UNIVERSIDADE FEDERAL FLUMINENSE

GABRIEL ALMEIDA DE OLIVEIRA

Uma implementação de mobilidade de dispositivos em Openflow

NITERÓI

2016

UNIVERSIDADE FEDERAL FLUMINENSE

GABRIEL ALMEIDA DE OLIVEIRA

Uma implementação de mobilidade de dispositivos em Openflow

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Redes de Computadores e Sistemas Distribuídos e Paralelos

Orientador:
Antonio Augusto de Aragão Rocha

NITERÓI

2016

Gabriel Almeida de Oliveira

Uma implementação de mobilidade de dispositivos em *Openflow*

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Redes e Sistemas Distribuídos e Paralelos

Aprovada em Abril de 2016.

BANCA EXAMINADORA

Prof. Antonio Augusto de Aragão Rocha - Orientador, UFF

Prof. Célio Vinicius Neves de Albuquerque, UFF

Prof. José Ferreira de Rezende, UFRJ

Niterói

2016

Dedicatória(s): Dedico este trabalho à minha mãe, Edivania Santos, cuja presença consigo sentir mesmo longe, sempre me apoiando onde quer que esteja.

Agradecimentos

Agradeço à minha mãe. Meu pilar, exemplo de vida, e minha maior motivação para seguir em frente, mesmo que tudo pareça perdido.

Ao meu orientador, Antonio Augusto, pela orientação, paciência e indicações.

Aos professores do Instituto de Computação, que me forneceram um ensino de alta qualidade. Em especial os professores que encontrei na minha pós graduação: Igor Moraes, Pedro Velloso, Fábio Protti, Célio Albuquerque, Lúcia Drummond e Débora Saade.

Aos amigos da UFF que me convenceram e continuaram comigo durante o mestrado, em especial Raphael Evangelista, Cinthya França e Hugo Barbalho.

Aos meus amigos da época da escola, que seguem comigo após mais de 20 anos juntos.

À impressora da Pós Graduação, por ter fornecido muito material de leitura nas longas voltas para casa.

Resumo

Atualmente, as pessoas estão sempre conectadas à Internet, seja através de um computador portátil com Modem 3G, um *smartphone* ou até dispositivos vestíveis, como relógios. Tais usuários estão em constante movimento, locomovendo-se entre trabalho e casa, e seus dispositivos lhes acompanhando e ingressando em redes *Wi-Fi* conhecidas ou voltando para suas redes 3G/4G de origem. Essa movimentação faz com que os dispositivos fiquem em constante utilização para que tais transições sejam feitas da forma mais transparente para o usuário, gastando mais recursos e bateria.

O problema está na forma que a Internet atual foi criada, já que ela não atende mais as necessidades atuais dos usuários e não permite evolução ou modificação. Uma rede planejada inicialmente para comunicação entre grupos de pesquisa, cada grupo com máquinas estacionárias e bem definidas, agora abriga os mais variados tipos de dispositivos, que podem ser estacionários ou transitar entre redes em alta velocidade. Como a Internet não suporta tais transições de rede, as aplicações precisam prever e lidar com a manutenção da comunicação.

Por este motivo, entre outros, é muito estudada a criação de uma "Internet do Futuro", modular e passível de mudança. Pesquisadores estudam a criação de uma Internet mais móvel, levando o custo de reconexão para a rede ao invés de deixá-la nos dispositivos do usuário. Algumas propostas envolvem recriar a rede, enquanto outras tentam reaproveitá-la de alguma forma, mantendo-a funcionando enquanto a nova arquitetura é implantada.

Neste trabalho, é proposta uma arquitetura de Internet, que leva em consideração a mobilidade de dispositivos entre redes. Serão discutidas formas de se resolver o problema da mobilidade e apresentados alguns trabalhos já existentes na literatura. Baseados nas propostas apresentadas, será formulada uma arquitetura. Por fim, será criada e simulada uma rede com a arquitetura proposta implantada, onde os tráfegos de algumas operações comuns realizadas por usuários serão reproduzidos.

Palavras-chave: Internet do Futuro, Mobilidade, Dispositivos móveis, Simulação, Mininet, OpenFlow.

Abstract

People are nowadays always connected to the Internet, be it by a notebook with a 3G dongle, a smartphone or even wearable devices, such as smartwatches. Those users are in constant movement, going from home to work, and their devices switching between Wi-Fi and 3G/4G networks. This mobility causes devices to be in constant use, so the mobility impact to the user is minimal, but draining more resources from the device.

The problem is the way the Internet was created, as it can't provide the functionalities today users require, as well as not allowing modifications to its core. A network that was going to be used only as a bridge between research groups having stationary devices, is now used by various kinds of devices, that can be stationary or roam between networks at a high speed. As the network is not designed to support those network transitions, developers and applications need to deal with the communication drops.

Because of those reasons, among others, the creation of an "Internet of the Future", modular and receptive to upgrading, is highly debated. Researchers study the creation of a more mobile Internet, taking the reconnection cost to the network instead of letting devices handle it. Some ideas involve recreating the network from a clean slate, while others try to use some parts of the current architecture, keeping itself online while the new architecture is deployed.

This work aims at proposing an Internet of the Future architecture that takes into account the users' and devices' mobility across multiple networks. Forms of solving the mobility problem will be discussed and existing works will be shown. Some architectures will be shown in more detail. Also, the proposed network will be deployed in a virtual network and mobility tests will be made on this network.

Keywords: Internet of the Future, Mobility, Mobile devices, Simulation, Mininet, OpenFlow.

Lista de Figuras

1.1 Número global de usuários de dispositivos móveis e estáticos (em milhões). Fonte: [66] 2

2.1 Utilizando o contexto para entregar mensagens para todos os membros de uma certa localização. 9

2.2 Redes Orientadas a Conteúdo - armazenamento na rede 11

2.3 *OpenFlow*: Conexão entre a infraestrutura e o plano de controle 13

3.1 Fluxo de pacotes no IP Móvel. 18

3.2 Barreira ao prover mobilidade em CCNs - fluxos bidirecionais 19

3.3 Exemplo de *Multihoming* na arquitetura *MobilityFirst* – encaminhamento do pacote 25

3.4 *MobilityFirst* – exemplo de mobilidade 26

4.1 Exemplo de mobilidade em redes SDN padrão 29

4.2 Proposta: Modelo Inicial 31

4.3 Proposta: Caso de mobilidade funcional 33

4.4 Mapa da rede (a) antes da mobilidade; e (b) depois da mobilidade 35

5.1 Fluxograma do módulo desenvolvido 38

5.2 Cenário de teste 1: Dispositivo no mesmo ponto de conexão mas com endereço diferente ao se reconectar 41

5.3 Cenário de teste 2: Topologia real 41

6.1 Cenário 1: Ping sem atraso na reconexão. 45

6.2 Cenário 1: Ping – Detalhe no Momento de Reconexão. 45

6.3 Cenário 1: Ping - Pacotes perdidos em 100 rodadas. 46

6.4 Cenário 2: Ping sem atraso na reconexão. 47

6.5 Cenário 2: Ping – Detalhe no Momento de Reconexão. 47

6.6 Cenário 1: Download de arquivo com mobilidade. 48

6.7 Cenário 2: Download de Arquivo. 48

6.8 Cenário 1: Jitter Médio percebido pelo nó móvel. 50

6.9 Cenário 1: Jitter Médio percebido pelo nó estático. 50

6.10 Cenário 2: Jitter médio percebido pelo nó móvel 51

6.11 Cenário 1: Taxa de *Download* - Streaming de vídeo. 51

6.12 Cenário 1: Detalhe no momento de reconexão 52

6.13 Cenário 2: Taxa de obtenção - Streaming de vídeo. 52

6.14 Cenário 2: Detalhe no momento de reconexão 53

Lista de Tabelas

1.1 Tempo médio (em horas) diário gasto por usuário adulto com mídias digitais nos EUA. Fonte: [66] 2

5.1 Ambiente de *hardware* e *software* utilizado nos experimentos. 40

5.2 Especificações do padrão 802.11n 42

Lista de Abreviaturas e Siglas

ICMP	:	Internet Control Message Protocol;
IP	:	Internet Protocol;
UDP	:	User Datagram Protocol;
TCP	:	Transmission Control Protocol;
ARP	:	Address Resolution Protocol;
MAC	:	Media Access Control;
RTT	:	Round-Trip Time;
PCM	:	Pulse-code modulation;
RTP	:	Real-time Transfer Protocol;
ISP	:	Internet Service Provider;
SDN	:	Software Defined Network;
CDN	:	Content Delivery Network;
CCN	:	Content Centric Network;
DNS	:	Domain Name System;
QoE	:	Quality of Experience;
GUID	:	Globally Unique IDentifiers;
NA	:	Network Address;
NCS	:	Name Certification Service;
GNRS	:	Global Name Resolution Service;
PIT	:	Pending Interest Table;
FIB	:	Forwarding Information Base;
API	:	Application Programming Interface;

Sumário

1 Introdução **1**

1.1 Motivação 3

1.2 Objetivos do Trabalho 4

1.3 Contribuições 4

1.4 Estrutura do Trabalho 5

2 Definições e Conceitos Gerais Sobre Arquiteturas de Internet do Futuro **6**

2.1 Definições básicas 6

Dispositivo móvel 6

Multicast e *Anycast* 7

Mobilidade de dispositivos 7

Mobilidade de usuários 7

Multihoming 7

"Internet Móvel" 8

Mobilidade baseada em *Host* ou rede 8

2.2 Novos Paradigmas e Terminologias 8

2.2.1 Contexto e sensibilidade ao contexto 8

2.2.2 Conteúdo e Redes Orientadas à Conteúdo 9

2.2.3 Redes Definidas por *Software* e Fluxos 10

2.3 Características Desejáveis 12

Alcançabilidade 13

Continuidade 13

Heterogeneidade 13
Conectividade 14
Escalabilidade 14
Adaptabilidade 15
Auto-gerenciabilidade 15
Armazenamento e processamento no núcleo da rede 15

3 Trabalhos Relacionados **17**

3.1 Auspice 21
3.2 MobilityFirst 23

4 Proposta **28**

4.1 Análise da proposta 32
4.2 Considerações 34

5 Implementação e Experimentos **37**

5.1 Implementação 37
5.1.1 Recebimento de pacotes 38
5.1.2 Mudança em interfaces 39
5.1.3 Detecção de novos nós 39
5.2 Experimentos 40

6 Resultados e Análise **44**

6.1 Tráfego ICMP 44
6.2 *Download* de arquivo via HTTP 46
6.3 Chamada de Áudio 49
6.4 *Streaming* de Vídeo 50
6.5 Análise dos Resultados 52

7 Conclusão e Trabalhos Futuros **54**

Referências **55**

Capítulo 1

Introdução

A Internet foi desenvolvida, inicialmente, para ser utilizada somente por cientistas e pesquisadores, indivíduos conhecidos e confiáveis, contava com poucos nós, estacionários e bem conhecidos, e pretendia apenas unir universidades e grupos de pesquisa [40, 37]. O objetivo destes cientistas era compartilhar recursos, como arquivos, bancos de dados, além de permitir acesso remoto a outros computadores e impressoras, recursos escassos na época [32]. Quatro décadas depois, é estimado em [11] que a rede possua quase 3 bilhões de usuários, e cada usuário pode acessá-la de vários dispositivos, com especificações e características distintas, podendo ser estacionários ou móveis. A Internet também deixou de ser usada unicamente para acesso remoto e compartilhamento de recursos, servindo de meio para várias aplicações com objetivos e serviços variados, como reprodução de vídeo e músicas ou jogos *on-line* [43]. Os objetivos das aplicações que utilizam a Internet se expandiram, mas a arquitetura permanece a mesma.

Atualmente, a Internet utiliza os endereços IP, que combina identidade e localidade [69, 37], para localizar dispositivos. O aumento cada vez mais acentuado de usuários e a utilização de dispositivos móveis desafia este princípio [17], que resulta em baixa mobilidade e capacidade de multiconexão, o chamado *multihoming* (o termo será melhor explicado na Subseção 2.1). O aumento das tabelas de roteamento, devido ao aumento de usuários, e a semântica múltipla do IP são considerados problemas chave da arquitetura atual da Internet [47]. A resolução de nomes atual também conta com a suposição de estações estacionárias, já que atrela um nome a um endereço IP. Porém, atualmente já são vendidos mais *smartphones* do que computadores pessoais [9], existindo uma migração para os dispositivos móveis [25]. A Tabela 1.1 evidencia a maior utilização de dispositivos móveis ao mostrar o tempo médio diário gasto por adulto consumindo mídias digitais nos Estados Unidos, de 2008 a 2015, segundo [66].

Tabela 1.1: Tempo médio (em horas) diário gasto por usuário adulto com mídias digitais nos EUA. Fonte: [66]

Ano	PCs	Smartphones	Outros Dispositivos
2008	2.2	0.3	0.2
2009	2.3	0.3	0.3
2010	2.4	0.4	0.4
2011	2.6	0.8	0.3
2012	2.5	1.6	0.3
2013	2.3	2.3	0.3
2014	2.4	2.6	0.3
2015	2.4	2.8	0.4

Além disso, o uso de dispositivos móveis para acesso à Internet também já é maior que o de estações estáticas [73], com aplicativos móveis gerando quase 10 vezes mais consumo de mídia em dispositivos móveis do que navegadores comuns [41]. Tal fato é mostrado no gráfico da Figura 1.1, que mostra o número global de usuários da Internet. Pode-se perceber que não só o número de dispositivos móveis já é maior, como também a curva de crescimento de tais dispositivos é mais acentuada que a de estações estáticas.

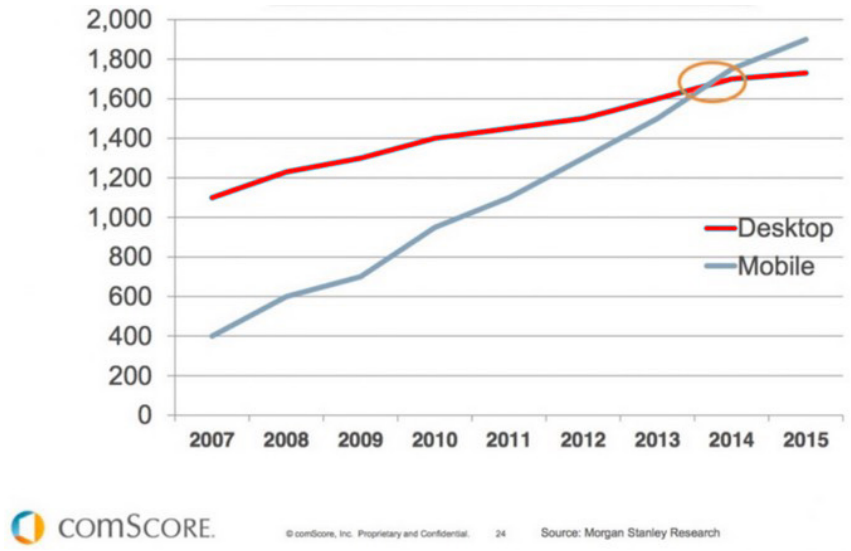

Figura 1.1: Número global de usuários de dispositivos móveis e estáticos (em milhões). Fonte: [66]

Outro agravante está no fato da arquitetura atual ser difícil de se modificar, devido à premissa inicial da Internet de deixar a complexidade na borda da rede, mantendo seu núcleo o mais simples possível, e intacto. Novas soluções são obrigadas a serem implementadas através de "remendos" em cima da arquitetura. Também são descobertas falhas nos protocolos que não podem ser resolvidas devido à falta de segurança projetada.

Apesar disso, a Internet ainda opera da mesma maneira de quando era dominada por máquinas estacionárias, sem levar em consideração que usuários, conteúdo e até mesmo redes (como redes intraveiculares) são móveis atualmente – os dispositivos móveis são a regra, não mais a exceção.

A vontade de se quebrar tal premissa inicial de não se alterar o núcleo da rede incentivou a pesquisa de novas arquiteturas de Internet. É muito estudado por pesquisadores a criação de uma "nova Internet", isto é, refazer a estrutura da Internet desde o princípio, levando em consideração as novas necessidades e requisitos a serem implementados na "nova Internet", assim como permitir uma fácil modificação da arquitetura, caso tais necessidades mudem com o tempo. Uma grande vertente destas pesquisas visa o projeto de arquiteturas que levam em consideração a mobilidade dos usuários.

Uma proposta de arquitetura de Internet do Futuro deve ser muito bem estudada, pois é uma arquitetura que será utilizada por milhões de pessoas, todos os dias. Por isso, é necessário que a arquitetura seja escalável, porém o menos complexa possível. Uma arquitetura deste tipo deve ser fácil de se adaptar para atender novas demandas, e ao mesmo tempo resiliente contra ataques. Estes são alguns requisitos, entre outros, que mostram a complexidade de se criar uma solução simples e efetiva.

1.1 Motivação

A mobilidade de usuários, atualmente, é um dos maiores desafios para a Internet do Futuro. Como dito anteriormente, dispositivos móveis são a maioria dos dispositivos conectados atualmente, sendo necessário garantir a conexão dos dispositivos enquanto eles se movem em alta velocidade entre redes. Os recursos destes dispositivos, como bateria e plano de dados, são críticos, porém é necessário tratar a mobilidade no próprio dispositivo, gastando tais recursos. Tirando a complexidade dos dispositivos e deixando-as no núcleo da rede, é possível fazer operações básicas dentro da infraestrutura, como criptografia e armazenamento, poupando recursos dos dispositivos.

Existem várias propostas, como [6, 62, 71, 65, 12, 19, 20, 21, 44, 72, 4, 67, 56, 57, 58], que tentam abordar o problema da mobilidade de maneiras diferentes. Porém todas possuem algum ponto fraco – viabilidade ou escalabilidade, entre outros. Portanto propor uma arquitetura é uma tarefa desafiadora, e a implementação dela mais ainda, devido às características imutáveis da arquitetura atual.

1.2 Objetivos do Trabalho

Esta dissertação tem como objetivo propor uma nova arquitetura de Internet do Futuro, sobreposta à rede atual, capaz de prover mobilidade aos usuários. Para tal, serão definidas características desejáveis em uma arquitetura que visa a mobilidade, e mostrados alguns modelos de arquiteturas já propostos, com suas características. Tomando como base as características e os trabalhos levantados, será proposta a arquitetura. A arquitetura será implementada e testada. Dada a dificuldade de se testar modelos e infraestruturas novas na atual Internet, os testes serão feitos em ambientes virtualizados. Por fim, será avaliado o impacto na rede causado pela arquitetura, utilizando as medidas mais percebidas pelos usuários, como vazão, perda de pacotes e latência [36].

1.3 Contribuições

As contribuições deste trabalho podem ser destacadas como as seguintes

- Análise do estado da arte da abordagem à mobilidade nos trabalhos relacionados à Internet do Futuro. Muitos trabalhos tentam propor uma nova arquitetura, porém nem todos levam em consideração a alta mobilidade de dispositivos. Aceitar uma nova arquitetura que considera estações estáticas não irá trazer benefícios para os usuários, que continuarão se movimentando entre redes sem suporte da mesma;
- Levantamento das características desejáveis numa arquitetura que leve em consideração a mobilidade. É possível utilizar estas características como base na validação de futuras propostas, possibilitando a resolução de problemas antes deles surgirem e analisar a viabilidade teórica;
- Proposta de uma arquitetura nova, que pode ser sobreposta à Internet atual, alterando apenas a borda da rede, ou substituí-la ao levar complexidade para o núcleo da rede, utilizando conceitos e ideias obtidas através da análise dos trabalhos relacionados;
- Modelagem de tipos de tráfego real e comumente utilizados pelos usuários em ambientes de testes;
- Modelagem de um ambiente de testes capaz de simular com precisão os tráfegos desejados;

- Avaliação de viabilidade da proposta feita, através do ambiente de testes elaborado e dos tráfegos modelados.

