Universidade do Minho
Escola de Engenharia

João Carlos Dias Veiga

Uma solução IPSec para comunicações seguras Anycast em redes IPv6

Outubro de 2011

João Carlos Dias Veiga
Uma solução IPSec para comunicações seguras Anycast em redes IPv6

UMinho | 2011

Universidade do Minho
Escola de Engenharia

João Carlos Dias Veiga

Uma solução IPSec para comunicações seguras Anycast em redes IPv6

Dissertação de Mestrado
Mestrado em Engenharia de Comunicações

Trabalho realizado sob a orientação do
Professor Doutor António Costa
e do
Professor Doutor Alexandre Santos

Outubro de 2011

Agradecimentos

Completar uma tese é um desafio e agradecer a todos aqueles que para ela contribuíram é também um. Tive a sorte de encontrar pelo caminho muitas pessoas que, de algum modo, tornaram isto possível, e quero portanto, agradecer a todos eles incluindo aqueles que não são aqui mencionados pelo nome.

Em primeiro lugar quero agradecer aos professores António Costa e Alexandre Santos pela orientação, por estarem sempre acessíveis para me ouvir e dar-me ideias que me permitiram realizar a minha tese de dissertação.

A todos os meus colegas de MIECOM.

Para os meus amigos mais próximos da universidade, Francisco Silva, Rui Vaz, Carlos Carneiro, Pedro Queirós, Nélson Marques, Rui Ferraz e Óscar Bravo, obrigado por todos os momentos que passamos juntos ao longo destes anos e pela vossa amizade!

Aos meus amigos de sempre, Rui, Duarte, Nelinho, Mário, Eduardo e Bruno, obrigado por estarem sempre ao meu lado e por serem os meus melhores amigos!

Para a Helena, muito obrigado por todo o apoio, amizade e amor, sem ti não seria a pessoa que sou hoje!

Para terminar, gostaria de agradecer e dedicar esta tese aos meus pais e irmã, por sempre acreditarem no meu trabalho e por todo o apoio e encorajamento pela minha vida inteira, sem o qual nada disto poderia ser possível.

Resumo

Esta dissertação tem como objectivos a análise das tecnologias IPSec e *Anycast*, o teste de implementações em cenários reais, e o desenvolvimento duma solução que permita aliar o balanceamento de carga com a segurança: o balanceamento de carga é obtido utilizando o *Anycast* e a segurança através do IPsec. Esta solução foi testada num cenário prático e concluiu-se sobre os seus resultados. A solução foi implementada utilizando o protocolo de *Internet* IPv6.

Inicialmente é efectuado um levantamento do estado de arte das tecnologias que assumem maior relevância: (1) IPv6; (2) balanceamento de carga; (3) IPSec; (4) e o *Anycast*. De seguida, testam-se as implementações existentes das tecnologias *Anycast* e IPSec. Inicialmente testam-se ambas as tecnologias em separado, e depois conjuntamente em cenários reais. Relativamente ao modelo de comunicação *Anycast*, foi verificado: (1) suporte em termos de SOs; (2) comportamento com os protocolos TCP e ICMPv6; (3) balanceamento de carga; (4) e redireccionamento em caso de falha. Em termos de IPSec, foi verficado: (1) suporte em termos de SOs; (2) ferramentas existentes para a sua configuração; (3) e ainda diferentes modos de configuração do IPSec. Após estes testes foram ainda implementadas as tecnologias *Anycast* e IPSec em simultâneo. Através destes testes, observou-se que sem alterações às tecnologias existentes o IPSec não pode ser utilizado conjuntamente com o *Anycast* se for configurado através do IKE, funcionando apenas caso o IPSec fosse configurado manualmente.

Nesta dissertação propõe-se uma solução capaz de permitir comunicações seguras entre um cliente e um conjunto de servidores com um mesmo endereço *Anycast*. A solução proposta é totalmente baseada no IPSec e a sua utilização não implica nenhuma alteração às tecnologias utilizadas. Visto não ser possível a utilização do IKE com o *Anycast*, a solução implementada utiliza o modo de configuração manual do IPSec para fornecer a segurança às comunicações. No entanto, o modo de configuração manual introduz sérios problemas de segurança no momento da troca das chaves secretas entre os dois extremos de uma conexão. Como tal, a solução desenvolvida implementa um mecanismo de troca das chaves secretas de um modo seguro. A segurança deste mecanismo é conseguida através de criptografia assimétrica.

A solução consiste em duas aplicações: `cliente` e `servidor`. As aplicações foram desenvolvidas na linguagem de programação `JAVA`. A aplicação `cliente` oferece ainda um GUI para a configuração dos parâmetros pretendidos para a conexão IPSec.

Abstract

This thesis aims to analyze the IPSec and Anycast, testing implementations in real scenarios, and developing a solution that will combine load balancing with security: the load balancing is achieved using the Anycast and security through IPsec. This solution was tested in a practical setting and found out about their results. The solution was implemented using the Internet protocol IPv6.

Initially a survey is made about the state of the art of the technologies that assume greater importance: (1) IPv6; (2) load balancing; (3) IPSec; (4) and Anycast. Then, we test existing implementations of the technologies Anycast and IPSec. First we test both technologies separately, and then together in real scenarios. For the Anycast communication model, was checked: (1) OS support; (2) behavior with TCP and ICMPv6 protocols; (3) load balancing; (4) and redirecting on failure. In terms of IPSec was checked: (1) OS support; (2) existing tools for configuration; (3) and about different ways to configure the IPSec. After these tests were also implemented simultaneously the Anycast and IPSec technologies. Through these tests, we found that with no changes to existing technologies IPSec can not be used in conjunction with the Anycast if configured via IKE. IPSec only works if it was manually configured.

Here, we propose a solution that can provide secure communications between a client and a set of servers with the same Anycast address. The proposed solution is completely based on IPSec and its use does not require any change to the technology used. Since it is not possible to use IKE with Anycast, the solution implemented uses the manual setting of IPSec to provide security to communications. However, the mode of manual configuration introduces serious security problems at the time of exchange of secret keys between two ends of a connection. As such, the solution developed implements a mechanism for the exchange of secret keys in a secure way. The security of this mechanism is achieved through asymmetric encryption.

The solution consists of two applications: `cliente` and `servidor`. The applications were developed in the `JAVA` programming language. The application `cliente` also offers a GUI for setting the desired parameters for the IPSec connection.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Código

Acrónimos

3DES	**T**riple **D**ata **E**ncryption **S**tandard
AARP	**A**nycast **A**ddress **R**esolving **P**rotocol
AES	**A**dvanced **E**ncryption **S**tandard
AH	**A**uthentication **H**eader
AOSPF	**A**nycast **O**pen **S**hortest **P**ath **F**irst
API	**A**pplication **P**rogramming **I**nterface
ARD	**A**nycast **R**eceiver **D**iscovery
ARIP	**A**nycast **R**outing **I**nformation **P**rotocol
AS	**A**utonomos **S**ystems
BGP	**B**order **G**ateway **P**rotocol
CAS	**C**ertificate **A**uthentication **S**erver
CIDR	**C**lassless **I**nter-**D**omain **R**outing
CORE	**C**ommon **O**pen **R**esearch **E**mulator
DES	**D**ata **E**ncryption **S**tandard
DDoS	**D**istributed **D**enial **o**f **S**ervice
DLL	**D**ynamic **L**inkable **L**ibrary
DNS	**D**omain **N**ame **S**erver
DoS	**D**enial **o**f **S**ervice
EGP	**E**xterior **G**ateway **P**rotocol
EIGRP	**E**nhanced **I**nterior **G**ateway **R**outing **P**rotocol
ESP	**E**ncapsulating **S**ecurity **P**ayload
G-CGA	**G**roup **C**ryptographically **G**enerated **A**ddresses
GUI	**G**raphical **U**ser **I**nterface
HMAC-SHA1	**H**ash-**B**ased **M**essage **A**uthentication **C**ode **S**ecure **H**ash **A**lgorithm-1

HOPS	**H**ost **P**roximity **S**ervice
ICMP	**I**nternet **C**ontrol **M**essage **P**rotocol
ICMPv6	**I**nternet **C**ontrol **M**essage **P**rotocol **V**ersion **6**
IETF	**I**nternet **E**ngineering **T**ask **F**orce
IGP	**I**nterior **G**ateway **P**rotocol
IKE	**I**nternet **K**ey **E**xchange
IPSec	**IP Sec**urity
IPv4	**I**nternet **P**rotocol **V**ersion **4**
IPv6	**I**nternet **P**rotocol **V**ersion **6**
MLD	**M**ulticast **L**istener **D**iscovery
NAT	**N**etwork **A**ddress **T**ranslation
OSPF	**O**pen **S**hortest **P**ath **F**irst
OSPFv3	**O**pen **S**hortest **P**ath **F**irst **V**ersion **3**
PIA-SM	**P**rotocol **I**ndependent **A**nycast-**S**parse **M**ode
PIM-SM	**P**rotocol **I**ndependent **M**ulticast-**S**parse **M**ode
PKI	**P**ublic **K**ey **I**nfrastructure
QoS	**Q**uality **o**f **S**ervice
RFC	**R**equest **F**or **C**omments
RIP	**R**outing **I**nformation **P**rotocol
RIPng	**R**outing **I**nformation **P**rotocol **n**ext **g**eneration
RP	**R**endezvous **P**oint
RSA	**R**ivest, **S**hamir, **A**dleman
SA	**S**ecurity **A**ssociation
SAD	**S**ecurity **A**ssociation **D**atabase
SATP	**S**ecure **A**nycast **T**unneling **P**rotocol
SP	**S**ecurity **P**olicy
SPD	**S**ecurity **P**olicy **D**atabase
SPF	**S**hortest **P**ath **F**irst
SPI	**S**ecurity **P**arameters **I**ndex
SRTP	**S**ecure **R**eal-**T**ime **T**ransport **P**rotocol
SSL	**S**ecure **S**ocket **L**ayer
TCP	**T**ransmission **C**ontrol **P**rotocol

UDP	**U**ser **D**atagram **P**rotocol
VLSM	**V**ariable-**L**ength **S**ubnet **M**asking
VPN	**V**irtual **P**rivate **N**etwork

Capítulo 1

Introdução

Neste capítulo da dissertação é efectuado o enquadramento do problema abordado, são demonstradas as motivações que nos levaram a dissertar sobre este problema e ainda apresentados os objectivos que nos propomos alcançar. São também apresentados os contributos desta dissertação. Para terminar este capítulo é efectuado uma descrição da estrutura da dissertação.

1.1 Enquadramento

A evolução da *Internet* e o rápido crescimento do seu número de utilizadores, levou a que desde cedo fosse previsto o esgotamento dos endereços IP disponibilizados pelo protocolo de *Internet* IPv4 [1]. Da necessidade de ter um espaço de endereçamento capaz de suportar este crescimento, foi desenvolvido o protocolo de *Internet* IPv6 [2]. O protocolo IPv6 é resultante de um esforço do IETF (*Internet Engineering Task Force*) no sentido de criar a "próxima geração do protocolo de *Internet*". O IPv6 para além de aumentar significativamente o espaço de endereçamento, de aproximadamente 4×10^9 para cerca de $3,4 \times 10^{38}$ endereços, possui ainda melhorias nas áreas de segurança, mobilidade e desempenho [2]. O formato do endereço IPv6 sofreu também alterações, passando dos 32 bits do IPv4 para 128, sendo então organizado como 8 grupos de 16 bits, do tipo 2001:690:2280:20:0000:0000:1234:56ab em notação hexadecimal. Um novo tipo de endereço foi ainda definido neste protocolo, o endereço *anycast*, que permite o envio de informação para um nó dentro de um grupo. Uma das principais alterações do IPv6 consiste num novo formato do cabeçalho. Algumas das motivações para esta alteração consistem no

aumento da velocidade de processamento de pacotes e em dotá-los de marcadores capazes de melhorar o serviço em termos de *Quality of Service* (QoS) ao nível do encaminhamento. As melhorias ao nível da segurança englobam a introdução do `IPSecurity` (IPSec) [3]. Através do IPSec é possível garantir autenticidade, confidencialidade, integridade, e ainda proteger uma conexão contra ataques por repetição. A implementação do IPv6 tem sido efectuada de um modo gradual, operando em simultâneo com o IPv4, sendo já suportado por inúmeras aplicações por todo o mundo.

Existem nos dias de hoje diversas situações, no âmbito da *Internet*, em que clientes precisam de localizar conteúdos ou serviços prestados pela rede. Tipicamente, estes conteúdos ou serviços estão localizados num servidor e o seu acesso está limitado pela largura de banda da rede e pelos recursos do próprio servidor, podendo consequentemente levar a uma degradação do serviço prestado ao cliente se o numero de clientes aumentar muito. De modo a contornar estas limitações, tornou-se prática comum replicar os servidores que prestam os serviços, aumentando a disponibilidade dos serviços e ainda distribuindo a carga pelas réplicas. Estes servidores podem ser classificados em réplicas locais ou réplicas distribuídas [4] . As réplicas locais, tipicamente utilizadas em *clusters*, podem estar contidas numa única sub-rede, enquanto que as réplicas distribuídas podem estar geograficamente distribuídas na *Internet* sendo este último modelo o objecto de estudo desta dissertação. Nas réplicas distribuídas, para que os clientes possam localizar o servidor apropriado, é necessário melhorar os mecanismos de localização utilizados para direcionar os clientes para os serviços. A utilização de um servidor DNS (*Domain Name Server*) [5], de um *Distributed Director* [6], ou de um *Host Proximity Service* (HOPS), são algumas das abordagens utilizadas para distribuir a carga pelos servidores utilizando o modelo de comunicação *Unicast*. No entanto, outra alternativa é a utilização de comunicações *Anycast*. Através do *Anycast* é possível suportar a replicação distribuída de servidores, distribuir a carga pelos mesmos, simplificar uma aplicação de rede e ainda satisfazer diferentes requisitos de QoS [7]. Enquanto que as primeiras soluções apresentadas são da camada aplicacional implicando mais processamento, o *Anycast* é uma solução ao nível da camada de rede não implicando o processamento existente nas restantes soluções. A descoberta de servidores DNS é um exemplo de uma utilização actual do modelo de comunicação *anycast*. Se um conjunto de servidores DNS utilizarem o mesmo endereço *Anycast*, no momento em que um cliente efectuar um pedido a um servidor DNS através da comunicação *Anycast*, o seu pedido será encaminhado para o servidor ”mais próximo”.

O *Anycast* é um novo modelo de comunicação nos standards do protocolo de *Internet* IPv6, tal como o *Unicast* e o *multicast*. Este novo modelo de comunicação caracteriza-se pelo facto da informação enviada para um endereço de rede *Anycast* ser recebida por um único nó de um conjunto de nós com este endereço. Na Figura 1.1 é possível observar o modelo de comunicação *Anycast*. Ao contrário do *multicast* que possui uma filosofia de "um para muitos" (*one to many*), ou seja, um pacote enviado para um endereço *multicast* é recebido por todos *hosts* com este endereço, o *Anycast* possui uma filosofia de "um para qualquer um" (*one to any*) [8]. Um endereço *Anycast* pode ser atribuído a um ou a mais nós de rede, tornando-se deste modo possível a um conjunto de servidores replicados que forneçam o mesmo serviço, possuir o mesmo endereço *Anycast*. No entanto, é também importante realçar que dois pacotes endereçados ao mesmo endereço *Anycast* podem chegar a dois servidores diferentes, tornando este modelo inadequado para cenários onde é necessário estabelecer conexões com os servidores de modo a manter um fluxo de informação. Um pacote endereçado a um servidor *Anycast*, é recebido pelo membro do grupo *Anycast* "mais próximo", sendo que a proximidade é definida pelas métricas de rede implementadas pelos encaminhadores. Um exemplo em que esta situação pode acontecer é a troca de um ficheiro entre um servidor e um cliente, em que o ficheiro, devido ao seu tamanho, necessita de ser divido em vários pacotes. Sendo assim, tendo em conta a dificuldade em manter um fluxo de informação com um servidor em concreto, em termos de segurança nas comunicações torna-se extremamente difícil o estabelecimento de canais seguros entre os clientes e os servidores *Anycast* utilizando o IPsec.

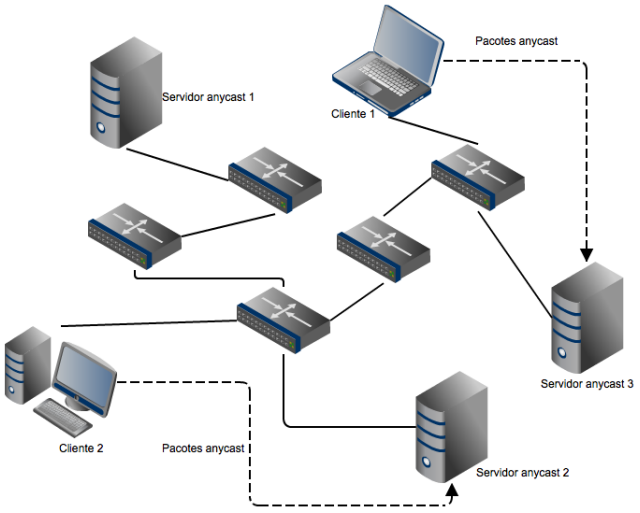

Figura 1.1: Modelo de comunicação *Anycast*

O *anycast* possui algumas limitações na sua utilização. Os endereços *Anycast* apenas podem ser atribuídos a *routers* e não a *hosts*, devido ao facto de não haver nenhum modo standard para *hosts* anunciarem a intenção de receberem pacotes através de endereços *Anycast*. A utilização dos endereços *anycast* como endereço de origem dos pacotes é também ela proibida. Isto deve-se ao facto de um endereço *Anycast* não identificar um único nó.

A utilização de endereços *Anycast* no acesso a servidores replicados acarreta enormes vantagens para estes. Através destes endereços é possível fazer o balanceamento de carga pelos servidores, redirecionar clientes para o servidor mais apropriado e melhorar ainda os mecanismos de descoberta de serviços, tudo isto através de uma solução ao nível de rede, evitando assim períodos de espera por processamentos que levam a más experiências de utilização. Devido às enormes vantagens introduzidas pelo *Anycast*, torna-se essencial conseguir ultrapassar as suas limitações, possibilitando assim a protecção das comunicações que utilizam este tipo de endereço.

O IPsec tem como objectivo proteger uma comunicação entre dois nós sejam eles *routers* ou *hosts*, como tal apenas o fluxo de dados trocados entre os dois nós que acordaram os pormenores de segurança é que será protegido. Por este motivo torna-se pouco apropriado a utilização do *anycast* com o IPsec. O não determinismo na entrega dos pacotes ao servidor correcto cria dificuldades à segurança das comunicações.

Hoje em dia existem diversos problemas de segurança muito frequentes na *Internet* tais como: desvio de sessão (*Session hijacking*); espionagem de tráfego (*Eavesdropping*); mascarar o endereço (*Spoofing*); ou ataques por homem-no-meio (*Man-in-the-midle*). O modo de proteção contra estes problemas consiste na utilização de técnicas criptográficas de chave secreta ou de chave pública para garantir confidencialidade, autenticação e integridade. Uma das aplicações mais populares do IPsec é a construção de *Virtual Private Network* (VPN) [9], que permitem a duas sub-redes criar conexões seguras através da *Internet* pública.

O IPsec consiste num conjunto de protocolos que visam proteger as comunicações das redes IP. O IPsec suporta autenticação ao nível da camada de rede, autenticação da origem dos dados, garante a integridade da informação, confidencialidade e ainda proteção contra ataques por repetição [10]. De modo a obedecer a todos os requisitos de segurança, o estabelecimento de uma conexão utilizando o IPsec, envolve a troca e o armazenamento de uma grande quantidade de informação. Como tal, e no âmbito de um serviço oferecido por um conjunto de servidores,

torna-se insustentável que todos os servidores que prestam o serviço possuam toda a informação de segurança relativas às conexões estabelecidas pelos vários servidores. Como consequência disto, sempre que uma conexão com um servidor falhe, será necessário o estabelecimento de uma nova conexão IPsec, negociando uma vez mais todos os pormenores necessários à conexão.

O objectivos deste trabalho consistem na análise das tecnologias IPSec e *Anycast*, e no desenvolvimento duma solução que permita aliar o balanceamento de carga com a segurança: o balanceamento de carga será obtido utilizando o *Anycast* e a segurança utilizando o IPsec.

Posteriormente, pretende-se testar a solução num cenário prático e concluir sobre os seus resultados. A solução foi implementada utilizando o protocolo de *Internet* IPv6.

1.2 Motivação

Nos últimos anos tem-se observado um rápido crescimento no número de utilizadores e dispositivos que utilizam a *Internet*, como tal tem-se também observado um consequente aumento na procura por conteúdos. Para suportar este consequente aumento na procura, são necessárias soluções eficientes e escaláveis para a distribuição dos conteúdos, permitindo assim que estes estejam sempre disponíveis. A distribuição dos conteúdos evoluiu assim de uma arquitectura que envolvia apenas um servidor, para uma arquitectura com múltiplos servidores distribuídos geograficamente. O que se pretende com esta distribuição geográfica, é que os servidores estejam mais próximos dos clientes melhorando assim a experiencia vivida por eles, reduzindo a carga na rede, e ainda acrescentando uma melhor tolerância a faltas [11].

A replicação dos servidores que oferecem os serviços revelou-se a melhor solução para a distribuição dos conteúdos. Através da replicação os servidores é possível aumentar a disponibilidade dos serviços e distribuir ainda a carga pelos servidores, proporcionando assim uma boa experiência de utilização aos clientes. Visto os servidores estarem geograficamente distribuídos, a localização dos mesmos bem como a escolha do servidor mais apropriado a atender os pedidos dos clientes, revelam-se de extrema importância para o bom funcionamento da replicação de servidores. A utilização de um servidor DNS, de um *Distributed Director*, ou de um HOPS, são algumas das abordagens utilizadas para suportar serviços replicados. Outra alternativa é a utilização de endereços *anycast*. Nesta dissertação propõe-se a utilização de uma solução ao

nível da camada de rede, os endereços *Anycast*, de modo a tornar a escolha do servidor a atender os pedidos dos clientes mais eficiente.

O aumento do número de utilizadores e dispositivos ligados à *Internet* levou ao esgotamento do espaço de endereçamento disponibilizado pelo protocolo de *Internet* IPv4. O esgotamento do espaço de endereçamento bem como a desactualização deste protocolo levou ao desenvolvimento do novo protocolo de *Internet*, o IPv6. A migração do protocolo IPv4 para o IPv6 está neste momento a decorrer, sendo já actualmente utilizado por diversas instituições por todo o mundo. O *Anycast* foi introduzido pelo IPv6, constituindo um novo modo de comunicação adicionado aos já existentes *Unicast* e *Multicast*. Apesar actualmente ainda não ser implementado por todas as instituições, o IPv6 consiste já numa realidade e será o protocolo de *Internet* utilizado num curto espaço de tempo. Como tal, nesta dissertação todo o trabalho efectuado será no âmbito do protocolo de *Internet* IPv6.

A segurança é hoje em dia um factor muito importante a ter conta em termos de redes de computadores. Quer seja pela confidencialidade dos conteúdos, quer seja pela garantia de autenticidade e integridade dos mesmos, a segurança nas redes de computadores tem de ser uma realidade e é necessário serem tomadas medidas no sentido de proteger as comunicações. O IPv6 introduziu o IPsec, que consiste num conjunto de protocolos que visam proteger as comunicações. Sendo o IPsec a melhor solução ao nível de segurança, pretende-se assim aliar a proteção oferecida pelo IPsec com as capacidades de distribuir a carga pelos servidores, isto é na escolha do melhor servidor e na descoberta de serviços do modelo de comunicação *Anycast*.

1.3 Objectivos

Os principais objectivos desta dissertação são:

- Analisar aprofundadamente o estado da arte do modelo de comunicação *Anycast*, do IPSec, e da utilização em simultâneo de ambas as tecnologias.
- Implementar o modelo de comunicações *Anycast* e do IPSec em situações concretas, tanto isoladamente como em simultâneo e concluir acerca dos resultados obtidos.

- Conceber e implementar uma solução que permita aliar o balanceamento de carga com a segurança: o balanceamento de carga será obtido utilizando o *Anycast* e a segurança utilizando o IPsec.

1.4 Estrutura da tese

Neste capítulo é efectuada uma breve introdução ao tema desta dissertacão. É descrita a motivação para a realização deste trabalho, juntamente com os respectivos objectivos a alcançados. Este capítulo termina com a descrição da estrutura adoptada para esta dissertação.

O Capítulo 2 apresenta o estado da arte relativamente ao protocolo de *Internet* IPv6, o estado da arte do IPSec e do modelo de comunicação *Anycast*. O Capítulo 2 apresenta ainda uma breve descrição de soluções de balanceamento de carga. No final deste capítulo é ainda efectuado um resumo do estado da arte das tecnologias abordadas.

No Capítulo 3 é efectuada a descrição de vários trabalhos relacionados com os temas abordados nesta dissertação: (1) o protocolo de segurança IPSec; (2) o modelo de comunicação *Anycast*; (3) e ainda sobre balanceamento de carga.

No Capítulo 4 são implementadas e testadas as tecnologias IPSec e *Anycast* em cenários concretos. Inicialmente é implementado modelo de comunicação *Anycast*, observando vários aspectos deste modelo. De seguida é implementado o IPSec. Depois são implementados o modelo de comunicação *Anycast* em simultâneo com o IPSec. Para terminar este capítulo, conclui-se acerca dos resultados obtidos nos testes efectuados neste capítulo.

O Capítulo 5 apresenta a solução proposta. Neste capítulo a solução é apresentada em detalhe, apresentando as decisões tomadas e os motivos que levaram a estas. São ainda efectuados alguns testes preliminares, concluindo de seguida acerca dos resultados obtidos nestes testes.

Por fim, o Capítulo 6 apresenta as conclusões desta dissertação, resumindo os objectivos cumpridos e as propostas de trabalho futuro.

Capítulo 2

Estado da arte

Esta dissertação incide sobre quatro temas: o novo protocolo de *Internet* IPv6; o balanceamento de carga; *Anycast*, o novo modelo de comunicação introduzido pelo IPv6; e o IPsec, tecnologia que tal como o *Anycast* é introduzido pelo IPv6. Nesta secção da dissertação é efectuado um levantamento do estado da arte destes quatro temas. Isto tem como objectivo demonstrar todos os pormenores relativos às tecnologias utilizadas, justificando assim o porquê das escolhas tomadas.

2.1 *Internet Protocol version* 6

O IPv6 é referido muitas vezes como a ”próxima geração do protocolo de *Internet*”, tendo sido desenvolvido com o intuito de ser o sucessor do protocolo de *Internet* IPv4. Várias alterações foram efectuadas relativamente ao IPv4, com o intuito de melhorar e ultrapassar as limitações deste. Através do IPv6 propriedades como flexibilidade, eficiência, capacidade e funcionalidade optimizada surgem como resposta às exigências crescentes dos utilizadores. Neste capítulo iremos demonstrar as alterações e o impacto destas no desempenho deste novo protocolo.

O número de utilizadores e dispositivos (por exemplo: telemóveis, impressoras, televisões, consolas, etc) que hoje em dia utilizam a *Internet* tem vindo a crescer exponencialmente. Para suportar este crescimento, o IPv4 desenvolveu soluções para contornar o problema da limitação

de recursos ao nível do espaço de endereçamento (por exemplo: *Network Address Translation* - NAT; *Variable-Length Subnet Masking* - VLSM; e *Classless Inter-Domain Routing* - CIDR), no entanto estas soluções apenas adiaram esta questão. Para compensar esta contínua falta de endereços o IPv6 surge assim como a melhor solução. O IPv6 aumentou as capacidades de endereçamento relativamente ao IPv4, de 32 *bits* para 128 *bits*. Com este aumento do espaço de endereçamento é possível suportar mais níveis de hierarquia de endereçamento, até cerca de $3,4 \times 10^{38}$ endereços, e uma arquitectura de endereçamento mais flexível [12]. O encaminhamento torna-se também mais escalável pois os endereços podem ser agregados de forma mais eficiente. Esta nova versão do protocolo de *Internet* inclui uma utilização mais ampla do modo de comunicação *Multicast*, melhorando o desempenho das redes. No IPv6 o modo de comunicações *Broadcast* foi retirado, sendo utilizado agora por sua vez o *Multicast* na maioria das aplicações que utilizavam as comunicações *Broadcast*. Um novo tipo de endereço é também introduzido pelo IPv6. O novo tipo de endereços chama-se *Anycast*, e possibilita o envio de informação para um nó de entre um grupo de nós. A simplificação do formato do cabeçalho dos pacotes foi outra das alterações introduzidas pelo protocolo IPv6. Vários parâmetros do cabeçalho foram removidos e outros alterados, estando alinhados em 64 *bits*, com o intuito de reduzir o tempo de processamento dos pacotes nos nós intermediários, e ainda para limitar a largura de banda ocupada pelo cabeçalho. O IPv6 introduz ainda um mecanismo de segurança ponto-a-ponto que opera ao nível da camada de rede, este mecanismo denomina-se IPsec.

