Mestrado em Engenharia Informática
Dissertação
Relatório Final

Desenvolvimento e Estudo de Soluções para Mobilidade Acima da Camada IP

Mara da Silva Benedito Martins
msbm@student.dei.uc.pt

Orientador:
Doutor Fernando Pedro Lopes Boavida Fernandes
Co-Orientador:
Mestre Pedro Vale Pinheiro

Data: 9 de Setembro de 2013

FCTUC DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Mestrado em Engenharia Informática
Dissertação
Relatório Final

Desenvolvimento e Estudo de Soluções para Mobilidade Acima da Camada IP

Mara da Silva Benedito Martins
msbm@student.dei.uc.pt

Orientador:
Doutor Fernando Pedro Lopes Boavida Fernandes
Co-Orientador:
Mestre Pedro Vale Pinheiro

Membros do Júri:
Presidente: António Jorge da Costa Granjal
Vogal: Alcides Miguel Cachulo Aguiar Fonseca
Orientador: Fernando Pedro Lopes Boavida Fernandes

Data: 9 de Setembro de 2013

FCTUC DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Resumo

Com a ascensão do surgimento de tecnologias de comunicação sem fio, diversos estudos vem sendo feito no âmbito de mobilidade de redes, a fim de desenvolver soluções que se adequem a esta nova realidade. Neste sentido, a pilha protocolar TCP/IP que hoje é a principal referência na área das comunicações em rede foi defrontada com a necessidade de se adequar a estas novas tecnologias de comunicação, sendo a transparência um dos principais requisitos deste cenário. Resume-se que, independente da localização de um dispositivo móvel, a pilha protocolar deverá providenciar que a camada de transporte e/ou aplicação não sejam afetadas pela constante mobilidade dos dispositivos.

Embora novas propostas tenham surgido como resposta para esta grande demanda, por exemplo, o Mobile IP, MIPv6, NEMO, LISP e outros, nenhuma foi capaz de solucionar o problema por completo, ou mesmo agregaram alguns malefícios em suas soluções como o aumento de *overheads* e falta de suporte transparente à mobilidade, que inviabilizaram a utilização destas propostas, onde muitas delas não chegaram a ser utilizadas na prática.

Tudo isso, tem gerado a necessidade de estudo e desenvolvimento de propostas com outras alternativas que tornem a mobilidade viável. Neste intuito, este trabalho visou no estudo das soluções para mobilidade de redes propostas pela comunidade científica, onde foram tomadas como referência para o desenvolvimento de uma nova abordagem para o apoio à mobilidade de redes definido na camada de transporte. Esta proposta teve como principal objetivo separar a identificação da localização dos endereços IP dos nós móveis, de forma a permitir a comunicação ponto a ponto, diminuindo desta forma os problemas comuns apresentado em outras propostas e garantindo também a transparência da solução.

Palavras-Chave

Identificadores e Localizadores, ILM, IP Móvel, LISP, Iptables, MIPV6, Mobilidade na Camada de Transporte, Mobilidade de Redes.

Índice

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

AR	Access Router
BA	Binding Acknowledgement
BU	Binding Update
CN	Correspondent Node
CoA	Care-of-Address
CoT	Care of Test
CoTI	Care of Test Init
DHCP	Dynamic Host Configuration Protocol
DHT	Distributed Hash Table
EID	Endpoint ID
ETR	Egress Tunnel Router
FA	Foreign Agent
FN	Foreign Nerwork
HA	Home Agent
HIP	Host Identity Protocol
HN	Home Network
HoA	Home Address
HoT	Home Test
HoTI	Home Test Init
ID	Identificador
IETF	Internet Engineering Task Force
ILM	Identifier and Locator Mapping
ILM-CN	Identifier and Locator Mapping Correspondent Node
ILM-IDd	Identifier and Locator Mapping Destination Identificator
ILM-Ids	Identifier and Locator Mapping Source Identificator
ILM-LOCd	Identifier and Locator Mapping Destination Locator
ILM-LOCs	Identifier and Locator Mapping Source Locator
ILM-MN	Identifier and Locator Mapping Mobile Node
ILM-Server	Identifier and Locator Mapping Server

ILM-UT	Identifier and Locator Mapping Update Table
IMS	Identity Management Server
IP	Internet Protocol
IPv4	Internet Protocol Version 4
IPv6	Internet Protocol Version 6
ITR	Ingress Tunnel Router
LISP	Locator/Identifier Separation Protocol
MN	Mobile Node
MR	Mobile Router
mSCTP	Mobile SCTP
MTU	Maximum Transfer Unit
NEMO	Network Mobility
PERL	Practical Extraction and Report Language
RLOC	Routing Locator
RLS	Registrars e Location Services
RR	Return Routability
RVS	Rendezvous Server
SCTP	Stream Control Transport Protocol
SIP	Session Initiation Protocol
TCP	Transmission Control Protocol
TIPS	Transparent Internet Protocol Sockets
UA	User Agent
VLAN	Virtual LAN

Capítulo 1

Introdução

A internet surgiu no início da década de 80, adotando a arquitetura de protocolos de comunicação TCP/IP, era formada por dispositivos fixos e teve como principal objetivo de sua criação, a comunicação baseada em trocas de arquivos e acesso remoto, onde o IP (Internet Protocol) tinha o intuito de garantir a conectividade fim a fim [Campista et al, 2010]. Com o passar dos anos, a internet revolucionou e popularizou-se, tornando impressionante a diversidade de serviços e novas aplicações que surgiram para este novo cenário, mas que muitas das vezes necessitou de mudanças em sua arquitetura original para que estes novos serviços coexistissem [Barbado, 2007].

Por diversos anos, os computadores permaneceram fixos no espaço físico e quase nunca eram realocados. Possuíam também o endereço IP que acumula duas funcionalidades para um único host: localizá-lo na topologia de rede, utilizado na maioria das vezes para o roteamento de pacotes e identificá-lo na internet, muito utilizado pelas aplicações acima da camada IP.

Nos dias atuais a propagação das tecnologias de comunicação sem fio vêm de forma a facilitar a utilização da internet móvel. Desta forma, quando um dispositivo conecta-se à internet e depois desloca-se para uma nova rede, seu endereço IP é alterado (processo este, conhecido como handover) [Barbado, 2007]. Este deslocamento, tende a trazer diversos problemas de conectividade, sendo uma delas a perda das conexões ativas anteriormente, pelo fato das aplicações utilizarem o endereço IP e Porta para a identificação e comunicação entre os dispositivos.

Diversas soluções foram propostas a fim de contornar o problema da interrupção da conexão no momento do handover, soluções em sua maioria aplicadas à camada IP da pilha protocolar TCP/IP, como o protocolo IPv4 Móvel [Perkins, 2002], IPv6 Móvel [Johnson, 2004] e LISP [Farinacci et al, 2012], entretanto essas soluções ainda são pouco suportadas, pois em muitas delas, problemas com a segurança, excesso de troca de mensagens entre diversos dispositivos e o corrompimento das conexões estabelecidas após o deslocamento e aquisição de um novo endereço IP inviabilizam a utilização destes protocolos.

1.1. Enquadramento

A grande limitação da arquitetura da internet é o fato de se utilizar o endereço IP para a identificação de dispositivos quanto para a sua localização.Assim, a atribuição de um IP para um dispositivo, depende da localidade onde este se encontra que por sua vez associa um dispositivo identificado por um endereço IP à localização que este endereço representa como caminho de roteamento. Para oferecer mobilidade aos dispositivos através da arquitetura original da internet, é necessário desenvolver novos métodos e procedimentos tidos como gerenciamento de mobilidade para contornar essas limitações, a fim de garantir a conectividade entre componentes de um sistema ou aplicação em um cenário que os dispositivos mudam constantemente de endereço IP. [Baptista, 2009].

O gerenciamento de mobilidade de nos dispositivos móveis e redes informáticas é um processo um tanto quanto recente, existindo uma gama de questionamentos sem soluções que necessitam de investigação profunda. Mesmo com a investigação consistente que se tem desenvolvido nesta área nestes últimos anos não existem, até o momento, soluções de mobilidade ao nível da camada IP que permitem a utilização em ambientes de produção. Entretanto, isto se deve ao fato da alta complexidade em se alterar os protocolos atuais da internet, trazendo uma complicação maior para dentro da rede. Percalços como estes e a contabilizar que o protocolo IP não foi desenvolvido para dar suporte à mobilidade, além das propostas hoje existentes criarem na verdade um remendo que produz mais problemas e ineficiências ao protocolo, é devidamente importante explorar diferentes soluções de mobilidade que se apliquem acima da camada IP, a fim de que esta seja mantida sem alterações e com a mínima funcionalidade.

Por estas razões, neste trabalho é proposto um estudo das diversas soluções de gerenciamento de mobilidade a fim de desenvolver um protocolo, o *Identifiers and Locators Mapping* (ILM) que suporte a mobilidade acima do nível da camada IP, que funcione na arquitetura atual da internet de forma a não alterar a pilha protocolar TCP/IP. A proposta do ILM é funcionar de forma transparente para os utilizadores sem qualquer interrupção para o sistema/aplicação dos dispositivos dos utilizadores. Esta proposta é baseada em protocolos híbridos, onde é sugerido a separação do IP e o mapeamento dinâmico entre identificadores e localizadores. A priori, questões relativas à segurança do protocolo ILM não serão discutidos neste trabalho.

1.2. Objetivos

Pretende-se com este trabalho prover um meio de garantir a mobilidade de rede através de uma nova proposta de solução acima da camada IP, necessariamente na camada de transporte. Será detalhada a arquitetura da solução que conduzirá no desenvolvimento de um package de software que, após instalação nos sistemas terminais, possibilitará que eles se desloquem transparentemente na internet sem perda de conectividade para as aplicações das conexões já em curso.

1.3. Estruturação da Dissertação

Este trabalho é estruturado como segue: o Capítulo 2 apresenta os principais protocolos propostos para solucionar o problemas relacionados a mobilidade de redes nas diversas camadas da pilha protocolar TCP/IP em geral e aponta alguns trabalhos correlatos com a proposta deste projeto com diferentes metodologias para a mobilidade de redes acima da camada IP; o Capítulo 3 apresenta a arquitetura proposta da solução ILM, a qual integra vários dos conceitos apresentados no capítulo 2 e o deselvolvimento da solução mencionado; O capítulo 4 apresenta os resultados e testes da solução; O capítulo 5 apresenta a conslusão e o capítulo 6 apresenta as referências utilizadas até para a construção da dissertação.

Capítulo 2

Estado da Arte

Atualmente, quando um dispositivo conecta-se à internet, a atribuição de um endereço IP para este dispositivo, é completamente dependente da localização onde ele se encontra. Desta forma, os protocolos que tendem solucionar os problemas e gerenciar a mobilidade, buscam superar a dependência entre a identificação e localização através de mecanismos que traduzem os endereços para o encaminhamento de pacotes para qualquer dispositivo, sendo ele fixo ou móvel [Pereira, 2008].

O Gerenciamento da Mobilidade possibilita sua implementação nas diversas camadas do modelo protocolar TCP/IP. A discussão sobre em qual camada deve ser implementada a mobilidade ainda é uma discussão aberta na comunidade acadêmica, sendo ainda necessário muito estudo e pesquisas na área. Atualmente existem diversas propostas para o Gerenciamento de Mobilidade nestas camadas, nas próximas sessões serão apresentadas as principais arquiteturas existentes [Baptista, 2009].

2.1. Mobilidade abaixo da Camada IP

Abaixo da camada IP, nas camadas de enlace e camada física não existem possibilidades de roteamento IP, onde um host conectado a uma rede não consegue realizar a comunicação com outro host em outra rede através dessas duas camadas protocolares de forma isolada. Para este roteamento, é irrevogavelmente necessário ao menos um protocolo pertencente à camada IP. Por esta questão, mecanismos presentes somente na camada de enlace por si só não podem fornecer inteiramente a mobilidade. Entretanto, para as soluções de mobilidade presente nas camadas superiores, o suporte da camada de enlace se faz necessário para reconfigurar o host em sua nova sub-rede de forma dinâmica durante o seu deslocamento [Eddy, 2004].

Outra função da camada de enlace em relação à mobilidade implementada nas camadas superiores é detectar quais os tipos de ligações estão disponíveis ao alcance do host em uma determinada área como 802.11, Ethernet, 3G, Bluetooth etc. Assim, a camada de enlace deve ter a capacidade de detectar qual tecnologia está disponível em uma determinada área e acoplar-se à topologia local de forma transparente, permitindo ao utilizador usufruir da conectividade em todo o seu percurso de mobilidade. Também enquanto a camada de enlace realiza uma transição entre redes distintas, a camada de rede necessita de reconfigurações para que o IP seja adequado à rede que o novo dispositivo passou a acessar [Baptista, 2009].

2.2. Mobilidade na Camada IP

Na camada IP, foi onde as soluções do gerenciamento da mobilidade foram mais explorados. Isso se deve ao fato de que é nesta camada em que os endereços IPs são distribuídos. Nos tópicos seguintes, serão descritos as principais abordagens desta camada.

2.2.1. Mobile IP

Através da internet, os endereços IP da camada de rede são atribuídos administravelmente à sub-redes específicas na internet. Assim, de acordo com a presente estrutura de roteamento da internet a abordagem padrão de gerenciamento de mobilidade é o Mobile IP [Perkins 2002], baseado na versão IPv4 e projetado pela IETF (*Internet Engineering Task Force*). A solução desta abordagem, indica que os endereços IPs sejam utilizados como identificadores dos dispositivos apesar de serem dependentes da localização. Isso se deve ao fato de que o endereço atribuído a cada dispositivo é o endereço IP da rede de origem a qual ele se conectou e não é modificado mesmo que o dispositivo migre para uma sub-rede diferente. O Mobile IP se baseia no conceitos de *Home Agent* (HA) e *Foreign Agent* (FA) para o roteamento de pacotes.

Para um melhor entendimento do funcionamento do Mobile IP, uma breve descrição das entidades que o compõe, se faz necessário:

Home Network (HN): É a rede doméstica em que o endereço IP do nó móvel é definido.

Foreign Nerwork (FN): É a rede estrangeira visitada pelo nó móvel quando ele migra da sua HN.

Mobile Node (MN): Um host ou roteador que muda seu ponto de acesso de uma rede para outra.

Correspondent Node (CN): É um host com o qual o MN está se comunicando. Ele pode ser móvel ou fixo.

Home Agent (HA): É um roteador presente na HN que faz o tunelamento de datagramas para entregar ao MN quando ele está fora da sua HN, além de manter sua localização atual.

Foreign Agent (FA): É um roteador presente na FN que fornece serviços de roteamento para o MN, enquanto estiver registrado nesta rede. O FA, entrega os datagramas para o MN que foram tunelados pelo HA.

Home Address (HoA): É o endereço IP original da HN. Este endereço não altera durante toda a comunicação dos nós, mesmo após a mobilidade.

Care-of-Address (CoA): É um endereço IP com o qual o MN está registrado na rede estrangeira.

Tunelamento: É um caminho seguro entre o HA e o FA que garante a entrega segura dos pacotes de comunicação dos nós.

Os passos seguintes, fornecem um esboço do funcionamento do Mobile IP que também é apresentado na Figura 1.

Como suporte para a mobilidade, o Mobile IP requer um HA na rede doméstica a qual o MN pertence. Sempre que ocorrer a mobilidade, as atualizações de localização, são enviadas através do MN para o seu HA que atualiza sua tabela de endereços, mas as ligações em curso, continuam a utilizar o endereço de HoA. Assim, quando as mensagens chegarem ao HA com destino ao MN, ele passa a enviar todos os pacotes destinados a este MN através de um tunelamento IP para a nova localização física deste nó, o CoA na FN.O ponto de conexão deste nó não importa, pois o HA envia os pacotes de dados utilizando seu endereço IP como remetente, mantendo assim as conexões pré existentes. Já que os

endereços IPs não mudam e a gestão da localização está implícita, uma vez que o MN mantém seu endereço global, o HA consegue realizar o mapeamento através da tabela de endereços e saber a localização atual do nó móvel [Perkins, 2002].

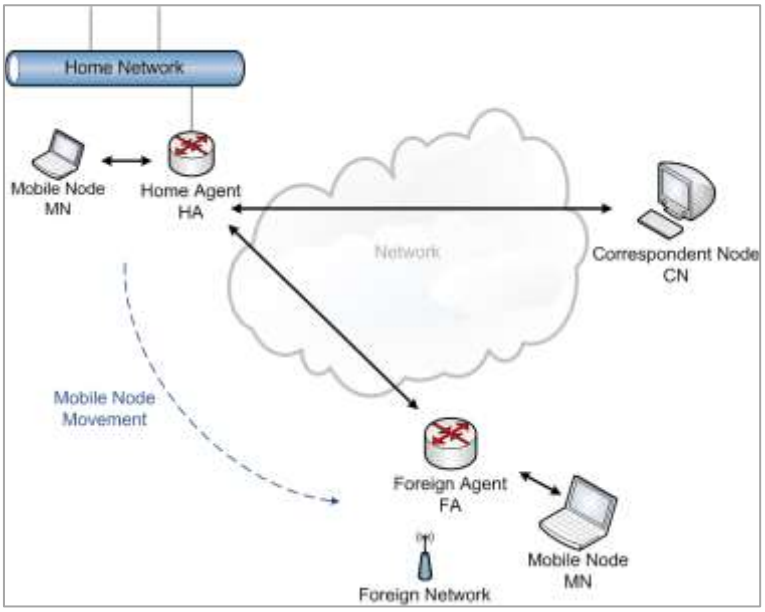


Figura 1 - Roteamento Mobile IP

Algumas desvantagens do Mobile IP é que a transparência total de mobilidade oferecida, nem sempre é o desejado, pois pode ser necessário realizar adaptações nas camadas superiores e a ciência desta mobilidade deverá ser explicitada [Baptista, 2009]. Também há necessidade de configurações em toda a rede (HA, FA, firewalls e roteadores) para a sua implantação. Outro ponto importante é que os usuários precisam de um endereço IP permanente, e para o IPv4, boa parte dos dispositivos atuais não possuem um endereço IP fixo, e sim adquirem via DHCP toda vez que se conectam à internet. Desta maneira, os conceitos do Mobile IP mesmo após serem amplamente discutidos, têm fortes indícios de que sua implantação é inviável em uma escala que deva oferecer alto nível de disponibilidade. Também vale ressaltar o fraco desempenho ocasionado pelo aumento do overhead ocasionado pelo encapsulamento, roteamento triangular e a grande espera nas atualizações dos registros [Martins, 2011].

2.2.2. MIPv6

O funcionamento do *Mobile Internet Protocol version 6* (MIPv6) [Johnson et al., 2004] se assemelha ao seu antecessor, o Mobile IP. Quando um nó móvel estiver fora de sua HN, uma mensagem é enviada para o HA, informando-o sobre a sua nova localização. E este, por sua vez, confere a autenticidade do seu emissor e atualizada sua tabela de endereços e cria-se um tunelamento IP entre o HA e o MN obtendo o mesmo roteamento triangular como pode ser visto na Figura 2.

Neste protocolo, como podemos perceber o MN e HA possuem as mesmas funcionalidades. Mas já não se faz necessário utilizar um FA, pois com o IPv6, a capacidade de endereçamento cresce e o gerenciamento destes endereços torna-se desnecessário, ao contrário do Mobile IP que utiliza o FA para gerenciar um mesmo endereço CoA utilizado em vários dispositivos móveis. Outra diferença entre os dois protocolos é que devido a autoconfiguração do IPv6, os registros de um novo IP (CoA) torna-se mais simples e rápido ocasionando uma maior agilidade do MIPv6.

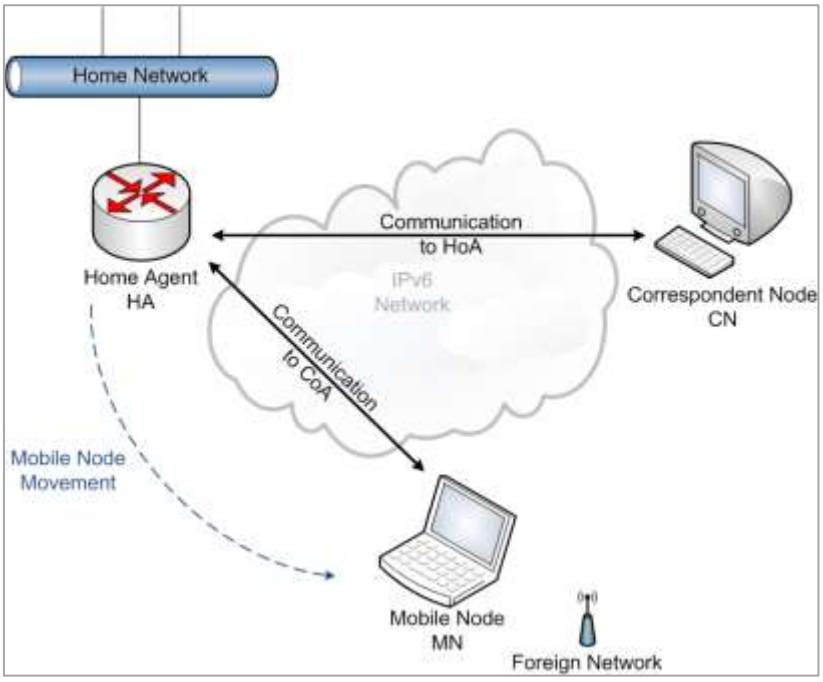


Figura 2 - Roteamento Inicial: Triangular

Como anteriormente mencionado, quando o MN setá fora de sua rede ele informa o HA sobre sua transição e o seu novo endereço, através de um Binding Update (BU), e então o HA envia uma confirmação de registro conhecida como Binding Acknowledgement (BA). E inicia-se a comunicação triangular. Entretanto, esta comunicação é muito ineficiente, pois os fluxos primeiramente devem passar pelo HA e depois serem encaminhados para o CoA do MN. Caso haja diversos nós registrados no HA, pode ocorrer congestionamentos dificultando a entrega correta dos pacotes.

Para solucionar o problema do roteamento triangular, o MIPv6 possui mecanismos de otimização de rotas, chamado de *Return Routability procedure* (RR) que pode ser visto na Figura 3. Este método é utilizado após o roteamento triangular, onde o MN envia uma mensagem para o CN em dois caminhos diferentes. Primeiro a mensagem *Home Test Init* (HoTI) é enviada através do tínel para o HA que encaminha em seguida para o CN. A segunda mensagem é enviada diretamente para o CN através da mensagem *Care of Test Init* (CoTI). Quando o CN recebe as mensagens, ele retorna a confirmação também por dois caminhos diferentes, o *Home Test* (HoT) passando pelo HA e o *Care of Test* (CoT) enviado diretamente ao MN. Por fim, para assegurar a legitimidade do CN, o MN envia um BU e após a resposta do CN por BA, a comunicação direta entre os nós é estabelecida e um tunelamento pode ser criado a fim de aumentar a segurança da comunicação [Johnson et al., 2004].

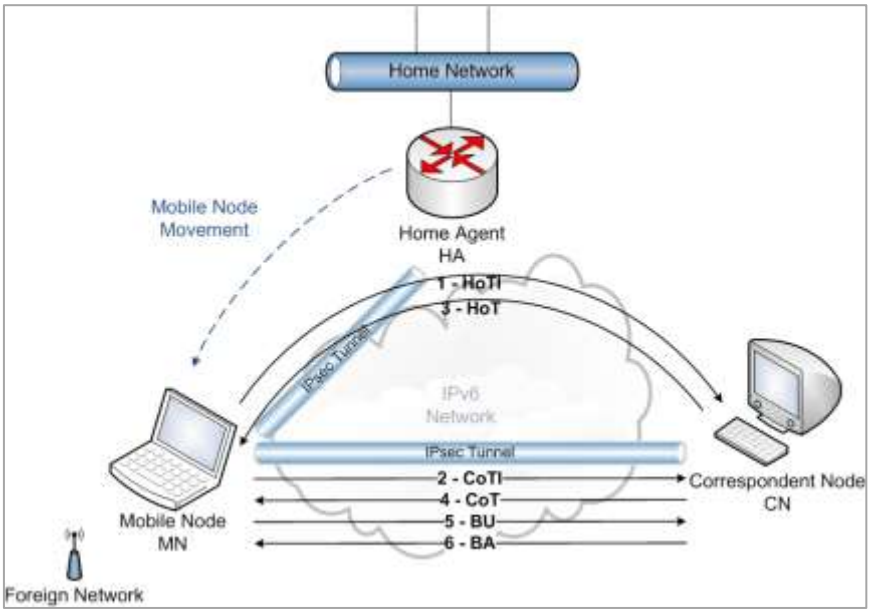


Figura 3 - Fluxo do Return Routability Procedure

Algumas desvantagens deste protocolo, também se deve ao mau desempenho, semelhante ao apresentado pelo Mobile IP, como o aumento do overhead causado pelo encapsulamento, roteamento triangular inicial e espera nos registros de atualização [Martins, 2011].

2.2.3. NEMO

Outro protocolo para suporte à mobilidade, desenvolvido para atuar na camada IP é o NEMO (*Network Mobility*) [Devarapalli, 2005], baseado no MIPv6 e também desenvolvido pela IETF. Neste protocolo, redes móveis podem se locomover entre diversos pontos da internet e as comunicações em curso permanecem ativas.