1.4 Estrutura do Trabalho

O restante deste trabalho está organizado da seguinte maneira:

No Capítulo 2, conceitos sobre a mobilidade de usuários e conteúdo na rede serão esclarecidos, abordando e definindo seus principais elementos e características. Também serão explicados novos paradigmas e mostradas características desejáveis em uma arquitetura de Internet do Futuro.

O Capítulo 3 mostra alguns trabalhos relacionados à mobilidade de dispositivos nas redes de computadores. Tais trabalhos servirão como base para a elaboração de uma nova proposta.

No Capítulo 4, uma nova arquitetura de Internet do Futuro será proposta, e seus detalhes serão explicados. A proposta será analisada em relação às características desejáveis e algumas considerações serão feitas.

No Capítulo 5, serão explicadas mais profundamente as etapas do desenvolvimento do modelo-teste, mostrando os detalhes de implementação. Também será explicada a modelagem dos ambientes de testes e dos experimentos realizados.

No Capítulo 6, estarão expostos os resultados obtidos nos experimentos, assim como a discussão acerca de tais resultados e conclusões que podem ser obtidas.

Por fim, o Capítulo 7 finaliza o trabalho, com as considerações finais e possíveis temas de trabalhos futuros.

Capítulo 2

Definições e Conceitos Gerais Sobre Arquiteturas de Internet do Futuro

A Internet segue o princípio básico de manter a complexidade nas bordas, com um núcleo mais simples que apenas armazena e encaminha (*store-and-forward*) pacotes [37]. Algumas propostas de arquiteturas de Internet do Futuro pretendem modificar este princípio, efetivamente construindo uma nova internet do princípio, como [62, 12, 4, 67]. Outras defendem que a dificuldade de modificar o núcleo da rede não é viável, defendendo a manutenção da rede atual com a implantação de outras redes adjacentes ou uma maior inteligência nas bordas da rede, como [56, 72]. Porém, todas concordam que a arquitetura atual limita o surgimento de novas tecnologias, e pretendem resolver tal problema.

2.1 Definições básicas

Nesta seção, apresentaremos alguns termos utilizados ao longo deste trabalho. Tais termos facilitam a compreensão dos trabalhos relacionados à Internet do Futuro.

Dispositivo móvel

Dispositivos móveis são aqueles capazes de transitar entre redes enquanto possuem conexões ativas com outros dispositivos. Exemplos de dispositivos móveis são celulares inteligentes (*Smartphones*), dispositivos vestíveis (como relógios e braçadeiras inteligentes), *Tablets* e até mesmo *Notebooks*, que podem possuir conexões *Wi-Fi* e permitir a conexão por *modems* 3G. Os termos "dispositivo", "nó", e "*host*" serão utilizados concomitantemente ao longo deste trabalho.

Multicast e *Anycast*

Multicast é quando uma mensagem é enviada para mais de um destinatário. Por exemplo, se mais de um usuário de uma mesma rede requisita um mesmo vídeo por *Streaming*, o dono do conteúdo consumido irá enviar os mesmos dados repetidas vezes para a mesma rede. O *Multicast* permite o envio dos dados apenas uma vez, e a rede pode duplicar os pacotes internamente e enviá-las para os receptores, causando uma menor carga na transmissão. Já o *Anycast* é quando o interesse é enviar os dados para o membro mais próximo de um determinado grupo ou rede. Este tipo de roteamento é particularmente útil em serviços descentralizados, como o DNS, já que a mesma informação está em vários lugares. O *Anycast* permite o envio da requisição para o meio, o nó (por exemplo, o servidor DNS) mais próximo irá receber a requisição e enviar a resposta.

Mobilidade de dispositivos

A mobilidade de dispositivos é feita quando um dispositivo móvel se move entre duas redes, podendo ser qualquer combinação entre redes *Wi-Fi* e redes de dados celular (como 3G/4G). Chamamos o momento de desconexão do dispositivo de início da mobilidade, e o fim da mobilidade ocorre quando o dispositivo está na sua nova rede, já com as conexões re-estabelecidas.

Mobilidade de usuários

A mobilidade de usuários é semelhante à mobilidade de dispositivos, porém leva em consideração que um usuário possui mais de um dispositivo. É possível se utilizar do *Multicast*, aliado às técnicas de localização dos usuários, para enviar informações para todos seus dispositivos simultaneamente, ou utilizar o *Anycast* para entregar mensagens a apenas um de seus dispositivos, sabendo a localização deste.

Multihoming

Multihoming é quando um dispositivo está em duas redes ao mesmo tempo. Como exemplo, alguns dispositivos móveis permitem a utilização da rede de dados celular como complemento para a rede *Wi-Fi*, caso a última esteja com sinal fraco. Nestes casos, é possível alcançar o dispositivo por qualquer uma das redes às quais ele pertence, através do *Multicast*.

"Internet Móvel"

Chamamos de "Internet Móvel" a arquitetura de Internet do Futuro que leva em consideração a mobilidade de usuários e dispositivos. Ou seja, uma arquitetura que esteja preparada para a alta taxa de mobilidade dos dias de hoje.

Mobilidade baseada em *Host* ou rede

Chamamos de Mobilidade baseada em *host* qualquer tipo de proposta que tenta prover mobilidade alterando o funcionamento ou instalando algo nos dispositivos dos usuários. Um exemplo seria uma aplicação que detecta mobilidades e avisa a um servidor da mudança, reparando a conexão. Já mobilidade baseada na rede é a proposta que prevê a mudança de rede, sem modificação nos dispositivos dos usuários, como redes SDN e propostas que visam modificar o núcleo da rede.

2.2 Novos Paradigmas e Terminologias

Existem três grandes paradigmas criados com a possibilidade de uma Internet do Futuro. Aqui iremos introduzir seus conceitos.

2.2.1 Contexto e sensibilidade ao contexto

Contexto é definido em [1] como qualquer informação que possa caracterizar a informação de uma entidade (pessoa, lugar, ou objeto) considerada relevante para a interação entre um usuário e uma aplicação. Um exemplo seria uma aplicação que utilize pesos e medidas. Dependendo de onde esta aplicação é utilizada, as unidades podem variar. Portanto, a localização do usuário ajuda a aplicação a formar um contexto, definindo configurações específicas, como a utilização do sistema métrico ou imperial.

Uma aplicação é sensível ao contexto se ela utiliza contexto para fornecer serviços ou informações relevantes ao usuário. Exemplos atuais de aplicações sensíveis ao contexto são os programas de chamadas de táxi, que localizam táxis perto do usuário, informam o tempo de chegada do táxi e permitem o acompanhamento em tempo real.

Uma rede é sensível ao contexto quando ela o utiliza para efetuar o encaminhamento. No exemplo anterior, a aplicação de táxis é responsável por detectar os taxistas próximos ao usuário. Caso a rede fosse sensível ao contexto, a aplicação poderia simplesmente enviar

um pacote de requisição cujo destinatário é "todos os taxistas próximos ao usuário". A rede, conhecendo a localização dos taxistas e dos usuários, multiplicaria os pacotes e os entregaria para os taxistas corretamente.

Utilizando o exemplo ilustrado pela Figura 2.1, é possível definir o contexto "Táxis em Niterói" (simbolizado pelo círculo azul). Todos os táxis que estão dentro da área receberão mensagens destinadas à esse contexto, e a rede se adapta automaticamente para entregar mensagens à membros que passam a atender ao contexto definido, e deixa de entregar mensagens que não atendem mais à este.

Figura 2.1: Utilizando o contexto para entregar mensagens para todos os membros de uma certa localização.

2.2.2 Conteúdo e Redes Orientadas à Conteúdo

Conteúdo pode ser páginas *Web*, arquivos de áudio ou vídeo, documentos de texto, ou metadados responsáveis por localizar, gerenciar e interpretar os próprios conteúdos [59]. A Internet é atualmente considerada como um sistema multimídia que entrega conteúdos, porém ela não é otimizada para tal. Existem algumas técnicas atualmente utilizadas para entregar conteúdos de forma eficiente, como o BitTorrent [13] e Redes de Distribuição de Conteúdo (CDNs) [53, 54, 61], mas são soluções implementadas na camada de aplicação.

As Redes Orientadas à Conteúdo(ROCs ou CCNs) [15, 16] prometem o acesso ao conteúdo independente de onde ele esteja localizado. Para isso, elas utilizam conceitos como armazenamento de dados no núcleo da rede e roteamento baseado no nome do conteúdo [32, 34], ao invés de baseado em estações (como atualmente). Esta forma de roteamento causa a obtenção do conteúdo sem a necessidade da localização deste ser conhecida.

As CCNs trabalham com o conceito de produtor e consumidor. Um consumidor envia requisições (pacotes de interesse) de um conteúdo desejado. Caso não haja uma cópia do conteúdo na rede, o pacote de interesse é recebido pelo produtor, enviando o conteúdo através de pacotes de dados. Além disso, roteadores armazenam os interesses que ainda não foram atendidos (na PIT - *Pending Interest Table*), além das interfaces de entrada do interesse e a interface pela qual ele foi encaminhado. A função desta tabela é diminuir o tráfego na rede, pois caso o roteador receba um pacote de interesse para um conteúdo já existente na PIT, é possível apenas catalogar o interesse atual e aguardar, já que o conteúdo já foi requisitado por algum outro nó. Quando o conteúdo for recebido pelo roteador, uma cópia é armazenada e o roteador irá enviar o conteúdo para todos os nós que o requisitaram.

Além da PIT, os roteadores utilizam outra tabela, a FIB (*Forwarding Information Base*). Esta tabela associa nomes a interfaces, permitindo o roteamento correto de pacotes. A única diferença desta tabela para uma tabela de roteamento IP padrão é que a FIB pode associar um nome às várias interfaces, devido à CCN permitir mais de uma fonte de dados.

A Figura 2.2, adaptada de [31], mostra um exemplo de obtenção de um conteúdo. Em (a), o cliente 1 busca determinado conteúdo. Os roteadores no caminho entre o cliente e o conteúdo, ao receberem este para encaminhamento, guardam uma cópia local deles (b). Caso o cliente 2 queira tal conteúdo (c), o roteador **D**, já tendo indexado tal conteúdo, é capaz de entregá-lo imediatamente, evitando o envio de um pedido até o produtor (d). De maneira geral, a requisição dos consumidores não necessariamente precisa chegar até o produtor, já que roteadores no caminho possuem cópias do conteúdo e podem entregá-lo.

2.2.3 Redes Definidas por *Software* e Fluxos

As Redes Definidas por *Software* (*Software Defined Networks* - SDNs) defendem a utilização de roteadores inteligentes, que se dividem em plano de dados e plano de controle. O plano de dados é de alta velocidade, e seu propósito é rotear os pacotes, enquanto o plano de controle inclui funções para configurar roteamento e encaminhamento, qualidade do serviço, entre outros [42].

Com a separação destes planos, é possível utilizar *software* para gerir o plano de controle, permitindo a implantação de serviços mais rapidamente, visto que a atualização de um *Software* é bem mais rápida e menos impactante do que a troca de uma parte do *Hardware* da rede. Além disso, podem ser disponibilizadas APIs que permitem o controle

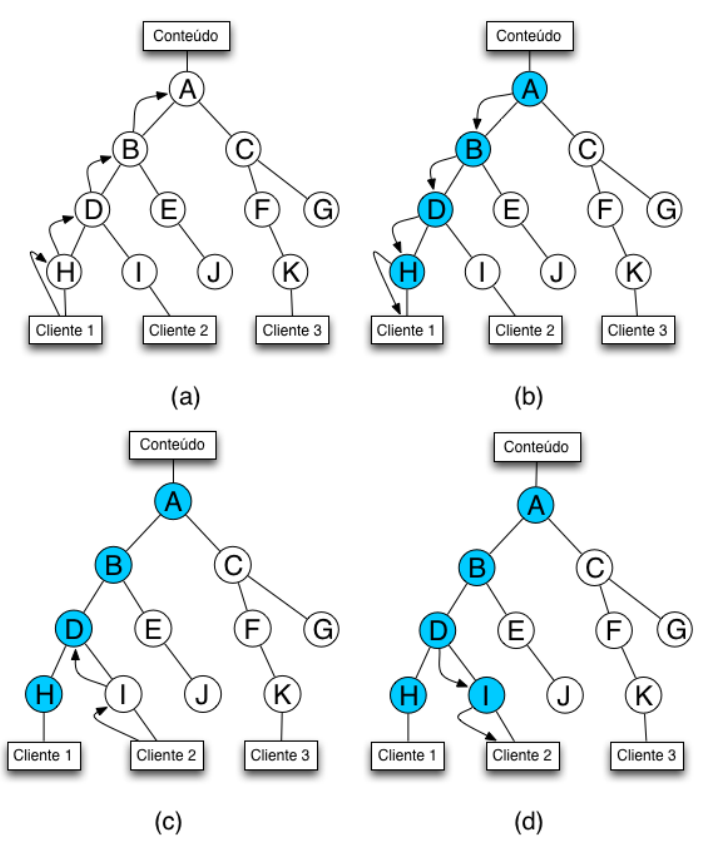

Figura 2.2: Redes Orientadas a Conteúdo - armazenamento na rede

e operação direta da rede, fazendo com que aplicações possam controlar a rede, tornando-a mais inteligente.

Antes de seguir em frente, é necessário introduzir o conceito de fluxos. Um fluxo pode ser uma conexão específica de uma aplicação ou o tráfego específico de um usuário ou contexto. Exemplos de fluxos são "pacotes da aplicação X", "pacotes enviados pelo usuário Y", ou "pacotes destinados ao país/cidade/etc Z". É possível definir fluxos bem específicos e tratá-los de acordo com as regras definidas no plano de controle.

Porém, fazer tal separação e manter a responsabilidade de gerir o plano de controle nos roteadores não irá trazer nenhum benefício. Com base nessa observação, foi sugerido em [45] que sejam utilizadas máquinas separadas para gerir o plano de controle. Estas máquinas receberam o nome de controladores. O controlador é responsável por fazer as decisões de roteamento e encaminhamento, através de regras que são instaladas numa tabela em cada roteador gerido por este controlador, a tabela de fluxos (*flow table*). Quando um roteador não sabe como encaminhar um pacote, uma cópia deste é enviada ao controlador, que irá executar os algoritmos de encaminhamento e instalará a regra a ser seguida no roteador. A partir daí, o roteador não precisará ir ao controlador para encaminhar

pacotes deste fluxo. Utilizando os exemplos de fluxos mencionados anteriormente, regras de fluxos válidas podem ser "enviar pacotes da aplicação X destinados ao usuário Y pela porta 5 do roteador", ou "descartar pacotes recebidos cujo emissor está no país K". A utilização inteligente de regras de fluxo pode ser uma grande aliada em arquiteturas que visam mobilidade, permitindo que pacotes sejam alterados conforme os usuários se movem entre redes. Tal princípio é a base da proposta apresentada neste trabalho, sendo mostrado com maior profundidade sua utilização no Capítulo 4.

O primeiro e mais utilizado padrão dentro do escopo das SDNs é o *OpenFlow* [46], que faz a ligação entre o plano de controle e a infraestrutura da rede. O *OpenFlow* permite acesso direto ao plano de dados da rede, viabilizando assim a existência das SDNs. A Figura 2.3 mostra a abstração fornecida pelo *OpenFlow* para os desenvolvedores e administradores de rede. O protocolo funciona como uma ponte entre a infraestrutura da rede (camada física) e o plano de controle, permitindo que desenvolvedores e administradores alterem configurações dos roteadores de maneira transparente, e fornece APIs que permitem a comunicação entre aplicações e o plano de controle.

Para que esta abstração ocorra, no entanto, é necessário que os roteadores atendam ao padrão *OpenFlow* [51]. Cabe então citar que o padrão foi bem aceito pelas empresas, que estão distribuindo cada vez mais roteadores atendendo os requisitos do *OpenFlow* [5]. Prova deste fato está na fundação da *Open Networking Foundation* (ONF) [52], em 2011. A ONF é uma fundação sem fins lucrativos, fundada por Deutsche Telekom, Facebook, Google, Microsoft e Yahoo, que visa promover a utilização de SDNs e do protocolo *OpenFlow*. Atualmente, a empresa possui mais de 120 membros, entre eles empresas como Cisco, Citrix, Dell, Extreme Networks, HP, Huawei e Oracle, mostrando a importância que a indústria tem dado para a adoção do protocolo.

Além dos roteadores precisarem atender o padrão *OpenFlow*, é necessário utilizar um controlador capaz de entender o protocolo *OpenFlow*. Existem várias alternativas [8, 24, 22, 50, 23, 28], cada uma com suas características. Uma comparação entre os controladores foge do escopo deste trabalho, mas existem trabalhos específicos sobre o tema [64, 33].

2.3 Características Desejáveis

Uma arquitetura de Internet do Futuro deve estar atenta à implementação de algumas características desejáveis. Vários autores [79, 6, 71, 77, 2, 70] discutem características implementadas em suas soluções. Abaixo seguem algumas delas.

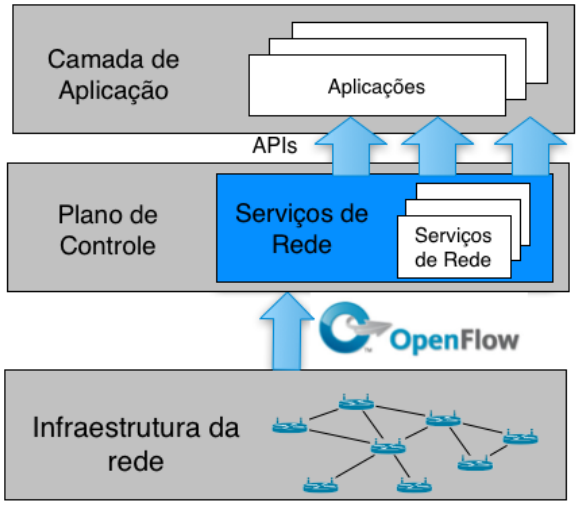

Figura 2.3: *OpenFlow*: Conexão entre a infraestrutura e o plano de controle

Alcançabilidade

Esta característica é a mais importante, e diz que um usuário na Internet deve ser sempre alcançável por outros usuários. Assim, a mobilidade do usuário não irá impedir a entrega de mensagens. Note que os nós não necessariamente precisam conhecer a localização exata do nó móvel, apenas garantir que mensagens possam ser enviadas e que o recebimento delas é conhecido. Isso deve ser garantido em qualquer lugar onde seja possível se conectar à Internet (desde a rede doméstica dos usuários até redes públicas).

Continuidade

Comunicações já estabelecidas não podem ser desfeitas devido ao movimento de usuários móveis. Uma comunicação não necessariamente significa uma conexão, ou seja, uma conexão pode ser desfeita se for re-estabelecida, mantendo o contexto à nível de aplicação. Além disso, usuários podem mudar o dispositivo que é utilizado (ou até utilizar vários dispositivos simultaneamente), podendo iniciar um *streaming* de um computador e retomar a mesma comunicação de seu celular, por exemplo.

Heterogeneidade

A Internet do Futuro deve levar em consideração que não existe mais um único tipo de dispositivo ou serviço. Os dispositivos atuais (computadores, celulares e *tablets*, entre outros) possuem especificações diferentes, como consumo de energia e de dados, poder computacional, e disponibilidade. Uma arquitetura deve também ser capaz de suportar

uma variedade grande de serviços com características diferentes, além de ser capaz de se expandir caso surja um serviço não previsto anteriormente.

Também é necessário trocar o conceito de localidade para a identificação de um dispositivo por algum conceito que vise a mobilidade, como um identificador para o próprio dispositivo ou mesmo para o usuário deste dispositivo. Uma boa prática seria fornecer nomes legíveis para humanos, que pode ser o mesmo nome utilizado pelos dispositivos ou um nome diferente (no mesmo modo que o IP é utilizado pelas máquinas e o nome de domínio é utilizado por humanos). É interessante que esse identificador seja auto-certificável, ou seja, um dispositivo pode autenticar outro dispositivo sem o uso de serviços certificadores de terceiros.

