4
Implementação e Ambiente de Simulação

Conforme apresentado no capítulo anterior, o protocolo HIP não estava implementado em nenhum framework do OMNeT++. Estudando os frameworks disponíveis para esse simulador, decidiu-se implementar esse novo protocolo no INETFramework.

Aproveitando a estrutura modular da ferramenta de simulação, o protocolo HIP foi implementado como um módulo simples e depois inserido na arquitetura entre a camada IP (composta, basicamente, pelos protocolos IPv6, ICMPv6, além do *Neighbour Discovery)*, e camada de transporte (composta, basicamente, pelos protocolos TCP e UDP), por meio da criação de um novo módulo composto.

Além da implementação dos mecanismos do protocolo HIP, foi preciso alterar alguns parâmetros e comportamentos de outros módulos no framework:

- O protocolo IPv6 implementa as portas para cada protocolo de transporte através do mapeamento do protocolo de trasnporte para uma porta num array. Para implementação HIP aumentou-se em uma unidade o tamanho do array de portas de transporte do módulo IPv6 para uso de tráfego HIP;
- Apesar da mobilidade definida pelo padrão IEEE 802.11 estar implementada, ou seja, apesar dos APs perceberem a entrada e saída de nós móveis da sua área de cobertura e aplicarem os processos de autenticação, associação e desassociação corretamente, não havia nenhum mecanismo implementado para que o nó móvel ao se associar a um novo AP obtivesse as informações necessárias da nova rede para atualizar sua tabela de endereçamento e sua rota default;
- Apesar do módulo UDP estar disponível, durante a execução dos testes verificou-se que existia um erro no tratamento de pacotes oriundos da camada IPv6. Esse erro foi resolvido;
- Dentro das aplicações UDP disponíveis não havia uma com características semelhantes a um tráfego VoIP, onde há períodos de

silêncio e de envio de informações. Para avaliar o comportamento desse tipo de tráfego com o protocolo HIP, desenvolveu-se uma aplicação UDP como esses requisitos;

- Quanto à definição do endereçamento IPv6 que cada nó presente na simulação assumiria, o framework funcionava apenas com um módulo de configuração que colocava todos os elementos numa mesma rede . Para testar o comportamento do protocolo HIP com a mudança do endereçamento IPv6, foi preciso implementar uma nova funcionalidade no módulo *RoutingTable6*.

4.1. Implementação do Protocolo HIP

Como apresentado no Capítulo 2, o protocolo HIP implementa a transparência de mobilidade IP gerenciando a troca do endereço IPv6 corrente pelo HIT nos datagramas destinados à camada de aplicação e, de forma inversa, ou seja, mapeando HITs em endereços IP nas mensagens destinados à camada de rede.

O novo módulo composto que representa a camada de rede com o protocolo HIP foi chamado de NetworkHIPLayer e é apresentado na figura 12.

O módulo HIP possui 10 portas, onde 5 são para entrada no módulo e 5 são para saída. Há uma porta de entrada e uma de saída para cada tipo de mensagem, a saber:

- mensagens TCP destinadas à camada IPv6;
- mensagens UDP destinadas à camada IPv6;
- mensagens IPv6 destinadas ao protocolo TCP;
- mensagens IPv6 destinadas ao protocolo UDP;
- mensagens HIP destinadas à camada IPv6;
- mensagens IPv6 destinadas à camada HIP.

A extensão do tamanho do array de portas do protocolo IPv6 foi feita nesse módulo para que as portas de entrada e saída definidas para a comunicação entre os módulos HIP e IPv6 fossem conectadas. As portas já definidas no módulo IPv6 para os protocolos de transporte TCP e UDP foram aproveitadas para conexão das portas HIP destinadas a esse tipo de comunicação.

