i por um tempo $\frac{1}{i(n-i)v\lambda_{const}}$ [Feller 1968].

Cálculo do custo

Se C_i representa a soma do custo de todas as i estratégias utilizando s_e mais o custo de todas as outras estratégias, então

$$C_i = [c_e * i + c_s(n-i)]\frac{1}{i(N-i)\lambda v} \quad \text{(XIX)}$$

Assim, o custo total na rede é

$$C_T = \sum_{i=1}^{n-1} C_i$$

$$C_T = \sum_{i=1}^{n-1} [c_e * i + c_s(n-i)] \frac{1}{i(n-i)\lambda v} \text{ (XX)}$$

Desenvolvendo a equação (XX), e seguindo os mesmos passos utilizados para calcular o tempo de difusão, é fácil mostrar que o custo total, dado por C_T, até a estratégia ser difundida é

$$C_T = \frac{c_e + c_s}{\lambda v}[\log(n-1) + \gamma + o(\frac{1}{n-1})] \text{ (XXI)}$$

Seja $K = \frac{1}{\lambda v}[\log(n-1) + \gamma + o(\frac{1}{n-1})]$, tem-se

$$C_T = (c_e + c_s)K$$

Definindo o custo adicional como a diferença entre o custo em usar a estratégia com o menor custo e utilizar a estratégia mais eficiente, então

$$C_{ad} = (C_s - C_e)K$$

Como $C_s = 2c_sK$ e $C_e = (c_e + c_s)K$, então

$$C_{ad} = (c_e - c_s)K \text{ (XXII)}$$

Cálculo do benefício

Considerando que b_e e b_s são o benefício por instante de tempo em utilizar a estratégia mais eficiente e a com o menor custo, respectivamente. Então, seguindo os mesmos argumentos matemáticos utilizados para o cálculo do custo, o benefício total dado por B_T é

$$B_T = \frac{b_e+b_s}{\lambda v}[\log(n-1) + \gamma + o(\frac{1}{n-1})] \text{ (XXIII)}$$

Por analogia, o benefício adicional é a diferença entre utilizar a estratégia mais eficiente e aquela com menor custo. Com isso,

$$B_e = (b_e + b_s)K \text{ e } B_s = 2b_sK$$

$$B_{ad} = (b_e - b_s)K \text{ (XXIV)}$$

Admitindo que B_{ad} e C_{ad} são da mesma magnitude, a escolha em tempo de execução entre a estratégia mais eficiente ou a menos custosa pode ser feita utilizando a razão $\frac{B_{ad}}{C_{ad}}$. Quando a aplicação não definir os níveis de QoS necessários, essa fração vai definir a escolha entre o benefício ou o custo. Se a razão for maior que 1, então a escolha é pela estratégia mais eficiente. Caso contrário, o custo é utilizado.

4.5 Conclusões

Este capítulo descreveu as principais contribuições desta dissertação. Primeiro, o *AESPmob* foi apresentado, mostrando, de uma maneira geral, como as aplicações utilizam os serviços essenciais. Depois foi detalhada a arquitetura do *AESPmob*, mostrando os módulos de contexto e QoS, além da gerência do sistema de *middleware*. Adicionalmente, o funcionamento do *AESPmob* foi descrito. Esse funcionamento está dividido em invocação dos serviços essenciais e difusão das estratégias.

Em seguida, foi apresentado um modelo estocástico, baseado em cadeias de Markov, para descrever o tempo de difusão das estratégias mais eficientes. Esse modelo considerou dois cenários principais: sem e com a saída de dispositivos. Nos dois cenários, o tempo de difusão diminui à medida que o número de dispositivos aumenta. Esse modelo mostrou ainda como a fator de difusão do *AESPmob* por ser adaptado

de acordo com o número de vizinhos percebidos em dado momento.

Ainda no modelo matemático, foi analisado como a eficiência e o custo variam com a difusão das estratégias. Adicionalmente, foi proposto um mecanismo para auxiliar na escolha em tempo de execução entre eficiência e custo na definição da estratégia a ser difundida.

O próximo capítulo apresenta as simulações feitas para validar o *AESPmob* e mostrar como a eficiência total na rede varia com a difusão das estratégias mais eficientes.

Capítulo 5

AVALIAÇÃO DO AESPmob

Este capítulo traz a implementação do *AESPmob* em simulador, enfatizando como a eficiência dos requisitos primários varia com a difusão das estratégias. Na seção 5.1 é feita breve introdução sobre o serviço essencial descoberta de serviços e o ambiente de simulação utilizado. Já na seção 5.2 são detalhadas as estratégias implementadas para avaliar *AESPmob*, além de descritas as métricas utilizadas para comparar as estratégias. Para finalizar, a seção 5.3 apresenta e discute os resultados da simulação.

5.1 Introdução

O objetivo principal do *AESPmob* é buscar uma rápida adaptação dos serviços essenciais às mudanças do ambiente, além de facilitar a evolução das estratégias que implementam esses serviços mediante a difusão das estratégias mais eficientes.

Dessa forma, buscando entender como a difusão dessas estratégias impacta na eficiência geral de um serviço essencial, foram implementadas em simulador quatro estratégias para a descoberta de serviços. A descoberta de serviços foi escolhida por ser amplamente discutida na literatura e bastante susceptível às variações da arquitetura e às condições momentâneas na camada de rede [Engelstad et al. 2006].

O simulador escolhido para implementar a difusão das estratégias proposta pelo

AESPmob foi o *Jist/Swans* [Cornell 2008]. Dentre os simuladores pesquisados, ele se destacou pela maior facilidade de implementação de protocolos na camada de aplicação, por ser mais eficiente em termos de processamento e consumo de memória, além de ser código aberto [Barr et al. 2005, Schoch et al. 2008].

Utilizando o *Jist/Swans*, o *AESPmob* foi simulado em uma rede *ad hoc* totalmente desestruturada, e os dispositivos dispersos inicialmente de forma aleatória seguindo distribuição uniforme, de acordo com os parâmetros mostrados na tabela 5.1 e tabela 5.2. A primeira tabela mostra os parâmetros referentes à mobilidade dos dispositivos. Como o objetivo da simulação é entender o impacto da difusão das estratégias na eficiência de um dado serviço essencial, esses parâmetros possuem pouca relevância e não foram analisados. Em particular, o tempo de simulação foi escolhido para que em todos os cenários simulados as estratégias sejam totalmente difundidas.

