# UMA AVALIAÇÃO DO PROTOCOLO HIP PARA PROVISÃO DE MOBILIDADE NA INTERNET

Lyno Henrique Gonçalves Ferraz

DEL / POLI / UFRJ

Projeto submetido para a obtenção do título de

**Engenheiro Eletrônico e de Computação**

ao Departamento de Eletrônica e de Computação

da Escola Politécnica da UFRJ

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
ESCOLA POLITÉCNICA
DEPARTAMENTO DE ELETRÔNICA E DE COMPUTAÇÃO

**Uma Avaliação do Protocolo HIP para Provisão de Mobilidade na Internet**

Autor:

---

Lyno Henrique Gonçalves Ferraz

Orientador:

---

Prof. Otto Carlos Muniz Bandeira Duarte, Dr.Ing.

Examinadores:

---

Prof. Luís Henrique Maciel Kosmalski Costa, Dr.

---

Prof. Miguel Elias Mitre Campista, D. Sc.

DEL

Março de 2010

*À minha família.*

# Agradecimentos

Agradeço aos meus pais e toda minha família pelo constante apoio e eterno carinho.

Ao professor Otto, meu orientador, e aos professores Luís Henrique e Miguel pelos diversos conselhos durante todo o trabalho.

Ao amigos do GTA, em especial Igor, Natalia, Marcelo e Carlo, pelos conselhos e pela grande ajuda.

Aos amigos da faculdade Dmitri, Felipe, Marcelo D., Danilo, Ramon, Marcelo L., Michel e Thiago por ajuda e apoio durante toda graduação.

A todos do departamento de eletrônica que construíram a minha formação como engenheiro.

A todos amigos que sempre estiveram do meu lado.

A todos que contribuíram direta ou indiretamente para a minha formação.

Resumo do Projeto Final apresentado ao DEL/UFRJ como parte dos requisitos necessários para a obtenção do grau de Engenheiro Eletrônico e de Computação.

# Uma Avaliação do Protocolo HIP para Provisão de Mobilidade na Internet

**Lyno Henrique Gonçalves Ferraz**

Março de 2010

Orientador: Otto Carlos Muniz Bandeira Duarte
Departamento: Engenharia Eletrônica e de Computação

O endereço IP (*Internet Protocol*) tem hoje a dupla função de "identificar" uma estação assim como especifica sua "localização" dentro de um domínio. Esta sobrecarga semântica do endereço IP traz problemas para estações móveis, onde uma estação muda de localização mantendo o mesmo endereço IP, e para estações e/ou serviços que se conectam à Internet por diferentes provedores de serviço (*Internet Service Provider* - ISP), onde uma mesma estação num mesmo local possui endereços IPs diferentes.

O *Host Identity Protocol* (HIP) surge como uma proposta para implementação de um sistema de nomeação melhor adaptado tanto para estações móveis quanto para estações com múltiplos domicílios (*multihoming*) na Internet, pois separa os dois papéis do endereço IP. Para realizar tal separação, HIP cria uma nova camada na pilha de protocolos TCP/IP, a camada de identificação. O endereço IP, o localizador na camada de rede, cumpre somente o papel de localizar as estações. A nova camada cumpre o papel de identificador da estação, de forma que, com o HIP, uma entidade pode manter sua identificação mesmo que ela mude de endereço IP ou mesmo que ela use diversos endereços IPs simultaneamente.

O objetivo deste projeto é avaliar experimentalmente o funcionamento do HIP em um ambiente real. Para tanto, são feitos experimentos usando estações equipadas com

interfaces de rede sem-fio. Esses experimentos buscam simular a caminhada de um usuário que, ao se deslocar, muda o seu ponto de acesso à Internet. Durante o trajeto, as mudanças na camada de rede são observadas quando se usa ou não o protocolo HIP, verificando o impacto do uso do HIP em cenários com mobilidade. Neste projeto, são avaliados tanto a conexão quanto o atraso ao se usar a pilha de protocolos TCP/IP e ao se usar a arquitetura HIP. Além disso, também foi medido o uso do HIP em redes cabeadas, para determinar a sobrecarga imposta pelo HIP quando este protocolo é utilizado em cenários sem mobilidade.

Palavras Chave: Rede de Computadores, Mobilidade, HIP

# Lista de Acrônimos

API : *Application Programming Interface*;

BEX : *Base Exchange*;

DHCP : *Dynamic Host Configuration Protocol*;

DHT : *Distribuited Hash Table*;

DNS : *Domain Name System*;

DSA : *Digital Signature Algorithm*;

DoS : *Denial of Service*;

ESP : *Encapsulated Security Payload*;

ESSID : *Extended Service Set Identifier*;

FDQN : *Fully Qualifield Domain Name*;

FTP : *File Transfer Protocol*;

HI : *Host Identifier*;

HIP : *Host Identity Protocol*;

HIT : *Host Identity Tag*;

HMAC : *Hash-based Message Authentication Code*;

HTTP : *Hypertext Transfer Protocol*;

IANA : *Internet Assigned Numbers Authority*;

IKEv2 : *Internet Key Exchange version two*;

ICMP : *Internet Control Message Protocol*;

IP : *Internet Protocol*;

IPSec : *IP Security*;

LDAP : *Lightweight Directory Access Protocol*;

LSI : *Local Scope Identifier*;

MitM : *Man in the Middle*;

MAC : *Medium Access Control*;

MODP : *More Modular Exponential*;

NAT : *Network Address Translation*;

NAI : *Network Address Identifier*;

ORCHID : *Overlay Routable Cryptografic Hash Identifier*;

PPP : *Point-to-Point Protocol*;

PCMCIA : *Personal Computer Memory Card International Association*;

QoS : *Quality of Service*;

RSA : *Rivest-Shamir-Aldeman*;

RFC : *Request for Comments*;

RTP : *Real-time Transport Protocol*;

RTSP : *Real Time Streaming Protocol*;

RTT : *Round Trip Time*;

RVS : *Rendezvous Server*;

SA : *Security Association*;

SHA : *Secure Hash Algorithm*;

SHA-1 : *SHA Family 1*;

SHIM6 : *Level 3 Shim for IPv6*;

SMA : *Secure Mobile Architecture*;

SPI : *Security Parameter Index*;

SSH : *Secure Shell*;

TCP : *Transmission Control Protocol*;

TCP/IP : *TCP IP Protocol Suite*;

TLV : *Type, Length and Value*;

UDP : *User Datagram Protocol*;

VoIP : *Voice over IP*;

# Sumário

# Lista de Figuras

# Lista de Tabelas

# Capítulo I

# Introdução

## I.1 Motivação

Quando a pilha de protocolos TCP/IP foi criada no final da década de 70 e início da década 80, não se pensava na dimensão que a Internet iria tomar [1]. Alguns cenários como a mobilidade de estações e estações com múltiplos domicílios (*multihoming*) na Internet não foram previstos. A pilha TCP/IP foi pensada para estações estáticas e as estações se conectavam à Internet por um único provedor de serviço sem nenhuma redundância. Naquela época, o papel do endereço IP de localizar as estações para o roteamento de pacotes também era suficiente para identificá-las de forma única, pois as estações raramente moviam-se. Quando a alocação dinâmica de IP foi criada com o *Point-to-Point Protocol* (PPP) [2] e *Dynamic Host Configuration Protocol* (DHCP) [3], a identificação única de estações por endereços IPs foi quebrada, pois uma mesma estação poderia ter IPs diferentes, dependendo do momento em que fosse feita a conexão com a rede. O conceito de identificação pelo endereço IP ficou ainda mais frágil com a criação do *Network Address Translation* (NAT) [4], que multiplexa várias estações com somente um endereço IP [5], permitindo um único endereço IP "identificar" diversas estações diferentes.

Além da natureza das estações finais, a evolução da Internet mudou também o tipo de

usuários que acessam a rede. No início, a Internet era utilizada apenas por uma pequena comunidade de usuários que confiavam uns nos outros. Posteriormente, com a comercialização da rede, o número de usuários aumentou significativamente e a confiança entre eles foi afetada. Hoje, existem os mais variados tipos de usuários, inclusive aqueles que desejam efetuar ações maliciosas. Para prover segurança entre as conexões são necessários mecanismos criptográficos.

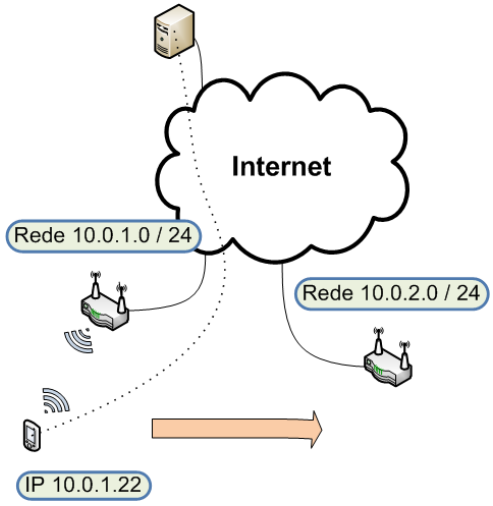
Em suma, o ambiente da Internet, a natureza dos usuários e os dispositivos mudaram desde sua criação. Com isso, requisitos como mobilidade, múltiplos domicílios de uma mesma estação na Internet e segurança surgiram. As soluções existentes para esses problemas na arquitetura TCP/IP são pouco eficientes e, então, se faz necessária uma mudança arquitetural na Internet para dar suporte aos novos cenários que existem na rede.

### I.1.1 Mobilidade

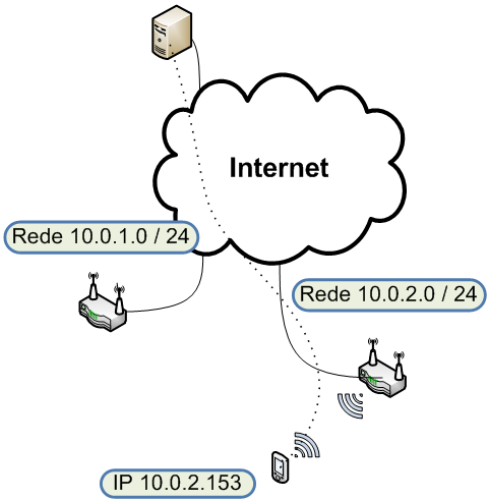
Neste trabalho, a mobilidade é considerada como o fenômeno de uma entidade mover-se mantendo ativo o contexto de comunicação [5], como mostrado na Figura I.1. Uma estação pode modificar o ponto de conexão com a rede livremente, e todos os contextos de conexão permanecerão ativos. Os processos não percebem a mobilidade, entretanto podem sentir uma diferença na qualidade de serviço.

Para compor um cenário real, assume-se, neste trabalho, que existem estações móveis conectadas a uma rede fixa. Isto representa o caso de uma estação móvel que acessa um servidor na Internet. Além disso, os prefixos dos endereços da camada de rede são determinados pela estrutura da rede. Ou seja, a topologia da rede determina qual endereço IP é atribuído a cada nó em cada momento. Isso reflete o fato que, em redes grandes, é importante manter os prefixos do endereço IP consistentes com a topologia para preservar as tabelas de roteamento com um tamanho gerenciável e sem redundâncias. Como consequência, todas as vezes que uma estação se move, seu endereço de rede também muda. Então, para continuar a comunicação a estação deve ser capaz de sinalizar essa

modificação de endereços aos pares ativos.




(a) Estação móvel com endereço IP igual a 10.0.1.22 conectado a um ponto de acesso da rede 10.0.1.0/24 inicia a mobilidade.




(b) A estação móvel muda sua localização topológica, saindo do alcance da rede 10.0.1.0/24 e entrando no alcance da rede 10.0.2.0/24, e, consequentemente, muda seu endereço IP.

Figura I.1: Mobilidade. A estação móvel muda de ponto de acesso e consequentemente de rede.

### I.1.2 Multihoming

A expressão *multihoming* ou multidomicílio se refere a uma situação na qual uma estação possui múltiplos caminhos paralelos de comunicação com a rede Internet [6]. Existem duas situações que caracterizam o multidomicílio. A primeira situação, chamada de multidomicílio de estações (*host multihoming*), é caracterizada pela existência de uma estação com diversas interfaces de rede, cada uma com um endereço IP diferente e independente. Com isso, um mesmo serviço provido por essa estação pode ser alcançado através de endereços IP diferentes. A segunda situação, chamada de multidomicílio de sítios (*site multihoming*), é caracterizada quando um sítio possui diversas rotas de saída para a Internet. Consequentemente, as estações deste sítio, mesmo com apenas uma interface de rede,

podem possuir endereços IP diferentes. A Figura I.2 mostra um exemplo de multidomicílio de sítio e de estação.

O multidomicílio (*Multihoming*) é um recurso muito importante e amplamente utilizado na Internet. O motivo de sua grande utilização é evitar o ponto único de falha. Para isso são criados caminhos redundantes de acesso a Internet como, por exemplo, por diferentes provedores de acesso. Entretanto, o uso do multidomicílio pode causar sérios problemas. Talvez um dos problemas mais graves é o grande aumento da tabela de roteamento. Isso ocorre quando o sítio ou estação que usa multidomicílio precisa anunciar as rotas para os provedores de serviço (*Internet Service Provider* - ISP).

Os endereços IPs são utilizados como identificadores e, portanto, devem ser únicos para as estações (ou outros elementos de rede como servidores). Entretanto, os endereços não são agregáveis para todos os provedores de serviço. Portanto, os provedores de serviço que não agregam os endereços IPs devem fazer o anúncio de rotas mais específicas. Ou seja, esses roteadores que não podem agregar a rota, devem inserir mais uma linha na tabela de roteamento para cada um desses sítios ou estações com multidomicílio. Essas rotas específicas serão incluídas em todos os roteadores até que os endereços possam ser agregados novamente. Como o multidomicílio (*multihoming*) é amplamente utilizado, então o número de linhas nas tabelas de roteamento cresce significativamente.

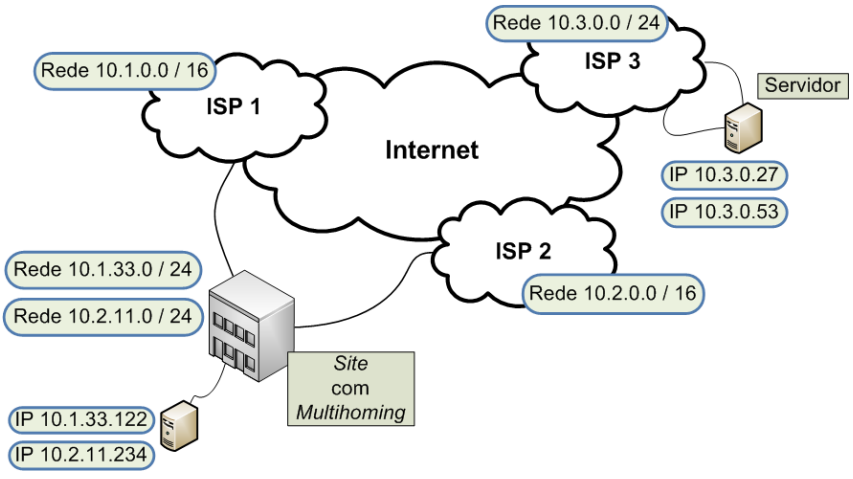
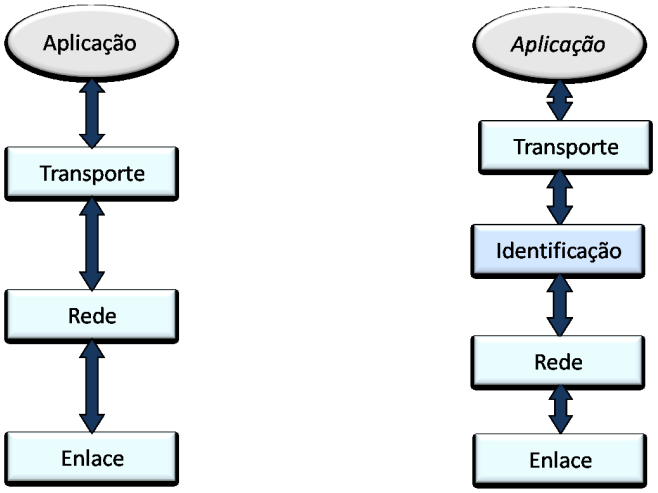



Figura I.2: Estações com multidomicílio (*multihoming*).

## I.2 Protocolo de Identificação de Estação - HIP

Para resolver os problemas apresentados, o Protocolo de Identificação de Estação (*Host Identity Protocol* - HIP) ataca a ambiguidade do IP [7]. O HIP faz a separação dos papéis do IP, distinguindo a função de localizador da função de identificador. Para realizar essa separação, uma nova camada na pilha de protocolos TCP/IP é criada, chamada de camada de identificação, como mostrado na Figura I.3. O IP continua a ser responsável pela localização e a função de identificar estações passa para a camada de identificação.




(a) Camadas convencionais da pilha TCP/IP.

(b) Camadas da pilha TCP/IP com o HIP.

Figura I.3: Arquiteturas de rede no modelo convencional e no modelo com o HIP.

A camada de identificação é inserida entre a camada de transporte e a camada de rede, como mostrado na Figura I.3(b). A camada de transporte passa a utilizar os dados da camada de identificação na conexão, ao invés de utilizar os dados da camada de rede diretamente. A camada de identificação, por sua vez, liga-se com a camada de rede. Portanto, quaisquer alterações nas camadas inferiores à camada de identificação não são percebidas pelas camadas superiores (transporte e aplicação).

Ao se tirar a função de identificação do IP, é necessário criar um método para se fazer o mapeamento dos endereços IP com as identidades de forma dinâmica e segura. Na

arquitetura atual, não existe a necessidade de fazer esse mapeamento, pois o IP exerce ao mesmo tempo o papel de localizador e identificador. Entretanto, ao separar a identificação da localização do IP, deve-se garantir a validade do endereço associado ao identificador.

No HIP, a identificação das estações é feita por meio de um identificador de natureza criptográfica, formado por uma chave pública. O motivo de o identificador ser uma chave pública é a possibilidade de utilização dessa chave pública para autenticação do usuário, assim como para a geração de uma chave de sessão e para encriptação de mensagens em uma conexão.