2.1.1 Formato do endereço

Como já foi afirmado, os endereços IPv6 possuem 128 *bits*, valor quatro vezes superior ao tamanho dos endereços IPv4. Na figura 2.1 podemos observar um exemplo de um endereço IPv6.

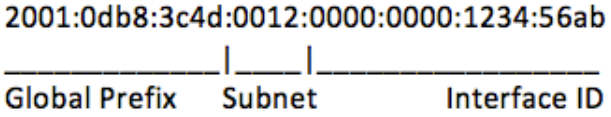

Figura 2.1: Exemplo de um endereço IPv6

Observando o exemplo podemos chegar à conclusão que o endereço apresenta uma representação consideravelmente diferente quando comparado com um endereço IPv4. A primeira diferença é como já foi referido o tamanho, que no IPv6 é consideravelmente maior. Este tipo de endereços é representado por 8 grupos de 16 *bits*, na notação *standard*, enquanto que no IPv4 são quatro grupos de 8 *bits*. Podemos ainda chamar à atenção o facto dos grupos de 16 *bits* serem expressos em hexadecimal.

2.1.2 Cabeçalho do pacote IPv6

O formato do cabeçalho IPv6 sofreu grandes alterações relativamente ao formato utilizado no IPv4. Entre as várias alterações introduzidas pelo novo formato do cabeçalho estão a adopção de um tamanho fixo para o cabeçalho, a remoção de parâmetros, a alteração de outros. Todas estas alterações tiveram como objectivo a optimização do novo protocolo da *Internet*.

O formato do cabeçalho IPv6 pode ser observado na figura 2.2.

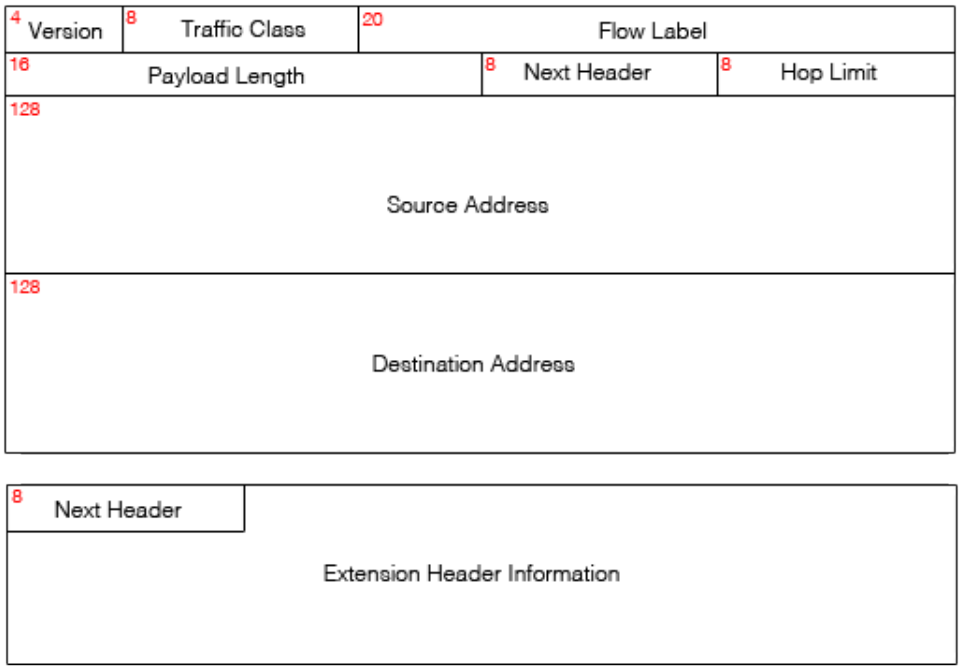

Figura 2.2: Formato do cabeçalho IPv6

- ***Version***: Número de versão do protocolo de *Internet* (4 *bits*);
- ***Traffic Class***: Classe de tráfego (8 *bits*);

- ***Flow Label***: Identificador de fluxo (20 *bits*);
- ***Payload Length***: Tamanho do *payload* do pacote IPv6 (16 *bits*);
- ***Next Header***: Identifica o tipo de cabeçalho imediatamente a seguir ao cabeçalho IPv6 (8 *bits*);
- ***Hop Limit***: Decrementado a cada nó que encaminha o pacote. O pacote é descartado caso o *hop limit* seja igual a zero (8 *bits*);
- ***Source Address***: Endereço do nó que originou o pacote (128 *bits*);
- ***Destination Address***: Endereço do nó para o qual o pacote se destina (128 *bits*);

2.1.3 *Anycast*, um novo modelo de comunicação

O IPv6 introduz um novo tipo de endereço designado por *Anycast*. Tal como o *Multicast*, um endereço *Anycast* identifica um grupo de *interfaces*. O que distingue estes dois modos de comunicação é o facto de no caso do *Anycast* a informação enviada para um endereço anycast ser recebida por um único nó de um conjunto de nós. Ao contrário do multicast que possui uma filosofia de ”um para muitos”(*one to many*), ou seja, um pacote enviado para um endereço *Multicast* é recebido por todos *interfaces* com este endereço, o *Anycast* então possui uma filosofia de ”um para qualquer um”(*one to any*).

O *Anycast* é descrito em maior detalhe na secção 2.4.

2.1.4 Segurança nas comunicações

A segurança na *Internet* é nos dias de hoje um dos temas mais em voga. Isto deve-se ao facto do teor de alguns dos conteúdos transmitidos através da *Internet* serem de extrema importância, necessitando que seja garantida a sua integridade e confidencialidade. Como tal, o novo protocolo de *Internet* não podia deixar este assunto de lado, introduzindo assim o IPSec.

O IPsec consiste num conjunto de protocolos que visam proteger as comunicações das redes IP. O IPsec suporta autenticação ao nível da camada de rede, autenticação da origem dos dados,

garante integridade, confidencialidade e ainda proteção contra ataques por repetição [10]. O IPsec é descrito em maior detalhe no secção 2.3.

2.1.5 Opções no IPv6

O modo como as opções são processadas é mais uma das alterações introduzidas pelo IPv6. Neste protocolo, as opções foram removidas do cabeçalho, facto que se verificava no IPv4. Pretende-se com esta alteração melhorar a eficiência do processamento dos pacotes, diminuindo os tempos de processamento, e ainda melhorar a eficiência do processamento das opções.

Como já foi afirmado, vários campos presentes no cabeçalho IPv4 foram removidas do cabeçalho IPv6. Alguns desses campos passaram a a ser opcionais sob a forma de cabeçalhos de extensão. Os cabeçalhos de extensão são cabeçalhos adicionais que são encapsulados dentro do pacote IPv6. Um pacote IPv6 pode transportar vários cabeçalhos. Cada cabeçalho possui o identificador do tipo de cabeçalho seguinte. Este identificador consiste no campo *next header* do cabeçalho IPv6. Depois dos cabeçalhos de extensão, vem a informação (cabeçalho + *payload*) relativa ao protocolo de nível superior (ICMP, TCP, UDP, etc).

As extensões aos cabeçalhos são opcionais, mas caso elas existam devem sempre seguir uma determinada ordem:

Cabeçalho IPv6

Extensão ao cabeçalho

- 0 - hop-by-hop Option Header
- 43 - Routing Header
- 60 - Destination Options Header
- 44 - Fragmentation Header
- 51 - Authentication Header

- 50 - Encapsulation Security Payload Header
- 135 - Mobility Header
- 59 - No Next Header

Camadas superiores

- 6 - TCP
- 17 - UDP
- 58 - ICMPv6

Os cabeçalhos de extensão *Hop-by-Hop Options* e *Destination Options* contêm opções, e utilizam uma estrutura semelhante para as armazenar.

A estrutura usada para armazenar as opções é a seguinte:

- *Option Type (8 bits)* - Tipo de opção
- *Option Data Length (8 bits)* - Número de octetos no campo *Option Data*
- *Option Data* - Opção de comprimento variável

Os dois *bits* mais significativos do campo *Option Type* indicam a acção a tomar quando um nó não reconhece a opção:

- 00 = Ignorar a opção e continuar
- 01 = Ignorar o pacote
- 10 = Ignorar o pacote e enviar a mensagem ICMP - *Parameter Problem* com Código 2

- 11 = Ignorar o pacote, enviar a mensagem ICMP - *Parameter Problem* com Código 2, apenas se o endereço de destino não for *multicast*

Para indicar se o campo *Option Data* pode sofrer alterações durante o seu percurso é utilizado o terceiro *bit* mais significativo. O valor zero indica que não se produzem alterações e o valor um indica que estas se podem produzir. Esta *flag* assume uma grande importância na presença de um cabeçalho de extenção de autenticação.

2.2 Balanceamento de carga

A utilização de múltiplos servidores para suportar serviços na *Internet*, é uma técnica muito utilizada nos dias de hoje por todo o mundo. Pretende-se com isto, distribuir uniformemente a carga pelos servidores de modo a optimizar a utilização dos recursos, minimizar o tempo de resposta, aumentar o *throughput* e evitar a sua sobrecarga. A utilização de múltiplos servidores possui ainda a vantagem de deixar de existir um ponto de falha único.

Ao balancear a carga pelos vários servidores é possível aumentar os valores da disponibilidade do serviço, pois mesmo que um servidor falhe existem sempre outros a manter o serviço disponível. Isto é conseguido devido à redundância conseguida ao replicar os conteúdos pelos múltiplos servidores.

Existem várias abordagens para balancear a carga pelos servidores, sendo as mais utilizadas as baseadas em:

- Servidores DNS;
- *Hardware*;
- *Software* específico.

A abordagem baseada em servidores DNS é também conhecida por DNS *round robin*. O balanceamento de carga nesta abordagem, é conseguido através da inserção de vários registos (endereços IP dos servidores) para o mesmo *hostname*. Assim, sempre que o servidor DNS recebe um pedido para o *hostname* do serviço, o servidor irá utilizar o algoritmo *round robin*

para escolher o servidor que irá responder ao pedido. Se se pretender que um servidor responda a mais pedidos que os outros, aumenta-se o número de registos com o endereço desse servidor no servidor DNS.

A abordagem *hardware*, pode encaminhar pacotes IP para vários servidores de um *cluster*. Este tipo de abordagem, tipicamente é utilizada para fornecer uma topologia robusta, com alta disponibilidade, mas vem com um custo muito mais elevado.

A maioria das abordagens utilizadas são baseadas em *software*, e muitas vezes vêm como componente integrada de servidores *web* e pacotes de aplicativos de *software* de servidor. São soluções mais baratas do que baseadas em *hardware*. São configuráveis com base nos requisitos, e podem ainda incorporar encaminhamento inteligente baseado em vários parâmetros de entrada.

2.3 Segurança IP

Existem nos dias de hoje diversos problemas de segurança muito frequentes na *Internet*, tais como :

- Desvio de sessão (*Session hijacking*);
- Espionagem de tráfego (*Eavesdropping*);
- Mascarar o endereço (*Spoofing*);
- Ataques por homem-no-meio (*Man-in-the-midle*).

O modo de proteção contra estes problemas consiste na utilização de técnicas criptográficas de chave secreta ou de chave pública para garantir confidencialidade, autenticação e integridade. Da necessidade de segurança nas comunicações surge o IPsec. O IPsec foi introduzido como *standard* no protocolo de *Internet* IPv6, a sua arquitectura foi inicialmente descrita no RFC 2401 [3]. Uma das aplicações mais populares do IPsec é a construção de VPNs, que permitem a duas sub-redes criar conexões seguras através da *Internet* pública.

O IPsec consiste num conjunto de protocolos que visam proteger as comunicações das redes IP. O IPsec suporta autenticação ao nível da camada de rede, autenticação da origem dos dados,

garante a integridade da informação, confidencialidade e ainda proteção contra ataques por repetição. Através do IPsec é possível a troca de informação na *Internet* de um modo seguro. O IPsec possui dois mecanismos de funcionamento: *Authentication Header* (AH) [14]; e *Encapsulating Security Payload* (ESP) [15]. O AH é utilizado pelo IPsec para garantir integridade e autenticação da origem, e o ESP para garantir integridade e confidencialidade. Opcionalmente o ESP pode também garantir autenticação de origem. Tipicamente são utilizados independentemente, no entanto é possível utilizar ambos os protocolos em simultâneo. O IPsec utiliza técnicas de criptografia simétrica como 3DES [16] para fornecer confidencialidade, e algoritmos baseados em sistemas de *hash* tal como HMAC-SHA1 [17] para assegurar a integridade.

É possível configurar o IPSec de duas formas: (1) configurando manualmente as SAs e SPs, introduzindo-as manualmente na *security association database* (SAD) e *security policy database* (SPD); (2) ou de forma automática através do *Internet Key Exchange* (IKE) [18]. A configuração manual requer a configuração das chaves secretas, utilizadas nas SAs, em ambos os nós da conexão IPsec. O IKE é responsável pela negociação de todos os parâmetros de segurança, incluindo as chaves de cifragem e autenticação.

O IPsec permite o tratamento de modo diferenciado do tráfego através da definição de políticas de segurança. As políticas de segurança são divididas em bases de dados de políticas de segurança (SPD), e em bases de dados de associações de segurança (SAD). As SPD definem o tratamento atribuído às diferentes classes de tráfego. As SAD definem o conjunto de parâmetros, algoritmos e conjuntos de chaves para os protocolos IPsec (AH e ESP).

2.3.1 Base de dados de políticas de segurança

A base de dados das políticas de segurança (SPD), possui um conjunto de informação que vai definir o modo como o tráfego, de entrada e saída, dos nós da conexão IPsec é tratado. A SPD possui uma entrada por cada classe de tráfego que se pretende aplicar um determinado tratamento. A comparação do tráfego com as entradas na SPD é efectuado através de selecionadores (*selectors*). A correspondência do tráfego com as entradas pode ser efectuada através de endereços IP, *subnets*, protocolos ou portas.

Todas as implementações do IPsec devem possuir duas políticas de segurança (*Security Policies* - SP) por cada um dos nós da conexão, uma aplicada ao tráfego de entrada e outro ao de

saída. Assim, tipicamente uma conexão IPsec entre dois nós deve envolver quatro SPs. São necessárias duas SP por nó, pois as regras aplicadas ao tráfego de entrada são diferentes das aplicadas ao tráfego de saída. Como já foi afirmado, as SP definem o comportamento a aplicar sobre o tráfego de entrada e saída dos nós IPsec. As regras possíveis de aplicar são:

- Aplicar IPsec aos pacotes;
- Enviar normalmente os pacotes;
- Descartar os pacotes.

Quando o IPsec é aplicado aos pacotes, existe uma referência para uma entrada na SAD que contém os parâmetros de processamento. Estes parâmetros podem ser definidos estaticamente configurados ou automaticamente através do IKE.

2.3.2 Base de dados de associações de segurança

É na base de dados de associações de segurança que as regras de processamento do IPsec são guardadas. As entradas nesta base de dados são chamadas associações de segurança (*security association* - SA), e podem ser configuradas manualmente, ou automaticamente através do IKE. Cada SA é identificada por um índice dos parâmetros de segurança (*Security Parameters Index* - SPI), endereço IP de destino e pelo protocolo de segurança.

Tipicamente uma SA deve incluir a seguinte informação [3]:

- ***Security Parameter Index (SPI)***: Valor de 32 *bits* utilizado para identificar exclusivamente uma SA;
- ***Sequence Number Counter***: Número de sequência do próximo pacote a ser transmitido usando este SA;
- ***Sequence Counter Overflow***: *Flag* que indica se o *overflow* do contador do número de sequência deve gerar um evento e prevenir a transmissão de pacotes adicionais através do SA, ou se *rollover* é permitido;

- ***Anti-Replay Window***: Contador de 64 bits, usado para determinar se um pacote de entrada, AH ou ESP, é uma repetição;
- ***Security protocol***: Protocolo de segurança utilizado com este SA. Actualmente o IPsec suporta dois protocolos: o ESP e o AH;
- ***IPsec protocol mode***: O IPsec possui dois modos de funcionamento: o túnel e o transporte;
- ***Lifetime of this SA***: Intervalo de tempo após o qual uma SA deve ser substituído por uma nova SA;
- ***Encryption Algorithm***: Algoritmo utilizado para cifrar informação. Este valor é apenas utilizado com o protocolo de segurança ESP;
- ***Authentication Algorithm***: Algoritmo utilizado para autenticação. Este valor é utilizado pelo ESP e pelo AH;
- ***Traffic based lifetime***: Quantidade de tráfego que pode ser processada pela SA.

Um pacote pode ter associado a si não apenas um, mas vários SAs. Esta situação, pode ocorrer por exemplo quando uma implementação IPsec utiliza os dois protocolos de segurança, ESP e AH, em simultâneo.

Numa implementação IPsec, sempre que são enviados ou recebidos pacotes através de um dos nós da implementação, o IPsec consulta a SPD de modo a saber o modo como deve tratar o pacote. Dependendo do tipo de tráfego, saída ou entrada, o IPsec age de modo diferente de acordo com a situação.

Quando um nó IPsec envia um pacote, primeiro é consultada a SPD para determinar o procedimento a ter com esse pacote. Caso seja necessário aplicar o IPsec ao pacote, haverá então na SAD um SA ou a coleção de SAs associadas a este pacote. Os SAs associados ao pacote vão determinar o modo como o pacote é processado.

Quando um nó IPsec recebe um pacote com cabeçalho IPsec, é consultada a SAD de modo a identificar os SAs associados ao pacote. A pesquisa na SAD pelos SAs correctos é efectuada através o SPI, do protocolo e do endereço IP de destino. Se for identificado algum SA a proteção IPsec é removida. Após ser removida a proteção, volta-se a verificar o pacote. Se o pacote

ainda se dirigir ao nó IPsec e contiver uma proteção IPsec, é localizada outra SA, e a proteção é removida. Este procedimento é repetido enquanto ainda houver proteções IPsec no pacote, e este for endereçado ao nó IPsec. Após removidas todas as proteções, é consultada a SPD de modo a verificar se as proteções associadas ao pacote são as que se verificaram na recepção. Caso sejam, o pacote é permitido ao pacote passar.

2.3.3 *Internet key exchange*

Com a grande difusão da implementação e utilização do IPsec, relevou-se necessária a criação de um protocolo standard, escalável e automático para a gestão dos SAs. Daqui nasce o IKE, um protocolo de troca de chaves *Diffie-Hellman* [19], utilizado para a criação de IPSec SAs. O IKE opera em duas fases. Numa primeira fase é criado um canal seguro entre os nós IKE. Na segunda fase, é negociado um IPsec SA. Todas as mensagens da segunda fase são protegidas pelo canal seguro criado na primeira fase. Todas as mensagens IKE são transportados utilizando UDP na porta 500 entre os nós IKE. Para que o IKE funcione, a SAD do IPsec deve possuir uma entrada que permite ao tráfego IKE para passar pelo IPsec sem proteção. Se essa entrada não estiver presente o IPSec irá bloquear os pacotes IKE.

Ao utilizar o IKE, a autenticação nas trocas de informação é um factor muito importante a ter em consideração. O IKE suporta autenticação com chaves previamente partilhadas e de chave pública. A autenticação é efectuada através do cálculo de um *hash* sobre partes selecionadas do *payload* dos pacotes enviados na primeira fase, e assinando o *hash* caso fosse utilizada criptografia de chave pública, ou incluindo o segredo previamente partilhado no *hash*. Sem a autenticação, um ataque do tipo homem-no-meio contra o IKE seria possível.

2.3.4 Modos de operação

Os dois protocolos disponibilizados pelo IPsec, AH e ESP, possuem ambos dois modos de funcionamento: o modo de transporte (*Transport mode*), utilizado para fornecer uma conexão segura entre dois nós ao encapsular o *payload* IP; e o modo túnel (*Tunnel mode*), neste modo o pacote IP é todo ele encapsulado de modo a criar um ”salto virtual seguro” entre dois nós. Tipicamente o modo túnel é utilizado na criação de VPNs.

No modo de transporte, o cabeçalho IPsec (AH ou ESP) é inserido depois do cabeçalho IP. Na imagem 2.3 está um exemplo da utilização do modo de transporte.

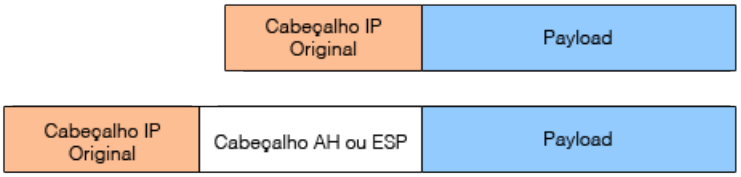

Figura 2.3: Modo de transporte (AH ou ESP)

No modo túnel, o pacote IP é encapsulado na sua totalidade no *payload* do novo pacote IP. Na imagem 2.4 está um exemplo da utilização do modo de túnel.

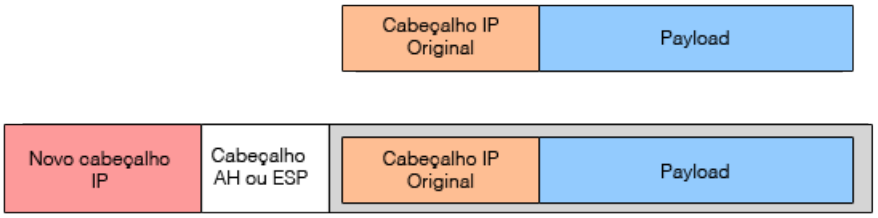

Figura 2.4: Modo de túnel (AH ou ESP)

2.3.5 *Authentication header*

O *authentication header* (AH) é utilizado para autenticação, tendo como objectivo garantir que estamos realmente em comunicação com o interlocutor que pensamos estar, detectar se houveram alterações na informação enquanto esta foi transmitida na rede, ou seja, garantindo a integridade da informação, e ainda para proteger as comunicações de ataques por repetição.

O AH processa uma mensagem de autenticação baseada no *hash* da maioria dos campos presentes na mensagem, juntando-lhe um segredo que depois é retirado. Essa mensagem é posteriormente inserida no campo *authentication data* do pacote a ser enviado. Quando o nó IPsec recebe um pacote protegido por AH, é calculada a mensagem de autenticação do mesmo modo que na origem, de modo a poderem ser comparados os valores recebido com o calculado. Caso sejam iguais, o receptor pode ter a certeza que o pacote não foi modificado porque para recalcular a mensagem de autenticação é necessário possuir a chave secreta [10].

Na figura 2.5 é possível observar o cabeçalho AH.

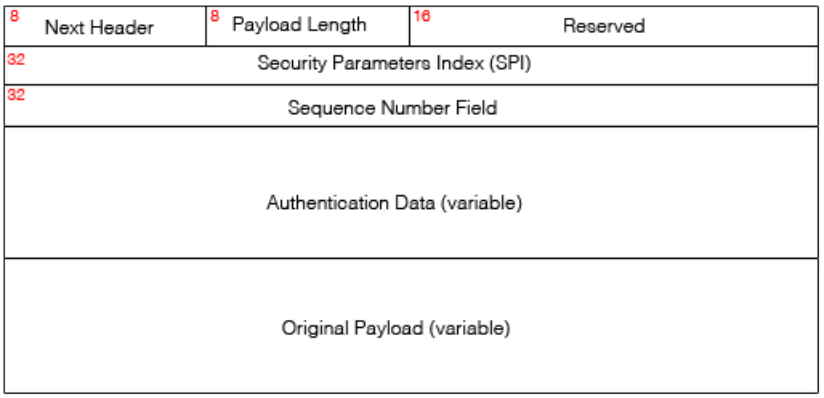

Figura 2.5: Cabeçalho AH

O cabeçalho AH possui os seguintes campos:

- ***Next header***: Identifica o tipo de protocolo utilizado pelo próximo cabeçalho;
- ***Payload length***: Tamanho, em ”palavras” de 32 *bits*, do cabeçalho AH, menos duas ”palavras”;
- ***Reserved***: Este campo está reservado para uso futuro;
- ***Security Parameters Index***: O número SPI da SA utilizada para proteger este pacote. O SPI, o endereço de destino, e o protocolo de segurança utilizados identificam uma SA;
- ***Sequence Number***: Contador de 32 *bits* que é incrementado a cada pacote. Ao utilizar este contador, o receptor pode distinguir se o pacote é ou não um pacote repetido;
- ***Authentication data***: Mensagem de autenticação baseada no *hash* da maioria dos campos presentes na mensagem.

2.3.6 *Encapsulated Security Payload*

Tal como o AH, o ESP garante integridade, autenticação de origem (apenas no modo túnel), proteção contra ataques por repetição e ao contrário do AH pode também garantir a confidencialidade.

O ESP utiliza algoritmos de cifragem simétrica para cifrar o *payload* dos pacotes. A confidencialidade é assim conseguida através da cifragem do pacote IP inteiro, sendo este posteriormente inserido num novo pacote IP. O algoritmo de cifragem utilizado é escolhido no momento da configuração do IPsec, sendo que existem várias opções de escolha (por exemplo: DES, AES, 3DES, etc) [10].

Na Figura 2.6 é possível observar o cabeçalho ESP.

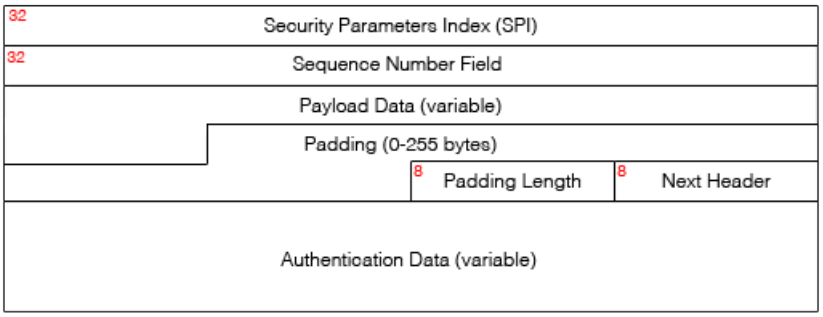

Figura 2.6: Cabeçalho ESP

O ESP pode ser utilizado para fornecer apenas autenticação ou confidencialidade. Se apenas é utilizada a confidencialidade, a protecção contra ataques por repetição não pode ser usada porque o campo do número de sequência não está autenticado.

2.3.7 Protecção contra ataques por repetição

A proteção contra ataques por repetição é uma propriedade partilhada por ambos os protocolos AH e ESP. Esta proteção é conseguida através da utilização de números de sequência e autenticação dos dados dos pacotes. Assim, caso seja utilizado o ESP sem autenticação não é possível fornecer proteção contra ataques por repetição, pois o número de sequência pode ser alterado.

Todos os pacotes IPsec contêm números de sequência, sendo que estes nunca se podem repetir no contexto da SA a que pertence. Quando o contador do número de sequência atinge o seu valor máximo, a respectiva SA não deve voltar a ser utilizada. Sempre que um nó IPsec envia um pacote, são incrementados os números de sequência do seu cabeçalho. Apenas após a actualização dos números de sequência é que é calculado o valor de autenticação. É através da autenticação dos dados que se garante que nenhum dos campos do pacote é alterado.

Quando um nó IPsec recebe um pacote com um cabeçalho IPsec, primeiro compara o valor do número de sequência com a sua janela de recepção. Apenas se o pacote passar pela validação do número de sequência, é que autenticação do pacote é verificada. Se a autenticação for verificada então a janela de recepção é atualizada.

2.4 Comunicações *Anycast*

O *Anycast* é um novo modelo de comunicação nos *standards* do protocolo de *Internet* IPv6, tal como o *Unicast* e o *Multicast*. Este novo modo de comunicação caracteriza-se pelo facto da informação enviada para um endereço ser recebida por um único nó de um conjunto de nós.

O modelo de comunicação *Unicast* permite a um nó de origem enviar um pacote para um nó destino único, ou seja, possui uma filosofia de ”um para um”. Por sua vez, o *Multicast* possui uma filosofia de ”um para muitos”(*one to many*), ou seja, um pacote enviado para um endereço *Multicast* é recebido por todos os *hosts* com este endereço. Comparando com os restantes modos de comunicação, o anycast possui uma filosofia de ”um para qualquer um”(*one to any*) [8], um pacote enviado para um endereço *Anycast* é recebido por um dos nós que possuem esse endereço.

Um endereço *Anycast* pode ser atribuído a um ou a mais terminais de rede, tornando-se deste modo possível a um conjunto de servidores replicados que forneçam o mesmo serviço, possuir o mesmo endereço *Anycast*. Um pacote endereçado a um servidor *Anycast*, é recebido pelo membro do grupo *Anycast* ”mais próximo”, sendo que a proximidade é definida pelas métricas de rede implementadas pelos encaminhadores. No entanto, é também importante realçar que dois pacotes endereçados ao mesmo endereço *Anycast* podem chegar a dois servidores diferentes, tornando este modelo inadequado para cenários onde é necessário estabelecer conexões com os servidores de modo a manter um fluxo de informação. Como tal, torna-se também impossível a fragmentação de pacotes, pois os pacotes fragmentados podem ser entregues a servidores diferentes [13].

O *Anycast* foi introduzido como *standard* no protolo de *Internet* IPv6, no entanto existem vários exemplos de implementação do *Anycast* no IPv4. O RFC1546 define um serviço *Anycast* experimental para o IPv4. Neste RFC, os endereços *Anycast* são diferenciados dos *Unicast* ao serem alocados numa gama de endereços diferentes.