Conectividade

Com a heterogeneidade, é necessário garantir que os diversos dispositivos conectados possam se comunicar entre si de forma transparente. Também é interessante poder diferenciar dispositivos que precisam estar conectados sempre (como servidores), e dispositivos que só necessitam de conexões pontuais (para atualizações ou *feedbacks*, por exemplo). De posse dessa informação, é possível prover níveis de conectividade diferentes para os dispositivos, otimizando os recursos disponíveis. Além disso, é necessário que a arquitetura seja capaz de identificar, endereçar e distinguir hosts, usuários, conteúdos e serviços.

Com mobilidade em mente, é interessante abstrair o conceito ponto-a-ponto da Internet atual. Com a existência de aplicações sensíveis ao contexto e colaborativas, além da distribuição de conteúdos é interessante que a arquitetura permita o envio de um mesmo dado para vários dispositivos (*Multicast*). Também deve-se considerar que um mesmo usuário possui vários dispositivos, e com o *Multicast* é possível enviar um dado para todos os dispositivos do mesmo usuário (como um *e-mail*, por exemplo). Além disso, dispositivos podem estar em mais de uma rede ao mesmo tempo, tornando possível o envio de pacotes por duas (ou mais) redes simultaneamente (*Multihoming*).

Escalabilidade

A escalabilidade de um sistema pode ser definida em relação a seu tamanho (quando a adição de novos usuários e recursos não causam impactos no sistema), em termos geográficos (quando a distância entre usuários ou recursos não impacta no sistema) ou em termos administrativos (quando o sistema se mantém fácil de gerenciar apesar de englobar

muitas organizações administrativas diferentes).

Não se pode esperar que os diversos dispositivos ligados à Internet sejam coordenados entre si. Como há um número vasto de dispositivos distintos que podem se conectar à Internet, pode haver problemas de performance na comunicação entre os dispositivos devido à falta de planejamento da própria rede. Também é impossível ter uma noção da topologia atual da rede justamente pelo conceito de mobilidade – usuários e dispositivos se movem entre redes, podendo estar ou não conectados. Por isso, é necessário que a arquitetura seja escalável em todos os pontos descritos, de forma a não se tornar um gargalo.

Adaptabilidade

A arquitetura deve ser capaz de se adaptar às novas mudanças no comportamento de seus usuários. No passado, a demanda era por conexões remotas e envio de *e-mails*. Não se pode pensar em uma arquitetura que se limite a atender as demandas atuais, de tal forma que seja necessário alterá-la mais uma vez caso as demandas mudem. A Internet do Futuro deve se adaptar às demandas dos usuários ao invés de precisar ser substituída como a Internet atual.

Auto-gerenciabilidade

Para suportar uma alta escalabilidade, é interessante que os dispositivos possam se gerenciar sem intervenção humana. É preciso definir mecanismos e técnicas para que os dispositivos possam executar tais ações. Os dispositivos da rede devem ser capazes de gerenciar as seguintes tarefas sem intervenção:

- Gestão Interna e Otimização: Os dispositivos devem detectar alterações na rede e se atualizar para se adequar às alterações feitas.
- Proteção e cura contra ataques: Um dispositivo deve possuir formas de se proteger contra ataques. Caso não consiga, é interessante que o ataque seja ao menos detectado, podendo se recuperar deste.

Armazenamento e processamento no núcleo da rede

O conceito de armazenamento e computação dos dados apenas nas extremidades da rede começa a se tornar obsoleto, com algumas empresas criando seus próprios nós para ar-

mazenar cópias de conteúdo mais próximo de usuários, reduzindo latências. Com vários dispositivos móveis possuindo planos de dados limitados, é interessante diminuir a quantidade de dados transferidos por dispositivo. Os roteadores atuais são capazes de guardar uma cópia dos conteúdos que foram fornecidos através dele (logo, mais próximo dos usuários), além de realizar algumas tarefas que demandam mais processamento (como encriptação dos dados a serem transferidos). Tais ajustes diminuem a distância que uma requisição precisa trafegar até obter uma resposta, e até obter o conteúdo em si, além da carga nos próprios dispositivos, diminuindo o tráfego total na rede e o consumo de bateria do dispositivo.

Capítulo 3

Trabalhos Relacionados

Tendo em mente as características desejadas apresentadas, há uma variedade de propostas para uma Internet do Futuro mais móvel. Algumas já possuem implementações capazes de executar sobre a Internet atual, para testes e demonstrações, enquanto outras são apenas esqueletos ou levantamento de requisitos e necessidades. Aqui serão listadas algumas delas, além de mostradas suas características mais marcantes.

O IP Móvel [56, 57, 58] foi o primeiro método capaz de fornecer mobilidade para os dispositivos móveis na Internet atual, e confia em agentes móveis capazes de armazenar informações sobre os usuários geograficamente próximos a ele. Os agentes móveis enviam à rede mensagens de anúncio periodicamente, e dispositivos podem também requisitar um anúncio. Após o dispositivo descobrir quem é o agente móvel daquela rede, ele precisa catalogá-lo como sendo de sua rede padrão ou não. Cada dispositivo possui apenas um agente móvel padrão. Quando um dispositivo sai de sua rede principal, seu endereço IP atual é registrado no agente móvel padrão. Quando um pacote é enviado ao endereço IP original do dispositivo, o agente móvel intercepta o pacote, enviando-o por um túnel até a rede nova, fazendo com que o pacote seja entregue ao usuário corretamente. No sentido inverso, o pacote pode ser entregue diretamente, sem precisar passar por agentes móveis.

A Figura 3.1 mostra o fluxo possível da proposta. Após a mobilidade do nó móvel **A**, o nó fixo **B** envia uma mensagem à rede padrão de **A** (1). A mensagem é interceptada pelo agente móvel padrão de **A**, que cria um túnel entre a rede atual e a rede padrão de **A**, encaminhando a mensagem através deste túnel (2). A mensagem é recebida pelo agente móvel da rede atual, que o envia até **A** (3). As respostas de **A** podem ser enviadas diretamente até **B** (4).

O IP Móvel possui diversas variações e melhorias [10, 55, 3, 74, 18], mas todos defen-

dem a utilização de agentes móveis.

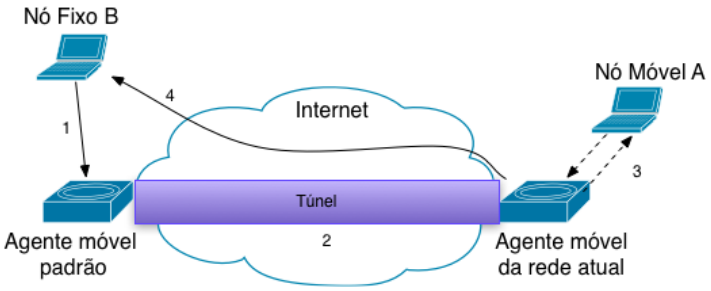

Figura 3.1: Fluxo de pacotes no IP Móvel.

Essa abordagem resolve diversos problemas, mas a troca de mensagens entre os nós e os agentes móveis causa uma maior latência no restabelecimento da conexão e o aumento da quantidade de mensagens de controle. O Projeto *4WARD* [4, 67] mostra que o IP Móvel não seria um bom método devido à natureza dos dispositivos móveis atuais, cuja principal característica é a rápida movimentação entre redes. Tal movimentação acelerada poderia levar a altos atrasos por criar túneis constantemente. Os autores propuseram uma arquitetura de Internet do Futuro que pretende virtualizar redes para que novos protocolos não precisem passar por padronizações demoradas, já que a virtualização é capaz de alterar as conexões e caminhos que passam pelas redes virtuais, realizando o mesmo trabalho dos túneis criados pelos agentes móveis. Além disso, a virtualização de redes permitiria que provedores implantassem novos serviços sem precisar de um consenso do resto da Internet, bastando apenas trabalhar em sua rede virtual designada. Por fim, a proposta ainda seria rentável devido à possibilidade de se dividir recursos da infraestrutura de rede.

Já o trabalho de [72] propôs uma arquitetura definida por *Software* que consiste em vários domínios colaborativos, cada um possuindo um controlador. Grupos de domínios formam uma confederação, e uma mesma aplicação de gerenciamento de mobilidade é instalada em todos os controladores desta confederação, permitindo a mobilidade de dispositivos entre os domínios desta. O plano de controle é responsável por manter mapeamentos de localização de cada nó móvel e por enviar tais mapeamentos para os roteadores quando for necessário. O identificador do nó é o seu endereço IP e a sua localização é dada por um prefixo atrelado ao primeiro roteador *OpenFlow* encontrado pelo nó. O IP fica atrelado ao nó mesmo em caso de mobilidade, mas a sua localização é refeita, agora com o prefixo do roteador da nova rede. Os autores obtiveram bons resultados nos itens viabilidade, escalabilidade e alta capacidade de adaptação da proposta.

Em [44] se discute o uso de CCNs para prover mobilidade. Tem sido demonstrado que arquiteturas orientadas a conteúdo podem suportar conversas ponto-a-ponto em tempo

real, como por exemplo chamadas de voz ou vídeo. No entanto, não foi levado em consideração a mobilidade do dispositivo em tempo real. Em [44], os autores ilustram os problemas de mobilidade em CCNs para aplicações em tempo real. Apesar das vantagens em relação ao armazenamento na rede comentadas na Subseção 2.2.2, aplicações que façam algum tipo de comunicação bidirecional (como chamadas de voz) sofrem do problema ilustrado na Figura 3.2. Neste exemplo, o nó móvel **NM1**, inicialmente na rede **B**, envia pacotes de interesse no conteúdo do nó **NM2**, que está na rede **A**. Ao receber o interesse, **NM2** envia pacotes com os dados de voz, e pacotes de interesse no conteúdo de **NM1**. Se **NM1** se move para a rede **C** (1), por exemplo, a conexão não pode ser estabelecida até que as FIBs atualizem suas entradas de alguma forma. Isto ocorre por causa do princípio de que, nas CCNs, os nomes identificam conteúdo, e não nós. Ainda na Figura 3.2, após a mobilidade de **NM1**, pacotes de interesse de **NM2** continuarão sendo encaminhados até a rede **B** (2). Como os pacotes de interesse não chegam em **NM1**, o nó não enviará os pacotes de dados, quebrando a conexão de voz. Ainda no trabalho de [44], são propostas algumas soluções para tal problema, como a utilização de um servidor para cada domínio (de forma análoga ao agente móvel), a utilização de mensagens de interesse que indicam a movimentação do nó ou a atualização proativa das FIBs.

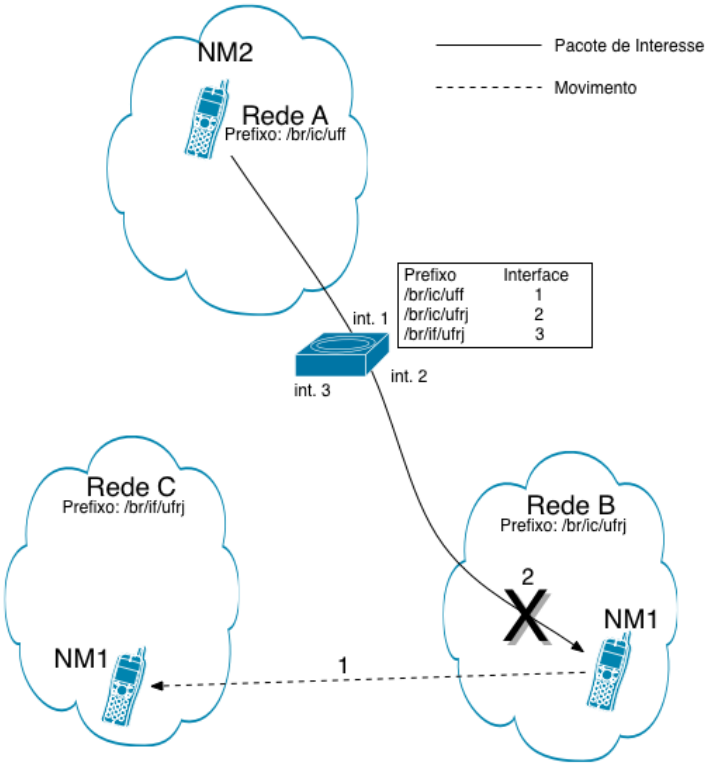

Figura 3.2: Barreira ao prover mobilidade em CCNs - fluxos bidirecionais

O Named Data Networking (NDN) [32] utilizou o código do CCN como base, alterando-

o para atender às necessidades do programa NSF *Future Internet Architecture* [49]. Dentre os projetos para o NDN pode-se citar o *Kite* [78], um esquema de suporte à mobilidade para produtores e consumidores. O *Kite* se utiliza das informações na PIT criadas por aplicações em nós móveis para encontrar tais nós, permitindo que estes sejam alcançados caso se movam durante uma comunicação. Para isso, o *Kite* pode agir de duas formas. Em sua forma direta, quando o nó móvel emite um interesse em um conteúdo de um nó estático. Já em sua forma indireta, o nó móvel gera constantes interesses em um conteúdo de um nó estático chamado de âncora. Um segundo nó estático pode enviar pacotes por um caminho que inclua a âncora, que pode determinar a localização do nó móvel de forma análoga ao método direto.

O *PERIMETER* [12, 19, 20, 21] é um projeto financiado pelo Serviço de Desenvolvimento e Pesquisa de Informações Europeu (EU ICT FP7 [14]), cujo objetivo é estabelecer um novo paradigma para a centricidade do usuário em redes. O projeto cita que a única métrica que considera o usuário de um serviço é a QoE. Assim, é criado um novo modelo de QoE, inferido de acordo com a resposta do usuário e de outros parâmetros técnicos, como custo da rede e nível de segurança. Além disso, são mostrados os seguintes parâmetros considerados necessários para se inferir o nível de QoE:

Performance dos servidores e dos dispositivos: Se os extremos da conexão (o dispositivo do usuário ou o servidor) são os gargalos desta, o usuário não dará uma resposta positiva sobre sua experiência;

Características mensuráveis da rede: Medidas como atraso na transferência e taxa de perda de pacotes podem gerar um nível de QoE baixo para o usuário;

Custo associado ao uso da rede: Se o custo for muito alto, ou o tipo de cobrança não for adequado à capacidade da rede, a resposta do usuário será negativa;

Características dos pontos de acesso: Fatores como disponibilidade, segurança, tempo para estabelecer conexões e perda de pacotes podem alterar a QoE de um usuário;

Segurança: Itens de segurança, como proteções contra ataques de negação de serviço, reputação e o nível de privacidade podem alterar a qualidade de experiência do usuário;

De posse do modelo acima, foi criado um *Framework* para prover aos usuários a melhor Qualidade de Experiência possível. Funcionando como uma rede sobreposta à atual

Internet, o *Framework* também possui um sistema de Autenticação, Autorização, Contabilidade e Reputação preservando a privacidade (Sistema PPA^{3}R), que provê controle da identidade e anonimidade dos usuários e suas preferências, além de cuidar de eventuais problemas com autenticação, reputação e cobrança dos serviços e usuários.

Na arquitetura proposta, ao invés do usuário estar atrelado à uma rede ou operador específico, ele possui uma conta que lhe fornece acesso à todas as redes disponíveis, permitindo assim a mobilidade entre as redes. Utilizando o modelo QoE, o *Framework* é capaz de alterar a rede de forma que a conexão sempre agrade o usuário, ou dá uma notificação caso a rede ótima não esteja disponível. Para que essa mudança dinâmica de redes seja possível, é necessário que os órgãos reguladores – os governos europeus, no caso do *PERIMETER* – criem regras permitindo os contratos propostos. O motivo dado pelos autores para esta intervenção é que há um benefício considerável para os usuários, que são representados pelos governos.

Abaixo, são descritos dois trabalhos que serviram de inspiração para o desenvolvimento desta dissertação. O primeiro é um servidor de nomes global descentralizado que é peça chave na nossa proposta [71, 65], enquanto o segundo é uma proposta completa de uma arquitetura de Internet do Futuro cuja principal prioridade é prover mobilidade [6, 62].

3.1 Auspice

O *Auspice* [71, 65] é um servidor de nomes global que visa prover alta mobilidade, atualmente sobreposto à Internet atual, considerado "massivamente escalável e geo-distribuído" pelos autores, e capaz de resolver identificadores flexíveis para localidades também flexíveis. Ele consiste em um grande número de servidores de nomes distribuídos geograficamente que hospedam réplicas de *resolvers*, um serviço responsável por fornecer os registros de um nome. Um registro mapeia um nome à seus atributos, como os seus endereços atuais e políticas de *multihoming*.

Para os autores do *Auspice*, um servidor de nomes global não deve forçar uma estrutura de nomeação específica. Nomes "planos", além de tornar desnecessário sua alteração conforme os dispositivos se movem, permitem suporte à autenticação na forma de identificadores auto-certificáveis. Para prover escalabilidade e confiança, uma única organização não deve ser responsável por prover o serviço para todos os nomes, mas a arquitetura deve permitir a existência de outros GNSes (além do *Auspice*), permitindo aos dispositi-

vos escolher livremente qual utilizar, analogamente ao sistema DNS atual.

O *Auspice* também é projetado para possuir baixos tempos de resposta e custo de atualização, além de consistência e alta disponibilidade. Isso é feito distribuindo *resolvers* geograficamente perto dos clientes, com uma quantidade de réplicas baixa o suficiente para ter baixo custo de atualização, mas grande o suficiente para prover alta disponibilidade. Todas essas metas são simples, mas fornecê-las ao mesmo tempo pode se tornar uma tarefa desafiadora.

A peça chave do *Auspice* é um motor que permite computar o número e a localização das réplicas de *resolver* para um nome baseado na demanda por aquele nome, além de distribuir novas réplicas para localizações com demanda crescente. Foi provado que um algoritmo exato para isso é computacionalmente difícil, então é utilizada uma heurística que pode ser executada independentemente por cada controlador de réplica.

Cada identificador (nome) armazenado no *Auspice* é associado a um número fixo de controladores de réplica, e um número variável de réplicas ativas. A localização dos controladores de réplica é conhecida e obtida através de funções *hash* (algoritmos que mapeiam dados de comprimento variável para dados de comprimento fixo, acelerando a busca) responsáveis por mapear um identificador até a localização do seu servidor de nomes. Os controladores de réplica formam o "plano de controle", e são responsáveis por determinar o número e a localização das réplicas ativas. Estas são responsáveis pelas entradas e por fazer as resoluções de nome.

Quando é feita a primeira resolução para um nome, o servidor de nomes local procura, através das funções *hash*, o controlador de réplica para aquele nome, que retorna uma lista com as suas réplicas ativas. O servidor local, então, faz a requisição do nome para a réplica ativa mais próxima a ele. A comunicação com o controlador de réplica não deve ser frequente, já que os servidores locais podem armazenar a localização da réplica ativa mais próxima via *cache*. Para decidir qual é o servidor mais próximo, os servidores locais mantém estimativas de latência para os outros servidores.

A consistência dos registros nas réplicas, no caso de alterações, é mantida através do método a seguir. O controlador de réplicas, ao receber um pedido de adição, remoção ou alteração de um registro, envia o registro para suas réplicas ativas, que apenas o armazena, sem alterar ainda suas respectivas tabelas. Quando um número suficientemente grande de confirmações de recebimento do registro pelas réplicas ativas for recebido, o controlador enviará um comando para que insiram o registro em suas tabelas.

Para permitir mobilidade entre as sessões, o *Auspice* provê uma API capaz de renegociar o estado da conexão e realocar as conexões ativas para outra combinação de endereço IP e porta. Um nó que sofre mobilidade pode utilizar de tal API para modificar todas as suas conexões ativas, adicionando seu endereço IP atual e restabelecendo a comunicação. Esta API é chamada de *Msocket* [76], e existe na forma de uma biblioteca que estende os *sockets* padrão.

