

**Cleiber Marques da Silva**

***Mobilidade em Redes IP: Análise dos Protocolos  
MIPv6 e HMIPv6***

São José – SC

Agosto / 2008

**Cleiber Marques da Silva**

***Mobilidade em Redes IP: Análise dos Protocolos  
MIPv6 e HMIPv6***

Monografia apresentada à Coordenação do Curso Superior de Tecnologia em Sistemas de Telecomunicações do Centro Federal de Educação Tecnológica de Santa Catarina para a obtenção do diploma de Tecnólogo em Sistemas de Telecomunicações.

Orientador:

Prof. Dr. Eraldo Silveira e Silva

CURSO SUPERIOR DE TECNOLOGIA EM SISTEMAS DE TELECOMUNICAÇÕES  
CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE SANTA CATARINA

São José – SC

Agosto / 2008

Monografia sob o título “*Mobilidade em Redes IP: Análise dos Protocolos MIPv6 e HMIPv6*”, defendida por Cleiber Marques da Silva e aprovada em 03 de abril de 2008, em São José, Santa Catarina, pela banca examinadora assim constituída:

---

Prof. Dr. Eraldo Silveira e Silva  
Orientador

---

Prof. Dr. Nome de Um Membro da Banca  
CEFET / SC

---

Prof. Dr. Nome de Outro Membro da Banca  
Departamento de Automação e Sistemas - UFSC

*In a world without fences and walls  
who needs Windows and Gates.*

# *Agradecimentos*

Dedico meus sinceros agradecimentos àqueles que muito me ajudaram para concluir este trabalho. Com certeza essas pessoas tornaram a realização deste trabalho uma tarefa prazerosa.

Outras pessoas poderiam ser aqui citadas, bem como a elaboração de um texto de agradecimento mais completo e emotivo.

# ***Resumo***

Este documento apresenta um modelo em L<sup>A</sup>T<sub>E</sub>X para concepção de monografias seguindo as normas das ABNT. Para a concepção deste modelo optou-se pela utilização da classe *AbnTex* haja visto que esta implementou, de forma satisfatória, as muitas normas ABNT necessárias para a elaboração de um texto acadêmico.

# *Abstract*

In this work, we present a L<sup>A</sup>T<sub>E</sub>Xtemplate to help our students to create their final text, that is, their monograph or dissertation. A dissertation is a document that presents the author's research and findings and is submitted in support of candidature for a degree or professional qualification.

# *Sumário*

## **Lista de Figuras**

## **Lista de Tabelas**

<b>1</b>	<b>Introdução</b>	p. 11
1.1	Motivação . . . . .	p. 11
1.2	Organização do texto . . . . .	p. 11
1.3	Objetivos . . . . .	p. 11
<b>2</b>	<b>Visão geral do protocolo MIPv6</b>	p. 12
2.1	Introdução . . . . .	p. 12
2.2	Terminologia do MIPv6 . . . . .	p. 13
2.3	Funcionamento . . . . .	p. 13
2.4	<i>Handover</i> . . . . .	p. 15
2.5	Envio de <i>Binding Updates</i> . . . . .	p. 16
2.6	Considerações sobre Segurança . . . . .	p. 17
2.6.1	<i>Return Routability Procedure</i> . . . . .	p. 17
<b>3</b>	<b>Plataforma de Testes</b>	p. 19
3.1	UML . . . . .	p. 19
3.2	Kernel . . . . .	p. 20
3.3	Preparando o sistema de arquivos . . . . .	p. 23
3.4	MIPL . . . . .	p. 25



3.4.1	Configurando MIPL . . . . .	p. 26
3.5	RADVD . . . . .	p. 27
3.5.1	Configurando RADVD . . . . .	p. 27
3.6	Gerador de tráfego . . . . .	p. 27
<b>4</b>	<b>Cenários de Testes</b>	p. 29
4.1	Cenário 1 . . . . .	p. 29
4.1.1	Resultados . . . . .	p. 32
4.1.2	Tempo de latência do <i>Handover</i> . . . . .	p. 33
4.2	Cenário 2 . . . . .	p. 36
4.3	Teste de desempenho do MIPv6 . . . . .	p. 38
<b>5</b>	<b>Estudo do código do MIPL</b>	p. 41
<b>6</b>	<b>Visão geral do protocolo HMIPv6</b>	p. 42
6.1	Terminologia do HMIPv6 . . . . .	p. 43
6.2	Funcionamento . . . . .	p. 43
6.3	Envio de <i>Binding Updates</i> . . . . .	p. 45
6.4	Descoberta de um MAP . . . . .	p. 46
<b>7</b>	<b>Conclusões</b>	p. 47
<b>Anexo A – Gerador de tráfego</b>		p. 48
<b>Anexo B – Cenários</b>		p. 53
<b>Referências Bibliográficas</b>		p. 60

## *Lista de Figuras*

2.1	Modos de operação simplificados do MIPv6 . . . . .	p. 15
2.2	Mensagens MIPv6 . . . . .	p. 18
4.1	Topologia do Cenário 1 . . . . .	p. 31
4.2	Diagrama de sequência do cenário 1 . . . . .	p. 37
4.3	Topologia do Cenário 2 . . . . .	p. 38
4.4	Taxa de transmissão em <i>Handover</i> . . . . .	p. 39
4.5	Gráfico: Pacotes Perdidos x Tempo Radvd . . . . .	p. 40
6.1	Mobilidade Local . . . . .	p. 44
6.2	Mobilidade Global . . . . .	p. 45

## *Lista de Tabelas*

4.1	Endereços do Cenário 1 . . . . .	p. 30
4.2	Tabela de roteamento do nó móvel no Cenário 1 . . . . .	p. 32
4.3	Tabela de roteamento do agente domiciliar no Cenário 1 . . . . .	p. 33
4.4	Mensagens do Cenário 1 com Tunelamento Bidirecional . . . . .	p. 34
4.5	Mensagens do Cenário 1 com Otimização de Roteamento . . . . .	p. 35

# ***1    Introdução***

Neste capítulo serão introduzidos todos os assuntos abordados por este documento. Pretende-se apresentar a motivação, os objetivos e a organização do texto.

## **1.1   Motivação**

A motivação deste documento foi a necessidade da elaboração de modelo para a concepção de monografias para o CEFET/SC.

## **1.2   Organização do texto**

## **1.3   Objetivos**

## 2 *Visão geral do protocolo MIPv6*

### 2.1 Introdução

Na internet, cada pacote associado a um fluxo de pacotes entre pontos comunicantes, é encaminhado em função do endereço de IP destino. Cabe ao protocolo de camada de rede chamado *Internet Protocol* (IP), escolher qual caminho o pacote deve seguir para chegar ao seu destinatário, em redes muito complexas como a *Internet* alguns protocolos de roteamento auxiliam na descoberta da localização dos destinos dinamicamente. Alguns exemplos de protocolos de roteamento são o *Open Shortest Path First* (OSPF), *Routing Information Protocol* (RIP) e *Border Gateway Protocol* (BGP).

Os endereços IP acabam definindo a localização geográfica de um ponto de acesso, os protocolos atuais da *Internet* assumem que o nó não muda seu endereço IP durante uma comunicação ou seja não troca o seu ponto de acesso.

Caso um nó mude seu ponto de acesso na *Internet*, normalmente ele deve reconfigurar um novo endereço IP e um roteador padrão, se uma comunicação estiver estabelecida quando o nó efetuar a mobilidade os protocolos de roteamento não serão mais capazes de entregar os pacotes corretamente. Para continuar sua comunicação sua nova rota deve ser propagada para toda estrutura de roteamento da rede, obviamente esta alternativa é inaceitável, pois seria inviável fazer isso em uma rede como a *Internet*.

Com a intenção de permitir que nós possam se movimentar para diferentes sub-redes e continuem sua comunicação, foi proposto o protocolo IP Móvel que garante que os pacotes sejam roteados para os nós móveis. Existem duas variações para o IP Móvel uma para o IPv4 e outra para o IPv6.

O foco deste trabalho será a versão para o IPv6, pois o IPv6 possui algumas vantagens sobre o IPv4 como maior número de endereços, suporte nativo para segurança, possibilidade de auto-configuração e suporte ao IP Móvel.

O protocolo móvel IPv6 (MIPv6) tem por objetivo permitir que nós IPv6 se desloquem entre sub-redes, com diferentes tecnologias de acesso como *Ethernet* e *Wireless LAN* e continuem suas comunicações. Sem suporte ao MIPv6 todos os pacotes destinados ao nó móvel quando ele estiver fora de sua rede origem serão perdidos.

O protocolo possibilita ao nó móvel comunicar-se com outros nós, móveis ou estáticos, após mudar seu ponto de conexão e ser alcançado pelo mesmo endereço IP, tornando a mobilidade transparente para as camadas superiores a de rede.

## 2.2 Terminologia do MIPv6

Alguns termos definidos pelo MIPv6:

**Nó móvel:** é o terminal de rede que pode trocar seu ponto de acesso a internet sem deixar de ser alcançável via seu endereço domiciliar.

**Endereço domiciliar:** é o endereço estático associado ao nó móvel na rede domiciliar.

**Link domiciliar:** *link* onde o prefixo de subrede do nó movel é definido.

**Link externo:** algum *link* que não seja *link* domiciliar do nó móvel.

**Care-of-address:** endereço IP dado ao nó móvel quando estiver em um *link* externo, o prefixo da rede deste endereço é o da rede externa.

**Nó correspondente:** é o nó com quem o nó móvel esta se correspondendo, pode ser móvel ou fixo.

**Agente domiciliar:** é um roteador no *link* domiciliar com quem o nó móvel registra seu *care-of-address* sempre que esta em um *link* externo.

**Movimento:** é uma troca de ponto de acesso na Internet.

**Binding:** é associação do endereço domiciliar do nó móvel com seu *care-of-address*.

## 2.3 Funcionamento

Todo o nó móvel está sempre acessível pelo mesmo endereço, independente de seu ponto de conexão com a *Internet*, este endereço é chamado de endereço domiciliar e representa o endereço IPv6 fixo do nó móvel na sua rede de origem.

Enquanto um nó móvel estiver fora de sua rede de origem ele terá no mínimo três endereços atribuídos a sua interface de rede, seu endereço domiciliar sempre permanente, um segundo referente ao enlace local que permite que nós presentes em um mesmo enlace se comuniquem sem roteadores. E um terceiro chamado de *care-of-address* que se refere a rede visitada, a cada nova o rede será configurado um novo *care-of-address*.

O *Care-of-address* pode ser obtido pelas formas convencionais do IPv6 por *stateless auto-configuration*, no qual o endereço é gerado por meio de informações divulgadas pelos roteadores e dos endereços disponíveis localmente, ou por *stateful autoconfiguration* onde o endereço e outros parâmetros de configuração é provido diretamente de um servidor, por exemplo os mecanismos DHCPv6 e PPPv6.

Na rede visitada após detectar o movimento e configurar seu *care-of-address*, o nó móvel envia uma mensagem icmp de *Binding Update* para seu agente domiciliar, e este faz a associação de seu endereço domiciliar com o seu *care-of-address*. O agente domiciliar registra a mensagem em seu *cache* e em resposta envia um *Binding Acknowledgement*. Então passa a atuar como um *proxy* interceptando todos os pacotes com destino ao nó móvel e enviando-lhes via túnel. O nó móvel pode informar também ao nó correspondente sua localização.

Duas formas de comunicação podem ser feitas entre o nó móvel e o correspondente. O primeiro modo com tunelamento bidirecional onde todos os pacotes são encaminhados via agente domiciliar, neste caso os pacotes com origem do nó correspondente são roteados para o agente domiciliar e este envia ao nó móvel via túnel. Os pacotes com destino ao nó correspondente são enviados via túnel ao agente domiciliar (túnel reverso) e estes são roteados normalmente da rede domiciliar para o nó correspondente. Neste modo o agente domiciliar utiliza *proxy Neighbor Discovery* para interceptar os pacotes com destino ao nó móvel. Podemos notar que neste modo o nó correspondente não precisa ter suporte ao MIPv6.

O segundo modo mais eficiente, com otimização de roteamento, o nó móvel também faz uma associação do seu *care-of-address* com o nó correspondente e este pode começar endereçar os pacotes diretamente ao *care-of-address*, eliminando assim o congestionamento e possíveis falhas no *link* entre o nó móvel e o agente domiciliar.

A figura 2.3 permite observar de forma simplificada os dois modos de operação do protocolo MIPv6.