O identificador do HIP pode ser utilizado em vários mecanismos de autenticação já existentes como o IKEv2 [8]. Apesar disso, a nova arquitetura apresentada introduz um novo protocolo (HIP) e uma troca criptográfica fim-a-fim própria, chamada de *HIP Base exchange* (BEX). Os protocolos definidos pelo HIP provêem mecanismos de confiança, mobilidade, *multihoming*, renumeração dinâmica de IP, tradução e transição de endereços IP, além de evitarem certos tipos de ataque de negação de serviços, como será descrito nas seções seguintes.

Cada estação possui ao menos um identificador. Embora uma mesma estação possa possuir mais de um identificador, como cada identificador é único, duas estações jamais possuirão algum identificador em comum. Além disso, os identificadores podem ser ou não publicados em sistema do tipo DNS. A não publicação de um identificador caracteriza uma forma de prover anonimato.

### I.2.1 Segurança

O Protocolo de Identificação de Estação (*Host Identity Protocol* - HIP) possui especificações próprias de segurança para evitar problemas com a separação do localizador do identificador. De fato, tanto a mobilidade quanto o multidomicílio (*multihoming*) podem causar vários problemas de segurança se não existirem mecanismos específicos na camada de identificação. Uma separação de localizador e identificador implica que mudanças de localizador devem ser notificadas ao outro par sempre que ocorrerem, para que o caminho

entre os dois nós pares seja sempre mantido corretamente. A mensagem que faz essa notificação pode ser utilizada para realização de ataques na rede.

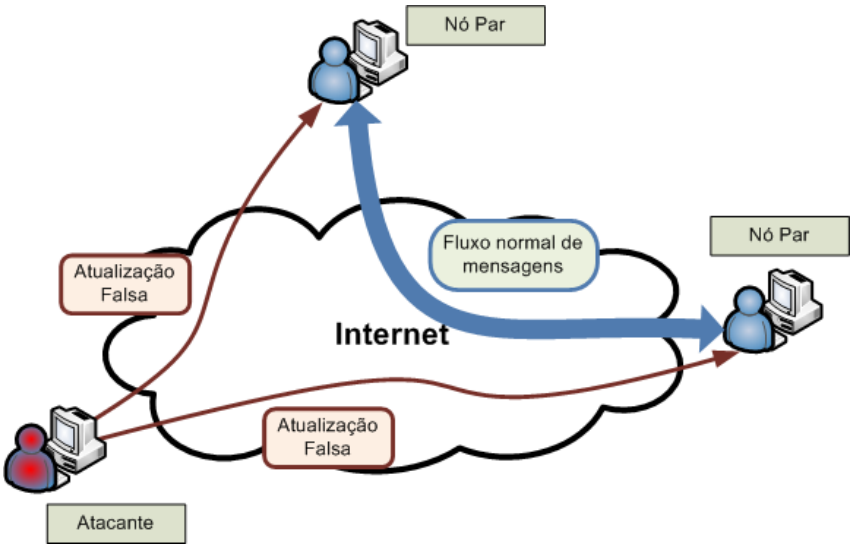
Existem dois problemas básicos: o furto de endereço, mostrado na Figura I.4, e a inundação de endereço, mostrada na Figura I.5 [5]. No furto de endereço, o atacante envia uma mensagem de atualização de um endereço que já está sendo utilizado por outro nó. Com isso, o nó atacante faz com que as mensagens que estavam sendo enviadas para o nó alvo sejam desviadas. Assim, o atacante pode realizar um ataque de mascaramento (*Masquerade*), homem no meio (MitM) ou negação de serviço (DOS). Na inundação de endereço, o nó malicioso anuncia que mudou de endereço para os seus nós pares, indicando como novo endereço IP o endereço IP do nó alvo do ataque, com o objetivo de fazer os nós pares enviarem mensagens para o nó alvo. Os nós pares acreditam que o nó se moveu para o endereço do nó alvo e inundam o enlace do nó alvo com pacotes não desejados.

Esses ataques são possíveis apenas se mensagens de atualização de endereço forem utilizadas sem algum mecanismo de segurança. O HIP propõe mecanismos específicos para evitar cada um desses ataques. No furto de endereço, a estação par, sempre que receber uma mensagem de atualização deve verificar se essa mensagem foi realmente enviada pelo dono do endereço antigo. Isso pode ser feito pela autenticação da assinatura da mensagem de atualização.

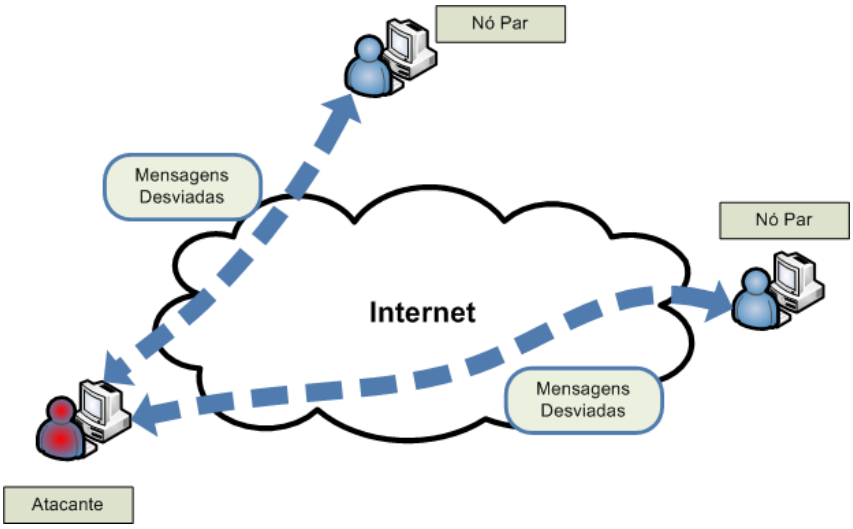
## I.3 Objetivos

O trabalho desenvolvido tem como objetivo avaliar experimentalmente o funcionamento do HIP em um ambiente real. Para tanto, são feitos experimentos com estações equipadas com interfaces de redes sem-fio. Esses experimentos buscam simular a caminhada de um usuário que ao se deslocar muda o seu ponto de acesso à Internet como mostrado na Figura I.6. Durante o trajeto, as mudanças na camada de rede são observadas quando se usa ou não o protocolo HIP.

Primeiramente para se avaliar o impacto da utilização da arquitetura do HIP, são fei-



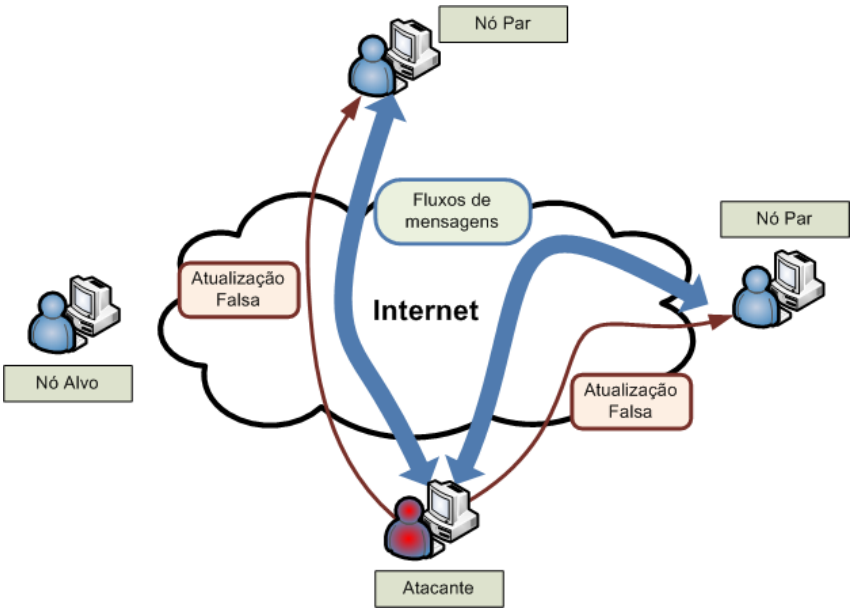

(a) Estações pares trocam mensagens normalmente. O atacante envia mensagens de atualização falsa para os pares.



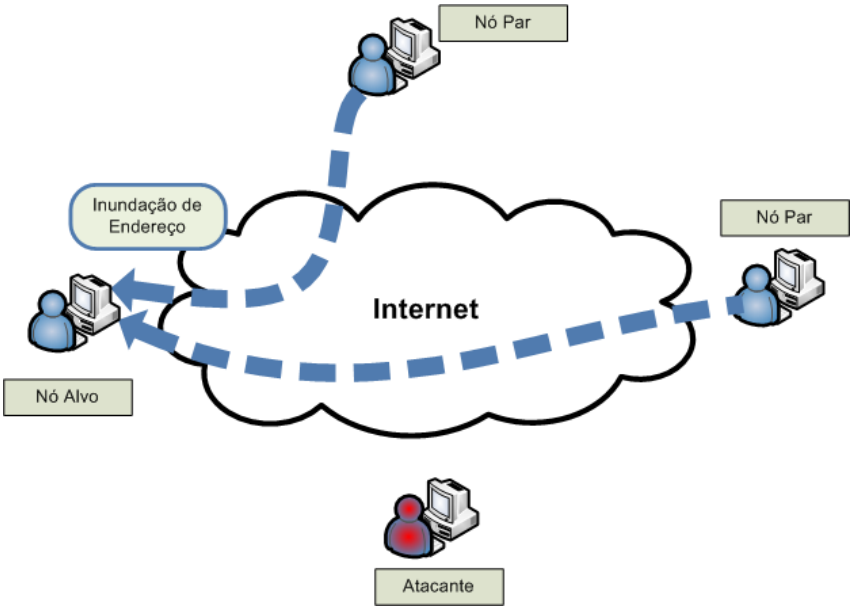

(b) As estações pares são forçadas a desviar as mensagens para o atacante.

Figura I.4: Furto de endereços. Neste caso o furto de endereços foi usado para realizar um ataque de homem no meio.

tos testes de atraso na rede cabeada. Posteriormente, para se avaliar o efeito da mudança de endereços, é feito um teste de exibição de um vídeo para simular uma conexão que o nó possa ter. Outras possibilidades são transferência de arquivos, conexão por `telnet`, etc. Em seguida, é avaliada a mobilidade. Ao se mover, um usuário muda o seu ponto de conexão com a rede e ao mesmo tempo deseja manter as conexões atuais ativas, sem




(a) As estações trocam mensagens normalmente com a estação maliciosa. Mensagens de atualização falsas são enviadas para os pares.




(b) As estações pares encaminham as mensagens para a estação alvo e sobrecarregam seu enlace.

Figura I.5: Inundação de endereços.

prejuízos significativos na qualidade do serviço recebido. Ao se mudar o ponto de conexão, consequentemente a faixa de endereços da rede também muda e com isso é possível deter-

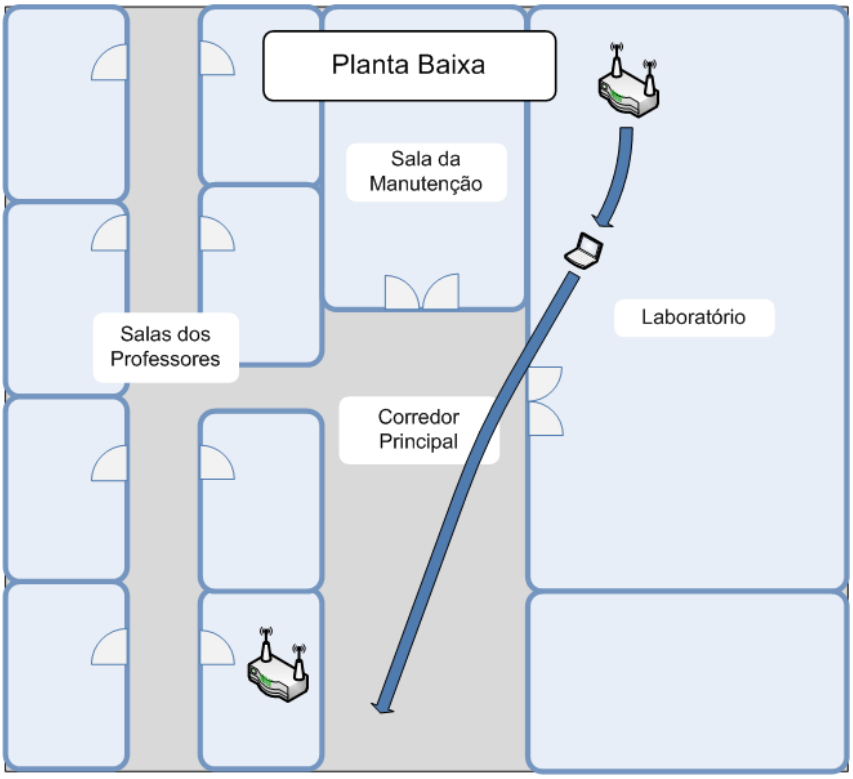



Figura I.6: Caminho do usuário com mudança de ponto de acesso. O usuário caminha do laboratório para o corredor. Em certo momento ocorre a mudança de ponto de acesso.

minar os efeitos do uso da arquitetura do HIP com mobilidade. O desempenho do HIP é avaliado através de medidas subjetivas, como o impacto da mobilidade na reprodução do vídeo. Também são realizadas medidas objetivas, tais como o atraso dos pacotes durante a movimentação do usuário e mudança de ponto de conexão.

## I.4 Organização do Texto

Este trabalho está organizado da seguinte forma. No Capítulo II é apresentado o Protocolo de Identificação de Estação (*Host Identity Protocol* - HIP). São descritas suas principais características e sub-protocolos envolvidos. No Capítulo III são apresentados, em detalhes, o cenário de testes e a plataforma de testes desenvolvida. Ainda nesse capítulo, são descritas as ferramentas usadas e criadas no desenvolvimento deste trabalho.

Detalhes referentes aos testes e à análise dos resultados são abordados no Capítulo IV. Por fim, no Capítulo V as conclusões sobre este trabalho são apresentadas e as considerações sobre trabalho futuros.

# Capítulo II

# Descrição do Protocolo de Identificação de Estação - HIP

## II.1 Identificadores

O Protocolo de Identificação de Estação (*Host Identity Protocol* - HIP) apresenta um novo domínio, chamado de domínio de identificação de nós da rede. Existem alguns tipos de identificadores dentro desse domínio. Dentre esses, o principal é o identificador de estação (*Host Identifier* - HI), descrito com mais detalhes na Seção II.1.1, o qual é uma chave pública. Outro identificador importante é a etiqueta de identidade de estação (*Host Identity Tag* - HIT), a qual é descrita na Seção II.1.2. Existe ainda um outro identificador, chamado de identificador de escopo local (*Local Scope Identifier* - LSI), utilizado localmente [7], conforme é descrito na Seção II.1.3.

A identidade das estações adiciona duas características essenciais nos protocolos da Internet. A primeira é a separação da camada de transporte e de rede. Isso possibilita modificações em uma das camadas sem que a outra perceba. Ademais, possibilita serviços fim-a-fim em ambientes de rede. A segunda característica é a autenticação dos nós da rede. Devido ao identificador ser uma chave pública, ele pode ser usado para autenticação em protocolos de segurança como o IPsec [9].

### II.1.1 O Identificador de Estação (*Host Identifier* - HI)

No Protocolo de Identificação de Estação, o HIP, o identificador de estação HI é uma chave pública de um par de chaves criptográficas assimétricas, composto por uma chave pública e sua chave privada correspondente. Então o nome representando a identidade no domínio de identificação, ou seja, o HI, é uma chave pública. A própria posse da chave privada indica a identidade. A estação é, então, definida como a proprietária da chave privada correspondente a chave pública definida pela identificação da estação, ou seja, o HI. Se mais de um nó possuir a chave privada, significa que a identidade identifica um grupo, podendo ser considerada como distribuída.

De acordo com a arquitetura do HIP, qualquer outra convenção de nomes na internet pode ser utilizada para os identificadores de estações. Entretanto, nomes não criptográficos devem ser usados apenas em situações de alta confiança e baixo risco. As identidades reais nunca não são utilizadas diretamente em protocolos da Internet. Os HI correspondentes podem ser armazenados em registros especializados do DNS [10] ou em diretórios do *Lightweigth Directory Access Protocol* (LDAP) [11] e são transmitidos nas trocas definidas pelo HIP. Outros identificadores como HIT e LSI são utilizados para representar a identidade em protocolos e interfaces de programa de aplicação (*Application Program Interface* - API).

### II.1.2 A Etiqueta de Identidade de Estação - HIT

A Etiqueta de Identidade de Estação (*Host Identity Tag*), HIT, é um resumo calculado com base em uma função *hash* do HI. A motivação por trás do HIT é padronizar um tamanho para ser usado nos protocolos envolvidos no HIP. Existem diversos algoritmos diferentes de chaves públicas que podem ser utilizados com comprimentos de chaves diferentes. Portanto, a chave pública HI não é indicada para ser usada como um identificador de pacotes ou índice para as diversas tabelas operacionais necessárias para o funcionamento dos protocolos. O HIT é um *hash* de 128 bits. Portanto, ao invés da chave pública HI, é o HIT, *hash* de HI de tamanho reduzido e fixo, que é utilizado no cabeçalho dos

pacotes HIP.

Nos pacotes dos protocolos do HIP, existem sempre dois HITs, Um para identificar o remetente e outro para identificar o destinatário. Ele deve, portanto, ser único no universo IP enquanto estiver em uso. Na eventualidade do HIT mapear mais de uma identidade, o que pode acontecer devido a colisões com a função *hash* [12], para diferenciar as identidades é usado o HI (chave pública). Se houver mais de uma chave pública para uma dada estação, o HIT é o indicativo de qual identidade correta a ser usada.

As propriedades do HIT são:

1. tem o mesmo comprimento que o endereço IPv6, para poder ser utilizado no campo de endereços em APIs e protocolos existentes;
2. é auto certificado, pois, dado um HIT, é computacionalmente difícil achar o HI correspondente;
3. a probabilidade de colisão entre dois HIT é bem baixa.