Já a segunda tabela apresenta os parâmetros escolhidos para simular o *AESPmob*. Foram realizadas simulações preliminares para definir os parâmetros fixos da simulação. A densidade e taxa de requisição de serviços foram escolhidos de forma a dificultar a descoberta de um serviço e a eficiência dos diferentes protocolos pudesse ser analisada. Os parâmetros variáveis são a taxa de difusão v do *AESPmob* e o número de dispositivos.

Tabela 5.1: Parâmetros da simulação

Parâmetro	**Valor**
Área	1000m x 1000m
Tempo total de simulação	400s
Tempo de aquecimento	40s
Tempo de resfriamento	40s
Raio transmissão	100m
Padrão de mobilidade	*Random way point*
Velocidade mínima	3 km/h
Velocidade máxima	5 km/h
Tempo de pausa	0s

A cada dispositivo da simulação foi atribuída uma das quatro estratégias para

Tabela 5.2: Parâmetros do *AESPmob*

Parâmetro	**Valor**
Número de dispositivos	20 - 100
Fator de difusão	0.2 - 0.8
Período entre verificações das estratégias	10s
Densidade dos serviços	10%
Taxa requisição dos serviços	10 serviços a cada 15s
Número de Simulações	500

o serviço essencial descoberta de serviços descritas na seção 5.2. Essa atribuição inicial das estratégias é feita de forma aleatória, seguindo a distribuição de probabilidade uniforme. Assim, durante a simulação, as estratégias são avaliadas segundo diferentes métricas de *QoS* e difundidas de acordo com essas métricas.

5.2 Descrição das estratégias

Utilizando o *Jist/Swans*, foram implementados quatro protocolos de descoberta de serviços (estratégias) escolhidos de forma representativa, um para cada classe de protocolos: 1) inundação, 2) *gossiping* probabilístico, 3) RAPID (protocolo com alta confiabilidade) e 4) estratégia híbrida proativa. Cada um desses protocolos possui características bem específicas e sua eficiência é altamente influenciada por características momentâneas da rede.

Inundação

O protocolo de descoberta de serviços baseado em inundação é caracterizado por um alto custo na rede, elevada taxa de entrega, baixa latência, e baixa complexidade de implementação. Para minimizar o custo na rede, o parâmetro TTL é utilizado para limitar o número máximo de retransmissões.

Aumentar o TTL induz um ganho na taxa de descoberta dos serviços, mas também introduz um alto custo na rede. Se a métrica utilizada para avaliar a eficiência da estratégia for taxa de entrega, essa estratégia é uma boa candidata

para ser difundida. Por outro lado, se a métrica considerar o custo, é provável que ela não seja difundida. Para a avaliação do *AESPmob*, o TTL foi fixado em 5, que é um valor suficientemente alto para que uma mensagem atinja toda a rede [Meyer et al. 2002].

Gossiping probabilístico

O *gossiping* [Khelil et al. 2007] procura manter a taxa de entrega semelhante ao inundação, mas minimizando o custo na rede, além de apresentar uma latência média. A complexidade de implementação também é baixa. No *gossiping*, os dispositivos retransmitem, com uma dada probabilidade, mensagens anteriores quando encontram novos vizinhos. Essa probabilidade de retransmissão influencia diretamente na taxa de descoberta dos serviços e no custo sobre a rede.

Na simulação do *AESPmob*, a probabilidade de retransmissão foi fixada em 0.5. Essa probabilidade permite que a taxa de entregue seja ainda alta, mas mantém o custo em um valor aceitável.

RAPID: realiable probabilistic dissemination in wireless networks

No RAPID [Drabkin et al. 2007], as mensagens são retransmitidas com baixa probabilidade e medidas de correção determinísticas são aplicadas para aumentar a confiabilidade da entrega. Adicionalmente, o cabeçalho contendo informações sobre uma dada mensagem é periodicamente retransmitido.

O RAPID possui elevada taxa de entrega das mensagens, um custo menor que o *gossiping*, uma latência média, e uma maior complexidade de implementação.

Estratégia híbrida proativa

Na estratégia híbrida proativa [Engelstad et al. 2006], os dispositivos que possuem os serviços proativamente enviam mensagens anunciando os serviços disponíveis. Antes de enviar para a rede a busca por um serviço, um dispositivo verifica se

já recebeu anteriormente um anúncio de serviço semelhante. Caso não tenha recebido, esse dispositivo inicia uma busca por inundação. Ao receber uma requisição, um dispositivo verifica se tem esse serviço, ou se já recebeu num anúncio de serviço com essas características.

Essa abordagem tem uma complexidade alta de implementação, pois anúncios de serviço podem se tornar inválidos. Ela possui ainda baixa latência, já que os anúncios podem ser acessados localmente, mas possui o maior custo na rede. A estratégia proativa apresenta ainda boa taxa de descoberta.

5.3 Resultados

A difusão das estratégias foi analisada de acordo com três métricas de *QoS* diferentes. A cada 10 segundos, mais um período entre 2 e 5s definido aleatoriamente com distribuição uniforme, um dispositivo escolhe um subconjunto de vizinhos para comparar as estratégias, de acordo com a taxa de difusão.

Essa comparação é feita de três maneiras: 1) número de serviços descobertos, onde as estratégias que apresentarem um número maior de serviços encontrados são difundidas; 2) razão entre o número de serviços descobertos e o número de mensagens enviadas, e as estratégias com uma razão maior são difundidas; e 3) latência média de descoberta, onde quanto menor a latência, mais eficiente é um serviço.

Uma característica importante que precisa ser verificada, é como o *AESPmob* analisa, em tempo de execução, qual estratégia é mais eficiente de acordo com a métrica de *QoS* utilizada. Para isso, foram implementadas três abordagens para analisar a eficiência de uma estratégia e comparadas com a abordagem onde as estratégias não são difundidas.