Testes foram realizados através de simulações, experimentações via PlanetLab[60] e emulação em um *cluster* com 16 nós. Foram avaliadas alterações (que podem ser adições ou remoções) de registros de serviços (como servidores *Web*), e de nomes de dispositivos móveis. Foram feitos testes comparando o *Auspice* a um esquema ótimo, que decide tanto a replicação de *resolvers* quanto o redirecionamento de pedidos para minimizar a latência. Resultados mostraram uma latência entre 1.1x e 2.1x do esquema ótimo, alertando que a heurística simples utilizada pelo *Auspice* pode ser melhorada.

O *Auspice* também foi avaliado em relação à provedores de DNS gerenciados atuais. Foi limitado o número de réplicas por nome do Auspice para 5, menos de um terço do número de localizações dos provedores de DNS gerenciados avaliados. As latências do *Auspice* foram similares às dos provedores de DNS no teste de pesquisa de registros, porém o *Auspice* foi cerca de 4 vezes mais rápido no teste de atualização de registros.

Por fim, outro teste interessante foi o de mobilidade durante uma sessão. Este experimento mostrou uma transição entre uma rede *Wi-Fi* para uma rede 4G. A conexão foi restabelecida após 412ms, porém a mesma leva aproximadamente 2 segundos para estabilizar (voltar à mesma vazão de antes da mobilidade). Os autores argumentam que o tempo necessário para o restabelecimento da conexão foi equivalente à 3 RTTs (*Round Trip Time*, o equivalente ao tempo necessário para enviar de uma mensagem e receber uma confirmação de recebimento), mas o tempo de estabilização da conexão ainda é alto.

3.2 MobilityFirst

O *MobilityFirst* [6, 62] é um projeto financiado pelo programa NSF *Future Internet Architecture* [49] e visa combater diretamente os desafios relacionados à acesso sem fio, mobilidade e confiança, enquanto provê novos serviços necessários para cenários de aplicações para uma Internet mais "móvel".

O *MobilityFirst* tem os seguintes objetivos em seu projeto:

Mobilidade ininterrupta: A arquitetura deve levar a mobilidade como princípio fundamental, e não tratá-la como casos pontuais;

Múltiplas raízes de confiança: Não deve haver uma única entidade de confiança, como atualmente há para traduzir nomes em endereços IP (o ICANN [29]);

Controle dos dados: Os receptores dos dados devem poder controlar o tráfego de entrada e bloquear tráfego não desejado;

Robustez: Um número pequeno de nós comprometidos não devem causar um impacto desproporcional na performance ou disponibilidade da rede;

Endereçabilidade de conteúdo: A rede deve facilitar recuperação de conteúdo como uma CCN;

Capacidade de evolução: A arquitetura deve permitir desenvolvimento rápido de novos serviços.

A arquitetura proposta utiliza GUIDs (IDentificadores Únicos Globais) para objetos ligados à rede, que podem ser dispositivos móveis, pessoas, grupos de dispositivos/pessoas, conteúdo ou serviços. As GUIDs do *MobilityFirst* são geradas através de funções *hash* aplicadas às chaves públicas daquele objeto de rede, e sua autenticação ocorre da seguinte forma. Caso um dispositivo X queira autenticar o dispositivo Y (ou seja, Y é realmente o dono da GUID Y), X envia um pedido de autenticação com um valor aleatório n. Y então retornará sua chave pública K_y, e o valor n cifrado utilizando sua chave privada. Primeiramente, X utilizará a função geradora em K_y, comparando o resultado com a GUID Y. Após isso, X tenta obter de volta n, decifrando o valor enviado por Y. Caso ambos os testes sejam verdadeiros, a autenticidade de Y está validada.

O *MobilityFirst* também suporta mais de um serviço de nomes (como o *Auspice*), e incentiva o uso de serviços de nome diferentes para modelos diferentes de GUIDs. Segundo os autores, num cenário ideal haveria um serviço de nomes diferente para dispositivos (móveis ou não), sensores, conteúdo, contexto e até mesmo redes. No último caso, o nome da rede seria o resultado de uma função *hash* aplicada à GUID. O resultado desta operação é chamado de NA – *Network Address*, ou endereço da rede.

Uma mensagem padrão do *MobilityFirst* contém as GUIDs de origem e destino, um código para o tipo de mensagem (que pode ser *Unicast, Multicast, Anycast, Multihomed*, busca de conteúdo ou de serviço), e, caso seja conhecido, um ou mais NAs que identificam a localização atual do destino. Os roteadores possuem duas tabelas, uma que atrela GUIDs

à NAs e outra que atrela NAs ao próximo salto ou interface de saída. Quando a lista de NAs está no pacote, a operação é similar à de um roteador convencional – verificar na tabela por qual interface encaminhar o pacote. Qualquer roteador pode requisitar novamente a localização da GUID, utilizando o serviço de nomes, alterando o cabeçalho do pacote e permitindo uma rápida recuperação em caso de mobilidade. Por fim, os roteadores intra-domínio possuem armazenamento para os dados enviados, permitindo assim o reenvio de uma mensagem cujo destino é um dispositivo que se desconectou desta rede.

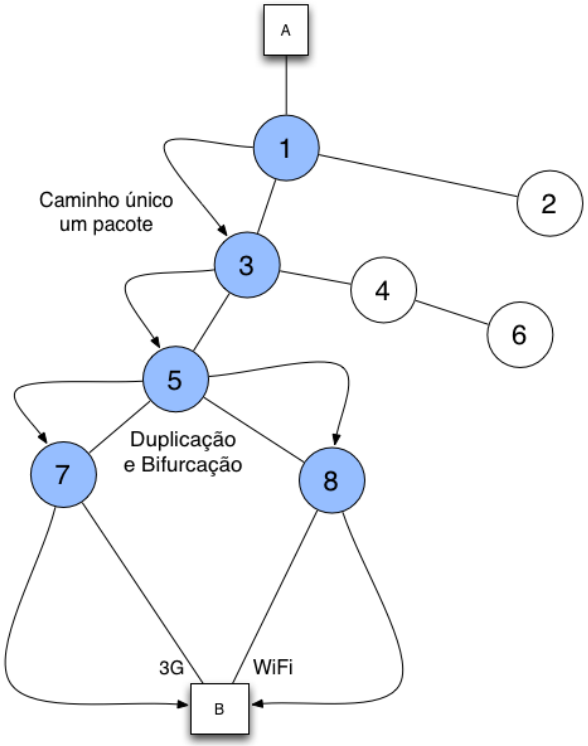

Figura 3.3: Exemplo de *Multihoming* na arquitetura *MobilityFirst* – encaminhamento do pacote

A Figura 3.3 exemplifica o encaminhamento de pacotes até um usuário em *multihoming*. Caso um usuário **A** queira enviar uma mensagem à todos os dispositivos de um outro usuário **B**, a origem, ao pesquisar por **B** em um serviço de nomes, obterá uma GUID de grupo que está atrelada à todos os NAs dos quais **B** faz parte. De posse desta GUID, o pacote *MobilityFirst* é criado e, ao ser enviado, será utilizado um algoritmo de encaminhamento do tipo "caminho comum mais longo", ou seja, o pacote será encaminhado sem duplicação o mais longe possível (no caso, até o roteador 5) antes de ser dividido em dois pacotes, cada um sendo enviado à um NA de destino.

O *MobilityFirst* trata a mobilidade da forma exemplificada pela Figura 3.4. Nela,

o dispositivo **A** envia um pacote para o dispositivo **B**, que se move do NA 7 para o NA 8 enquanto o pacote está em trânsito (1). O pacote será enviado até o NA 7, onde ocorrerá uma falha na entrega, devido à mobilidade. O pacote será então armazenado no roteador de borda (2), enquanto pesquisas são feitas, em algum serviço de nomes, em busca da localização atual de **B** (utilizando sua GUID, contida no pacote *MobilityFirst*). Ao receber a informação que agora **B** está no NA 8, o roteador altera o pacote com o novo destino, encaminhando-o até o dispositivo corretamente (3).

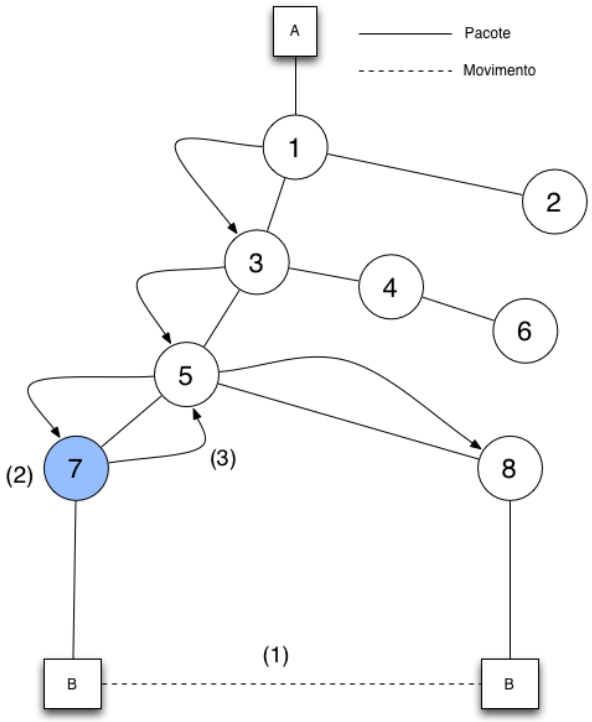

Figura 3.4: *MobilityFirst* – exemplo de mobilidade

Existem alguns trabalhos que avaliaram o *MobilityFirst* em ambientes de testes [39, 35]. Em [35], um protótipo das funcionalidades do *MobilityFirst* foi implementado utilizando o protocolo *OpenFlow*. Em seus testes, os autores mostraram que a vazão da conexão para pacotes grandes (na ordem de grandeza de *megabytes*) é maior do que roteadores definidos por *Software* padrão, porém para pacotes pequenos, a vazão é baixa. Além disso, como era esperado, pacotes que necessitam de busca em um serviço de nomes tem um atraso maior devido à comunicação entre o controlador e o GNS. Por fim, os autores detectaram uma grande carga no controlador quando muitos pacotes necessitam de processamento.

Em [39], são utilizadas redes virtuais como forma de evitar o roteamento via endereço IP, reduzindo o atraso criado por este roteamento. Para isso, foi utilizado a implementação

de [35] como base. A idéia consiste em associar pacotes a redes virtuais, transformando a rede em circuitos virtuais que ligam um roteador à outro. Pacotes são enviados por estes circuitos, que podem se alterar de acordo com a mobilidade dos dispositivos. Porém, foi descoberto que tal aumento no processamento causa um atraso de aproximadamente 680ms para cada 1000 pacotes, mostrando que ainda há espaço para evolução nesta técnica.

Capítulo 4

Proposta

Como já foi mostrado em trabalhos relacionados, o IP Móvel não é uma solução escalável devido à quantidade de mensagens de controle introduzidas na rede. Como o núcleo da rede atual não é mutável, a virtualização de redes proposta por [67, 4] e o uso de CCNs do trabalho de [44] também não são viáveis no quesito aplicabilidade, e a atualização de informações pelos controladores de [72] podem sobrecarregar a rede em casos extremos. As propostas de [12, 19, 20, 21, 71, 65] partem do pressuposto de uma rede totalmente nova, então também não são viáveis atualmente. Por fim, como o *Msocket* é apenas uma biblioteca, apesar de permitir mobilidade sem a necessidade de mudanças em sistemas operacionais ou na infraestrutura de rede, é necessário que todas as aplicações sejam reescritas utilizando sua biblioteca.

Soluções de mobilidade baseadas em *hosts* podem ser inviáveis pela quantidade de dispositivos, assim como alterações bruscas no núcleo da rede são inviáveis pela dificuldade de implantação. Por isso, a proposta deste trabalho consiste em levar a complexidade da mobilidade para a borda da rede, mas ainda fora dos dispositivos finais. Assim, é permitida a mobilidade de dispositivos em qualquer tipo de aplicação sem a necessidade de alteração em seus códigos e no núcleo da rede. Para ser uma solução mais viável, apenas os roteadores de borda das redes necessitarão de mudanças, deixando o núcleo da rede intacto.

A solução desenvolvida neste trabalho é inspirada pela ideia de [72], utilizando roteadores inteligentes com controladores externos. Com o auxílio da Figura 4.1, será explicado o comportamento padrão destes roteadores. O nó fixo **NF1** deseja enviar uma mensagem até o nó móvel **NM1**. Como se trata de uma conexão nova, apenas **S3** conhece **NM1**, já que o dispositivo está conectado a este roteador. Quando **NF1** enviar um pacote ao

roteador de primeiro salto **S1** (1), este irá consultar seu controlador **C1** (2), que irá executar os algoritmos de encaminhamento necessários. O controlador retorna a regra de fluxo, informando que o pacote deve ser encaminhado pela interface 1 do roteador (3). O roteador **S2**, que também não sabe a localização de **NM1**, também irá pedir uma regra à **C1** (4). Analogamente, **C1** informa que **S2** deve encaminhar o pacote pela interface 2. **S3** recebe o pacote e, sabendo que NM1 está em sua interface 1, irá entregar o pacote corretamente (6).

Agora, ainda com o auxílio da Figura 4.1, suponha que **NM1** se moveu para **S4** (7). Como as Tabelas de Fluxos de **S1**, **S2** e **S3** ainda estão com as informações antigas fornecidas por **C1**, os pacotes de **NF1** para **NM1** serão encaminhados até a interface 1 de **S3**, onde o pacote se perderá. Portanto, **NF1** não poderá enviar mensagens para **NM1** até que as tabelas sejam atualizadas.

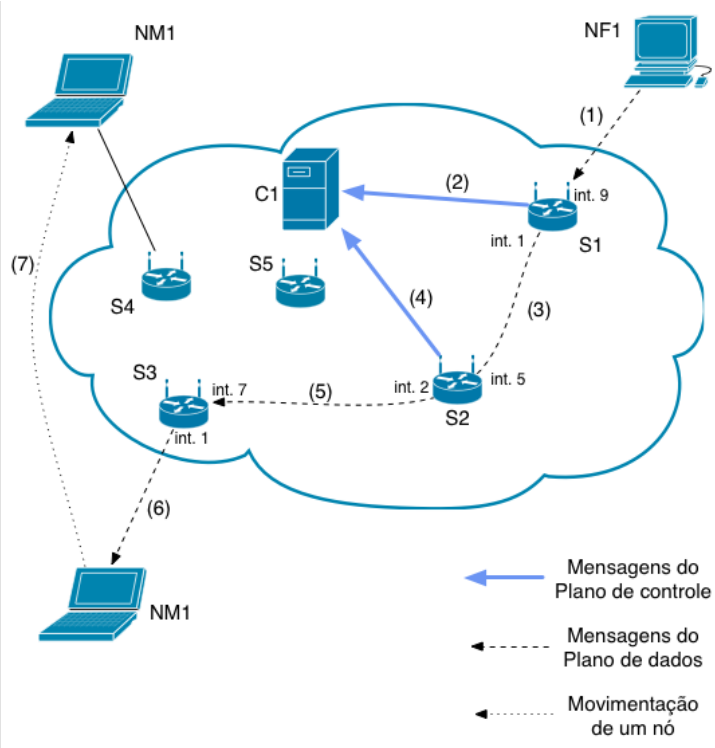

Figura 4.1: Exemplo de mobilidade em redes SDN padrão

Nesta proposta, é utilizado o poder do *Auspice* para armazenar a localização atual dos nós, e a proposta recorre a ele no caso de mobilidade. Mas é preciso saber quais informações são necessárias armazenar no GNS, para que o controlador possa alterar os pacotes corretamente. Como é desejável manter a conexão, é necessário saber o endereço IP antigo (antes da mobilidade) e novo (depois da mobilidade) do nó móvel. Assim, pode-se capturar os pacotes que possuem o endereço MAC do nó móvel no cabeçalho da camada

de enlace, assim como o endereço IP antigo no cabeçalho da camada de rede, alterando o cabeçalho ao inserir o endereço IP novo como destinatário. Além disso, uma regra de fluxo básica informa ao roteador por qual interface o pacote deve ser enviado, então esta informação também deve ser armazenada. Por fim, é necessário um identificador para o dispositivo. Como utilizar o endereço IP traria todos os problemas relacionados à ele e já discutidos, foi escolhido utilizar o endereço MAC da interface conectada ao roteador. Como a rede não conhece nenhuma informação do nó além do endereço MAC e do endereço IP, não é possível utilizar um identificador por dispositivo sem alterações nos sistemas dos usuários. O endereço MAC é único para cada interface, então é possível prover mobilidade para conexões utilizando esta mesma interface.

Na Figura 4.2, quando **NM1** move para S4 (1), o roteador **S4** avisa ao controlador **C1** que o nó está em sua rede (2). **C1**, então, atualiza o *Auspice* com as informações de localização de **NM1** – IP atual, roteador e interface (em relação à **S4**) (3). Assim, se **NF1** tentar se comunicar com NM1, **C1**, ao ser consultado por **S1** (4), irá requisitar ao *Auspice* a entrada relativa à **NM1** (5) e assim, podendo gerar corretamente a regra de encaminhamento.

Nesta proposta, porém, existe um problema na mobilidade durante a conexão. Caso **NF1** tenha se comunicado com **NM1** antes da mobilidade (enquanto ele estava em **S3**), as tabelas de fluxos de S1 ainda conterão informações que encaminharão os pacotes destinados à **NM1** até **S3**, causando falhas até que a tabela seja atualizada. Além disso, entradas na tabela de fluxos não são removidas se houver alguma conexão que a utilize, então se a mobilidade foi feita durante uma conexão, a regra ficará na tabela por um tempo determinado de inatividade até ser deletada (*timeout*).

Para mitigar tais problemas, serão utilizadas algumas estruturas auxiliares, armazenadas nos controladores. São elas:

- Tabela de nós conhecidos (TNC): Cada controlador possui uma tabela de nós conhecidos, além de informações sobre o seu roteador de próximo salto e em qual interface deste o nó está conectado;
- Tabela de nós desconectados (TND): Esta tabela guarda os nós que estavam se comunicando, e se desconectaram por algum motivo;
- Tabela de conexões ativas (TCA): Esta tabela guarda uma tupla de dois nós que indica a existência de uma conexão entre eles no momento da mobilidade.

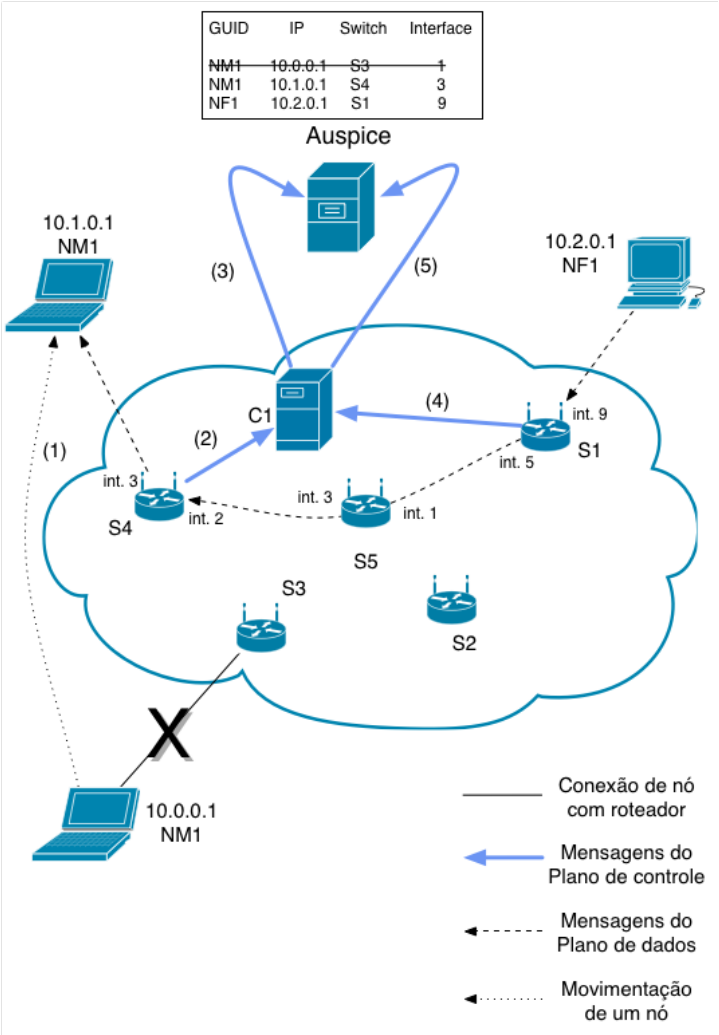

Figura 4.2: Proposta: Modelo Inicial

Assim que um controlador recebe um pacote cuja origem ou destino é um nó ainda ausente na TNC, o controlador faz a adição das informações do nó, para uso posterior. Se um nó ainda não está na TNC, ele ainda não enviou nem recebeu nenhuma mensagem, logo não há regras de fluxo associadas à ele em nenhum roteador controlado por este controlador, podendo então ignorar a mobilidade sofrida por ele.