A utilização de um endereço *Unicast* partilhado por vários *hosts*, *shared-unicast address*, é outra das implementações do *Anycast* no IPv4. Esta técnica é utilizada para balancear carga por múltiplos servidores, por exemplo em servidores DNS e em servidores *web*.

2.4.1 Propriedades e limitações do *Anycast*

O facto dos endereços *Anycast* utilizarem o mesmo formato dos endereços *Unicast* torna estes endereços, sem mais nenhuma informação adicional, indistinguíveis.

A atribuição de endereços *Anycast* foi limitada a nós que sejam *routers* [12]. Esta propriedade deve-se ao facto de não haver nenhum modo normalizado para *hosts* anunciarem a intenção de receberem pacotes através de endereços *Anycast*. O facto de todos os nós *Anycast* serem *routers* torna o encaminhamento de pacotes *Anycast* mais simples.

Como os endereços *Anycast* não identificam um nó de origem único, a sua utilização como endereço de origem num pacote IP foi desaconselhada.

Dois pacotes enviados pelo mesmo *host* para um endereço *Anycast*, podem chegar a dois servidores *Anycast* diferentes, dependendo da estabilidade da tabela de *routing*. Esta propriedade do *Anycast* inviabiliza também a fragmentação de pacotes que possuam um endereço *Anycast* como endereço de destino[13]. A utilização do ligações TCP torna-se também difícil devido a esta propriedade do *Anycast*.

Uma das principais limitações deste modelo de comunicação é o de não existirem protocolos de encaminhamento *Anycast* devidamente reconhecidos e implementados. O tráfego *Anycast* é tratado pelos encaminhadores como tráfego *Unicast*, não aproveitando todas as vantagens deste modelo.

2.4.2 Conjugação do *Anycast* com o IPSec

O IPsec identifica os seus nós através de pares de endereços origem/destino. Tendo em conta os problemas, previamente evidenciados, associados à utilização dos endereços *Anycast* tanto no campo de origem como no de destino, torna-se muito difícil a utilização do IPsec em pacotes *Anycast*. Ao utilizarmos a configuração manual o modelo de confiança utilizado pelo IPsec em

endereços *Anycast* torna-se confuso. Visto o IPsec utilizar o endereço de destino para identificar a chave secreta que deverá ser utilizada, todos os nós com o mesmo endereço *Anycast* deverão possuir a mesma chave secreta. Como foi afirmado anteriormente, o IPsec possui mecanismos de proteção contra ataques por repetição. Devido ao não determinismo na entrega de pacotes *Anycast* dois pacotes podem chegar a dois servidores diferentes, logo o contador de repetições não será actualizado de modo consistente provocando o mau funcionamento do mecanismo de proteção contra ataques por repetição. Devido ao não determinismo na entrega de pacotes *Anycast* torna-se também impossível a utilização do IKE para a troca de chaves secretas. Tal como no caso da configuração manual, todos os nós com o mesmo endereço *Anycast* devem possuir também a mesma chave secreta.

2.5 Resumo

Na secção do estado da arte, é efectuada uma revisão técnica sobre o estado da arte das tecnologias IPv6, balanceamento de carga, *Anycast* e IPsec. Estes quatro temas foram abordados devido ao facto desta dissertação incidir essencialmente sobre eles.

A solução desenvolvida e apresentada nesta tese, foi desenvolvida no âmbito do novo protocolo de *Internet*, o IPv6. Isto deve-se ao facto, de tal como é explicado no capítulo de estado da arte do IPv6, o IPv6 ser num futuro próximo a realidade da *Internet*. As melhorias apresentadas por este protocolo são bastante significativas para o desempenho da *Internet*, e visto este protocolo estar inclusive a ser utilizado por várias aplicações por todo o mundo revela-se como a escolha mais apropriada para o ambiente em que a solução apresentada deve funcionar.

Devido ao enorme número de serviços disponibilizados na *Internet* que são suportados por vários servidores replicados, o balanceamento de carga revela-se um dos temas em voga nos dias de hoje. Nesta dissertação é apresentado o *Anycast* como um modo de efectuar o balanceamento de carga.

O *Anycast* é apresentado como um modo eficiente de balancear a carga pelos vários servidores, ao encaminhar os clientes para os servidores ”mais próximos” deles. A vantagem desta solução é tratar-se de uma solução ao nível da rede, evitando longos tempos de processamento

das soluções a nível aplicacional. Esta solução possui ainda a vantagem do *Anycast* ser uma tecnologia nativa no IPv6, algo que não se verificava no IPv4. No entanto, o anycast possui algumas limitações/características que inviabilizam a sua utilização com o TCP e consequentemente com o IPsec.

O IPsec é outro dos temas abordados nesta dissertação. O IPsec surge com o IPv6, introduzindo enormes melhorias em termos de segurança, tendo sido por isso a solução escolhida para prover a segurança na solução proposta.

Capítulo 3

Trabalhos relacionados

Ao efectuar o estudo acerca do estado da arte das tecnologias envolvidas nesta dissertação, foram também estudados trabalhos relacionados com as tecnologias abordadas. Não foram encontrados trabalhos que procurassem responder ao problema colocado nesta dissertação, no entanto foram encontrados outros que, devido ao seu conteúdo, estão relacionados com o problema colocado.

Este capítulo está dividida em três secções. Na secção *IP security*, são apresentados trabalhos na área do IPsec. Na secção *Anycast*, são apresentados trabalhos que procuram responder às limitações deste protocolo. Na última secção são apresentados trabalhos relacionados com o tema do balanceamento de carga.

Neste capítulo não são apresentados trabalhos que associem as comunicações *Anycast* com o IPSec, devido ao facto de não ter sido encontrado nenhum trabalho que se inserisse neste contexto.

3.1 Segurança IP

Nesta secção são apresentados dois trabalhos que abordam a temática da utilização do IPsec em *clusters* de servidores. Os trabalhos utilizam abordagens diferentes, apresentando soluções distintas para resolver o mesmo problema.

3.1.1 Uma solução IPSec para *clusters*

Em *Ipsec clustering* [20] é apresentada uma implementação de um *cluster* IPsec, que permite a distribuição da implementação do IPsec em várias máquinas dum modo em que a segurança do IPsec é mantida. Este mecanismo torna possível a alta disponibilidade e tolerância a falhas a um nível que seria muito difícil de obter com a implementação de um único nó. Se um membro do *cluster* falhar, os outros membros podem começar a lidar com tarefas que foram tratadas pelo membro que falhou.

Antti Nuopponen et al. propõem uma implementação de um *cluster* IPsec que funciona como um nó IPsec único, do ponto de vista do cliente. Supõe-se que os membros do *cluster* estão ligados utilizando uma conexão *Ethernet*, que é fisicamente protegida e não pode ser adulterada. Supõe-se também que os nós são capazes de se comunicar usando *frames Ethernet*.

O modelo de *cluster* escolhido neste trabalho, é um *cluster* com uma arquitectura *Master*/*Slave* que utiliza uma abordagem *shared ip address*, utilizando o servidor *master* para encaminhar pacotes para outros membros do *cluster*.

Como o IPsec opera na camada IP, o cluster precisa de ter um endereço IP único. Caso contrário, o *cluster* não seria um nó IPsec único.

Este modelo foi escolhido porque torna mais fácil preservar a função de proteção contra ataques por repetição do IPsec em situações de falha. Num sistema baseado numa arquitectura *master*/*slave* o *master* pode acompanhar os números de sequência, pois é ele que encaminha pacotes.

Neste modelo todos os nós do *cluster* possuem uma SAD com os mesmos SAs. Sempre que um nó do *cluster* estabelece uma nova SA, é efectuado o *broadcast* da SA para todos os membros do cluster. O *broadcast* é efectuado através da rede física que liga os membros do *cluster*. Os valores dos números de sequência dos SAs, são também disponibilizados a todos os membros do *cluster*, possibilitando assim que qualquer um dos membros do *cluster* possa atender um pedido de um cliente que estabeleceu inicialmente uma SA com outro membro do *cluster*.

Como o *cluster* representa um nó único do ponto de vista do cliente, e este é representado por um endereço IP *Unicast*, o estabelecimento da conexão IPsec é efectuado de um modo normal. Como todos os membros do *cluster* possuem o mesmo endereço IP *Unicast*, e todos eles possuem

a mesma SAD e a mesma informação acerca dos números de sequência dos SAs, qualquer um dos membros do *cluster* pode responder ao pedido do cliente.

Este trabalho foi desenvolvido no âmbito de uma tese de mestrado na *Helsinki University of Technology*.

3.1.2 *Gateway* IPSec para uma arquitectura de *clusters*

Os *gateways* IPSec, tornaram-se actualmente numa das soluções mais utilizadas para fornecer serviços de segurança a clientes dentro de uma sub-rede protegida. A velocidade de processamento de um *gateway* IPSec é fundamental para *throughput* da rede, sendo por conseguinte tipicamente utilizados *clusters* para aumentar essa mesma capacidade de processamento.

Em *Clustered Architecture for High-Speed IPsec Gateway* [21], é proposta uma arquitectura de *cluster* descentralizada com um esquema de balanceamento de carga baseado nos pacotes, desenhado de modo a aumentar o desempenho de um *gateway* IPsec. Cada nó do *cluster* é capaz de processar todos os pacotes de entrada, seja para tal necessário encaminhar, encapsula-los à entrada ou extraí-los à saída do túnel IPSec.

Nesta arquitectura os pacotes são entregues aos nós do *cluster* através do protocolo de comunicação *multicast*. Através da arquitectura proposta, todos os nós do *cluster* recebem todos os pacotes endereçados ao *cluster*. Como tal para que qualquer um dos nós do *cluster* possa responder a cada um dos pacotes de entrada, é necessário que todos os nós possuam a mesma informação de segurança, ou seja, os mesmos SAs. O *cluster* tem como objectivo ser visto como uma única entidade, como tal e visto todos os nós do *cluster* receberem todos os pacotes, é necessário que apenas um dos nós do *cluster* responda a cada um dos pacotes de entrada. De modo a decidir qual dos nós responde a um determinado pacote de entrada, todos os nós possuem um *driver* que à chegada de um pacote actualiza a respectiva SA e decide se este nó responde ou descarta este pacote. Este *driver* garante que apenas um dos nós do *cluster* responde a cada um dos pacotes.

Este trabalho foi desenvolvido na *National Chiao Tung University*, e submetido no *Workshop on Cryptology and Information Security*.

3.2 Modelo de comunicação *Anycast*

Existem vários motivos para que a utilização do anycast seja ainda efectuada de um modo muito limitado, sendo alguns dos principais motivos: não existirem protocolos de encaminhamento *Anycast* devidamente reconhecidos e implementados; o facto de ser desaconselhada a utilização do *Anycast* com o protocolo TCP; e ainda não existirem também protocolos de gestão de grupos anycast devidamente reconhecidos e normalizados. Como tal, os trabalhos acerca do anycast têm incidido essencialmente sobre estes assuntos.

Nesta secção são apresentados alguns trabalhos que visam ultrapassar algumas das limitações do *Anycast*.

3.2.1 Anytun - *secure anycast tunneling protocol*

O `Anytun` [22] consiste numa implementação do *secure anycast tunneling protocol* (SATP). O SATP é definido em [23], e define um protocolo utilizado nas comunicação entre qualquer combinação de extremidades *Unicast* e *Anycast* do túnel. Ele permite todos os protocolos de túneis. O SATP inclui criptografia e autenticação de mensagens baseados nos métodos utilizados pelo *Secure Real-time Transport Protocol* (SRTP). O *anycast* é suportado tanto nos emissores como nos receptores.

Através do `anytun` é possível construir *clusters* VPN com balanceamento de carga entre os servidores. Os servidores VPN possuem o mesmo endereço IP, e é o protocolo de *routing* que é responsável por adicionar e remover servidores VPN. Isto leva a que não sejam necessárias nenhumas alterações nos clientes quando novos servidores VPN são adicionados ou removidos.

A implementação do balanceamento de carga a nível global, é possível utilizando rotas BGP [24] anunciando o espaço de endereçamento dos servidores de túneis em múltiplas localizações.

Actualmente o `anytun` é apenas suportado em Linux.

3.2.2 Gestão segura de grupos *Anycast*

O desenvolvimento de protocolos de gestão de grupos *Anycast* é um dos assuntos abordados pela comunidade científica, no que diz respeito ao *Anycast*. A inexistência de nenhum protocolo devidamente reconhecido e normalizado tem sido um dos principais motivos apresentados para a não utilização das comunicações *Anycast*. De modo a apoiar a utilização de *hosts* como membros do grupo *Anycast*, é necessário um mecanismo para informar o sistema de *routing* acerca da identificação dos hosts *Anycast*. Isto é realizado através de protocolos de gestão de grupos, que correspondem essencialmente a alterações de protocolos de gestão de grupos *Multicast* [25] [26].

Em [27] é proposto o Gothic. O Gothic consiste numa arquitetura de controlo de acesso seguro ao grupo para *Multicast* e *Anycast*. Esta arquitectura propõe-se utilizar as assinaturas com base em certificados PKI para os membros *Anycast* aderirem ao sistema de *routing*.

Em [28] é proposto o *Group Cryptographically Generated Addresses* (G-CGA). Neste método, assume-se que o principal motivo para a gestão dos membros do grupos *Multicast* e *Anycast* poderem ser abusados é que os *routers* não podem determinar se um determinado *host* está autorizado a participar num grupo. Em [28] é proposta uma solução para IPv6 baseada G-CGA. Através deste método, é possível limitar um determinado tipo de ataques DoS. Este método é totalmente distribuído, não necessitando duma entidade confiável ou que seja pré-estabelecida uma associação de segurança entre os *routers* e os *hosts*. O problema deste método é que pode introduzir sérios problemas de segurança se as chaves de grupo forem perdidas. Como tal, é necessária a actualização periódica das chaves.

Em [7] é proposta uma arquitectura para a gestão dos membros do grupo *Anycast*. Segundo [7], para garantir a segurança *Anycast*, os seguintes requisitos devem ser garantidos:

1 - Os grupos *Anycast* que não estejam autenticados não podem ser adicionados ao sistema de *routing*.

2 - Os servidores *Anycast* que não estejam autenticados não podem ser adicionados ao grupo.

3 - A transmissão de informação deve ser efectuada de um modo seguro durante a fase de autenticação.

Para os requisitos 1 e 2, são utilizados certificados PKI como o método de identificação dos servidores *Anycast* e como método de autenticação dos membros *Anycast*. No entanto, é utilizado um *Certificate Authentication Server* (CAS) para que as tarefas de certificar e autenticar serem separadas dos *routers*. O terceiro requisito é cumprido utilizando o IPSec para garantir a segurança da camada de rede e o Secure Socket Layer (SSL) [29] nas camadas de aplicação e transmissão.

3.2.3 Protocolo de resolução de endereços *Anycast*

O *Anycast Address Resolving Protocol* (AARP) [30] [31] foi desenvolvido por Satoshi Doi et al, e tem como objectivo tornar possível a utilização dos protocolos TCP e UDP com serviços que utilizem endereços *Anycast*.

O AARP consiste numa API podendo apenas ser invocada pela camada de aplicação. Esta API funciona de um modo semelhante a uma *Dynamic Linkable Library* (DLL), substituindo a API original fornecida pelo sistema operativo.

O estabelecimento de conexões TCP e UDP utilizando do AARP, é conseguido através da conversão dum endereço *anycast* num endereço *Unicast* correspondente. Quando uma aplicação tenta criar uma conexão, TCP ou UDP, com um endereço *Anycast*, o AARP é invocado descobrindo o endereço *Unicast* de uma das máquinas com o endereço *Anycast*, e estabelecendo uma conexão com essa máquina através do seu endereço *Unicast*. Este procedimento é efectuado sem que a aplicação tenha conhecimento disso.

A descoberta do endereço *Unicast* é efectuada através do protocolo ICMPv6. Quando o AARP é invocado, este envia um pacote ICMPv6 *ECHO REQUEST* para o endereço *anycast*, ao qual este vai responder com um ICMPv6 *ECHO REPLY* com o seu endereço *Unicast* no campo do endereço de origem. O AARP ao receber o ICMPv6 *ECHO REPLY* descobre o endereço *Unicast* e cria a conexão com esse endereço. Isto é possível devido ao facto dos endereços *Anycast* não poderem ser utilizados como endereços de origem.

O AARP possui ainda uma tabela com os endereços *anycast* e os endereços *Unicast* descobertos. Uma entrada na tabela é válida por um tempo predeterminado, e sempre que esse tempo expira o processo de descoberta do endereço *Unicast* a partir do *Anycast* é efectuado de novo.

3.2.4 *Protocol Independent Anycast - Sparse Mode*

Os endereços *Anycast* são actualmente pouco utilizados, sendo uma das razões para tal, a inexistência de um protocolo de *routing Anycast*.

O *Protocol Independent Anycast - Sparse Mode* (PIA-SM) [32] foi desenvolvido por Satoshi Matsunaga et al, e consiste num protocolo de *routing Anycast*.

O PIA-SM, foi desenvolvido a partir do *Protocol Independent Multicast - Sparse Mode* (PIM-SM) que consiste num protocolo de *routing Multicast*. Um dos motivos que levou a esta decisão, baseia-se no pressuposto que os protocolos *Anycast* e *Multicast* possuem muitas semelhanças.

Este protocolo de *routing Anycast*, combina o mecanismo do *routing Multicast* para fazer a gestão de múltiplos receptores, e o mecanismo de encaminhamento de pacotes do *routing Unicast*.

As decisões acerca de qual dos receptores irá receber uma mensagem *Anycast* são, no PIA-SM, baseadas em métricas. Deste modo é assim possível encaminhar um pacote para o receptor mais apropriado. Tal como no PIM-SM, o PIA-SM faz a gestão dos receptores *Anycast* compondo uma árvore de distribuição para cada endereço *Anycast*, com a raiz no *Rendezvous Point* (RP).

Este protocolo de *routing Anycast* foi inicialmente proposto em [31], juntamente com mais dois protocolos. Posteriormente, o PIA-SM foi implementado em [32], no sistema BSD alterando a implementação do PIM-SM. Este protocolo foi implementado modificando um protocolo de *routing* já existente, de modo a facilitar a sua implementação e utilização generalizada.

3.2.5 Protocolos de *routing Anycast*

Em *Design, Implementation and Evaluation of Routing Protocols for IPv6 Anycast Communications* [33] são propostos dois protocolos de *routing Anycast*, o ARIP e o AOSPF. É também proposto um protocolo de gestão dos grupos *Anycast*, o Anycast Receiver Discovery (ARD).

Na especificação dos protocolos ARIP e AOSPF é utilizado o mesmo espaço de endereçamento dos endereços *Unicast* para os endereços *Anycast*, e propõe uma arquitectura de rede que

possibilita uma implementação gradual dos protocolos, ou seja, não necessitam que todos os routers ”falem” estes protocolos para que funcionem.

Estes protocolos foram implementados modificando protocolos de routing já existentes, de modo a facilitar a sua implementação e utilização generalizada.

Os protocolos ARD, ARIP e AOSPF foram implementados através do Zebra, alterando as implementações do Multicast Listener Discovery (MLD), RIPng e OSPFv3.

3.3 Balanceamento de carga usando comunicações *Anycast*

O balanceamento de carga é apontado como uma das principais vantagens da utilização das comunicações *Anycast* para o suporte de serviços replicados. Nesta secção são apresentados dois trabalhos que incidem sobre este tema.

3.3.1 Utilização de endereços *Anycast* para distribuição de carga e localização de servidores

Em *Using IP Anycast for load distribution and server location* [4] é proposta uma abordagem para suportar serviços replicados através do uso de endereços anycast. Um endereço anycast é um endereço IP que pode ser atribuído a um ou mais *hosts*. Diferentes servidores que suportem o mesmo serviço podem ter todos o mesmo endereço anycast. Um *host* é considerado membro de um grupo de *Anycast* simplesmente pelo facto do endereço *Anycast* estar atribuído a um dos seus *interfaces*. Um pacote endereçado a um endereço *Anycast* é entregue a um dos membros do grupo *Anycast*. Esta abordagem é elegante, pois o encaminhamento de pacotes para os endereços *Anycast* é tratado diretamente pelos *routers* que têm acesso a informações precisas da topologia da rede.

Tipicamente, o serviço IP é limitado apenas à entrega de pacotes a *hosts* sem considerar como as aplicações os usam. Neste contexto, o serviço *Anycast* não é excepção. Porém a maioria, se não mesmo todas as aplicações, têm alguma noção de estado que liga os sucessivos pacotes. Claramente, se como resultado da utilização de comunicações *Anycast*, os sucessivos pacotes são encaminhados para servidores *Anycast* diferentes, as informações de estado serão perdidas,

tornando as comunicações *Anycast* apenas úteis para algumas aplicações. De modo a garantir que os pacotes *Anycast* pertencentes a um único fluxo não são encaminhadas para diferentes *hosts*, é necessário efectuar algumas modificações na *stack* TCP / IP do *host*. A ideia deste trabalho é a de fixar o fluxo de informação ao *host* que respondeu ao primeiro pacote. Isto é muito semelhante ao *route pinning* no contexto de QoS *routing*. A associação é efectuada através da inserção de uma *loose source route option* em todos os pacotes do mesmo fluxo. Neste trabalho são implementadas estas modificações na *stack* TCP / IP do AIX 4.2 que consiste numa variante da implementação BSD 4.x.

3.3.2 Utilização de comunicações *Anycast* em DNS

O modelo de comunicação *Anycast* é actualmente amplamente utilizado em serviços DNS. Os endereços IP de muitos DNS *nameservers* de nível superior correspondem a grupos *Anycast*. Os pedidos de clientes enviados para estes endereços são entregues, pela infraestrutura de *Internet routing*, à réplica mais próxima do grupo *Anycast* correspondente. Operadores DNS utilizam o modelo de comunicação *Anycast* por várias razões: reduzir a latência dos pedidos, aumentar a confiabilidade e disponibilidade, bem como a resiliência a ataques DDoS. Embora seja geralmente aceite que a utilização d *Anycast* no DNS foi um passo positivo, não têm sido efectuados estudos que avaliem a melhoria de desempenho oferecido pelo *Anycast*. O trabalho *On the Use of Anycast in DNS* [34] apresenta o primeiro estudo abrangente sobre a matéria.

Este trabalho pretende responder às seguintes perguntas: (1) O anycast reduz a latência dos pedidos? (2) A seleção anycast realmente encaminha os clientes para o servidor DNS mais próximo? (3) As zonas que utilizam o modelo *Anycast* possuem menor número de interrupções? Para responder a estas questões, é realizado um estudo de medição usando clientes implementados através do PlanetLab, para medir as características de desempenho de quatro zonas *top-level* utilizando *Anycast* e comparando estas com zonas que não usam *Anycast*.

Os resultados obtidos podem ser resumidos da seguinte forma: em geral, a utilização do *Anycast* diminui a latência média dos pedidos. Enquanto o número de falhas dos pedidos é relativamente pequena (<= 0,9 %), possuem um período de duração longo (50% duram mais de 100 segundos), afectados por longos períodos de convergência de *routing* BGP. O tipo de esquema de *Anycast* utilizado, independentemente dos servidores possuírem visibilidade local ou global,

determina a percentagem de pedidos dirigidas para o servidor *Anycast* mais próximo. Esse valor varia de cerca de 37% para zonas com poucos nós globais para cerca de 80% para zonas, onde todos os nós são globais. Foi também descoberto um *trade-off* entre o aumento da eficácia do *Anycast* no direcionamento dos pedidos para o servidor mais próximo e estabilidade da própria zona. Para as zonas que anunciam todos os seus membros do grupo *Anycast* globalmente, os clientes escolhem na maioria das vezes o servidor mais próximo. O efeito negativo é que a zona se torna vulnerável ao aumento do número de trica de servidores e interrupções.

Capítulo 4

Implementação e análise das tecnologias

As comunicações *Anycast* e o IPsec, são duas tecnologias que assumem uma enorme importância nesta dissertação. Como tal, nesta dissertação é efectuada uma descrição do estado da arte destas tecnologias, baseada na documentação mais relevante existente sobre estes dois temas.

Após o levantamento do estado da arte das tecnologias, estas foram implementadas e testadas em situações concretas, utilizando para tal as ferramentas adequadas. Inicialmente efectuaram-se testes com as tecnologias IPSec e *Anycast* em separado, e depois com as duas em simultâneo.

Nesta secção apresentam-se os testes realizados e os resultados obtidos nos mesmos, concluindo de seguida sobre esses mesmos resultados.

4.1 Modelo de comunicação *Anycast*

O modelo de comunicação *Anycast*, é uma tecnologia relativamente recente e apenas nativa no IPv6. Como tal, nesta secção são apresentadas as plataformas que actualmente suportam o *Anycast*, bem como o comportamento deste modelo quando deparado com determinadas situações.

A implementação do *Anycast* e os testes realizados foram todos desenvolvidos num ambiente IPv6.

4.1.1 Selecção do endereço *Anycast*

Na implementação do *Anycast*, o primeiro aspecto que foi tomado em consideração, foi a escolha do endereço.

Segundo o `rfc` 2526 [35], a gama de endereços *Anycast* possíveis de ser atribuídos quando estes são do formato EUI-64, que são os disponibilizados na rede utilizada para os testes, são os últimos 128 endereços (últimos 7 *bits*), com todos os restantes *bits* (57 *bits*) a um, menos o `bit universal/local` que tem de ser colocado a zero para indicar que o identificador do *interface* não é único globalmente. Como tal, e tendo em conta que a rede que utilizada para os testes possui o prefixo 2001:690:2280:20::/64, que se enquadra na gama de endereços IPv6 atribuídos à Universidade do Minho, o endereço escolhido foi:

2001:690:2280:20:fdff:ffff:ffff:ff81 ,

sendo os primeiros 64 *bits* o prefixo da rede, e os últimos 64 o identificador de *interface*.

4.1.2 Análise de suporte em várias plataformas

Tendo em conta o panorama actual do IPv6 em termos de implementação a nível mundial, considerou-se necessário verificar quais os sistemas operativos que actualmente suportam o *Anycast*.

Plataforma	Suporte *Anycast*
Mac OS X 10.6.7	Sim
Ubuntu 10.10	Não
Windows	Sim

Tabela 4.1: Plataformas que suportam IPv6 Anycast

Como é possível observar pela tabela 4.1, dos SO observados apenas o sistema operativo Ubuntu não suporta a atribuição de endereços *Anycast*. No caso dos sistema Mac OS X e Windows a atribuição de endereços *Anycast* consiste numa opção disponibilizada nativamente. Foram testados estes três SOs, pois são representativos das famílias BSD [36] , Linux e Windows respectivamente segundo a ordem apresentada na tabela.

No sistema operativo Mac OS X, a atribuição de um endereço *Anycast* é efectuado com o comando representado no Código 4.1.

```
1
2 ifconfig interface inet6 address/prefix anycast
```

Código 4.1: Comando da *shell* para atribuir endereço *Anycast* a um *interface*, no *kernel* BSD

No código 4.2 é possível observar um exemplo.

```
1
2 ifconfig en0 inet6 2001:690:2280:20:fdff:ffff:ffff:ff81/128 anycast
```

Código 4.2: Exemplo de atribuição de um endereço *Anycast* a um *interface*, no *kernel* BSD

No sistema operativo Windows é possível configurar o *Anycast* através dos comandos `Netsh`. O `Netsh` é uma ferramenta da linha de comandos para configurar *interfaces*, endereços, *caches*, e rotas em IPv6. Para a atribuição de um endereço *Anycast* primeiro é preciso entrar no contexto `netsh interface ipv6`, e depois executar o comando representado no Código 4.3.

```
1
2 add address interface={Nome ou indice do interface} address={IPv6 address} type={unicast/
      anycast}
```

Código 4.3: Comando da `shell` para atribuir endereço *Anycast* a um *interface*, no SO Windows Vista

No código 4.4 é possível observar um exemplo.

```
1
2 add address interface=10 address=2001:690:2280:20:fdff:ffff:ffff:ff81/128 type=anycast
```

Código 4.4: Exemplo de atribuição de um endereço *Anycast* a um *interface*, no Windows Vista

4.1.3 Testes efectuados

Nesta secção são apresentados os testes efectuados ao *Anycast* e os respectivos resultados. Os testes foram realizados em máquinas com o SO Mac OS X 10.6.7 e Ubuntu 10.10. No entanto os endereços *Anycast* apenas foram atribuídos às máquinas com o SO Mac OS X 10.6.7, devido aos motivos apresentados na secção anterior.

Os testes efectuados à implementação *Anycast*, visam aferir o comportamento do *Anycast* nos seguintes aspectos do *Anycast*:

- Utilização do endereço *Anycast* como endereço de origem;
- Recuperação em caso de falha;
- Balanceamento de carga;
- Estabelecimento de conexões TCP.

4.1.3.1 Utilização do endereço *Anycast* como endereço de origem

Segundo a especificação do modelo de comunicação *Anycast*, estes endereços não devem ser utilizados como endereços de origem nos pacotes. Para verificar se tal se verificava na realidade, efectuou-se o seguinte teste.