No NEMO, as funcionalidades tidas pelo MN em outros protocolos, são transferidas para o *Mobile Router* (MR), ou seja, o tratamento das locomoções do o ponto de interconexão com a internet são tratadas somente no MR sem qualquer alterações para os MNs.

O funcionamento deste protocolo pode ser visto na Figura 4.

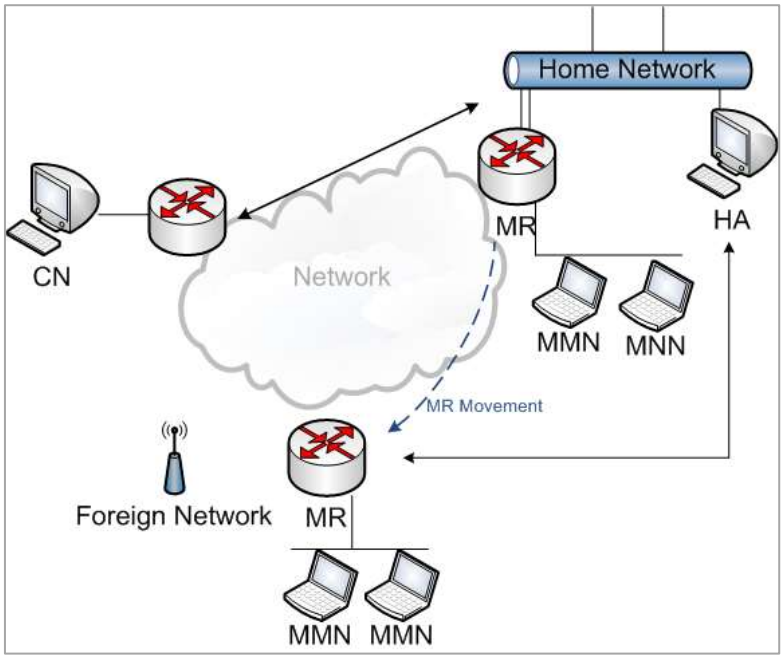


Figura 4 - Roteamento NEMO

Inicialmente, quando o MR migra para uma nova rede, ele conecta-se a um novo roteador de acesso e adquire um novo endereço denominado CoA. Então este endereço é enviado através de uma mensagem de atualização BU para o HA de sua rede de origem. Após o recebimento do BU, o HA, associa o endereço HoA do MR com o CoA e envia uma mensagem de confirmação BA para o MR validando o processo. Uma vez finalizado o processo de atualização e confirmação entre o MR e HA, um túnel bidirecional entre eles é criado. À partir deste momento, toda vez que um CN enviar uma mensagem para um MN desta rede móvel, estas são enviadas para o HA que mapeia o endereço real do MR, encapsula-as e as envia através do túnel já criado. Ao chegarem ao MR, essas mensagens são desencapsuladas e repassadas à rede móvel e entregues ao MN de destino. O mesmo processo ocorre nas trocas de mensagens com origem ao MN para o CN [Devarapalli, 2005], [Pereira, 2008].

2.2.4. LISP

O LISP (Locator/Identifier Separation Protocol) [Farinacci et al, 2012], é um protocolo que para prover o gerenciamento da mobilidade, separa o endereçamento IP entre identificadores chamado de *Endpoint ID* (EID) e localizadores chamados de *Routing Locator* (RLOC). O EID é o endereço original de um nó quando este conecta-se à internet e o RLOC, nada mais é que um endereço IP de um roteador na borda da rede [Loureiro et al., 2012].

O funcionamento do LISP se baseia no encapsulamento de pacotes de entre dois EIDs através dos roteadores de borda *Ingress Tunnel Router* (ITR) e *Egress Tunnel Router* (ETR). A função do roteador ITR, é receber os pacotes dos EIDs, encapsulá-los e encaminhá-los para o destino e a função do ETR é receber estes pacotes, desencapsula-os e entregá-los à EID adequada.

Desta maneira, supõe-se que o domínio do destino já tenha sido determinado pelo router de borda ITR e um mapeamento na rede é realizado a fim de determinar o caminho de roteamento até o ETR. Assim, as mensagens enviadas do EID ao destinatário, são encapsuladas pelo ITR com um novo cabeçalho, com o endereço RLOC do destinatário no

campo de destino. Logo que as mensagens chegarem ao ETR, elas serão desencapsuladas e roteadas para o endereço EID do destino. O processo de comunicação do LISP descrito, pode ser visto na Figura 5 [Farinacci et al, 2012].

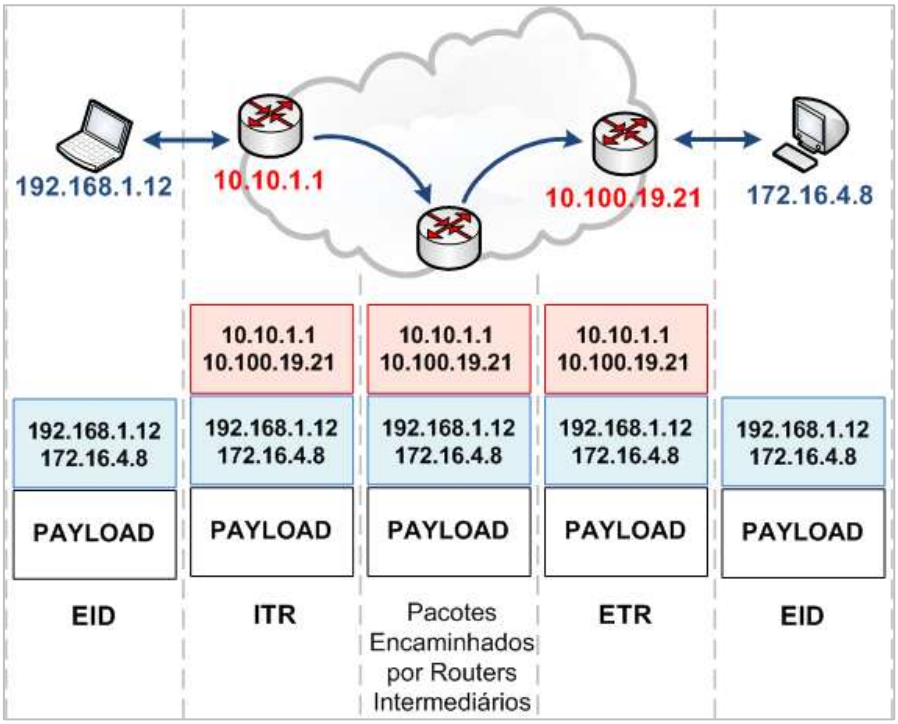


Figura 5 - Roteamento de Mobilidade LISP

O LISP é tido como um dos protocolos promissores para o gerenciamento da mobilidade, entretanto as suas desvantagens fazem com que a comunidade acadêmica continue na busca para melhorar seu desempenho e também por novas propostas. No LISP, com o encapsulamento e desencapsulamento duplo constante de pacotes, problemas com o tamanho do pacote *Maximum Transfer Unit* (MTU) podem ocorrer, pois como os pacotes são encapsulados com novos cabeçalhos e o MTU definidos pelos roteadores pode ser extrapolado. Problemas de aumento de overhead, latência nas pesquisas do EID e RLOC podem ocasionar atrasos, que podem ser letais para aplicações em tempo real [Martins, 2011].

2.3. Mobilidade entre a Camada IP e a Camada de Transporte

2.3.1. HIP

O protocolo de gerenciamento de mobilidade *Host Identity Protocol* (HIP) [Moskotlvitz & Nikander, 2006], também utiliza o mecanismo de separar do endereço IP, o papel de localizador e identificador. Ele introduz uma nova camada entre a camada de transporte e a camada IP da pilha protocolar TCP/IP que é um nó identificador que faz o uso de criptografia de chaves públicas.

Figura 6 - Pilha Protocolar HIP

No HIP, é atribuído para cada nó, um nome globalmente único (namespace) que é baseado em criptografia de chaves públicas, desta maneira este namespace passa a ser utilizado também para autenticar as comunicações entre os nós.

Para desacoplar o identificador e localizador do endereço IP, o protocolo HIP faz o uso do namespace que é utilizado como identificador que atualiza as associações HIP entre estes nós através das trocas de mensagens [Baptista, 2009]. Já o endereço IP é tido como o localizador, é gerido somente na camada de rede sem alterações da camada HIP, assim elimina-se a utilização do endereço IP nas camadas de transporte e aplicação. Neste protocolo, um servidor de localização independente *Rendezvous Server* (RVS), que tem o papel semelhante ao HA do protocolo *Mobile IP* é utilizado[Moskotlvitz & Nikander, 2006].

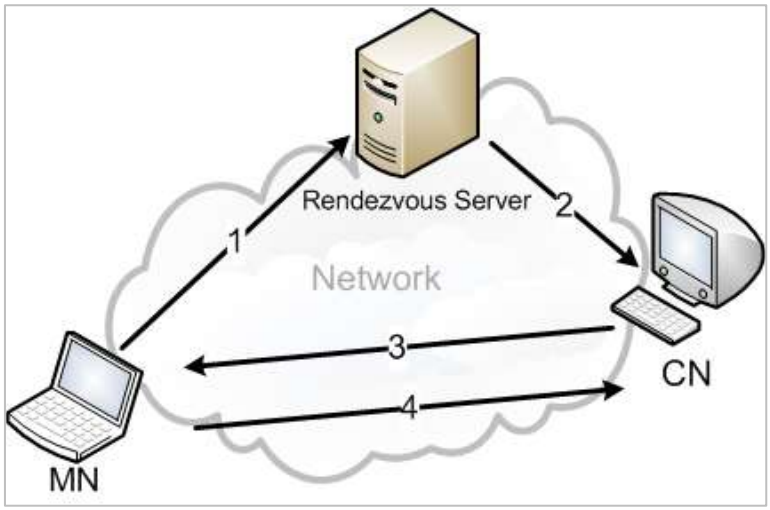


Figura 7 - Roteamento HIP

No funcionamento do HIP, visto na Figura 7, quando o MN deseja comunicar-se com o CN, ele realiza uma consulta ao DNS solicitando o namespace do nó e a sua chave pública. Ao receber essas informações, o MN envia uma mensagem para iniciar a comunicação através do RVS que por sua vez valida o namespace e encaminha a mensagem para o CN. Após a mensagem que possui o endereço IP e o namespace do MN chegar ao CN, a comunicação entre os nós será estabelecida e passará ser direta sem a intermediação do RVS. Quando o MN migrar para uma nova rede, ele receberá um novo endereço IP que será enviado para o RVS e para o CN e assim as mensagens das ligações em curso passam a ser enviadas para este novo endereço sem qualquer interrupção.

Esta solução oferece mobilidade de redes transparente para as camadas superiores e o gerenciamento entre múltiplos domínios de endereçamento IP. Entretanto a inserção de uma nova camada na pilha protocolar TCP/IP e adaptações nos terminais de comunicação têm um grande impacto na infraestrutura que limitam a implantação desta solução no ambiente de produção. Questões relacionadas ao má desempenho, como o aumento do overhead ocasionado na camada do HIP e a perda de pacotes quando dois terminais se movem ao mesmo tempo também precisam ser mitigados [Baptista, 2009] e [Martins, 2011].

2.4. Mobilidade na Camada de Transporte

2.4.1. mSCTP

Para que seja possível desenvolver uma solução de gerenciamento de mobilidade na camada de transporte é necessário ter uma forma de vincular dinamicamente os endereços de uma conexão quando ocorrer a mobilidade dos dispositivos. Além disso, várias modificações devem ser realizadas nesta camada.

Um protocolo, chamado *Stream Control Transport Protocol* (SCTP) [Stewart et al., 2000] proposto com o objetivo de viabilizar a confiabilidade de sistemas de sinalização para telefonia voz sobre IP (VoIP) tem sido visto como um forte candidato para ser estendido para o gerenciamento da mobilidade. O SCTP faz uso de diversas características de um protocolo bastante estabelecido, o TCP, fazendo a utilização das conexões confiáveis ponto a ponto orientadas à conexões IP [Baptista, 2009]. A nova funcionalidade deste protocolo é que através dele, é possível utilizar diversas interfaces de rede em uma mesma conexão identificadas por diferentes endereçamentos IPs. E as interfaces neste protocolo são distribuídas entre primária e secundária.

Quando um dispositivo móvel deseja se conectar-se a um outro dispositivo, o SCTP envia para o endereço da interface primária previamente configurado na associação (comunicação SCTP entre os dispositivos) e caso este endereço esteja inalcançável, é então realizado o envio para o endereço da interface secundária como pode ser visto na Figura 8.

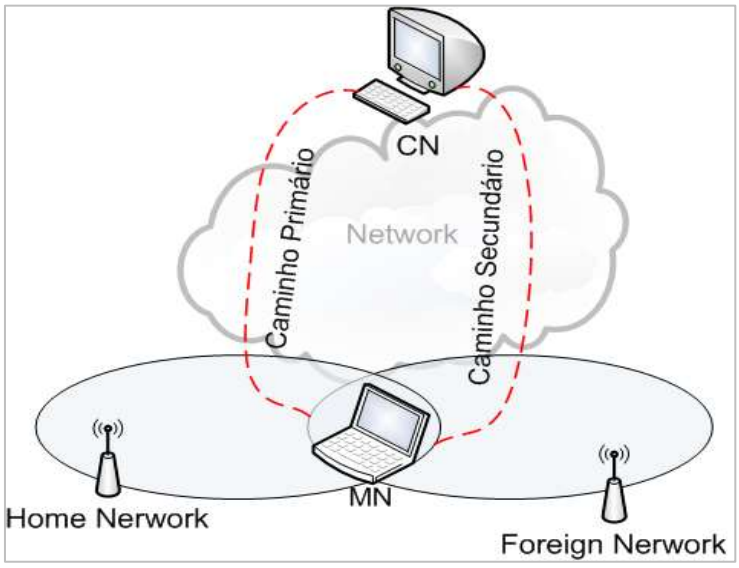


Figura 8 - Associação SCTP com Caminho Primário e Secundário

Os endereços IPs que pertencem à associação, devem ser trocados entre os dispositivos antes de estabelecer uma comunicação e devem ser mantidos até o término da

comunicação. Mas quando ocorrer a mobilidade, este protocolo por si só não é capaz de manter as conexões em curso, desta forma, uma extensão do SCTP chamada de *Mobile SCTP* (mSCTP) proposta por [Riegel & Tuexen, 2003] permite que os endereços de uma associação possam sem alterados ou removidos durante a comunicação.

A maioria das poucas soluções propostas para o gerenciamento da mobilidade na camada de transporte até o momento, normalmente exigem alterações também na camada de aplicação, como por exemplo, para gerenciar a localização. Entretanto, as soluções nesta camada, normalmente apresentam menos problemas em relação à segurança da rede [Eddy, 2004].

2.5. Mobilidade na Camada de Aplicação

2.5.1. SIP

As soluções de gerenciamento na camada de aplicação, possuem vantagens semelhantes aos apresentados na camada de transporte mas sem as alterações desta camada. Entretanto, as soluções presentes puramente na camada de aplicação não conseguem manter as conexões ativas durante a mobilidade de um dispositivo, pois essas soluções não têm acesso aos módulos de conexão abertos dentro da camada de transporte [Eddy, 2004]. Porém os principais protocolos e aplicações que utilizam a internet, se comunicam utilizando o protocolo TCP e assim beneficiam-se de implementações simples na camada de aplicação, como por exemplo os serviços de navegação web, correio eletrônico e etc que se baseiam pequenas transações [Baptista, 2009].

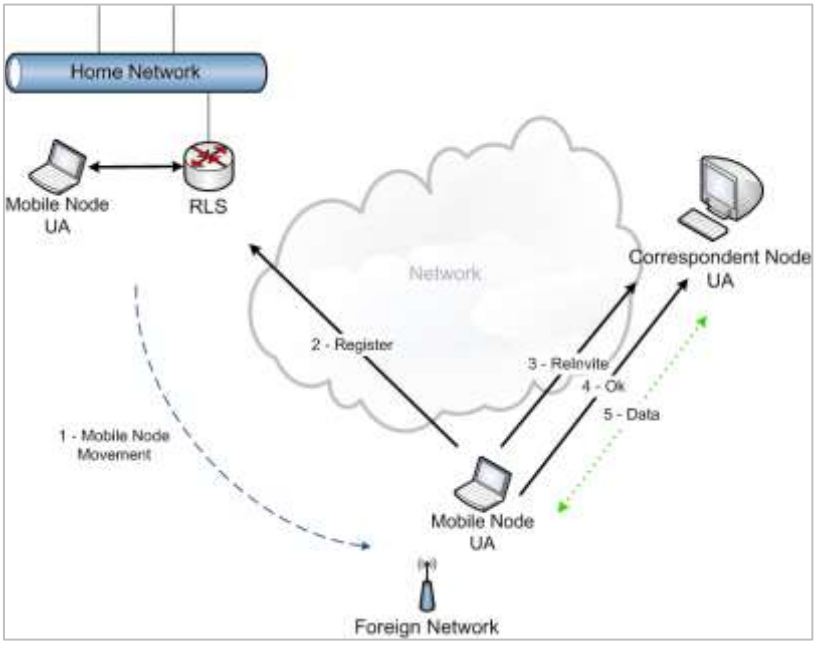


Figura 9 - Roteamento SIP

A solução para o gerenciamento da mobilidade nesta camada, utilizando o protocolo *Session Initiation Protocol* (SIP) [Rosenberg et al., 2002] foi proposta por [Wedlund & Schulzrinne, 1999]. Neste protocolo a mobilidade é assistida pelo fato dos *User Agents* (usuários) realizarem registros de seus endereços independentemente da sua localização em um servidor de registro chamado *Registrars* e *Location Services* (RLS). Em seu funcionamento visto na Figura 9, inicialmente, o *User Agent* (UA) está conectado em sua rede doméstica

contendo o RLS doméstico. Toda a vez que ele realizar uma transição para uma nova rede, o UA envia mensagens de atualização para o RLS que age de forma semelhante ao Home Agent presente no *Mobile IP*, mas realizando o mapeamento dos usuários e não dos dispositivos.

Sempre que um UA de um nó correspondente se comunica com um UA, uma mensagem INVITE é enviada e o RLS que contém o endereço real deste UA reencaminha a mensagem para o destino. Quando ocorre a mobilidade do UA com alguma ligação em curso, uma mensagem RE-INVITE é enviada do UA para o UA correspondente informando seu novo endereço IP. Por sua vez, o UA correspondente deve detectar esta mensagem como uma atualização de endereços e não como uma mensagem para o início de uma nova ligação. Após a correta atualização de endereços, a comunicação continua sem interrupções [Baptista, 2009].

2.6. AKARI

Diversos protocolos foram apresentados nos tópicos anteriores para solucionar o problema da mobilidade de redes. Entretanto, nenhum deles foi capaz de proporcionar uma solução que viabilizasse sua produção em grande escala. Desta maneira, estudos e novas propostas continuamente vem surgindo a fim de suprir esta necessidade.

Neste item, será apresentado uma proposta similar a desta tese que utiliza como base os protocolos apresentados anteriormente. A solução para o gerenciamento de mobilidade aqui apresentado, o AKARI, se baseia no pressuposto da separação do endereço IP entre identificadores e localizadores.

A solução que permite a mobilidade de redes apresentada na arquitetura Akari [Harai et al., 2008], utiliza entidades distintas para identificar e localizar os nós em uma rede. Entretanto, a solução apresentada nesta arquitetura difere das mencionadas anteriormente por permitir a sua aplicação em arquiteturas pós IP ou mesmo não-IP.

A arquitetura Akari, é um projeto japonês iniciado no ano de 2006 com a participação do governo, universidades e setor privado, tendo como objetivo desenvolver uma rede de nova geração até o ano de 2015. Seu princípio básico é criar uma arquitetura onde a camada de IP deve ter a mínima funcionalidade possível e tornar a internet em um ambiente livre para qualquer evolução e desenvolvimento, tendo a capacidade de suprir toda a demanda da sociedade entre 50 a 100 anos [Harai et al., 2008] ,[Martins, 2011]. No ano de 2011, na 7º Conferência Internacional on IP + Optimal Network (iPOP) em Kwasaki, no Japão, já foi realizado uma demonstração da prova de conceito do Akari.

Nesta solução, um nó pode ser identificado pelo nome de forma global ou local. A identificação local é formada por uma combinação de palavras sobre as principais características do nó, como número de série, contexto série, etc e são únicos na rede local, normalmente utilizados para identificar os nós e para a manutenção da rede. A identificação por nomes de forma global é formada pela combinação do nome local e do nome do Servidor de Gerenciamento de Identificação (IMS). Este servidor é utilizado para suportar a mobilidade e prover a segurança no acesso à rede. Também, um nó pode ser identificado por um identificador (ID), que é inserido no cabeçalho dos pacotes.

Para que o desacoplamento do endereço IP seja possível e o mapeamento entre o identificador e localizador funcione, uma nova camada chamada de Identidade foi inserida entre a camada de transporte e a camada de IP da pilha protocolar TCP/IP como é visto na Figura 10.

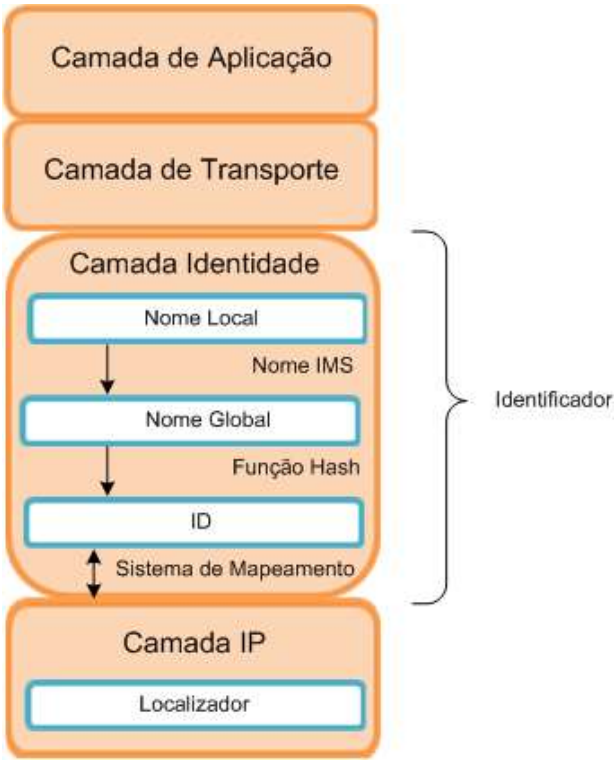


Figura 10 - Pilha Protocolar Akari

No Akari, a comunicação se dá da seguinte forma: inicialmente, os dados vindos da camada de aplicação chegam à camada de transporte por identificadores chamados de primários de fonte e destino. Na camada de transporte o cabeçalho de transporte é adicionado e os dados são encaminhados para a camada de identidade utilizando uma outra interface identificada pelo identificador primário. Quando os dados chegarem à camada de identidade, é feito um mapeamento do identificador primário para um identificador ativo que ao fim será inserido no cabeçalho dos dados. Antes dos dados serem enviados para a camada IP, um novo mapeamento entre o identificador ativo e o localizador é realizado. Em seguida, na camada de IP, os localizadores de fonte e destino são adicionados ao cabeçalho do pacote de dados e finalmente enviado ao seu destino [Harai et al., 2008] ,[Martins, 2011].

Nesta arquitetura, os identificadores ativos e localizadores inseridos pela camada de identidade e pela camada IP respectivamente são visíveis em toda a rede, sendo que em alguns serviços o identificador ativo continua inalterado enquanto o localizador é modificado de forma dinâmica dando o devido suporte à mobilidade de redes.

Para esta solução, o aumento do overhead na camada de identidade tende a ser significativo [Martins, 2011].

2.7. Análise Comparativa das Soluções

Compete neste ponto analisar as soluções supracitados a fim de verificar as diferenças e semelhanças e as principais contribuições entre as arquiteturas para a mobilidade de redes, além de explanar as diferenças destas soluções com a proposta desta tese a ser apressentada no próximo capítulo. A Tabela 1 apresenta o resumo das comparações destas soluções.

	Flexibilidade de Roteamento	**Segurança**	**Problemas e Desempenho**	**Perda de Pacotes**
Mobile IP	Somente IP	IPSec	Aumento de overhead pelo encapsulamento, roteamento triangular, tempo elevado nos registros de atualização	Devido a demora no processamento de registro de atualização
MIPv6	Somente IP	IPSec	Aumento de overhead pelo encapsulamento, aumento no atraso da entrega de pacotes, demora na atualização do handover	Devido à sobrecarga da rede e latência entre os handovers
NEMO	Somente IP	IPSec	Limitações de encaminhamento, problemas com otimização de rotas e delegação de prefixos	Quando ocorre o handover
LISP	Somente IP	Procssos de controle para a obtenção do mapeamento EID-to-RLOC	Aumento de overhead, latência nas pesquisas do EID e RLOC e atrasos	Quando ocorre atraso de mapeamento
HIP	IP ou não IP	Baseado em Criptografia de chaves Públicas	Ataque de negação de serviço. Aumento de overhead na camada do HIP	Quando ocorre o double jump.
mSCTP	Somente IP	Troca de chave especial, cookie entres os nós comunicantes	Não apto a realizar handover com tráfego de tempo real, implementação imatura, não possui gerenciador de localização	Quando ocorre o double jump.
SIP	Somente IP	Confidencialidade e integridade das mensagens através de cifras	Demora para a migração do handover, deturpando o desempenho de aplicações em tempo real, possui poucas oportunidades de otimização de tempo	Depende da velocidade em que os outros protocolos que criam e reestabelecem a conexão
AKARI	Flexível. Rotemaneto independente da arquitetura TCP/IP	Baseado em Criptografias de chave Públicas e função hash	Aumento do overhead na camada identidade	_____

Tabela 1 - Algumas Comparações das Soluções de Gerenciamento de Mobilidade

A escolha da arquitetura é o princípio para se criar uma solução que tem o intuito de trabalhar ao nível das camadas de rede, considerando que o roteamento e questões de segurança dependem da mandeira na qual a arquitetura foi concebida.