A TND possui dois propósitos. O primeiro é cortar conexões cujo destino é algum nó contido nela, pois isso geraria uma regra de fluxo desatualizada, comprometendo a entrega correta após o início da mobilidade. O segundo propósito é servir de gatilho para a deleção de todas as regras de fluxo que encaminhariam pacotes até um nó recém adicionado na tabela.

Por fim, o objetivo da TCA é saber quais conexões estavam ativas antes da mobilidade, tornando possível retomar a conexão quando o nó móvel se reconectar.

Assim, a proposta evolui para o mostrado na Figura 4.3. Quando **NM1** se desconecta de **S3** (1), **S3** avisa à **C1** que houve uma desconexão (2), e o controlador acessa a tabela de fluxos de **S3** (3), inserindo na TCA todos os fluxos relativos à **NM1**. Além disso, informações sobre **NM1** são inseridas na TND, ativando o gatilho e apagando os fluxos relacionados à **NM1** de **S3**. Desta forma, quando um novo pacote destinado à **NM1** for recebido por **S3**, este irá até **C1** em busca de uma regra nova. Quando **S5** informar que **NM1** está conectado à ele (4), **C1**, após atualizar o *Auspice* (5), irá buscar na TCA todos os fluxos relacionados à **NM1** e criará uma regra de fluxo, alterando o endereço IP dos pacotes destinados ao IP antigo de **NM1** e encaminhando-os pela interface correta. Portanto, pacotes destinados à **NM1** seguirão até **S3** com o cabeçalho "antigo" de **NM1**, onde terão o cabeçalho alterado e serão encaminhados corretamente até **NM1**. Para se manter o estado da conexão (quando utilizados protocolos que permitam tal manutenção), a mudança do endereço IP é feita em ambos os sentidos da conexão. Ou seja, pacotes de **NM1** destinados à **NF1** terão seus cabeçalhos alterados, re-inserindo o endereço IP antigo de **NM1**. Desta forma, quando **NF1** receber o pacote, será atribuído à conexão ativa.

4.1 Análise da proposta

Aqui, a proposta apresentada será analisada em relação às características desejadas da Seção 2.3 (alcançabilidade, continuidade, heterogeneidade, conectividade, escalabilidade, adaptabilidade, auto-gerenciabilidade e armazenamento e processamento no núcleo da rede).

Como a única entidade que necessita saber a localização atualizada do nó é o *Auspice*, a alcançabilidade é garantida. Um nó conectado à Internet sempre poderá ser alcançado, mesmo que os outros nós não conheçam sua localização, já que, caso isso ocorra, o roteador de próximo salto irá requisitar uma regra de seu controlador, e este buscará a localização do nó móvel no *Auspice*. Como os pacotes de conexões que estavam ativas no momento de mobilidade são alterados, a continuidade também é garantida, já que conexões ativas serão retomadas assim que o nó se reconectar.

A proposta deste trabalho afeta qualquer dispositivo que possua uma interface com endereço MAC, e que receba um endereço IP ao se conectar à Internet. Como todo tipo de dispositivo necessita, atualmente, destes dois endereços, é garantida a heterogeneidade. A garantia à adaptabilidade é herdada da definição de SDN de que é possível adicionar

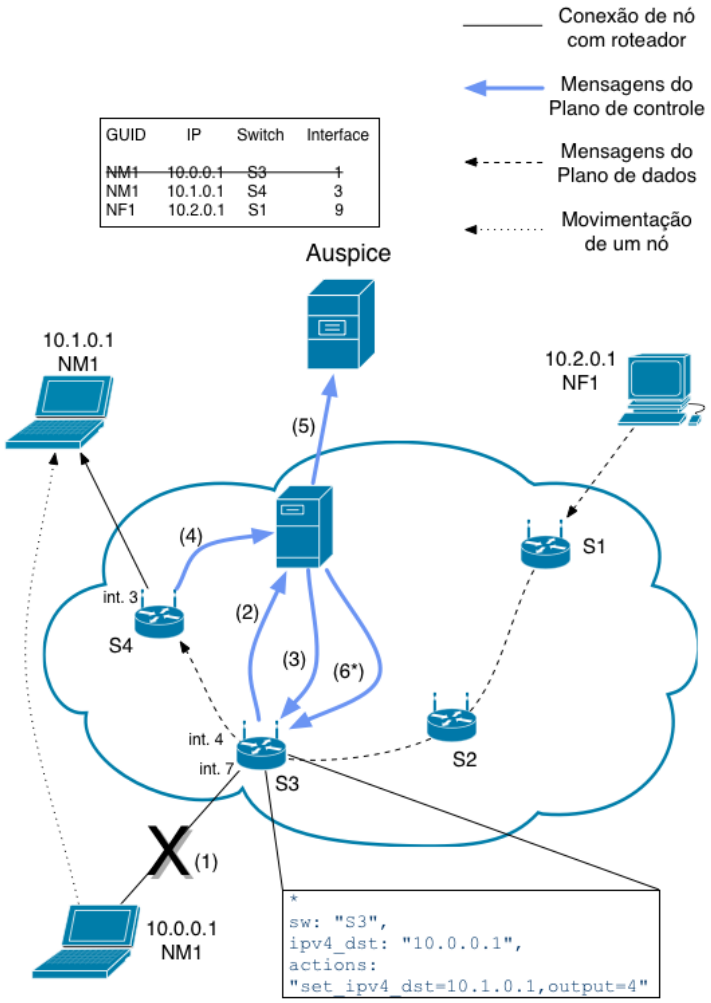

Figura 4.3: Proposta: Caso de mobilidade funcional

novas APIs, suportando assim demandas novas que venham a surgir.

A conectividade, auto-gerenciabilidade e armazenamento e processamento no núcleo da rede não são garantidas à princípio, mas, como esta proposta é uma arquitetura SDN, é possível criar APIs novas que atacam este problema. Além disso, a proposta apresentada consegue distinguir pacotes que são destinados à nós desconectados graças à TND, portanto um roteador (ou até mesmo um controlador) com espaço de armazenamento suficiente poderia armazenar tais pacotes enquanto o nó móvel não se reconecta.

A escalabilidade é parcialmente garantida já que o *Auspice* é considerado "massivamente escalável". Com o aumento de dispositivos se movimentando entre redes, é possível aumentar o número de réplicas do *Auspice* e de controladores. Porém, com o aumento da taxa de mobilidade na rede, os controladores terão que apagar as tabelas de fluxo constantemente. Esta questão será tratada melhor na Seção 4.2.

Por fim, é importante ressaltar que a co-existência da proposta apresentada com a Internet atual é factível, já que é apenas necessário que a rede que queira implantar a proposta suporte SDNs. Por isso, é possível que algumas redes utilizem a proposta, enquanto outras não. Neste caso, a mobilidade seria garantida apenas nas primeiras redes.

4.2 Considerações

A proposta descrita possui uma limitação se comparada à implementação tradicional do *MSocket*. A limitação está no próprio endereço MAC, visto que seu uso pode impossibilitar a mobilidade entre interfaces, ou *multihoming*. É um caso comum de mobilidade a ocorrida entre uma rede *Wi-Fi* e uma 3G, sendo que o nó móvel possui uma interface para se conectar a redes *Wi-Fi* e outra para redes 3G. Neste caso, o mesmo dispositivo terá dois endereços MAC (um para cada interface) e, portanto, duas GUIDs diferentes. Logo, o sistema não entenderá que houve mobilidade. Um modo de resolver tal limitação é utilizar o mesmo artifício usado na *MobilityFirst*, que consiste em criar um identificador por dispositivo, e não por interface (como o endereço MAC atualmente utilizado como identificador em nossa proposta). Esta modificação faz com que seja necessário alterar o dispositivo do usuário, de forma que suas interfaces enviem o identificador do nó junto do seu pacote.

Outra limitação está no fato de o *Auspice* aceitar atualizações das informações dos dispositivos ao recebê-las de qualquer roteador, sem levar em consideração a segurança. Um controlador malicioso pode fazer um ataque de negação de serviço ao enviar atualizações inexistentes sobre a mobilidade de um nó para o *Auspice*, fazendo com que outros controladores, ao buscarem tais informações, gerem regras que não permitirão que pacotes sejam encaminhados até tal nó. Uma solução possível é permitir a atualização apenas pelo controlador da rede em que o nó se encontrava antes da mobilidade, podendo delegar atualizações a respeito deste nó apenas ao controlador do novo roteador após a mobilidade.

Por fim, há uma limitação quanto ao roteamento inter-domínio e a sincronização de controladores. Na Figura 4.4(a), $\mathbf{C_a}$, $\mathbf{C_n}$, $\mathbf{S_a}$ e $\mathbf{S_n}$ são controladores responsáveis por controlar 4 redes distintas mas conectadas entre si de alguma forma. Será utilizado o mesmo nome para o controlador e para a rede controlada por este. **C** e **S** são nós que atualmente estão em $\mathbf{S_a}$ e $\mathbf{C_a}$, respectivamente, e que estão se comunicando. Como já

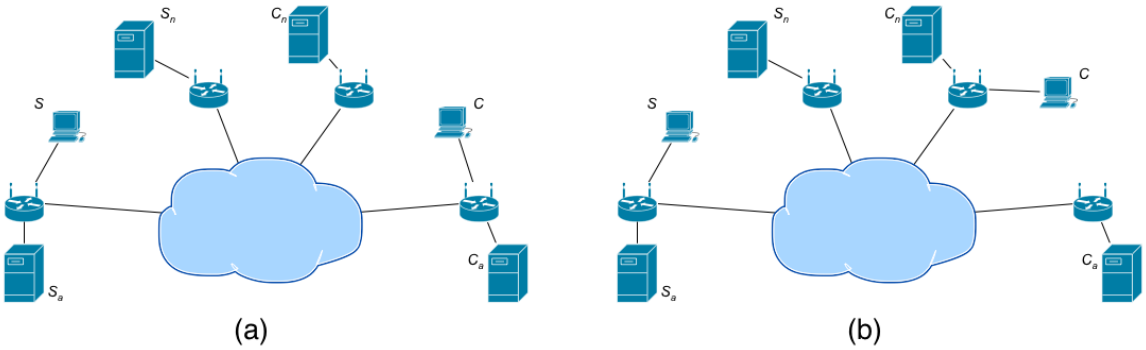

Figura 4.4: Mapa da rede (a) antes da mobilidade; e (b) depois da mobilidade

houve algum contato entre **C** e **S**, as tabelas de fluxo dos roteadores da rede $\mathbf{C_a}$ possuem regras para encaminhar pacotes até $\mathbf{S_a}$, e vice-versa.

Caso **C** se mova para a rede $\mathbf{C_n}$, como na Figura 4.4(b), o controlador $\mathbf{C_n}$ irá atualizar o *Auspice* com as novas informações de **C**, mas o roteador de $\mathbf{S_a}$ não irá acionar seu controlador, por já existir uma entrada referente à **C** em sua tabela de roteamento. Portanto, todos os nós destas redes irão enviar pacotes para $\mathbf{C_a}$, acreditando que **C** ainda está nesta rede. Caso, por exemplo, **S** se mova para $\mathbf{S_n}$, a conexão se retomaria corretamente, pois $\mathbf{S_n}$ não possui informações de **C**, indo então consultar o *Auspice*. Atualmente o algoritmo apaga, nas tabelas de fluxo dos controladores, qualquer informação sobre o nó **C** logo que é detectada sua desconexão. Porém, essa tática não é escalável, como já foi dito anteriormente.

Uma solução possível para este problema consiste em aproveitar as informações do *Auspice*, que, antes de ser atualizado por $\mathbf{C_n}$, possui a informação que **C** estava na rede $\mathbf{C_a}$, além do fato que as tabelas de fluxo de $\mathbf{C_a}$ possivelmente ainda possuem regras dos nós com os quais **C** se comunicou. Assim, logo que $\mathbf{C_n}$ aceitar a entrada de **C** na rede, $\mathbf{C_n}$ pode pedir a tabela de fluxos de $\mathbf{C_a}$ e, com ela, informar aos controladores necessários (no caso, $\mathbf{S_a}$) que eles precisam se atualizar. A não existência de informações na tabela de fluxos de $\mathbf{C_a}$ sinaliza que não há conexões ativas, pois, sem regras na tabela de fluxos, não há como encaminhar pacotes e no início de uma conexão, é instalada uma regra na tabela de fluxos.

Uma solução mais robusta para a sincronização de roteadores é se aproveitar do algoritmo de *Paxos* [38]. O *Paxos* é um algoritmo utilizado em sistemas distribuídos que consiste em escolher um valor dentre os fornecidos por vários sistemas. Como exemplo, em servidores de nomes distribuídos, o *Paxos* ajudaria a resolver um endereço IP, escolhendo um nome dentre os fornecidos pelos vários servidores.

Na proposta apresentada, é possível utilizar o *Paxos* para escolher a localização atual de um nó. Para isso, o algoritmo receberia, de todos os controladores que possuem informações sobre o nó em questão, suas entradas nas tabelas de fluxo. A entrada correta deve ser escolhida pelo *Paxos*, e controladores que possuírem entradas diferentes atualizariam suas tabelas com a entrada escolhida. Desta forma, todos os controladores estariam sincronizados com a mesma informação sobre nós móveis.

Capítulo 5

Implementação e Experimentos

Neste capítulo será detalhada a implementação da proposta apresentada no Capítulo 4, além de definidos os experimentos e ambientes de teste.

5.1 Implementação

A proposta deste trabalho foi implementada como um módulo executado por controladores *OpenFlow*. Foi escolhido o *Floodlight* [24] para implementar o *OpenFlow*, pela facilidade de programação, já que é baseado em Java [26]. O *Floodlight* está configurado para executar este módulo antes de qualquer outro, garantindo assim que os outros algoritmos executados pelo controlador receberão o pacote com as informações atualizadas dos nós. O módulo possui cerca de 800 linhas de código e, graças à API orientada a eventos fornecida pelo *Floodlight*, é acionado quando os seguintes eventos ocorrem:

Recebimento de Pacotes: Os controladores recebem pacotes em caso de um roteador não saber roteá-lo;

Mudança em interfaces: Quando um roteador detecta o estabelecimento ou quebra da conexão de um enlace em uma de suas interfaces;

Detecção de novos nós: Quando um roteador percebe um novo nó em sua rede;

Na inicialização do controlador, são criadas as tabelas TNC, TND e TCA, inicialmente vazias. Também são criadas tabelas auxiliares que guardam nós cuja mobilidade foi detectada. É criada uma tabela para a mudança do ponto de conexão do nó, e outra para a mudança de seu endereço IP. Além disso, o módulo se registra para receber informações

na ocorrência dos eventos acima, e inicializa sua conexão com o *Auspice*. O fluxograma do módulo é mostrado na Figura 5.1. Abaixo serão descritos detalhadamente os passos tomados ao se receber uma notificação de cada evento registrado.

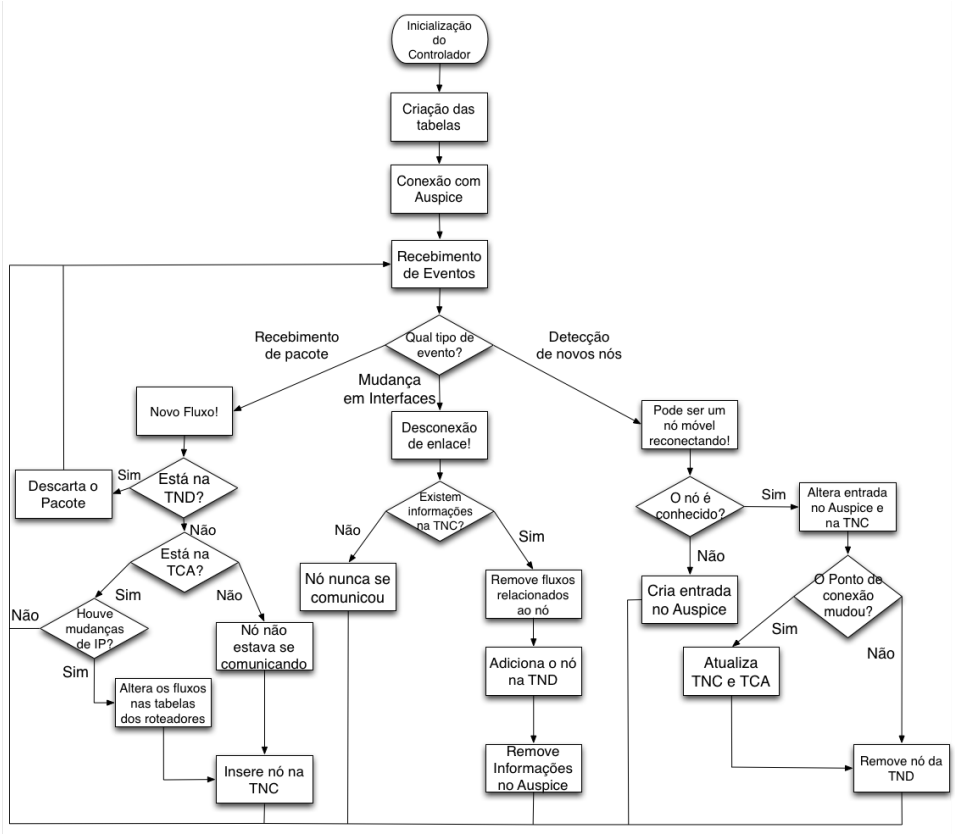

Figura 5.1: Fluxograma do módulo desenvolvido

5.1.1 Recebimento de pacotes

Este evento é acionado quando um roteador envia um pacote ao controlador, por não saber como entregá-lo ao seu destino. Portanto, é seguro dizer que este é um novo fluxo, então é necessário tratá-lo. Primeiramente, o módulo checa se o destino está na TND, descartando o pacote caso esteja (neste caso, é possível armazenar o pacote para entrega posterior). Após, é checado se o nó está na TCA, o que sinaliza a existência de uma conexão antiga entre estes dois nós que é necessária restaurar. Para isso, é verificado se ocorreu mudança no endereço IP, através da tabela auxiliar. Caso afirmativo, todos os fluxos contidos neste roteador cujo destino é o nó móvel são alterados, inserindo no pacote o novo endereço IP do nó.

Após estas checagens, o nó é inserido (caso seja o primeiro fluxo no qual o nó participa) ou atualizado (caso o nó já seja conhecido) na TNC. Por fim, o pacote é devolvido para

o *Floodlight*, que irá entregá-lo para os próximos módulos e algoritmos.

5.1.2 Mudança em interfaces

Este evento pode ocorrer quando uma interface é adicionada ou removida, ou quando um novo nó é (des)conectado à uma interface. A alteração na interface em si não é preocupante, apenas a mudança de enlaces. Por isso, é feita uma verificação de qual tipo de mudança ocorreu. Caso o evento tenha sido de ativação de um enlace, também não é necessária nenhuma ação, pois neste momento o módulo possui apenas informações sobre o enlace, quando são necessárias informações sobre o nó em si. Portanto, este caso será tratado quando o nó for detectado.