A primeira maneira utilizada nesta dissertação para quantificar a eficiência de uma estratégia é utilizar apenas a visão local de cada dispositivo. Essa é a abordagem local e utiliza um histórico local do uso da estratégia. Esse histórico é comparado

com o nível de eficiência fornecido por um determinado vizinho e a estratégia que apresentar maior eficiência é difundida.

Outra maneira utiliza informações compartilhadas pelos vizinhos. Essa abordagem é chamada de cooperação e quantifica a eficiência de uma estratégia utilizando uma média dos níveis de eficiência recebidos de vários dispositivos durante determinado período. Essa abordagem troca mais mensagens que a anterior, mas permite melhor julgamento sobre as estratégias mais eficientes.

A terceira forma utiliza informações compartilhadas por todos os dispositivos da rede, e é chamada de abordagem global. Como a avaliação é feita em simulador, uma variável compartilhada (estática) guarda o nível de eficiência das estratégias, e é acessada no momento de definir quais estratégias serão difundidas. Essa abordagem global é a melhor maneira de avaliar uma estratégia, no entando, em dispositivos reais, o custo na rede é muito alto, pois muitas mensagens são trocadas para manter o valor da eficiência das estratégias consistente.

A abordagem global possi uma maior chance de tomar decisões corretas, já que a decisão sobre qual a estratégia mais eficiente é feita baseada em informações compartilhadas por todos os nós. Esse comportamento é comprovado observando que e a abordagem global apresenta maior eficiência em todos os cenários simulados.

O problema em utilizar uma abordagem global é que, em ambientes reais, não é simples coletar e manter atualizadas informações compartilhadas por todos os dispositivos. Mais ainda, essa abordagem normalmente insere um custo proibitivo na rede. Por outro lado, a abordagem cooperativa não é tão eficiente quanto a global, mas tem um custo menor. Já decisões baseadas em informações locais podem não ser confiáveis, mas também apresentam o menor custo dentre as três abordagens.

5.3.1 Eficiência total da descoberta de serviços

Para medir a eficiência total da descoberta de serviços, as estratégias são aleatoriamente atribuídas aos nós e a simulação iniciada. A partir de então, o nível de

eficiência de cada estratégia é monitorado. O objetivo é identificar quais estratégias são mais eficientes em cada cenário e difundir essa estratégia, verificando como a eficiência total do serviço essencial descoberta de serviços se comporta.

As figuras 5.1 e 5.2 mostram como a eficiência total varia à medida que a simulação acontece. Nessas figuras, a difusão das estratégias é feita utilizando a métrica de *QoS* número de serviços descobertos. O eixo Y mostra como o número de serviços descobertos varia durante a execução com a difusão das estratégias. São analisadas diferentes abordagens de avaliação: global, cooperação, local e sem difusão. Já o eixo X representa o tempo da simulação.

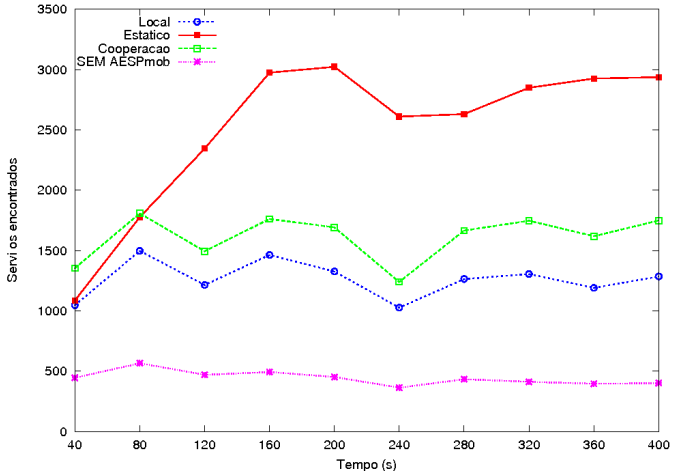


Figura 5.1: Número de Serviços Descobertos para 20 nós e fator de difusão de 0.4

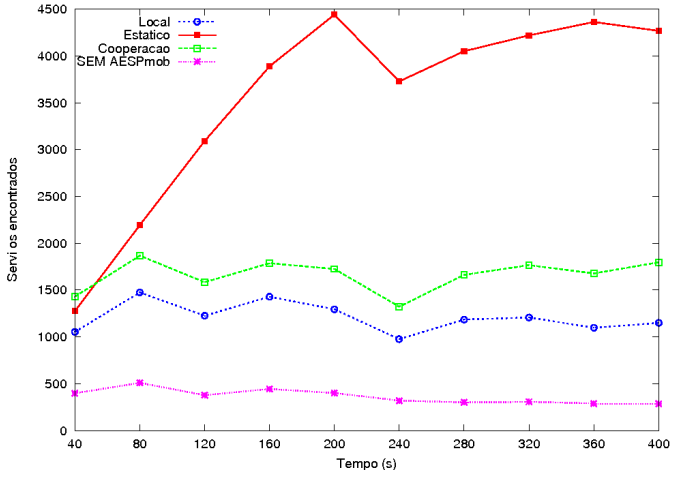


Figura 5.2: Número de Serviços Descobertos para 60 nós e fator de difusão de 0.2

A figura 5.1, onde foi simulado um cenário com 20 nós e as estratégias difundidas com fator 0.4, e a figura 5.2, para 60 dispositivos e fator de difusão 0.2, mostram como a eficiência varia com a difusão das estratégias segundo a métrica de *QoS* número de serviços descobertos. Nelas, a eficiência da abordagem global (estática) varia até o tempo de simulação 160 segundos, quando a estratégia mais eficiente é totalmente difundida e todos os nós estão utilizando a mesma estratégia. Após esse momento, a eficiência geral dessa abordagem permanece constante.

As abordagens utilizando cooperação e local também apresentam um ganho de eficiência, mas a uma taxa menor. Isso acontece porque decisões baseadas em informações locais, ou utilizando um subconjunto pequeno de nós, não se mostram confiáveis como a abordagem global, e algumas estratégias ineficientes são difundidas. Mesmo que estratégias ineficientes sejam, porém, eventualmente difundidas, as mais eficientes prevalecem no final da execução. Assim, as duas abordagens apresentaram uma eficiência maior do que na abordagem sem difusão.