O Mobile IP, MIPv6, NEMO, SIP, mSCTP e o LISP utilizam a atual arquitetura de nomes, onde todos utilizam dois nomes hierárquicos (conhecido também como endereço de internet) para dar suporte ao então "descoplamento" do endereço IP entre identificadores e localizadores, sendo esta, uma abordagem inflexível pela a utilização do endereço IP. Ao contrário do HIP que utiliza um nome hierárquico como única identificação do *host* e o endereço IP como o localizador deste *host*. No AKARI, os identificadores de *hosts* utilizam parcialmente os nomes para critérios de hierarquia e também para questões de segurança.

As abordagens inflexíveis (Mobile IP, MIPv6, NEMO, SIP, mSCTP e o LISP) ou mesmo o HIP que utiliza do método de identificação baseado por criptografias de chaves públicas, possuem a vantagem das aplicações já criadas para a Internet atual poderem continuar sendo utilizados sem qualquer necessidade de alterações. Porém, são impedidas de serem utilizadas em em arquiteturas não-IP ou Pós IP como o AKARI [Martins, 2011].

O protocolo *Mobile* IP, um dos pioneiros apresentados como solução para a mobilidade de redes foi a base para a construção de outros procolocos, como o MIPv6 e o LISP. Nele, há uma vantagem primordial para uma solução móvel, a transparência para o usuário, ou seja, mantém as conexões após o handover de forma transparente para as camadas superiores. Entretanto, suas desvantagens fizeram com que este protocolo se tornasse inviável para a mobilidade, desvantagens como o roteamento triangular, excesso de encapsulamento, e um longo tempo no registro de atualização da sua nova posição no *Home Agent*. Já o MIPv6, uma variante do *Mobile* IP, possui praticamente as mesmas desvantagens mencionadas, entretanto o roteamento triangular deste protocolo é tratado.

O LISP, tem como vantagem a sua implementação que pode ser realizada e testada de forma gradativa. Outra fator relevante é que ele utiliza o método de separação da localização e identificação do *host*. Entretanto, esta separação resulta em um processo maior de controle do mapeamento do EID-*t*o-RLOC, o que muitas vezes o torna demorado para aplicações de mobilidade de redes (latência do mapeamento), e o excesso de encapsulamento destes dispositivos de borda aumentam o *overhead* do protocolo.

A arquitetura do protocolo NEMO, que é uma variante do MIPv6, possui praticamente suas mesmas vantagens, entretanto encontra dificuldades na prática da sua implementação devido aos problemas constatados no encaminhamento e otimização de rotas das redes em movimento.

O HIP atenua os problemas relacionados ao DNS, pois faz a utilização do servidor *Rendezvous*, entretanto a modificação necessária na pilha protocolar dos dispositivos torna complexo a sua implementação. Também está propício a ataques de negação de serviço, e de aumento de overhead na camada HIP.

O mSCTP tem como principal vantagem o suporte nativo tanto para IPv4 quanto para a arquitetura IPv6, além da grande quantidade de informações para tomadas de decisões no *handoff* por se tratar também de um protocolo de sinalzação. Entretanto, as implemetações do SCTP ainda são recentes, predispondo interpretações erradas e implementações complexas. Outro fator agravante deste protocolo é a falta de um gerenciador de mobilidade, impendindo a comunicação em caso de *double jump*. Já o protocolo SIP tem a vantagem e as facilidades presentes na camada de aplicação, tendo a expressiva estrutura da Intenet facilitando o gerenciamento das sessões sendo utilizado como protocolo de sinalização [Pereira, 2009]. Seu gerenciamento de mobilidade torna-se

nativo devido a sua estrutura, fazendo deste protocolo o mais propício para aplicações em tempo real. Mas vale ressaltar que o SIP realiza *handover* com perdas de pacotes ou um *delay* considerável. Também não é considerado na utilização com *handover* horizontal (entre células da mesma tecnologia).

Capítulo 3

ILM: Identifier and Locator Mapping

Para que a persistência das comunicações seja mantida, os mecanismos que dão suporte a mobilidade de redes devem ser capazes de detectar o deslocamento dos terminais móveis e contornar os problemas causados por este deslocamento, de forma que o usuário possa mover-se por diversas redes e subredes e não dar-se conta da mobilidade.

Tratando-se do desenvolvimento desses mecanismos, a decisão de determinada camada da pilha protocolar TCP/IP faz com que a implementação seja conduzida por diferentes caminhos. Visto com mais detalhes no capítulo 2, as soluções para a mobilidade nas camadas inferiores, como a de enlace por si só não é capaz de realizar roteamento IP, entretanto as soluções de mobilidade centralizadas requerem os redirecionadores desta camada para prover o uma mobilidade transparente. Já nas camadas superiores, a transporte e aplicação as ligações e a mobilidade são realizadas sem a interferência do proxy por ter uma abordagem fim-a-fim.

Na mobilidade, casos de *handover* suaves ou chamados de *smooth handovers* (quando há dois caminhos ativos e dois fluxos correspondentes, pelo menos por um tempo mínimo) faz com que a camada de transporte seja a mais apropriada para sustentar a mobilidade independente da infraestrutura e origem/destino da comunicação [Eddy, 2004].

3.1. Arquitetura ILM

A proposta para o gerenciamento de mobilidade de redes desenvolvida, denominada *Identifier and Locator Mapping* (ILM), é uma solução tida com base nas propostas de [Farinacci et al, 2012] e [Moskotlvitz & Nikander, 2006] por ser baseada em protocolos híbridos, onde é sugerido a separação do IP entre identificadores e localizadores e por atuar na camada de transporte, o que mantém a lógica e o fluxo da comunicação de dados na pilha protocolar TCP/IP, tornando a comunicação transparente para a camada de aplicação, como pode ser visto na Figura 11.

O ILM é uma proposta simples, que não faz a utilização de diversos intermediadores como visto no LISP, que a todo o momento realiza o mapeamento de todas as ligações através dos roteadores de borda, utilizando um endereço IP virtual para as suas comunicações. O ILM propõe uma solução de comunicação fim-a-fim com o gerenciamento de mobilidade sendo tratado diretamente entre os terminais móveis sem a intervenção de outros dispositivos no caso de *single jump* (deslocamento de um só terminal), porém, na ocorrência do *double jump* (deslocamento de ambos terminais), um terceiro dispositivo se faz necessário para gerenciar a mobilidade, se igualando as propostas vistas anteriormente com suas respectivas desvantagens. Porém, considerando que a taxa de ocorrência do *double jump* é mínima, a utilização de um terceiro dispositivo neste caso não altera o desempenho da solução proposta. O ILM visa amenizar problemas de latência nos *handovers* e o mau desempenho através de uma solução minimalista com o mínimo de recursos e infraestrutura possíveis.

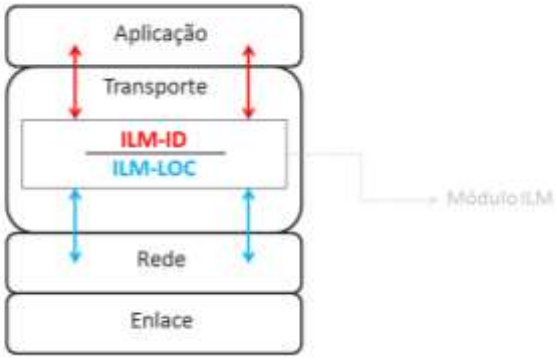


Figura 11 - Pilha Protocolar TCP/IP com ILM

Para obter o gerenciamento de mobilidade desejado, o cenário do ILM é constituído pelos seguintes agentes e identificadores:

Nó Móvel (ILM-MN – *Mobile Node*): altera seu ponto de conexão entre uma rede a outra sem alterar o endereço IP de origem. Uma vez possuindo conectividade em um ponto da rede, ele poderá continuar sua comunicação com diversos nós da internet, seja qual for sua localização.

Nó correspondente (ILM-CN - *Correspondent Node*): é o dispositivo com o qual um nó móvel se comunica. O ILM-CN pode ser tanto fixo quanto móvel. Para ele, a mobilidade do ILM-MN é transparente, pois ele sempre referencia o nó móvel através de seu endereço de origem, independente da localização deste nó.

Identificador de Origem (ILM-IDs - *Source Identificator*): é o endereço IP original atribuído ao ILM-MN e ao ILM-CN quando estes se conectam à rede representando o endereço da sua rede de origem *home adress* (HA).

Localizador de Origem (ILM-LOCs - *Source Locator*): quando um ILM-MN ou ILM-CN realizar um deslocamento e conectar-se à uma nova rede, este receberá um novo endereço IP, chamado de endereço de localização.

Identificador de Destino (ILM-IDd - *Destination Identificator*): quando o ILM-MN estabelecer uma comunicação com o ILM-CN, o módulo ILM de ambos dispositivos preencherá o campo IDd da tabela de atualização, com o IP fixo do nó destino, denominado identificador de destino.

Localizador de Destino (ILM-LOCd - *Destination Locator*): quando o ILM-MN ou o ILM-CN realizar um movimento e conectar-se à uma nova rede, o ILM atribuirá ao campo ILM-LOCd o novo endereço IP do nó destino, chamado localizador de destino.

Servidor de Registro (*ILM-Server*): quando houver o deslocamento dos dispositivos ILM-MN e ILM-CN simultaneamente, os nós poderão buscar as informações da nova localização do nó destino neste servidor.

Tabela de atualização (ILM-UT - *Update Table*): tabela com as informações dos localizadores e identificadores dos nós em comunicação que é atualizada a cada deslocamento.

A solução desenvolvida para o gerenciamento da mobilidade foi baseado em IO:: Module Base e na subclasse IO::Socket::INET [Barr, 1997] e em *Iptables* [Russell et. al, 1999] para dar

suporte a mobilidade em conexões fim a fim. E para que fosse possível obter o gerenciamento de mobilidade desejado, a estratégia de desenvolvimento da solução constituiu nas seguintes etapas:

i) Criação de um *Daemon* Servidor: Solução em *background* disponível nos terminais móveis sempre em escuta de possíveis ligações clientes a fim de atualizar o endereço IP do dispositivo correspondente.
ii) Identificação do Terminal Móvel: Provê a identificação do mapeamento IP possibilitando detectar a mobilidade de redes.
iii) Serviço de Atualização: Provê a atualização dos endereços IPs das ligações em curso nos terminais móveis.

A solução implementada indica um baixo custo de implantação, por não requerer modificações das funções e métodos presentes no módulo *socket* juntamente com o iptables, com a vantagem de não sobrecarregar a solução com a criação de um novo protocolo.

3.2. Modelo de Comunicação ILM

O ILM realiza um mapeamento entre os identificadores estáveis e os localizadores dinâmicos retirando a necessidade de um um servidor para mapeamento global em caso de deslocamento *single jump*. Sabendo que após o deslocamento de um nó para uma nova rede, todas as aplicações da ligação em curso são desconectadas pelo kernel do sistema, o ILM propõe o seguinte modelo de comunicação que pode ser visto na Figura 12.

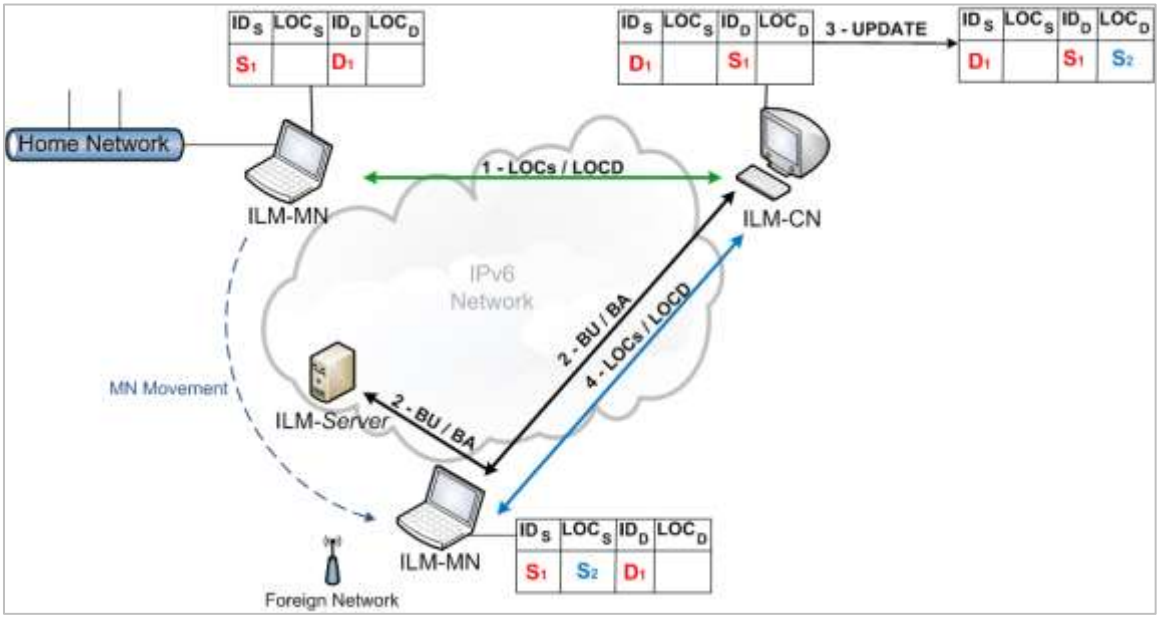


Figura 12 - Roteamento ILM com *single jump*

(1) O *Mobile Node* conecta-se a rede, denominada *Home Netwok* (HN) e recebe o endereço IP original desta rede localizado na tabela *Mobile Node* como S1.

(2) O mesmo ocorre com o *Correspondent Node*, este, recebe um endereço IP da sua HN, localizado na tabela do *Correspondent Node* como D1.

(3) Após o início da comunicação entre o *Mobile Node* e *Correspondent Node*, a tabela de atualização ILM-UT é preenchida no módulo ILM presente de ambos dispositivos.

Lembrando que, no primeiro momento sem mobilidade dos terminais, não existe endereços localizadores.

(4) Num segundo momento, o *Mobile Node* migra para uma nova rede, denominada Foreign Network (FN) e lhe é atribuído um novo endereço IP.

(5) Imediatamente o *Mobile Node* envia um *Bind Uptade* (BU) via *sockets* para o *Correspondent Node* e ILM-*Server* com o seu novo endereço IP e estes, retornarão com um *Bind Acknowledge* (BA). O envio do BU para o ILM-*Server*, garante que, em caso de deslocamento *double jump*, os nós poderão recorrer ao ILM-*Server* para obter a nova localização do nó.

(6) O *Mobile Node* e ILM-CN insere na tabela ILM-UT nos campos *Source Locator* e *Destination Locator* respectivamente o novo endereço IP, localizado na tabela do *Mobile Node* como S2.

(7) À partir daí, quando os pacotes vierem do *Mobile Node* para o *Correspondent Node* com um novo endereço IP, o módulo ILM presente na camada de transporte realizará uma alteração no cabeçalho destes pacotes através do tradutor de endereço NAT, alterando o endereço de origem para o IP original do HN do *Mobile Node* consultando a ILM-UT antes de enviar para a camada de aplicação.

(8) Quando os pacotes vierem da camada de aplicação do *Correspondent Node* para *Mobile Node*, estes terão o endereço original do *Mobile Node* como destino, logo o ILM interceptará estes pacotes e realizará a alteração do destino para o novo endereço IP do *Mobile Node* mediante consulta da tabela ILM-UT.

O processo descrito com o deslocamento *single jump*, ocorre de forma bidirecional, ou seja, permite a mobilidade também do *Correspondent Node*. E quando houver o deslocamento *double jump* os nós realizarão o BU e não obterão o BA. Desta forma, após um *time out*, os nós recorrerão ao *ILM-Server* para obter o novo endereço IP do nó correspondente e assim, ambos atualizarão a tabela ILM-UT e o processo continuará igualmente como descrito no *single jump*.

A Figura 13, apresenta o diagrama de sequência do modelo de comunicação do ILM.

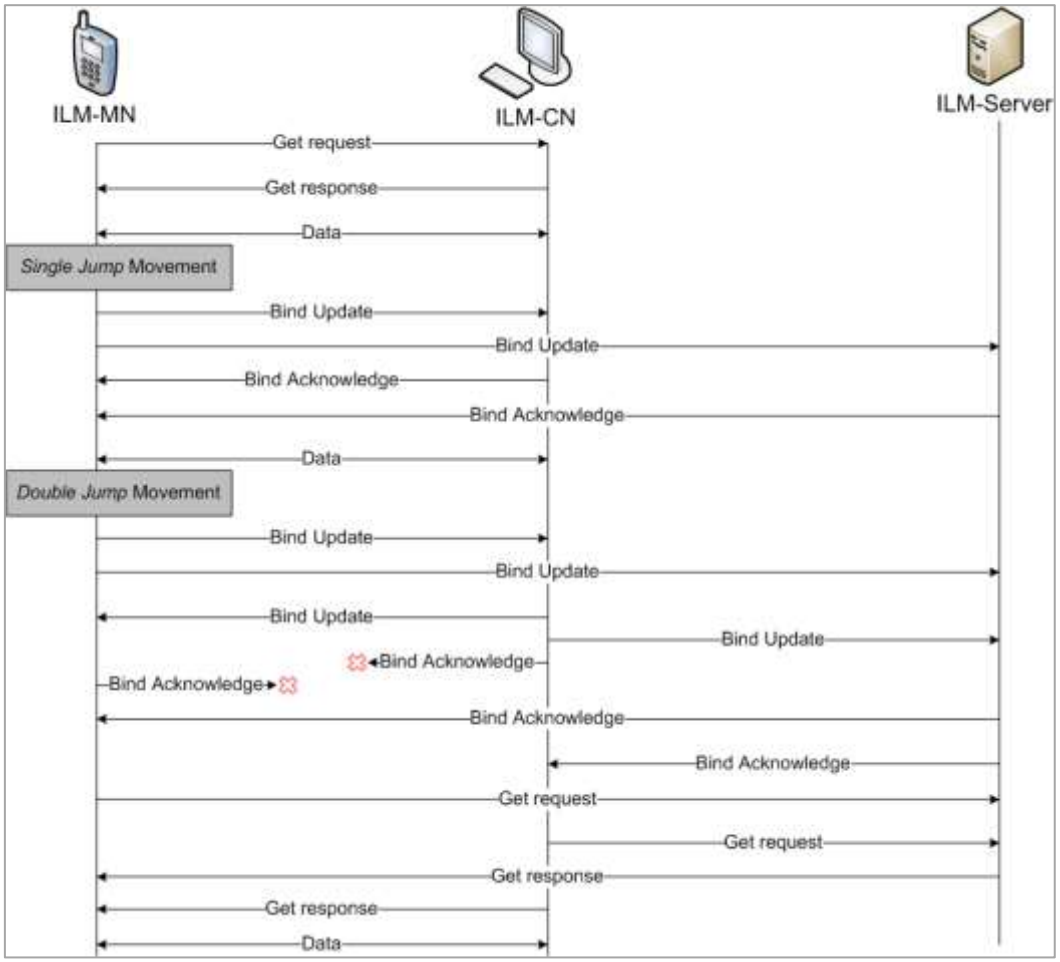


Figura 13 – Diagrama de Sequência do ILM

(1) O *Mobile Node* envia um *Get Request* para o *Correspondent Node* a fim de estabelecer uma comunicação.

(2) O *Correspondent Node* retorna um *Get Responde* para a solicitação do *Mobile Node*.

(3) A comunicação entre os nós é estabelecida e a troca de dados é realizada.

(4) O *Mobile Node* desloca-se para uma nova rede (*single jump*) e envia um BU com o novo endereço de rede para o *Correspondent Node* e para o ILM-*Server*.

(5) Tanto o *Correspondent Node* e ILM-*Server* retornam com um BA.

(6) A comunicação entre os nós continua, sem interrupções.

(7) Simulando o *double jump*, o *Mobile Node* e *Correspondent Node* deslocam-se ao mesmo para novas redes e enviam o BU para o endereço IP original dos nós juntamente para o ILM-*Server*.

(8) Ambos nós não recebem o BA da solicitação anteriormente enviada, somente o BA do ILM-*Server*.

(9) Passado um tempo limite sem retorno das solicitações, tanto o *Mobile Node* e *Correspondent Node* recorrerão ao ILM-*Server* para obter o novo endereço do nó correspondente.

(10) O ILM-*Server* retorna a solicitação dos nós com um *Get Response*.

(11) Sabendo o endereço do nó correspondente, a comunicação entre o *Mobile Node* e *Correspondent Node* é continuada.

3.3. Ambiente de Testes

A proposta do ILM é desenvolver um módulo de gerenciamento de mobilidade na camada de transporte dos dispositivos móveis para que seja mantida a comunicação fim a fim mesmo após o deslocamento. Entretanto, para implementar a mobilidade fim a fim, seria necessário criar uma camada intermédia ao nível do kernel, que no Linux possui um sistema monolítico definido por [Tanenbaum, 1999] onde é chamado de a "grande fusão" por não existir uma estrutura visível a nível de camadas, obrigando cada procedimento possuir uma interface bem definida em termos de parâmetros e uma perfeita integração de todo o módulo kernel com a nova camada. Esta camada intermédia, deveria ser capaz de detectar a mobilidade de rede e atualizar o endereço IP dos pacotes das ligações em curso sem que os procedimentos de *Networking* do kernel enviarem os pacotes TCP com *flags* FIN e RST, como visto na Tabela 2, para a outra ponta da comunicação, ou seja, impedir que o kernel desconecte todas as ligações em curso assim que adquirir um novo endereço IP.

<table>
<tr><td colspan="2">Porta Origem</td><td colspan="2">Porta Destino</td></tr>
<tr><td colspan="4">Número de Sequência</td></tr>
<tr><td colspan="4">Acknowlegement</td></tr>
<tr><td>Tamanho</td><td>Reservado</td><td>Flags</td><td>Window</td></tr>
<tr><td colspan="2">Checksum</td><td colspan="2">Urgent Pointer</td></tr>
<tr><td colspan="4">Opções (Se houver)</td></tr>
<tr><td colspan="4">Dados</td></tr>
</table>

Tabela 2 - Cabeçalho dos pacotes TCP

Onde,

Porta de Origem e Destino: Indicam os processos que estão enviando os dados e os processos que estão recebendo estes dados respectivamente.

Número de Sequência: Este campo informa os bytes enviados.

Acknowlegement: Indica os bytes que foram enviados e recebidos sem erros pelo destino e também a sequência do próximo byte a ser esperado.

Tamanho: Tamanho total do pacote/*frame*.

Reservado: Este campo ainda não foi utilizado.

Flags: Indicam o tipo de pacote a ser enviado:

URG: Bit de urgência, no qual deve ter prioridade e lido com urgência pela aplicação (Ex. o Ctrl + C).

ACK: Bit de reconhecimento, indica a confirmação da recepção de um segmento.

PSH: Bit de push, indica que a origem solicitou ao destino, o envio rápido de todos os dados do seu buffer para a aplicação.

RST: Bit reset, indica que a conexão foi resetada pela origem.

SYN: Bit de sincronização, primeiro bit para o estabelecimento de uma conexão, sincronizando os bits de sequência da conexão.

FIN: Bit de finalização, indica que o transmissor solicitou o término da conexão, informando que este é o pacote de término da ligação.

Window: Tamanho da janela para o controle de fluxo.

Checksum: Permite verificar os erros das transmissões.

Tal método foi analisado durante os estudos deste projeto e constatou-se que seria necessário um tempo bem maior para o desenvolvimento desta solução ao nível do kernel, sendo imprescindível a recompilação e sua substituição por completo para ele possa ter acesso a um novo recurso e a sua nova arquitetura e sistemas de arquivos. Assim, visando a complexidade dos recursos e de lógica, optou-se por desenvolver solução em uma estrutura simplificada. Ressaltando que a priore, o intuito é verificar se a atual proposta é viável e suscetível a aperfeiçoamentos e continuidade de desenvolvimento.

Uma vez definido o modelo de comunicação do gerenciador de mobilidade proposto, cabe mostrar como se dará a integração das entidades que foram utilizadas no cenário de desenvolvimento, que visou garantir a emulação do ambiente de comunicação e mobilidade entre dois dispositivos sem alteração da pilha protocolar TCP/IP.

A Figura 14, apresenta o cenário para o desenvolvimento e validação da arquitetura.