Colocar os HIs ou HITs no cabeçalho de cada pacote não é recomendado, pois aumentaria muito a sobrecarga da rede. Então, para se evitar este problema, é definido um mapeamento padrão entre os pacotes de dados e os HIs. O mapeamento é realizado com o *Security Parameter Index* (SPI) presente no cabeçalho extra do *Encapsulated Security Payload* (ESP) [13, 14], que permite encriptar os dados e autenticar as mensagens. A utilização do SPI e ESP nos pacotes de dados do HIP são descritas na seção II.3.3.

O HIT é gerado através de um tipo de *Overlay Routable Cryptografic Hash Identifier* (ORCHID) [15], o qual é baseado na aplicação do algoritmo *Secure Hash Algorithm version 1* (SHA-1) no identificador da estação, o HI. O ORCHID possui um prefixo alocado do bloco IPv6 e é definido no documento RFC 4843 [15].

### II.1.3 O Identificador de Escopo Local - LSI

O Identificador de Escopo Local (*Local Scope Identifier* - LSI) é uma representação local da identidade. Ele pode ter 32 ou 128 bits (normalmente 32 bits). Ele é utilizado para facilitar a utilização de identidades em protocolos e APIs existentes. O seu problema é ser somente válido em um escopo local, ou seja, não é possível utilizá-lo como um identificador global.

Exemplos de seu uso podem ser: o endereço em um comando `ftp`, `ping`, endereço de uma chamada de *socket*, etc. Ele atua como uma conexão da arquitetura HIP com protocolos e APIs (IPv4 principalmente) já existentes.

## II.2 O Protocolo de Identificação de Estação - HIP

O cabeçalho HIP poderia ser carregado por todos os pacotes IP. Entretanto, por possuir um tamanho relativamente grande (40 bytes), é desejável comprimir o cabeçalho. O cabeçalho deve estar presente somente em pacotes de controle para estabelecer ou modificar conexões. O método definido para fazer essa compressão é o (ESP) que realiza um mapeamento das identidades com um dos parâmetros de seu cabeçalho o SPI (descrito com mais detalhes na Seção II.3.3). Mais tarde outros métodos poderão ser definidos. O número de protocolo a ser preenchido no cabeçalho IP definido pela IANA para o HIP é o 139.

### II.2.1 Criação de uma associação

O sistema que inicia a troca HIP é chamado de iniciador, e seu par de respondedor. Essa distinção desaparece após a troca de bases (*Base Exchange* - BEX) ser completada. A troca de bases tem o propósito de estabelecer uma conexão entre os pares. Esse é um acordo de quatro mensagens. A primeira mensagem, a mensagem **I1**, inicia a troca; as últimas três: **R1**, **I2** e **R2**, constituem uma troca autenticada de chaves Diffie-Hellman.

Durante o protocolo Diffie-Hellman, o material criptográfico é criado. Caso outras chaves criptográficas sejam necessárias, como a chava para ser usada com ESP, elas são derivadas do mesmo material criptográfico.

O protocolo começa quando o iniciador envia o pacote **I1**. Este pacote funciona como um gatilho, para iniciar as operações. Ele contém somente o HIT do iniciador e pode conter também o HIT do respondedor, caso ele seja de conhecimento do iniciador. Em alguns casos é possível trocar esse pacote gatilho por outras formas de gatilho, nos quais o protocolo se inicia com o pacote **R1**.

O segundo pacote, o pacote **R1**, começa de fato a troca de bases. O pacote contém um desafio criptográfico que o iniciador deve solucionar antes de continuar com o protocolo. O nível de dificuldade do desafio pode ser ajustado com o nível de confiança do iniciador, carga corrente ou quaisquer outros fatores. O pacote **R1** contém os parâmetros iniciais Diffie-Hellman e uma assinatura que cobre parte da mensagem. Alguns campos são deixados de fora da assinatura para suporte de pacotes R1 pré-criados.

No terceiro pacote, o pacote **I2**, o iniciador deve exibir a solução do desafio. Se a resposta não for correta o pacote é descartado. Ele também carrega as informações Diffie-Hellman necessárias para o respondedor. O pacote é inteiramente assinado. O pacote **R2** é o último pacote do protocolo. Ele finaliza a troca de bases e é assinado. A Figura II.1 ilustra esse processo que será explicado com mais de talhes na Seção II.2.4.

### II.2.2 Formato da Mensagem HIP

Os pacotes de controle do HIP possuem tamanho fixo com o formato mostrado na Figura II.2:

O cabeçalho HIP foi desenvolvido para ser uma extensão do cabeçalho IPv6. O campo *Next Header* (8 bits) foi desenvolvido para trabalhar com IPv6. Entretanto, as implementações atuais não utilizam o campo. Esse campo é preenchido com o valor IPPROTO_NONE (número 59) que corresponde a inexistência de próximo cabeçalho.

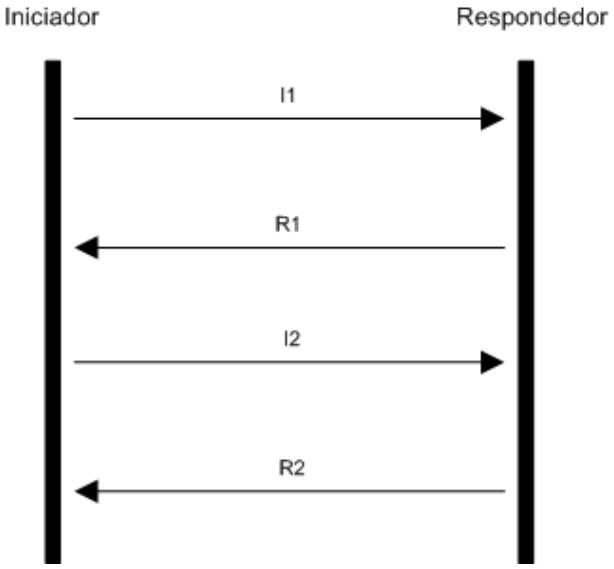



Figura II.1: Mensagens do protocolo de troca de bases (BEX).

O Campo *Header Length* (8 bits) indica o tamanho do cabeçalho HIP. É medido em unidades 8 bytes, exceto os primeiros 8 bytes. Como todos os cabeçalhos devem possuir os HITs do remetente e do destinatário, então o menor número desse campo é 4. O tamanho do campo de parâmetros é limitado dessa forma por:

$$Tamanho_{max\_parametros} = Tamanho_{max} - Tamanho_{HIT}$$

$$Tamanho_{max\_parametros} = (255 * 8) - (4 * 8) = 2008 \text{ bytes}$$

O campo *Packet Type* (7 bits) indica o tipo do pacote HIP. Os tipos de pacote **I1**, **R1**, **I2** e **R2** são dos tipos 1, 2, 3 e 4 respectivamente. Caso o pacote possua um tipo desconhecido, ele deve ser descartado.

O campo *version* (4 bits) indica a versão do HIP. O valor atual é 1. Esse valor deve ser incrementado somente se houver mudanças que não sejam compatíveis com versões anteriores.

Os próximos 3 bits são reservados e devem ser enviados com o valor zero e ignorados na recepção.

Os dois valores de valor fixo no cabeçalho (bits número 16 e 31) estão reservados para

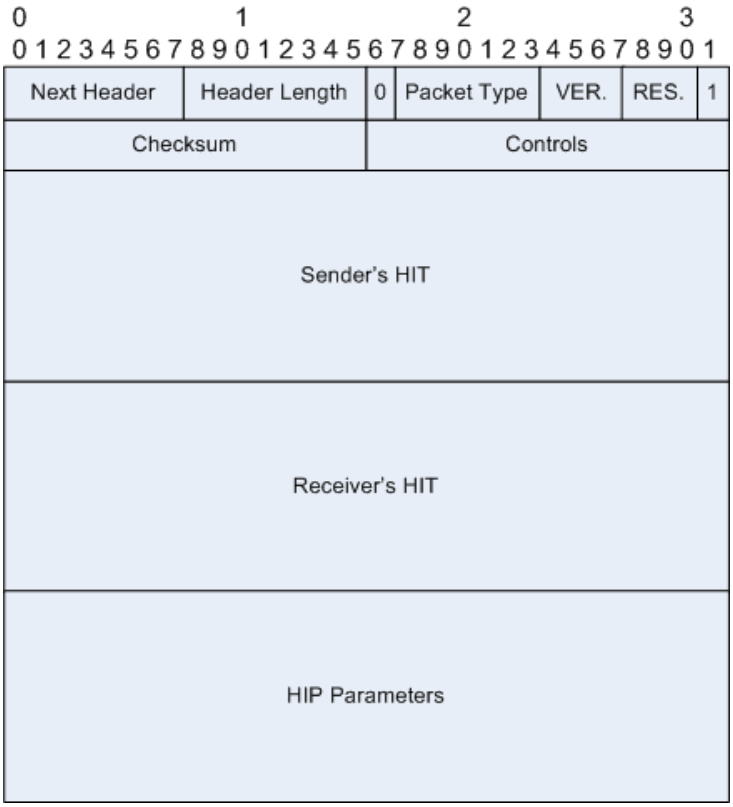



Figura II.2: Cabeçalho dos pacotes HIP.

potencial compatibilidade com o protocolo *Level 3 Shim for IPv6* (SHIM6) [16]. Se este não for utilizado, os valores devem ser zero e devem ser ignorados na recepção.

O campo *checksum* de 16 bits tem como finalidade detectar erros acidentais. O campo cobre os endereços IP de origem e destino, então deve ser recalculado em sistemas NAT conscientes da utilização do HIP.

O campo de *controls* (16 bits) informa a estrutura do pacote e as capacidades da estação. Os seguintes campos foram definidos (Figura II.3):


Figura II.3: Valores definidos para campo *controls*. Somente o foi definido **A** (Anônimo).

**A** - Anônimo (*Anonymous*). Se este campo for selecionado, a identidade do remetente do pacote é anônima, ou seja, não está publicada ou listada em algum diretório. Esse controle é marcado em pacotes **R1** ou **I2**.

Os campos restantes foram reservados para uso futuro e devem ser enviados com valor zero.

A seguir os valores dos HITs do remetente e do destinatário. Ambos os campos possuem o comprimento de 128 bits.

Os parâmetros dos HIP seguem o cabeçalho comum e definem as informações de sinalização HIP trocada entre os pares. São codificadas utilizando o formato *Type Length Value* (TLV) descrito mais a frente. Os tipos são mostrados na Tabela II.1.

A organização dos parâmetros em um pacote HIP deve ser de maneira incremental, ou seja, os parâmetros com menor índice de tipo devem ser incluídos antes dos parâmetros com maior índice de tipo. Se o ordenamento dos parâmetros não for incremental, o pacote é considerado como mal formado e é descartado.

Devido a esse ordenamento incremental, os parâmetros e tipos definidos foram espaçados para permitir extensões futuras. Parâmetros numerados de 0 até 1023 são utilizados na troca de bases do HIP e mensagens de atualização e são cobertos pela assinatura. Os 1024 até 2047 são reservados. Parâmetros numerados de 2048 até 4095 estão relacionados aos tipos de transformação utilizados (como ESP). Os números de 4096 até $(2^{16} - 2^{12})$ 61439 são reservados. De 62440 até 62463 são utilizados para assinaturas e MACs (*Medium Access Control*) assinados. De 62464 até 63487 são utilizados para parâmetros que caem fora área assinada. Os números de 63488 até 64511 são reservados para serviços Rendezvous e outros. Os números de 64512 até 65535 são reservados.

### II.2.3 Formato TLV

O formato dos parâmetros pode ser visto na Figura II.4. Os campos são os seguintes:

Tabela II.1: Parâmetros HIP existentes.

| TLV | Tipo | Tamanho | Descrição |
|---|---|---|---|
| R1_COUNTER | 128 | 12 | Contador da mensagem **R1** que indica a geração de desafios válidos |
| PUZZLE | 257 | 12 | $K$ e $\#I$ aleatório |
| SOLUTION | 321 | 20 | $K$ e $\#I$ aleatório e a solução de desafio $J$ |
| SEQ | 385 | 4 | Número da identificação do pacote de atualização |
| ACK | 449 | variável | Número da identificação do pacote de atualização |
| DIFFIE_HELLMAN | 513 | variável | Chave pública |
| HIP_TRANSFORM | 577 | variável | Tipo dos algoritmos de encriptação e integridade |
| ENCRYPTED | 641 | variável | Parte encriptada do pacote **I2** |
| HOST_ID | 705 | variável | Identidade da estação como o nome completo (*Fully Qualified Domain Name FQDN ou Network Access Identifier (NAI)*) |
| CERT | 768 | variable | Certificado do HI utilizado para transferir certificados. Seu uso ainda não foi definido, mas será especificado quando preciso. |
| NOTIFICATION | 832 | variable | Dados informacionais. |
| ECHO_REQUEST_SIGNED | 897 | variable | Dados opacos assinados para serem transmitidos de volta. |
| ECHO_RESPONSE_SIGNED | 961 | variable | Dados opacos assinados transmitidos de volta ao pedinte. |

| TLV | Tipo | Tamanho | Descrição |
|---|---|---|---|
| HMAC | 61505 | variable | Código de autenticação baseado em HMAC, com material criptográfico de HIP_TRANSFORM |
| HMAC_2 | 61569 | variable | Código de autenticação baseado em HMAC, com material criptográfico de HIP_TRANSFORM. A diferença para o HMAC é que o HOST_ID é incluído no cálculo do HMAC_2. |
| HIP_SIGNATURE_2 | 61633 | variable | Assinatura do pacote. **R1** |
| HIP_SIGNATURE | 61697 | variable | Assinatura do pacote. |
| ECHO_REQUEST_UNSIGNED | 63661 | variable | Dados opacos não assinados para serem transmitidos de volta. |
| ECHO_RESPONSE_UNSIGNED | 63425 | variable | Dados opacos não assinados transmitidos de volta. |

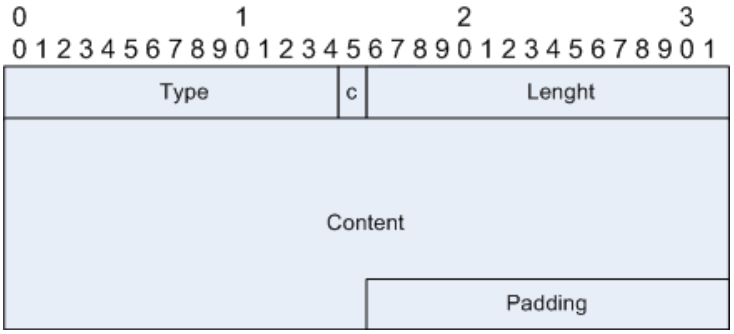



Figura II.4: Formato *Type, Length and Value* (TLV) dos parâmetros.

- *Type* - Código do tipo do parâmetro. Tamanho de 16 bits, o campo c considerado parte do campo tipo.
- *c* - critical. Possui valor 1 se o parâmetro for crítico e precisa ser reconhecido pelo destino. Consequentemente os tipos possuem valor impar se forem críticos ou par se não forem.
- *Length* - Tamanho do campo Contents em bytes. O tamanho do parâmetro TLV total é um múltiplo de 8 bytes. O tamanho total é descrito pela fórmula abaixo. $\texttt{total\_Length} = 11 + \texttt{length} - (\texttt{length} + 3) \bmod 8$
- *Contents* - Conteúdo específico do parâmetro. Tamanho variável.
- *Padding* - utilizado para preencher os bytes restantes para o tamanho total seja múltiplo de 8 bytes. Possui o tamanho variável de 0 a 7 bytes.

Os parâmetros são descritos com maiores detalhes na explicação do protocolo de troca de bases.

### II.2.4 O Protocolo de Troca de Bases - BEX

O processo começa assim que o iniciador tem conhecimento da identidade do par, o respondedor. Essa identidade pode ser obtida através de requisições em sistemas DNS

especializados, ou outras infraestruturas como LDAP ou tabelas *hash* distribuídas (*Distributed Hash Table* - DHT).

Contudo essa infraestrutura pode não dar suporte a certos casos como em ambientes par-a-par e temporários. Nestes casos o modo oportunístico pode ser utilizado. Nesse modo, o iniciador não possui informações a priori do nó par. O modo oportunístico é baseado no *"leap-of-faith"*, ou seja crença sem evidências. Isso significa que a troca de mensagens está suscetível a ataque de homem no meio (*Man in the Middle* - MitM) na conexão inicial. Esse sistema é similar ao *Secure Shell Protocol* (SSH) [17], no qual a chave pública do servidor é adicionada após a conexão.

Como dito anteriormente o protocolo de troca de base é um acordo de quatro mensagens. Ele é descrito com mais detalhes abaixo.

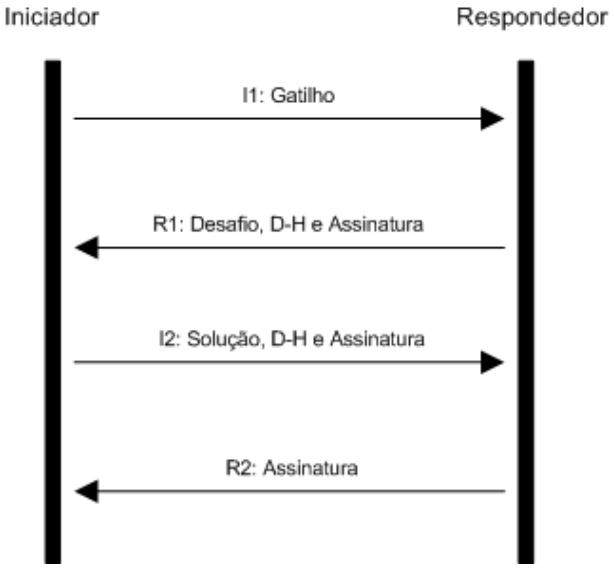



Figura II.5: Processo de troca de bases *Base Exchange* (BEX).

O primeiro pacote (o pacote **I1**) é o tipo 1 (*Packet Type* = 1). Ele é um pacote bem simples e não possui nenhum parâmetro. É o primeiro pacote do protocolo e funciona como um gatilho para iniciá-lo. O pacote pode ter o HIT do Respondedor se o possuir, ou pode este pode ser nulo para se fazer o protocolo no modo oportunístico. O pacote **I1** está representado na Figura II.6

```
I1
  Cabeçalho:
    Packet Type = 1
    SRC HIT = HIT do Iniciador
    DST HIT = HIT do Respondedor, ou NULO
  IP(HIP())
```

Figura II.6: O pacote **I1**.