Além das portas, o módulo HIP possui alguns parâmetros que devem ser passados através do arquivo ini durante a execução de uma simulação: procDelay, que define o tempo de atraso devido a processamento do módulo que se deseja simular; DNSFile, arquivo que possui as entradas necessárias para pesquisa de mapeamento HIT-IPv6; rvsHITAddress, endereço do Servidor Rendez-vous; e hipAddress, o endereço HIT que o dispositivo está usando na simulação.

A definição das portas e os parâmetros do módulo HIP encontram-se no arquivo HIP.ned que está localizado na pasta /Network/IPv6.

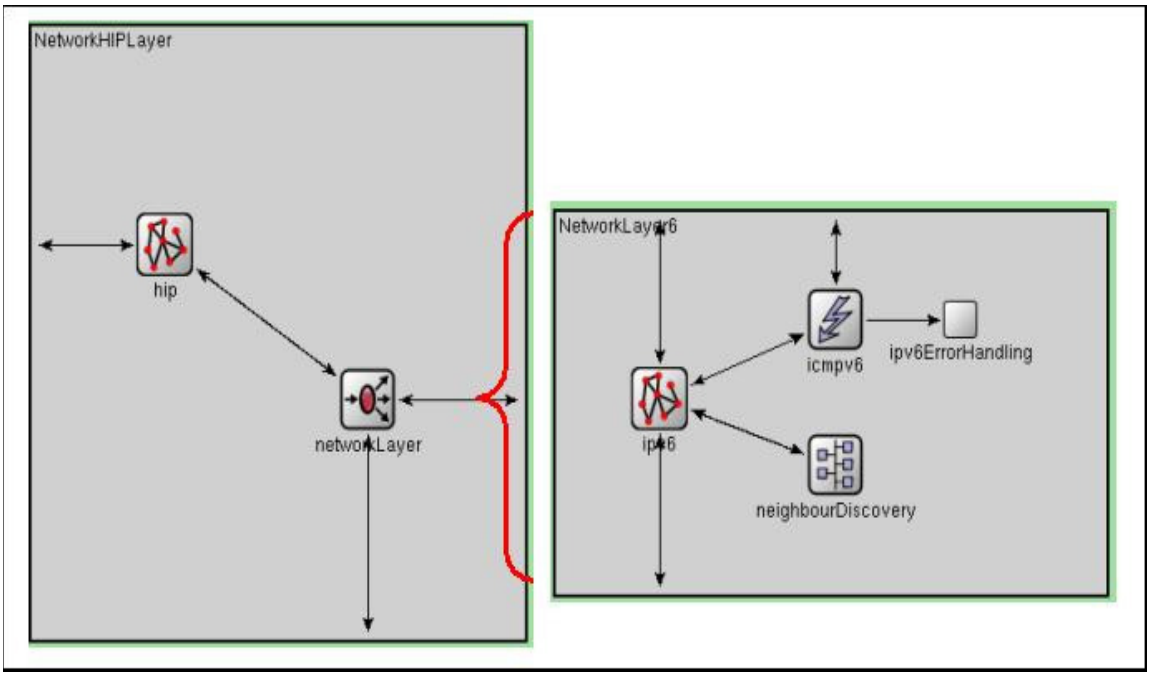

Figura 12: Novo módulo NetworkHIPLayer

A interface de comunicação entre a camada de rede e a camada de transporte foi aproveitada na implementação do protocolo HIP. Assim, estudou-se os campos definidos na estrutura *IPv6ControlInfo* utilizada na camada IPv6 para encaminhamento das mensagens aos protocolos de transporte para identificar o formato e o tipo da variável que continha o endereço IP de origem e destino, que a camada de transporte estava esperando uma vez que o endereço HIT passaria a exercer esse papel.

Para manter a compatibilidade do endereço HIT com a estrutura já existente na interface com a camada de transporte, o endereço HIT foi definido como a mesma estrutura utilizada para representar endereços IPv6, a *IPv6InterfaceData*. Apenas para efeito de identificação de quando a estrutura representa um endereço

IPv6 e quando representa um endereço HIT, definiu-se o uso de um prefixo único para todos os endereços HIT: “50:50:50:50”.

