Cleiber Marques da Silva
Filipe Medeiros de Almeida

Plataforma para o Estudo de Mobilidade na Camada de Rede

São José – SC

Março / 2009

Cleiber Marques da Silva
Filipe Medeiros de Almeida

Plataforma para o Estudo de Mobilidade na Camada de Rede

Monografia apresentada à Coordenação do Curso Superior de Tecnologia em Sistemas de Telecomunicações do Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina para a obtenção do diploma de Tecnólogo em Sistemas de Telecomunicações.

Orientador:
Prof. Ms. Eraldo Silveira e Silva

CURSO SUPERIOR DE TECNOLOGIA EM SISTEMAS DE TELECOMUNICAÇÕES
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA

São José – SC

Março / 2009

Monografia sob o título *"Plataforma para o Estudo de Mobilidade na Camada de Rede"*, defendida por Cleiber Marques da Silva e Filipe Medeiros de Almeida e aprovada em 06 de Março de 2009, em São José, Santa Catarina, pela banca examinadora assim constituída:

Prof. Ms. Eraldo Silveira e Silva
Orientador - IF-SC - Campus S]ao José

Prof. Ms. Marcelo Sobral
IF-SC - Campus São José

Prof. Dr. Roberto Willrich
INF-UFSC-SC

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

Agradecimentos

Aos meus queridos pais e familiares, que nunca mediram esforços para que eu pudesse atingir meus objetivos. Além do apoio, confiança, dedicação e presença constante em todos os momentos da minha vida.

Aos meus amigos que ao longo dos últimos anos, trabalhei, aprendi, festejei, troquei ideias. A estes quero agradecer de forma especial pelo incentivo, apoio e amizade.

Ao professor Eraldo, pelos esclarecimentos das dúvidas, todos os comentários sempre úteis, coloboração e dedicação. Além da prontidão para me auxiliar em todos os momentos durante a elaboração deste trabalho.

Cleiber Marques da Silva

Aos meus pais e familiares, pela base sólida que sempre me deu força para encarar a vida de frente. Principalmente aos meus pais, por serem um exemplo de força de vontade, mas, que acima de tudo, me ensinaram a percorrer meu próprio caminho.

Aos amigos com quem tive todo o prazer em trocar experiências, tanto em questões academicas quanto no amadurecimento como pessoa, sendo na nossa querida instituição de tijolos à vista, nas nossas festas ou nos bares da região.

A todos os professores que contribuíram decisivamente para a minha, e nossa, formação academica, profissional e pessoal. Em especial ao meu orientador professor Eraldo, por todo o conhecimento passado, pelas excelentes supervisões, orientação e por sempre acreditar no potencial dos alunos.

Filipe Medeiros de Almeida

Resumo

A investigação de tecnologias que possibilitam a mobilidade de terminais no âmbito da Internet vem ganhando um grande espaço na comunidade científica. As soluções de mobilidade na camada de rede, tal como o IPv6 Móvel e seus derivados, apresentam-se como uma forma transparente e elegante de tratar o deslocamento de um nó móvel entre sub-redes. Neste trabalho, descreve-se a implementação de uma plataforma de testes baseada em máquinas virtuais UML, com fins de facilitar os estudos dos referidos protocolos, mais especificamente do IPv6 móvel e IPv6 móvel hierárquico. Visto a ausência de um código recente deste último, também foi implementada uma versão experimental do IP Móvel Hierárquico. Como contribuição adicional, são apresentadas análises preliminares do funcionamento e desempenho dos protocolos estudados, a partir de cenários de testes realizados sobre a plataforma.

Palavras-chave: IP Móvel, IP Móvel Hierárquico, IP Móvel para o Linux, Linux Modo Usuário, Plataforma de Testes.

Abstract

The investigation of technologies that enable the mobility of terminals in the Internet has gained a large space in the scientific community. The solutions for mobility in the network layer, such as Mobile IPv6 and its derivatives, appear as a transparent and elegant way to deal with the displacement of a mobile node between subnets. This work describes the implementation of a test platform based on UML virtual machines with the purpose of facilitating studies of these protocols, specifically the Mobile IPv6 and the Hierarchical Mobile IPv6. The lack of an updated code for the last one motivated an implementation of a experimental version of the Hierarchical Mobile IP. As an additional contribution, it is discussed preliminary results on the operation and performance of the studied protocols, extracted from test scenarios performed on the platform.

Keywords: Mobile IP, Mobile IP Hierarchical, Mobile IP for Linux, User Mode Linux, Platform of tests.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

AH *Authentication Header*

API *Application Programming Interface*

ARP *Address Resolution Protocol*

BA *Binding Acknowledgement*

BE *Binding Error*

BGP *Border Gateway Protocol*

BU *Binding Update*

CoA *Care-of-Address*

CoTI *Care-of Test Init*

CoT *Care-of Test*

DAD *Duplicate Address Detection*

DHCPv6 *Dynamic Host Configuration Protocol version 6*

ESP *Encapsulating Security Payload Header*

FMIPv6 *Fast Handovers for Mobile IPv6*

GPLv2 *General Public License version 2*

GUML *GUI Management console for User Mode Linux*

GUML4MIP *GUI Management console for User Mode Linux for Mobile IP*

HMIPv6 *Hierarchical Mobile Internet Protocol version 6*

HoTI *Home Test Init*

HoT *Home Test*

HTTP *HyperText Transfer Protocol*

ICMPv4 *Internet Control Message Protocol version 4*

ICMPv6 *Internet Control Message Protocol version 6*

IETF *Internet Engineering Task Force*

IGMP *Internet Group Management Protocol*

IP *Internet Protocol*

IPSec *Internet Protocol Security*

IPv4 *Internet Protocol version 4*

IPv6 *Internet Protocol version 6*

Kbm *Binding Management Key*

LCoA *Link Care-of-Address*

MAC *Media Access Control*

MAP *Mobile Anchor Point*

MIP *Mobile Internet Protocol*

MIPv6 *Mobile Internet Protocol version 6*

MTU *Maximum Transmit Unit*

NA *Neighbor Advertisement*

ND *Neighbor Discovery*

NS *Neighbor Solicitation*

OSPF *Open Shortest Path First*

POSIX *Portable Operating System Interface*

PPPv6 *Point to Point Protocol version 6*

PyGTK *Python Gimp Tool Kit*

QoS *Quality of Service*

RA *Router Advertisement*

RD *Router Discovery*

RADVD *Router Advertisement Daemon*

RCoA *Regional Care-of-Address*

RFC *Request For Comments*

RRP *Return Routability Procedure*

RS *Router Solicitation*

RIP *Routing Information Protocol*

RPDB *Routing Policy Database*

TCP/IP *Transmission Control Protocol over Internet Protocol*

UML *User Mode Linux*

VLAN *Virtual Local Area Networks*

1 Introdução

1.1 Motivação

A utilização de dispositivos móveis para o acesso a Internet vem crescendo acentuadamente nos últimos anos. As novas aplicações multimídias incentivam esta expansão, aliadas a uma variedade de novas tecnologias de acesso sem fio que permitem suportar taxas de transmissão em níveis relativamente altos. Algumas destas tecnologias, consideradas como camada de enlace e física do ponto de vista da arquitetura *Transmission Control Protocol over Internet Protocol* (TCP/IP), possuem mecanismos para o tratamento de movimentos entre estações base, minimizando os efeitos destas operações para o usuário.

Contudo, a utilização de redes *Internet Protocol* (IP) em situações de mobilidade pode impactar consideravelmente as camadas superiores, independentemente da tecnologia de acesso. Comunicações em andamento podem ser interrompidas de forma brusca e novas comunicações podem ser inviabilizadas quando um terminal móvel se movimenta para uma nova sub-rede IP. Este fato advém do caráter de posicionamento topológico dos endereçamentos IP. Uma mudança de sub-rede inviabiliza, a priori, a utilização de um endereço IP da rede de origem, dado que os protocolos de roteamento subjacentes não podem atualizar as rotas em tempo real, evidenciando um problema de escalabilidade.

Uma série de propostas da *Internet Engineering Task Force* (IETF) vem de encontro a este problema. Estas propostas orbitam em torno do *Mobile Internet Protocol* (MIP), nas versões 4 e 6. O estudo e teste destes protocolos envolve, normalmente, arranjos relativamente complexos, com estruturas de rede sem fio além de entidades associadas aos protocolos de mobilidade da camada de rede. Esta constatação é um dos fatores primeiros que motivaram o desenvolvimento deste trabalho.

1.2 Objetivos

O presente trabalho visa o desenvolvimento de uma plataforma para o estudo de protocolos de mobilidade na camada de rede, em particular os protocolos *Mobile Internet Protocol version 6* (MIPv6) e *Hierarchical Mobile Internet Protocol version 6* (HMIPv6).

Como especificações básicas desta plataforma pode-se enumerar:

- a execução real dos protocolos, de forma que se possa configurá-los e executá-los de forma muito próxima de cenários reais;
- facilidades de construção de cenários, sem a necessidade de arranjos de rede sem fio, reproduzindo, no entanto, os movimentos que impactam a camada de rede;
- pronta disponibilidade de ferramentas de medição, análise de tráfego, bem como protocolos de roteamento dinâmicos;

1.3 Organização do texto

Este trabalho está organizado da seguinte forma. O capítulo 2 resume os dois protocolos de mobilidade de interesse: o MIPv6 e sua extensão, o HMIPv6. O capítulo 3 apresenta a plataforma de mobilidade desenvolvida com apoio de máquinas virtuais. Tendo em vista que foram realizadas modificações em um código aberto do MIPv6, para que se comportasse tal como o HMIPv6, no capítulo 4 é apresentada uma visão geral deste código, bem como as modificações realizadas no mesmo para a obtenção do HMIPv6. No capítulo 5 a plataforma de mobilidade é explorada para a construção de alguns cenários de mobilidade com o MIPv6 e o HMIPv6. O capítulo 6 conclui e apresenta as perspectivas futuras de trabalhos usando a plataforma.

2 *Protocolos de Mobilidade da camada Rede*

2.1 O Problema da Mobilidade na Camada de Rede

Em redes IP, cada pacote integrante de um fluxo de pacotes entre dois pontos comunicantes, é encaminhado em função do seu endereço de IP destino. Cabe ao protocolo de camada de rede, chamado *Internet Protocol* (IP) (POSTEL, 1981), determinar rotas e encaminhar pacotes para que eles alcancem o seu destino. Em muitos casos, quando as redes são complexas, os protocolos de roteamento proporcionam a descoberta da localização dos destinos dinamicamente. Alguns exemplos de protocolos de roteamento são o *Open Shortest Path First* (OSPF), o *Routing Information Protocol* (RIP) e o *Border Gateway Protocol* (BGP).

Os endereços IP acabam definindo a localização geográfica de um hospedeiro. Os protocolos atuais da *Internet* assumem que o nó não muda seu endereço IP durante uma comunicação, ou seja, não altera o seu ponto de conexão à rede.

Caso um nó mude seu ponto de conexão, ele deverá, normalmente, configurar um novo endereço IP e, possivelmente, um novo roteador padrão. Se uma comunicação estiver estabelecida quando o nó efetuar a mobilidade, os protocolos de roteamento não corrigirão a rota para o novo destino. Para continuar a comunicação, sua nova rota deveria ser propagada para toda estrutura de roteamento da rede. Obviamente esta alternativa é inaceitável, pois seria inviável fazer isso em uma rede IP, por questões de escalabilidade.

Com a intenção de permitir que os nós possam se movimentar para diferentes subredes e continuar suas comunicações, a IETF propôs o *Mobile Internet Protocol* (MIP) que garante que os pacotes sejam roteados para os nós móveis. Existem duas variações para o MIP: uma para o *Internet Protocol version 4* (IPv4) (PERKINS, 2002) e outra para o *Internet Protocol version 6* (IPv6) (JOHNSON; PERKINS; ARKKO, 2004). O foco deste trabalho será na versão para o IPv6, pois este possui algumas vantagens sobre o IPv4 como maior número de endereços, suporte nativo para segurança, possibilidade de autoconfiguração e suporte ao MIP.

Em adição, a IETF propôs o HMIPv6 (SOLIMAN et al., 2005) que permite aumentar o desempenho do protocolo MIPv6 através do tratamento localizado da mobilidade um mesmo domínio. Na sequência, serão apresentados brevemente os referidos protocolos.

2.2 Visão Geral do Funcionamento do MIPv6

O protocolo IPv6 móvel (MIPv6) tem por objetivo, portanto, permitir que nós IPv6 se desloquem, de forma transparente, entre subredes, com diferentes tecnologias de acesso, tais como *Ethernet* e *Wireless LAN*, e mantenham as comunicações em andamento. Novas comunicações também serão possibilitadas. Sem suporte ao MIPv6 todos os pacotes destinados ao nó móvel, quando ele estiver fora de sua rede origem, seriam perdidos.

No contexto do MIPv6 um nó móvel está sempre acessível por um mesmo endereço IP, independente de seu ponto de conexão com a *Internet*. Este endereço é chamado de endereço domiciliar e é o endereço IPv6 fixo global do nó móvel na sua rede de origem. Todos os nós correspondentes utilizam este endereço para se comunicar com o nó móvel.

Quando um nó móvel estiver fora de sua rede de origem ele terá no mínimo dois endereços atribuídos a sua interface de rede:

1. Um endereço domiciliar que será permanente, e que, supostamente será utilizado pelos nós correspondentes em suas sessões de comunicação;

2. Um endereço chamado de *Care-of-Address* (CoA), que se refere a rede visitada. A cada nova rede será configurado um CoA.

O CoA pode ser obtido pelas formas convencionais do IPv6, ou seja, por *stateless autoconfiguration* (THOMSON; NARTEN, 1998), no qual o endereço é gerado por meio de informações divulgadas pelos roteadores das subredes, ou por *stateful autoconfiguration*, onde o endereço e outros parâmetros de configuração são providos diretamente de um servidor, por exemplo, os mecanismos *Dynamic Host Configuration Protocol version 6* (DHCPv6) e *Point to Point Protocol version 6* (PPPv6).

Na rede visitada, após detectar o movimento e configurar seu CoA, o nó móvel envia uma mensagem *Internet Control Message Protocol version 6* (ICMPv6) de *Binding Update* (BU) para seu Agente Domiciliar, para que este faça a associação de seu endereço domiciliar com o seu CoA. Esta informação é registrada em um *cache* local do Agente Domiciliar. Este, em resposta, envia um *Binding Acknowledgement* (BA) para o nó móvel. O Agente Domiciliar

passa, então, a atuar como um *proxy*, interceptando todos os pacotes com destino ao nó móvel e enviando-os, via túnel, para este nó. Opcionalmente, o nó móvel pode também informar ao nó correspondente sua localização. Neste caso, assim que o nó correspondente atualizar seu *cache*, ele encaminhará os pacotes diretamente ao nó móvel.

Duas formas de comunicação podem ser realizadas entre o nó móvel e o correspondente (ver figura 2.1). No primeiro modo, com tunelamento bidirecional, todos os pacotes são encaminhados via Agente Domiciliar e o nó correspondente não precisa ter suporte ao MIPv6. Os pacotes com origem no nó correspondente são roteados para o Agente Domiciliar e este envia ao nó móvel via túnel. No sentido contrário, pacotes enviados pelo nó móvel, 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. O Agente Domiciliar utiliza *proxy Neighbor Discovery* para interceptar os pacotes com destino ao nó móvel.

O segundo modo, mais eficiente, é chamado de *"otimização de roteamento"*. O nó móvel, adicionalmente ao registro com o Agente Domiciliar, também faz uma associação do seu CoA com o nó correspondente e este pode começar a endereçar os pacotes diretamente ao CoA, melhorando a escabilidade do protocolo e minimizando o tráfego de rede.

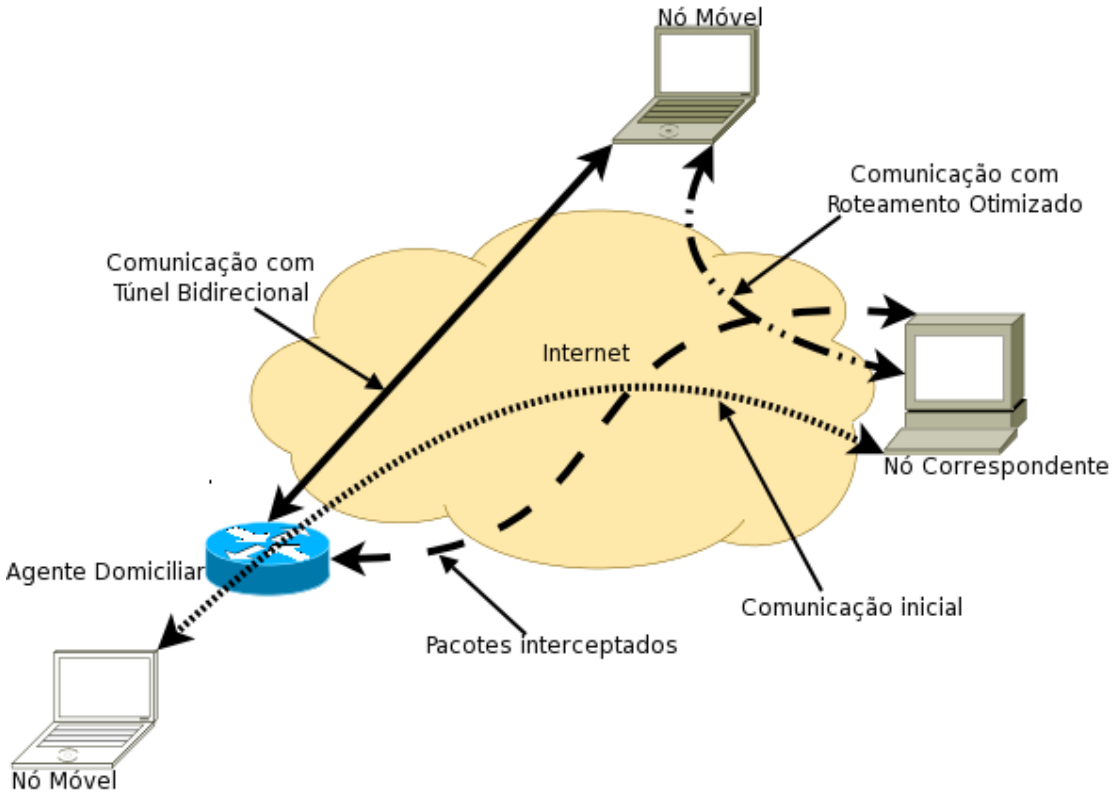

Figura 2.1: Modos de operação simplificados do MIPv6

Como citado anteriormente, 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 CoA. Ele poderá usar qualquer um destes como endereço fonte para se comunicar. Com o intuito de manter a transparência para as camadas superiores à de rede, o nó móvel irá utilizar geralmente seu endereço domiciliar. É interessante observar que os pacotes enviados pelo nó móvel deverão ser modificados, inserindo-se o CoA no campo endereço de origem e movendo o endereço domiciliar para campo *home address*, em um novo tipo de cabeçalho de extensão de roteamento(KOODLI; PERKINS, 2007). No receptor do pacote, estas alteraçõesdevem ser revertidas para manter a transparência para as camadas superiores.

2.2.1 O Procedimento de Handover no MIPv6

Conforme a *Request For Comments* (RFC) 3775 (JOHNSON; PERKINS; ARKKO, 2004), o *Handover* de camada 3 pode ser definido como o processo em que o nó móvel muda seu ponto de acesso, percebe a mudança de subrede, configura um novo CoA e o registra com o seu Agente Domiciliar/correspondentes. Note que o gerenciamento do *handover* pode se dar em diferentes camadas da arquitetura de rede. O *handover* de camada enlace refere-se, por exemplo, a descoberta e conexão há um novo ponto de acesso sem necessariamente causar um *handover* de camada 3. Já o *handover* na camada rede, no caso da arquitetura IPv6, envolve:

1. Descoberta de um novo roteador;
2. Auto configuração do CoA;
3. Teste de duplicidade do CoA (DAD), e;
4. Registro com o agente domiciliar e o nó correspondente.

O início de um *handover* de camada 3 se dá a partir do protocolo *Neighbor Discovery* (ND) presente no IPv6. Este protocolo utiliza-se, dentre outros, dos mecanismos *Router Discovery* (RD) para realizar a detecção de uma nova rede.

O RD (NARTEN; NORDMARK; SIMPSON, 1998) é o mecanismo que permite que nós IPv6 descubram roteadores existentes no seu enlace, através das mensagens *Router Advertisement* (RA) e *Router Solicitation* (RS). Um roteador IPv6 periodicamente envia mensagens de RA para todo o enlace. Desta forma, os nós podem configurar seu endereço de rede (*stateless autoconfiguration*) e o roteador padrão. O nó também pode enviar uma mensagem de RS recebendo como resposta um RA.

Desta forma, o nó móvel pode perceber o movimento quando receber um RA de um novo roteador ou quando perceber que seu roteador não esta mais alcançável. Neste caso, ele deve requisitar um novo roteador através da mensagem RS. Por meio das mensagens trocadas na descoberta do novo roteador, o nó móvel é capaz de gerar seu CoA. Então, um teste de duplicidade (DAD) deve ser efetuado para verificar se não há um endereço igual ao CoA no *link*. Com o sucesso do DAD, o registro com o Agente Domiciliar e o nó correspondente deve ser efetuado, finalizando o processo de *handover*.