O segundo pacote (o pacote **R1**) é do tipo 2 (*Packet Type* = 2). Este pacote serve como resposta do pacote **I1** e tem como funções principais: enviar o desafio, enviar parâmetros do acordo Diffie-Hellman e troca de capacidades de encriptação e integridade.

O respondedor pode ser anônimo, então o bit de controle A deve ser marcado.

Os HITs do remetente e destinatário devem ser iguais aos enviados pelo iniciador, ou seja, o HIT do respondedor e o HIT do iniciador respectivamente. Caso o iniciador utilizou o modo oportunístico, o respondedor pode escolher usar quaisquer um de seus HITs.

O parâmetro R1_COUNTER é utilizado para indicar a geração dos desafios. Soluções de desafios cuja geração é anterior a atual, não são aceitas.

O desafio (PUZZLE) consiste de um número aleatório $\#I$ e dificuldade $K$. O valor $\#I$ não é coberto pela assinatura e deve ser zerado durante o cálculo da assinatura. Isso permite o respondedor ter pacotes **R1** previamente computados. O mecanismo de desafio e suas vantagens são descritos posteriormente.

Diffie-Hellman é utilizado para a criação de uma chave secreta compartilhada entre os pares, por um meio inseguro e sem segredos prévios. Essa chave secreta é usada para algoritmos de encriptação. O parâmetro DIFFIE_HELLMAN define os parâmetros Diffie-Hellman. Eles são o $p$, $g$ e o $g^{segredo\_respondedor} \bmod p$ [18]. O grupo *More Modular Exponential* (MODP) [19] é enviado em um campo do parâmetro, e os números $p$ e $g$ são escolhidos com base nesse grupo. Podem ser enviados até dois grupos diferentes de MODP de Diffie-Hellman. Grupos diferentes servem para oferecer algoritmos mais fracos para dispositivos com menor poder computacional. É recomendado ao iniciador escolher

o grupo que possui os algoritmos mais fortes.

HIP_TRANSFORM define os algoritmos de encriptação e integridade que o respondedor suporta. Eles são ordenados por ordem de preferência. Todas as implementações devem permitir o uso de AES [20] com HMAC-SHA-1-96 [21].

O identificador do nó (HOST_ID). Ou seja, a chave pública e o algoritmo utilizado para sua geração (RSA ou DSA) assim como a FDQN.

O parâmetro ECHO_REQUEST_SIGNED e ECHO_REQUEST_UNSIGNED contém dados opacos que o nó quer receber sem modificações no parâmetro de resposta apropriado ECHO_RESPONSE_SIGNED ou ECHO_RESPONSE_UNSIGNED respectivamente.

O pacote **R1** é assinado e a assinatura é codificada no parâmetro HIP_SIGNATURE_2. Esse parâmetro define o valor da assinatura e o algoritmo utilizado (RSA ou DSA). A assinatura é feita zerando os campos e HIT de destino, *checksum*, valor do desafio $\#I$, e inclusive os dados opacos se utilizados.

É possível que o respondedor possua pacotes **R1** pré-computados para minimizar os efeitos de ataque de negação de serviço. Nesse caso o processamento necessário antes do envio do pacote é colocar o valor do HIT do destinatário, escolher e colocar um valor de $\#I$ e, por fim, colocar os campos de dados opacos.

O pacote **R1** está representado na Figura II.7

Quando o iniciador recebe o pacote **R1**, ele prepara o terceiro pacote (o pacote I2), que é do tipo 3 (*Packet Type* = 3). O propósito do pacote é entregar a solução, enviar os parâmetros Diffie-Hellman e seleção dos mecanismos de integridade e encriptação.

O iniciador pode ser anônimo, então o bit de controle A deve ser marcado.

O parâmetro R1_COUNTER indica a geração do desafio utilizado para gerar a resposta. Esse valor deve ser o mesmo enviado pelo respondedor.

A solução é codificada no parâmetro SOLUTION.

```
R1
  Cabeçalho:
    Packet Type = 2
    SRC HIT = HIT do Respondedor
    DST HIT = HIT do Iniciador
  IP ( HIP ( [ R1_COUNTER, ]
    PUZZLE,
    DIFFIE_HELLMAN,
    HIP_TRANSFORM,
    HOST_ID,
    [ ECHO_REQUEST, ]
    HIP_SIGNATURE_2 )
    [, ECHO_REQUEST_UNSIGNED ])
```

Figura II.7: O pacote **R1**.

DIFFIE_HELLMAN contém os parâmetros do protocolo Diffie-Hellman ($p$, $g$ e a chave pública que é $g^{segredo\_iniciador} \bmod p$). O iniciador escolhe o grupo mais forte suportado por ele e cria o material criptográfico. A chave secreta gerada pelo protocolo será: $(g^{segredo\_respondedor} \bmod p)^{segredo\_iniciador} \bmod p$

O iniciador coloca em HIP_ENCRIPTION os algoritmos selecionados para serem utilizados para encriptação e integridade. Este deve ser um dos anunciados pelo mesmo parâmetro enviado pelo respondedor.

A identidade do iniciador pode ser encriptada ou não pelo algoritmo definido no parâmetro HIP_TRANSFORM. O parâmetro ECHO_RESPONSE é uma cópia do parâmetro ECHO_REQUEST enviado pelo respondedor. Dependendo se o pedido desse parâmetro é do tipo assinado ou não, a resposta (ECHO_RESPONSE) é assinada ou não.

A integridade do pacote é garantida com o parâmetro *Keyed-HASH Authentication Code* (HMAC). Este campo deve ser validado pelo respondedor ao receber o pacote **I2**. O HMAC cobre a mensagem toda, exceto os parâmetros que o seguem (HIP_SIGNATURE e ECHO_RESPONSE).

A assinatura é calculada sobre todo o pacote exceto o parâmetro que segue este. O respondedor deve validar a assinatura. Para isso ele pode utilizar o HI provido nesse

pacote ou o HI adquirido por outros meios.

O pacote **I2** está representado na Figura II.8

```
I2
    Cabeçalho:
      Packet Type = 3
      SRC HIT = HIT do Iniciador
      DST HIT = HIT do Respondedor
    IP ( HIP ( [R1_COUNTER,]
      SOLUTION,
      DIFFIE_HELLMAN,
      HIP_TRANSFORM,
      ENCRYPTED { HOST_ID } ou HOST_ID,
      [ ECHO_RESPONSE_SIGNED ,]
      HMAC,
      HIP_SIGNATURE
      [, ECHO_RESPONSE] ) )
```

Figura II.8: Pacote **I2**.

Ao receber o pacote **I2** o respondedor pode calcular a chave secreta com base na fórmula: $(g^{segredo\_iniciador} \bmod p)^{segredo\_respondedor} \bmod p$ Ele pode gerar o material criptográfico e utilizar os algoritmos de encriptação e integridade.

O pacote **R2** é o quarto e último pacote do protocolo de troca de bases ($Packet\ TYPE = 4$). Ele finaliza o protocolo e possui somente dois parâmetros. O parâmetro HMAC_2 é calculado sobre todo o pacote sem o parâmetro HIP_SIGNATURE e a ainda com o HOST_ID concatenado. Após o cálculo, o parâmetro HOST_ID é removido. O outro parâmetro é HIP_SIGNATURE. Esta é uma assinatura do pacote inteiro. Os dois parâmetros devem ser verificados na recepção pelo iniciador.

O pacote **R2** está representado na Figura II.9

```
R2
  Cabeçalho:
    Packet Type = 4
    SRC HIT = HIT do Respondedor
    DST HIT = HIT do Iniciador
  IP ( HIP ( HMAC_2, HIP_SIGNATURE ) )
```

Figura II.9: O pacote **R2**.

## II.3 A Segurança no HIP

### II.3.1 Mecanismo Desafio

O propósito do mecanismo de desafio é proteger o respondedor de alguns tipos de ameaças de negação de serviço. Ele permite o respondedor atrasar a criação da conexão até a chegada do pacote **I2**. Além disso, o desafio permite ao respondedor, com baixo custo computacional, avaliar se o iniciador é sincero no pedido de conexão e está realmente disposto de gastar tempo e processamento para criar a conexão.

O respondedor pode permanecer sem estar conectado e pode rejeitar a maioria dos pacotes com origens forjadas, pois o desafio é baseado no HIT do iniciador. A ideia é o respondedor ter certo número de pacotes **R1** prontos, e escolhe um deles baseados em informações do pacote **I1**. Quando o respondedor posteriormente receber o pacote **I2**, ele poderá verificar se a solução do desafio foi feita utilizando o HIT do iniciador. Isso torna difícil para o atacante trocar as mensagens **I1** e **R1** iniciais e gerar um grande número de **I2** que pareçam vir de diferentes HITs.

O respondedor pode escolher a dificuldade do desafio dependendo do nível de confiança que se tem no iniciador. A assinatura do pacote **R1** não inclui o desafio. Isso possibilita a criação prévia de pacotes **R1** e incluir o desafio na hora do envio do pacote. O respondedor deveria usar heurísticas para determinar quando está sob ataque de negação de serviço, e escolher o nível do desafio apropriadamente.

Para a criação de um desafio o respondedor cria um número aleatório $\#I$ e seleciona

um nível de desafio $K$. Ele então envia esses números para o iniciador. O iniciador deve então achar a solução que é um certo número $J$. Deve se feito um *hash* utilizando o algoritmo RHASH do número $\#I$ concatenado com os HITs do iniciador e respondedor e com o número $J$. Os $K$ bits menos significativos desse *hash* devem ser iguais a zero. O iniciador deve então gerar o número $J$ diversas vezes até achar um, tal que o resultado do HASH tenha os primeiros $K$ bits iguais a zero. Estatisticamente, quanto maior o número de bits $K$ iguais a zero, maior o número de tentativas para achar o número $J$.

O respondedor deve gerar o número $\#I$ de maneira que o iniciador não possa adivinhá-lo. Uma forma de fazer isso é criar o $\#I$ como um *hash* de vários números concatenados: um número secreto criado aleatoriamente, os HITs e os IPs dos pares. O número secreto é válido somente por um período de tempo.

Ao receber a solução, o respondedor usa as informações recebidas (como HITs e IPs envolvidos, geração e número secreto correspondente) para validá-la. Como as informações do iniciador (HIT e IP) estão disponíveis em seus pacotes, o respondedor não precisa guardar essas informações do iniciador ao receber o pacote I1. Ao receber pacotes cujas informações sejam incompatíveis, o respondedor simplesmente os descarta.

Os respondedores podem ainda incluir dados opacos no parâmetro  O iniciador deve então copiar os dados recebidos nesse parâmetro para o parâmetro ECHO_RESPONSE. O valor colocado nesse parâmetro pode ser um segredo, o número $\#I$, ou qualquer dado relacionado. Com isso é possível verificar se foi ele mesmo que enviou o pacote **R1** e o desafio $\#I$. O segredo deve ser modificado periodicamente.

É recomendado o respondedor guardar desafios antigos por um tempo determinado. Isso possibilita iniciadores que são mais lentos solucionar os desafios. Ao mesmo tempo, o uso desse desafio já solucionado deve ser limitado para evitar ataques baseados nele.

### II.3.2 Ataques de Repetição

O HIP inclui alguns meios de se proteger contra ataques de repetição. Respondedores estão protegidos contra repetições de **I1** justamente pelo fato de não guardarem nenhuma informação ou estado a respeito dos iniciadores. Para responder esses pacotes ele adiciona algumas informações a pacotes **R1** já preparados.

Iniciadores estão protegidos contra repetições de pacotes **R1** pelo parâmetro R1_COUNTER. Esse parâmetro é incrementado monotonicamente. Assim, caso o iniciador receba pacote cujo valor desse parâmetro seja igual ou menor a um pacote já recebido, esse pacote é descartado.

Respondedores estão protegidos contra repetições de pacotes **I2**s pelo mecanismo de desafio e o uso opcional de dados opacos no parâmetro ECHO_RESPONSE.

Iniciadores estão protegidos contra repetições de pacotes **R2** e pacotes de atualização devido à verificação do HMAC, que é menos custosa computacionalmente.

### II.3.3 *Encapsulated Security Payload* (ESP)

O protocolo de troca de bases faz autenticação entre os pares e geram também chaves e material criptográfico. Entretanto, não provê nenhuma maneira de proteger a comunicação fim-a-fim. Isso pode ser feito com o ESP.

O estabelecimento de comunicação segura entre os nós feita com ESP requer algumas modificações em pacotes do protocolo de troca de bases. As modificações servem para configurar o ESP, escolha do algoritmo de encriptação e de *hash* utilizado. O protocolo está representado na Figura II.10

Um parâmetro chamado ESP_TRANSFORM deve ser adicionado no pacote **R1**. Nele o respondedor anuncia os as transformações (algoritmos) que ele tem suporte. O iniciador escolhe uma dentre as anunciadas pelo respondedor e a coloca no parâmetro ESP_TRANSFORM do pacote **I2**.

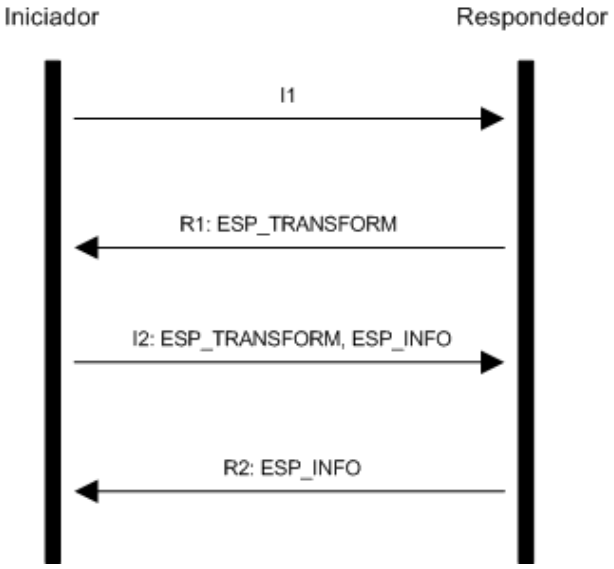



Figura II.10: Mensagens modificadas do protocolo de troca de bases (BEX) para funcionar com ESP.

As representações das identidades dos nós não são carregadas em todos os pacotes trafegados. Ao invés disso, o número *Security Parameter Index* (SPI) é colocado para indicar o nó par correto. Esse número é definido pelo parâmetro ESP_INFO no pacote **I2** e **R2**. O valor SPI enviado é o valor desejado para o par utilizar para identificar a conexão segura.

Durante a troca de bases são configuradas duas associações seguras (*Security Association* - SA), uma para pacotes enviados e outra para pacote que chegam no nó. A SA que indica a chegada de pacotes é a mesma que indica a saída de pacotes no nó par e vice-versa. As associações seguras utilizam o material criptográfico gerado no BEX para criar as chaves necessárias. Essas chaves serão utilizadas para encriptar e para garantir a integridade com HMAC. Então cada associação segura define chaves usadas.

Os SPIs são utilizados para achar a associação segura correta dos pacotes recebidos. Portanto, eles indicam ao receptor quais chaves que devem ser utilizadas para interpretar os dados do pacote, ou seja, para decriptar e verificar a integridade. Os SPI, quando utilizados no HIP, possuem mais um significado. Eles são uma representação comprimida

de um par de HITs. Eles podem servir para sistemas intermediários prover serviços tais como tradução de endereços.

Os SPIs possuem significado somente no receptor. Isso, pois o nó pode escolher o SPI que o par utilizará para identificar a associação. Ele pode inclusive ser o mesmo para vários pares diferentes. Então, o par <SPI, IP destino> identifica o HIT de destino em dado momento.

O ESP usado com HIP é da mesma maneira que com IPsec no modo transporte. A diferença é o sentido estendido do SPI, que representa um par de HITs.

## II.4 Mobilidade e Multidomicílio

O HIP cria uma arquitetura que separa a camada de transporte da camada de rede. A camada de identificação fica entre essas camadas, e utiliza chaves pública/privada para identificação ao invés de endereços IP. Quando HIP é utilizado, as camadas superiores ligam-se na camada de identificação, não mais na camada de rede. Isso possibilita mudanças na camada de rede, como mudanças de IP devido à mobilidade ou devido a protocolos como DHCP ou PPP, ou até renumeração de endereços IP. Possibilita também a presença de múltiplos endereços IP para uma única identidade, como no caso de multidomicílio (*multihoming*) de estação. Nesse cenário, o IP serve somente para encaminhar pacotes, como localizadores e não mais também como identificadores. Contudo, cada estação deve saber ao menos um endereço IP dos seus pares para que possa se comunicar com eles.

A arquitetura HIP define um parâmetro chamado LOCATOR, que é um conceito de localizador geral. Um localizador especifica um ponto de conexão com a rede. Esse localizador possibilita notificar aos pares endereços alternativos nos quais a estação é alcançável. Os localizadores podem ser meros endereços IP, ou podem também conter informações adicionais para auxiliar o tratamento de pacotes, como por exemplo, números SPI para saber a associação segura correta [22].

### II.4.1 O Pacote de Atualização

```
UPDATE
  Cabeçalho:
    Packet Type = 6
    SRC HIT =  HIT remetente
    DST HIT = HIT destinatário
  IP (HIP ( [SEQ, ACK, ] HMAC,HIP_SIGNATURE ))
```

Figura II.11: O formato do pacote de atualização.

O pacote de atualização é do tipo 6. Ele pode ser utilizado nas seguintes situações: expiração de uma conexão segura e reconexão (todas SA tem um tempo de vida determinado), inclusão de nova associação segura, mudança do endereço IP. O pacote está representado na Figura II.11.

Caso o parâmetro SEQ for utilizado o pacote deverá ser confirmado posteriormente através de um pacote de atualização com o parâmetro ACK pelo par. O valor do parâmetro ACK deve ser o mesmo do parâmetro SEQ enviado. Caso o pacote não possua SEQ, significa que ele não precisará ser confirmado. O parâmetro SEQ deve ser incrementado para manter a coerência dos pacotes, ou seja, a confirmação deve ser feita para o pedido de confirmação correspondente.

HMAC e HIP_SIGNATURE são utilizados da mesma maneira que os pacotes do protocolo de troca de bases. Outros parâmetros podem ser incluídos no pacote de atualização, tais como LOCATOR, ESP_INFO, ECHO_REQUEST e ECHO_RESPONSE.