Este teste consiste no envio de `ICMPv6 ECHO REQUESTS` para o endereço *Anycast*, e observar qual o endereço de origem utilizado nas respostas.

```
jveiga@kepler:~$ ping6 2001:690:2280:20:fdff:ffff:ffff:ff81
PING 2001:690:2280:20:fdff:ffff:ffff:ff81(2001:690:2280:20:fdff:ffff:ffff:ff81) 56 data bytes
64 bytes from 2001:690:2280:20:21c:b3ff:feb0:bcca: icmp_seq=1 ttl=64 time=335 ms
64 bytes from 2001:690:2280:20:21c:b3ff:feb0:bcca: icmp_seq=2 ttl=64 time=2.03 ms
64 bytes from 2001:690:2280:20:21c:b3ff:feb0:bcca: icmp_seq=3 ttl=64 time=0.610 ms
64 bytes from 2001:690:2280:20:21c:b3ff:feb0:bcca: icmp_seq=4 ttl=64 time=0.967 ms
64 bytes from 2001:690:2280:20:21c:b3ff:feb0:bcca: icmp_seq=5 ttl=64 time=0.677 ms
64 bytes from 2001:690:2280:20:21c:b3ff:feb0:bcca: icmp_seq=6 ttl=64 time=32.2 ms
64 bytes from 2001:690:2280:20:21c:b3ff:feb0:bcca: icmp_seq=7 ttl=64 time=45.5 ms
64 bytes from 2001:690:2280:20:21c:b3ff:feb0:bcca: icmp_seq=8 ttl=64 time=0.606 ms
64 bytes from 2001:690:2280:20:21c:b3ff:feb0:bcca: icmp_seq=9 ttl=64 time=23.5 ms
64 bytes from 2001:690:2280:20:21c:b3ff:feb0:bcca: icmp_seq=10 ttl=64 time=5.33 ms
^C
--- 2001:690:2280:20:fdff:ffff:ffff:ff81 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9006ms
rtt min/avg/max/mdev = 0.606/44.728/335.697/98.178 ms
```

Figura 4.1: Envio de `ICMPv6 ECHO REQUESTS`

Este teste foi efectuado através do comando `Ping6` da linha de comandos. Como é possível observar pela Figura 4.1, os `ICMPv6 ECHO REQUESTS` são enviados para o endereço *Anycast*, mas os `ICMPv6 ECHO REPLYs` são enviados com o endereço de origem o endereço *Unicast* da máquina.

4.1.3.2 Tempo de recuperação em caso de falha e balanceamento de carga

A redundância fornecida pela utilização dos endereços *Anycast* e o balanceamento de carga, são apresentados como duas das vantagens introduzidas nativamente pelo *Anycast*. Como tal, considerou-se necessário observar estes dois aspectos em situações reais, e concluir sobre o seu comportamento.

De modo a avaliar estes dois aspectos elaborou-se uma arquitectura de rede capaz de os colocar à prova. Para que esta arquitectura seja utilizada para testar tanto o *Anycast* em termos de balanceamento de carga como em termos de redireccionamento em caso de falha, a arquitectura de rede foi manipulada de teste para teste consoante o protocolo de *routing* utilizado.

Devido à inexistência de um protocolo de *routing Anycast*, devidamente reconhecido e implementado, os pacotes *Anycast* são tratados pelos protocolos de *routing Unicast*. Como tal, a arquitectura foi implementada em configurações de encaminhamento externo com o protocolo *Border Gateway Protocol* (BGP), e de encaminhamento interno com o *Open Shortest Path First* (OSPF). Sendo o BGP o protocolo *standard* em termos de protocolos *Exterior Gateway Portocols* (EGP), pretende-se com a implementação deste protocolo avaliar o modo como as

comunicações entre Sistemas Autónomos (AS) diferentes tratam as comunicações *Anycast*. O BGP é um protocolo baseado em políticas, utiliza um algoritmo de vector de distâncias e não possui suporte para balanceamento de carga. O protocolo IGP implementado foi o OSPFv3. O OSPF é um protocolo baseado no estado das ligações, onde todos os nós conhecem a topologia da rede completa e calculam os seus caminhos mais curtos. O algoritmo utilizado no cálculo dos caminhos é o de caminhos mais curtos (SPF) de Dijkstra. Os caminhos são, normalmente, caracterizados por uma métrica baseada na largura de banda. O OSPF suporta ainda *multipath routing*, distribuindo equitativamente a carga por um máximo de 4 rotas de igual custo por destino, sendo este um dos principais motivos que levaram à sua escolha. As limitações do protocolo IGP RIP em termos de número de saltos e o facto deste protocolo não implementar balanceamento de carga levaram à exclusão deste protocolo na realização dos testes. Outros protocolos como o EIGRP que possuem capacidade de balancear carga não foram seleccionados por serem protocolos proprietários.

Na Figura 4.2, é apresentada a arquitectura de rede utilizada para a realização dos testes. Esta arquitectura foi utilizada com reconfigurações nos *routers*, com os protocolos OSPF e BGP.

De modo a avaliar os diferentes aspectos referidos, os testes são realizados com os servidores à mesma ”distância” do cliente e a ”distâncias” diferentes. Para simular estas ”distâncias”, no OSFP foi utilizado o parâmetro `COST` e no BGP o parâmetro `WEIGHT`, manipulando assim a arquitectura de rede.

A implementação da arquitectura de rede, foi realizada através da configuração duma rede emulada.

Para efectuar os testes foram implementados dois programas: `cliente HTTP`, e `servidorHTTP`. O programa `servidor HTTP` implementa um servidor com o protocolo HTTP 1.1 que possui o objecto `index.html`. Por seu lado, o programa `cliente HTTP`, é responsável por fazer 100 pedidos consecutivos do objecto `index.html`. Após receber os 100 objectos termina a sua execução. O programa `cliente HTTP`, possui ainda a particularidade de determinar o tempo que demora a receber os 100 objectos.

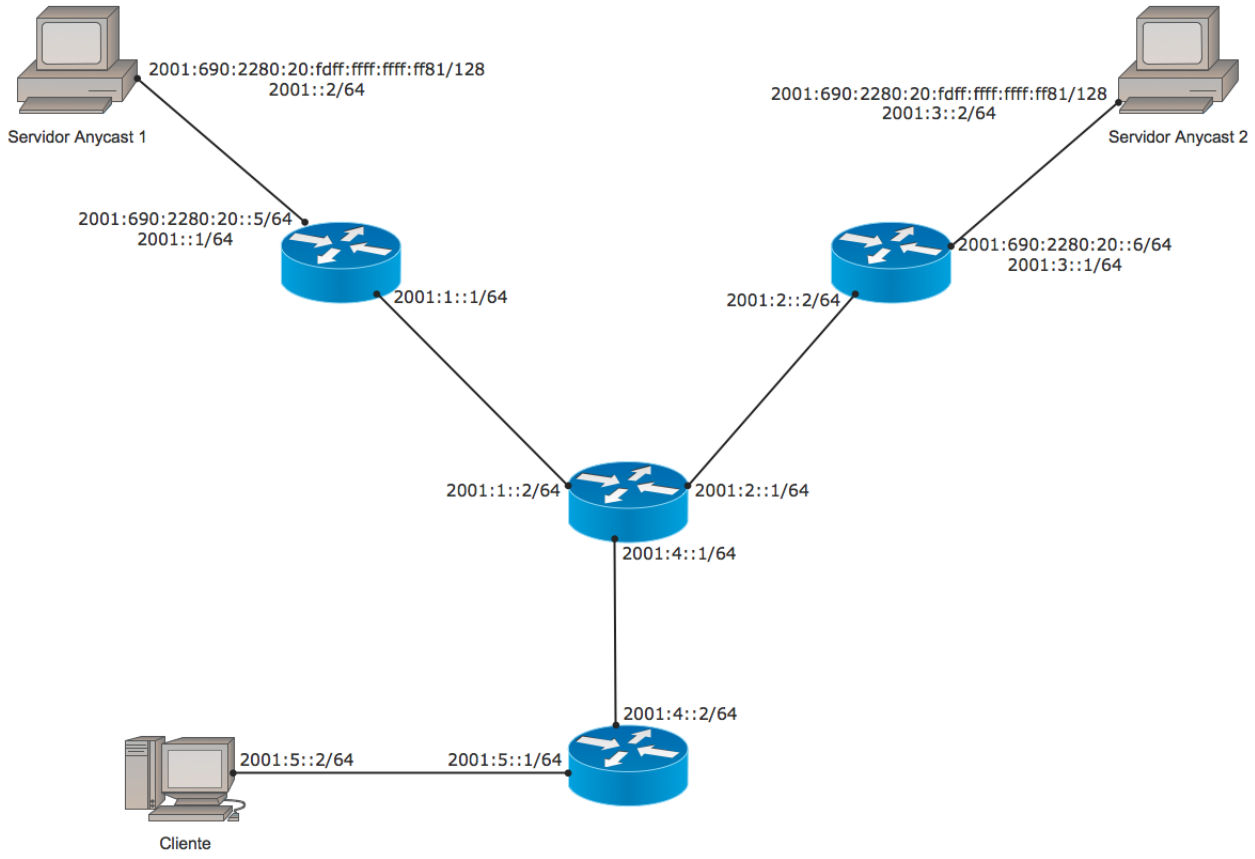

Figura 4.2: Arquitectura de rede

Os programas foram desenvolvidos na linguagem de programação `JAVA` [37].

De modo a demonstrar o modo como os protocolos de *routing Unicast* OSPF e BGP lidam com o *Anycast* em termos de balanceamento de carga e o redireccionamento em caso de falha, são efectuados 4 testes diferentes utilizando a arquitectura de rede apresentada e os programas `cliente HTTP` e `servidor HTTP`. Os testes foram realizados em cada um dos protocolos de *routing* e implementados através da rede emulada. Os testes são os que se seguem:

- **1º Teste** - Com os servidores *Anycast* 1 e 2 à mesma "distância" colocar os programas `servidor HTTP` a correr nos servidores, o programa `cliente HTTP` a correr no cliente, e observar se é efectuado algum balanceamento de carga;
- **2º Teste** - Com os servidores *Anycast* 1 e 2 à mesma "distância" colocar os programas `servidor HTTP` a correr nos servidores, o programa `cliente HTTP` a correr no cliente, e enquanto o programa `cliente HTTP` está a correr causar uma falha no servidor *Anycast* que está a responder aos pedidos. De seguida observa-se se o cliente é reencaminhado para o outro servidor *Anycast*;

- **3º Teste** - Com os servidores *Anycast* 1 e 2 a ”distâncias” diferentes colocar os programas `servidor HTTP` a correr nos servidores, o programa `cliente HTTP` a correr no cliente, e observar se é efectuado algum balanceamento de carga;

- **4º Teste** - Com os servidores *Anycast* 1 e 2 a ”distâncias” diferentes colocar os programas `servidor HTTP` a correr nos servidores, o programa `cliente HTTP` a correr no cliente, e enquanto o programa `cliente HTTP` está a correr causar uma falha no servidor *Anycast* que está a responder aos pedidos. De seguida observa-se se o cliente é reencaminhado para o outro servidor *Anycast*;

Na implementação dos testes, são aplicadas métricas que dependem do protocolo de encaminhamento utilizado para implementar as distâncias do cliente aos servidores. As falhas são simuladas desligando as ligações (colocando *down*) entre o R2 e o R3 e R4. Para fosse possível em termos de tempo para desligar as ligações, foi diminuída a largura de banda das ligações entre R2 e o R3 e R4, aumentando assim o tempo de resposta dos servidores. Os testes foram realizados 15 vezes cada, medindo sempre os tempos obtidos de modo a tirar conclusões sobre os mesmos.

A rede emulada foi implementada utilizando a ferramenta *Common Open Research Emulator* (CORE) [38]. A ferramenta fornece um GUI para desenhar as topologias de rede e configurar os equipamentos activos (ex. *routers*, *switchs*, *Access Points*). O CORE constrói uma representação de uma rede de computadores real que corre em tempo real. As redes emuladas podem ser conectadas a redes e *routers* físicos, ou mesmo a outras redes emuladas noutras máquinas. O *software* de *routing* utilizado pelo CORE é o Zebra/Quagga [39].

Inicialmente serão apresentados os resultados obtidos nos quatro testes efectuados em situações de encaminhamento interno com o protocolo de *routing* OSPF, e de seguida serão apresentados os obtidos em situações de encaminhamento externo com o BGP. De modo a tornar mais perceptível os resultados, serão também apresentados alguns gráficos com médias dos tempos obtidos nos testes, e o respectivo desvio padrão.

Após a rede emulada estar implementada, foram medidos os tempos que cada um dos servidores *Anycast* levam para responder aos pedidos do programa `cliente HTTP` em situação normal. Estes tempos são apresentados nos gráficos das Figuras 4.3 e 4.4, e têm como objectivo tornar visível o tempo que foi necessário, em média, para que um cliente fosse reencaminhado em caso de falha de um servidor. Como é possível observar pelos gráficos, os servidores *Anycast* (1ª e 2ª coluna de cada gráfico) demoram em média 32 segundos a responder aos 100 pedidos. Nos resultados, os tempos de redireccionamento em caso de falha, são a relação entre o tempo obtido quando uma falha ocorre e o tempo que os servidores levam a responder numa situação normal (em média 32 segundos).

Os dois servidores *Anycast* possuem o mesmo tempo de resposta média aos 100 pedidos, e por este motivo a as ”distâncias” relativas ao cliente vão ser variadas utilizando a métrica `COST`, não afectando assim os tempos de resposta. Assim, a diferença nos tempos de resposta em caso de falha varia apenas no tempo que o cliente leva a ser redireccionado.

O primeiro protocolo de *routing Unicast* a ser testado foi o OSPF.

Ao realizar o primeiro teste, não se verificou nenhum balanceamento de carga, ou seja, todos os pedidos foram para o mesmo servidor. Este resultado foi inesperado pois a especificação OSPF indica que quando existem dois caminhos de igual custo, o tráfego é balanceado de igual modo pelos caminhos. Como tal, este facto pode dever-se à implementação do OSPFv3 utilizada pelo CORE, pois após este teste foi possível observar que o *software* de *routing* do CORE não possui suporte *multipath routing* no contexto IPv6.

No segundo teste, verifica-se que ao originar uma falha no servidor que está a responder aos pedidos o cliente é reencaminhado para o outro servidor *Anycast*. No gráfico representado na Figura 4.3 é possível observar a média dos tempos obtidos no teste.

Com o terceiro teste, não se verificou nenhum balanceamento de carga, ou seja, todos os pedidos foram para o mesmo servidor. Este resultado foi o esperado devido à especificação OSPF.

No quarto teste, verifica-se que ao originar uma falha no servidor que está a responder aos pedidos o cliente é reencaminhado para o outro servidor *Anycast*. No gráfico representado na Figura 4.3 é possível observar a média dos tempos obtidos no teste.

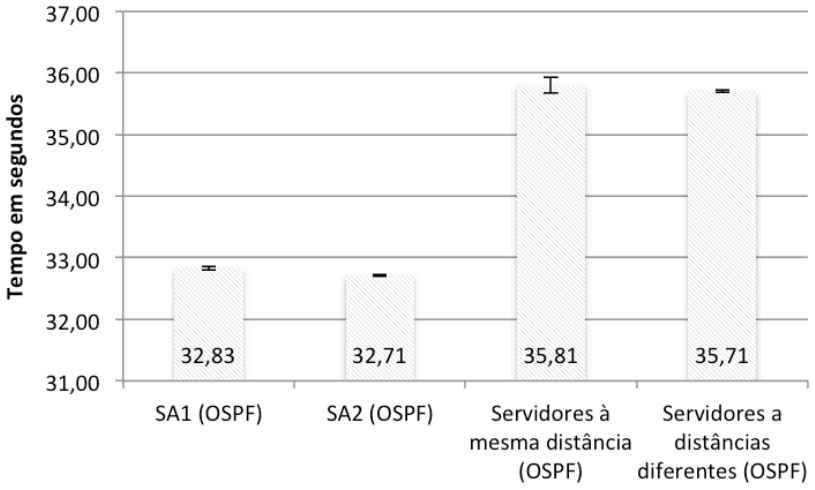

Figura 4.3: Resultados obtidos nos testes 2 e 4 na implementação OSPF

As colunas SA1 e SA2 do gráfico representado na Figura 4.3 demonstram o tempo que os servidores SA1 e SA2 levam a responder aos 100 pedidos sem a ocorrência de falhas. Estes valores apresentados servem meramente de referência. É possível observar pelo gráfico representado na Figura 4.3 que em caso de falha o cliente é reencaminhado para outro servidor, sendo que a diferença dos tempos apresentados nas situações em que os servidores estão à ”mesma distância” e a ”distâncias diferentes” não se revelaram estatisticamente relevantes.

Após os testes sobre o protocolo OSPF na rede emulada, configurou-se a mesma arquitectura de rede para utilizar o protocolo de *routing* BGP. Para tal, os dois servidores foram colocados em AS diferentes.

O primeiro e o terceiro teste não foram realizados com a implementação BGP. Esta decisão foi tomada pelo facto do BGP não efectuar nenhum balanceamento de carga.

No segundo teste, verifica-se que ao originar uma falha no servidor que está a responder aos pedidos o cliente é reencaminhado para o outro servidor *Anycast*. No gráfico representado na Figura 4.3 é possível observar a média dos tempos obtidos no teste.

No quarto teste, verifica-se que ao originar uma falha no servidor que está a responder aos pedidos o cliente é reencaminhado para o outro servidor *Anycast*. No gráfico representado na Figura 4.4 é possível observar a média dos tempos obtidos no teste.

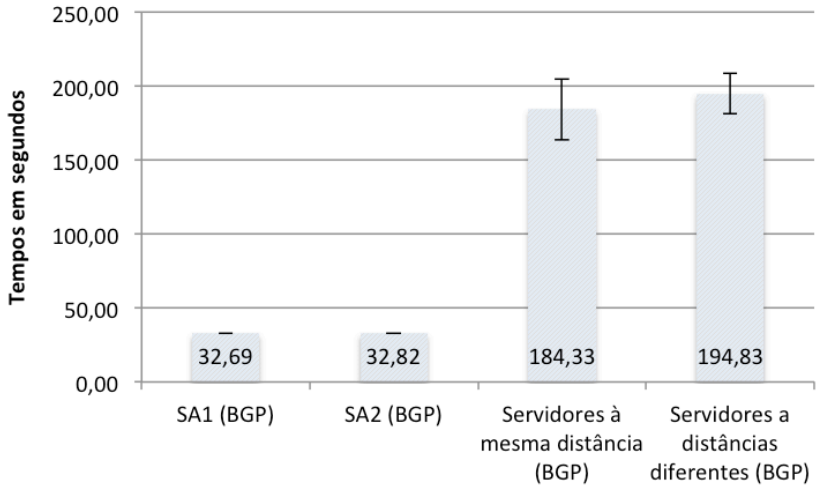

Figura 4.4: Resultados obtidos nos testes 2 e 4 na implementação BGP

As colunas SA1 e SA2 do gráfico representado na Figura 4.4 demonstram o tempo que os servidores SA1 e SA2 levam a responder aos 100 pedidos sem a ocorrência de falhas. Estes valores apresentados servem meramente de referência. É possível observar pelo gráfico representado na Figura 4.4 que em caso de falha o cliente é reencaminhado para outro servidor, sendo que a diferença dos tempos apresentados nas duas situações não se revelaram estatisticamente relevantes.

Após observar os tempos obtidos nos testes, é possível observar a diferença de valores obtidos pelo BGP e OSPF, revelando-se este último muito mais rápido nas situações de falha. No entanto é importante referir que estes tempos constituem referências e não resultados absolutos, devido ao facto de serem utilizadas as configurações por defeito dos protocolos OSPF e BGP, e como tal os valores por defeito dos seus *timers*, sendo ainda necessário relembrar que as ligações entre os *Routers* R2, R3 e R4 foram deliberadamente alteradas no CORE, de modo a tornar as ligações mais lentas.

4.1.3.3 Estabelecimento de conexões TCP

O estabelecimento de conexões TCP com um endereço *Anycast* é, segundo a especificação do *Anycast*, impossível devido ao facto dos endereços *Anycast* não poderem ser utilizados como endereços de origem.

Neste teste foram utilizados dois programas, `servidor` e `cliente`, desenvolvidos apenas para testes. O programa `cliente` tenta criar um `socket` TCP com o endereço *Anycast*, em caso de sucesso o servidor envia um ficheiro para o cliente através da conexão criada. Estes programas foram desenvolvidos na linguagem de programação `JAVA`.

Após efectuado o teste verificou-se que é possível o estabelecimento de uma conexão TCP com um endereço *Anycast*. Como pode ser observado pela Figura 4.5, verifica-se que o servidor utiliza na conexão o endereço *Anycast* como endereço de origem. A Figura 8 consiste numa captura de imagem ao programa de captação e análise de tráfego `Wireshark`, durante a troca do ficheiro entre o cliente e o servidor *Anycast*.

Para a realização deste teste foram utilizadas duas máquinas com o SO Mac OS X 10.6 a correr o programa `servidor`, e uma máquina com o SO Ubuntu 11.04 para correr o programa `cliente`. São utilizados dois servidores neste teste de modo a verificar se todos os pacotes TCP são enviados para o servidor correcto. Neste teste todas as máquinas estavam ligadas na mesma rede. Ao realizar este teste utilizando estes programas, observou-se que o cliente conseguia estabelecer um `socket` TCP com um dos servidores, e que durante a conexão nenhum pacote foi enviado para o servidor errado. Os programas cliente e servidor correram sem a ocorrência de nenhum problema, conseguindo efectuar a troca dos objectos. Conclui-se que os motivos que tornam o estabelecimento da conexão TCP possível nesta situação, são o facto do endereço *Anycast* não ser, nesta situação, diferenciado do *Unicast*, pois verificou-se que ao observar o tráfego TCP trocado entre o cliente e o servidor, através do programa de captura de tráfego `Wireshark`, os pacotes TCP enviados pelo servidor possuiam o endereço *Anycast* no campo endereço de origem do pacote, o que não era suposto.

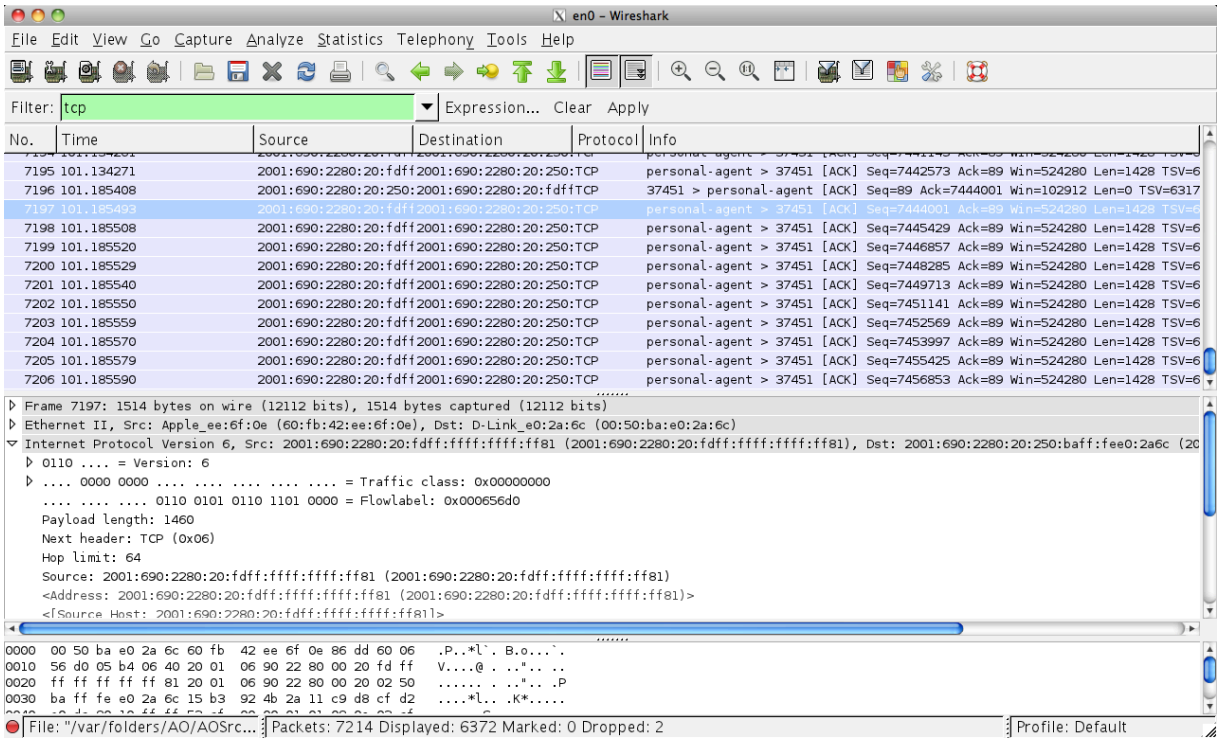

Figura 4.5: Criação de uma conexão TCP com um endereço *Anycast*

4.2 Segurança IP

O IPSec é uma ferramenta de segurança introduzida como *standard* pelo IPv6 amplamente utilizada nos dias de hoje. Nesta secção são apresentadas as plataformas que actualmente suportam IPSec, ferramentas para a utilização do IPSec e ainda alguns exemplos de implementação.

4.2.1 Análise de suporte em várias plataformas

O IPSec, ao contrário do protocolo *Anycast*, é uma ferramenta com standards bem definidos e já emplamente utilizada. No entanto, considerou-se necessário verificar se esta tecnologia é suportada nos diversos SOs.

Como é possível observar pela tabela 4.2, todos os SOs testados possuem suporte para IPSec nativamente. Foram testados estes três SOs, pois são representativos das famílias BSD, Linux e Windows respectivamente segundo a ordem apresentada na tabela.

Plataforma	Suporte IPSec
Mac OS X 10.6.7	Sim
Ubuntu 10.10	Sim
Windows	Sim

Tabela 4.2: Plataformas que suportam IPSec

4.2.2 Ferramentas utilizadas

Na secção 2.3 é efectuado um estudo ao estado de arte do IPSec, onde esta tecnologia é descrita em detalhe. Nessa secção, é ainda explicado o modo como esta tecnologia opera e quais os parâmetros configuráveis.

Ao utilizarmos o IPSec, necessitamos de definir muito bem quais as políticas de segurança e as associações de segurança que pretendemos aplicar ao tráfego, pois são a chave para o bom funcionamento do IPSec.

Para implementar o IPSec nos SOs Ubuntu e Mac OS X, podemos utilizar o `IPsec-Tools`. `IPsec-Tools` [40] é uma *framework* IPSec da `KAME` [41] para a implementação IPsec nas plataformas: Linux 2.6, NetBSD e FreeBSD. O `IPsec-Tools` é constituído pelas seguintes ferramentas:

- **`Setkey`** - Ferramenta para manipular e consultar a base de dados de políticas de segurança (SPD) e a base de dados se associações de segurança (SAD);
- **`Racoon`** - *Daemon* IKE para geração automática de chaves para conexões IPsec, e negociação de SAs;
- **`Racoonctl`** - Ferramenta de controlo baseado em `bash` para o `Racoon`.

A ferramenta `setkey` [42] permite manipular directamente as bases de dados SAD e SPD, para tal são utilizados dois comandos: o comando `add` e o `spdadd`.

O comando `add` adiciona uma associação de segurança na SAD. Para tal são necessários os endereços IP de origem e destino, o protocolo IPsec (AH ou ESP), o SPI e os algoritmos pretendidos. O algoritmo de autenticação é especificado com o parâmetro `-A`, o algoritmo de

cifragem através do parâmetro `-E`, e o de compressão através do `-C` (a compressão IP ainda não é suportada). De seguida o algoritmo de autenticação/cifragem deve ser especificado. A chave pode ser formatada em aspas duplas ”ASCII” ou em hexadecimal com um 0x inicial.

O comando `spdadd` adiciona as políticas de segurança na SPD. Estas políticas definem quais pacotes que devem ser protegidos por IPsec e quais os protocolos e chaves a usar. O comando necessita dos endereços IP de origem e de destino dos pacotes a serem protegidos, o protocolo (e porto caso se pretenda) para proteger e a política a usar (`-P`). A política especifica a direcção (`in / out`), a acção de aplicar (`ipsec / discard / none`), o protocolo (`AH / ESP / ipcomp`), o modo (`transport/tunnel`) e o nível (`use / require`).

A ferramenta `racoon` [43] consiste num *daemon* IKE para troca de chaves utilizado no protocolo IPsec.

No caso de chaves previamente partilhadas, os *hosts* são autenticados dinâmicamente pelo `racoon` utilizando chaves secretas partilhadas por ambos os *hosts*.

No caso da negociação automática de chaves secretas, os *hosts* são autenticados utilizando certificados que comprovam a sua identidade.

O ficheiro de configuração do `racoon`, racoon.conf, está localizado na directoria /etc/racoon/. É através deste ficheiro que são definidos os parâmetros utilizados na negociação das SAs, e caso seja necessário das chaves secretas. A configuração do `racoon` consiste em inserir secções com os parâmetros pretendidos para as conexões no ficheiro de configuração.

Existem dois tipos de secções: `remote` e `sainfo`. As secções `remote` definem os parâmetros para as conexões com os sistemas remotos, que consiste na Fase 1. Por seu lado, as secções `sainfo` fornecem as instruções para a criação das SAs, tais como os algoritmos de cifragem e autenticação a usar, que por sua vez consiste na Fase 2.

As secções `remote` e `sainfo` possuem várias directivas utilizadas na sua configuração. Na secção `remote` existem as seguintes directivas:

- **`exchange_mode`** - Directiva que especifica o modo de negociação da Fase 1;

- **identifier** - Directiva que especifica o próprio identificador utilizado no sistema remoto;
- **lifetime time** e **lifetime byte** - Directivas que especificam o tempo de vida da SA IKE;
- **proposal** - Directiva que especifica uma proposta sobre a Fase 1;
- **encryption_algorithm** - Directiva que especifica o algoritmo de cifragem utilizado na Fase 1. O racoon pode usar DES, 3DES, RC5, IDEA, CAST, BLOWFISH como algoritmos de cifragem;
- **hash_algorithm** - Directiva que especifica algoritmo de *hash*. O racoon pode usar MD5 e SHA1 como algoritmo de *hash*;
- **authentication_method** - Directiva especifica o método de autenticação na Fase 1;
- **dh_group** - Directiva que especifica um grupo para troca de chaves *Diffie-Hellman*. O racoon pode utilizar 1, 2 e 5 como grupo;

Na secção sainfo existem as seguintes directivas:

- **pfs_group** - Directiva especifica um grupo para troca de chaves *Diffie-Hellman* na Fase 2;
- **lifetime time** e ***lifetime byte*** - Directivas que especificam o tempo de vida da SA IPsec;
- **pfs_group** - Directiva especifica um grupo para troca de chaves *Diffie-Hellman* na Fase 2;
- **encryption_algorithm** - Directiva que especifica a proposta de algoritmo de cifragem para o ESP. O racoon pode utilizar DES, 3DES, CAST, Blowfish, Twofish, Rijndael e NULL como algoritmo de cifragem para ESP;
- **authentication_method** - Directiva que especifica a proposta de algoritmo de autenticação para o ESP e AH. O racoon pode utilizar HMAC-SHA1, MD5 e HMAC-Keyed-MD5 como algoritmo de autenticação;
- **compression_algorithm** - Directiva que especifica a proposta de algoritmo de compressão para IPCOMP. Neste momento o racoon pode utilizar DEFLATE.

4.2.3 Configurar IPSec

Nesta secção são apresentados alguns exemplos de implementações IPSec utilizando as ferramentas apresentadas na secção anterior.

Nesta fase de testes, o IPSec foi implementado tanto em IPv4 como em IPv6. No entanto, visto o âmbito desta dissertação ser o IPv6, todos os exemplos apresentados são em IPv6 salientando, caso existam, as diferenças com as configurações IPv4.

Os exemplos apresentados representam três modos diferentes de configurar o IPSec:

- 1º Modo - Configuração manual da SPD e SAD com chaves secretas partilhadas por ambos os nós, através da ferramenta `setkey`;
- 2º Modo - Configuração manual da SPD e geração automática das SAs pelo `racoon`, utilizando chaves secretas partilhadas por ambos os nós;
- 3º Modo - Configuração manual da SPD e geração automática das SAs e das chaves secretas pelo `racoon`, utilizando para tal certificados de chave pública/privada.

De seguida são apresentados exemplos para cada um dos modos de configuração IPSec. Os exemplos que se seguem foram todos desenvolvidos nos SOs Mac OS X e Ubuntu.

Os exemplos apresentados são representativos de uma das extremidades da conexão. Isto deve-se ao facto das configurações serem semelhantes em ambas as extremidades, diferenciando-se apenas nas direcções aplicadas às mesmas.

4.2.3.1 1º Modo - Configuração manual da SPD e da SAD

Neste modo de configuração as SA e as SP são manualmente inseridas na SAD e SPD respectivamente. Tal como é referido na secção 2.3, para cada nó de uma conexão são necessárias pelo menos duas SA e duas SP, podendo assim definir o tratamento que será aplicado ao tráfego de entrada e saída.

De modo a implementar o IPSec, neste exemplo é apenas utilizada a ferramenta `setkey` para configurar o IPSec entre as máquinas com as endereços 2001:690:2280:20:250:baff:fee0:2a6c e 2001:690:2280:20:62fb:42ff:feee:6f0e. Assim sendo, através do `setkey` as SAs e as SPs são introduzidas manualmente na SAD e SPD respectivamente.

No exemplo que se segue, o IPSec é configurado para fornecer proteção AH e ESP com chaves secretas previamente partilhadas, a todo o tráfego trocado entre as máquinas com os endereços 2001:690:2280:20:250:baff:fee0:2a6c e 2001:690:2280:20:62fb:42ff:feee:6f0e.

Como é possível observar no Código 4.5, inicialmente introduzimos as SAs AH com as chaves secretas 0xc0291ff014dccdd03874d9e8e4cdf3e6 e 0x96358c90783bbfa3d7b196ceabe0536b. São introduzidas duas SAs, uma para o tráfego de entrada e outra para o de saída. As SAs são inseridas com os valores 0x200 e 0x300 para os respectivos SPIs. Depois é definido o protocolo de autenticação através do parâmetro `-A`. O protocolo de autenticação utilizado neste caso é o hmac-md5.

```
1 add 2001:690:2280:20:250:baff:fee0:2a6c 2001:690:2280:20:62fb:42ff:feee:6f0e ah 0x200 -A hmac-
      md5 0xc0291ff014dccdd03874d9e8e4cdf3e6;
2
3 add 2001:690:2280:20:62fb:42ff:feee:6f0e 2001:690:2280:20:250:baff:fee0:2a6c ah 0x300 -A hmac-
      md5 0x96358c90783bbfa3d7b196ceabe0536b;
```

Código 4.5: 1º modo de configuração: introduzir SAs AH na SAD

De seguida introduzimos as SAs ESP com as chaves secretas na sua representação hexadecimal :

- 0x7aeaca3f87d060a12f4a4487d5a5c3355920fae69a96c831
- 0xf6ddb555acfd9d77b03ea3843f2653255afe8eb5573965df

São introduzidas duas SAs, uma para o tráfego de entrada e outra para o de saída. As SAs são inseridas com os valores 0x201 e 0x301 para os respectivos SPIs, definindo de seguida o protocolo de cifragem através do parâmetro `-E`. O protocolo de autenticação utilizado neste caso é o 3des-cbc. Estas configurações são apresentadas no Código 4.6.

```
1 add 2001:690:2280:20:250:baff:fee0:2a6c 2001:690:2280:20:62fb:42ff:feee:6f0e esp 0x201 -E 3des
      -cbc 0x7aeaca3f87d060a12f4a4487d5a5c3355920fae69a96c831;
2
3 add 2001:690:2280:20:62fb:42ff:feee:6f0e 2001:690:2280:20:250:baff:fee0:2a6c esp 0x301 -E 3des
      -cbc 0xf6ddb555acfd9d77b03ea3843f2653255afe8eb5573965df;
```

Código 4.6: 1º modo de configuração: introduzir SAs ESP na SAD

Agora são introduzidos as SPs na SPD. As SPs indicam qual o tipo de tratamento que se deve dar ao tráfego, como tal, não é necessário possuirmos SPs diferentes para o AH e ESP. Nas SPs especifica-se qual o tráfego que requer tratamento especial, neste caso é o trocado entre os endereços 2001:690:2280:20:62fb:42ff:feee:6f0e e 2001:690:2280:20:250:baff:fee0:2a6c. Neste caso não foi especificada nenhum porto, logo a política abrange todo o tráfego trocado entre os dois endereços referidos, que chega a qualquer um dos portos da máquina.

O tipo de protocolo de comunicação ao qual se pretende aplicar o tratamento é também um dos parâmetros configuráveis. Neste caso pretende-se aplicar o tratamento a todo o tráfego, como tal utilizou-se o parâmetro `any` em vez de especificarmos um protocolo.

De seguida é definido a direcção, a acção a aplicar, o protocolo utilizado, o modo e ainda o nível. Visto pretender-se aplicar o IPSec ao tráfego de entrada e saída teremos uma SP com a direcção `in` e outra `out`. A seguir é definida a acção, neste caso é `ipsec`. Por último é necessário definir o protocolo, o modo e o nível estão dependentes do protocolo escolhido. Assim, visto serem aplicados os dois protocolos AH e ESP, é necessário definir o modo e o nível para cada um deles. Neste exemplo, optou-se pela utilização do modo `transport` em ambos os protocolos, e também do nível `require` em ambos. No Código 4.7 estão representados os comandos utilizados.

```
1 spdadd 2001:690:2280:20:250:baff:fee0:2a6c 2001:690:2280:20:62fb:42ff:feee:6f0e
2  any -P in ipsec esp/transport//require ah/transport//require;
3
4 spdadd 2001:690:2280:20:62fb:42ff:feee:6f0e 2001:690:2280:20:250:baff:fee0:2a6c
5  any -P out ipsec esp/transport//require ah/transport//require;
```

Código 4.7: 1º modo de configuração: introduzir SPs na SPD

Após aplicarmos estas configurações, podemos consultar a SAD através do comando `setkey -D`, e a SPD através do comando `setkey -D -P`. As Figuras 4.6 e 4.7 representam as bases de dados SAD e SPD da máquina 2001:690:2280:20:62fb:42ff:feee:6f0e, após aplicadas as configurações.

```
dhcp-46:~ joaoveiga$ sudo setkey -D
2001:690:2280:20:62fb:42ff:feee:6f0e 2001:690:2280:20:250:baff:fee0:2a6c
        esp mode=any spi=769(0x00000301) reqid=0(0x00000000)
        E: 3des-cbc  f6ddb555 acfd9d77 b03ea384 3f265325 5afe8eb5 573965df
        seq=0x0000000d replay=0 flags=0x00000040 state=mature
        created: Jun  6 10:27:16 2011   current: Jun  6 10:29:16 2011
        diff: 120(s)    hard: 0(s)      soft: 0(s)
        last: Jun  6 10:28:27 2011      hard: 0(s)      soft: 0(s)
        current: 1072(bytes)    hard: 0(bytes)  soft: 0(bytes)
        allocated: 13   hard: 0 soft: 0
        sadb_seq=3 pid=3790 refcnt=15
2001:690:2280:20:250:baff:fee0:2a6c 2001:690:2280:20:62fb:42ff:feee:6f0e
        esp mode=any spi=513(0x00000201) reqid=0(0x00000000)
        E: 3des-cbc  7aeaca3f 87d060a1 2f4a4487 d5a5c335 5920fae6 9a96c831
        seq=0x00000000 replay=0 flags=0x00000040 state=mature
        created: Jun  6 10:27:16 2011   current: Jun  6 10:29:16 2011
        diff: 120(s)    hard: 0(s)      soft: 0(s)
        last: Jun  6 10:28:27 2011      hard: 0(s)      soft: 0(s)
        current: 680(bytes)     hard: 0(bytes)  soft: 0(bytes)
        allocated: 12   hard: 0 soft: 0
        sadb_seq=2 pid=3790 refcnt=2
2001:690:2280:20:62fb:42ff:feee:6f0e 2001:690:2280:20:250:baff:fee0:2a6c
        ah mode=any spi=768(0x00000300) reqid=0(0x00000000)
        A: hmac-md5  96358c90 783bbfa3 d7b196ce abe0536b
        seq=0x0000000d replay=0 flags=0x00000040 state=mature
        created: Jun  6 10:27:16 2011   current: Jun  6 10:29:16 2011
        diff: 120(s)    hard: 0(s)      soft: 0(s)
        last: Jun  6 10:28:27 2011      hard: 0(s)      soft: 0(s)
        current: 1384(bytes)    hard: 0(bytes)  soft: 0(bytes)
        allocated: 13   hard: 0 soft: 0
        sadb_seq=1 pid=3790 refcnt=2
2001:690:2280:20:250:baff:fee0:2a6c 2001:690:2280:20:62fb:42ff:feee:6f0e
        ah mode=any spi=512(0x00000200) reqid=0(0x00000000)
        A: hmac-md5  c0291ff0 14dccdd0 3874d9e8 e4cdf3e6
        seq=0x0000000c replay=0 flags=0x00000040 state=mature
        created: Jun  6 10:27:16 2011   current: Jun  6 10:29:16 2011
        diff: 120(s)    hard: 0(s)      soft: 0(s)
        last: Jun  6 10:28:27 2011      hard: 0(s)      soft: 0(s)
        current: 968(bytes)     hard: 0(bytes)  soft: 0(bytes)
        allocated: 12   hard: 0 soft: 0
        sadb_seq=0 pid=3790 refcnt=2
```

Figura 4.6: SAD, 1º modo

```
dhcp-46:~ joaoveiga$ sudo setkey -D -P
2001:690:2280:20:250:baff:fee0:2a6c[any] 2001:690:2280:20:62fb:42ff:feee:6f0e[any]
 any
        in ipsec
        esp/transport//require
        ah/transport//require
        spid=1 seq=1 pid=3794
        refcnt=2
2001:690:2280:20:62fb:42ff:feee:6f0e[any] 2001:690:2280:20:250:baff:fee0:2a6c[any]
 any
        out ipsec
        esp/transport//require
        ah/transport//require
        spid=2 seq=0 pid=3794
        refcnt=2
dhcp-46:~ joaoveiga$
```

Figura 4.7: SPD, 1º modo

4.2.3.2 2º Modo - Configuração manual da SPD e SAs negociadas com chaves previamente partilhadas

Neste modo de configuração as SPs são manualmente inseridas na SPD, no entanto, as SAs são negociadas através da ferramenta `racoon`. O modo de configuração utilizado para a negociação das SAs é o que requer uma partilha prévia das chaves secretas entre as duas extremidades da conexão. Tal como é referido na secção 2.3, para cada nó de uma conexão são necessárias pelo menos duas SAs e duas SPs, podendo assim definir o tratamento que será aplicado ao tráfego de entrada e saída.

Visto as SAs neste exemplo serem negociadas automaticamente, neste exemplo para além da ferramenta `setkey` é também necessário utilizar o `racoon` de modo a configurar o IPSec entre as máquinas 2001:690:2280:20:62fb:42ff:feee:6f0e e 2001:690:2280:20:216:cbff:feac:5b2f. Assim sendo, através do `setkey` serão introduzidas as SPs na SPD e através do `racoon` as SAs serão introduzidas na SAD.

No exemplo que se segue, o IPSec é configurado para fornecer proteção ESP com chaves secretas previamente partilhadas, a todo o tráfego trocado entre as máquinas com os endereços 2001:690:2280:20:62fb:42ff:feee:6f0e e 2001:690:2280:20:216:cbff:feac:5b2f.

Inicialmente são introduzidos as SPs na SPD. As SPs indicam qual o tipo de tratamento que se deve dar ao tráfego. Nas SPs especifica-se qual o tráfego que requer tratamento especial, neste caso é o trocado entre as máquinas com os endereços 2001:690:2280:20:62fb:42ff:feee:6f0e e

2001:690:2280:20:216:cbff:feac:5b2f. Neste caso não foi especificado nenhum porto, logo a política abrange todo o tráfego trocado entre os dois endereços referidos, que chega a qualquer um dos portos da máquina.

O tipo de protocolo de comunicação ao qual se pretende aplicar o tratamento é também um dos parâmetros configuráveis. Neste caso pretende-se aplicar o tratamento a todo o tráfego, como tal utilizou-se o parâmetro `any` em vez de especificarmos um protocolo.

De seguida é definida a direcção, a acção a aplicar, o protocolo utilizado, o modo e ainda o nível. Visto pretender-se aplicar o IPSec ao tráfego de entrada e saída teremos uma SP com a direcção `in` e outra `out`. A seguir é definida a acção, neste caso é `ipsec`. Por último é necessário definir o protocolo, o modo e o nível estão dependentes do protocolo escolhido, neste caso o ESP. Neste exemplo, optou-se pela utilização do modo `transport` e do nível `require`. Os comandos utilizados nesta configuração estão representados no código 4.8.

```
1 spdadd 2001:690:2280:20:62fb:42ff:feee:6f0e 2001:690:2280:20:216:cbff:feac:5b2f any -P out
      ipsec esp/transport//require;
2
3 spdadd 2001:690:2280:20:216:cbff:feac:5b2f 2001:690:2280:20:62fb:42ff:feee:6f0e any -P in
      ipsec esp/transport//require
```

Código 4.8: 2º modo de configuração: introduzir SPs na SPD

Ao configurar o IPSec em IPv6 utilizando o `racoon` para negociar as SAs, verificou-se que para a utilização do nível `require` nas SPs o tráfego do protocolo ICMPv6 não poderia ser protegido pelo IPSec. Ou seja, caso fossem apenas introduzidas as duas SPs apresentadas, as SAs não iriam ser negociadas. Como tal, são necessárias duas novas SPs. Esta particularidade não se verifica ao configurar o IPSec, nas mesmas circunstâncias, em IPv4. A utilização do nível `use` em vez de `require` não necessita que sejam inseridas estas duas novas SPs. Através dos testes, verificou-se que o problema apenas ocorria se o processo de geração das SAs fosse despoletado pelo tráfego ICMPv6. Para iniciar uma negociação de SAs, primeiro o cliente e o servidor *Anycast* necessitam dos endereços *link-layer* de ambos. Após a análise do tráfego e dos resultados obtidos, observou-se que que o cliente despoletava a negociação de SAs através dum pacote ICMPv6, de seguida são trocadas mensagens `Neighbor solicitation`, que utilizam o protocolo ICMPv6. A mensagem `Neighbor solicitation` ao chegar ao servidor

vai despoletar também a criação das SAs. O servidor envia a resposta, mas esta quando chega ao destino não consegue passar pelos filtros IPSec pois existem SPs que requerem protecção ESP/AH, despoletando de novo a criação das SAs, gerando um *loop*, e não criando as SAs.

Nas SPs a inserir, é definido o tipo de protocolo de comunicação ao qual se pretende aplicar o tratamento, que neste caso é o ICMPv6. De seguida é definida a direcção, a acção a aplicar. Visto pretender-se que os pacotes deste protocolo não sejam protegidos, não se pretende aplicar nenhuma acção, logo o parâmetro utilizado é `none`. Os comandos utilizados nesta configuração estão representados no código 4.9.

```
1 spdadd 2001:690:2280:20:62fb:42ff:feee:6f0e 2001:690:2280:20:216:cbff:feac:5b2f ipv6-icmp -P
      out none;
2
3 spdadd 2001:690:2280:20:216:cbff:feac:5b2f 2001:690:2280:20:62fb:42ff:feee:6f0e ipv6-icmp -P
      in none;
```

Código 4.9: 2º modo de configuração: introduzir SPs ICMPv6 na SPD

Após inserir as SPs na SPD, o próximo passo é definir os parâmetros que serão utilizados para a negociação das SAs. Para tal, é necessário alterar o ficheiro de configuração do `racoon`, o ficheiro racoon.conf.

Neste exemplo a autenticação dos *hosts* é efectuada através da utilização de chaves previamente partilhadas, como tal primeiro é necessário definir o caminho para o ficheiro onde a chave está armazenada. Neste caso é o ficheiro psk.txt. O comando utilizado para definir o caminho está representado no Código 4.10.

```
1 path pre_shared_key "/etc/racoon/psk.txt" ;
```

Código 4.10: 2º modo de configuração: definir o caminho para o ficheiro onde a chave está armazenada

De seguida definem-se os parâmetros utilizados na primeira fase de negociação das SAs. São estes os parâmetros utilizados no estabelecimento das conexões com os sistemas remotos. Estes parâmetros são definidos introduzindo uma secção `remote` no ficheiro de configuração

do `racoon`. O sistema remoto com o qual se pretende estabelecer uma conexão é o identificado com o endereço IP 2001:690:2280:20:216:cbff:feac:5b2f.

Neste exemplo o método de autenticação é o de chaves secretas partilhadas. As configurações utilizadas estão representadas no Código 4.11.

```
remote 2001:690:2280:20:216:cbff:feac:5b2f
{
        exchange_mode aggressive,main;
        doi ipsec_doi;
        situation identity_only;

        my_identifier address;

        lifetime time 2 min;
        initial_contact on;
        proposal_check obey;

        proposal {
                encryption_algorithm 3des;
                hash_algorithm sha1;
                authentication_method pre_shared_key ;
                dh_group 2 ;
        }
}
```

Código 4.11: 2º modo de configuração: configuração do `racoon`, secção `remote`

Agora definem-se os parâmetros utilizados na segunda fase de negociação das SAs. Estes parâmetros fornecem as instruções para a criação das SAs, tais como os algoritmos de cifragem e autenticação a usar. Estes parâmetros são definidos introduzindo uma secção `sainfo` no ficheiro de configuração do `racoon`. As configurações utilizadas neste exemplo são configurações por defeito, utilizadas por várias conexões. Para que estas sejam utilizadas por várias conexões, é utilizado o parâmetro `anonymous` indicando que qualquer conexão que não possui uma secção `sainfo` própria pode utilizar esta. As configurações utilizadas estão representadas no Código 4.12.

```
sainfo anonymous
{
        pfs_group 1;
```

```
4         lifetime time 2min;
5         encryption_algorithm 3des ;
6         authentication_algorithm hmac_sha1;
7         compression_algorithm deflate ;
8 }
```

Código 4.12: 2º modo de configuração: configuração do `racoon`, secção `sainfo`

Para terminar as configurações necessárias, falta apenas definir a chave secreta a utilizar nesta conexão. De modo a definir a chave secreta, é necessário alterar o ficheiro psk.txt. No ficheiro psk.txt é necessário acrescentar uma linha com o endereço IP do sistema remoto da conexão e à sua frente a chave secreta a usar, como podemos observar no Código 4.13.

```
1
2 2001:690:2280:20:216:cbff:feac:5b2f    password
```

Código 4.13: 2º modo de configuração: definir chave secreta a utilizar nas conexões

4.2.3.3 3º Modo - Configuração manual da SPD e SAs e chaves secretas negociadas

Neste modo de configuração as SPs são manualmente inseridas na SPD, no entanto, as SAs são negociadas automáticamente utilizando a ferramenta `racoon` para o efeito. O modo de configuração utilizado para a negociação das SAs é o de negociação automática das chaves secretas entre as duas extremidades da conexão. Tal como é referido na secção 2.3, para cada nó de uma conexão são necessárias pelo menos duas SAs e duas SPs, podendo assim definir o tratamento que será aplicado ao tráfego de entrada e saída.

Visto as SAs neste exemplo serem negociadas automaticamente, neste exemplo para além da ferramenta `setkey` é também necessário utilizar o `racoon` de modo a configurar o IPSec entre as máquinas 2001:690:2280:20:62fb:42ff:feee:6f0e e 2001:690:2280:20:216:cbff:feac:5b2f. Assim sendo, através do `setkey` serão introduzidas as SPs na SPD e através do `racoon` as SAs serão introduzidas na SAD.

No exemplo que se segue, o IPSec é configurado para fornecer proteção ESP com chaves secretas negociadas através do IKE, a todo o tráfego trocado entre as máquinas com os endereços 2001:690:2280:20:62fb:42ff:feee:6f0e e 2001:690:2280:20:216:cbff:feac:5b2f.

Neste exemplo, tal como acontece no 2º modo de configuração as SPs são inseridas na SPD com a ferramenta `setkey`. Visto a única diferença entre os modos 2 e 3 de configurar o IPSec ser na configuração do `racoon`, neste exemplo são utilizadas as mesmas configurações da SPD utilizadas no 2º modo. Como tal essas configurações não serão explicadas mais uma vez.

O uso de chaves secretas partilhadas é difícil porque estas não são facilmente partilhadas e uma vez partilhadas não são mais secretas. No entanto, existe tecnologia de criptografia assimétrica para ajudar a resolver este problema.

Neste exemplo, de modo a negociar automáticamente as chaves secretas, são utilizados certificados X.509. Como tal, é necessário gerar um par de chaves pública/privada. Para gerar o par de chaves é utilizada ferramenta `openssl` [44].

O `OpenSSL` possui uma vasta infraestrutura de chaves que podem ou não ser certificadas por autoridades de certificação. No entanto para este exemplo não será necessário a utilização de uma autoridade de certificação por uma questão de comodidade, sendo recomendável usar autoridades de certificação pois um certificado auto-assinado não pode ser validado.

Primeiro é necessário emitir um ”pedido de certificado” para o nosso *host*, chamado de ”*laptop*”, como pode ser observado no Código 4.14.

```
1
2 openssl req -new -nodes -newkey rsa:1024 -sha1 -keyform PEM -keyout \ laptop.private -outform
      PEM -out request.pem
```

Código 4.14: 3º modo de configuração: emitir um pedido de certificado

De seguida atribuí-se o pedido ao *host*. O modo com é efectuado o pedido está representado no Código 4.15.

```
1
2 openssl x509 -req -in request.pem -signkey laptop.private -out \laptop.public
```

Código 4.15: 3º modo de configuração: atribuir certificado ao *host*

Após ambos os *hosts* possuírem os certificados, é necessário que ambos partilhem as suas chaves públicas. De seguida é necessário definir o caminho para a pasta onde estão os certificados. No Código 4.16 está representado o modo como o caminho é definido.

```
1
2 path certificate "/etc/racoon/certs";
```

Código 4.16: 3º modo de configuração: definir o caminho para a pasta onde estão os certificados

O modo de autenticação neste caso é diferente do apresentado no 2º modo configuração, como tal a secção `remote` irá também possuir algumas diferenças.

Comparativamente ao exemplo apresentado no 2º modo, é possível observar que agora é utilizado o `rsasig` como método de autenticação indicando que são necessários um par de chaves pública/privada RSA. Para a autenticação ser efectuada com sucesso o *host* precisa de possuir o seu par de chaves pública/privada, neste caso `laptop.public` e `laptop.private`, e a chave pública do *host* remoto, neste caso `server.public`. As configurações utilizadas estão representadas no Código 4.17.

```
1
2 remote 2001:690:2280:20:216:cbff:feac:5b2f
3 {
4         exchange_mode main,aggressive;
5         my_identifier asn1dn;
6         peers_identifier asn1dn;
7
8         certificate_type x509 "laptop.public" "laptop.private";
9
10        peers_certfile x509 "server.public";
11        proposal {
12                encryption_algorithm 3des;
13                hash_algorithm sha1;
14                authentication_method rsasig;
15                dh_group 2 ;
16        }
17 }
```

Código 4.17: 3º modo de configuração: configuração do `racoon`, secção `remote`

Para terminar definem-se os parâmetros utilizados na segunda fase de negociação das SAs. Neste exemplo foram utilizadas as mesmas configurações apresentadas no 2º modo de configurações pois estas não têm influência no modo de configuração de chaves secretas automaticamente negociadas. As configurações utilizadas estão representadas no Código 4.18.

```
1
2 sainfo anonymous
3 {
4         pfs_group 1;
5         lifetime time 2min;
6         encryption_algorithm 3des ;
7         authentication_algorithm hmac_sha1;
8         compression_algorithm deflate ;
9 }
```

Código 4.18: 3º modo de configuração: configuração do `racoon`, secção `sainfo`

4.3 *Anycast* com IPSec

A utilização das tecnologias *Anycast* e IPSec em simultâneo é, segundo a bibliografia, desaconselhada. O facto do *Anycast* não ser um protocolo adequado ao TCP e dos seus endereços não identificarem um *host*, são alguns dos aspectos apontados para a impossibilidade da utilização de ambas as tecnologias em simultâneo. No entanto, apesar destes aspectos considerou-se importante verificar se de facto é possível o estabelecimento de um canal seguro, utilizando o IPSec, com um endereço *Anycast*.

Antes de passar à implementação, vários aspectos foram tomados em consideração de forma a verificar se nenhum pormenor de segurança é comprometido.

O primeiro aspecto a ter em consideração é o facto do modelo de comunicação *Anycast* não garantir que dois pacotes de um mesmo fluxo de informação sejam enviados para o mesmo *host*. Em termos de segurança, visto o *Anycast* identificar uma entidade/serviço este aspecto não é

comprometedor, pois para o cliente é indiferente não ser sempre o mesmo *host* a responder, desde que isso não afecte a QoS do serviço.

O segundo aspecto é o facto de um endereço *Anycast* não identificar um *host*, mas sim uma entidade/serviço, e como tal, ser desaconselhada a atribuição deste tipo de endereços a *hosts*. Este aspecto poderá ser comprometedor em termos de segurança, pois ao utilizarmos o IPSec é necessário o estabelecimento das SAs entre as duas extremidades de uma conexão, tipicamente dois *hosts*. Neste caso a conexão é entre um *host* e um serviço, como tal o estabelecimento das SAs será mais complexo de modo a que a segurança não seja comprometida.

O terceiro aspecto consiste no facto de não existir qualquer protocolo, reconhecido pelas entidades competentes, de acesso a um grupo *Anycast*. Este aspecto pode comprometer a segurança de um grupo, e consiste em mais um dos motivos que leva a que seja desaconselhada a utilização de endereços *Anycast* em *hosts*. No entanto este aspecto está fora do âmbito desta dissertação, e como tal, iremos partir sempre do pressuposto que os grupos *Anycast* estão seguros, ou seja, que nenhum *host* se juntou ao grupo de um modo malicioso.