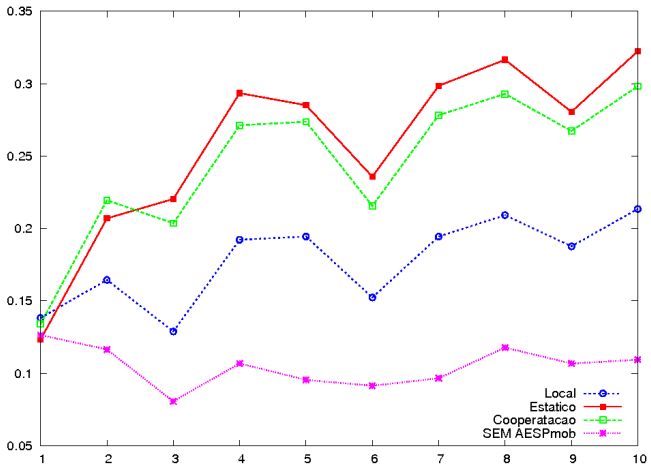


Figura 5.3: Razão entre serviços descobertos e mensagens enviadas para 80 nós e fator de difusão de 0.6

Nas figuras 5.3 e 5.4 são analisados o modo como a eficiência da razão entre o número de serviços descobertos e mensagens enviadas varia com a difusão das estratégias. Todas as abordagens com difusão apresentaram um ganho de eficiência. É possível perceber por essas figuras que a diferença entre as abordagens globais

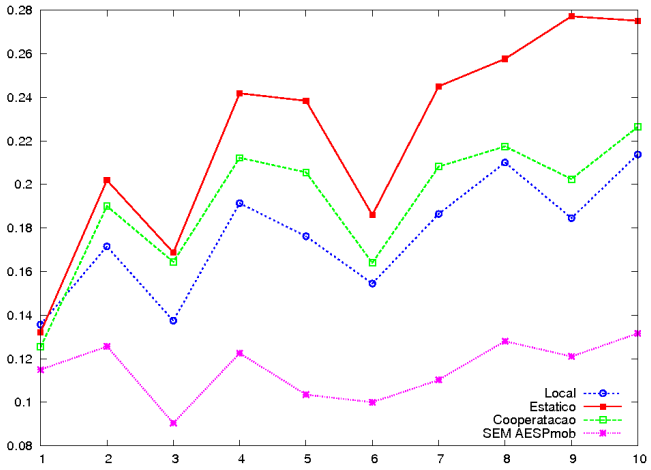


Figura 5.4: Razão entre serviços descobertos e mensagens enviadas para 60 nós e fator de difusão de 0.8

e cooperação é menor do que na métrica do número de serviços descobertos. Isso acontece porque as estratégias baseadas em inundação e *gossiping* enviam muito mais mensagens que as outras estratégias, o que diminui o valor da razão, e são facilmente identificadas como ineficientes.

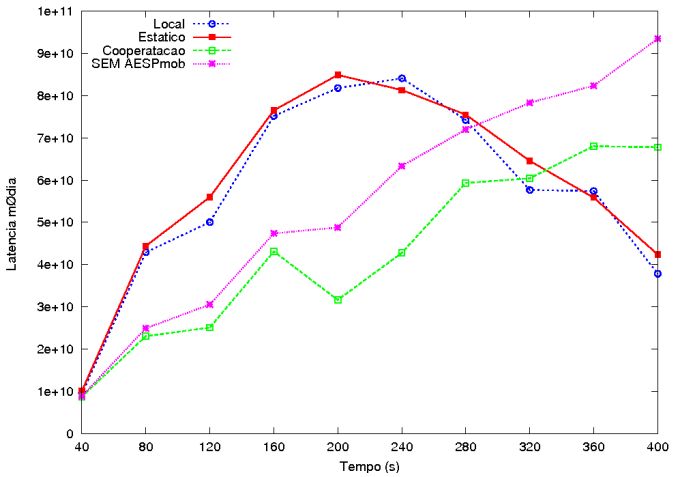


Figura 5.5: Latência média de descoberta para 40 nós e fator de difusão de 0.2

Já nas figuras 5.5 e 5.6, a métrica de *QoS* considerada é a latência média de descoberta. Particularmente utilizando essa métrica, a abordagem sem difusão apresentou inicialmente desempenho melhor, mas, à medida que a simulação acontecia e as estratégias são difundidas, as abordagens com difusão atingiram um nível de eficiência melhor. Isso aconteceu porque a latência de descoberta é mais suscetível

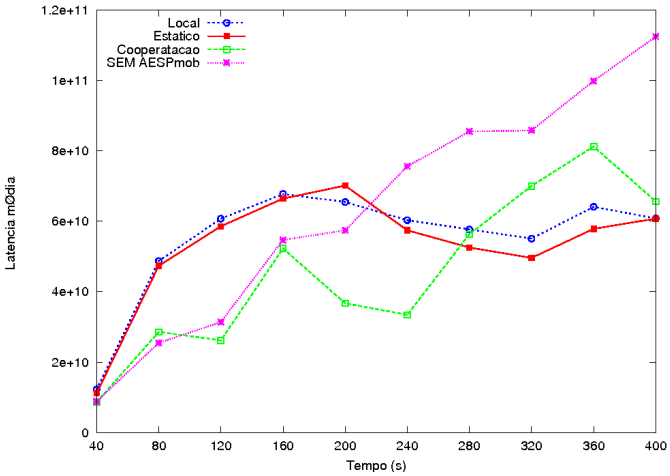
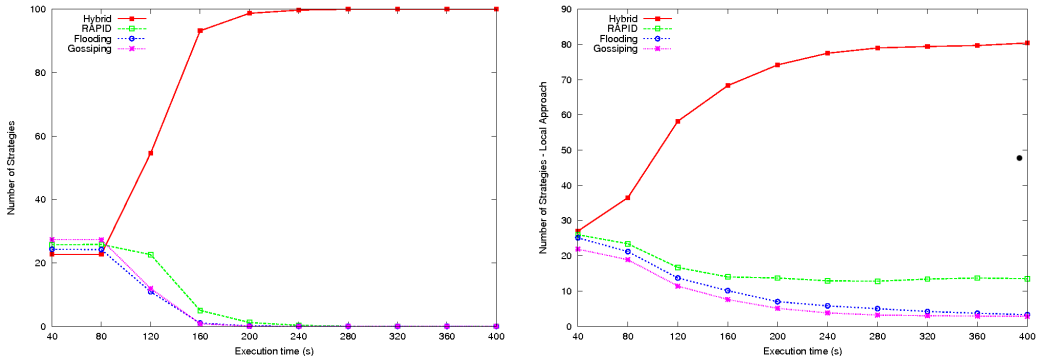