Enquanto estiver em uma rede visitada o nó móvel terá mais de um endereço configurado em sua interface, o endereço domiciliar e um ou mais *care-of-address*, ele pode usar qualquer um desses endereços para se comunicar. Com o intuito de manter a transparência para as camadas

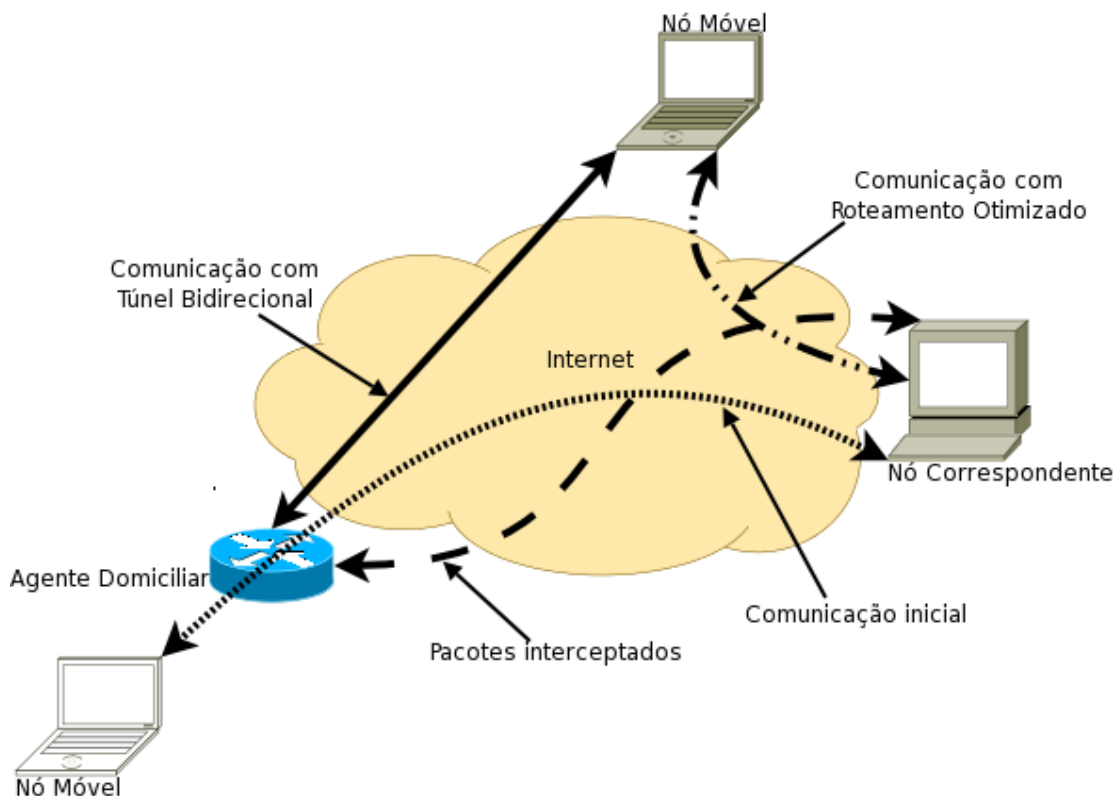


Figura 2.1: Modos de operação simplificados do MIPv6

superiores a de rede geralmente o nó móvel irá utilizar seu endereço domiciliar.

Ao enviar estes pacotes, eles deverão ser modificados inserindo o *care-of-address* no campo endereço de origem e movendo o endereço domiciliar para o campo *home address*. No receptor do pacote estas alterações devem ser revertidas para manter a transparência para as camadas superiores.

## 2.4 Handover

Uma das mais complicadas tarefas em mobilidade é o handover e a detecção do movimento do nó móvel por meio do protocolo MIPv6. O processo de *handover* pode ser dividido em dois tipos, um efetuado pela camada enlace referente a descoberta e conexão há uma nova rede. Outro conhecido como *handover* na camada rede efetuado pelo MIPv6 e se refere a descoberta do novo roteador, auto configuração do *care-of-address*, teste de duplicidade do *care-of-address* (DAD) e o registro com agente domiciliar e o nó correspondente.

Para realizar a tarefa de detecção de movimento o MIPv6 utiliza-se de facilidades do protocolo IPv6 *Neighbor Discovery*, incluindo *Router Discovery* e *Neighbor Unreachability Detection*.



Com o protocolo *Neighbor Unreachability Detection* é possível perceber que o roteador padrão não está mais alcançável, neste caso o nó deve descobrir um novo roteador padrão.

E o *Router Discovery* é o protocolo que permite que nós IPv6 descubram roteadores existentes no seu enlace, através das mensagens *Router Advertisement* e *Router Solicitation*. Um roteador IPv6 periodicamente envia mensagens de *Router Advertisement* para todo o enlace, desta forma os nós podem configurar seu endereço de rede e roteadores padrão. O nó também pode enviar uma mensagem de *Router Solicitation* recebendo como resposta um *Router Advertisement*.

Desta forma o nó móvel pode perceber o movimento quando receber um *Router Advertisement* de um novo roteador ou quando perceber que seu roteador não está mais alcançável, então deve requisitar um novo. Por meio das mensagens trocadas na descoberta do novo roteador o nó móvel é capaz de gerar seu *care-of-address*, então o teste de duplicidade deve ser feito para verificar se não há um endereço igual no *link*, com sucesso o registro com o agente domiciliar e o nó correspondente deve ser efetuado finalizando o processo de *handover*.

## 2.5 Envio de Binding Updates

Ao detectar que não está em sua rede domiciliar o nó móvel deve enviar um *Binding Update* ao seu home agente para informar o seu *care-of-address*. No pacote de *Binding Update* os seguintes requisitos devem ser seguidos:

- bit H (*home registration*) deve estar ligado;
- bit A (*acknowledge*) deve estar ligado;
- o pacote deve ter a opção de agente domiciliar preenchida;
- endereço de origem do pacote deve ser o *care-of-address* a ser registrado, a menos que a sub-opção de *care-of-address* alternativo esteja sendo utilizada;
- o tempo de vida do *binding* deve ser menor ou igual ao do *care-of-address*.

Quando o nó móvel retornar a sua rede de origem deve enviar um *Binding Update* para o seu agente domiciliar, para que este pare de interceptar e tunelar os pacotes destinados a ele.

As mensagens de *Binding Updates* trocadas nos dois modos de operação do MIPv6 podem ser observadas na figura 2.6.1.

## 2.6 Considerações sobre Segurança

O uso de *Binding Updates* não autenticados é tido como um sério problema de segurança, pois, um nó mal intencionado pode forjar um registro com o nó correspondente e passar a receber os pacotes destinados ao nó móvel, com a intenção de solucionar este problema de segurança o protocolo MIPv6 implementa alguns mecanismos.

Para dar mais proteção no processo de *Binding Update* com o agente domiciliário pode-se utilizar o protocolo IPsec nativo no IPv6 para tornar o processo mais seguro. A troca de mensagens pode ser feita utilizando o protocolo AH do IPsec, para isso é necessário um relacionamento prévio de segurança entre o nó móvel e o agente domiciliário, ou seja, uma chave secreta deve ser configurada nos dois nós.

No processo do registro do nó móvel com o nó correspondente torna-se impossível utilizar o IPsec para prover segurança no processo, pois o nó correspondente pode estar localizado em qualquer lugar na Internet. Para tornar o processo seguro faz-se necessário o uso de algum mecanismo global de autenticação automática. A solução proposta para esse problema é conhecida como *Return Routability Procedure*.

### 2.6.1 *Return Routability Procedure*

Na tentativa de tornar mais seguro o registro do nó móvel com o nó correspondente o MIPv6 introduz uma solução bastante inteligente o *Return Routability Procedure*, a idéia do processo é que o nó correspondente obtenha garantias que o nó móvel seja alcançável pelo seu endereço domiciliário e o *care-of-address*. Somente assim o nó correspondente está apto para receber *Binding Updates*.

O teste consiste em enviar duas mensagens ao nó correspondente uma via agente domiciliário (*Home Test Init*) e outra diretamente ao nó correspondente (*Care-of Test Init*). Em resposta o nó correspondente envia duas mensagens (*Home Test* e *Care-of Test*) a partir dos dados destas mensagens o nó móvel é capaz de gerar uma chave de segurança denominada *binding management key* (Kbm).

Na figura 2.6.1 pode-se observar o fluxo das mensagens utilizados no *Return Routability Procedure*.

Para autorizar o *Binding Update* o nó móvel gerou a Kbm que permite uma verificação do nó correspondente, ao receber o *Binding Update* o nó correspondente é capaz de recalcular a Kbm e permitir a associação do *care-of-address* com o endereço domiciliário.

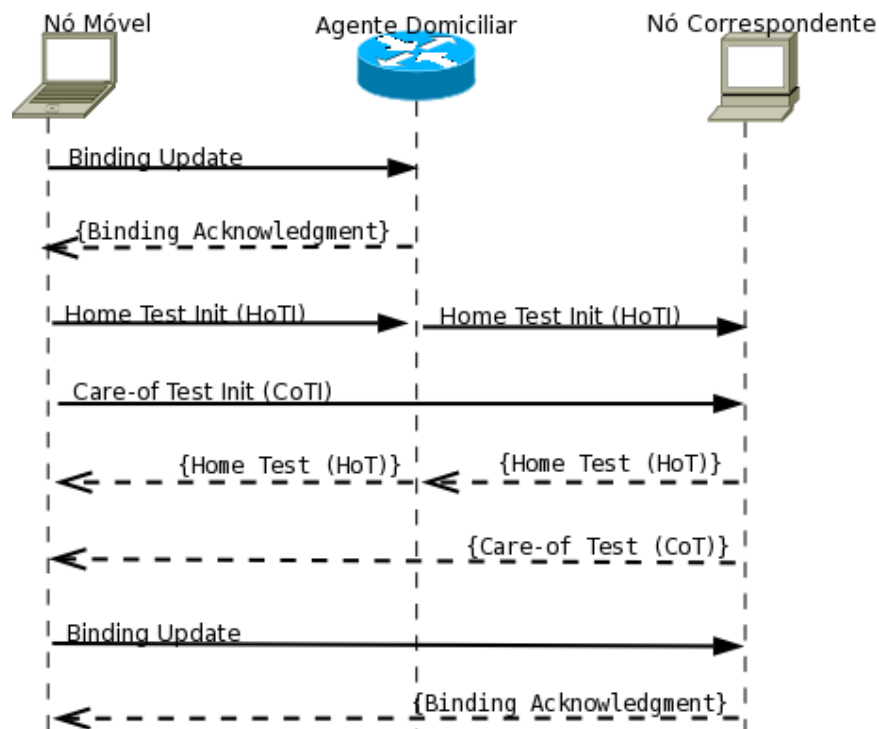


Figura 2.2: Mensagens MIPv6

É importante salientar que o *Return Routability Procedure* não é totalmente seguro. Pois, um atacante pode capturar as mensagens enviadas pelo nó correspondente e forjar mensagens de *Binding Updates*. Obviamente o mesmo atacante conseguirá fazer o mesmo ataque em uma rede IPv6 sem mobilidade. Podemos concluir que o *Return Routability Procedure* não introduz riscos adicionais ao protocolo IPv6 básico.

## 3 *Plataforma de Testes*

O objetivo deste capítulo é descrever todos os procedimentos efetuados para montar um ambiente de simulação, que irá auxiliar na análise do protocolo MIPv6, vamos usar simulação devido a facilidade e rapidez para construção dos cenários, além disso realizar os experimentos fisicamente envolvem um alto custo e demandam muito tempo. No ambiente simulação faremos o uso da máquina virtual *User Mode Linux* (UML), pois é incorporada ao kernel e apresenta um alto desempenho em execução.

Nos testes envolvendo o protocolo é necessário um Kernel Linux compilado com suporte ao MIPv6, uma instalação do *Mobile IPv6 for Linux* (MIPL) uma implementação do IPv6 Móvel para o Linux, e a instalação do *Router ADvertisement Daemon* (RADVD) necessário em redes IPv6 para gerar *router advertisement* e escutar *router solicitations*.

### 3.1 UML

UML (*User Mode Linux*) é uma máquina virtual para sistemas Linux, é uma implementação que permite que o Kernel Linux rode no sistema operacional Linux como um processo normal, diferente de outras tecnologias de virtualização que emulam uma plataforma física. Com UML é possível criar máquinas virtuais para ajudar no desenvolvimento de aplicações, serviços de rede, implementação clusters de rede, testes de segurança.