Após analisados os possíveis problemas de segurança, e pressupostos a ter em conta, decidiu-se implementar as duas tecnologias, *Anycast* e IPSec, em simultâneo e analisar os resultados obtidos. Para a implementação destas tecnologias foram utilizadas os conhecimentos apresentados nas secções 4.1 e 4.2 desta dissertação. A implementação foi realizada utilizando o protocolo de *Internet* IPv6.

4.3.1 Arquitectura geral

Esta implementação possui dois servidores *Anycast* que fornecem o mesmo serviço, o qual irá ser requisitado por um cliente. Pelos motivos apresentados no capítulo 4.1.2, são utilizadas duas máquinas com o SO Mac OS X para os servidores, e para o cliente é utilizada uma máquina com o SO Ubuntu 10.10.

Para a realização dos testes serão utilizados os programas Cliente/Servidor HTTP utilizados na secção 4.1.3. Estes programas consistem num servidor HTTP que possui o objecto ”index.html”, e num programa cliente que faz 100 pedidos desse objecto. Estes programas foram desenvolvidos na linguagem de programação `JAVA`, e implementam o protocolo HTTP 1.1. Para cada pedido pelo objecto ”index.html” é criado uma conexão TCP entre o cliente e o servidor.

O endereço *Anycast* utilizado pelos servidores é o:

2001:690:2280:20:fdff:ffff:ffff:ff81 ,

o qual já tinha sido apresentado na secção 4.1.1 . Para configurar os servidores com este endereço foram utilizadas as configurações apresentadas no secção 4.1.2.

Na secção 4.2.3 são apresentados três modos de configurar o IPSec: 1º modo, configuração manual da SPD e SAD com chaves secretas partilhadas por ambos os nós, através do `setkey`; 2º modo, configuração manual da SPD através do `setkey` e geração automática das SAs pelo `racoon`, utilizando chaves secretas partilhadas por ambos os nós; 3º modo, configuração manual da SPD através do `setkey` e geração automática das SAs e das chaves secretas pelo `racoon`, utilizando para tal certificados de chave pública/privada. Assim sendo, nesta implementação são testados os três modos de configuração.

No primeiro e no segundo modo de configuração é utilizada uma chave secreta préviamente partilhada, como tal essa chave secreta deve ser igual em todos os servidores *Anycast*. No terceiro modo de configuração, são utilizados pares de chaves pública/privada, sendo que estas devem ser iguais em todos os servidores *Anycast*.

Os servidores *Anycast* fornecem um serviço, e do ponto de vista do cliente eles constituem uma única entidade. Como tal, as configurações IPSec do lado do cliente visam o estabelecimento do IPSec com o endereço *Anycast*. Assim sendo, o cliente deve possuir apenas uma configuração que deve ser utilizada independentemente do servidor com o qual se conecta.

Do lado do grupo *Anycast*, todos os seus membros devem possuir a mesma configuração IPSec, sendo assim possível ao cliente ligar-se a qualquer um dos membros do grupo.

4.3.2 Resultados obtidos

Neste capítulo são apresentados os resultados da utilização em simultâneo dos endereços *Anycast* com o IPSec. Para a configuração do IPSec são utilizados os três modos apresentados no capítulo 4.2.3.

4.3.2.1 1º Modo de configuração do IPSec

Inicialmente são atribuídos a ambos os servidores o endereço *Anycast*, e o IPSec é configurado tanto nos servidores como no cliente através do `setkey`. Neste modo de configuração as SA e as SP são manualmente inseridas na SAD e SPD respectivamente. As configurações utilizadas neste modo são iguais às apresentadas na secção 4.2.3.

Os dois servidores *Anycast* vão possuir configurações iguais, sendo assim possível um canal seguro entre o cliente e o serviço. Assim, independentemente de qual o servidor a responder, para o cliente o resultado será sempre o mesmo.

Após serem atribuídos endereços *Anycast* aos servidores e o IPSec ser configurado em todas as máquinas, o programa `servidor HTTP` é colocado a correr nos servidores, e no cliente é colocado a correr o programa `cliente HTTP`. Enquanto o cliente efectua os 100 pedidos pelo objecto `index.html`, é possível observar pela ferramenta de captura de tráfego `Wireshark` que os pacotes trocados entre o endereço do cliente e o endereço *Anycast* vão todos protegidos com um cabeçalho ESP. Assim sendo, com esta configuração é possível a utilização das comunicações *Anycast* e do IPSec em simultâneo.

4.3.2.2 2º e 3º Modo de configuração do IPSec

No 2º e 3º modo, ao contrário do 1º as SAs são negociadas pelo *deamon* `racoon`, e são inseridas na SAD por esta mesma ferramenta. Este pormenor destas configurações pode gerar problemas de confiança na negociação dos SAs impedindo a sua criação, pois enquanto que o cliente tenta criar uma sessão IPSec com o endereço *Anycast*, este pode responder com o seu endereço *Unicast*. No entanto, ambas as configurações foram testadas.

No 2º modo a negociação das SAs requer uma partilha prévia das chaves secretas entre as duas extremidades da conexão. Por seu lado, no 3º modo a negociação das SAs requer a negociação automática das chaves secretas entre as duas extremidades da conexão, utilizando pares de chaves pública/privada para se autenticarem.

Em ambos os modos de configuração, os dois servidores *Anycast* vão possuir configurações iguais, sendo assim possível um canal seguro entre o cliente e o serviço. Assim, independentemente de qual o servidor a responder, para o cliente o resultado será sempre o mesmo.

Ao implementar estes dois modos de configuração obtiveram-se os mesmos resultados. Após serem atribuídos endereços *Anycast* aos servidores e o IPSec ser configurado em todas as máquinas, os *deamons* `racoon` são colocados a correr, e o que se verifica é que os SAs não são negociados. Após consulta do mecanismo de *log* do `racoon`, verifica-se que as SAs não são negociadas pois a primeira fase de negociação, a fase de autenticação entre o cliente e o servidor, falha não sendo assim possível o estabelecimento de uma sessão IPsec entre o cliente e o servidor. Após a análise do tráfego e dos resultados obtidos, observou-se que quando o cliente despoletava a negociação de SAs, para iniciar a negociação primeiro o cliente e o servidor *Anycast* necessitam dos endereços *link-layer* de ambos, trocando para tal mensagens `Neighbor solicitation`. Estas mensagens são trocadas através do protocolo ICMPv6, visto o servidor não responder a estes pacotes com o endereço *Anycast* no campo origem, mas sim com o *Unicast*, é criado um problema de confiança resultando numa falha na fase de autenticação do IKE, não permitindo a criação das SAs.

4.4 Conclusão

Através dos testes realizados nesta secção, foi possível observar o comportamento das comunicações *Anycast* e do IPSec em situações reais. A solução proposta nesta dissertação tem em conta os resultados obtidos nestes testes.

Em termos de suporte dos SOs, foram testados os SOs Mac OS 10.6 e o Ubuntu 10.10 das famílias UNIX BSD e Linux, verificando-se que actualmente apenas o BSD possui suporte *Anycast*.

De modo a observar o comportamento das comunicações *Anycast* com o protocolo ICMPv6, foi atribuído a uma máquina com o SO Mac OS X 10.6 o endereço de rede *Anycast* já previamente mencionado 2001:690:2280:20:fdff:ffff:ffff:ff81. De seguida através de uma outra máquina, utilizando o comando `Ping6 2001:690:2280:20:fdff:ffff:ffff:ff81` da linha de comandos, foram enviados sucessivos pacotes ICMPv6 para o endereço *Anycast*. Ao observarmos as respostas verificou-se que o servidor *Anycast* responde sempre com o seu endereço *Unicast* no campo endereço de origem, neste caso 2001:690:2280:20:21c:b3ff:feb0:bcaa.

Para observar o comportamento das comunicações *Anycast* com TCP, foram criados dois programas de teste na linguagem de programação `JAVA`. Os programas consistem num cliente e num servidor. O servidor possui três objectos de tamanhos diferentes em memória, e está à escuta de conexões TCP no porto 5555. O programa cliente, cria um `socket` TCP com o par endereço *Anycast* e porto 5555, estabelecendo uma conexão TCP, e pede ao servidor os três objectos mencionados. Após receber os objectos termina a ligação. Observou-se com este teste, que é possível o estabelecimento de conexões TCP com endereços *Anycast*. Este comportamento pode dever-se à implementação da biblioteca de `sockets` TCP, pois esta não consegue distinguir um endereço de rede *Anycast* de um *Unicast*. Após a análise do tráfego TCP trocado entre o cliente e o servidor, através do programa de captura de tráfego `Wireshark`, os pacotes TCP enviados pelo servidor possuíam o endereço *Anycast* no campo endereço de origem do pacote, o que não era suposto. Este factor, aliado ainda ao facto do pacote possuir o endereço de âmbito local do servidor, permitiram que fosse possível a utilização do *Anycast* com o TCP dum modo correcto.

Após observar os tempos obtidos nos testes às implementações BGP e OSPF da arquitectura de rede, é possível observar a diferença de valores obtidos pelo BGP e OSPF, revelando-se este último consideravelmente mais rápido nas situações de falha. Os testes efectuados permitiram assim verificar que o *routing*, pode ter um papel importante nos tempos de reacção a falhas.

Após testar os dois modos de funcionamento, IKE através do `racoon` e configuração manual através do `setkey`, verificou-se que não é possível, sem alterações, a utilização do IKE com o *Anycast*. No entanto, manipulando a SAD e a SPD com o `setkey`, utilizando chaves

previamente partilhadas, verificou-se ser possível o estabelecimento de um canal IPSec entre o cliente e o servidor *Anycast*. Apesar deste modo funcionar com comunicações *Anycast*, é importante referir que este modo requer a troca prévia das chaves secretas entre as extremidades de uma conexão. Esta troca pode então introduzir alguns problemas de segurança dependendo do ambiente utilizado para a troca das chaves.

Capítulo 5

Solução proposta

Nesta secção é apresentada a solução desenvolvida, os pressupostos assumidos, e ainda pormenores relativos à implementação.

5.1 Descrição

Após implementadas as tecnologias IPSec e *Anycast*, concluíu-se que estas não funcionam em simultâneo utilizando o IKE, devido aos problemas surgidos na fase de autenticação. Através da configuração manual foi possível a utilização de ambas as tecnologias. No entanto este modo de configuração implica a troca prévia das chaves secretas, acrescentando aqui alguns problemas de segurança. Como tal, a solução proposta procura oferecer segurança às comunicações entre clientes e serviços *Anycast*, através do IPSec, resolvendo os problemas da troca das chaves secretas, possibilitando assim a comunicação segura entre eles.

O objectivo da solução desenvolvida, é proteger a interação entre um cliente e um serviço identificado por um endereço *Anycast*, através do IPSec. Funcionalmente a proposta está estruturada em 8 fases, que podem ser observadas na Figura 5.1. Correspondem a:

1 - O cliente escolhe os parâmetros IPSec pretendidos;

2 - São geradas as chaves secretas necessárias;

3 - O cliente descobre o endereço *Unicast* do servidor ”mais próximo”;

4 - O cliente estabelece uma conexão TCP com o endereço *Unicast* do servidor mais próximo;

5 - O cliente pede ao servidor a chave pública do grupo *Anycast*, através da conexão TCP;

6 - O cliente envia ao servidor os parâmetros IPSec pretendidos para a comunicação segura, e as chaves secretas cifradas com a chave pública do grupo *Anycast*;

7 - O cliente termina a conexão TCP com o servidor, e ambos criam as SPs e SAs e introduzem estas na SPD e SAD respectivamente;

8 - Por último o servidor partilha com os restantes membros do grupo *Anycast* a informação de segurança, de um modo seguro.

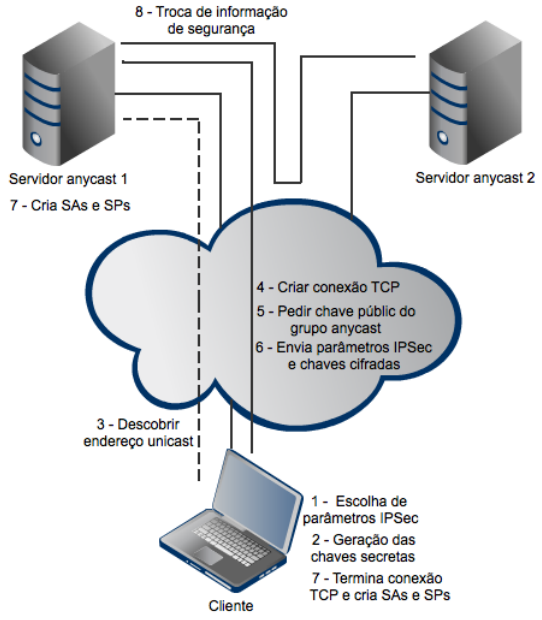

Figura 5.1: Interação do cliente com o servidor

Esta solução foi concretizada com dois programas (`cliente` e `servidor`), desenvolvidos na linguagem de programação `JAVA`. O programa `cliente` deve correr na máquina cliente e o programa `servidor` em todos os membros do grupo *Anycast*. Estes programas utilizam o modo de configuração manual do IPSec, mas as chaves são geradas automáticamente pelo programa `cliente` após o utilizador escolher os parâmetros de segurança, e trocadas de um modo seguro. Utilizando criptografia assimétrica e confiando na chave pública das entidades envolvidas, garante-se que a troca de chaves secretas é feita de forma segura. Após a negociação das chaves e troca segura destas, o cliente poderá ligar-se ao serviço *Anycast* utilizando o IPSec, independentemente do servidor para qual é encaminhado.

A metodologia utilizada no desenvolvimento desta solução tem em conta o estado da arte de ambas as tecnologias, não implicando nenhuma alteração às mesmas. Esta solução permite assim, a troca segura das chaves secretas por ambas as extremidades da conexão, o que constituía um dos principais problemas de segurança introduzido pela configuração manual do IPSec, oferecendo assim uma alternativa à utilização do IKE com o *Anycast*.

Na escolha dos parâmetros IPSec, é disponibilizado ao cliente um `GUI` onde este poderá escolher todos os parâmetros IPSec. O `GUI` está representado na Figura 5.2. De todo os parâmetros necessários para o estabelecimento da conexão IPSec, apenas as chaves secretas não são inseridas pelo cliente, pois estas são geradas pela aplicação.

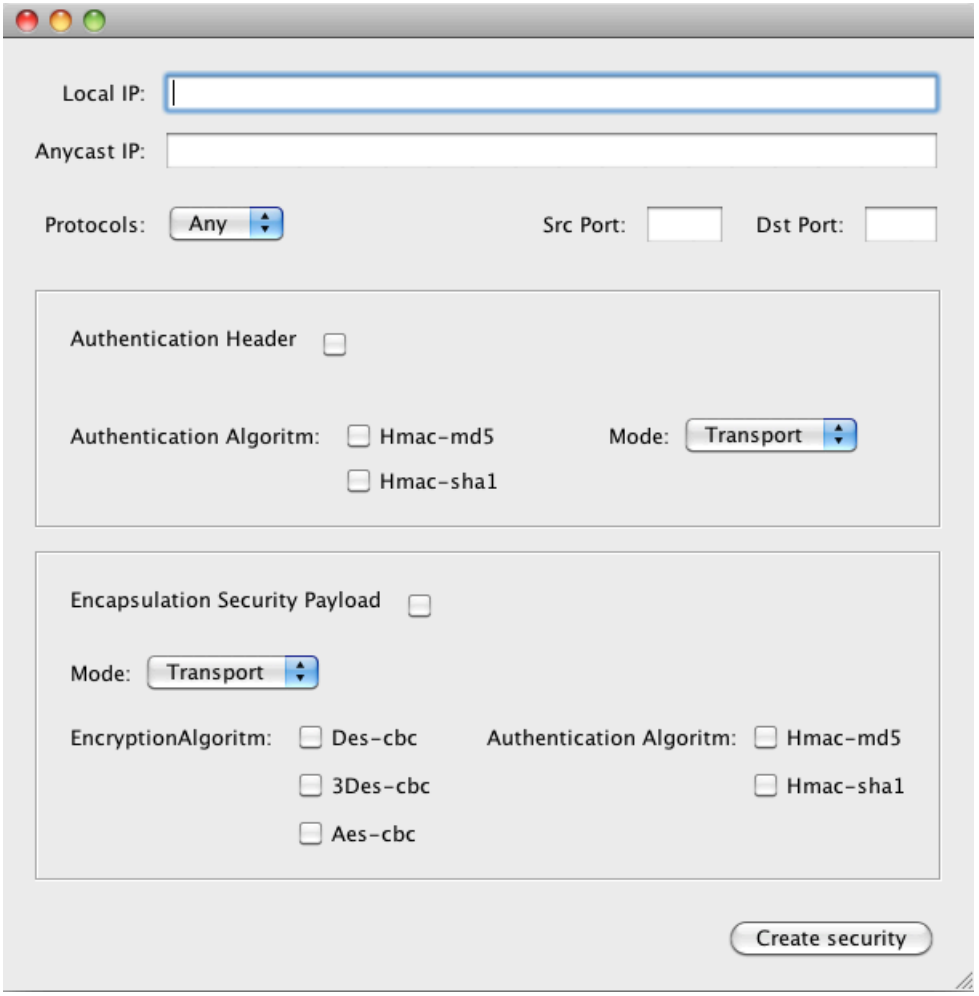

Figura 5.2: `GUI` para escolha dos parâmetros IPSec

5.2 Pressupostos

No desenvolvimento da solução proposta, foi necessário assumir alguns pressupostos e tomar determinadas decisões.

O primeiro pressuposto assumido visa identificar em que contexto é que a solução proposta deve ser utilizada. Ao utilizar a solução proposta o grupo *Anycast* deve identificar um serviço único, ou seja, todos os servidores devem possuir o mesmo comportamento e possuir a mesma informação. Caso isto não se verifique, tendo em conta que a decisão sobre para qual servidor o tráfego *Anycast* é encaminhado é efectuada pelo protocolo de *routing*, pode levar a resultados inesperados.

A existência de um protocolo de gestão de grupos *Anycast* é um dos pressupostos assumidos, existindo já, como foi referido neste artigo, várias propostas neste sentido. Assume-se ainda que o protocolo de gestão de grupos *Anycast* fornece sempre ao novo servidor que se junta ao grupo o par de chaves RSA pública/privada do grupo *Anycast*. Através deste requisito, e tendo em conta que toda a informação secreta é trocada através do grupo *Multicast*, apenas os servidores que tenham sido devidamente aceites pelo protocolo de gestão é que poderão decifrar a informação secreta trocada nas mensagens. Como é referido nesta dissertação, a atribuição de endereços *Anycast* a *hosts* é desaconselhada devido ao facto de não existir nenhum protocolo de gestão de grupos *Anycast* devidamente reconhecido e implementado. Ao assumir que é utilizado um protocolo de gestão do grupo *Anycast*, é resolvido o problema da atribuição do endereço *Anycast* a *hosts*.

A utilização dos endereços *Anycast* visam identificar um serviço, ou um conjunto de máquinas com um mesmo comportamento. Como tal, ao utilizar o *Anycast* pretende-se oferecer redundância em termos da máquina para a qual são enviados pacotes *Anycast*. Assim sendo, do ponto de vista do cliente, é irrelevante saber qual a máquina para a qual os seus pacote *Anycast* são enviados. Como é referido nesta dissertação, a utilização do endereço *Anycast* como endereço de origem é desaconselhada visto não identificar uma única máquina. No entanto, o endereço *Anycast* identifica um grupo, como tal o cliente não precisa de identificar o servidor com quem está a interagir pois apenas necessita de saber que está a interagir com o serviço identificado pelo endereço *Anycast*.

5.3 Implementação

Tal como é referido na descrição da solução, e demonstrado com a Figura 5.1, a solução está estruturada em 8 fases. Nesta secção, as 8 fases mencionadas são descritas ao pormenor.

Na Figura 5.3 estão representadas as oito fases através de um diagrama de fluxo. Pretende-se com este diagrama demonstrar qual a sequência de execução da aplicação proposta.

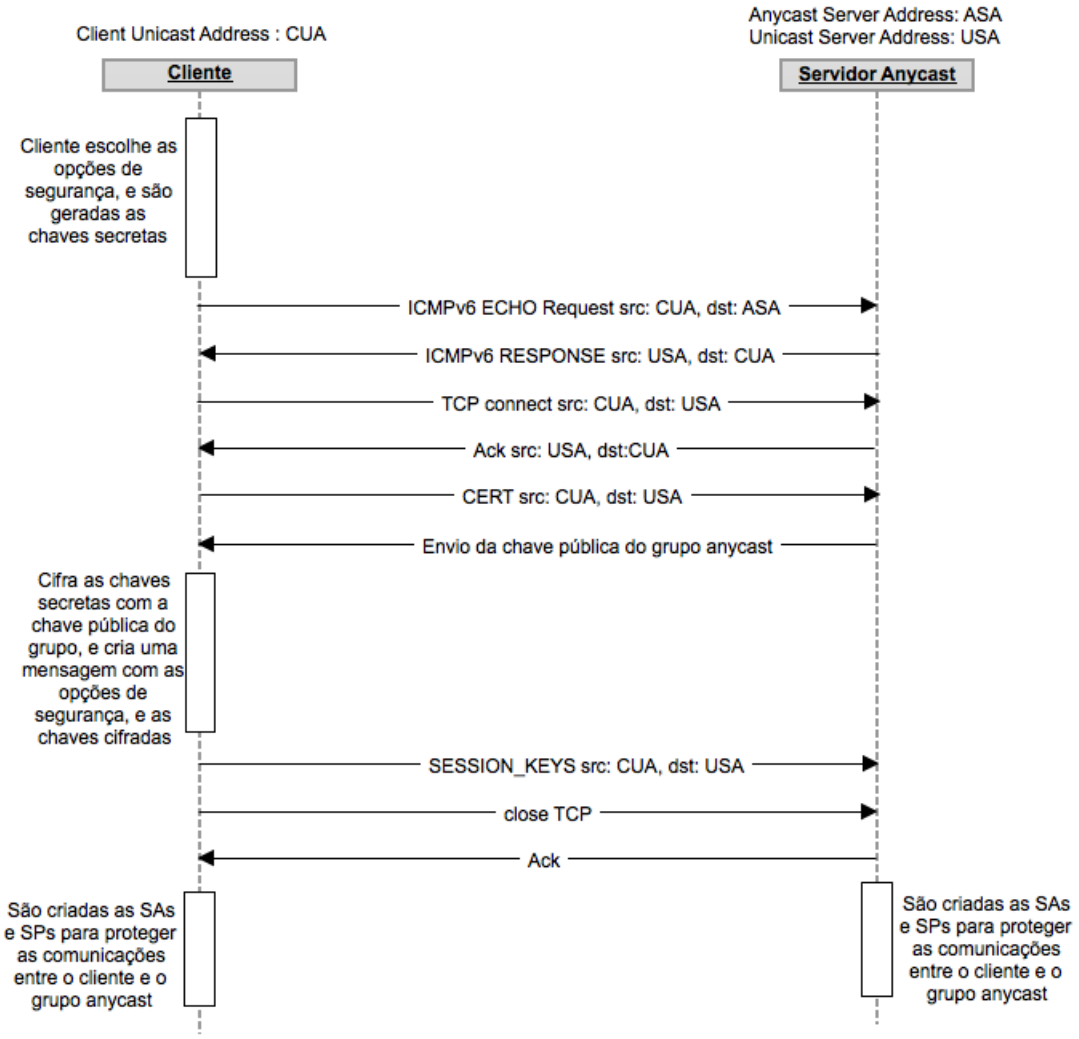

Figura 5.3: Fluxograma da interação detalhada do cliente com o servidor

5.3.1 1ª Fase - Escolha dos parâmetros de configuração do IPSec

A primeira fase consiste na escolha dos parâmetros IPSec pretendidos para a comunicação protegida. Para o efeito, é disponibilizado ao cliente um `GUI` onde este poderá escolher os seguintes

parâmetros:

- **Modo de operação** - Existem dois modos de operação: `transport` e `tunnel`. Nesta versão da solução, apenas é possível a utilização do modo `transport`.
- **Modo de funcionamento** -Existem dois modos de funcionamento: AH e ESP. Através da solução é possível a utilização de ambos os modos em separado ou com os dois em simultâneo.
- **Algoritmos de autenticação** - Os algoritmos de autenticação disponibilizados pela solução são: o Hmac-md5 e Hmac-sha1. Estes algoritmos podem ser utilizados tanto com o modo de funcionamento AH como ESP.
- **Algoritmos de cifragem** - Os algoritmos de cifragem disponilizados pela solução são: o DES, o 3DES e o AES. Estes algoritmos podem apenas ser utilizados com o modo de funcionamento ESP.
- **Tráfego que pretende proteger** - Através da solução é possível aplicar as configurações IPSec a: todo o tráfego (`any`), ao tráfego TCP, ao UPD e ainda ao ICMPv6.
- **Portos** - É ainda possível configurar os portos, tanto de origem como destino, aos quais pretende aplicar estes filtros IPSec.

Relativamente à informação necessária para a criação das SAs e SPs, apenas as chaves secretas não são inseridas pelo cliente, pois estas são geradas pelo programa cliente. O `GUI` para escolha de parâmetros IPSec está representado na Figura 5.2.

5.3.2 2ª Fase - Geração das chaves secretas

Após a escolha dos parâmetros, são geradas as chaves secretas por parte do programa `cliente`. As chaves geradas dependem dos protocolos de autenticação e cifragem escolhidos. Todas as chaves são geradas do mesmo modo, sendo que para cada uma delas é necessário introduzir o tamanho da chave em *bits*. No código 5.1 é demonstrado como se gera uma chave para o algoritmo DES.

```
1 public SecretKey genDESSecretKey(int bits) throws Exception
2 {
3     KeyGenerator kg = KeyGenerator.getInstance("DES");
4     kg.init(bits);
5     SecretKey key = kg.generateKey();
6
7     return key;
8 }
```

Código 5.1: Função para gerar uma chave do algoritmo DES

Inicialmente é criado um objecto da classe `KeyGenerator`, utilizando para tal o método `getInstance()`. É através do método `getInstance()` que é definido o tido de algoritmo pretendido, sendo que este é passado como parâmetro neste método. De seguida inicializa o objecto `KeyGenerator` através do método `init()`, passando como parâmetro o tamanho de chave em *bits*. Para terminar é criada a chave secreta através do método `generateKey()`. Este método retorna um objecto do tipo `SecretKey`, com a chave secreta.

5.3.3 3ª Fase - Descoberta do endereço *Unicast* do servidor *Anycast* mais próximo

A terceira fase consiste na descoberta do endereço *Unicast* do servidor *Anycast* "mais próximo". Como foi observado noutros capítulos deste artigo, os servidores *Anycast* respondem a pacotes ICMPv6 com o seu endereço *Unicast* no campo endereço de origem. Assim, de modo a obter o endereço *Unicast* do servidor é enviado para o endereço *Anycast* um pacote `ICMPv6 ECHO REQUEST`. Ao receber a resposta do servidor, o programa cliente retira o endereço *Unicast* do servidor da resposta ao pacote ICMPv6. A abordagem utilizada é semelhante à apresentada em [30] [31]. Na Figura 5.4 está representada esta interacção.

Devido ao facto do `JAVA` não implementar, nativamente, o protocolo ICMPv6, foi necessário recorrer à biblioteca `RockSaw` [45]. `Rocksaw` é uma API para a realização de *network* I/O com IPv4 e IPv6 `raw sockets` em `JAVA`.

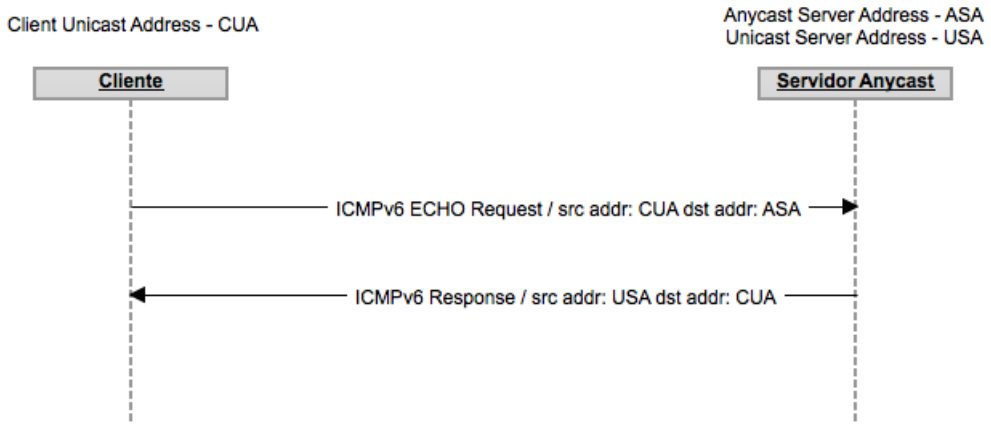

Figura 5.4: Descoberta do endereço *Unicast*

5.3.4 4ª Fase - Estabelecimento de uma conexão TCP de controlo

De seguida é criada uma conexão TCP com o endereço *Unicast* do servidor que respondeu ao pacote ICMPv6. Toda a troca de mensagens/informação através da conexão TCP é efectuada através de um formato de mensagem desenvolvido para estes programas, utilizando para o efeito objectos serializáveis. O objecto utilizado nas mensagens é do tipo `ProtocolObject`, e a sua estrutura está representada na Figura 5.5.