Além da definição de um novo módulo de rede, o NetworkHIPLayer, foi necessário gerar novas definições de dispositivos básicos e de nós móveis que utilizassem o novo módulo de rede. Essas novas definições são chamadas de StandardHostHIP6, WirelessHIPHost6 que estão localizados, respectivamente, em /Nodes/IPv6 e /Nodes/Wireless.

Ao deixar de utilizar o módulo *FlatNetworkConfigurator6,* utiliza um servidor DNS foi preciso criar um dispositivo que pudesse consultado sempre que surgisse a necessidade de identificar os endereços HIT e IPv6 de um nó. A solução definida nessa implementação do protocolo HIP foi a que prevê acesso dos dispositivos HIP a um servidor DNS ou um servidor Rendez-vous que possua as informações de mapeamento HIT – IPv6. Ao invés de implementar um servidor DNS, na simulação, todos os dispositivos são informados de seus endereços IPv6 e HIT, além do endereço do seu respectivo Servidor Rendez-vous.

Os outros endereços IPv6 são obtidos através de consultas ao Servidor Rendez-vous sempre que uma nova conexão é solicitada pela camada de aplicação e a pesquisa à tabela local de mapeamentos HIT-IPv6 retorna nulo. Nesse caso, o dispositivo que deseja iniciar a aplicação, coloca a mensagem da aplicação em uma fila e envia uma mensagem ao Servidor Rendez-vous solicitando o endereço IPv6 correspondente ao HIT informado. Quando o dispositivo recebe a resposta do Servidor Rendez-vous, ele retira a solicitação da aplicação da fila e envia a solicitação ao módulo HIP novamente.

O módulo HIP ao receber a mensagem da aplicação verificará se já existe alguma comunicação entre a origem e o destino especificados na mensagem e tomará as devidas providências para encaminhamento da mensagem ou estabelecimento de comunicação antes do envio da mesma segundo as informações obtidas nessa consulta. A partir daqui a comunicação entre os dispositivos envolvidos no fluxo de dados da aplicação é controlada pela máquina de estados do protocolo HIP.

O módulo que define o Servidor Rendez-vous é chamado de RVS_HIP e difere dos módulos de dispositivos básicos por não possuir portas de entrada e saída para a camada de transporte e por usar o módulo HIPRendezvous no lugar do módulo HIP na camada de rede.

O código do módulo HIPRendezvous difere do código HIP no envio de mensagens para estabelecimento de comunicação HIP entre o servidor Rendez-vous e os dispositivos que desejem se registrar nele e na simplificação do tratamento de mensagens entrantes decorrente da não existência de comunicação com as camadas de transporte e aplicação.

Nessa implementação, o Servidor Rendez-vous divulga apenas um serviço e apenas responde à pedidos de conexão; nunca as inicia. Também não existe falha na consulta aos serviços disponíveis nem no registro dos nós móveis.

A definição das mensagens de estabelecimento de comunicação entre dois dispositivos HIP e entre dispositivo HIP e seu Servidor Rendez-vous encontram-se no arquivo HIP.msg. É acrescido a essa mensagem a estrutura de dados *IPv6ControlInfo*, conforme explicado anteriormente.

No arquivo .msg apenas informamos os campos que compõem aquele tipo de mensagem com os respectivos valores esperados (tipo da variável). O OMNeT++ possui uma ferramenta que transforma essa definição em código C++ já implementando a construção e o desmonte da mensagem, bem como a inserção e a obtenção dos dados contidos em cada campo.

Os arquivos que contém o código fonte do protocolo HIP (HIP.cc, HIP.h, HIPRendezvous.cc, HIPRendezvous.h e HIP.msg) encontram-se no diretório /Network/IPv6.