Para utilizar uma máquina virtual UML somente é necessário uma versão do Kernel Linux compilado para a arquitetura UML e um sistema de arquivos. No endereço <http://uml.nagafix.co.uk/> há disponível vários sistemas de arquivos, no ambiente de simulação utilizaremos um baseado no Linux Debian. Para iniciar a construção do ambiente de simulação vamos fazer o download do sistema de arquivos:

```
# mkdir vm
# cd vm
# wget http://uml.nagafix.co.uk/Debian-4.0/Debian-4.0-x86-root_fs.bz2
```

```
# bunzip2 Debian-4.0-x86-root_fs.bz2
# mv Debian-4.0-x86-root_fs DebianFS
```

Um pacote de ferramentas para UML chamado *uml\_utilities* deve ser instalado no sistema hospedeiro para permitir interligar as máquinas virtuais em rede, e auxiliar no ambiente de testes.

```
# wget http://prdownloads.sourceforge.net/user-mode-linux\
/uml_utilities_20040406.tar.bz2
# tar jxvf uml_utilities_20040406.tar.bz2 && cd tools
# make && make install
```

O *uml\_switch* que acompanha este pacote não permite a simulação de mobilidade, algumas alterações no código do *uml\_switch* foram feitas pelo desenvolvedor do projeto "Avaliação do Protocolo Fast Handover MIPv6" que permitem a simulação da mobilidade. Será preciso desta versão para o ambiente:

```
# wget http://algun.lugar
# tar xvzf uml_switch.tar.gz && cd uml_switch
# make && make install
```

## 3.2 Kernel

Para o ambiente de simulação precisamos um kernel compilado para a arquitetura UML com suporte a MIPv6. No kernel o suporte nativo do protocolo MIPv6 está disponível desde a versão 2.6.17, nestes cenários de testes será utilizado a versão 2.6.25.6 do kernel. Para iniciar o trabalho de compilação é necessário fazer o download dos códigos fontes do kernel:

```
# wget http://kernel.org/pub/linux/kernel/v2.6/linux-2.6.25.6.tar.bz2
# tar jxvf linux-2.6.25.6.tar.bz2
# cd linux-2.6.22.6
```

Configurando a arquitetura na qual o kernel será compilado:

```
# export ARCH=um
```

Configurando o kernel:

```
# make defconfig
# make menuconfig
```

É interessante habilitar a opção HOSTFS para as máquinas virtuais terem acesso ao sistema de arquivos do hospedeiro.

```
UML-specific options
-->Host filesystem [HOSTFS]
```

Para configurar o kernel com suporte ao MIPv6 estas são as opções mínimas que devem ser setadas:

```
CONFIG_EXPERIMENTAL=y
CONFIG_SYSVIPC=y
CONFIG_PROC_FS=y
CONFIG_NET=y
CONFIG_INET=y
CONFIG_IPV6=y
CONFIG_IPV6_MIP6=y
CONFIG_XFRM=y
CONFIG_XFRM_USER=y
CONFIG_XFRM_SUB_POLICY=y
CONFIG_INET6_XFRM_MODE_ROUTEOPTIMIZATION=y
```

Opções que o agente domiciliar e o nó móvel necessitam:

```
CONFIG_IPV6_TUNNEL=y
CONFIG_IPV6_MULTIPLE_TABLES=y
```

Opção que o nó móvel também necessita:

```
CONFIG_IPV6_SUBTREES=y
```

Para alguns indicadores de movimento do nó móvel, pode ser setado:

```
CONFIG_ARPD=y
```

Para suporte ao IPsec é necessário pelo menos:

CONFIG\_INET6\_ESP=y

Para utilizar IPsec nos túneis:

CONFIG\_NET\_KEY=y

CONFIG\_NET\_KEY\_MIGRATE=y

No menu de configuração estas são as opções que devem ser setadas:

Code maturity level options

--> Prompt for development and/or incomplete code/drivers [CONFIG\_EXPERIMENTAL]

General setup

--> System V IPC [CONFIG\_SYSVIPC]

Networking

--> Networking support [CONFIG\_NET]

--> Networking options

--> Transformation user configuration interface [CONFIG\_XFRM\_USER]

--> Transformation sub policy support [CONFIG\_XFRM\_SUB\_POLICY]

--> Transformation migrate database [CONFIG\_XFRM\_MIGRATE]

--> PF\_KEY sockets [CONFIG\_NET\_KEY]

--> PF\_KEY MIGRATE [CONFIG\_NET\_KEY\_MIGRATE]

--> TCP/IP networking [CONFIG\_INET]

--> The IPv6 protocol [CONFIG\_IPV6]

--> IPv6: AH transformation [CONFIG\_INET6\_AH]

--> IPv6: ESP transformation [CONFIG\_INET6\_ESP]

--> IPv6: IPComp transformation [CONFIG\_INET6\_IPCOMP]

--> IPv6: Mobility [CONFIG\_IPV6\_MIP6]

--> IPv6: IPsec transport mode [CONFIG\_INET6\_XFRM\_MODE\_TRANSPORT]

--> IPv6: IPsec tunnel mode [CONFIG\_INET6\_XFRM\_MODE\_TUNNEL]

--> IPv6: MIPv6 route optimization mode [XFRM\_MODE\_ROUTEOPTIMIZATION]

--> IPv6: IPv6-in-IPv6 tunnel [CONFIG\_IPV6\_TUNNEL]

--> IPv6: Multiple Routing Tables [CONFIG\_IPV6\_MULTIPLE\_TABLES]

--> IPv6: source address based routing [CONFIG\_IPV6\_SUBTREES]

File systems

--> Pseudo filesystems

```
--> /proc file system support [CONFIG_PROC_FS]
```

Para verificar se o kernel está configurado corretamente para MIPv6 existe o *shell script* **chkconf\_kernel.sh** que acompanha o pacote do MIPL.

Após a configuração o kernel vamos compilá-lo e os seus módulos devem ser copiados para o sistema de arquivos, por isso serão instalados no diretório `uml-modules` para depois serem copiados.

```
# make
# strip linux
# make modules_install INSTALL_MOD_PATH=uml-modules
# unset arch
```

Para copiar os módulos do kernel para o sistema de arquivos:

```
# cd ..
# mkdir loop
# mount -o loop DebianFS loop
# cp -rf linux-2.6.25.6/uml-modules/lib/modules/* loop/lib/modules
```

É preciso copiar os arquivos fontes do kernel para o sistema de arquivos, pois o MIPL vai precisar para a sua compilação.

```
# cp -r linux-2.6.25.6.tar.bz2 loop/usr/src/
# cd loop/usr/src
# tar jxvf linux-2.6.25.6.tar.bz2
# ln -s linux-2.6.25.6 linux
# rm linux-2.6.25.6.tar.bz2
# cd ../../../../
# umount loop
```

### 3.3 Preparando o sistema de arquivos

Alguns pacotes de desenvolvimento, bibliotecas deverão ser instalados no sistema de arquivos, pois os programas MIPL e RADVD serão compilados na máquina virtual. E para auxiliar na análise dos cenários serão instaladas algumas ferramentas de rede.



Para poder fazer o download dos pacotes é necessário a criação de uma rede entre o hospedeiro e a máquina virtual. Neste caso deve-se criar uma interface virtual no hospedeiro, o módulo **tun** deve estar carregado no sistema.

```
# modprobe tun
# tuncctl -u <nome do usuário>
```

Configurando a interface virtual do hospedeiro, habilitando o roteamento de pacotes e o nat para os pacotes da máquina virtual:

```
# ifconfig tap0 192.168.1.1 netmask 255.255.255.0
# echo 1 > /proc/sys/net/ipv4/ip_forward
# iptables -t nat -I POSTROUTING -o eth0 -j MASQUERADE
# iptables -I FORWARD -i tap0 -j ACCEPT
# iptables -I FORWARD -o tap0 -j ACCEPT
```

Iniciando a máquina virtual, para se logar use *root* sem senha:

```
# cp linux-2.6.25.6/linux .
# ./linux ubda=DebianFS mem=128M eth0=tuntap,tap0
```

Configurando a rede na maquina virtual:

```
uml# ifconfig eth0 192.168.1.2 netmask 255.255.255.0
uml# route add default gw 192.168.1.1
uml# echo nameserver <seu_dns> > /etc/resolv.conf
```

Na distribuição Linux Debian há um gerenciador de pacotes o **apt** que facilita a instalação de programas, ele permite procurar e instalar pacotes. Para utilizar é necessário configurar um repositório de pacotes:

```
uml# echo deb http://security.debian.org/ etch/updates main contrib > \
/etc/apt/source.list
```

Os seguinte pacotes serão instalados:

- gcc-4.1

- libc6-dev
- make
- libtool
- automake1.9
- autoconf2.13
- bison
- patch
- flex
- indent
- telnet
- tcpdump
- tethereal
- iproute
- subversion

Para instalar utilizando o *apt*:

```
uml# apt-get update
uml# apt-get install gcc-4.1 libc6-dev make libtool automake1.9 \
autoconf2.13 bison patch flex indent telnet tcpdump tethereal \
iproute subversion
```

## 3.4 MIPL

O MIPL (*Mobile IPv6 for Linux*) é uma implementação de Suporte a Mobilidade no IPv6 (RFC 3775), desenvolvida na Universidade Tecnológica de Helsinki. É um programa em *user space* que trabalha junto com MIPv6 habilitado no kernel.

Vamos fazer o download da ultima versão estável do MIPL:

```
uml# wget ftp://ftp.linux-ipv6.org/pub/usagi/patch/mip6/umip-0.4/daemon\
/tarball/mip6-daemon-umip-0.4.tar.gz
uml# tar xvzf mip6-daemon-umip-0.4.tar.gz && cd mip6-daemon-umip-0.4
```

Vamos configurar o MIPL para buscar os cabeçalhos do kernel no local onde foram copiados, habilitaremos a compilação de um virtual terminal onde poderemos acessar para acompanhar o protocolo em execução.

```
uml# CPPFLAGS='-isystem /usr/src/linux/include' ./configure --enable-vt
```

Compilando e instalando:

```
uml# make
uml# make install
```

### 3.4.1 Configurando MIPL

Nesta seção pretende-se mostrar como configurar o *daemon* do MIPL para rodar como um nó móvel, um agente domicili ou um nó correspondente, explanando todas as opções do seu arquivo de configuração.

Esta são as opções comuns ao nó móvel, ao agente domicili e ao nó corresponde do arquivo de configuração do MIPL.

**NodeConfig CN — HA — MN;**

Indica se o *daemon* deve rodar como nó correposdente, agente domicili ou nó móvel. A configuração padrão é CN.

**DebugLevel number;**

Indica ao *daemon* o nível de *debug*, se o valor for maior que zero, o *daemon* irá imprimir mensagens de *debug* no console. A configuração padrao é 0.

**DoRouteOptimizationCN boolean;**

Indica se o nó deve participar da otimização de roteamento com o nó móvel. A configuração padrão é habilitado.

Opções comuns ao nó móvel e ao agente domiciliário: Falta explicar vários parâmetros ainda.

## 3.5 RADVD

O RADVD (*Router ADvertisement Daemon*) é um serviço para roteadores IPv6. Ele envia mensagens *Router Advertisement* e responde a *Router Solicitation* especificadas na RFC 2641. Essas mensagens são utilizadas para *stateless autoconfiguration*. Vamos fazer o download do pacote, compilar e instalar:

```
uml# wget http://www.litech.org/radvd/dist/radvd-1.1.tar.gz
uml# tar xvzf radvd-1.1.tar.gz && cd radvd-1.1
uml# make && make install
```

### 3.5.1 Configurando RADVD

Opções do arquivo de configuração ...

## 3.6 Gerador de tráfego

Para auxiliar nos testes do protocolo MIPv6 foi criado uma simples aplicação chamada *gen* que permite gerar trafegos TCP e UDP na rede. O programa foi desenvolvido utilizando a linguagem C e a API de *sockets*, é semelhante a ferramenta *ping*, porém utiliza as camadas de aplicação e de transporte. Com o *gen* permite-se observar a transparência do protocolo com as camadas superiores a de rede.

Vamos fazer o download do programa e sua compilação:

```
uml# wget http://
uml# gcc -o gen gen.c
uml# mv gen /bin
```

Para utilizar o programa, é preciso que um nó da rede inicie o *gen* e fique esperando pacotes, desta forma:

```
mn# gen
```

E outro nó deve iniciar o gen passando como parâmetro o endereço IP do outro nó, desta forma:

```
cn# gen 2000:a::1
```

## 4 *Cenários de Testes*

Após os estudos bibliográficos sobre o protocolo MIPv6 e a preparação de um ambiente de simulação que possibilita a montagem de cenários. Iniciou-se a etapa para definir os cenários que permitiriam analisar o comportamento, testar a funcionalidade do protocolo, perdas e atrasos na entrega dos pacotes.

Para analisar alguns mecanismos do protocolo e testar seus modos de operação, dois cenários base foram definidos. No primeiro cenário uma comunicação entre um dispositivo móvel e um dispositivo estático, no segundo comunicação entre dois dispositivos móveis. Com intuito de comparação de desempenho os dois cenários serão testados nos dois modos de operação do protocolo MIPv6, o primeiro com tunelamento bidirecional e o segundo com otimização de roteamento.

No anexo B estão disponíveis dois *scripts* que foram feitos para auxiliar na montagem das topologias de rede e na configuração dos cenários. Também está disponível os arquivos de configuração do MIPL e do RADVD para cada máquina virtual dos cenários.

Para auxiliar na geração de *logs* que tendem a enriquecer análise dos cenário foram utilizadas as ferramentas *tcpdump*, *ping6* e *gen*.

### 4.1 **Cenário 1**

O primeiro cenário de teste consiste em gerar um tráfego com o *gen* ou *ping*, com o fluxo de dados do nó correspondente ao nó móvel, provocar uma situação de mobilidade no nó móvel, depois o retorno a sua rede domiciliar. Em primeiro momento utilizando o modo com tunelamento bidirecional e num segundo com otimização de rotas.

O cenário é formado por três redes interligadas entre si e cinco nós, sua topologia pode ser observada na figura 4.1, todos os endereços atribuídos as interfaces dos nós durante a realização do cenário e seus endereços de hardware estão disponíveis na tabela 4.1.

1. **Nó Móvel (MN)**: Nó que inicia o experimento na rede **2000:a::** depois se move para a rede **2000:d::**.
2. **Correspondente (CN)**: Nó da rede que se comunica com o nó móvel durante o experimento.
3. **AR1**: É o roteador que interliga a rede domiciliar **2000:a::** com as outras e oferece o serviço de agente domiciliar.
4. **AR2**: É o roteador que interliga a rede **2000:c::** com as outras, é onde se encontra o nó correspondente.
5. **AR3**: É o roteador que interliga a rede **2000:d::** com as outras, é para onde o nó móvel se move.