### II.4.2 Multidomicílio

Multidomicílio (*multihoming*) significa que um nó pode ser alcançável por diversos caminhos diferentes ao mesmo tempo. No contexto de HIP, isso significa que o nó possui vários localizadores simultaneamente, Ele pode anunciar os pares os múltiplos endereços e ainda definir o preferencial. O HIP pode ser integrado com SHIM6, para uma suporte a multihoming mais completo.

### II.4.3 Mobilidade

Quando um nó se move pela rede, ele anuncia aos seus pares a mudança do endereço através de mensagens de atualização (mensagem UPDATE). Nessas mensagens, são incluídos os localizadores (LOCATOR) nos quais o nó é alcançável. O pacote de atualização deve ser confirmado (pacote ACK). O par pode verificar a autenticidade do pacote através da assinatura e do HMAC.

Na ocasião de uma atualização, o nó pode decidir se deseja selecionar outras chaves para a conexão segura e ainda se deverão ser criadas novas chaves com o protocolo Diffie-Hellman.

Quando o HIP é utilizado com ESP para proteger as mensagens, o nó pode receber pacotes de qualquer endereço. Ele pode modificar o próprio IP e continuar a enviar pacotes com a mesma conexão segura. Entretanto, os seus pares não poderão enviar pacotes para os novos endereços sem antes atualizar de maneira segura e confiável os endereços associados.

### II.4.4 Atualização

O protocolo de atualização pode ocorrer de várias maneiras. A maneira mais simples é a modificação de localizador de um nó sem refazer as conexões seguras ou atualizar as chaves. Pode ocorrer também uma atualização do endereço, e ao mesmo tempo, reselecionar novamente as chaves da associação segura. O nó pode ainda recriar o material criptográfico com o protocolo Diffie-Hellman. Outros casos de atualização são quando um nó deseja informar localizadores alternativos, devido a *multihoming* de estação, ou devido a multidomicílio de sítio.

Existe também a mobilidade que utiliza um elemento de rede como ponto de encontro. Esse elemento é chamado de Rendezvous [23]. Ele é necessário quando ambos nós são móveis. Esse caso é explicado na Seção II.4.5.

**Mobilidade com Par Único de SA (sem Troca de Chaves)**

A estação móvel pode mudar seu endereço IP. Isso pode ocorrer por um movimento da estação e troca de ponto de conexão com a rede. Outras causas são: um enlace PPP reconectado, uma nova associação DHCP ou novo prefixo IPv6 anunciado no link. Para manter as conexões existentes a estação móvel deve informar aos seus pares a mudança de IP.

Este exemplo considera que a estação possui uma interface única, um endereço IP e um par de SA (uma para envio e outra para recepção) e não ocorrem mudanças nas chaves das conexões seguras.

O procedimento está representado na Figura II.12

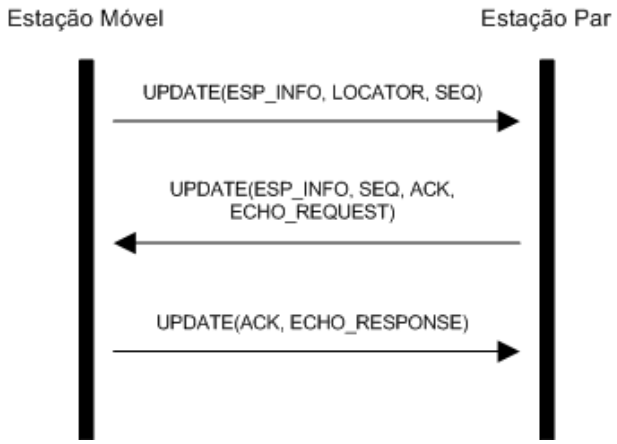



Figura II.12: Mensagens do protocolo de atualização sem troca de chaves.

Passos da mobilidade:

1. A estação móvel fica por um período curto de tempo desconectado do par, enquanto troca de endereço IP. Ao receber o novo IP, o nó móvel envia seu localizador (LOCATOR) em uma mensagem de atualização (UPDATE) para o par. O localizador contém somente o novo endereço IP e o tempo de vida. A mensagem contém ainda o parâmetro ESP_INFO, que contém os valores do SPI velho e do novo. Neste caso, os valores do SPI velho e do novo são os mesmos. Apesar de não ocorrerem

mudanças o parâmetro é incluído, pois poderá ser inspecionado por *middleboxes* especializadas. A mensagem inclui o parâmetro SEQ, que indica que a mensagem deve ser confirmada pelo receptor pelo parâmetro ACK.

2. O par recebe o UPDATE e o valida. Ele modifica todas as suas conexões com o nó móvel para ter o endereço novo e prepara uma mensagem de UPDATE de retorno. O par deve fazer a verificação de endereço, para evitar atualizações falsas. Ele coloca o parâmetro ECHO_REQUEST na mensagem para tal função. O par coloca o parâmetro ACK correspondente ao SEQ recebido e um novo SEQ. O parâmetro ESP_INFO com os valores dos SPI novo e velho também é adicionado na mensagem. A mensagem é então enviada.

3. A estação móvel para terminar o procedimento de atualização, precisa replicar os parâmetros recebidos. Ele adiciona na última mensagem o ACK e o ECHO_RESPONSE com o que foi recebido no ECHO_REQUEST. Os pares consideram o endereço como verificado, e já podem utilizá-lo sem maiores problemas.

#### Mobilidade com Par Único de SA (com Troca de Chaves)

O processo ocorre de maneira semelhante ao processo simples sem troca de chaves. Somente alguns parâmetros são modificados. E, no caso, de reconstruir o material criptográfico, um novo parâmetro deverá ser adicionado. O procedimento está representado na Figura II.13

O parâmetro ESP_INFO deverá ter o valor antigo do SPI e deverá incluir o novo valor desejado para o SPI. O parâmetro DIFFIE_HELLMAN é adicionado caso o material criptográfico seja refeito e funciona da mesma maneira que o protocolo de troca de bases.

#### Multidomicílio de Estação

Uma estação pode possuir múltiplas interfaces de rede ou endereços IP, denominado multidomicílio de estação (*Host Multihoming*). Se ele desejar, pode anunciar aos pares

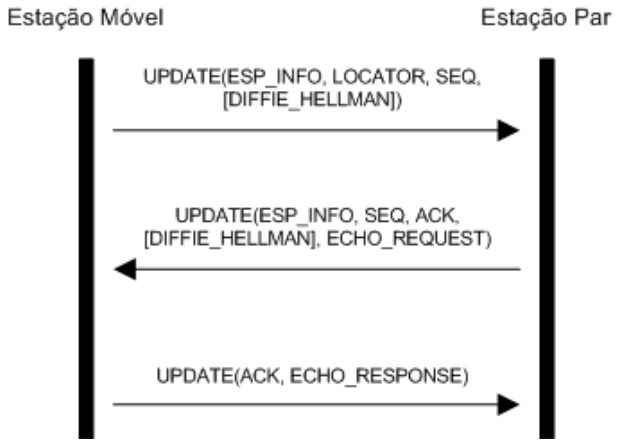



Figura II.13: Mensagens do protocolo de atualização com troca de chaves.

os endereços extras utilizando o parâmetro LOCATOR. Nós que possuem mais de uma interface devem anunciar a interface preferencial para os pares,

Para evitar a janela do ESP contra ataques de repetição, ele deve usar uma SA para cada interface. O nó deve então, criar um novo par de conexões seguras. O procedimento está representado na Figura II.14.

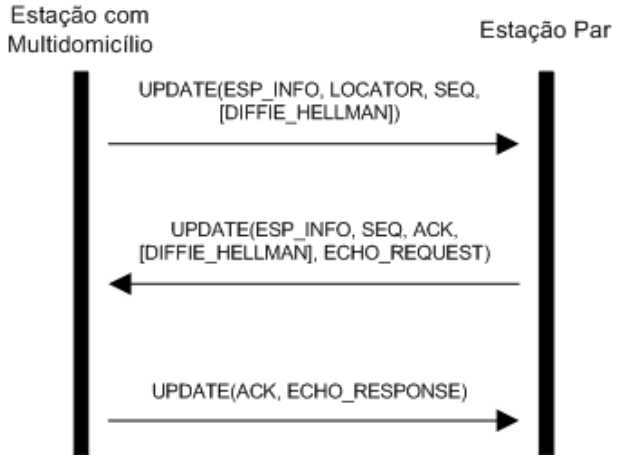



Figura II.14: Mensagens do protocolo de atualização para multidomicílio.

Para adicionar SAs, o nó envia na mensagem de UPDATE o LOCATOR com ESP_INFO. O valor do SPI antigo em ESP_INFO deve ser zero, que indica que não havia conexões anteriores. O parâmetro LOCATOR deve conter o endereço extra e deve

também conter todos os demais endereços ativos.

O procedimento é feito da mesma maneira que o de atualização de endereço.

#### Multidomicílio de sítios

Uma estação pode conter vários endereços globais roteáveis, denominado de multidomicílio de sítio (*Site Multihoming*). Isso pode decorrer do fato de um sítio estar conectado com múltiplos provedores de serviço. Pode também ser o caso que um provedor de serviço distribui aos nós tanto endereços IPv4 quanto endereços IPv6.

A estação pode ainda ter multidomicílio (*multihoming*) de estação ao mesmo tempo. Ele pode ser móvel também. Nesse caso o nó pode escolher quais endereços divulgar na hora da atualização.Esse caso é gerenciado da mesma maneira que o caso de multidomicílio de estação. Definições mais completas são encontradas na especificação do SHIM6.

### II.4.5 Rendezvous

A arquitetura HIP introduz o mecanismo Rendezvous para auxiliar uma estação a se comunicar com outra estação que se move constantemente. O mecanismo faz o uso de um novo elemento, o servidor Rendezvous (*Rendezvous Server* - RVS). Esse serve como ponto de encontro inicial entre as estações. Os clientes do RVS registram seus HITs e IPs. Após esse registro outras estações podem iniciar a troca de bases utilizando o endereço IP do servidor Rendezvous.

A mobilidade com o HIP permite que a estação informe aos pares sobre a mudança de seus localizadores. Ademais, a arquitetura presume que inicialmente ambas as estações são alcançáveis mutuamente. Contudo, tal estação pode desejar também ser alcançável a outra que não está ciente da mudança de localizador.

As estações que desejam utilizar os serviços do servidor Rendezvous, podem registrar os seus HITs e endereços IPs atuais [24]. O servidor então encaminha pacotes destinados

aos HITs das estações aos endereços correspondentes. Os clientes do RVS deveriam colocar em seus registros de sistemas como o DNS o endereço de seu servidor de Rendezvous. Ao se fazer requisições de endereços nesses serviços tipo DNS, o endereço do RVS que será divulgado, e assim as estações serão constantemente alcançáveis.

O procedimento de troca de bases é então ligeiramente alterado. Assume-se que o nó $R$ (respondedor) já se registrou com o RVS e os registros de seus HITs estão atrelados ao endereço do RVS.

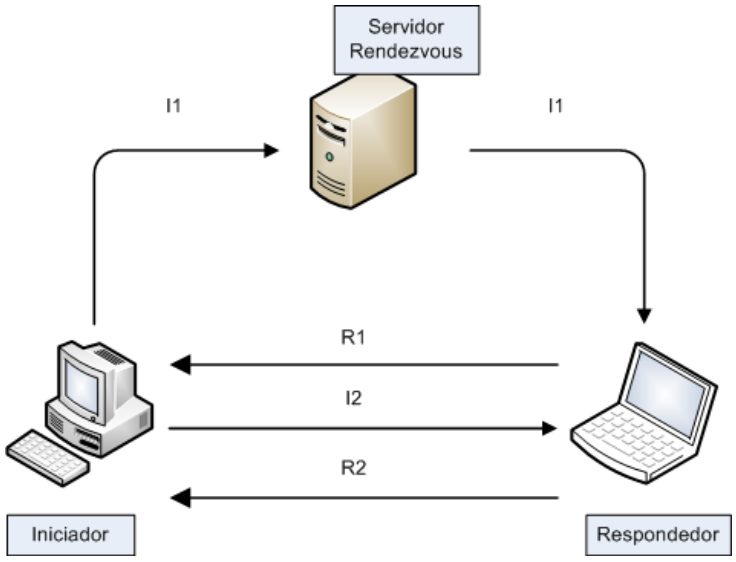



Figura II.15: Mecanismo do ponto de encontro (Rendezvous). O iniciador envia a primeira mensagem para o servidor de Rendezvous. Este encaminha para o Respondedor, que responde diretamente para o iniciador.

Ao iniciador (nó $I$) tentar estabelecer uma conexão com a estação $R$, ela necessita primeiramente enviar o primeiro pacote da troca de bases para um dos endereços de $R$ ou um de seus servidores Rendezvous. A estação $I$ obtém o endereço do RVS através de um registro DNS e envia o pacote **I1**, gatilho para troca de bases, para o RVS. O RVS ao perceber que o pacote **I1** que chegou não se destina a ele próprio verifica se o HIT de destino corresponde a um de seus clientes. Ao achar o cliente ele encaminha o pacote **I1** ao endereço atual dele.

O resto do procedimento de troca de bases ocorre normalmente, sem mais interferências

do servidor Rendezvous. Isso ocorre, pois após receber o pacote R1, a estação $I$ já possui o endereço atual da estação $R$. O procedimento está ilustrado na Figura II.15.

# Capítulo III

# O Cenário de Testes de Mobilidade

A fim de testar a viabilidade da mobilidade ao se utilizar o protocolo HIP foi projetado um cenário de testes. O objetivo dos testes é verificar a mobilidade, ou seja, a mudança do ponto de conexão com a rede, preservando as conexões de transporte (UDP e/ou TCP). ativas.

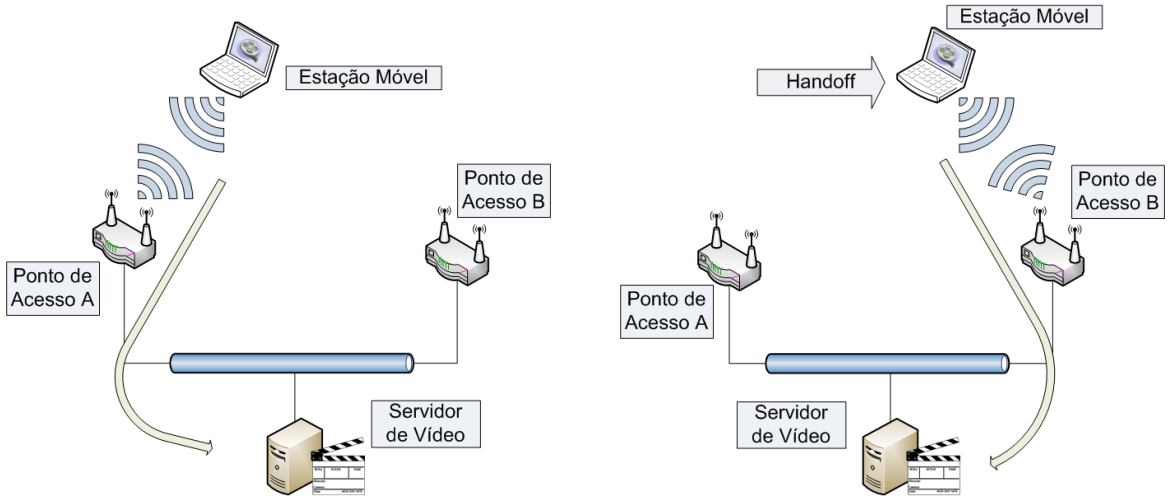
Este capítulo descreve o cenário de testes e as ferramentas usadas para a construção do cenário.

## III.1 Descrição do Cenário

O cenário de teste da mobilidade constitui-se de uma estação móvel que se conecta à rede através de um ponto de acesso. Esta estação recebe um fluxo contínuo(*streaming*) de vídeo encapsulado via HTTP (*Hypertext Transfer Protocol*)

Este cenário simula uma caminhada de um usuário que está utilizando um serviço de vídeo. O usuário está conectado a outros pares, possuindo, portanto, conexões de transporte (UDP e/ou TCP) ativas. Para simular essas conexões ativas, criou-se um servidor de vídeo ao qual o usuário se conecta para assistir ao fluxo de vídeo recebido.

Em dado momento o usuário entra em movimento e muda de ponto de acesso por um




(a) Situação antes do handoff.

(b) Situação após o handoff.

Figura III.1: Cenário de testes.

processo de *handoff*. *Handoff* é o processo de mudança de ponto de acesso enquanto ocorre transferência de dados. Este processo não interrompe nenhuma conexão ou transferência de dados. Os elementos utilizados nos testes são: uma estação móvel, um servidor de vídeo e dois pontos de acesso. Tais elementos são descritos a seguir.

### III.1.1 Elementos Utilizados nos Testes

#### A Estação Móvel

A estação móvel utilizada foi o portátil IBM Thinkpad Intel Pentium M 1.7 GHz com 512 MB de memória RAM mostrado na Figura III.2(a). Esta estação já vem equipada com uma interface de rede sem fio. Essa interface foi utilizada para fazer a comunicação entre a estação móvel e o servidor de vídeo. A estação móvel foi também equipada com uma placa Linksys PCMCIA WPC55AG, mostrada na Figura III.2(b), a fim de monitorar o enlace sem fio sem interferir no tráfego de vídeo. O sistema operacional instalado no portátil é o Linux, distribuição Debian Etch, com núcleo (*kernel*)2.6.24-1-686.

Para monitorar e configurar o enlace sem fio foram utilizadas as ferramentas `iwconfig`

(a) Estação móvel utilizada, IBM Thinkpad.

(b) Placa PCMCIA Lynksys WPC55AG.

e `iwlist`. Para configurar a rede e as rotas foram utilizadas as ferramentas `ifconfig` e `route`. Uma explicação mais detalha e as configurações aplicadas seguem na Seção III.2

### Servidor de Vídeo

Como servidor de vídeo foi utilizado um computador de mesa Intel Core2 Duo 2.4 GHz com 2 GB de memória RAM. O sistema operacional do servidor é o Linux, distribuição Debian Lenny, com núcleo (*kernel*) 2.6.26.2-amd64. Assim, como na configuração da estação móvel, para configurar a rede e as rotas foram utilizadas as ferramentas `ifconfig` e `route`.