Figura 5.6: Latência média de descoberta para 60 nós e fator de difusão de 0.6

às variações das camadas inferiores, como mobilidade dos dispositivos. Dessa forma, mesmo na abordagem global, estratégias ineficientes são difundidas.

5.3.2 Difusão das estratégias eficientes

No início da simulação, as quatro estratégias são atribuídas aleatoriamente aos nós segundo uma distribuição de probabilidade uniforme. Dessa forma, quando a simulação foi iniciada e a eficiência das estratégias monitorada, o número de cada estratégia sendo utilizada pode aumentar ou diminuir de acordo com seu nível de eficiência. O objetivo é maximizar o número de estratégias eficientes e minimizar o número de ineficientes.



(a) Visão global (b) Visão local

Figura 5.7: Número de cada estratégia para 100 nós e um fator de difusão de 0.4

A figura 5.7 mostra o número de cada uma das quatro estratégias sendo utilizadas à medida que a simulação acontece. Nessa figura, a métrica utilizada na difusão é o número de serviços descobertos. A figura 5.7a mostra como o número de cada estratégia varia usando a abordagem global, e a figura 5.7b exibe o modo como esse número varia utilizando a abordagem local.

Segundo a métrica número de serviços descobertos, a estratégia mais eficiente é a híbrida. A abordagem global (figura 5.7a) difunde as estratégias de maneira muito mais rápida do que a local (figura 5.7b). Aproximadamente no tempo de simulação de 160 segundos, a abordagem global tem difundido totalmente a estratégia híbrida, e todas as outras estratégias eliminadas. Já na abordagem local, a estratégia híbrida não é totalmente difundida, e outras estratégias menos eficientes ainda permanecem sendo utilizadas ao final da simulação.

5.4 Conclusões

Esse capítulo apresentou a implementação do modelo de avaliação, difusão e evolução dos serviços essenciais proposto pelo *AESPmob*. Para isso, foram utilizadas quatro estratégias diferentes para o serviço essencial descoberta de serviços.

As estratégias implementadas foram inundação, o *gossiping* probabilístico, o RAPID e a híbrida proativa. Durante a simulação, a eficiência dessas estratégias foi avaliada segundo três métricas distintas: número de serviços descobertos, dazão entre número descoberto e mensagens enviadas e latência média de descoberta.

Para avaliar a eficiência das estratégias de acordo com cada métrica, três abordagens distintas foram utilizadas: uma baseada na visão global, a segunda em uma visão cooperativa e a última utilizando apenas informações locais. Dentre essas três, a visão global apresentou a melhor eficiência e tempo de difusão menor em todos os casos. Além disso, todas as três abordagens foram mais eficientes do que aquela abordagem onde as estratégias não são difundidas.

O próximo capítulo discute as conclusões e as contribuições dessa dissertação, além de levantar os trabalhos futuros.

Capítulo 6

CONCLUSÃO E TRABALHOS FUTUROS

Esta dissertação propôs o AESPmob, um modelo autônomo e evolutivo para provisão de serviços essenciais em redes móveis. Os principais objetivos desse modelo são o constante monitoramento dos serviços essenciais, o compartilhamento das estratégias mais eficientes para permitir uma rápida adaptação dos protocolos na camada de aplicação às variações na camada de rede, além de possibilitar uma evolução dos serviços essenciais.

Dessa forma, mediante o compartilhamento das estratégias mais eficientes, o nível de eficiência total dos serviços essenciais melhora, além de facilitar a manutenção e evolução das diferentes estratégias que implementam esses serviços.

O restante deste capítulo lista as principais contribuições desta dissertação na seção 6.1 e os trabalhos futuros na seção 6.2.

6.1 Contribuições

A alta dinamicidade e a incerteza inerentes aos sistemas móveis requerem soluções que sejam capazes de rapidamente perceber mudanças no ambiente e identificar quais ações precisam ser tomadas para minimizar o impacto no funcionamento das

aplicações. Em sistemas computacionais, essa alta responsividade a estímulos do ambiente é alcançada através do uso de soluções autônomas.

Nesse contexto, a principal contribuição desta dissertação é o AESPmob, um modelo que identifica as mudanças que acontecem no ambiente, adapta e evolui o comportamento dos serviços essenciais para melhor lidar com essas mudanças.

Dessa forma, as aplicações móveis são construídas utilizando serviços chamados essenciais, que são funcionalidades comuns às aplicações móveis distribuídas. Diferentes estratégias para esses serviços são adequadas aos mais diversos contextos. O papel do AESPmob é medir a adequação dessas estratégias às variações do contexto.

Para entender como as estratégias se adaptam aos diferentes contextos no qual elas serão utilizadas, além de perceber como requisitos inerentes às aplicações afetam o comportamento das diferentes estratégias, a segunda contribuição desta dissertação é um *survey* detalhado dos serviços essenciais. Nele, as diferentes maneiras de implementar um serviço foram analisadas, o que permitiu identificar dependências internas a um serviço essencial, além de perceber como cada serviço influencia e é influenciado pelos outros serviços essenciais.

Depois de detalhar os serviços essenciais, esta dissertação descreveu como as diferentes características de cada arquitetura de rede móvel influenciam na escolha das diferentes implementações.