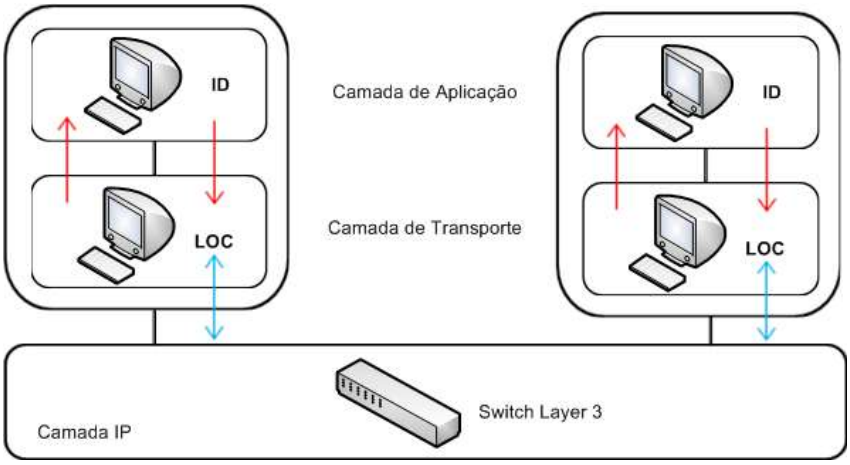


Figura 14 - Ambiente de Testes do ILM

O cenário é composto por dois computadores desktops comuns, denominados Cliente 1 e 2 simulando a camada de aplicação. O papel destes dois dispositivos na arquitetura é iniciar e receber uma comunicação através de uma aplicação. Os IPs destas camadas, são denominados os Identificadores, endereços que não sofrem alterações ao longo da mobilidade. A utilização destes computadores, inibe a primeira ação do kernel em desconectar as ligações imediatamente assim que que ocorrer o *handover*, visto que a mobilidade e seu gerenciamento ocorrem na camada de transporte, ou seja, no computador

que possui o endereço IP Localizador. A nível de testes estes identificadores 1 e 2 possuem os endereços IPs estáticos 192.168.5.2 e 192.168.6.2 respectivamente.

Também compõem o cenário, outros dois computadores desktops denominados *Router* 1 e *Router* 2 que simulam a camada de transporte. A atividade destes computadores é desempenhar o roteamento entre todos os dispositivos do cenário, além de que seus endereços IPs são denominados os Localizadores na arquitetura ILM, sendo estes endereços que sofrerão alterações de mobilidade durante os percursos das ligações. No início das ligações os endereços localizadores 1 e 2 possuem os respectivos endereços IPs para ligações externas 192.168.3.2 e 192.168.4.2 na interface em1 adquiridos por DHCP e para as ligações internas os IPs 192.168.5.1 e 192.168.6.1 nas interfaces p4p1 e p5p1 respectivamente.

A configuração dos computadores utilizados é apresentado na Tabela 3.

	Router 1	**Router 2**	**Cliente 1**	**Cliente 2**
Memória	1 GB	8 GB	512 MB	512 MB
Hard Disk	80 GB	1 TB	40 GB	40 GB
Processador	Intel Pentium IV 2.80 GHz	Intel Core i7 3.4 GHz	Intel Pentium IV 2.53 GHz	Intel Pentium IV 2.60 GHz
Sistema Operacional	Fedora 17	Fedora 17	Fedora 17	Fedora 17
Kernel	3.3.4-5	3.3.4-5	3.3.4-5	3.3.4-5

Tabela 3 - Configuração dos Computadores Utilizados

Um switch layer 3 da Cisco 3550, é responsável por simular a camada de rede e onde é realizado a troca de IPs dos localizadores quando se desejar simular a mobilidade de um ou dos dois nós. Também vale ressaltar que os IPs do switch são utilizados basicamente para a interligação dos equipamentos e transmissão de informações, também sem funcionalidades em relação à arquitetura do ILM. Para que fosse possível emular a mobilidade dos nós, foi criado no *switch* uma divisão da rede física em diferentes domínios de disseminação denominados Virtual LANs (VLANs) . Do atual entendimento lógico, estes domínios funcionam como se fossem redes independentes, desde cablagem aos equipamentos, não havendo ligações direta entre os dispositivos conectados a diferentes VLANs. A Figura 15, exibe a alternância de uma rede para a VLAN 8, de forma a simular a mobilidade de um nó através dos comandos *interface fastethernet* e *switchport access* do *switch*. O tempo gasto para que o *Mobile Node* receba o endereço IP da nova VLAN via DHCP é de 23 segundos.

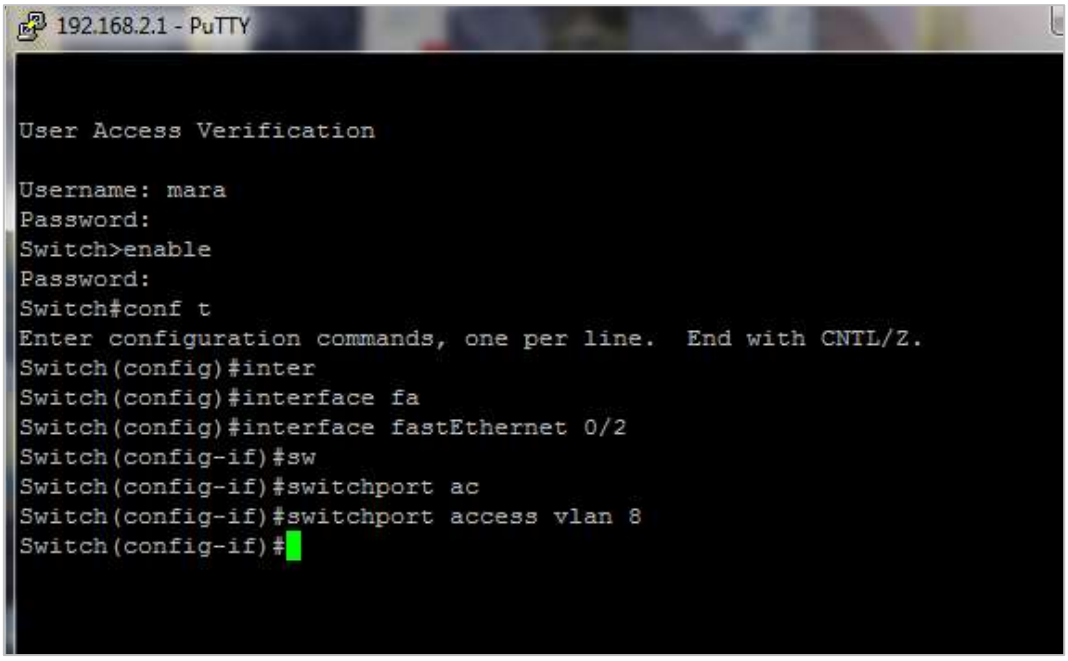


Figura 15 - Simulando a mobilidade dos MN pelo Switch.

Para que houvesse comunicação entre as VLANs, *routers* (desktop localizadores 1 e 2) fizeram-se necessários para realizar a comunicação que pode ser vista na Tabela 4.

(Router 1)	(Router 2)
#route -nn Tabela de Roteamento IP do Kernel Destino Gateway ... Iface 192.168.5.0 192.168.3.2 ... p4p1	**#route -nn** Tabela de Roteamento IP do Kernel Destino Gateway ... Iface 192.168.6.0 192.168.4.2 ... p5p1

Tabela 4 - Roteamento do Kernel - Router 1 e 2

A divisão lógica de 6 VLANs (de 3 a 8) foi criada no *switch*, onde o *router* 1 pertence à VLAN 3 (com o IP 192.168.3.1) e pertence também à VLAN 5. Isto se fez necessário para que houvesse comunicação direta com o cliente 1 que pertence também à VLAN 5. O mesmo ocorre com o *router* 2, que por hora está presente tanto na VLAN 4 (IP 192.168.4.1) quanto na VLAN 6, juntamente com o cliente 2, como pode ser visto na Figura 16. As VLANs 7 (192.168.7.1) e 8 (192.168.8.1) são utilizadas para realizar saltos, ou seja, a mobilidade dos nós para outros domínios. As VLANs 3, 4, 7 e 8 foram configuradas por DHCP para que fosse possível prover um novo endereçamento para os hosts após a mobilidade e as restantes VLANs, configuradas em modo estático.

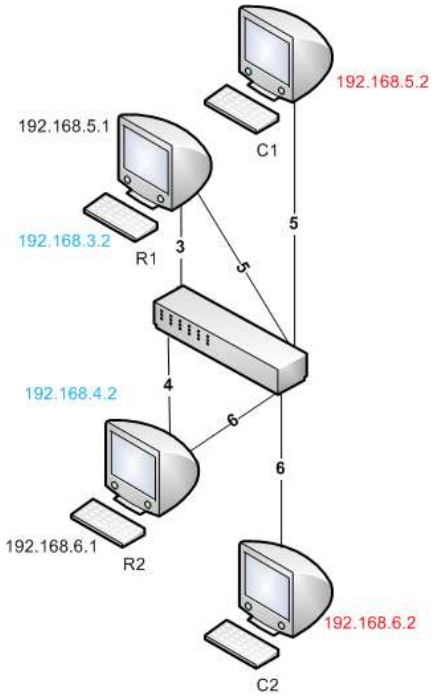


Figura 16 - Integração dos dispositivos e VLANs

Para exemplificar o funcionamento da comunicação no ambiente acima proposto, será aqui descrito os passos da comunicação original e após a mobilidade de rede:

(1) O Cliente 1, inicia uma comunicação cujo o destino é o *Router* 2.

(2) Ao passar pelo *Router* 1, o endereço de origem dos pacotes é alterado para a interface de saída do *Router* 1.

(3) Os pacotes da comunicação chegam ao destino, o *Router* 2 e são redirecionados para o Cliente 2.

(4) No retorno da comunicação, o Cliente 2 envia os pacotes da comunicação para o *Router* 1, passando pelo *Router* 2 que também altera o endereço de origem para a sua interface de saída.

(5) Assim que os pacotes chegarem ao *Router* 1, estes serão encaminhados para o Cliente 1, completando assim a comunicação entre dois terminais, sem a mobilidade.

(6) Vamos considerar a mobilidade do *Router* 2 com a comunicação anterior em curso através da alteração de sua VLAN de 4 para 8. O *Router* 2 envia seu novo endereço IP para o *Router* 1 assim que a mobilidade for detectada.

(7) O *Router* 1 altera o endereço de origem dos pacotes para o endereço antigo do Router 2 e encaminham para o Cliente 1.

(8) No retorno da comunicação, quando as mensagens do Cliente 1 com destino ao endereço IP antigo do *Router* 2 (IP final 4.2), passarem pelo *Router* 1, o endereço de destino é alterado para o endereço real do *Router* 2 (IP final 8.2).

Desta forma, a mobilidade não deve sentida pelas aplicações, que estão sempre se comunicando com os endereços *Routers* estabelecidos no início da comunicação, independente da mobilidade.

É importante ressaltar que a comunicação se faz fim a fim, do *Router* 1 para o *Router* 2 e que o encaminhamento dos pacotes para os clientes é realizado com o intuito de não haver a desconexão imediata ao nível do kernel, diferentemente da solução LISP.

3.4. O Módulo Socket

A solução de gerenciamento de mobilidade na camada de transporte, contou com um módulo básico de *sockets*, o IO:: *Module Base* [BIBLIOGRAFIA]que fornece uma abordagem orientada a objetos para a programação via *sockets*. Parte da solução ILM foi implementada por meio de uma adequação das funções específicas deste módulo por se adequar nas primitivas de comunicação preexistentes na camada de transporte. Para o tratamento da qualidade das transmissões fim a fim, funções presentes nesta camada vinculadas ao controle de recebimento de dados e controle de fluxo são utilizadas. Desta forma, segue as funções de comunicação utilizadas pertencentes ao IO:: *Module Base*: *socket, new, accept, autoflush, send, recv, gethostbyname, hostname, peerhost, peerport* e *close*.

3.4.1. *Daemon* Servidor *Socket*

Ao inicar o módulo ILM, em ambos terminais móveis um *daemon socket* é criado como uma aplicação servidora que está sempre em *listening*. Ela foi criada com a finalidade de receber e prover ao ILM a informação do novo endereço IP do terminal móvel em comunicação em caso de *handover*, basicamente esta aplicação está atenta aos *Bind Updates* enviados pelos dispositivos. O código apresentado na Tabela 5 mostra que o servidor está em escuta de dados TCP na porta 7001 e caso ele receba alguma informação do cliente, esta informação é lida até o fim e armazenada na variável $data. A função *autoflush* solicita o envio da resposta imediatamente sem o armazenamento da informação no *buffer* de saída. Posteriori, esta informação que na verdade é o endereço IP do *Correspondent Node*, é disponibilizada em um arquivo para a leitura do ILM.

```
use IO::Socket::INET;
use strict;
...
my $server_socket = IO::Socket::INET->new(
          LocalPort=>'7001',
          Proto=>'tcp',
          Reuse=> 1,
          Listen=> 5
) or die "não foi possivel criar o socket.($!)\n";
...
$server_socket->listen();
$server_socket->autoflush(1);
...
while ($client = $server_socket->accept() )
{
          ...
          print "\nConnected from:", $client->peerhost();
          print "\nPorta:", $client->peerport(), "\n";
          while (1)
          {
                    ...
                    $client->recv($resultado, 1024);
                    open (OUT, ">iprcv.txt");
                    print OUT "$resultado";
                    close (OUT);
                    ...
                    last;
          }
          chomp;
          close $client;
}
```

Tabela 5 - Daemon Socket Servidor

3.4.2. Identificação do Mobile Node e Bind Update

Para detectar o estabelecimento de uma ligação, foram analisados as *flags* do cabeçalho dos pacotes de todas as mensagens recebidas em ambos *hosts* através da ferramenta tcpdump. Esta ferramenta permite a monitoração de todo o tráfego da rede, exibindo todo cabeçalho dos pacotes. A detecção do *handshake* das ligações em curso foi adquirida através do seguinte comando e saída exibido na Tabela 6.

```
tcpdump -n -i em1 'tcp[13] & 2 != 0 && tcp[13] & 16 != 0'

13:47:53.595830 IP router1.ssh > 192.168.4.2.35252: Flags [S.], seq 1373686591, ack
3989505587, win 14480, options [mss 1460,sackOK,TS val 6239111 ecr 6224083,nop,wscale
7], length 0
```

Tabela 6 - Saída da Monitoração das Flags SYN-ACK

A posição da *flag* no cabeçalho TCP está localizada na décima quarta posição, com os índices iniciados em 0, logo sua localização está na posição 13, justificando a busca do tcpdump 'tcp[13]'. A *flag* SYN é segundo bit de mais baixa ordem, logo uma comparação bit a bit é realizada através da regra '2 != 0' do tcpdump, onde o resultado será 00000000 caso o bit SYN estiver desligado e 00000010 se estiver ligado. A mesma comparação ocorre caso a *flag* ACK estiver ativada através dos bits 00010010.

Ao ser detectado a *flag* SYN-ACK, sua saída será direcionada a um arquivo para receber o tratamento através da expressão regular "s/. (\d+\.\d+\.\d+\.\d+\.\d+) .*/" afim de retornar somente o endereço IP e a porta da ligação e em seguida armazená-los em uma lista bidirecional para que seja possível estabelecer uma ligação *socket* futura com o servidor. Desta maneira, o endereço IP de todas as ligações estabelecidas no dispositivo móvel é armazenado nesta lista, com o intuito de se obter o controle de quais endereços IPs o dispositivo móvel mantém ligações. Toda a vez que se é identificado através do comando tcpdump apresentado na Tabela 7, uma flag RST ou FIN, o endereço IP correspondente à essa *flag* é apagado da lista de endereços IPs válidos.

```
tcpdump –i em1 tcp[tcpflags] & (tcp-fin | tcp-rst) != 0

16:09:06.726810 IP router1.cdc > 192.168.8.2.52009: Flags [R], seq 229017455, win 0,
length 0
```

Tabela 7 - Saída da Monitoração das Flags FIN e RST

Tendo em mãos o endereço IP do servidor, que na verdade é o endereço do *Correspondent Node*, uma verificação contínua do endereço do *Mobile Node*, atravé do *socket* cliente é realizada através das funções hostname( ) e gethostbyname( ), que retornam o endereço atual do dispositivo, com o objetivo de detectar uma possível mobilidade. As funções utilizadas são apresentatas à seguir.

```
$Hostname = Hostname();

$IpAddrAtual = Pinet_ntoa(scalar gethostbyname($HostName || 'localhost'));
```

Após o *Mobile Node* se locomover e adquirir um novo endereço IP, o cliente *socket* estabelece uma ligação com o servidor e realiza o envio instantâneo do seu novo IP, denominado como *Bind Uptade* como pode ser visto na Tabela 8.

```
use IO::Socket::INET;
use Socket;
use Sys::Hostname;
...
system("/usr/sbin/tcpdump -n -i p2p1 'tcp[13] & 2 !=0  && tcp[13] & 16 !=0 ' >syn-ack.txt -Z
root &");

system("/usr/sbin/tcpdump -i p2p1 'tcp[tcpflags] & (tcp-fin | tcp-rst) !=0' >fin-rst.txt -Z
root &");
...
# Daemon Cliente.
while (1) {
           ...
                     if(/(\d+\.\d+\.\d+\.\d+\.\d+:)/)
                     {
                                $name=$1;
                               $aux1=$name;
                                $aux1=~s/.\d+://;
                                chomp($aux1);
                                if ($name) {
                                          $name=~s/\d+\.\d+\.\d+\.\d+./$1/;
                                          $aux2=$name;
                                          chop($aux2);
                                ...
           $HostName = hostname();
           $IpAddrAtual = inet_ntoa(scalar gethostbyname($HostName || 'localhost'));
           print "\n\nMeu IP Atual: $IpAddrAtual";
           ...
           my $aux="";
           open AX, "route -n | grep ^0.0.0.0 |";
           chop($aux=<AX>);
           close AX;
           $aux=~s/.* ([^ ]+)$/$1/;
           if ($aux) {
                     open AX, "ifconfig $aux | grep 'inet ' |";
                     chop($aux=<AX>);
                     $aux=~s/.*inet (\d+\.\d+\.\d+\.\d+) .*/$1/;
                     close AX;  }
           $IpAddrAtual=$aux;
...
                     if ($newIpAddr eq $IpAddrAtual)
                     {
                        print "\n Mobilidade não Detectada: ",$IpAddrAtual, "\n\n";
                     }
                     else
                     {
                        print "\nMobilidade Detectada. Meu Novo IP: ",$newIpAddr, "\n";
                     for ($j=0; $j <= $ip; $j++) {
                                $name = $ip[j];
                                $port = $porta[j];
$client_socket->autoflush(1);
my $client_socket = IO::Socket::INET->new('PeerAddr' => $name, 'PeerPort'  => $port, Proto =>
'tcp') or last "\nNao Foi criado o Socket $!\n";
... }

my $client_socket = IO::Socket::INET->new('PeerAddr' => '192.168.9.2', 'PeerPort'  => '7001',
Proto => 'tcp') or last "\nNao Foi criado o Socket $!\n";
...
```

Tabela 8 - Socket Cliente

O *Bind Update* será realizado para todos os endereços presentes na lista de IPs válidos mencionados anteriormente. Após o *Bind Update*, o *socket* é fechado e somente reiniciado em caso de nova mobilidade. Um *daemon* também foi estabelecido para detectar o *handover*.

3.5. Atualização de Localização através de Iptables

Após a mobilidade e o *Correspondent Node* receber o novo endereço IP do *Mobile Node*, uma atualização de endereços das ligações em curso se fez necessário, com o intuito de alterar o novo endereço IP dos pacotes para IP o anterior estabelecido no início da ligação e enviá-los para a camada de aplicação, além de alterar também o destino destes pacotes no

retorno da comunicação. Isso se fez com o intuito da aplicação não sentir a mobilidade estabelecida de forma transparente. Inicialmente, cogitou-se a utilização das regras *mangle* para realizar diferentes marcações nos pacotes de cada ligação estabelecida e traduzi-las através do *Network Address Translation* (NAT) [Egevang et al., 1994]. E a posteriori em manipular o NAT sem a tabela de estados.

3.5.1. CONNMARK

Foi proposto inserir uma marcação através do CONNMARK, um recurso bastante imponente do Netfilter que oferece uma forma de se ter uma marca nas conexões de entrada presente na tabela de estado do kernel do linux. Foi realizado marcações das conexões em cada ligação indicada pela *flag* SYN, de maneira que fosse possível diferenciar as ligações mesmo se após o *handover* um novo dispositivo adquirisse o endereço IP identificador de uma ligação ainda em curso.

A utilização das regras *mangle*, permitiu realizar um tratamento diferenciado em todas as ligações, porque o CONNMARK marca todos os pacotes da mesma conexão subsequente ao seu estabelecimento. Posteriormente as regras NAT foram utilizadas para fazer a tradução de endereços para cada conexão marcada pelo *mangle*. Um exemplo minimalista e direto pode ser visto na tabela 9.

Programa	Tabela e Chain	Expressão	Ação
Iptables	-t mangle -A PREROUTING	-p tcp --sport 223 -s 192.168.4.2 -j CONNMARK	--set-mark 123
Iptables	-t nat -A PREROUTING	-m mark --mark 1 -j	DNAT --to-destination 192.168.8.2

Tabela 9- Exemplo de regras Iptables para redirecionamento

As regras mencionadas indicam a marcação de **--set mark 123** para a conexão que possui como origem o endereço IP 192.168.4.2 visto na Figura 19. Já a regra NAT logo à seguir, realiza a alteração do destino dos pacotes marcados previamente para o IP 192.168.8.2 através do DNAT.

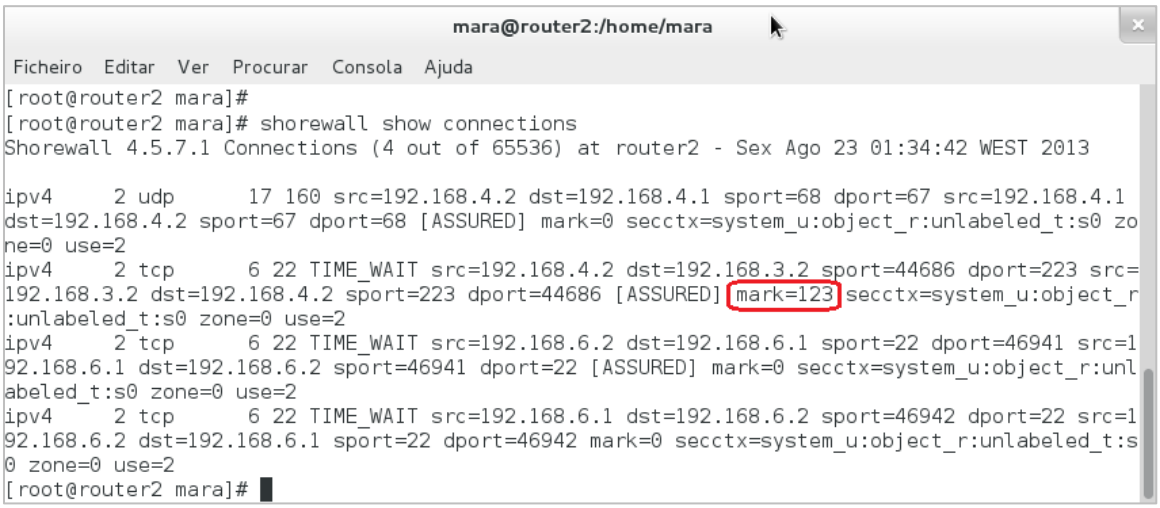


Figura 17 - Marcação dos pacotes através do CONNMARK

A captação das conexões marcadas pelo CONNMARK foi realizada através da ferramenta de alto nível para configurações Netfilter denominada Shoreline Firewall (Shorewall).

Entretanto, esta proposta foi descontinuada após se descobrir através dos testes que as marcações feitas na tabela mangle não persistem ao passar por algum tipo de NAT, tornando inviável a marcação de pacotes através da tabela *mangle* como possível parte de uma decisão de roteamento. A fim de tentar contornar este obstáculo, se pensou em salvar as marcações *mangle* através do **--save mark** antes da regra NAT e restaurá-lo posteriormente com a máquina de estado através da regra **--restore-mark** que restaura as marcações nos pacotes das conexões marcadas pelo CONNMARK visto na Tabela 10.

Programa	Tabela e Chain	Expressão	Ação
Iptables	-A PREROUTING -t mangle	–i em1 -j CONNMARK	--save-mark
Iptables	-A POSROUTING -t mangle	-j CONNMARK	--restore-mark

Tabela 10 - Salvar e Restaurar as marcações pelo CONNMARK

Esta solução também tornou-se impraticável por duas razões. Primeiro se fez necessário desativar a tabela de estado *connection tracking* (conntrack) para a manipulação desejada da tabela NAT e o CONNMARK utiliza esta tabela para determinar quais pacotes serão marcados, sendo eles com o estado ESTABLISHED ou RELATED e sem esta tabela é impossível restaurar qualquer marcação salva anteriormente. A segunda questão está relacionado ao CONNMARK que está disponível em todas as *chains* e tabelas do iptables, porém sem a desativação do conntrack, somente o primeiro pacote de uma ligação atravessa a tabela NAT, sendo assim o CONNMARK não tem nenhum efeito para os pacotes subsequentes.