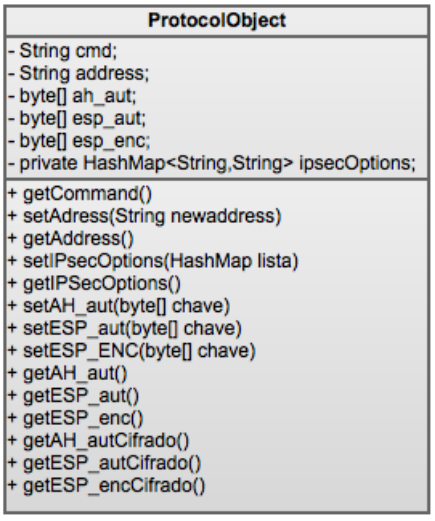

Figura 5.5: Estrutura do objecto utilizado na troca de informação

Este objecto é utilizado para a troca de mensagens não apenas nas interações Cliente/Servidor, mas também entre Servidores. Como tal, é utilizado tanto no tráfego *Unicast* como *Multicast*.

Existem Quatro tipos de mensagens para as interações cliente/servidor e servidor/servidor:

- `CERT` - Este tipo de mensagem é utilizado nas comunicações cliente/servidor através do protocolo TCP, para o cliente pedir a chave pública do grupo *Anycast* ao servidor.
- `SESSION_KEYS` - Este tipo de mensagem é utilizado nas comunicações cliente/servidor através do protocolo TCP, e é utilizado para o cliente enviar ao servidor os parâmetros pretendidos para a proteção IPSec, juntamente com as chaves secretas cifradas com a chave pública do grupo *Anycast*.
- `NEW_CLIENT` - Este tipo de mensagem é utilizado nas comunicações entre servidores *Anycast* através do grupo *Multicast*. Este tipo de mensagem tem como objectivo notificar todos os elementos do grupo *Anycast* que um novo cliente possui configurações IPSec com o grupo *Anycast*. Nesta mensagem são enviados os parâmetros pretendidos para a proteção IPSec, juntamente com as chaves secretas cifradas com a chave pública do grupo *Anycast*.
- `NEW_SERVER` - Este tipo de mensagem é utilizado nas comunicações entre servidores *Anycast* através do protocolo TCP. Quando um novo servidor se junta ao grupo *Multicast*, após aceite pelo protocolo de gestão do grupo *Anycast*, conecta-se por TCP a um dos servidores do grupo. De seguida envia uma mensagem do tipo `NEW_SERVER` de modo a pedir informações acerca de todos os clientes ligados ao grupo *Anycast*.

O tipo de mensagem é identificado no objecto `ProtocolObject` através do campo `cmd`.

5.3.5 5ª Fase - Obtenção da chave pública do grupo

Depois de criada a conexão TCP, o cliente pede ao servidor a chave pública RSA do grupo *Anycast*. Para o pedido da chave pública, o cliente envia ao servidor uma mensagem do tipo `CERT`. A certificação da chave pública consiste num problema de segurança bem identificado, mas fora do âmbito deste trabalho.

5.3.6 6ª Fase - Envio de parâmetros IPSec

Após receber a chave, o cliente envia para o servidor os parâmetros IPSec préviamente escolhidos, e as chaves secretas geradas pelo programa cliente cifradas com a chave pública do grupo *Anycast*. Para enviar esta informação o cliente envia ao servidor uma mensagem do tipo `SESSION_KEYS`. A cifragem das chaves secretas é efectuada com o auxílio da biblioteca de criptográfica *Bouncy Castle* [46]. A função utilizada para decifrar pode ser observada no código 5.2.

```
1 public byte[] cifra(String plainText) throws NoSuchAlgorithmException,
2                                       InvalidKeyException, Exception {
3
4     Security.addProvider(new org.bouncycastle.jce.provider.BouncyCastleProvider());
5     Cipher cipher = Cipher.getInstance("RSA/None/NoPadding", "BC");
6     cipher.init(Cipher.ENCRYPT_MODE, this.sec.loadServerPubKey());
7
8     return cipher.doFinal(plainText.getBytes());
9 }
```

Código 5.2: Função para cifrar informação com o algoritmo RSA

Primeiro é definido o *provider*, sendo neste caso o *Bouncy Castle*. De seguida é criado um objecto do tipo `Cipher`. Este objecto é criado com o método `getInstance()`, passando como parâmetros o algoritmo a usar, e o *provider* (neste caso BC, *Bouncy Castle*). De seguida inicializa-se o objecto do tipo `Cipher` através do método `init()`, passando como parâmetro o modo de funcionamento (neste caso `ENCRYPT_MODE`, ou seja, modo de cifragem), e a chave RSA pública utilizada para cifrar. A chave pública é um objecto do tipo `RSAPublicKey` e é retornada pela função `loadServerPubKey()`.

5.3.7 7ª Fase - Criação das SAs e SPs

Quando o servidor recebe esta informação por parte do cliente, decifra as chaves secretas e juntamente com a restante informação acerca dos parâmetros IPSec possui assim toda a informação

necessária para criar as SAs e SPs. De seguida a conexão TCP entre o cliente e o servidor termina, e são criadas as SAs e as SPs e inseridas na SAD e SPD respectivamente, com o auxílio da ferramenta `setkey`. A decifragem das chaves secretas é efectuada com o auxílio da biblioteca criptográfica *Bouncy Castle*. A função utilizada para decifrar pode ser observada no código 5.3.

```
1  public byte[] decifra(String cifrado) throws NoSuchAlgorithmException,
2                              NoSuchProviderException, NoSuchPaddingException,
3                              InvalidKeyException, Exception {
4
5     Security.addProvider(new org.bouncycastle.jce.provider.BouncyCastleProvider());
6     Cipher cipher = Cipher.getInstance("RSA/None/NoPadding", "BC");
7     cipher.init(Cipher.DECRYPT_MODE, loadServerPrivKey());
8
9     return cipher.doFinal(cifrado.getBytes());
10 }
```

Código 5.3: Função para decifrar informação com o algoritmo RSA

Primeiro é definido o *provider*, sendo neste caso o *Bouncy Castle*. Depois é criado um objecto do tipo `Cipher`. Este objecto é criado com o método `getInstance()`, passando como parâmetros o algoritmo a usar, e o *provider* (neste caso BC, *Bouncy Castle*). De seguida inicializa-se o objecto do tipo `Cipher` através do método `init()`, passando como parâmetro o modo de funcionamento (neste caso `DECRYPT_MODE`, ou seja, modo de decifragem), e a chave RSA privada utilizada para cifrar. A chave pública é um objecto do tipo `RSAPrivateKey` e é retornada pela função `loadServerPrivKey()`.

5.3.8 8ª Fase - Partilha da informação no grupo *Anycast*

Para terminar, o servidor envia por *Multicast* a informação de segurança, respectiva a este cliente, para todos os elementos do grupo *Anycast*. Na Figura 5.6 está representada esta interação através de um fluxograma. A informação de segurança, tal como se verifica na comunicação entre o cliente e o servidor, é enviada cifrada com a chave pública do grupo *Anycast*, garantindo assim a sua confidencialidade. É ainda importante referir que esta informação é trocada pelos servidores

Anycast, para em caso de falha qualquer servidor do grupo poder atender o cliente de um modo seguro.

O grupo *Multicast* tem como objectivo permitir aos servidores partilharem a informação sobre os clientes de um modo mais eficiente. Toda a informação secreta transmitida através do grupo *Multicast* é cifrada com a chave pública do grupo. Esta decisão foi tomada para que apenas os servidores do grupo *Anycast* possam ter acesso à informação secreta. Assim, mesmo que entidades maliciosas capturem pacotes do grupo *Multicast*, não irão conseguir extrair daí a informação secreta.

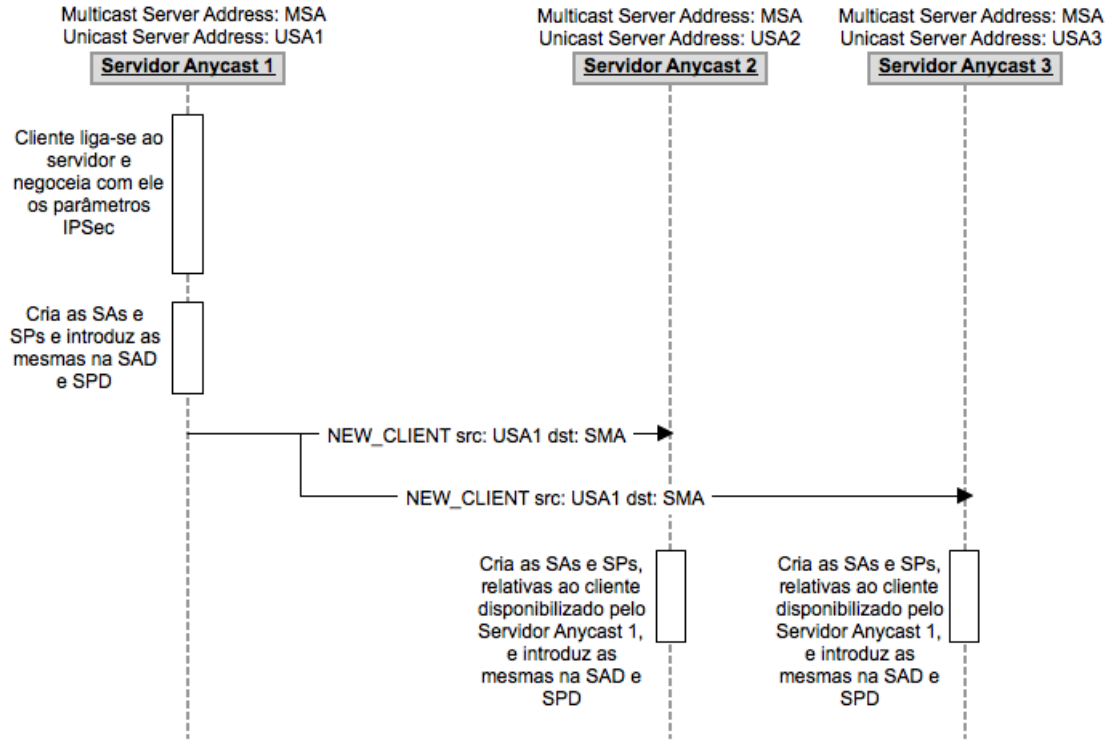

Figura 5.6: Fluxograma da interação detalhada do cliente com o servidor

5.3.9 Novo servidor *Anycast*

Quando um novo servidor se junta ao grupo *Anycast*, tal como é referido nos pressupostos da solução, este recebe um conjunto de informação que lhe vai permitir interagir com os restantes elementos do grupo. A informação que o novo servidor recebe é: par de chaves RSA pública/privada do grupo; e uma lista dos endereços *Unicast* dos servidores do grupo *Anycast*. O

servidor após receber esta lista, estabelece uma conexão TCP com um dos servidores da lista (o primeiro da lista) e pede a este a lista de clientes e a informação de segurança de cada um deles. Este pedido é efectuado através de uma mensagem `NEW_SERVER`. O servidor que recebe esta mensagem, primeiro envia ao novo servidor o número de clientes, de seguida envia uma mensagem do tipo `NEW_CLIENT` com os parâmetros IPSec e as chaves secretas, cifradas com a chave pública do grupo *Anycast*, por cada um dos clientes. O novo servidor após receber o número de clientes sabe quantas mensagens do tipo `NEW_CLIENT` vai receber. À medida que o novo servidor recebe as mensagens, vai criando as respectivas SAs e SPs de cada cliente e insere-as na SAD e SPD respectivamente. Após todo este processo, o cliente está pronto para receber pedidos de todos os clientes.

No fluxograma apresentado na Figura 5.7, está representado o processo descrito nesta secção através de um exemplo em que o grupo *Anycast* possui 3 clientes.

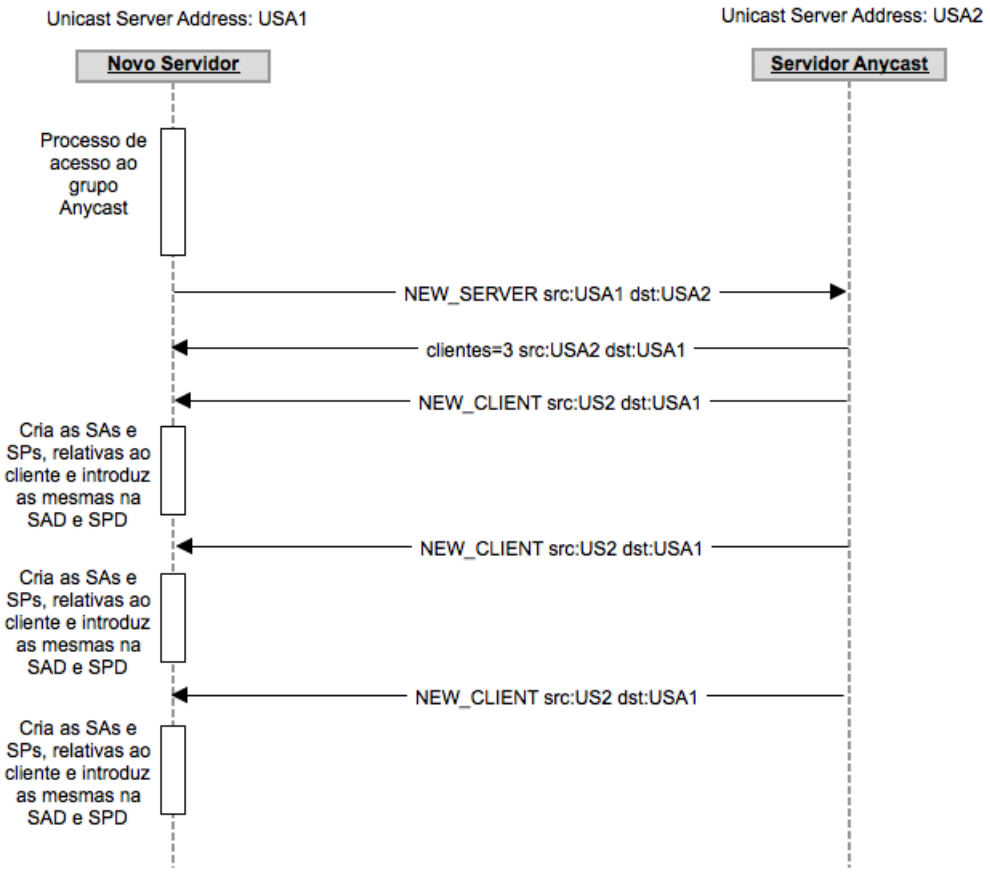

Figura 5.7: Processo efectuado quando um novo servidor se junta ao grupo *Anycast*

5.4 Testes à solução

Nesta secção são explicados em detalhe os testes efectuados à solução desenvolvida e apresentados os resultados obtidos nos mesmos.

5.4.1 Cenário de testes

Para a realização dos testes, foram utilizadas três máquinas. Uma máquina com o SO Ubuntu utilizado para implementar um cliente, e duas máquinas com o SO Mac OS X para implementar dois servidores. As três máquinas estão ligadas à rede 2001:690:2280:20::/64, e todas à mesma distância umas das outras em termos de métricas de rede. Os servidores *Anycast* possuem o endereço de rede 2001:690:2280:20:fdff:ffff:ffff:ff81/128.

Ao cliente foi atribuído o endereço de rede *Unicast* 2001:690:2280:20:208:54ff:fe0e:74af. Aos servidores 1 e 2 para além dos endereços *Anycast*, foram também atribuídos a estes os endereços *Unicast* 2001:690:2280:20:216:cbff:feac:5b2f/64 e 2001:690:2280:20:62fb:42ff:feee:6f0e/64, respectivamente.

As decisões acerca dos SOs e do endereço *Anycast* a utilizar, foram tomadas em conformidade com a informação apresentada no Capítulo 4.

5.4.2 Testes e resultados obtidos

De modo a testar o funcionamento da solução e demonstrar a utilidade desta ferramenta, foram efectuados os seguintes testes:

1 - Primeiro iniciam-se os dois servidores *Anycast*, colocando o programa `servidor`, da solução desenvolvida, e o programa `servidor HTTP`, previamente apresentado no Capítulo 4, a correr. Após os servidores estarem preparados, coloca-se a correr o programa `cliente` da solução. Após preenchido o formulário disponibilizado no GUI do programa `cliente` e da protecção IPSec estar criada, coloca-se a correr o programa `cliente HTTP`, previamente apresentado no Capítulo 4. De seguida observa-se o comportamento das três máquinas.

2 - Neste teste, todo o procedimento do teste 1 é repetido com apenas uma alteração na fase final do teste. Após o programa `cliente HTTP` estar a correr, é gerada uma falha no servidor *Anycast* que está a responder aos pedidos. Visto haver dois servidores *Anycast*, o cliente será reencaminhado para o outro servidor. De seguida observa-se o comportamento das três máquinas.

5.4.2.1 Teste 1 - Comportamento em situação normal

Para efectuar este teste, o formulário do `GUI` foi preenchido de modo às comunicações entre o grupo *Anycast* e o cliente serem protegidas com os seguintes parâmetros: (1) AH no modo `transport`, com o algoritmo de autenticação HMAC-MD5; (2) e ESP no modo `transport`, com o algoritmo de encriptação 3DES-CBC. Na Figura 5.8 é possível observar como preencher o formulário do `GUI` para obter estas configurações.

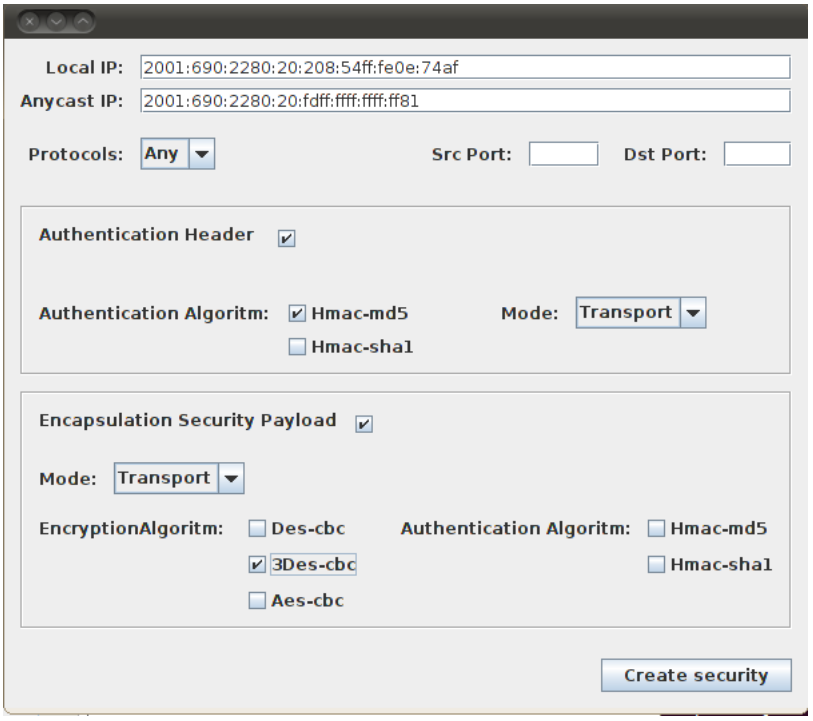

Figura 5.8: Teste 1, formulário do `GUI` do programa `cliente` preenchido

Ao implementar este teste verificou-se que o primeiro servidor *Anycast* a responder foi o servidor 1. Após o cliente negociar com este os parâmetros IPSec, ambos criaram as respectivas SAs e SPs e introduziram as mesmas na SAD e SPD respectivamente. Através dos comandos

previamente apresentados no Capítulo 4, foi possível observar a SAD e SPD do servidor e do cliente. As Figuras 5.10, 5.9, 5.12 e 5.11 representam as SADs e as SPDs do cliente e do servidor 1, respectivamente. Como é possível observar pelas figuras, as informações de segurança foram trocadas com sucesso, e as SAs e SPs foram inseridas, tanto no cliente como no servidor 1, dum modo correcto possuindo ambas as máquinas as chaves secretas e as configurações necessárias para proteger as suas comunicações.

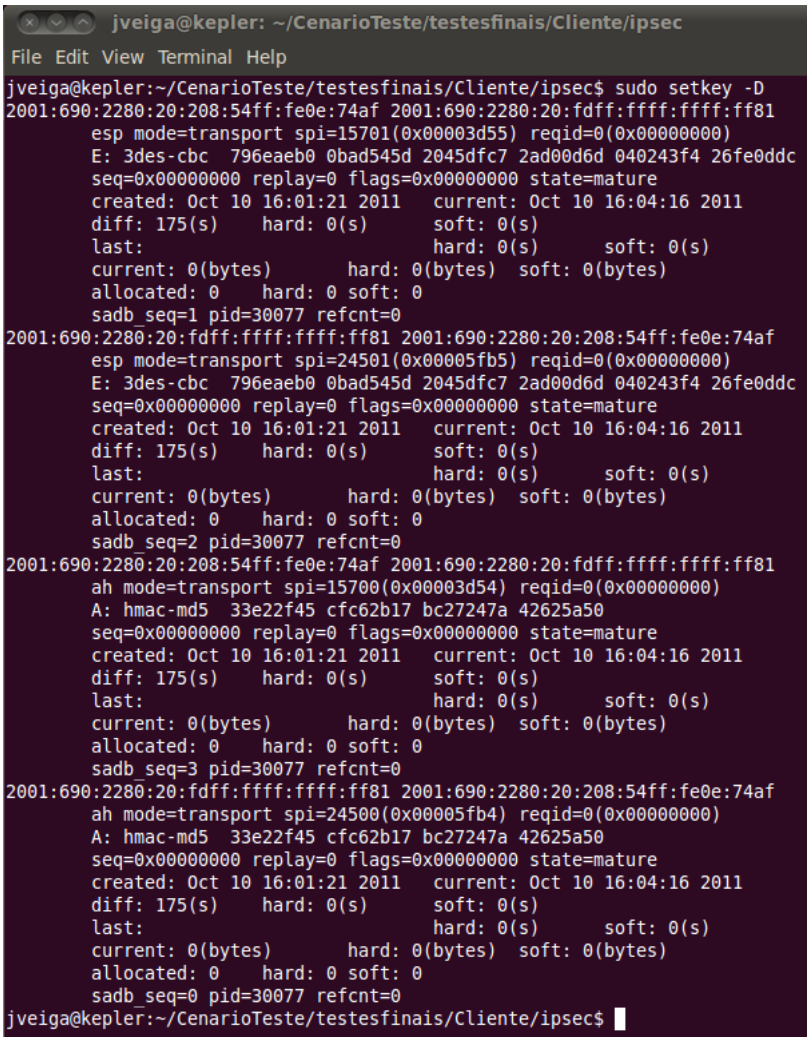

Figura 5.9: Teste 1, SAD do cliente

De seguida, visto o servidor 1 transmitir aos restantes elementos do grupo *Anycast* as informações de segurança do novo cliente, observou-se a SAD e SPD do servidor 2 de modo a verificar se este possui as configurações IPSec para proteger as comunicações com o cliente. Após consultar a SAD e a SPD, foi possível observar que este já possuía as SAs e SPs necessárias para proteger as comunicações com o cliente. Nas Figuras 5.13 e 5.14 estão representadas

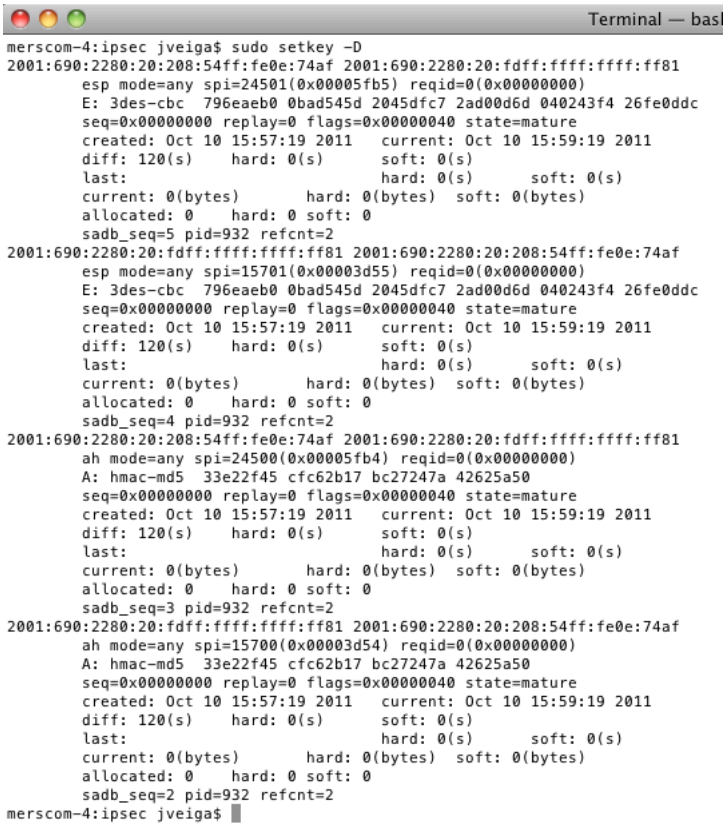

Figura 5.10: Teste 1, SAD do servidor 1

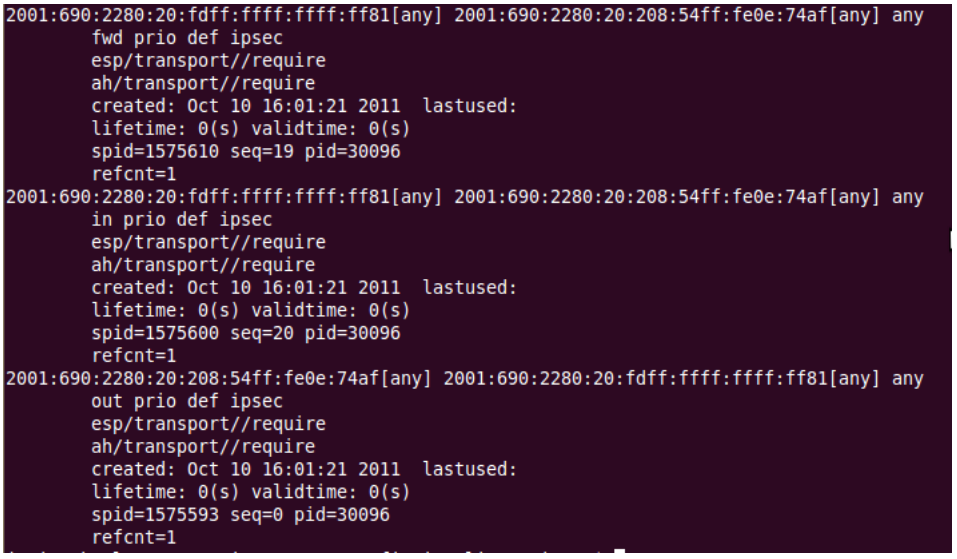

Figura 5.11: Teste 1, SPD do cliente

a SPD e a SAD do servidor 2. Como é possível observar através destas figuras, o servidor possui configurações de segurança semelhantes ao servidor 1, tal como era esperado.

Após verificar que todos os servidores *Anycast*, neste caso o 1 e 2, possuem a informação

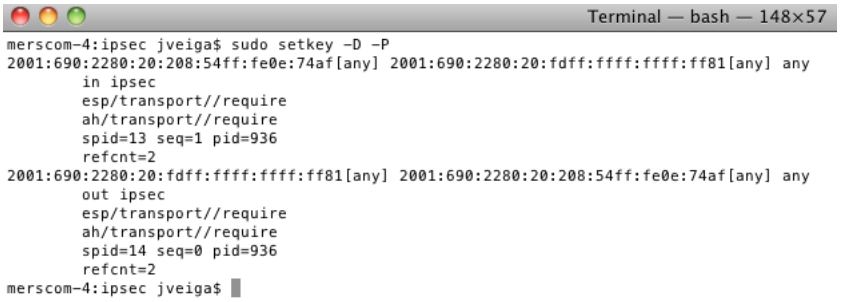

Figura 5.12: Teste 1, SPD do servidor 1

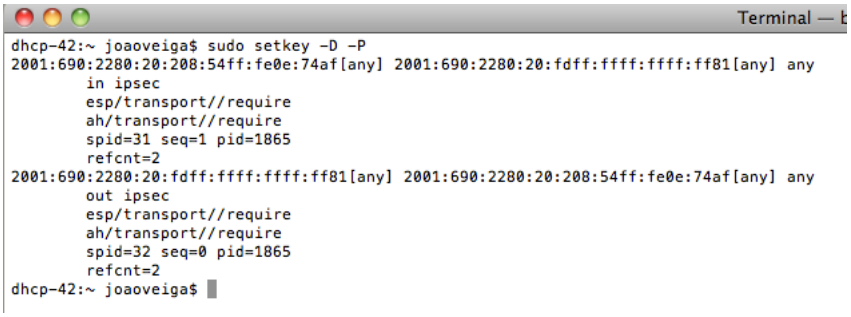

Figura 5.13: Teste 1, SPD do servidor 2

de segurança necessária para proteger as comunicações com o cliente, coloca-se o programa `cliente HTTP` a correr. Depois de colocar o programa a correr, observou-se que foi o servidor 1 a responder aos pedidos. De seguida, e enquanto que o servidor respondia aos pedidos, observou-se o tráfego trocado entre o servidor 1 e o cliente, através do `Wireshark` observando-se que todos os pacotes em ambos os sentidos eram protegidos pelo IPSec. Na Figura 5.15 é apresentado um *printscreen* do tráfego trocado entre o cliente e o servidor 1, observado através do `Wireshark`, demonstrando que todo o tráfego está protegido com IPSec.