Caso o evento tenha sido gerado pela quebra de um enlace, o módulo procura na TNC informações sobre o nó, já que neste ponto são conhecidas apenas informações sobre o enlace em si. A informação mais importante que pode ser obtida é o endereço MAC, e com este é possível obter, através da TNC, todas as informações necessárias sobre o nó que se desconectou. De posse destas informações, o módulo procura fluxos relacionados ao nó, armazenando-os na TC e os removendo do roteador que gerou o evento, caso existam. Por fim, o nó é adicionado à TND e sua entrada no *Auspice* é removida.

5.1.3 Detecção de novos nós

Este evento pode ser causado tanto pela adição de um nó desconhecido, quanto pela mudança do endereço IP ou ponto de conexão (interface do roteador) de um nó conhecido. Caso um nó desconhecido seja conectado, a única ação é a criação de uma entrada para ele no *Auspice*.

Caso o endereço IP de um nó seja alterado, o módulo primeiramente busca a entrada do nó no *Auspice*, e verifica o(s) novo(s) endereço(s) IP do nó, alterando a TNC conforme necessário. Então, o módulo atualiza o *Auspice* com as novas informações e adiciona o nó à tabela auxiliar correspondente.

Por fim, caso ocorra a mudança do ponto de conexão de um dispositivo, o módulo busca pelo nó na TNC. A não existência do nó na tabela sinaliza que nunca ocorreu comunicação com este nó, já que nenhum pacote com suas informações foi recebido pelo controlador. Portanto, pode-se ignorar sua mobilidade. Caso existam informações sobre o nó na TCA, o módulo checa se o ponto de conexão é diferente dos contidos na TCA, atualizando o *Auspice* caso necessário. Além disso, se o ponto de conexão for diferente, o

controlador sinaliza que este nó é móvel, adicionando-o à tabela auxiliar correspondente. Por fim, o nó é removido da TND.

5.2 Experimentos

Para validar a proposta apresentada, foi utilizada a ferramenta *Mininet* [48]. O *Mininet* é um emulador que permite a criação de redes virtuais com dispositivos, controladores *OpenFlow* e roteadores. Tal ferramenta foi escolhida por permitir a utilização de comandos *Unix* reais, numa forma de tornar o ambiente de testes mais similar à uma rede real. Para a implementação do controlador *OpenFlow*, foi escolhido o *Floodlight* [24], como explicado anteriormente. Nos experimentos, o controlador *OpenFlow* está na máquina física descrita pela Tabela 5.1, assim como a máquina virtual onde o *Mininet* é executado, um Ubuntu 14.04 LTS utilizando 1GB de memória RAM e até 100% de um núcleo da máquina física. A versão do *Floodlight* utilizada é a 0.90, e o *Mininet* está em sua versão 2.1.0.

Tabela 5.1: Ambiente de *hardware* e *software* utilizado nos experimentos.

Processador	Intel Core i7 2.3GHz
Sistema Operacional (SO)	Mac OS X 10.10.5
Arquitetura do SO	x86_64
Memória RAM	16 GB 1600 MHz DDR3
Número de Núcleos	8

Foram implementados dois cenários. No primeiro cenário, ilustrado na Figura 5.2, o dispositivo **A** se comunica com o dispositivo **B** e, num dado momento, **A** perde sua conexão com a rede, se reconectando após um dado tempo, mas com um endereço IP diferente. Neste cenário, há apenas um roteador, conectado à um controlador *OpenFlow*. Este é um cenário simples e sem tráfego concorrente, que é utilizado para checar a viabilidade da proposta e servir de base para um cenário mais complexo.

O segundo cenário, cujo mapa de rede está representado na Figura 5.3, se dá numa topologia real, obtida através do *Rocketfuel* [68]. O *Rocketfuel* é um motor de mapeamento que utiliza informações de roteamento de vários pontos estratégicos, além de convenções de nomes de roteadores para inferir topologias de ISPs. Também é utilizado um algoritmo para encontrar endereços IP referentes à interfaces diferentes de um mesmo roteador. A topologia utilizada possui 200 nós e 30 roteadores, e todos estão conectados entre si de alguma forma. Os roteadores estão conectados a um controlador *Openflow*.

Como o *Rocketfuel* fornece apenas a topologia, sem informações de vazão, pesos ou

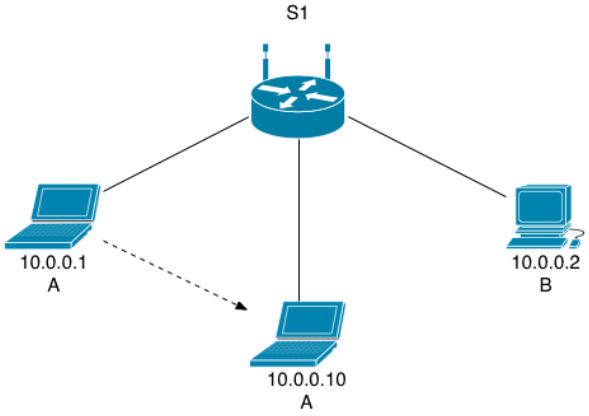

Figura 5.2: Cenário de teste 1: Dispositivo no mesmo ponto de conexão mas com endereço diferente ao se reconectar

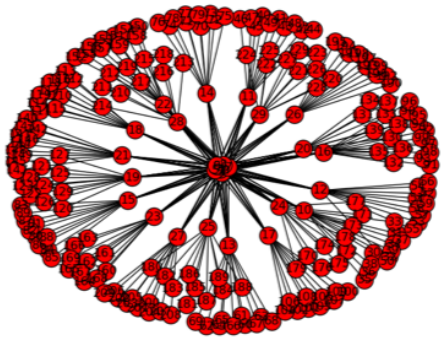

Figura 5.3: Cenário de teste 2: Topologia real

tráfego, foi utilizado o *Fast Network Simulation Setup* (FNSS) [63] para a configuração dos nós e a importação da topologia para o *Mininet*. O FNSS é um conjunto de ferramentas que ajuda a simplificar a configuração de cenários de experimentos em redes. O FNSS é capaz de:

- Interpretar topologias de um conjunto de dados (como o *Rocketfuel*) ou um gerador de topologias;
- Configurar enlaces com capacidades, pesos e atrasos;
- Implantar aplicações e pilhas de protocolos em nós;
- Gerar tráfego na rede;
- Gerar agendas de eventos na rede;

- Implantar a topologia de rede em vários simuladores e emuladores.

Utilizou-se o FNSS para configurar os enlaces da topologia obtida através do *Rocketfuel* de acordo com os padrões 802.11n [75], ilustrado na Tabela 5.2, e exportá-la para o *Mininet*. Neste cenário, também foram separados dois nós **A** e **B** que irão se comunicar entre si, e em dado momento **A** irá se mover, alterando seu endereço IP. Ao contrário do primeiro cenário, outros nós podem se comunicar entre si, gerando tráfego concorrente, e assim uma maior carga na rede.

Tabela 5.2: Especificações do padrão 802.11n

Banda dos canais	40MHz
Frequência do sinal	2.4GHz ou 5GHz
Taxa de Dados	150Mbps ou 17,88MB/s

Para verificar se o módulo proposto causa algum impacto na rede sob a perspectiva do usuário, foram coletadas medidas relacionadas ao atraso fim-a-fim, vazão e taxa de perda de pacotes, as medidas mais percebidas por usuários[36]. Os resultados do Cenário 2 ajudarão a avaliar a escalabilidade da proposta, devido à introdução de outros nós e de tráfego concorrente. Foram avaliados os protocolos ICMP, UDP e TCP. Será chamado de Período A o período antes da mobilidade, e Período B será o período após a mobilidade. O Momento de Reconexão será o instante em que o nó sofre mobilidade, se desconectando e reconectando à rede.

Foram escolhidos os seguintes tipos de tráfego para os experimentos:

- Teste de latência com pacotes ICMP entre os dispositivos **A** e **B**. A envia um pacote ICMP *Echo Request* por segundo para B e calcula-se o *Round Trip Time*;
- Teste de perda com pacotes ICMP entre os dispositivos **A** e **B**, em relação ao intervalo entre pacotes. A envia pacotes ICMP *Echo Request* para B num intervalo que pode ir de 0.1 segundo até 2.0 segundos, e calculado o número de pacotes perdidos durante a reconexão de **A**;
- Teste de envio de áudio PCM/UDP. Os dispositivos **A** e **B** iniciam uma conexão de áudio PCM por 120 segundos, um fluxo RTP utilizando o codec G.711 [30], com um intervalo entre pacotes de 30ms, consumindo uma banda de 67 kbits/s. Cada nó respeita o tempo de fala do outro, ou seja, enquanto um nó envia dados, outro apenas recebe. Tal teste calculará a variação do atraso (*jitter*) e a taxa de perda durante a simulação;

- *Streaming* de vídeo via HTTP. Um arquivo no formato MP4 de 50MB em resolução 720p é disponibilizado na rede, e o nó **A** consome o vídeo. Neste teste será calculada a vazão de uma conexão HTTP;

- *Download* de arquivo via TCP. Um nó será um servidor HTTP e o nó que sofrerá mobilidade irá requisitar um arquivo de 512MB. Será calculada, assim, a vazão desta conexão. Utilizou-se uma janela de 1 segundo para o cálculo da vazão.

Foram efetuadas 20 rodadas de testes para cada tipo de tráfego nos dois cenários. No teste de perda de pacotes ICMP, foram efetuadas 100 rodadas por ser mais suscetível a itens externos (como filas de roteadores cheias). Além disso, em ambos os cenários o nó se conecta imediatamente, sem considerar o tempo de *handoff* da conexão, já que o objetivo destas avaliações era avaliar possíveis tempos de atraso na reconexão introduzidos pela proposta.

Capítulo 6

Resultados e Análise

Neste capítulo serão apresentados os resultados obtidos nos testes definidos no Capítulo 5. Para cada gráfico apresentado, foi considerada a média dos 20 testes e utilizado um intervalo de confiança de 95%. Os dados em azul são referentes ao experimento utilizando o módulo proposto, enquanto os dados em laranja indicam um experimento de controle na mesma topologia e com o mesmo tráfego, porém nestes testes o módulo não estava ativo e não ocorreu mobilidade de nenhum nó. Este experimento é utilizado como base de referência, pois quanto mais próximo o modelo proposto estiver desta base, menos perceptível será a mobilidade para os usuários.

6.1 Tráfego ICMP

Neste experimento, avalia-se o tempo entre o envio de um pacote e o recebimento da confirmação de recebimento deste (RTT), e a taxa de perda de pacotes ICMP.

Os Gráficos das Figuras 6.1 e 6.2 mostram os resultados para os testes ICMP (Ping) do primeiro cenário. Era esperado um aumento brusco do RTT no Momento de Reconexão, com uma diminuição logo após o restabelecimento da conexão, voltando rapidamente aos valores do Período A. Isso se daria devido ao tempo no qual o nó estaria desconectado, somado ao tempo que o módulo levaria para criar as regras e alterar os pacotes. Como a Figura 6.2 demonstra, os resultados seguiram o que era esperado, com um pico de 1,1s, em média, e uma queda brusca, se reestabilizando. Outro detalhe está no fato de, apesar de um atraso mais alto, nenhum pacote foi perdido por nenhum dos nós nesta bateria de testes, o que mostra que o módulo atua rapidamente a ponto de não romper a conexão. Em média, o RTT dos nós móveis segue o mesmo padrão dos nós que não se moveram durante o experimento.

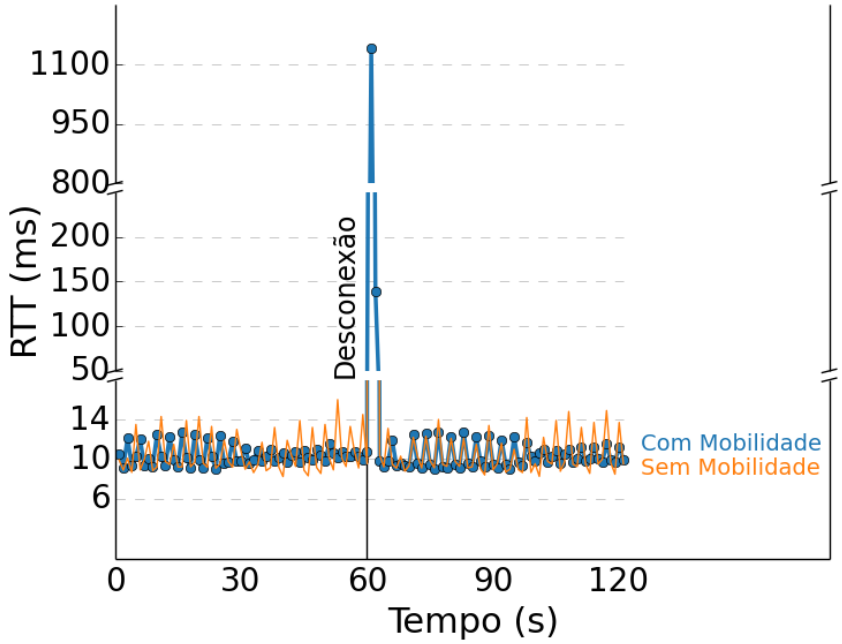

Figura 6.1: Cenário 1: Ping sem atraso na reconexão.

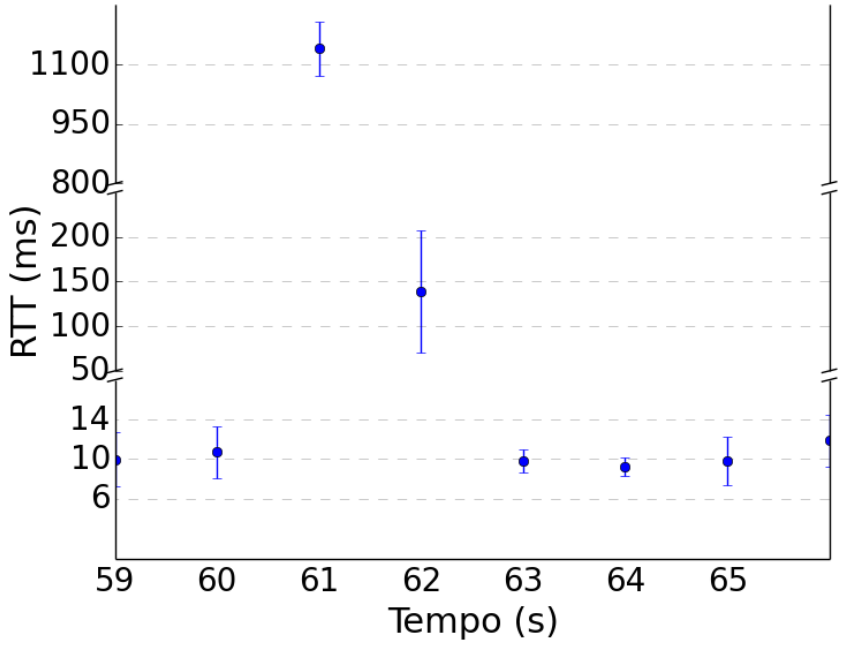

Figura 6.2: Cenário 1: Ping – Detalhe no Momento de Reconexão.

O próximo teste, cujos resultados se encontram na Figura 6.3, relaciona a quantidade de pacotes perdidos com a velocidade de envio de pacotes ICMP. Como se pode perceber, após um intervalo de 0,5s entre pacotes, a taxa de pacotes perdidos foi de 2,3% (lembrando que exclusivamente neste teste foram feitas 100 rodadas de teste para cada taxa de envio). Esta taxa de perda, associado ao fato de, no teste anterior, não ter ocorrido nenhuma perda de pacotes em todas as 20 rodadas, mostra que perdas de pacote podem acontecer, mas

são raras. Devido ao tempo necessário para a atualização do *Auspice* pelo controlador, elaboração das regras de fluxo novas e a instalação desta nos roteadores, era previsto uma taxa de perda inversamente proporcional ao intervalo entre pacotes, o que de fato ocorreu.

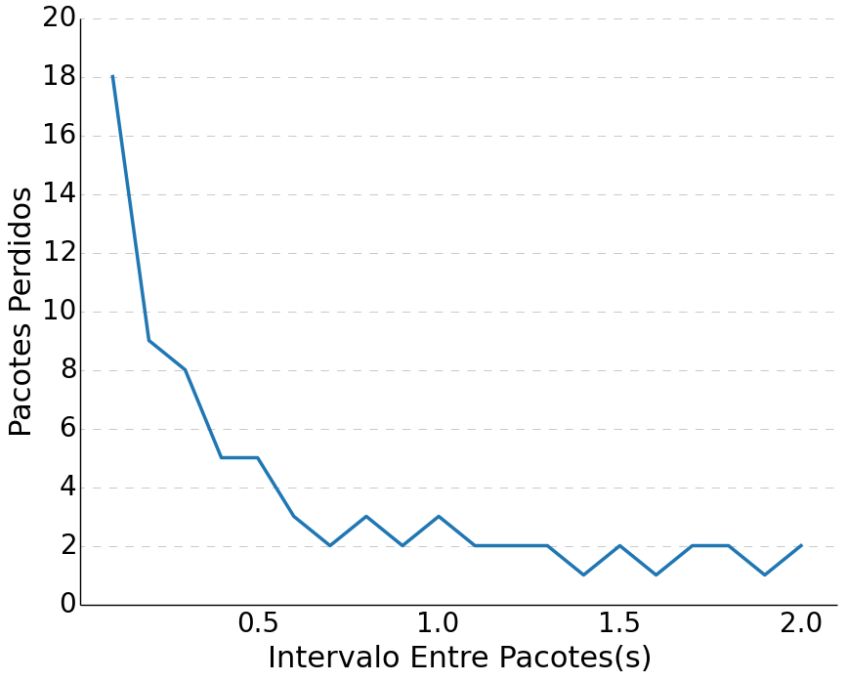

Figura 6.3: Cenário 1: Ping - Pacotes perdidos em 100 rodadas.

Os resultados para o teste ICMP do segundo cenário estão nas Figuras 6.4 e 6.5. Neste, o pico gerado após a reconexão foi ainda menor do que o obtido no cenário 1, sendo de apenas 150ms, em média. A reestabilização também ocorreu rapidamente, com o próximo pacote após o pico já estando nos níveis de antes da reconexão. Assim como no primeiro cenário, nenhum pacote foi perdido nesta bateria de testes.

6.2 *Download* de arquivo via HTTP

Neste teste, é testado se a mobilidade do usuário causa uma queda grande na vazão (ou até rompimento da conexão) do *Download* de um arquivo via HTTP.

O gráfico da Figura 6.6 mostra os resultados do teste de *Download* para o primeiro cenário. Esperávamos uma diminuição na taxa no Momento de Reconexão devido ao controle de congestionamento do TCP, com uma recuperação lenta porém constante da taxa de *Download*. Os resultados mostram intervalo de confiança muito alto, com uma recuperação relativamente rápida das taxas, em média. Atribuímos o elevado intervalo de confiança após a mobilidade ao controle de congestionamento do TCP, que ocorreram

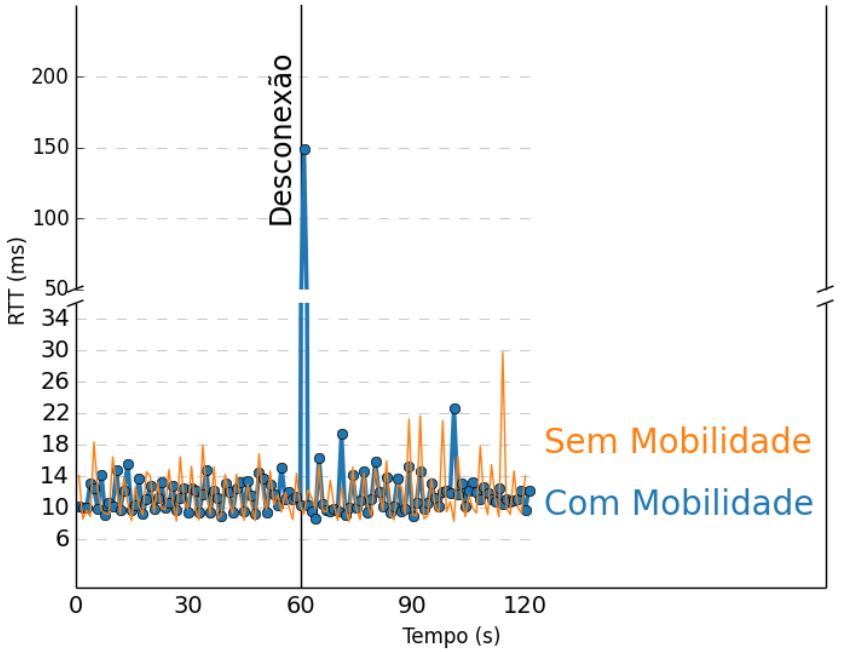

Figura 6.4: Cenário 2: Ping sem atraso na reconexão.