Nó	Interface	MAC	Endereço	Tipo
MN	eth0	92:09:4F:D5:EF:EA	2000:a::1	Domiciliar
			2000:a::9009:4fff:fed5:efea	Auto-configurado
			2000:d::9009:4fff:fed5:efea	Care-of-address
CN	eth0	D6:7F:B0:C3:C9:0C	fe80::9009:4fff:fed5:efea	Local
			2000:c::1	Global
			fe80::d47f:b0ff:fec3:c90	Local
AR1	eth0	AE:4D:A8:10:EB:1D	2000:a::2	Global
	eth1	1E:2D:C8:43:E6:C7	fe80::ac4d:a8ff:fe10:eb1d	Local
			2000:b::1	Global
			fe80::1c2d:c8ff:fe43:e6c7	Local
AR2	eth0	36:A7:3F:06:58:8D	2000:c::2	Global
	eth1	4A:19:FD:12:48:34	fe80::34a7:3fff:fe06:588d	Local
			2000:b::2	Global
			fe80::4819:fdff:fe12:4834	Local
AR3	eth0	B2:4C:E9:EC:6C:42	2000:d::2	Global
	eth1	2A:E2:57:C7:4F:E4	fe80::b04c:e9ff:feec:6c42	Local
			2000:b::3	Global
			fe80::28e2:57ff:fec7:4fe4	Local

Tabela 4.1: Endereços do Cenário 1

Após todo o cenário iniciado, a idéia é começar o envio de mensagens UDP por meio do *gen*, e monitorar todo o funcionamento do MIPv6 durante o processo de *handover* utilizando o *tcpdump*. Para isso os seguintes comandos foram disparados no nó móvel e no nó correspondente:

```
mn# gen -l -d -p udp -s 400 &
mn# tcpdump -vvvSX
```

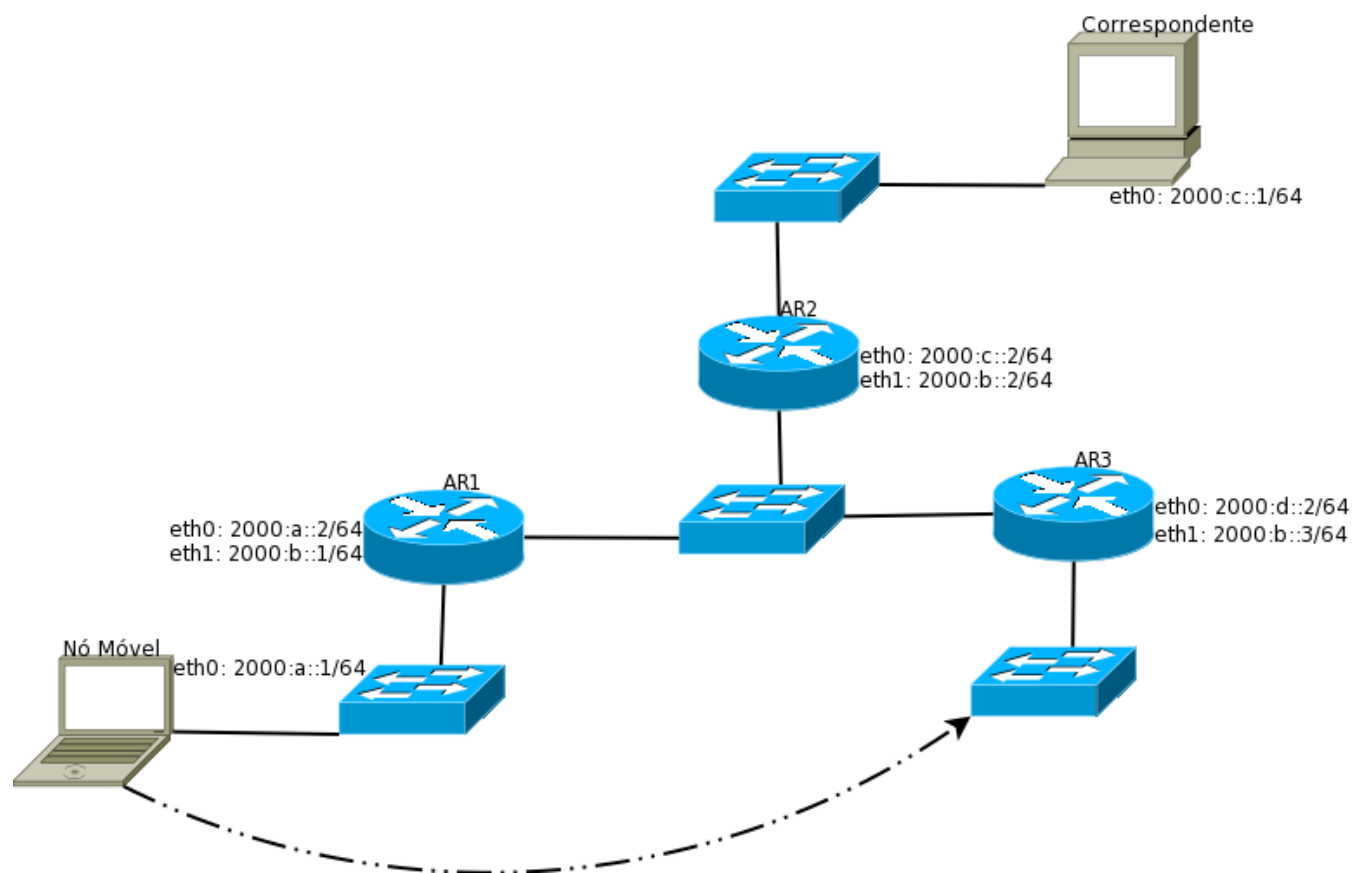


Figura 4.1: Topologia do Cenário 1



```
cn# gen -a 2000:a::1 -d -t -p udp -s 400
```

Depois em um segundo momento repetimos o mesmo cenário utilizando o modo de operação com otimização de roteamento. Para auxiliar na obtenção dos *logs* para análise desta vez utilizaremos o utilitário *ping6*, pois a otimização de roteamento somente se aplica para novas conexões e o *gen* mantém a mesma conexão, desta forma não conseguiríamos ver os processos de registro com o nó correspondente e o *Return Routability Procedure*.

### 4.1.1 Resultados

Com a realização dos experimentos podemos constatar o funcionamento do protocolo e perceber continuação da comunicação entre os pontos comunicantes após a mobilidade de forma transparente as camadas superiores a de rede, observar todas as mensagens trocadas no processo e estimar o tempo de latência com a mobilidade. O primeiro resultado que se pode obter com a realização do Cenário 1, foi o perfeito funcionamento do MIPv6, ocorreram algumas perdas de pacotes, mas o nó móvel conseguiu continuar sendo alcançado pelo seu endereço domiciliar mesmo não estando em sua rede domiciliar.

Os dados obtidos com a saída do comando *tcpdump* foram compilados em forma de uma tabela para facilitar a visualização dos mesmos e podem ser observados na tabela 4.4.

Analisando os dados recolhidos conseguimos observar exato momento que ocorreu a mobilidade para a outra rede, até então o nó móvel recebia

Podemos constatar mudanças nas tabelas de roteamento do nó móvel e do agente domiciliar depois da mobilidade nas respectivas figuras 4.2 e 4.3, após receber o *Binding Update* e o agente domiciliar criar o túnel com o nó móvel, é adicionada a seguinte rota a sua tabela de roteamento, que todos os pacotes com destino ao endereço domiciliar devem ser encaminhados para o túnel. E no nó móvel os pacotes com destino ao nó correspondente e ao agente domiciliar são encaminhados pelo túnel, caracterizando o tunelamento bidirecional.

Destino	Via	Proximo salto	Considerações
default	eth0	fe80::ac4d:a8ff:fe10:eb1d	Na rede domiciliar
default	eth0	fe80::b04c:e9ff:feec:6c42	Na rede visitada
2000:a::2	ip6tnl1	2000:a::2	
2000:c::1	ip6tnl1	2000:c::1	

Tabela 4.2: Tabela de roteamento do nó móvel no Cenário 1

Destino	Via	Proximo salto	Considerações
2000:c::/64	eth1	2000:b::2	Antes da mobilidade do nó móvel
2000:d::/64	eth1	2000:b::3	
2000:c::/64	eth1	2000:b::2	Após a mobilidade do nó móvel
2000:d::/64	eth1	2000:b::3	
2000:a::1	ip6tn11	2000:a::1	

Tabela 4.3: Tabela de roteamento do agente domiciliar no Cenário 1

### 4.1.2 Tempo de latência do *Handover*

O tempo de latência de um *handover* segundo [] (por referência) pode ser dividida em quatro fases, a partir dos dados colhidos com Cenário 1 vamos tentar reproduzi-las, exemplificando com os nossos tempos:

1. **Detecção de Movimento (*TD*):** Em um cenário real representaria o tempo do *handover* na camada de enlace até o primeiro *Router Advertisement*. Como neste ambiente de testes não conseguimos simular a camada enlace não se pode precisar com a exatidão a latência envolvida no processo. Porém, para fins de estudo consideraremos para nosso cenário o tempo entre o último pacote do *gen* recebido na rede domiciliar e o primeiro *Router Advertisement* na rede visitada.

$$TD = t1 - t0$$

$$TD = 24.23 - 21.99$$

$$TD = 2.24$$

2. **Configuração do *Care-of-address* (*TA*):** Tempo que entre o primeiro *Router Advertisement* e o envio do *Binding Update*.

$$TA = t2 - t1$$

$$TA = 25.55 - 24.23$$

$$TA = 1.32$$

3. **Registro com agente domiciliar (*TR*):** Intervalo de tempo entre o envio do *Binding Update* ao agente domiciliar e o recebimento do *Binding Acknowledgement*.

$$TR = t3 - t2$$

$$TR = 26.58 - 25.55$$

$$TR = 1.03$$

Tempo (s)	Origem	Destino	Conteúdo
21:59.952226	fe80::ac4d:a8ff:fe10:eb1d	ff02::1	RA, Flags [Home Agent]
22:00.525313	fe80::ac4d:a8ff:fe10:eb1d	2000:a::1	NS, who 2000:a::1
22:00.525398	2000:a::1	fe80::ac4d:a8ff:fe10:eb1d	Neighbor Advertisement
22:00.656486	2000:c::1	2000:a::1	Gen seq#=5
22:01.664448	2000:c::1	2000:a::1	Gen seq#=6
<b>22:03.467587</b>	<b>fe80::b04c:e9ff:feec:6c42</b>	<b>ff02::1</b>	<b>Router Advertisement</b>
22:03.482118	::	ff02::16	HBH, multicast listener
22:03.799224	::	ff02::1:ffd5:efea	NS, who fe80::9009:4fff:fed5:efea
22:04.285049	::	ff02::1:ffd5:efea	NS, who 2000:d::9009:4fff:fed5:efea
<b>22:05.288026</b>	<b>2000:d::9009:4fff:fed5:efea</b>	<b>2000:a::2</b>	<b>BU seq#=26356 AH</b>
22:05.304531	fe80::9009:4fff:fed5:efea	ff02::16	HBH, multicast listener
22:06.305532	fe80::b04c:e9ff:feec:6c42	ff02::1:ffd5:efea	NS, who 2000:d::9009:4fff:fed5:efea
22:06.305650	2000:d::9009:4fff:fed5:efea	fe80::b04c:e9ff:feec:6c42	Neighbor Advertisement
<b>22:06.305972</b>	<b>2000:a::2</b>	<b>2000:d::9009:4fff:fed5:efea</b>	<b>BA seq#=26356 lifetime=262140</b>
22:06.316950	fe80::b04c:e9ff:feec:6c42	ff02::1	Router Advertisement
22:06.512529	fe80::9009:4fff:fed5:efea	fe80::ac4d:a8ff:fe10:eb1d	NS, who fe80::ac4d:a8ff:fe10:eb1d
22:06.700567	2000:a::2	2000:d::9009:4fff:fed5:efea	2000:c::1 ↯ 2000:a::1 (Gen seq#=11)
22:07.510304	fe80::9009:4fff:fed5:efea	fe80::ac4d:a8ff:fe10:eb1d	NS, who fe80::ac4d:a8ff:fe10:eb1d
22:07.705972	2000:a::2	2000:d::9009:4fff:fed5:efea	2000:c::1 ↯ 2000:a::1 (Gen seq#=12)
22:08.338339	fe80::b04c:e9ff:feec:6c42	ff02::1	Router Advertisement
.	.	.	.
.	.	.	.
.	.	.	.
22:18.785859	2000:a::2	2000:d::9009:4fff:fed5:efea	2000:c::1 ↯ 2000:a::1 (Gen seq#=23)
<b>22:19.815983</b>	<b>fe80::ac4d:a8ff:fe10:eb1d</b>	<b>ff02::1</b>	<b>RA, Flags [Home Agent]</b>
22:19.839337	::	ff02::16	HBH, multicast listener
22:19.877057	::	ff02::16	HBH, multicast listener
22:20.170279	::	ff02::1:ffd5:efea	NS, who 2000:a::9009:4fff:fed5:efea
22:20.585230	::	ff02::1:ffd5:efea	NS, who fe80::9009:4fff:fed5:efea
22:20.879643	::	ff02::16	HBH, multicast listener
22:21.585309	::	ff02::1:ff00:1	NS, who 2000:a::1
22:21.606725	fe80::9009:4fff:fed5:efea	ff02::16	HBH, multicast listener
22:21.767709	fe80::ac4d:a8ff:fe10:eb1d	ff02::1	Neighbor Advertisement
<b>22:21.772629</b>	<b>2000:a::1</b>	<b>2000:a::2</b>	<b>BU seq#=26357</b>
22:21.785931	fe80::9009:4fff:fed5:efea	ff02::16	HBH, multicast listener
22:21.792882	fe80::ac4d:a8ff:fe10:eb1d	ff02::16	HBH, multicast listener
22:21.812524	fe80::ac4d:a8ff:fe10:eb1d	ff02::1:ff00:1	NS, who 2000:a::1
22:21.812620	2000:a::1	fe80::ac4d:a8ff:fe10:eb1d	Neighbor Advertisement
22:21.812932	2000:c::1	2000:a::1	Gen seq#=27
<b>22:21.874997</b>	<b>2000:a::2</b>	<b>2000:a::1</b>	<b>BA seq#=26357 lifetime=0</b>
22:21.904397	2000:a::1	ff02::1	Neighbor Advertisement
22:22.569757	fe80::ac4d:a8ff:fe10:eb1d	ff02::1	RA, Flags [Home Agent]
22:22.814652	2000:c::1	2000:a::1	Gen seq#=28

Tabela 4.4: Mensagens do Cenário 1 com Tunelamento Bidirecional

Tempo (s)	Origem	Destino	Conteúdo
50:21.991721	2000:a::1	2000:c::1	ICMP6, echo reply
<b>50:24.231519</b>	<b>fe80::b04c:e9ff:feec:6c42</b>	<b>ff02::1</b>	<b>Router Advertisement</b>
50:24.247992	::	ff02::16	HBH, multicast listener
50:24.545598	::	ff02::1:ffd5:efea	NS, who 2000:d::9009:4fff:fed5:efea
50:24.879467	::	ff02::1:ffd5:efea	NS, who fe80::9009:4fff:fed5:efea
50:25.548165	fe80::b04c:e9ff:feec:6c42	ff02::1	Router Advertisement
<b>50:25.557223</b>	<b>2000:d::9009:4fff:fed5:efea</b>	<b>2000:a::2</b>	<b>BU seq#=47808 AH</b>
50:25.576407	::	ff02::16	HBH, multicast listener
50:26.581135	fe80::b04c:e9ff:feec:6c42	ff02::1:ffd5:efea	NS, who 2000:d::9009:4fff:fed5:efea
50:26.581276	2000:d::9009:4fff:fed5:efea	fe80::b04c:e9ff:feec:6c42	Neighbor Advertisement
<b>50:26.581588</b>	<b>2000:a::2</b>	<b>2000:d::9009:4fff:fed5:efea</b>	<b>BA seq#=47808 lifetime=262140</b>
50:26.987535	fe80::9009:4fff:fed5:efea	fe80::ac4d:a8ff:fe10:eb1d	NS, who fe80::ac4d:a8ff:fe10:eb1d
50:27.009344	2000:a::2	2000:d::9009:4fff:fed5:efea	2000:c::1 ; 2000:a::1, (icmp6)
50:27.009538	2000:d::9009:4fff:fed5:efea	2000:a::2	2000:a::1 ; 2000:c::1, (icmp6)
<b>50:27.013543</b>	<b>2000:d::9009:4fff:fed5:efea</b>	<b>2000:a::2</b>	<b>2000:a::1 ; 2000:c::1, (HoTI)</b>
<b>50:27.015165</b>	<b>2000:d::9009:4fff:fed5:efe</b>	<b>2000:c::1</b>	<b>CoTI Care-of Init</b>
<b>50:27.019919</b>	<b>2000:a::2</b>	<b>2000:d::9009:4fff:fed5:efea</b>	<b>2000:c::1 ; 2000:a::1, (HoT)</b>
<b>50:27.029192</b>	<b>2000:c::1</b>	<b>2000:d::9009:4fff:fed5:efea</b>	<b>CoT</b>
<b>50:27.031650</b>	<b>2000:d::9009:4fff:fed5:efea</b>	<b>2000:c::1</b>	<b>BU seq#=1077 A</b>
<b>50:27.037401</b>	<b>2000:c::1</b>	<b>2000:d::9009:4fff:fed5:efea</b>	<b>BA seq#=1077 lifetime=420</b>
50:27.365152	fe80::b04c:e9ff:feec:6c42	ff02::1	Router Advertisement
50:27.982719	fe80::9009:4fff:fed5:efea	fe80::ac4d:a8ff:fe10:eb1d	Neighbor Solicitation
50:28.014358	2000:c::1	2000:d::9009:4fff:fed5:efea	ICMP6, echo request
50:28.014494	2000:d::9009:4fff:fed5:efea	2000:c::1	ICMP6, echo reply

Tabela 4.5: Mensagens do Cenário 1 com Otimização de Roteamento

4. **Otimização de Roteamento (TR):** Intervalo de tempo entre o envio das mensagens do *Return Routability Procedure* e o recebimento do *Binding Acknowledgement* do nó correspondente.

$$TO = t_4 - t_3$$

$$TO = 27.03 - 27.01$$

$$TO = 0.2$$

Obviamente podemos calcular o tempo de latência do handover com a seguinte formula:

$$Handover = TD + TA + TR + TO$$

Na figura 4.1.2 é possível observar um diagrama de sequência simplificado mostrando todo o processo de mobilidade. Entretanto, mesmo o cenário sendo simulado e os tempos medidos são estimados, segundo algumas bibliografias estudadas que realizaram experimentos fisicamente os tempos medidos na simulação estão dentro do esperado.

Refazer o diagrama evidenciando TD, TA, TR, TO

## 4.2 Cenário 2

O segundo cenário de teste não passa de uma variação do primeiro onde iremos estabelecer uma comunicação entre dois nós móveis e provocar uma situação de mobilidade nos dois nós. A idéia deste cenário é analisar o comportamento do protocolo quando em uma comunicação ambas as partes se movimentam, comparando com o primeiro em questões como perda de pacotes e tráfego de sinalização.

Este cenário de teste consiste de quatro redes e seis nós, a sua topologia de rede pode ser observada na figura 4.2.

1. **Nó Móvel 1:** Nó que inicia o experimento na rede **2000:a::** depois se move para a rede **2000:d::**.
2. **Nó Móvel 2:** Nó que inicia o experimento na rede **2000:c::** depois se move para a rede **2000:e::**.
3. **AR1:** É o roteador que interliga a rede **2000:a::** com as outras e oferece o serviço de agente domiciliar.

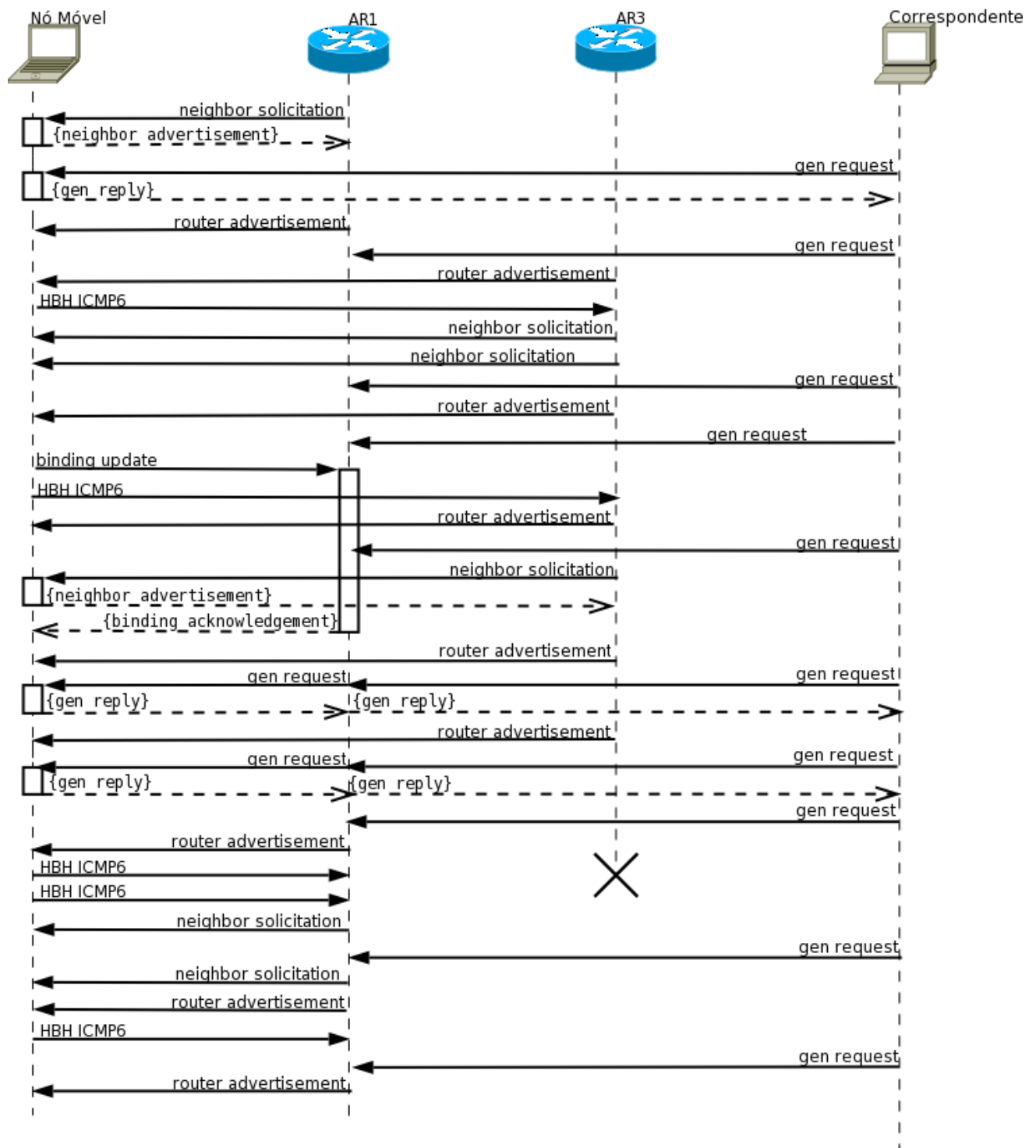


Figura 4.2: Diagrama de sequência do cenário 1

4. **AR2:** É o roteador que interliga a rede **2000:c::** com as outras e oferece o serviço de agente domiciliar.
5. **AR3:** É o roteador que interliga a rede **2000:d::** com as outras, é para onde o nó móvel 1 se move.
6. **AR4:** É o roteador que interliga a rede **2000:e::** com as outras, é para onde o nó móvel 2 se move.

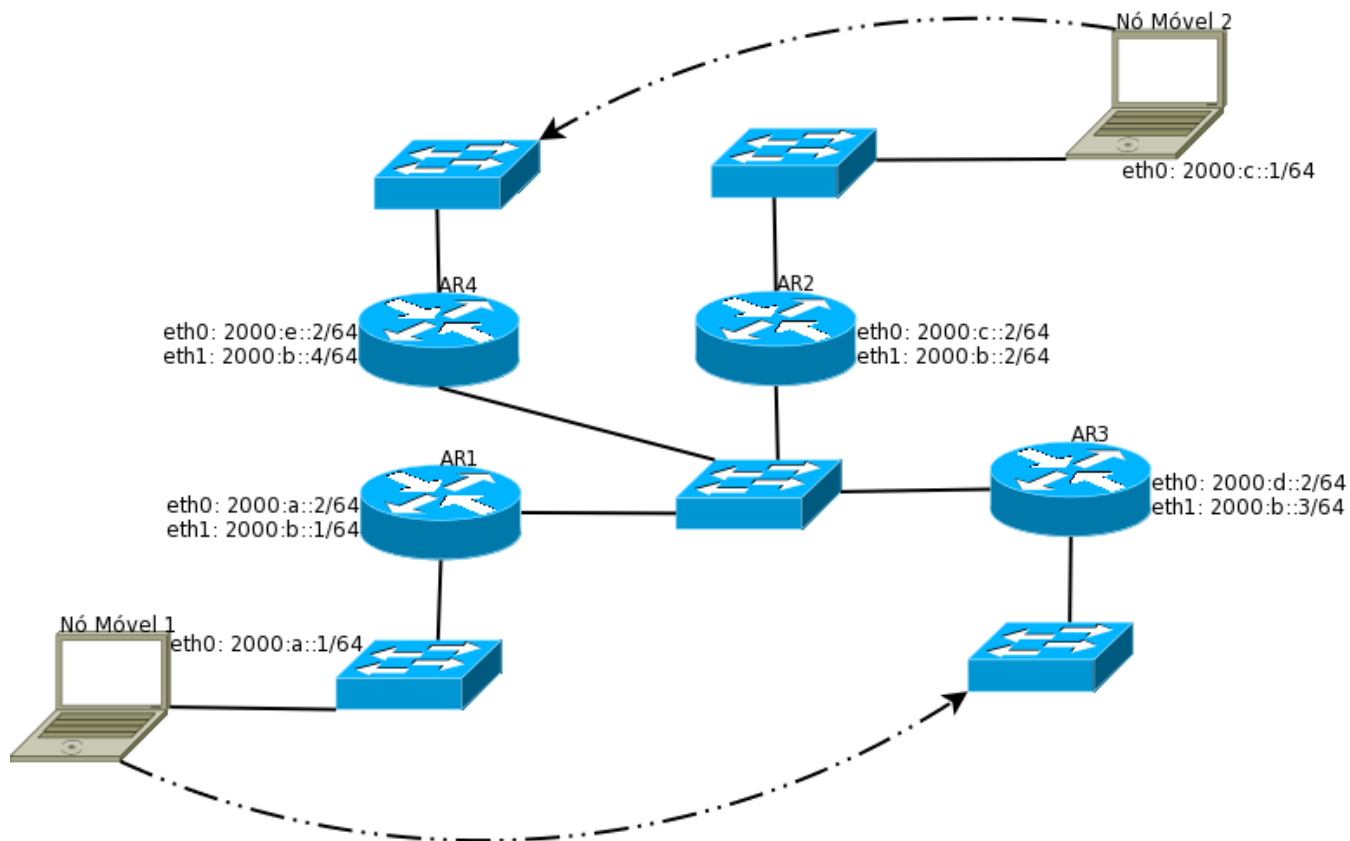


Figura 4.3: Topologia do Cenário 2

## 4.3 Teste de desempenho do MIPv6

Utilizando o Cenário 1 como referência e com o auxílio da ferramenta *gen*, outros testes foram realizados na tentativa de testar o desempenho do protocolo e consequentemente o das máquinas virtuais.

O computador utilizado para a realização dos testes de desempenho possui as seguintes características: *Athlon XP 2600*, 512MB de memória e rodando *Gnu/Linux Slackware 12.0*, kernel *linux-2.6.21.5*.

O primeiro teste pretende verificar a perda na taxa de transmissão em um processo de *handover*. Utilizando a ferramenta *gen* que permite gerar tráfegos específicos e relatórios que extraem a taxa de transmissão e a perda de pacotes, e utilizando o utilitário *GNUPLOT* somos capazes de gerar gráficos que avaliam esse desempenho.

Ostestes realizados foram feitos utilizando tunelamento bidirecional, o tempo configurado para o intervalo entre as mensagens de *Router Advertisement* foi de 30 à 70 milisegundos. Os comandos disparados nos nós móvel e correspondente foram os seguintes:

```
mn# gen -l -d -p udp -s 14000 -f /host/log/  
cn# gen -a 2000:a::1 -d -p udp -s 14000
```

O gráfico obtido no teste, pode ser observado na figura 4.3.

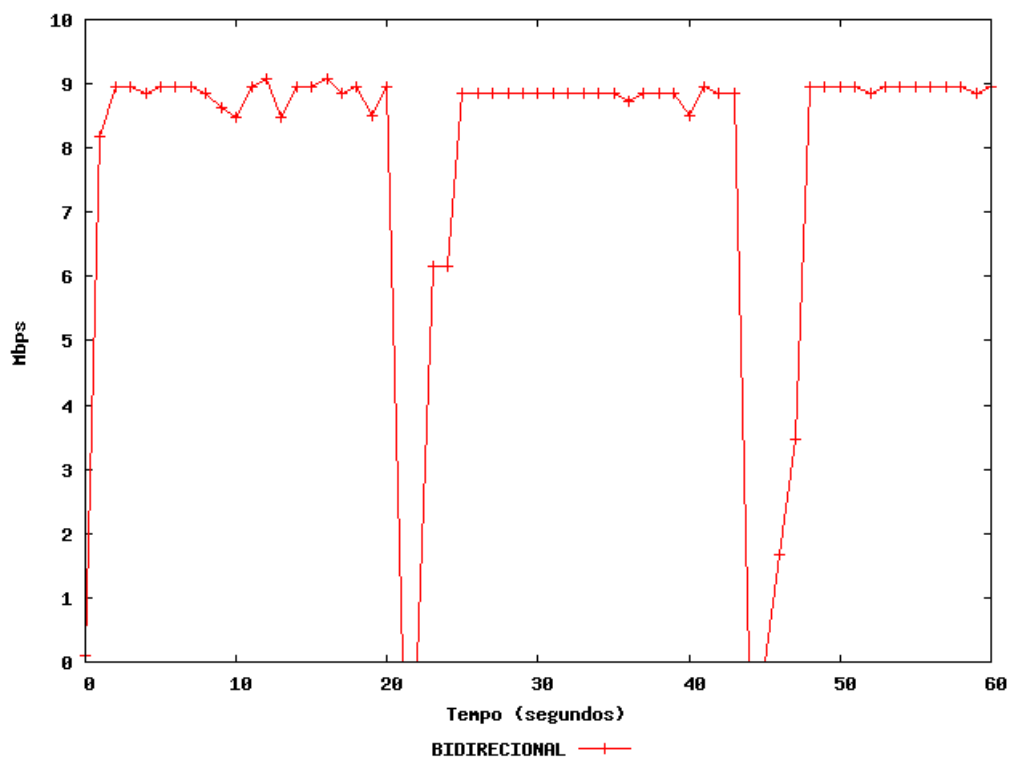


Figura 4.4: Taxa de transmissão em *Handover*

A partir dos dados levantados com o teste podemos tirar várias conclusões, entre as quais destacam-se que o *Troughput* suportado pela rede gira em torno de 9 Mbps, este dado está muito relacionado ao desempenho da máquina hospedeiro, que cada *handover* é de aproximadamente 3 segundos, há perdas consideráveis de pacotes que prejudicariam muito comunicações de tempo real como videoconferência e *VoIP*, porém, navegação em páginas da *Internet* e visualização de *e-mail* a instabilidade é tolerável.



Outro teste proposto, visa verificar se há uma melhora no processo de *handover* diminuindo o intervalo entre as mensagens de *Router Advertisement*, pois é por meio desta mensagem que o nó móvel detecta o movimento e desencadeia todo o processo de *handover*. Para a realização do teste foram realizadas as mesmas configurações do teste anterior, somente foi alterado os valores de intervalo entre as mensagens de *Router Advertisement*. O resultado deste teste de desempenho do MIPv6 esta disponível na figura 4.3.

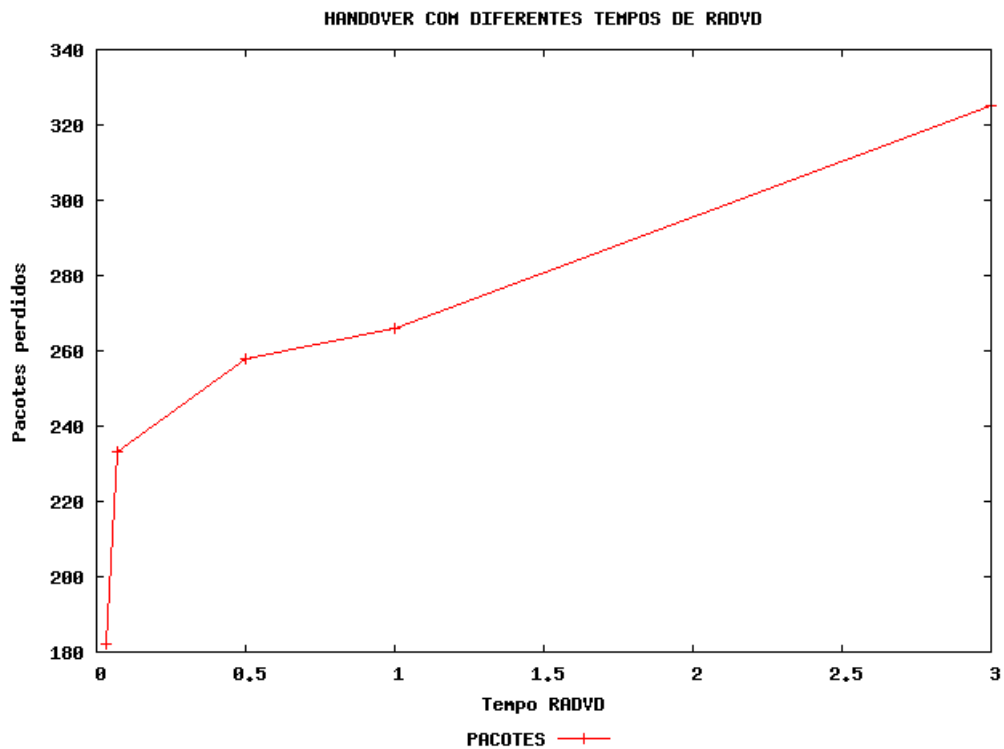


Figura 4.5: Gráfico: Pacotes Perdidos x Tempo Radvd

É possível verificar com este teste, que o intervalo entre as mensagens de *Router Advertisement* esta diretamente ligado com a perda de pacotes durante o *handover*. Porém, não é a solução para o problema do tempo de *handover*, além de não ser uma boa pratica um administrador rede configurar um tempo muito pequeno para o intervalo, pois iria acabar inundando a sua rede com mensagens *multicasting* e prejudicando a perfomance principalmente em redes sem fio. Concereteza esta não é melhor maneira de contornar o problema do tempo do *handover*, para este problema há outras propostas como o *Fast Handover MIPv6* (FMIPv6);

## **5    *Estudo do código do MIPL***

Vai dar trabalho...

## 6 *Visão geral do protocolo HMIPv6*

O modelo do protocolo MIPv6 apresenta alguns problemas que comprometem a sua escalabilidade na *Internet*, alguns desses puderam ser observados nos cenários estudados. Dentre estes problemas podemos destacar:

- tempo na detecção do movimento;
- tempo na configuração do novo endereço na rede visitada, o *care-of-address*;
- tempo do registro com o seu agente domiciliar;

Em um ambiente utilizando o MIPv6 com um excessivo numero de nós móveis a tendência é a perda na qualidade de serviço e o aumento do *delay* na entrega dos pacotes.

Com o intuito de minimizar os efeitos dos longos retardos no envio dos *Binding Updates*, para o registro com o agente domiciliar ou o nó correspondente, e reduzir o tráfego de pacotes de controle na *Internet* foi proposto um protocolo complementar ao MIPv6 o IPv6 Móvel Hierárquico (HMIPv6).

A idéia principal do protocolo é tratar a mobilidade global e local de formas distintas, uma movimentação local pode ser entendida como uma que acontece em um mesmo *site* e a global entre *sites*. Podemos definir um *site* como uma dimensão arbitrária, e pode ser, por exemplo, uma rede de uma empresa ou universidade com uma ou mais sub-redes.

Um estudo sobre os padrões de mobilidade analisou diversos profissionais, mostrou que 69% das movimentações são locais independente deles possuírem algum dispositivo móvel. Este dado mostra que o modelo hierárquico é mais adequado para o uso na *Internet*. As principais vantagens seriam a diminuição no tempo do *handover* e do tráfego enviado para a *Internet* com sinalização.

## 6.1 Terminologia do HMIPv6

**Roteador de acesso (RA):** é o roteador padrão do nó móvel, recebe os dados de saída do nó móvel.

**Mobility Anchor Point (MAP):** é um roteador localizado na rede visitada, o MAP é usado como um agente domiciliar pelo nó móvel. Um ou mais MAPs podem existir em uma rede visitada.

**Nó móvel HMIPv6:** é um nó móvel capaz de receber e processar mensagens do roteador padrão que contenham a opção MAP. Um nó móvel HMIPv6 deve ser apto para enviar *Binding Updates* locais com o bit M ligado.

**Regional care-of-address (RCoA):** é o endereço obtido pelo nó móvel na rede visitada com o mesmo prefixo de rede do MAP. Ele é auto-configurado quando recebe a opção de MAP divulgada nas mensagens do roteador padrão.

**Link care-of-address (LCoA):** é o endereço configurado pelo nó móvel com o mesmo prefixo do roteador padrão. Uma simples referência ao care-of-address.

**Binding Update local:** é o *Binding Update* que o nó móvel envia para o MAP fazer a associação do LCoA com o RCoA.

## 6.2 Funcionamento

A principal mudança no HMIPv6 em relação ao MIPv6 foi à introdução de um novo agente, o MAP que nada mais é do que um roteador utilizado para gerenciar a mobilidade. O funcionamento do agente domiciliar e do nó corresponde permanece idêntico ao do MIPv6. Assim como no MIPv6 o funcionamento do HMIPv6 independe da tecnologia de acesso.

Assim que um nó móvel entra em um *site* com a presença de um MAP, ele se conecta com um RA e auto-configura dois endereços o RCoA baseado no mesmo prefixo de rede do MAP e outro o LCoA com o mesmo prefixo do RA. Então o nó móvel envia um *Binding Update* para o MAP para que este faça a associação do RCoA com o LCoA. Outros *Binding Updates* também são enviados para o agente domiciliar e os nós correspondentes, neste caso para que estes façam a associação do endereço domiciliar com o RCoA. Se o nó móvel estiver se comunicando com correspondentes locais, ou seja, do mesmo *site* ele deve enviar um *Binding Update* para a associação do endereço domiciliar com o LCoA.

O funcionamento do MAP é semelhante ao do agente domiciliado, ele intercepta os pacotes originados do agente domiciliado e dos nós correspondentes destinados ao RCoA e então os envia via túnel ao LCoA. Um túnel bidirecional é estabelecido entre o MAP e o nó móvel, que é usado para enviar todos os pacotes oriundos do nó móvel, exceto os com destino aos correspondentes locais.