A partir desse estudo detalhado dos serviços essenciais nas diferentes tecnologias de redes móveis, a adaptação e a evolução de cada serviço essencial acontecem de forma transparente às aplicações. Ela é feita pelo AESPmob de forma colaborativa utilizando informações locais e obtidas em dispositivos próximos. A ideia é que, à medida que uma nova implementação é identificada como mais eficiente, ela seja difundida para ser utilizada por todos os dispositivos.

Desde o momento em que as estratégias mais eficientes são difundidas, a eficiência total do serviço essencial cresce. Essa premissa foi validada de duas maneiras. Na primeira, um modelo analítico descreve como o tempo de difusão, o benefício inserido

e o custo associado variam à medida que o número de dispositivos e a taxa de difusão das estratégias varia.

Esse modelo é mais uma contribuição desta dissertação. Ele pode ser utilizado para descrever o tempo de difusão de informações em cenários totalmente distribuídos, onde a entrada e saída de dispositivos é considerada.

O modelo analítico serviu ainda para guiar a adaptação do fator de difusão, utilizando como parâmetro apenas o número de vizinhos percebidos em um dado momento. Em redes densas, essa adaptação permite minimizar o custo associado à difusão das estratégias, sem alterar significativamente o tempo de difusão. Já nas redes esparsas, a adaptação do fator de difusão procura minimizar o impacto da saída dos dispositivos.

A segunda forma de validar o ganho em desempenho é utilizando um simulador. Na simulação é verificado que a eficiência total do serviço essencial realmente aumenta. Percebeu-se, contudo, que a forma como uma estratégia é avaliada possui relevância significativa no ganho inserido e no tempo de difusão, pois quanto maior a participação dos nós no processo de coleta e agregação dos dados referentes à eficiência das estratégias, maior a chance de escolher uma estratégia mais eficiente.

6.2 Trabalhos futuros

O próximo passo é analisar como a coleta das informações sobre a eficiência pode ser melhorada para aumentar a probabilidade de uma escolha correta das estratégias mais eficientes. Abordagens locais são menos confiáveis, porém, mais leves. Por outro lado, o uso de informações globais insere um custo muitas vezes proibitivo, mas com a vantagem de fornecer um guia mais correto.

A seguir, é necessário ainda entender mais detalhadamente como os mecanismos de descrição de *QoS* existentes podem ser utilizados pelo *AESPmob*. Levantamentos preliminares mostram que será necessária a criação de um modelo que descreva as

características de cada serviço essencial.

Existe ainda a necessidade do desenvolvimento do AESPmob em dispositivos reais para verificar como as premissas utilizadas no modelo analítico e na descrição do *AESPmob* se comportam quando praticados.

Referências Bibliográficas

[Andrade e Logrippo 2006] Andrade, R. M. C. and Logrippo, L. (2006). Morar: A pattern language for mobility and radio resource management. In Dragos Manusecu, Markus Völter, J. N., editor, *Pattern Language of Program Design 5*, chapter 10, pages 213–256. Addison-Wesley.

[Andrews 1991] Andrews, G. R. (1991). *Concurrent programming: principles and practice*. Benjamin-Cummings Publishing Co., Inc., Redwood City, CA, USA.

[Baduel e Matsuoka 2007] Baduel, L. and Matsuoka, S. (2007). A peer-to-peer infrastructure for autonomous grid monitoring. *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*, pages 1–8.

[Bailey 1975] Bailey, N. (1975). *The Mathematical Theory of Infectious Diseases and its Applications*. Griffin, London.

[Bar-Yossef et al. 2008] Bar-Yossef, Z., Friedman, R., and Kliot, G. (2008). Rawms - random walk based lightweight membership service for wireless ad hoc networks. *ACM Trans. Comput. Syst.*, 26(2):1–66.

[Barr et al. 2005] Barr, R., Haas, Z. J., and van Renesse, R. (2005). Jist: an efficient approach to simulation using virtual machines: Research articles. *Softw. Pract. Exper.*, 35(6):539–576.

[Benda et al. 2006] Benda, P., Jisl, P., Pěchouček, M., Suri, N., and Carvalho, M. (2006). A distributed stand-in agent based algorithm for opportunistic resource

allocation. In *SAC '06: Proceedings of the 2006 ACM symposium on Applied computing*, pages 119–125, New York, NY, USA. ACM.

[Bhat e Miller 2002] Bhat, U. N. and Miller, G. K. (2002). *Elements of Applied Stochastic Processes*. Wiley Series in Probability and Statistics. Wiley-Interscience.

[Boley et al. 2001] Boley, H., Tabet, S., and Wagner, G. (2001). Design rationale for ruleml: A markup language for semantic web rules. pages 381–401. SWWS.

[Cabri et al. 2006] Cabri, G., Ferrari, L., Leonardi, L., Mamei, M., and Zambonelli, F. (2006). Uncoupling coordination: Tuple-based models for mobility. In Bellavista, P. and Conrradi, A., editors, *The Handbook of Mobile Middleware*, chapter 10, pages 229–256. Prentice Hall.

[Chang et al. 2007] Chang, L., Lin, F., and Shi, Z. (2007). A dynamic description logic for semantic web service. In *SKG '07: Proceedings of the Third International Conference on Semantics, Knowledge and Grid*, pages 74–79, Washington, DC, USA. IEEE Computer Society.

[Conti et al. 2004] Conti, M., Gregori, E., and Maselli, G. (2004). Cooperation issues in mobile ad hoc networks. *Distributed Computing Systems Workshops, 2004. Proceedings. 24th International Conference on*, pages 803–808.

[Conti et al. 2005] Conti, M., Gregori, E., and Turi, G. (2005). A cross-layer optimization of gnutella for mobile ad hoc networks. In *MobiHoc '05: Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, pages 343–354, New York, NY, USA. ACM.

[Cornell 2008] Cornell, U. (2008). Jist/swans java in simulation time/scalable wireless ad hoc network simulator. http://jist.ece.cornell.edu/.

[Correa e Barbosa 2009] Correa, R. C. and Barbosa, V. C. (2009). Partially ordered distributed computations on asynchronous point-to-point networks. *Parallel Comput.*, 35(1):12–28.