Dessa forma, a implementação do protocolo HIP consiste em:

- Uma consulta a um dispositivo externo (Servidor Rendez-vous) que possui as informações de endereçamento dos demais nós que se registraram, caso não possua em sua tabela local essas informações;
- A definição dos estados do protocolo HIP e do mecanismo de controle dos mesmos que, por meio da troca de mensagens, estabelece e mantém comunicação com um ou mais dispositivos remotos. A troca dos endereços HIT pelos IPv6 e vice-versa que encontram-se na estrutura IPv6ControlInfo é de responsabilidade desse protocolo assim como o encaminhamento dos pacotes e datagramas recebidos para as portas corretas após a devida troca de endereçamento;
- Gerência da fila de mensagens que aguardam conclusão de estabelecimento de comunicação HIP e/ou resposta do Servidor

Rendez-vous com endereço IPv6 correspondente ao endereço HIT informado;

- O gerenciamento dessas conexões por meio do armazenamento de informações sobre a comunicação HIP que está estabelecida ou se estabelecendo em um vetor. Existe uma entrada para cada fluxo, ou seja, para cada nó existem duas entradas na tabela de comunicações HIP em curso: uma no sentido destino – origem e uma no sentido inverso.

Como o foco deste trabalho não é o aspecto de segurança na troca de mensagens ou no estabelecimento de conexão e como o framework não possuía a implementação do IPSec, não se fez uso desse protocolo na implementação do HIP. Ao invés disso, verifica-se através dos endereços IPv6 de origem e destino ou endereços HIT de origem e destino o estado da comunicação HIP para então definir o que será feito com a mensagem. O atraso de processamento desse protocolo pode ser introduzido utilizando-se o parâmetro procDelay do módulo HIP.

Após a definição da lógica básica de comportamento do protocolo HIP iniciou-se o processo de integração desse módulo ao framework.

Os principais pontos a serem respondidos eram: como receber a informação de que o link foi perdido e recuperado; quais mensagens devem ser solicitadas pelo dispositivo móvel ao estabelecer acesso numa nova rede; o que precisa acontecer na tabela de endereços de interfaces do dispositivo que perde a conexão com a rede e depois a re-estabelece; a nova rota default é aprendida e a antiga é descartada quando ocorre uma mudança de rede de acesso.

Para implementar isso, foi preciso verificar os triggers existentes na implementação do IEEE 802.11 e, através do uso do módulo *Notification Board*, implementar no módulo HIP as seguintes ações na ocorrência desses triggers:

- NF_L2_BEACON_LOST: armazenamento do endereço IPv6 numa variável, retirada de endereço IPv6 referente ao HIT do dispositivo em questão da tabela local de mapeamento HIT-IPv6, liberação de endereço IPv6 do AP antigo e remoção da rota default e das entradas do destCache;

- NF_L2_ASSOCIATED: envio de solicitação de mensagem RouterAdvertisement para configuração de novo endereço IPv6 na nova rede;
- NF_IPv6_INTERFACECONFIG_CHANGED: obtenção de novo endereço, atualização de endereço IPv6 referente ao HIT do dispositivo em questão na tabela local de mapeamento HIT-IPv6, envio de mensagem Update para todos os dispositivos que possuem comunicação estabelecida, usando para isso informação de endereço IPv6 antigo armazenado na variável na perda de conexão L2 e o novo endereço.

No módulo *Neighbour Discovery* foi implementado o envio do trigger NF_IPv6_INTERFACECONFIG_CHANGED toda vez que fosse gerado um novo endereço IPv6 para a interface WLAN.

Como não foi utilizado o módulo *FlatNetworkConfigurator6*, foi preciso inserir no módulo *RoutingTable6* uma função que executa um parsing do arquivo XML utilizado pela simulação onde são definidas as informações de endereçamento e características das interfaces, tais como: habilitação ou não de envio de *Router Advertisements*, habilitação ou não de roteamento, entre outras; e de rotas que não serão divulgadas pelas mensagens de *Router Advertisements* enviadas pelos roteadores adjacentes.