3.5.2. NAT e Connection Tracking (Conntrack)

Dando continuidade a esta proposta de solução ainda não explorada no meio acadêmico para tratamento da mobilidade de redes, um estudo aprofundado do iptables, principalmente da tabela NAT foi realizado a fim de identificar os problemas e propor as devidas soluções para a sua aplicação ao ILM.

O Iptables é um firewall que utiliza uma cadeia de regras que operam dentro do kernel do linux com o propósito de filtrar os pacotes baseando-se em regras específicas ou em um conjunto de regras. O iptabales é um firewall do tipo *statefull*, denominado um firewall com estado por fazer a utilização específica de uma tabela de estados designado *Connection Tracking* (conntrack). Quando o firewall utiliza um modo de filtragem do tipo *stateless* (Ipchains) a execução de cada pacote roteado é feito individualmente tornando mais fácil o seu tratamento e sua implementação por ter uma resolução mais rápida ao se comparar com um tipo *statefull*, além de um melhor desempenho. Entretanto, sua configuração é mais trabalhosa e menos segura já que cada pacote recebido é considerado novo, independentemente de sua direção, seja uma requisição ou uma resposta de alguma solicitação prévia. Vale ressaltar que o ipchains foi descontinuado à partir da versão 2.4 do kernel.

O modelo de filtragem do tipo *statefull* (Iptables)elabora um forte sistema firewall que utiliza o conntrack para registrar e informá-lo das conexões previamente estabelecidas,

evitando assim certos tipos de ataques como o famoso *Stealth Scan* que basicamente envia um pacote com a flag ACK ativa, sem ter iniciado uma conexão através da flag SYN. Ataque este bastante utilizado em firewalls *stateless*. O iptables também é conhecido como filtro de inspeção de estados por tomar as decisões de filtragem buscando informações fundamentada na tabela de estados e também por considerar todas as informações dos pacotes e não somente o cabeçalho, funcionando da seguinte maneira:

(1) O Cliente, inicia uma comunicação e envia um pacote com a flag SYN para o firewall.

(2) O pacote atravessa as regras definidas pelo firewall.

(3) Caso o pacote não pertença a nenhuma regra, ele é descartado e a conexão rejeitada.

(4) Se o pacote pertencer a alguma regra que não seja o descarte, a sessão é então cadastrada na tabela de estados conntrack, presente na memória do kernel do linux.

(5) Em seguida, o pacote é então direcionado para seu destino e todos os pacotes seguintes serão encaminhados diretamente para a tabela de estados sem passar pelas regras do firewall.

Este procedimento clarifica as questões sobre o não funcionamento da regra NAT e *mangle* utilizada na solução anterior.

3.5.2.1. Connection Tracking (Conntrack)

A tabela de estados conntrack é responsável por registrar todas as ligações que passam pelo firewall. Esses registros são fundamentados nas informações dos protocolos presentes nos pacotes da comunicação. Também são criadas regras exclusivas, designando ao firewall quando a tabela de estados deve ser consultada [Fávero, 2007]. Uma entrada no conntrack é adicionada, toda vez que uma conexão que passa pelo firewall é iniciada seja originada nele ou fora dele, sendo as informações registradas as que seguem:

- Versão do protocolo IP (IPv4 ou IPv6);
- Indicação numérica do protocolo IP;
- Protocolo da conexão;
- Indicação numérica do protocolo da conexão;
- Tempo restante para a remoção da ligação na tabela do conntrack;
- Endereço IP de origem e endereço IP de destino;
- Porta de origem e porta de destino;
- Todas as informações descritas inversamente no qual os pacotes de retorno da ligação devem conter;
- O estado do pacote;
- Valor de marcação interna do pacote, não relacionado as marcações de conexão.

A Figura 18 exibe uma entrada do conntrack presente em /proc/net/nf_conntrack.

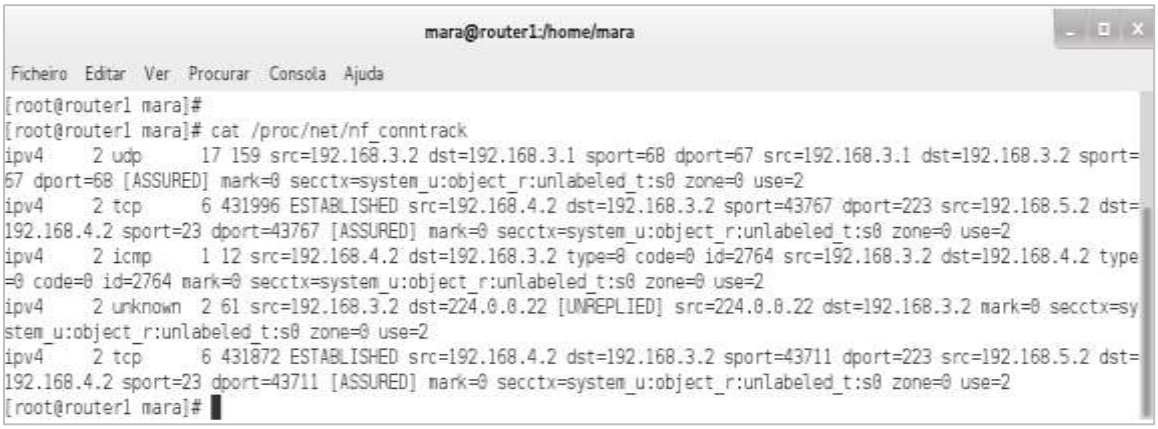


Figura 18 - Entrada da tabela de estados Conntrack (/proc/net/nf_conntrack)

O conntrack verifica as condições das regras de cada pacote através da extensão *state* que possui permissão para acessar as informações do estados e ativar esta verificação. Os possíveis estados dos pacotes são divididos nos quatro seguintes:

- NEW: Este estado indica o primeiro pacote de alguma conexão detectada pelo dispositivo, ou seja, qualquer pacote não se referindo somente a conexões TCP. Um erro bastante comum é cogitar que o estado NEW refere-se somente aos pacotes do tipo TCP-SYN e não aos outros protocolo UDP ou ICMP.

- ESTABLISHED: Os pacotes com este estado estão relacionados a alguma conexão estabelecida, permitindo a passagem dos pacotes posteriores. Este estado possui uma ação poderosa ao tipo *statefull* do firewall.

- RELATED: Este estado está relacionado com pacotes que iniciem uma nova conexão em alguma porta, mas que estejam associados com alguma conexão previamente existente. Como por exemplo o protocolo FTP, utilizado para transferência de dados, mensagens ICMP ou mesmo alguma resposta DNS. Vale ressaltar que somente o netfilter não é capaz de gerenciar duas conexões sem solicitar o auxílio de módulos específicos destes protocolos.

- INVALID: Este estado indica que o pacote não foi identificado por algum motivo e está associado a qualquer pacote não presente na tabela de estados. Recomenda-se bloquear este tipo de pacote por estar relacionado a mensagens de erros que não pertencem a alguma conexão reconhecida.

Devido a este mecanismo *statefull* do iptables e a predominância em tentar utilizá-lo como roteamento e tradução de endereços para aplicar ao ILM, decidiu-se criar um script automatizado de regras na qual não fosse consultada a tabela de estados, ou mesmo que não fosse armazenado as entradas no conntrack, já que ele é iniciado automaticamente quando o iptables se torna ativo. Logo, pensou-se em utilizar o firewall em modo *stateless*, aplicando a tabela *raw* presente somente nas versões mais recentes do kernel do linux (superior a versão 3.0.0) que possui a função de isentar o rastreamento das conexões em combinação com a *target* NOTRACK [Eychenne, 2013]. Esta tabela possui as *chains* PREROUTING e OUTPUT e tem o poder de alterar os pacotes no mais baixo nível do kernel, alterando-os em modo bruto, entrando em ação bem antes do conntrack e das tabelas mangle→ nat→filter→security do iptables, pois as políticas de filtragem deste firewall depende da

configuração das regras das tabelas, estando diretamente ligado a ordem como o iptables lê essas regras. A Figura 19, exemplifica melhor este comentário.

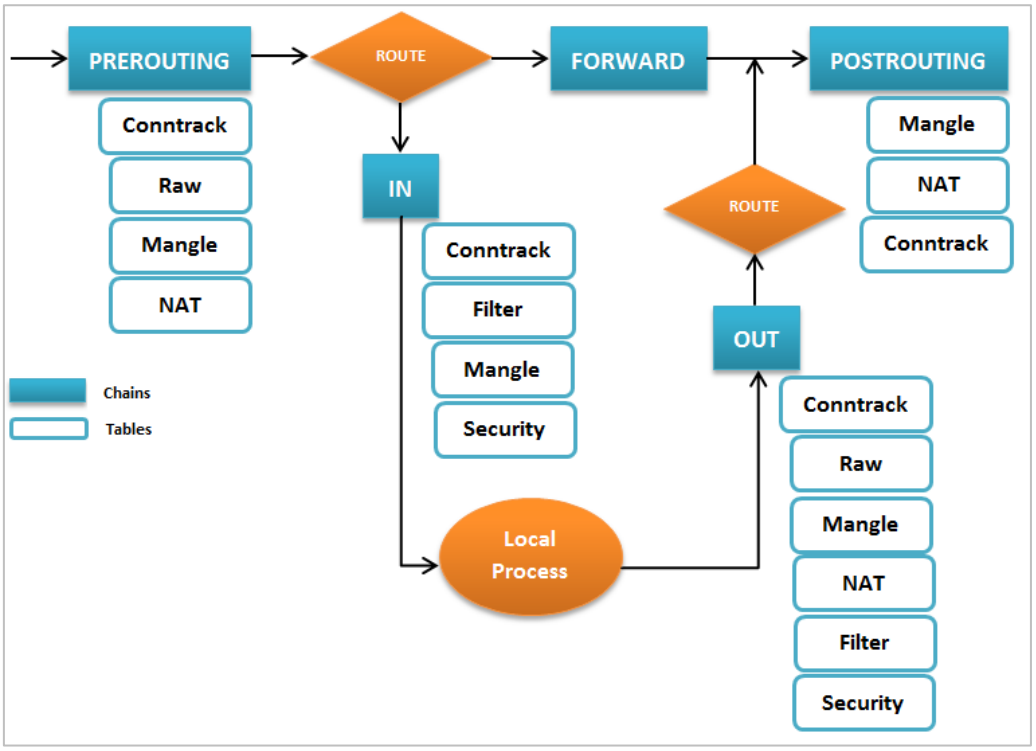


Figura 19 - Caminho dos Pacotes no Firewall IPtables.

Para a aplicação da tabela *raw* ao ILM utizou-se algumas das regras iptables apresentadas na Tabela 11.

Programa	Tabela e Chain	Expressão	Ação
Iptables	-t raw -I PREROUTING	-s 192.168.4.2 -p tcp --dport 23 -j	NOTRACK
Iptables	-t raw -I PREROUTING	-j	NOTRACK
Iptables	-t raw -I OUTPUT	-j	NOTRACK
iptables	-A FORWARD	-m state --state	UNTRACKED -j ACCEPT

Tabela 11 – Desabilitando o rastreamento das conexões através da tabela raw.

A regra presente na linha 2 da Tabela 11 configura uma exceção no mecanismo conntrack do kernel para os pacotes cuja origem possui o endereço IP 192.168.4.2. Ela foi utilizada a fim de que esta conexão se tornasse *stateless* e autorizasse a tradução de endereços através da tabela NAT após o início da conexão. As regras das linhas 3 e 4 configuraram a exceção de todo o tráfego de entrada e saída que passa pelo firewall. A ultima regra, permite que todos os pacotes com o estado UNTRACKED, definidos pelas regras anteriores sejam roteados. Entretanto os testes mostraram que como mencionado, a entrada dos pacotes descritos nas expressões das regras acima não são registrados pelo conntrack, porém foi constatado que o NOTRACK ocasiona um *jump* nas *chains* do iptables, ou seja nenhum outro pacote da ligação passou por outras regras presentes no firewall, logo, as regras da tabela NAT não foram lidas como previa-se.

Para realizar o *debugging* do iptables e poder ter uma melhor análise sobre em quais tabelas e *chains* o pacote das conexões passaram, foi utilizado a regra TRACE também pertencente a tabela raw cuja regras são apresentadas na Tabela 12.

Programa	Tabela e Chain	Expressão	Ação
Iptables	-t raw -I PREROUTING	-j	TRACE
Iptables	-t raw -I OUTPUT	-j	TRACE

Tabela 12 - Regras de Depuração do Iptables

Para visualizar o debugging, foi necessário configurar o arquivo de log. Primeiramente foi carregado o módulo de log do iptables através de modprobe ipt_LOG e modprobe ip6t_LOG. Depois foi definido a saída do log do kernel inserindo a linha **kern.* /var/log/kern.log** dentro do arquivo /etc/rsyslog.conf. Por ultimo, foi setado o selinux context através das linhas de comando **su -c "touch /var/log/kern.log"** e **su -c "chcon system_u:object_r:var_log_t:s0 /var/log/kern.log"** e o rsyslog reiniciado com **su -c "service rsyslog restart".**

Através das regras da Tabela 11, foi possível constatar verificando o log do kernel que, como mencionado anteriormente, quando há a utilização da regra NOTRACK os pacotes saltam a tabela mangle e a tabela NAT. No caso mais específico da Figura 20, a regra NOTRACK estava ativa, juntamente com as regras SNAT (ida e volta) e regras DNAT do ILM. Entretanto o pacote passou pela tabela raw na chain PREROUTING e pela tabela filter na chain OUTPUT, ignorando completamente as regras estabelecidas pela tabela NAT.

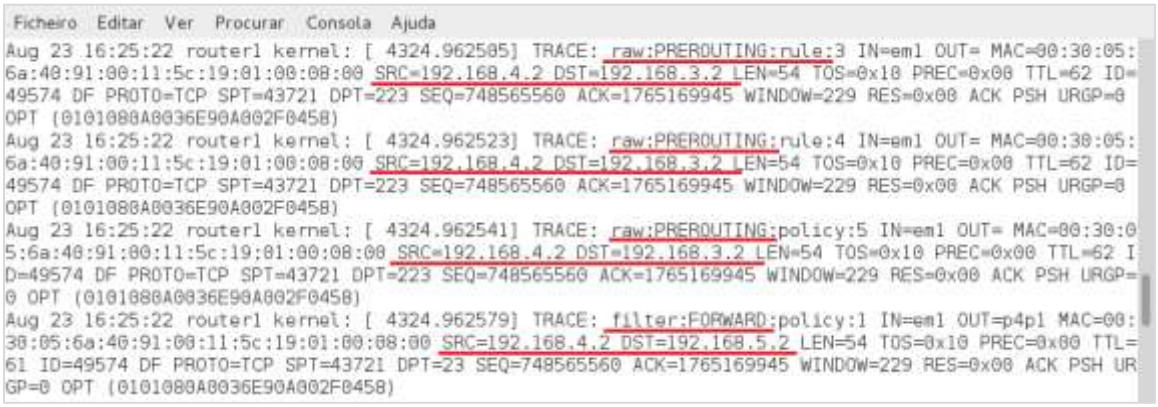


Figura 20 - Depuração do iptables com o NOTRACK (/var/log/kern.log)

Outra solução, realizada de um modo mais abrupto, foi desabilitar permanentemente o módulo conntrack dos dispositivos relacionados ao iptables e ao kernel. Para isso, foi necessário criar módulos *blacklists* via modprobe através do arquivo **blacklist.conf** localizado em /etc/modprobe.d/. A Tabela 13 exibe os módulos desabilitados.

```
/etc/modprobe.d/blacklist.conf

# Desabilitar nf_conntrack
rmmod nf_conntrack
rmmod nf_conntrack_ipv6
rmmod xt_conntrack
rmmod nf_conntrack_ftp
rmmod xt_state
rmmod iptable_nat
```

```
rmmod ipt_MASQUERADE
rmmod ipt_REDIRECT
rmmod nf_nat
rmmod nf_conntrack_ipv4
```

Tabela 13 - Blacklists para desabilitar o Conntrack

A remoção do conntrack não alterou em nada o ambiente estabelecido, o sistema continuou estável e o iptables finalmente passou a trabalhar em modo *stateless* como desejado. A Figura 21 exibe os pacotes passando pela tabela raw, NAT, e filter.

```
Ficheiro  Editar  Ver  Procurar  Consola  Ajuda
Aug 23 16:19:57 router1 kernel: [ 3999.668813] TRACE: raw:PREROUTING:rule:4 IN=em1 OUT= MAC=00:30:05:6a:
40:91:00:11:5c:19:01:00:08:00 SRC=192.168.4.2 DST=192.168.3.2 LEN=60 TOS=0x10 PREC=0x00 TTL=62 ID=49545
DF PROTO=TCP SPT=43721 DPT=223 SEQ=748565474 ACK=0 WINDOW=14600 RES=0x00 SYN URGP=0 OPT (020405B40402080
A0031F26B0000000001030306)
Aug 23 16:19:57 router1 kernel: [ 3999.668832] TRACE: raw:PREROUTING:policy:5 IN=em1 OUT= MAC=00:30:05:6
a:40:91:00:11:5c:19:01:00:08:00 SRC=192.168.4.2 DST=192.168.3.2 LEN=60 TOS=0x10 PREC=0x00 TTL=62 ID=4954
5 DF PROTO=TCP SPT=43721 DPT=223 SEQ=748565474 ACK=0 WINDOW=14600 RES=0x00 SYN URGP=0 OPT (020405B404020
80A0031F26B0000000001030306)
Aug 23 16:19:57 router1 kernel: [ 3999.668868] TRACE: nat:PREROUTING:rule:1 IN=em1 OUT= MAC=00:30:05:6a:
40:91:00:11:5c:19:01:00:08:00 SRC=192.168.4.2 DST=192.168.3.2 LEN=60 TOS=0x10 PREC=0x00 TTL=62 ID=49545
DF PROTO=TCP SPT=43721 DPT=223 SEQ=748565474 ACK=0 WINDOW=14600 RES=0x00 SYN URGP=0 OPT (020405B40402080
A0031F26B0000000001030306)
Aug 23 16:19:57 router1 kernel: [ 3999.668896] TRACE: filter:FORWARD:policy:1 IN=em1 OUT=p4p1 MAC=00:30:
05:6a:40:91:00:11:5c:19:01:00:08:00 SRC=192.168.4.2 DST=192.168.5.2 LEN=60 TOS=0x10 PREC=0x00 TTL=61 ID=
49545 DF PROTO=TCP SPT=43721 DPT=23 SEQ=748565474 ACK=0 WINDOW=14600 RES=0x00 SYN URGP=0 OPT (020405B404
02080A0031F26B0000000001030306)
Aug 23 16:19:57 router1 kernel: [ 3999.668912] TRACE: nat:POSTROUTING:policy:2 IN= OUT=p4p1 SRC=192.168.
4.2 DST=192.168.5.2 LEN=60 TOS=0x10 PREC=0x00 TTL=61 ID=49545 DF PROTO=TCP SPT=43721 DPT=23 SEQ=74856547
4 ACK=0 WINDOW=14600 RES=0x00 SYN URGP=0 OPT (020405B40402080A0031F26B0000000001030306)
```

Figura 21 - Depuração do iptables sem o NOTRACK (/var/log/kern.log)

3.5.2.2. NAT

Outra particularidade utilizada no ILM e já comentada, foi o impregamento da tabela NAT com o objetivo de traduzir o endereço IP dos pacotes das ligações já em curso de acordo com o novo endereço IP adquirido pelo terminal móvel após a mobilidade. A decisão da sua utilização se deu pela simplicidade ao aplicá-la através do iptables e também pela inovação da proposta. A ideia base era criar uma solução que alterasse o mínimo possível o ambiente do dispositivo móvel e reduzisse ao máximo possível os dispositivos excedentes como vistos nos protocolos do Capítulo 2.

A tabela NAT tem o objetivo de controlar a tradução de endereços de pacotes que não são gerados localmente. Esses pacotes quando atravessam o *script* de roteamento do linux sofrem influência do NAT para realizar alterações de origem e destino através da compilação do NF_IP_PRE_ROUTING, NF_IP_POST_ROUTING, NF_IP_LOCAL_IN e NF_IP_LOCAL_OUT [Egevang & Francis, 1994].

As *chains* que fazem parte da tabela NAT são PREROUTING, OUTPUT e POSTROUTING, examinadas necessariamente nesta ordem. A *chain* PREROUTING é utilizada para alterar o endereço de destino do pacote através do desvio de compilação NF_IP_PRE_ROUTING. Este NAT é denominado *destination* NAT (DNAT) de forma que a alteração de endereço do destino é realizada antes do roteamento do pacote . O DNAT também é realizado através da *chain* OUTPUT para as conexões realizadas localmente. O POSTROUTING tem a função de alterar o endereço de origem dos pacotes através do desvio de compilação NF_IP_POST_ROUTING denominado *source* NAT (SNAT), onde os pacotes são alterados após o seu roteamento. Existe um tipo especial de SNAT conhecido como MASQUERADE utilizado quando se tem uma conexão com endereço dinâmico, pois

através dele é possível alterar o endereço de origem dos pacotes de acordo com a interface de rede externa do dispositivo.

Explicitado as características básicas do NAT, neste ponto é relevante demonstrar como é feito a junção da tabela NAT com a tabela de estado conntrack, tornando-o *statefull*. Como se sabe, as regras que utilizam SNAT e MASQUERADE são criadas para os pacotes de ida, que saem do router, não havendo regras para a uma tradução justa do endereço IP quando os pacotes retornam para o router. Desta forma, para que seja possível esta comunicação, o NAT trabalha diretamente com a tabela conntrack, averiguando suas entradas a fim de reaver o correto endereço traduzido anteriormente pelo router e fazer seu correto encaminhamento. Ou seja, a regra SNAT/MASQUERADE é desfeita através da verificação da tabela de estados e o pacote retorna para a estação que o originou ao invés de voltar para o serviço local do firewall. Logo o NAT não funcionaria sem a tabela conntrack ativa.

Como mencionado no tópico anterior, decidiu-se tornar o iptables e principalmente o NAT do tipo *stateless* removendo o conntrack dos dispositivos para aplicá-lo ao ILM. Logo, para que fosse possível o NAT refazer o correto encaminhamento dos pacotes de ida quanto de volta, foi necessário criar regra por regra para cada conexão estabelecida no dispositivo.

Programa	Tabela e Chain	Expressão	Ação
Iptables	-t nat -I POSTROUTING	-s 192.168.8.10 –p tcp -o p4p1 -j SNAT	--to-source 192.168.4.2:5203
Iptables	-t nat - A PREROUTING	-s 192.168.4.2 -i p4p1 -j DNAT	--to-destination 192.168.8.10

Tabela 14 – Tradução de Endereços com SNAT sem o conntrack

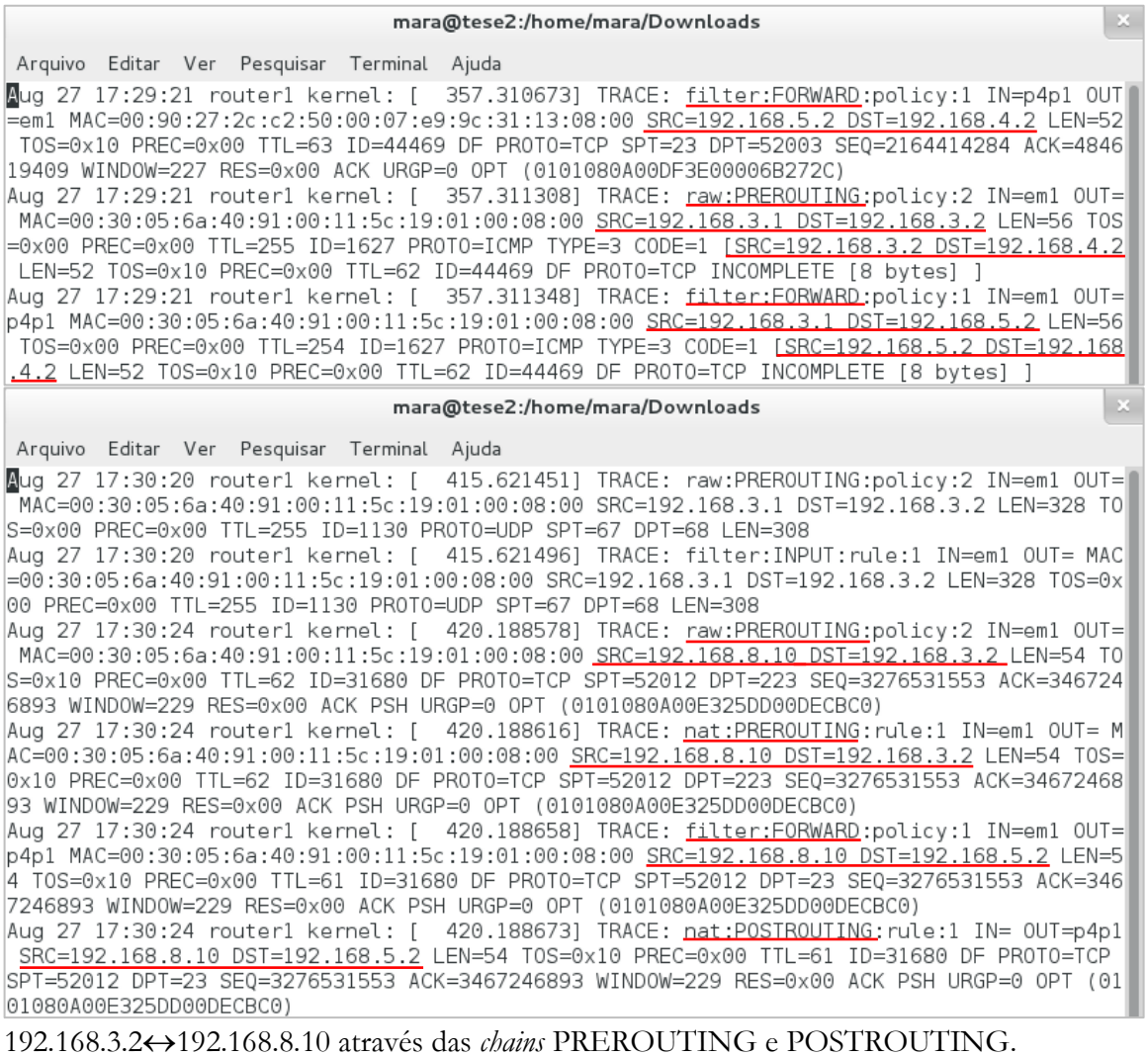
A Tabela 14, apresenta duas *chains* utilizadas para uma correta tradução dos endereços após a mobilidade. Neste caso, supõe-se que há uma ligação em curso proveniente do *Router 2* cujo IP é o 192.168.4.2 e porta 5203. Após a mobilidade este router adquiriu o endereço IP 192.168.8.10. O que a regra apresentada na linha 2 da tabela realiza, é um SNAT no Router 1, alterando a origem de todos os pacotes cujo o endereço IP é 192.168.8.2 para o IP e porta (192.168.4.2: 5203) da ligação anterior em curso. Esta alteração é direcionada para a interface interna do router que será reencaminhada para o Cliente 1. A regra da linha 3, altera o endereço IP de destino (192.168.4.2: 5203) proveniente da interface interna (Cliente 1) para o endereço IP real do Router 2 192.168.8.2. Nesta regra não é necessário determinar a porta de origem ou destino da ligação, pois quase sempre o NAT busca manter a porta da ligação. A exceção a esta regra é uma situação específica que será mencionada posteriormente. Ressalta-se que esta regra DNAT somente é necessária pela não utilização do conntrack pelo NAT. Este par de regras é criado para cada conexão estabelecida quando ocorrer a mobilidade de redes.