### Os Pontos de Acesso

Para que a estação móvel possa se conectar à rede cabeada, são necessários pontos de acesso sem fio. Nos testes deste projeto, foram utilizados os roteadores Linksys mostrados na Figura III.1.1. Ambos os roteadores sem fio funcionam com o sistema operacional OpenWRT [25], que é um sistema Linux para dispositivos embarcados. Assim, diversas ferramentas que estão disponíveis no Linux, estão também instaladas nesse sistema operacional. Os roteadores funcionam como ponto de acesso para as estações móveis. Cada roteador pertence a duas redes: a rede cabeada e a rede sem fio.


(c) Roteador Linksys WRT54G.

(d) Roteador Lynksys WRT350N.

### III.1.2 Configuração dos Elementos

A Figura III.2 mostra o cenário de testes com mais detalhes. O servidor de vídeo se conecta, através de uma rede cabeada, aos roteadores sem fio, que servem como *gateways* para as suas respectivas redes sem fio. Como a estação móvel não pertence à rede cabeada e o servidor de vídeo não pertence a nenhuma das redes sem fio, então é necessário configurar rotas para todas as redes em cada um dos elementos.

Para se montar este cenário diversas ferramentas foram utilizadas, conforme descrito a seguir.

## III.2 Ferramentas

- `ifconfig`: ferramenta utilizada para configurar o endereço IP de cada máquina;
- `route`: ferramenta utilizada para configurar as rotas;
- `iwconfig` e `iwlist`: ferramentas utilizadas para configurar as interfaces de rede sem fio;
- `VideoLAN`: ferramenta utilizada para fazer a transmissão, recepção e exibição do vídeo;

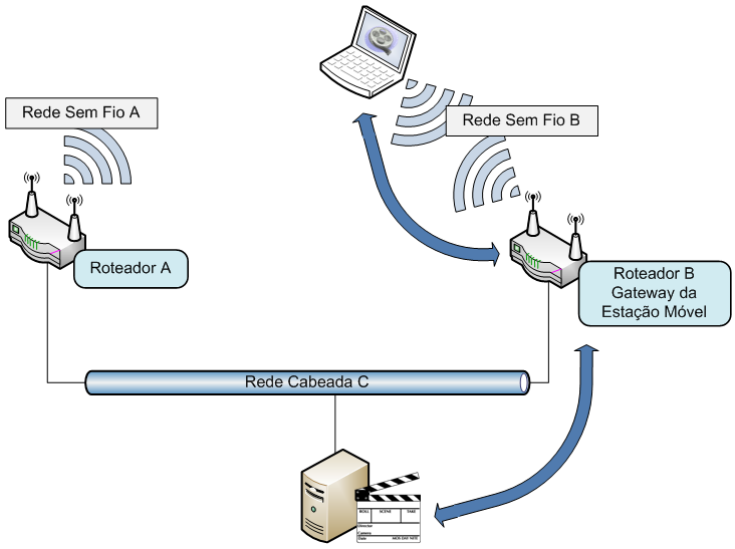



Figura III.2: Cenário de testes detalhado.

- `ping` e `tcpdump`: ferramentas utilizadas para fazer os testes de atraso;
- `OpenHIP`: ferramenta utilizada para fazer a comunicação por meio dos protocolos do HIP. Esta ferramenta implementa uma interface especial que faz as comunicações quando se utiliza o HIP;
- `Handoff.py`: ferramenta desenvolvida para se fazer a mudança de pontos de acesso. Esta ferramenta analisa o meio sem fio e, quando há a necessidade, muda o ponto de acesso.

### III.2.1 OpenHIP

As estações que implementam o HIP, para se comunicarem, utilizam a ferramenta `OpenHIP` [26]. Esta ferramenta constitui-se de dois programas de linha de comando: `hitkey` e `hip`. O programa `hitgen` é utilizado para vários propósitos. Ele serve para gerar a identidade da estação, ou seja, para gerar o par de chaves pública/privada, o HIT e o LSI. O `hitgen` também é utilizado para gerar o arquivo de configuração para o programa `hip`, que é utilizado para fazer o tratamento dos pacotes. Para isso, o `hip` cria um dispositivo virtual (*driver* TAP). Esse dispositivo é responsável por pegar os pacotes

e enviá-los para o programa `hip`. A arquitetura do `OpenHIP` está representada na Figura III.3.

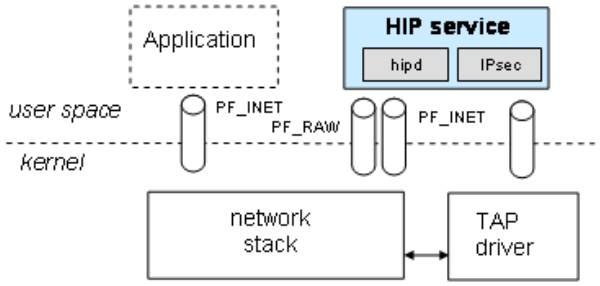



Figura III.3: Arquitetura do OpenHIP. Figura retirada do site `http://www.openhip.org`

A maneira que a ferramenta `OpenHIP` faz para manter a compatibilidade com as aplicações existentes é mapear os identificadores LSI e HIT para o dispositivo virtual. Este dispositivo virtual encaminha os pacotes para o programa `hip` que faz o tratamento adequado dos mesmos. Em seguida o programa `hip` envia os pacotes para a interface de rede fídica.

Os endereços criados para LSI possuem 32 bits e são do tipo 1.x.x.x. A tabela de roteamento do *kernel* é configurada de forma que a interface de saída da rede 1.0.0.0/8 seja o dispositivo virtual TAP. Dessa maneira, é possível diferenciar o tráfego normal e o tráfego HIP. O mesmo ocorre para o HIT, mas os endereços em questão são de 128 bits, pois são endereços IPv6.

Quando o pacote chega ao processo `hip`, ele é processado para implementar as funcionalidades do HIP. O processo primeiramente traduz o endereço LSI (ou HIT) para o endereço IP associado àquela identidade. Caso não haja informações sobre essa tradução, é feita uma requisição ao servidor DNS. Outra possibilidade é a inclusão da identidade em um arquivo de identidades.

O passo seguinte é verificar se já existe uma conexão segura ativa com o par (SA). Caso não haja conexões seguras com o par, é feito o procedimento de troca de bases (BEX) com o par. Caso haja conexão ativa com o par, o processo faz a encriptação que fora definida

durante o BEX. Nesta implementação, existe o limite de 1 SA por par. A ferramenta `OpenHIP` implementa diversos procedimentos definidos nas RFCs relacionadas [9, 14, 24, 23, 22, 27, 28]. Os procedimentos implementados são os seguintes:

- troca de bases e esp com IP v4;
- troca de bases e esp com IPv6;
- mobilidade;
- transgressão de NAT;
- servidor *rendezvous*;
- requisições ao DNS;
- mobilidade de roteadores;
- HIP com *Secure Mobile Architecture* (SMA).

Existem diversos parâmetros de configuração da ferramenta. Os parâmetros relevantes serão discutidos conforme forem citados.

### III.2.2 VideoLAN

O servidor de vídeo opera com a ferramenta `VideoLAN` [29], composta pelo VLC (VideoLAN Client) e pelo VLS (VideoLAN Server). Estes dois programas trabalham com o padrão MPEG (1, 2 e 4) e suportam encapsulamento via HTTP (*Hypertext Transfer Protocol*). Além disso, estão disponíveis para diversas plataformas, incluindo Windows e Linux, e seus códigos-fonte são abertos.

Tanto a ferramenta VLS quanto a VLC, inicialmente só um cliente, podem ser usadas para distribuir vídeo. O VLS não possui nenhuma interface gráfica. Sua configuração é feita via arquivos de configuração e o programa é executado por linha de comando. O VLS pode ser usado como um servidor de fluxos elementares MPEG-2 (DVD), canais de

TV aberta, vídeos ao vivo e fluxos originados por uma placa codificadora MPEG-2. Já o VLC possui uma interface gráfica amigável e pode distribuir vídeos nos formatos MPEG-2 e MPEG-4. As duas ferramentas podem usar os protocolos *User Datagram Protocol* (UDP) e *Real-time Transport Protocol* (RTP) para transmissão dos pacotes de vídeo e o *Real Time Streaming Protocol* (RTSP) para requisição de conteúdo. O VLC foi utilizado tanto para exibição do vídeo quanto para o *streaming* HTTP.

Na criação do servidor de vídeo (também na conexão com ele), é necessário informar o endereço IP que está associado ao servidor. O endereço utilizado foi o endereço que está associado à interface que faz a comunicação usando HIP, ou seja, o endereço LSI criado junto com a identidade (pois se utiliza o IPv4). Nesse caso, percebe-se a importância de identificadores para manter a compatibilidade com os aplicativos existentes.

### III.2.3 `ping` e `tcpdump`

A ferramenta `ping` é um programa de linha de comando normalmente utilizado para verificar a alcançabilidade de uma dada estação de destino. O `ping` envia mensagens ICMP(*Internet Control Message Protocol*) para o destino fornecido e mede o tempo de ida e volta (*Round Trip Time* - RTT), descontado o tempo de processamento da mensagem. A ferramenta exibe o RTT de cada mensagem. Para a ferramenta `ping` funcionar, é necessário digitar o nome do programa e um endereço. Esse endereço pode ser um nome de um servidor DNS, ou pode ser um endereço IP. Como se está trabalhando com o HIP, deve-se digitar um endereço que na verdade mapeie para uma identidade HIP. O endereço utilizado foi o LSI, definido quando foi criada a identidade do servidor.

O `tcpdump` é um programa farejador (*sniffer*) de tráfego. Um farejador monitora e registra o tráfego de dados em uma interface específica. Para analisar todo o tráfego que passa por um segmento de rede, mesmo os pacotes que não são destinados à máquina que está rodando o `tcpdump`, a interface de rede deve operar em modo promíscuo. Os farejadores são úteis para a detecção e solução de problemas na rede. Entretanto, podem ser usados para capturar senhas e dados confidencias, que trafegam pela rede sem qualquer

proteção. O `tcpdump`, que é chamado via linha de comando, foi utilizado para observar o RTT e determinar quando a estação móvel consegue se comunicar com o servidor de vídeo. O `tcpdump` foi também utilizado para analisar os pacotes ICMP que eram enviados e recebidos e, a partir daí, calcular o tempo que a estação móvel ficou sem conexão.

### III.2.4 `ifconfig` e `route`

A ferramenta `ifconfig` é um programa de linha de comando utilizado para configurar os endereços IP de uma interface. Neste trabalho, o `ifconfig` foi utilizado para configurar o endereço IP após a mudança de ponto de acesso, pois cada ponto de acesso está em uma rede diferente. Poderia ser utilizada outra maneira de se configurar o endereço IP, como por exemplo, fazer uma requisição DHCP (*Dynamic Host Configuration Protocol*). Mas, para os fins do teste deste trabalho, é necessária maior rapidez na configuração do IP, para medir os tempos associados aos protocolos do HIP, o que justifica a solução adotada.

A ferramenta `route` é utilizada para gerenciar as rotas do dispositivo, seja ele um servidor, um roteador ou uma estação. A ferramenta de linha de comando foi utilizada para garantir que a estação utilizaria os roteadores como *gateways* e também para que o servidor de vídeo consiguisse enviar pacotes para as redes sem fio.

A utilização do `ifconfig` e `route` foi necessária para montar a topologia física da rede mostrada na Figura III.2.

### III.2.5 `iwconfig` e `iwlist`

A ferramenta `iwconfig` é utilizada para realizar conexões com os pontos de acesso sem fio. As informações necessárias para a configuração são: o nome do *Extended Service Set ID* (ESSID), o endereço MAC (*Medium Access Control*) da interface de rede do ponto de acesso e a frequência de transmissão. Para descobrir esses valores é necessário ou saber anteriormente ou fazer uma pesquisa nos arredores da estação móvel. A ferramenta `iwlist` serve para esse propósito. Ambas as ferramentas são utilizadas pela interface de

comando.

### III.2.6 `handoff.py`

A mudança de ponto de acesso é também conhecida como *handoff*, que inclui um processo de negociação com o novo ponto de acesso. Não existem ferramentas conhecidas, bem estabelecidas para realizar esse processo e com as peculiaridades dos testes desenvolvidos. Portanto, foi necessário desenvolver uma ferramenta para este fim.

A ferramenta `handoff.py` foi desenvolvida na linguagem de programação Python [30] e integra algumas funcionalidades de outras ferramentas existentes. Seu funcionamento básico é monitorar o enlace sem fio através da ferramenta `iwlist`. Quando certa condição definida for verificada, a troca de ponto de acesso é efetuada. Para tal, são utilizadas as ferramentas `iwconfig` para configurar o novo ponto de acesso, `ifconfig` e `route` para corrigir os endereços IP da estação móvel e do *gateway*.

O monitoramento do enlace sem fio, feito através da ferramenta `iwlist`, é realizado na interface de rede sem fio adicional da estação móvel (cartão PCMCIA). Isso é feito para que a monitoração não atrapalhe o tráfego na interface sem fio, pois a ferramenta que realiza a varredura de redes sem fio disponíveis não permite que a interface atue em outra atividade.

A variável utilizada para fazer a troca de pontos de acesso é a potência do sinal recebido na estação. Quando o sinal cai a um nível mais baixo que o sinal proveniente do outro ponto de acesso, é requisitado o *handoff*. Foi também definida uma histerese para garantir que não haja mudança de ponto de acesso devido a variâncias do nível de sinal. O código-fonte da ferramenta `handoff.py` encontra-se no apêndice A

# Capítulo IV

# Testes e Resultados

O ambiente de testes desenvolvido permitiu a realização de testes comparativos em dois cenários: usando HIP e não usando HIP.

Neste capítulo os testes são apresentados e os resultados são discutidos.

## IV.1 Testes com Ping

Foram feitos diversos testes com a ferramenta `ping`, para avaliar o desempenho do HIP frente ao desempenho sem o HIP.

### IV.1.1 Teste com Ping na Rede Cabeada

Primeiramente foram feitos testes para avaliar os atrasos provenientes do uso do HIP Para tanto, foram efetuados dois testes descritos a seguir.

O ambiente de testes utilizado está descrito na Seção III.1 e representado na Figura III.1. Para esse teste, foi alterado da seguinte maneira: a estação móvel foi conectada por meio de um cabo à interface ethernet do roteador Linksys ao invés de utilizar o enlace sem fio como no cenário anterior.

**Teste de RTT Usando o HIP**

O objetivo deste teste é verificar quanto tempo extra é necessário para o processamento de pacotes ao utilizar o HIP.

O teste é realizado da seguinte maneira: a estação móvel envia pacotes de requisição de eco ICMP (ICMP *Echo Request*) e aguarda os pacotes de resposta ao eco ICMP (ICMP *Echo Reply*).

Na primeira parte do teste, é medido o tempo de ida e volta do pacote (RTT) utilizando a pilha de protocolos TCP/IP convencional. Dessa forma, o pacote é enviado diretamente para a interface física de saída. Com esta configuração, o valor estimado para o RTT pela ferramenta `ping` corresponde ao tempo em que o pacote percorre os enlaces e filas dos caminhos de ida e volta.

Na segunda parte do teste é utilizada a pilha de protocolos TCP/IP com HIP. Neste teste, após a abertura de conexão onde ocorre a troca de bases, o pacote é enviado para a interface virtual TAP, que envia o pacote para o processo `hip`. O processo `hip` mapeia o identificador do destinatário para seu endereço IP correspondente e efetua a encriptação[1] do pacote utilizando ESP. Quando o pacote é recebido pelo destinatário, existe uma etapa de processamento semelhante onde o pacote é mapeado para a associação segura correta e decriptado.

O tempo medido neste teste inclui todo o processamento do pacote, pois as medidas são feitas nas interfaces virtuais.

A Figura IV.1 mostra os resultados do teste. O eixo das ordenadas mostra os valores de RTT obtidos em milissegundos. O eixo das abcissas mostra o instante (em milissegundos) que cada pacote de requisição de eco ICMP foi enviado. Os pacotes foram enviados a cada 10 milissegundos, para atingir a precisão desejada.É possível observar que o tempo de RTT médio na rede cabeada foi de 0,4 ms. Vale lembrar que o teste foi efetuado na rede interna do laboratório do Grupo de Teleinformática e Automação da UFRJ com somente um salto, na Internet são esperados valores de RTT significativamente maiores.

[1] A encriptação utilizada foi AES-CBC com HMAC-SHA1.

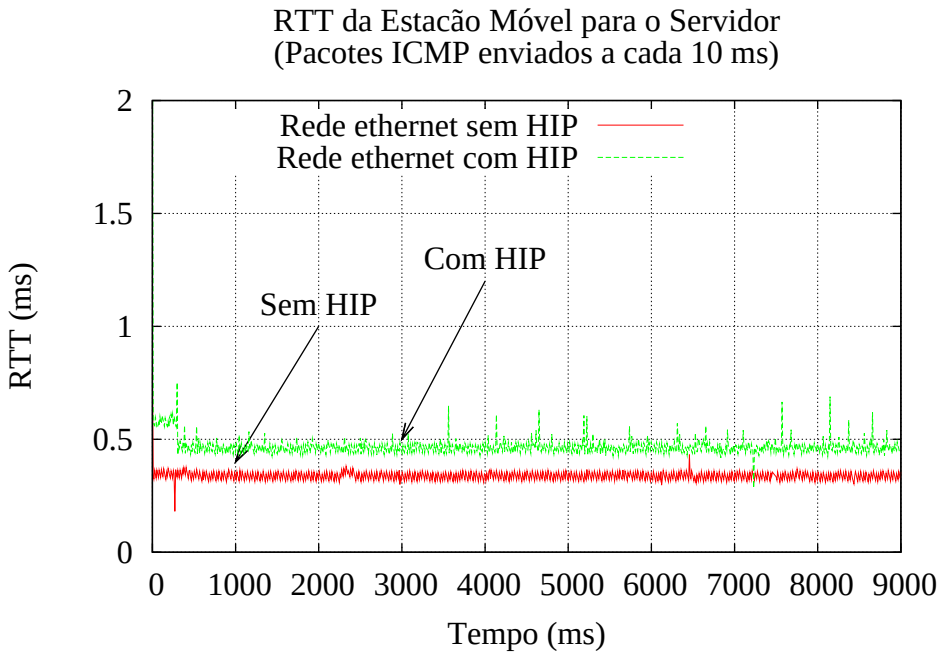



Figura IV.1: Tempo de ida e volta de um ping em uma rede cabeada usando HIP.