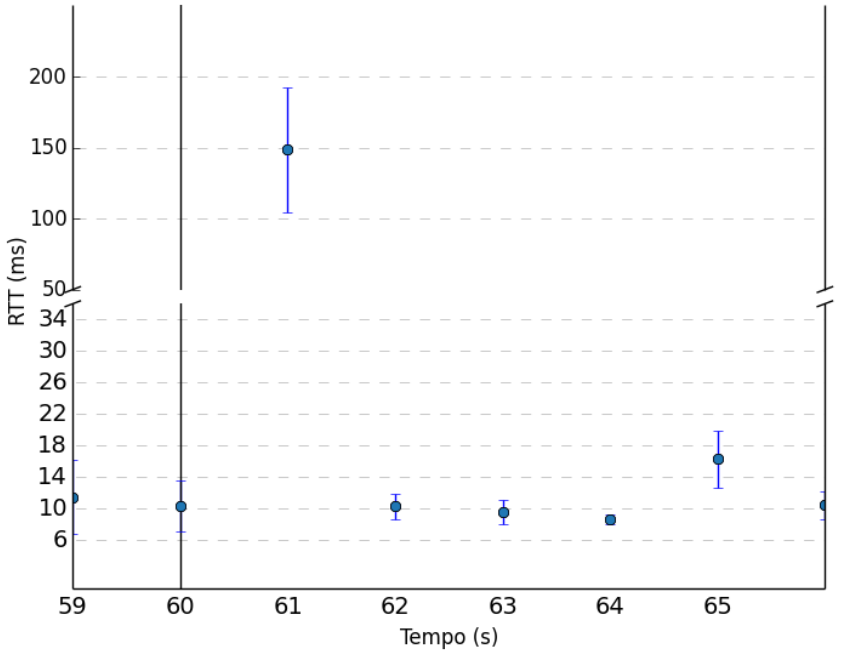

Figura 6.5: Cenário 2: Ping – Detalhe no Momento de Reconexão.

mesmo sem tráfego concorrente, como pode ser observado pelos resultados do experimento sem mobilidade.

A Figura 6.7 mostra os resultados do teste de *Download* de arquivos para o segundo cenário. Os resultados se mostraram parecidos ao esperado no que se refere ao Período B (recuperação lenta e constante até as taxas obtidas no Período A). Porém, o download se

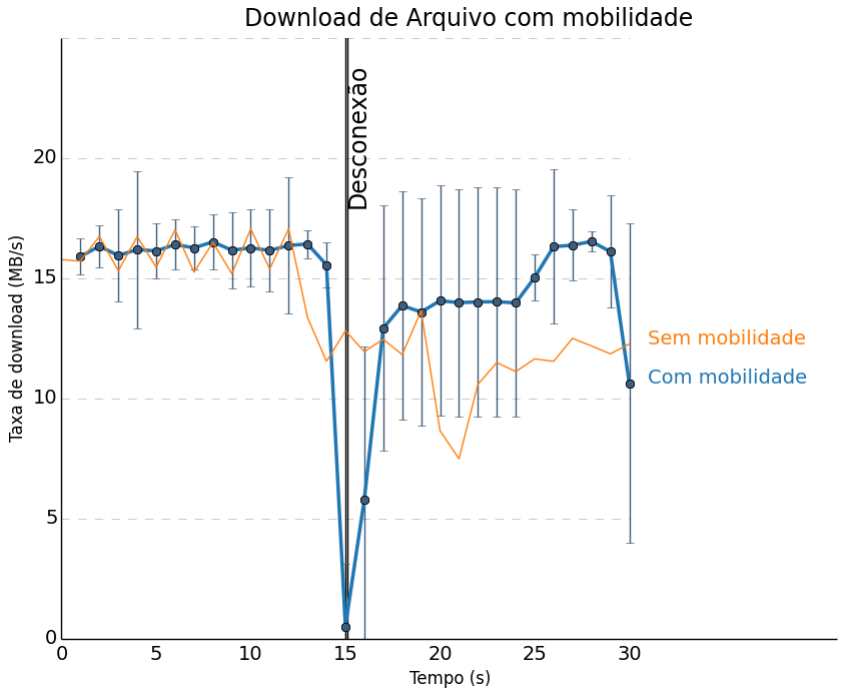

Figura 6.6: Cenário 1: Download de arquivo com mobilidade.

manteve ativo, não zerando no Momento de Reconexão, superando as expectativas.

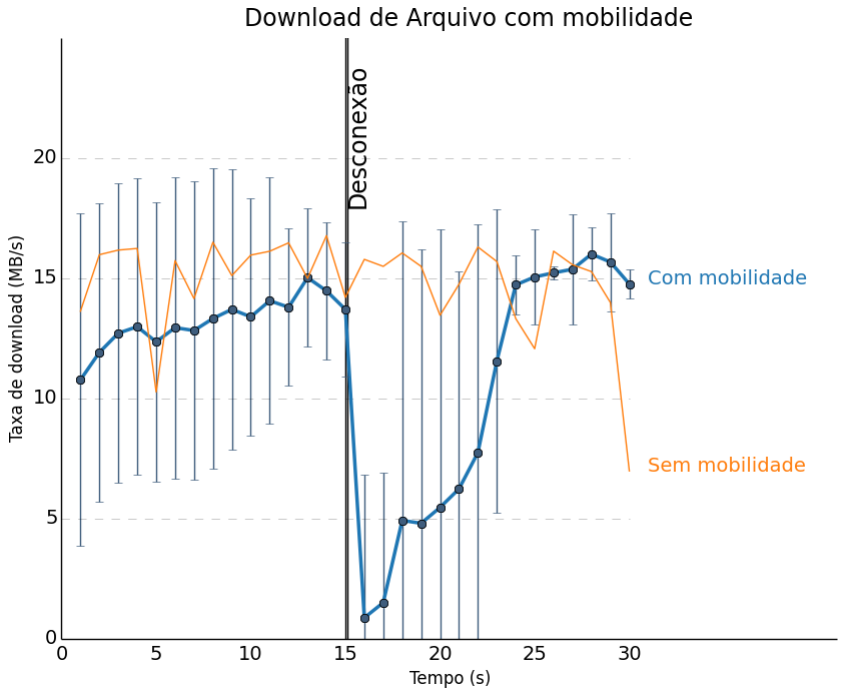

Figura 6.7: Cenário 2: Download de Arquivo.

6.3 Chamada de Áudio

Neste experimento, é avaliada a variação da latência (*Jitter*) médio de uma conexão de áudio entre dois nós. O *Jitter* é dado pela seguinte fórmula:

$$J_i = J_{i-1} + \frac{(d_i - d_{i-1}) - J_{i-1}}{16} \tag{6.1}$$

Ou seja, para estimar o *Jitter* J_i do pacote i, é calculado a diferença entre o tempo de trânsito do pacote i e do pacote *i-1*, subtraído do *Jitter* do pacote *i-1*, dividido por 16 (valor escolhido por fornecer uma boa taxa de redução de ruídos enquanto mantém uma taxa de convergência aceitável, segundo [7]) e adicionado ao valor do *Jitter* do pacote *i-1*. Esta é fórmula utilizada pelo protocolo RTP [27] para calcular o *Jitter* de seus pacotes.

Assim, por mais que latências altas ocorram, a média do *Jitter* será pequena pois a variação entre as latências é pequena. Utilizamos esta métrica para chamadas de áudio devido à natureza desta, pois não é necessário que o áudio chegue rapidamente ao destino, apenas que não cheguem pacotes com uma variação alta de entrega entre eles, causando áudio "picotado".

As Figuras 6.8 e 6.9 mostram os resultados para o teste de chamada de áudio no primeiro cenário, sob a perspectiva do nó móvel e estático, respectivamente. Era esperado, no Momento de Reconexão, um aumento semelhante ao obtido no teste de ICMP, com uma recuperação também semelhante para ambos os nós. Como os gráficos mostram, de fato o nó móvel teve um aumento no *Jitter* no Momento de Reconexão, rapidamente voltando a ter taxas parecidas com as do Período A. O nó estático teve uma variação pequena no Momento de Reconexão, não ultrapassando 1.1ms. O fato dos nós respeitarem o tempo de fala do outro, e a forma que os nós se comportaram no Momento de Reconexão mostram que a mobilidade do nó móvel se deu num momento em que o mesmo estava enviando informações em todas as rodadas.

Para o segundo cenário, a Figura 6.10 retrata os resultados para o teste de chamada de áudio. Foi gerado apenas o gráfico sob a perspectiva do nó móvel pois o nó estático poderia se conectar com outros nós, afetando significantemente os dados caso isso ocorresse. Os resultados foram satisfatórios, havendo uma alta significativa no Momento de Reconexão, mas menor do que alguns momentos sem mobilidade.

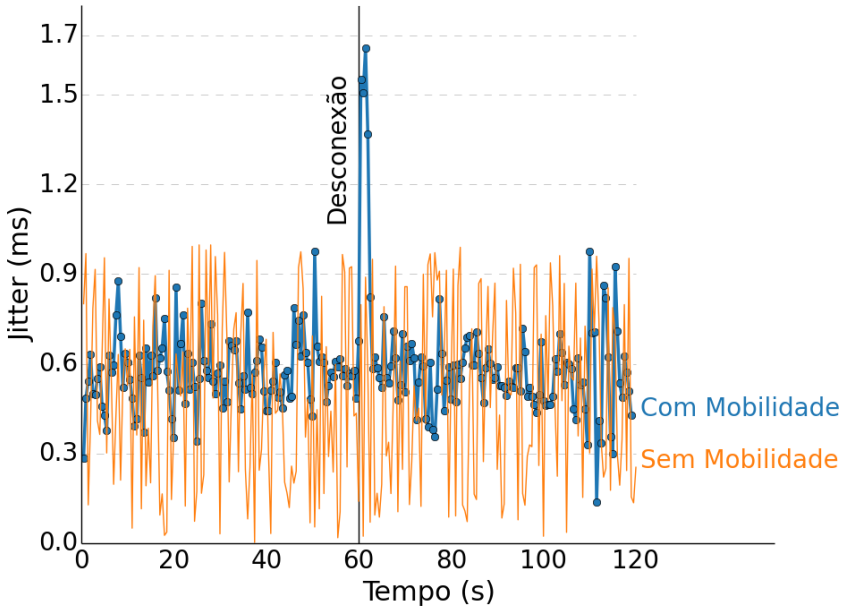

Figura 6.8: Cenário 1: Jitter Médio percebido pelo nó móvel.

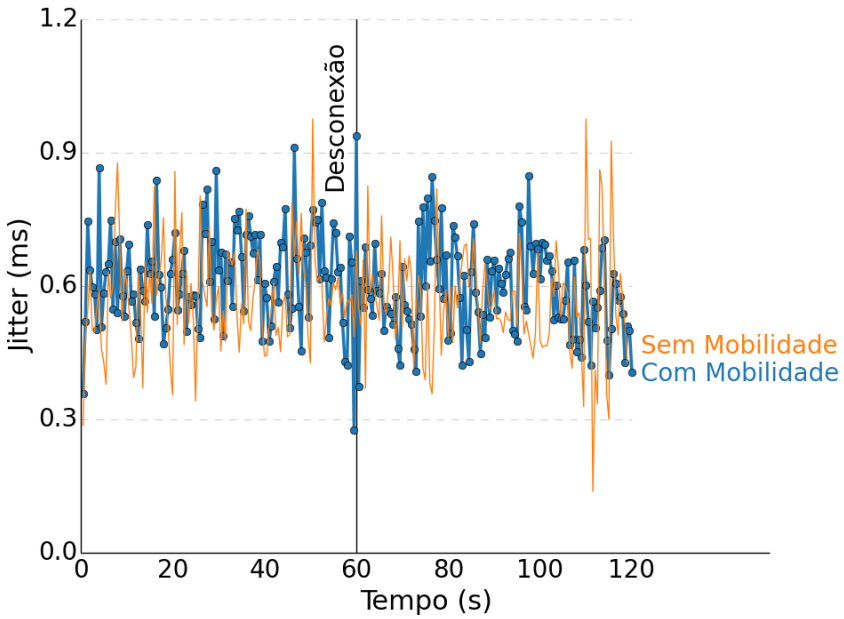

Figura 6.9: Cenário 1: Jitter Médio percebido pelo nó estático.

6.4 *Streaming* de Vídeo

As Figuras 6.11 e 6.12 mostram os resultados do experimento de *Streaming* de vídeo do primeiro cenário. Como o experimento foi feito utilizando o protocolo TCP, era esperado um resultado semelhante com o obtido no experimento de *Download* TCP. O Momento de Reconexão foi semelhante, tendo uma diminuição na taxa quando houve a mobilidade, e uma recuperação rápida, porém as taxas nos Momentos A e B foram diferentes, com intervalos de confiança gradativamente menores após a mobilidade. Como as taxas dos

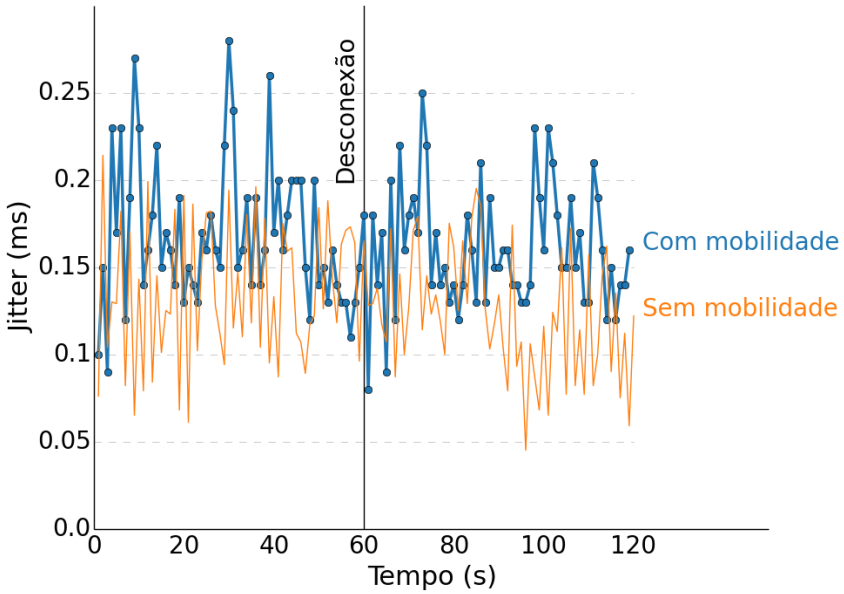

Figura 6.10: Cenário 2: Jitter médio percebido pelo nó móvel

experimentos sem mobilidade seguem o mesmo padrão, tal diferença é devida a pacotes perdidos em alguma rodada, causando uma queda na velocidade da conexão e, assim, o aumento no intervalo de confiança

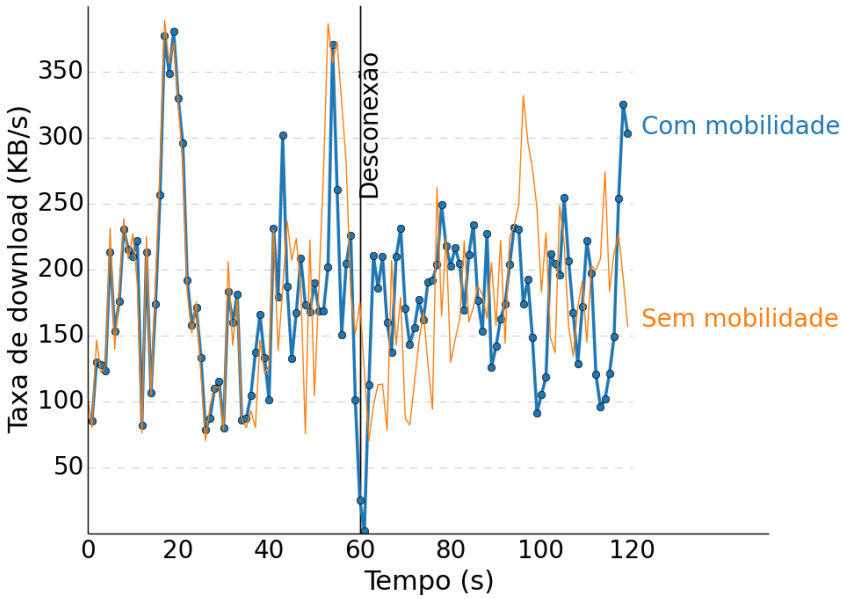

Figura 6.11: Cenário 1: Taxa de *Download* - Streaming de vídeo.

Por fim, as Figuras 6.13 e 6.14 mostram os resultados do experimento no segundo cenário. Enquanto não havia mobilidade, a média da vazão dos dois experimentos (com mobilidade e sem mobilidade) coincidiram, com uma pequena queda durante a reconexão.

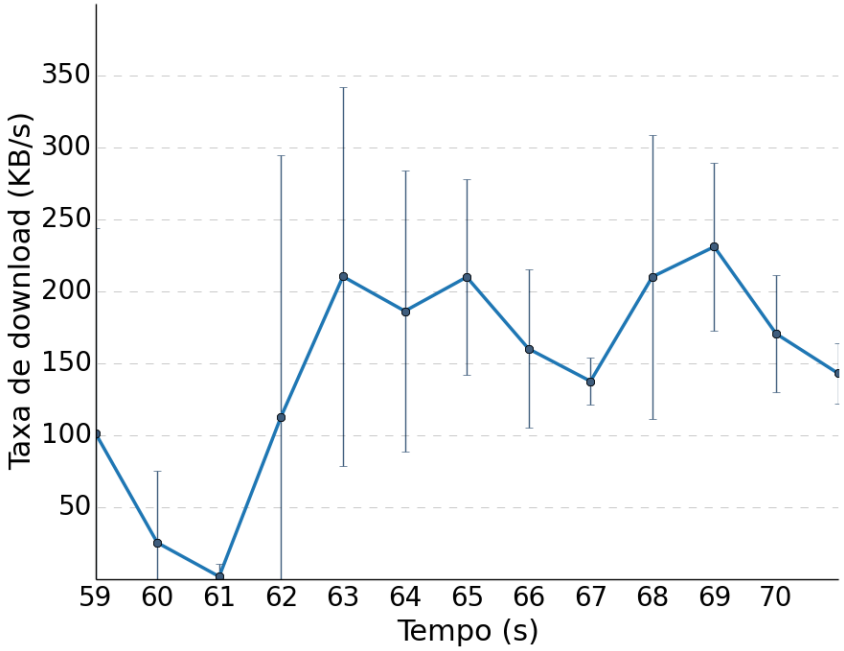

Figura 6.12: Cenário 1: Detalhe no momento de reconexão

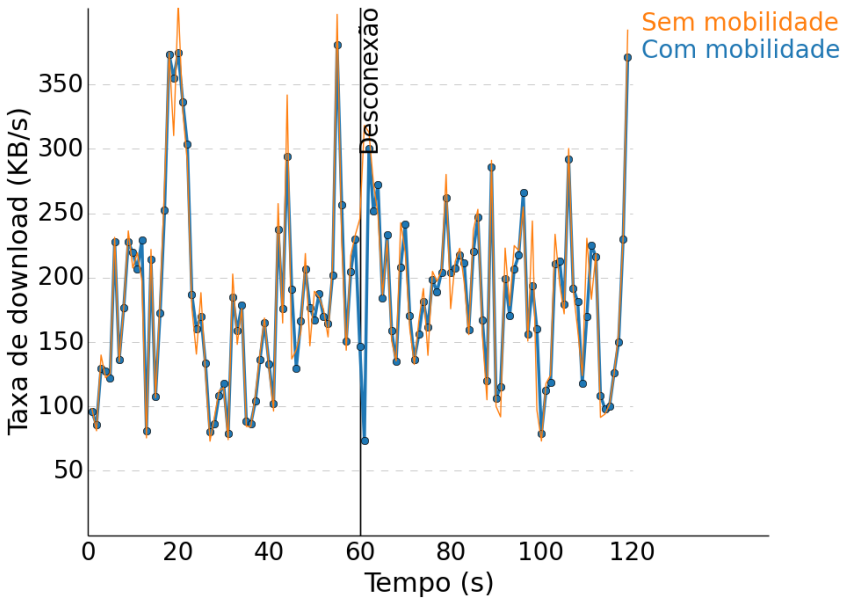

Figura 6.13: Cenário 2: Taxa de obtenção - Streaming de vídeo.