Para verificar se os endereços IPv6 e HIT fornecidos que serão utilizadas na criação de mensagens e preenchimento da estrutura IPv6ControlInfo são válidos, utiliza-se as funções de IPAddressResolver. Porém o tipo de retorno dessas funções não era compatível com o tipo de variáveis definidas no protocolo HIP. Por essa razão replicou-se as funções de busca de endereços IPv6 e de remoção desses endereços na tabela de interfaces alterando o tipo de retorno de IPvXAddress para IPv6Address.

Além disso, para que o protocolo HIP seja considerado um protocolo válido, é preciso que ele esteja listado no arquivo IPProtocolId.msg. Foi inserido ao final da lista nesse arquivo um nome para o protocolo HIP (IP_PROT_IPv6_HIT) e seu respectivo número (253).

Outro arquivo do framework que foi alterado para garantir o funcionamento do protocolo HIP foi o netconf2.dtd. Nele foram inseridos os tags <hip> </hip> com as mesmas funcionalidades dos tags <interface> </interface> para que a

função que aplica parsing aos arquivos XML não tive problemas na leitura do arquivo que contém as informações de DNS a serem aprendidas pelos dispositivos HIP.

Para obtenção de informações a respeito do tráfego UDP tipo VoIP, ou seja, tráfego alternando momentos de "fala" (transmissão de dados – período on) com momentos de "silêncio" (sem transmissão de dados – período off). Esses períodos foram definidos como parâmetros que podem receber como valor contantes numéricas ou funções de distribuição exponenciais e randômicas.

Além desses parâmetros, é possível definir o endereço IPv6 do dispositivo de destino, a porta de origem e de destino da aplicação, o tempo de início de envio de pacotes, o tamanho dos pacotes e o intervalo entre os pacotes.

Para armazenamento de informações sobre as mensagens enviadas, foram geradas estruturas adicionais de armazenamento de informações sobre o tráfego ao arquivo de definição de mensagem UDPVideoStreamMsg para inserir na mensagem envia o tempo de criação da mesma. No módulo receptor, no caso UDPSink, foram criados vetores que armazenam o número de seqüência das mensagens recebidas e as informações de tempo de criação da mensagem e de recebimento da mesma. Essas informações permitem a verificação do delay fim a fim das mensagens e da quantidade de perdas dos mesmos. Fazendo a diferença de tempo de chegada entre a última mensagem recebida na rede antiga e a primeira mensagem recebida na rede nova, tem-se o tempo de handoff.

Além disso, no protocolo HIP adicionou-se um vetor para armazenamento de informações de ocorrência de *handoff:* contagem do número de handoffs e a diferença de tempo entre ocorrência de perda de conexão L2 e re-estabelecimento de conexão L3.

Algumas facilidades do HIP não foram implementadas, a saber:

- Solicitação de fechamento de comunicações HIP: as comunicações estabelecidas ficam ativas até o fim da simulação;
- CBA: os nós ao receberem uma mensagem de Update, passam a usar o endereço informado logo que verificam a acessibilidade do nó através do mesmo;
- Mensagens ICMPv6 para notificações de erros.

4.2.
Cenário de Simulação

O cenário definido para simulação é apresentado na figura 13:

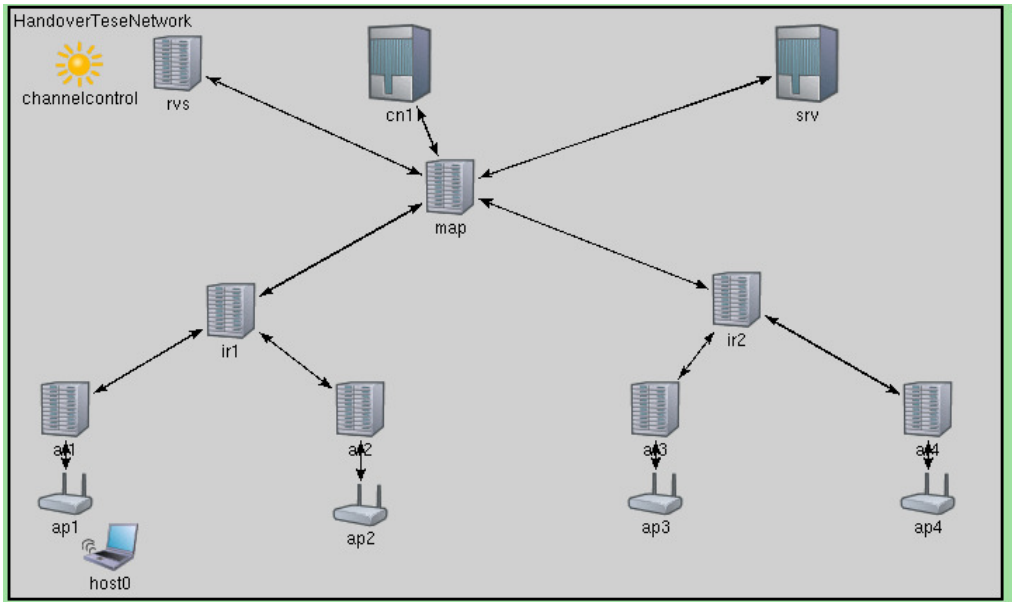

Figura 13: Cenário de simulação do protocolo HIP no INETFramework

A esse cenário são acrescentados outros nós móveis para verificar o comportamento do protocolo com compartilhamento de acesso e tráfego concorrente. O valor limite utilizado nesse estudo foi o de 50 nós móveis.

O nó móvel host0 passa pelos quatro APs usando um trajeto bem definido e uma velocidade de locomoção fixa, enquanto os outros dispositivos móveis se movimentavam aleatoriamente e com velocidade que varia dentro de um intervalo apenas para forçar o compartilhamento da banda disponível e dos recursos dos demais dispositivos que compõem o cenário.

O cenário é composto por um roteador central, identificado como MAP, conectado a dois roteadores, os roteadores IR1 e IR2, e aos servidores existentes, CN1, RVS e SRV. Cada um desses rotedores intermediários está conectado a dois outros roteadores, AR1, AR2, AR3 e AR4, que por sua vez estão conectados a access point, AP1, AP2, AP3 e AP4.

Os dispositivos móveis que participam do cenário estão recebendo dados de CN1 ou enviando dados para CN1. O dispositivo móvel que está sendo considerado para análise, host0, recebe informações do servidor SRV que gera o tráfego usando a aplicação UDP semelhante ao tráfego VoIP. A aplicação UDP que foi iniciada no servidor SRV é a UDPSink.

Os resultados da simulação são armazenados de duas formas:

- Em variáveis de coleta de estatísticas, que vão armazenando informações ao longo da simulação e
- No log gerado durante a execução da simulação.

As variáveis de coleta de estatísticas são implementadas no código C++ dos próprios módulos e podem ser tratadas por ferramentas existentes no OMNeT++: *Scalars* [21] e *Plove* [21]. A ferramenta *Scalars* permite a visualização das variáveis estatísticas escalares que servem como contadores de tipos de pacotes ou informações de protocolos ao longo da simulação. Já a ferramenta *Plove* analisa os dados contidos em vetores que são preenchidos ao longo da simulação permitindo uma visão do comportamento de determinadas facilidades ao longo do tempo.

Além dos formatos de armazenamento disponibilizados pelo próprio framework, os vetores inseridos na aplicação UDP que gera tráfego semelhante ao tráfego VoIP, chamada de UDPAppOnOff, também geram arquivos no formato csv que são tratados posteriormente.