5.4.2.2 Teste 2 - Comportamento em situação de falha

Pretende-se com este teste, demonstrar que através da solução desenvolvida é possível um serviço *Anycast*, suportado por múltiplos servidores, proteger as comunicações com os clientes independentemente do servidor que atende o cliente.

O procedimento inicial deste teste é semelhante ao efectuado no teste 1, como tal foram utilizadas as mesmas opções IPSec, e os servidores estão a correr os mesmos programas. Após o cliente negociar com um dos servidores *Anycast* os parâmetros de segurança e este os divulgar,

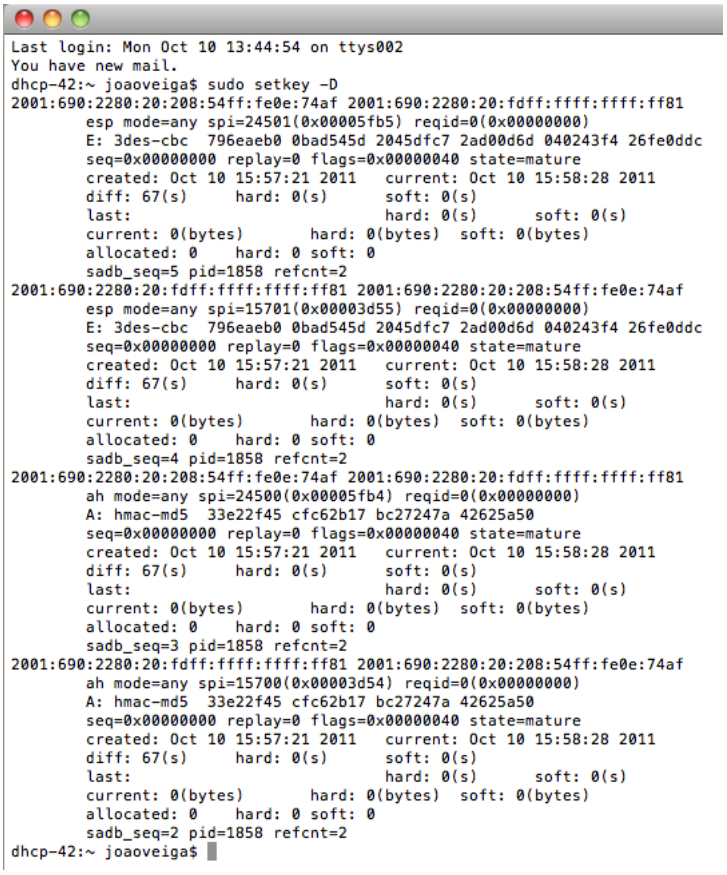

Figura 5.14: Teste 1, SAD do servidor 2

de um modo seguro, para os restantes membros do grupo *Anycast*, é colocado a correr na máquina cliente o programa `cliente HTTP`. O programa `cliente HTTP` foi alterado para a realização deste teste, em vez de cem efectua mil pedidos pelo objecto `index.html`. Esta alteração foi efectuada de modo a levar mais tempo a efectuar a totalidade dos pedidos, e assim permitir gerar a falha no servidor que está a responder aos pedidos. A falha no servidor é causada através da interrupção da ligação do servidor à rede.

Ao colocar o programa `cliente HTTP` observou-se que o servidor 1 estava a responder aos pedidos, como tal gerou-se a falha nesse servidor. Após a falha, os pedidos passaram a ser encaminhados para o servidor 2. De seguida, e enquanto o servidor estava a responder aos pedidos, observou-se através do `Wireshark` que o tráfego continuava a ser protegido pelo IPSec. O programa `cliente HTTP` terminou com sucesso, sem a ocorrência de nenhum problema. Na Figura 5.16 é apresentado um *printscreen* do tráfego trocado entre o cliente e o servidor 2, observado através do `Wireshark`, demonstrando que todo o foi reencaminhado para o servidor 2 e que este está protegido com IPSec.

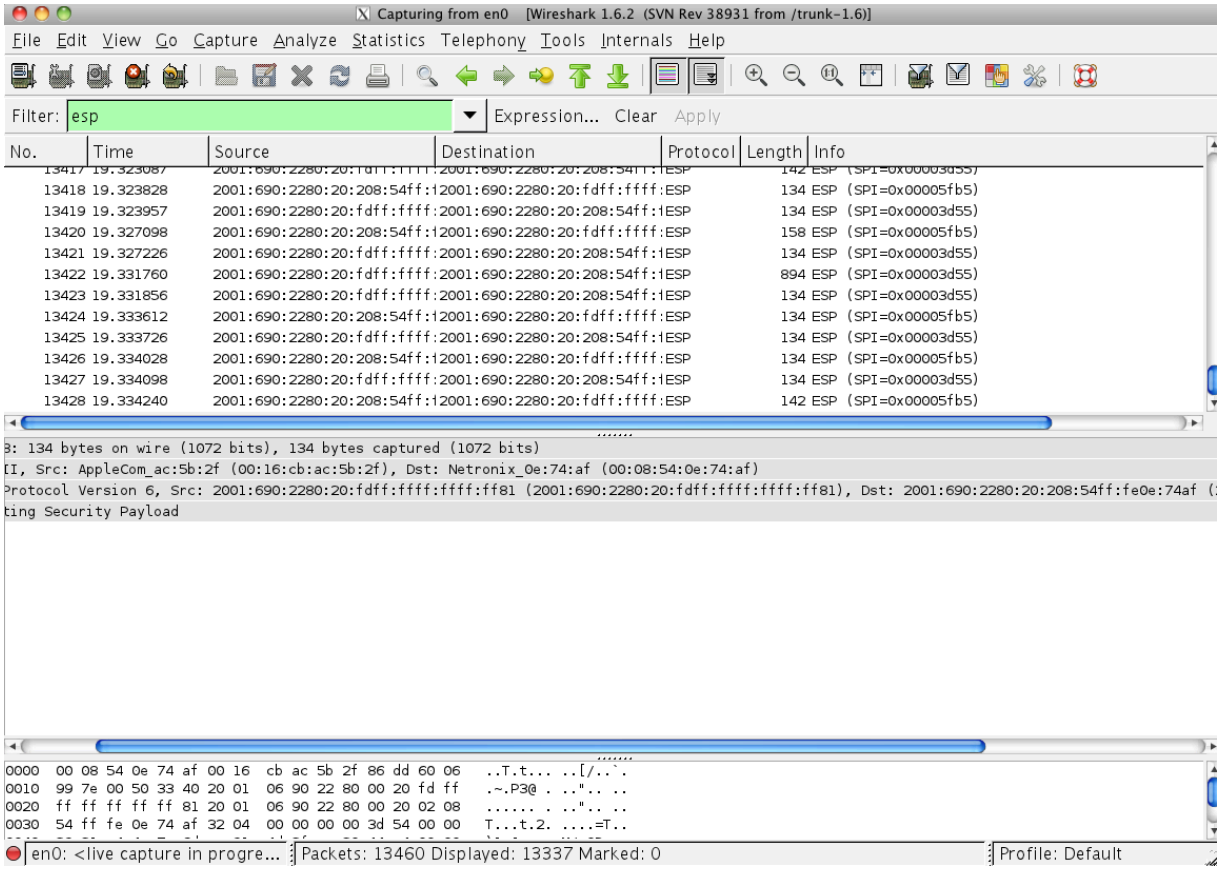

Figura 5.15: Teste 1, Captura de tráfego entre o cliente e o servidor 1

5.5 Conclusão

A solução proposta tem como objectivo permitir a um cliente negociar os pormenores de segurança IPSec com um serviço *Anycast* de um modo seguro. De modo a facilitar a utilização da solução, é disponibilizado ao cliente um `GUI` para que este possa escolher, de um modo simples, os parâmetros que pretende para a conexão IPSec. O `GUI` está representado na Figura 5.2. A partir de fase da escolha dos parâmetros IPSec, todas as restantes fases operam sem que o cliente tenha conhecimento disso. Assim, exceptuando a fase de escolha dos parâmetros IPSec, a solução proposta para o estabelecimento da proteção IPSec opera como uma camada de abstração invisível ao utilizador.

Nesta dissertação comprova-se que é possível a utilização do IPSec com comunicações *Anycast* através da configuração manual do IPSec. No entanto, este modo de configuração, ao contrário do IKE, introduz alguns problemas de segurança. O facto do modo como as chaves secretas são trocadas não ser controlado, torna este modo de configuração muito vulnerável.

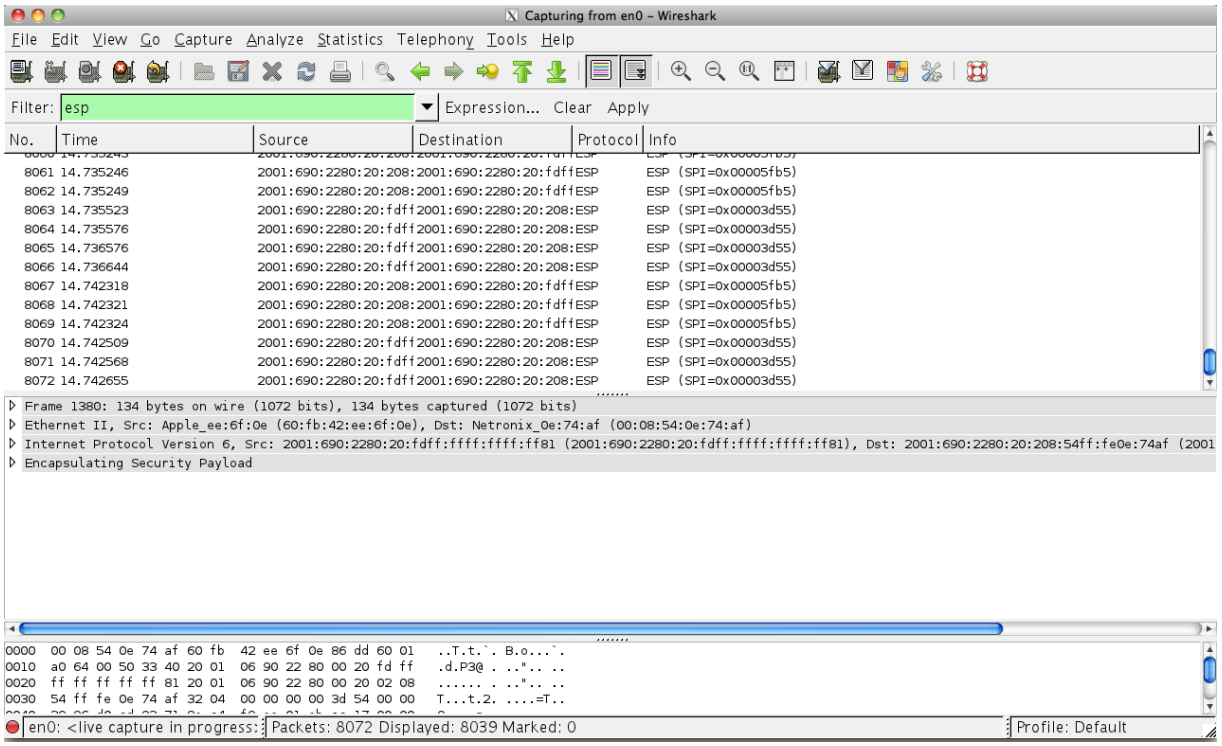

Figura 5.16: Teste 1, Captura de tráfego entre o cliente e o servidor 2

Pretende-se então com esta solução, tornar a utilização do modo de configuração manual do IPSec mais segura.

No desenvolvimento da solução, utilizou-se o IKE como um modelo a seguir para o desenvolvimento da solução. Como tal, a solução proposta, antes de criar e inserir as SAs e SPs na SAD e SPD respectivamente, executa um protocolo de negociação das chaves secretas necessárias e troca estas de um modo seguro. Após a negociação e troca das chaves secretas, as SAs e SPs são criadas e inseridas de seguida na SAD e SPD respectivamente, através do modo de configuração manual do IPSec utilizando para o efeito a ferramenta `setkey`.

A segurança na troca das chaves secretas, utilizada na solução, é conseguida através de criptografia assimétrica. Para tal, é utilizado um par de chaves pública/privada do algoritmo RSA para o grupo *Anycast*. Toda a informação secreta enviada entre o cliente e o servidor, ou entre servidores é cifrada com a chave pública do grupo *Anycast* garantindo assim a sua confidencialidade. No desenvolvimento da solução considerou-se a utilização do protocolo *Diffie-Hellman* com certificados para a troca de informação secreta. No entanto, optou-se por não se utilizar este protocolo por este necessitar que ambas as extremidades possuíssem um certificado. Embora o protocolo *Diffie-Hellman* possa ser utilizado sem certificados, esta solução iria introduzir

vulnerabilidades na segurança (p.ex. ataques por homem-no-meio).

Para além da interação com os clientes, os servidores interagem ainda uns com os outros de modo a manterem uma coerência nas informações de segurança dos clientes. Para o efeito, é utilizado um grupo *Multicast* ao qual todos os servidores do grupo de juntam. Sempre que um servidor pretende enviar informação para o grupo *Anycast*, e não para apenas um servidor em concreto, utiliza o grupo *Multicast*.

Após as 8 fases apresentadas serem concluídas com sucesso, o cliente pode ligar-se a qualquer um dos servidores *Anycast* que as suas comunicações serão protegidas pelo IPSec.

Ao testar a solução num cenário prático com um cliente e dois servidores, observou-se que a solução possui o comportamento esperado. Quando o cliente pretende proteger as suas comunicações com o serviço *Anycast*, ao utilizar a solução proposta toda a informação enviada para o endereço *Anycast* é protegida, tal como as suas respostas, independentemente do servidor que responde aos seus pacotes. Este comportamento foi observado, através de dois testes: (1) após a utilização do programa `cliente` da solução, colocou-se a correr o programa `cliente HTTP` e observou-se o comportamento; (2) após a utilização do programa `cliente` da solução, colocou-se a correr o programa `cliente HTTP` e enquanto este executa, gerou-se uma falha no servidor *Anycast* que estava a responder aos pedidos, observando o seu comportamento. O que se observou foi que a informação de segurança é partilhada de um modo seguro, e que independentemente do servidor que responde aos pedidos do cliente, as suas comunicações são sempre protegidas pelo IPSec.

Capítulo 6

Conclusão e trabalho futuro

Nesta secção são apresentadas as conclusões obtidas após o estudo efectuado às tecnologias *Anycast* e IPSec, aos testes efectuados às implementações destas tecnologias e ainda à solução proposta e implementada nesta dissertação.

É ainda apresentado um conjunto o trabalho futuro abordando questões suscitadas durante o desenvolvimento.

6.1 Conclusão

Existem nos dias de hoje diversas situações, no âmbito da *Internet*, em que clientes precisam de localizar conteúdos ou serviços prestados pela rede. Tipicamente, estes conteúdos ou serviços estão localizados num servidor e o seu acesso está limitado pela largura de banda da rede e pelos recursos do próprio servidor, podendo consequentemente levar a uma degradação do serviço prestado ao cliente. De modo a contornar estas limitações, tornou-se prática comum replicar os servidores que prestam os serviços, aumentando a disponibilidade dos serviços e ainda distribuindo a carga pelas réplicas. Uma abordagens utilizada para distribuir a carga pelos servidores é utilizar o modelo de comunicação *Anycast*. O *Anycast* é um novo modelo de comunicação nos standards do protocolo de *Internet* IPv6, tal como o *Unicast* e o *multicast*. Este novo modelo de comunicação caracteriza-se pelo facto da informação enviada para um endereço de rede *Anycast* ser recebida por um único nó de um conjunto de nós que partilham este mesmo endereço.

Existem actualmente diversos problemas de segurança muito frequentes na *Internet*. O modo de proteção habitual contra estes problemas consiste na utilização de técnicas criptográficas de chave secreta ou de chave pública para garantir confidencialidade, autenticação e integridade. O IPsec tem como objectivo proteger uma comunicação entre dois nós sejam eles *routers* ou *hosts*, como tal apenas o fluxo de dados trocados entre os dois nós que acordaram os pormenores de segurança é que será protegido. Por este motivo torna-se pouco apropriado a utilização do *Anycast* com o IPsec. O não determinismo na entrega dos pacotes ao servidor correcto põe em risto a segurança das comunicações.

Nesta dissertação, inicialmente, é efectuado um levantamento do estado da arte das tecnologias IPv6, IPSec, *Anycast* e ainda sobre balanceamento de carga. O protocolo de *Internet* IPv6 foi estudado devido ao facto dos estudos e testes efectuados nesta dissertação estarem inseridos no panorama IPv6, sendo inclusive a solução desenvolvida apenas operável em IPv6. O IPSec é uma tecnologia consolidada e amplamente utilizada, sendo estudas todas as suas potencialidades, e funcionalidades procurando ainda entender o comportamento desta tecnologia quando utilizada em simultâneo com o *Anycast*. As comunicações *Anycast* foram estudadas observando todo o potencial e limitações desta tecnologia. O balanceamento de carga é apresentado como uma das principais vantagens da utilização das comunicações *Anycast*, como tal considerou-se necessário efectuar um levantamento do estado de arte acerca do balanceamento de carga. Ao efectuar o levantamento do estado de arte das tecnologias, efectuou-se também um levantamento dos trabalhos que devido ao seu conteúdo estão relacionados com o tema desta dissertação. Nesta pesquisa foram encontrados trabalhos relacionados com o *Anycast*, com o IPSec e ainda com balanceamento de carga, mas não foram encontrados trabalhos que abordassem estas duas tecnologias em simultâneo.

Após o levantamento do estado de arte e dos trabalhos relacionados, são implementadas e testadas as tecnologias *Anycast* e IPSec, avaliando o comportamento destas em situações concretas. As tecnologias são inicialmente implementadas e testadas separadamente, sendo de seguida implementadas e testadas em simultâneo. Relativamente ao *Anycast*, foram observados os comportamentos destas comunicações em termos de: (1) endereçamento; (2) suporte dos SOs; (3) comportamento na resposta a *Pings*; (4) comportamento no estabelecimento de conexões TCP;

(5) balanceamento de carga; (6) e tempo de resposta a falhas. De modo a observar as comunicações *Anycast* em termos dos aspectos 5 e 6, foi criada uma arquitectura de rede capaz de colocar estes aspectos à prova. Esta arquitectura foi utilizada com implementações dos protocolos de *routing Unicast* OSPF e BGP, sendo os testes repetidos para ambas as implementações. Foram utilizados protocolos de routing *Unicast* nos testes, devido ao facto de não existir nenhum protocolo de *routing Anycast* devidamente reconhecido e implementado.

Relativamente ao IPSec foram observados os seguintes aspectos: (1) suporte dos SOs; (2) e aplicações existentes para implementar o IPSec. Foram também explorados três modos de configurar o IPSec através das ferramentas apresentadas, e utilizando os dois modos de configuração do IPSec, o IKE e a configuração manual.

De seguida foram implementadas as tecnologias *Anycast* e IPSec em simultâneo utilizando para tal os dois modos de configurar o IPSec. Através dos resultados obtidos foi possível observar que as comunicações *Anycast* não podem ser utilizadas com o IKE, podendo no entanto ser utilizadas com a configuração manual do IPSec.

Nesta dissertação é proposta uma solução capaz de permitir a utilização do IPSec para proteger as comunicações entre um cliente e um serviço *Anycast*, tornando assim possível aliar as vantagens do IPSec e das comunicações *Anycast*. A metodologia utilizada no desenvolvimento desta solução tem em conta o estado da arte de ambas as tecnologias, não implicando nenhuma alteração às mesmas. A utilização do *Anycast* com o IPSec apenas é possível se for utilizada a configuração manual do IPSec, como tal a solução desenvolvida utiliza este modo de configuração. No entanto, este modo de configuração introduz problemas de segurança na troca das chaves secretas. Por este motivo, esta solução possui uma camada de abstração que através de criptografia assimétrica permite a troca segura das chaves secretas por ambas as extremidades da conexão, o que constituía um dos principais problemas de segurança introduzido pela configuração manual do IPSec, oferecendo assim uma alternativa segura e imediata à utilização do IKE com o *Anycast*.

O protótipo foi desenvolvido na linguagem de programação JAVA, e oferece um GUI ao cliente para que este possa escolher os parâmetros da conexão IPSec que pretende estabelecer com o serviço suportado pelas comunicações *Anycast*. Todo o processo de criação das chaves secretas e transmissão destas, de um modo seguro, para o grupo *Anycast* acontece sem que o

utilizador tenha conhecimento disso. Já foram efectuados testes à solução, mas é necessário testar mais exaustivamente a solução proposta com cenários mais complexos.

Os testes efectuados permitiram verificar a operabilidade e as funcionalidades da solução e constatar que o *routing*, pode ter um papel importante nos tempos de reacção a falhas. No entanto não existe nenhuma proposta especificamente em *Anycast*, mas sim em *Unicast* (com os custos de ter muitas rotas). Outras soluções devem ser avaliadas.

6.2 Trabalho futuro

Como trabalho futuro propõe-se uma re-factorização do código desenvolvido de modo a melhorar a sua arquitectura e facilitar a modificação para inclusão de novas funcionalidades.

Propõe-se ainda uma melhoria no sistema distribuído utilizado para manter a coerência da informação de segurança, utilizado pelo grupo *Anycast*, de modo a optimizar a troca de mensagens, diminuindo o impacto destas na rede.

Apesar da certificação das chaves públicas consistir num problema fora do âmbito deste trabalho, é no entanto um problema de segurança bem identificado, podendo por isso no futuro ser incluído um módulo, na solução proposta, de certificação das chaves públicas.

Foi observado através do estudo efectuado nesta dissertação, que a utilização das comunicações *Anycast* está seriamente limitada pelo facto de não existir nenhum protocolo de *routing Anycast* devidamente reconhecido e implementado. Como tal, como trabalho futuro propõe-se também um estudo mais aprofundado acerca deste assunto.

Referências

[1] U. o. S. C. Information Sciences Institute, "Rfc 791: Internet protocol," *Network Working Group*, 1981.

[2] S. Deering and R. Hinden, "Rfc 2460: Internet protocol, version 6 (ipv6)," *Network Working Group*, 1998.

[3] K. S. S. Kent, "Rfc 4301: Security architecture for the internet protocol," *Network Working Group*, 2005.

[4] R. Engel, V. Peris, D. Saha, E. Basturk, and R. Haas, "Using ip anycast for load distribution and server location," 1998.

[5] T. Brisco, "Rfc 1794: Dns support for load balancing," *Network Working Group*, 1995.

[6] *Cisco Distributed Director*, Cisco Systems, 1996.

[7] Z. Zhou, G. Xu, J. He, and C. Deng, "Research of secure anycast," *Hybrid Information Technology, International Conference on*, vol. 1, pp. 186–190, 2006.

[8] L. U. Scott Weber, Liang Cheng, "A survey of anycast in ipv6 networks," *EEE Communications Magazine*.

[9] Y. R. E. Rosen, "Rfc 4364: Bgp/mpls ip virtual private networks (vpns)," *Network Working Group*, 2006.

[10] S. Friedl. (2010) An illustred guide to ipsec. [Online]. Available: http://www.unixwiz.net/techtips/iguide-ipsec.html

[11] G. Agarwal, R. Shah, and J. Walrand, "Content distribution architecture using network layer anycast," in *Proceedings of the Second IEEE Workshop on Internet Applications*

(wiapp '01), ser. WIAPP '01. Washington, DC, USA: IEEE Computer Society, 2001, pp. 124–. [Online]. Available: http://dl.acm.org/citation.cfm?id=882509.885275

[12] S. D. R. Hinden, "Rfc 2373: Ip version 6 addressing architecture," *Network Working Group*, 1998.

[13] K. Jun-ichiro itojun Hagino, "An analysis of ipv6 anycast," Tech. Rep., 2003.

[14] S. Kent, "Rfc 4302: Ip authentication header," *Network Working Group*, 2005.

[15] ——, "Rfc 4303: Ip encapsulating security payload (esp)," *Network Working Group*, 2005.

[16] W. S. P. Karn, P. Metzger, "Rfc 1851: The esp triple des transform," *Network Working Group*, 1995.

[17] R. G. C. Madson, "Rfc 2404: The use of hmac-sha-1-96 within esp and ah," *Network Working Group*, 1998.

[18] E. C. Kaufman, "Rfc 4306: Internet key exchange (ikev2) protocol," *Network Working Group*, 2005.

[19] E. Rescorla, "Diffie-hellman key agreement method," *Network Working Group*, 1999.

[20] A. Nuopponen, S. Vaarala, and T. Virtanen, "Ipsec clustering," in *SEC'04*, 2004, pp. 367–380.

[21] S. Shieh and Y.-Z. Lai, "Clustered architecture for high-speed ipsec gateway," *Workshop on Cryptology and Information Security*.

[22] Secure anycast tunneling protocol. [Online]. Available: http://www.anytun.org/

[23] O. Gsenger, "Secure anycast tunneling protocol (satp)," *Network Working Group*, 2008.

[24] K. P. D. Meyer, "Rfc 4274: Bgp-4 protocol analysis," *Network Working Group*, 2006.

[25] W. Z. P.-O. A. Weijia Jia, Gaochao Xu, "Efficient internet multicast routing using anycast path selection," *JOURNAL OF NETWORK AND SYSTEMS MANAGEMENT Volume 10, Number 4, 417-438,*, vol. 10, no. 4, pp. 417–438, December 2002.

[26] D. Katabi and J. Wroclawski, "A framework for scalable global ip-anycast (gia)," *SIGCOMM Comput. Commun. Rev.*, vol. 30, pp. 3–15, August 2000. [Online]. Available: http://doi.acm.org/10.1145/347057.347388

[27] P. Judge and M. Ammar, "Gothic: A group access control architecture for secure multicast and anycast," 2002.

[28] C. Castelluccia and G. Montenegro, "Securing group management in ipv6 with cryptographically generated addresses," in *Proceedings of the Eighth IEEE International Symposium on Computers and Communications*, ser. ISCC '03. Washington, DC, USA: IEEE Computer Society, 2003, pp. 588–. [Online]. Available: http://dl.acm.org/citation.cfm?id=839294.843401

[29] C. A. T. Dierks, "The tls protocol version 1.0," *Network Working Group*, 1999.

[30] S. Doi, S. Ata, H. Kitamura, M. Murata, and H. Miyahara, "Protocol design for anycast communication in ipv6 network," in *in Proceedings of 2003 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM'03*, 2003, pp. 470–473.

[31] S. Doi, S. Ata, H. Kitamura, and M. Murata, "Ipv6 anycast for simple and effective service-oriented communications," *IEEE Communications Magazine*, vol. 42, pp. 163–171, 2004.

[32] S. Matsunaga, S. Ata, H. Kitamura, and M. Murata, "Design and implementation of ipv6 anycast routing protocol: Pia-sm," *Advanced Information Networking and Applications, International Conference on*, vol. 2, pp. 839–844, 2005.

[33] S. Doi, S. Ata, H. Kitamura, and M. Murata, "Design, implementation and evaluation of routing protocols for ipv6 anycast communication," in *Proceedings of the 19th International Conference on Advanced Information Networking and Applications - Volume 2*, ser. AINA '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 833–838. [Online]. Available: http://dx.doi.org/10.1109/AINA.2005.156

[34] S. Sarat, V. Pappas, and A. Terzis, "On the use of anycast in dns," 2004.

[35] D. Johnson and S. Deering, "Rfc 2526: Reserved ipv6 subnet anycast addresses," *Network Working Group*, 1999.

[36] Bsd. [Online]. Available: http://www.bsd.org/

[37] Java. [Online]. Available: http://www.java.com/

[38] T. H. J. Ahrenholz, C. Danilov and J. Kim, "Core: A real-time network emulator," *Proceedings of IEEE MILCOM Conference*, 2008.

[39] Quagga routing suite. [Online]. Available: http://www.quagga.net/

[40] Ipsec-tools. [Online]. Available: http://ipsec-tools.sourceforge.net/

[41] The kame project. [Online]. Available: http://www.kame.net/

[42] setkey(8) - linux man page. [Online]. Available: http://linux.die.net/man/8/setkey

[43] racoon – ike (isakmp/oakley) key management daemon. [Online]. Available: http://netbsd.gw.com/cgi-bin/man-cgi?racoon++NetBSD-current

[44] Openssl. [Online]. Available: http://www.openssl.org/

[45] "Rocksaw raw socket library," 2011. [Online]. Available: http://www.savarese.org/software/rocksaw/

[46] The legion of the bouncy castle. [Online]. Available: http://www.bouncycastle.org/java.html