6.5 Análise dos Resultados

Os resultados dos testes são muito otimistas, mostrando que é possível reestabelecer a conexão de um nó após sua mobilidade sem alterações no núcleo da rede ou nos dispositivos finais. Porém, apesar da mobilidade ser realizada, alguns experimentos obtiveram uma variação grande entre as rodadas de teste, o que indica que o algoritmo utilizado pode ser otimizado. O fato de, em alguns testes, ocorrerem raras perdas de pacotes ou uma vazão

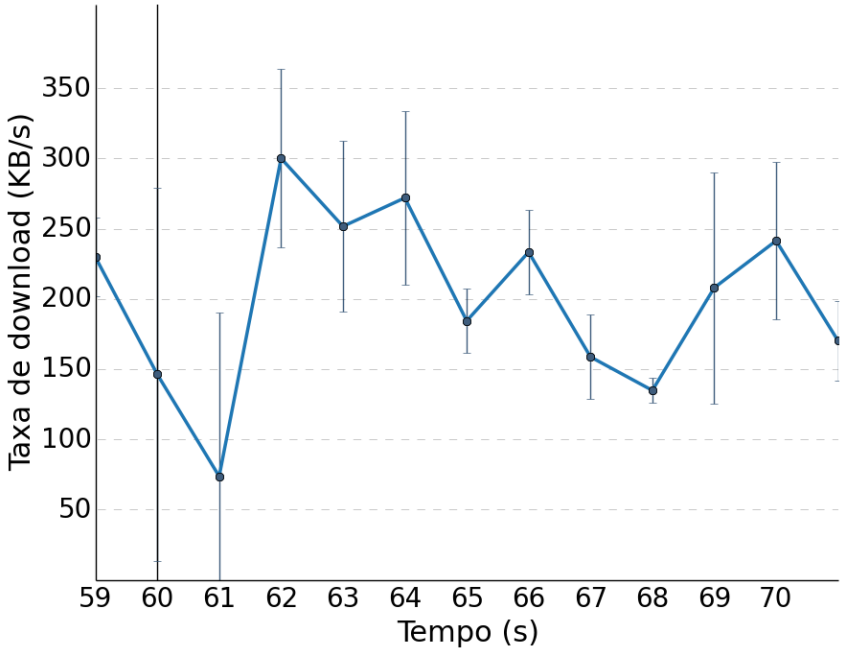

Figura 6.14: Cenário 2: Detalhe no momento de reconexão

baixa mostra a necessidade de guardar os pacotes que foram enviados durante a execução do algoritmo, para posterior entrega. Desta forma, nenhum pacote seria perdido, e assim, a vazão se manteria alta.

Percebe-se que, em ambos os cenários, houveram resultados semelhantes, o que leva a crer que o tráfego concorrente gerado no cenário 2, assim como a topologia mais robusta deste cenário, não resultou em impactos relevantes na implementação da proposta. Portanto, é possível concluir que a proposta deste trabalho é escalável em relação ao tamanho da rede regida por apenas um controlador. Com um bom algoritmo de sincronização de controladores, a proposta apresentada pode ser uma candidata para a implementação de mobilidade em arquiteturas de Internet do Futuro.

Capítulo 7

Conclusão e Trabalhos Futuros

Este trabalho aborda a questão da alta mobilidade de usuários e dispositivos entre redes e a incapacidade da Internet atual de lidar com tal mobilidade. Atualmente, ao mover-se entre redes, o custo do restabelecimento da conexão é atribuído às aplicações, o que gera um gasto maior de recursos e energia. Foram mostrados alguns trabalhos que visam retirar tal custo dos dispositivos, levando-os à borda ou ao núcleo da rede.

Também foi apresentada a proposta, implementação e avaliação de suporte à mobilidade de usuários e dispositivos entre redes. A proposta utiliza o Serviço Global de Nomes *Auspice* para guardar informações de localidade, e roteadores com *OpenFlow*, que podem acessar tais informações para atualizar suas informações de roteamento.

Por causa das limitações mostradas na Seção 4.2, a proposta deste trabalho ainda não é escalável em relação à quantidade de controladores e está suscetível a ataques de negação de serviço. Os trabalhos futuros estão na implantação da arquitetura em ambientes reais e resolução das limitações, que são:

- Algoritmo de captura de pacotes para entrega posterior (após a mobilidade);
- Utilização de um identificador único por dispositivo, e não por interface de rede (endereços MAC);
- Escalabilidade da solução, sincronizando os roteadores com o algoritmo de *Paxos* [38];
- Resolução de problemas de segurança, não permitindo que qualquer controlador mude informações de qualquer nó no *Auspice*.

Referências

[1] Abowd, G. D.; Dey, A. K.; Brown, P. J.; Davies, N.; Smith, M.; Steggles, P. Towards a better understanding of context and context-awareness. In *Handheld and ubiquitous computing* (1999), Springer, pp. 304–307.

[2] Åhlund, C.; Wallin, S.; Andersson, K.; Brännström, R. A service level model and Internet mobility monitor. *Telecommunication Systems 37*, 1-3 (2008), 49–70.

[3] Arkko, J.; Johnson, D. B.; Perkins, C. E. Mobility support in IPv6.

[4] Baucke, S. Flexible architecture for the Future Internet. *7th Workshop ITG-5.2 "Future of Networks"* (2007).

[5] Braun, W.; Menth, M. Software-Defined Networking using OpenFlow: Protocols, applications and architectural design choices. *Future Internet 6*, 2 (2014), 302–336.

[6] Bronzino, F.; Nagaraja, K.; Seskar, I.; Raychaudhuri, D. Network service abstractions for a mobility-centric Future Internet architecture. *Proceedings of the eighth ACM international workshop on Mobility in the evolving internet architecture* (October 2013), 5–10.

[7] Cadzow, J. A. *Foundations of Digital Signal Processing and Time Series Analysis*. Prentice Hall PTR, 1987.

[8] Cai, Z. *Maestro: achieving scalability and coordination in centralizaed network control plane*. Tese de Doutorado, Rice University, 2012.

[9] Canalys. Canalys. Disponível em: `http://www.canalys.com/newsroom/smart-phones-overtake-client-pcs-2011`, 2011. Acessado em 17/09/2013.

[10] Cheshire, S.; Baker, M. Internet mobility 4×4. In *ACM SIGCOMM Computer Communication Review* (1996), vol. 26, ACM, pp. 318–329.

[11] CIA. Central Intelligence Agency - The World Factbook. Disponível em: `https://www.cia.gov/library/publications/the-world-factbook/geos/xx.html`, 2015. Acessado em 11/06/2015.

[12] Cleary, F.; Fiedler, M.; Ridel, L.; Toker, A. C.; Yavuz, B. PERIMETER: Privacy-preserving contract-less, user centric, seamless roaming for always best connected Future Internet. *22nd Wireless World Research Forum Meeting* (2009).

[13] Cohen, B. The bittorrent protocol specification, 2008.

[14] CORDIS. Community research and development information service. Disponível em: http://cordis.europa.eu/fp7/ict/, 2013. Acessado em 23/11/2013.

[15] de Brito, G. M.; Velloso, P. B.; Moraes, I. M. Redes orientadas a conteúdo: Um novo paradigma para a internet. *Minicursos do Simpósio Brasileiro de Redes de Computadores-SBRC 2012* (2012), 211–264.

[16] de Brito, G. M.; Velloso, P. B.; Moraes, I. M. Uma análise do desempenho de redes sem-fio orientadas a conteúdo. *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)* (2014).

[17] de Meer, H.; Hummel, K. A.; Basmadjian, R. Future internet services and architectures: trends and visions. *Telecommunication Systems* (2012), 1–2.

[18] Devarapalli, V.; Wakikawa, R.; Petrescu, A.; Thubert, P. Network mobility (NEMO) basic support protocol. Tech. rep., 2004.

[19] Dillon, E. PERIMETER: A user-centric approach to surfing the web on your mobile. *Waterford Institute of Technology - Research Matters Issue 11* (Spring 2010), 16–17.

[20] Dillon, E.; Power, G.; Cleary, F. Experimental testing in the Future Internet PERIMETER project. *FIS'10 Proceedings of the Third Future Internet conference on Future internet* (2010), 30–39.

[21] Dillon, E.; Power, G.; Ramos, M. O.; Rodríguez, M. A. C.; Argente, J. R.; Fiedler, M.; Tonesi, D. S. PERIMETER: A Quality of Experience framework.

[22] Erickson, D. The Beacon OpenFlow Controller. In *HotSDN* (2013), ACM.

[23] EugeneNg, Z. A. C. T. Maestro: Balancing fairness, latency and throughput in the openflow control plane. Tech. rep., Rice University, 2011.

[24] Floodlight. Project Floodlight – open source software for building Software-Defined Networks. Disponível em: http://www.projectfloodlight.org/floodlight/, oct 2015. Acessado em 24/11/2015.

[25] Gartner. Gartner research. Disponível em: http://www.gartner.com/newsroom/id/1759714, 2011. Acessado em 17/09/2013.

[26] Gosling, J. *The Java language specification*. Addison-Wesley Professional, 2000.

[27] Group, N. W., et al. RFC 1889-RTP: a transport protocol for real time applications, 1996.

[28] Gude, N.; Koponen, T.; Pettit, J.; Pfaff, B.; Casado, M.; McKeown, N.; Shenker, S. Nox: towards an operating system for networks. *ACM SIGCOMM Computer Communication Review 38*, 3 (2008), 105–110.

[29] ICANN. Internet Corporation for Assigned Names and Numbers. Disponível em: https://www.icann.org, 1998. Acessado em 02/02/2016.

[30] ITU-T Recommendation G.711. Pulse code modulation of voice frequencies.

[31] Jacobson, V.; Mosko, M.; Smetters, D.; Garcia-Luna-Aceves, J. Content-centric networking. *Whitepaper, Palo Alto Research Center* (2007), 2–4.

[32] Jacobson, V.; Smetters, D. K.; Thornton, J. D.; Plass, M. F.; Briggs, N. H.; Braynard, R. L. Networking named content. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies* (2009), ACM, pp. 1–12.

[33] Khondoker, R.; Zaalouk, A.; Marx, R.; Bayarou, K. Feature-based comparison and selection of software defined networking (sdn) controllers. In *Computer Applications and Information Systems (WCCAIS), 2014 World Congress on* (2014), IEEE, pp. 1–7.

[34] Koponen, T.; Chawla, M.; Chun, B.-G.; Ermolinskiy, A.; Kim, K. H.; Shenker, S.; Stoica, I. A data-oriented (and beyond) network architecture. In *ACM SIGCOMM Computer Communication Review* (2007), vol. 37, ACM, pp. 181–192.

[35] Krishnamoorthy, A. *Implementation and evaluation of the MobilityFirst protocol stack on software-defined network platforms*. Tese de Doutorado, Rutgers, The State University of New Jersey, 2013.

[36] Kuipers, F.; Kooij, R.; De Vleeschauwer, D.; Brunnström, K. *Techniques for measuring Quality of Experience*. Springer, 2010.

[37] Kurose, J. F.; Ross, K. W. *Redes de Computadores e a Internet: Uma abordagem top-down, 3ª Edição*. Editora Pearson, São Paulo-SP, 2006.

[38] Lamport, L. The part-time parliament. *ACM Transactions on Computer Systems (TOCS) 16*, 2 (1998), 133–169.

[39] Lara, A.; Ramamurthy, B.; Nagaraja, K.; Krishnamoorthy, A.; Raychaudhuri, D. Using openflow to provide cut-through switching in mobilityfirst. *Photonic Network Communications 28*, 2 (2014), 165–177.

[40] Leiner, B. M.; Cerf, V. G.; Clark, D. D.; Kahn, R. E.; Kleinrock, L.; Lynch, D. C.; Postel, J.; Roberts, L. G.; Wolff, S. S. The past and future history of the Internet. *Communications of the ACM 40*, 2 (1997), 102–108.

[41] Lella, A.; Lipsman, A. The US mobile app report. Acessado em 02/02/2016.

[42] Li, C.-S.; Liao, W. Software defined networks. *IEEE Communications Magazine 51*, 2 (2013), 113–113.

[43] Linksys. Internet speed today: The evolution of home Wi-Fi. Disponível em: `http://www.linksys.com/us/home-wifi-internet-speed-evolution/`, 2016. Acessado em 13/02/2016.

[44] Luo, Y.; Eymann, J.; Timm-Giel, A. Mobility support for content centric networking. *Telecommunication Systems 59*, 2 (2015), 271–288.

[45] McKeown, N. Software-defined networking. *INFOCOM keynote talk 17*, 2 (2009), 30–32.

[46] McKeown, N.; Anderson, T.; Balakrishnan, H.; Parulkar, G.; Peterson, L.; Rexford, J.; Shenker, S.; Turner, J. (openflow): enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review 38*, 2 (2008), 69–74.

[47] Meyer, D.; Zhang, L.; Fall, K. Report from the IAB workshop on routing and addressing, 2007. RFC 4984.

[48] Mininet. Mininet: An instant virtual network on your laptop (or other PC). Disponível em: `http://mininet.org`. Acessado em 29/10/2015.

[49] National Science Foundation. NSF Future Internet Architecture Project. Disponível em: `http://www.nets-fia.net`, 2010. Acessado em 12/02/2016.

[50] Ng, E. Maestro: A system for scalable OpenFlow control. Tech. rep., TSEN Maestro-Technical Report TR10-08, Rice University, 2010.

[51] Open Networking Foundation. OpenFlow switch specification. *Version 1.3.4* (2014).

[52] Open Networking Foundation. Open Networking Foundation. Disponível em: `https://www.opennetworking.org/`, 2016. Acessado em 21/01/2016.

[53] Pallis, G.; Vakali, A. Insight and perspectives for content delivery networks. *Communications of the ACM 49*, 1 (2006), 101–106.

[54] Peng, G. Cdn: Content distribution network. *arXiv preprint cs/0411069* (2004).

[55] Perera, E.; Sivaraman, V.; Seneviratne, A. Survey on network mobility support. *ACM SIGMOBILE Mobile Computing and Communications Review 8*, 2 (2004), 7–19.

[56] Perkins, C. Mobile IP. *IEEE Communications Magazine* (1997).

[57] Perkins, C. IP mobility support for IPv4.

[58] Perkins, C.; Calhoun, P.; Bharatia, J. Mobile IPv4 challenge/response extensions (revised). Tech. rep., 2007.

[59] Plagemann, T.; Goebel, V.; Mauthe, A.; Mathy, L.; Turletti, T.; Urvoy-Keller, G. From content distribution networks to content networks—issues and challenges. *Computer Communications 29*, 5 (2005), 551–562.

[60] PlanetLab. Planet Lab. Disponível em: `https://www.planet-lab.org`, 2013. Acessado em 24/11/2013.

[61] Rabinovich, M.; Spatscheck, O. *Web caching and replication*. Addison-Wesley Boston, USA, 2002.

[62] Raychaudhuri, D.; Nagaraja, K.; Venkataramani, A. MobilityFirst: A robust and trustworthy mobility-centric architecture for the Future Internet. *ACM SIGMOBILE Mobile Computing and Communications Review* (2012), 2–13.

[63] Saino, L.; Cocora, C.; Pavlou, G. A toolchain for simplifying network simulation setup. In *Proceedings of the 6th International ICST Conference on Simulation Tools and Techniques* (ICST, Brussels, Belgium, Belgium, 2013), SIMUTOOLS '13, ICST.

[64] Shalimov, A.; Zuikov, D.; Zimarina, D.; Pashkov, V.; Smeliansky, R. Advanced study of SDN/OpenFlow controllers. In *Proceedings of the 9th Central & Eastern European Software Engineering Conference in Russia* (2013), ACM, p. 1.

[65] Sharma, A.; Tie, X.; Uppal, H.; Venkataramani, A.; Westbrook, D.; Yadav, A. A global name service for a highly mobile Internet.

[66] SmartInsights. Mobile marketing statistics compilation. Disponível em: `http://www.smartinsights.com/mobile-marketing/mobile-marketing-analytics/mobile-marketing-statistics/`, 2015. Acessado em 04/02/2016.

[67] Söllner, M.; Görg, C.; Pentikousis, K.; Lopez, J. M. C.; de Leon, M. P.; Bertin, P. Mobility scenarios for the Future Internet: The 4WARD approach. *Em 11th International Symposium on Wireless Personal Multimedia Communications (WPMC 2008)* (2008).

[68] Spring, N.; Mahajan, R.; Wetherall, D. Measuring isp topologies with rocketfuel. *SIGCOMM Comput. Commun. Rev. 32*, 4 (aug 2002), 133–145.

[69] Tanenbaum, A. *Rede de Computadores. 4ª Edição.* Editora Campus, 2003.

[70] Tanenbaum, A. S.; Van Steen, M. *Sistemas Distribuídos: Princípios e Paradigmas. 2ª Edição.* Editora Pearson Prentice Hall, 2007.

[71] Venkataramani, A.; Sharma, A.; Tie, X.; Uppal, H. Design requirements for a global name service for a mobility-centric, trustworthy Internetwork. *Proceedings of the 2013 Fifth International Conference on Communication Systems and Networks (COMSNETS)* (2013), 1–9.

[72] Wang, Y.; Bi, J.; Zhang, K. Design and implementation of a software-defined mobility architecture for IP networks. *Mobile Networks and Applications 20*, 1 (2015), 40–52.

[73] Watch, S. E. Mobile now exceeds PC: The biggest shift since the Internet began. Disponível em: `https://searchenginewatch.com/sew/opinion/2353616/mobile-now-exceeds-pc-the-biggest-shift-since-the-internet-began`, 2016. Acessado em 13/02/2016.

[74] Wedlund, E.; Schulzrinne, H. Mobility support using SIP. In *Proceedings of the 2nd ACM international workshop on Wireless mobile multimedia* (1999), ACM, pp. 76–82.

[75] Xiao, Y. IEEE 802.11n: enhancements for higher throughput in wireless LANs. *Wireless Communications, IEEE 12*, 6 (Dec 2005), 82–91.

[76] Yadav, A.; Sharma, A.; Venkataramani, A.; Cecchet, E.; Westbrook, D. msocket: System support for developing seamlessly mobile, multipath, and middlebox-agnostic applications. *Proceedings of the UMass SCS Technical Report* (2014).

[77] Zhang, P.; Durresi, A.; Barolli, L. A survey of Internet mobility. In *Network-Based Information Systems, 2009. NBIS'09. International Conference on* (2009), IEEE, pp. 147–154.

[78] Zhang, Y.; Zhang, H.; Zhang, L. Kite: A mobility support scheme for NDN. In *Proceedings of the 1st international conference on Information-centric networking* (2014), ACM, pp. 179–180.

[79] Zorzi, M.; Gluhak, A.; Lange, S.; Bassi, A. From today's INTRAnet of things to a future INTERnet of things: A wireless- and mobility-related view. *IEEE Wireless Communications* (Dezembro 2010), 44–51.