[Coulouris e Dollimore 2005] Coulouris, G. F. and Dollimore, J. (2005). *Distributed systems: concepts and design.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 4th ed. edition.

[Coutaz et al. 2005] Coutaz, J., Crowley, J. L., Dobson, S., and Garlan, D. (2005). Context is key. *Commun. ACM*, 48(3):49–53.

[Dario Bruneo e Scarpa 2006] Dario Bruneo, A. P. and Scarpa, M. (2006). Mobile middleware: Definition and motivation. In Bellavista, P. and Conrradi, A., editors, *The Handbook of Mobile Middleware*, chapter 7, pages 146–167. Prentice Hall.

[Drabkin et al. 2007] Drabkin, V., Friedman, R., Kliot, G., and Segal, M. (2007). Rapid: Reliable probabilistic dissemination in wireless ad-hoc networks. *Reliable Distributed Systems, 2007. SRDS 2007. 26th IEEE International Symposium on*, pages 13–22.

[Eiben e Schippers 1998] Eiben, A. E. and Schippers, C. A. (1998). On evolutionary exploration and exploitation. *Fundam. Inf.*, 35:35–50.

[Eiben e Smith 2003a] Eiben, A. E. and Smith, J. E. (2003a). *Introduction to Evolutionary Computing*, chapter What is an Evolutionary Algorithm, pages 15–36. SpringerVerlag.

[Eiben e Smith 2003b] Eiben, A. E. and Smith, J. E. (2003b). *Introduction to Evolutionary Computing*, chapter Introduction, pages 1–14. SpringerVerlag.

[Engelstad et al. 2006] Engelstad, P. E., Zheng, Y., Koodli, R., and Perkins, C. E. (2006). Service discovery architectures for on-demand ad hoc networks. *Inter-*

national Journal of Ad Hoc and Sensor Wireless Networks, Old City Publishing (OCP Science), 2(1):27–58.

[Erl 2005] Erl, T. (2005). *Service-Oriented Architecture : Concepts, Technology, and Design*. Prentice Hall PTR.

[Eugster et al. 2003] Eugster, P. T., Felber, P. A., Guerraoui, R., and Kermarrec, A.-M. (2003). The many faces of publish/subscribe. *ACM Comput. Surv.*, 35(2):114–131.

[Feller 1968] Feller, W. (1968). *An Introduction to Probability Theory and Its Applications, Volume 1*. Wiley.

[Foster e Kesselman 2003] Foster, I. and Kesselman, C. (2003). *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[Fugetta et al. 1998] Fugetta, A., Picco, G. P., and Vigna, G. (1998). Understanding code mobility. *IEEE Transactions on Software Engineering*, 24(5):342–361.

[Ganek 2006] Ganek, A. (2006). Overview of autonomic computing: Origins, evolution, direction. In Parashar, M. and Hariri, S., editors, *Autonomic Computing: Concepts, Infrastructure, and Applications*, chapter 1, pages 4–18. CRC.

[Grigoras 2006] Grigoras, D. (2006). Challenges to the design of mobile middleware systems. In *PARELEC '06: Proceedings of the international symposium on Parallel Computing in Electrical Engineering*, pages 14–19, Washington, DC, USA. IEEE Computer Society.

[Groenevelt et al. 2005] Groenevelt, R., Nain, P., and Koole, G. (2005). The message delay in mobile ad hoc networks. *Perform. Eval.*, 62:210–228.

[Gupta e Kumar 1998] Gupta, P. and Kumar, P. (1998). Critical power for asymptotic connectivity in wireless networks. *Stochastic Analysis, Control, Optimizations, and Applications: A Volume in Honor of W.H. Fleming*, pages 547–566.

[Hales e Arteconi 2006] Hales, D. and Arteconi, S. (2006). Slacer: A self-organizing protocol for coordination in peer-to-peer networks. *IEEE Intelligent Systems*, 21(2):29–35.

[Handorean et al. 2003] Handorean, R., Payton, J., Julien, C., and Roman, G.-C. (2003). Coordination middleware supporting rapid deployment of ad hoc mobile systems. In *ICDCSW '03: Proceedings of the 23rd International Conference on Distributed Computing Systems*, page 362, Washington, DC, USA. IEEE Computer Society.

[Herrmann et al. 2007] Herrmann, K., Mühl, G., and Jaeger, M. A. (2007). Meshmdl event spaces - a coordination middleware for self-organizing applications in ad hoc networks. *Pervasive Mob. Comput.*, 3(4):467–487.

[IBM 2005] IBM (2005). An architectural blueprint for autonomic computing.

[Imrich Chlamtac e Liu 2003] Imrich Chlamtac, M. C. and Liu, J. J. N. (2003). Mobile ad hoc networking: imperatives and challenges. *Ad Hoc Networks*, 1(1):13–64.

[Issarny et al. 2007] Issarny, V., Caporuscio, M., and Georgantas, N. (2007). A perspective on the future of middleware-based software engineering. In *FOSE '07: 2007 Future of Software Engineering*, pages 244–258, Washington, DC, USA. IEEE Computer Society.

[Jong et al. 2008] Jong, S. d., Tuyls, K., and Verbeeck, K. (2008). Fairness in multi-agent systems. *Knowl. Eng. Rev.*, 23(2):153–180.

[Kawadia e Kumar 2005] Kawadia, V. and Kumar, P. R. (2005). A cautionary perspective on cross-layer design. *Wireless Communications, IEEE*, 12:3–11.

[Khelil et al. 2007] Khelil, A., Marrón, P. J., Becker, C., and Rothermel, K. (2007). Hypergossiping: A generalized broadcast strategy for mobile ad hoc networks. *Ad Hoc Netw.*, 5(5):531–546.

[Kiess e Mauve 2007] Kiess, W. and Mauve, M. (2007). A survey on real-world implementations of mobile ad-hoc networks. *Ad Hoc Netw.*, 5(3):324–339.

[Liu et al. 2006] Liu, C., Peng, Y., and Chen, J. (2006). Web services description ontology-based service discovery model. In *WI '06: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 633–636, Washington, DC, USA. IEEE Computer Society.