Na figura 2.2 é possível observar um diagrama de sequência do funcionamento do MIPv6 abordado nesta seção. Neste cenário um nó correspondente se comunica com o nó móvel através de um fluxo de dados. Inicialmente, o nó está na sua rede domiciliar e depois mudará para outra rede.

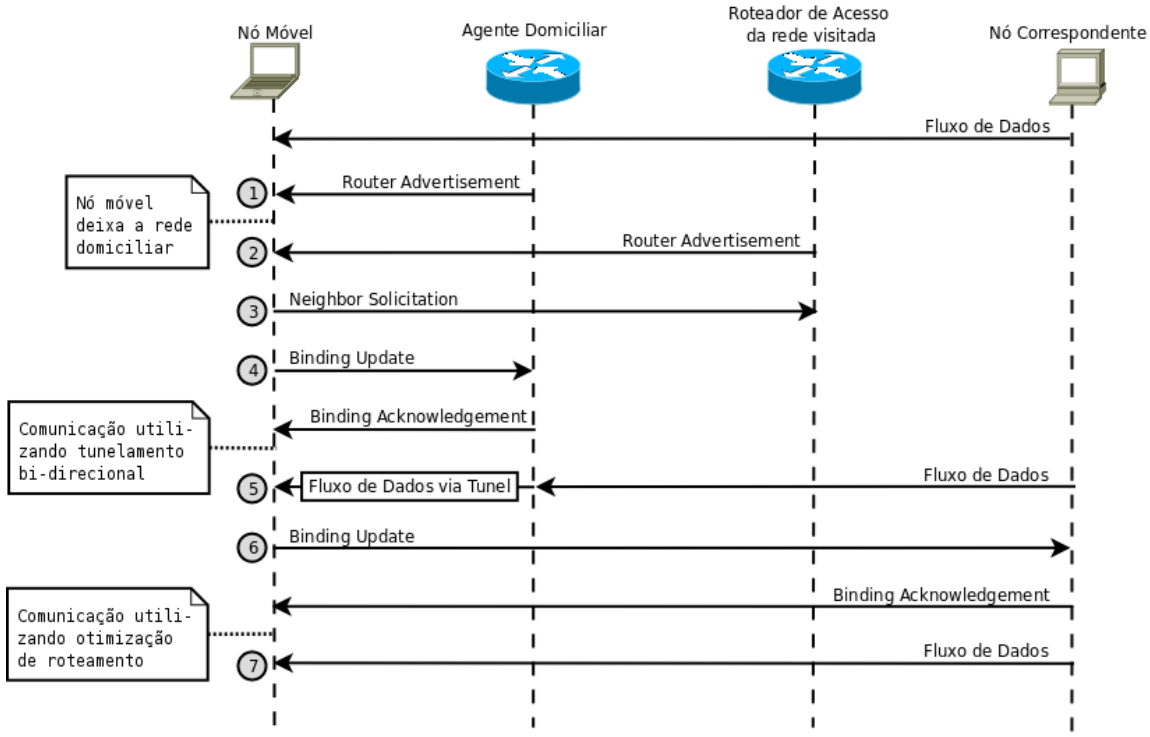

Figura 2.2: Diagrama de sequência do MIPv6

Após o fluxo de dados estabelecido, ocorrem as seguintes trocas de mensagens:

1. Inicialmente, o nó móvel recebe uma mensagem de RA do roteador de acesso da rede domiciliar. O nó compara os prefixos de seu endereço domiciliar e o prefixo do endereço IP divulgado pelo roteador de acesso e observa que está na sua rede domiciliar;

2. Até então o nó móvel estava trocando mensagens com o nó correspondente em sua rede domiciliar. Quando seu ponto de acesso muda, o movimento é detectado através da mensagem de RA do roteador de acesso da rede visitada;

3. Com as informações recebidas na mensagem do roteador, o nó móvel é capaz de gerar o seu CoA. Ele efetua, então, o teste de duplicidade de endereço com as mensagens de *Neighbor Solicitation* (NS);

4. Com o sucesso do DAD, o nó móvel faz o registro com o seu agente domiciliar, enviando o BU e recebendo como resposta um BA;

5. A partir deste ponto, o fluxo de dados entre o nó móvel e o nó correspondente é realizado através do tunelamento bidirecional, via Agente Domiciliar;

6. Para otimizar o roteamento, o nó móvel faz o registro do seu CoA com o nó correspondente. Assim, a comunicação entre eles se dá de forma normal, sem a necessidade de encaminhar os pacotes ao agente domiciliar;

7. O fluxo de dados é conduzido diretamente ao nó móvel tendo como endereço destino o CoA. O endereço domiciliar é conduzido no cabeçalho de mobilidade do IPv6 (*Mobility Header*).

2.2.2 Mensagens do MIPv6

O MIPv6 define um novo cabeçalho de extensão ao protocolo IPv6, o *Mobility Header*. Este cabeçalho de mobilidade é utilizado pelos nós móveis e correspondentes, bem como pelo agente domiciliar, para construir as mensagens utilizadas no MIPv6.

As mensagens enviadas utilizando o cabeçalho de mobilidade são as seguintes:

1. ***Binding Refresh Request* (BRR)**: mensagem enviada pelos nós correspondentes ao nó móvel, para requisitar a atualização do registro de mobilidade;

2. ***Home Test Init* (HoTI)**: mensagem enviada pelo nó móvel para o nó correspondente, via agente domiciliar, utilizada para iniciar o procedimento de *Return Routability Procedure* (RRP). Requisita o *home keygen token*. utilizado pelo nó móvel para gerar a chave *Binding Management Key* (Kbm);

3. ***Care-of Test Init* (CoTI)**: mensagem enviada pelo nó móvel para o nó correspondente, pertencente ao processo de *Return Routability*. Requisita o *care-of keygen token* utilizado pelo nó móvel para gerar a chave Kbm;

4. ***Home Test* (HoT)**: mensagem em resposta ao HoTI enviada pelo nó correspondente ao nó móvel;

5. ***Care-of Test*** **(CoT)**: mensagem em resposta ao CoTI enviada pelo nó correspondente ao nó móvel;

6. ***Binding Update*** **(BU)**: mensagem utilizada para notificar o agente domiciliar e os nós correspondentes, do novo CoA;

7. ***Binding Acknowledgement*** **(BA)**: mensagem utilizada para confirmar o BU;

8. ***Binding Error*** **(BE)**: mensagem enviada pelos nós correspondentes ao nó móvel, para informar erro na mobilidade relatada.

2.2.3 Considerações sobre Segurança

O uso de BUs 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 (ARKKO; DEVARAPALLI; DUPONT, 2004), o protocolo MIPv6 implementa alguns mecanismos.

Para dar mais proteção no processo de BU com o Agente Domiciliar pode se utilizar o protocolo *Internet Protocol Security* (IPSec) nativo no IPv6. A troca de mensagens é feita utilizando um dos cabeçalhos de segurança:

- *Authentication Header* (AH) (KENT; ATKINSON, 1998a): representa um cabeçalho de extensão do protocolo IPv6 e foi criado para identificar/autenticar os pontos comunicantes.

- *Encapsulating Security Payload Header* (ESP) (KENT; ATKINSON, 1998b): representa um cabeçalho de extensão do protocolo IPv6 que fornece integridade e confidencialidade aos datagramas IP por meio da cifra dos dados contidos no datagrama.

Para o uso de um dos cabeçalhos de segurança é necessário um relacionamento prévio de segurança entre o nó móvel e o agente domiciliar, 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 ser estar localizado em qualquer lugar na *Internet* e uma chave secreta não pode ser pré-configurada. 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 RRP.

Return Routability Procedure

Na tentativa de tornar mais seguro o registro do nó móvel com o nó correspondente o MIPv6 introduz o RRP (JOHNSON; PERKINS; ARKKO, 2004). A idéia do processo é que o nó correspondente obtenha garantias de que o nó móvel seja alcançável pelo seu endereço domiciliar e pelo seu CoA. Somente assim o nó correspondente estará apto para receber BUs.

O teste (NIKANDER et al., 2005) consiste em enviar, a partir do nó móvel, duas mensagens ao nó correspondente: uma via agente domiciliar (*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 Kbm.

Na figura 2.3 pode-se observar o fluxo das mensagens utilizados no RRP.

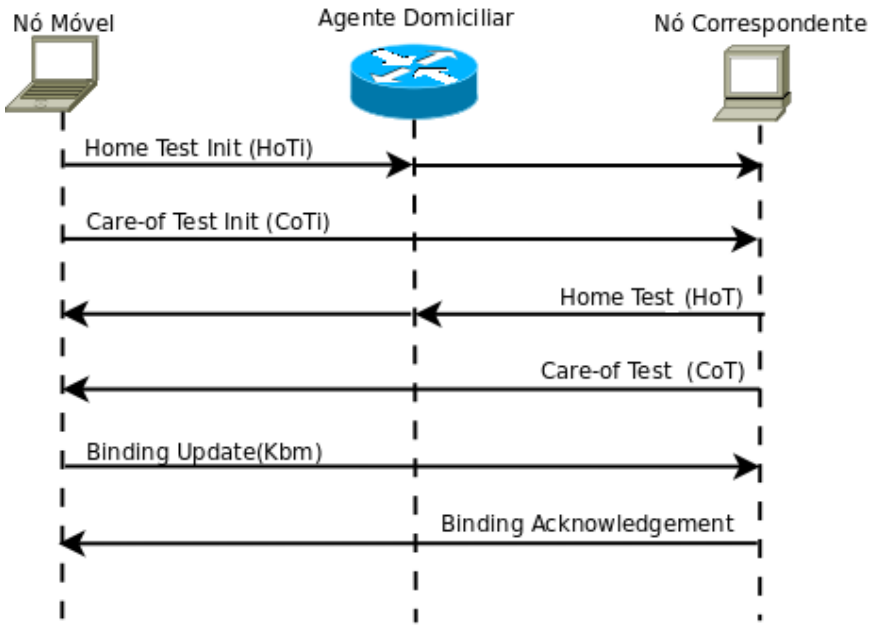

Figura 2.3: Fluxo das mensagens no Return Routability Procedure

Para autorizar o BU o nó móvel gerou a chave Kbm que permite uma verificação por parte do nó correspondente. Ao receber o BU o nó correspondente é capaz de recalcular a Kbm e permitir a associação do CoA com o endereço domiciliar.

É importante salientar que o RRP não é totalmente seguro, pois um atacante pode capturar as mensagens enviadas pelo nó correspondente e forjar mensagens de BUs. Obviamente o mesmo atacante conseguirá fazer o mesmo ataque em uma rede IPv6 sem mobilidade. Pode-se concluir que o RRP não introduz riscos adicionais ao protocolo IPv6 básico.

2.3 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 neste trabalho. Dentre estes problemas podemos destacar:

- tempo na detecção do movimento;
- tempo na configuração do novo endereço na rede visitada, o CoA;
- tempo do registro com o seu Agente Domiciliar;

Em um ambiente utilizando o MIPv6 com um excessivo número de nós móveis a tendência é que se tenha 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 BUs, para o registro com o Agente Domiciliar ou com os nós correspondentes e reduzir o tráfego de pacotes de controle na *Internet*, foi proposto um protocolo complementar ao MIPv6, o *Hierarchical Mobile Internet Protocol version 6* (HMIPv6) (SOLIMAN et al., 2005).

A idéia principal do protocolo é tratar a mobilidade global e local de formas distintas. Uma movimentação local pode ser entendida como a que acontece dentro de um mesmo domínio e a global entre domínios diferentes. Podemos definir um domínio como uma dimensão arbitrária, por exemplo, uma rede de uma empresa ou universidade com uma ou mais subredes.

Um estudo (KIRBY, 1995) 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 de sinalização enviado para à *Internet*.

2.3.1 Funcionamento do HMIPv6

A principal mudança no HMIPv6 em relação ao MIPv6 foi à introdução de um novo agente, o *Mobile Anchor Point* (MAP), que nada mais é do que um roteador utilizado para gerenciar a mobilidade em um determinado domínio. O funcionamento do Agente Domiciliar e dos nós correspondentes permanece idêntico ao do MIPv6. Assim como no MIPv6, o funcionamento do HMIPv6 não depende da tecnologia de acesso.

Quando um nó móvel entra em um domínio com a presença de um MAP, ele irá receber uma mensagem RA com informações sobre os roteadores de acesso. Isso permite que ele autoconfigure dois endereços: o *Regional Care-of-Address* (RCoA), que será baseado em um prefixo de uma rede do MAP, e o *Link Care-of-Address* (LCoA), que utilizará o mesmo prefixo anunciado pelo roteador padrão.

Após gerar os endereços RCoA e LCoA, o nó móvel irá atualizar suas associações. Ele enviará um *Binding Update* para o MAP, para que este faça a associação do RCoA com o LCoA. Outros BUs também serão enviados para o Agente Domiciliar e, opcionalmente, para 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 correspondes locais, ou seja, do mesmo domínio, ele deve enviar um BU para à associação do endereço domiciliar com o LCoA.

O funcionamento do MAP é semelhante ao do Agente Domiciliar: ele intercepta pacotes destinados ao RCoA, originados pelo tunelamento entre Agente Domiciliar e nó móvel, ou pelos pacotes provindos diretamente dos nós correspondentes. Neste último caso, o destino dos pacotes provindos dos nós correspondentes é o RCoA, sendo que o endereço domiciliar é colocado no cabeçalho de mobilidade do IPv6. Seja qual for o caso, o MAP envia os pacotes destinados ao RCoA pelo túnel bidirecional, estabelecido entre o MAP e o LCoA. Este túnel é também usado para enviar todos os pacotes oriundos do nó móvel, exceto os com destino aos correspondentes locais.

Dois comportamentos podem ser distinguidos no protocolo HMIPv6: quando o nó móvel se desloca dentro de um mesmo domínio (mobilidade local) e entre domínios diferentes (mobilidade global):

Mobilidade local: quando o nó móvel muda somente o seu endereço LCoA, ou seja, muda de roteador de acesso mas continua sob a cobertura do mesmo MAP. Neste caso, é necessário o envio de BUs para o MAP e os correspondentes locais, atualizando a associação RCoA-LCoA. Como resultado, os pacotes com origem no agente domiciliar e nos nós correspondentes continuarão a serem enviados para o mesmo RCoA, minimizando o tempo do *handover* e a perda de pacotes. Também notamos que todo tráfego gerado é dentro do mesmo domínio.

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

A figura 2.4 ilustra um exemplo de domínios HMIPv6 onde o nó móvel está inicialmente

atrelado a rede domiciliar. Em seguida, realiza mobilidade para o *domínio 1*, tendo como roteador padrão o RA1, e utilizando o MAP1. Após isso, o nó móvel realiza uma mobilidade local, mudando o roteador padrão para RA2. Por último, realiza uma mobilidade global, passando a ter o RA3 como roteador padrão, e passando a utilizar o MAP2.

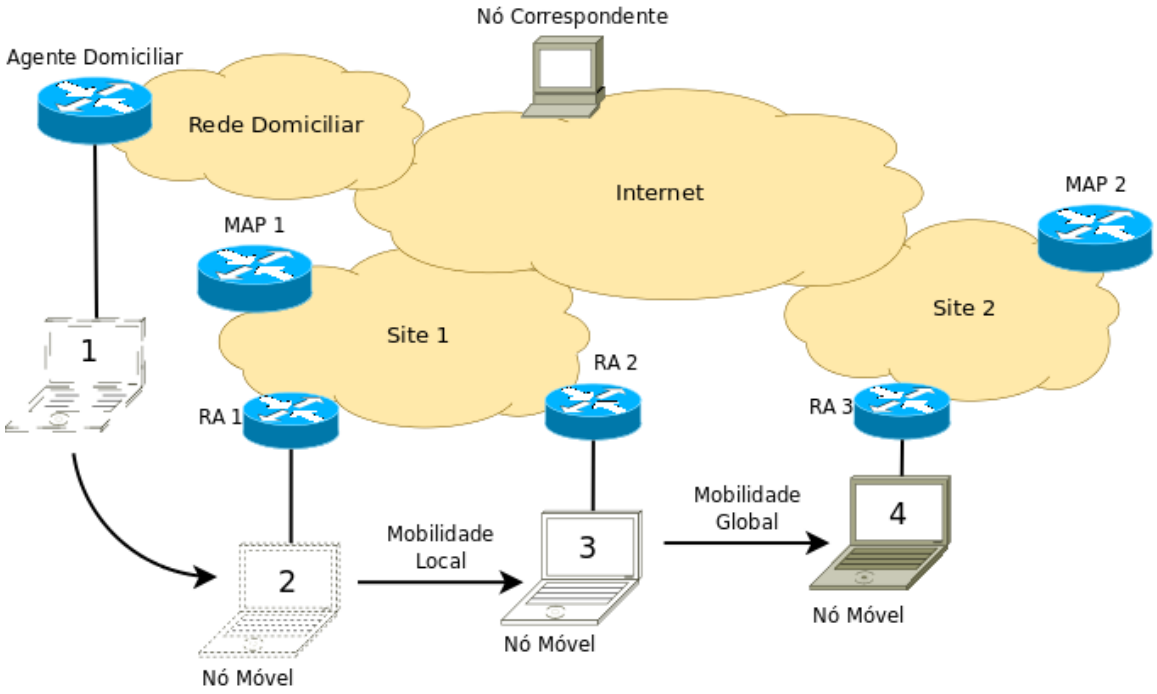

Figura 2.4: Exemplo de domínios do IP Móvel Hierárquico

A figura 2.5 apresenta a sequência de mensagens trocadas durante o movimento do nó móvel no cenário da figura 2.4, sendo suprimido os processos de DAD e RRP.

As trocas de mensagens que ocorrem na figura 2.5 são descritas a seguir:

1. Inicialmente o nó móvel está na rede domiciliar se comunicando com o um correspondente.

2. O MAP1 envia mensagens de RA. Os roteadores de acesso RA1 e RA2, sob a cobertura de MAP1, irão anexar as informações recebidas das mensagens RA do MAP 1 e, encaminhar para suas subredes, via RA. O nó móvel recebe uma mensagem de RA do RA1, realizando a mobilidade para a respectiva rede;

3. O nó móvel realizará o registro com o MAP1, enviando-lhe um BU, associando desta forma o endereço domiciliar com o LCoA1. Após efetuar o registro, o MAP1 enviará um BA;

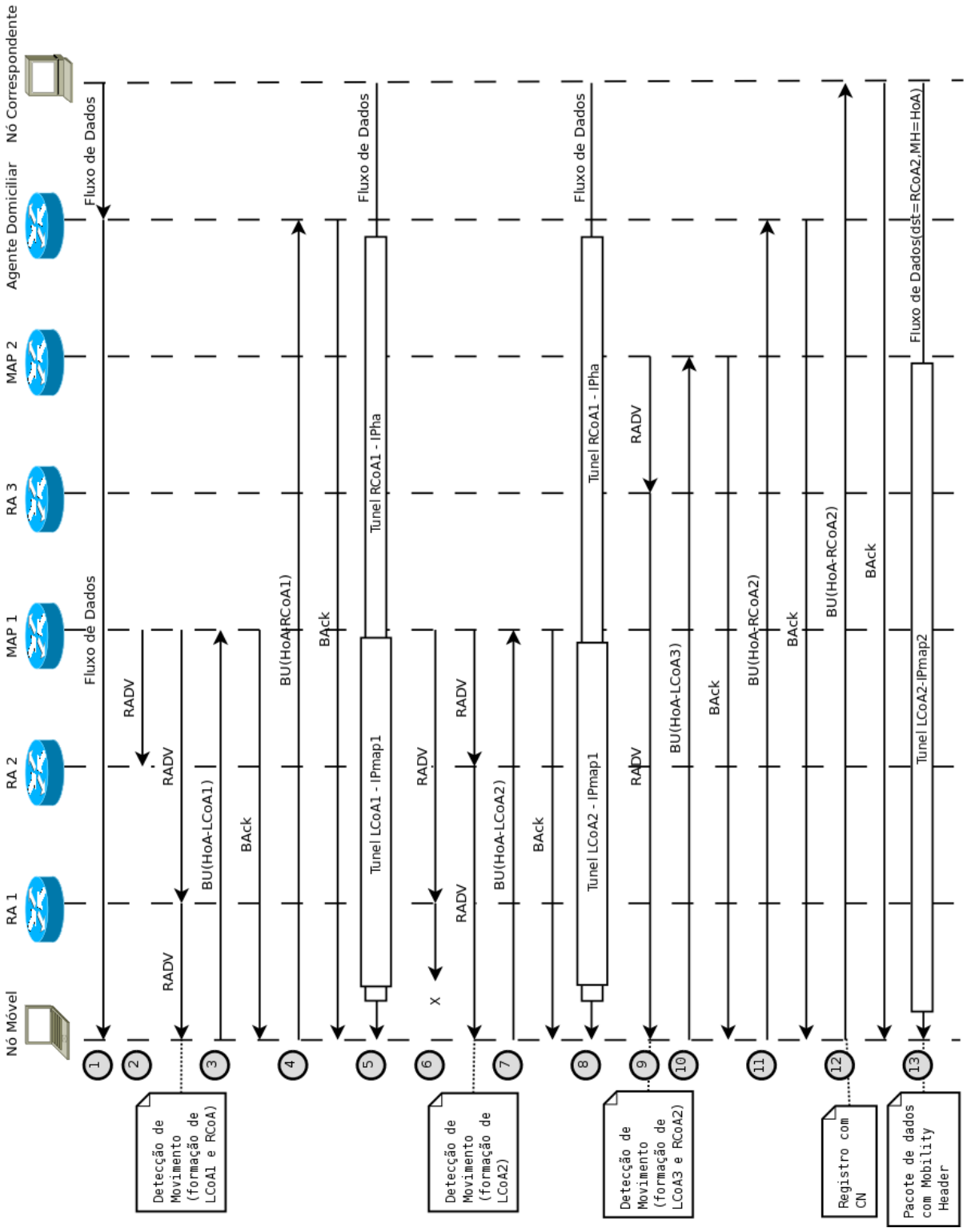

Figura 2.5: Diagrama de troca de mensagens do HMIPv6

4. O próximo nó com quem o nó móvel realizará o registro será o agente domiciliar. A diferença será que o registro irá associar o endereço domiciliar com o RCoA1;

5. O fluxo de dados percorre o seguinte caminho: sai do nó correspondente com destino ao endereço domiciliar. Como o nó móvel não está na rede domiciliar, o Agente Domiciliar interceptará os pacotes, e encaminhará para o endereço registrado, via túnel, no caso o RCoA1. Quando chegar no MAP1, será tunelado novamente até o LCoA1. O fluxo de dados passa pelo RA1, mas ele só encaminha o pacote para o nó móvel, onde termina os túneis;

6. O nó móvel se movimenta e sai do alcance do RA1 e entra na cobertura de RA2. Como foi feita uma mobilidade local, será necessário somente configurar um novo LCoA (LCoA2);

7. Será necessário atualizar o registro com o MAP1, associando o LCoA2 ao endereço domiciliar. Não é necessário novo registro com o MAP1;

8. O trajeto do fluxo de dados será alterado do MAP1 para frente, sendo transparente para o nó correspondente e para o Agente Domiciliar. O túnel entre o MAP1 e o nó móvel terá como destino o LCoA2, e passará pelo RA2, o roteador padrão do nó móvel.