O tempo de RTT médio na rede cabeada com a utilização do HIP foi de 0,5 ms. Isso significa que o processamento dos pacotes HIP, inclusive encriptação e decriptação efetuadas ambas tanto no remetente quanto no destinatário (2 pacotes ICMP, requisição e resposta de eco), durou 0,1 ms em média. Com base nesses resultados, conclui-se que o atraso relacionado ao processamento por pacote é pequeno se comparado com os atrasos de trânsito na rede.

**Atualização de Endereço de Estações Móveis**

O objetivo desse teste é verificar o tempo necessário para a atualização de endereço de estações móveis.

O teste é realizado da seguinte maneira: a estação móvel envia continuamente pacotes de requisição de eco ICMP e aguarda os pacotes de resposta de eco ICMP. Em um dado momento a estação móvel modifica seu endereço IP para forçar a mensagem de atualização de endereço do HIP. Durante o período de atualização do endereço, os pacotes enviados para a estação móvel são perdidos e com base nessa informação é possível determinar o

tempo gasto até a atualização ser concluída.

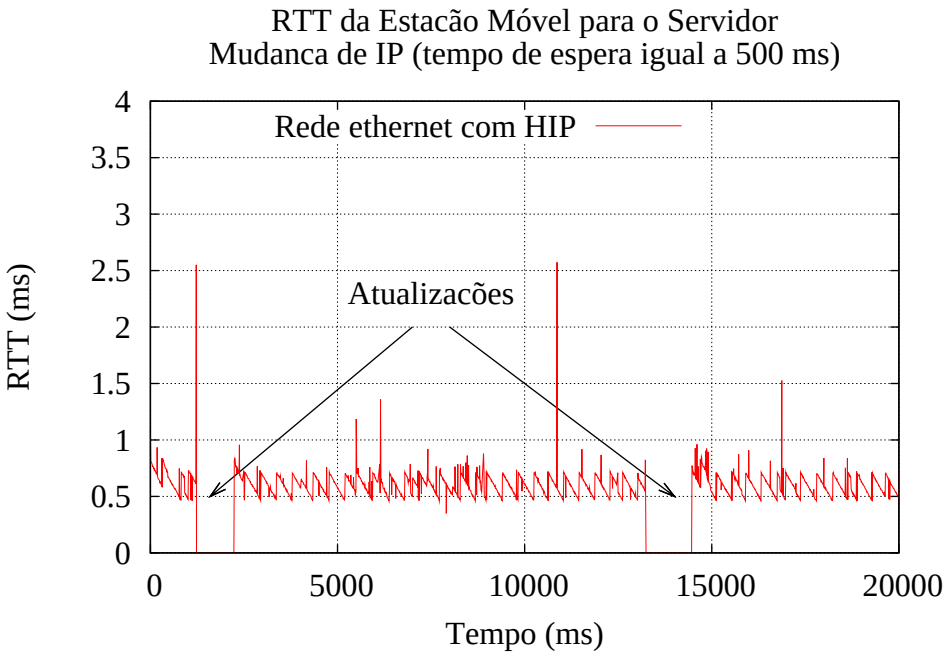



Figura IV.2: Tempo de atualização de endereço de estações móveis na rede cabeada.

A Figura IV.2 mostra os resultados do teste. O eixo das ordenadas mostra os valores de RTT obtidos em milissegundos. O eixo das abcissas mostra o instante (em milissegundos) que cada pacote de requisição de eco ICMP foi enviado. Os pacotes foram enviados a cada 10 milissegundos, para obter a precisão desejada. O endereço IP foi modificado duas vezes durante o experimento, nos instantes indicados na Figura IV.2. Nos instantes que seguem a atualização os pacotes ICMP de resposta são enviados ao endereço IP antigo da estação, motivo pelo qual não aparecem no gráfico.

Com base na análise dos resultados percebe-se que o tempo de atualização é da ordem de 1 segundo. O motivo pelo qual isso ocorre foi descoberto ao se analisar o código fonte da ferramenta OpenHIP. O processo `hip` que captura os pacotes HIP que chegam na interface de rede possui um mecanismo de espera. Este mecanismo de espera faz com que o processo `hip` espere por um tempo determinado antes de fazer o tratamento dos pacotes HIP. Isso foi feito para evitar o consumo excessivo de recursos computacionais com leituras excessivas no *buffer* de entrada de pacotes.

Tabela IV.1: Tempo do procedimento de atualização.

| Configuração | Média (ms) | desvio padrão (ms) |
|---|---|---|
| Tempo de espera de 500 ms | 996 | 8 |
| Tempo de espera de 100 ms | 216 | 23 |

O tempo de espera configurado por padrão é de 500 milissegundos. Relembrando o procedimento de atualização, que neste caso ocorre sem troca de chaves (Figura II.12), percebe-se que o procedimento é realizado com três pacotes, sendo dois enviados pela estação móvel em momentos distintos e um pelo servidor. Como o servidor espera por duas mensagens, espera um total de 1 segundo. Somente passado esse segundo, o servidor envia as respostas de eco ICMP para a estação móvel.

Experimentou-se também variar o parâmetro de amostragem do buffer da interface de rede para comprovar esse tempo de espera. Foi efetuado um teste com tempo de espera de 100 milissegundos. Foram efetuadas dez rodadas para se calcular o tempo total obtido. Os resultados estão na Tabela IV.1. Pode-se perceber que esse parâmetro afeta significativamente o desempenho da atualização de endereços.

O valor do parâmetro de amostragem utilizado nos testes posteriores foi o valor padrão, ou seja, 500 milisegundos.

### IV.1.2 Teste com Ping na Rede Sem Fio

Foram efetuados testes com e sem HIP para avaliar seu desempenho no ambiente sem fio. Dois testes foram feitos, o primeiro avalia o desempenho no enlace sem fio e o outro avalia a mobilidade, ou seja, efetua o *handoff* entre os pontos de acesso e a atualização de endereços.

**Atraso no Enlace Sem Fio**

Para obter uma estimativa da ordem de grandeza dos atrasos relacionados a rede

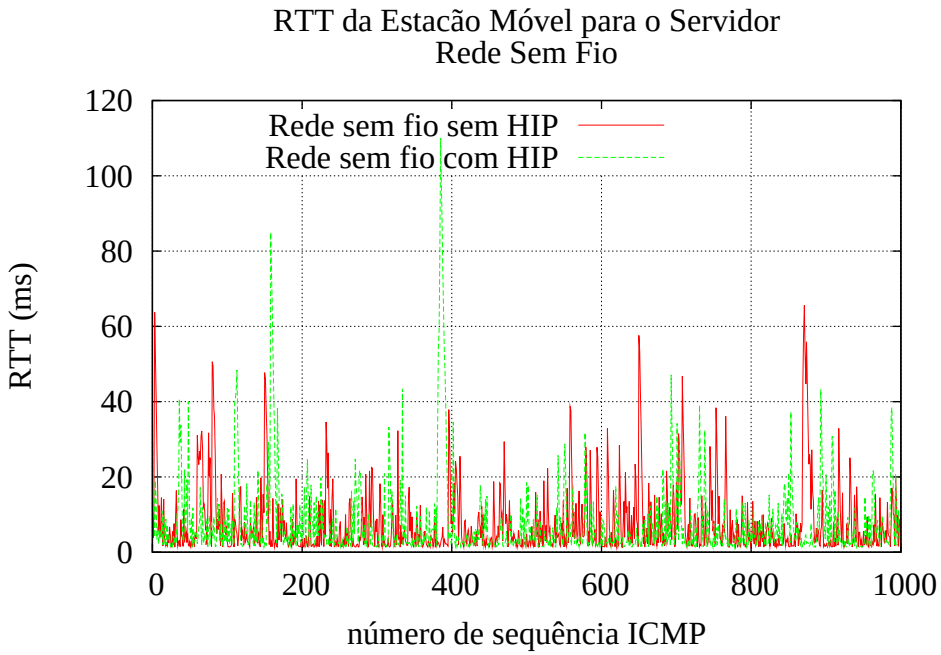



Figura IV.3: Tempo de Ida e Volta no enlace sem fio.

sem fio, foi efetuada a medição de RTT com e sem HIP. O resultado obtido está na Figura IV.3. O eixo das ordenadas mostra os valores de RTT obtidos em milissegundos. O eixo das abcissas mostra os números de sequência das mensagens de requisição de eco ICMP enviadas.

Percebe-se que a ordem de grandeza dos atrasos é bem superior à ordem de grandeza dos atrasos da rede cabeada e que há maior variação nos valores.

**Atualização de Endereço com *Handoff***

Foi realizado um teste para observar o impacto da mudança de ponto de acesso no processo de *handoff*. Para tal, foi efetuado o teste de mobilidade.

O usuário inicia o programa `ping`, que envia pacotes de requisição de eco ICMP, e a ferramenta `handoff.py`. O usuário então inicia uma caminhada em direção ao outro ponto de acesso. Em dado momento, a intensidade do sinal do segundo ponto de acesso supera a intensidade do sinal do primeiro ponto de acesso, e, ao detectar esse evento, a estação conecta-se ao segundo ponto de acesso.

O resultado obtido está na Figura IV.4. O eixo das ordenadas mostra os valores de RTT obtidos em milissegundos. O eixo das abcissas mostra o número de sequência das mensagens de requisição de eco ICMP enviadas.

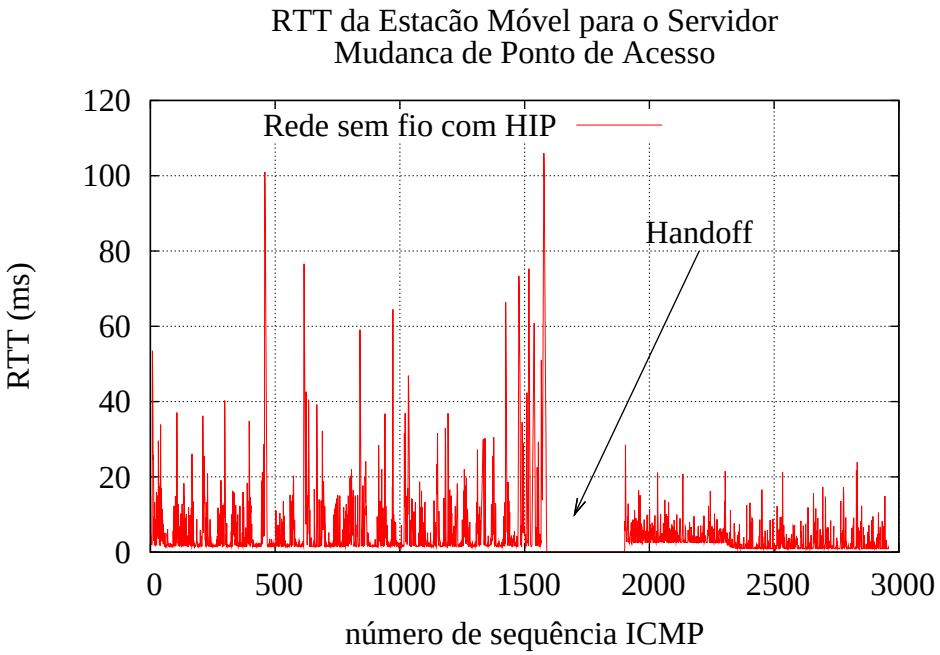



Figura IV.4: Atualização de endereço com o handoff.

Pode-se observar que o processo de *Handoff* e a troca de mensagens de atualização do HIP duraram cerca de 3 segundos. Esse tempo decorre do fato de que a troca de pontos de acesso não é instantânea, o que resulta em perdas de mensagens de atualização e, consequentemente, maior tempo até o processo de atualização de endereço ser concluído.

## IV.2 Teste de Exibição do vídeo

O teste de exibição de vídeo tem o objetivo de avaliar subjetivamente o impacto da mobilidade para um usuário. O teste foi efetuado para verificar se a mobilidade provoca perda na qualidade do serviço oferecido.

O teste foi realizado da seguinte maneira:

- o usuário encontra-se perto de um ponto de acesso, ao qual está inicialmente conectado;
- o usuário faz uma conexão HTTP com um servidor de vídeo;
- após o acúmulo de uma determinada quantidade de quadros no *buffer* o filme começa a ser exibido;
- em seguida o usuário inicia uma caminhada.
- quando a intensidade do sinal do outro ponto de acesso supera a intensidade do sinal do primeiro ponto de acesso, há a troca de pontos de acesso.

Depois da troca de pontos de acesso o filme continua a ser exibido, pois as conexões não foram perdidas nesse processo. O processo é representado na Figura IV.5.

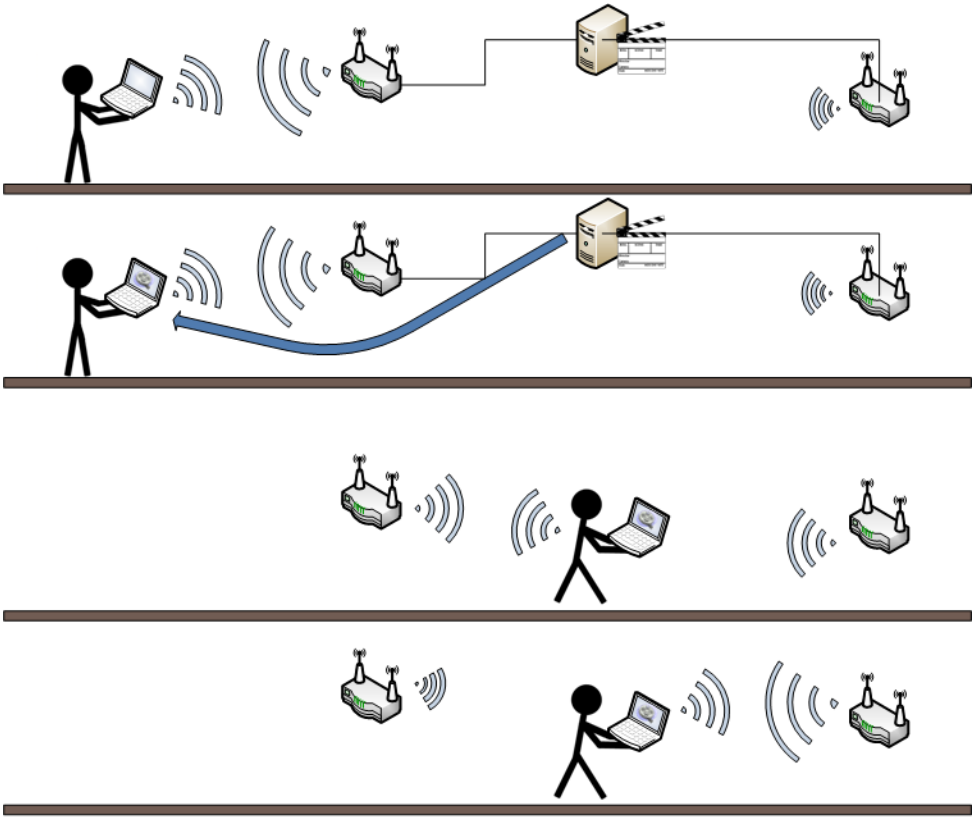


Figura IV.5: Representação do teste de mobilidade.

A avaliação subjetiva mostrou um grande impacto no vídeo que estava sendo exibido. O filme exibido ficou parado por cerca de 3 segundos. Entretanto após a troca de ponto de acesso o vídeo continuou a ser exibido sem impactos visíveis.

Algumas soluções já conhecidas poderiam ser utilizadas para prevenir o impacto na reprodução do vídeo, tal como aumento do *buffer* de exibição do vídeo. Esse tipo de solução é aceitável por se tratar de um cenário peculiar.

# Capítulo V

# Conclusão

O *Host Identity Protocol* (HIP) é uma proposta para resolver o problema da mobilidade na Internet e possibilitar o multidomicílio (*multihoming*). Com o Protocolo de Identificação de Estação (*Host Identity Protocol* - HIP), uma nova camada, a camada de identificação, é criada. Esta camada retira do IP o papel de identificação. O IP passa a ser somente o endereço topológico (localizador) de uma estação, enquanto que a camada de identificação cuida das funcionalidades relacionadas a identidade. Algumas dessas funções incluem a autenticação de pares, a segurança fim-a-fim, etc.

A separação dos papéis do IP de localizador e identificador causa a separação da camada de transporte da camada de rede. Com isso, é possível que uma camada mude sem afetar a outra. Portanto, com essa arquitetura é possível realizar a mobilidade. O endereço IP, ou o endereço topológico, pode ser modificado livremente sem que a conexão fim-a-fim seja afetada. As camadas superiores à camada de identificação não sentem nenhuma diferença devido à mobilidade. Da mesma maneira, é possível relacionar a camada de identificação a mais de um endereço IP, ou seja, é possível implementar o multidomicílio (*multihoming*).

Essa separação traz alguns benefícios como uma maior agregação dos endereços IP e, consequentemente, a redução das tabelas de roteamento. Com o HIP é possível associar os diversos endereços topológicos à identidade de um sítio com múltiplos acessos com

a Internet por diferentes provedores de serviço. Portanto, não é mais necessário criar diversas rotas para esse sítio.

Este trabalho teve como objetivo avaliar o HIP como solução de mobilidade para a Internet. Para isso foi desenvolvido um cenário de testes, que simula a mobilidade. Este cenário ilustra uma situação na qual um usuário que utiliza um serviço de *streaming* de vídeo realiza a mobilidade. O cenário constitui-se de uma estação móvel, utilizada para assistir o vídeo; dois pontos de acessos, utilizados com *gateways* para a estação móvel; e um servidor de faz o *streaming* do vídeo. Tanto a estação móvel quanto o servidor utilizam a ferramenta `OpenHIP` para implementar os protocolos do HIP. Esses dispositivos rodam o sistema operacional Linux. Os roteadores Linksys rodam o sistema operacional OpenWRT. Através desse cenário foi possível avaliar experimentalmente o desempenho dos protocolos da arquitetura HIP para a mobilidade.