As principais formas de atuação propostas pelo protocolo são quando o nó móvel se desloca dentro de um mesmo *site* (mobilidade local) e entre *sites* (mobilidade global):

**Mobilidade local:** quando o nó móvel troca somente o seu LCoA, é necessário o envio de *Binding Updates* para o MAP e os correspondentes locais. Como resultado os pacotes com origem do agente domiciliado e dos nós correspondentes ainda são enviados para o mesmo RCoA, assim minimizando o tempo do *handover* e a perda de pacotes, também notamos que todo tráfego gerado é dentro do mesmo *site*.

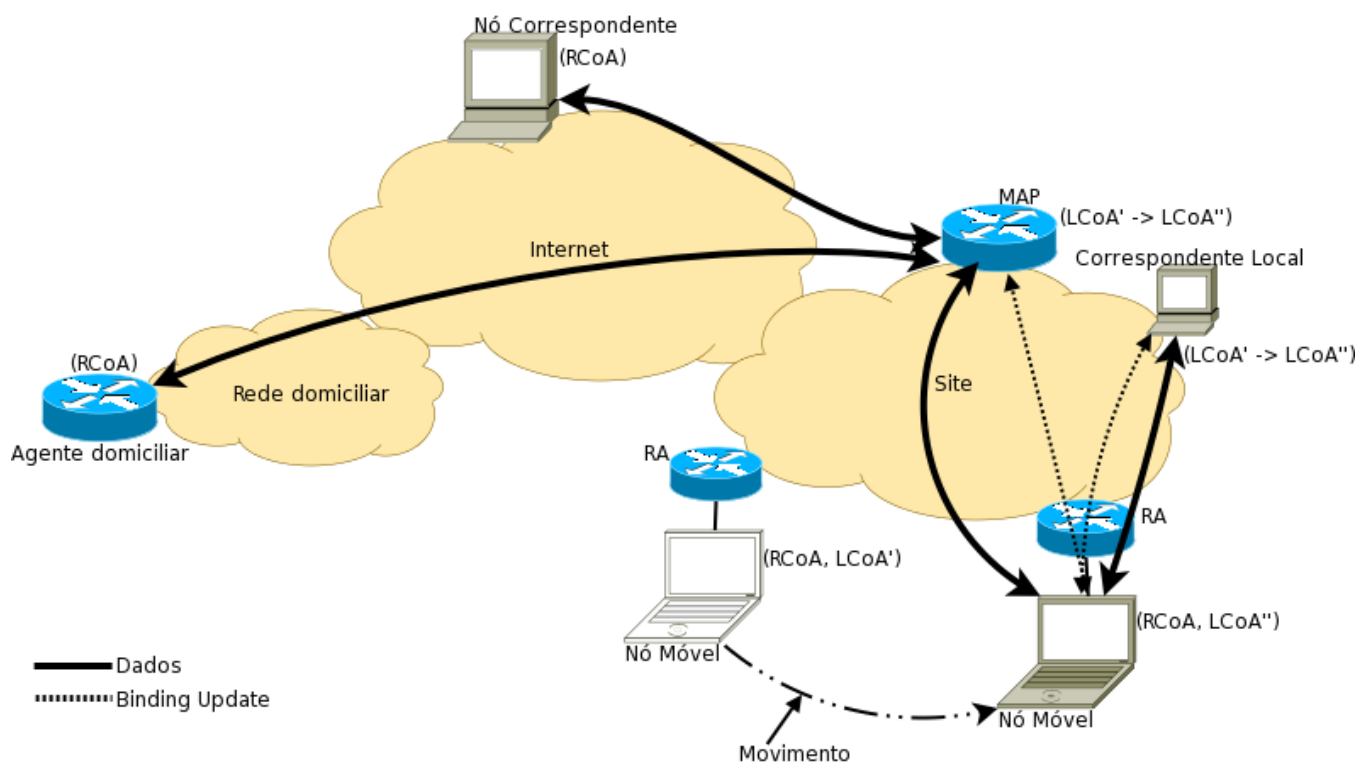


Figura 6.1: Mobilidade Local

**Mobilidade global:** neste caso o nó móvel reconfigura o seu RCoA e LCoA, e envia *Binding Updates* para o MAP, os nós comunicantes fora do *site* informando o novo RCoA e os nós do mesmo site para informar o LCoA.

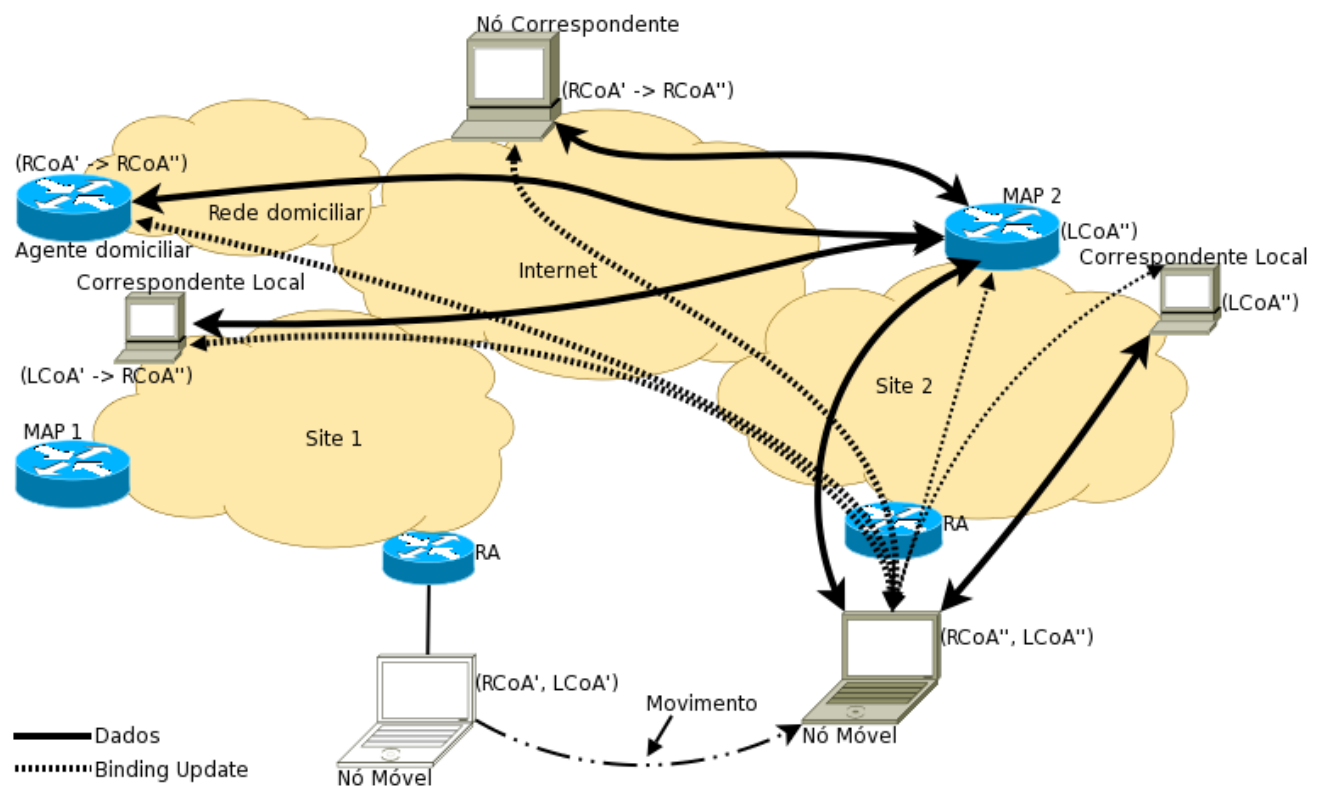


Figura 6.2: Mobilidade Global

Podemos perceber que em algumas vezes o nó móvel deve preferir atuar como no MIPv6 para tornar mais simples o seu funcionamento. Neste caso a rede visitada se encontra próxima ao agente domiciliar e não se faz necessário o uso do MAP, a discussão sobre este tipo de cenário não faz parte do escopo deste documento.

### 6.3 Envio de *Binding Updates*

Após o evento de mobilidade e o nó móvel auto-configurou o seu RCoA ele precisa enviar um *Binding Update* para o MAP. O pacote desta mensagem deve seguir os seguintes requisitos, definidos em:

- bit A (*acknowledge*) deve estar ligado;
- bit M (*map registration*) deve estar ligado;
- o RCoA na opção de agente domiciliar;
- o LCoA deve ser usado como endereço de origem do *Binding Update*;

O *Binding Update* irá associar o RCoA com o LCoA, similarmente como um endereço domiciliar. O MAP fará o teste de duplicidade (DAD) do RCoA, somente no primeiro *Binding Update*, e retornará um *Binding Acknowledgement* para o nó móvel.

A especificação do protocolo permite que um nó móvel tenha mais de um RCoA se recebeu mais de uma opção de MAP, para todos os RCoAs devem ser feito um *Binding Update*. O nó móvel não deve fazer um multi-nível hierárquico, pois irá diminuir a eficiência e a performance do protocolo muitos

Depois de fazer o registro com o MAP o nó móvel deve fazer a associação do RCoA com o endereço domiciliar no agente domiciliar e nos nós correspondentes. A mensagem de *Binding Update* deve ser enviada utilizando a opção de *care-of-address* alternativo preenchida com o RCoA e o tempo de vida deve ser menor do recebido no *Binding Acknowledgement* do registro com o MAP.

## 6.4 Descoberta de um MAP

Esta seção descreve como um nó móvel obtém o endereço do MAP e seu prefixo de rede, e como ARs presentes no *site* descobrem os MAPs. Dois métodos são definidos para o descobrimento do MAP:

**Descobrimento dinâmico:** é baseado na propagação da opção de MAP nas mensagens de *Router Advertisements*. Na opção de MAP estão presentes o endereço global do MAP, seu prefixo de rede, um vetor de distância baseado nos saltos da mensagem até a chegada no nó móvel, esta opção influenciará na escolha do MAP padrão, e um MAP particular como preferência. Neste método os ARs devem ser configurados para receberem as mensagens com opções de MAP e re-enviarem incrementando o vetor de distância.

**Configuração manual:** neste método a opção de MAP é configurada manualmente nos ARs por um administrador da rede, em muitos *sites* esse é o mecanismo padrão. Também deve ser possível a configuração por mecanismos dinâmicos.

## **7      *Conclusões***

Este trabalho procurou mostrar como deverá ser a apresentação da monografia a ser submetida à Coordenação do Curso Superior de Tecnologia em Sistemas de Telecomunicações do Centro Federal de Educação Tecnológica de Santa Catarina para a obtenção do diploma de Tecnólogo em Sistemas de Telecomunicações.

No capítulo 1 foi feita uma pequena introdução. No capítulo foram apresentados alguns comentários sobre figuras. E no capítulo foi apresentada uma forma para inserir tabela.

Como trabalho futuro, fica a reescrita do texto deste documento de forma que ele possam indicar informações específicas a formatação do documento. Como o tamanho da fonte utilizada, o espaçamento da borda, o alinhamento e numeração das seções e capítulos, etc.



## ***ANEXO A – Gerador de tráfego***

```
/*
 * gen-0.1
 *
 * (c) 2008 Cleiber Marques
 *
 * This program is free software; you can redistribute it and /or
 * modify it under the terms of the GNU General Public License
 * as published by the Free Software Foundation; either version
 * 2 of the License, or (at your option) any later version.
 *
 */

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <time.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>

#define PORT 4950
#define SIZE_PACK 100
#define IPV6
```

```
/*#define LOOP*/

int main(int argc, char **argv)
{
    int fd, numbytes, len;
    char buf[SIZE_PACK];
#ifdef IPV6
    struct sockaddr_in6 gen, rcv;
#else
    struct sockaddr_in gen, rcv;
#endif
    struct tm *local;
    time_t t;
    char addr[INET6_ADDRSTRLEN];
    /* Cria um socket UDP */
#ifdef IPV6
    printf("Program version IPv6\n");
    if (!(fd = socket(AF_INET6, SOCK_DGRAM, 0))) {
#else
    printf("Program version IPv4\n");
    if (!(fd = socket(AF_INET, SOCK_DGRAM, 0))) {
#endif
        fprintf(stderr, "Error in create socket\n");
        exit (1);
    }
    bzero(&gen, sizeof(gen));
#ifdef IPV6
    gen.sin6_family = AF_INET6;
    gen.sin6_port = htons(PORT);
#else
    gen.sin_family = AF_INET;
    gen.sin_port = htons(PORT);
#endif
    if(argc == 1) {
#ifdef IPV6
```

```
    gen.sin6_addr = in6addr_any;
#else
    gen.sin_addr.s_addr = INADDR_ANY;
#endif

    /* Bind */
    if (bind(fd, (struct sockaddr *)&gen, sizeof(gen)) == -1) {
        fprintf(stderr, "Error in bind\n");
        exit(1);
    }

    while(1) {
        len = sizeof(rcv);
        /* Recebe pacote */
        if ((numbytes=recvfrom(fd, buf, SIZE_PACK , 0,
            (struct sockaddr *)&rcv, &len)) == -1) {
            fprintf(stderr,"Error in recvfrom\n");
            exit(1);
        }

        /* Pega a hora do sistema e imprime */
        t = time(NULL);
        local = localtime(&t);

#ifdef LOOP
        /* Envia pacote */
        if ((numbytes = sendto(fd, buf, SIZE_PACK, 0,
            (struct sockaddr *)&rcv, sizeof(gen))) == -1)
            fprintf(stderr, "Error in sendto\n");
#endif

        /* Imprime log */
#ifdef IPV6
        inet_ntop(AF_INET6, &rcv.sin6_addr, addr, sizeof(addr));
        printf("Receive packet [%d:%d:%d] %d bytes from %s seq=%d\n", local->tm_hour,
            local->tm_min, local->tm_sec, numbytes, addr, buf[0]);
#else
        printf("Receive packet [%d:%d:%d] %d bytes from %s seq=%d\n", local->tm_hour,
            local->tm_min, local->tm_sec, numbytes, inet_ntoa(rcv.sin_addr), buf[0]);
#endif
    }
}
```

```
    }
} else {
    int count = 0, i;
#ifdef IPV6
    inet_pton(AF_INET6, argv[1], &gen.sin6_addr);
#else
    inet_pton(AF_INET, argv[1], &gen.sin_addr);
#endif
    while(1) {
        srand(time(NULL));

        /* Completa o pacote com numeros randomicos */
        for(i=0; i<SIZE_PACK; i++)
            buf[i] = rand() % 10;
        /* Envia pacote */
        if ((numbytes = sendto(fd, buf, SIZE_PACK, 0,
            (struct sockaddr *)&gen, sizeof(gen))) == -1) {
            fprintf(stderr, "Error in sendto\n");
            exit(1);
        }
        /* Pega a hora do sistema e imprime */
        t = time(NULL);
        local = localtime(&t);
#ifdef LOOP
        len = sizeof(rcv);
        /* Recebe pacote */
        if ((numbytes=recvfrom(fd, buf, SIZE_PACK , 0,
            (struct sockaddr *)&rcv, &len)) == -1) {
            fprintf(stderr,"Error in recvfrom\n");
            exit(1);
        }
#endif
        /* Imprime log */
        printf("Send packet [%d:%d:%d] %d bytes from %s seq=%d\n",
            local->tm_hour, local->tm_min, local->tm_sec, numbytes, argv[1], buf[0]);
    }
}
```

```
        sleep(1);  
    }  
}  
}
```

## ***ANEXO B – Cenários***

```
#!/bin/sh
#
# cenario.sh: script que gera as máquinas e switches virtuais para os
#             dois cenários definidos no trabalho.
# Uso:  ./cenario.sh <cenario> <modo>
# cenario: numero do cenário, 1 ou 2;
# modo: modo de operação, 1 para tunelamento bidirecional,
# 2 para otimização de rotas.
#
WD=~ /vm
FSYS=DebianFS
LINUX=$WD/linux
CENARIO=$1
$MOD0=$2
echo "Iniciando Cenario $CENARIO"
echo "Limando cows"
rm $WD/cow-*
echo "Limando pipes"
rm /tmp/net*
echo $MOD0 > $WD/conf
echo "Lancamento dos switches"
xterm -geometry 80x5 -T NET100 -e "uml_switch_mob -mob -unix /tmp/net100.ctl" &
if [ $CENARIO -eq 1 ]
then
    xterm -geometry 80x5 -T NET200 -e "uml_switch -unix /tmp/net200.ctl" &
else
    xterm -geometry 80x5 -T NET200 -e "uml_switch_mob -mob -unix /tmp/net200.ctl" &
```

```
fi
    xterm -geometry 80x5 -T NETGEN -e "uml_switch -unix /tmp/netgen.ctl" &
    sleep 2
    echo "Lancamento das maquinas virtuais UML"
    xterm -T Mobile -e "$LINUX umid=MN ubda=$WD/cow-MN,$WD/$FSYS mem=128M \
    eth0=daemon,,,/tmp/net100.ctl" &
    sleep 1
    xterm -T AR1 -e "$LINUX umid=AR1 ubda=$WD/cow-AR1,$WD/$FSYS mem=128M \
    eth0=daemon,,,/tmp/net100.ctl eth1=daemon,,,/tmp/netgen.ctl" &
    sleep 1
    if [ $CENARIO -eq 1 ]
    then
        xterm -T Correspondent -e "$LINUX umid=CN ubda=$WD/cow-CN,$WD/$FSYS \
        mem=128M eth0=daemon,,,/tmp/net200.ctl" &
    else
        xterm -T 'Mobile 2' -e "$LINUX umid=MN2 ubda=$WD/cow-MN2,$WD/$FSYS \
        mem=128M eth0=daemon,,,/tmp/net200.ctl" &
    fi
    sleep 1
    xterm -T AR2 -e "$LINUX umid=AR2 ubda=$WD/cow-AR2,$WD/$FSYS mem=128M \
    eth0=daemon,,,/tmp/net200.ctl eth1=daemon,,,/tmp/netgen.ctl" &
    sleep 1
    xterm -T AR3 -e "$LINUX umid=AR3 ubda=$WD/cow-AR3,$WD/$FSYS mem=128M \
    eth0=daemon,,,/tmp/net100.ctl \ eth1=daemon,,,/tmp/netgen.ctl" &
    sleep 1
    if [ $CENARIO -eq 2 ]
    then
        xterm -T AR4 -e "$LINUX umid=AR4 ubda=$WD/cow-AR4,$WD/$FSYS mem=128M \
        eth0=daemon,,,/tmp/net200.ctl eth1=daemon,,,/tmp/netgen.ctl" &
    fi
    PID='ps x | grep mob | tail -2 | head -1 | cut -c 1-5'
    echo "Pressione <enter> para a mobilidade"
    read a
    echo $PID
    kill -USR1 $PID
```

```
echo "Pressione <enter> para retorno a rede"
read a
kill -USR1 $PID
echo "Pressione <enter> para parar a simulação"
read a
kill $(pidof linux)
kill $(pidof xterm)
echo "Fim da simulacao"

#!/bin/bash
#
# cenario.conf: script que configura as máquinas virtuais de acordo com sua
#                função na topologia do cenário.
# Nota: Deve ser copiado para o arquivo /etc/rc.local do sistema de arquivos.
#
mount none /host -t hostfs -o /home/user/vm
OPTION='cat /host/conf'
NODO=$(basename $(cat /proc/cmdline | cut -f 1 -d " " | cut -f 2 -d "=" \
    | cut -f 1 -d ","))
if [ "$NODO" = "cow-MN1" ]
then
    ip -6 route flush all
    ip link set dev lo up
    ip link set dev eth0 down
    ip link set dev eth0 up
    ip -6 addr add 2000:a::1/64 dev eth0
    /sbin/sysctl -w net.ipv6.conf.all.forwarding=0
    /sbin/sysctl -w net.ipv6.conf.all.autoconf=1
    /sbin/sysctl -w net.ipv6.conf.all.accept_ra=1
    /sbin/sysctl -w net.ipv6.conf.all.accept_redirects=1
    sleep 6 # Para garantir que home agent inicie antes
    if [ $OPTION -eq 1 ]
    then
        echo "Iniciando MIPv6 MN"
        /usr/local/sbin/mip6d -c /host/conf/mip6d.conf.MN &
    else
```



```
        echo "Iniciando MIPv6 MN com Route Optimization"
        /usr/local/sbin/mip6d -c /host/conf/mip6d.conf.MN &
    fi
elif [ "$NODO" = "cow-MN2" ]
then
    ip -6 route flush all
    ip link set dev lo up
    ip link set dev eth0 down
    ip link set dev eth0 up
    ip -6 addr add 2000:c::1/64 dev eth0
    /sbin/sysctl -w net.ipv6.conf.all.forwarding=0
    /sbin/sysctl -w net.ipv6.conf.all.autoconf=1
    /sbin/sysctl -w net.ipv6.conf.all.accept_ra=1
    /sbin/sysctl -w net.ipv6.conf.all.accept_redirects=1
    sleep 6 # Para garantir que home agent inicie antes
    if [ $OPTION -eq 0 ]
    then
        echo "Iniciando MIPv6 MN"
        /usr/local/sbin/mip6d -c /host/conf/mip6d.conf.MN2 &
    elif [ $OPTION -eq 1 ]
    then
        echo "Iniciando MIPv6 MN com Route Optimization"
        /usr/local/sbin/mip6d -c /host/conf/mip6d.conf.MN &
    fi
elif [ "$NODO" = "cow-AR1" ]
then
    ip -6 route flush all
    ip link set dev lo up
    ip link set dev eth0 down
    ip link set dev eth0 up
    ip link set dev eth1 down
    ip link set dev eth1 up
    ip -6 addr add 2000:a::2/64 dev eth0
    ip -6 addr add 2000:b::1/64 dev eth1
    /sbin/sysctl -w net.ipv6.conf.all.forwarding=1
```

```
/sbin/sysctl -w net.ipv6.conf.all.proxy_ndp=1
/sbin/sysctl -w net.ipv6.conf.all.autoconf=0
/sbin/sysctl -w net.ipv6.conf.all.accept_ra=0
/sbin/sysctl -w net.ipv6.conf.all.accept_redirects=0
ip -6 route add 2000:c::/64 via 2000:b::2
ip -6 route add 2000:d::/64 via 2000:b::3
ip -6 route add 2000:e::/64 via 2000:b::4
/usr/local/sbin/mip6d -c /host/conf/mip6d.conf.HA &
sleep 1
/usr/local/sbin/radvd -C /host/conf/radvd.conf.AR1 &
elif [ "$NODO" = "cow-AR2" ]
then
    ip -6 route flush all
    ip link set dev lo up
    ip link set dev eth0 down
    ip link set dev eth0 up
    ip link set dev eth1 down
    ip link set dev eth1 up
    ip -6 addr add 2000:c::2/64 dev eth0
    ip -6 addr add 2000:b::2/64 dev eth1
    /sbin/sysctl -w net.ipv6.conf.all.forwarding=1
    /sbin/sysctl -w net.ipv6.conf.all.autoconf=0
    /sbin/sysctl -w net.ipv6.conf.all.accept_ra=0
    /sbin/sysctl -w net.ipv6.conf.all.accept_redirects=0
    ip -6 route add 2000:a::/64 via 2000:b::1
    ip -6 route add 2000:d::/64 via 2000:b::3
    ip -6 route add 2000:e::/64 via 2000:b::4
    /usr/local/sbin/mip6d -c /host/conf/mip6d.conf.HA &
    sleep 1
    /usr/local/sbin/radvd -C /host/conf/radvd.conf.AR2 &
elif [ "$NODO" = "cow-AR3" ]
then
    ip -6 route flush all
    ip link set dev lo up
    ip link set dev eth0 down
```

```
ip link set dev eth0 up
ip link set dev eth1 down
ip link set dev eth1 up
ip -6 addr add 2000:b::3/64 dev eth1
ip -6 addr add 2000:d::2/64 dev eth0
/sbin/sysctl -w net.ipv6.conf.all.forwarding=1
/sbin/sysctl -w net.ipv6.conf.all.autoconf=0
/sbin/sysctl -w net.ipv6.conf.all.accept_ra=0
/sbin/sysctl -w net.ipv6.conf.all.accept_redirects=0
ip -6 route add 2000:a::/64 via 2000:b::1
ip -6 route add 2000:c::/64 via 2000:b::2
ip -6 route add 2000:e::/64 via 2000:b::4
sleep 1
/usr/local/sbin/radvd -C /host/conf/radvd.conf.AR3
elif [ "$NODO" = "cow-AR4" ]
then
    ip -6 route flush all
    ip link set dev lo up
    ip link set dev eth0 down
    ip link set dev eth0 up
    ip link set dev eth1 down
    ip link set dev eth1 up
    ip -6 addr add 2000:b::4/64 dev eth1
    ip -6 addr add 2000:e::2/64 dev eth0
    /sbin/sysctl -w net.ipv6.conf.all.forwarding=1
    /sbin/sysctl -w net.ipv6.conf.all.autoconf=0
    /sbin/sysctl -w net.ipv6.conf.all.accept_ra=0
    /sbin/sysctl -w net.ipv6.conf.all.accept_redirects=0
    ip -6 route add 2000:a::/64 via 2000:b::1
    ip -6 route add 2000:c::/64 via 2000:b::2
    ip -6 route add 2000:d::/64 via 2000:b::3
    sleep 1
    /usr/local/sbin/radvd -C /host/conf/radvd.conf.AR4
elif [ "$NODO" = "cow-CN" ]
then
```

```
ip -6 route flush all
ip link set dev lo up
ip link set dev eth0 down
ip link set dev eth0 up
ip -6 addr add 2000:c::1/64 dev eth0
/sbin/sysctl -w net.ipv6.conf.all.forwarding=0
/sbin/sysctl -w net.ipv6.conf.all.autoconf=1
/sbin/sysctl -w net.ipv6.conf.all.accept_ra=1
/sbin/sysctl -w net.ipv6.conf.all.accept_redirects=1
ip -6 route add ::/0 via 2000:c::2
if [ $OPTION -eq 2 ]
then
    echo "Iniciando MIPv6 CN com Route Optimization"
    /usr/local/sbin/mip6d -c \
        /host/conf/mip6d.conf.CN
fi
fi
```

## *Referências Bibliográficas*