9. O nó móvel recebe uma mensagem RA com informações de um outro MAP através do RA3, o que indica que ocorreu uma mobilidade global está em andamento. Um novo LCoA será formado (LCoA3) e um novo RCoA(RCoA2);

10. O primeiro registro será com o novo MAP, o MAP2. A associação feita será do endereço domiciliar com o LCoA3;

11. O registro no Agente Domiciliar será atualizado, associando o endereço domiciliar com o RCoA2;

12. O registro com o nó correspondente é realizado, associando o endereço domiciliar com RCoA2. Após o registro, o agente domiciliar será utilizado somente por outro correspondente para encontrar o nó móvel e estabelecer uma comunicação;

13. O fluxo de dados será endereçado ao RCoA2, e utilizará um cabeçalho de mobilidade, indicando que o pacote é endereçado ao endereço domiciliar do nó móvel.

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.

2.3.2 Envio de *Binding Updates*

Após o evento de mobilidade global, o nó móvel autoconfigura um novo RCoA e um novo LCoA e ele precisa enviar um BU para o MAP. O BU irá associar o RCoA com o LCoA, de forma similar a associação do CoA com o endereço domiciliar no MIPv6. O MAP fará o DAD do RCoA, somente no primeiro BU, e retornará um BA para o nó móvel.

A especificação do protocolo permite que um nó móvel tenha mais de um RCoA. Isto se aplica para o caso deste ter recebido mais de uma opção de MAP. Nesta situação, deverá ser feito um Binding Update para cada RCoA .

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 BU deve ser enviada utilizando a opção de CoA alternativo preenchida com o RCoA e o tempo de vida deve ser menor que o do recebido no BA do registro com o MAP.

2.3.3 Descoberta de um MAP

O nó móvel e os roteadores de acesso deve obter de alguma forma o endereço do e o prefixo de rede do MAP. Dois métodos são definidos para o descobrimento do MAP:

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

Configuração manual: neste método, a opção de MAP é configurada manualmente nos roteadores de acesso, por um administrador da rede. Em muitos domínios esse é o mecanismo padrão.

3 *Plataforma de Testes para Mobilidade em Redes IP*

3.1 Descrição Geral da Plataforma

A plataforma desenvolvida neste trabalho tem como objetivo prover o usuário com:

- um sistema que permita facilidades no estudo dos protocolos de mobilidade da camada de rede, em particular o MIPv6 e HMIPv6. Outros protocolos poderão ser instalados no futuro. Em adição, os protocolos não devem ser executados como simulações de comportamento, mas sim na forma de implementações reais;
- facilidades para a rápida construção de cenários de rede, sem se ater a procedimentos de configuração e instalação de equipamentos de rede sem fio;
- facilidades para a rápida instalação da plataforma nas máquinas hospedeiras.

Os requisitos básicos do projeto descartam a utilização de simuladores de rede como NS2 (**??**) e OMNeT (VARGA et al., 2001):

- O OMNeT é um ambiente de simulação de eventos discretos com código aberto. É baseado em componentes(modular) e possui uma interface gráfica para vizualização da execução. Sua principal aplicação é a simulação de redes de comunicação, mas devido à sua arquitetura flexível e genérica, é utilizado com sucesso também em outras áreas como a simulação de sistemas complexos de TI, redes de filas ou arquiteturas de hardware.
- O NS2 é..

A utilização destes simuladores é interessante para a análise de desempenho dos protocolos mas não refletem os problemas reais de configuração e execução do sistema, uma vez que tendem a abstrair detalhes de funcionamento considerados irrelevantes na simulação. Além

disto, as implementações do MIP e HMIP sobre estes simuladores são bastantes precárias. O NS2 não possui nem mesmo uma versão estável do IPv6.

Na concepção da plataforma de testes, optou-se por utilizar um ambiente baseado em máquinas virtuais devido a rapidez na realização de experimentos: em somente uma máquina de hospedagem é possível o desenvolvimento, análise e testes de protocolos de rede. Além disto, existem vantagens econômicas e de redução de complexidade, pois não há necessidade de diversos equipamentos para a realização dos experimentos.

Em geral, uma plataforma de testes de mobilidade envolve o uso de redes sem fio. No entanto, como o objetivo principal da plataforma é possibilitar o estudo de protocolos de camada 3, decidiu-se por construir um ambiente de teste através de máquinas virtuais *User Mode Linux* (UML) (DIKE et al., 2009) adaptando os *switches* virtuais para produzir a mobilidade. Desta forma, evita-se a complexidade adicional de instalação de hardware para a rede sem fio.

Os seguintes componentes podem ser identificados nesta plataforma:

- máquinas e *switches* virtuais UML para construção de domínios e dos nós móveis. Os *switches* virtuais foram modificados para a simulação de mobilidade;
- Interface gráfica para permitir a rápida construção de cenários e para possibilitar o comando de mobilidade;
- Ferramentas para estudo dos protocolos: gerador de tráfego, medidor de banda e de perda de pacotes;
- Protocolos de Mobilidade da Camada 3: instalação, nas UMLs, de uma versão do MIPv6, do gerador de mensagens de advertência de roteador do IPv6 *Router Advertisement Daemon* (RADVD) e de uma versão do HMIPv6. Como será visto adiante, não existe uma versão atualizada do protocolo HMIPv6. Parte do trabalho foi dedicada a uma extensão do MIPv6 para o HMIPv6;
- Outros protocolos e ferramentas de apoio: alguns protocolos e ferramentas adicionais foram instalados, tais como o protocolos RIP e o protocolo OSPF, através do pacote Zebra (ISHIGURO et al., 2009).

Na sequência, apresentamos alguns detalhes sobre os componentes da plataforma sendo que os aspectos relacionados aos protocolos de mobilidades serão discutidos separadamente nos próximos capítulos.

3.2 A UML como Bloco Básico da Plataforma de Testes

3.2.1 UML - *Linux* em Modo Usuário

A máquina virtual UML foi escolhida como base para a construção das redes virtuais na plataforma de testes. Ela oferece várias possibilidades de construção de redes, além de ser incorporada ao *kernel* do Linux e de apresentar um alto desempenho na execução de programas. Isto permite iniciar múltiplas instâncias UML em uma mesma máquina hospedeira e a construção de complexos cenários de rede.

O Linux executa a máquina UML como um processo comum no sistema. O processo é executado em um ambiente isolado protegendo a camada física da máquina hospedeira, o que não impõe restrições de uso para a máquina virtual, tornando um ambiente excelente para testes (LURUO; KHANVILKAR, 2005). A figura 3.1 mostra o funcionamento da UML em uma máquina hospedeira.

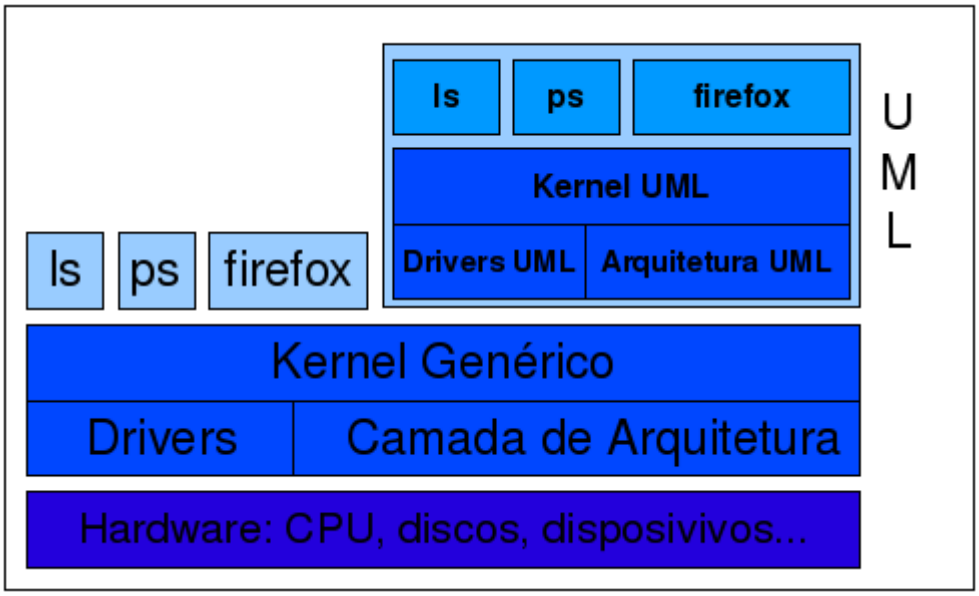

Figura 3.1: Funcionamento do Linux em Modo Usuário

É comum chamar a UML de máquina virtual, porém, a tecnologia de virtualização UML constitue-se, na verdade, em um sistema operacional virtual. O termo máquina virtual seria mais aplicável em tecnologias de virtualização tal como a *VMWare*, onde realmente é emulada uma plataforma física, processador e periféricos, sendo que o sistema operacional é executado nesta plataforma emulada.

A UML permite construir sistemas que em um ambiente real seriam muito difíceis de reproduzir, por exemplo, é possível criar uma máquina com *n* interfaces de rede e discos. Devido

a sua flexibilidade, a UML pode ser utilizada para inúmeras aplicações tais como:

- Consolidação de servidores de rede: é possível reproduzir todo sistema em UML e testá-lo antes de colocá-lo em produção;
- Em ambientes de ensino: com UML os alunos podem desenvolver as atividades ensinadas sem a preocupação de danificar o sistema;
- No desenvolvimento de aplicações em nível usuário, protocolos de rede, do *kernel linux*;

Para utilizar a UML é preciso um *kernel* Linux compilado para a arquitetura *um* e um sistema de arquivos, que nada mais é do que um simples arquivo contendo a imagem de um sistema, onde temos toda a estrutura de diretórios e os demais componentes para uma execução normal de um Linux. A UML realiza as operações de escrita e leitura de arquivos no sistema de arquivos, analogamente ao que sistema hospedeiro realiza no disco rígido.

Para esta plataforma de testes, o *kernel* linux-2.6.27 foi compilado com vários recursos de rede. Um sistema de arquivos foi construído a partir da distribuição *GNU/Linux Debian Lenny*. A idéia foi construir um pequeno sistema dedicado para laboratórios de redes. Ele inclui as implementações dos protocolos de rede e um conjunto de utilitários para solucionar problemas de rede.

Os principais componentes e recursos de rede disponíveis no *kernel* compilado e no sistema de arquivos são:

- Implementações do MIPv6 e HMIPv6;
- Implementações dos protocolos de roteamento RIPv2, RIPng, OSPFv2, OSPFv3 e BGP4;
- Utilitários de rede: *tcpdump*, *ping*, *ping6*, *traceroute*, *ip*(mostra e manipula rotas, dispositivos, políticas de roteamento e túneis), *iptables*, *tc*;
- Ferramentas IPSec;
- *Kernel* suporte para *Netfilter*;
- *Kernel* suporte para Qualidade de serviço (QoS);
- *Kernel* suporte para IP Seguro (IPSec) pra IPv4 e IPv6;
- *Kernel* suporte a mobilidade e otimização de roteamento em redes IPv6;

Outros componentes e recursos podem facilmente ser adicionados a plataforma de testes.

3.2.2 Redes Virtuais Móveis com a Máquina UML

UML oferece dois tipos de redes para as máquinas virtuais: uma que permite conectar a UML com o hospedeiro e uma outra denominada redes virtuais isoladas onde somente fazem parte instâncias UML. No primeiro tipo de rede as tecnologias de transporte utilizados pela UML são: *TUN/TAP*, *Ethertap*, *SLIP* e *Slirp*. No segundo tipo: *switch* virtual e *Multicast*.

Nos cenários de rede, para testes dos protocolos de rede, foram utilizadas as redes virtuais isoladas. O programa *uml_switch*, integrante do pacote UML, permite a interligação das máquinas virtuais. Ele é um utilitário que implementa um *switch Ethernet* ou um *hub* em *software*. As máquinas UML se conectam ao *switch* e se comunicam por meio de um arquivo de domínio *UNIX sockets* no hospedeiro.

Na figura 3.2 observamos as duas formas de redes virtuais UML. Note que a flexibilidade da máquina virtual permite à instância UML participar das duas redes.

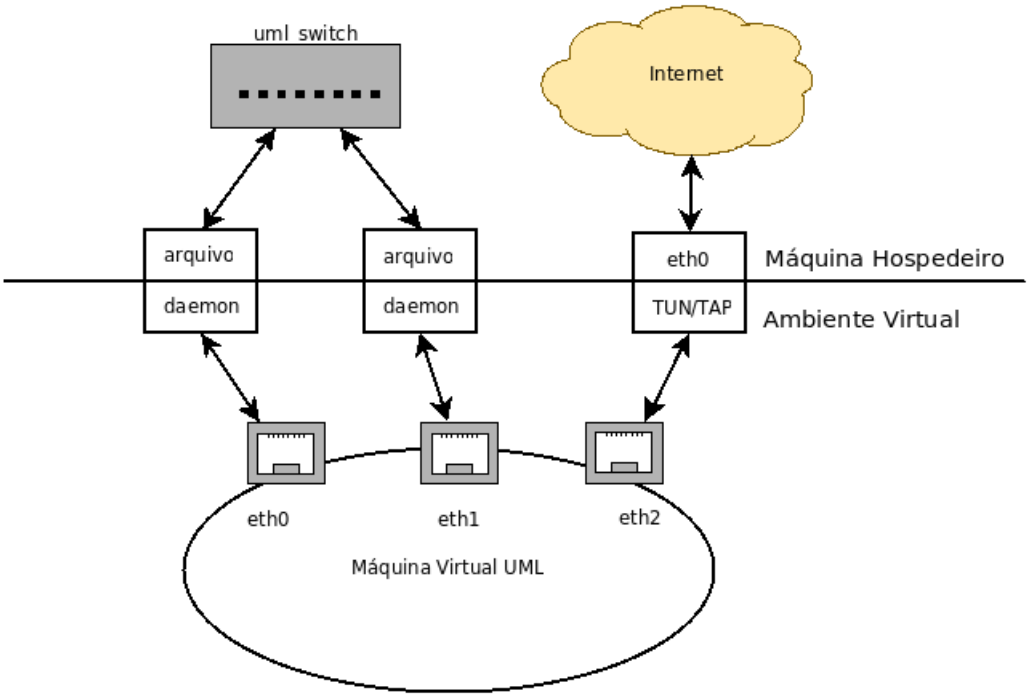

Figura 3.2: Redes Virtuais UML

Em geral, uma plataforma de testes de mobilidade envolve o uso de redes sem fio, contudo os protocolos estudados são em níveis de camada 3 sendo a mudança de portas em um *switch* suficiente para verificar boa parte do comportamento dos protocolos. Tornou-se necessário então fazer alterações no programa do *uml_switch*, que é um software livre sobre a *General Public License version 2* (GPLv2), escrito na linguagem de programação C.

O uml_switch para Mobilidade

A principal alteração no código fonte do *uml_switch* foi a inclusão da possibilidade de segmentação de redes, permitindo a criação de uma espécie de *Virtual Local Area Networks* (VLAN). Desta forma é possível realizar a mobilidade dos nós conectados ao *switch*. Após a mudança, cada nó conectado ao *switch* pertence a uma VLAN e é possível mover o nó para outra VLAN.

Outra implementação que foi realizada no *uml_switch* foi a adição de uma *thread* que responde um servidor *telnet*, que basicamente tem duas funções. A primeira é permitir o usuário efetuar a mobilidade dos nós conectados no *switch*. A outra função, é listar os nós conectados. Esse recurso torna fácil a implementação de uma interface gráfica do *switch*, pois um outro programa, em outra linguagem de programação, pode utilizar-se deste recurso para gerar a interface.

Os comandos que o servidor telnet do uml_switch pode aceitar são:

- *list*: Lista as portas do switch e as VLANs a elas associadas;
- *move* porta VLAN: move uma porta do *switch* para uma VLAN;
- *quit*: encerra a conexão.

3.3 GUML4MIP - Interface Gráfica de controle de terminais UML para o IP Móvel

3.3.1 A GUML4MIP do ponto de vista do usuário

A máquina virtual UML tem se mostrado uma ótima ferramenta para a plataforma de teste. Porém, a sua utilização na construção de redes virtuais não é muito prática, porque para cada cenário é necessário a criação de *scripts* para iniciar as máquinas e configurar as redes. Isto torna a construção de cenários demorada e não tão evidente, além do fato de que o usuário necessita de bons conhecimentos do sistema operacional *linux*, indo um pouco contra a idéia proposta pela plataforma.

Pelo motivo apontado teve-se a idéia de construir uma interface gráfica, amigável para o usuário, de forma a possibilitar: a construção rápida de cenários com redes, o controle da mobilidade dos nós e a centralização de todos os terminais das máquinas virtuais em uma única

janela. Esta última característica evitaria que, em cenários muito complexos, um grande número de janelas fossem abertas, diminuindo a produtividade durante sua utilização.

Após algumas pesquisas na *internet* foi encontrado o projeto de código aberto *GUI Management console for User Mode Linux* (GUML) (PALMER, 2009), implementada com a linguagem de programação *Python* e que já contempla muitos dos recursos pretendidos. Decidiu-se fazer alterações no código fonte deste projeto de forma a implementar os novos recursos almejados.

Como citado, apesar de possuir vários recursos, o GUML não fornece suporte para configuração de cenários de rede integrado na ferramenta. Não é seu intuito ser uma interface para construção de cenários de redes móveis e sim para controle de terminais UML. Por esse motivo optou-se fazer uma derivação do projeto e criar um outro um chamado *GUI Management console for User Mode Linux for Mobile IP* (GUML4MIP), baseado-se no seu código fonte.

Os principais recursos incorporados ao GUML4MIP, que possibilitam e melhoram o suporte as redes virtuais, são:

- Adição de parâmetros no arquivo de configuração, passando a ser possível especificar:
  - A função da máquina UML: um roteador ou um nó hospedeiro;
  - Endereço IP das interfaces das máquinas UML;
  - A VLAN que a interface da máquina UML será atrelada;
  - Serviços adicionais que poderão ser iniciados no *boot* da máquina UML. Por exemplo, o *daemon* do MIPL ou do RADVD;
- Geração automática de *scripts* de configuração de rede;
- Roteamento dinâmico utilizando o *Zebra*;
- Possibilidade de carregar novos cenários de rede em tempo de execução da interface;
- Interface gráfica do *uml_switch* (ver figura 3.4). Com este recurso o usuário pode visualizar o estado do *uml_switch* durante a execução do cenários;
- Controle da mobilidade das máquinas UML por meio da interface do *uml_switch*.

Foi também adicionado ao pacote do GUML4MIP um aplicativo, chamado *create_fs*, que constrói automaticamente o sistema de arquivos especificado neste capítulo para a plataforma de testes.

Para utilizar o GUML4MIP o usuário deve configurar um arquivo para cada máquina UML. No arquivo de configuração é possível declarar:

- O sistema de arquivo que a máquina virtual irá utilizar;
- Quantidade de memória para a máquina virtual;
- Interfaces de rede;
- Discos;

Um exemplo de arquivo de configuração do GUML4MIP pode ser observado no anexo A.

Na figura 3.3 podemos observar o visual da interface, rodando um cenário de rede, e os recursos disponíveis ao usuário. Abaixo uma explicação detalhada das funções que o usuário pode realizar na interface:

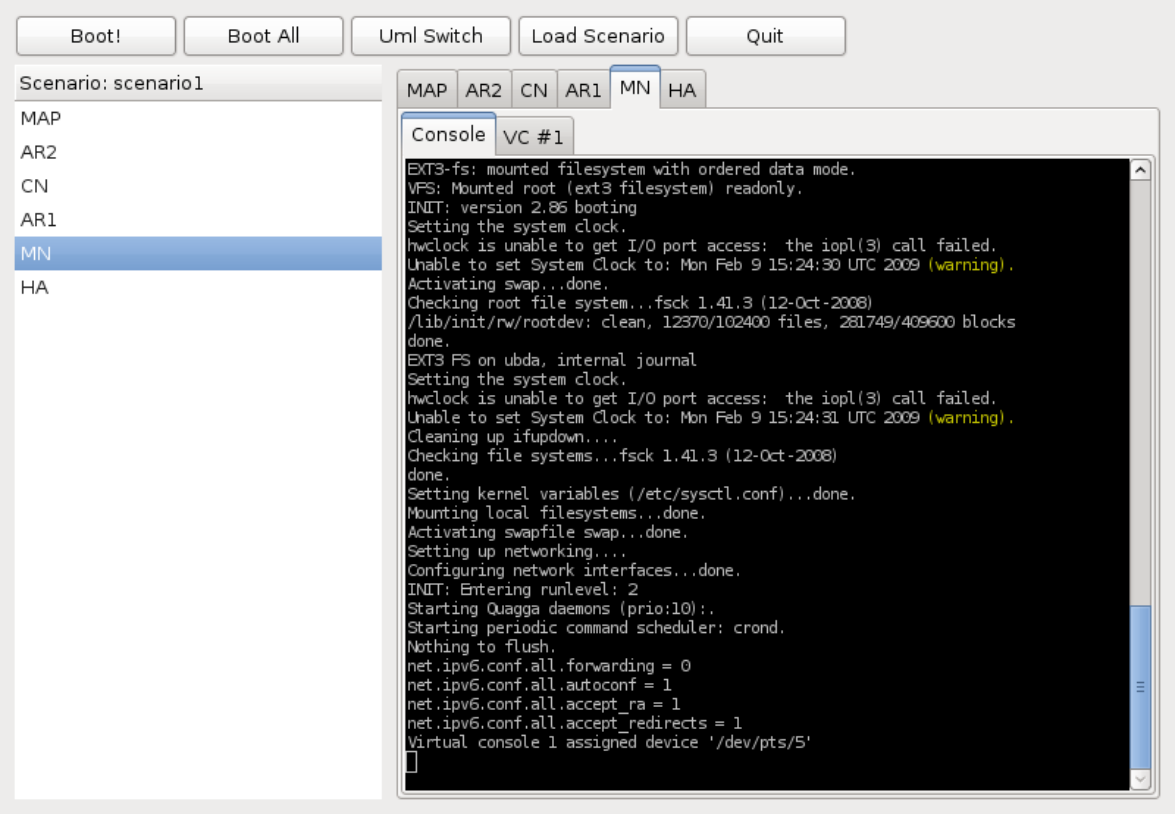

Figura 3.3: Interface Gráfica de controle de terminais UML para o IP Móvel

Boot ! Clicando neste botão, o usuário inicia a máquina UML selecionada no quadro que lista as máquinas do cenário;

Boot All Este botão permite que o usuário inicie todas as máquinas UML configuradas para o cenário, a direita por meio das abas, o usuário pode acessar o terminal das máquinas virtuais;

Uml Switch Clicando neste botão, uma nova janela será aberta. Na janela do *uml_switch*, figura 3.4, o usuário pode observar todas as máquinas virtuais conectados a ele, e a qual VLAN cada interface de rede esta associada. Caso o usuário queira efetuar alguma mobilidade basta clicar no número da VLAN a qual queira se associar;

Load Scenario Por meio deste botão, o usuário busca o diretório onde estão os arquivos de configuração de um novo cenário de rede. O ultimo cenário executado é salvo e na próxima execução do programa já é carregado automaticamente.

Quit Encerra o programa.

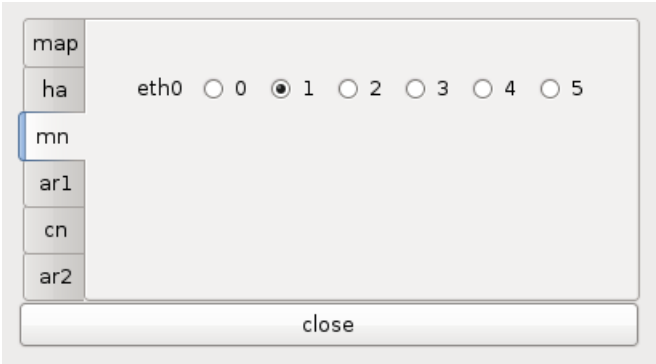

Figura 3.4: Interface Gráfica para o *uml switch*

3.3.2 Funcionamento do GUML4MIP

O programa GUML4MIP possui um funcionamento bem simples, por esse motivo se torna uma aplicação versátil e facilmente alterável. Na figura 3.5 é mostrado um diagrama dos principais blocos do GUML4MIP. A relação e o funcionamento dos blocos são descritos a seguir.:

setup.py: Programa que desempenha a função de instalador e desinstalador do GUML4MIP. Também é dele a função de criar o sistema de arquivos que será utilizado pelo *kernel* UML, instalar no hospedeiro o pacote do *uml_switch:* modificado com suporte a mobilidade e os pacotes dos protocolos de rede no sistema de arquivos.

guml4mip.py: Constitui-se no bloco principal de todo o diagrama. Ele comanda todas as funções do programa e também a interface com o usuário. Seu funcionamento é dividido nas etapas a seguir:

1. Ao iniciar, é realizada a leitura dos arquivos de configuração das máquinas UML no diretório do cenário a ser executado. Na primeira execução do programa, o

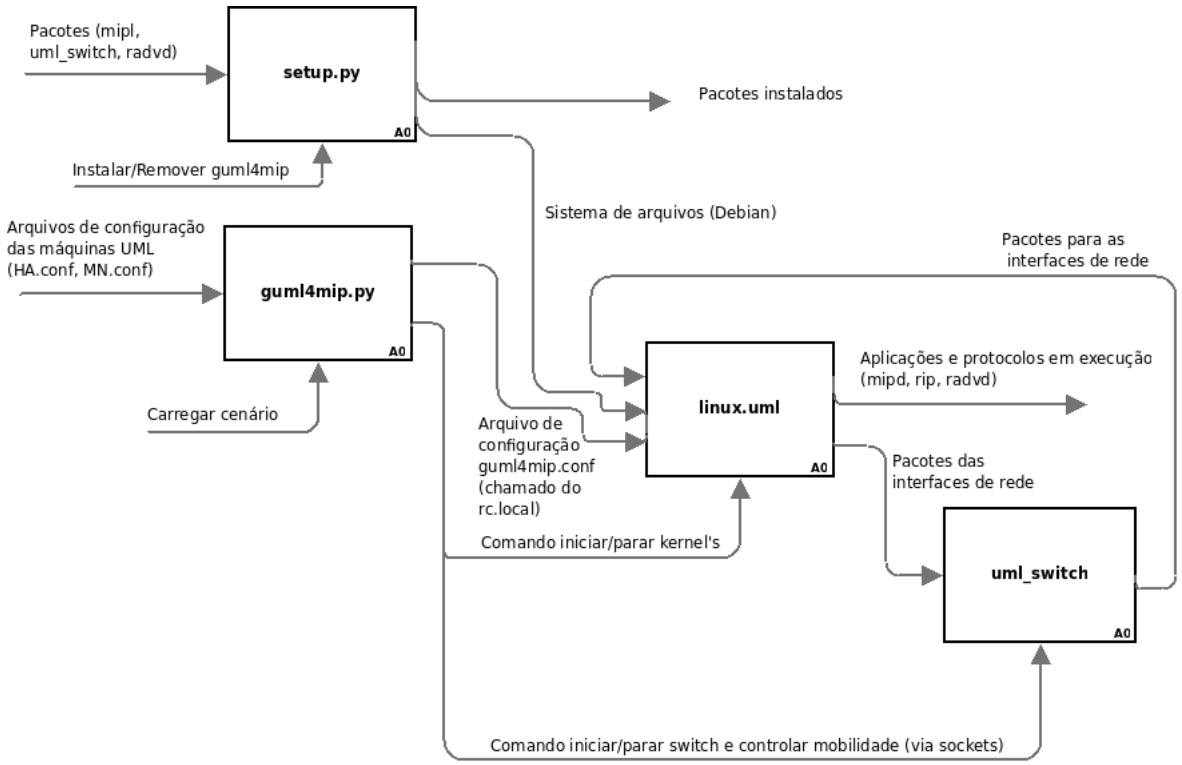

Figura 3.5: Funcionamento simplificado do GUML4MIP

cenário carregado é um exemplo que acompanha o pacote do GUML4MIP. Nas outras execuções, o último cenário executado é o padrão;

2. A partir dos arquivos de configuração são criadas as linhas de comando que executarão as máquinas UML e o arquivo **guml4mip.conf**. Este último é o *script* que configura as interfaces de rede das máquinas UML e que inicia os *daemons* especificados pelo usuário;

3. O processo do *uml_switch* é iniciado, processo este no qual as máquinas UML irão se conectar. Em paralelo uma *thread* é criada e se conecta ao servidor; *telnet* do *uml_switch*. Por meio desta conexão, o GUML4MIP monitora as conexões no *uml_switch*, apresentando em uma interface gráfica o estado do *switch*. A partir desta interface p ode-se provocar a mobilidade das máquinas UML;

4. Os recursos do programa são disponibilizadoas ao usuário em uma janela construída em *Python Gimp Tool Kit* (PyGTK);

5. Ao utilizar função que permite carregar um novo cenário, através do botão *Load Scenario*, o programa encerra os processos das máquinas do cenário anterior e finaliza a conexão com servidor telnet do *uml_switch*. O mesmo acontece quando é acionado o botão *Quit*, porém, neste caso reiniciam-se todas as etapas.

linux.uml: São os processos (*kernel*) das máquinas UML iniciados pelo programa, configuradas conforme seu arquivo de configuração e conectadas em sua VLAN. Seus terminais tornam-se disponíveis na interface;

uml_switch: Processo do *uml_switch* sempre iniciado no começo de um novo cenário de rede. Responsável por segmentar a rede em VLANs, entregar os pacotes as portas corretas e responder as requisições da sua interface gráfica, via *telnet*.

4 *Implementação do HMIPv6 baseando-se no projeto MIPL*

4.1 Implementações existentes para Linux dos protocolos de Mobilidade

O projeto *Mobile IPv6 for Linux* (MIPL) (NUORVALA; PETANDER; TUOMINEN, 2009) é uma implementação de suporte a mobilidade no IPv6 (RFC 3775), desenvolvida na Universidade Tecnológica de Helsinki . O código foi desenvolvido acompanhando os *drafts* da RFC e as versões do *kernel linux*. No início do desenvolvimento, a versão do kernel utilizada foi a 2.3.59 e o *draft* 8, porém, devido as substanciais mudanças na especificação do MIPv6, os gestores do projeto decidiram dividir o código fonte uma parte em espaço de kernel e outra em espaço de usuário. A última versão foi o MIPL 1.1 que trabalhava com a versão 2.4.26 do *kernel linux*.

Mais adiante, iniciou-se o desenvolvimento do MIPL2 pela Universidade de Helsink juntamente com o projeto USAGI (JUN, 2009). Nesta nova versão houveram mudanças significativas: o código foi praticamente todo reescrito e muitas funcionalidades foram adicionadas em espaço de usuário. Um *daemon* passou a controlar a sinalização e a detecção de movimento. A tarefa do *kernel* passou a ser apenas uma camada de suporte ao MIPv6.

O MIPL2 passou a suportar as últimas especificações do MIPv6 e passou a utilizar *framework* XFRM para o suporte a IPSec. É interessante observar também que, a partir das últimas versões do kernel do linux, o suporte ao MIPv6 já faz parte da linha principal de desenvolvimento, não sendo mais necessário aplicar *patchs*.

No que se refere ao HMIPv6, uma implementação foi proposta pela Universidade de Monash (MOORE,), baseando-se no MIPL 0.94 e na versão 2.4 do *kernel linux*. Também foram realizadas alterações no RADVD 0.7.2 para este divulgar mensagens de MAP. Neste momento, o projeto está parado e não atende mais as necessidades deste trabalho, pois, ocorreram mudanças significativas no desenvolvimento do MIPL. Atualmente a grande maioria dos sistemas já utilizam a série 2.6 do *kernel*, então pode-se concluir que não há nenhuma implementação aberta e

recente do HMIPv6.

O trabalho propõe-se, então desenvolver uma implementação experimental atualizada do HMIPv6, baseando-se na última versão estável do projeto MIPL e utilizando a versão 0.7.2 do RADVD alterado pelo projeto da Universidade de Monash.

4.2 Estudo do código do projeto MIPL

Com fins de gerar uma versão do HMIPv6 a partir do projeto MIP, iniciou-se um estudo aprofundado do código fonte do MIPL. O estudo do código do MIPL2 foi realizado na sua última versão estável, anunciada pelo projeto USAGI como umip-0.4. O MIPL é um projeto de código aberto sobre a licença GPL versão 2, escrito na linguagem de programação C.

O programa é executado na forma de um *daemon* chamado de *mip6d*, ou seja, um processo executado no segundo plano e que não está associado a um terminal controlador (STEVENS; FENNER; RUDOFF, 2005). Este mesmo *daemon* desempenha o papel de nó móvel, de agente domiciliar e de nó correspondente. A sua função é definida em um arquivo de configuração, sendo que todos os nós são em potencial um nó correspondente.

O foco principal do estudo foi na parte responsável pelo funcionamento do nó móvel, pois as principais mudanças do MIPv6 para o HMIPv6 é a funcionalidade deste nó. O agente MAP possui, basicamente, o mesmo comportamento de um agente domiciliar, a não ser pelo de fato de possuir uma rede com prefixo a partir do qual serão criados RCoAs. Além do que, todo o projeto é de alta complexidade, envolvendo o uso de várias bibliotecas, o que demandaria um tempo grande de estudo. É bom lembrar também que, com fins de delimitar o escopo do trabalho, foram desconsiderados todos os mecanismos de segurança do HMIPv6.

4.2.1 Interação entre o *kernel* e o MIPL

Geralmente tudo o que fizemos, enquanto trabalhamos em um sistema Linux, está sendo executado em em espaço de usuário, por exemplo: um editor de texto, um navegador, servidor X. Em contraste com isso, o *kernel* executa as suas tarefas em espaço de *kernel*, com acesso à instruções privilegiadas, total controle da memória e do sistema de interrupções. O principal motivo para esta separação entre dados dos processos do usuário e do *kernel*, é evitar que um perturbe o outro, o que resultaria numa diminuição do desempenho, problemas de segurança e instabilidades do sistema (ALTMEYER, 2009).

Os motivos levantados acima foram os que levaram os desenvolvedores a passarem a maio-

ria da implementação do protocolo MIPv6 para espaço de usuário, o que torna desenvolvimento mais fácil, seguro e independente de versão de *kernel*. Porém o MIPL precisa de uma interface para a troca de informações entre o *kernel* e espaço de usuário. A troca de informação do entre *kernel* e o espaço de usuário é realizada através do *netlink*. O *netlink* consiste em uma extensão da interface de soquetes padrão, que proporciona uma comunicação bidirecional entre *kernel* e o espaço de usuário. O soquete *netlink* usa endereços da família AF_NETLINK, comparado ao AF_INET usado pelos soquetes TCP/IP (DHANDAPANI; SUNDARESAN, 1999).

O soquete *netlink* é a forma utilizada pelo MIPL para:

- acessar as informações do MIPv6 no *kernel*;
- alterar endereços IP's;
- manipular tabelas de roteamento e regras das políticas de uso destas tabelas;
- criar túneis.

Os soquetes brutos também são utilizados pelo MIPL para permitir a leitura e envio de pacotes puramente IP sem a interferência do *kernel* (STEVENS et al., 2003). Em condições normais quando o kernel não entende o campo de protocolo do datagrama IP, como por exemplo, os que possuem cabeçalho de mobilidade, eles são passados diretamente para um soquete bruto. O único processamento feito pelo *kernel* é a verificação mínima de alguns campos de cabeçalho IP.

Esta capacidade permite que tais pacotes possam ser tratados por aplicações independentes de inserção de códigos especiais em *kernel*, neste caso o *daemon* MIPL.

4.2.2 *Threads* importantes do MIPL

No seu funcionamento, o *daemon* do MIPL necessita realizar várias tarefas, por exemplo, interagir com o *kernel*. Para realizar as várias funções que permitem o funcionamento do protocolo MIPv6, o programa utiliza a técnica de *Multithreading*. A seguir são listadas *threads* que executam no *daemon* e a figura 4.2 (YOSHIFUJI, 2005) mostra a forma que algumas delas se relacionam com o *kernel*.

runner

Durante a sua execução, o *mip6d* necessita agendar tarefas para serem executadas. Por exemplo, caso o tempo de vida de um endereço expirar ele precisa ser removido, para isso, o

daemon cria uma tarefa que remove o endereço da interface, e a insere em uma fila global de tarefas.

Para inserir tarefas na fila o *daemon* utiliza a função *add_task_abs*, passando como parâmetro:

- o tempo em milisegundos para a execução da tarefa;
- ponteiro para função que deve ser chamada quando o tempo da tarefa expirar.

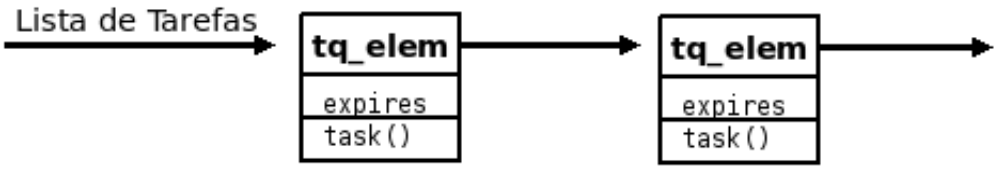

Figura 4.1: Fila de tarefas agendadas do MIPL

A *thread runner*, fica percorrendo a fila de tarefas(ver figura 4.1) e quando o tempo da tarefa expira, executa uma função associada, previamente registrada.

icmp6_listen

O MIPL cria uma *thread* para escutar mensagens ICMPv6 e para isso cria um soquete bruto. Porém, há muitas mensagens ICMPv6, por isso, para reduzir o número de pacotes passados do *kernel* para a aplicação, é fornecido um filtro especifico pela aplicação. Na tabela 4.1 é possível observar as mensagens ICMPv6 que serão tratadas pelo MIPL.

Função mip6d	Mensagem Filtrada	Função
Nó Móvel	ND_ROUTER_ADVERT	Anúncio de roteadores
	ND_NEIGHBOR_ADVERT	Anúncio de vizinhança
	MIP_HA_DISCOVERY_REPLY	Resposta de descoberta de agente domiciliar
	ICMP6_PARAM_PROB	Problema de Parâmetro
	ICMP6_DST_UNREACH	Destino inalcançável
Agente Domiciliar	MIP_PREFIX_SOLICIT	Solicitação de prefixo
	MIP_HA_DISCOVERY_REQUEST	Descoberta de agente domiciliar
	ND_ROUTER_ADVERT	Anúncio de roteadores
	ICMP6_DST_UNREACH	Destino inalcançável;

Tabela 4.1: Filtros para os pacotes ICMPv6 feitos pelo MIPL

Durante a execução do *daemon* é feita a instalação de manipuladores, que tratam as mensagens, em uma lista global. Ao receber uma das mensagens ICMPv6 filtradas, a *thread* percorre a lista de manipuladores e chama a função para tratar a mensagem.

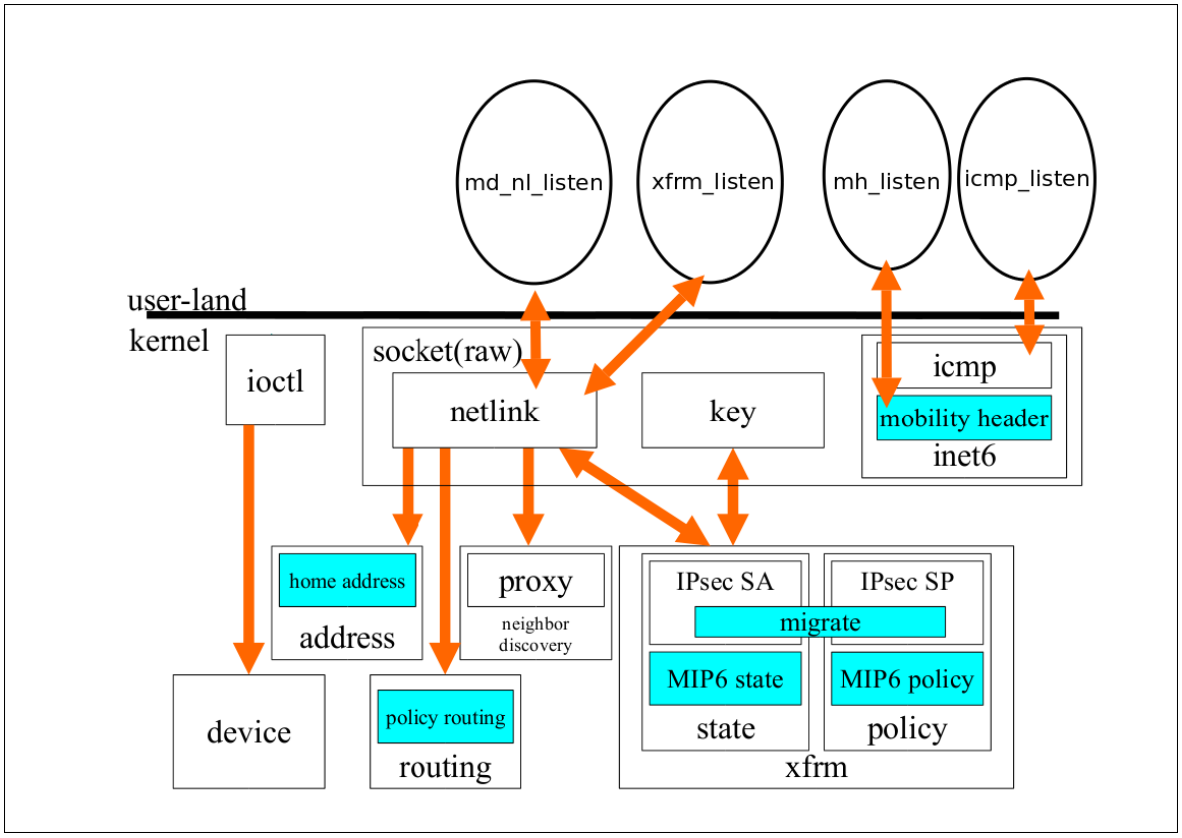

Figura 4.2: Interfaces utilizadas pelo MIPL para fazer a comunicação com o kernel

mh_listen

Thread criada para escutar a chegada de pacotes IPv6 com cabeçalho de mobilidade. O seu funcionamento é semelhante a *thread icmp6_listen*: ao receber uma mensagem com o cabeçalho de mobilidade chama um manipulador, previamente instalado em uma lista global, para tratar a mensagem.

xfrm_listen

O *daemon* cria a *thread xfrm_listen* que escuta mensagens *netlink* do *framework* XFRM. Um manipulador é instalado para tratar essas mensagens. As mensagens tratadas são: XFRM_MSG_ACQUIRE e XFRM_MSG_REPORT.

md_nl_listen

O *mip6d* cria uma *thread* para tratar mensagens *netlink* do tipo NETLINK_ROUTE enviadas pelo *kernel*. Esta *thread* é executada somente no nó móvel.

A *thread* ao receber uma mensagem *netlink*, chama o manipulador instalado para tratá-la. As mensagens tratadas pelo manipulador são as que informam:

- a interface ou ligação: ligado ou desligado;
- as mudanças no estado da alcançabilidade do roteador padrão;
- os *Care-of-address* novos ou removidos.

vt_server_recv

É possível habilitar em tempo de compilação um terminal virtual. O terminal virtual permite que os usuários se conectem ao *daemon*, via protocolo *telnet*, para obter informações sobre o MIPv6, por exemplo:

- O tempo de vida do prefixo do agente domiciliar;
- O estado do *Binding cache*;
- A Lista de *Bindig Updates*.

A *thread vt_server_recv* é criada para responder as conexões para o terminal virtual, que é executado na porta 7777.

sigh

O *mip6d* implementa uma *thread* que é responsável por fazer o tratamento de alguns sinais do sistema. Os sinais que passam a ser tratados pela aplicação são:

SIGHUP: Significa o reinicio do programa, o *daemon* ao receber esse sinal executa a função *reinit*;

SIGINT: Causa a interrupção do programa, em termos práticos é o sinal recebido quando se pressiona as teclas *Control+C*. Neste caso ao receber este sinal o *daemon* chama a função *terminate*;

SIGTERM: Solicitação de término do programa, neste caso o *daemon* também chama a função *terminate*.

É de extrema importância o tratamento destes sinais pelo *daemon*, pois ao finalizar o programa é preciso deletar eventuais endereços IP's, túneis, tabelas de roteamento.

4.2.3 Detecção de Movimento

O algoritmo de detecção de movimento do MIPL é inteiramente baseado na escuta das mensagens de anúncio de roteadores. O nó móvel está constantemente escutando as mensagens de RA, sendo que o nó percebe o movimento quando:

- o seu roteador padrão torne-se inalcançável; para tanto, o nó móvel de tempos em tempos, verifica o roteador com um *Neighbor Solicitation*;
- passa a receber mensagens de outro roteador e para de escutar mensagens do anterior.

Embora o MIPL seja uma extensão da pilha IPv6 padrão do *Linux*, apresenta algumas características sobre administração de roteadores:

1. O MIPL possui a sua própria lista de roteadores. Esta lista contém o atual roteador de acesso, bem como roteadores que não são atualmente utilizados, mas com tempo de vida não expirados;
2. o MIPL força atualizações na tabela de roteamento após receber um anúncio de novo roteador, e apaga todas as informações de roteamento que não são do prefixo atual.

Devido a estas duas características, o MIPL sempre escolhe um novo roteador. Este método de seleção agressiva de roteadores traz uma melhora na detecção de movimento, ou seja, no tempo de *handover*.

4.3 Alterações no MIPL para obter o HMIPv6

A primeira implementação necessária foi dar suporte ao MIPL a leitura das mensagens RA divulgadas pelos MAP's, pois são elas que permitem a criação do endereço RCoA. No manipulador de mensagens RA foi adicionado o suporte para a leitura da opção de MAP na mensagem RA. Foi criada uma estrutura de dados para o MAP e funções para manipulação desta estrutura. Na estrutura de dados do roteador foi adicionado uma lista de MAP's. A estrutura é mostrada a seguir.

```
1 struct map_list_entry
2 {
3     struct list_head list;
4     struct timespec timestamp;
```

```
5     struct tq_elem tqe;
6     struct nd_opt_map map;
7     struct in6_addr rcoa;
8     int used;
9 };
```

A estrutura contêm:

- os campos da opção de MAP da mensagem RA;
- tempo de vida do roteador que entregou a mensagem;
- endereço RCoA criado a partir do endereço do MAP;
- estrutura de dados que permite criar tarefas para serem agendadas;

No processamento da mensagem RA foram adicionados novos eventos de movimento: **ME_MAP_NEW**, **ME_MAP_UPDATE** e **ME_MAP_EXPIRED** Estes eventos tem a função de colher os dados que serão utilizados no momento do registro e manter os dados para o funcionamento do protocolo.

Foi criada também uma estrutura de dados para o RCoA tendo sido adicionada na estrutura do agente domiciliar. As informações contidas na estrutura serão utilizadas no momento da formação das mensagens de BU e no controle do registro.

```
1  struct rcoa_addr_info {
2      struct mn_addr rcoa; /* Home address for MAP */
3      uint8_t mlen;
4      uint8_t map_reg_status;
5      struct in6_addr map_prefix;
6      struct in6_addr map_addr;
7      int pend_ba;
8      int site;
9      int if_tunnel;
10 };
```

Outas implementações foram realizadas no nó móvel, quais sejam:

- foi acrescentado ao mecanismo de registro, um suporte ao envio do BU para o MAP;
- alteração da mensagem de BU para o agente domiciliar;
- criação do túnel, com o MAP;

- e modificação do túnel com o agente domiciliar.

Algo interessante de salientar é que toda a parte de tomada de decisão sobre o movimento se manteve e as alterações atuam somente na forma do registro com os agentes.

4.3.1 Detalhes do funcionamento

A seguir, uma descrição das implementações efetuadas no MIPL em um movimento para um novo domínio:

1. Ao receber uma mensagem RA de um novo roteador de acesso:
   - as opções de MAP são lidas;
   - uma lista de MAP's é criada na estrutura do roteador.
2. Um novo MAP dispara um evento de movimento do tipo **ME_MAP_NEW** que:
   - faz escolha de um MAP na lista de roteadores, baseado na distância em saltos do MAP;
   - cria um túnel para o MAP;
   - adiciona o RCoA na interface do túnel;
   - define o RCoA na estrutura do agente domiciliar, utilizado no envio do BU.
3. o movimento é detectado, usando o código normal do MIPL. O registro deve ser, então, realizado com os agentes:
   - enviado BU para o MAP com os campos:
     - Origem = RCoA
     - Destino = endereço MAP
     - *Care-of-Adress* = LCoA
   - enviado BU para agente domiciliar com os campos:
     - Origem = endereço domiciliar
     - Destino = endereço agente domiciliar
     - *Care-of-Adress* = RCoA
4. Após o envio dos BU's, os túneis devem ser modificados:
   - Túnel entre nó móvel e MAP: o ponto final é a interface de rede.

- Túnel entre nó móvel e agente domiciliar: o ponto final é o túnel entre nó móvel e MAP.

5. Periodicamente, com a chegada dos RA's, o evento de movimento **ME_MAP_UPDATE** é disparado e o tempo de vida do RCoA é atualizado.

4.3.2 Problemas conhecidos

Considerando o caráter experimental do trabalho, a disponibilidade de tempo para a sua execução e a alta complexidade do código do MIPL, pode-se dizer que muito trabalho deve ainda ser realizado para a obtenção de uma versão completa do HMIPv6. No que se refere a implementação efetuada, alguns problemas podem, em particular, ser apontados:

- ausência de suporte a otimização de rotas;
- ausência de suporte a segurança (IPSec);
- ausência de suporte ao HMIPv6 em arquivo de configuração, somente em tempo de compilação;
- sem tarefas para deletar MAP's expirados;
- problema ao receber BA provindo do MAP, o que ocasiona problemas em mobilidade em um mesmo domínio;
- problemas no retorno para rede domiciliar.

5 *Cenários de Testes*

5.1 Cenários Estudados

Após os estudos bibliográficos sobre o protocolo MIPv6 e a preparação da Plataforma de Testes para Mobilidade, iniciou-se a etapa de definição e implementação de cenários de testes. Esta etapa teve por objetivo analisar algumas funcionalidades dos protocolos instalados, demonstrando desta forma as potencialidades do ambiente desenvolvido.

Para analisar alguns mecanismos dos protocolos e testar os seus modos de operação, dois cenários base foram definidos: cenário 1, com o MIPv6, e o cenário 2, com o HMIPv6. Para o primeiro cenário, duas variações foram realizadas. Na primeira, descrita na figura 5.1, é testado o modo de operação por tunelamento bidirecional do MIPv6, sem a otimização de rota. A segunda variação testa o modo de operação por otimização de roteamento. O cenário 2, descrito na figura 5.4, tem como objetivo verificar minimamente o funcionamento do HMIPv6 em comunicações onde os nós correspondentes não executam otimização de roteamento.

Os arquivos para a configuração dos cenários usados no GUML4MIP estão disponíveis no diretório da documentação do projeto (SILVA; ALMEIDA, 2009).

5.1.1 Fundamentos para a Análise dos Cenários

Na análise dos cenários estudados foram focados os seguintes aspectos:

- a sequência e a natureza das mensagens trocadas entre os vários nós. Neste ponto, foram utilizadas as saídas geradas pelo programa *tcpdump*;
- as tabelas de roteamento e as regras de roteamento analisadas em situações chave, por exemplo, antes e depois da mobilidade;
- os aspectos de desempenho e perdas de pacotes, tendo ciência das limitações desta análise em um ambiente virtual. Os dados obtidos foram também confrontados com os resultados

esperados através de uma abordagem analítica;

- a criação e a manipulação dos túneis entre o agente domiciliar e o nó móvel, antes e depois do movimento.

5.1.2 Considerações sobre o Tempo de latência do *Handover*

O processo de *handover* acontece quando o nó móvel muda seu ponto de conexão de uma sub-rede para outra. O tempo de latência envolvido neste processo (COSTA et al., 2002) pode ser dividido em quatro fases:

1. **Detecção de Movimento (*TD*)**: Em um cenário real, representa o tempo do *handover* na camada de enlace até o primeiro RA. Na plataforma de testes construída não foi simulada a camada enlace e, portanto, não se pode precisar com a exatidão a latência envolvida no processo. Porém, para fins de estudo, considerar-se-á o tempo entre a entrega do último pacote, de alguma conexão, recebido na rede domiciliar ($t0$) e o primeiro RA na rede visitada ($t1$).

$$TD = t1 - t0 \tag{5.1}$$

2. **Configuração do CoA (*TA*)**: Tempo entre o primeiro RA e o envio do BU ($t2$).

$$TA = t2 - t1 \tag{5.2}$$

3. **Registro com agente domiciliar (*TR*)**: Intervalo de tempo entre o envio do BU ao agente domiciliar e o recebimento do BA($t3$).

$$TR = t3 - t2 \tag{5.3}$$

4. **Otimização de Roteamento (*TO*)**: Intervalo de tempo entre o envio das mensagens do RRP ($t4$) e o recebimento do BA do nó correspondente.

$$TO = t4 - t3 \tag{5.4}$$

Obviamente, pode-se estimar o tempo de latência do *handover* com a seguinte fórmula:

$$TH = TD + TA + TR + TO \tag{5.5}$$

5.1.3 Metodologia para Realização dos Cenários

Nos cenários propostos, o nó móvel inicialmente estará em sua rede domiciliar e depois irá movimentar-se e visitar outras redes. Durante o procedimento de movimento, testes serão realizados para a coleta de dados que irão ajudar na análise do protocolo.

Em um teste, o nó correspondente começa a enviar mensagens ICMPv6, por meio do *ping6*, para o nó móvel:

```
cn# ping6 mn
```

O nó móvel executa o *tcpdump* em modo *verbose* (-vvv) para realizar o monitoramento da troca de mensagens do funcionamento do MIPv6 durante o processo de *handover*:

```
mn# tcpdump -vvv
```

Outro procedimento de teste, que é eventualmente realizado durante a execução dos cenários, é uma conexão utilizando o protocolo da camada transporte TCP. Tal teste visa comprovar a transparência do processo de mobilidade para as camadas superiores à de rede, além de permitir gerar um fluxo de dados maior do que com as mensagens ICMPv6 e de possibilitar uma melhor observação do impacto do protocolo MIPv6 sobre as conexões.

Para realização do teste é iniciado, no nó móvel, um servidor TCP. Este servidor responde a requisição GET do protocolo *HyperText Transfer Protocol* (HTTP), além de ser capaz de gerar tráfego. Para realizar esta tarefa é utilizado a ferramenta *netcat6*, da forma:

```
mn# echo "200 OK" > header; cat header /dev/zero | nc6 -l -p 80
```

Esta linha significa:

echo "200 OK" > header: cria um cabeçalho HTTP com a mensagem "200 OK", para responder as requisições ao servidor;

nc6 -l -p 80: inicia um soquete TCP que (-l) escuta a (-p 80) porta 80;

cat header `/dev/zero`**:** concatena para a entrada padrão do servidor o arquivo contendo a mensagem HTTP e o dispositivo caracter `/dev/zero` que é responsável por gerar o tráfego.

O nó correspondente irá se conectar ao servidor utilizando da ferramenta *wget*. Também no nó correspondente executamos a ferramenta *speedometer* que é capaz de medir a taxa média de dados de uma interface de rede. A saída do *speedometer* permite criar um gráfico para visualizar o efeito do MIPv6 em uma conexão.

```
cn# wget -6 --limit-rate=1M http://mn -o /dev/null -O /dev/null &
cn# speedometer -p -i 0.1 -rx eth0
```

Sobre os comandos executados no nó correspondente as seguintes considerações:

- O gerador produz uma taxa muito elevada de dados o que ocasiona o travamento das máquinas virtuais. Para resolver este problema a taxa é limitada (–limit-rate=1M) para 1MBps;
- o *speedtometer* vai monitorar a recepção da interface eth0 em intervalos de 0.1 segundos.

5.1.4 Subsídios para interpretação dos pacotes IPv6 no *tcpdump*

Nas listagens do *tcpdump*, utilizadas na seções que se seguem, os pacotes serão mostrados da seguinte forma:

<estampa_tempo> IP6 (.. next header <tipo_next_header> ..) <endereço_fonte> <endereço_destino>

No caso de túneis, onde existe encapsulamento de IP sobre IP, o formato é repetido e o campo *next header* do primeiro nível indica que o próximo cabeçalho é um pacote IPv6. No segundo nível, o *next header* é a carga original transportada, por exemplo, um pacote ICMP6:

<estampa_tempo> IP6 (.. next header IPv6 ..) <endereço_fonte> <endereço_destino> IP6 (.. next header ICMP6 ..) <endereço_fonte> <endereço_destino>

É interessante lembrar que os RA são mensagens ICMP6, assim como os *echo request* e *echo reply* produzidos pelo *ping6*.

O *tcpdump* apresenta os cabeçalhos de mobilidade (BU e BA) com *next-header unknown* no caso do BU e de *source routing* no caso do BA. Entretanto, o *next-header* é detalhado na sequência mostrando o tipo e eventualmente o DSTOPT (campo *destination option*).

5.2 Cenário 1 - Uso do Protocolo MIPv6

5.2.1 Descrição da Topologia

O primeiro cenário de teste é formado por três redes interligadas entre si e cinco nós. A sua topologia pode ser observada na figura 5.1, onde o número nos balões cinza representam a VLAN no *uml_switch*.

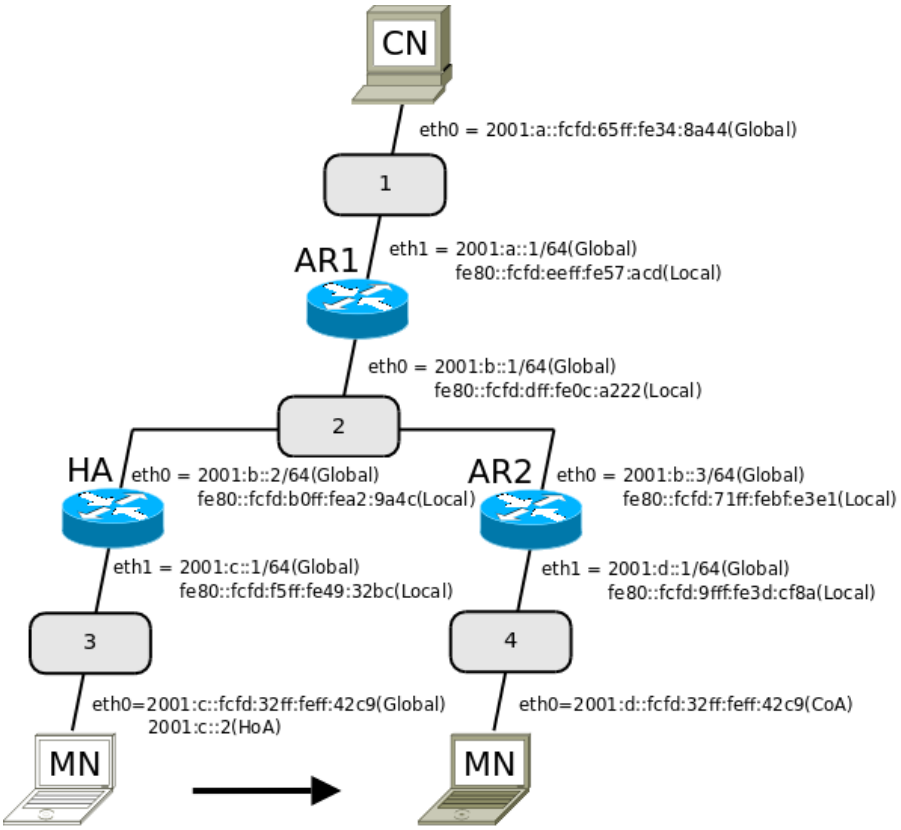

Figura 5.1: Topologia do Cenário 1

Na rede com prefixo 2000:a::/64 está o nó correspondente, com quem o nó móvel está se comunicando. Na rede com o prefixo 2000:c::/64 está presente um roteador (HA) que oferece o serviço de agente domiciliar ao nó móvel. A rede com prefixo 2000:d::/64 é a rede que o nó móvel deve visitar.

5.2.2 Variação 1: com Tunelamento Bidirecional e Sem Otimização de Rota

Análise da Troca de Mensagens

Os dados obtidos na saída do comando *tcpdump* podem ser observados na figura 5.2. Analisando os dados recolhidos consegue-se observar:

```
15:04:15.648283 IP6 (hlim 255, next-header ICMPv6 (58) payload length: 56)
        fe80::fcfd:f5ff:fe49:32bc > ff02::1: ICMP6, router advertisement,
        length 56, Flags [home agent], pref medium, router lifetime 9s
15:04:15.947080 IP6 (hlim 62, next-header ICMPv6 (58) payload length: 64)
        2001:a::fcfd:65ff:fe34:8a44 > 2001:c::2: ICMP6, echo request,
        length 64, seq 3
15:04:15.947174 IP6 (hlim 64, next-header ICMPv6 (58) payload length: 64)
        2001:c::2 > 2001:a::fcfd:65ff:fe34:8a44: ICMP6, echo reply,
        length 64, seq 3
15:04:16.468360 IP6 (hlim 255, next-header ICMPv6 (58) payload length: 56)
    (1) fe80::fcfd:71ff:febf:e3e1 > ff02::1: ICMP6, router advertisement,
        length 56, Flags [none], pref medium, router lifetime 9s
15:04:16.518273 IP6 (hlim 1, next-header Options (0) payload length: 56)
        :: > ff02::16: HBH (rtalert: 0x0000) (padn)ICMP6,
        multicast listener report v2, length 48, [|icmp6]
15:04:16.926324 IP6 (hlim 255, next-header ICMPv6 (58) payload length: 24)
    (2) :: > ff02::1:ffff:42c9: [icmp6 sum ok] ICMP6, neighbor solicitation,
        length 24, who has 2001:d::fcfd:32ff:feff:42c9
15:04:17.290338 IP6 (hlim 255, next-header ICMPv6 (58) payload length: 24)
        :: > ff02::1:ffff:42c9: [icmp6 sum ok] ICMP6, neighbor solicitation,
        length 24, who has fe80::fcfd:32ff:feff:42c9
15:04:18.297878 IP6 (hlim 64, next-header unknown (60) payload length: 56)
    (3) 2001:d::fcfd:32ff:feff:42c9 > 2001:c::1: DSTOPT (homeaddr: 2001:c::2)
        mobility: BU seq#=12103 AH lifetime=262140(alt-CoA: 2001:0:3100::)
15:04:19.326743 IP6 (hlim 255, next-header ICMPv6 (58) payload length: 32)
        fe80::fcfd:71ff:febf:e3e1 > ff02::1:ffff:42c9: [icmp6 sum ok] ICMP6,
        neighbor solicitation, length 32, who has 2001:d::fcfd:32ff:feff:42c9
15:04:19.326875 IP6 (hlim 255, next-header ICMPv6 (58) payload length: 32)
        2001:d::fcfd:32ff:feff:42c9 > fe80::fcfd:71ff:febf:e3e1: ICMP6,
        neighbor advertisement, length 32, tgt is 2001:d::fcfd:32ff:feff:42c9,
        destination link-address option (2), length 8 (1): fe:fd:32:ff:42:c9
15:04:19.327784 IP6 (hlim 63, next-header Routing (43) payload length: 40)
    (4) 2001:c::1 > 2001:d::fcfd:32ff:feff:42c9: srcrt (type=2, [0]2001:c::2)
        mobility: BA status=0 seq#=12103 lifetime=262140(padn)
15:04:19.340546 IP6 (hlim 255, next-header ICMPv6 (58) payload length: 56)
        fe80::fcfd:71ff:febf:e3e1 > ff02::1: ICMP6, router advertisement,
        hop limit 64, Flags [none], pref medium, router lifetime 9s
15:04:19.979737 IP6 (hlim 63, next-header IPv6 (41) payload length: 104)
    (5) 2001:c::1 > 2001:d::fcfd:32ff:feff:42c9:
        IP6 (hlim 62, next-header ICMPv6 (58) payload length: 64)
        2001:a::fcfd:65ff:fe34:8a44 > 2001:c::2: [|icmp6]
15:04:19.979921 IP6 (hlim 64, next-header IPv6 (41) payload length: 104)
        2001:d::fcfd:32ff:feff:42c9 > 2001:c::1:
        IP6 (hlim 64, next-header ICMPv6 (58) payload length: 64)
        2001:c::2 > 2001:a::fcfd:65ff:fe34:8a44: [|icmp6]
```

Figura 5.2: Cenário 1 : Captura de mensagens com Tunelamento Bidirecional

1. O nó móvel recebe uma mensagem de RA do roteador de acesso da rede visitada (src

= fe80::fcfd:71ff:febf:e3e1), o que indica que ocorreu mobilidade. Este movimento foi produzido pela comutação da VLAN 3 para a 4. Antes de realizar o movimento o nó móvel recebia *echo request* do nó correspondente (src = 2001:a::fcfd:65ff:fe34:8a44) e RA do Agente domiciliar (src = fe80::fcfd:f5ff:fe49:32bc);

2. O nó móvel realiza o teste de duplicação de endereço (DAD) do CoA gerado com o prefixo da rede visitada (2001:d::fcfd:32ff:feff:42c9);

3. O nó móvel envia a mensagem de BU para o agente Domiciliar informando o endereço CoA e o seu endereço domiciliar HoA. Note que o CoA é o endereço fonte do pacote e o HoA é informado na opção DSTOPT;

4. O agente domiciliar envia um BA, confirmando o registro;

5. O nó móvel volta a receber *echo request* do nó correspondente, só que agora através do túnel. Note os endereços IP fonte e destinos deste pacote. O primeiro nível IP do agente domiciliar(2001:c::1) para o CoA do nó móvel (2001:d::fcfd:32ff:feff:42c9). No segundo nível, do nó correspondente (2001:a::fcfd:65ff:fe34:8a44) para o HoA (2001:c::2).

Análise das Tabelas de Roteamento e de Regras de Roteamento

Após o nó móvel deixar a rede domiciliar, podemos constatar mudanças nas tabelas de roteamento do nó móvel e do agente domiciliar, que são mostradas nas respectivas tabelas 5.1 e 5.2.

Destino	Via	Próximo salto	Considerações
2001:a::/64	eth0	fe80::fcfd:dff:fe0c:a222	
2001:b::/64	eth0	::	
2001:c::2	**eth1**	**2001:c::2**	Antes da mobilidade do nó móvel
2001:c::/64	eth1	::	
2001:d::/64	eth0	fe80::fcfd:9fff:fe3d:cf8a	
2001:a::/64	eth0	fe80::fcfd:dff:fe0c:a222	
2001:b::/64	eth0	::	
2001:c::2	**ip6tnl1**	**2001:c::2**	Após a mobilidade do nó móvel
2001:c::/64	eth1	::	
2001:d::/64	eth0	fe80::fcfd:9fff:fe3d:cf8a	

Tabela 5.1: Tabela principal de roteamento do agente domiciliar

Após o agente domiciliar receber o BU, enviado pelo nó móvel para efetuar o registro, é adicionada na sua tabela de roteamento uma rota para que todos os pacotes com destino ao endereço domiciliar sejam encaminhados para o túnel. Após o nó móvel receber o RA na rede

Destino	Via	Próximo salto	Considerações
2001:c::2	ip6tnl1	::	Na rede domiciliar
2001:c::/64	eth0	::	
default	eth0	fe80::fcfd:f5ff:fe49:32bc	
2001:c::2	ip6tnl1	::	Na rede visitada
2001:d::/64	eth0	::	
default	eth0	fe80::fcfd:71ff:febf:e3e1	

Tabela 5.2: Tabela principal de roteamento do nó móvel

visitada, o roteador padrão é alterado. é deletada a rota para rede 2001:c::/64 e adicionada uma rota para a rede 2001:d::/64, onde a interface eth0 está atrelada.

Desde as versões do *kernel linux* 2.x pode-se ter múltiplas tabelas de roteamento no kernel. Estas tabelas são identificadas por um número variando de 1 a 255 ou por um nome de acordo com o arquivo `/etc/iproute2/rt_tables`. Com novas tabelas pode-se criar uma estrutura flexível para implementar uma Politica de Roteamento (LYRA; TORRES,).

Existem três tabelas que são criadas pelo *kernel*. A *local*(255), é uma tabela de roteamento especial mantida pelo kernel, onde usuários não podem adicionar novas entradas nessa tabela. A *main*(254), é a tabela de roteamento principal, onde ficam as rotas quando adicionamos com o comando `route` ou `ip route`. A *default*(253) é outra tabela especial e a *unspec*(0), qualquer operações sobre esta tabela refletem em todas as tabelas.

O nó móvel adiciona regras na tabela 252, mostradas na tabela 5.3, para que os pacotes com destino ao nó correspondente e ao agente domiciliar sejam encaminhados via túnel.

Destino	Via	Próximo salto	Considerações
*	*	*	Na rede domiciliar
2001:a::fcfd:65ff:fe34:8a44	ip6tnl1	2001:a::fcfd:65ff:fe34:8a44	Na rede visitada
2001:c::1	ip6tnl1	2001:c::1	
default	ip6tnl1	::	

Tabela 5.3: Tabela de Roteamento 252 do nó móvel

No agente domiciliar é adicionada uma rota na tabela 252, para que os pacotes com destino ao nó móvel(2001:c::2) e origem no agente domiciliar(2001:c::1) sejam encaminhados via eth1 para a rede visitada.

Destino	Origem	Via	Considerações
2001:c::2	2001:c::1	eth1	Após a mobilidade do nó móvel

Tabela 5.4: Tabela de Roteamento 252 do agente domiciliar

O Banco de Dados da Política de Roteamento ou *Routing Policy Database* (RPDB) controla

a ordem na qual o *kernel* faz sua busca através das tabelas de roteamento. Cada regra possui uma prioridade, e as regras são examindadas sequencialmente da regra 0 até a regra 32767.

O MIPv6 adiciona algumas regras no RPDB direcionando para tabela 252, no nó móvel e no agente domiciliar, mostradas nas tabelas 5.5 e 5.5, respectivamente.

Prioridade	Regra	Ação
0:	from all	lookup local
1001:	**from 2001:c::2**	**lookup 252**
1002:	from fe80::/64	lookup main
1002:	from 2001:d::/64	lookup main
1003:	from 2001:c::2	blackhole
32766:	from all	lookup main

Tabela 5.5: Regras de Roteamento do Nó móvel

Prioridade	Regra	Ação
0:	from all	lookup local
1005:	**from 2001:c::2**	**lookup 252**
32766:	from all	lookup main

Tabela 5.6: Regras de Roteamento do Home Agente

Detalhes da Formação de Túneis

O túnel no nó móvel é criado antes de acontecer qualquer mobilidade. No agente domiciliar o túnel com o nó móvel é criado no recebimento do BU. As tabelas 5.8 e 5.7 apresentam o endereçamento do túnel, no HA e MN, antes e depois da mobilidade.

Nome	Remoto	Local	Interface	Considerações
ip6tnl1	2001:c::1	2001:c::2	eth0	Antes da mobilidade do nó móvel
ip6tnl1	2001:c::1	2001:d::fcfd:32ff:feff:42c9	eth0	Após a mobilidade do nó móvel

Tabela 5.7: Descrição dos túneis do nó móvel

Nome	Remoto	Local	Interface	Considerações
*	*	*	*	Antes da mobilidade do nó móvel
ip6tnl1	2001:d::fcfd:32ff:feff:42c9	2001:c::1	eth0	Após a mobilidade do nó móvel

Tabela 5.8: Descrição dos túneis do agente domiciliar

5.2.3 Variação 2: MIPv6 com Otimização de Rota

Análise da Troca de Mensagens

Os dados obtidos na saída do comando *tcpdump* podem ser observados na figura 5.3. Analisando os dados recolhidos consegue-se observar:

1. O nó móvel recebe uma mensagem de RA do roteador de acesso da rede visitada, o que indica que ocorreu mobilidade. Antes de realizar o movimento o nó móvel respondia um *echo request* do nó correspondente;

2. O nó móvel realiza o teste de duplicação de endereço (DAD) do CoA gerado com o prefixo da rede visitada (2001:d::fcfd:32ff:feff:42c9);

3. O nó móvel a mensagem de BU para o Agente domiciliar informando o atual endereço. Note que o CoA é o endereço fonte do pacote e o HoA é informado na opção DSTOPT;

4. O agente domiciliar envia um BA, confirmando o registro;

5. O nó móvel volta a receber *echo request* do nó correspondente, só que ainda através do túnel. A otimização de rota só irá ocorrer após o processo de RRP. Note os endereços IP fonte e destinos deste pacote. O primeiro nível IP do agente domiciliar(2001:c::1) para nó móvel (2001:d::fcfd:32ff:feff:42c9). No segundo nível, do nó correspondente (2001:a::fcfd:65ff:fe34:8a44).

6. O nó móvel inicia o processo de *Return Routability Procedure* (RRP) enviando via agente domiciliar a mensagem de *Home Test Init*(HoTi) ao nó correspondente, para adquirir a *home keygen token*.

7. O nó móvel recebe a mensagem de *Home Test*(HoT) do nó correspondente, via agente domiciliar.

8. O nó móvel da continuação ao processo de RRP enviando a mensagem de *Care-of Test Init*(CoTi) ao nó correspondente, para adquirir a *care-of keygen token*. Essa mensagem não passa pelo túnel, ela é endereçada diretamente ao nó correspondente.

9. O nó móvel recebe a mensagem de *Care-of Test*(CoT) do nó correspondente, não passando pelo agente domiciliar.

10. O nó móvel envia um BU ao nó correspondente com o Kbm, concluindo o processo de RRP.

Figura 5.3: Cenário 1 : Captura de mensagens com Otimização de Rota

11. O nó móvel recebe uma mensagem de *echo request*, endereçada ao CoA, não passando mais pelo agente domiciliar. Note que a opção DSTOPT que trás o endereço domiciliar (2001:c::2) do nó móvel.

Estado das Tabelas de Roteamento e de Regras

Sobre as tabelas e as regras de roteamento, quando o nó móvel e o nó correspondente estão configurados para realizar a otimização de rota, não existe nenhuma alteração em relação a variação 1.

Detalhes da Formação de Túneis

Sobre os túneis, não há nenhuma alteração em relação ao cenário configurado com tunelamento bidirecional sem otimização de rota.

5.3 Cenário 2 - Uso do Protocolo HMIPv6

5.3.1 Descrição da Topologia

Com o fim de testar se a implementação experimental do HMIPv6 atende as funcionalidades do protocolo, o cenário 2 foi montando com os seguintes componentes (ver figura 5.4):

- Rede domiciliar com o prefixo 2001:d::/64 (VLAN 4), onde estão o agente domiciliar (HA) e o nó móvel (MN), no início do teste;
- Rede com o prefixo 2001:a::/64 (VLAN 1), onde estão presentes um roteador de acesso (AR1) e o nó correspondente (CN). Este último irá se comunicar com o nó móvel durante o teste;
- Um domínio formado pelo roteador MAP, que executa o *daemon* MIPL na função de agente domiciliar, e dois roteadores de acesso (AR2 e AR3), que possibilitam o teste de mobilidade em um mesmo domínio. Fazem parte deste domínio as sub-redes com os prefixos: 2001:c::/64 (VLAN 3), 2001:e::/64 (VLAN 5) e 2001:f::/64 (VLAN 6).

Durante a execução do cenário:

1. O nó móvel iniciará na rede domiciliar se comunicando via *ping6* com o nó correspondente.

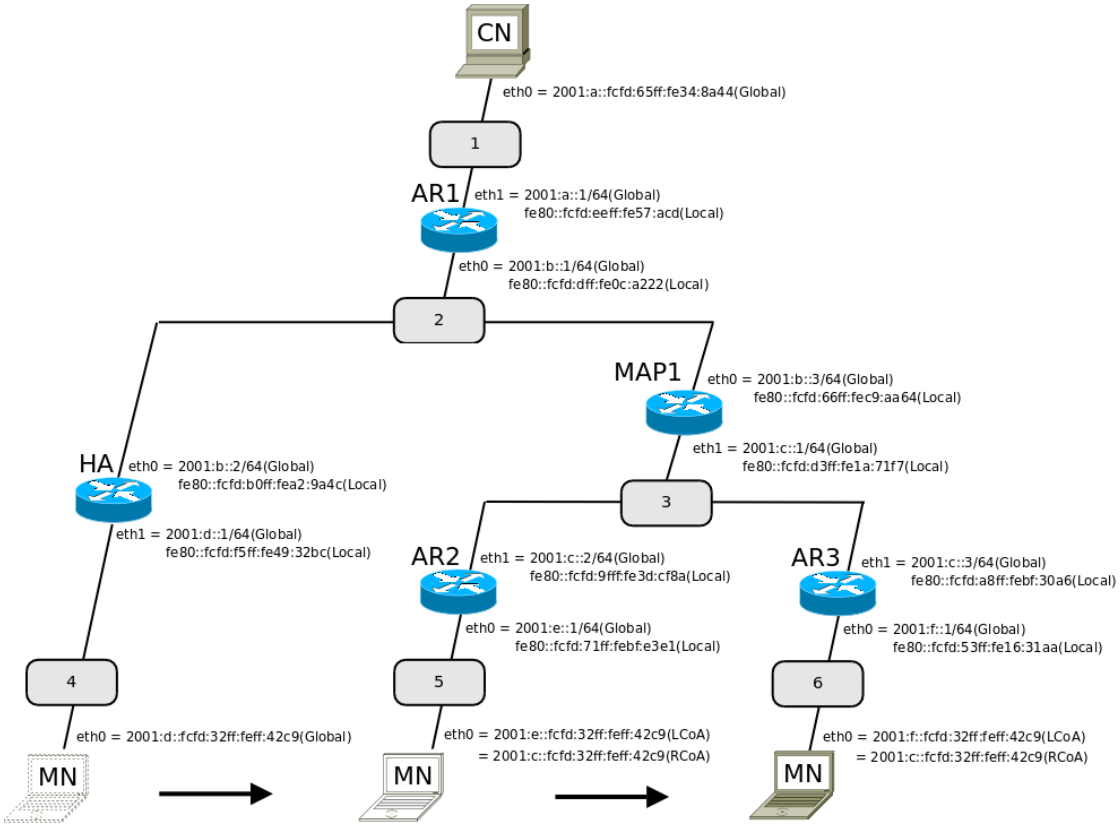

Figura 5.4: Topologia do Cenário 2

2. Uma situação de mobilidade será ocasionada e o nó irá migrar para a VLAN 5, domínio do MAP1, caracterizando uma mobilidade global.

3. Na segunda situação de mobilidade o nó irá migrar para a VLAN 6 caracterizando uma mobilidade local.

5.3.2 Análise da Troca de Mensagens

Mobilidade Global

Durante a primeira situação de mobilidade do cenário 2, uma mobilidade global, os dados coletados com a ferramenta *tcpdump* estão disponíveis na figura 5.5. Por meio deles pode-se observar que:

1. Inicialmente o nó móvel está na sua rede domiciliar recebendo *echo request*s do nó correspondente e mensagens RA do agente domiciliar. Pode-se evidenciar este fato pelo campo src=fe80::fcfd:f5ff:fe49:32bc do RA que é o endereço link local do agente domiciliar. A mensagem 1 evidencia a mobilidade do nó móvel pois ele passa a receber mensagens RA

```
21:19:57.710562 IP6 (hlim 255, next-header ICMPv6 (58) payload length: 56)
        fe80::fcfd:f5ff:fe49:32bc > ff02::1: ICMP6, router advertisement,
        length 56, Flags [home agent], pref medium, router lifetime 9s
21:19:58.466671 IP6 (hlim 62, next-header ICMPv6 (58) payload length: 64)
        2001:a::fcfd:65ff:fe34:8a44 > 2001:d::2:
        ICMP6, echo request, seq 128
21:19:58.466715 IP6 (hlim 64, next-header ICMPv6 (58) payload length: 64)
        2001:d::2 > 2001:a::fcfd:65ff:fe34:8a44:
         ICMP6, echo reply, seq 128
21:19:59.879303 IP6 (hlim 255, next-header ICMPv6 (58) payload length: 80)
  (1)   fe80::fcfd:9fff:fe3d:cf8a > ff02::1: ICMP6, router advertisement,
        length 80, Flags [none], pref medium, router lifetime 9s
21:19:59.894923 IP6 (hlim 1, next-header Options (0) payload length: 56)
        :: > ff02::16: HBH(rtalert:0x00) ICMP6,
        multicast listener report v2, length 48, 2 group record(s)
        [gaddr ff02::1:ff00:2 to_ex, 0 source(s)][|icmp6]
21:20:00.035383 IP6 (hlim 255, next-header ICMPv6 (58) payload length: 24)
  (2)   :: > ff02::1:ffff:42c9: [icmp6 sum ok] ICMP6, neighbor solicitation,
        length 24, who has 2001:e::fcfd:32ff:feff:42c9
21:20:00.357481 IP6 (hlim 255, next-header ICMPv6 (58) payload length: 24)
        :: > ff02::1:ffff:42c9: [icmp6 sum ok] ICMP6, neighbor solicitation,
        length 24, who has fe80::fcfd:32ff:feff:42c9
21:20:02.867308 IP6 (hlim 64, next-header unknown (60) payload length: 40)
        2001:e::fcfd:32ff:feff:42c9 > 2001:c::1:
  (3)   DSTOPT (homeaddr: 2001:c::fcfd:32ff:feff:42c9)
        mobility: BU seq#=18242 AH lifetime=262140
21:20:02.870531 IP6 (hlim 64, next-header unknown (60) payload length: 56)
        2001:c::fcfd:32ff:feff:42c9 > 2001:d::1:
  (4)   DSTOPT (homeaddr: 2001:d::2)
        mobility: BU seq#=4741 AH lifetime=262140(alt-CoA: 2001:0:3100::)
21:20:03.901379 IP6 (hlim 255, next-header ICMPv6 (58) payload length: 32)
        fe80::fcfd:9fff:fe3d:cf8a > ff02::1:ffff:42c9: ICMP6,
        neighbor solicitation, who has 2001:e::fcfd:32ff:feff:42c9
21:20:03.901449 IP6 (hlim 255, next-header ICMPv6 (58) payload length: 32)
        2001:e::fcfd:32ff:feff:42c9 > fe80::fcfd:9fff:fe3d:cf8a: ICMP6,
        neighbor advertisement, tgt is 2001:e::fcfd:32ff:feff:42c9,
        destination link-address option (2), length 8 (1): fe:fd:32:ff:42:c9
21:20:03.902553 IP6 (hlim 63, next-header Routing (43) payload length: 40)
        2001:c::1 > 2001:e::fcfd:32ff:feff:42c9:
  (5)   srcrt(len=2, segleft=1, rsv=0x0, [0]2001:c::fcfd:32ff:feff:42c9)
        mobility: BA status=0 seq#=18242 lifetime=262140(padn)
21:20:03.902558 IP6 (hlim 63, next-header IPv6 (41) payload length: 80)
        2001:c::1 > 2001:e::fcfd:32ff:feff:42c9:
  (6)   IP6 (hlim 63, next-header Routing (43) payload length: 40)
        2001:d::1 > 2001:c::fcfd:32ff:feff:42c9: [|srcrt]
21:20:04.512773 IP6 (hlim 63, next-header IPv6 (41) payload length: 144)
        2001:c::1 > 2001:e::fcfd:32ff:feff:42c9:
        IP6 (hlim 63, next-header IPv6 (41) payload length: 104)
        2001:d::1 > 2001:c::fcfd:32ff:feff:42c9: [|ip6]
21:20:04.512856 IP6 (hlim 64, next-header IPv6 (41) payload length: 104)
        2001:c::fcfd:32ff:feff:42c9 > 2001:d::1:
  (7)   IP6 (hlim 64, next-header ICMPv6 (58) payload length: 64)
        2001:d::2 > 2001:a::fcfd:65ff:fe34:8a44: [|icmp6]
```

Figura 5.5: Cenário 2: Captura de mensagens Mobilidade Global

do roteador de acesso AR2 (src=fe80::fcfd:71ff:febf:e3e1, endereço link local de AR2) com a opção de MAP propagada pelo roteador MAP. Esta opção não é mostrada pelo tcpdump. Com este RA o nó móvel forma os endereços LCoA(2001:e::fcfd:32ff:feff:42c9) e RCoA(2001:e::fcfd:32ff:feff:42c9);

2. O nó móvel realiza o DAD do LCoA gerado com o prefixo da rede visitada (endereço fe80::fcfd:32ff:feff:42c9);

3. O nó móvel envia a mensagem de BU para o MAP (dst=2001:c::1) informando a dupla (LCoA,RCoA). Note que LCoA é o endereço fonte do pacote e o RCoA é informado na opção DSTOPT;

4. O nó móvel envia a mensagem de BU para o Agente Domiciliar (dst=2001:d::1) informando a dupla (RCoA,HoA). Note que RCoA é o endereço fonte do pacote e o HoA é informado na opção DSTOPT;

5. O nó móvel recebe a mensagem de BA do MAP como confirmação de registro bem sucedido;

6. O nó móvel recebe a mensagem de BA do agente domiciliar, como confirmação de registro bem sucedido, via MAP, de forma tunelada. Note no primeiro nível os endereços IP do MAP (2001:d::1) e IP LCoA (2001:e::fcfd:32ff:feff:42c9). Na sequência também chega um pacote provindo do nó correspondente que, embora não explícito pelo *tcpdump*, provavelmente pertence a um *echo request* duplamente tunelado. Note os endereços IPs fonte e destino destes pacotes: no primeiro nível IP do MAP para IP LCoA e, no segundo nível, IP do Agente domiciliar para IP RCoA;

7. Envia um *echo reply* ao nó correspondente, com destino ao agente domiciliar, com a mensagem tunelada ao nó correspondente, via MAP.

Mobilidade Local

Na segunda situação de mobilidade do cenário 2, uma mobilidade local, as mensagens trocadas pelo nó móvel estão disponíveis na figura 5.6. Por meio delas pode-se observar que:

1. É realizada a mobilidade e passa receber mensagens RA do roteador de acesso AR3 ainda sobre cobertura do mesmo MAP.

2. Realiza o DAD do novo LCoA gerado.

3. Envia a mensagem de BU para o MAP com os seguintes campos:

   - Care-of-Address = LCoA;
   - Endereço domiciliar = RCoA, gerado a partir da opção de MAP da mensagem RA.

   Como é uma mobilidade local não é necessário enviar mensagem ao agente domiciliar.

4. Recebe a mensagem de BA do MAP como confirmação de registro bem sucedido.

5. Recebe um *echo reply* do nó correspondente, via MAP com a mensagem tunelada do agente domiciliar.

5.3.3 Estado das Tabelas de Roteamento e de Regras

Sobre as tabelas e as regras de roteamento, a implementação do HMIpv6 não faz nenhuma alteração. As tabelas de roteamento do cenário 2 se diferem muito pouco das do cenário 1, tanto na mobilidade global como na local. As tabelas 252 (5.9) e a principal (5.10), após a primeira situação de mobilidade estão disponíveis logo abaixo. E seguem algumas considerações sobre as regras e tabelas de roteamento do cenário 2:

Destino	**Via**	**Próximo salto**
2001:a::fcfd:65ff:fe34:8a44	ip6tnl1	2001:a::fcfd:65ff:fe34:8a44
2001:d::1	ip6tnl1	2001:d::1
default	ip6tnl1	::

Tabela 5.9: Cenário 2: Tabela de Roteamento 252

Destino	**Via**	**Próximo salto**
2001:d::fcfd:32ff:feff:42c9	ip6tnl2	::
2001:e::2	ip6tnl1	::
2001:10::/64	eth0	::
default	eth0	fe80::fcfd:53ff:fe16:31aa

Tabela 5.10: Cenário 2: Tabela principal de roteamento

- antes da primeira situação de mobilidade, as tabelas e regras de roteamento do cenário 2 são iguais as cenário 1 antes da mobilidade;
- após a mobilidade:
  - as regras de roteamento são as mesmas que as do cenário com MIPv6;

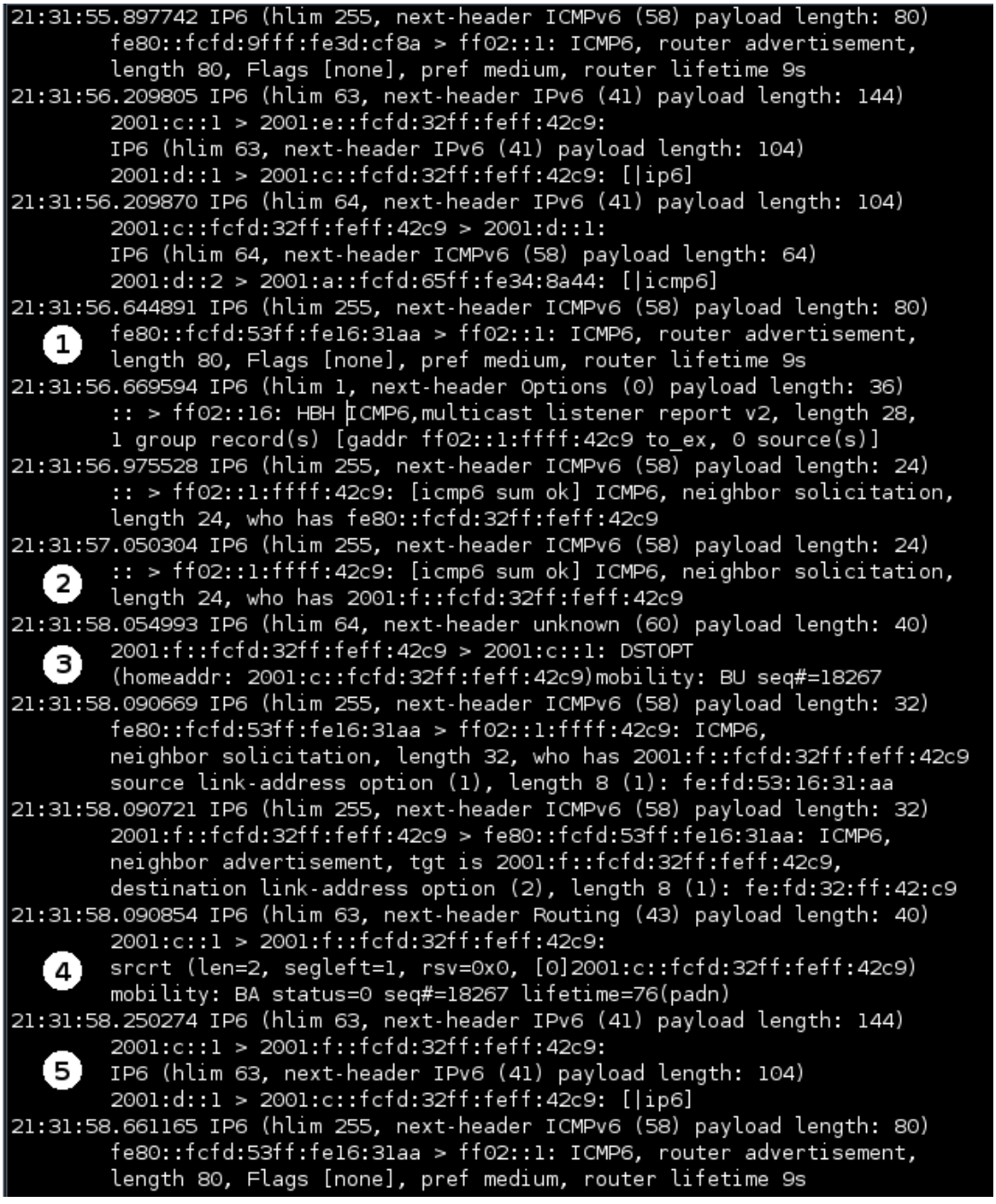

Figura 5.6: Cenário 2: Captura de mensagens Mobilidade Local

- a tabela 252 se mantém a mesma que a do cenário com MIPv6.
- a rota padrão é alterada para o roteador de acesso AR3 e depois na segunda situação de mobilidade para o roteador AR4;
- adicionada a rota que pacotes com destino ao RCoA são via ip6tnl2;
- as outras rotas se mantém as mesmas que as de um cenário com MIPv6.

5.3.4 Detalhes da Formação de Túneis

Os detalhes da formação dos túneis do cenário 2 diferem um pouco do cenário 1, a implementação do HMIPv6 é a responsável. Nas tabelas 5.11 e 5.12 é possível ver os túneis presentes no nó móvel após as mobilidades global e local. As principais diferenças do cenário 1 são:

- a criação do túnel (ip6tnl2) entre o LCoA do nó móvel (2001:e::fcfd:32ff:feff:42c9) e o MAP1 (2001:c::1). Este túnel é necessário, pois os pacotes oriundos do Agente domiciliar serão endereçados agora ao RCoA(2001:c::fcfd:32ff:feff:42c9). Como no MAP1 há uma associação LCoA-RCOA, este túnel passa a encaminhar os pacotes do HA ao nó móvel;
- para que os pacotes vindos via MAP1 destinados ao endereço domiciliar (2001:d::2) sejam entregues ao nó móvel é necessário a alteração dos pontos finais do túnel entre o nó móvel e o Agente domiciliar, para o RCoA e o endereço do agente domiciliar, e a interface que este túnel está ligado agora deve ser o outro túnel criado entre nó móvel e o MAP1.

Os túneis dos MAP1 e agente domiciliar se mantém iguais aos de um cenário rodando o MIPv6.

Nome	Remoto	Local	Interface
ip6tnl1	2001:d::1	2001:c::fcfd:32ff:feff:42c9	ip6tnl2
ip6tnl2	2001:c::1	2001:e::fcfd:32ff:feff:42c9	eth0

Tabela 5.11: Cenário 2: Túneis Mobilidade Global

Após a mobilidade local verificamos que os dois túneis (ver 5.12) continuam presentes no nó móvel. Porém, o ponto local do túnel *ip6tnl2* foi alterado para o novo LCoA gerado (2001:f::fcfd:32ff:feff:42c9) na sub-rede do AR4.

Nome	Remoto	Local	Interface
ip6tnl1	2001:d::1	2001:c::fcfd:32ff:feff:42c9	ip6tnl2
ip6tnl2	2001:c::1	2001:f::fcfd:32ff:feff:42c9	eth0

Tabela 5.12: Cenário 2: Túneis Mobilidade Local

5.4 Análise dos Tempos de *Handover*

O tempo do *handover* é um dos maiores problemas do MIPv6, esta seção do trabalho pretende fazer uma análise deste processo, apontar as causas e possíveis soluções. Como base para estudo dos tempos de *handover* foi utilizado o cenário 1 com tunelamento bidirecional.

Para verificar a perda na taxa de transmissão em um processo de *handover*, foi realizado o segundo teste especificado na seção sobre a metodologia na realização dos cenários. Utilizando a ferramenta *GNUPLOT* com a saída do teste foi gerado um gráfico, que pode ser observado na figura 5.7.

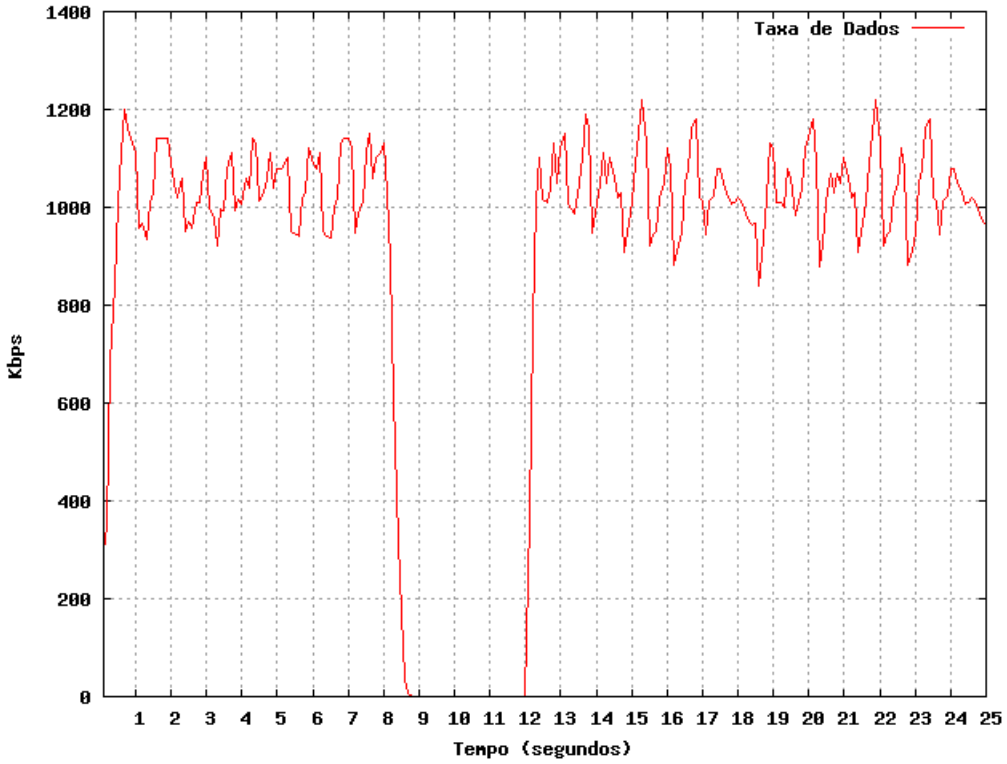

Figura 5.7: Taxa de transmissão em *Handover*

A partir da figura é possível concluir que cada *handover* é de aproximadamente 3 segundos, há perdas consideráveis de pacotes que prejudicariam muito comunicações interativas como videoconferência e *VoIP*, porém, navegação em páginas da *Internet* e visualização de *e-mail* a instabilidade é tolerável.

Como o cenário de estudo foi simulado todos os tempos medidos são estimados. Porém, segundo algumas bibliografias estudadas que realizaram experimentos fisicamente os tempos medidos na simulação estão dentro do esperado.

Na tabela 5.13, encontram-se os tempos das fases do processo de *handover* referentes ao cenário estudado.

Fase	**Tempo (ms)**	**Media %**
TD	721	23,11
TA	1371	43.92
TR	1029	32.97
TH	3121	100

Tabela 5.13: Latência no *Handover* do cenário estudado

Algumas tentativas podem ser feitas para tentar diminuir a latência do *handover* nas fases de detecção de movimento e de configuração do CoA.

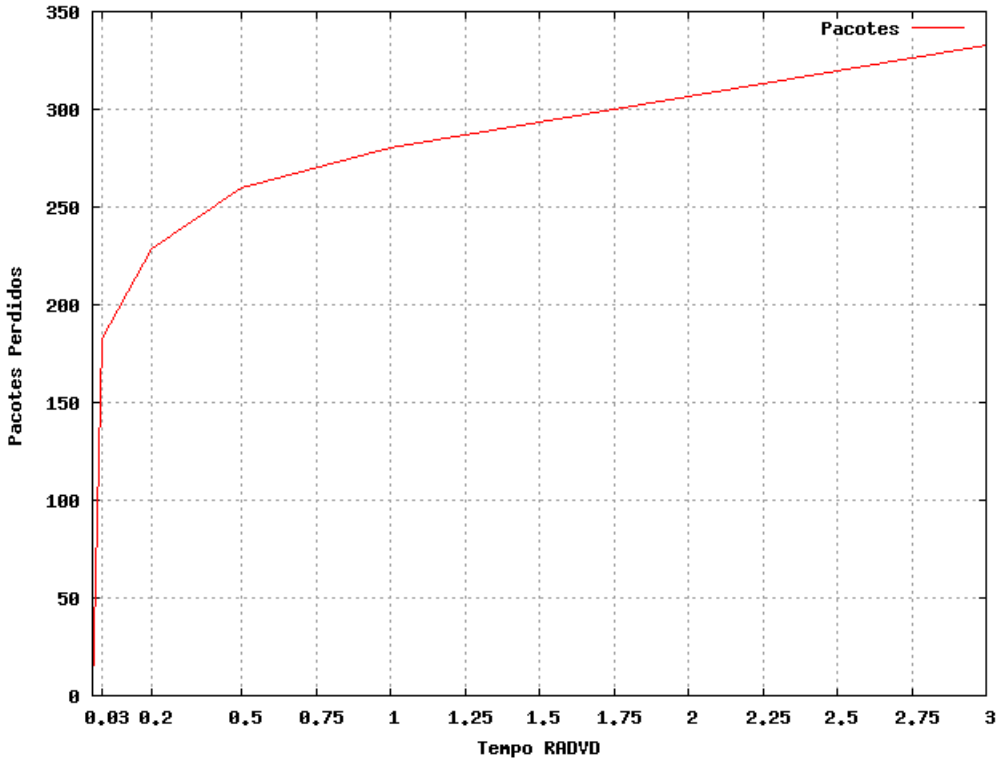

Figura 5.8: Diferentes intervalos entre as mensagens *Router Advertisement*

Na tentativa de diminuir o tempo da fase de detecção de movimento, podemos diminuir o intervalo entre as mensagens de RA do roteador presente na rede que irá ser visitada pelo nó móvel. Pois, é por meio desta mensagem que o nó móvel detecta o movimento e desencadeia o processo de registro.

Para verificar se há uma melhora no processo de handover. O cenário estudado foi repetido variando o intervalo entre as mensagens de RA. O resultado deste teste esta disponível na figura 5.8.

Com os dados levantados, é possível perceber que o intervalo entre as mensagens de RA esta diretamente ligado com a perda de pacotes durante o *handover*. Porém, esta não é uma boa prática para tentar amenizar a latência do handover, pois um número muito grande de mensagens *multicasting* inundaria a rede prejudicando a performance principalmente em redes sem fio.

No processo de formação de um novo CoA o DAD é necessário para se certificar que o endereço formado é exclusivo. Para o teste ser bem sucedido nenhum nó vizinho deve enviar um *Neighbor Advertisement* (NA), em resposta ao teste, ou NS, como o mesmo endereço em questão, em um período de *1000ms* segundo a RFC 2462 (THOMSON; NARTEN, 1998). Este tempo de espera compromete a fase de configuração do CoA.

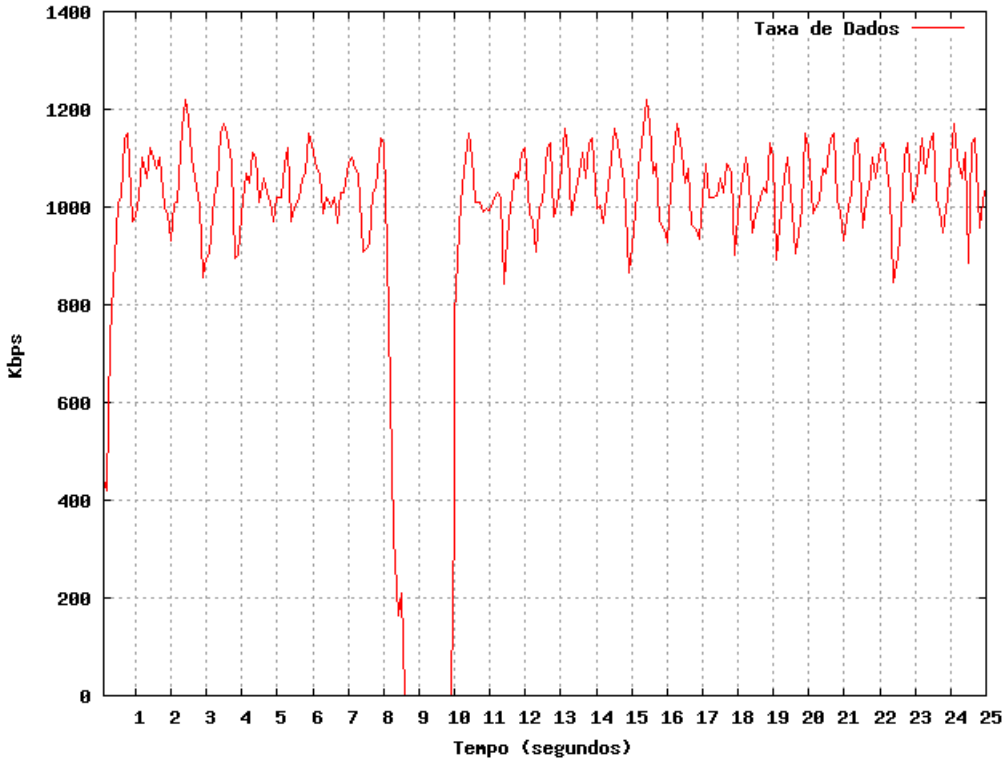

Figura 5.9: Taxa de transmissão em *Handover* sem DAD e baixo intervalo de RA

Existe uma variável chamada *dad_transmits* no *kernel* do Linux que é usada para configurar o DAD em um nó. Com a intenção de diminuir a latência na fase de configuração do CoA, foi configurado a variável com o valor 0, isso significa que o procedimento de DAD é cancelado, e foi alterado o valor do intervalo das mensagens de RA no roteador de acesso da rede visitada para 0.03 segundos. Então foi repetido o teste. O resultado pode ser observado na figura 5.9.

A Partir, deste teste é possível perceber uma queda de 3 segundos para aproximadamente 1 segundo no tempo do *handover* com as mudanças realizadas. Entretanto há um protocolo dedicado a acelerar o processo de *handover* o *Fast Handovers for Mobile IPv6* (FMIPv6). A idéia do FMIPv6 é providenciar informação relativa à camada de rede, com o objetivo de prever ou responder prontamente a um evento de *handover* (KOODLI, 2005).

5.5 Conclusões sobre a Análise dos Cenários

Com a realização dos experimentos podemos constatar o funcionamento do protocolo, 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.

Na realização do segundo cenário, foi possível verificar minimamente o funcionamento do HMIPv6, claro que os problemas ainda existentes na implementação não permitiram comprovar uma melhora no *handover*. Porém, pode-se ver um avanço já que não existe nenhuma versão recente deste protocolo.

Com a realização dos cenários é possível verificar que o *handover* é o é o grande problema MIPv6, por isso, a necessidade de outras extensões para trabalharem juntos com o MIPv6 como: o HMIPv6 e FMIPv6.

6 *Conclusões e Perspectivas*

O trabalho apresentou uma plataforma de testes para avaliação dos protocolos de mobilidade da camada rede, qual possibilitou a avaliação das extensões de mobilidade, que mostrou um funcionamento estável. A plataforma de testes implementada atingiu o objetivo desejado e vem demonstrando bons resultados nos cenários realizados, principalmente por:

- permitir realizar mobilidade entre sub-redes, devido a implementação realizada no *switch* virtual;
- facilitar a construção e realização dos cenários;
- não necessitar de conhecimentos específicos obre as máquinas virtuiais UML, compilação do *kernel linux* e dos *daemons* dos protocolos;
- por ser flexível podendo ser utilizada para outros estudos sobre redes de computadores;

Na realização do trabalho foi gerado uma versão experimental do HMIPv6, a qual permite estudos iniciais já que não havia nenhuma versão recente. Mesmo não tendo alcançado uma versão completa do protocolo, a realização do trabalho contribui para o enriquecimento de vários conhecimentos. Princilpamente na linguagem de programação C, desenvolvimento em sistemas *linux*, *Application Programming Interface* (API) de soquetes IPv6, *Portable Operating System Interface* (POSIX) *Threads*.

Como perspectivas de trabalhos futuros, pode-se enumerar aspectos associados à plataforma de mobilidade e às extensões ao protocolo MIPL. No que se refere a plataforma, os seguintes trabalhos podem ser citados, além das correções dos problemas conhecidos já apontados no capítulo 4:

1. Concepção de um simulador da rede sem fio nas máquinas UML, de forma similar ao desenvolvido em (GUFFENS; BASTIN; BONAVENTURE,), possibilitando a análise dos problemas associados ao enlace e seus impactos na camada de rede;

2. Implementação de modelos de mobilidade clássicos, de forma a produzir movimentos circulares, retilíneos, randomizados, ou mesmo, a acoplagem a um simulador de movimentos urbanos, tal como o (KRAJZEWICZ et al., 2002);

3. Incrementos na linguagem de descrição da UML, de forma a incorporar aspectos de configuração do IPv4 e facilitar a incorporação de novos protocolos e *deamons*;

4. Melhorias na configuração dos terminais virtuais e consoles, de forma a obter Uma melhor visualização do sistema por parte do usuário.

5. Desenvolvimento de uma interface gráfica para a configuração das máquinas UML;

6. Criação automática de uma conexão virtual entre a máquina hospedeira e as máquinas virtuais;

7. Melhorias no instalador do sistema.

No que se as extensões ao MIPL, as seguintes perspectivas são vislumbradas:

- implementação da otimização de rotas no HMIP, ou seja, a realização de BUs diretamente com os nós correspondentes;

- Exploração da implementação do HMIP para estudar melhorias na seleção de MAPs por parte dos nós móveis;

- Estudo da integração do HMIP com protocolos de sinalização de QoS, desenvolvendo módulos/bibliotecas para o interfaceamento entre eles;

- Incorporação do protocolo FMIP (KOODLI, 2005) na plataforma de mobilidade;

Finalmente, a plataforma guml4mip está sendo avaliada para uso em sala de aula na disciplina de Redes de Computadores III do Curso de Tecnologia em Telecomunicações do Campus São José do IF-SC. Além disto, existem demandas externas de acadêmicos interessados na sua utilização para fins de pesquisa. Por este motivo, entende-se que o projeto atingiu plenamente os seus objetivos e plantou sementes para futuros trabalhos na instituição.

APÊNDICE A – IPv6

Considerando o grande crescimento da *Internet*, o protocolo que é utilizado hoje, o *Internet Protocol version 4* (IPv4), é considerado incapaz de suprir a necessidade da demanda por novos endereços na *Internet*.

A previsão atual para a exaustão de todos os endereços IPv4 livres é Março 2011 (ARANO, 2009), o que significa que o uso do *Internet Protocol version 6* (IPv6) é inevitável num futuro próximo.

Com o intuito de resolver o problema de números de endereço e as principais falhas encontradas no IPv4, foi proposto o protocolo IPv6. As principais melhorias são:

- Ampliação do Espaço de Endereçamento de 32 bits para 128 bits;
- Melhoria na estrutura (formato) do pacote IP;
- Melhoria no processo de autoconfiguração;
- Inserção de mecanismos para tratamento de mobilidade, segurança e facilidades para o gerenciamento de *Quality of Service* (QoS).

A.1 A estrutura de cabeçalhos

Um datagrama IPv6 é constituído por um cabeçalho base de tamanho fixo de 40 bytes, seguido de zero ou mais cabeçalhos de extensão. Na figura A.1, é ilustrado um cabeçalho IPv6 com um cabeçalho de extenção de mobilidade.

O fato do cabeçalho IPv6 possuir um tamanho fixo é uma vantagem, pois acelera o processamento dos pacotes nos roteadores, uma vez que não há necessidade de calcular o tamanho de alguns campos e do cabeçalho como um todo.

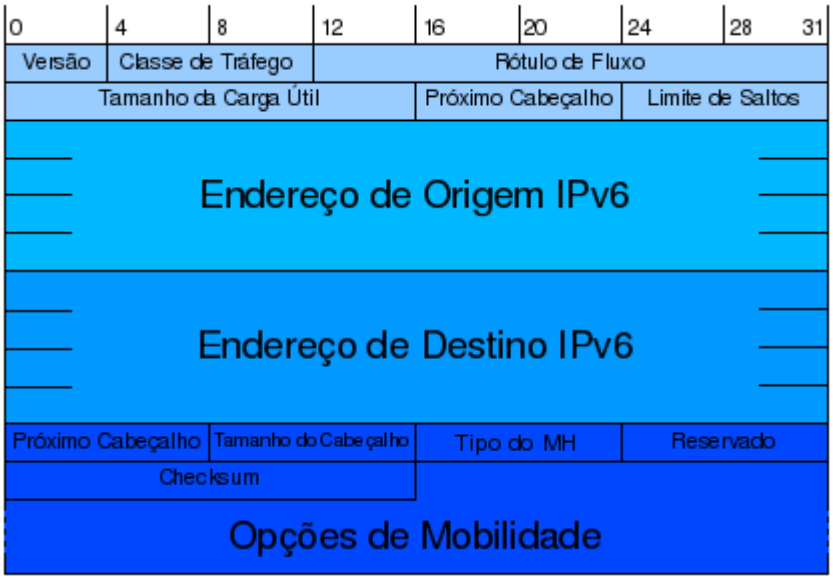

Figura A.1: Estrutura do Cabeçalho do IPv6

A redução do números de campos utilizados, excluindo os campos com pouca utilidade prática, também contribui para a diminuição do tempo gasto em processamento pelos roteadores. Entre eles, dois que merecem destaque, *checksum* e o de fragmentação.

A função do campo de *checksum* era detectar erros que afetassem ao cabeçalho IP, não detectando no entanto erros no restante do pacote. Hoje os mecanismos de deteção de erros Ethernet e PPP são bastante eficientes, o que permitiu a exclusão do campo *checksum*. Com isso os roteadores não precisam recalcular o *checksum* a cada salto, melhorando a sua performace.

Quanto ao campo fragmentação, este foi excluído, pois decidiu-se que pacotes não serão mais fragmentados por roteadores. Caso algum roteador da rota detecte que o pacote é muito grande para o *Maximum Transmit Unit* (MTU) então ele avisa o nó fonte com ICMPv6 *Packet Too Big*, que inclui o tamanho da MTU do enlace problema.

Desta forma há um ganho de performance no roteamento, pois é eliminada a necessidade de um roteador fragmentar vários pacotes.

A.2 O endereçamento IPv6

O endereçamento no IPv6 passa a ser de 128 bits, incluindo o prefixo de rede e sufixo do host. Não existem classes de endereços, como acontecia no IPv4.

Os endereços IPv6 são representados normalmente como oito grupos de 4 dígitos hexadecimais. Por exemplo:

```
3ffe:88a6:3a58:3d8a:021e:8cff:fe91:f6dd
```

Se um grupo de dígitos for zeros, eles pode ser omitido. Por exemplo:

```
3ffe:88a6:3a58:0000:0000:0000:0000:f6dd
```

é o mesmo endereço IPv6 que:

```
3ffe:88a6:3a58::f6dd
```

No IPv6 existem três tipos de endereços:

- *unicast* - cada endereço identifica uma interface (dispositivo).
- *multicast* - cada endereço identifica múltiplas interfaces. É enviada uma cópia para cada interface. O IPv6 eliminou o *broadcast* e enriqueceu o processo de *multicast*;
- *anycast* - cada endereço pode ser atribuído a múltiplas interfaces. Um datagrama é enviado para um dos dispositivos, por exemplo, o mais próximo.

A.3 ICMPv6

O *Internet Control Message Protocol version 6* (ICMPv6) (CONTA; DEERING; GUPTA, 2006) permite enviar mensagens de erro e informação.

Em comparação ao *Internet Control Message Protocol version 4* (ICMPv4), podedos citar dentro das principais mudanças para o ICMPv6 a incorporação da função de converter o endereço IP em endereço físico, função que no IPv4 era feita por um protocolo a parte, o *Address Resolution Protocol* (ARP) (PLUMMER, 1982).

As mensagens de descoberta de informações de vizinhança *Neighbor Discovery* (ND), podem ser usadas para desempenhar a função equivalente ao ARP/RARP, para detecção de IP duplicado *Duplicate Address Detection* (DAD), fazer anúncio ou solitação de roteadores e outros.

O protocolo *Internet Group Management Protocol* (IGMP) que trata o multicast também foi incorporado no ICMPv6.

No MIP, para fazer os registro de localização, também é utilizado as mensagens ICMP.

A.4 Autoconfiguração

A autoconfiguração é um dos pontos fortes do IPv6. Ela visa liberar o usuário da tarefa de configuração, tornando-a automática e transparente. Ela pode ser feita por *stateful autoconfiguration*, onde o endereço e outros parâmetros de configuração são providos diretamente de um servidor, por exemplo, os mecanismos *Dynamic Host Configuration Protocol version 6* (DHCPv6) e *Point to Point Protocol version 6* (PPPv6). Ou pode ser feita por por *stateless autoconfiguration* (THOMSON; NARTEN, 1998), no qual o endereço é gerado combinando os prefixos divulgados pelos roteadores e o *Media Access Control* (MAC) do nó.

ANEXO A – Arquivo de Configuração do GUML4MIP

```
[general]
; Nome para a máquina UML.
name = MN1

; Quantidade de memoria (em MB) que será alocado para a máquina UML.
mem = 128

; Opções extras que podem ser adicionadas a linha de comando do kernel UML.
;extraopts = "devfs=mount ubd=3"

; Escpecifica o tipo de função da maquina UML,
; podendo ser 'router' ou 'node'. Opção padrão é nó.
;type = node

[eth0]
; Configuração de Rede
; Você pode criar varias seções ethN como esta, para configurar
; multiplas interfaces de rede.

; Nota somente o tipo = daemon esta propriamente suportado e testado
; até o momento.
type = daemon

; Endereço MAC atribuído para a interface. Use mac = random para gerar
; um MAC aleatório.
mac = random

; Endereço IP atribuído para a interface.
ip = 2000:a::1/64

; O socket do switch virtual que esta interface irá se conectar.
```

```
32 ;socket = /tmp/net.ctl
33
34 ; Você pode especificar qual Vlan a maquina UML irá iniciar.
35 ; A opção padrão é na Vlan 0.
36 ;vlan = 0
37
38 [disks]
39 ; Você pode especificar os discos da máquina UML, todos devem ter o
40 nome ubdN (onde N é um dígito).
41 ubd0 = /usr/share/guml4mip/vm/lenny.img
42 ;ubd1 = /var/lib/uml/example_swap
43
44 [daemons]
45 ; Você pode especificar alguns daemons para a máquina UML executar
46 ; todos devem ter o nome daemonN (onde N é um dígito).
47 daemon0 = /usr/local/sbin/mip6d -c /host/conf/mip6d.conf.HA
48 ;daemon1 = /usr/local/sbin/radvd-hmip -c /host/conf/radvd.conf.AR1
```

Listing A.1: Exempo de arquivo de configuração de uma máquina UML, para o GUML4MIP

Referências Bibliográficas

ALTMEYER, O. Project: SeKeRo. 2009. Disponível em: <http://krypt.cs.uni-sb.de/projects/sekero/docu/node17.html>.

ARANO, T. *IPv4 Exhaustion Counter*. jan. 2009. Disponível em: <http://www.potaroo.net/tools/ipv4/index.html>.

ARKKO, J.; DEVARAPALLI, V.; DUPONT, F. *Using IPsec to Protect Mobile IPv6 Signaling Between Mobile Nodes and Home Agents*. IETF, jun. 2004. RFC 3776 (Proposed Standard). (Request for Comments, 3776). Updated by RFC 4877. Disponível em: <http://www.ietf.org/rfc/rfc3776.txt>.

CONTA, A.; DEERING, S.; GUPTA, M. *Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification*. IETF, mar. 2006. RFC 4443 (Draft Standard). (Request for Comments, 4443). Updated by RFC 4884. Disponível em: <http://www.ietf.org/rfc/rfc4443.txt>.

COSTA, X. P. et al. A MIPv6, FMIPv6 and HMIPv6 handover latency study: analytical approach. *IST Mobile and Wireless Telecommunications Summit*, 2002.

DHANDAPANI, G.; SUNDARESAN, A. Netlink Sockets–Overview. *Information and Telecommunications Technology Center, Department of Electrical Engineering & Computer Science, The University of Kansas*, 1999.

DIKE, J. et al. The User-mode Linux Kernel Home Page. 2009. Disponível em: <http://user-mode-linux.sourceforge.net>.

GUFFENS, V.; BASTIN, G.; BONAVENTURE, O. *An emulation infrastructure for multi-hop wireless communication networks*. [S.l.].

ISHIGURO, K. et al. *Quagga Routing Suite*. 2009. Disponível em: <http://www.quagga.net>.

JOHNSON, D.; PERKINS, C.; ARKKO, J. *Mobility Support in IPv6*. IETF, jun. 2004. RFC 3775 (Proposed Standard). (Request for Comments, 3775). Disponível em: <http://www.ietf.org/rfc/rfc3775.txt>.

JUN, M. *USAGI Project - Linux IPv6 Development Project*. 2009. Disponível em: <http://www.linux-ipv6.org>.

KENT, S.; ATKINSON, R. *IP Authentication Header*. IETF, nov. 1998. RFC 2402 (Proposed Standard). (Request for Comments, 2402). Obsoleted by RFCs 4302, 4305. Disponível em: <http://www.ietf.org/rfc/rfc2402.txt>.

KENT, S.; ATKINSON, R. *IP Encapsulating Security Payload (ESP)*. IETF, nov. 1998. RFC 2406 (Proposed Standard). (Request for Comments, 2406). Obsoleted by RFCs 4303, 4305. Disponível em: <http://www.ietf.org/rfc/rfc2406.txt>.

KIRBY, G. Locating the User. *Communication International*, 1995.

KOODLI, R. *Fast Handovers for Mobile IPv6*. IETF, jul. 2005. RFC 4068 (Experimental). (Request for Comments, 4068). Disponível em: <http://www.ietf.org/rfc/rfc4068.txt>.

KOODLI, R.; PERKINS, C. *Mobile Inter-networking with IPv6: Concepts, Principles and Practices*. [S.l.]: Wiley-Interscience, 2007.

KRAJZEWICZ, D. et al. SUMO (Simulation of Urban MObility); An open-source traffic simulation. In: *4th Middle East Symposium on Simulation and Modelling (MESM2002)*. [S.l.: s.n.], 2002. p. 183–187.

LURUO, K.; KHANVILKAR, S. Virtual Networking with User-Mode Linux. *Linux for You–Pro, Mar*, 2005.

LYRA, C.; TORRES, P. *Construindo Roteadores com Linux*.

MOORE, R. Implementation of Hierarchical Mobile IPv6 for Linux. Disponível em: <www.ctie.monash.edu.au/ipv6/hmipv6.htm>.

NARTEN, T.; NORDMARK, E.; SIMPSON, W. *Neighbor Discovery for IP Version 6 (IPv6)*. IETF, dez. 1998. RFC 2461 (Draft Standard). (Request for Comments, 2461). Updated by RFC 4311. Disponível em: <http://www.ietf.org/rfc/rfc2461.txt>.

NIKANDER, P. et al. *Mobile IP Version 6 Route Optimization Security Design Background*. IETF, dez. 2005. RFC 4225 (Informational). (Request for Comments, 4225). Disponível em: <http://www.ietf.org/rfc/rfc4225.txt>.

NUORVALA, V.; PETANDER, H.; TUOMINEN, A. *Mobile IPv6 for Linux (MIPL)*. 2009. Disponível em: <http://www.mobile-ipv6.org>.

PALMER, M. *A GUI management console for User Mode Linux*. 2009. Disponível em: <http://www.hezmatt.org/ mpalmer/guml/>.

PERKINS, C. *IP Mobility Support for IPv4*. IETF, ago. 2002. RFC 3344 (Proposed Standard). (Request for Comments, 3344). Updated by RFC 4721. Disponível em: <http://www.ietf.org/rfc/rfc3344.txt>.

PLUMMER, D. *Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware*. IETF, nov. 1982. RFC 826 (Standard). (Request for Comments, 826). Disponível em: <http://www.ietf.org/rfc/rfc826.txt>.

POSTEL, J. *Internet Protocol*. IETF, set. 1981. RFC 791 (Standard). (Request for Comments, 791). Updated by RFC 1349. Disponível em: <http://www.ietf.org/rfc/rfc791.txt>.

SILVA, C. M. da; ALMEIDA, F. M. de. *Uma plataforma para estudo do IP móvel baseado no projeto guml*. 2009. Disponível em: <http://code.google.com/p/projfin-hmip/>.

SOLIMAN, H. et al. *Hierarchical Mobile IPv6 Mobility Management (HMIPv6)*. IETF, ago. 2005. RFC 4140 (Experimental). (Request for Comments, 4140). Disponível em: <http://www.ietf.org/rfc/rfc4140.txt>.

STEVENS, W.; FENNER, B.; RUDOFF, A. *Unix network programming:(The) sockets networking API. Volume 1*. [S.l.]: Bookman, 2005.

STEVENS, W. et al. *Advanced Sockets Application Program Interface (API) for IPv6*. IETF, maio 2003. RFC 3542 (Informational). (Request for Comments, 3542). Disponível em: <http://www.ietf.org/rfc/rfc3542.txt>.

THOMSON, S.; NARTEN, T. *IPv6 Stateless Address Autoconfiguration*. IETF, dez. 1998. RFC 2462 (Draft Standard). (Request for Comments, 2462). Disponível em: <http://www.ietf.org/rfc/rfc2462.txt>.

VARGA, A. et al. The OMNeT++ discrete event simulation system. In: *Proceedings of the European Simulation Multiconference (ESM'2001)*. [S.l.: s.n.], 2001.

YOSHIFUJI, H. IPv6 Development Status 2005. In: . [s.n.], 2005. Disponível em: <http://www.linux-ipv6.org/materials/200507-NETCONF/img27.html>.