As regras de ida e retorno são necessárias também para o MASQUERADE utilizadas no ILM para uma perfeita comunicação exclusiva para o ambiente proposto. Os pacotes vindos da camada de aplicação do computador identificador, serão mascarados para o endereço localizador, o IP do router. E como regra de retorno, foi necessário realizar um DNAT para a entrega do pacote ao destino correto. As regras utilizadas são exibidas na Tabela 15.

Programa	Tabela e Chain	Expressão	Ação
Iptables	-t nat -A POSTROUTING	-s 192.168.4.2 -o em1 -j	MASQUERADE
Iptables	-t nat - I PREROUTING	-p tcp --dport 223 –j DNAT	--to-destination 192.168.6.2

Tabela 15 – Tradução de Endereços com Masquerade sem o conntrack

A Figura 22, também retirada do *debugging* do iptables (/var/log/kern.log) prova que após a retirada do conntrack o NAT passou a trabalhar em modo *stateless* e traduziu corretamente o endereço dos pacotes das ligações em curso. Primeiramente é visto ligações passando pela tabela raw e filter entre os IPs 192.168.5.2↔192.168.4.2 e 192.168.3.2↔192.168.4.2. Em seguida, na mesma ligação a tabela NAT é vista com a tradução dos endereços já realizada entre os IPs 192.168.5.2↔192.168.8.10 e



192.168.3.2↔192.168.8.10 através das *chains* PREROUTING e POSTROUTING.

Figura 22 - Depuração do NAT com o TRACE sem o conntrack

3.6. ILM-Server

O ILM-*Server* possui as mesmas características e funcionalidades do *Daemon Socket* servidor e cliente apresentadas no tópico 3.4.1. Ele fica sempre na escuta a espera de receber mensagens dos clientes quando a mobilidade *double jump* ocorrer. Em uma lista bidirecional as informações recebidas também são armazenadas (o endereço IP e portas anteriores e atuais dos nós) para que seja possível responder as requisições dos clientes em relação ao nó correspondente em comunicação prévia. O ILM-*Server* só é contactado quando o cliente do *Mobile Node* tenta realizar a ligação *socket* com o servidor do *Correspondet Node* e esta tentativa falha. O cliente então realizará a ligação para o ILM-*Server* solicitando o endereço atual do nó.

Capítulo 4

Avaliação da Solução

A avaliação da solução desenvolvida foi realizada a todo momento com o auxílio de diversas ferramentas para monitoramento e análises do tráfego de rede e system calls além de protocolos de rede originados na camada de aplicação. Dentre eles estão:

- Tcpdump
- Trace
- Strace
- Shorewall
- Wireshark
- Telnet
- SSH

Neste capítulo, supunha-se realizar um cenário de emulação de mobilidade de redes e testar a solução de acordo os parâmetros de latência do handover, análises paramétricas para medir o desempenho do ILM, impacto do ambiente construído para a solução em relação ao ambiente e perda de pacotes. Entretanto, devido as conclusões da solução apresentada os testes e análises serão voltados para clarificar a viabilidade da solução.

Ao longo do capítulo 3, foram apresentados os resultados parciais das fases desenvolvidas e das fases inviáveis criadas para o ILM com suas respectivas explicações. Neste momento, os teses e análises serão concentrados nos módulos socket e na tabela NAT. Para isso a emulação de uma ligação e mobilidade dos nós foi realizada.

Inicialmente o Cliente 2 (192.168.6.2) inicia uma ligação através do telnet para o Router 1 (192.168.3.2) através da porta 223 como apresentado na Figura 23.

Figura 23 - Ligação com o Telnet

Ao passar pelo Router 2 (192.168.4.2) um MASQUERADE é realizado e o endereço IP de origem deste pacote sai com o endereço atual da interface externa do Router 2 e segue para o Router 1. Esta tradução é necessária para simular a comunicação fim a fim entre o Router 1 e Router 2. Na Figura 24 através da interface p5p1 é possível avistar a ligação telnet vinda do Cliente 2 com destino ao Router 1 e logo mais abaixo através da interface em1

verifica-se que esta mesma ligação foi mascarada para o IP do Router 2. Uma ressalva importante a ser referida, é que todas as funcionalidades implementadas atuam de forma bidirecional.

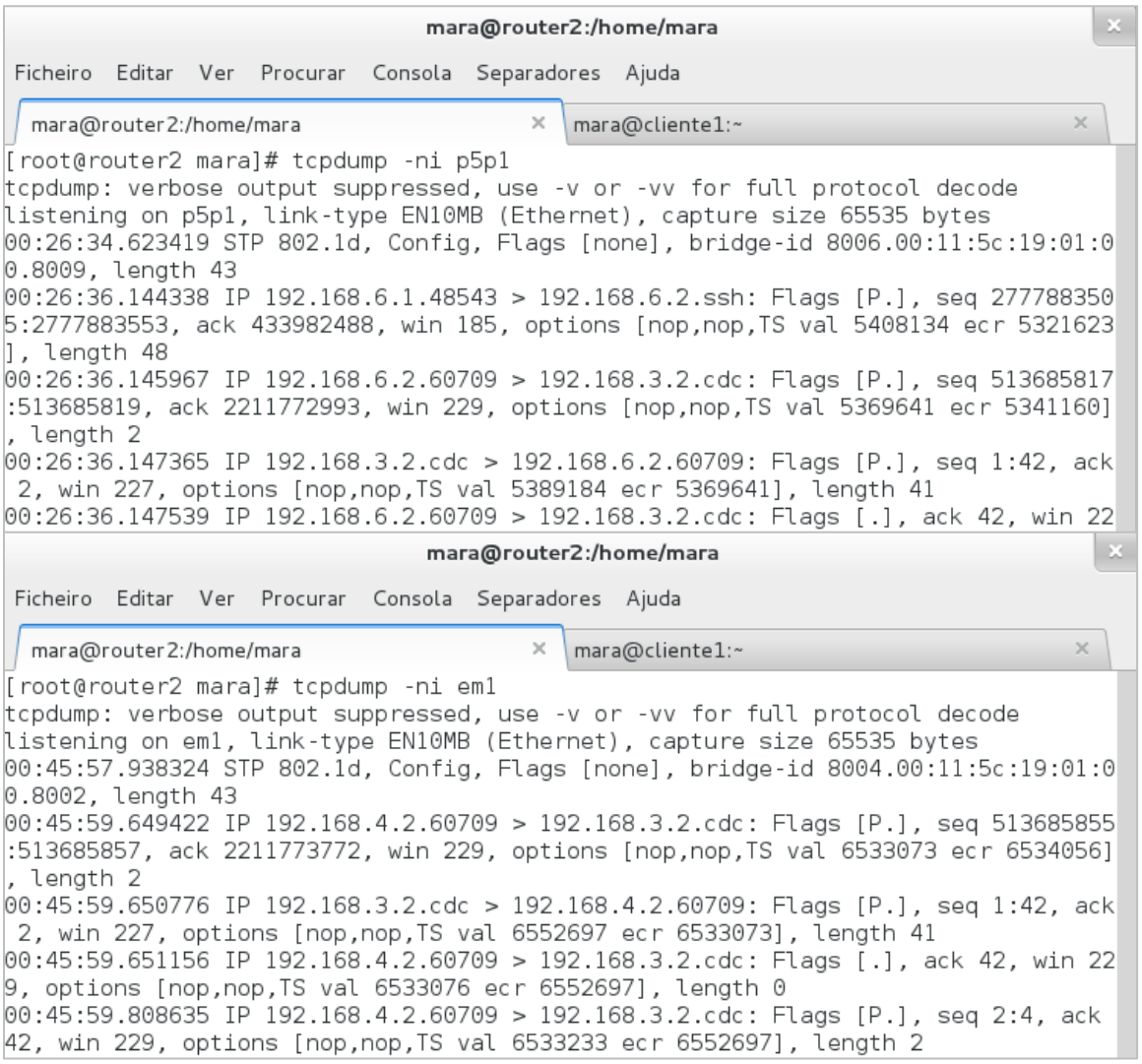


Figura 24 - Tradução IP com MASQUERADE

Quando os pacotes chegam ao destino, no Router 1, o DNAT previamente definido na Tabela 15 encaminha os pacotes da ligação para o Cliente 1 (192.168.5.2). Neste momento o *handshake* é realizado e a conexão estabelecida.

Simulando a mobilidade do Router 2, a troca da VLAN 4 para a VLAN 8 acontece e após 23 segundos (em média) o Router 2 adquire um endereço IP novo via DHCP na rede 8, o que pode justificar algumas diferenças (irrelevantes) de endereços IPs nas imagens dos testes aqui apresentados, que hora estão com o IP 192.168.8.2, outras com o o IP final 8.9, 8.10 e etc. O mesmo acontece para as imagens com os IPs da VLAN 4.

Em ambos terminais móveis o socket servidor está ativo e após a identificação do *handover* pelo *Mobile Node*, ele recebe do cliente socket o endereço IP atual, o endereço IP anterior e a porta utilizada na ligação. A Figura 25 mostra quando o servidor socket recebe estas informações e finaliza a conexão com o cliente imediatamente, pois ele somente deve se conectar em casos de mobilidade.

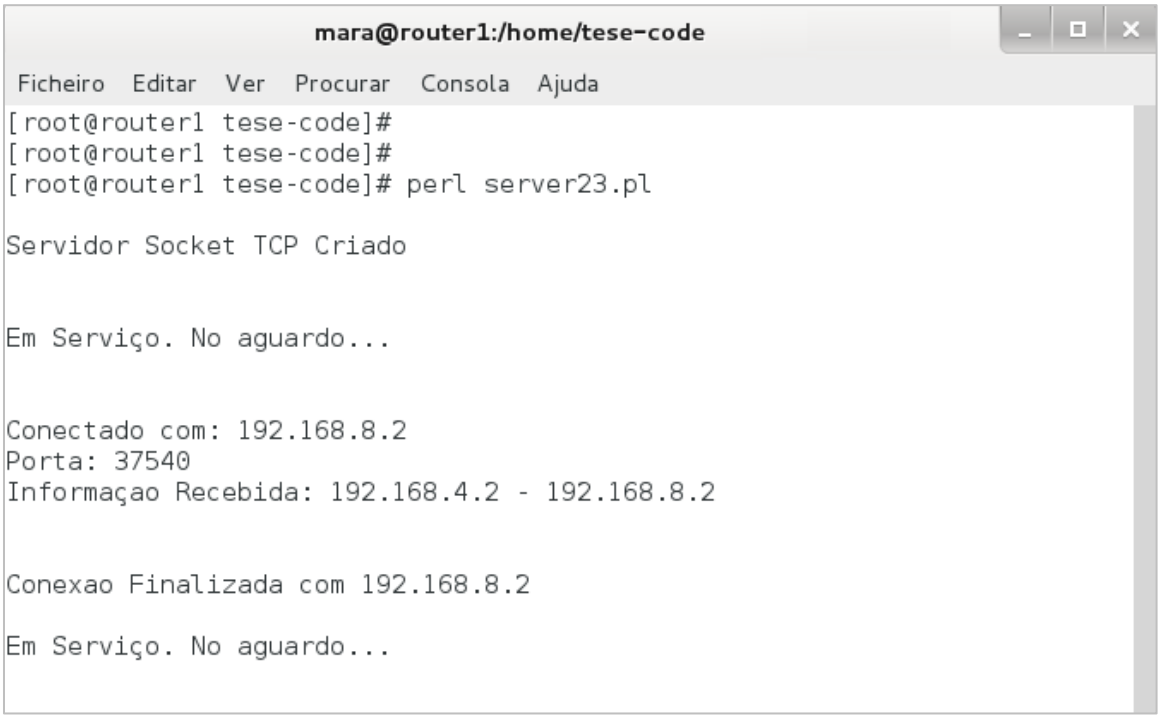


Figura 25 - Recebimento do Endereço IP pelo Socket Servidor

No Router 1, as regras SNAT e DNAT apresentadas anteriormente na Tabela 14 são criadas de acordo com essas informações (endereço IP e porta) através do script de tradução do ILM, visto parcialmente na Figura 26.

Figura 26 – Script de tradução de endereços ILM

Após o processo de *handover* e os módulos do ILM estarem praticamente finalizados, o tcpdump foi utilizado para monitorar o tráfego desta ligação. Primeiramente, foi constatado que o Cliente 2 não se deu conta da mobilidade do Router 2 e continuou enviando pacotes normalmente sem qualquer anomalia na sequência, *acknowledgment*, *window* e nos dois *timestamps* dos pacotes. A Figura 27 comprova esta teoria.

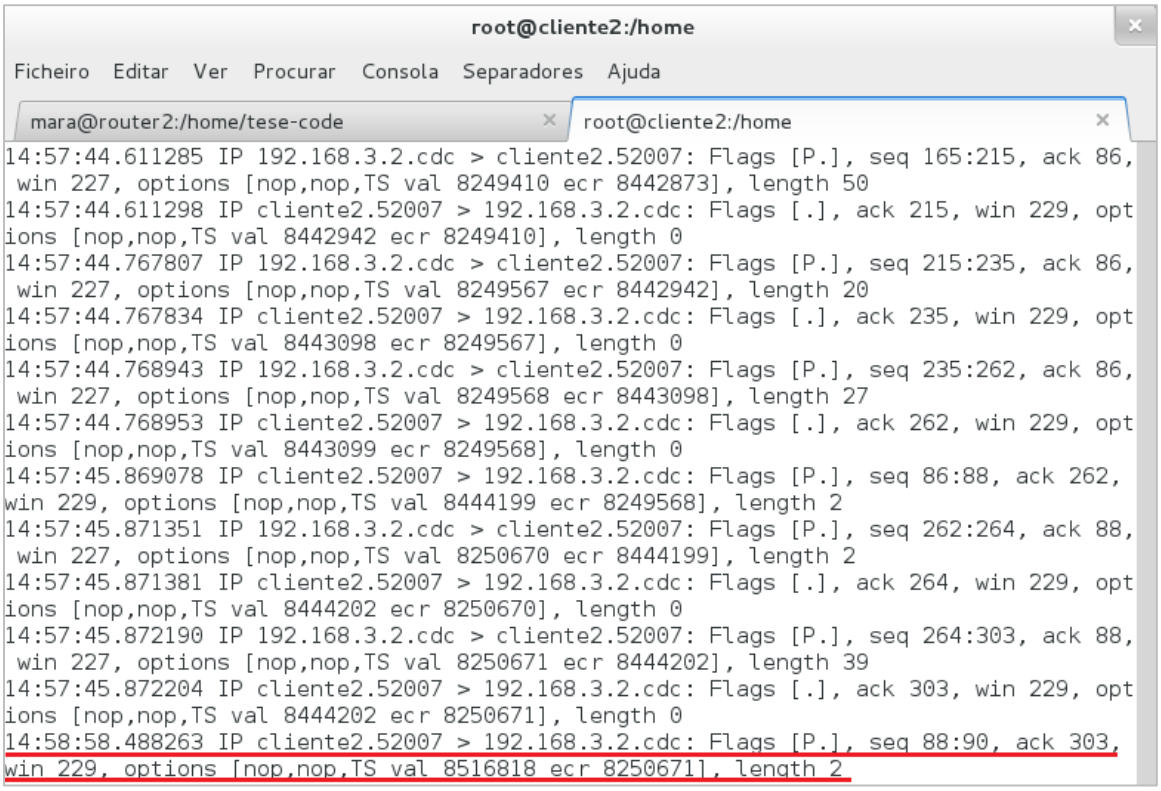


Figura 27 - Transparência da ligação após a mobilidade vista pelo Cliente 2

Esta solução ILM desenvolvida através de regras iptables e sockets apresentou uma instabilidade após a mobilidade de redes quando ocorre a tradução dos endereços devido a uma particularidade de mapeamento de portas. O padrão da tabela NAT procura modificar a conexão o mínimo possível, alterando apenas as regras definidas pelos usuários, assim as portas são remapeadas quando não existem regras específicas que dizem ao contrário. Entretanto há exceções, e uma delas é o Mapeamento de Portas de Origem Implícito [Russell, 2001]. Este mapeamento ocorre de forma automática e implícita quando necessário, mesmo quando não há utilização de regras NAT para uma determinada conexão.

Quando uma conexão se estabelece, uma porta é definida para ela permanecendo ocupada até o término da ligação e em alguns casos algum tempo após. Se uma conexão qualquer fazer o requerimento de uma porta que já esteja ocupada, em uso, o mapeamento de porta de origem implícito ocorre. Supõe-se a seguinte situação de masquerading vista em [Russell, 2001]:

(1) Estabelece-se uma conexão web de um dispositivo cujo IP é 192.1.2.3 vinda da porta 1025 para www.uc.pt com porta 80.

(2) É feito um MASQUERADE nesta conexão e os pacotes terão como origem o endereço IP 10.1.2.3.

(3) A máquina responsável pelo masquerading tenta se conectar com o endereço www.uc.pt porta 80 vindo do endereço 10.1.2.3 porta 1025.

(4) A tabela NAT realização uma alteração na porta de origem de 1025 para 1026 da segunda conexão a fim de evitar conflitos.

As portas do mapeamento de origem implícito são divididas em 3 classes (portas abaixo de 512, portas entre 512 e 1023 e portas de 1024 e acima), sendo que jamais uma porta será mapeada de forma implícita para um classe diferente, além do mapeamento do endereço da porta ser feito o mais próximo possível do seu endereço original disponível em sua classe.

Na solução ILM emulada, após a tradução de endereços o Cliente 2 continuou enviando mensagens para o Router 1, entretanto não houve qualquer resposta porque os pacotes recebidos foram descartados (DROP) em vez de serem encaminhados para o Cliente 1. Esta anomalia se dá pelo fato da regra SNAT utilizada, apresentada abaixo, especificar uma porta de ligação, a 55970, que já está ocupada pela própria conexão anterior a mobilidade não havendo assim maneiras de mapear a conexão de acordo com os requisitos determinados pela regra.

```
Iptables -t nat -I POSTROUTING -s 192.168.8.10 -p tcp -o p4p1 -j SNAT--to-source 192.168.4.2:55970
```

Devido a regra específica do SNAT determinando a porta da ligação, o mapeamento de origem implícito não ocorreu. A Figura 28 constata esta experiência. Depois da mobilidade até o término da captura, os pacotes passaram a ser descartados e consequentemente as requisições do Cliente 2 não foram respondidas.

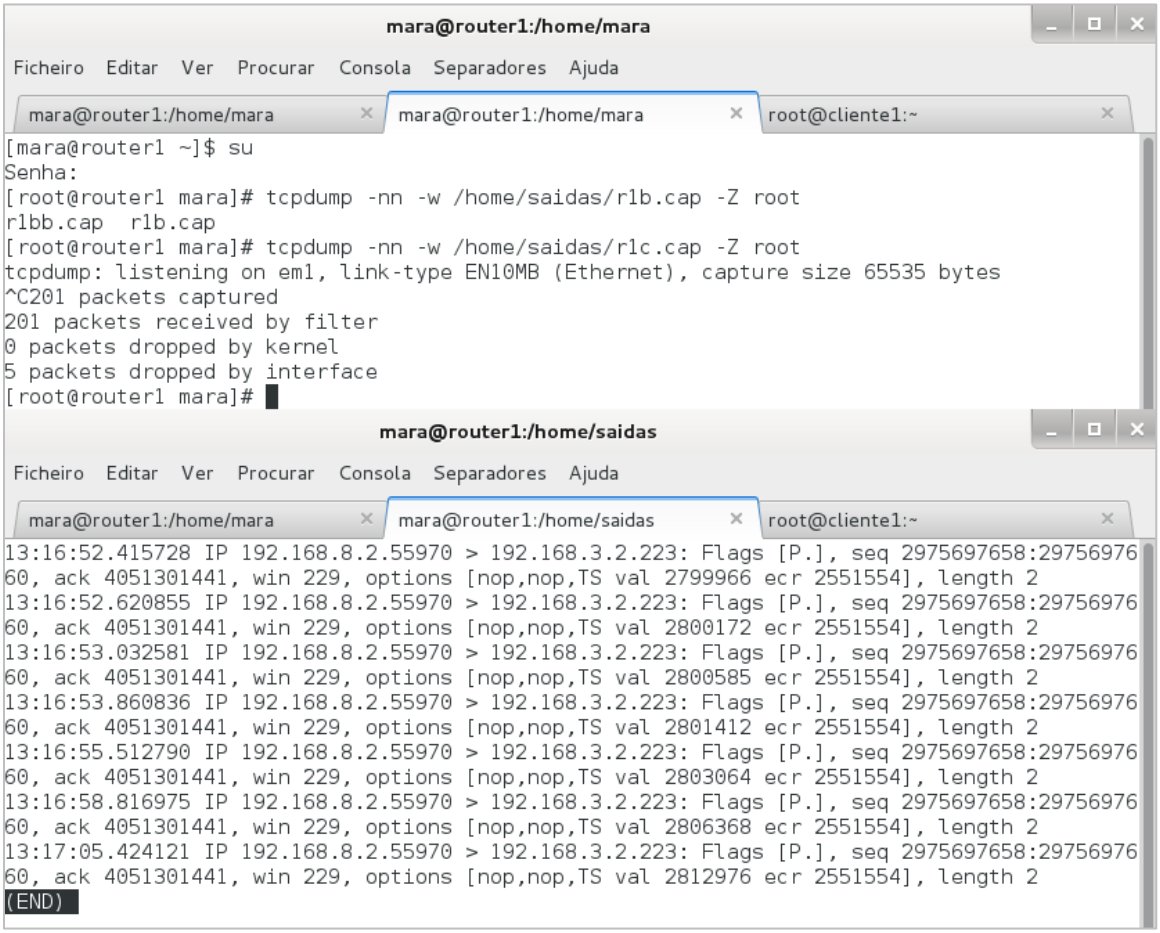


Figura 28 - Descarte dos Pacotes por Conflitos de Porta

Entretanto, se a regra SNAT determinar somente os endereços IPs sem uma porta específica, visto através da regra abaixo, a tradução acontece e os pacotes chegam ao Cliente 1 e retornam ao Cliente 2 passando pelos routers.

```
Iptables -t nat -I POSTROUTING -s 192.168.8.10 -o p4p1 -j SNAT--to-source 192.168.4.2
```

Sem a definição da porta na regra, acontece exatamente o mapeamento de origem implícito e a porta definida para a ligação passa a ser 1024 em vez de 52005. A sequência das Figuras 29, 30 e 31 seguintes, foram capturadas pela ferramenta de análise de tráfego Wireshark e mostram com detalhes este mapeamento implícito.

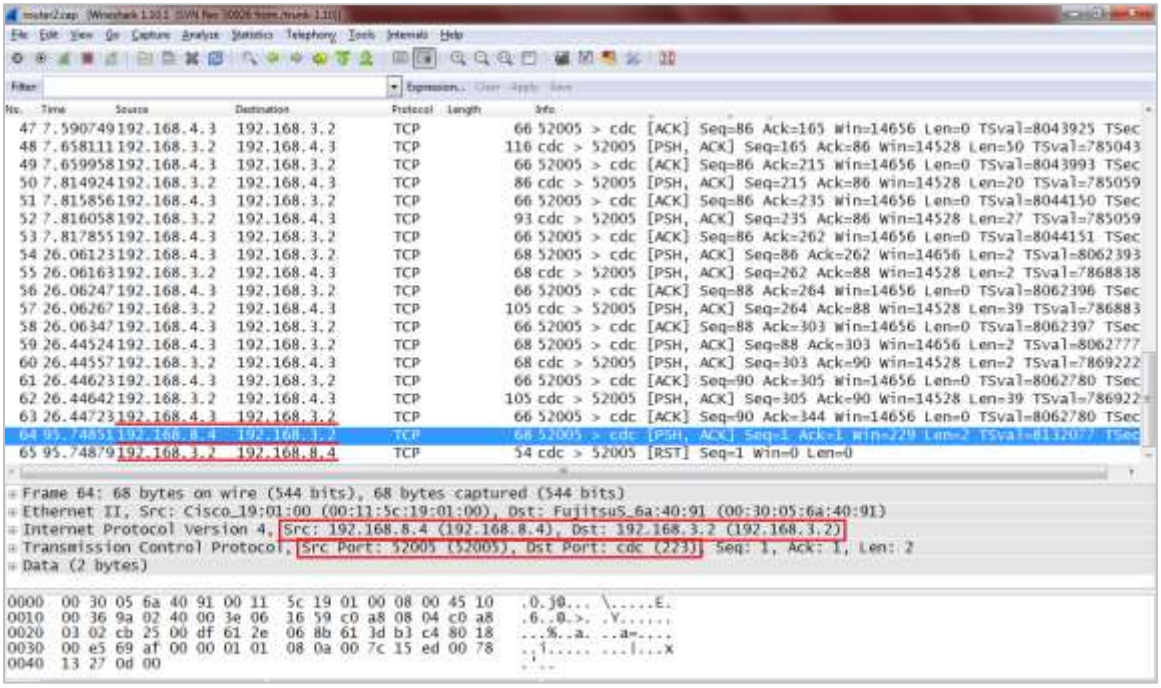


Figura 29 - Análise de tráfego do Router 1

A Figura 29 mostra o tráfego do Router 1 e exibe o exato momento em que ocorre a tradução dos endereços IPs. É possível perceber que o endereço das portas dos pacotes neste momento continua idêntico aos endereços dos pacotes anteriores, mas os números da *sequence* e do *acknowledgment* do pacote foram alterados. Esta alteração faz com que o NAT suponha que este pacote é o primeiro de uma nova ligação, razão para gerar um possível conflito entre as conexões.

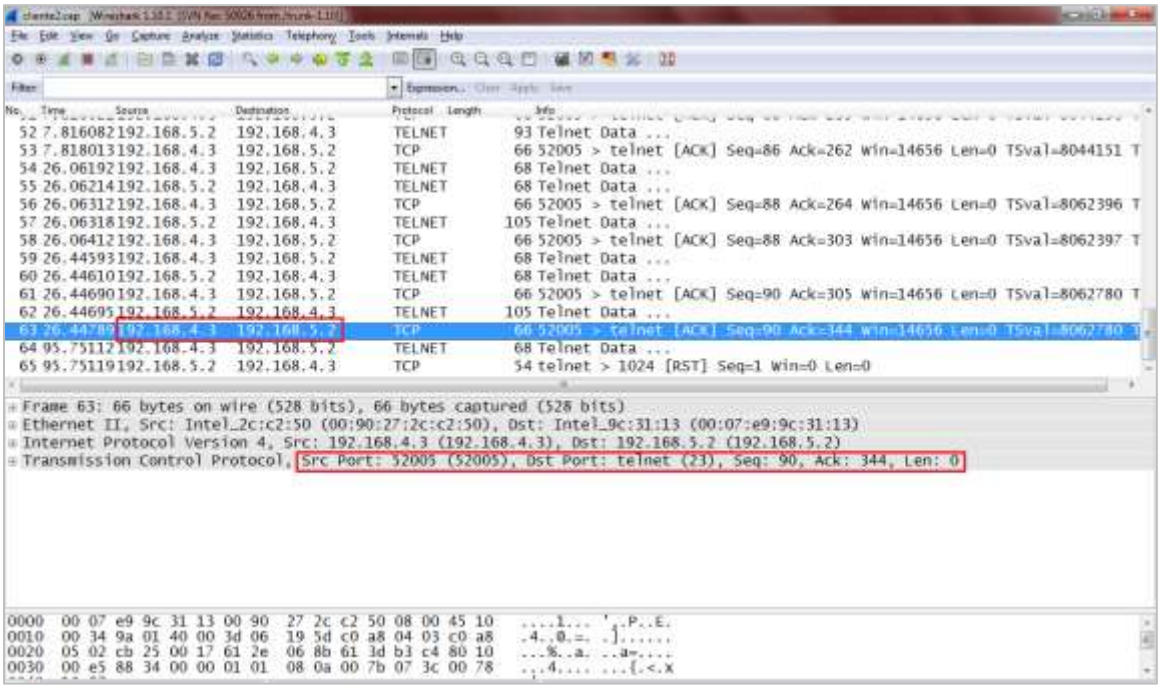


Figura 30 - Análise de Tráfego Cliente 1 Antes do Handover

A Figura 30 acima, exibe o tráfego do Cliente 1 antes da mobilidade. É possível identificar que o pacote recebido do Cliente 2, mascarado pelo o Router 2 como 192.168.4.3 possui a porta de origem como 52005 e mantém o número de sequência dos pacotes.

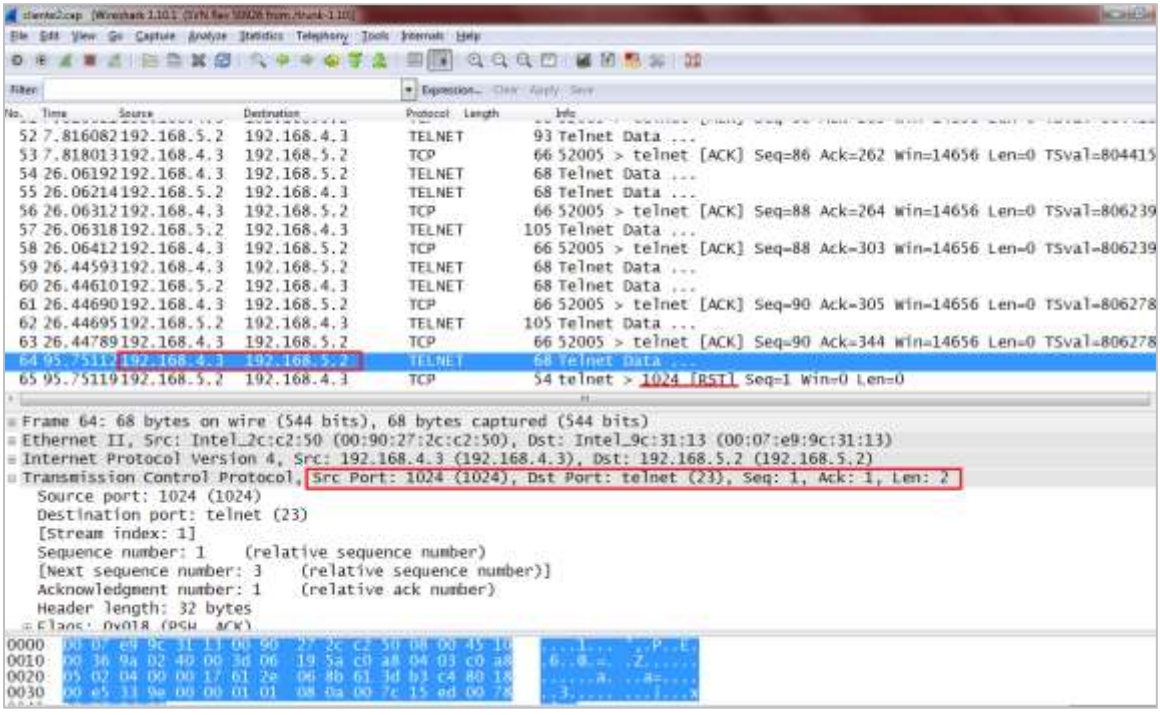


Figura 31 - Análise de Tráfego Cliente 1 Após o Handover

Após o handover, o Cliente 1 recebe o pacote com o novo número IP de origem do Router 2 mas traduzido para o endereço IP anterior à mobilidade 192.168.4.3. É neste momento em que é identificado a alteração na porta de origem, antes como 52005 e agora como 1024, sendo ele o primeiro número da ultima classe de portas (>=1024). Ou seja, o

mapeamento de origem implícito foi aplicado nesta conexão para evitar o conflito das ligações evitando o descarte das mensagens.

Também na Figura 31, o pacote TCP seguinte com origem no Cliente 1 envia uma mensagem com a *flag* RST, determinando o término da conexão. Esta interrupção não foi causada pelo telnet, sua ocorrência foi em um nível mais baixo, sendo causada ao nível do kernel, bloqueada no *address space* do kernel (endereçamentos que corresponde a um host ou periféricos) através da chamada **exit_group(1)**. A ferramenta de análise de *system calls* Strace apresentada na Figura 32 confirma esta conclusão. O comando utilizado é o que segue:

```
strace nc tcpdump -r cliente2.cap
```

```
mara@tese2:/home/mara/Downloads
Arquivo  Editar  Ver  Pesquisar  Terminal  Ajuda
fstat64(3, {st_mode=S_IFREG|0644, st_size=68412, ...}) = 0
mmap2(NULL, 68412, PROT_READ, MAP_PRIVATE, 3, 0) = 0xb7741000
close(3)                                = 0
open("/lib/libbsd.so.0", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\1\1\1\0\0\0\0\0\0\0\0\0\3\0\3\0\1\0\0\0\0J\205K4\0\0\0"..., 512) = 512
fstat64(3, {st_mode=S_IFREG|0755, st_size=48920, ...}) = 0
mmap2(0x4b853000, 52512, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x4b853000
mmap2(0x4b85f000, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xb) = 0x4b
85f000
close(3)                                = 0
open("/lib/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\1\1\1\3\0\0\0\0\0\0\0\0\3\0\3\0\1\0\0\0\220\307kK4\0\0\0"..., 512) = 512
fstat64(3, {st_mode=S_IFREG|0755, st_size=1999400, ...}) = 0
mmap2(0x4b6a3000, 1759932, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x4b6a3000
mmap2(0x4b84b000, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1a8) = 0
x4b84b000
mmap2(0x4b84e000, 10940, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x4b
84e000
close(3)                                = 0
mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb7740000
set_thread_area({entry_number:-1 -> 6, base_addr:0xb77406c0, limit:1048575, seg_32bit:1, contents
:0, read_exec_only:0, limit_in_pages:1, seg_not_present:0, useable:1}) = 0
mprotect(0x4b84b000, 8192, PROT_READ)   = 0
mprotect(0x4b69f000, 4096, PROT_READ)   = 0
munmap(0xb7741000, 68412)               = 0
write(2, "nc: ", 4nc: )                     = 4
write(2, "port number invalid: cliente2.ca"..., 33port number invalid: cliente2.cap) = 33
write(2, "\n", 1
)                       = 1
exit_group(1)                           = ?
+++ exited with 1 +++
[root@tese2 Downloads]#
```

Figura 32 – Debugging de Saída do Cliente 1 pelo STRACE

Como já foi previamente comprovado, o *debugging* to strace determina a razão para o término da ligação, sendo este o número inválido da porta da conexão.

Já o reinício dos número da sequência e do *acknowledgment* do pacote está relacionado com as diretivas das implicações de segurança para o NAT [Gont & Srisuresh, 2009], desenvolvidas pela *Internet Engineering Task Force* (IETF). O NAT por si só busca alterar o mínimo possível no cabeçalho dos pacotes, mas as diretivas de segurança fazem alterações nos cabeçalhos dos pacotes afim de afirmar a interoperabilidade da conexão, ou seja, garantir a capacidade de comunicação transparente. Isto justifica o restabelecimento dos campos seq e ack. A forma como isso é feito não foi esclarecida em [Gont & Srisuresh, 2009].

Protocolos mais sensíveis em relação à questões de segurança como o SSH rejeitam qualquer pacote com alteração brusca nos campos *sequence* e *acknowledgment*. Logo esta solução mesmo que se fosse viável, não poderia ser utilizada por este tipo de aplicação.

As diretivas de segurança do NAT, utilizam um gerador de número de sequência iniciais (ISN) a fim de dificultar a descoberta da sequência utilizada pelos pacotes ou mesmo

prever o número da sequência seguinte do pacote da conexão. Também foi desenvolvido pelas diretivas um gerador de data e hora (*Timestamp*) com as mesmas justificativas. Desta maneira, mesmo que se desejasse realizar uma alteração manual dos números de sequence e *acknowledgment*, com o intuito de englobar os protocolos de comunicação sensíveis à segurança a esta solução, esses geradores de sequência dificultariam ou praticamente impediriam seu prosseguimento.

4.1. Considerações sobre Aspectos de Segurança

Apesar dos resultados da solução desenvolvida apresentar efeitos inesperados, os mecanismos utilizados para o seu desenvolvimento abrem diversas lacunas nas questões de segurança que devem ser discutidos.

O primeiro ponto a ser destacado, é a não utilização da tabela de estados conntrack no iptables. Esta tabela também pode ser considerada como um mecanismo de segurança da ferramenta, já que para ela manter o estado das conexões, o tráfego de rede é examinado em sua totalidade, não permitindo qualquer alteração da ligação enquanto estiver registrada nesta tabela. Sem o conntrack, o risco de um pacote qualquer (proveniente de um intruso) que possua no campo *flag* a informação SYN ou mesmo ACK chegar a um dos filtros do firewall e comece imediatamente a fazer parte de algum canal de comunicação estabelecida é alto. Este método é tipicamente conhecido como ataque dos pacotes fragmentados, IP falsificado ou mesmo como *Spoofing*. Ataques *Main in the Middle* também são favorecidos nesta solução.

A separação do identificador e localizador aumenta o risco de interceptações e ataques na rede, visto que a cada *handover* uma mensagem de notificação com o novo número IP deve ser feita para manter a ligação em curso entre os dois nós. Neste caso pode haver um ataque conhecido como inundação, quando o atacante envia uma mensagem como se fosse uma atualização de nó e então passe a interceptar as ligações de um nó móvel. Ataques de furtos de endereços também está suscetível de ocorrer. Neste ataque, o intruso envia uma mensagem de atualização de um nó qualquer com o endereço de algum outro nó, desviando desta forma os pacotes do correto destino. Esses dois ataques são plausíveis de acontecer caso nenhum mecanismo de segurança seja implementado.

Na tentativa de mitigar estes problemas, a princípio seria necessário desenvolver um forte esquema de segurança para preencher as diversas lacunas apresentadas anteriormente. Este esquema deve ser bem analisado e estruturado já que as soluções de segurança apresentados nos protocolos para o gerenciamento da mobilidade desenvolvidos até o momento são os principais motivos para o mau desempenho da solução final.

Como pontos de atenuação destes problemas, a ação primordial a ser realizada seria o bloqueio de todos os pacotes não desejados através do firewall iptables. As as ligações que não são filtradas pelas regras pré estabelecidas devem ser descartadas.

Mais do que necessário, devido as lacunas mencionadas, um sistema de detecção de intrusos deveria ser implementado, já que o firewall não é capaz de detectar um ataque mais composto por diversos eventos. Uma solução possível seria a implementação de um sistema de detecção baseada em redes Petri, que é conhecido por possuir um forte poder para especificar ações ordenadas e tempo restritos destas ações, tornando-o bastante eficiente. Entretanto o desempenho da solução com a utilização deste mecanismo deve ser bem analisado.

E por fim, para que seja possível prover a segurança entre as ligações, algum mecanismo de criptografia se faz necessário.

Capítulo 5

Conclusão

As metodologias apresentadas nesta tese visou gerenciar a mobilidade de redes para comunicações fim a fim através da camada de transporte utilizando o firewall iptables. Na verdade, o iptables trabalha na camada de transporte, atuando na determinação e filtro das portas de origem e destino mas também trabalha na camada de rede atuando diretamente no tratamento dos endereços IP de origem ou destino sem levar em conta as portas.

A decisão em se utilizar o iptables, se deu pelo conhecimento das funcionalidades básicas de suas tabelas e *chains* que a princípio supunha ser suficiente para um correto tratamento dos pacotes após a mobilidade de redes.

Inicialmente, a marcação de pacotes através da tabela *mangle* seria o ponto forte da solução, porém os testes foram claros e demonstraram a perda da marcação realizada nestes pacotes após a necessária tradução dos endereços feitas pela tabela NAT. Desta maneira o foco da solução foi desenvolver a maioria das regras diretamente através da tabela NAT. Esta solução permitiria a sua utilização mesmo quando o desenvolvimento fosse implementado no ambiente real, fim a fim, pois as *chains* desta tabela garantem a tradução dos endereços dos pacotes gerados localmente através da *chain* OUTPUT, substituída pela *chain* PREROUTING no ambiente proposto por esta tese, possuindo o mesmo propósito e sintaxe. As outras *chains* possuem o funcionamento idêntico em ambas arquiteturas.

Em teoria a tradução dos endereços não ocorre em ligações em curso, entretanto a remoção de um mecanismo forte do iptables conhecido como conntrack foi realizada, tornando-o *stateless*. Assim, os principais problemas em se utilizar o NAT no ILM seriam resolvidos. Mas os resultados foram contrários às expectativas. Após o procedimento de mobilidade um mecanismo não previsto, denominado mapeamento de origem implícito foi ativado e a porta dos pacotes trocadas, causando assim seus descartes e inviabilizando a solução.

O mecanismo de atualização de endereços foi baseado em *sockets*, utilizado somente para atualização dos endereços após a mobilidade (*bind uptade*). O único ponto relevante a ser levantado, é a falta de segurança desta atualização, mencionada nos aspectos de segurança.

Supondo que a solução fosse viável, alguns erros agravantes da solução poderiam ser tratados pela forma em que o iptables trabalha. Supondo que ocorra a mobilidade de redes e o endereço IP altere de 192.168.4.2:123 para 192.168.8.2:123 e as respectivas regras NAT para a correta tradução dos endereços IP fossem criadas, e um terceiro nó passasse a adquirir o endereço IP 192.168.4.2:456 e entrasse em contato com o mesmo dispositivo da ligação em curso acima, este terceiro nó receberia normalmente os pacotes de retorno da ligação pelo fato das traduções serem feitas em conjunto de endereço IP e porta. Logo não haveria problema algum em se reutilizar o mesmo endereço IP. Já que a porta 123 estaria ocupada, e jamais seria utilizada pelo terceiro nó.

O NAT é um mecanismo que ainda possui certa instabilidade e apresentando erros em determinados casos com identificação de causas desconhecidas. Isso se deve ao fato de não se saber a fundo como o NAT trabalha. No caso específico desta solução o problema não foi originado pelo NAT, mas o real motivo por ele alterar os campos *sequence* e *acknowledgment* não estão tão claros.

A utilização de iptables para gerenciar a mobilidade de redes então se torna inviável pelo fato das análises apresentarem um comportamento não satisfatório, mas também pela instabilidade da tabela NAT, além das questões graves de segurança a serem mitigadas que provavelmente iria afetar o desempenho final do ILM.

4.2. Trabalhos Futuros

O gerenciamento da mobilidade de redes é um tema bastante discutido e atual onde vale a pena manter os estudos e desenvolvimentos para encontrar uma solução utilizável. Desta maneira, como trabalhos futuros, continuando na linhagem minimalista na busca de eliminar dispositivos excedentes mantendo uma ligação fim a fim, é proposto utilizar sockets internos do sistema (*Internal Sockets*), que trabalha diretamente ao nível do kernel através da API *Berkeley Sockets*. Lembrando que esta API é diferente do módulo *sockets* utilizado para o desenvolvimento da comunicação desta tese. Esta API possui funções específicas que dão suporte ao tratamento da mobilidade de redes. Dentre as mais poderosas estão o *socket( ), read( ), write( ), connet( ) e buffer_write( ).*

O que se propõe basicamente é desenvolver primeiramente um mecanismo *Keep Alive*, com o intuito de manter as ligações supostamente ativas na camada de aplicação por um período determinado de tempo, afim de que a aplicação não perceba o término da conexão quando ocorrer a mobilidade de redes.

Através das funções mencionadas anteriormente o restabelecimento da ligação pode ser feita através da função *connect( )* utilizando os mesmos parâmetros da ligação em curso anterior com o objetivo de não causar interrupções por parte da camada de aplicação. A função *buffer* pode ser utilizada para reenviar os pacotes perdidos neste perído de perda, identificação e reestabelecimento da conexão.

As questões de segurança desta possível solução devem ser analisados futuramente mas a utilização de chaves públicas e privadas pode ser viável.

Cronograma

Este tópico detalha as atividades desenvolvidas durante todo o período de investigação da tese. Ressaltando as atividades realizadas no primeiro semestre letivo apresentadas na qualificação defendida em 8 de Fevereiro de 2013, as atividades desenvolvidas ao longo do segundo semestre letivo de 2012/2013 e as respectivas considerações.

Estágios do Projeto	SET				OUT				NOV				DEZ			
Detalhamento da Proposta				■												
Estudo do Estado da Arte					■	■										
MIPv4,MIPv6, HIP, LISP,SHIM6					■	■	■	■								
Escrita Capítulo 2									■	■						
MIPv4									■							
MIPv6									■							
NEMO									■							
LISP										■						
AKARI										■						
mSCTP										■						
SIP										■						
Estudo Trabalhos Correlatos											■					
CB, NB, AKARI, TIPS											■					
Detalhamento da Arquitetura												■	■	■	■	
Especificação												■				
Definição do Ambiente												■	■			
Escrita da Arquitetura												■	■	■	■	
Estudo de Implementação											■	■	■	■	■	
Estudo Iptables											■	■	■			
Estudo sobre a Linguagem PERL											■	■	■	■		
Estruturação da Implementação														■	■	
Conclusão da Escrita														■	■	■

Tabela 16 - Cronograma das atividades do Primeiro Semestre Letivo

Como pôde ser visto na Tabela 16, durante primeira semana do semestre letivo, houve uma reunião com os orientadores para o detalhamento da proposta da tese. Nas três semanas seguintes, foi realizado um estudo aprofundado das arquiteturas criadas para solucionar problemas relacionados com a mobilidade de rede. Nas duas semanas seguintes, ao fim de Outubro e início de Novembro, a escrita do capítulo 2 (estado da arte) foi realizada. Em meados de Novembro, uma pesquisa sobre os trabalhos correlatos com a proposta da tese foi realizada, a fim de verificar se haveria trabalhos semelhantes e descobrir as diferenças entre a propostas. Seguidamente, foi realizado num período de uma semana, a escrita dos trabalhos correlatos. Durante todo o mês de Novembro foi estudado e proposto a arquitetura da solução, juntamente com os requisitos e definição do ambiente de desenvolvimento. À partir do momento em que se

soube da utilização da ferramente Iptables e a linguagem de programação Perl para o desenvolvimento do projeto, um estudo mais aprofundado foi realizado entre Novembro e Dezembro. No mês de Dezembro foi estruturado a implementação da solução e finalizado a escrita para a qualificação.

Será apresentado à seguir o cronograma do segundo semestre proposto durante a qualificação para o desenvolvimento da tese e o cronograma real praticado com suas respectivas justificativas quando se tratar do delongamento do prazo inicialmente proposto.

Estágios do Projeto	**FEV**				**MAR**				**ABR**				**MAI**				**JUN**				**JUL**				**AGO**			
Configuração do Ambiente																												
Configuração do Ambiente																												
Teste do Ambiente																												
Teste do Ambiente																												
Desenvolvimento Regras Iptables																												
Criar Regras Iptables																												
Testar Regras Iptables																												
Desenvolvimento Regras + Teste																												
Desenvolvimento ILM - BU																												
Testar BU																												
ILM – BU + Testes																												
Desenvolvimento ILM																												
ILM – BU + Regras Ipatables																												
ILM – BU + Ipatables																												
Testes da Solução																												
Testes da Solução																												
Conclusão da Escrita																												
Conclusão da Escrita																												

Tabela 17 - Cronograma das atividades do Segundo Semestre Letivo

O cronograma da Tabela 17, apresenta em azul o plano de trabalho planejado e em laranjado o realizado. É claramente perceptível que o cronograma proposto praticamente não foi seguido, mas isto se deu a diversos fatores de risco que previamente não foram identificados, ou sequer colocados em pauta que contribuíram diretamente para a entrega da tese no mês de Setembro.

A configuração do ambiente, planejada para o a segunda semana de Fevereiro se estendeu até Abril pelos seguintes fatores apresentados na Tabela 18.

Data	Fatores de Risco	Mitigação	Solução
11/02/13	Somente 1 monitor disponível para configurar e utilizar o ambiente	Alternância da utilização do monitor para a instalação do Sistema Operacional nos computadores.	Fevereiro: Foi disponibilizado pelo CIUC na segunda e semana de Fevereiro um monitor adicional.
11/02/13	Computadores sem leitor de CD/DVD e a não disponibilidade imediata de um pendrive bootável.	Simulação da configuração do ambiente iniciada no Cisco Packet Tracer para definir as etapas necessárias.	Fevereiro: Leitor de DVD foi disponibilizado pelo CIUC na segunda semana de Fevereiro.
18/02/13	2 computadores sem memória suficiente para a instalação do Fedora.	Testes do ambiente configurado no Cisco Packet Tracer para detectar possíveis problemas.	Fevereiro: Memórias de 512 MB foram disponibilizadas pelo CIUC na terceira semana de Fevereiro.
18/02/13	1 Computador sem qualquer atividade, sem ligação.	Configuração do ambiente em uma máquina virtual à partir do meu computador pessoal.	Março: Disponibilização do seu computador de trabalho no CIUC pelo co-orientador Pedro Vapi.
14/02/13	Necessidade de mais placas de rede para montar o Router 1 e Router 2 no ambiente.	Configuração e testes da comunicação Cliente e Router.	Março: 2 placas de rede foram disponibilizadas pelo CIUC.
19/02/13	1 dos computadores tinha um comportamento instável, desligando constantemente e perdendo todas as configurações realizadas.	Utilização deste computador como cliente, por se ter menos atividades e configurações mais simples de serem reconfiguradas.	Abril: Aquisição de um novo computador para incorporar o ambiente pelo orientador Fernando Boavida.
23/04/13	A entrada do conector do monitor do novo computador era DVI-D e o monitor disponível com entrada VGA.	-	Abril: Compra de um adaptador de monitor DVI-D para VGA para a instalação e configuração da máquina.

10/06/13	1 dos computadores apresentava comportamentos inadequados ao ligar-se desligar-se até o HD queimar no final de julho	Manter este computador como cliente e reiniciá-lo somente quando necessário.	Julho: Um novo HD foi disponibilizado pelo CIUC na ultima semana de Julho e as configurações foram refeitas.

Tabela 18 - Fatores de Risco do Projeto

Durante todo o mês de Fevereiro e Março o ambiente foi configurado, reconfigurado e testado na medida da disponibilidade dos equipamentos, entretanto sempre ao desligá-lo por completo e religá-lo as configurações eram perdidas e conflitos entre o Network Manager e o Service Network surgiam mesmo após tê-los desabilitados permanentemente. O ambiente se estabeleceu e manteve-se em correto funcionamento com a chegada do novo computador e com a remoção do computador cliente mais problemático. A reconfiguração de parte do sistema foi realizada novamente em julho pelo fato do HD de um dos computadores queimar.

Enquanto não era possível configurar e testar o ambiente por completo, foi dado início ao desenvolvimento do ILM-BU no mês de Março, realizado através de sockets em máquinas virtuais do meu computador pessoal. O único empecilho foi que os testes no ambiente real só puderam ser realizados ao fim de Abril e início de Maio.

Em Abril e Maio a criação do script das regras e testes do iptables foram iniciados mesmo sem o ambiente estar estabilizado. Nesta altura, foi possível detectar que a proposta inicial não era factível e nesta ocasião novas propostas de soluções foram pensadas e também desenvolvidas até o mês de Julho.

Os testes e análises se estenderam pelo mês de Julho e início de Agosto juntamente com a escrita da tese.

É importante ressaltar que este estágio foi realizado no departamento de Gestão de Sistemas e Infra-Estruturas de Informação e Comunicação (GSIIC)/Centro de Informática da Universidade de Coimbra (CIUC) da Universidade de Coimbra e sempre que foi solicitado algum equipamento a resposta foi imediata e o retorno o mais rápido possível, pois mesmo que algum equipamento necessário não estivesse disponível no momento, a busca para adquiri-lo não era cessada.

Referências

[Baptista, 2009] BAPTISTA, G., L., B. *Gerenciamento de Mobilidade e Tratamento Desconexão Baseado em SIP*, Fevereiro 2009.

[Barbado, 2007] BARBADO, W. 2007. *A Mobilidade na Internet Através do Protocolo HIP (Host Identify Protocol)*. Revista de Ciências Exatas e Tecnologia, v.2, nº2.

[Barr, 1997] BARR, G. *IO::Socket Documentation*. Web Page: *http://perldoc.perl.org/IO/Socket.html*. 1997.

[Campista at al, 2010] CAMPISTA, M., E., M., FERRAZ, L., H., G., MORAES, I., M., LANZA, M., L., D, COSTA, L., H., M, DUARTE O., C., M., B. *Interconexão de Redes na Internet do Futuro: Desafios e Solucões*, Simpósio Brasileiro de Redes de Computadores, Gramado, RS, Brasil, pp. 47-101, Maio 2010.

[Devarapalli, 2005] DEVARAPALLI, V., WAKIKAWA, R., PETRESCU, A. and THUBERT, P. *Network Mobility (NEMO) Basic Support Protocol*, IETF RFC 3963, Janeiro 2005.

[Eddy, 2004] EDDY, W. *At Layer does Mobility Belong?* Communications Magazine, IEEE, v.42, nº 10, p.155-159, 2004.

[Egevang & Francis, 1994] EGEVANG, K. and FRANCIS, P. *The IP Network Address Translator (NAT)*, IETF RFC 1631, Maio 1994.

[Eychenne, 2013] EYCHENNE, H. *Iptables Linux Manual Page*. Web Page: *http://ipset.netfilter.org/iptables.man.html*. 2013.

[Farinacci et al, 2012] FARINACCI, D., FULLER, V., MEYER, D. and LEWIS, D. *Locator/ID Separation Protocol (LISP)*. IETF draft-ietf-lisp-24.txt, Novembro 2012.

[Fávero, 2007] FÁVERO, A., L. *Metodologia para Detecção de Incoerências entre Regras em Filtros de Pacotes*. Janeiro 2007.

[Giust, 2011] GIUST, F. *Client-based and Network-based solutions for Distributed Mobility Management*. Setembro 2011.

[Gont & Srisuresh, 2009] Gont, F. and Srisuresh, P. *Security implications of Network Address Translators (NATs)*. IETF draft-gont-behave-nat-security-03, Outubro 2009.

[HARAI et al., 2008] HARAI , H, et al. *New Generation Network Architecture AKARI Conceptual Design*. Web Page: *http://akari-project.nict.go.jp/eng/concept-design/AKARI_fulltext_e_translated_version_1_1.pdf* . 2007.

[Johnson et al., 2004] JOHNSON, D., PERKINS, C. and ARKKO, J. *Mobility Support in IPv6*, IETF RFC 3775, Junho 2004.

[Kimura, 2008] KIMURA, B., Y., L. *TIPS: Uma Plataforma para a Mobilidade IP sobre a Camada de Transporte*. Maio 2008.

[Wall et at., 1991] WALL, L., CHRISTIANSEN, T., DFOY, B., ORWANT, J. 1991. *Programming Perl*. O.Reilly Media, Sebastopol, CA.

[Leffler et al., 1989] LEFFLER, S., MCKUSICK, M., LEFFLER, M., QUARTERMAN. J. 1989. *The Design and Implementation of the 4.3BSD UNIX Operating System*. Addison-Wesley Publishing Company, MA.

[Loureiro et al., 2012] LOUREIRO, C. A. H., TAROUCO, L., M., R., GRANVILLE, L., Z., BERTHOLDO, L. M. *Uma análise das implementações de protocolos IPv6 puros e híbridos para provimento de mobilidade em IPv6*, simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos, Ouro preto, MG, Brasil, Abril 2012.

[Martins, 2011] MARTINS, B., M. *Desacoplamento de Identificadores e Localizadores: Uma Comparação de Abordagens e Proposta de Arquitetura*, Fevereiro 2011.

[Moskotlvitz & Nikander, 2006] MOSKOWITZ, R. and NIKANDER, P. *Host Identity Protocol (HIP) Architecture*, IETF RFC 4423, Junho 2004.

[Pereira, 2008] PEREIRA, T., M. *Gerenciamento de Políticas de Qualidade de Serviço com Suporte à Mobilidade*, Junho 2008.

[Perkins, 2002] PERKINS, C. *IP Mobility Support for Ipv4*. IETF RFC 3344, Agosto 2002.

[Purdy, 2004] PURDY, G., N. 2004. *Linux iptables Pocket Reference*. O.Reilly Media, Sebastopol, CA.

[Riegel & Tuexen, 2003] RIEGEL, M., TUEXEN, M. Mobile SCTP. IETF RFC 2026, *draft-riegel-tuexen-mobile-sctp-01.tx*, Fevereiro 2003.

[Rosenberg et al., 2002] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M. and Schooler, E. *Session Initiation Protocol*. IETF RFC 3261, Junho 2002.

[Russell et. al, 1999] RUSSELL P., MAZIOLI G., FOLLE, A. 1999. *Firewall: Segurança de redes Linux*. Alfamídia Ltda, Porto Alegre, RS.

[Russell, 2001] RUSSELL R. *NAT HOWTO*. Web Page: *http://www.netfilter.org/documentation/HOWTO/pt/NAT-HOWTO.txt*. 2001.

[Stewart et al., 2000] STEWART, R., XIE, Q., MORNEAULT, k., SHARP, C., SCHWARZBAUER, H., TAYLOR, T., RYTINA, I., KALLA, M., ZHANG, L. and PAXSON, V. *Stream Control Transmission Protocol*. IETF RFC 2960, Outubro 2000.

[Tanenbaum, 1999] TANEMBAUM, ANDREW S. 1999. *Sistemas Operacionais Modernos*. Editora LTC, Rio de Janeiro, RJ.

[Wedlund & Schulzrinne, 1999] SCHULZRINNE, H., WEDLUND, E. *Application-Layer Mobility Using SIP*. Second ACM/IEEE International Conference on Wireless and Mobile Multimedia, 1999.

Anexos

server-socket.pl

```
#! /usr/bin/perl

# Universidade de Coimbra
# Mestrado em Engenharia Informática
# Ano 2011/2012 - 2012/2013
# Mara da Silva Benedito Martins - 2011101057 - msbm@student.dei.uc.pt

use IO::Socket::INET;
use strict;

my ($client, $resultado);

my $server_socket = IO::Socket::INET->new(
        LocalPort=>'7001',  # Porta que vai ficar em listening.
        Proto=>'tcp',           # Protocolo.
        Reuse=> 1,                      # Permite reutilização da porta.
        Listen=> 5                      # Número maximo de clientes conectados.
) or die "não foi possivel criar o socket.($!)\n";

#Socket Criado com sucesso.
print "\nServidor Socket TCP Criado\n";

# Socket Servidor em Escuta.
$server_socket->listen();

# Envio imediato das mensagens sem armazenamento no buffer.
$server_socket->autoflush(1);
```

```
print "\n\nEm serviço. No aguardo...\n";

# Condição para aceitar cliente socket se existir alguma mensagem a ser recebida.
while ($client = $server_socket->accept() )
{
        print "\nConnected from:", $client->peerhost();
        print "\nPorta:", $client->peerport(), "\n";

        # Condição looping para receber todas as informações do cliente.
        while (1)
        {
                print "\nAceitando Sockt do Cliente...\n";
                # Recebendo informações do cliente e armazenando na variável $resultado.
                $client->recv($resultado, 1024);
                print "\nA informação recebida foi: ", $resultado;

                # Armazenamento da informação em um arquivo texto.
                open (OUT, ">iprcv.txt");
                print OUT "$resultado";
                close (OUT);
                # Saindo da condição looping após o recebimento da mensagem.
                last;
        }
        # Remove qualquer seqüência posterior que corresponde ao valor atual de $/.
        chomp;
        # Termina a ligação com o cliente
        close $client;
        print "\nConnection Closed from:", $client->peerhost();
        print "\n\nEm serviço. No aguardo...\n";
}
```

cliente-socket.pl

```
#! /usr/bin/perl

# Universidade de Coimbra
# Mestrado em Engenharia Informática
# Ano 2011/2012 - 2012/2013
# Mara da Silva Benedito Martins - 2011101057 - msbm@student.dei.uc.pt



use IO::Socket::INET;
#use strict;
#use warnings;
use Socket;
use Sys::Hostname;

my ($IpAddrAtual, $IpAddr, $HostName);
my @AaA;
my @ip;
my @porta;
$i='0';

# Detecção do estabelecimento da ligação. Flag SYN-ACK.
system("/usr/sbin/tcpdump -n -i p2p1 'tcp[13] & 2 !=0  && tcp[13] & 16 !=0 ' >syn-ack.txt -Z root &");
print "\n";
sleep 2;

# Detecção do término da ligação. Flag FIN ou RST.
system("/usr/sbin/tcpdump -i p2p1 'tcp[tcpflags] & (tcp-fin | tcp-rst) !=0' >fin-rst.txt -Z root &");
```

 # Daemon Cliente.

```
while (1) {
        open (ARQ,'syn-ack2.txt');
        while(<ARQ>)
        {       $name=""; $porta=""; $aux1=""; $aux2="";


                # Expressões Regulares para abstrair o número IP e Porta das ligações em estabelecidas.
                if(/(\d+\.\d+\.\d+\.\d+\.\d+:)/)
                {
                        $name=$1;
                   $aux1=$name;
                        $aux1=~s/.\d+://;
                        chomp($aux1);
                        if ($name)      {
                                $name=~s/\d+\.\d+\.\d+\.\d+./$1/;
                                $aux2=$name;
                                chop($aux2);
                        }
                print "\n\nIP Correspondent Node (Server Socket): ", $aux1;
                        $ip[i]=$aux1;
                        $porta[i]=$aux2;
                        print "\nPorta: ", $aux2;
                        $i = $i+1;
                }
                else
                {print "\n\nSem Nova Ligação...\n\n"; }
        }
        open (OUT,'fin-rst2.txt');
        while(<OUT>){
                $out=""; $aux3=""; $aux4="";
```

```
        # Expressões regulares para abstrair o número IP e Porta das ligações finalizadas ou interrompidas.
        if(/(\d+\.\d+\.\d+\.\d+\.\d+:)/) {
                $out=$1;
                $aux3=$out;
                $out=~s/.\d+://;
                chomp($out);
                $aux3=$out;
                print "\nremover IP: ",$out;
                if ($out){
                        $out=~s/\d+\.\d+\.\d+\.\d+./$1/;
                        $aux4=$out;
                        chop($aux4);
                        $aux4=$out;
                        print "\naux3: ",$aux3;
                        print "\naux4: ",$aux4;
                }
        }
        # Eliminando o endereço IP de ligações finalizadas do do vetor.
        foreach $out (0 .. $ip)
        {
                delete $ip[$out];
        }
}
print $aux3;
print "\n\nporta: ", $porta[i];
print "\nip: ", $ip[i];

print "\nporta foreach: \n";
foreach (@porta)
{
        print "$_ \n";
```

```
}
print "\n",@ip;

# Detecta o endereço IP atual através da interface principal do computador.
$HostName = hostname();
$IpAddrAtual = inet_ntoa(scalar gethostbyname($HostName || 'localhost'));
print "\n\nMeu IP Atual: $IpAddrAtual";
sleep 1;

#Detecta endereço IP da interface desejada.
my $aux="";
open AX, "route -n | grep ^0.0.0.0 |";
chop($aux=<AX>);
close AX;
$aux=~s/.* ([^ ]+)$/$1/;
if ($aux) {
        open AX, "ifconfig $aux | grep 'inet ' |";
        chop($aux=<AX>);
        $aux=~s/.*inet (\d+\.\d+\.\d+\.\d+) .*/$1/;
        close AX;
}
$IpAddrAtual=$aux;

my $newIpAddr = inet_ntoa(scalar gethostbyname($HostName || 'localhost'));

my $aux2="";
open AX, "route -n | grep ^0.0.0.0 |";
chop($aux2=<AX>);
close AX;
$aux2=~s/.* ([^ ]+)$/$1/;
if ($aux2)
```

```
{
        open AX, "ifconfig $aux2 | grep 'inet ' |";
        chop($aux2=<AX>);
        $aux2=~s/.*inet (\d+\.\d+\.\d+\.\d+) .*/$1/;
        close AX;
        print "\n\nO IP dentro do Looping aux2:",$aux2;
}
$newIpAddr=$aux2;


# Condição para o endereço IP não tiver vazio ao entrar no IF. Necessário pelo DHCP demorar 23 segundos em média para atualização de IP.
if ($newIpAddr ne "")
{
        # Verifica se o dispositivo realizou o handover.
        if ($newIpAddr eq $IpAddrAtual)
        {
                print "\n Mobilidade não Detectada: ",$IpAddrAtual, "\n\n";
                sleep 2;
}
        else
        {
                print "\nMobilidade Detectada. Meu Novo IP: ",$newIpAddr, "\n\n";


                #Envia seu novo endereço IP para todos os dispositivos em comunicação através do vetor @ip.
                for ($j=0; $j <= $ip; $j++) {
                        $name = $ip[j];
                        $port = $porta[j];
                        print "\nnome ip array: ",$name;
                        print "\nporta array: ", $port;
```

```
                    # Determina o envio imediato das mensagens sem
armazenamento no buffer.
                    $client_socket->autoflush(1);


                    # Cria o Socket, mas sai dente looping em caso de erro em
vez de fechar o aplicativo completamente.
                    my $client_socket = IO::Socket::INET->new('PeerAddr' =>
$name, 'PeerPort'  => $port, Proto => 'tcp') or last "\nNao Foi criado o Socket $!\n";



                    print "\nSocket 2 foi Criado\n";


                    my $data = $newIpAddr . $porta;
                    # Envia as informações para o servidor socket.
                    $client_socket->send($data);
                    print "\nOs IPs sao diferentes. Inicial:",$IpAddrAtual, " e
Atual:",$newIpAddr, "\n\n";
                    sleep 2;
                    print "\n-> Novo IP enviado para o Servidor","\n";
                    sleep 2;
                    $IpAddrAtual=$newIpAddr;


                    # Finaliza a ligação do cliente socket atual.
                    $client_socket->close();
                    print "\nSocket no Cliente Finalizado\n\n";
                }
                    # Envia o novo endereço IP para o ILM-Server.
                    my $client_socket = IO::Socket::INET->new('PeerAddr' =>
'192.168.9.2', 'PeerPort'  => '7001', Proto => 'tcp') or last "\nNao Foi criado o Socket
$!\n";
                    $client_socket->send($data);
                    $client_socket->close();


        }
```

```
        }
        $port = "";
        $name = "";
        open (ARQ, '>syn-ack2.txt');
        print ARQ " ";
        close (ARQ);

        open (OUT,'>fin-rst2.txt');
        print OUT " ";
        close (OUT);
}
```

ilm-script-iptables.pl

```
#! /usr/bin/perl

# Universidade de Coimbra
# Mestrado em Engenharia Informática
# Ano 2011/2012 - 2012/2013
# Mara da Silva Benedito Martins - 2011101057 - msbm@student.dei.uc.pt

use strict;
use warnings;
use Socket;
use Sys::Hostname;

my ($IpLOCd, $newIpLOCd);

# Endereços IP fixos.
my $IpIDs ='192.168.5.2';
my $IpIDd ='192.168.6.2';

#Interface Interna e Externa
my $ethcliente = 'p4p1';
my $ethrede = 'em1';



#Ler o IP do Correspondent Node à partir de um arquivo
open (ARQUIVOIP, "<iprecebido.txt") || die "\nArquivo nao encontrado\n";
while (<ARQUIVOIP>)
{
        $IpLOCd = "$_";
        $aux1=$IpLOCd;
```

```
        $aux1=~s/.\d+ //;
        chomp($aux1);
        if ($IpLOCd)
        {
                $IpLOCd=~s/\d+\.\d+\.\d+\.\d+./$1/;
                $$aux2=$IpLOCd;
                chop($aux2);
        }
        #Porta da comunicação R1 <-> R2
        my $port = $aux2;
}
        close(ARQUIVOIP);


        print "\nEndereço IP LOCALIZADOR do Correspondent Node (ROUTER): ",$IpLOCd,"\n";
        print "\nEndereço da Porta de Comunicação: ",$port,"\n";


        #Detectar o Endereço IP para uma Interface
        #Pegando o IP da máquina Atual
        #my $Hostname = hostname();
        #my $IpLOCs = inet_ntoa(scalar gethostbyname($Hostname || 'localhost'));


        #Detectar o Endereço IP da Interface Desejada do Computador, endereço LOCALIZADOR (ROUTER).
        my $aux="";
        open AX, "route -n | grep ^0.0.0.0 |";
        chop($aux=<AX>);
        close AX;
        $aux=~s/.*([^ ]+)$/$1/;
        if ($aux)
        {
                open AX, "ifconfig $aux | grep 'inet ' |";
```

```
        chop($aux=<AX>);
        $aux=~s/.*inet (\d+\.\d+\.\d+\.\d+) .*/$1/;
        close AX;
}
my $IpLOCs=$aux;

print "\nMeu Endereço IP LOCALIZADOR: ",$IpLOCs,"\n";
sleep 2;

my @cont1 = ("1","$IpIDs","$IpLOCs","$port","$IpIDd","$IpLOCd");
#print "\nImpressao da tabela do ILM\n";
#print join(", ",@cont1, "\n");

# Ativando o roteamento NAT
my $numero = '1';
open (ARQ, ">/proc/sys/net/ipv4/ip_forward");
print ARQ "$numero";
close(ARQ);
print "\nAtivado o roteamento NAT\n";
sleep 2;

#Limpando as chains do iptables
system('/usr/sbin/iptables -F -t filter');
system('/usr/sbin/iptables -F -t nat');
system('/usr/sbin/iptables -F -t mangle');
system('/usr/sbin/iptables -F -t raw');
system('/usr/sbin/iptables -F -t security');
system('/usr/sbin/iptables -X');
system('/usr/sbin/iptables -X -t nat');
system('/usr/sbin/iptables -X -t mangle');
system('/usr/sbin/iptables -X -t raw');
```

```
system('/usr/sbin/iptables -X -t security');

#Criando as regras iniciais do iptables
system('/usr/sbin/iptables -t raw -I PREROUTING -j TRACE');
system('/usr/sbin/iptables -t raw -I OUTPUT -j TRACE');

#Aceitando todos os pacotes TCP e UDP
system("/usr/sbin/iptables -I INPUT -p ALL -i $ethrede -j ACCEPT");

# Regras Iniciais de Ida e Volta (Sem Conntrack) para serem iniciadas pelo Telnet e SSH

#Alterando a origem dos pacotes Identificadores (Cliente) para o IP da interface do Localizador (Router)
system('itables -t nat -A POSTROUTING -s $IpLOCd -o em1 -j MASQUERADE');

#Redirecionando todos os pacotes a serem utilizados pelo Telnet para o Identificador
system('iptables -t nat - I PREROUTING -p tcp --dport 223 –j DNAT --to-destination $IpIDd');

#Redirecionando todos os pacotes a serem utilizados pelo SSH para o Identificador
system('iptables -t nat - I PREROUTING -p tcp --dport 222 –j DNAT --to-destination $IpIDd');


#Daemon para detectar troca de IP do R1 alterar as regras iptables
while (1)
{
open (ARQUIVOIP, "<iprecebido.txt") || die "\nArquivo nao encontrado\n";
while (<ARQUIVOIP>)
{
$newIpLOCd = $_;
```

```
            print "Verificando Mobilidade de Redes\n";
            sleep 2;
        }
        close(ARQUIVOIP);
        if ($IpLOCd ne $newIpLOCd)
        {
            print "\nO IP do Correspondent Node mudou de ",$IpLOCd ," para
",$newIpLOCd, "\n";
            sleep 2;
            system('/usr/sbin/iptables -F -t filter');
            system('/usr/sbin/iptables -F -t nat');
            system('/usr/sbin/iptables -F -t mangle');
            system('/usr/sbin/iptables -F -t raw');
            system('/usr/sbin/iptables -F -t security');
            system('/usr/sbin/iptables -X');
            system('/usr/sbin/iptables -X -t nat');
            system('/usr/sbin/iptables -X -t mangle');
            system('/usr/sbin/iptables -X -t raw');
            system('/usr/sbin/iptables -X -t security');
            my $numero = '1';
            open (ARQ, ">/proc/sys/net/ipv4/ip_forward");
            print ARQ "$numero";
            close(ARQ);
            system('/usr/sbin/iptables -t raw -I PREROUTING -j
        TRACE');
            system('/usr/sbin/iptables -t raw -I OUTPUT -j TRACE');
            system('iptables -t nat - I PREROUTING -p tcp --dport 223 –j
DNAT--to-destination $IpIDd');
            system('iptables -t nat - I PREROUTING -p tcp --dport 222 –j
DNAT--to-destination $IpIDd');
            system('itables -t nat -A POSTROUTING -s $IpLOCd -o em1 -j
        MASQUERADE');
            system("usr/sbin/iptables -t nat -I POSTROUTING -s newIpLOCd
–p tcp -o $ethcliente -j SNAT --to-source IpLOCd:$port");
```

```
                system("usr/sbin/iptables -t nat - A PREROUTING -s IpLOCd -i
$ethcliente -j DNAT --to-destination newIpLOCd");
        }
        else
        {
                print "\nNão Houve Handover...\n";
        }
        sleep 3;
        $IpLOCd=$newIpLOCd;
}
```