Os experimentos realizados mostram que a utilização do HIP não acarreta atrasos significativos devido a processamento de pacotes. Além disso, o desempenho do protocolo de atualização de endereços mostrou-se aquém do esperado. Os atrasos medidos e percebidos na exibição do vídeo, com o uso da ferramenta `OpenHIP`, foram da ordem segundos. Esse tempo é elevado para diversas aplicações em tempo real como VoIP (*Voice over IP*) ou videoconferência. Entretanto, para exibição de vídeo sobre demanda, os atrasos podem ser contornados se forem utilizados alguns recursos, tais como o aumento do *buffers* de recepção.

Foi também testado um mecanismo para se realizar o *handoff* entre pontos de acessos. Para fins de teste, o mecanismo cumpre o papel. Entretanto, para se prover mobilidade na Internet entre pontos de acesso é necessário um mecanismo mais eficiente. O mecanismo implementado, faz o uso de duas interfaces de rede e monitora constantemente o enlace sem fio causando alto consumo energético. Logo, deve ser implementado um mecanismo que seja rápido e que evite perdas de pacotes para que o procedimento de atualização de endereços do HIP funcione corretamente. Outra preocupação é o gerenciamento eficiente de energia, que em qualquer dispositivo móvel é de suma importância.

Como trabalhos futuros pretende-se refinar o cenário de testes. Pretende-se testar e avaliar o impacto em mais aplicações como transferência de arquivo, conexões remotas, etc. Também serão testadas outras ferramentas que implementem HIP e serão criados mecanismos melhores e mais eficientes para se fazer o *handoff*.

# Referências Bibliográficas


[1] MARCELO D. D. MOREIRA, NATALIA C. FERNANDES, L. H. M. K. C. E. O. C. M. B. D., "Internet do Futuro: Um Novo Horizonte", *Minicursos do Simpósio Brasileiro de Redes de Computadores - SBRC'2009*, pp. 1–59, 2009.

[2] SIMPSON, W., *The Point-to-Point Protocol*. RFC 1661, IETF, Jul. 1994.

[3] DROMS, R., *Dynamic Host Configuration Protocol*. RFC 2131, IETF, Mar. 1997.

[4] HOLDREGE, P. S. E. M., *IP Network Address Translator (NAT) Terminology and Considerations*. RFC 2663, IETF, Aug. 1999.

[5] PEKKA NIKANDER, J. Y. E. J. W., "Integrating Security, Mobility, and Multi-Homing in a HIP Way". In: *Network and Distributed Systems Security Symposium (NDSS'03)*, pp. 87–99, Feb. 2003.

[6] J. ABLEY, K. LINDQVIST, E. D. B. B. E. V. G., *IPv4 Multihoming Practices and Limitations*. RFC 4116, IETF, Jul. 2005.

[7] NIKANDER, R. M. E. P., *Host Identity Protocol Architecture*. RFC 4423, IETF, May 2006.

[8] KAUFMAN, C., *Internet Key Exchange (IKEv2) Protocol*. RFC 4306, IETF, Dec. 2005.

[9] R. MOSKOWITZ, P. NIKANDER, P. J. E. T. H., *Host Identity Protocol*. RFC 5201, IETF, Apr. 2008.

[10] LAGANIER, P. N. E. J., *Host Identity Protocol (HIP) Domain Name System (DNS) Extension*. RFC 5205, IETF, Apr. 2008.

[11] ZEILENGA, K., *Lightweight Directory Access Protocol (LDAP): Technical Specification Road Map*. RFC 4510, IETF, Jun. 2006.

[12] PRENEEL, B., "The State of Cryptographic Hash Functions". In: *Lectures on Data Security, Modern Cryptology in Theory and Practice, Summer School, Aarhus, Denmark, July 1998*, pp. 158–182, London, UK, 1999.

[13] KENT, S., *IP Encapsulating Security Payload (ESP)*. RFC 4303, IETF, Dec. 2005.

[14] P. JOKELA, R. M. E. P. N., *Using the Encapsulating Security Payload (ESP) Transport Format with the Host Identity Protocol (HIP)*. RFC 5202, IETF, Apr. 2008.

[15] P. NIKANDER, J. L. E. F. D., *An IPv6 Prefix for Overlay Routable Cryptografic Hash Identifier (ORCHID)*. RFC 4843, IETF, Apr. 2007.

[16] BAGNULO, E. N. E. M., *Shim6: Level 3 Multihoming Shim Protocol for IPv6*. RFC 5533, IETF, Jun. 2009.

[17] LONVICK, T. Y. E. C., *The Secure Shell (SSH) Authentication Protocol*. RFC 4252, IETF, Jan. 2006.

[18] RESCORLA, E., *Diffie-Hellman Key Agreement Method*. RFC 2631, IETF, Jun. 1999.

[19] KOJO, T. K. E. M., *RFC3526 - More Modular Exponential (MODP) Diffie-Hellman groups*. RFC 3526, IETF, May 2003.

[20] S. FRANKEL, R. G. E. S. K., *The AES-CBC Cipher Algorithm and Its Use with IPsec*. RFC 3602, IETF, Sep. 2003.

[21] GLENN, C. M. E. R., *The Use of HMAC-SHA-1-96 within ESP and AH*. RFC 2404, IETF, Nov. 1998.

[22] P. NIKANDER, T. HANDERSON, C. V. E. J. A., *End-Host Mobility and Multihoming with the Host Identity Protocol*. RFC 5206, IETF, Apr. 2008.

[23] EGGER, J. E. L., *Host Identity Protocol (HIP) Rendezvous Extension*. RFC 5204, IETF, Apr. 2008.

[24] J. LAGANIER, T. K. E. L. E., *Host Identity Protocol (HIP) Registration Extension*. RFC 5203, IETF, Apr. 2008.

[25] "OpenWRT", Feb. 2004, `http://www.openwrt.org`.

[26] "OpenHIP", Mar. 2005, `http://www.openhip.org/`.

[27] A. ADAMS, J. N. E. W. S., *Basic HIP Extensions for Traversal of Network Address Translators*. Work in progress, <draft-ietf-hip-nat-traversal-09.txt>, IETF, Oct. 2009.

[28] J. MELEN, J. YLITALO, P. S. E. T. H., *Host Identity Protocol-based Mobile Router (HIPMR)*. Work in progress, <draft-melen-hip-mr-02>, IETF, May 2009.

[29] "VideoLAN", Feb. 2004, `http://www.videolan.org`.

[30] "Python", 1990, `http://www.python.org`.

[31] TANENBAUM, A. S., *Computer Networks*. Terceira ed. Prentice Hall Inc, 1996.

[32] AL-SHRAIDEH, F., "Host Identity Protocol". In: *International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies*, pp. 87–99, Feb. 2003.

[33] D. COOPER, S. SANTESSON, S. F. S. B. R. H. E. W. P., *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. RFC 5280, IETF, May 2008.

# Apêndice A

# Códigos Fonte

Neste apêndice todos os códigos fonte que foram gerados são mostrados.

Os códigos foram feitos para a criação da ferramenta `handoff.py`. A ferramenta está dividida em quatro arquivos listados a seguir.

- Arquivo principal `handoff.py`.
- Arquivo auxiliar `iwlist.py`.
- Arquivo auxiliar `iwconfig.py`.
- Arquivo auxiliar `cell.py`.

**Arquivo** `handoff.py`

```
#!/usr/bin/python
"""
Usage: python handoff.py [OPTION]
Do the Handoff based on the signal level of

  -h, --help                show this help
  -d, --dhcp                the IP request is made by dhcp
                                (this option excludes -p and -g)
  -y, --hysteresis          the signal level hysteresis used
```

```
                                to do the handoff def: 15 dbm
  -s, --scan-interval        the scan interval to analize the
                                signal level
  -i, --interface            the interface to use the scans
  -s, --scan-interface       the interface used to scan
  -e, --essids               the essid list to scan (separate them
                                by comma, no spaces)
  -p, --static-ips           the static ip list (separate them by
                                comma, no spaces)
  -g, --static-gateways      the static gw ip list (separate them
                                by comma, no spaces)

Examples:
  python handoff.py -y 20
  python handoff.py -s 2 --essids=teste1,teste2

"""
# Script that do the handoff automaticaly

from iwlist import Iwlist
from iwconfig import Iwconfig
import time
import sys
import getopt
import os

# Handoff class

class Handoff():

    def __init__(   self, s = 0.5, y = 15,
                    e = ['teste_hip-2','teste_hip-16'],
                    p=['192.168.2.115','192.168.16.115'],
                    g=['192.168.2.1','192.168.16.1'],
                    i = 'wlan1', c = 'eth1'):

        self.SCAN_INTERVAL = s
        self.HYSTERESIS = y
        self.ESSIDS = e
        self.INTERFACE = i
        self.SCAN_INTERFACE = c
        self.STATIC_IP_DICT = {}
        self.STATIC_IP_GW_DICT = {}
        if p and g:
            for ind in range(len(e)):
```

```
                self.STATIC_IP_DICT[e[ind]] = p[ind]
                self.STATIC_IP_GW_DICT[e[ind]] = g[ind]
        self.IWLIST_OBJ = Iwlist(self.SCAN_INTERFACE)
        self.IWCONFIG_OBJ = Iwconfig(self.INTERFACE)
        self.cellConnected = None
        print 'SCAN_INTERVAL = %2.3f' %
               self.SCAN_INTERVAL
        print 'HYSTERESIS = %u' % self.HYSTERESIS
        print 'INTERFACE = %s' % self.INTERFACE
        print 'STATIC_IP_DICT = %s' %
              self.STATIC_IP_DICT
        print 'STATIC_IP_GW_DICT = %s' %
              self.STATIC_IP_GW_DICT

    def run(self):
        print 'Running handoff'

        try:
            while 1:
                # sleep
                time.sleep(self.SCAN_INTERVAL)

                self.cellConnected =
                    self.IWCONFIG_OBJ.getCellConnected()
                if not self.cellConnected:
                    self.connect()
                    continue

                print '    "%s" : %s dbm' %
(self.cellConnected.getEssid(),
self.cellConnected.getSignalLevel())

                # run iwlist
                self.IWLIST_OBJ.run()

                cellList = []
                for essid in self.ESSIDS:
                    if essid in self.IWLIST_OBJ.cells:
                        cellList.append(
                            self.IWLIST_OBJ.cells[essid])


                elected = None
                for cell in cellList:
                    if cell.getEssid() !=
```

```
                        self.cellConnected.getEssid():
                    print '      "%s" : %s dbm' %
(cell.getEssid(), cell.getSignalLevel())
                    if cell.getSignalLevel() >
(self.cellConnected.getSignalLevel()
+ self.HYSTERESIS):
                        elected = cell

            if elected:
                self.connect(elected)
        except:
            raise

    def connect(self, elected = None):
        print '  Trying to connect...'
        try:
            if not elected:
                self.IWLIST_OBJ.run()
                cellList = []
                for essid in self.ESSIDS:
                    if essid in self.IWLIST_OBJ.cells:
                        cellList.append(
                             self.IWLIST_OBJ.cells[essid])

                for cell in cellList:
                    if elected:
                        if cell.getSignalLevel() >
                                  elected.getSignalLevel():
                            elected = cell
                    else:
                        elected = cell
            # connect
            if elected:
                self.IWCONFIG_OBJ.setEssid(elected.getEssid())
                self.IWCONFIG_OBJ.setAp(elected.getAddress())
                self.cellConnected =
                          self.IWCONFIG_OBJ.getCellConnected()
                if self.cellConnected:
                    os.popen('ifconfig %s %s' %
(self.INTERFACE,
self.STATIC_IP_DICT[elected.getEssid()])
).read()
                    os.popen('route add default gw %s' %
(self.STATIC_IP_GW_DICT[elected.getEssid()])
).read()
```

```
                    print '  Connected to "%s"' %
(self.cellConnected.getEssid())
                    print '  With signal level = %s' %
(self.cellConnected.getSignalLevel())
                else:
                    print '  Not connected!'
        except:
            if elected:
print '**  "%s" : %s dbm **' %
(elected.getEssid(), elected.getSignalLevel())
            raise

# the main procedure

def main():
    # parse command line options
    try:
        opts, args = getopt.getopt( sys.argv[1:],
                "hdy:s:i:c:e:p:g:",
                ["help", "dhcp", "hysteresis=",
                 "scan-interval=", "interface=",
                 "scan-interface=", "essids=",
 "static-ip=","static-gateway="])
    except getopt.error, msg:
        print msg
        print "for help use --help"
        sys.exit(2)
    # default options
    y = 15
    s = 0.5
    i = 'wlan1'
    c = 'eth1'
    e = ['teste_hip-2','teste_hip-16']
    p=['192.168.2.115','192.168.16.115']
    g=['192.168.2.1','192.168.16.1']
    # process options
    for o, a in opts:
        if o in ("-h", "--help"):
            print __doc__
            sys.exit(0)
        if o in ("-d", "--dhcp"):
            print "IP request by dhcp not implemented!"
            sys.exit(0)
        if o in ("-y", "--hysteresis"):
            y = int(a)
```

```
        if o in ("-s", "--scan-interval"):
            s = float(a)
        if o in ("-i", "--interface"):
            i = a
        if o in ("-c", "--scan-interface"):
            c = a
        if o in ("-e", "--essids"):
            e = a.split(',')
        if o in ("-p", "--static-ips"):
            p = a.split(',')
        if o in ("-g", "--static-gateways"):
            g = a.split(',')
    if ((not c) and (i)):
        c = i
    # process arguments
    #for arg in args:
    #    process(arg) # process() is defined elsewhere

    #print opts
    #print args
    handoff = Handoff(y=y,s=s,i=i,c=c,e=e,p=p,g=g)
    handoff.run()

if __name__ == "__main__":
    main()
```

**Arquivo** `iwlist.py`

```
#!/usr/bin/python

# Iwlist class
from cell import Cell


class Iwlist():

    def __init__(self,interface=None):
        self.interface = interface
        self.iwlistOutput = None
        #the list of cells
        self.cellList = None
        # a dictionary which the
#    key is the essid and the value is the cell
        self.cells = None
```

```
    def getIwlistOutput(self):
        return self.iwlistOutput

    def run(self):
        try:
            import os
            self.iwlistOutput =
                os.popen('iwlist %s scan' %
          (self.interface)).read()
            #iwlistLines = self.iwlistOutput.splitlines()

            self.cellList = []
            cellStrList = self.iwlistOutput.split('Cell')
            for cell in cellStrList:
                if 'Address' in cell:
                    self.cellList.append(Cell(cell))
            self.cells = dict( [(cell.getEssid(),cell)
for cell in self.cellList])
        except:
            raise
```

**Arquivo** `iwconfig.py`

```
#!/usr/bin/python

# Iwlist class
from cell import Cell


class Iwconfig():

    def __init__(self,interface=None):
        self.interface = interface
        self.iwconfigOutput = None
        #the list of cells
        self.cellConnected = None

    def getIwconfigOutput(self):
        return self.iwconfigOutput

    def getCellConnected(self):
        self.run()
        return self.cellConnected
```

```
    def run(self):
        try:
            import os
            self.iwconfigOutput =
                os.popen('iwconfig %s' %
        (self.interface)).read()
            if not 'ESSID' in self.iwconfigOutput:
                raise self.iwconfigOutput
            cell = Cell(self.iwconfigOutput,'iwconfig')
            if not cell.connected: cell = None
            self.cellConnected = cell
        except:
            self.cellConnected = None
            #raise

    def setEssid(self, essid):
        try:
            import os
            self.iwconfigOutput =
                os.popen('iwconfig %s essid %s' %
                (self.interface,essid)).read()
        except:
            raise

    def setAp(self, ap):
        try:
            import os
            self.iwconfigOutput =
                os.popen('iwconfig %s ap %s' %
        (self.interface,ap)).read()
        except:
            raise
```

**Arquivo** `cell.py`

```
#!/usr/bin/python

# Cell class

class Cell():

    def __init__(self,
            strCell=None,
```

```
            parseType='iwlist'):
        self.strCell = strCell
        self.cellNumber = None
        self.address = None
        self.essid = None
        self.signalLevel = None
        self.cellLines = None
        self.parseType = parseType
        self.connected = None

        if self.parseType == 'iwlist':
            self.parseOutput =
                self.parseIwlistOutput
        elif self.parseType == 'iwconfig':
            self.parseOutput =
                self.parseIwconfigOutput
        else:
            raise "Parse type does not exist."

        if self.strCell:
            self.parse()

    def setStrCell(self, strCell):
        self.strCell = strCell

    def getCellNumber(self):
        if not self.cellNumber: self.parse()
        return self.cellNumber

    def getAddress(self):
        if not self.address: self.parse()
        return self.address

    def getEssid(self):
        if not self.essid: self.parse()
        return self.essid

    def getSignalLevel(self):
        if not self.cellNumber: self.parse()
        return self.signalLevel

    def parseOutput(self):
        pass

    def parse(self):
```

```
        if not self.strCell:
            raise 'Cell string must be given.'
        self.parseOutput()

    def parseIwlistOutput(self):
        lines = self.strCell.splitlines()

        for line in lines:
            line = line.strip()
            if 'Address' in line:
                # get the cell number
                self.cellNumber =
                    int(line.split(' ')[0])
                # get the address
                self.address =
                    line.split(' ')[3]

            elif 'ESSID' in line:
                # get the essid
                self.essid =
                    line.split(':')[1].strip('"')

            elif 'Signal level' in line:
                # get the signal level
                self.signalLevel = int(
line.split(' ')[3].split('=')[1])

        self.cellLines = lines

    def parseIwconfigOutput(self):
        lines = self.strCell.splitlines()

        for line in lines:
            line = line.strip()
            if 'ESSID' in line:
                if 'unassociated' in line:
                    self.connected = False
                    return
                # get the essid
                self.essid =
                    line.split(':')[1].strip().strip('"')
                self.connected = True


            elif 'Access Point' in line:
```

```
                # get the address
                self.address =
                    line.strip().split(' ')[7]

            elif 'Signal level' in line:
                # get the signal level
                self.signalLevel = int(
line.strip().split(' ')[4].split('=')[1])

        self.cellLines = lines
```