[Liu e Issarny 2004] Liu, J. and Issarny, V. (2004). Qos-aware service location in mobile ad hoc networks. *Mobile Data Management, 2004. Proceedings. 2004 IEEE International Conference on*, pages 224–235.

[Loureiro et al. 2006] Loureiro, E., Bublitz, F., Barbosa, N., Perkusich, A., Almeida, H., and Ferreira, G. (2006). A flexible middleware for service provision over heterogeneous pervasive networks. In *WOWMOM '06: Proceedings of the 2006 International Symposium on on World of Wireless, Mobile and Multimedia Networks*, pages 609–614, Washington, DC, USA. IEEE Computer Society.

[Mamei e Zambonelli 2005] Mamei, M. and Zambonelli, F. (2005). *Field-Based Coordination for Pervasive Multiagent Systems (Springer Series on Agent Technology)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[Martins et al. 2006] Martins, F., Maia, M. E. F., Andrade, R. M. C., dos Santos, A. L., and de Souza, J. N. (2006). A grid computing diagnosis model for tolerating manipulation attacks. *International Transactions On Systems Science And Applications*, 2(135).

[Merwe et al. 2007] Merwe, J. V. D., Dawoud, D., and McDonald, S. (2007). A survey on peer-to-peer key management for mobile ad hoc networks. *ACM Comput. Surv.*, 39(1):1.

[Meyer et al. 2002] Meyer, K. D., Bishop, J. M., and Nasuto, S. J. (2002). Small-world effects in lattice stochastic diffusion search. In *ICANN '02: Proceedings of the International Conference on Artificial Neural Networks*, pages 147–152, London, UK. Springer-Verlag.

[Oreizy et al. 1999] Oreizy, P., Gorlick, M. M., Taylor, R. N., Heimbigner, D., Johnson, G., Medvidovic, N., Quilici, A., Rosenblum, D. S., and Wolf, A. L. (1999). An architecture-based approach to self-adaptive software. *IEEE Intelligent Systems*, 14(3):54–62.

[Oreizy et al. 1998] Oreizy, P., Medvidovic, N., and Taylor, R. N. (1998). Architecture-based runtime software evolution. In *ICSE '98: Proceedings of the 20th international conference on Software engineering*, pages 177–186, Washington, DC, USA. IEEE Computer Society.

[Popescu-Zeletin et al. 2006] Popescu-Zeletin, R., Arbanowski, S., Steglich, S., and Radusch, I. (2006). Mobile computing. In Bellavista, P. and Conrradi, A., editors, *The Handbook of Mobile Middleware*, chapter 2, pages 28–49. Prentice Hall.

[Ravi et al. 2004] Ravi, N., Borcea, C., Kang, P., and Iftode, L. (2004). Portable smart messages for ubiquitous java-enabled devices. *Mobile and Ubiquitous Systems: Networking and Services, 2004. MOBIQUITOUS 2004. The First Annual International Conference on*, pages 412–421.

[Rezende et al. 2008] Rezende, C., Rocha, B. P. S., and Loureiro, A. A. F. (2008). Publish/subscribe architecture for mobile ad hoc networks. In *SAC '08: Proceedings of the 2008 ACM symposium on applied computing.*

[Riva et al. 2007] Riva, O., Nadeem, T., Borcea, C., and Iftode, L. (2007). Context-aware migratory services in ad hoc networks. *IEEE Transactions on Mobile Computing*, 6(12):1313–1328.

[Schade et al. 2004] Schade, S., Sahlmann, A., Lutz, M., Probst, F., and Kuhn, W. (2004). Comparing approaches for semantic service description and matchmaking. In *On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE*, chapter 15, pages 1062–1079. Springer Berlin / Heidelberg.

[Schoch et al. 2008] Schoch, E., Feiri, M., Kargl, F., and Weber, M. (2008). Simulation of ad hoc networks: ns-2 compared to jist/swans. In *Simutools '08: Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, pages 1–8, ICST, Brussels, Belgium, Belgium. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[Seyed Amin Hosseini e Seno 2007] Seyed Amin Hosseini Seno, Rahmat Budiarto, T.-C. W. (2007). Survey and new approach in service discovery and advertisement for mobile ad hoc networks. *IJCSNS International Journal of Computer Science and Network Security*, 7(2):275–284.

[Sharma et al. 2007] Sharma, G., Mazumdar, R., and Shroff, N. B. (2007). Delay and capacity trade-offs in mobile ad hoc networks: a global perspective. *IEEE/ACM Trans. Netw.*, 15(5):981–992.

[Sterritt et al. 2005] Sterritt, R., Parashar, M., Tianfield, H., and Unland, R. (2005). A concise introduction to autonomic computing. *Advanced Engineering Informatics*, 19(3):181–187.

[Subramanian e Chung 2001] Subramanian, N. and Chung, L. (2001). Software architecture adaptability: an nfr approach. In *IWPSE '01: Proceedings of the 4th*

International Workshop on Principles of Software Evolution, pages 52–61, New York, NY, USA. ACM.

[Tesauro et al. 2004] Tesauro, G., Chess, D. M., Walsh, W. E., Das, R., Segal, A., Whalley, I., Kephart, J. O., and White, S. R. (2004). A multi-agent systems approach to autonomic computing. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 464–471, Washington, DC, USA. IEEE Computer Society.

[Tracy Camp 2002] Tracy Camp, Jeff Boleng, V. D. (2002). A survey of mobility models for ad hoc network research. *Wireless Communications and Mobile Computing*, 2(5):483–502.

[W3C 2008] W3C (2008). Owl web ontology language overview. http://www.w3.org/TR/owl-features/. Acessado em 5 de Fevereiro de 2009.

[Yan et al. 2007] Yan, J., Kowalczyk, R., Lin, J., Chhetri, M. B., Goh, S. K., and Zhang, J. (2007). Autonomous service level agreement negotiation for service composition provision. *Future Gener. Comput. Syst.*, 23(6):748–759.

[Yu 2007] Yu, W. (2007). Game theoretic analysis of cooperation stimulation and security in autonomous mobile ad hoc networks. *IEEE Transactions on Mobile Computing*, 6(5):507–521. Fellow-Liu,, K. J. Ray.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo