Firewall Traversal in Mobile IPv6 Networks

Dissertation
zur Erlangung des Doktorgrades
der Mathematisch-Naturwissenschaftlichen Fakultäten
der Georg-August-Universität zu Göttingen

vorgelegt von
Niklas Steinleitner
aus Marburg

Göttingen 2008

D7

Referent:	Prof. Dr. Hogrefe
Korreferent:	Prof. Dr. Fu
Tag der mündlichen Prüfung:	09.10.2008

Abstract

Middleboxes such as firewalls are an important aspect for a majority of IP networks today. Current IP networks are predominantly based on IPv4 technology, and hence various firewalls as well as *Network Address Translators* (NATs) have been originally designed for these networks. Deployment of IPv6 networks is currently work in progress. Given the fact that Mobile IPv6 is a recent standard, most firewalls available for IPv6 networks still do not support Mobile IPv6. Unless firewalls are aware of Mobile IPv6 protocol details, they will either block communication traffic under Mobile IPv6, or carefully deal with the traffic. This is a major impediment to the successful deployment of Mobile IPv6.

This thesis describes the problems and impacts of having middleboxes in Mobile IPv6 environments. Therefore, it firstly explains which types of middleboxes are given, what exactly a middlebox is and how such a middlebox works and secondly identifies the problems and explains the impacts of having firewalls in Mobile IPv6 environments. Afterwards, it studies several state-of-the-art middlebox traversal solutions, which can be regarded as potential solutions to deal with the Mobile IPv6 firewall traversal problems. It explains in detail how these solutions work, and evaluates them in terms of their applicability for Mobile IPv6 firewall traversal.

As the main contribution, this thesis proposes two solutions in detail, able to overcome the Mobile IPv6 firewall traversal problem. The first one, the *NSIS based Mobile IPv6 firewall traversal*, bases on the *Next Steps in Signaling* (NSIS) framework and the *NAT/Firewall NSIS Signaling Layer Protocol* (NAT/FW NSLP). Afterwards, it presents the second proposed solution, the *Mobile IPv6 Application Layer Gateway*. It explains in detail how these approaches are able to handle the problems and impacts of having firewalls in Mobile IPv6 environments.

Additionally, this thesis presents how the *NSIS based Mobile IPv6 firewall traversal* and the *Mobile IPv6 Application Layer Gateway* proof-of-concept implementations, developed as part of this thesis, have been implemented. Finally, it evaluates and analyses the developed proof-of-concept implementations and the two proposed approaches in general.

Zusammenfassung

Middleboxes, wie zum Beispiel Firewalls, sind ein wichtiger Aspekt für eine Großzahl moderner IP-Netzwerke. Heute IP-Netzwerke basieren überwiegend auf IPv4 Technologien, daher sind viele Firewalls und *Network Address Translators* (NATs) ursprünglich für diese Netzwerke entwickelt worden. Die Entwicklung von IPv6 Netzwerken findet zur Zeit statt. Da Mobile IPv6 ein relativ neuer Standard ist, unterstützen die meisten Firewalls die für IPv6 Netzwerke verfügbar sind, noch kein Mobile IPv6. Sofern Firewalls sich nicht der Details des Mobile IPv6 Protokolls bewusst sind, werden sie entweder Mobile IPv6 Kommunikation blockieren oder diesen sorgfältig handhaben. Dieses stellt einen der Haupthinderunggründe zum erfolgreichen Einsatz von Mobile IPv6 da.
Diese Arbeit beschreibt die Probleme und Auswirkungen des Vorhandenseins von Middleboxes in Mobile IPv6 Umgebungen. Dazu wird zuerst erklärt welche Arten von Middleboxes es gibt, was genau eine Middlebox ist und wie eine solche Middlebox arbeiten und zweitens die Probleme identifiziert und die Auswirkungen des Vorhandenseins von Firewalls in Mobile IPv6 Umgebungen erklärt. Anschließend werden einige State-of-the-Art Middlebox Traversal Ansätze untersucht, die als mögliche Lösungen um die Mobile IPv6 Firewall Traversal Probleme zu bewältigen betrachtet werden können. Es wird detailiert erklärt wie diese Lösungen arbeiten und ihre Anwendbarkeit für Mobile IPv6 Firewall Traversal evaluiert.
Als Hauptbeitrag bringt diese Arbeit zwei detailierte Lösungsansätze ein, welche das Mobile IPv6 Firewall Traversal Problem bewältigen können. Der erste Lösungsansatz, der *NSIS basierte Mobile IPv6 Firewall Traversal*, basiert auf dem *Next Steps in Signaling* (NSIS) Rahmenwerk und dem *NAT/Firewall NSIS Signaling Layer Protocol* (NAT/FW NSLP). Anschließend wird der zweite Lösungsansatz vorgestellt, der *Mobile IPv6 Application Layer Gateway*. Diese Arbeit erklärt detailiert, wie diese Lösungsansätze die Probleme und Auswirkungen des Vorhandenseins von Middleboxes in Mobile IPv6 Umgebungen bewältigen.
Desweitern stellt diese Arbeit vor, wie die *NSIS basierte Mobile IPv6 Firewall Traversal* und die *Mobile IPv6 Application Layer Gateway* Proof-of-Concept Implementierungen, die im Rahmen dieser Arbeit entwicklet wurden, implementiert wurden. Abschließend werden die Proof-of-Concept Implementierungen sowie die beiden Lösungsansätze allgemein evaluiert und analysiert.

Acknowledgements

I like to thank my supervisors Prof. Dr. Hogrefe and Prof. Dr. Fu for their kind support, their knowledge and the valuable suggestions and the discussions I shared with them as well as for the possibility to research under excellent conditions. The work at the Telematics Group and the Computer Networks Group in Göttingen has been a very positive experience and also a nice time.
I would also like to thank my colleagues in Göttingen who have accompanied me through the years. It has always been a pleasure to work with them and the discussions I shared with them were very fruitful.
This thesis would not be in its current shape without the comments of numerous people. I truly appreciate the efforts of Ingo Juchem, Florian Brauel, and Dr. Henrik Brosenne for reading this thesis, finding errors and providing helpful suggestions for improving this thesis. I also thank my students Swen Weiland and Jan Demter for contributing to this work with their considerable efforts and Martin Stiemerling, Hannes Tschofenig, Suresh Krishnan, and Henning Peters for the fruitful discussions over the past years.
Finally, I am deeply indebted to my parents, my family, and Doro for their patience and continuous support over the years and during the preparation and writing of this thesis.

Contents

1 Introduction

1.1 Motivation

Middleboxes [CB02], such as *Firewalls* (FW) and *Network Address Translators* (NAT) [EF94], are an important component for a majority of *Internet Protocol* (IP) [Pos81b] networks today. Current IP networks are predominantly based on IPv4 [Pos81b] technology, and hence various firewalls as well as NATs have been originally designed for these networks. NATs are necessary to overcome the IPv4 address space limitations of many network domains; firewalls are common to protect systems and networks against unsolicited traffic and attacks from the outside network, in particular the Internet. Middleboxes normally work on transport and higher layers and are not part of the traditional end-to-end Internet architecture. Deployment of IPv6 [DH98] networks is currently work in progress. However, some firewall products for IPv6 networks are already available [v6St]. It is foreseen that firewalls will become an indispensable device for protecting against unwanted traffic in operational IPv6 networks, especially in enterprise environments.

One main disadvantage of middleboxes is that they often implement knowledge about higher layer protocols and applications to perform. This prevents the deployment of new network protocols or applications, as long as the middleboxes do not implement the new protocol or are configured sufficiently. For example, a firewall is usually configured for only a set of common network protocols by allowing the corresponding ports, which highly affects the deployment of new applications. In addition, nowadays middleboxes are commonly only supports of traditional transport protocols, like *Transmission Control Protocol* (TCP) [Pos81c] and *User Datagram Protocol* (UDP) [Pos80], and reject other protocols, such as *Internet Protocol Security* (IPsec) [KS05] and *Stream Control Transmission Protocol* (SCTP) [SXM+00]. Furthermore, middleboxes usually maintain state tables to allow traffic for already established sessions, whereas these states need to be periodically refreshed in a certain manner.

Firewalls and NATs are a subject of active research and discussion in the *Internet Engineering Task Force* (IETF) [IETF] and research community [GF07, JTS08]. The IETF is currently developing recommendations for the operation of NATs; this work is done in the *BEHAVE working group* [BEHAVE]. The goal is to provide guidelines how NATs should behave with common protocols, e.g., guidelines for *TCP* [GBF+08],

UDP [AJ07], and *Internet Control Message Protocol* (ICMP) [SFSG08]. Nevertheless, even if middleboxes do behave more standardised, this will not overcome all the existing problems.
Given the fact that *Mobile IPv6* [JPA04] is a recent standard, most firewalls available for IPv6 networks still do not support Mobile IPv6. Unless firewalls are aware of Mobile IPv6 protocol details, they will have to either block Mobile IPv6 communication traffic, or carefully deal with the traffic per-connection, or allow this traffic in general through manual pre-configuration. This could be a major impediment to the successful deployment of Mobile IPv6.
To overcome the problems, it requires the usage of a middlebox configuration solution. In general we can distinguish between two types of middlebox configuration: the *implicit* and the *explicit* approaches. The implicit middlebox configuration is triggered by data traffic. A stateful packet filtering firewall or a NAT establish state information in the local state table based on the header information in the data traffic itself. To the category of implicit approaches belong *Simple Traversal of UDP Through Network Address Translators* (STUN) [RWHM03] and *Traversal Using Relays around NAT* (TURN) [RMM08], since these signaling protocols do not interact with the middlebox itself but rather implement a hole-punching behavior as middleboxes treat these messages as ordinary data traffic.
A more advanced technique would be to enable clients, or more precisely applications, to signal their solicited traffic as well as communication characteristics and profiles explicitly to the middleboxes. With such an explicit approach, the number of state-refreshing or keep-alive messages could be significantly decreased, and middleboxes could be made directly aware of the communication traffic and characteristics of the end-hosts. In such an explicit approach the intention is to interact with the middleboxes and therefore the middlebox has to implement additional protocols. *Application Layer Gateway* (ALG) [SH99], *Policy-Based Networks* (PBN) [YPG00] solutions like *Common Open Policy Service* (COPS) [DBC+00], *Middlebox Communication* (MIDCOM) [SKR+02] or the *Next Steps in Signaling* (NSIS) [HKLdB05] *NAT/Firewall NSIS Signaling Layer Protocol* (NAT/FW NSLP) [STAD08] are examples of this approach.
The main difference between the two approaches is the flexibility regarding the firewall pinhole creation versus the need to enhance existing middleboxes to understand additional protocols. Implicit approaches are less flexible regarding the creation of pinholes which often leads to the need to tunnel one protocol on top of another one. Additionally, since there is no interaction with the middlebox and the end-host, it is unclear how long the established state is kept alive at the middlebox. As a consequence, more frequent refresh messages have to be sent to assure that the state is not discarded. Finally, explicit approaches provide better security properties. However, the major disadvantage is the slow adoption of new protocols at middleboxes.
A number of different middlebox configuration protocols and proposals have been pro-

posed over the last years and several new are currently being developed. Some existing middlebox traversal solutions, such as *SOCKS* [LGL+96], *Universal Plug and Play* (UPnP) [UPnP], *Middlebox Communication* (MIDCOM) [SKR+02], *Simple Traversal of UDP Through Network Address Translators* (STUN) [RWHM03], *Traversal Using Relay NAT* (TURN) [RMM08], *Interactive Connectivity Establishment* (ICE) [Ros07] or *Policy-Based Networks* (PBN) [YPG00] solutions like *Common Open Policy Service* (COPS) [DBC+00] potentially can be extended for performing middlebox or firewall traversal even in Mobile IPv6 networks. However, some of them require prior knowledge of the existence of firewalls and most do not address the issue of discovering firewalls. Furthermore, they do not support the node mobility case intentionally and thus may require significant efforts to be extended for use in Mobile IPv6 networks.
One promising solution is *Application Layer Gateway* (ALG) [SH99]. ALGs are application specific agents that allow an application on a client in one network or domain to connect to its counterpart running on a host in a different network or domain transparently. They typically operate inside routers along with the firewall and NAT component. Currently, ALGs are often used for Voice over IP (VoIP) signaling to traverse firewalls and NATs. The ALG investigate the payload of for example *Session Initiation Protocol* (SIP) [RSC+02] messages which arrive at the middlebox and either translate the IP address and port from one address domain to another or analyse the packets and opens required firewall pinholes. This solution is potentially able to allow Mobile IPv6 to work in presence of middleboxes.
Another recent initiative within the IETF, *Next Steps in Signaling* (NSIS) [HKLdB05], has developed a signaling protocol for firewall and NAT traversal, namely the *NAT/-Firewall NSIS Signaling Layer Protocol* (NAT/FW NSLP) [STAD08]. NSIS utilises a two-layer signaling paradigm, which defines a lower layer protocol for general extensible IP signaling and a layer for various signaling applications such as signaling for NAT and firewall traversal. Since its initial design, NSIS has considered node mobility as its potential use scenarios [SFJ+08] but the feasibility has never been proved and not implemented. Additionally, how the NSIS NAT/firewall traversal signaling protocol supports IPv6 mobility is not specified, neither have the required firewall pinhole formats been considered in the design phase.

1.2 Scope of this Thesis

This thesis will give an introduction of the problems and impacts when middleboxes, more precisely, firewalls are placed in Mobile IPv6 environments. Furthermore, it identifies several potential state-of-the-art approaches and evaluates whether they might be applicable for Mobile IPv6 firewall traversal. In the following, two different solutions developed to overcome the Mobile IPv6 firewall traversal problem will be introduced and described in detail. Both have been designed as main contribution of

this thesis, namely the *NSIS based Mobile IPv6 firewall traversal* [SFL07] in Section 5.3 and the *Application Layer Gateway based Mobile IP6 firewall traversal* [KSYB08, KSSB08] in Section 5.4. This applicability has also been verified with the help of two proof-of-concept implementations. Finally, this thesis evaluates and analyses the developed proof-of-concept implementations and the two proposed approaches in general.

1.3 Structure of this Thesis

This thesis is structured as follows. Firstly, Chapter 2 introduces some fundamentals and basics on middleboxes, firewalls and the Mobile IPv6 protocol. Chapter 3 describes the problems and impacts of having firewalls in Mobile IPv6 environments. In Chapter 4, several state-of-the-art solutions are introduced and evaluated, Chapter 5 presents the two different Mobile IPv6 middlebox traversal solutions developed as part of this thesis, one based on the NSIS signaling layer protocol for NAT/firewall traversal [STAD08], namely the *NSIS based Mobile IPv6 firewall traversal* [SFL07] in Section 5.3 and the other based on Application Layer Gateways [SH99], namely the *Application Layer Gateway based Mobile IP6 firewall traversal* [KSYB08, KSSB08] in Section 5.4 and explains how they can be utilised for firewall traversal in Mobile IPv6 networks. Chapter 6 discusses the security, authentication and authorisation aspects which has to be considered for Mobile IPv6 firewall traversal and presents how they can be accomplished regarding the solutions presented in Chapter 5. Chapter 7 presents the proof-of-concept implementations, developed within this thesis and Chapter 8 evaluates the two approaches in general as well as the developed proof-of-concept implementations. Chapter 9 discusses open issues and further work and Chapter 10 summarises and concludes this thesis.

2 Definitions and Basic Concepts

2.1 Middleboxes

RFC 3234 [CB02] defines the term middleboxes as follows:

"A middlebox is ... any intermediary device performing functions other than the normal, standard functions of an IP router on the datagram path between a source host and destination host."

RFC 3234 identifies several different types of middleboxes. One way of classifying them is by their behaviours. The first group operates on packets, does not modify application-layer payloads and does not insert additional packets. This group includes *Network Address Translators* (NAT) [EF94], *NAT Protocol Translation* (NAT-PT) [TS00], *SOCKS* gateways [LGL+96], IP tunnel [Sim95] endpoints, *Virtual Private Network* (VPN) gateways [GLH+00], packet classifiers, markers, schedulers, transport relays, IP firewalls and application firewalls. There are other middleboxes, such as *Transmission Control Protocol* (TCP) [Pos81c] performance-enhancing proxies, *Application Layer Gateways* (ALG) [SH99], gatekeepers and session control boxes, transcoders, proxies, caches, modified *Domain Name System* (DNS) servers [Moc87a, Moc87b], content and applications distribution boxes and load balancers, which modify application-layer payloads. The most common ones among these middlebox types may be NAT, firewalls and VPN gateways. In this thesis the term middlebox refers to firewalls and NATs only, other types are out of scope.

2.2 Firewalls

2.2.1 Definition

A firewall is one of many possible measures to protect a system, a device, a network or a domain from a number of Internet security threats. A firewall makes examinations and/or modifications of the data flowing through the network or domain border at a transition point possible, on the basis of the given security requirements. Such a transition point is normally a border between two networks or domains, which runs under separate administration policies. Accordingly, a firewall can only be used as

a measure against those threats which can be neutralized at an appropriate network transition point. While more detailed descriptions on how a firewall fundamentally works and could be built may be found in for example [CB94], [ZCCR06], and [Spe07], the following subsections give an overview of the tasks expected for firewalls as well as the different types of firewalls.

2.2.2 Task of a Firewall

A firewall implements certain functions for the purpose of neutralising certain threats. From the available literature there are different views on which tasks and/or functions a firewall exactly has to perform. Theoretically, all measures to prevent threats at a network or domain border can be performed in a firewall. In practice, a firewall performs the following tasks: identity-referred access control, filtering and modification of data, hiding of structures as well as logging of security-relevant events [Roe06]. These tasks are briefly described below:

- *Access control*: the firewall must be able to accomplish an access control function based on the identities of sender and receiver of the data flows. The identities can be determined by the analysis of the data, or by other methods, for example a node's IP-address, a specified process identified by a port on a node or a user's identification.

- *Filtering and modification*: based on the information obtained by the analysis of the data flows over a firewall, the firewall must decide which of the conveyed data represent a threat. Due to this decision the data are forwarded or blocked and thus dropped. Additionally, it may specify rules whether and how data must be modified before forwarding.

- *Hiding*: it is desirable for a firewall to be able to hide the network topology at the network or domain border on one side from the other side. This reduces the possibility to procure information about possible targets to an aggressor.

- *Logging*: a firewall must be able to perform the logging function for identifying the irregularities from the data history; otherwise attacks on systems will be unnoticed unless the system becomes malfunctioned or other observable events take place.

2.2.3 Types of Firewalls and Mode of Operation

There are different types of firewall concepts and technologies; the most common are the *Stateless Packet Filter*, the *Stateful Packet Filter*, the *Application Layer Gateway* (ALG) or *Proxy* and the *Network Address Translation* (NAT). This section give an

overview about the most common firewall concepts, describes their mode of operation and their assets and drawbacks.

2.2.3.1 Stateless Packet Filter

The *Stateless Packet Filter* is the simplest concept to implement a firewall. The first implementation was done by Jeffrey C. Mogul in 1989 [Mog89]. This first packet filter *"screend"* was a userspace program, which makes the decision whether a packet is allowed to traverse or not.
In general, a stateless packet filter is realised as part of a network component which is used for connecting two different networks or domains. When the network component is extended with firewall functionality, the packet filter examines, typically based on pre-configured firewall rules, whether the packets traversing the network component represent a threat. If a packet represents a threat, it is dropped. The values within the packets are compared with the rules specified before in order to decide whether a threat is present or not. Mostly, the values of the header fields of the network- and transport-layer are used for the analysis. Two types of packet filters are commonly used: in stateless packet filter firewalls, any examination of a packet does not consider earlier analysed packets; in contrast, in a stateful packet filter (see Section 2.2.3.2), the examination of a packet takes account of previous analyses packets.
Most packet filters are IP packet filters, also capable to analyse *Transmission Control Protocol* (TCP) [Pos81c], *User Datagram Protocol* (UDP) [Pos80] and *Internet Control Message Protocol* (ICMP) [Pos81a] packets. An IP packet filter examines the packet (primarily the IP header) according to the pre-configured firewall rules, for example performing *allow* or *deny* functions depending on the type of the packet (e.g., TCP or UDP), the IP addresses of the sender and receiver, and the port numbers of the sender and receiver. Packets addressed to a certain receiver can be recognised belonging to either *allow* or *deny* category, which in turn results in those packets to be either allowed to traverse or to be dropped.
In a stateless packet filter, each packet is analysed by oneself without the coherence to other packets; the relation to other packets is ignored. Therefore, a classical packet filter is not able to detect whether a arrived packet belongs to an already established connection or session. This concept is realised with help of a stateful packet filter, described in Section 2.2.3.2.

A stateless packet filter firewall achieves the tasks of a firewall specified in Section 2.2.2 in the following way:

- *Access control*: the packet filter can verify the identity of the communication participants on the basis of their port numbers and their IP addresses.

- *Filtering and modification*: the firewall decides whether the data is forwarded or not. A modification of the data is not possible before the forwarding process.
- *Hiding*: hiding of the internal network structure is not possible with a filter.
- *Logging*: if a packet is dropped by the firewall and is not forwarded, this is noted accordingly.

2.2.3.2 Stateful Packet Filter

Commercial *Stateful Packet Filters* (SPF) have been available since 1993. The first Open Source products were *"IP-Filter"* (IPF) [Ree96] and *"sf Firewall"* [sfFW96], both published in 1996. Nowadays, the Linux Open Source *"netfilter/iptables"* [netfilter] represents one of the most common and most powerful stateful packet filter.
A stateful packet filter is a packet filter which is able to fall back at the examination of the current packets to the information which was gained by the examination of earlier arrived packets. In its typical use in handling TCP [Pos81c] traffic, a stateful packet filter is able to examine whether a TCP packet belongs to an already established TCP-connection or not. Therefore, the stateful firewall heavily depends on the three-way handshake of the TCP protocol and needs to remember the state of a connection in a *state table*. When a client behind a stateful packet filter initiates a new connection, it sends a packet with the *SYN bit* set in the message header. The firewall considers any packet with a *SYN bit* as an indication of creating new connections by a TCP client. The response to such a packet is a message in which both the *SYN* and the *ACK bit* are set. Upon the receipt of this response, the client responds with a message in which only the *ACK bit* is set, and the connection enters the *ESTABLISHED* state. If a packet is later examined by the stateful filter, it can determine whether the packet corresponds to an already established connection through matching an existing state in the state table. The stateful packet filter also examined whether the context fits into the current connections by inspecting the connection state and the sequence number. A stateful packet filter is also capable to perform stateful packet filtering for protocols different from TCP, like UDP [Pos80] or ICMP [Pos81a], and to maintain their established connections in the state table.
Nowadays, stateful packet filters are widely spread. This is mainly based on their convenience, their low cost implementability and their applicability even for high-speed networks, and in most scenarios, their sufficient security.

A firewall solely based on stateful filters can fulfill the following tasks:

- *Access control*: the stateful filter can determine the identity of the communication participants through their port numbers, their IP addresses and the state of

network communications (such as TCP connection setup). Only packets matching a known connection state will be allowed by the firewall; others will be rejected and dropped.

- *Filtering and modification*: the firewall decides whether the data is forwarded or not. A modification of the data is possible before the forwarding process.
- *Hiding*: hiding of the internal network structure is not possible with a filter.
- *Logging*: if a packet is dropped or modified by the firewall, this is noted accordingly.

2.2.3.3 Application Layer Gateway

An *Application Layer Gateway* (ALG) [SH99], also known as *Application Level Gateway* or *Proxy*, is an intermediary program which acts as both a server and a client for the purpose of making requests on behalf of other clients. The first proxy was a *File Transfer Protocol* (FTP) [PR85] proxy written by Marcus Ranum in 1991. In a proxy, all packets are serviced internally or by passing them on, with possible translation, to other servers. During the packet processing the proxy can examine whether the data in the packets represent threats from the application level and decide whether to allow passing through or to drop the packets. Therefore, the proxy has to be aware of the application protocols details to examine whether the packets represent threats. Different from packet filters, typically the proxy can only use the data which are gained from the application layer and cannot directly use the information contained in the data of a lower-layer (which is transparent to proxies).
One special type of proxy is the *Curcuit Relay*. Curcuit Relay is a proxy which works on the transport-layer and thus is not capable to examine the application layer protocols. This concepts lead to the development of the *SOCKS Protocol* [LGL+96], described in Section 4.2.

A firewall which is realised only by the use of a proxy can fulfill the following tasks:

- *Access control*: the proxy can verify the identity of the communication participants based on the user-referred information which may be contained within the application layer.
- *Filtering and modification*: the firewall decides whether the data is forwarded or not. The modification of the data is possible before the forwarding process.
- *Hiding*: as the communication end-points are completely separated from each other on all layers, the network topology may be hidden by a proxy.
- *Logging*: if a packet is dropped or modified by the firewall, this is noted accordingly.

2.3 Network Address Translation

Originally, *Network Address Translation* (NAT) [EF94] were designed to overcome the shortage of IP address space in the Internet. NATs allow the translation of local IP addresses into one globally unique IP address. This is done by mapping a few real IP addresses which are required for Internet communication to the many local IP addresses, whereas the local IP address can be shared over the Internet. This process is called *Network Address Translation* or *Network Masquerading*.
As traffic passes from the local network to the Internet, the source address in each packet on the fly can be translated by a NAT device from the private addresses to the public address(es). When a reply returns to the NAT, it uses the connection tracking data it stored during the outbound phase to determine where on the internal network to forward the reply. The TCP or UDP client port numbers are used to demultiplex the packets if the NAT has only one public IP address. Both IP address and port number are translated when multiple public addresses are available on packet return. To a system on the Internet, the NAT device itself appears to be the source or destination for this traffic. How the addresses and port numbers are converted is defined by the rules of the NAT. If the address and port are converted for a packet, the same conversion must be also done for all other packets of the same data stream. Subsequently, an entire subnet can be mapped into one IP address. Nodes can be addressed over the NAT only if the rules define an appropriate address conversion.
The main disadvantage of network address translation is that it breaks the end-to-end connectivity and does not allow a true end-to-end connectivity which is required by some applications, e.g., real-time application like VoIP. Such applications require the use of for example a proxy to successfully traverse the NAT. However, this complicates and slows down the communication process, especially the communication initialisation.

A firewall that only uses a NAT component can fulfill the following tasks:

- *Access control*: the NAT can verify the identity of the communication end-points on the base of their port numbers and their IP addresses.
- *Filtering and modification*: the NAT decides whether the data are forwarded or not and whether an address conversion for a certain communication connection is accomplished or not. A modification of the data is necessary before forwarding in order to change the used IP addresses and port numbers.
- *Hiding*: the network structure is completely hidden by the NAT.
- *Logging*: if data are dropped or modified by the NAT, this is noted accordingly.

2.4 Mobile IPv6

The *Internet Protocol version 6* (IPv6) [DH98], developed by the *Internet Engineering Task Force* (IETF) [IETF] *IP Version 6 working group* [IPv6] in 1998, itself does not provide a node the feasibility to stay reachable while roaming around in different IPv6 networks. Due to the lack of mobility support, packets intended to such a moving *Mobile Node* (MN) would never reach the mobile node while it is away from the home network. The mobile node could change its IP address each time it moves to a new point-of-attachment. However, in this case the mobile node would not be able to maintain already established transport-layer connections.

Mobile IPv6 (MIPv6) [JPA04, ADD04], developed by the IETF *Mobility for IPv6 working group* [MIP6], is a protocol developed in 2004 as an extension of *IPv6* [DH98] to support node mobility. Mobile IPv6 is an update of *Mobile IP* [Per96], which was designed for IPv4 node mobility, and adds roaming capabilities of mobile nodes into IPv6 networks.

The major benefit of this standard is to keep alive any communication between a *Correspondent Node* (CN) and a mobile node while the mobile node moves from one to another IPv6 network and thus changing its point-of-attachment. This is feasible as the mobile node is always addressable via its *Home Address* (HoA), a IP address assigned to the mobile node within its home network. The mobile node is always reachable at its home address, independent from its current point-of-attachment in the IPv6 Internet. If the mobile node is at home, packets addressed to this home address are normally routed to the mobile node.

While the mobile node is away from its home network and attached to a foreign network, it maintains at least one *Care-of Address* (CoA). The care-of address is an IP address with the subnet prefix of the foreign network and addresses the mobile node in the foreign network. This care-of address can for example be derived through IPv6 auto-configuration. While the mobile node is staying in the foreign network, packets addressed to the care-of address will be routed to the mobile node.

Figure 2.1 depicts how Mobile IPv6 works. To be able to forward packets to the mobile node while it is not in the home network, a so called *Binding* between the mobile node's home address and the care-of address is created. While the mobile node is away from home, for example, if the mobile node moves to a foreign network (black line in Figure 2.1), a node on its home link is taking care of packets address to the mobile node's home address. This node is called *Home Agent* (HA). When the mobile node receives a new care-of address, it associates the care-of address with its home address by performing a binding registration with the home agent. This is done by sending a *Binding Update* (BU) message to the home agent (red line in figure). The home agent replies to the mobile node with a *Binding Acknowledgement* (BA) message, which finishes the binding registration. The home agent now intercepts packets addressed to

the mobile node's home address and forwards them using an *Internet Protocol Security* (IPsec) [KS05] *Encapsulation Security Protocol* (ESP) [Ken05] tunnel to the mobile node's care-of address [ADD04].

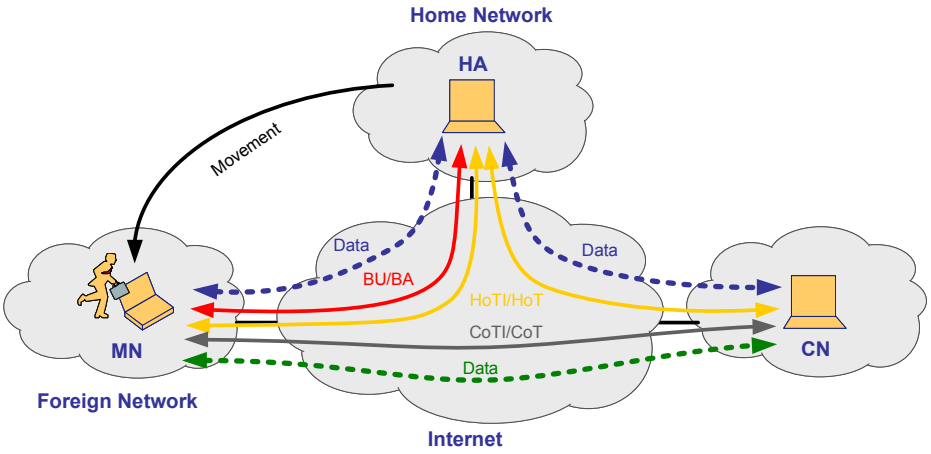


Figure 2.1: Mobile IPv6

The mobile node could also send a Binding Update to a potential correspondent node to update the correspondent nodes binding cache. Therefore, it first has to complete the *Return Routability Procedure*. Goal of the return routability procedure is to ensure that the mobile node is addressable at its home address and its claimed care-of address. A correspondent node only accepts Binding Updates from the mobile node which successfully performs the return routability procedure to ensure that the data traffic is directed or received from the mobile node using the care-of address.
Therefore, Mobile IPv6 performs tests whether packets addressed to the home address and care-of address are routed to the mobile node. *"The mobile node can pass the test only if it is able to supply proof that it received certain data (the 'keygen tokens') which the correspondent node sends to those addresses. These data are combined by the mobile node into a binding management key."* [JPA04]
The message flow for the return routability procedure is also shown in Figure 2.1 (yellow and gray lines). The mobile node sends two messages: the *Home Test Init* (HoTI) and *Care-of Test Init* (CoTI) messages at the same time. The HoTI message is sent to the correspondent node via the home agent, whereas the CoTI message is sent directly, not via the home agent. The correspondent node responses to the HoTI message with the *Home Test* (HoT) message, which is again sent via the home agent. The *Care-of Test* (CoT) message is sent by the correspondent node in response to the CoTI message. This message is again sent directly to the mobile node.
These four messages implement the return routability procedure. The procedure requires some processing at the involved nodes to calculate some tokens and cookies.

More details are described in [JPA04]. Nevertheless, the HoT and CoT messages can be returned quickly. After having successfully completed the return routability procedure, the mobile node is able to update its binding cache at the correspondent node.
Two communication modes between a mobile node and a correspondent node are possible. The first mode, the so called *Bi-directional Tunneling*, does not require the correspondent node to support Mobile IPv6. In this mode, packets from the correspondent node to the mobile node are routed to the home agent which tunnels them to the mobile node by performing *IPv6 encapsulation* [CD98]. Packets from the mobile node to the correspondent node are tunneled to the home agent, so called *Reverse Tunneled*, and than routed to the correspondent node. The blue lines in Figure 2.1 represent the packets being sent in this mode.
The second mode, called *Route Optimisation*, requires the correspondent node to support Mobile IPv6. Here, the mobile node registers its current binding also to the correspondent node, thus the correspondent node is able to address packets directly to the mobile node's care-of address. This direct communication between the mobile node and correspondent node allows to use the shortest path and rapidly decreases the traffic at the home agent. This is depicted with the green lines in Figure 2.1.

3 Problem Statement

3.1 The Middlebox Traversal Problem

Firewalls and NATs are normally located at the edge of networks or domains. This is not only the case in business networks, even many residential users deploy NATs and firewalls, often without knowing about this. This section presents an overview about the problems and impacts of having middleboxes in IPv4 and IPv6 networks. This is done with the help of VoIP traffic as an example to highlight in which cases these problems appear. In the following, this section describes in detail the problems and impacts of having firewalls placed in Mobile IPv6 environments.
The problem basically consists of two sub-problems. While today's firewalls are able to dynamically open, maintain and close multiple ports when required, e.g., for VoIP protocols, they remain ineffective at securely supporting unsolicited incoming media flows, e.g. VoIP calls. NATs prevent two-way communication, as the private IP addresses and ports inserted by nodes behind a NAT in the packet payload are not routable in public networks. In general, traversal of unsolicited incoming packets which are not allowed through pre-configuration is not possible with existing firewalls or NATs without the help of additional mechanisms. However, letting such packets pass through firewalls or NATs is essential for today's communication, like for example VoIP protocols.

3.1.1 The Firewall Challenge

A firewall is one of many possible measures to protect a system or network from a number of Internet security threats. A firewall makes examinations and/or modifications of the data flowing over the network or domain border at a transition point possible on the basis of the given security requirements. Such a transition point is normally a border between two networks or domains which runs under separate administration policies. Accordingly, a firewall can only be used as a measure against those threats which can be neutralised at an appropriate network transition point. So, firewalls are used to protect networks against certain types of attacks from different networks and the Internet. This is done by blocking traffic based on different parameters, mostly, the source and destination address, the source and destination ports and the traffic type.

Firewalls also base their decision on the direction of the traffic flow. Incoming traffic, from the untrusted public side, respectively the Internet, is normally only allowed if that session has been initiated from a node on the trusted private network or domain before, or has been allowed by manual pre-configuration, whereas outgoing traffic from the trusted private domain is allowed by default. Figure 3.1 depicts this problem. *Node A* initiates communication to *Node B*, the subsequently related communication from *Node B* back to *Node A* is allowed as the communication was initiated from the trusted private domain. When *Node C* tries to initiate communication to *Node A*, the firewall blocks this unsolicited incoming traffic, as it has not been initiated from the trusted private domain before. Manual pre-configuration is one possibility to allow this kind of communication to traverse the firewall. Because of the unpredictability of today's communication like VoIP and the enormous effort to manual pre-configure all kind of unsolicited incoming packets, this approach is not an option.

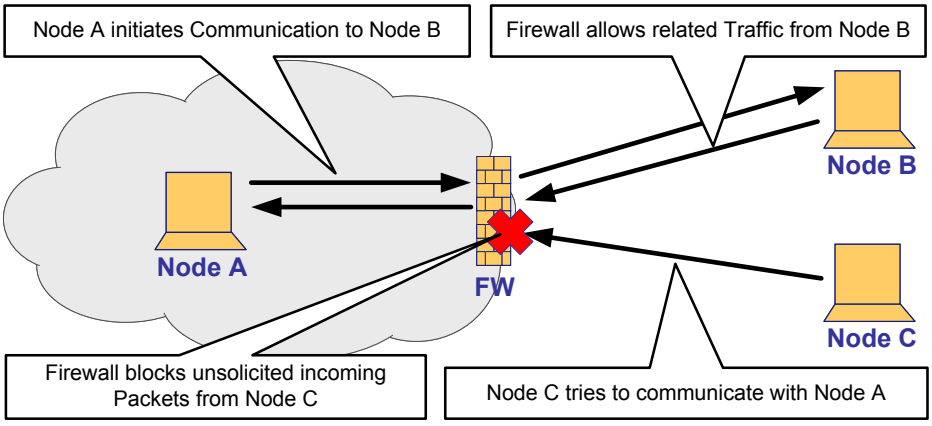


Figure 3.1: The Firewall Challenge

However, a lot of today's Internet communication, like VoIP telephony, is based on unsolicited incoming traffic, from unknown and untrusted sources. Evidently, this is a problem of the firewall policies as described above, which block unsolicited incoming traffic. As administrators are not likely to change these policies and to allow this kind of unrestricted communication, this prevents the communication to be established as shown in Figure 3.1.

Any solution which wants to overcome this challenge must allow this traffic to successfully traverse the firewall. In addition, it should not require major changes to the firewalls and should not reduce the level of security provided by it.

3.1.2 The NAT Challenge

As described in Section 2.3, NATs translate IP addresses and port numbers from private networks with private address ranges into public addresses that are routable in the Internet.
When a node from the private network sends for instance VoIP traffic to a node on the public network, the NAT dynamically assigns an unused port number at the public address of the NAT. By maintaining a *state table* which links the private address and port number to the public address of the NAT and the allocated port number, called *Binding*, the NAT is able to identify incoming traffic intended for a private node, to re-write the address and the port number and to forward the traffic to the private node. Figure 3.2 depicts the NAT challenge.

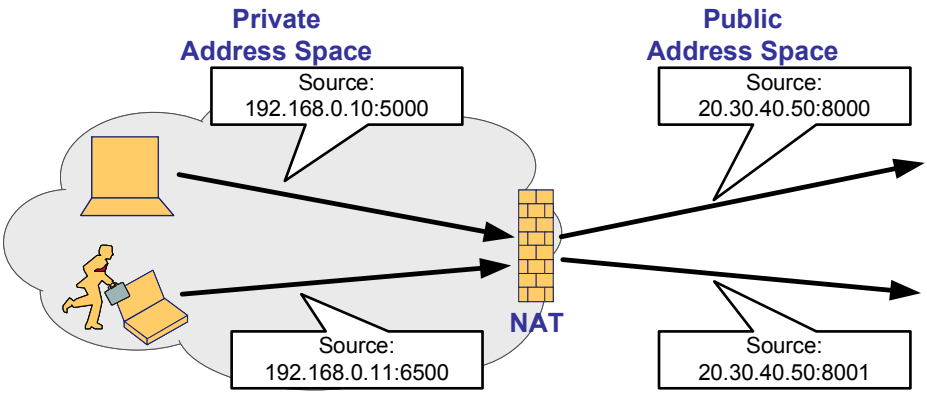


Figure 3.2: The NAT Challenge

Receiving unsolicited incoming traffic is a more difficult issue, as the private address of the receiver node behind the NAT is not routable in the public network. Therefore, the sending node from the Internet must somehow trigger the internal node to create a binding in the NAT which would allow the public node to address the receiver and to traverse the NAT by using this binding. Otherwise, unsolicited incoming traffic of applications like VoIP are not able to reach the node behind the NAT.
Any solution which wants to solve this challenge must allow the communication traffic as well as the unsolicited incoming connections of for example a VoIP application to traverse the NAT. It should also minimise changes or dependencies on upgrading NATs.

3.1.3 The Mobile IPv6 Firewall Traversal Challenge

To identify how firewall traversal can be achieved in Mobile IPv6 environments, it is necessary to understand the detailed problems and impacts of having firewalls in such

environments.
Mobile IPv6 [JPA04, ADD04] introduces several new types of messages, which can be categorised into registration messages *(Binding Update* (BU), *Binding Acknowledgements* (BA)), Home/Care-of Testing messages *(Home Test Init* (HoTI), *Home Test* (HoT), *Care-of Test Init* (CoTI), *Care-of Test* (CoT)) and data traffic. A new mobility header is introduced in all these new messages, and all messages between the *Mobile Node* (MN) and the *Home Agent* (HA) are IPsec ESP encapsulated [ADD04]. When a user moves to a visited network or domain, a firewall – no matter whether it is located in the home network, the visited network or the access network of the corresponding node – will affect the Mobile IPv6 signaling and data messages. For instance, *Route Optimisation* (RO), an integral part of the Mobile IPv6 specification, does not work with the state-of-the-art firewalls that utilise stateful packet filtering. This set of extensions is a fundamental part of the protocol, enabling optimised routing of packets between a mobile node and its correspondent node, thus providing optimised communication performance. However, firewall technologies do not support Mobile IPv6 or are not even aware of IPv6 mobility extension headers. Since most networks in the current business environment deploy firewalls, this may prevent future large-scale deployment of Mobile IPv6. Secondly, another mode of communication in Mobile IPv6, namely *Bi-directional Tunneling*, does not work under some scenarios, for example, when a firewall is placed in the access network or the home network. In addition, it is difficult or, in some scenarios, even impossible for the Mobile IPv6 Binding Update messages to traverse firewalls as they are encapsulated using IPsec ESP. In summary, these deployment issues with firewalls occur due to the nature that the commonly used firewalls possess [LFPT06]:

- do not allow IPsec ESP traffic to traverse which is used for Mobile IPv6 registration messages (e.g., Binding Update, Home Test Init) between MN and HA and thus preventing MNs from updating their binding cache and performing route optimisation,
- block unsolicited incoming traffic and hence drop connection setup requests or packets,
- do not understand the Mobile IPv6 *Mobility Header* [JPA04],
- do not understand data packets encapsulated in Mobile IPv6 and likely drop them.

The following subsections explore these problems in detail from both operational and technical aspects regarding the relevant scenarios.

3.2 Mobile IPv6 Firewall Traversal - Scenarios and Issues

Without loss of generality, let us consider a typical roaming scenario where a mobile Node with a *Personal Digital Assistant* (PDA) is roaming outside of its/their company (hereafter, the so-called *Mobile Service Provider*, or MSP) into a visited network (*Access Service Provider*, or ASP) which is also a corporate network. The MN wants to communicate with his home network or its home agent in order to register its new location and additionally with another node, the corresponding node, for data communication. The visited network could be protected by a firewall, thus parts of the traffic to the mobile node may be blocked. Besides, both the home network and the network of the correspondent node may deploy firewalls. These three possible firewall placements introduce several problems which could prevent Mobile IPv6 from operating successfully in the presence of firewalls. In all cases, pinholes have to be open on the firewalls for enabling successful communication. These problems can be differentiated under three basic scenarios:

- Firewall located at the edge of the MN's ASP,
- Firewall located at the edge of the CN's ASP,
- Firewall located at the edge of the MN's MSP.

The following subsections investigate these three basic scenarios individually, and present how firewalls might prevent Mobile IPv6 from a successful operation.

3.2.1 Firewall Located at the Edge of the MN's ASP

The first scenario assumes that the MN roaming to another network (i.e., ASP, which deploys a firewall, the ASP-FW) wants to enjoy communication with its company or *Internet Service Provider* (ISP) (MSA/MSP/ASA). Therefore, the MN needs to traverse the ASP-FW. Figure 3.3 depicts how the components are placed in this scenario. Several issues need to be considered:

- To be able to use IPsec ESP [ADD04] between MN and HA, *Security Associations* (SA) have to be established before. Therefore, *Internet Key Exchange* (IKE) [HC98] and *IKEv2* [Kau05] packets have to be exchanged between these nodes. Firewalls may drop the IKE packets and circumvent the establishment of the IKE security association and thus the sending of IPsec ESP protected messages like Binding Update and Binding Acknowledgement. A possible solution might be to manually pre-configure the firewalls to allow IKE/IKEv2 signaling to bypass, although a dynamic solution should be preferable.

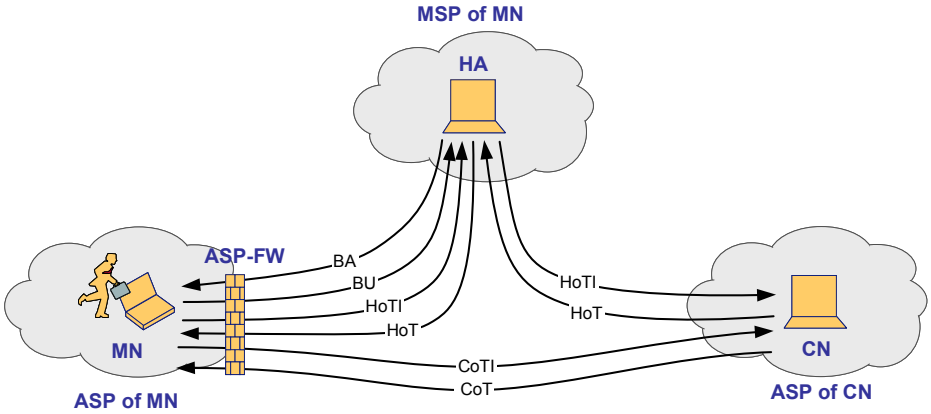


Figure 3.3: Firewall Located at the Edge of the MN's ASP

- Binding Updates and Binding Acknowledgements should be protected by IPsec ESP, but many firewalls drop IPsec ESP packets because they cannot determine whether inbound ESP packets are authorised. A possible solution might be to manually pre-configure the ASP-FW so that Mobile IPv6 traffic is allowed to traverse. However, not every administrator would permit IPsec traffic in general, so it should be possible to dynamically install these firewall rules.

- The ASP-FW may drop the Home Test messages and may prevent the completion of the *Return Routability Test* (RRT) procedure, as the Home Test messages of the RRT are protected by IPsec ESP in tunnel mode. Therefore, either manual pre-configuration or dynamic on-demand configuration of firewall pinholes rules on the ASP-FW are possible solutions for these type of messages. The ASP-FW also may prevent correspondent nodes from establishing communications, e.g., route optimisation traffic, as incoming packets are dropped since the packets do not match any existing firewall state.

- If the MN successfully sends a Binding Update to its HA and the subsequent traffic is sent from HA to MN (in bi-directional tunneling), there is also no corresponding state on the firewalls, and they drop the incoming packets. Hence, it is necessary to dynamically configure the ASP-FW to allow this data traffic to traverse.

- If the MN roams and moves to another access network protected by a different firewall, all new incoming packets are dropped as they do not match any existing firewall state.

3.2.2 Firewall Located at the Edge of the CN's ASP

Here, a MN wants to communicate with a CN that deploys a firewall at the edge of its network or domain. Therefore, the traffic from the MN to the CN needs to traverse the CN's ASP-FW. Figure 3.4 depicts how the components are placed in the second scenario. Several issues need to be considered:

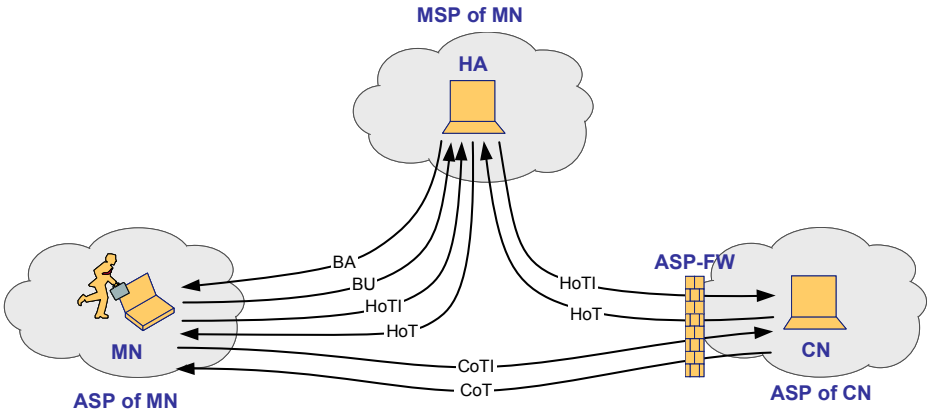


Figure 3.4: Firewall Located at the Edge of the CN's ASP

- The Care-of Test Init message is sent using the care-of address of the MN as the source address. Such a packet does not match any state in the protecting firewall, as the states in the firewall are bounded to the old address of the MN. The CoTI message will thus be dropped by that firewall, and as a consequence, the RRT cannot be completed and route optimisation cannot be performed. Every packet has to be tunneled through the home network, more precisely the home agent.

- The Home of Test Init message is sent using the home address of the MN as the source address. Such a packet might not match any state in the protecting firewall. Thus, the HoTI message might be dropped by that firewall, and as a consequence, the RRT cannot be completed.

- The Binding Update message to the CN is sent using the care-of address of the MN as the source address. Such a packet does not match any state in the protecting firewall, as the states are bound to the old address. The Binding Update message will be dropped and the binding cache at the CN cannot be updated. Thus, route optimisation cannot be performed. Every packet has to be tunneled through the home network, more precisely the home agent.

- If the Binding Update to the CN is successful, the firewall still drops packets that are coming from the CoA, because these incoming packets are sent from the CoA and do not match any existing firewall state.

3.2.3 Firewall Located at the Edge of the MN's MSP

In this scenario, the MN roaming to another foreign network (ASP) wants to enjoy communicating with a CN and his own company (MSP), and the MSP deploys a firewall at its network or domain border. The MN traffic needs to traverse the MSP-FW to be able to run Mobile IPv6. Figure 3.5 depicts how the components are placed in this third scenario. Several issues need to be considered:

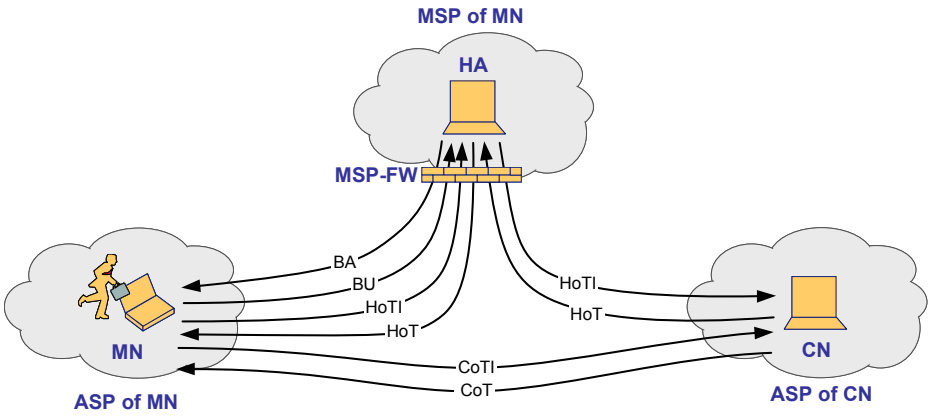


Figure 3.5: Firewall Located at the Edge of the MN's MSP

- If the firewall protects the home agent by blocking ESP traffic, some of the Mobile IPv6 signaling (e.g., Binding Update and HoTI) may be dropped at the firewall. This prevents mobile nodes from updating their binding cache and performing route optimisation, since these messages must be protected by IPsec ESP. Manual pre-configuration is a solution, but also has some problems as mentioned before.

- If the firewall is a stateful packet filter and protects the home agent from unsolicited incoming traffic, the firewall may drop connection setup requests from correspondent nodes, as well as packets from mobile nodes.

3.3 Summary

This section has given an overview about the problems and impacts of having middleboxes in IPv6 environments. Several problems consist which prevent Mobile IPv6 to work in presence of firewalls. In general, the problems and impacts of having firewalls placed in Mobile IPv6 environments can be solved by enabling the following messages or types of traffic to successfully traverse the firewalls:

- IKE/IKEv2 Signaling for establishing Security Association,
- Binding Update and Binding Acknowledgement,
- Return Routability Test,
- Binding Update and Binding Acknowledgement CN,
- Route Optimisation,
- Bi-directional Tunneling.

Any solution that allows these messages to successfully traverse the middleboxes between the Mobile IPv6 nodes is generally applicable for Mobile IPv6 firewall traversal. Therefore, any proposed solution presented in this thesis handle these cases in detail and thus show it applicability to overcome the described problems. The following Chapter 4 now presents the state-of-the-art middlebox traversal solutions as they all representing potential solutions for Mobile IPv6 firewall traversal.

4 State of the Art in Middlebox Traversal

This chapter introduces several state-of-the-art approaches for middlebox traversal. They represent potential solutions to solve the problems that occur when firewalls are placed in Mobile IPv6 environments as described in Chapter 3. After describing the potential firewall traversal solutions in Sections 4.1 to 4.9, Section 4.10 evaluates the applicability of the most promising candidates for Mobile IPv6 firewall traversal. Section 4.11 summarises this chapter.

4.1 Universal Plug and Play

Universal Plug and Play (UPnP) [UPnP] is a technique that is aimed to achieve easy control of the network and middleboxes in small office or home office (SOHO) scenarios. UPnP was initiated by *Microsoft* [Micro] and developed by the *UPnP-Forum* [UPnP]. The UPnP architecture is designed to address a number of issues, and to allow the easy configuration of small networks. It allows client applications to discover and configure nodes, including networking nodes, services components, firewalls and NATs, if these nodes support UPnP.
The main disadvantage of the UPnP approach is related to security. It does not satisfactorily solve the security problems. UPnP relies on the firewalls or NATs to open pinholes to the outside world under the dynamic control of the UPnP client; for example a VoIP application using *Session Initiation Protocol* (SIP) [RSC+02]. This capability to allow a client to control the NAT or firewall independently is most likely contrary to most security and firewall policies and therefore may not be accepted by large corporate customers.
To put it in a nutshell, UPnP enables plug and play for network nodes, which is easy for everyday user. It is suitable for many services and nodes, and not just for middlebox control. However, UPnP is designed for small networks and hence only works in a single network and not across domains. Therefore, it is not suitable for multi-hop installations. Besides, it does not support any kind of policy control or security authorisation mechanism, so every node on the local network or domain can configure or open the middlebox and access the Internet; even Microsoft recommends that UPnP should be disabled for high security. Thus, UPnP is limited to small

installations, not suitable for middlebox control across different domains and therefore not applicable for Mobile IPv6 firewall traversal.

4.2 SOCKS

The *SOCKS Protocol Version 5* [LGL+96] defines a protocol which allows nodes on a private network or domain to establish communication with nodes from another network or domain, e.g., the Internet. Therefore, a so called *SOCKS Server* is located on the network border, which relays the communication sessions of the *SOCKS Clients* up on their request.
When a SOCKS client wants to establish a TCP session with a node from the Internet, it sends a connection request to the SOCKS server. The request specifies the direction of the TCP session to be established and whether the SOCKS server has to act as the initiator or responder for this session. The SOCKS server responses to the request and informs the SOCKS client about the external IP address and port number which will be used for the TCP session. The SOCKS server now forwards traffic between the private and public TCP sessions, as long as either of them is terminated. UDP sessions are also initialised to the SOCKS server via a TCP session. The SOCKS client reaches the IP address and the UDP port number of the UDP relay server. The SOCKS server now forwards datagrams between the private and the public network. The SOCKS client adds an additional SOCKS header to every packet, which carries the IP address and UDP port of the receiver in the public network. The SOCKS server also adds this SOCKS header when forwarding datagrams to the SOCKS client. Figure 4.1 shows a SOCKS connection scenario.

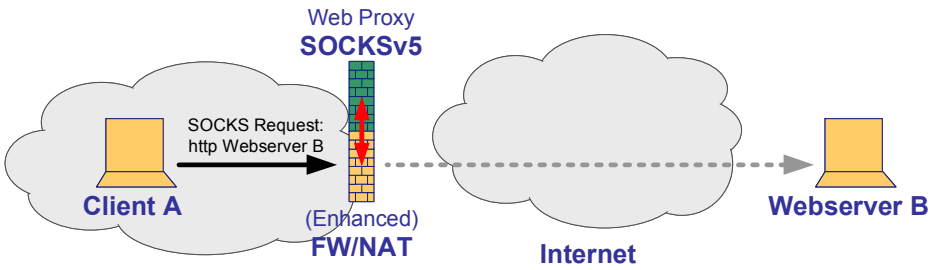


Figure 4.1: A SOCKS Connection Scenario

SOCKS supports both IPv4 and IPv6 as well as a *SOCKS-based IPv6/IPv4 gateway mechanism* [Kit01] which allows translation between IPv6 and IPv4 nodes. However, SOCKS is not applicable for generic server applications as only one passive TCP session per request is allowed. Additionally, the conveyance of IP header parameters is not defined with the exception of IP addresses.

Applications must typically be modified to support SOCKS. Nevertheless, SOCKS is widely spread and many implementations for clients and servers are available, especially for *Hypertext Transfer Protocol* (HTTP) [FGM+99] and *File Transfer Protocol* (FTP) [PR85]. Furthermore, manual pre-configuration on the SOCKS client is required, as the SOCKS server address must be provided before. This significant limits or even circumvents the applicability of SOCKS in mobile environments.

4.3 Application Layer Gateway

Application Layer Gateways (ALGs) [SH99] are application specific agents that are aware of the protocols details of a specific protocol, e.g., *Session Initiation Protocol* (SIP) [RSC+02] for VoIP, and are able to understand the protocol messages and their dependencies within the communication. Thus, they are able to allow applications in different networks and domains behind middleboxes to connect each other transparently. An ALG processes the application traffic while transit and assists the middlebox in implementing its function. Therefore, the ALG performs a deep packet-inspection of packets and comprehends the application protocol which are supported, e.g., SIP for VoIP. The work of the ALG is transparent to end-hosts and does not terminate or influence sessions between the end-hosts.
The ALG interacts with a middlebox to set up middlebox state, firewall pinholes or access control filters or uses the middlebox state information. The ALG may modify application specific payload, for instance the application signaling to change private IP addresses to public IP addresses or the ports used by signaling and media traffic. The ALG also performs whatever else necessary to allow the application specific traffic to traverse the middlebox.
The application layer gateway technique requires the replacement of the existing middlebox with an ALG, but ALGs may also be co-resided with middleboxes. Many vendors provide software updates for their middleboxes to support ALG functionality for specific applications.
So, ALG may operate inside routers along with the firewall and NAT components; however, middlebox and ALG are not the same. ALG performs the application specific payload and notify the middlebox to open specific firewall pinholes or to add additional state information. The ALG functionality is application specific and requires the examination and the re-composition of network- or above-layers payload, whereas middleboxes usually only operate on the network- or above-layers header.
The complexity of ALGs depends on the application level knowledge required to process the payload and maintain or manipulate states. Ideally, the ALGs should be simple and not require excessive computation or state storage. Depending on the protocol, an ALG may be difficult or easy to implement. However, in some cases it may not be possible at all, e.g., when the payload is encrypted and is opaque to the

ALGs.

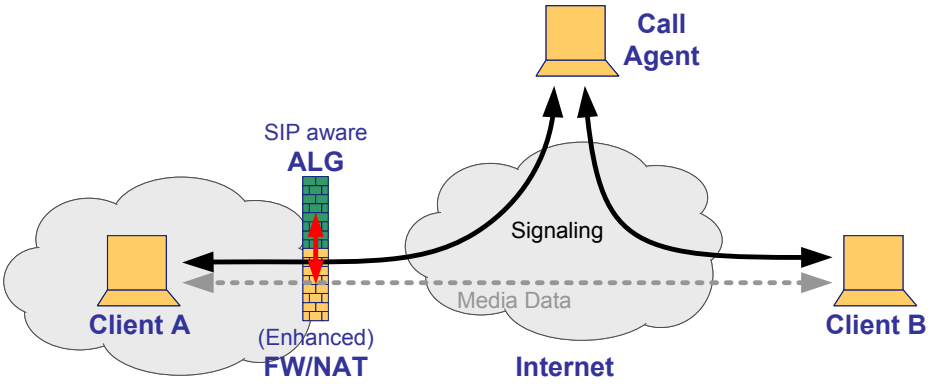


Figure 4.2: Application Layer Gateway Scenario

Nowadays, ALGs are often used for VoIP signaling to traverse firewalls and NATs. The ALG investigate the payload of for example SIP messages which arrive at the middlebox and either translates the IP address and port or analyses the packets and opens required firewall pinholes. Figure 4.2 depicts such an Application Layer Gateway scenario.
However, when several middleboxes exist in the data path, each of them needs to be updated to achieve middlebox traversal. This would slow down the deployment of ALGs for new protocols or restrict the early support to large cooperated networks. Another disadvantage of this approach is that the ALG performance may become the bottleneck of the middlebox under heavy load.

4.4 Middlebox Communication

As another middlebox traversal solution, the IETF *MIDCOM working group* [MIDCOM] has standardised the *Middlebox Communication* (MIDCOM) framework [SKR+02], *MIDCOM Requirements* [SMS+02], *MIDCOM Semantics* [SQT08] and *MIDCOM Management Information Base* (MIB) [QSS08] for communication between applications servers and middleboxes. There are three main entities in the MIDCOM framework: *Middleboxes*, *MIDCOM Agent*, *and MIDCOM Policy Decision Point (PDP)*.
The MIDCOM framework allows trusted third parties, so called *MIDCOM Agents*, to request middleboxes to perform middlebox functionality without being aware of application protocol details. Instead, a combination of application awareness and knowledge of the middlebox function could be possessed in the MIDCOM agent. This

combination allows the MIDCOM agent to enable the application traffic to traverse the middlebox. However, the task of middlebox discovery becomes a problem.
The main idea of MIDCOM is to have several MIDCOM agents assisting one middlebox, so that the middleboxes can stay focused on their services like firewall and NAT. As such, the middlebox communication protocol allows for selective communication between a specific MIDCOM agent and one or more middlebox functions on the interface. A MIDCOM agent is performing ALG functionality and is located outside of the middlebox. It performs and analyses the application specific payload and notifies the middlebox to open specific firewall pinholes, to add additional state information, modify application specific payload, or perform whatever else is necessary to allow application traffic to traverse the middlebox. MIDCOM agent may be located in application servers, proxy servers or end-hosts. The only requirement is that the entity should be able to establish a trust relationship between itself and the middlebox.
The MIDCOM policy decision point allows the middleboxes to verify authorisation of a MIDCOM agent. Therefore, it also acts as a policy repository, holding MIDCOM related policy profiles in order to make authorisation decisions. For example, a MIDCOM agent assisting a firewall might request the firewall to allow access for application specific, dynamically generated, session traffic. Before the signal and datagram arrive at the middlebox, the MIDCOM agent uses the MIDCOM configuration protocol to configure the corresponding port on the firewall, via an established MIDCOM session. If there is no MIDCOM session available, a session establishment procedure is required. After receiving a configuration or session establishment message, the middlebox validates whether the agent is authorised to perform this action. If the profile of MIDCOM agent does not exist in the middlebox, it may appeal to the PDP via the policy interface. The interaction between middleboxes and PDPs is defined in [QSS08].
Before any transaction on a middlebox policy rule is possible, a valid MIDCOM session must be established. All transactions are transmitted within the scope of a session, however, there might be several sessions between the same agent and middlebox at the same time. A MIDCOM session is an authenticated and authorised association between one agent and one middlebox. Sessions are initiated by agents and can be terminated by either the agent or the middlebox.
As already described, the SIP protocol [RSC+02] can be used to establish VoIP sessions. As an example, Figure 4.3 depicts a possible SIP middlebox traversal scenario within the MIDCOM framework.
The main idea of MIDCOM is to move application knowledge from the middlebox into a trusted third entity, providing a generalised communications interface between them. This decomposition provides a number of advantages, including improved performance, lower software development and maintenance costs, improved ability to support traversal of packet filters by complex protocols, and easier deployment of new applications. Nevertheless, some disadvantages exist. The discovery of middleboxes is

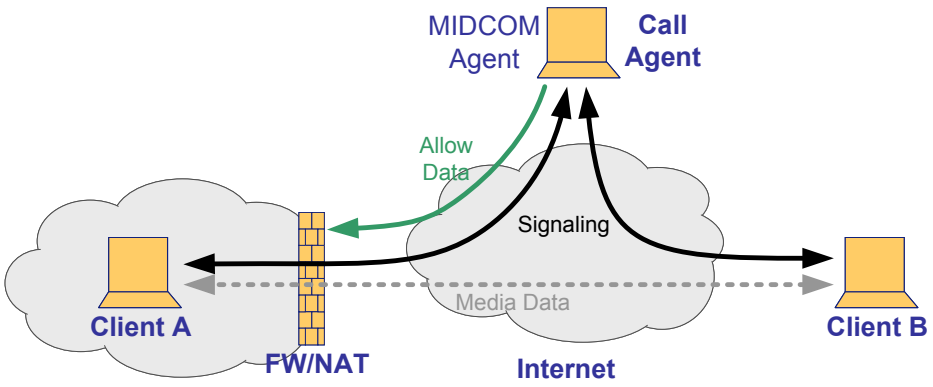


Figure 4.3: SIP Application Traverses Middlebox in MIDCOM Framework

out of scope of MIDCOM. If several firewalls are located in the data path, the MIDCOM agent will not be aware of the existence of some firewalls without performing the discovery procedure, and thus is unable to configure them.
The *NEC's Simple Middlebox Control* (SIMCO) [SQC06] protocol is a lightweight implementation of the MIDCOM protocol used to control middleboxes such as firewalls and network address translators. It fully complies with the MIDCOM semantics [SQT05] and hence has the same advantages and disadvantages as the MIDCOM protocol. An Open Source SIMCO implementation is available at [Simco].

4.5 Policy-Based Networks

Policy-Based Networks (PBN) [YPG00] is a network model for granting access and allocation of specific resources located at a network node. PBN is based on specific policies derived from criteria defined by network administrators. The PBN concept is defined in [YPG00], which describes the framework, the network elements as well as the interaction among them.
In PBNs, there is no definition what a policy is or how a policy has to be expressed. Policies are usually described in high-level languages. The more flexible are the policies, the more powerful is the PBN, however, policies have to be translated into specific rules in order to allocate the requested network resources.
The PBN concept was designed as a proposal for *Resource ReSerVation Protocol* (RSVP) [BZB+97] and it *Integrated Services* [Wro97,SPG97]. Thus, it was developed to dynamically allocating bandwidth among different routers within an administrative domain according to specific policies stored in the routers. However, the PBN concepts are generic enough to be extended to other network resources different from network bandwidth, even as a solution for middlebox traversal.

The PBN solution presented here is an only possible approach under investigation, not a real description or proposal that is currently working or standardised. The PBN model itself is simple and the main components are depicted in Figure 4.4.

- *Policy Decision Point* (PDP) is the network entity where the decision about accepting or denying the network resource (in place, a firewall pinhole) is taken, based on both, the information provided in the resource request and the policies stored at the policy repository.
- *Policy Enforcement Point* (PEP) is the network entity where the policy is really enforced once the PDP has taken a decision about a resource request, in place, the middlebox.
- *Policy Repository* is the entity where the policies and profiles are stored.
- *Policy Editor* is usually the administrator of the PBN. This entity is in charge of editing and configuring the policies in order to accept or deny network resource requests (i.e., firewall pinholes).

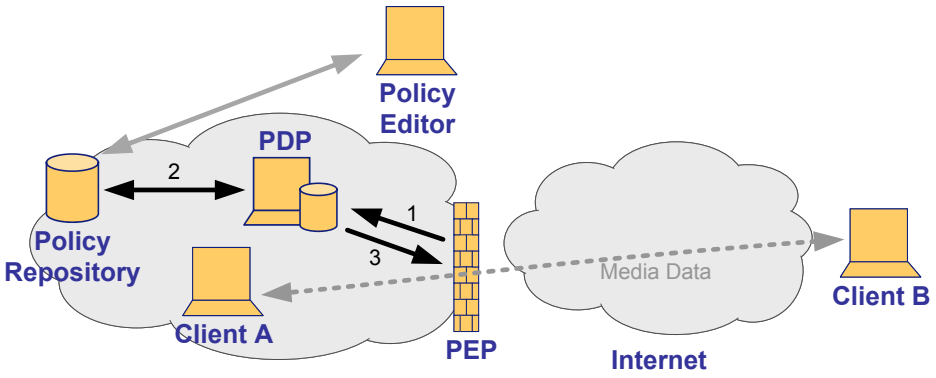


Figure 4.4: The Policy-Based Network Components and Scenario

The policy-based networks middlebox traversal procedure could work as follows (compare Figure 4.4):

1. The PEP detects that a network resource hosted by itself needs to be allocated. Such a network resource (i.e., firewall pinhole) needs to be authorised before being allocated, so it makes a resource request to the PDP. A request consists of specific data required by the PDP, such as the network resource to be allocated, the credentials of the user asking for the resource, and the user's IP address.

2. After the PDP receives the resource request from the PEP it determines the policies that has to be applied from the policy repository.

3. By using the information received on the resource request as well as the policies stored in the repository, the PDP takes the decision whether this firewall pinhole request is allowed or not. The PDP translates the policies into firewall rules to be applied on the PEP in case it is allowed. This firewall pinhole rules are sent to the PEP and enforced there.

Several architectural models exists; they depend on the location of the PDP. The before-mentioned example illustrates the generic architectural model which explains the functionality of PBNs. Obviously, some kind of communication between the PEP and the PDP is necessary. There are some alternatives, e.g., *Common Open Policy Service* (COPS) [DBC+00], *Simple Network Management Protocol* (SNMP) [LMS02] and *SNMPv2* [Pre02]. The most important requirement is that the protocol is able to provide some critical features like security in the communications or the ability to transport parameters between PDP and PEP. According to this model, the PBN approach can be a potentially used to enable middlebox traversal. However, following assumptions need to be considered:

- The network resource to be allocated is a firewall pinhole at the middlebox, i.e., a firewall pinhole for VoIP traffic.
- The PEP is the middlebox where the policies/firewall pinholes are enforced to allow or deny certain kind of traffic.
- The PDP is a node located at the administrative domain which takes the decision about allowing or denying firewall pinholes on the middlebox (PEP).
- The policy repository is the database where the policies and profiles are stored. It might be part of the *Authentication, Authorisation, and Accounting* (AAA) [VCF+00] infrastructure or not.

4.6 Session Traversal Utilities for NAT

The initial version of the *Simple Traversal of UDP Through Network Address Translators* (STUN) protocol [RWHM03] was developed by the IETF *MIDCOM working group* [MIDCOM]. It has been revisied by the IETF *BEHAVE working group* [BEHAVE], and is now called *Session Traversal Utilities for NAT* (STUN) [RMMW08]. STUN allows a client to discover whether it is behind a NAT or firewall, the type of the NAT or firewall and to identify the public IP address and ports which are used. This is achieved with the help of a special server in the public address space,

the *STUN Server*. The client sends an exploratory message to the STUN server, the STUN server then examines this message and informs the client about the public IP address and the ports used by middleboxes. When the client obtains this information, it uses this public IP address and ports as its source information and receives the packets arriving at the public address and port (although having been processed by the NAT or traversed the firewall), without intervening the STUN server. Figure 4.5 depicts such a STUN scenario.

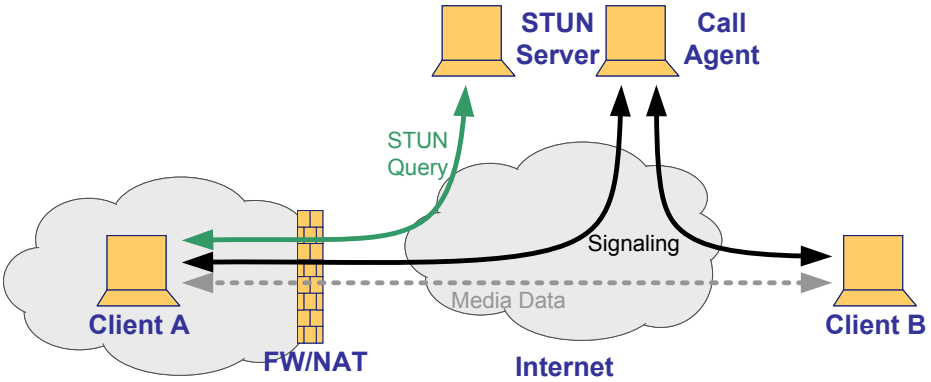


Figure 4.5: Session Traversal Utilities for NAT Scenario

The STUN approach has two main disadvantages. Firstly, STUN does not address the need to support TCP based communication since it is only works for UDP. Secondly, STUN does not work with the most common NAT type in corporate networks, the *Symmetric NAT*. A symmetric NAT is one where all requests from the same internal IP address and port, to a specific destination IP address and port, are mapped to the same external IP address and port. If the same host sends a packet with the same source address and port, but to a different destination, a different mapping is used. With a symmetric NAT, only the external host that receives a packet can send a UDP packet back to the internal host. The destination address of real outgoing UDP traffic from the NAT should be the external host, not the STUN server. However, the information learned from the STUN server when a client issues the exploratory procedure will be the STUN server's address and port number. As this is incorrect, the communication attempt will fail.
The same problem occurs with incoming traffic. STUN relies on the fact that an outgoing port which is bound to the STUN server can be reached by any traffic from any part of the network, independent of the source IP address. Thus, everyone can use the binding in reverse direction and reach the receivers port on the client. Thereby, NATs are vulnerable to port scan attacks and generally create major security concerns. Therefore, this approach is not able to solve all problems and introduces additional security risks.

The IETF *BEHAVE working group* [BEHAVE] has proposed an additional mechanism, *Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)* [RMM08], which was designed to solve the traversal problems for symmetric NATs.

4.7 Traversal Using Relays around NAT

The *Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)* protocol [RMM08] allows a client behind a NAT or firewall to receive incoming data over TCP or UDP connections by the use of a *TURN Server* placed in the public Internet. The TURN server relays packets from an external IP address and port towards the client if the client had previously sent a packet through the TURN server towards that IP address and port. In contrast to STUN, TURN restricts the binding to only a single address, as if all internal clients were communicating with this single address. Thus, TURN provides the same security functions as symmetric NATs and firewalls, but turns the middlebox state tables so that the client behind a NAT or firewall can be the receiver of a connection. Figure 4.6 depicts such a TURN scenario.

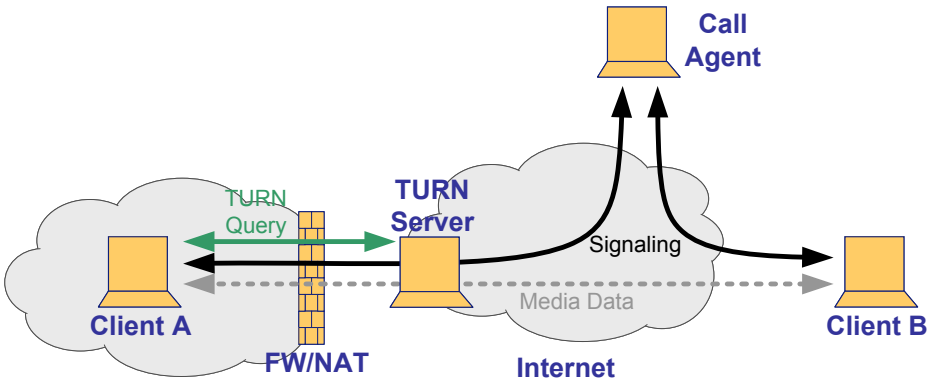


Figure 4.6: Traversal Using Relays around NAT Scenario

Like STUN, TURN relies on a server which is inserted inside the media and the signaling path. This TURN server must be located in the service provider network or in the Internet. The TURN client sends an exploratory message to this server and the TURN server responds with the public IP address and the ports used by the NAT or firewall. This information is used for the establishment messages and for subsequent data traffic. Since there is no change in the destination address, TURN can be used with symmetric NATs and firewalls.

4.8 Interactive Connectivity Establishment

As described before, STUN and TURN have particular advantages and limitations. Thus, the IETF *MMUSIC working group* [MMUSIC] has proposed a framework called *Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols* [Ros07]. The *MMUSIC working group* has also defined a two-phase exchange [RS02] of *Session Description Protocol* (SDP) [HJP06] messages to allow the establishment of multimedia sessions. This offer/answer approach is used by several protocols, e.g., SIP [RSC+02].
Interactive Connectivity Establishment is a technique for NAT and firewall traversal for UDP-based media streams established by the offer/answer model, but can be extended to handle additional transport protocols, for example for TCP as specified in [Ros08].
However, ICE is not a new protocol. It is an extension to the offer/answer model and thus creates a framework that joins different techniques, such as STUN (Section 4.6) and TURN (Section 4.7). It works by performing connectivity checks by peer-to-peer connectivity tests and including IP addresses and ports in the SDP offers and answers. This is done by utilising the revised STUN specification, called *Session Traversal Utilities for NAT* [RMMW08]. Additionally, ICE uses the STUN extension *Traversal Using Relays around NAT (TURN)* [RMM08]. The ICE framework allows the client to learn the topology it encounters and the variety of middleboxes that may exist between a client and a network. Hence, the client learns how to communicate with other clients and to successfully traversing existing middleboxes. Thus, ICE offers a middlebox traversal solution for applications. For example, it allows SIP-based VoIP calls to cross different types of middleboxes. Therefore, ICE determines what type of firewalls exists between the clients and uses the best relevant techniques to address the communication between the clients based on the available connection paths.

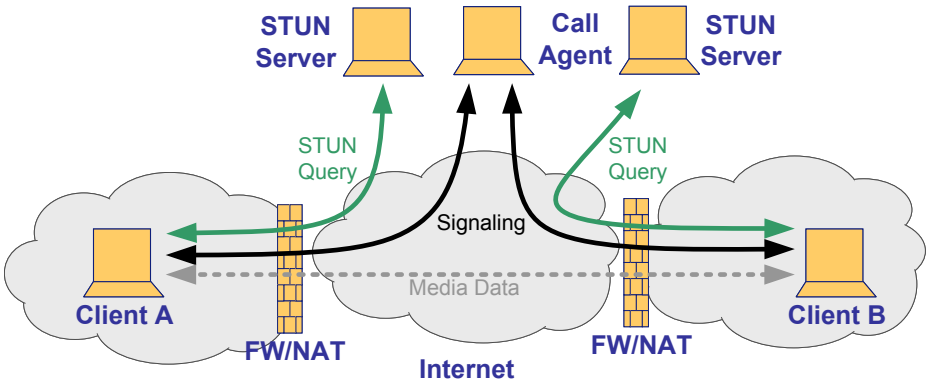


Figure 4.7: Interactive Connectivity Establishment Scenario

In a typical ICE deployment environment, as depicted in Figure 4.7, two clients want to communicate with each other. They can communicate indirectly with the help of some signaling protocols like SIP utilising offer/answer exchange of SDP messages [RS02], but they are not able to communicate directly. Initially, the clients are not aware of their own network topologies; i.e., they do not know whether they are behind middleboxes or not. ICE now offers the possibility to learn their topologies and thus to find potential path candidates to communicate with one another.
In the depicted ICE scenario, the two client are *Client A* and *Client B*. Both clients are placed behind middleboxes and both might not be aware of their topology nor the type of the middleboxes. The clients are able to initiate an offer/answer exchange of SDP messages via a rendezvous server, for example a SIP server, and thus establish a session between Client A and Client B. This is achieved with the help of STUN and TURN servers in the network. Consequently, the clients discover a set of candidate transport addresses which allows to communicate with each other. A candidate transport address is a combination of an IP address and a port for a particular transport protocol, whereas the address might be a transport address of a client network interface, a translated transport address on the public side of a middlebox (so called *Server Reflexive Address*) or a transport address allocated from a TURN server (so called *relayed address*). Any discovered candidate transport addresses of the clients can potentially be used to communicate with each other. Effectively, many combinations will not work, for instance candidates with an IP address which are only validate in private networks. ICE allows the clients to discover which combinations of candidate addresses will work by trying all possible combinations until it finds all working candidates.
The advantage of ICE is that it builds a unified framework to utilise different traversal protocols and avoid their individual drawbacks. The main disadvantages of ICE are its complexity and its long candidate discover procedure delay.

4.9 Next Steps in Signaling and the NAT/Firewall NSIS Signaling Layer Protocol

The *Next Steps in Signaling* (NSIS) framework [HKLdB05] has been developed by the IETF *NSIS working group* [NSIS] with the goal of supporting various signaling applications which install and manipulate certain control states in the network. Such states are meaningful for data flows and are installed and manipulated on network nodes supporting NSIS (NSIS Entities, NEs) along the data path. Not every node has to be such an NE. For instance, in the the *NAT/Firewall NSIS Signaling Layer Protocol* (NSLP) [STAD08] case only NAT/firewall boxes need to be the NEs along the data path of a data flow besides the end-hosts. The basic protocol concept does not

depend on any signaling application. Two NSIS entities that communicate directly are said to be in a *peer relationship*. Thereby, either or both NEs can store state information about the other NE, but it is not mandatory to establish a long-term signaling connection between them.

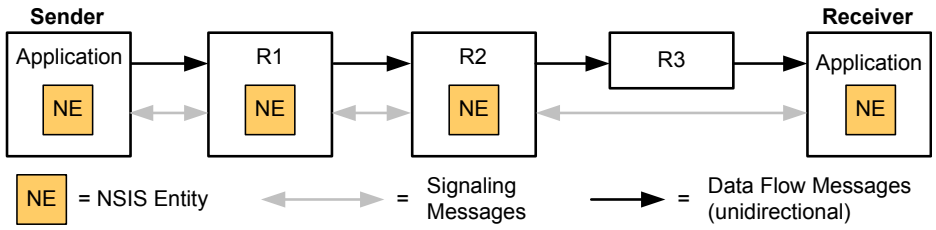


Figure 4.8: Simple Signaling and Data Flow Example

Figure 4.8 shows one of the simplest possible signaling configurations. A data flow is flowing from the sender via different routers to the receiver. The two end-hosts and two of the routers (*R1* and *R2*) contain NEs that exchange signaling messages through the flow. *R3* does not contain an NE and forwards only the data. The signaling messages exchange is possible in both directions. Before a data flow is sent, an NSIS signaling procedure will take place along the NEs in the data path, including discovering their existence and signaling the application-specific states (e.g., firewall configurations for corresponding data traversal).

4.9.1 NSIS Layered Model Overview

In order to meet the modular requirements for NSIS, the NSIS protocol is structured in two layers:

- The *NSIS Transport Layer Protocol* (NTLP), which is responsible for moving signaling messages around and nevertheless independent from the underlying signaling application. The NTLP is implemented by *General Internet Signaling Transport* (GIST) [SH08]. Several open source implementations are available, e.g., *FreeNSIS* [FreeNSIS].
- The *NSIS Signaling Layer Protocol* (NSLP), which allows application based functionality, such as message formats and sequences.

Figure 4.9 illustrates this modular NSIS approach and the mutual influence between the NTLP and the NSLP.
Functionality within the NTLP is restricted only to transport- and lower-layer operations. Other operations are relocated to the signaling application layer. A short introduction of the NTLP can be described as follows.

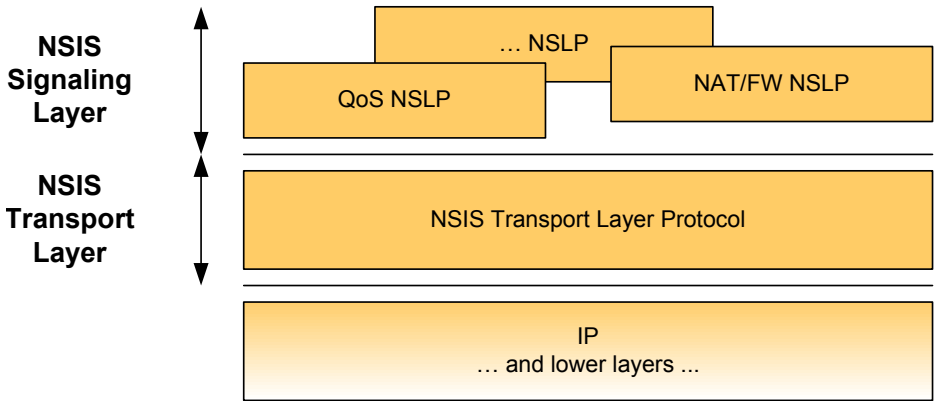


Figure 4.9: The NSIS Protocol Components

When an NSLP signaling message needs to be sent, the NSLP gives it over to the NTLP together with the information to which flow it belongs (so-called *flow identifier*). The NTLP has to care about how the message is sent to the next NE along the path. The NTLP does not need to have any knowledge about addresses, capabilities, or status of other NEs along the path, only of the NEs that it directly peers with. Having received an NSIS message, each intermediate NTLP either directly forwards it or, if the signaling application runs locally, passes the message to the NSLP for further processing. After processing, the NSLP can use the original message or generate another message and hand it over to the NTLP. Through this procedure end-to-end NSIS message delivery can be achieved. This restriction of the NTLP to peer-relationship scope simplifies the management and the complexity of the NTLP at the cost of an increased functionality, complexity of the NSLPs and deployment complexity, as some components (e.g., middleboxes) on the path need to run NSIS.

4.9.2 The NAT/Firewall NSLP Protocol

The IETF *NSIS working group* [NSIS] is currently finalising the *NAT/Firewall NSIS Signaling Layer protocol* (NAT/FW NSLP) specification [STAD08], which describes scenarios, problems and solutions for path-coupled network address translator and firewall signaling. The NAT/FW NSLP is one of the two NSLPs that the working group has been developing. Section 8.1.2 and [SPTF06] show that NSIS and the NAT/FW NSLP framework is able to support firewall signaling for up to tens of thousands of flows in parallel even in a low-end environment; and the overall performance bottleneck is the utilised firewall implementation, not on the signaling implementation.
The main goal of NSIS NAT/FW signaling is to enable communications between two endpoints across different networks or domains in case of the existence of NATs and

firewall middleboxes. Firstly, it is assumed that these middleboxes will be configured in such a way that NSIS NAT/FW signaling messages can traverse them. Then, the NSIS NAT/FW NSLP protocol is used to dynamically install additional policy rules in all NAT/FW NSLP-aware middleboxes along the path. Firewalls will be configured to forward desired data packets according to the policy rules which are established by the NAT/FW NSLP signaling.
The signaling traffic of an application behind a middlebox has to traverse all middleboxes along the data path to establish communication with a corresponding application on the other end-host. To achieve middlebox traversal, the application triggers the local NSIS entity to signal along the data path. If the local NSIS entity supports NAT/FW NSLP signaling, the knowledge of this application is used to establish policy rules and NAT bindings in all middleboxes along the path, which allows the data to travel from the sender to the receiver. Clearly, it is necessary for intermediate middleboxes to support NAT/FW NSLP, but not necessary for other intermediate nodes to support NAT/FW NSLP or even NSIS.
Figure 4.10 shows a common topology for the use of NAT/FW NSLP. This scenario is separated into two distinct administrative domains, namely *Domain A* and *Domain B*.

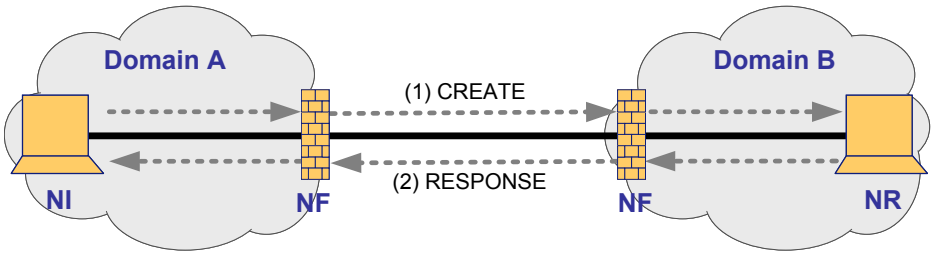


Figure 4.10: NAT/FW NSLP Firewall Traversal Scenario

The *NSLP Initiator* (NI) sends NSIS NAT/FW NSLP signaling messages along the data path to the *NSLP Responder* (NR). It is assumed that NI, NR and every intermediate middlebox implements the NAT/FW NSLP. Thereby, no knowledge about the next middlebox along the path is required; this is done by on-path next-hop discovery. The signaling messages reach different intermediate NSIS nodes (i.e., *NSLP Forwarder* or NF) and every NAT/FW NSLP node processes the signaling messages and, if necessary, installs additional rules for the following data packets. The NAT/FW NSLP supports several types of signaling messages, most notably the *CREATE* and the *EXTERNAL* messages:

- The *CREATE* message is sent from the source address to the destination address, processed by every middlebox and forwarded to the destination.

- The *EXTERNAL* (EXT) message is sent from the source address to an external address (e.g., the HA's address or the CN's address), is intercepted by the edge firewall and not forwarded to the destination address. This allows signaling pinholes at the edge-firewall without introducing long end-to-end signaling delays.

- The *RESPONSE* message is used as a response to CREATE and EXT request messages.

Policy rules for firewalls are represented by a common 5-tuple, namely the source and destination addresses, the transport protocol and the source and destination port, in addition to the *rule action* with the value *allow* or *deny*. Such a policy rule in NAT/FW NSLP is bounded to a specified session. Different from other signaling applications where policy rules are carried in one object, the policy rules in NAT/FW NSLP are divided into the *action* (allow/deny), the *flow identifier* and further information. The *Message Routing Information* (MRI) in the NTLP carries the *filter specification*; the additional information such as *lifetime*, *session ID*, *message sequence number*, *authorisation objects* and the specified *action* are carried in NSLP's objects.

4.10 Evaluation and Applicability for Mobile IPv6 Firewall Traversal

In the following subsections, all the aforementioned solutions will be analysed in detail regarding their applicability for Mobile IPv6 firewall traversal. As discussed before, the use of *UPnP* (Section 4.1) and *SOCKS* (Section 4.2) is limited to small installations and not fully applicable for Mobile IPv6 firewall traversal. Therefore, it is not necessary to study them further. *Application Layer Gateway* (Section 4.3) and *Middlebox Communication* (Section 4.4) will be evaluated together, as they share many similarities. *STUN, TURN and ICE* (Sections 4.6 to 4.8) are also evaluated together as only a combination of these approaches can be applied for Mobile IPv6 firewall traversal.
A middlebox traversal solution needs to meet several conditions and requirements to be applicable for Mobile IPv6 firewall traversal. They can be divided into several groups:

The first group, the *general requirements*, is fundamental for Mobile IPv6 firewall traversal, namely the support for IPv6, the support for IP mobility, in fact the support for Mobile IPv6, and the applicability for large networks and domains, e.g., for ISP networks or large company networks. A potential solution must support this requirements or either must be able to circumvent them.

The second group, the *security requirements*, is also fundamental, and thus must also be supported by the firewall traversal solution. Here, the support for authorisation, authentication and accounting interaction to achieve acceptable authorisation of Mobile IPv6 firewall traversal based on users' service profiles is required. In case of an explicit firewall traversal approach, the solution should also provide the possibility to protect the signaling, e.g. using secured tunneling. Obviously, an implicit approach does not need to provide this.

Thirdly, the *deployment requirements* depict whether the firewall traversal solutions requires additional entities or software to perform. A middlebox traversal solution may require additional network entities, additional software or application logic on the middleboxes, or additional software on sender and receiver. For easy deployment, a potential firewall traversal solution should depend on as less as possible deployment requirements.

Fourly, the *operational requirements* reflect the applicability in widespread operational environments. A firewall traversal solution should provide an easy setup and administration feasibility, as this increases the possibility to be adopted. Additionally, it should provide reliability, i.e., the approach should be reliable and should guarantee that the firewall pinhole creation progress, if is enforced, is successfully. Ideally, a firewall traversal solution may already be available, deployed and operational as widespread solution significant simplifies the deployment and acceptability.

Finally, the *performance and scalability requirements* should be satisfied. This includes low session setup time to be able to deal with fast handover, ideally, *on the fly* and to perform well under heavy load, e.g., to scale well even with thousands of simultaneous firewall traversal sessions. Other performance and scalability requirements are explicit session teardown, the cognition of broken sessions and thus contemporary deletion of the firewall pinholes of broken sessions, and adequate refresh timers for the session states.

In the following subsections, the potential firewall traversal solutions will be evaluated in detail regarding their applicability for Mobile IPv6 firewall traversal with help of this criteria.

4.10.1 Application Layer Gateway and Middlebox Communication

4.10.1.1 Evaluation and Applicability

The *Application Layer Gateway* (Section 4.3) and the *Middlebox Communication* (Section 4.4) approaches satisfy the *general requirements*. Both support IPv4 as well as

IPv6. ALGs and MIDCOM agents perform the application traffic and facilitate firewall traversal based on the protocol knowledge. However, these approaches do not support Mobile IPv6 by nature. To apply them for Mobile IPv6 firewall traversal, it is necessary to extend them to understand and support Mobile IPv6. This is mostly an implementation issue and does not circumvent the applicability. If the underlying network topology is complicated, it is difficult to configure every middlebox in data path to establish end-to-end connectivity, as the MIDCOM agents must be aware of the network topology. Therefore, firewall discovery is an important issue for the MIDCOM protocol, but this functionality is out of the scope of MIDCOM. However, if the approach wants to offer an end-to-end connectivity in practical large network, this functionality is necessary. Some NAT discovery methods are proposed in the IETF *MIDCOM working group*, and may be applied here. When utilise the ALG approach, firewall discovery is not an issues.

As the decision, whether a packet and the corresponding flow is enforced on the middlebox will be taken locally, the security depends on the local implementation, i.e. the AAA infrastructure. Therefore, these approaches have considered AAA issues. Other security mechanism will cooperate with them to implement AAA. Protected signaling is not required for the ALG and MIDCOM approaches. Albeit the described limitations, these solutions achieve the *security requirements* they are not fully apply to the technique they implement.

Whereas the ALG approach does not require additional hardware entities, the MIDCOM approach needs additional functionalities to accomplish protocol operations. These functionalities may reside in existing network entities or not. Furthermore, these approaches require additional software on each middlebox to offer the communication and configuration interface, and implement protocol knowledge to understand and analyse the communication at the application layer to successfully offer firewall traversal. So, the *deployment requirements* are noticeable as these approaches depend on significant modifications on the middleboxes, whereas the sender and receiver do not need to be changed.

These approaches meet the *operational requirements* as they provide easy setup and administration possibilities. Moreover, ALGs are widespread and used since many years which should increase the acceptability and the faster deployment. In contrast to other approaches, ALG and MIDCOM are reliable approaches, i.e., if the firewall pinhole creation progress is enforced and the AAA aspects are given, they can guarantee that the firewall pinhole creation progress is successful and do not rely on change.

The approaches also satisfies the *performance and scalability requirements*. When using ALG or MIDCOM to deal with middlebox traversal, the main latency of session

setup time may include middlebox discovery and state setup. However, the session setup time in general is much faster compared to several implicit and explicit solutions as the ALG or MIDCOM approaches provide middlebox traversal nearly *on the fly*. As already described, the performance of the middleboxes may limit the applicability of these protocols in large networks as they may become the bottleneck of these middlebox traversal protocols. With modern environments this is not a critical issue and thus these protocols are fully suitable. Both approaches do not provide the capability to teardown established sessions. Thus, they depend on the soft state approach of the middlebox and have to constantly refresh the state table as the middleboxes delete the firewall pinholes for broken, not refreshed, sessions after a specified session timeout.

4.10.1.2 Mobile IPv6 Application Layer Gateway

According to this evaluation, both, the Application Layer Gateway and Middlebox Communication approach are basically applicable for Mobile IPv6 firewall traversal. As ALGs are widespread and deployed since many years, they represent the better solution alternative for Mobile IPv6 firewall traversal. Moreover, a working ALG solution would ease the development of an MIDCOM based approach.
To summarise, Application Layer Gateways are one of the most promising solutions to deal with the problems and impacts of having firewalls in Mobile IPv6 environment. Therefore, one main contribution of this thesis was to design and develop an Application Layer Gateway based Mobile IPv6 firewall traversal solution. This leads to the *Mobile IPv6 Application Layer Gateway* [KSYB08, KSSB08], introduced in Section 5.4.

4.10.2 Policy-Based Networks

4.10.2.1 Evaluation and Applicability

Besides the fact that the *Policy-Based Networks* (PBN) [YPG00] approach is still not technically mature fully developed since it includes open issues that need to be further analysed, it achieves the *general requirements*. The PBN approach is not aware of the network layer, so it is applicable for both IPv4 and IPv6 networks. However, as theoretical framework, the PBN solution has currently no support for mobility, but requires some kind of mobility support in order to let the PEP (i.e., the firewall) parse Mobile IPv6 signaling and extract some information to discover the MSA/ASA. The PBN solution is based on the communication between PDP and MSA/ASA and all the required logic to allow or deny Mobile IPv6 traffic in the firewall is based on MN's profile stored in the MSA and on policies stored in the repository. The main constraint for being applied in large networks is the ability of both the PEP and the PDP to detect Mobile IPv6 traffic and to take decisions respectively about such traffic

as fast as required which depends on the hardware features and communications with the MSA/ASA.

The PBN approach implements the *security requirements*. It requires authentication and authorisation for both the PDP and the PEP in order to trust each other. There is not any protocol to be considered as standard for communication between the PDP and the PDP. There are several alternatives that can be considered like *Common Open Policy Service* (COPS) [DBC+00], *Simple Network Management Protocol* (SNMP) [LMS02], *SNMPv2* [Pre02], and *Session Initiation Protocol* (SIP) [RSC+02]. The most promising protocols are COPS and SIP as they provide good authentication and authorisation capabilities. However, some kind of extensions are required in order to let them transport PBN-specific parameters related to the allocation of the Mobile IPv6 pinhole in the PEP. Signaling between PDP and PEP is not standardised, so different alternatives are possible (COPS, SNMP or SIP). In general, some of them do not support security in the communications. Also some kind of authentication and authorisation is required to allow each PDP to query the MSA/ASA about the MN's profile. This part is more standardised and protocols like *RADIUS* [RWRS00] or *DIAMETER* [CLG+03] could be used, maybe with extensions to support specific parameters related to the allocation of the Mobile IPv6 firewall pinhole in the PEP.

The *deployment requirements* are remarkable as the PBN approach requires significant modifications on the middleboxes and additional hardware entities, but the sender and receiver do not need to be modified. Two new hardware entities are needed, the PDP and the policy repository. It also requires some protocol knowledge to allow the PEP functionality to work and to implement the PEP functionality in the firewall to allow the Mobile IPv6 traffic to traverse it.

The *operational requirements* can only be analysed unsatisfactory. How the policy is implemented is the main constraint about PBN. This is an open issue which needs to be addressed. Besides, the solution is only an approach under investigation without standardisation or implementations. Furthermore, there are some important open issues which need to be studied further like the discovery of MSA/ASA being contacted by the PDP, the discovery of the MN's identity and authorisation and authentication between the PEP and the PDP. Additionally, the key point for reliability is whether the PDP can find out the MSA/ASA being contacted to allow or deny the Mobile IPv6 traffic sent by the MN. This could not be done under some circumstances like using IPsec for ciphering Mobile IPv6 signaling between the MN and the HA, so it can not guarantee that the firewall pinhole creation process succeeds under every condition.

The PBN approach also satisfies the *performance and scalability requirements*. The key for fast session setup time besides the end-to-end delay is the queries to the policy repository which is assumed to be fast enough. Many alternatives are feasible for

implementing fast policy repository like databases. Under heavy load the firewall will have to parse a lot of Mobile IPv6 signaling and contact the PDP/MSA/ASA several times. This may take longer than expected and desired depending on the hardware features of both PEP (i.e., the firewall) and PDP, as well as the network conditions between the PDP and the MSA/ASA. The PBN does not have an explicit soft state refreshment, so the firewall pinhole is closed after timeouts from the last allowed Mobile IPv6 packets are sent.

4.10.2.2 Policy-Based Networks for Mobile IPv6 Firewall Traversal

Summing up, *Policy-Based Networks* (PBN) [YPG00] solution should be able to handle with the problems and impacts of having firewalls in Mobile IPv6 environments but contains problems and open issues which need to be studied further. As other approaches represents better solution alternatives, this subsection only briefly describes how PBN could be potentially used for firewall traversal in Mobile IPv6 environments. Having the Mobile IPv6 components in mind, the *Policy-Based Networks for Mobile IPv6 Firewall Traversal* solution should be based on policy decisions taken locally in each administrative domain; so every domain has its own PDP. The architectural components and the way how they interact are depicted in Figure 4.11.

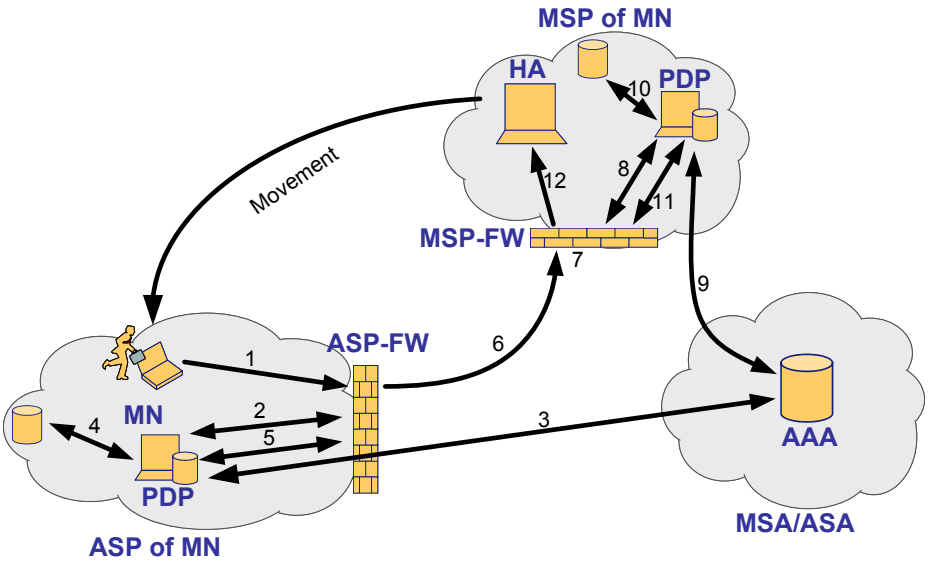


Figure 4.11: Policy-Based Networks for Mobile IPv6 Firewall Traversal

In general the policies, i.e. firewall pinholes, will be bound to the mobile node. Therefore, MN's identity will be one of the parameters to be checked by the PDP to decide whether this request is permitted. For this reason, the local PDP must communicate

with the MSA/ASA of the MN, as it is the only entity which knows the user's service profile and can therefore allow user-specific actions for all the PEPs involved in the communication. On the other hand, each administrative domain has its own PDP as well as the corresponding local policy repository. Thus, the domain administrator finally can manage which types of requests are permitted within this domain.
It is assumed that the firewall is configured to allow the IKEv2 signaling in order to let the MN establish a SA with the other peer. If this is not the case it is required to allocate a firewall pinhole in the firewall for IKEv2 signaling. That might also be achieved by using the PBN architecture in a similar way that the one used for opening the Mobile IPv6 firewall pinhole described below. In this case, the resource to be allocated is the IKEv2 signaling pinhole rather than the firewall pinhole for Mobile IPv6 traffic. Other alternatives might be also valid but they will not be further discussed. The most important point is that in case the firewall denys the IKEv2 signaling, it is required to allocate a firewall pinhole for it as previous step to the allocation of the firewall pinhole for Mobile IPv6 traffic. Otherwise Mobile IPv6 traffic will not be able to traverse the firewall.
Figure 4.11 shows the operation where the mobile node has already established an IKEv2 SA with the home agent and wants to start to communicate with the HA in order to send binding updates to it.
In this scenario all involved domains deploy their own firewall which must provide firewall pinholes to allow the Mobile IPv6 traffic to traverse. Such firewalls are considered the PEPs according to the PBN approach. Each PEP interacts with the local PDP, located in the same network domain as the firewall, to accept or to deny the Mobile IPv6 traffic.
The sequence to configure the middleboxes to support Mobile IPv6 firewall traversal could be as follows (compare Figure 4.11:

1. As the IKEv2 SA is assumed to being already configured, the MN sends the BU to the HA and the Mobile IPv6 packet is intercepted by the firewall. This triggers the event that the firewall (PEP) collects certain parameters related to the packet, e.g., type of the packet, source address (CoA) and destination address (HA's address).

   The main issue here is that, if IPsec is used for protecting BU/BA, it is impossible for the firewall to extract the identity of the MN from the encapsulated BU (i.e., it's *Network Access Identifier* (NAI) [PLK+05]). Therefore the PDP is not aware of the MN's home domain and its MSA/ASA to be contacted for checking MN's credentials and authorisation.

   This issue seems to have no solution unless some kind of correlation between the HA address and the MSA/ASA can be established. Some heuristics might be applied, like HA address reverse resolution which could provide domain information about the MSP and afterwards extracting the MSA/ASA by using DNS

SRV records configured for the MSP domain. However, this solution seems to be quite unrealistic as it requires the coordination and configuration of many agents, like MSP, DNS reverse resolution for HA, and DNS SRV RR for MSA/ASA.

In case the *Authentication Protocol* [PLK+06] is used rather than IPsec to protect Mobile IPv6 signaling between MN and HA, the BU is not ciphered and has to include the NAI of the MN. This provides the PEP the opportunity to extract the MN identity as well as its domain which allows to figure out the MSA/ASA easily.

2. The firewall works as PEP and has an interface to communicate to the PDP located at its domain. The PEP asks the PDP whether the Mobile IPv6 traffic from the MN is allowing or not. Some kind of authentication and authorisation should be done between the PEP and the PDP when asking the PEP for the decision or when the PDP forces the configuration of a firewall pinhole on the PEP.

3. Based on the information provided by the PEP (e.g., HA address or MN CoA), the PDP figures out the MSA/ASA regarding the MSP, as described in step 1. The PDP contacts the MSA/ASA to validate the MN's right to use the mobility service. The MN's profile is stored in the MSA/ASA, thus it is the only entity that can provide information about the rights that the MN owns. To work in a secure manner, the MSA/ASA only sends information about the MN's profile if the PDP is properly authenticated and authorised.

4. If the MSA/ASA authorises the MN to use the mobility service, the PDP may collect local policies related to the foreign node (MN). With the MSA/ASA notification and the local policies, the PDP makes a decision about allowing or denying Mobile IPv6 traffic coming from or going to the MN. If either or both the MSA/ASA or local policies do not allow Mobile IPv6 traffic for the MN and/or MSP, the PDP does not force the firewall pinhole allocation on the PEP and the process stops.

5. The PDP translates the policies into firewall pinhole rules to be applied at the firewall located at the ASP.

6. The Mobile IPv6 traffic is now able to traverse through the firewall and is forwarded to the HA.

7. The incoming Mobile IPv6 traffic triggers an event at the firewall located at the MSP to collect the parameters related to such the packet, e.g., type of the packet, source addresses (CoA and HoA) and destination address (HA's address). The firewall might contact local databases to get some of these parameters.

8. The firewall works as PEP and has a interface to communicate to the PDP located at its domain. The PEP asks the PDP whether the Mobile IPv6 traffic from the MN is allowed or not. Again some kind of authentication and authorisation should be done between the PEP and the PDP when asking the PEP for the decision or when the PDP forces the configuration of a firewall pinhole on the PEP.

9. In this case, the PDP is within the MSP domain and is directly connected to MSA/ASA. The PDP contacts the MSA/ASA to validate whether the MN has the rights to use the mobility service. In order to work in a secure manner, the MSA/ASA sends information about the MN's profile if the PDP is authenticated and authorised.

10. If the MSA/ASA authorises the MN to use the mobility service, it makes a decision whether the Mobile IPv6 traffic coming from or going to the MN is allowed or not. If either or both the MSA/ASA or local policies do not allow the Mobile IPv6 traffic for the MN and/or ASP, the PDP does not force the firewall pinhole allocation on the PEP and the process stops here.

11. The PDP translates the policies into firewall pinhole rules to be applied at the firewall located at the MSP.

12. The Mobile IPv6 traffic is now able to traverse through both, the ASP and MSP firewalls and the MN and HA can communicate.

This procedure allows the MN to communicate with the HA and vice versa but potentially could be extended to also allow the MN to communicate with the CN and the HA to communicate with the CN. Thus, it could be applied to allow Mobile IPv6 firewall traversal.
The main advantage of this middlebox traversal approach is that the decision whether a firewall pinhole request is allowed or not takes place within the domain where the firewall is located. The decision is not only based on the MN's profile located at the MSA but is also based on local policies stored within the domain.
This PBN approach for Mobile IPv6 middlebox traversal has the following requirements:

- If IPsec is used to cipher Mobile IPv6 signaling, some mechanism to find out the MSA/ASA is required.

- The firewall has to be aware of Mobile IPv6 protocol details if the policies consider properties like *Mobility Header* (MH) [JPA04], HoTI, HoT. Furthermore, the firewall also must have PEP capabilities.

- Policies are an important element and have to be clearly specified in order to let the Mobile IPv6 traffic accepted properly. They must be designed for accepting or denying Mobile IPv6 traffic based on the criteria defined by network administrators.

- Some kind of coordination has to be done among entities (ASP/MSP/MSA/ASA) to allow the respective PDPs to query the MSA/ASA about the MN's profile.

- Communication between firewalls (PEPs) and PDP must be secured which means being both authenticated and authorised.

4.10.3 STUN, TURN and ICE

4.10.3.1 Evaluation and Applicability

The STUN (Section 4.6), TURN (Section 4.7) and ICE (Section 4.8) approaches successfully solve the *general requirements*. All these protocols work for both IPv4 and IPv6 networks but do not have support for mobility. As it is necessary to know the required mobility parameters to initiate the firewall traversal, some methods need to be developed to obtain these parameters for firewall traversal. However, when utilise UDP encapsulation, this is not needed. STUN, TURN and ICE are already widely used and do not have any specific constraints in large networks.

The STUN, TURN and ICE approaches do not achieve the *security requirements*. These approaches do not have any explicit signaling traffic and do not need protected signaling. The most important issue with this implicit approaches is that they are reliable but can not ensure the firewall traversal progress (in the sense that it has been specifically designed mostly for NAT traversal it may not work for certain firewall scenarios especially for mobile environments) and has insufficient support authorisation, authentication and accounting.

The *deployment requirements* of these approaches are very small as only some additional network entities are needed, e.g., STUN- or TURN-servers, as well as additional software on both sender and receiver side. There is no need to deploy additional software or protocol knowledge on the middleboxes.

STUN, TURN and ICE are widely spreaded, well accepted and the software is available. So a potential solution could be easily implemented. Additionally, setup and administration of STUN, TURN and ICE is easy for both, the users and administrators, and thus this *operational requirements* are fulfilled. However, the approaches

rely on the right behaviour of the involved middleboxes, and therefore, can not guarantee that the firewall pinhole creation process succeeds but might fail under certain conditions.

The *performance and scalability requirements* of these approaches are difficult to analyse and evaluate. These approaches require firewall discovery and active communication with the STUN- and TURN-servers, which highly effects the session setup time, an important issues, especially in mobile environments. However, there are no visible constraints under heavy load, but the performance of the STUN- and TURN-server may become a bottleneck. The STUN, TURN and ICE solutions are not explicitly signaled and the timeout of the firewall pinholes relies on soft state approach of the present middlebox implementation and configuration.

4.10.3.2 Mobile IP Interactive Connectivity Establishment

The STUN, TURN and ICE frameworks represent one potential solution alternative for Mobile IPv6 firewall traversal. On the one hand, this solution is a promising candidate as the *deployment requirements* and some *operational requirements* are very small and the protocols are already widely deployed. However, these protocols have several disadvantages, most notable the insufficient support for authorisation, authentication and accounting.
Recently, the *Mobile IP Interactive Connectivity Establishment* (M-ICE) [Tsc08], that is based on the *Interactive Connectivity Establishment* (ICE) [Ros07] methodology, has been proposed. ICE is based on STUN [RWHM03] and TURN [RMM08]. With M-ICE, the ICE framework is applied to Mobile IPv6. M-ICE uses STUN for connectivity checks, a modified return routability procedure and UDP encapsulation of the signaling traffic as described in [Baj08]. After performing M-ICE, two Mobile IPv6 end-points will be able to communicate directly or through a relay via NATs, *Network Address and Port Translators* (NAPTs) and firewalls.
Firstly, M-ICE gathers the MN's candidates and afterwards signals them to the CN by including them in the CoTI-ICE message. When the CN receives this message, M-ICE starts to gather its candidates and provides them in the CoT-ICE message. At that time both nodes could pair these candidates up and start connectivity checks by using STUN. After finishing this, they both have a prioritised list of working candidate pairs.
ICE works reliably, as it is widely used for VoIP. For interacting with middleboxes, an extension to STUN has been proposed that enables STUN-aware middleboxes to participate in the signaling exchange, see [WRT07]. Authorisation functionality has been proposed with [Win06].
M-ICE is a very recent proposal, however, it introduces a long delay for gathering the candidates. Obviously, this limits its applicability for seamless handover with Mobile

IPv6 firewall traversal. Furthermore, neither a detail evaluation nor a prototype implementation of M-ICE is currently available.

4.10.4 NSIS and the NAT/Firewall NSLP

4.10.4.1 Evaluation and Applicability

The *Next Steps in Signaling* (NSIS) [HKLdB05] and the *NAT/Firewall NSIS Signaling Layer Protocol* (NAT/FW NSLP) [STAD08] based approaches satisfy the *general requirements*. They support IPv4 and IPv6 and NSIS (including NAT/FW NSLP) is designed for wide-area or global Internet usage. Therefore, both are fully applicable for large networks. Admittedly, the current NSIS NAT/FW NSLP does not explicitly support Mobile IPv6. However, it should be easy to add this support by either extending it to support Mobile IPv6, or make it be able to interact with Mobile IPv6.

NSIS and NAT/FW NSLP already support an authorisation framework [MSTB08] and signaling traffic protected by SCTP [FDC08]. In general, each network- or transport-layer security protocol should be applicable for NSIS. Hence, this approach fully fulfills these *security requirements*. Besides, as an explicit firewall traversal approach, NSIS and the NAT/FW NSLP should be reliable, as they rely on soft state and not rely on a centralised entity. However, it has not been experimented in the Internet scale or large deployments.

The *deployment requirements* of the NSIS based approach are remarkable as they require NSIS and NAT/FW NSLP to run on sender- and receiver-side and on all intermediate middleboxes. So, significant modifications on the middleboxes are needed, though, as an explicit firewall traversal approach, it is not required that a middlebox understands the protocol logic of the application. Additionally, NSIS do not need any additional network entities or special hardware.

In general, the setup and administration for NSIS and NAT/FW NSLP should be easy, as they only need information from local interfaces and the routing table. NSIS and the NAT/FW NSLP have been implemented, e.g., by the University of Göttingen [FreeNSIS] but have not been experimented in the open Internet or large ISP deployments yet. Therefore, it is available but it is difficult to draw conclusions for *operational requirements*.

This approach is an explicit firewall traversal solution and the path needs to be signaled before the data traffic. Additionally, NSIS requires path discovery before the NSLP signaling can be done and therefore might have an increased session setup time. So, the session setup time for this approach is higher then for an *on the fly* approach. However, it has been proven that the utilised middlebox implementation becomes the

bottleneck of a "heavy-load-scenario" not NSIS or the NAT/FW NSLP, as described in Section 8.1.2 and in [SPTF06]. As a soft state protocol, all NSIS and NAT/FW NSLP states need to be refreshed during a specified time value, otherwise the session state times out and will be deleted. So, the *performance and scalability requirements* are achieved.

4.10.4.2 NSIS Based Mobile IPv6 Firewall Traversal

According to this evaluation, an NSIS and NAT/FW NSLP based approach is fully applicable for Mobile IPv6 firewall traversal.
Mobile IPv6 firewall traversal based on NSIS and NAT/FW NSLP is one of the most promising solutions to deal with the problems and impacts of having firewalls in Mobile IPv6 environment. One main contribution of this thesis is the development of the *NSIS based Mobile IPv6 firewall traversal* solution, introduced in Section 5.3.

4.11 Summary

The potential middlebox traversal solutions investigated could be summarised – if feasible – in Table 4.1 according to the above evaluation results. From this table it can be seen that each solution has its advantages and disadvantages in dealing with firewall traversal problems in a Mobile IPv6 environments, whereas the light-grey fields represent small disadvantages which are more or less easy to overcome and the dark-grey fields represent major issues or disadvantages.
The overall evaluation can be concluded as follows:
Although the *Policy-Based Networks* approach is applicable and has been used in some network scenarios, it do not offer properties like the support for Mobile IPv6 traffic and has disadvantages and some open issues. *STUN, TURN, and ICE* offers a good potential candidate, however, like all solution alternatives, it also has some disadvantages and conditions. [Tsc08] and [Baj08] recently proposed the *Mobile IP Interactive Connectivity Establishment* (M-ICE), an ICE based solution to deal with the Mobile IPv6 firewall traversal problems, but this approach introduces a long delay for gathering the candidates. This limits it applicability for seamless handover with Mobile IPv6 firewall traversal.
The *NSIS and NAT/FW NSLP* based solution, as well as the *Application Layer Gateway* based solution are both applicable to overcome the problems and impacts when having firewalls in Mobile IPv6 environment. According to the evaluation and applicability study, they represent the most promising solution alternatives for Mobile IPv6 firewall traversal, and this thesis therefore focuses on these two candidates.
After having presented the state-of-the-art approaches for middlebox traversal and evaluating their applicability for Mobile IPv6 firewall traversal in this chapter, the

	ALG, MIDCOM	PBN	STUN, TURN, ICE	NSIS and NAT\FW
General Requirements				
IPv6 Support	Yes	Yes	Yes	Yes
Mobile IPv6 Support	No	No	Not required	Not required
Applicability for large Networks	Yes	Yes	Yes	Yes
Security Requirements				
Authorisation, Authentication, and Accounting Issues	Limited	Yes	Limited	Yes
Protected Signaling	Not required	Yes	Not required	Yes
Deployment Requirements				
Additional Network Entities/Hardware	No	Yes	Yes	No
Additional Software on Middleboxes	Yes	Yes	No	Yes
Additional Application Logic on Middleboxes	Yes	Yes	No	Yes
Additional Software on Sender and Receiver	No	No	Yes	No
Operational Requirements				
Easy Setup and Administration	Yes	No	Yes	Limited
Reliability	Yes	Yes	Limited	Yes
Available, deployed and operational	Yes	Limited	Yes	Limited
Performance and Scalability Requirements				
Low Session Setup	Yes	Yes	Limited	Limited
Performance/Scalability	Yes	Limited	Yes	Yes

Table 4.1: Evaluation and Applicability Summary

following Chapter 5 now presents how they can be utilised to achieve Mobile IPv6 firewall traversal.

5 Proposed Mobile IPv6 Firewall Traversal Solutions

5.1 Motivation and Overview

Several aspects could prevent Mobile IPv6 from operating successfully in the presence of firewalls; these problems and impacts are introduced in Chapter 3. This could be a major impediment to the successful deployment of Mobile IPv6. To overcome these problems, this chapter presents two different solutions, appropriate to allow Mobile IPv6 to work in presence of middleboxes. Both have been designed and developed as main contribution of this thesis and are standardised within the IETF [IETF]. Section 5.3 describes the *NSIS based Mobile IPv6 firewall traversal* solution, which has been developed within the *EU Framework Programme 6* [EUFP6] *ENABLE* [ENABLE] project. Section 5.4 describes the second solution, based on *Application Layer Gateway*. This solution was developed within the IETF *Mobile IPv6 Firewall Traversal Design Team*, established by the IETF *MEXT working group* [MEXT]. The *Mobile IPv6 Application Layer Gateway* solution is a MEXT working group draft.

5.2 Framework

When utilising NSIS and the NAT/FW NSLP or an Application Layer Gateway approach for Mobile IPv6 firewall traversal and having the Mobile IPv6 components in mind, the generic Mobile IPv6 firewall traversal scenario looks like depicted in Figure 5.1.
The two approaches have different requirements and assumptions to allow Mobile IPv6 firewall traversal in this framework. For example, the NSIS based Mobile IPv6 firewall traversal requires the MN, HA and CN to support NSIS, the NAT/FW NSLP and a modified Mobile IPv6 implementation, whereas in the Mobile IPv6 Application Layer Gateway solution, MN, HA and CN do not need to be updated. The NSIS based Mobile IPv6 firewall traversal requirements and assumptions are described in Section 5.3.1, the Mobile IPv6 Application Layer Gateway requirements and assumptions are described in Section 5.4.1. However, both solutions are applicable to work in the generic Mobile IPv6 firewall traversal scenario as shown in Figure 5.1.

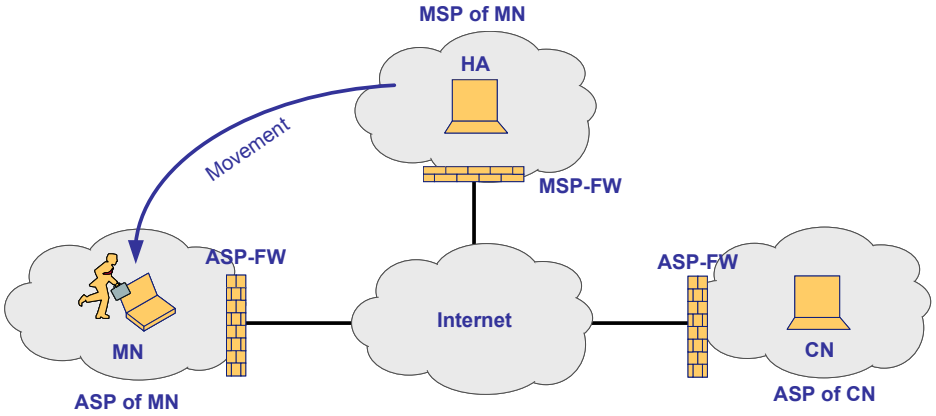


Figure 5.1: Generic Mobile IPv6 Firewall Traversal Scenario

5.3 NSIS Based Mobile IPv6 Firewall Traversal

Section 4.9 already introduced the *Next Steps in Signaling* (NSIS) [HKLdB05] framework, developed with the goal to support various signaling applications, which installs and manipulates certain control states in the network. It also introduces the IETF *NAT/Firewall NSIS Signaling Layer Protocol* (NAT/FW NSLP) [STAD08], which describes scenarios, problems and solutions for path-coupled network address translator and firewall signaling. The main goal of NSIS NAT/FW NSLP signaling is to enable communications between two end-points across different networks or domains in case of the existence of NATs and firewall middleboxes. Therefore, the NSIS NAT/FW NSLP protocol is used to dynamically install additional policy rules in all NAT/FW NSLP aware middleboxes along the path. Firewalls will be configured to forward desired data packets according to the policy rules which are established by the NAT/FW NSLP signaling.
As described in Chapter 3, the standard Mobile IPv6 does not work in presence of firewalls. To tackle this issue, one possible approach is to utilise a signaling protocol for install some firewall pinholes rules which allow these Mobile IPv6 messages to traverse the firewalls. The *NSIS* [SH08] *NAT/FW NSLP*, as described in [STAD08], allows an end system to establish, maintain and delete middlebox state (i.e., firewall pinholes rules), and thus permits the packets to traverse these middleboxes. The following subsections describe how the NSIS NAT/FW NSLP could be utilised and extended to address the aforementioned problems. This approach has been submitted to the IETF in [SFL07] and also proposed at [SFH+07].
Therefore, Section 5.3.1 firstly presents the requirements and assumptions, the essential extensions (Section 5.3.3) as well as the generic architecture (Section 5.3.4). Secondly, Section 5.3.5 discusses the scenarios and the corresponding problems as de-

scribed in Chapter 3 and presents how the different scenarios can be solved by using NSIS and the NAT/FW NSLP. Whereas Section 5.3.5 only describes how Mobile IPv6 firewall traversal can be accomplished with the current NSIS and NAT/FW NSLP specification and the small extensions presented in Section 5.3.3, Section 5.3.6 introduces some major improvements for the *NAT/FW NSLP Proxy Mode* which allow the Mobile IPv6 firewall traversal based on NSIS to significantly reduce the requirements and to perform faster by utilising the *Improved NAT/FW NSLP Proxy Mode*. Finally, Section 5.3.8 summarises the *NSIS based Mobile IPv6 firewall traversal* approach.

5.3.1 Requirements and Assumptions

The applicability of NSIS and the NAT/FW NSLP for Mobile IPv6 firewall traversal subjects to several requirements and assumptions:

- All networks and nodes MUST support IPv6; IPv4/IPv6 interworking as well as dual-stack solutions are out of scope and not supported.
- All involved Mobile IPv6 nodes, i.e., MN, HA and CN (as depict in Figure 5.1), MUST support NSIS, the NAT/FW NSLP and a modified Mobile IPv6 implementation able to interact with NSIS and the NAT/FW NSLP to perform Mobile IPv6 firewall traversal signaling.
- All firewalls at the edge of the different networks or domains, as well as all potentially intermediate middleboxes MUST support NSIS, the NAT/FW NSLP and MUST implement the *netfilter/ip6tables* [netfilter] module for performing middlebox functionality. In addition, ip6tables must be able to analyse the *IPv6 Mobility Header*, the *IPv6 Routing Header* and the *IPv6 Destination Options Header*.
- The firewalls MUST be configured in such a way, that NSIS NAT/FW NSLP signaling messages can traverse them. This has to be done by manual preconfiguration.
- For authentication and authorisation aspects, MSA and MSP MUST be colocated.
- The NAT/FW NSLP on all nodes MUST support the required extensions as described in Section 5.3.3.
- In this scenario, firewalls at the edge of the networks CAN be deployed, however, they MUST NOT be deployed as it is the case for other solutions.

5.3.2 The NAT/Firewall NSIS Signaling Layer Protocol

Section 4.9 introduces the IETF *NAT/Firewall NSIS Signaling Layer Protocol* (NAT/FW NSLP) [STAD08]. However, to utilise the NAT/FW NSLP protocol for Mobile IPv6 firewall traversal it requires several extensions. To understand this necessarily, a more detailed understanding of the NAT/FW NSLP is necessary. Therefore, this section introduces the NAT/FW NSLP in more detail, especially the protocol framework, the protocol operations, the protocol messages and protocol message objects. Afterwards, Section 5.3.3 presents the required NAT/FW NSLP extensions.

5.3.2.1 Protocol Overview

As described in Section 4.9, the NSIS NAT/FW NSLP is carried over the *NSIS Transport Layer Protocol* (NTLP) defined in [HKLdB05]. The interworking with the NTLP and the other components is shown in Figure 5.2. A normal NAT/FW NSLP messages is initiated by the *NSLP Initiator* (NI), may be handled by *NSLP Forwarder* (NF) and finally be processed by the *NSLP Responder* (NR). Therefore, it is required that at least the NI and NR implement the NAT/FW NSLP; NFs need only to implement the NAT/FW NSLP, if they provide middlebox functionality. NSIS nodes without the NAT/FW NSLP functionality only forward the packets.

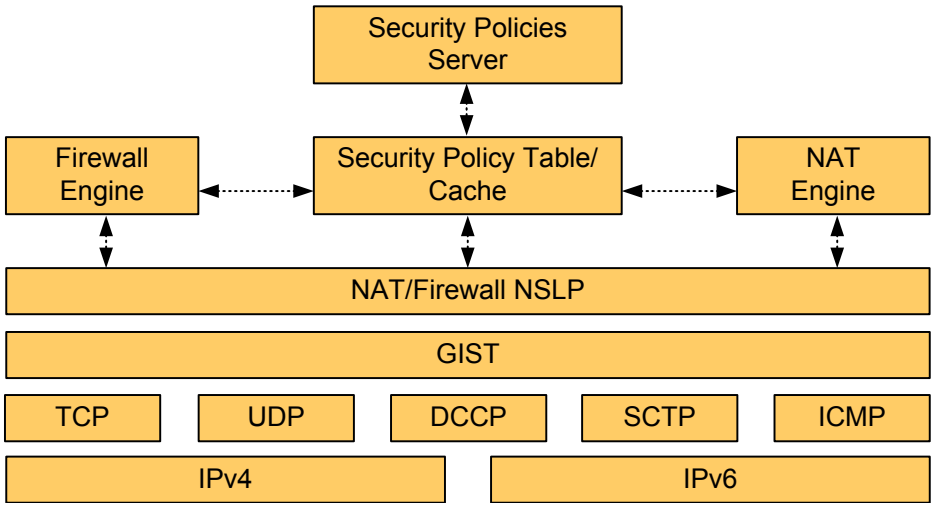


Figure 5.2: NAT/Firewall NSIS Signaling Layer Protocol Interworking

Figure 5.3 shows a common topology for the use of NAT/FW NSLP. The following sequence shows a typical NAT/FW NSLP event procedure:

- The NSLP initiator generates NAT/FW NSLP messages (e.g., a CREATE message) and sends those to the NSLP responder. The NI must not necessarily

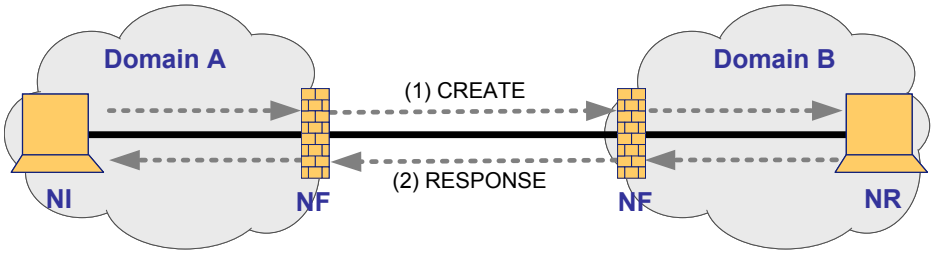


Figure 5.3: NAT/FW NSLP Firewall Traversal Scenario

be the data sender; when using the *Reserve External Address* signaling, the NI must be the data receiver.

- Each traversed NSLP forwarder which supports the NAT/FW NSLP processes the NSLP messages, checks their local policy rules and forwards the messages toward the next NSIS node until the messages reach the NR.
- The NSLP responder checks the received messages, processes them, generates RESPONSE messages and sends them back to the NI over the same NF chain. That is guaranteed via the established reverse message routing state in the NTLP. In some scenarios the NR may not necessarily be the data receiver.
- Each traversed NF which supports NAT/FW NSLP processes the RESPONSE messages again.
- When the NI received a successful response, the data sender can start to send data flow to the data receiver.

This NAT/FW NSLP behaviour enables communication between the end-hosts but only for scenarios in which there are only firewalls on the path or NATs on sender side. The NAT/FW NSLP also supports scenarios with NATs on the receiver side. The above described appropriation assumes that both end-hosts are NSIS aware and fails when only one end is NSIS aware. The NAT/FW NSLP partly supports those scenarios by using a so-called *Proxy Mode*. As NAT traversal is not required in Mobile IPv6 environment, this thesis only focuses on the firewall part.
The basic functionality of the NAT/FW NSLP is to enable data flows to traverse firewalls and NATs with the help of opening firewall pinholes and creating NAT bindings on these middleboxes. This is needed as firewalls normally work on *deny-all* policies; they block all traffic that does not match any firewall filter rule or any existing firewall state for established sessions. NATs normally block all traffic that does not match any existing configured or installed binding or session. In contrast, some scenarios require to support *allow-all* policy firewalls. That means, that only explicitly blocked

traffic is not allowed to traverse these firewalls. The NAT/FW NSLP also provides the capability to install deny policy rules at firewalls.
NSIS and the NAT/FW NSLP protocol works on soft states, so every installed state on NAT/FW NSLP aware NSIS nodes will de-installed or teardowned after a specified period of time if it has not been refreshed in the meantime. Only a just-in-time received session extension request can prevent this behaviour. To delete no longer needed NAT/FW NSLP reservations or sessions, the NAT/FW NSLP provides also an explicit state deletion mechanism.

5.3.2.2 Protocol Operations

This subsection presents a short overview about the message types used by the NAT/FW NSLP. It later describes the important NAT/FW NSLP protocol operations, i.e., how to create a session, keeping it alive and how to delete it. Furthermore, it provides an overview about the mechanisms to reserve the external address.

The NAT/FW NSLP protocol utilities four messages:

- *CREATE*: The *CREATE* message is the most important component of the NAT/FW NSLP messages. It is a request message sent from the source address to the destination address, processed by every middlebox and forwarded to the destination. Thus, it could be used to create, refresh and delete NAT/FW NSLP sessions on all middleboxes along the path from source to destination.
- *EXTERNAL*: The *EXTERNAL* (EXT) request is basically used for reserving an external address and port number. As the EXT message is sent from the source address to an external address (e.g., the HA's address or the CN's address), is intercepted by the edge firewall and not forwarded to the destination address, it could also be used to create, refresh and delete NAT/FW NSLP sessions at edge-firewalls without introducing long end-to-end signaling delays.
- *RESPONSE*: The *RESPONSE* message is used to respond to CREATE, EXT and NOTIFY messages.
- *NOTIFY*: An asynchronous message which is used by NAT/FW NSLP aware NEs to alert upstream and downstream NAT/FW NSLP NEs about specific events, notably in case of failures.

The important operations will be described in more detail in the following subsections.

5.3.2.2.1 Message Types Overview

It is advantageous to show the basic message types and message formats at first. A NAT/FW NSLP message is composed of a *Common NSLP Header* and one or

more objects, which follow the header. *"The NSLP header is carried in all NAT/FW NSLP messages and objects are Type-Length-Value (TLV) encoded using big endian (network ordered) binary data representations. Header and objects are aligned to 32 bit boundaries and object lengths that are not multiples of 32 bits must be padded to the next higher 32 bit multiple. The whole NSLP message is carried as payload of a NTLP message."* [STAD08] The NSLP Header is the first part of the message and contains four fields, the *NSLP Message Type*, the *P flag* and two reserved fields. The message types are similar to the four messages types described before. The *P flag* is used to indicate the usage of the *Proxy Mode*; the reserved fields must be set to zero and are used in other NSLPs as a flag field. The total length is 32 bits. Figure 5.4 shows the *Common NSLP Header*.

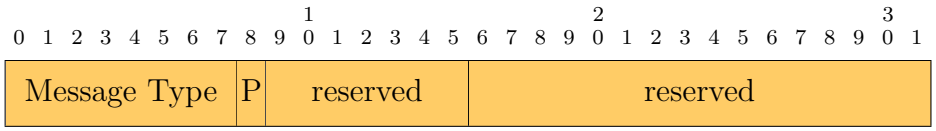


Figure 5.4: Common NSLP Header

Each NAT/FW NSLP object uses the *Common NSLP Object Header* format which is illustrated in Figure 5.5. The *Common NSLP Object Header* contains two fields, the *Object Type* and the *Object Length* and additional flags. In total it is 32 bits long.

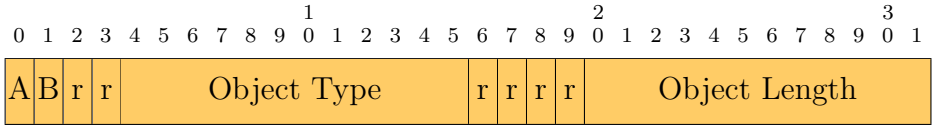


Figure 5.5: Common NSLP Object Header

The possible values of *Object Type* and *Object Length* are described at the end of this subsection. The *Object Length* is the total length of the object without the *Common NSLP Object Header*. Fields marked with "r" are reserved for future use. The first two bits are used to signal the favoured treatment for objects which are not defined in the current framework. The NAT/FW NSLP reuses parts of the categories which are defined in GIST [SH08].

- AB=00 ("Mandatory"): If the object is not recognised, the entire message containing it must be rejected with an error RESPONSE.
- AB=01 ("Optional"): If the object is not understood, it should be deleted and then the rest of the message processed as usual.

- AB=10 ("Forward"): If the object is not understood, it should be retained unchanged in any message forwarded as a result of message processing, but not stored locally.
- AB=11 This combination must not be used and an error RESPONSE must be generated.

Defined NAT/FW NSLP objects are:

- Signaling Session Lifetime Object, Length 1.
- External Address Object (IPv4/IPv6), Length 2/5.
- Extended Flow Information Object, Length 1.
- Information Code Object, Length 1.
- Nonce Object, Length 1.
- Message Sequence Number Object, Length 1.
- Data Terminal Information Object (IPv4/IPv6), Length 3/6.
- ICMP Types Object, Length variable.

For more details to NAT/FW NSLP messages and NSLP objects check [STAD08] or A.3.

5.3.2.2.2 Session Creating

The NAT/FW NSLP uses the CREATE request to establish data flow between two end-hosts in attendance of middleboxes. Therefore, the NI generates a CREATE message and hands it over to the NTLP. The NTLP forwards the message to NR. Forwarding is managed hop-by-hop and is transparent for NFs which do not implement NAT/FW NSLP and non NSIS aware routers between NSLP hops. Each NAT/FW NSLP aware NF along the path processes the message, but can also reject those messages based on local policy rules. When the message reaches the NR and the NR accepts it, it creates a RESPONSE message to this request and the response is forwarded hop-by-hop back to the NI. Figure 5.6 displays the message flow in case of session creating.
Furthermore, the CREATE message is not only used for session creating. Combined with the lifetime object it is used for several purposes, for session creating, session modification, session refreshing and session deleting. So, the processing methods for a CREATE message depend on the message function, the lifetime and on the node at which the processing happens. The following shortly displays how a CREATE message is processing in the different NSIS node types:

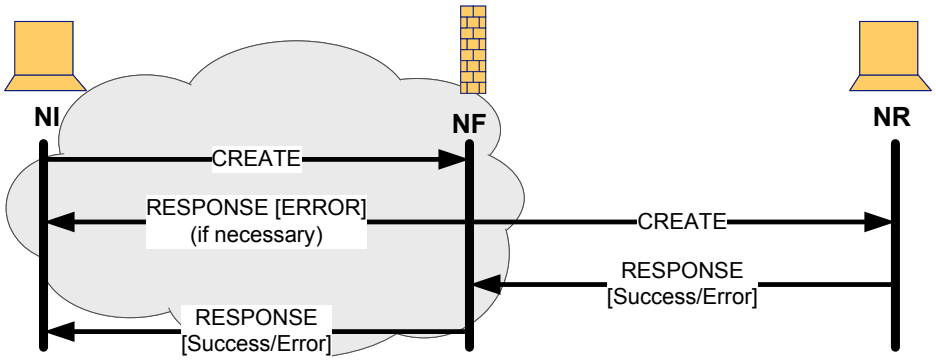


Figure 5.6: Create Message Flow

- NAT/FW NSLP Initiator: The NI only generates the initial CREATE message and hands it over to the NTLP. If the NI receives a successful response message, the session is established and the data path is configured to send data messages to the NR. If the NI receives an error RESPONSE message, it may try to generate the CREATE message again or send a failure report to the application.

- NAT/FW NSLP Forwarder (only firewall): Every NFs which receives an initial CREATE message must first check authentication and authorisation with its local policies. This authentication and authorisation bases on the combination of the NTLPs *Message-Routing-Information* (MRI) and the CREATE payload. If a firewall receives an initial CREATE message, the NSLP remembers the requested policy rules, but does not install the policy rules at this time. Afterwards, the message is forwarded to the next hop. After receiving a successful response to such a request, the NSLP locally installs the before remembered policy rules.

- NAT/FW NSLP Receiver: If a NR receives an initial CREATE message it first must check authentication and authorisation. If authentication and authorisation is successful and the request is accepted, the NR must reply with a success RESPONSE message. If authentication and authorisation fails, the NR should reply with an error RESPONSE message. The RESPONSE message is sent back hop-by-hop towards the NI, so every intermediate NF process it.

5.3.2.2.3 Reserving External Addresses

The above described CREATE-mechanism works also well in presence of NATs and firewall on the path, if the data sender is located behind a NAT. It is catchier if the data receiver is located behind a NAT (compare Figure 5.7), as the NSIS firewall signaling must know the destination address in advance and may also be able to

reach it. The NAT/FW NSLP solves this problem by using a common peer-to-peer networking approach, where the two end-points learn each others IP addresses with the help of a third party, so-called *Application Server*. Figure 5.7 describes a typical peer-to-peer networking environment where a third-party helps two end-points to learn of each others IP addresses.

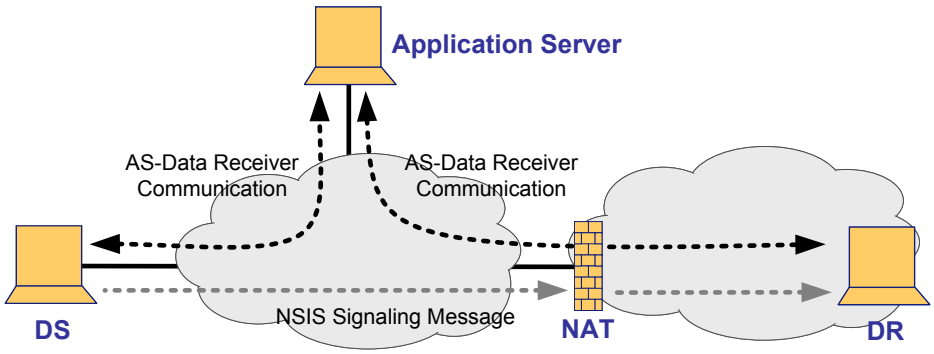


Figure 5.7: Data Receiver behind NAT

If the data receiver is located behind a NAT, a signaling message will be addressed to the allocated IP address at the NAT. If no NAT forwarding state exists, the signaling message will terminate at the NAT device, because the message transmitted by the data sender can not install the NAT binding there and the data sender has to know the detailed topology of the data receiver side. So, data receivers behind a NAT must reserve an external IP address and port number to enable a following CREATE message. When using the *Reserve External Address* signaling, the NSIS initiator is typically the node that will become the data receiver for the upcoming data flow. To distinguish this initiator from the normal case, where the NI is associated with the data sender, the NI is denoted by *NI+* and the NSIS responder is similarly denoted by *NR+*. Figure 5.8 shows the reservation message flow.
The NI+ sends an *EXTERNAL* message to the *Signaling Destination Address* (SDA) (for more details on signaling destination address see [STAD08]). The message is sent downstream in reverse direction of the normal message routing until it reaches the edge-NAT. The NI+ includes the data sender's address information object, which is used by the edge-NAT to restrict the possible NI addresses to only this address. So the NI+ can specify the IP address and port from where the following NSIS firewall signaling message must arrive. The EXT message creates NAT session states at any intermediate NF and ensures that the edge-NAT is discovered. If the EXT message reaches the edge-NAT device, it responds with a RESPONSE message, which includes a success object and the public reachable IP address and port number. Now it is possible to forward an incoming CREATE signaling message towards the NR, which

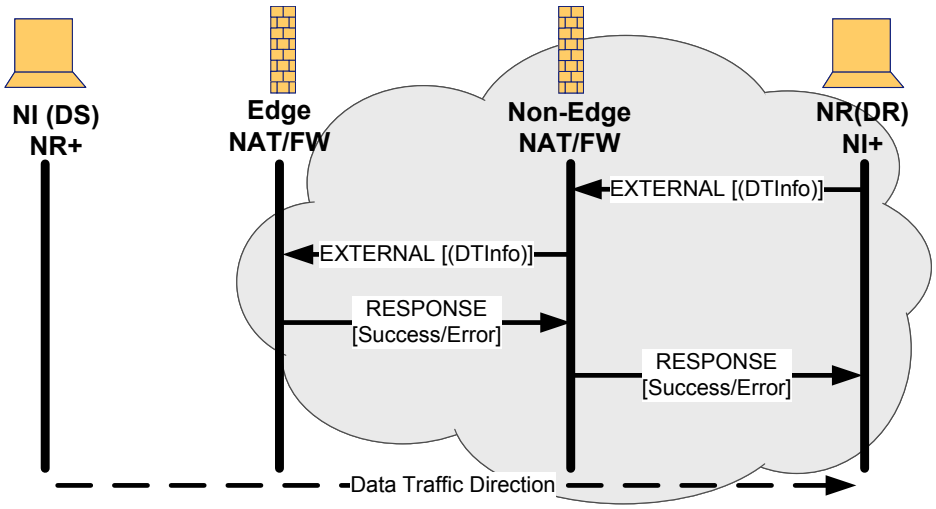


Figure 5.8: External Message Flow

is located behind a NAT.
In the firewall case the procedure is similar. The NI+ sends an EXT message to the SDA and the message is sent downstream until it reaches the edge-firewall. When utilising EXT to install firewall pinholes at the edge-firewall, the firewall pinhole format is included in the EXT message. If the EXT message reaches the edge-firewall, it responds with a RESPONSE message which includes a success object. The EXT and the corresponding RESPONSE message creates NAT/FW NSLP session states at any intermediate firewall between NI+ and the edge-firewall and ensures that the edge-firewall is discovered. When the RESPONSE message reaches the NI+, it is possible to use this firewall pinhole and traverse the firewall at the edge of NI+ domain or network.

5.3.2.2.4 Session Refresh

All NAT/FW NSLP sessions are maintained on a soft state basis. That means, that sessions as well as their policy rules, which are not refreshed, are removed automatically after a specified timeout. Each soft state must refresh by the same message type it was created, e.g., a state created by a CREATE must be maintained by CREATE. Refresh messages must carry the exact MRI and session ID as the initial message and a lifetime object with a lifetime greater than zero. NFs receiving session refresh messages must first check authentication and authorisation. The NF should check the lifetime with its local policies if it can accept it for this session. NRs which receive such a refresh message, must reply with a RESPONSE message and forward it towards the NI. So, intermediate NFs can update their refresh period. For more details

about the session lifetime calculation and the interaction with the message refresh rate see [STAD08]. Figure 5.9 shows a refresh message flow, with CREATE as an example.

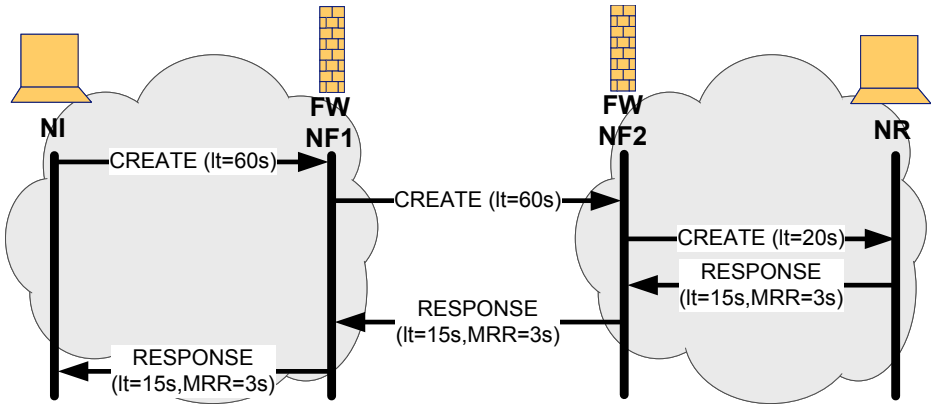


Figure 5.9: State Refresh Message Flow, CREATE as Example

5.3.2.2.5 Session Deleting

NSLP initiators can delete NAT/FW NSLP sessions anytime by sending a CREATE, or EXT message with a lifetime object which is set to zero. Sessions installed by CREATE can only be deleted by a CREATE message, which is similar for EXT sessions. Each NAT/FW NSLP node receiving such a message must first check authentication and authorisation and must immediately delete the session and associated policy rules if authentication and authorisation is verified. Afterwards the message is forwarded as long as it reaches the NR. Figure 5.10 shows an example delete message flow. Consider that non message with zero lifetime value generates any response.

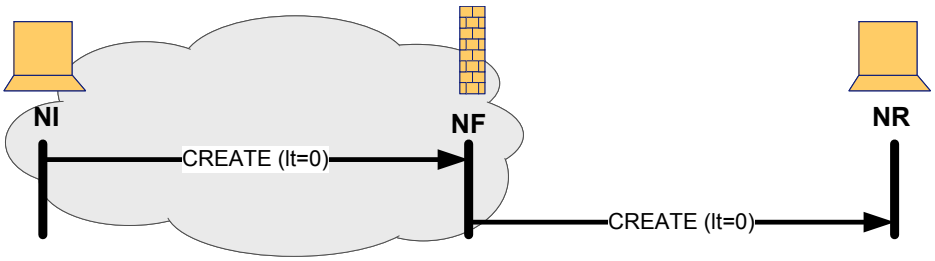


Figure 5.10: Delete Message Flow, CREATE as Example

5.3.3 Required NSIS and NAT/FW NSLP Extensions

NSIS and the NAT/FW NSLP provide most of the features needed for a NSIS and NAT/FW NSLP based Mobile IPv6 firewall traversal approach. However, some required features are not included in the current specification. The following section introduces these extensions and features.
Since the initial NSIS specification [SH08], NSIS has been considering node mobility as a potential use scenario. An analysis regarding the applicability of NSIS for signaling in mobile environment is done in [SFJ+08], but the feasibility has never been proved much less implemented. Therefore, the required interworking aspects or interfaces are not defined or specified.
Obviously, to be applicable for Mobile IPv6 firewall traversal, NSIS should be able to work in mobile environments. Therefore, it requires some kind of *NSIS autoconfiguration*, i.e., NSIS – in particular the utilised NSIS implementation – should be able to configure and re-configure itself, e.g., in case of IP address changes after handovers. Moreover, to trigger this re-configuration, NSIS needs to be able to detect the event of an IP address change or handover. This *NSIS autodetect IP address changes* feature requires either to statically observe the local addresses, e.g., via a fork, or, as a more elegant solution, to interwork with the local Mobile IPv6 implementation and let the Mobile IPv6 implementation trigger the NSIS implementation in case of a handover or IP address change.
The NSIS based Mobile IPv6 firewall traversal as described in this thesis, utilises the Mobile IPv6 and NSIS interworking approach. Therefore, several interfaces have been designed to allow the different implementations to interwork among them. These interfaces, as well as the interface messages, are described in detail in Section 7.1.
Besides the interworking, additional NAT/FW NSLP objects are required to perform the NSIS based Mobile IPv6 firewall traversal as later described in Section 5.3.5. The NSIS based Mobile IPv6 firewall traversal approach installs firewall pinholes of different formats, however, most of these firewall pinholes can not be signaled with the current NAT/FW NSLP specification [STAD08], as it lacks the required objects. This includes the capability of signaling firewall pinholes for:

- *Mobility Header*, and the requested mobility header type,
- *IPv6 Routing Header*, and the requested routing header type,
- *IPv6 Destination Options Header*, and the requested header type.

Therefore, additional NAT/FW NSLP *Object Types* have been specified, which allow the NAT/FW NSLP to signal the required firewall pinhole formats and to install this kind of firewall pinhole upon receiving an authorised request of this format. The usage of this new NAT/FW NSLP object types is similar to the other NAT/FW NSLP object types, as described in Section 5.3.2.2.1.

The new defined NAT/FW NSLP objects are:

- Mobility Header Object, Length 1 (Figure 5.11),
- IPv6 Routing Header Object, Length 1 (Figure 5.12),
- IPv6 Destination Options Header Object, Length 1 (Figure 5.13).

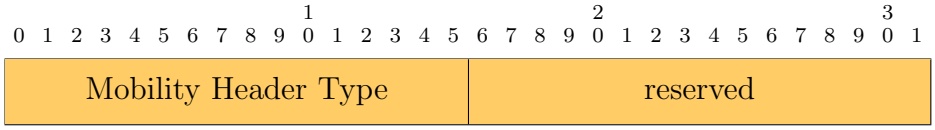


Figure 5.11: Mobility Header Object

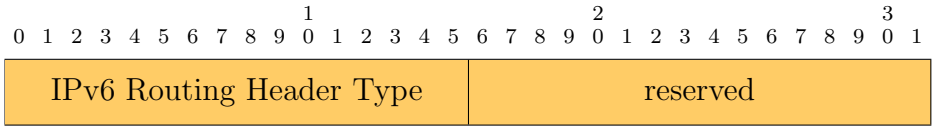


Figure 5.12: IPv6 Routing Header Object

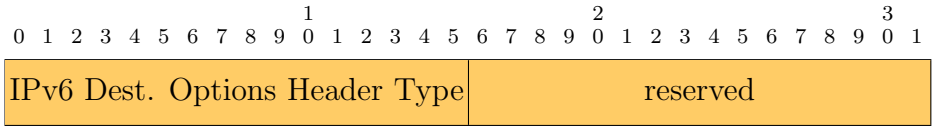


Figure 5.13: IPv6 Destination Options Header Object

Only if NSIS and the NAT/FW NSLP implement the required extensions as described in this section, they can be utilised for Mobile IPv6 firewall traversal.

5.3.4 Architecture

Figure 5.14 shows the *Generic NSIS Based Mobile IPv6 Firewall Traversal Architecture*. This only represents one possible architecture, others might implement the same using a different architecture. As already explained in Section 5.3.3, there is the need for interfaces between the Mobile IPv6, the NSIS, and the NAT/FW NSLP implementations and the Mobile IPv6 firewall traversal daemon. Details on how such interfaces can be implemented are described in Section 7.1. Moreover, Section 7.1 also specify the other software modules and software interfaces.

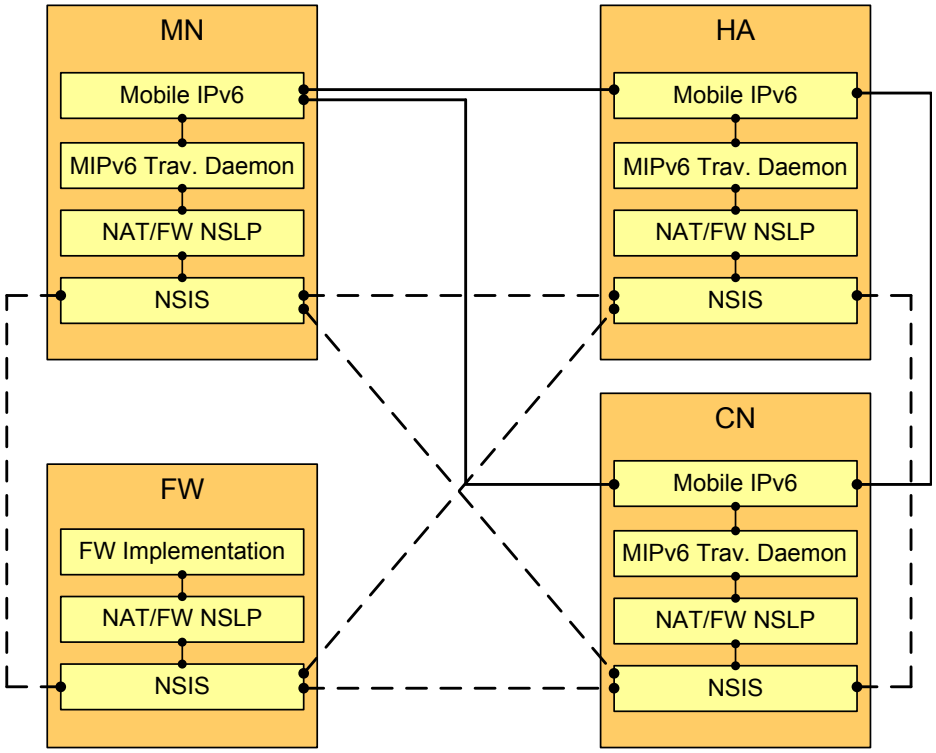


Figure 5.14: Generic NSIS Based Mobile IPv6 Firewall Traversal Architecture

5.3.5 Scenarios and Solutions

The following subsections discuss the scenarios and the corresponding problems as described in Chapter 3 and introduce how these different scenarios can be solved when utilising NSIS and the NAT/FW NSLP for Mobile IPv6 firewall traversal. However, these subsections only introduce how Mobile IPv6 firewall traversal can be accomplished with the current NSIS and NAT/FW NSLP specification and the small extensions presented in Section 5.3.3. Therefore, they deal with the three basic scenarios as described in Section 3.2, namely, firewall located at the edge of the MN's ASP, firewall located at the edge of the CN's ASP, and firewall located at the edge of the MN's MSP, and explain in detail how the messages or the type of traffic are enable

to successfully traverse the firewalls, as specified in Section 3.3:

- IKE/IKEv2 Signaling for establishing SAs,
- Binding Update and Binding Acknowledgement,
- Return Routability Test,
- Binding Update and Binding Acknowledgement CN,
- Route Optimisation,
- Bi-directional Tunneling.

Later, Section 5.3.6 describes some major improvements and extensions to the NAT/FW NSLP and Section 5.3.7 shows how this influences the NSIS based Mobile IPv6 firewall traversal approach.

5.3.5.1 Firewall Located at the Edge of the MN's ASP

In this scenario the mobile node is in a network or domain which is protected by a firewall (ASP-FW) that employs stateful packet filtering, as depicted in Figure 5.15. The external correspondent node and the home agent are also shown in the figure. The MN is located in a visited network and is expecting to communicate with the CN. If the MN initiates normal data traffic there is no problem with the stateful packet filter firewall, as the communication is initiated from the internal network. However, if the communication is initiated from the external network, e.g., from the CN, the stateful packet filtering firewall will block these packets, as they do not match any state in the firewall. This section explains how the NSIS based approach manages the Mobile IPv6 signaling traffic problems, specified in Section 3.2.1, for this scenario.

5.3.5.1.1 IKE/IKEv2 Signaling for Establishing SAs

In case the firewall is a stateful packet filter – which can be considered as the common case – the IKE/IKEv2 signaling for establishing SAs between the MN and the HA can traverse the firewall without assistance. This is possible as the signaling is initiated by the MN from the internal network or domain, and thus, the subsequently following traffic from the external HA matches an established state in the firewall state table. However, if the firewall is not a stateful packet filter, the reserve signaling from the HA is not able to traverse the firewall. NAT/FW NSLP can be utilised to configure the firewall to allow the associated IKE/IKEv2 packets to traverse. Therefore, the firewall should allow IKE/IKEv2 packets (to UDP port 500) to bypass. Having only the firewall located at the edge of the MN's ASP scenario in mind, it would be sufficient to install one firewall pinhole for the IKE/IKEv2 signaling from the HA to the

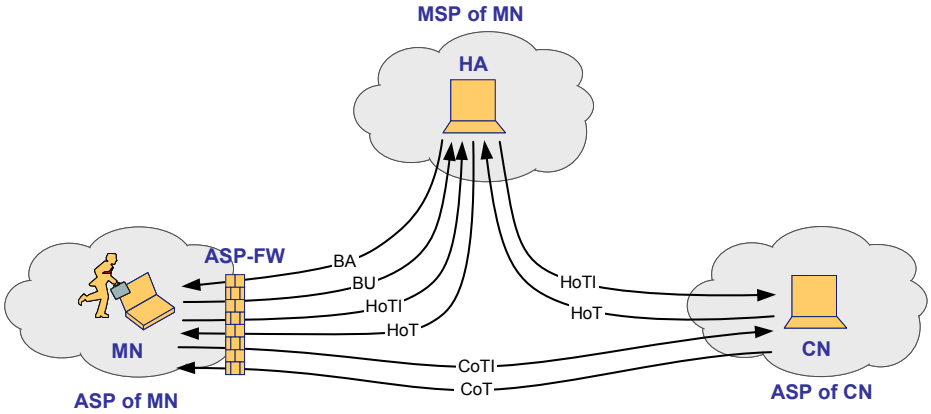


Figure 5.15: Firewall Located at the Edge of the MN's ASP

MN; considering also the firewall located at the edge of the MN's MSP scenario (Section 5.3.5.3), an additional firewall pinhole in the firewall at the edge of the MN's MSP to allow the IKE/IKEv2 signaling from the MN to the HA is required. Therefore, the procedure in real environments would be to install both kinds of firewall pinholes.

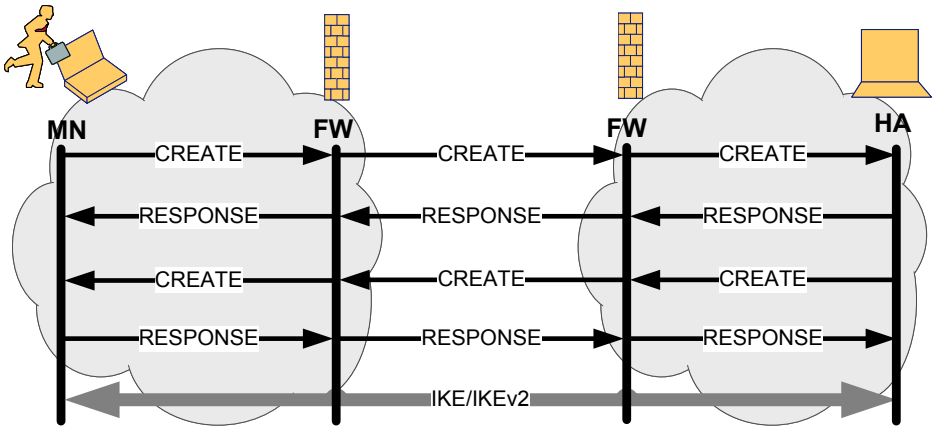


Figure 5.16: NSIS Firewall Signaling for IKE/IKEv2 Signaling

Figure 5.16 depicts the signaling messages flow which enables the IKE/IKEv2 signaling for establishing SAs to traverse the firewalls. As can be seen from the figure and as discussed before, two kinds of firewall pinholes are installed; first the firewall pinhole for IKE/IKEv2 signaling from the MN to the HA, afterwards, a firewall pinhole for the reverse direction, from the HA to MN, is installed. The format of the firewall pinholes for allowing IKE/IKEv2 signaling for establishing SAs is specified in A.1.1.

5.3.5.1.2 Binding Update and Binding Acknowledgement

IPsec protected Binding Updates cause problems in some deployment environments, as described in Chapter 3. As a solution, NAT/FW NSLP can be used to dynamically configure the firewall to allow the IPsec packets to traverse, before sending the Binding Update. Therefore, *IP Protocol ID 50* should be allowed in the filter policies in order to allow IPsec ESP. As a Mobile IPv6 implementation is utilised which is able to interact with the NAT/FW NSLP implementation (compare Section 5.3.1), the firewall pinhole policies can be more precise and also match the *Security Parameter Index* (SPI), valid for the communication between MN and HA. A.1.2 specifies the firewall pinhole format to allow the Binding Update to traverse the firewalls.

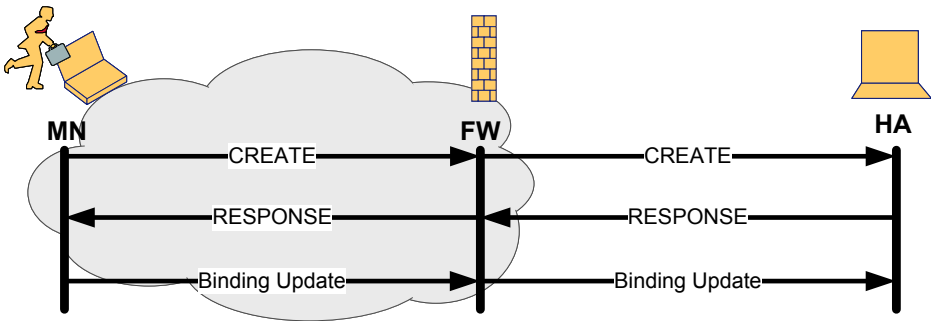


Figure 5.17: NSIS Firewall Signaling for Binding Update

Figure 5.17 shows the signaling message flow for this signaling. If the firewall is a stateful packet filter, the subsequent Binding Acknowledgement from the HA to the CoA can pass the firewall, as it matches an existing state in the table. However, in case the firewall is not a stateful packet filter, or stateful packet filtering is not implemented for IPsec traffic, a second firewall pinhole to allow the Binding Acknowledgement traverse the firewall is required. The firewall pinhole format is the same as for the Binding Update, but with inverted source- and destination-addresses and specified in A.1.3. Figure 5.18 depicts the signaling message flow which enables the Binding Acknowledgement to successfully traverse the firewalls.

Therefore, both NSIS firewall signaling as shown in Figure 5.17 and Figure 5.18, are required to allow the MN to update its binding cache by sending the Binding Update and receiving the Binding Acknowledgement from the HA. It has to be noticed that the signaling as specified in this section not only allow the BU and BA to traverse the firewall at the edge of the MN's ASP, but also the firewall at the edge of the MN'S MSP. Moreover, it allows the Binding Update and Binding Acknowledgement to successfully traverse all firewalls on the path between the MN and the HA.

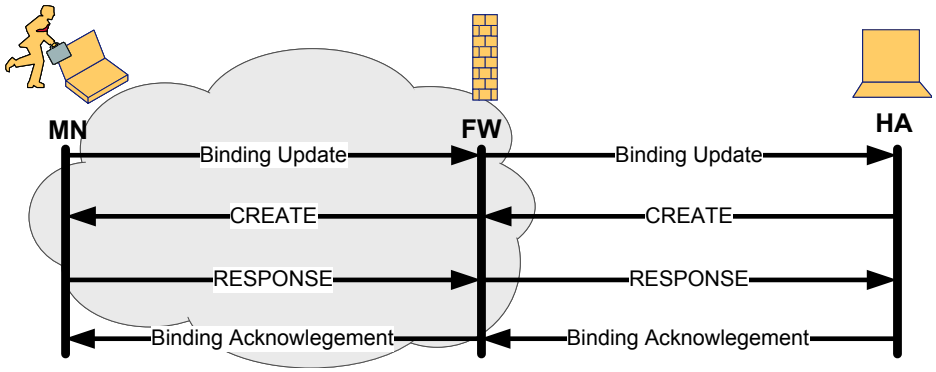


Figure 5.18: NSIS Firewall Signaling for Binding Acknowledgement

5.3.5.1.3 Return Routability Test

Immediately after moving into a new network, the MN acquires a new CoA, performs the firewall pinhole creation for the Binding Update and Binding Acknowledgement as described before, and sends the Binding Update to the HA. When the MN start to perform the *Return Routability Test* (RRT), it first sends the *Home Test Init* (HoTI) message to the HA. The HoTI message from the MN to the HA, as well as the responding *Home Test* (HoT) message from the HA to the MN, is IPsec encapsulated and might re-use the IPsec firewall pinholes between MN and HA created for the Binding Update and Binding Acknowledgement. In this case, no additional firewall pinholes are required to traverse the firewalls.

However, depending on the local Mobile IPv6 configuration, it might be required to install additional firewall pinholes, namely, in case the HoTI and HoT are configured to use different SPI than the Binding Update and Binding Acknowledgement. When meeting such a configuration, the HoTI and HoT messages can not re-use the previously installed firewall pinhole rule for BU and BA, and thus, they will be dropped by the firewalls. Therefore, the MN initiates NSIS firewall signaling and open firewall pinholes for the HoTI and the HoT messages by allowing this different SPI.

As it might be the case that the firewall is not a stateful packet filter firewall, again firewall pinholes for both directions could be required. The message flows for this firewall pinhole signaling are depicted in Figure 5.19 and Figure 5.20, the firewall pinhole formats are specified in A.1.4 and A.1.5.

After having successfully installed these firewall pinholes at the firewalls, the HoTI and HoT messages are able to traverse all firewalls between the MN and the HA. When a firewall is deployed at the edge of the MN's MSP or at the edge of CN ASP, the HoTI and HoT messages from the HA to the CN and vice versa might be dropped. These scenarios are handled in Section 5.3.5.2 and Section 5.3.5.3.

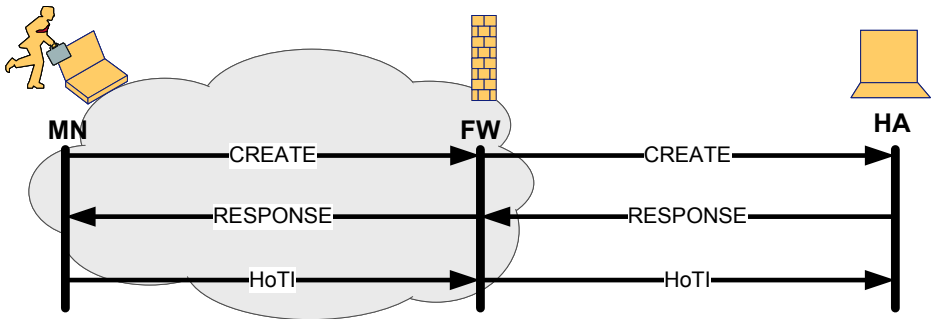


Figure 5.19: NSIS Firewall Signaling for Home Test Init

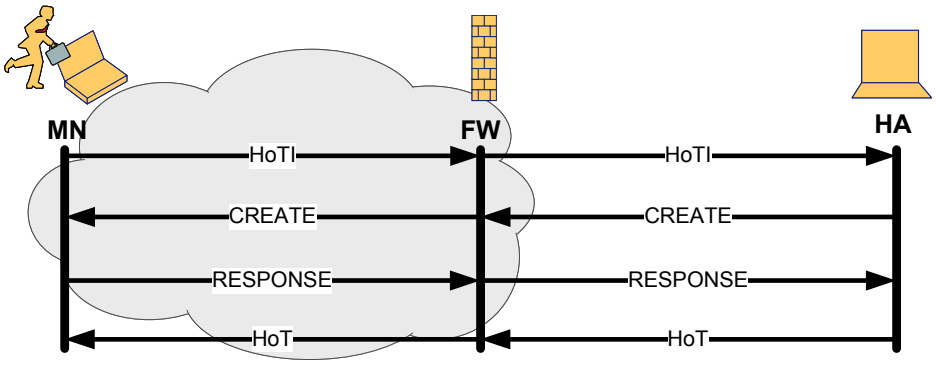


Figure 5.20: NSIS Firewall Signaling for Home Test

When performing the RRT, apart from sending the HoTI message via the HA, the MN also sends the *Care-of Test Init* (CoTI) message directly to the CN and expects a *Care-of Test* (CoT) message as response. The CoTI message is allowed to traverse the firewall at the edge of the MN's ASP as it comes from the internal domain.

However, even when deploying a stateful packet filtering firewall, the subsequently CoT message can not traverse the firewall at the edge of the MN's ASP as the CoT message does not match a state in the firewall state table previously installed by the CoTI message. This bases on the fact that the CoTI message format is different from the CoT message format. The CoTI consists of *Mobility Header* of *Type 2*, whereas the CoT consists of *Mobility Header* of *Type 4*. Thus, the firewall at the edge of the MN's ASP will drop the subsequently CoT from the CN to the MN. NSIS firewall signaling is required to signal for an additional firewall pinhole to allow the CoT message to traverse.

The signaling message flow for this firewall pinhole is shown in Figure 5.21, the firewall pinhole format is specified in A.1.7. If the CN's ASP deploys a firewall at the edge of

it network or domain, the NAT/FW NSLP has to signal for a firewall pinhole before being able to successfully send the CoTI to the CN. This scenario is described in Section 5.3.5.2 in detail.
However, such an approach where external nodes are able to install firewall pinhole in a foreign ASP-FW clearly requires a strong authentication and authorisation framework. Chapter 6 discusses this in more detail and presents several potential candidates. This solution works under the assumption that the firewalls will allow NSIS messages from external network to bypass, by applying a delayed packet filter state establishment and authorisation from the CN. However, operators might be reluctant to allow NSIS message from external network as this might lead to *Denial of Service* (DoS) attacks. The CN might therefore be required to authorise the traversal of NSIS signaling message implicitly to reduce unwanted traffic. To avoid this complexity, it is also possible to ask the CN to open pinholes in the firewall on behalf of the MN. However, this solution may not work in some scenarios due to routing asymmetry as explained in [STAD08].

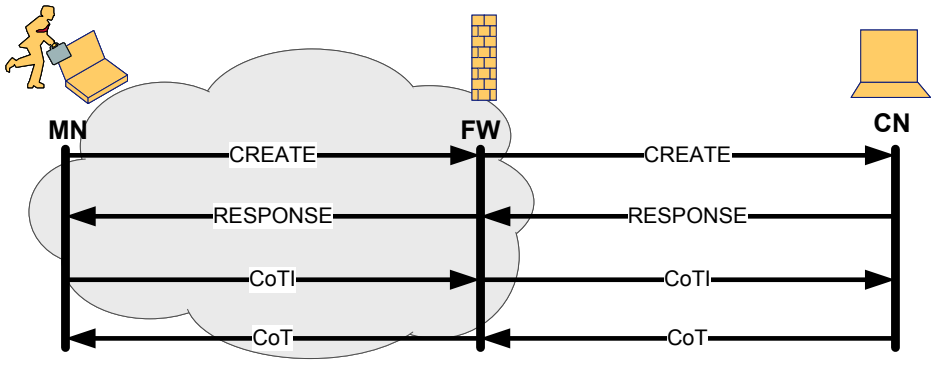


Figure 5.21: NSIS Firewall Signaling for Care-of Test

5.3.5.1.4 Binding Update and Binding Acknowledgement CN

Once the RRT is successful, the MN sends the Binding Update message to the CN. Even if the deployed firewall at the edge of the MN's ASP is a stateful packet filtering firewall and the BU message to the CN installs a firewall state at the stateful packet filter, the subsequent BA is not able to re-use this state. This bases on the fact that the BA message format is different from the BU message format. The BU consists of a *IPv6 Destination Options Header* and *Mobility Header* of *Type 5*, whereas the BA consists of a *Routing Header for IPv6* of *Type 2* and *Mobility Header* of *Type 6*. Thus, the firewall at the edge of the MN's ASP will drop the subsequent BA from the CN to the MN.

Therefore, a firewall pinhole at the edge of the MN's ASP is needed which allows the BA messages traverse. This is done by the MN using to the EXT mechanism, to install a firewall pinhole with a different source address then the MNs CoA address, in this case, the CN address. Besides, letting the MN taking care of this signaling and thus keeping the changes on the CN as small as possible, it also avoids long end-to-end delays which occur when the CN signals for this firewall pinhole. The signaling message flow is depicted in Figure 5.22, and the firewall pinhole format is specified in A.1.8.

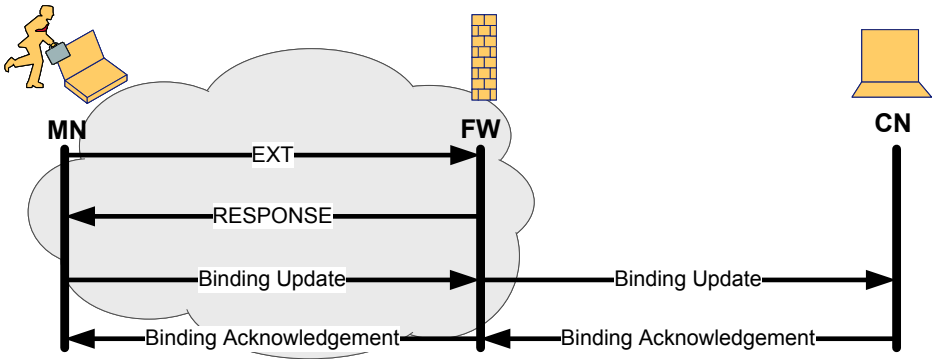


Figure 5.22: NSIS Firewall Signaling for Binding Acknowledgement from CN

5.3.5.1.5 Route Optimisation

After sending the BU to the CN, the MN might want to continue sending data traffic. The packet format of route optimised data traffic is very similar to the packet format of the before sent Binding Update and Binding Acknowledgement. In case the firewall pinholes for the BU and the BA only match against the *IPv6 Destination Options Header*, respectively the *Routing Header for IPv6* of *Type 2* and not against the *Mobility Header*, the route optimised data traffic would be able to re-use these firewall pinholes and no additional signaling would be required. However, ISPs might want to allow the Binding Update and Binding Acknowledgement to traverse, but not the route optimised data traffic to traverse by default, e.g., for selling the route optimisation as service. Thus, the ISPs might deploy the more precise firewall pinholes for the BU and BA to the CN, and the route optimisation is not able to work without additional policy rules. If ISPs want to allow route optimisation by default, the firewall pinholes for the BU and BA to the CN should not match against the *Mobility Header*. This allows the route optimised data traffic to re-use the firewall pinholes and no additional NSIS firewall signaling is required in this scenario.
If an ISPs want to install a firewall pinhole which allows only the route optimised

data traffic, the NSIS firewall signaling is very similar to the signaling for the Binding Update and Binding Acknowledgement from CN as described in Section 5.3.5.1.4. The data traffic from the MN to the CN can traverse the firewall as it comes from the internal trusted domain, but the subsequently data traffic requires a firewall pinhole at the edge of the MN's ASP to traverse. In this scenario it does not matter whether the MN is data sender or data receiver. Even if the MN is the data sender, and thus, the traffic is initiated from the internal trusted domain, the firewall pinhole has to be the same, as if the MN is the data receiver. The signaling message flow is the same as for the Binding Acknowledgement as depicted in Figure 5.22. However, the firewall pinhole format is different compared to the Binding Acknowledgement and is specified in A.1.9.

5.3.5.1.6 Bi-directional Tunneling

Since the MN had earlier initiated firewall pinholes for the purpose of Binding Update, Binding Acknowledgement, Home Test Init, and Home Test, no additional signaling is required as the before installed rules match again the addresses, the IPsec ESP encapsulation and the SPI. Therefore, the bi-directional tunneled data traffic can re-use the firewall pinholes as described in Section 5.3.5.1.2 and Section 5.3.5.1.3, and thus is able to traverse the firewall. The message flow is shown in Figure 5.23. Independently whether the MN is the data sender or data receiver, no firewall pinholes signaling is necessary at all in this scenario.

5.3.5.2 Firewall Located at the Edge of the CN's ASP

In this scenario the correspondent node is in a network or domain, which is protected by a firewall (ASP-FW) that employs stateful packet filtering, as depict in Figure 5.24. The external mobile node and its associated home agent are also shown in the figure. The MN is located in a visited network or domain which do not deploy a firewall at the edge of the network and is expecting to communicate with the CN. If the CN initiates normal data traffic, there is no problem with the stateful packet filtering firewall, as the communication is initiated from the internal network. However, if the communication is initiated from the external network, e.g., from the MN or the HA, the stateful packet filtering firewall will block these packets, as they do not match any state in the firewall. This section explains how the NSIS Mobile IPv6 firewall traversal approach overcomes the Mobile IPv6 signaling traffic problems, specified in Section 3.2.2, in this scenario.

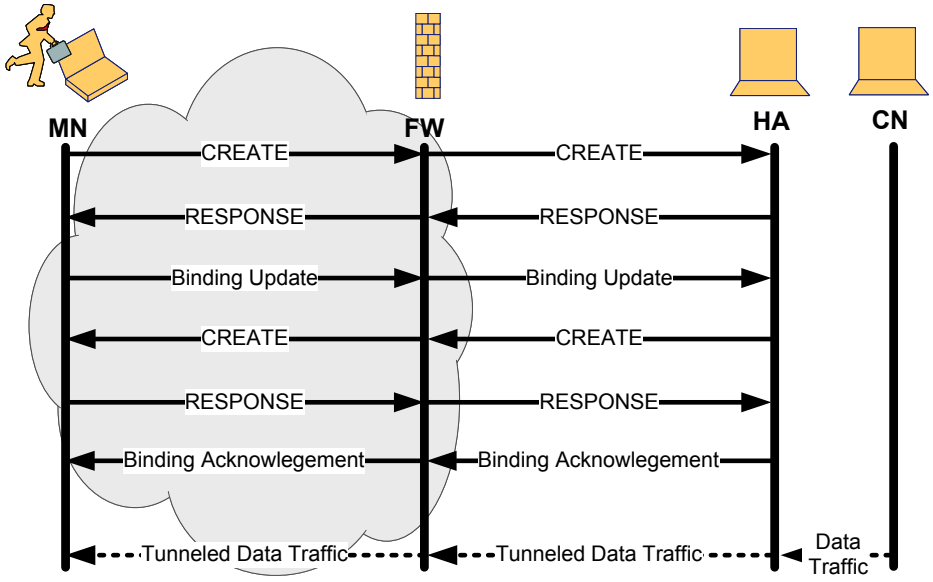


Figure 5.23: NSIS Firewall Signaling for Bi-directional Tunneled Data Traffic

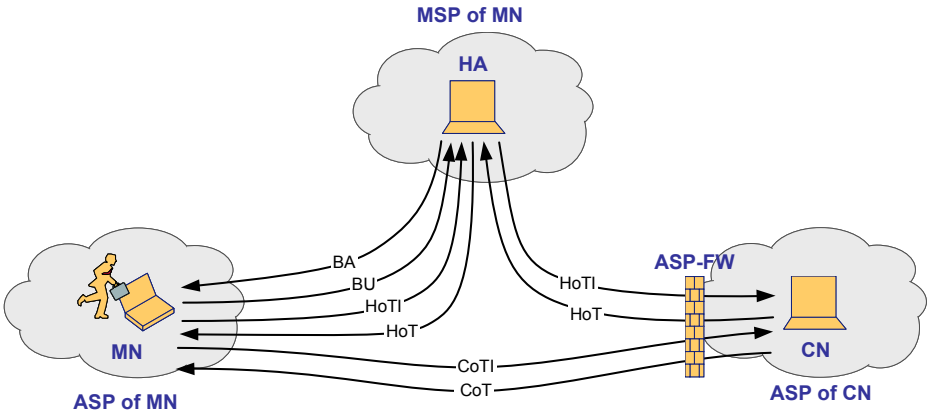


Figure 5.24: Firewall Located at the Edge of the CN's ASP

5.3.5.2.1 Binding Update and Binding Acknowledgement

Immediately after moving into a new network, the mobile acquires a new CoA, performs the Binding Update and Binding Acknowledgement, and thus updates the binding cache at the home agent. Up to this point, the Mobile IPv6 procedure works successfully in this scenario, as no firewall is present in the signaling path for Binding Update and Binding Acknowledgement.

5.3.5.2.2 Return Routability Test

If the MN is out of its home network, it has to perform the return routability test before sending the Binding Update to the CN. When it starts to perform the Return Routability Test, it sends the Home Test Init through the HA to the CN and expects a HoT message from the CN along the same path. As the HoTI message from the HA to the CN does not match any state, the firewall at the edge of the CN's ASP will drop this message. The same problem occurs if the MN sends the Care-of Test Init message directly to the CN, this message will also be dropped by the firewall. Thus, two additional firewall pinholes are required to let the HoTI and the CoTI successfully traverse the firewall at the edge of the CN's ASP. The first firewall pinhole, the one for the HoTI message from the HA to the CN, is installed by the HA. The NSIS firewall signaling is shown in Figure 5.25 and the firewall pinhole format is specified in A.1.4.

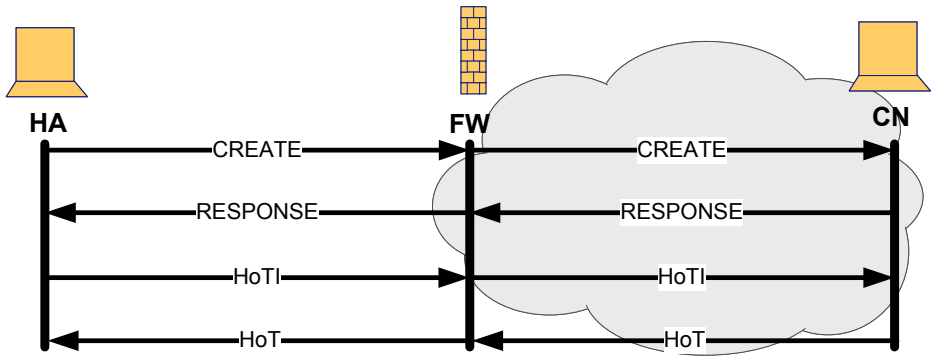


Figure 5.25: NSIS Firewall Signaling for Home Test Init

The installed firewall pinhole allows the HoTI message to reach the CN, which responses with the HoT message. The HoT message can traverse the firewall without assistance as it comes from the trusted internal domain. However, the RRT procedure still can not be executed, as the CoTI is still not able to traverse the firewall. The MN initiates a NSIS NAT/FW NSLP session by sending a CREATE message to the CN to install a policy rules for the CoTI message. The NSIS firewall signaling to allow the CoTI message is shown in Figure 5.26; the format for this kind of firewall pinhole is specified in A.1.6. The responding CoT message from the CN to the MN is able to traverse the firewall without additional firewall pinholes.

When the MN receives both, the CoT and the HoT messages, the RRT procedure is finished. Now the MN is able to update the binding cache at the CN.

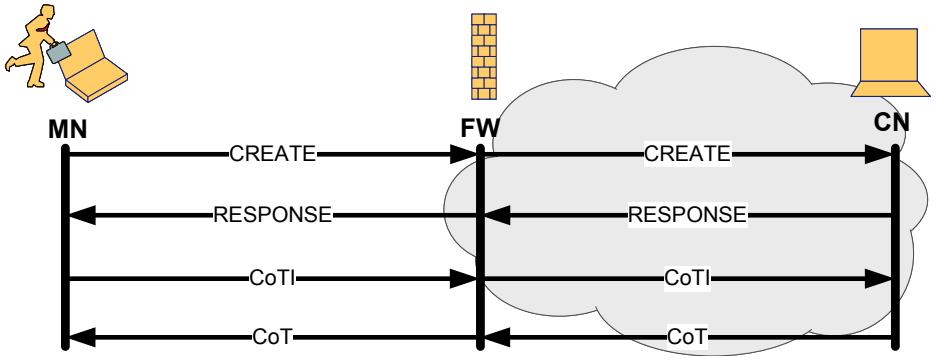


Figure 5.26: NSIS Firewall Signaling for Care-of Test Init

5.3.5.2.3 Binding Update and Binding Acknowledgement CN

Therefore, the MN sends the Binding Update message to the CN. As the Binding Update message does not match any previously installed firewall pinhole, the firewall at the edge of the CN's ASP will drop the BU from the CN to the MN. A firewall pinhole at the edge of the CN's ASP is needed which allows the BU message to traverse. This is done by the MN using the CREATE mechanism. The signaling message flow is depicted in Figure 5.27; the firewall pinhole format is specified in A.1.8. The subsequent Binding Acknowledgement from the CN to the MN can traverse the firewall as it comes from the internal domain.

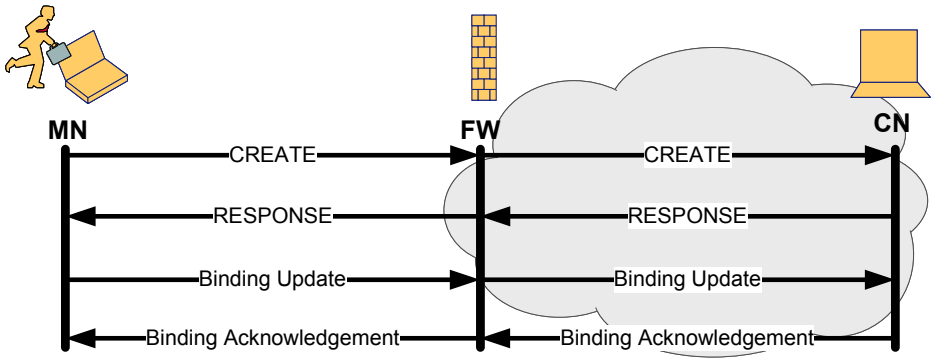


Figure 5.27: NSIS Firewall Signaling for Binding Update to CN

5.3.5.2.4 Route Optimisation

After sending the BU to the CN, the MN or the CN might want to continue sending route optimised data traffic. The packet format of route optimised data traffic is

very similar to the packet format of the before sent Binding Update and Binding Acknowledgement, as described in Section 5.3.5.1.5. So, the route optimised data traffic potentially would be able to re-use these firewall pinholes and no additional signaling would be required. In case the ISP might want to allow the Binding Update and Binding Acknowledgement to traverse, but not the route optimised data traffic to traverse by default as described in Section 5.3.5.1.5, additionally firewall pinholes for the route optimisation data traffic are required.

Even if the deployed firewall at the edge of the CN's ASP is a stateful packet filtering firewall and the route optimised data traffic was initiated by the CN, and installs a firewall state at the stateful packet filter, the subsequently route optimised data traffic is not able to re-use this state. This bases on the fact that the route optimised data traffic message formats are different, depending on the direction. The route optimised data traffic from the MN to the CN consists of a *IPv6 Destination Options Header*, whereas the route optimised data traffic from the CN to the MN consists of a *IPv6 Routing Header* of *Type 2*. Thus, the firewall at the edge of the CN's ASP will drop the route optimised data traffic from the MN to the CN, independently whether or not it was initiated from its internal domain before.

As intended keeping the modification on the CN as small as possible, the MN takes care of this signaling. However, due to the fact that this approach requires the CN to be NSIS and NAT/FW NSLP aware, the EXT mechanism would be also applicable here, and would avoid long end-to-end delays. The MN install a filter policy for route optimised data traffic between MN and CN using CREATE. The NSIS firewall signaling message flow is depicted in Figure 5.28; the firewall pinhole format is specified in A.1.9.

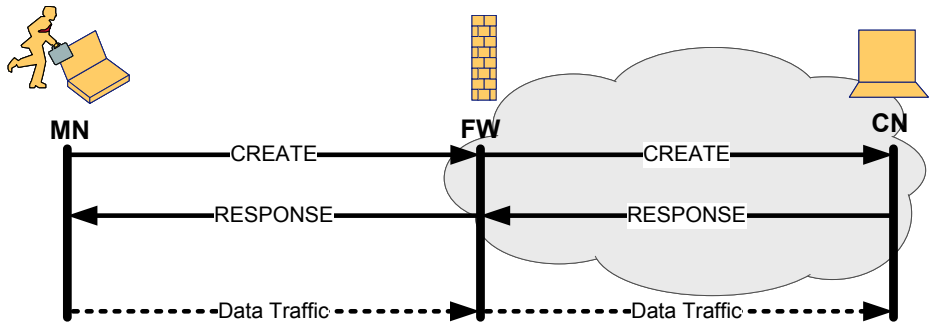


Figure 5.28: NSIS Firewall Signaling for Route Optimised Data Traffic

5.3.5.2.5 Bi-directional Tunneling

If the firewall at the edge of the CN's ASP is a stateful packet filtering firewall, there is no need for any signaling if the CN initiates the data traffic. The CN sends the bi-directional tunneled data traffic and hence the stateful packet filter will store relevant state information and accepts packets from the reverse direction.

If the HA is the data sender, either the CN has to initiate the signaling using EXT or the HA using CREATE, in order to configure the firewall to allow the data traffic traverse from the HA to CN. As the intention is to keep the changes on the CN small, the HA signals for these firewall pinholes.

To support that function, the Mobile IPv6 implementation at the HA will need to be changed that to it triggers the local NSIS based Mobile IPv6 firewall traversal implementation in the event of receiving bi-directional tunneled data traffic from the MN. The local NSIS based Mobile IPv6 firewall traversal implementation is able to trigger the firewall pinhole creation process. The NSIS signaling flow for this firewall pinhole is shown in Figure 5.29; the firewall pinhole format is specified in A.1.10.

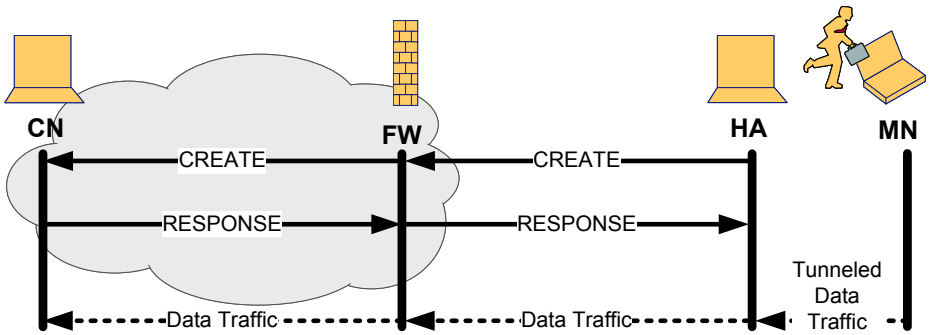


Figure 5.29: NSIS Firewall Signaling for Bi-directional Tunneled Data Traffic

5.3.5.3 Firewall Located at the Edge of the MN's MSP

In this scenario the mobile node's MSP is in a network or domain which is protected by a firewall (MSP-FW) that employs stateful packet filtering, as depicted in Figure 5.30. The MN and the CN are also shown in the figure, both are in a network which does not deploy a firewall at its edge. This section explains how the NSIS based approach handles the Mobile IPv6 signaling traffic problems, specified in Section 3.2.3, in this scenario.

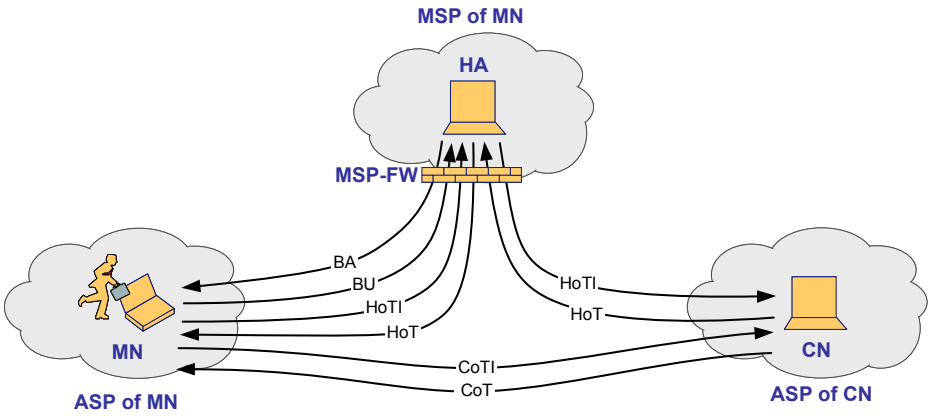


Figure 5.30: Firewall Located at the Edge of the MN's MSP

5.3.5.3.1 IKE/IKEv2 Signaling for Establishing SAs

As already described in Section 5.3.5.1.1, the firewall located at the edge of the MN's MSP scenario requires an additional firewall pinhole in the firewall at the edge of the MN's MSP to allow the IKE/IKEv2 signaling from the MN to the HA to traverse the firewall. NAT/FW NSLP is utilised to configure the firewall to allow the associated IKE/IKEv2 packets to traverse. Therefore, the firewall is configured to allow IKE/IKEv2 packets (to UDP port 500) to bypass.
Figure 5.31 depicts the signaling messages flow which enables the IKE/IKEv2 signaling for establishing SAs to traverse the firewalls. As can be seen from the figure, two kinds of firewall pinholes are installed; first the firewall pinhole for IKE/IKEv2 signaling from the MN to the HA, afterwards, a firewall pinhole for the reverse direction, from the HA to MN, is installed. This second firewall pinhole is required if the firewall at the edge of the MN's ASP is not an stateful packet filter; in this case the IKE/IKEv2 signaling in the reverse direction from the HA to MN will be dropped. The format of the firewall pinholes for allowing IKE/IKEv2 signaling for establishing SAs is specified in A.1.1.

5.3.5.3.2 Binding Updates and Binding Acknowledgement

After entering a new network or domain, the MN tries to send a Binding Update to its home agent. But as it is initiated from the external network, i.e., by the MN, it will be blocked by the firewall at the edge of the MN's MSP. Additionally, the Binding Update message to the HA is assumed to be IPsec encapsulated, and this might cause problems, as some primitive firewalls do not recognise IPsec traffic and hence drop the packets because of the absence of any transport header, as described in Section 3.1.3. The MN initiates NSIS firewall signaling to create firewall pinholes

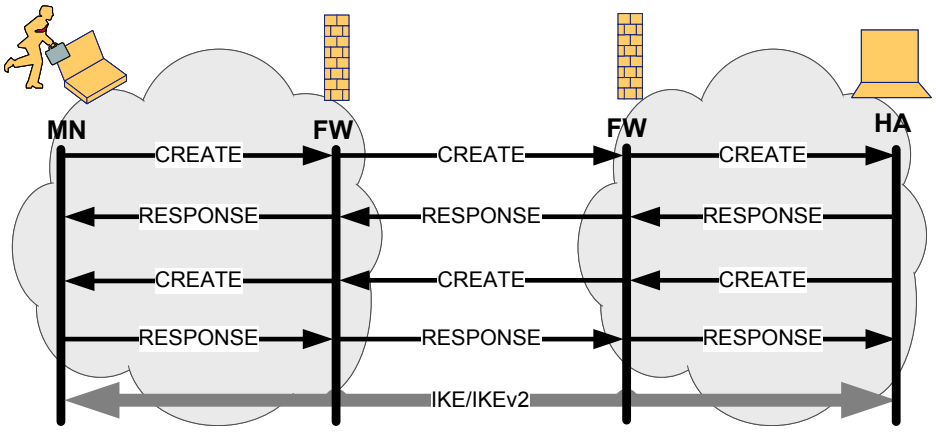


Figure 5.31: NSIS Firewall Signaling for IKE/IKEv2 Signaling

that allow the Binding Update message and Binding Acknowledgement to pass through the firewall. Afterwards, the MN is able to send the Binding Update message to the HA and to receive the corresponding Binding Acknowledgement. The signaling for this kind of firewall pinholes has already been described in Section 5.3.5.1.2 and is depicted in Figure 5.17 and Figure 5.18. The format of the firewall pinholes are specified in A.1.2 and A.1.3.

5.3.5.3.3 Return Routability Test

In some configurations, as explained in Section 5.3.5.1.3, the rules previously installed in the firewall will neither allow the HoTI message nor the HoT message between the MN and the HA to pass through. Hence, the MN has to install a different set of firewall pinholes for these signaling messages by initiating another NAT/FW NSLP signaling exchange as described in Section 5.3.5.1.3, and depicted in the Figures 5.19 and 5.20. Afterwards, it sends the HoTI message to the HA. The HA installs firewall pinholes between the HA and the CN to allow the HoTI and HoT message to traverse, and accordingly sends the HoTI to the CN. Thus, the subsequent HoT message from the CN to the HA is allowed to traverse the firewall. The HoT message from the HA to the MN is also allowed as it matches the IPsec firewall pinholes installed before. The RRT accomplishes successfully. The detailed message flow between the HA and the CN is shown in Figure 5.32.

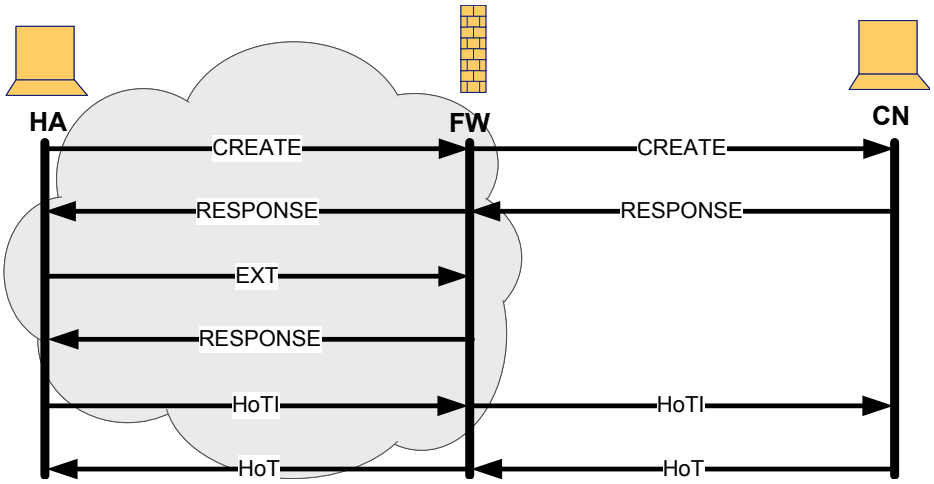


Figure 5.32: NSIS Firewall Signaling for HoTI and HoT

5.3.5.3.4 Route Optimisation

After successfully completing the RRT, the MN might want to send route optimised data traffic to the CN. In this scenario, no additional firewall pinholes or NSIS firewall signaling are required, as the MN sends the route optimised data traffic directly to CN, and none of these networks deploy a firewall. This is applicable for both cases when either MN or CN is the data senders. The situation is different when the data traffic should be send via bi-directional tunneling.

5.3.5.3.5 Bi-directional Tunneling

In this scenario, the bi-directional tunneled data traffic from the MN to the HA, and vice versa, is able to traverse the firewall, as it matches the firewall pinhole rules installed before, depending on the local configuration either the firewall pinholes for the Binding Update and Binding Acknowledgement, or for the HoTI and HoT message. However, the bi-directional tunneled data traffic still needs to traverse the firewall again, after the packets are de-encapsulated by the HA and sent to the CN. In case the traffic is initiated by the MN, no additional firewall pinhole is necessary; the traffic is initiated from the internal network, i.e., by the HA after de-encapsulating the packets from the MN and sent to the CN. This installs a firewall pinhole at the stateful packet filtering firewall at the edge of the MN's MSP and allows the packets to pass in the reverse direction. If the firewall is not a stateful packet filter, or the CN is the data sender for the bi-directional tunneled data traffic, additional NSIS firewall pinhole signaling is required, as explained in Section 5.3.5.2.5. The NSIS

firewall signaling flow for this firewall pinhole is shown in Figure 5.29, the firewall pinhole format is specified in A.1.10.

5.3.6 Improved NAT/FW NSLP Proxy Mode

The *NSIS based Mobile IPv6 firewall traversal* approach, as specified before, works under the requirements and assumptions described in Section 5.3.1. These include simple accomplishable assumptions, e.g., all networks and nodes must be IPv6, the MSA and MSP be co-located, but also hard to achieve or at least in the initial phases of the incremental NSIS deployment unrealistic requirements, e.g., that all involved Mobile IPv6 nodes, namely, MN, HA and CN, must support NSIS, the NAT/FW NSLP and a modified Mobile IPv6 implementation. Especially the assumption that the CN must support NSIS, the NAT/FW NSLP and a modified Mobile IPv6 implementation is unlikely, at least in the initial NSIS migration. Therefore, this situation will occur quite frequently, and hence a scenario where the CN does not support NSIS and the NAT/FW NSLP should be supported. Such an scenario highly impacts the NSIS and NAT/FW NSLP protocols suites, as it requires NSIS and NAT/FW NSLP to work properly without the need for supporting true end-to-end NSIS firewall signaling to the correspondent application. The support for such a scenario would be practical even in the late phases of NSIS and NAT/FW NSLP migration, as the assumption of the CN to be NSIS and NAT/FW NSLP aware is quite markable. This is possible by utilising an *improved NAT/FW NSLP proxy mode*.
Two approaches are feasible for the improved NAT/FW NSLP proxy mode. The first approach is pre-configured, i.e., the firewall at the edge of the corresponding node is pre-configured to work in proxy mode and take care of the NSIS firewall signaling for the corresponding node, i.e., the NSIS responder. The second possible approach works on NSIS firewall signaling timeouts and also performs in the case if no firewall is deployed at the edge of the firewall, whereas the pre-configured approach requires the firewall as the NSIS responder. Both approaches are intended to work in the scenario shown in Figure 5.33.
The first approach is explicit. Therefore, the firewall at the edge of a network or domain is pre-configured to work in proxy mode and to take care of the NSIS signaling for nodes in its network or domain. This could be done in different ways: either the firewall is pre-configured to take care of the NSIS firewall signaling for the whole network, i.e., a network prefix, or the firewall is configured to act as NR for some pre-configured IP-addresses, e.g., the corresponding node.
If the firewall receives an NSIS initiation message (QUERY) or a NAT/FW NSLP CREATE messages, it first compares this request with the local configuration, the pre-configured IP-addresses or the network prefix, and acts as NSIS responder for this signaling if the request matches the pre-configuration. Afterwards, the firewall decides

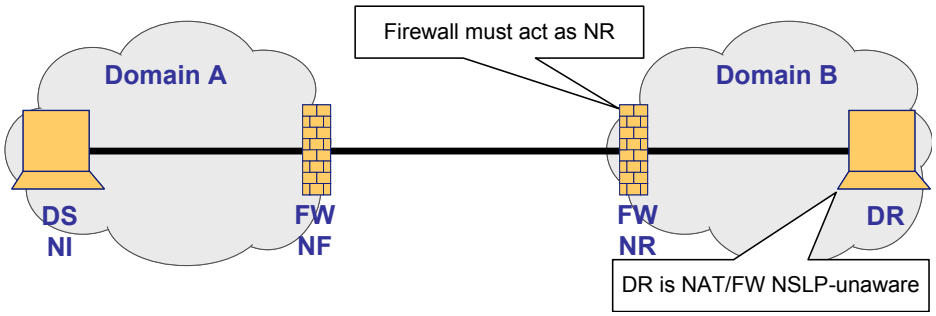


Figure 5.33: Improved NSIS and NAT/FW NSLP Proxy Mode Scenario

whether this request, i.e., the NSIS initiator, is allowed to signal and is allowed to install this certain kind of firewall pinholes using NSIS firewall signaling. This decision could be made by requesting the local configuration of the firewall, or by a local policy repository as part of the authentication, authorisation, and accounting infrastructure. Figure 5.34 depicts how the pre-configured proxy mode approach works.

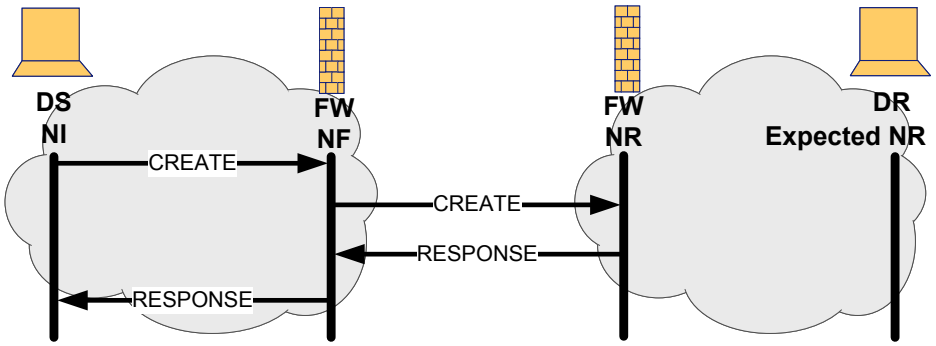


Figure 5.34: Pre-configured NSIS and NAT/FW NSLP Proxy Mode

The advantage of this approach is that the decision whether or not a firewall pinhole request is allowed is still taken within the domain of the expected NR and with the help of the same authentication, authorisation, and accounting infrastructure. The disadvantage of this approach is that the firewall at the edge of the CN's ASP must be deployed. Otherwise, the NI would not be able to signal for firewall pinholes, as no NR would response to the request message. This is a problem if firewalls are deployed between the NI's edge firewall and the domain of the expected NR. Additionally, it requires modifications to NSIS to be able to support this pre-configured proxy mode, as the node has to take care for both kind of signaling, the NSIS and the NAT/FW NSLP signaling.

The second approach can also be described with the help of the scenario, shown in Figure 5.33. In this scenario, the domain of the NI has at least one NSIS and NAT/FW NSLP aware firewall or there is a NSIS aware firewall on the path to the expected NR, but the NR itself is not NSIS and NAT/FW NSLP aware. Goal of this approach is to let the last firewall on the path to act as NR, and thus, allows NSIS and NAT/FW NSLP to work in scenarios where the expected NR, e.g., the correspondent node in Mobile IPv6, does not support NSIS and the NAT/FW NSLP.
This approach is implicit and works on NSIS and NAT/FW NSLP timeouts. The NI sends the NSIS initiation or NAT/FW NSLP creation message and the message traverse all NFs on the signaling path, but no NR responds back. Therefore, the response timer on the NFs will expire, as no response message is received. The last NF notices this timeout and decides locally, whether itself is able and allowed to act as NR for this NSIS firewall signaling. This decision could be made by the local configuration of the firewall, or by requesting a local policy repository as part of the authentication, authorisation, and accounting infrastructure. If the NF is allowed to act as NR for this NSIS firewall signaling, it responses back to the NI. This response must be sent for the NSIS and the NAT/FW NSLP initiation messages, respectively the QUERY and the CREATE message. The NF responses with a RESPONSE message and might install the requested firewall pinholes. The message flow of this approach is depicted in Figure 5.35, using the network deployment shown in Figure 5.33.

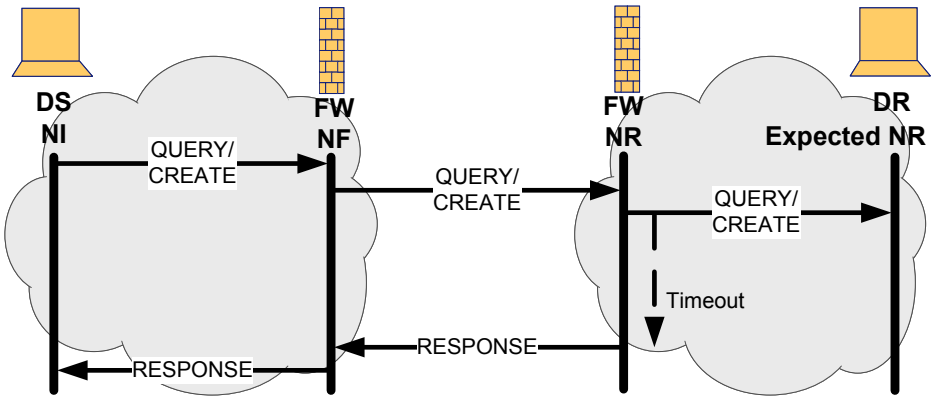


Figure 5.35: Timeout Based NSIS and NAT/FW NSLP Proxy Mode

In contrast to the pre-configured approach described before, the implicit timeout based approach does not require the firewall at the edge of the expected NR to be deployed. Any last NF on the signaling path can potentially act as NR of the NSIS firewall signaling, even if the last NF on the signaling path is in the domain of the NI. The NSIS timeout based proxy mode as described exposes that NSIS firewall signaling messages could by terminated somewhere along the signaling path.

However, this complicates allowing end-to-end security mechanisms to be applied, as the NSIS end-to-end security mechanisms is intended to enable authorisation by both end-hosts, and to connect the NSIS end-to-end signaling together with application layer signaling. But this approach would be broken when applying the timeout based proxy mode.
Due to this security considerations, the improved NSIS based Mobile IPv6 firewall traversal utilises the pre-configured proxy mode as described before and therefore requires the firewall at the edge of the CN's ASP to be deployed. Further investigation of the authentication and authorisation aspects of the timeout based approach is necessary to successfully deploy it for NSIS based Mobile IPv6 firewall traversal.

5.3.7 Improved NSIS Based Mobile IPv6 Firewall Traversal

As explained in Section 5.3.6, the assumption that all involved Mobile IPv6 nodes, namely, MN, HA and CN, must support NSIS, the NAT/FW NSLP and a modified Mobile IPv6 implementation is quite unrealistic, at least in the initial NSIS migration. This is especially the case for the assumption that the CN must support NSIS, the NAT/FW NSLP and a modified Mobile IPv6 implementation. Such a situation will occur quite frequently, and hence a scenario where the CN does not support NSIS and the NAT/FW NSLP should be supported. Therefore, Section 5.3.6 has proposed an improved NAT/FW NSLP proxy mode, which allows the *NSIS based Mobile IPv6 firewall traversal* approach, as specified in Section 5.3.5, to support such a scenario where the CN is not NSIS and NAT/FW NSLP aware.
When utilising the *improved NAT/FW NSLP proxy mode* as introduced in Section 5.3.6, all NSIS firewall signaling which is intended to the CN, will be intercepted and processed by the firewall at the edge of the CN's ASP. Obviously, this firewall has to support NSIS and the NAT/FW NSLP. As the firewall now acts as NSIS Responder for the received request, it replies to the received CREATE message with a RESPONSE message and installs the requested firewall pinhole rule.
As mentioned before, this is the case for all NSIS firewall signaling addressed to the CN. Thus, all *NSIS based Mobile IPv6 firewall traversal* message flows to the CN would be intercepted and processed by the edge firewall. The correspondent node can stay focused on Mobile IPv6 and does not require to support NSIS, the NAT/FW NSLP and a modified Mobile IPv6 implementation. Hence, the requirement to support NSIS, the NAT/FW NSLP and a modified Mobile IPv6 implementation is not longer valid.
Figure 5.36 exemplifies how the NSIS firewall signaling flow would look with the improved NAT/FW NSLP proxy mode for the CoTI message as it was specified for the normal NSIS based Mobile IPv6 firewall traversal in Section 5.3.5.2.2, depicted in Figure 5.26.
The MN initiates a NSIS NAT/FW NSLP session by sending a CREATE message to the CN to install firewall pinhole rule for the CoTI message. The NSIS firewall

signaling to allow the CoTI message is shown in Figure 5.36. In contrast to the flow described in Section 5.3.5.2.2, the CREATE message from the MN is now intercepted and processed by the firewall at the edge of the CN's ASP. The firewall processes the message and, in case the request is permitted, installs the firewall pinhole and replies with the RESPONSE message to the MN. As can be seen from the figure, the CN is not longer involved in the *NSIS based Mobile IPv6 firewall traversal* procedure.

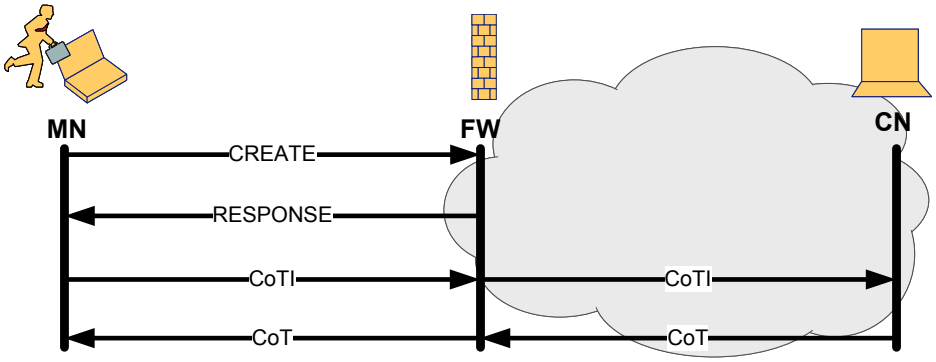


Figure 5.36: NSIS Firewall Signaling with Improved NAT/FW NSLP Proxy Mode

5.3.8 Summary

This section has described how the NSIS NAT/FW NSLP can address the issues caused by stateful packet filter firewalls encountered in Mobile IPv6 environments. It has utilised an IETF NSIS NAT/FW NSLP based firewall traversal solution and introduced some essential extensions which allow it to handle all the problems and impacts of having firewalls in Mobile IPv6 environments in all the different scenarios, as described in Chapter 3. It has to be noted that a real scenario could include a combination of some set of these cases. This approach has been submitted to the IETF in [SFL07] and has also been proposed at [SFH+07].
In contrast to other middlebox configuration solutions, the NSIS based solution can offer a solution for all deployment scenarios, assuming that the MN, the CN, the HA and the firewalls are NSIS and NAT/FW NSLP aware. However, in contrast to other explicit middlebox configuration approaches, NAT/FW NSLP requires more signaling overhead but does not require knowledge about the topology and avoids possible performance problems caused by the deep packet inspection of other approaches.
Whereas Section 5.3.5 has only described how Mobile IPv6 firewall traversal can be accomplished with the current NSIS and NAT/FW NSLP specification and the essential extensions presented in Section 5.3.3, Section 5.3.6 has introduced some major improvements for NSIS and the NAT/FW NSLP which allow the Mobile IPv6 firewall

traversal based on NSIS to significantly reduce the requirements and to perform faster by utilising the improved NSIS and NAT/FW NSLP.
An important aspect for firewall traversal or firewall signaling is how to ensure that only authorised hosts are allowed to perform actions. This authentication and authorisation considerations and aspects are discussed in detail in Section 6.1. Later, Section 7.1 presents an overview on the *NSIS based Mobile IPv6 firewall traversal* proof-of-concept implementation, developed as part of this thesis.

5.4 Mobile IPv6 Application Layer Gateway

Several problems and impacts could prevent Mobile IPv6 from operating successfully in the presence of firewalls as introduced in Chapter 3. To overcome these problems, this section introduces one possible solution, based on *Application Layer Gateways* (ALGs) [SH99]. This solution was developed and standardised within the IETF *MEXT working group* [MEXT]. The *Mobile IPv6 Application Layer Gateway* solution has been adopted as working group draft. It represents one key contribution of this thesis.

Section 4.3 already introduces the *Application Layer Gateway* [SH99] technique. ALGs are application specific agents that are aware of the protocol details of a specific protocol, e.g., *Session Initiation Protocol* (SIP) [RSC+02] for VoIP, and is able to understand the protocol messages and their dependencies within the communication. Thus, it is able to allow applications in different networks and domains behind middleboxes to connect each other transparently. An ALG processes the application traffic while transit and assists the middlebox in implementing its function. Therefore, the ALG performs a deep packet-inspection of packets and understands the application protocols which are supported. The work of the ALG is transparent to end-hosts, and does not terminate or influence sessions between the end-hosts. The ALG interacts with a middlebox to set up middlebox states, firewall pinholes or access control filters or uses the middlebox state information.

As described in Chapter 3, the standard Mobile IPv6 does not work in presence of firewalls. One possible approach to tackle this issue is to utilise the Application Layer Gateway technique for Mobile IPv6 firewall traversal. The following sections describe how Application Layer Gateways could be utilised to overcome the Mobile IPv6 firewall traversal problems. This approach has been adopted by the IETF in [KSYB08] and [KSSB08].

Firstly, Section 5.3.1 presents the requirements and assumptions for the Application Layer Gateway based Mobile IPv6 firewall traversal approach. The approach itself is divided into two parts, more precisely two guidelines. Section 5.4.2 present the guidelines for firewall administrators regarding Mobile IPv6 traffic, i.e., which type of traffic has to be allowed by pre-configuration, whereas Section 5.4.3 present guidelines for firewall vendors to help them implement their firewalls in a way that allows Mobile IPv6 signaling and data messages to traverse. Only a combination of these two parts can achieve *Mobile IPv6 Application Layer Gateway* firewall traversal, i.e., both need to be deployed together. Section 5.4.4 summarises the *Mobile IPv6 Application Layer Gateway* firewall traversal section.

5.4.1 Requirements and Assumptions

When utilising Application Layer Gateway for Mobile IPv6 firewall traversal, several requirements and assumptions have to be considered:

- All networks and nodes MUST support IPv6; IPv4/IPv6 interworking as well as dual-stack solutions are out of scope and not supported.
- All involved Mobile IPv6 nodes, i.e., MN, HA and CN (as depict in Figure 5.1), MUST support a Mobile IPv6 implementation.
- All firewalls at the edge of the different networks or domains, as well as all potentially intermediate middleboxes MUST support a *Mobile IPv6 Application Layer Gateway* implementation and MUST perform middlebox functionality, e.g., by implementing the *netfilter/ip6tables* [netfilter]. In addition, the firewall must be able to analyse the *IPv6 Mobility Header*, the *IPv6 Routing Header* and the *IPv6 Destination Options Header*.
- The firewalls MUST be configured as described in Section 5.4.2. This has to be done by manual pre-configuration.
- The firewalls MUST implement the recommendations for firewall vendors, as described in Section 5.4.3.
- The *Mobile IPv6 Application Layer Gateway* firewall traversal approach does not require any firewall MUST to be deployed. However, any firewall – independent of its position – CAN be deployed.

5.4.2 Guidelines for Firewall Administrators Regarding Mobile IPv6 Traffic

This section presents guidelines for firewall administrators to help them configure their firewalls in a way that allows the Mobile IPv6 signaling and data messages to pass through. This work is also standardised by the IETF in [KSYB08]. It assumes that the firewalls include some kind of stateful packet filtering capability as described in Section 5.4.3 and standardised in [KSSB08]. The static firewall pinholes rules that need to be configured are described in this section. In some scenarios, the support of additional mechanisms to create firewall pinholes required for Mobile IPv6 signaling and data traffic to pass through will be necessary. A possible solution, describing the dynamic capabilities needed for the firewalls to create firewall pinholes based on Mobile IPv6 signaling traffic is described in Section 5.4.3.

5.4.2.1 Firewall Located at the Edge of the MN's MSP

This section presents the guidelines for configuring a firewall that is located at the edge of the MN's MSP, respectively protects the home agent, as depicted in Figure 5.37.

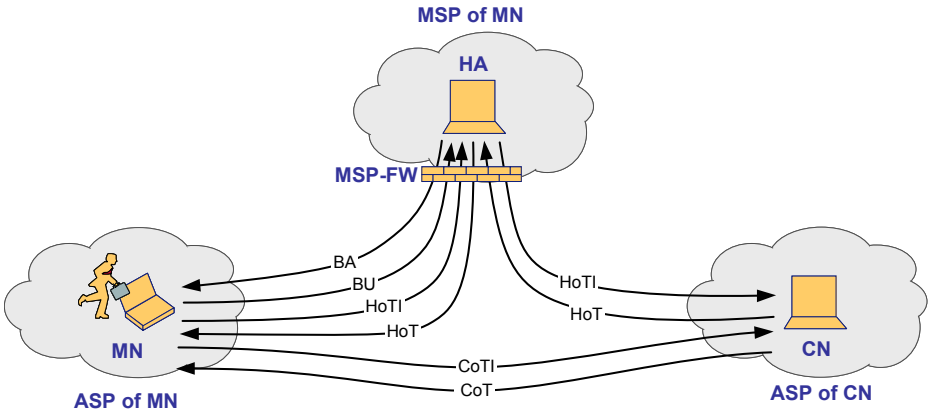


Figure 5.37: Firewall Located at the Edge of the MN's MSP

For each type of traffic that needs to traverse this firewall, guidelines are presented on how to identify and to allow that certain kind of traffic. The following types of traffic must be considered:

- IKE/IKEv2 Signaling for establishing SAs,
- Binding Update and Binding Acknowledgement,
- Return Routability Test,
- Bi-directional Tunneling.

5.4.2.1.1 IKE/IKEv2 Signaling for Establishing SAs

The MN and the HA exchange IKE/IKEv2 signaling in order to establish the security associations before sending the Binding Update. The security associations will later be used for securing the mobility signaling messages. Hence, these messages need to be allowed to traverse the firewall at the edge of MN's MSP. The pattern presented in Table 5.1 matches these messages and allows them to traverse the firewall. A firewall administrator must open a firewall pinhole at the firewall at the edge of the MN's MSP to let the IKE/IKEv2 signaling to traverse. The IKE/IKEv2 signaling in reverse direction is allowed to traverse as it comes from the internal side.

Destination Address:	HA Address
Transport Protocol:	UDP
Destination UDP Port:	500

Table 5.1: Pattern for IKE/IKEv2 Signaling for Establishing SAs

5.4.2.1.2 Binding Update and Binding Acknowledgement

After establishing the IKE/IKEv2 security associations, the MN tries to sends the Binding Update to the HA. The signaling between the MN and HA is protected using IPsec ESP. These messages are critical to the Mobile IPv6 protocol; if these messages are discarded, Mobile IPv6 as specified in [JPA04] will cease to work. In order to permit these messages to pass through, the firewall has to be configured to allow these messages using the patterns specified in Table 5.2. The corresponding Binding Acknowledgement is allow to traverse the firewall at the edge of the MN's MSP as it comes from internally.

Destination Address:	HA Address
Next Header:	50 (ESP)
Mobility Header Type:	5 (BU)

Table 5.2: Pattern for Binding Update Message

5.4.2.1.3 Return Routability Test

Configuring the firewall for the pattern presented in Table 5.3 allows the HoTI messages of the return routability test between the MN and the HA to traverse. The HoT message from the HA to the MN is allowed to traverse the firewall at the edge of MN's MSP as it comes from the internal trusted domain.

Destination Address:	HA Address
Next Header:	50 (ESP)
Mobility Header Type:	1 (HoTI)

Table 5.3: Pattern for HoTI Message

5.4.2.1.4 Bi-directional Tunneling

The firewall pinhole as described in Section 5.4.2.1.2 potentially allows the data traffic through the HA to pass the firewall as this traffic is protected using IPsec ESP and

might re-use the firewall pinhole. However, if the firewall is configured very restrictively and allows only IPsec ESP encapsulated traffic with *Mobility Header Type 5*, an additional firewall pinhole to allow all IPsec ESP encapsulated traffic is required to let the bi-directional tunneled data traffic traverse. This pattern is presented in Table 5.4.

Destination Address:	HA Address
Next Header:	50 (ESP)

Table 5.4: Pattern for Bi-directional Tunneling

5.4.2.2 Firewall Located at the Edge of the CN's ASP

This section presents the guidelines for configuring a firewall at the edge of CN's ASP if the node behind it should be able to act as Mobile IPv6 correspondent node, as depicted in Figure 5.38.

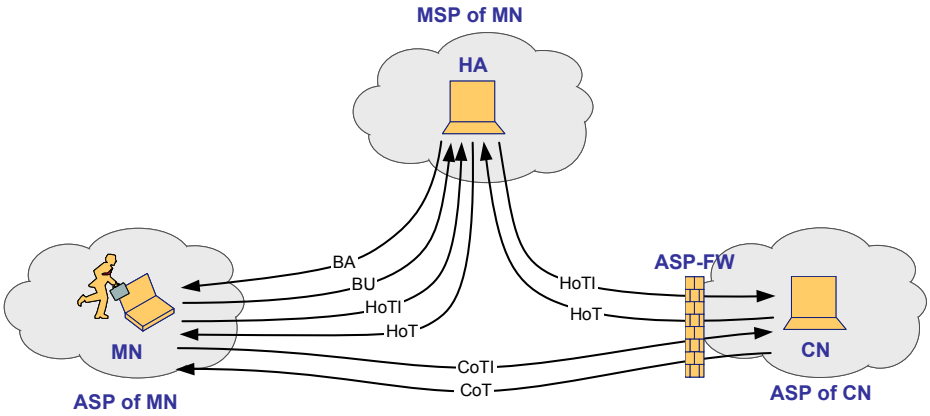


Figure 5.38: Firewall Located at the Edge of the CN's ASP

For each type of traffic that needs to traverse this firewall, guidelines are presented on how to identify and allow that traffic. The following types of traffic must be considered:

- Return Routability Test,
- Binding Update and Binding Acknowledgement CN,
- Route Optimisation,
- Bi-directional Tunneling.

5.4.2.2.1 Return Routability Test

Parts of the return routability test signaling have to pass through the HA, namely the HoTI and the HoT messages. Without assistance, the HoTI message from the HA to the CN is not able to traverse the firewall. If only a few privileged nodes (like servers) are allowed to be contactable by outside nodes, the pattern presented in Table 5.5 allows the HoTI messages to reach these nodes:

Destination Address:	CN Address
Mobility Header Type:	1 (HoTI)

Table 5.5: Pattern for HoTI Message

The CN Address describes the address(es) of the privileged node(s) which should be able to act as CN. This firewall pinhole allows the HoTI message from the HA to the CN to traverse the firewall at the edge of the CN's ASP. The HoT message from the CN to the MN through the HA can traverse the firewall without any assistance. Hence no additional firewall pinhole is required.
Route optimisation allows direct communication of data packets between the MN and a CN without tunneling it back through the HA. To get route optimisation work, the MN has to send a CoTI message directly to the CN, which responses with a CoT message. However, a stateful packet filter firewall would prevent the CoTI message to traverse as there is no established state on the firewall. If only a few privileged nodes (like servers) are allowed to be contactable by outside nodes, the pattern presented in Table 5.6 allows the CoTI messages to reach these nodes.

Destination Address:	CN Address
Mobility Header Type:	2 (CoTI)

Table 5.6: Pattern for CoTI Message

The CN Address describes the address(es) of the privileged node(s) which should be able to act as CN. The CoT message from the CN to the MN can traverse the firewall without any assistance. Hence no additional firewall pinhole is required.

5.4.2.2.2 Binding Update and Binding Acknowledgement CN

After successfully performing the return routability procedure, the MN sends the BU to the CN and expects the BA. Since this BU does not match any previous installed firewall pinhole rule, an additional firewall pinhole is required. When only a few privileged nodes (like servers) are allowed to be contactable by outside nodes, the pattern presented in Table 5.7 allows the BU messages to reach these nodes.

Destination Address:	CN Address
Mobility Header Type:	5 (BU)

Table 5.7: Pattern for Binding Update CN Message

The CN Address describes the address(es) of the privileged node(s) which should be able to act as CN. This allows the BU to traverse the firewall; the BA can pass the firewall without any assistance. Therefore, the Binding Update and Binding Acknowledgement sequence can be performed successfully.

5.4.2.2.3 Route Optimisation

Also the route optimised data traffic from the MN directly to the CN can not traverse the firewall without assistance. A dynamically created firewall pinhole such as the one specified in Section 5.4.3.3 will allow this traffic to traverse.

5.4.2.2.4 Bi-directional Tunneling

The bi-directional tunneling data traffic from the MN to the CN via the HA can not traverse the firewall without assistance. A dynamically created firewall pinhole such as the one specified in Section 5.4.3.3 will allow this traffic to traverse.

5.4.2.3 Firewall Located at the Edge of the MN's ASP

This section presents the guidelines for configuring a firewall at the edge of MN's ASP that protects the network a mobile node visiting, as shown in Figure 5.39.
For each type of traffic that needs to traverse this firewall, guidelines are presented on how to identify and allow that traffic. The following types of traffic must be considered:

- IKE/IKEv2 Signaling for establishing SAs,
- Binding Update and Binding Acknowledgement,
- Return Routability Test,
- Binding Update and Binding Acknowledgement CN,
- Route Optimisation,
- Bi-directional Tunneling.

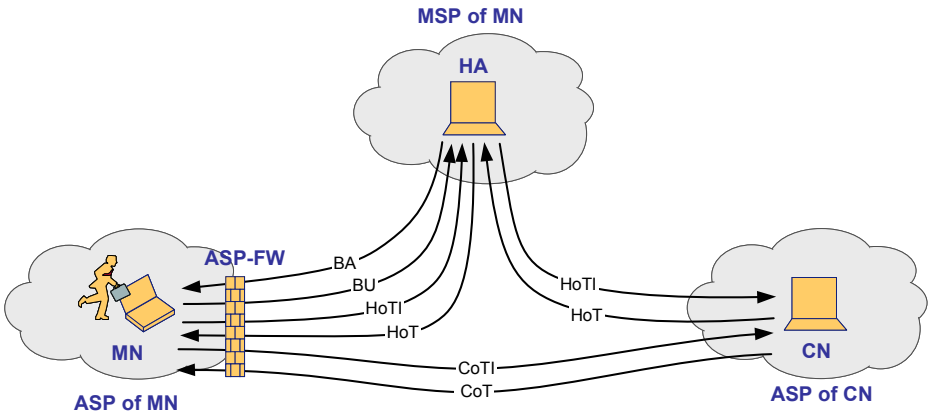


Figure 5.39: Firewall Located at the Edge of the MN's ASP

5.4.2.3.1 IKE/IKEv2 Signaling for Establishing SAs

Before sending the Binding Update, the MN and HA exchange IKE/IKEv2 signaling in order to establish the security associations. The security associations will be used for securing the mobility signaling messages. Due to variable source/destination IP addresses and MN always as initiator, the pattern presented in Table 5.8 will let the IKE/IKEv2 negotiation pass through the firewall at the edge of the MN's ASP.

Source Address:	Visited Subnet Prefix
Transport Protocol:	UDP
Destination UDP Port:	500

Table 5.8: Pattern for IKE/IKEv2 Signaling for Establishing SAs

5.4.2.3.2 Binding Update and Binding Acknowledgement

As described before, signaling between the MN and HA is protected using IPsec ESP. Many firewalls are configured to block incoming ESP packets. Moreover, from the view of the firewall, both source and destination addresses of these messages from/to the mobile node are variable. In Mobile IPv6, the mobile node is always the initiator for the BU. Since MN's CoA is not able to be known in advance, the firewall can use the following patterns to allow these messages to traverse.
The pattern presented in Table 5.9 allows the Binding Update to traverse the firewall at the edge of the MN's ASP. The subsequently Binding Acknowledgement would be dropped by the firewall as it does not match any firewall pinhole. Section 5.4.3

Source Address:	Visited Subnet Prefix
Next Header:	50 (ESP)
Mobility Header Type:	5 (BU)

Table 5.9: Pattern for Binding Update Message

describes how to enable the Binding Acknowledgement to traverse.

5.4.2.3.3 Return Routability Test

Source Address:	Visited Subnet Prefix
Next Header:	50 (ESP)
Mobility Header Type:	1 (HoTI)

Table 5.10: Pattern for HoTI Message

The pattern presented in Table 5.10 allows the HoTI message to traverse the firewall at the edge of the MN's ASP. Some firewall administrators might deploy very restrictive firewall rules and might not allow Mobile IPv6 by default. In this case, the pattern presented in Table 5.11 might be required to allow the CoTI message to traverse the firewall.

Source Address:	Visited Subnet Prefix
Mobility Header Type:	2 (CoTI)

Table 5.11: Pattern for CoTI Message

5.4.2.3.4 Binding Update and Binding Acknowledgement CN

The pattern presented in Table 5.12 allows the Binding Update from the MN to the CN to traverse the firewall at the edge of MN's ASP, in case it is configured very restrictively. The corresponding Binding Acknowledgement from the CN to the MN requires a dynamically installed firewall pinhole to traverse, as specified in Section 5.4.3.2.

Source Address:	Visited Subnet Prefix
Mobility Header Type:	5 (BU)

Table 5.12: Pattern for Binding Update CN

5.4.3 Guidelines for Firewall Vendors Regarding Mobile IPv6 Traffic

This section presents guidelines for firewall vendors to help them implement their firewalls in a way that allows Mobile IPv6 signaling and data messages to traverse as standardised in [KSSB08]. Thus, it describes how to implement stateful packet filtering capability for Mobile IPv6.

5.4.3.1 Mobile IPv6 Firewall Primitives

5.4.3.1.1 Requirements

The *Mobile IPv6 Application Layer Gateway* firewall traversal approach assumes that the firewalls are capable of deep packet inspection, at least up to the *Mobility Header*. For *Route Optimisation* and *Bi-directional Tunneling* it also requires the inspection of the *IPv6 Destination Options Header* and *IPv6 Routing Header*. It also assumes that the firewalls are capable of creating filters based on arbitrary fields of the contents of a signaling packet.

5.4.3.1.2 Detecting and Parsing the Mobility Header

The *Mobility Header* is the basic primitive in all Mobile IPv6 signaling messages. Thus, the firewalls need to be able to recognise the presence of the mobility header and to be able to parse the contents of the mobility header. The mobility header as well as its format is described in [JPA04]. Firewalls must at least understand the contents of the *Mobility Header Type* field that describes the type of signaling message carried.

5.4.3.1.3 Parsing Mobility Options

The mobility header can carry additional information in terms of mobility options as described in [JPA04]. Some of these mobility options need to be understood for proper creation of state on the firewalls. Hence, firewalls must be able to parse the mobility options defined in [JPA04].

5.4.3.2 Allowing Signaling Response Packets

The Mobile IPv6 signaling messages are performed as a request-response pair. The request message is usually allowed by setting up a static firewall pinhole rule to allow the traffic to traverse, as described in Section 5.4.2. The response message on the other hand can be dynamically allowed if the firewall can automatically setup a filter for the response packets when the request packet passes through. This is not trivial, but straightforward. There are three message pairs that are of importance to Mobile

IPv6 signaling, namely the BU/BA, HoTI/HoT and CoTI/CoT pairs. When the first message in the pair traverses the firewall in one direction, the firewall must setup a filter rule to allow the corresponding second message to traverse in the other direction. For example, consider a received packet that matches a static firewall pinhole rule configured on a firewall, as exemplified in Table 5.13.

Destination Address:	Address of HA
Next Header:	50 (ESP)
Mobility Header Type:	5 (BU)

Table 5.13: Pattern for Received Binding Update Message

This firewall pinhole rule allows a Binding Update message to the HA address to traverse the HA firewall. Once a packet that matches this rule passes through the firewall, the firewall must setup a dynamic filter for the corresponding return packet as exemplified in Table 5.14.

Source Address:	Destination Address from Packet
Destination Address:	Source Address from Packet
Next Header:	50 (ESP)
Mobility Header Type:	6 (BA)

Table 5.14: Corresponding Dynamic Return Filter for Binding Acknowledgement

This firewall pinhole rule ensures that the return Binding Acknowledgement packet can traverse the firewall. The firewall pinhole rules can be generalised as shown in Table 5.15. Such dynamic rules can be timed out after 420 seconds – the maximum lifetime of a *Binding Cache Entry* – unless renewed by new mobility messages.

Passing Packet MH Type	Setup Return Filter with MH Type
Mobility Header Type: 1 (HoTI)	Mobility Header Type: 3 (HoT)
Mobility Header Type: 2 (CoTI)	Mobility Header Type: 4 (CoT)
Mobility Header Type: 5 (BU)	Mobility Header Type: 6 (BA)

Table 5.15: Message Pairs in Mobile IPv6

5.4.3.3 Allowing Data Packets Based on Signaling

Once the Mobile IPv6 signaling completes, the MN might send data traffic to the CN or vice versa. The traffic filters for the data traffic can be inferred from the contents of the signaling messages that setup the session. This section describes how firewalls

can intelligently set up firewall pinhole filters for data traffic based on signaling traffic. The following example describes how to set up a firewall pinhole filter for allowing incoming route optimised data traffic messages from a CN to a MN after the MN sent a BU message to a CN.
When the BU message from MN to CN (Mobility Header Type 5) traverses the firewall, the firewall extracts the home address from the *Home Address Option* [JPA04] of the packet.

Destination Address:	Source Address of the Packet (MN CoA)
Source Address:	Destination Address of the Packet (CN)
Routing Header Type 2 Address:	MN HoA (from Packet)

Table 5.16: Pattern for Route Optimisation from CN to MN

The firewall installs the firewall pinhole rule specified in Table 5.16 in order to let the return traffic traverse. This pattern allows all route optimised traffic coming from the CN to the MN to pass through. Additionally, the firewall installs a second firewall pinhole rule in order to also let the data traffic from the MN to the CN traverse. This firewall pinhole is specified in Table 5.17. This pattern permits all route optimised traffic coming from the MN to the CN to traverse.

Source Address:	Source Address of the Packet (MN CoA)
Destination Address:	Destination Address of the Packet (CN)
Next Header:	60 (IPv6 Destination Options Header)
Home Address Dest. Option:	MN HoA (from Packet)

Table 5.17: Pattern for Route Optimisation from MN to CN

A firewall at the edge of the MN's MSP installs the firewall pinhole rule shown in Table 5.18 when receiving a Binding Update in order to let the incoming bi-directional tunneled data traffic traverse the firewall.

Destination Address:	Source Address of the Packet (MN HoA)
Source Address:	Destination Address of the Packet (CN)

Table 5.18: Pattern for Bi-directional Tunneling

5.4.4 Summary

This section has proposed one possible Mobile IPv6 firewall traversal solution to overcome the problems and impacts when having firewalls placed in Mobile IPv6 envi-

ronments, as described in Chapter 3. The solution based on *Application Layer Gateway* (ALGs) [SH99] techniques and was developed and standardised within the IETF *MEXT working group* [MEXT], in [KSYB08] and [KSSB08].
Therefore, it firstly has described the requirements and assumptions for the *Mobile IPv6 Application Layer Gateway* firewall traversal approach. The proposed approach itself is divided into two parts, more precisely two guidelines. Section 5.4.2 has presented the guidelines for firewall administrators regarding Mobile IPv6 traffic, i.e., which type of traffic has to be allowed by pre-configuration. Section 5.4.3 has presented guidelines for firewall vendors to help them implement their firewalls in a way that permits Mobile IPv6 signaling and data messages to traverse. It should be noticed, that only a combination of these two parts can achieve *Application Layer Gateway based Mobile IPv6 firewall traversal*.
Obviously, authentication and authorisation is an important aspect for firewall traversal. The authentication and authorisation aspects are discussed in detail in Section 6.2. Section 7.2 presents an overview about the *Mobile IPv6 Application Layer Gateway* firewall traversal proof-of-concept implementation, developed for this thesis.

6 Security, Authorisation and Authentication Considerations

One major aspect for firewall traversal in general or for firewall signaling is how to ensure that only authorised hosts are allowed to perform actions. This leads to the security requirement to provide authentication and authorisation.
This chapter discusses the security, authorisation and authentication aspects which have to be considered for the Mobile IPv6 firewall traversal solutions, proposed in Chapter 5. Section 6.1 discusses the security aspects for the *NSIS based Mobile IPv6 firewall traversal* solution and presents several potential solutions to deal with it. Section 6.2 discusses the security aspects for the *Mobile IPv6 Application Layer Gateway* firewall traversal solution.

6.1 NSIS Based Mobile IPv6 Firewall Traversal

An NSIS and NAT/FW NSLP based approach in general poses two fundamental security requirements. The first requirement is a mechanism which allows a firewall to trust firewall traversal signaling from a node as well as the corresponding firewall pinhole rules. The second requirement is to enable the firewall traversal signaling to be transported in a secure manner, for instance in a secure channel, to circumvent malicious nodes to listen or tamper firewall signaling.
The NSIS framework [HKLdB05] consists of two layers, the message transport layer and the signaling application layer, in this case the NAT/FW NSLP. As the NAT/FW NSLP bases on the NSIS framework, two security issues need to be considered. The first issue is to apply certain authentication mechanisms in the message transport layer to establish trust relationship between two nodes. GIST may either drop a request, or deliver it together with the available authenticated information about the initiator to the NSLP for further processing. The second issue is to deploy service authorisation mechanisms in the NAT/FW signaling application layer, for instance by introducing an authorisation entity. This authorisation entity can make authorisation decisions about what entity can be authorised and what kind of resource that entity is allowed to access. This decision is taken with the help of authentication identifier. For example, the NAT/FW NSLP could ask this identity from an access control list to identify what kind of firewall pinhole rules are allowed to create by an initiator node

respectively identity. However, authenticating an identity or identifier may not be necessary in every case, e.g., if the authorisation decision is not based on the identity or identifier.

[MSTB08] basically specifies how authorisation could be achieved for the QoS NSLP [MKM08] and the NAT/FW NSLP [STAD08] using an authorisation token based approach. The approach re-uses the authorisation token format specified for RSVP [BZB+97] and allows to exchange information between nodes in order to authorise access to resources, e.g., firewall pinholes.

There are several other existing security mechanisms which can potentially be used to secure the NAT/FW NSLP signaling and communication. However, each security mechanism or infrastructure subjects to certain technical assumptions and might only be applied in certain scenarios. Therefore, a single solution might not be enough for all use cases. Thus, in addition to the already proposed mechanism in [MSTB08], this section shortly discusses several potential authentication and service authorisation architecture solutions, namely:

- Transport Layer Security (TLS) with X.509 Public Key Infrastructure (PKI),
- Extensible Authentication Protocol (EAP),
- Authorisation Tokens,
- Security Assertion Markup Language (SAML),
- Generic Bootstrapping Architecture (GBA),
- Generic Service Authorisation and Bootstrapping Architecture (GSABA).

6.1.1 Transport Layer Security with X.509 Public Key Infrastructure

Transport Layer Security (TLS) [DA99] is a widespread security approach to protect transport- and application-layer protocols. TLS supports authenticating of parties using *Public-Key Certificates* [DA99], *Pre-Shared Keys* [ET05], and *Kerberos* [MH99]. TLS is usually run over TCP [Pos81c] or SCTP [SXM+00], but *Datagram TLS* [RM06] also allows the usage of unreliable transport protocols like UDP [Pos80]. The following shortly describes how TLS authenticated using certificates could be utilised to protect GIST messaging over TCP in a *X.509* [CSF+08] *Public Key Infrastructure* (PKI). The detailed authentication procedure is exemplified in Figure 6.1.

- When a NSIS signaling layer protocol, e.g., the NAT/FW NSLP, is started, it must at least configure the keys or certificates for establishing and accepting messaging associations protected with TLS.

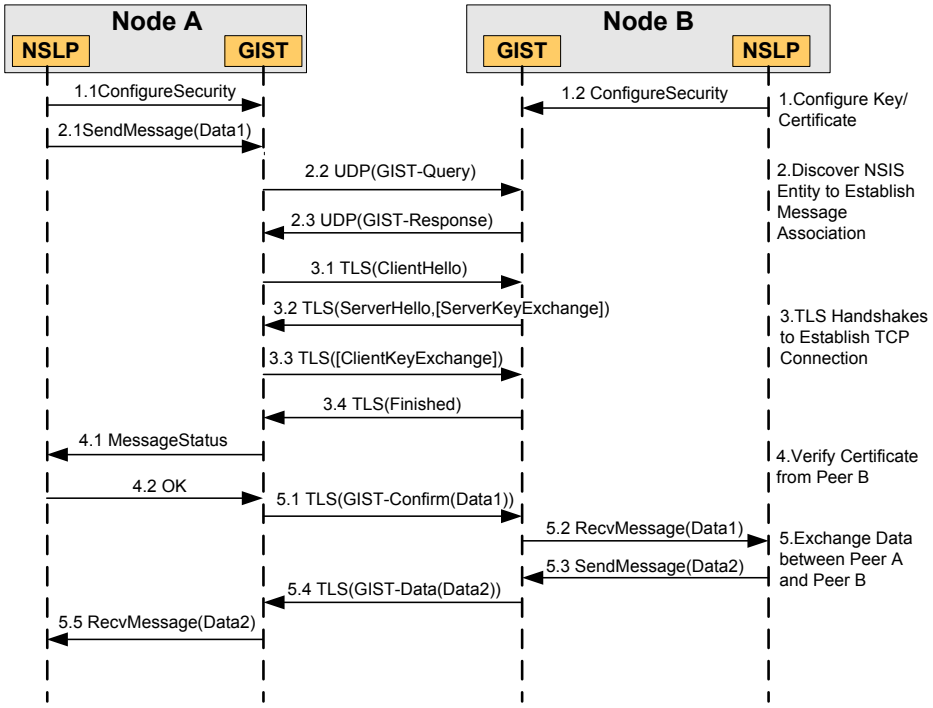


Figure 6.1: Example Authentication Procedure for TLS with X.509 PKI

- The NSLP on Node A starts to send a message. The local NSIS NTLP notices that a new messaging association is required and initiates QUERY/RESPONSE with Node B.
- Afterwards, Node A establishes a TCP connection to Node B and starts the TLS handshake.
- The NTLP on Node A asks the local NSLP to verify the certificates from Node B.
- If the certificates are verified, data exchange between Node A and Node B takes place.

Obviously, TLS depends on certificates to derive key materials. However, it is difficult to integrate TLS using the X.509 PKI within the existing AAA infrastructure. Therefore, this approach is suitable when an existing authentication infrastructure based on X.509 PKI certificates is encountered, which is clearly not the case for all possible NSIS scenarios. The assumption that all users have X.509 certificates is unrealistic in many environments. However, the approach allows flexibility in terms of what kind

of semantics the certificates have. X.509 certificates bind together one or more identifiers and a public key, but they might have additional authorisation semantics. The certificate either implicitly or explicitly authorises to perform some kind of actions, e.g., to install some kind of firewall pinhole rules.
The TLS with X.509 PKI approach can be used to implement authentication between two neighbouring nodes. However, it is not appropriate to provide authentication in foreign networks as not all middleboxes install certificates.

6.1.2 Extensible Authentication Protocol

To support different deployment scenarios, NSIS signaling should have flexible authentication and authorisation capabilities. A possible solution is to interact with the AAA infrastructure for computing the authorisation decision of various NSLP requests, e.g. NAT/FW NSLP requests. The *Extensible Authentication Protocol* (EAP) [ABV+04] is a common solution to support existing credentials in different usage scenarios. EAP provides the following capabilities [TE06]:

- Flexible support for authentication and key exchange protocols.
- Ability to re-use existing long-term credentials and already deployed authentication and key exchange protocols.
- Integration into the existing AAA infrastructure, namely *RADIUS* [RWC00] and *Diameter* [EHZ05].
- Ability to execute the authorisation decision at the user's home network based on the capabilities of protocols as RADIUS and Diameter.

Therefore, it needs to be considered how EAP authenticated TLS exchange can be used to protect GIST message over TCP. The authentication procedure of *TLS Inner Application* (TLS/IA) [FBWS+06] is exemplified in Figure 6.2, and can be illustrated with the following steps:

- When an NSLP is started, it must at least configure the keys or certificates for establishing and accepting messaging associations protected with TLS.
- When the NSLP on Node A sends a message; the local NTLP notices that a new messaging association is required, and initiates QUERY/RESPONSE with Node B.
- Node A establishes a TCP connection to Node B, and starts the TLS handshake with Node B. During this TLS handshake, Node B initiates the EAP process with AAA infrastructure to obtain server key material.

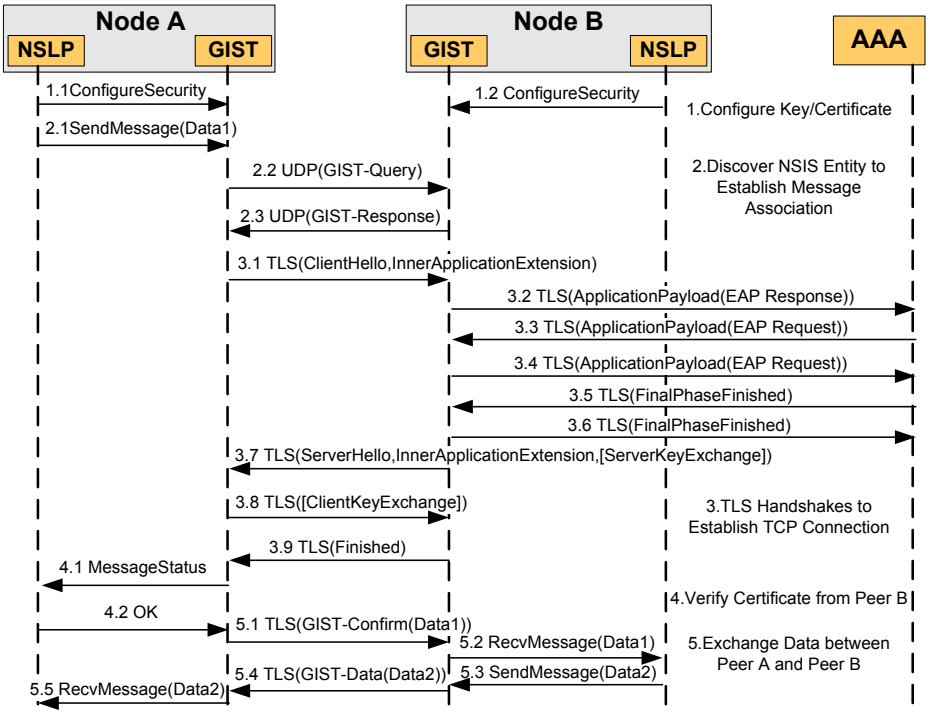


Figure 6.2: Example Authentication Procedure for TLS/IA

- Node A verifies the certificate or key material from Node B.
- Afterwards, data exchange between Node A and Node B takes place.

The main benefit of the TLS/IA approach is that it does not require modifications in the NSIS protocol suite. This bases on the fact that the EAP exchange is encapsulated within the TLS handshake exchange. The AAA interaction, triggered as part of the TLS/IA (and the EAP method processing), performs only authentication to provide key materials to the third party. When authentication and key exchange is performed with TLS/IA, the NSLP payload is not yet processed.
Hence, authorisation of specific NSLP operation (such as a request for firewall pinholes) can only be provided at the NSLP layer. Therefore, additional key establishment and a separate exchange might be required at the NSLP. This is demonstrated with the NAT/FW NSLP and gives an example how the EAP integration into NSLP can be used to protect NSLP. The authentication procedure of the EAP in NSLP approach is shown in Figure 6.3 and demonstrates how the EAP integration into the NSLP can be used to protect the NAT/FW NSLP.

- Node A sends a CREATE message with EAPoL-Start option included.

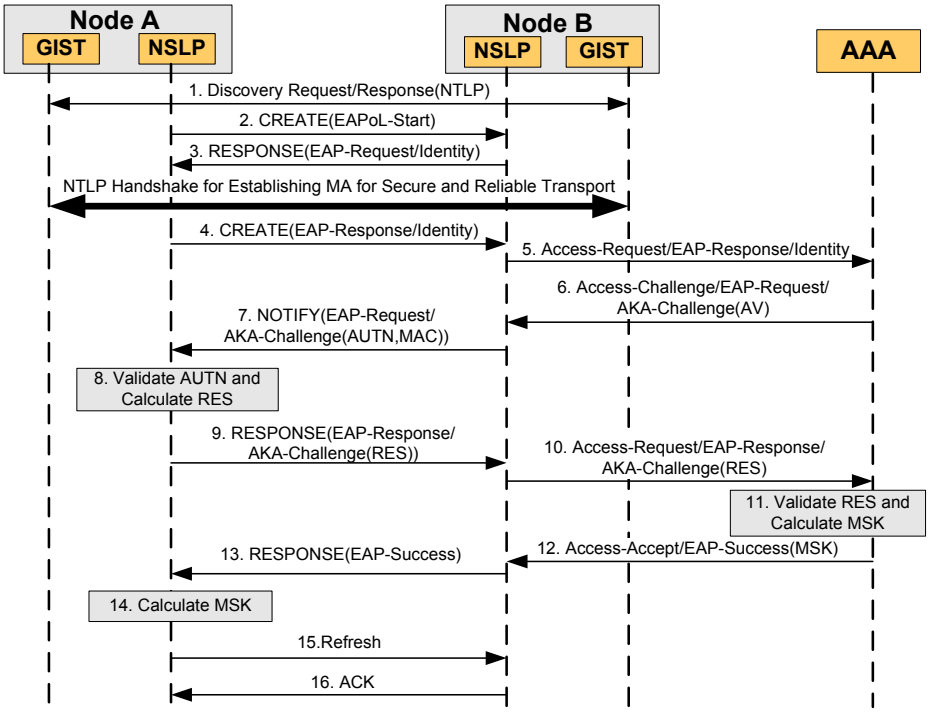


Figure 6.3: Example Authentication Procedure for EAP in NSLP

- Node B replies with a RESPONSE message with EAP-Request option for Node A's identity.
- Node A sends a CREATE message with EAP-Response option and it's identity included.
- Node B forwards the EAP-Response option to the AAA in *Access Request*.
- AAA initialises *Access Challenge* with AKA challenge to Node A.
- Node B forwards AKA challenge in NOTIFY message to the Node A.
- Node A validates the authentication token and calculates RES.
- Node A replies with a RESPONSE message; the AKA-challenge is included in the EAP-Response option.
- Node B forwards the EAP-Response to the AAA.
- The AAA validates the RES in the AKA challenge and calculates MSK.

- The AAA replies with Access-Accept message and EAP-success option included.
- Node B forwards EAP-Success in RESPONSE message to Node A and Node A calculate MSK.

This approach allows a seamless interworking between EAP and NAT/FW NSLP protocol with a proper encapsulation of the EAP payloads into NAT/FW NSLP messages. However, EAP methods should deal with the fragmentation, reliability and re-transmission of the EAP data into the NAT/FW NSLP messages. For integrating EAP into the NAT/FW NSLP, a new NSLP payload object should be defined to carry the EAP packets. The EAP session will be established between a NAT/FW NSLP initiator, a NAT/FW NSLP policy aware node and an authorising entity. A NAT/FW NSLP policy aware node should be authenticated and authorised to be one side in an AAA NAT/FW NSLP authorisation session. In addition, the authorisation decision is based not only on an authenticated identity, but also on the description of requested NAT/FW rules parameters, which would clearly identify the type of the requested service.
Both of the described EAP authentication procedures build trust relationship between two neighbouring nodes. The difference between them is that EAP is carried in different layers. The EAP in NSLP approach can be used to generate session credentials that secure NSLP signaling from being tampered by malicious nodes, while for the TLS/IA approach, key materials can not be easily obtained from GIST layer to secure NSLP signaling.
However, the integration with EAP is prepossessing since the TLS server is able to relay EAP payloads to the existing AAA infrastructure to offload authentication, authorisation and accounting tasks. Another advantage of using TLS-EAP is the smooth integration with the AAA infrastructure and the simple enhancements needed for EAP integration into TLS due to the extensibility of the NSIS framework. A weakness of the TLS-EAP approach is the additional message exchange since EAP is quite chatty.

6.1.3 Authorisation Token

Authorisation Token is one potential application of authorisation assertions and trait-based authorisation [PPST06]. A user is authenticated and authorised through one protocol, and can re-use the resulting authorisation assertion in other, unrelated protocol exchanges.
Authorisation token based security mechanisms are usually used to complete authentication and authorisation with other security infrastructures. This considers a basic authorisation framework based on authorisation token and the authorisation procedure is described as follows:

- A node requests an authorisation token from a third grant entity.
- The third grant entity replies to the node with the corresponding token.
- The node sends an authorisation request to assertion verification entity with the node's token.
- The assertion verification entity exchanges messages with the third grant entity to validate the token.
- If the token is valid, the assertion verification entity responds to the node with the authorisation result.

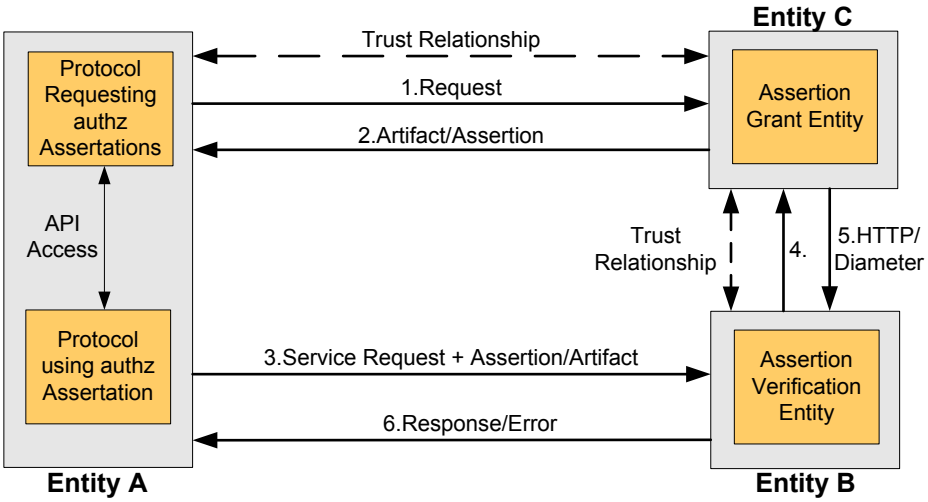


Figure 6.4: Authorisation Framework Based on Authorisation Token

The authorisation token approach allows the third party to make authorisation decisions on what entity can be authorised and what resource that entity is allowed to access, e.g., firewall pinholes rules.
According to [MSTB08], the authorisation token format is defined as depicted in Figure 6.5. The authorisation token mainly consist of the authorisation identifier, the source and destination address, the session start the end time, and the authentication data. The firewall policy element of the session is included in the authentication data field.
Different authentication mechanisms have different formats for the authorisation token. This approach considers two types of authorisation token formats, one is a symmetric shared key based authorisation token, the other is a public key based authorisation token. The main difference between the two token types is that different

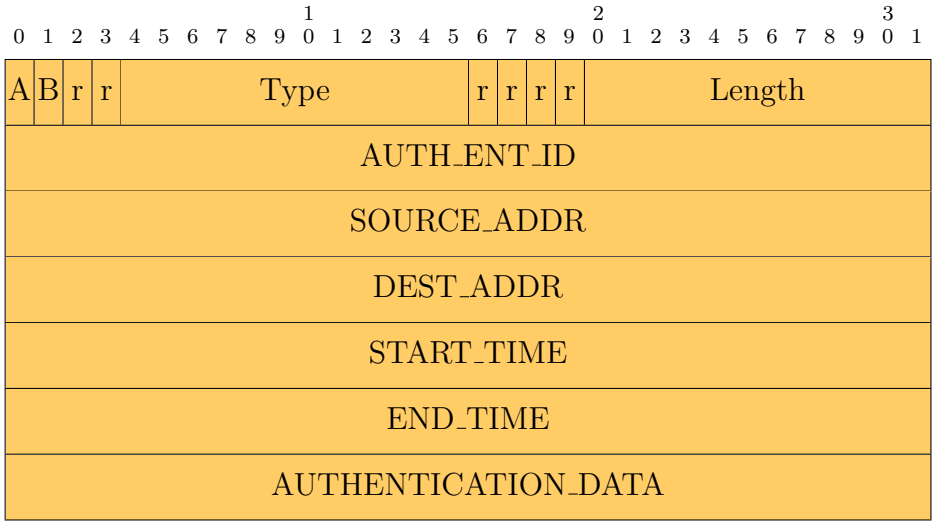


Figure 6.5: Authorisation Token Format

key environments have different *AUTH_ENT_IDs*. In a symmetric shared key environment, a token can be used with the AAA infrastructure; *AUTH_ENT_ID* is usually the IPv6 address, the *Fully Qualified Domain Name* (FQDN) [EB97] or the *Network Access Identifier* (NAI) [PLK+05]. While in public key environment, a token can be used with digital certificates, *AUTH_ENT_ID* is usually the *X509_V3_CERT* [CSF+08] or *PGP_CERT* [CDFT98] (Pretty Good Privacy).

6.1.4 Security Assertion Markup Language

The *Security Assertion Markup Language* (SAML) is an XML-based framework for creating and exchanging security information. In the course of making, or relying upon such assertions, SAML system entities may use SAML protocols, or other protocols, to communicate with an assertion itself, or the subject of an assertion.
Thus one can employ SAML to make and encode statements such as "Alice has these profile attributes and her domain's certificate is available over there, and I'm making this statement that she is allowed to traverse firewalls within this particular domain." Then, an end-host can cause such an assertion to be conveyed to some party, for example a firewall, who can rely on it for computing an authorisation decision, for example using it as input into some local policy evaluation for granting the establishment of a firewall pinhole.
A possible approach of applying SAML for NAT/FW NSLP signaling in Mobile IPv6 environments is as follows: The MN first asks the *Identity Provider* (IdP) to get such an assertion before starting the signaling with the firewall. The IdP will authenticate the user or end-host and will return an assertion in case of success. When interacting

with a firewall, the MN will attach the SAML assertion to the message. The firewall verifies whether the assertion is valid and, if the MN is authorised, perform the indicated action (e.g., creating a firewall pinhole) for further communication. An error is returned in case the end-host is not authorised. Despite the popularity of SAML for identity management there is also a disadvantage, namely the large size of the XML-based assertions when conveyed by value. To overcome this limitation a reference to a SAML assertion can be used instead. In this case, the firewall has to resolve the reference firstly to obtain the assertion, for example using an HTTP lookup.

6.1.5 Generic Bootstrapping Architecture

The *3GPP Generic Bootstrapping Architecture* (GBA) [3GP08b] is an authentication system with three parties. A trusted third party (*Bootstrapping Server Function* or BSF), is involved in authentication and key exchange between two other nodes, the client (*User Equipment* or UE) and the server (*Network Application Function* or NAF).
Goal of the GBA architecture is to isolate the knowledge of the long-term secrets and the credentials to one single trusted node, the Bootstrapping Server Function. Thus, the *actual servers* (NAFs) do not need to have access to the clients' long-term credentials. Additionally, the NAF does not need to know the type of credentials used between the client and the BSF.
The GBA approach uses authentication entity identifiers to request key material from the AAA infrastructure and establishes trust relationship with an authenticated entity. A possible approach is to use *Pre-shared Key TLS* [ET05] like *GAA HTTPS services* [3GP08a], but this requires the querying node to know the *Fully Qualified Domain Name* (FQDN) [EB97] in advance. As a solution, the FQDN could be obtained by the GIST-RESPONSE message. In this case, the node must be able to verify the FQDN.
Figure 6.6 depicts how the *3GPP Generic Bootstrapping Architecture* can be used to protect GIST messages.
The *3GPP Generic Bootstrapping Architecture* [3GP08b] provides a potential solution for securing the NSIS and NAT/FW NSLP signaling. However, it lacks support for service authorisation. Therefore, the *Generic Service Authorisation and Bootstrapping Architecture* (GSABA) [KTF+06] should be considered as a more promising candidate.

6.1.6 Generic Service Authorisation and Bootstrapping Architecture

The *Generic Service Authorisation and Bootstrapping Architecture* (GSABA) [KTF+06] is a generic framework for service authorisation and bootstrapping. It is integrated in the existing AAA infrastructure and provides the

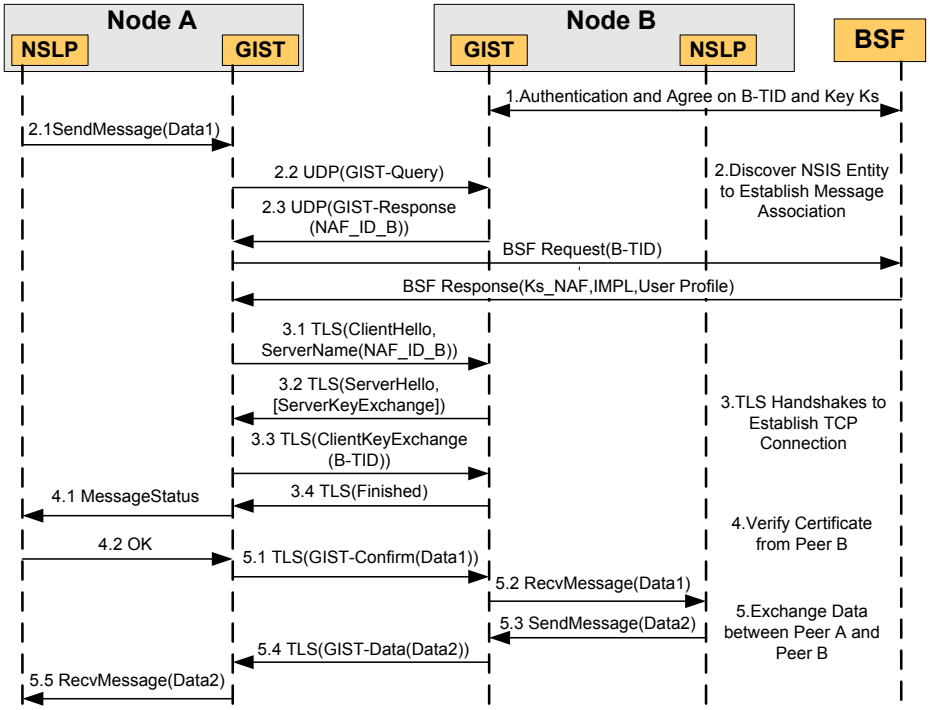


Figure 6.6: Example Authentication Procedure for GBA

end-host as well as the service providing server with the required information for service access based on credentials.

Figure 6.7 illustrates the basic architecture elements of GSABA. The main component is the *Bootstrapping Agent* (BA), which consists of the bootstrapping and service configuration function (*Bootstrapping Configuration Agent* or BCA) and the service authorisation function (*Bootstrapping Authorisation Agent* or BAA). Both functions could be co-located on the same entity or could be distributed. The BCA is responsible for providing necessary bootstrapping information to the mobile node. The BAA is responsible for asserting authorisation decision statements. In addition, the BAA also has to authenticate the MN.

The decision statements are based on the MN's profile, which is available in the authorising domain. In the roaming case (i.e., the authorising domain is not the same as the service providing domain), the BAA proxy located in service providing domain proxies authentication and authorisation requests to the BAA of the authorising domain. The statements and the parameters need to be conveyed to the MN. The MN communicates with the BCA to obtain bootstrapping and service authorisations. The MN performs the actual *Service Protocol* (SP) and thus consumes the service.

The authorisation decisions are taken in the GSABA server and the authorisation

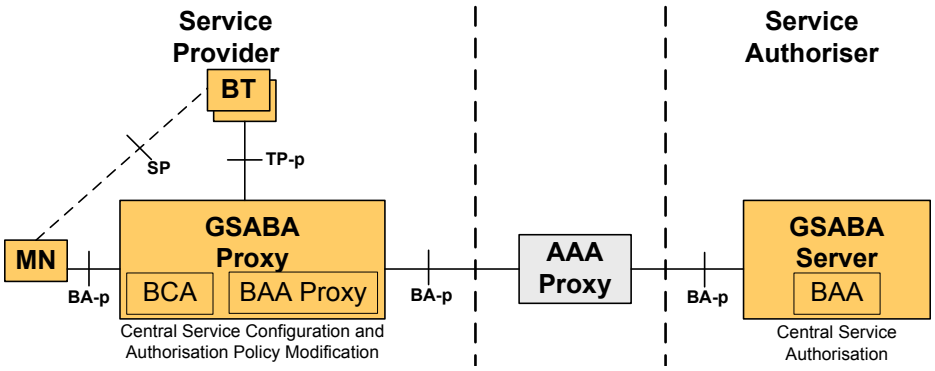


Figure 6.7: The GSABA Architecture

statement will be relayed to the GSABA proxy which the MN is connected to. The *BA-p* interface is the interface between the GSABA proxy and the GSABA server. As in the existing AAA infrastructure, the *BA-p* interface can include multiple AAA proxies which do not have to be aware of the GSABA.
The *Bootstrapping Target* (BT) is the entity that offers the requested service. It is responsible for obtaining the service specific information and MN related information from the BCA in order to execute the service protocol and provide the service to the MN. The interface *TP-p* is responsible for interaction between the GSABA proxy and the BT. It is used to deliver the bootstrapping data (i.e., service key and the authorisation information). The transport protocol can be *RADIUS* [RWC00] or *Diameter* [EHZ05] with the AAA security mechanisms.
In Mobile IPv6 firewall traversal scenarios, as shown in Figure 6.8, the firewall acts as the Bootstrapping Target and the GIST traffic is secured by TLS/PSK.

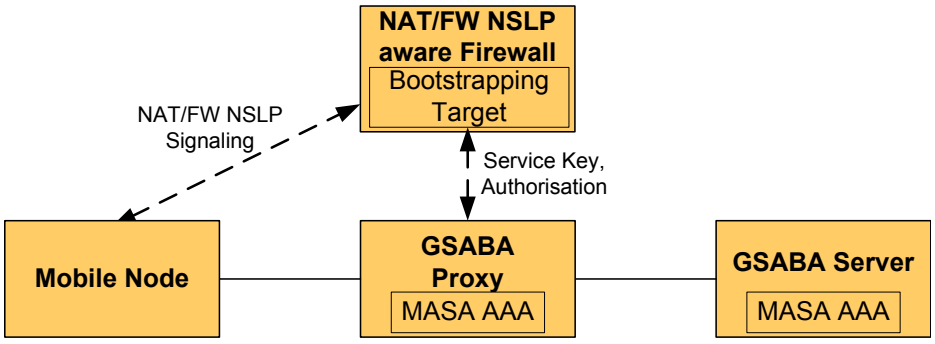


Figure 6.8: Example Authentication Procedure for GSABA

The NAT/FW NSLP service key is derived from the GSABA key. The NSIS and

NAT/FW NSLP aware firewall has to verify the authenticated entity based on the service key. The GSABA proxy/server is in charge of the delivery of the service key for user authentication and the protection of the signaling traffic between the MN and the firewalls. Besides, the GSABA proxy/server is also in charge of the authorisation of the firewall pinhole configuration on the firewalls.

As discussed before, the set up of the security association between the MN and the firewall may be done with a full EAP exchange transported over NSIS. Since GSABA enables an integration of the service bootstrapping with the network access authentication, this can be optimised by deriving keying material from the initial EAP authentication carried out during the network access phase, thus the message exchanged can be slashed. It is not necessary to go back to the home network for keying material during the firewall traversal phase if the MN has already been authenticated by the visited network.

From the GIST point of view, TLS with PSK is used to secure reliable messaging over TCP. A new MA-Protocol-ID value, "TLS/PSK" may need to be assigned to allow the ciphersuite negotiation in the GIST QUERY/RESPONSE phase.

The integration of GSABA into the NSIS and NAT/FW NSLP based Mobile IPv6 firewall traversal requires the investigation of the three basic scenarios, as introduced in Chapter 3.

6.1.6.1 Firewall Located at the Edge of the MN's ASP

When the MN is connected to an ASP, it has to be authenticated against the GSABA server in the home network. After a successful access authentication, the GSABA key has been derived by the MN and the GSABA server and transported to the GSABA proxy. This key is used as the root key to derive service specific keys for the bootstrapping of further services. When the MN wants to install firewall pinhole rules at the firewall, it starts bootstrapping the NSLP application with the NSIS and NAT/FW NSLP aware firewall. Therefore, the MN derives the GIST key from the GSABA key and starts a GIST TLS/PSK handshake with the firewall. The firewall fetches the GIST key from the GSABA proxy and authenticates the MN. After establishing the secured GIST TLS tunnel, the MN could start the NAT/FW NSLP signaling; e.g., sending a CREATE/EXT message through the firewall to the local HA. The firewall fetches the authorisation decision statements from the GSABA proxy after receiving the CREATE/EXT message.

Since opening firewall pinholes is usually a local decision which is taken by the domain where the firewall is located, the home domain is not involved in the authorisation.

6.1.6.2 Firewall Located at the Edge of the CN's ASP

In this scenario, the MN is typically located in a different network than the CN, while there is a firewall located at the edge of the CN's ASP. When the MN signals the CREATE message, the firewall at the CN's ASP rejects the CREATE request and responds with an error message, since the MN can not be authenticated.
In order to establish the required security association, the firewall includes its domain name in the responding error message. The MN now derives a specific NSLP key bound to this domain name from the GSABA key. The MN creates an *AUTH Object* which is protected by that NSLP key and will be used to authenticate itself in the CN's domain. The MN re-sends the CREATE message with the AUTH object included. At this time when the firewall receives this message, it can fetch the NSLP key from the GSABA server through the GSABA infrastructure. This requires that the MN shares a trust relationship with the CN's ASP and in turn with the firewall (e.g., the MN is an employee client that has moved outside of the enterprise network, or the MN's visited network has a roaming agreement with the CN's ASP). Thus, the firewall is able to authenticate the MN, authorise the request based on the CN's profile, and forward the CREATE message towards the CN.
It is important to notice that the NSLP key is bound to the whole domain rather than the edge firewall itself. This allows the NAT/FW NSLP signaling to traverse multiple firewalls within the CN's domain without deriving individual keys for each firewall. In the case that firewalls which belong to multiple domains are located on the path, different keys have to been generated for each domain.

6.1.6.3 Firewall Located at the Edge of the MN's MSP

Similar to the scenario described in Section 6.1.6.1, the MN derives the GIST key from the GSABA key and starts the GIST TLS/PSK handshake. The firewall fetches the GIST key as well as the authorisation decision statements from the GSABA proxy to authenticate the MN and authorise the CREATE/EXT request.

6.1.7 Summary

From the security perspective, when the NAT/FW NSLP is used to realise Mobile IPv6 firewall traversal, there is a need to establish secure tunnels between the arbitrary two neighbouring nodes before delivering NSLP messages. Furthermore, firewalls need to authenticate NSLP message before allowing NSLP messages to traverse. Using the existing security infrastructure, trust relationship between nodes can be established in two layers.
Transport Layer Security (TLS) [DA99] with *X.509* [CSF+08] *Public-Key Certificates* [DA99], as described in Section 6.1.1, is suitable when an existing authentication

infrastructure based on X.509 certificates is present. This is clearly not the case in all possible NSIS based Mobile IPv6 firewall traversal scenarios.

Extensible Authentication Protocol (EAP) [ABV+04] and *3GPP Generic Bootstrapping Architecture* (GBA) [3GP08b] approaches, as described in Section 6.1.2 and Section 6.1.5, are appropriate to establish trust relationship between GIST layers or between NSLP layers. However, two exchanges in two layers are completely isolated, which easily leads to man-in-the-middle attacks.

Authorisation Token (Section 6.1.3) is usually used to make authorisation decisions with other authentication framework and can not be used independently.

The *Security Assertion Markup Language* (SAML), as described in Section 6.1.4, has several drawbacks which limit the usage for NSIS based Mobile IPv6 firewall traversal, e.g., the large size of the XML-based assertions when conveyed by value. Additionally, there is no SAML *profile* or *binding* defined that describes in detail how SAML assertions are carried within NSIS.

In the *Generic Service Authorisation and Bootstrapping Architecture* (GSABA) [KTF+06], as described in Section 6.1.6, the transport layer is secured by TLS in the message transport layer. Additional service authorisation based on authorisation tokens can be provided at the signaling layer and the PSK is shared by both, the transport layer, and the signaling layer. Thus, it provides a generic service authorisation and authentication framework.

As a result, GSABA is the most favorite approach for securing the NSIS and NAT/FW NSLP based Mobile IPv6 firewall traversal approach. The detailed message flows based on GSABA are described in Section 6.1.8.

6.1.8 Message Flows for the NSIS Based Mobile IPv6 Firewall Traversal with GSABA Integration

This section presents in detail the message flows for the *NSIS based Mobile IPv6 firewall traversal* approach with GSABA integration, as introduced in Section 6.1.6. It should be noticed that this section presents the NSIS firewall signaling as specified in Section 5.3.5, not the improved *NSIS based Mobile IPv6 firewall traversal*, as described in Section 5.3.7. However, the message flow is similar to this approach, apart from the fact that the NSIS firewall signaling to the CN is processed by it edge firewall.

This section does not present the complete message flows as specified in Section 5.3.5, it only presents the initial flows and the required GSABA authorisation and authentication. The later following NSIS firewall signaling is as specified in Section 5.3.5 and non GSABA authorisation and authentication would take place as the involved nodes are already authorised and authenticated at this point. Nevertheless, the section shows one example message flow for CREATE and EXT respectively.

6.1.8.1 Firewall Located at the Edge of the MN's ASP

Figure 6.9 and Figure 6.10 illustrate the service authorisation message flow of the NSIS based Mobile IPv6 firewall traversal in the scenario, where a firewall is located at the edge of the MN's ASP.

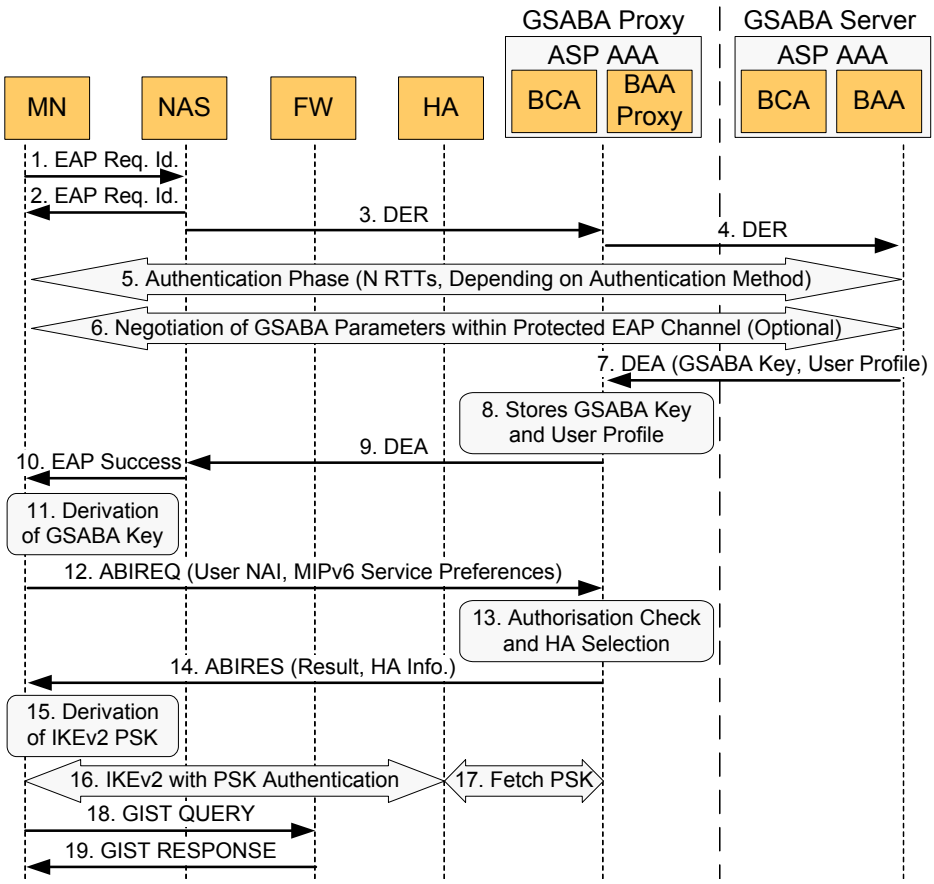


Figure 6.9: Firewall Located at the Edge of the MN's ASP (I)

Steps (1) to (11) show the procedure for network access authentication. Upon successful authentication of the MN's credential, as depicted in steps (1) to (6), the GSABA server replies with a *Diameter-EAP-Answer* (DEA) with the Result-Code AVP indicating the success (steps (7) and (9)). The message also includes the GSABA key and the user profile of the MN (i.e., the list of services that the MN is allowed to access from the ASP).
After the successful EAP network authentication, the MN derives the GSABA key locally (step (11)). The GSABA key serves as the root key for the derivation of other keys to secure subsequent message changes with different entities (i.e., GSABA proxy,

HA, and firewalls).
The MN derives the BCA-p key from the GSABA key for the establishment of a secure channel with the GSABA proxy to request bootstrapping of Mobile IPv6 by sending an *ABIREQ* message (step (12)). Based on the user profile, the GSABA proxy performs the Mobile IPv6 authorisation check. In this scenario, a local HA is selected (step (13)) and communicates to the MN in the ABIRES messages (step (14)).
The MN bootstraps the IPsec SAs with the HA assigned by the GSABA proxy. Therefore, the MN can either perform an additional EAP exchange, or as an optional optimisation, derive the IKEv2 PSK for IKEv2 authentication from the GSABA key (step (15)).

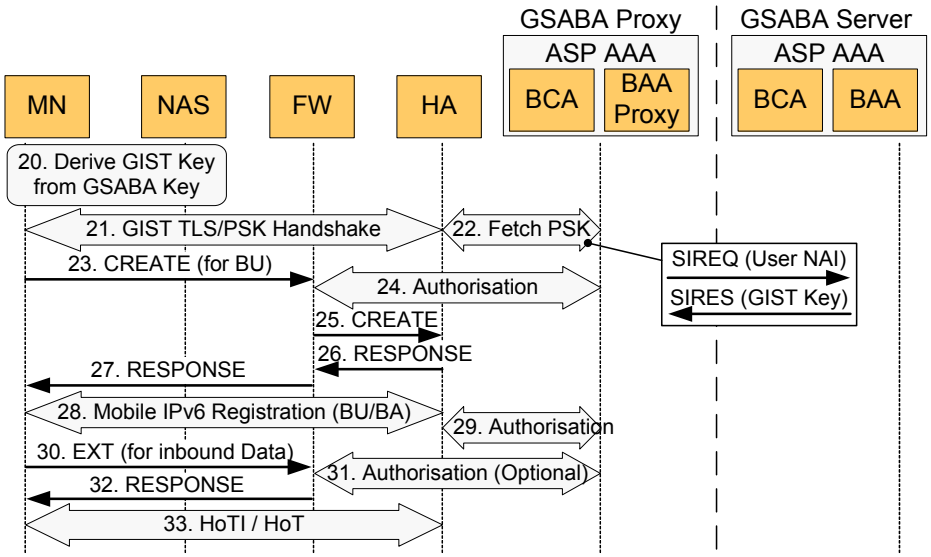


Figure 6.10: Firewall Located at the Edge of the MN's ASP (II)

To establish the MN and firewall security associations, the MN and the GSABA proxy derives a GIST key from the GSABA key (step (20)) in Figure 6.10). The GIST key is used as the shared key to establish TLS/PSK connections (step (21)) during the GIST QUERY/RESPONSE exchange. Upon receiving the TLS ClientHello message, the firewall fetches the GIST-key from the GSABA proxy using the *SIREQ/SIRES* messages over the TP-p interface (step (22)). Once the MN has been authenticated by the firewall and the TLS tunnel has been established, the firewall is able to process NSIS NAT/FW-NSLP messages sent by the MN.
As discussed in Section 5.3.5, the MN has to create a firewall pinhole on the firewall at the edge of its ASP to allow the IPsec packets to traverse, before sending the Binding Update. Upon receiving the NAT/FW NSLP CREATE message (step (23)),

the firewall has to perform the authorisation request/response exchanges with the GSABA proxy (step (24)). The firewall pinhole configuration at the firewall is a local decision, taken by the domain where the firewall is located, here, the ASP. In principle there is no need to go back to the home domain to request authorisation. The authorisation decision is made by the GSABA proxy based on the MN's user profile. If the request is authorised, the firewall forwards the CREATE message to the next NAT/FW NSLP hop on the path. When receiving a RESPONSE message from the HA, the firewall installs the firewall pinhole for the BU message and forwards the RESPONSE towards the MN. Now, the MN is able to update the Mobile IPv6 binding cache at the HA by sending the Binding Update and receiving the BA (step (28)).
Figure 6.10 also shows the signaling message flows for inbound route optimised data traffic as an example for a EXT message exchange (steps (30) to (32)). Every firewall pinhole request (e.g., EXT for inbound data, CREATE for BU/HOTI) has to be authorised by the network. The authorisation may be made locally by the firewall, if the ASP AAA has already authorised the MN for NAT/FW NSLP signaling in a previous step, as depicted in step (31).

6.1.8.2 Firewall Located at the Edge of the CN's ASP

Figure 6.11 shows the service authorisation message flow of the NSIS based Mobile IPv6 firewall traversal in the scenario, where a firewall is located at the edge of the CN's ASP.
In this scenario, the CN needs to establish a security association between the firewall and itself firstly. To open the firewall pinhole for the CoTI message, the MN sends a CREATE message to the CN. This CREATE message reaches the firewall first (step (1)). Since the firewall is unable to authenticate the MN at this point, the firewall rejects the CREATE message with an ERROR RESPONSE message (step (2)). The firewall also includes its domain name in the RESPONSE message.
Upon receiving the ERROR RESPONSE message, the MN knows about the firewall on the path against which the MN has to be authenticated. Therefore, the MN derives a NSLP key from the GSABA key and creates an AUTH object protected by the NSLP key (step (3)). The NSLP is bound to the domain name included in the RESPONSE message.
It is important to notice that the NSLP key is not bound to the firewall which rejected the first CREATE message, but bound to the domain where the firewall is located, here, the CN's ASP domain. This allows the NAT/FW NSLP signaling to traverse multiple firewalls which may be located within the CN's ASP without performing multiple firewall detection exchanges and deriving individual keys for each firewall. If multiple domains are involved, different keys have to be generated carrying out by the CREATE/RESPONSE exchanges.

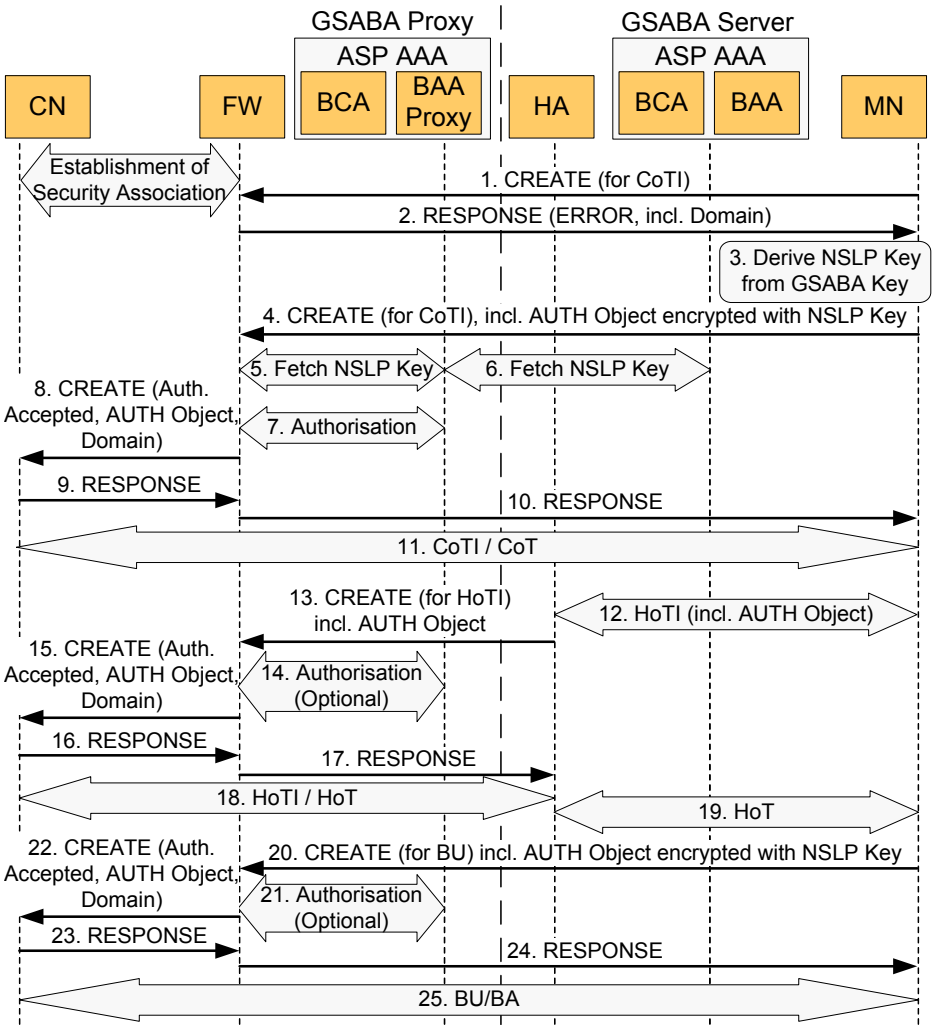


Figure 6.11: Firewall Located at the Edge of the CN's ASP

Now, the MN sends the CREATE message again, but includes the AUTH object in the CREATE message (step (4)). When the firewall at the edge of the CN's ASP receives this message with the additional AUTH object, it fetches the NSLP key of the MN from the GSABA proxy in order to authenticate the MN (step (5)). If the GSABA key is not available at the GSABA proxy, the GSABA proxy has to contact the GSABA server in the home domain and has to fetch the key (step (6)).
When the MN is authenticated and the CREATE message is authorised successfully (step (7)), the firewall forwards the CREATE message towards the CN (step (8)), which replies with a RESPONSE message (step (9) and (10)). The MN and the CN are now able to send the CoTI/CoT messages for route optimisation (step (11)).

The message flow for the HoTI message is different as the MN sends the HoTI message including the AUTH object to its HA (step (12)). The HA then sends a CREATE message for opening firewall pinholes to the firewall at the edge of the CN's ASP (step (13)). The firewall could now authorise the CREATE message from the MN's HA (step (14)). Afterwards, the CREATE message is forwarded to the CN (step (15)) which responses with the RESPONSE message (step (16) and (17)). Now the HoTI message can be send from the HA to the MN (step (18)) and the CN responses with the HoT message to the MN through the HA (step (18) and (19)). To be able to send the BU to the CN, the MN sends a CREATE message to the CN (step (20)). This message is again processed and authenticated by the firewall before reaching the CN (steps (21) and (22)). When the RESPONSE message from the CN reaches the MN (step (23) and (24)), the firewall pinhole for the BU is installed. The BU/BA messages between the MN and the CN now can traverse the firewall without problems (step (25)).

6.1.8.3 Firewall Located at the Edge of the MN's MSP

Figure 6.12 and Figure 6.13 illustrate the message flow for the service authorisation of the NSIS based Mobile IPv6 firewall traversal in the scenario, where a firewall is located at the edge of the MN's MSP.
The MN first needs to be authenticated by the GSABA server to get the GSABA key. This message flow is shown in Figure 6.12 (steps (1) to (8)).

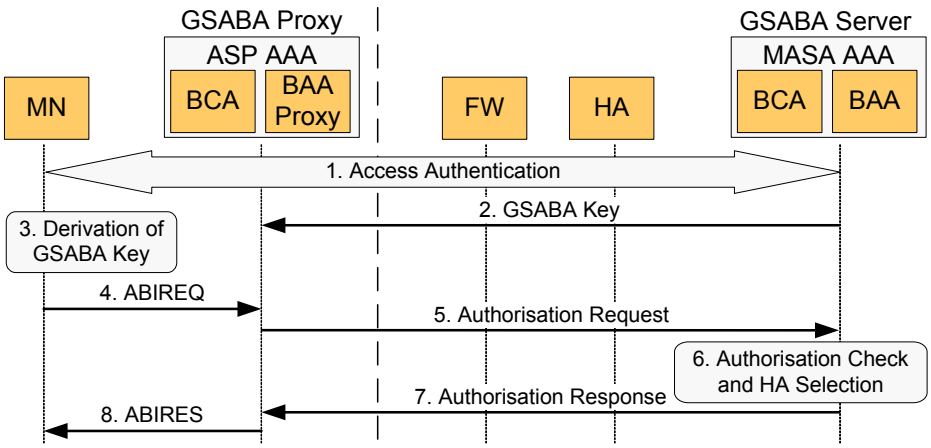


Figure 6.12: Firewall Located at the Edge of the MN's MSP (I)

Afterwards, the MN performs GIST QUERY/RESPONSE (steps (1) and (2)). To establish the MN and the firewall security associations, the MN and the GSABA proxy derive a GIST key from the GSABA key (step (3)). The GIST key is used

as the shared key to establish TLS/PSK connections during the GIST TLS/PSK handshake (step (4)).

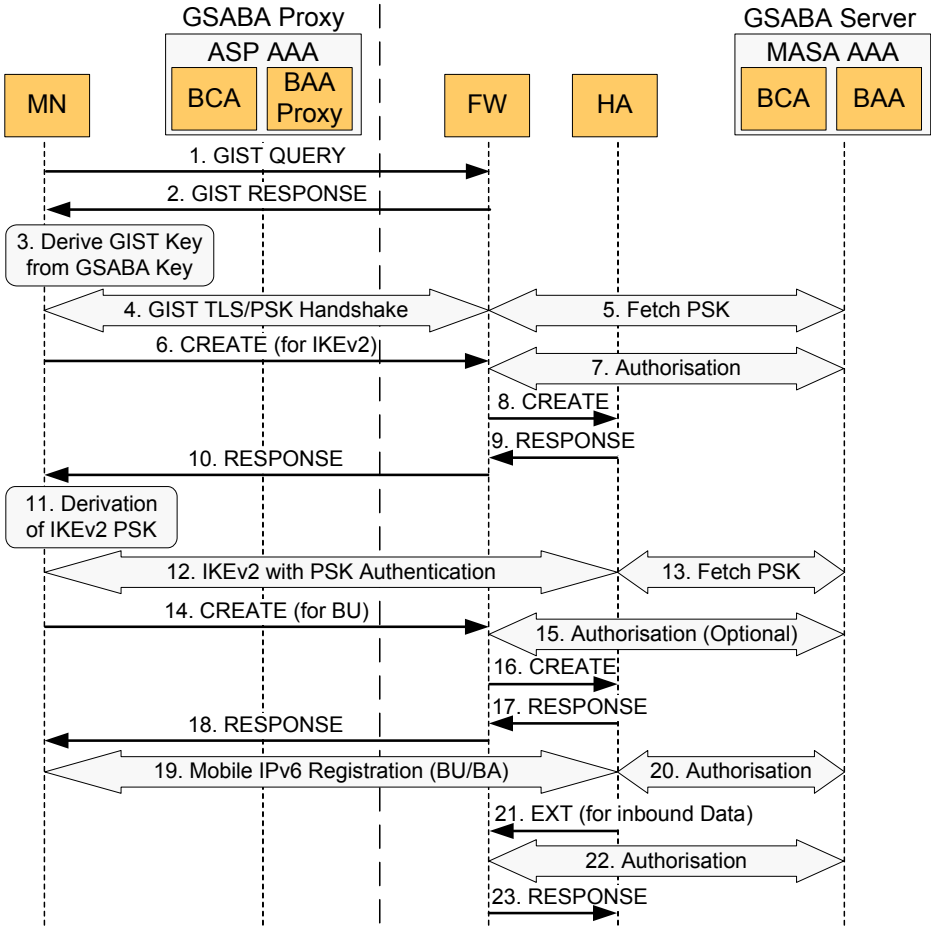


Figure 6.13: Firewall Located at the Edge of the MN's MSP (II)

The firewall fetches the PSK from the GSABA server (step (5)) and the MN sends a CREATE message to install firewall pinholes to allow IKEv2 signaling traffic to traverse the firewall at the edge of the MN's MSP (step (6)). The firewall checks the authorisation at the GSABA server (step (7)) and decides whether the CREATE message can traverse the firewall. If it is allowed to traverse, the firewall forwards the CREATE message to the HA (step (8)), which sends the RESPONSE message back to the MN (step (9) and (10)).
Afterwards, the MN derives IKEv2 PSK (step (11)) and performs the IKEv2 authentication against the HA using the derived PSK (step (12)). Therefore, the HA fetches the PSK from the GSABA server (step (13)).

As discussed in Section 5.3.5, the MN has to create a firewall pinhole on the firewall at the edge of its MSP to allow the IPsec packets to traverse before sending the Binding Update. Upon receiving the CREATE message (step (14)), the firewall optional performs authorisation with the GSABA proxy (step (15)). The authorisation decision is made by the GSABA proxy based on the MN's user profile (e.g., whether the user is permitted to configure firewall pinholes for Mobile IPv6 traffic). If the request is authorised, the firewall forwards the CREATE message to the HA (step (16)). When the RESPONSE message from the HA reaches the MN (steps (17) and (18)), the firewall pinholes for the BU are configured and the BU takes place (steps (19) and (20)).
As showed in Figure 6.13, the HA also installs firewall pinhole rules, e.g., for bi-directional tunneled data traffic using the EXT message (steps (21) to (23)).

6.2 Mobile IPv6 Application Layer Gateway

The *Mobile IPv6 Application Layer Gateway* approach as described in Section 5.4 specifies recommendations for firewall vendors and administrators to allow Mobile IPv6 traffic to pass through firewalls unhindered. It recommends a liberal setting of firewall pinhole rules so that all legitimate traffic is allowed to traverse. Since some of the traffic is encrypted it is not possible for firewalls to recognise whether it is safe or not. The *Mobile IPv6 Application Layer Gateway* approach recommends a liberal setting so that all legitimate traffic can traverse. This means that some malicious traffic may be permitted by these firewall pinhole rules. These rules may allow the initiation of *Denial of Service* attacks against Mobile IPv6 capable nodes, namely the MNs, CNs and the HAs.
One of the main goals of any firewall is to prevent unsolicited traffic from entering the network. The proposed solution allows such traffic into the network, albeit with a number of restrictions.
In a typical enterprise environment, an administrator can not distinguish Mobile IPv6 capable nodes from other nodes. In such a situation any node in the network or domain may end up receiving unsolicited packets from outside the firewall. The risk in this case is that such packets could trigger unknown vulnerabilities in any of these nodes, causing denial-of-service or worse attacks. This issue is compounded in a mobile service provider environment by the risks specific to such environments like end-point battery exhaustion and bandwidth misusage.
[GHJP00] specifies Mobile IP authentication, authorisation and accounting requirements, [LC04] specifies authentication, authorisation and accounting requirements for the *Session Initiation Protocol* (SIP) [RSC+02]. The requirements for a Mobile IPv6 firewall traversal solutions are very similar, however, further study on this issues is necessary for an Application Layer Gateway based Mobile IPv6 firewall traversal ap-

proach. Nevertheless, [LC04] shows how an authentication and authorisation solution for the Application Layer Gateway based approach basically could look like, whereas the *SIP Proxies* have to be replaced by the *Mobile IPv6 Application Layer Gateway* aware firewalls.

7 Implementation

This chapter presents the *NSIS based Mobile IPv6 firewall traversal* and *Mobile IPv6 Application Layer Gateway* firewall traversal proof-of-concept implementations developed as part of this thesis. The approaches have been described in detail in Section 5.3 and Section 5.4. This chapter focus on the proof-of-concept implementations.

7.1 NSIS Based Mobile IPv6 Firewall Traversal Implementation

The *NSIS based Mobile IPv6 firewall traversal* proof-of-concept implementation presented in this section is publicly available at [NSISMIP6]. It utilises the Open Source NSIS and NAT/FW NSLP implementations of the University of Göttingen, available at [FreeNSIS], the Mobile IPv6 implementation *MIPL 2.0.2* [MIPL] developed by *GO-Core* in co-operation with the *USAGI/WIDE Project* [USAGI] and the *netfilter/iptables* [netfilter] firewall implementation. The developed proof-of-concept implementation implements the NSIS based Mobile IPv6 firewall traversal approach as described in Section 5.3.5, not the improved NSIS based Mobile IPv6 firewall traversal, introduced in Section 5.3.6.

7.1.1 Reference Architecture

Figure 7.1 shows the NSIS based Mobile IPv6 firewall traversal software architecture. It should be noted that this reference architecture is based on the assumption that the MSA and the MSP are co-located.
The functional elements (orange rectangle) that compose this architecture are:

- Mobile Node (MN),
- Home Agent (HA),
- Corresponding Node (CN),
- Firewall (FW).

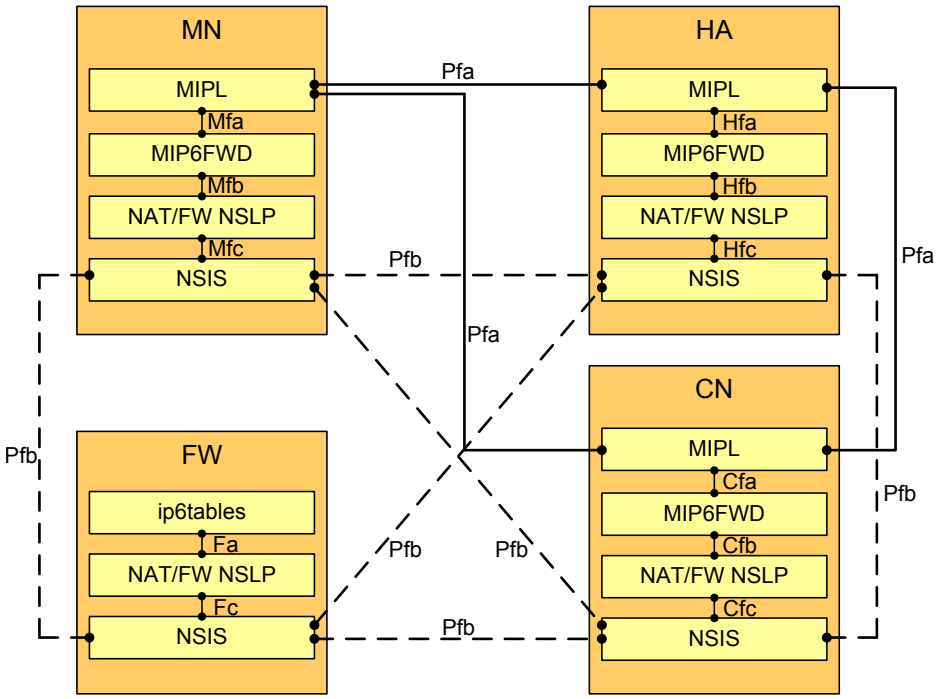


Figure 7.1: NSIS Based Mobile IPv6 Firewall Traversal Architecture

The yellow rectangles represent the software modules, namely *MIPL*, *MIP6FWD*, *NAT/FW NSLP*, *NSIS* and *ip6tables*. Whereas the main interfaces are represented with black connectors. Both software modules and interfaces are described in the following subsections.

7.1.2 Software Modules and Interfaces

7.1.2.1 NSIS

The University of Göttingen has implemented the GIST protocol, which is the implementation of the NSIS NTLP protocol [SH08], in C++, using *Linux Kernel* 2.6. The *FreeNSIS* implementation is full con format to the GIST protocol and its API [SH08]. The code is publicly available in [FreeNSIS].
The implementation architecture is shown in Figure 7.2. It is currently based on a single process approach using a main event loop based on *XORP* [XORP], which is used to implement socket maintenance and callbacks as well as timer callbacks. This design has no additional overhead for maintaining and synchronising multiple threads, which results in a high throughput and a rather simple implementation.
Besides the event loop, a key component in the GIST implementation is state man-

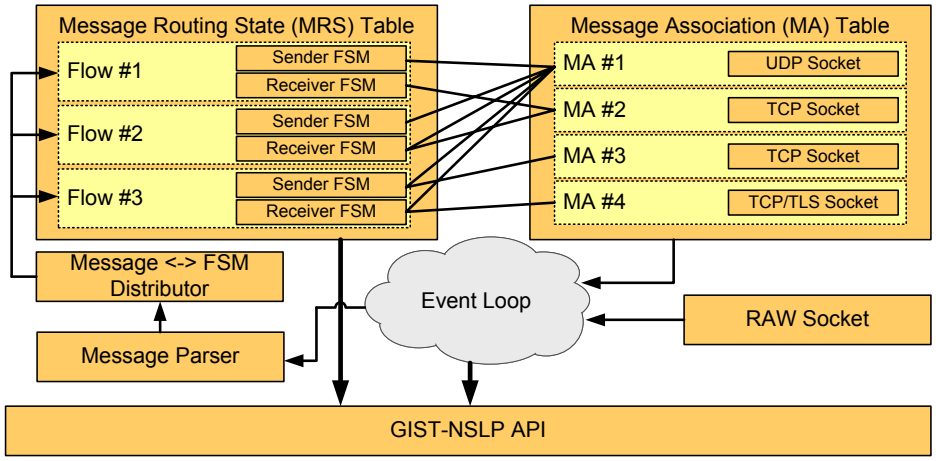


Figure 7.2: NSIS Implementation Architecture

agement. In order to support tens of thousands of signaling sessions efficiently, a hash table was used to manage the *Message Routing States* (MRSs), associated with linked lists to resolve conflicts. A standard lookup takes a constant time, however in the worst case, all table entries would be compared to find a given MRS.
To search the MRS table, one needs to know the associated key information, namely the session ID, the NSLP ID and *Message Routing Information* (MRI). Nevertheless, this subjects to some limitations, e.g., it is not possible to search for all MRSs using a specific MRI. Such a search feature may be useful to find MRSs that are affected by a detected link failure. A possible solution is to maintain specialised hash tables for link failures, which would permit quick searches. However, this approach would add maintenance overhead to every MRS table operation.
In addition to managing MRSs, a GIST implementation has to manage *Message Associations* (MAs) for *C-mode* operations. If two peers already have a MA and a new session is being established on the same path, the MA should be re-used to minimise resource usage. This feature implies that there should be a way to search the MA table for a MA that can be re-used for a certain session. The *FreeNSIS* implementation uses a second hash table to accomplish that goal. The upstream *Peer Information* (PI) serves as the key information. The UDP socket is treated as a *"virtual"* MA for the convenience of unifying the socket interface module.
Another important component of the *FreeNSIS* implementation is the *Finite State Machine* (FSM) to maintain states for each session. The GIST finite state machine [TTF+08] is implemented based on a combination of the *XORP* timer class and an FSM framework that was originally written for a *Linux* device driver.
Every MRS is associated with two FSMs, one for the upstream peer and another one for the downstream peer. There is no need for a global table of FSMs, because every

MRS provides pointers to the associated FSMs. In addition, every MA has a list of FSMs with which it is associated, so that the state machines can be informed, e.g., when a loss of connectivity with its current peer takes place.

7.1.2.2 NAT/FW NSLP

The NAT/FW NSLP [STAD08] daemon is implemented in user-space using C++. The code builds upon the *FreeNSIS* implementation developed by the University of Göttingen and introduced in Section 7.1.2.2. Both implementations are freely available in a single release [FreeNSIS]. The *FreeNSIS* implementation offers an API for NSLPs to use its generic transport services via *UNIX* sockets. The NAT/FW NSLP daemon itself also offers an API to upper layers to allow applications to trigger signaling flows, such as accepting inbound connections at an edge firewall. As depicted in Figure 7.3, the NAT/FW NSLP implementation consists of six main parts:

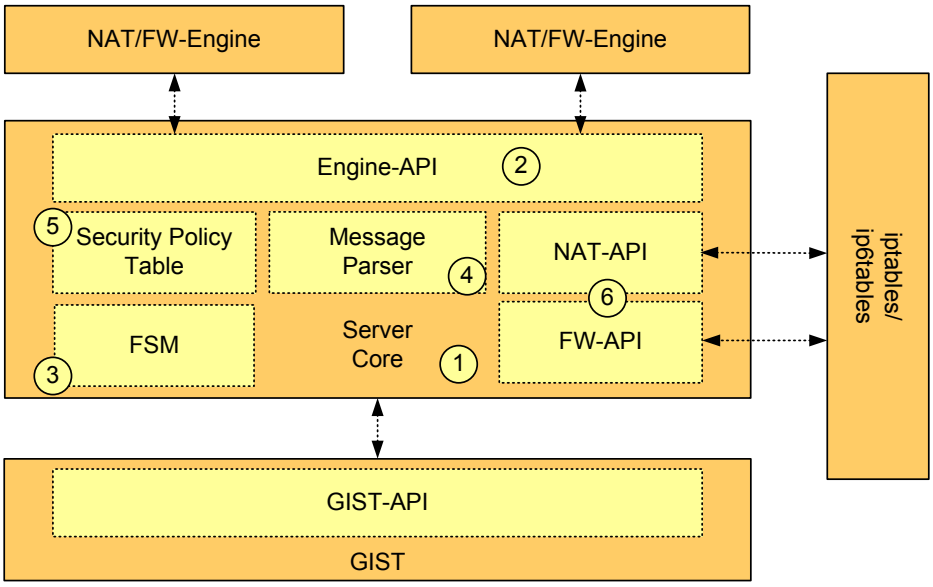


Figure 7.3: NAT/FW NSLP Implementation

1. Server core connecting to GIST-API and delegating callbacks to the other components,
2. NAT/FW engine-API,
3. Protocol behaviour defined in a finite state machine,
4. Message parsing and construction,

5. Security policy table and

6. APIs to firewall and NAT.

NAT/FW engines, (e.g., the *MIP6FWD*, described in Section 7.1.2.3) which enforce the NAT bindings and firewall policy rules for corresponding data traffic, are connected to the NAT/FW NSLP daemon via the NAT/FW engine-API.
The *Linux Kernel netfilter* [netfilter] module and its *iptables* front end was chosen as the NAT and firewall, because of its availability and complete coverage of required features. The use of the low-level iptables API *"libiptc"* is still discouraged by its developers because a lack of robustness and missing documentation. To avoid problems and incompatibility with different iptables versions, a *"system()"* call to invoke an iptables process with according parameters was chosen, although this approach is known to be inefficient. NAT/FW NSLP imposes only a small set of requirements on the used firewall and NAT, as NAT/FW NSLP supports only a small subset of functionality to any possible firewall or NAT implementation and thus replacing the currently used firewall and NAT can be done easily.
It was shown during the *FreeNSIS* development that having an efficient finite state machine in source code that represents similar sets of states, transitions and actions as in the state machine specification, simplifies the understanding of the code without sacrificing performance. A C++ template was written to allow re-usability among GIST and NSLP daemon development, enabling a mapping between the definition of a finite state machine, including states, transitions and actions to corresponding variables, function pointers and executable code.
The NAT/FW NSLP state machine [WSF+07] lists three possible initial states, a host being in an initiator, a forwarder or a receiver idle state. The decision whether a message has to be forwarded or delivered can not be made solely on the destination address in GIST MRI as in the NAT case; it is being rewritten, similarly as IP headers in NATs. Moreover, it depends on the current NAT configuration (or alternatively, often called reservation) status at the host where a message is received. The idea of the state machine giving a high-level overview for protocol understanding misses some aspects, such as locator rewrite and reservation dependency, that were fundamental aspects during implementation. Incoming message are processed by GIST and delivered to an NSLP if the NSLP is supported on that node. The NSLP decides whether to accept the message or to forward it. In the NAT/FW NSLP there are messages that are meaningful either just for NAT or just for firewall. The daemon configuration allows for the setting of flags to indicate whether a NAT/FW NSLP host is running a firewall, a NAT or both. After a message is accepted, basic validity checks are performed and the daemon will try to associate an existing state with the incoming message based on the session ID carried in NSLP payload. If there is no state installed yet, a new state machine object with a new session ID needs to be

created. As mentioned above, it must be distinguished whether the host is the NR of the signaling path or whether it is a NF. Depending on the initial state, the protocol behaviour for this session is different. In contrast, if the action is triggered via the API the host is always the NI and a new state machine object with a new session ID is created. Now, as the state machine object is created and the initial state is determined, the transition is applied on it. Transitions are modelled as a set of states, events and function pointers on the state machine object. According to a given state and an incoming event, the corresponding function in the state machine is called. This keeps the function call overhead very small. The function bodies contain all relevant code that defines the protocol behaviour, such as state manipulation, NAT and firewall interaction, message parsing and construction.
Fa is the interface between the NAT/FW NSLP on the firewall and ip6tables, as depicted in Figure 7.1. This interface translates the parameters of the firewall pinhole used inside the NAT/FW NSLP to a format adoptable by ip6tables. Furthermore, its builds a command line string which can install or delete the firewall pinhole into/from the local firewall implementation and triggers this by using a *"system()"* system call. **Fc** is the interface between the NAT/FW NSLP and NSIS and similar to **Mfc**, which is described in detail in Section 7.1.2.5.3.

7.1.2.3 MIP6FWD

The *MIP6FW daemon* (MIP6FWD) module is a simple daemon used to translate and forward triggers from the MIPL module arrival via the interfaces **Mfa**, **Hfa** or **Cfa** to the NAT/FW NSLP module via the interface **Mfb**, **Hfb** or **Cfb**. The NAT/FW NSLP uses the interface **Mfc**, **Hfc** or **Cfc** to communicate with the NSIS NTLP. It also waits for the response of the requests and forwards it back via the arrival interface.

7.1.2.4 ip6tables

The ip6tables module implements the firewall functionality. The *netfilter/iptables* [netfilter] release 1.3.8 (or above) can be used. The Mobile IPv6 firewall traversal implementation requires the following patches/extensions to be installed on all firewalls:

- **mh**: this module adds Mobility Header support for IPv6.
- **rt**: this module adds IPv6 Routing Header support.
- **dst**: this module allows to match the parameters in IPv6 Destination Options Header.

7.1.2.5 MIPL (MN)

The MIPL module implements the mobile node functionality. The release used is *MIPL 2.0.2* [MIPL] developed by GO-Core in co-operation with the *USAGI/WIDE Project* [USAGI]. The NSIS based Mobile IPv6 firewall traversal requires some extensions to the MIPL code:

- When the MIPL implementation on the MN gets a new CoA and wants to send a Binding Update to the HA, the MIPL implementation is halted and triggers the MIP6FWD to trigger a re-configuration via the interface **Mfa**. The MIP6FWD now triggers the NAT/FW NSLP via interface **Mfb**. When the NSIS auto reconfiguration, as described in Section 5.3.3, is finished successfully and the MIPL receives a corresponding response, it triggers the MIP6FWD to install firewall pinholes for the Binding Update via interface **Mfa**, which again uses **Mfb** to trigger the NAT/FW NSLP. If the MIPL knows about the successful pinhole creation to the HA, it is resumed and sends out the Binding Update to the HA as it now can traverse the firewalls.

- If the MIPL implementation wants to send data traffic to a CN, either directly to the CN or via the HA, it is halted again and triggers the MIP6FWD to install the required firewall pinholes via the interface **Mfa**, which forwards this trigger to the NAT/FW NSLP using the interface **Mfb**. If the MIPL knows about the successful pinhole creation, it is resumed and sends out the data traffic as it now can traverse the firewalls.

- The MIPL implementation on the MN communicates with the MIPL implementation on the HA and the CN using the interface **Pfa**, which implements the Mobile IPv6 protocol [JPA04].

7.1.2.5.1 Mfa (MIPL – MIP6FWD)

Mfa is an inter-process interface implemented using a TCP socket. This interface is used to trigger the MIP6FWD in order to trigger the firewall pinholes between the nodes when they are required. Therefore, a trigger message (e.g., of type CREATE) is used, which signals the MIP6FWD the format of the required firewall pinhole. The format for each type of firewall pinhole is described in detail in A.1. After triggering the MIP6FWD, the MIPL implementation is halted and waits for a response message from the MIP6FWD. With the response, the MIP6FWD informs the MIPL implementation about the success of the firewall pinhole creation.
The create trigger message format, exchanged between MIPL and MIP6FWD, is depicted in Figure 7.4, the message fields are defined as follows:

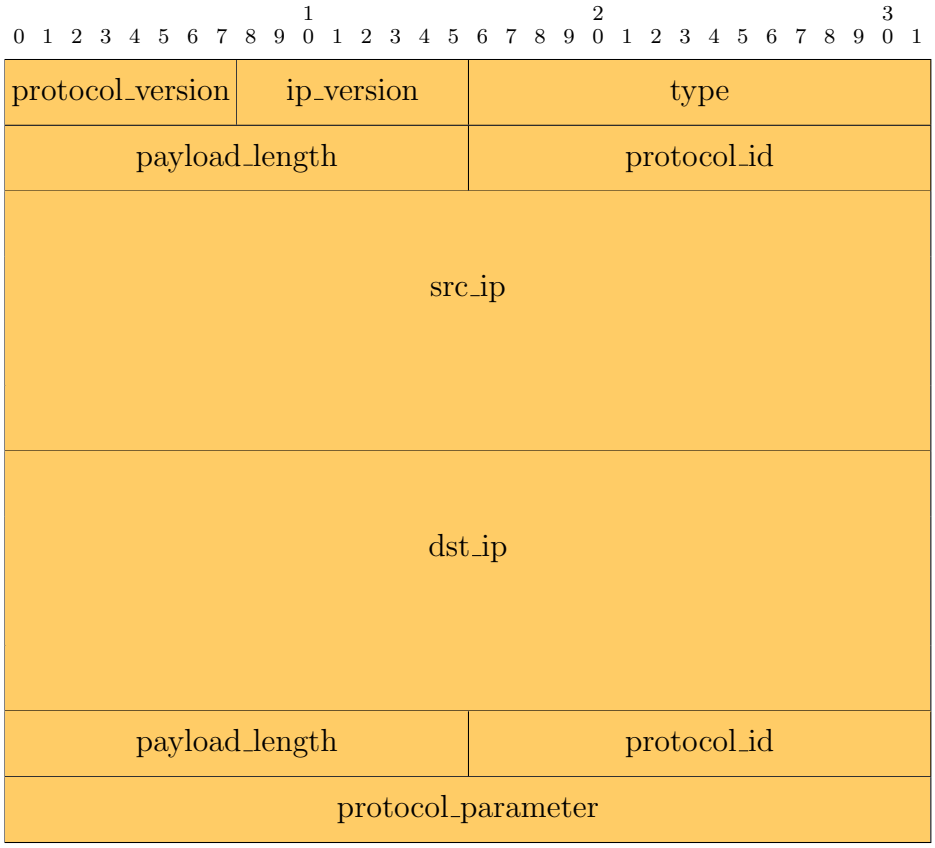


Figure 7.4: MIPL-MIP6FWD Create Message

- **protocol_version**: specifies the version of the interface between MIPL and MIP6FWD. Current version is "1".

- **ip_version**: defines whether the IP version of the request is IPv4 or IPv6. In Mobile IPv6 firewall traversal case, it is always set to IPv6.

- **type**: defines the type of the packet. Possible types are:

  – MIPTRIGGER_MSG_ERROR (0x00),
  to inform the MIPL implementation about an occurred error.
  – MIPTRIGGER_MSG_ACK (0x01),
  to inform the MIPL about the success of a firewall pinhole request.
  – MIPTRIGGER_MSG_CREATE (0x02),
  to trigger MIP6FWD for firewall pinhole creation using a NAT/FW NSLP CREATE message.

  - MIPTRIGGER_MSG_STATUS (0x03),
    to exchange the status of the MIPL and the MIP6FWD implementations.
  - MIPTRIGGER_MSG_NSIS_AUTOCONF (0x04),
    when the MIP6FWD receives this message, it triggers the NSIS NAT/FW NSLP implementation to re-configure its IP addresses. This is required after a handover; otherwise NSIS would not know of its new addresses.
  - MIPTRIGGER_MSG_EXT (0x05),
    to trigger MIP6FWD for firewall pinhole creation using a NAT/FW NSLP EXT message.

- **payload_length**: defines the length of the payload.
- The following fields specify the format of the requested firewall pinhole:
  - **protocol_id**: protocol-id of the firewall pinhole,
  - **src_ip**: source IP address of the firewall pinhole,
  - **dest_ip**: destination IP address of the firewall pinhole,
  - **src_port**: source port of the firewall pinhole,
  - **dest_port**: destination port of the firewall pinhole,
  - **protocol_parameter**: for special types for this protocol, e.g., the type of a mobility header.

Unlike the create and ext message, the other message types, namely the error, response, status and autoconf messages have the same message format. These messages are also used for exchange between MIPL and MIP6FWD and have the format as shown in Figure 7.5.

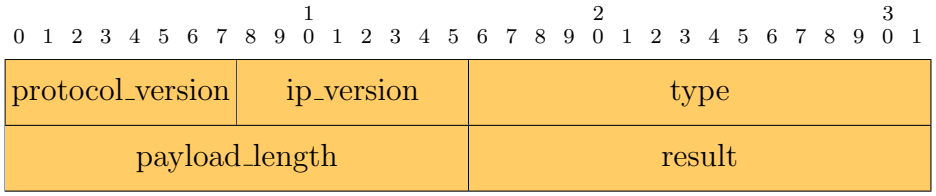


Figure 7.5: MIPL-MIP6FWD Response Message

The format of the **protocol_version, ip_version, type and payload_length** fields have the same specification as for the create message.

- **result**: Informs the MIPL specification about the result of a firewall pinhole creation request. Possible values are:
  - MIPTRIGGER_RESULT_UNKNOWN (0x00),

- MIPTRIGGER_RESULT_ACK (0x01),
- MIPTRIGGER_RESULT_NACK (0x02),
- MIPTRIGGER_RESULT_NACK_FW (0x03),
- MIPTRIGGER_RESULT_PROTOCOL_ERROR (0x05),
- MIPTRIGGER_RESULT_QUEUE_BUSY (0x10),
- MIPTRIGGER_RESULT_QUEUE_EMPTY (0x11).

The rules that the MIPL and the MIP6FWD modules have to observe are:

- As soon as MIPL implementation knows the new care-of address, it sends a MIPTRIGGER_MSG_NSIS_AUTOCONF to the MIP6FWD.
- When the MIP6FWD receives a MIPTRIGGER_MSG_NSIS_AUTOCONF messages, it triggers the NSIS NAT/FW NSLP to re-configure its IP addresses and informs the MIPL implementation about the success (compare Section 7.1.2.5.2).
- After receiving a successful MIPTRIGGER_MSG_NSIS_AUTOCONF response, the MIPL implementation begins to start the firewall pinhole creation progress for the first firewall pinhole (e.g., the firewall pinhole for the BU or IKEv2 signaling messages between the MN and the HA). Therefore, it triggers the MIP6FWD with a MIPTRIGGER_MSG_CREATE message to install this firewall pinhole and waits for the response.
- When the MIP6FWD receives an MIPTRIGGER_MSG_CREATE message, it computes this request and triggers the NAT/FW NSLP to install the firewall pinhole as it is requested. It later informs the MIPL implementation about the success of the firewall pinhole creation request.
- If the MIPL implementation receives a successful MIPTRIGGER_MSG_ACK to a MIPTRIGGER_MSG_CREATE message, it resumes the normal MIPL implementation. This could be to trigger the creation of further firewall pinholes, to trigger NSIS for a new re-configuration or to resume the normal MIPL process.

The interaction between MIPL and MIP6FWD on the MN is depicted in the flow diagram in Figure 7.6.

7.1.2.5.2 Mfb (MIP6FWD – NAT/FW NSLP)

Mfb is an inter-process interface implemented using a TCP socket. This interface is used to trigger the NAT/FW NSLP to trigger the pinhole creation progress upon the event of receiving a message at the MIP6FWD. Therefore, a trigger message is used, which signals to the NAT/FW NSLP implementation the type of the requested

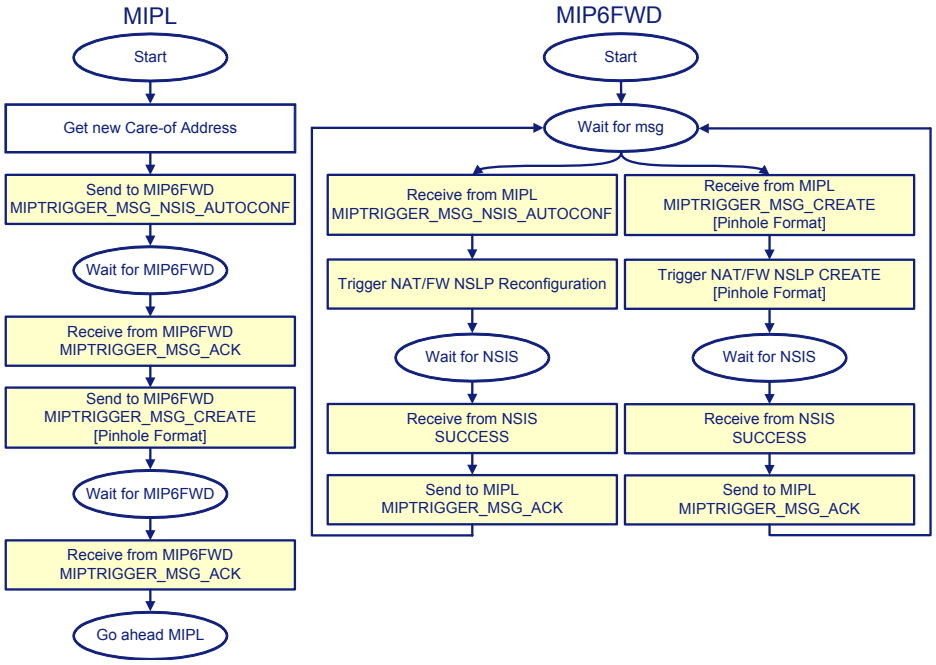


Figure 7.6: Interaction between MIPL and MIP6FWD (MN)

firewall pinhole. After triggering the NAT/FW NSLP, the MIP6FWD waits for a response from the NAT/FW NSLP implementation. With the response the NAT/FW NSLP implementation informs the MIP6FWD about the success of the firewall pinhole creation.
The trigger message exchanged between the MIP6FWD and the NAT/FW NSLP has the format as depicted in Figure 7.7. It should be notice, that this message format is used for all kind of signaling between an application and the NAT/FW NSLP. Therefore, it also consist of fields which are not necessary for the Mobile IPv6 firewall traversal case.

- **type**: defines the type of the packet. Possible values are:
  - NATFW_TYPE_EXT (0x01),
    to signal for a EXT message.
  - NATFW_TYPE_CREATE (0x02),
    to signal for a CREATE message.
  - NATFW_TYPE_NOTIFY (0x03),
    to signal for a NOTIFY message
  - NATFW_REREADROUTINGTABLE (0x04),
    when the NAT/FW NSLP receives this kind of messages, it triggers the

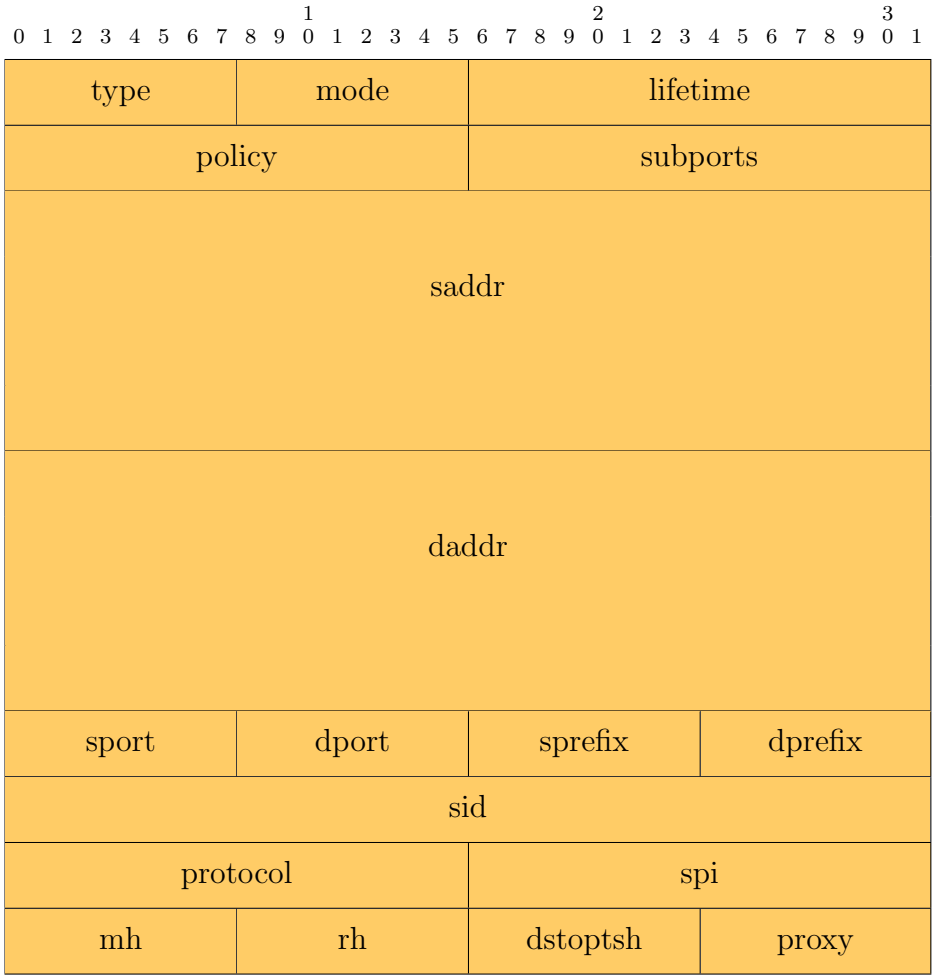


Figure 7.7: MIP6FWD-NATFW Trigger Message

NSIS implementation to re-configure its IP addresses. This is required after a handover; otherwise NSIS would not know about its new addresses.

- **mode**: specifies the mode of the message. Possible values are NATFW_MODE_CMODE (0x01) and NATFW_MODE_DMODE (0x02).
- The following fields specify the format of the requested firewall pinhole:
  - **lifetime**: specifies the lifetime of a firewall pinhole in seconds. If the firewall pinhole is not refreshed before the lifetime has expired, it will be automatically removed.
  - **policy**: defines the policy of the firewall pinhole. Possible values are ALLOW (0x01) and DENY (0x02).

  - **subports**: defines a range of ports (if the firewall pinhole requires them).
  - **saddr, sport, sprefix**: source address, source port and source prefix.
  - **daddr, dport, dprefix**: destination address, destination port and destination prefix.
  - **protocol**: defines the protocol the firewall pinhole matches to.
  - **spi**: specifies a SPI (if the firewall pinhole requires).
  - **mh, rh, dstoptsh**: to signal for a special type of Mobility Header, Routing Header or Destination Options Header.

- **sid**: The session ID of this request. Used to identify the request, e.g., in the RESPONSE message.

- **proxy**: flag to trigger the NAT/FW NSLP to use the proxy mode (not used in the current NSIS based Mobile IPv6 firewall traversal implementation).

The response message format, sent from the NAT/FW NSLP to the MIP6FWD to inform it about the result of a request is shown in Figure 7.8.

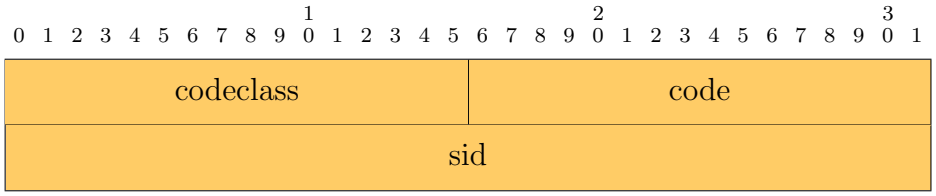


Figure 7.8: MIP6FWD-NATFW Response Message

- **codeclass**: specifies the codeclass of the response message.

- **code**: The result of the firewall pinhole creation request. Possible values are SUCCESS (0x01) or FAILURE (0x02).

- **sid**: The session ID of the firewall pinhole creation request. Used to identify the primarily request.

7.1.2.5.3 Mfc (NAT/FW NSLP – NSIS)

The interface **Mfc** implements the NAT/FW NSLP protocol [STAD08]. It communicates with other NSIS and NAT/FW NSLP implementations on other nodes via the interface **Pfb**. The interface **Pfb** implements the NSIS protocol [SH08]. For more details on the NAT/FW NSLP see Section 4.9.

7.1.2.6 MIPL (HA)

This MIPL module implements the home agent functionality. The same release used for the MN (MIPL 2.0.2) has been used as a starting point for the development on the HA. However, the NSIS based Mobile IPv6 firewall traversal has required some extensions to the MIPL source code:

- When the MIPL implementation on the HA receives a specific message (e.g., BA, HoTI or bi-directional data traffic) it triggers the MIP6FWD via the interface **Hfa** to install the corresponding firewall pinhole and waits for the response. The MIP6FWD itself again triggers the NAT/FW NSLP to trigger this firewall pinhole via the interface **Hfb**. When the MIPL implementation receives a successful response it resumes the normal MIPL progress.

- The MIPL implementation on the HA communicates with the MIPL implementation on the MN and the CN using the interface **Pfa**, which implements the Mobile IPv6 protocol [JPA04].

7.1.2.6.1 Hfa (MIPL – MIP6FWD)

Hfa is an inter-process interface implemented using a TCP socket. This interface is very similar to the interface **Mfa**, but reacts on different events. In contrast to the **Mfa** interface, which reacts on a new CoA, the **Hfa** interface interacts on the event of receiving a BU message, a HoTI message or on receiving bi-directional data traffic from the MN.
If this happens, it triggers the MIP6FWD to open firewall pinholes between the nodes, e.g., for the HoTI message between the HA and the CN. Therefore, a trigger message (e.g., of type CREATE) is used, which signals to the MIP6FWD, the format of the required firewall pinhole. After triggering the MIP6FWD, the MIPL implementation is halted and waits for a response message from the MIP6FWD. With the response the MIP6FWD informs the MIPL implementation about the success of the firewall pinhole creation. These messages are similar to the messages as explained in Section 7.1.2.5.1. The rules that the MIPL and the MIP6FWD modules must observe include:

- As soon as the MIPL implementation receives a specific message (e.g., BA, HoTI or bi-directional data traffic), it sends a MIPTRIGGER_MSG_NSIS_CREATE message to the MIP6FWD to install the corresponding pinhole and waits for the response. For the example of receiving a HoTI, it triggers a firewall pinhole for the HoTI between HA and CN, as specified in Section 5.3.

- When the MIP6FWD receives an MIPTRIGGER_MSG_CREATE message, it computes this request and triggers the NAT/FW NSLP to install the firewall

pinhole as it is requested. It later informs the MIPL implementation about the success of the firewall pinhole creation request.

- If the MIPL implementation receives a successful MIPTRIGGER_MSG_ACK to a MIPTRIGGER_MSG_CREATE message, it resumes the normal MIPL implementation. This could be to trigger the creation of further firewall pinholes, to trigger NSIS for a new re-configuration or the normal MIPL process. In our example, this means to send the HoTI to the CN, as it can now traverse the firewalls.

The interaction between MIPL and MIP6FWD on the HA is depicted in the flow diagram in Figure 7.9.

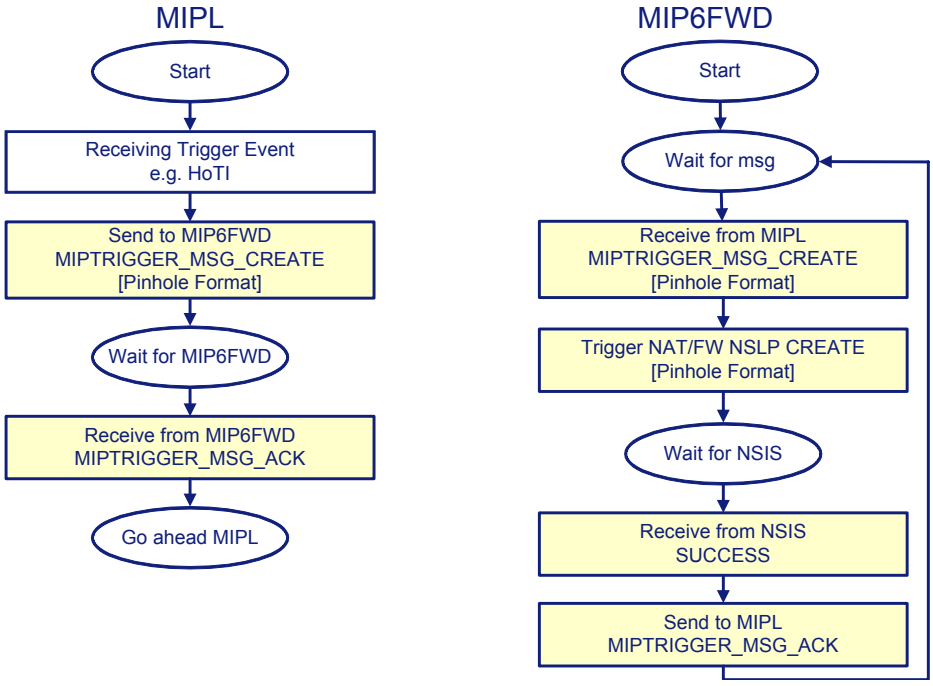


Figure 7.9: Interaction between MIPL and MIP6FWD (HA)

7.1.2.6.2 Hfb (MIP6FWD – NAT/FW NSLP)

Hfb represents the same interface as **Mfb**, described in Section 7.1.2.5.2.

7.1.2.6.3 Hfc (NAT/FW NSLP – NSIS)

Hfc is the same interface as **Mfc**, which is described in detail in Section 7.1.2.5.3.

7.1.2.7 MIPL (CN)

This MIPL module implements the corresponding node functionality. The same release used for the MN (MIPL 2.0.2) has been used as a starting point for the development on the CN. However, the NSIS based Mobile IPv6 firewall traversal has required some extensions to the MIPL source code:

- When the MIPL implementation on the CN receives a specific message (e.g., HoTI, CoTI or data traffic), it triggers the MIP6FWD via the interface **Cfa** to install the corresponding firewall pinhole and waits for the response. The MIP6FWD itself again triggers the NAT/FW NSLP to trigger this firewall pinhole via the interface **Cfb**. When the MIPL implementation receives a successful response it resumes the normal MIPL progress.

7.1.2.7.1 Cfa (MIPL – MIP6FWD)

Cfa is an inter-process interface implemented using a TCP socket. This interface is very similar to the interface **Mfa**, but reacts on different events. In contrast to the **Mfa** interface, which reacts on a new CoA, the **Cfa** interface interacts on the event of receiving a HoTI message, a CoTI message or on receiving/sending bi-directional/route optimised data traffic from/to the MN.
If this happens, it triggers the MIP6FWD in order to trigger a firewall pinhole between the nodes, e.g., for the HoT message between the CN and the HA. Therefore, a trigger message (e.g., of type CREATE) is used, which signals the MIP6FWD the format of the required firewall pinhole. After triggering the MIP6FWD, the MIPL implementation is halted and waits for a response message from the MIP6FWD. With the response, the MIP6FWD informs the MIPL implementation about the success of the firewall pinhole creation. The messages are similar to the messages explained in Section 7.1.2.5.1. The rules that the MIPL and the MIP6FWD modules have to observe are:

- As soon as the MIPL implementation receives a specific message (e.g., HoTI, CoTI or data traffic), it sends a MIPTRIGGER_MSG_NSIS_CREATE message to the MIP6FWD to install the corresponding firewall pinhole and waits for the response. For the example of receiving a HoTI, it triggers a firewall pinhole for the upcoming HoT between CN and HA, as specified in Section 5.3.

- When the MIP6FWD receives a MIPTRIGGER_MSG_CREATE message, it computes this request and triggers the NAT/FW NSLP to install the firewall pinhole as it is requested. It later informs the MIPL implementation about the success of the firewall pinhole creation request.

- If the MIPL implementation receives a successful MIPTRIGGER_MSG_ACK to a MIPTRIGGER_MSG_CREATE message, it resumes the normal MIPL implementation. This could be the installation of further firewall pinholes, or the normal MIPL process. In the example, this means to send the HoT to the HA, as it now can traverse the firewalls.

The interaction between MIPL and MIP6FWD on the CN is depicted in the flow diagram in Figure 7.10.

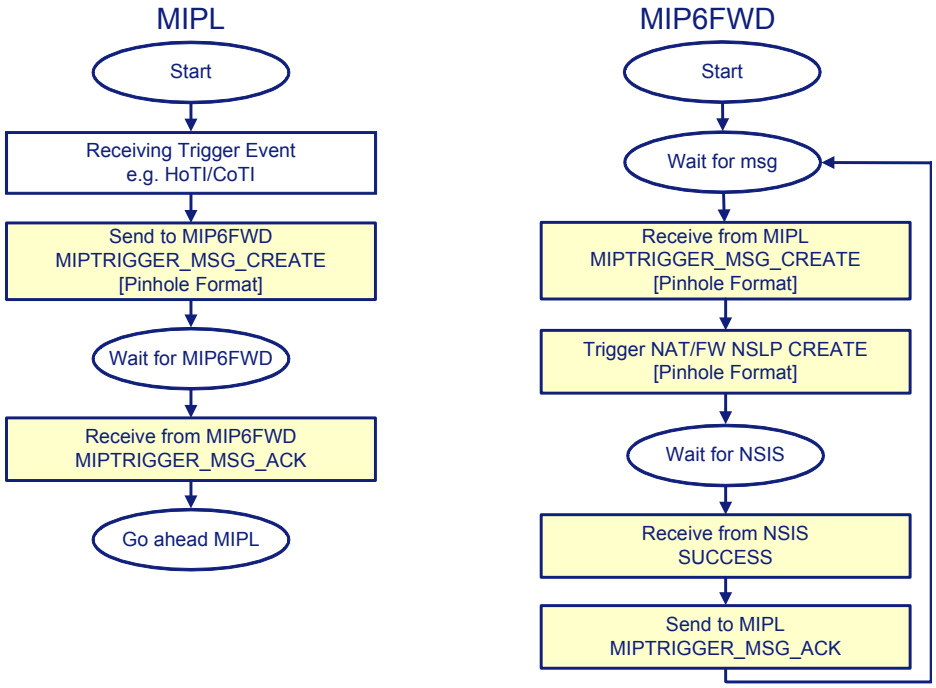


Figure 7.10: Interaction between MIPL and MIP6FWD (CN)

7.1.2.7.2 Cfb (MIP6FWD – NAT/FW NSLP)

Cfb represents the same interface as **Mfb**, described in Section 7.1.2.5.2.

7.1.2.7.3 Cfc (NAT/FW NSLP – NSIS)

Cfc is the same interface as **Mfc**, which is described in detail in Section 7.1.2.5.3.

7.2 Mobile IPv6 Application Layer Gateway

The *Mobile IPv6 Application Layer Gateway* proof-of-concept implementation is still in an unstable condition. The prototype implementation is realised as a *netfilter/iptables* [netfilter] module and is available at [MIP6ALG].
The prototype implementation currently only supports the basic primitives as described in Section 5.4.3.1, more precisely, the primitives described in Section 5.4.3.1.1, Section 5.4.3.1.2 and Section 5.4.3.1.3. However, the possibility to implement these requirements already proves that the *Mobile IPv6 Application Layer Gateway* is technically feasible and implementable.

7.3 Summary

This chapter has presented the *NSIS based Mobile IPv6 firewall traversal* and the *Mobile IPv6 Application Layer Gateway* firewall traversal proof-of-concept implementations. The approaches have been described in detail in Section 5.3 and Section 5.4. The described proof-of-concept implementations developed as part of this thesis have proved that firewall traversal in Mobile IPv6 environments is technically feasible and implementable for both the *NSIS based Mobile IPv6 firewall traversal* and the *Mobile IPv6 Application Layer Gateway* firewall traversal.
Chapter 8 evaluates the proposed Mobile IPv6 firewall traversal solutions as well as the proof-of-concept implementations, described in this chapter.

8 Evaluation

This chapter evaluates the Mobile IPv6 firewall traversal solutions proposed in Chapter 5 as well as their in Chapter 7 described proof-of-concept implementations. This has to be done either with the help of a performance testing and/or with the help of a mathematical model. Section 8.1 evaluates the *NSIS based Mobile IPv6 firewall traversal* approach and implementation while Section 8.2 evaluates the *Mobile IPv6 Application Layer Gateway* firewall traversal approach.

8.1 NSIS Based Mobile IPv6 Firewall Traversal

This section evaluates the *NSIS based Mobile IPv6 firewall traversal* approach, as introduced in Section 5.3 and its implementation described in Section 7.1. The *NSIS based Mobile IPv6 firewall traversal* bases on the *Next Steps in Signaling* (NSIS) [HKLdB05] *NSIS Transport Layer Protocol* (NTLP) [SH08] and the *NAT/Firewall NSIS Signaling Layer Protocol* (NSLP) [STAD08]. Due to this modular design it is required to study the performance of the NTLP and NSLP layers individually before being able to draw conclusions from their impact on the *NSIS based Mobile IPv6 firewall traversal* approach.
Therefore, this section firstly examines the performance of the NSIS NTLP and the utilised *FreeNSIS* [FreeNSIS] implementation in Section 8.1.1. Secondly, it presents a performance testing for the NAT/FW NSLP implementation in Section 8.1.2. Finally, the *NSIS based Mobile IPv6 firewall traversal* approach is analysed with the help of a mathematical model in Section 8.1.3.

8.1.1 NSIS Performance Testing

The performance of the utilised *FreeNSIS* [FreeNSIS] NSIS NTLP implementation of the University of Göttingen has already been studied several times. Results of the overhead and performance study are presented in for instance [DFT+06].
[DFT+06] has shown that the modularity design of the GIST implementation provides a flexible way for state management, message processing, and any type of application-specific signaling purposes. It confirms that the result of the modular design is improved extensibility, security, and transport properties at the cost of additional overhead. The *FreeNSIS* implementation performed efficiently when serving a

large number of sessions up to 60,000 with acceptable processing and round-trip times. Therefore, NSIS itself, more precisely *FreeNSIS*, is fully adaptive for an NSIS based Mobile IPv6 firewall traversal approach. However, as the performance also relies on the NAT/FW NSLP implementation, the next section examines its performance in detail.

8.1.2 NAT/FW NSLP Performance Testing

An implementation of the NAT/FW NSLP offers various possibilities for performance testing and evaluation. A large number of different tests could be run in order to examine, for example, the scalability or the message parsing of the implementation as well as the iptables performance. Due to the scope of this thesis – Mobile IPv6 firewall traversal – only the performance of the firewall related implementation is evaluated. It was intended that the main part of this performance testing focused on the evaluation of the NAT/FW NSLP implementation. In the course of the tests it became obvious that not the NAT/FW NSLP implementation itself was the bottleneck with regard to performance, but rather the firewall implementation (i.e., management of the policy rule set). Therefore, this evaluation deals also with performance testing of the *netfilter/iptables* [netfilter] implementation; however, only were it is connected to the NAT/FW NSLP framework.

8.1.2.1 Testing Methodology and Testbed Setup

Before presenting results of the tests, an overview on the testing methodology and the testbed will be given. This performance testing represented a large challenge due to the fact that currently no evaluation of such a middlebox traversal implementation is available. This demanded all performance testing methodology to be self designed and acquired. Therefore, the results presented in this evaluation can not be directly compared to results of other evaluation approaches.

8.1.2.1.1 Testbed Setup

The testing experiments were run on standard low-end PCs with a *Linux Kernel* 2.6.12. They were equipped with the following hardware:

- Via Eden CPU 533 MHz
- 3 Realtek 100Mbps NICs
- 256 MB SDRAM PC 133
- 20 GB HDD

Figure 8.1 shows how the nodes were connected for the experiments. The NSLP forwarders both run a firewall. Most experiments were performed with the topology shown in Figure 8.1, but some experiments were performed with only one NSLP forwarder, namely the session setup time and the round trip time testing.

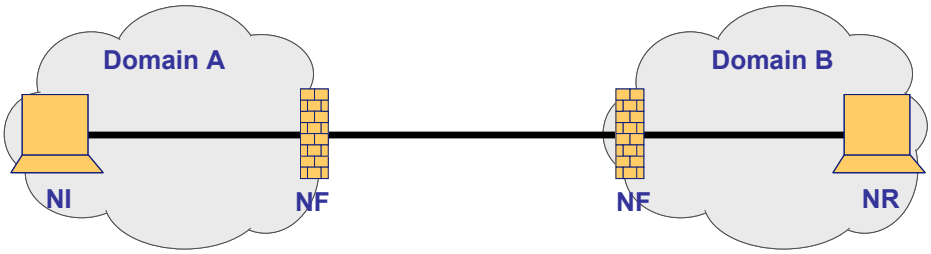


Figure 8.1: Performance Testing Testbed Setup

8.1.2.1.2 Measurement Methods and Tools

The following methods and tools were used for measurement and testing. To calculate the session setup time and round trip time (Section 8.1.2.2.1 and Section 8.1.2.2.3) timestamps have been inserted directly into the source code. A simple script calculates the time between the two timestamps corresponding to one session.
A similar approach was used to calculate the processing time for the implementation (Section 8.1.2.2.2). Also, the processing time for the NAT/FW NSLP protocol behaviour and for iptables could be examined individually. A *PHP* script similar to *Linux* process monitoring tool *"top"* [Top] was used to monitor CPU and memory consumption (Section 8.1.2.2.4). *Gnuplot* [Gnup] was used to generate plots and summaries from the before extracted reports.

8.1.2.2 Testing Results

This section presents the results of the different performance experiments. Aim of these experiments was to test how the scalability of the implementation behaves in the presence of increasing numbers of sessions on core routers. Thereby, limits and bottlenecks of the NAT/FW NSLP and iptables implementations could be detected. The tests were performed using a lightweight *"NatFwClient"* test-application. The application was run on the NSLP initiator and creates a new NAT/FW NSLP session every $\frac{3}{4}$ second. Each session had the same NSLP responder but different source and destination ports. Thereby it could assure that the NSLP forwarders had to create a new firewall pinhole for each session. All tests started with zero sessions and

generated up to 15,000 sessions between NI and NR without deleting or teardown existing sessions during the tests. The maximum of 15,000 sessions was causes by the testbed machines and the iptables implementations; this will be described later with help of the testing results in more detail. GIST was configured to use a refresh rate of 180 seconds and the NAT/FW NSLP is configured to a refresh rate of 90 seconds. Taking account of possible measurement inaccuracy and errors due to the experimental environments, all tests were repeated several times. The results in following subsections represent the average of those tests.

8.1.2.2.1 Session Setup Time

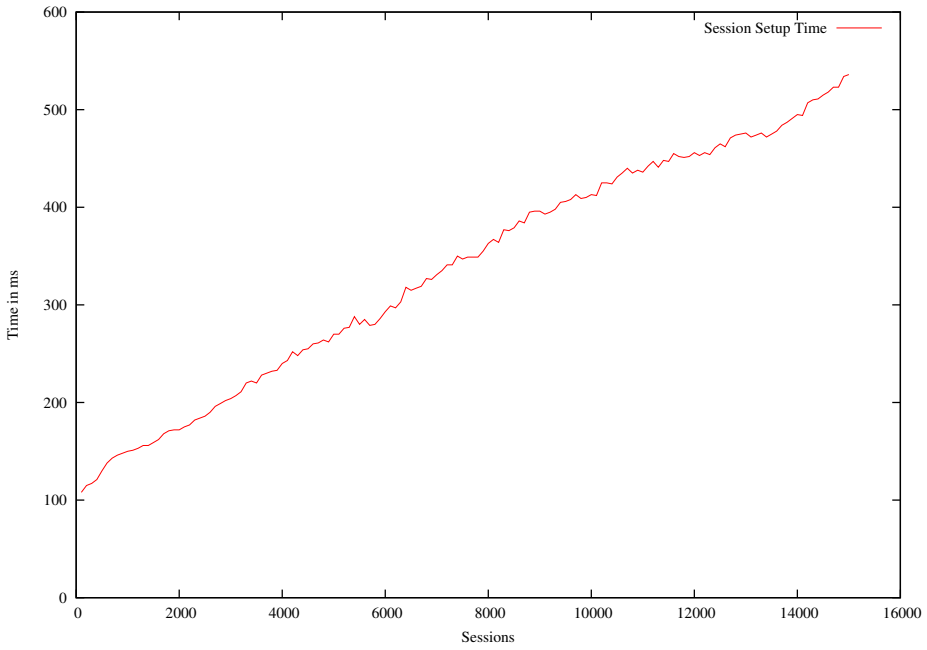


Figure 8.2: Session Setup Time

The first performance test examined the session setup time of the implementation, meaning how long it takes to establish a NAT/FW NSLP session between NI and NR. The experiment has been performed using only three nodes of the testbed, the NSLP initiator, the NSLP responder and one NSLP forwarder. This is due to the fact that two NF would increase the session setup time and would make it more difficult to get results of the scalability of the implementation. Figure 8.2 shows the session setup time under different number of sessions. The first observation is that the increase of the session setup time is nearly linear. It can be noted that the implementation does not have low session setup time with only hundreds of sessions.

When serving for approximately 15,000 sessions, the NAT/FW NSLP implementation seemed overloaded with a setup time of about 520 *ms*. However, iptables is the much more crucial factor, which will be examined in the next experiment.

8.1.2.2.2 Processing Time

As shown above, results of the session setup time tests show the serving session number is limited by a certain number. This motivates to examine more closely the processing time distribution of the individual implementation components and to determine the performance bottleneck. Therefore, timestamps were inserted in the source code and the same test was executed again. Several timestamps assure that processing time for the NAT/FW NSLP implementation and the processing time of iptables can be observed individually. The iptables processing time can only be measured on a node using iptables. Hence, measurement was run on a NSLP forwarder and thus the NAT/FW NSLP processing time represents only the values for a NF.
Figure 8.3 shows the processing time for only the NAT/FW NSLP implementation, and the overall processing time together within the iptables costs. As can be seen immediately, the processing time of the NAT/FW NSLP implementation is almost equal to 40-50 *ms*, even with increasing number of sessions. In contrast, the processing time for adding a rule into the iptables rule set may be acceptable for up to 6,000 sessions; with more than 6000 sessions it increases rapidly and amounts to nearly 400 *ms* with 15,000 sessions. This shows that the iptables system-call to open a firewall pinhole is the bottleneck in the experiments.
These high values may be caused by the used machines in our testbed, which are, due to their hardware, not comparable with a core router machine. Much crucial for the bad iptables performance, however, is the way iptables implements the rule set; namely as a linked list. iptables uses a simple packet classification algorithm which traverses the rules in a chain linearly per packet until a matching rule is found or not. However, this approach lacks efficiency [KG04, HPS03]. So, the linear packet filtering approach is not feasible if many rules are inside the chains. The cost for inserting rules increases rapidly with increasing numbers of rules in the rule set and the processing time per packet also increases. Another critical aspect is that iptables stalls the packet processing during chain updates, e.g., when inserting a rule.
[KG04] and [HPS03] have investigated this behaviour of the iptables implementation. [KG04] has demonstrated that a high-end router needs more than 400 seconds to add 16,384 rules into the rule set and that the cost to add a rule increases the more rules already exists. Several solutions were proposed over the time to solve this problem: *"ipset"* [IPset] is not a complete replacement or extension of iptables itself, but a new matching fully integrated into iptables. ipset supports matching of IP addresses, netblocks or port numbers stored in bitmaps, hashes or trees. *"nf-HiPAC"* [HiPAC] is a full featured packet filter for *Linux* and uses a binary tree to store and evaluate the

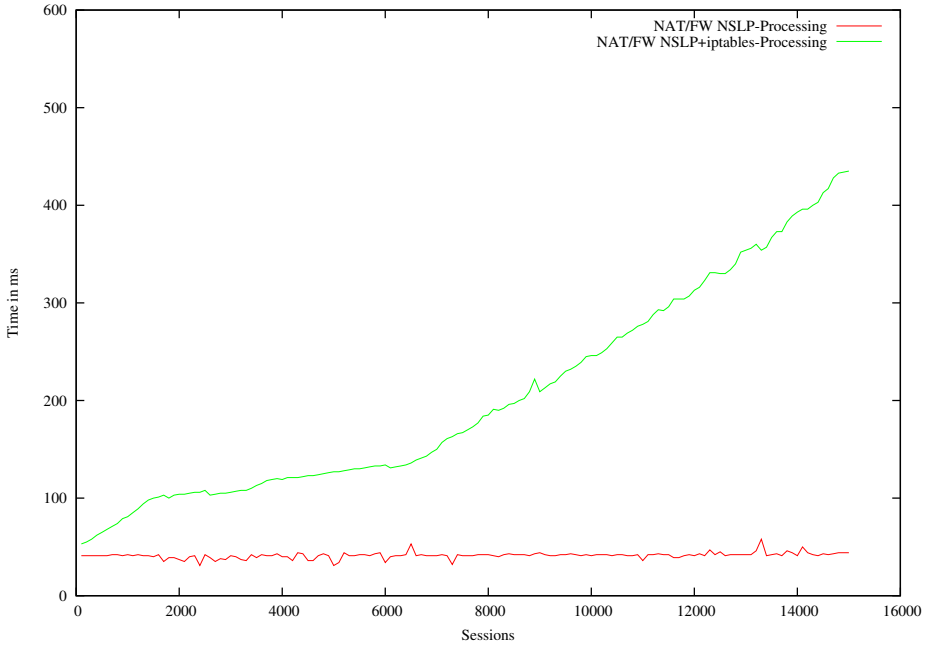


Figure 8.3: Processing Time

rule. The user interface closely follows the syntax of iptables and besides the native nf-HiPAC matches, it can use non-native matches from iptables. However, nf-HiPAC supports only the filter table and does not provide the same functionality for network address translation.

[KG04] and [HPS03] show that both implementations reduce the number of memory lookups per packet. They are applicable for environments with large rule sets or high bandwidth networks and outperform iptables independent of the number of rules. Thereby, they do not impose any overhead, even for very small rule sets. They also can update their rule sets dynamically without stalling the packet processing in contrast to iptables, which has a worse update performance and stalled packet processing during updates.

In order to be able to examine the performance of the implementation with large number of sessions, iptables would have to be replaced by a better scaling implementation. However, this performance study does not aim to be an evaluation of firewall implementations, rather it focuses on the firewall (and other middleboxes) traversal issue. Therefore, instead of using a different firewall implementation, the firewall was changed to allow-all policy and the calls to iptables are not performed. The experiment is performed to examine only the NAT/FW NSLP implementation without the overhead of a firewall implementation. This experiment approach is likely a good

approximation to some realistic scenarios, as [KG04] and [HPS03] show that the overhead of these firewall implementations can be neglected if the firewall is reasonably implemented. Figure 8.4 shows the results of this experiment.
As it can be seen, the NAT/FW NSLP implementation is now able to handle up 35,000 sessions, if no iptables implementation is present. This confirms the assumption that iptables represents the bottleneck in the experiments. The maximum of 35,000 sessions is caused by the low-end PC hardware of the testbed machines; the implementation may be able to handle more sessions in a different environment.

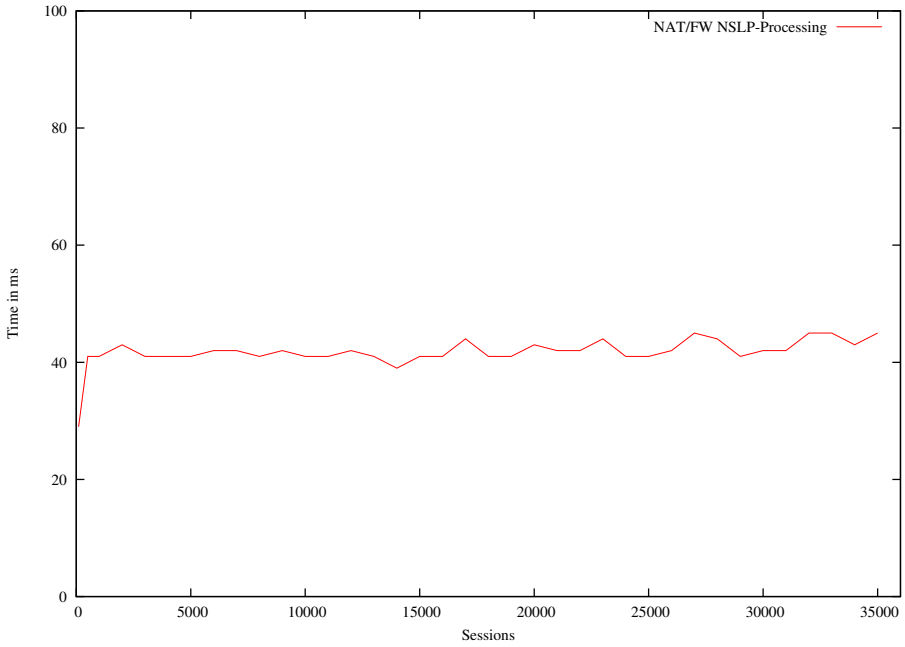


Figure 8.4: NAT/FW NSLP Processing Time, no iptables-calls

The experiment demonstrated that the NAT/FW NSLP implementation can handle a larger number of sessions of up to an amount of 35,000 sessions. Hence the session setup time experiment was run again. In this test – similar to in the previous one – the iptables-calls were not performed. Therefore, the results should demonstrate an approximation for results with a better scaling firewall implementation.
Figure 8.5 shows the results of this experiment, and compares them with the results when the iptables-call was performed. As the figure shows, the session setup time without the iptables-call increases slowly linear. With 35,000 sessions the session setup time remains under 150 ms. Figure 8.5 also shows the impact of the iptables implementation on the session setup time. With the iptables-calls, a maximum of 15,000 sessions could be created and the session setup time increases over 550 ms.

Without the iptables-calls – probably also with a better performed firewall implementation – 35,000 sessions could be created and the corresponding session setup time increases only very slowly, up to less than 150 ms.
Performance tests with others firewall implementations than iptables are not possible in the scope of this evaluation. However, they represent one interesting proposal for further study and performance testing.

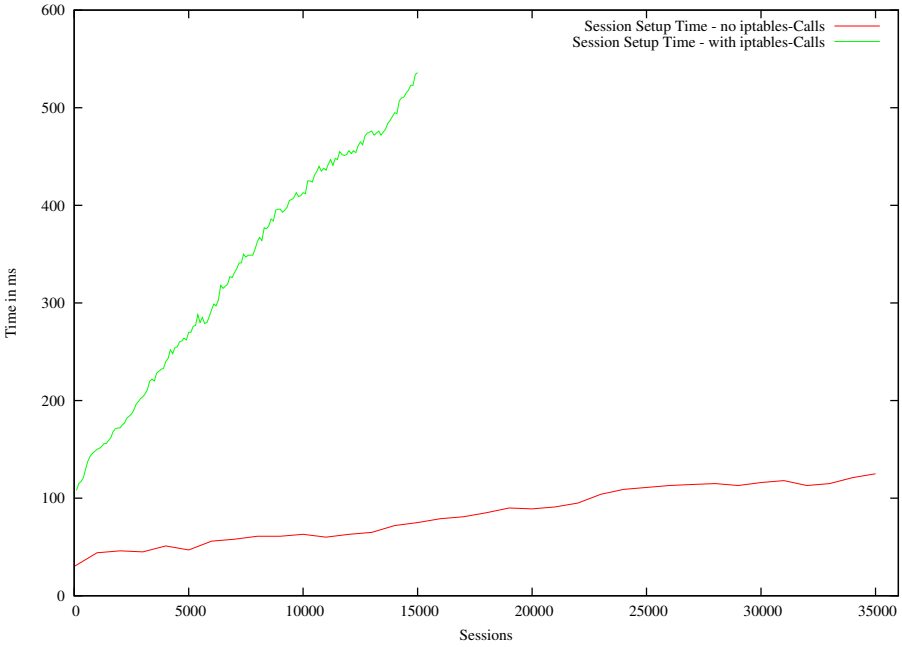


Figure 8.5: Session Setup Time

8.1.2.2.3 Round Trip Time

Figure 8.6 shows the round trip time (RTT) under different number of sessions. Therefore, one session is created and the round trip time for the refresh message of this session is measured under increasing numbers of sessions. The round trip time with only a few sessions is very low, nearly 5 ms. With below 4,000 sessions, the RTT is around 50 ms, however it increases linearly when the number of sessions grows above 6,000.
The figure shows the average of the measured RTT for NAT/FW NSLP refresh messages. A detailed investigation showed that most messages had an RTT of around 50 ms, but some messages had a high RTT. That is due to the fact that iptables stalls the packet processing during a rule set update. If there are no rule set update present when the message arrives, the RTT is furthermore around 50 ms. When a message

correlated with a rule set update, the message processing has to wait until the rule set update is finish. With another firewall implementation it can be expected that the RTT will be almost around the above noted 50 *ms*.

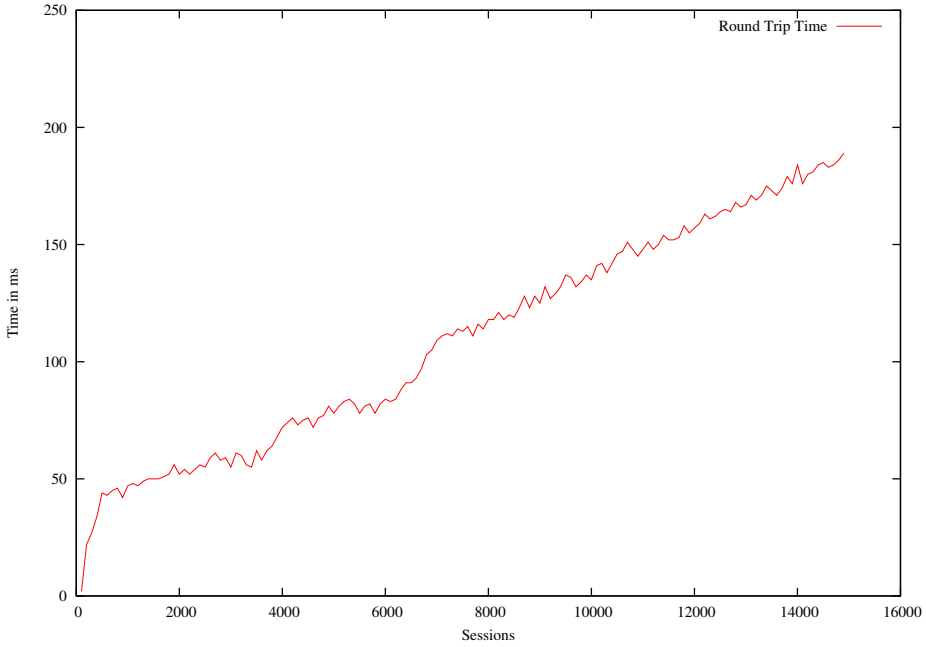


Figure 8.6: Round Trip Time

8.1.2.2.4 CPU and Memory Consumption

The final experiment was to measure the CPU and memory consumption of the NAT/FW NSLP implementation. As described above it was only possible to perform tests for less than 15,000 sessions when iptables was running (and less than 35,000 sessions if iptables is not running). That is likely due to the fact that the hardware of the testbed machines can not handle more sessions.
Figure 8.7 shows the CPU load of the NSLP forwarder under different numbers of sessions. The first observation is that the CPU load of the iptables implementation increases rapidly if it has to handle more than 2,000 sessions. The figure also confirms that the iptables implementation is the bottleneck in the experiments.
Furthermore, it can be seen that the NAT/FW NSLP implementation does not introduce a large overhead. It can also be seen that the maximum number of 35,000 sessions depends on the GIST implementation. That is due to the fact that the used GIST implementation [FreeNSIS] currently uses timers of *XORP* [XORP], which brings a large overhead to the GIST implementation, as already investigated in [DFT+06].

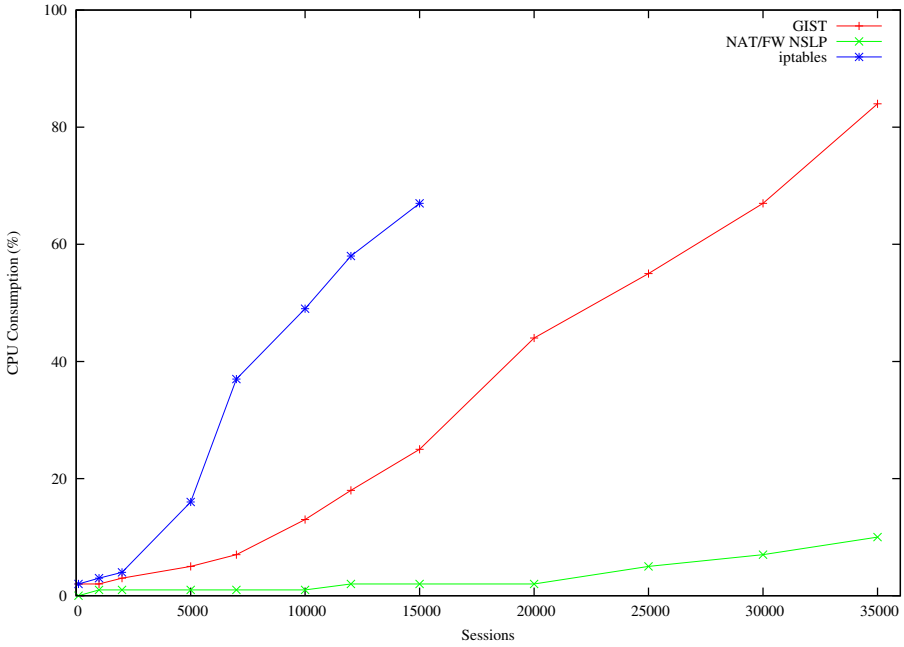


Figure 8.7: CPU Consumption

Figure 8.8 presents the impact of the number of sessions on the memory utilisation of the GIST and NAT/FW NSLP implementations at an NSLP forwarder. The memory utilisation of the GIST implementation increases nearly linear. The memory utilisation of the NAT/FW NSLP implementation also increases nearly linearly, however more slowly. This also suggests – on more advanced hardware – that the implementation may handle many more sessions.

8.1.3 Analytical Study of NSIS Based Mobile IPv6 Firewall Traversal Signaling Performance

After having analysed the NSIS and NAT/FW NSLP performance in Section 8.1.1 and Section 8.1.2, this section analyses the *NSIS based Mobile IPv6 firewall traversal* signaling performance with the help of a simple mathematical model. The analytical model may not be very comprehensive; it only presents an initial analysis, even through a simple mathematical model.

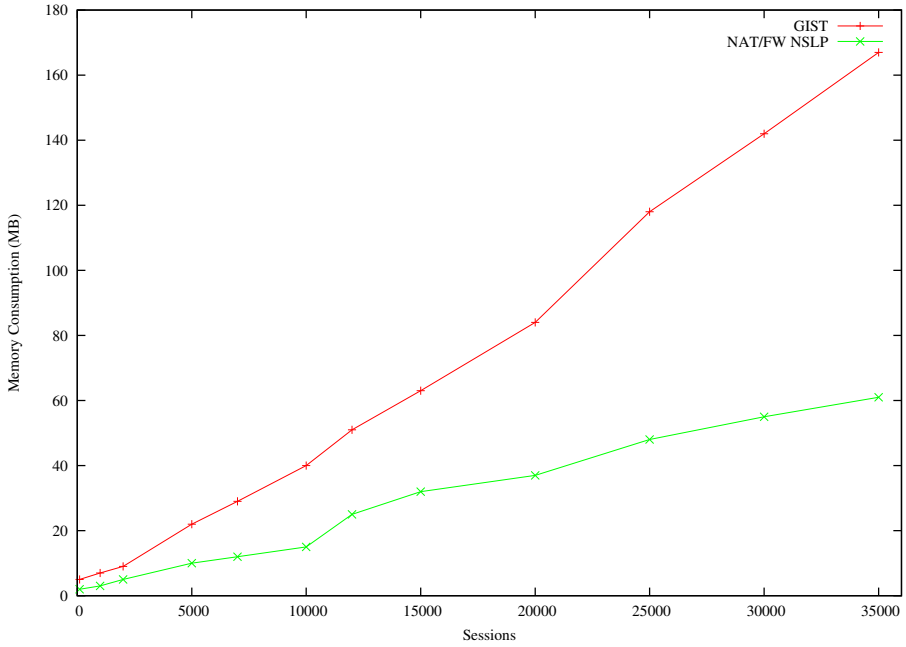


Figure 8.8: Memory Consumption

8.1.3.1 Analytical Study Scenario

Figure 8.9 depicts the topology of the scenario used for the analytical study. A typical roaming scenario where a mobile node is roaming out of its company network into a visited network, which is also a corporate network, is assumed for this study. The MN wants to communicate with its home network or its home agent in order to register its new location and additionally with another node, the corresponding node, for data communication.
The visited network deploys a firewall at the edge of its network, and thus the MN is located behind a firewall, for simplification hereafter called the MN-FW. Besides, the home network and the network of the correspondent node both deploy firewalls, namely the HA-FW and the CN-FW, as depicted in Figure 8.9.

8.1.3.2 Assumptions and Parameters

The following analysis examines in detail the delay for the NSIS based Mobile IPv6 firewall traversal approach introduced by both, the wireless and the wired part. The firewall traversal signaling delay will be analysed for the MN initiated handover case to install all firewall pinholes at the firewalls to allow Mobile IPv6 to work in presence of the firewalls. With respect to the NSIS and NAT/FW NSLP signaling performance

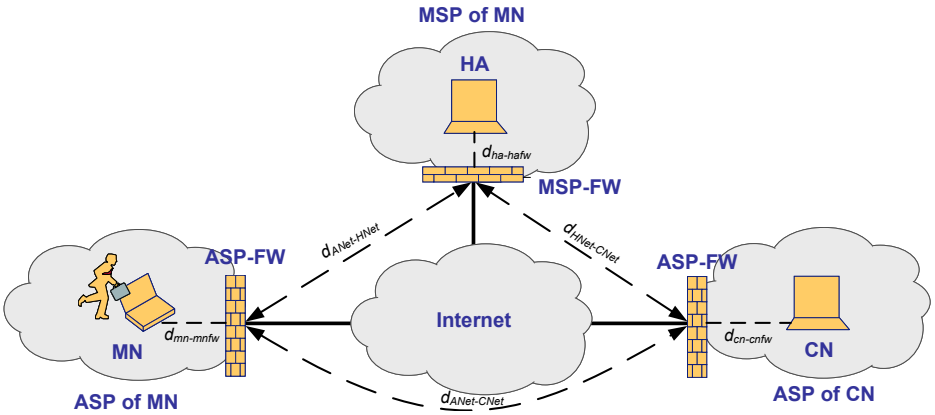


Figure 8.9: Analytical Study Scenario

study presented in Section 8.1.1 and in Section 8.1.2, the local processing delays and the local triggers on the involved nodes can be neglected compared to the transmission delay and thus will be ignored in the analysis. The firewalls are assumed to be stateful packet filter firewalls, therefore, the signaling which would be required if the firewall are stateless packet filter firewall must not be performed. In addition, the uplinks and downlinks are assumed to be equal, even in the wireless link.

Besides the scenario topology, Figure 8.9 also depicts the parameters used for the analysis. d_{x-y} denotes the transmission delay between the entities x and y. For example, $d_{mn-mnfw}$ is the required transmission delay for packets between the MN and the MN-FW, whereas $d_{ANet-HNet}$ represents the transmission delay for packets between the access network of the mobile node and its home network.

8.1.3.3 NSIS Based Mobile Firewall Traversal Signaling Delay Study

It should be noticed that the study presented in this section analyses the NSIS based Mobile IPv6 firewall traversal solution as introduced in Section 5.3.5, not the improved solution as described in Section 5.3.7. However, as the improved NSIS based Mobile IPv6 firewall traversal approach only reduces the requirements of the solution – the corresponding node is not required to be NSIS and NAT/FW NSLP aware – the reducement of signaling delay is negligible, as it only saves the transmission delay of the link $d_{cn-cnfw}$ several times. Due to the fact that the transmission delay for $d_{cn-cnfw}$ can be assumed to be very low in most cases, the results are also appropriate for the improved NSIS based Mobile IPv6 firewall traversal solution as described in Section 5.3.7.

8.1.3.3.1 IKE/IKEv2 Signaling for Establishing SAs

According to the study scenario in Figure 8.9, and the NSIS firewall signaling as described in Section 5.3.5.1.1, the delay caused by the NSIS firewall signaling for the IKE/IKEv2 signaling for establishing SAs is:

1. the time required to send the CREATE message for the IKEv2 signaling from the MN to the MN-FW ($d_{mn-mnfw}$); plus
2. the time required to send the CREATE message for the IKEv2 signaling from the MN-FW to the HA-FW ($d_{ANet-HNet}$); plus
3. the time required to send the CREATE message for the IKEv2 signaling from the HA-FW to the HA ($d_{ha-hafw}$); plus
4. the time required to send the RESPONSE message for the IKEv2 signaling from the HA to the HA-FW ($d_{ha-hafw}$); plus
5. the time required to send the RESPONSE message for the IKEv2 signaling from the HA-FW to the MN-FW ($d_{ANet-HNet}$); plus
6. the time required to send the RESPONSE message for the IKEv2 signaling from the MN-FW to the MN ($d_{mn-mnfw}$).

The delay caused by the NSIS firewall signaling for the IKE/IKEv2 signaling for establishing SAs is:

$$l_{ike} = 2d_{mn-mnfw} + 2d_{ANet-HNet} + 2d_{ha-hafw}$$

8.1.3.3.2 Binding Update and Binding Acknowledgement

The delay caused by the NSIS firewall signaling for the firewall pinholes for the Binding Update and Binding Acknowledgement, as described in Section 5.3.5.1.2, is:

1. the time required to send the CREATE message for the BU from the MN to the MN-FW ($d_{mn-mnfw}$); plus
2. the time required to send the CREATE message for the BU from the MN-FW to the HA-FW ($d_{ANet-HNet}$); plus
3. the time required to send the CREATE message for the BU from the HA-FW to the HA ($d_{ha-hafw}$); plus
4. the time required to send the RESPONSE message for the BU from the HA to the HA-FW ($d_{ha-hafw}$); plus

5. the time required to send the RESPONSE message for the BU from the HA-FW to the MN-FW ($d_{ANet-HNet}$); plus
6. the time required to send the RESPONSE message for the BU from the MN-FW to the MN ($d_{mn-mnfw}$); plus
7. the time required to send the CREATE message for the BA from the HA to the HA-FW ($d_{ha-hafw}$); plus
8. the time required to send the CREATE message for the BA from the HA-FW to the MN-FW ($d_{ANet-HNet}$); plus
9. the time required to send the CREATE message for the BA from the MN-FW to the MN ($d_{mn-mnfw}$); plus
10. the time required to send the RESPONSE message for the BA from the MN to the MN-FW ($d_{mn-mnfw}$); plus
11. the time required to send the RESPONSE message for the BA from the MN-FW to the HA-FW ($d_{ANet-HNet}$); plus
12. the time required to send the RESPONSE message for the BA from the HA-FW to the HA ($d_{ha-hafw}$).

The RESPONSE message as described in steps (4)-(6) to the initial CREATE message and the CREATE message in steps (7)-(9) are sent at the same time. Thus the transmission delay is added only once. Therefore, the delay caused by the NSIS firewall signaling for the firewall pinholes for the Binding Update and Binding Acknowledgement is:

$$l_{buba} = 3d_{mn-mnfw} + 3d_{ANet-HNet} + 3d_{ha-hafw}$$

8.1.3.3.3 Return Routability Test

The NSIS firewall signaling for the firewall pinholes for the return routability test, as described in Section 5.3.5.1.3, Section 5.3.5.2.2 and Section 5.3.5.3.3 is complicated. Therefore, the following analyses the signaling delay caused by the NSIS firewall signaling for the different Mobile IPv6 return routability test messages individually.
The delay caused by the NSIS firewall signaling for the firewall pinholes for the HOTI message between the MN and the HA is:

1. the time required to send the CREATE message for the HoTI message from the MN to the MN-FW ($d_{mn-mnfw}$); plus

2. the time required to send the CREATE message for the HoTI message from the MN-FW to the HA-FW ($d_{ANet-HNet}$); plus

3. the time required to send the CREATE message for the HoTI message from the HA-FW to the HA ($d_{ha-hafw}$); plus

4. the time required to send the RESPONSE message for the HoTI message from the HA to the HA-FW ($d_{ha-hafw}$); plus

5. the time required to send the RESPONSE message for the HoTI message from the HA-FW to the MN-FW ($d_{ANet-HNet}$); plus

6. the time required to send the RESPONSE message for the HoTI message from the MN-FW to the MN ($d_{mn-mnfw}$).

The delay caused by the NSIS firewall signaling for the firewall pinholes for the HoT message between the HA and the MN is:

7. the time required to send the CREATE message for the HoT from the HA to the HA-FW ($d_{ha-hafw}$); plus

8. the time required to send the CREATE message for the HoT from the HA-FW to the MN-FW ($d_{ANet-HNet}$); plus

9. the time required to send the CREATE message for the HoT from the MN-FW to the MN ($d_{mn-mnfw}$); plus

10. the time required to send the RESPONSE message for the HoT from the MN to the MN-FW ($d_{mn-mnfw}$); plus

11. the time required to send the RESPONSE message for the HoT from the MN-FW to the HA-FW ($d_{ANet-HNet}$); plus

12. the time required to send the RESPONSE message for the HoT from the HA-FW to the HA ($d_{ha-hafw}$).

The delay caused by the NSIS firewall signaling for the firewall pinholes for the HoTI message between the HA and the CN is:

13. the time required to send the CREATE message for the HoTI message from the HA to the HA-FW ($d_{ha-hafw}$); plus

14. the time required to send the CREATE message for the HoTI message from the HA-FW to the CN-FW ($d_{HNet-CNet}$); plus

15. the time required to send the CREATE message for the HoTI message from the CN-FW to the CN ($d_{cn-cnfw}$); plus

16. the time required to send the RESPONSE message for the HoTI message from the CN to the CN-FW ($d_{cn-cnfw}$); plus

17. the time required to send the RESPONSE message for the HoTI message from the CN-FW to the HA-FW ($d_{HNet-CNet}$); plus

18. the time required to send the RESPONSE message for the HoTI message from the HA-FW to the HA ($d_{ha-hafw}$); plus

The delay caused by the NSIS firewall signaling for the firewall pinholes for the HoT message between the CN and the HA is:

19. the time required to send the EXT message for the HoT from the HA to the HA-FW ($d_{ha-hafw}$); plus

20. the time required to send the RESPONSE message for the HoT from the HA-FW to the HA ($d_{ha-hafw}$).

The delay caused by the NSIS firewall signaling for the firewall pinholes for the COTI message between the MN and the CN is:

21. the time required to send the CREATE message for the CoTI from the MN to the MN-FW ($d_{mn-mnfw}$); plus

22. the time required to send the CREATE message for the CoTI from the MN-FW to the CN-FW ($d_{ANet-CNet}$); plus

23. the time required to send the CREATE message for the CoTI from the CN-FW to the CN ($d_{cn-cnfw}$); plus

24. the time required to send the RESPONSE message for the CoTI from the CN to the CN-FW ($d_{cn-cnfw}$); plus

25. the time required to send the RESPONSE message for the CoTI from the CN-FW to the MN-FW ($d_{ANet-CNet}$); plus

26. the time required to send the RESPONSE message for the CoTI from the MN-FW to the MN ($d_{mn-mnfw}$); plus

The delay caused by the NSIS firewall signaling for the firewall pinholes for the COT message between the CN and the MN is:

27. the time required to send the EXT message for the CoT from the MN to the MN-FW ($d_{mn-mnfw}$); plus

28. the time required to send the RESPONSE message for the CoT from the MN-FW to the MN ($d_{mn-mnfw}$).

Some of this NSIS firewall signaling takes place at the same time. The signaling as described in steps (4), (7), (13) and (19) is initiated at the same time. Additionally, the signaling as described in steps (1), (21) and (27) is also initiated at the same time. The delay for steps (27)-(28) is always lower than for steps (21)-(26) ($2d_{mn-mnfw} < 2d_{mn-mnfw} + 2d_{ANet-CNet} + 2d_{cn-cnfw}$). This is also the case for the delay of the steps (19) and (20), which is always lower than the delay for steps (7)-(12). Therefore, the overall signaling caused by the NSIS firewall signaling for the return routability test is:

$$l_{hot(i)} = d_{mn-mnfw} + d_{ANet-HNet} + d_{ha-hafw} + max(2d_{mn-mnfw} + 2d_{ANet-HNet} + 2d_{ha-hafw}, 2d_{ha-hafw} + 2d_{HNet-CNet} + 2d_{cn-cnfw})$$

$$l_{cot(i)} = 2d_{mn-mnfw} + 2d_{ANet-CNet} + 2d_{cn-cnfw}$$

$$l_{rrt} = max(l_{hot(i)}, l_{cot(i)})$$

8.1.3.3.4 Binding Update and Binding Acknowledgement CN

The delay caused by the NSIS firewall signaling for the firewall pinholes for the Binding Update and Binding Acknowledgement to the CN, as described in Section 5.3.5.1.4 and Section 5.3.5.2.3, is:

1. the time required to send the CREATE message for the BU from the MN to the MN-FW ($d_{mn-mnfw}$); plus

2. the time required to send the CREATE message for the BU from the MN-FW to the CN-FW ($d_{ANet-CNet}$); plus

3. the time required to send the CREATE message for the BU from the CN-FW to the CN ($d_{cn-cnfw}$); plus

4. the time required to send the RESPONSE message for the BU from the CN to the CN-FW ($d_{cn-cnfw}$); plus

5. the time required to send the RESPONSE message for the BU from the CN-FW to the MN-FW ($d_{ANet-CNet}$); plus

6. the time required to send the RESPONSE message for the BU from the MN-FW to the MN ($d_{mn-mnfw}$); plus

7. the time required to send the EXT message for the BA from the MN to the MN-FW ($d_{mn-mnfw}$); plus

8. the time required to send the RESPONSE message for the BA from the MN-FW to the MN ($d_{mn-mnfw}$).

As the NSIS firewall signaling for the BU and the BA is taking place at the same time and the signaling delay of the CREATE message is always higher than for the EXT message ($2d_{mn-mnfw} + 2d_{ANet-CNet} + 2d_{cn-cnfw} > 2d_{mn-mnfw}$), the delay caused by the NSIS signaling for the Binding Update and Binding Acknowledgement to the CN is:

$$l_{bubacn} = 2d_{mn-mnfw} + 2d_{ANet-CNet} + 2d_{cn-cnfw}$$

8.1.3.3.5 Route Optimisation

The delay caused by the NSIS firewall signaling for the firewall pinholes for route optimised data traffic between the MN and the CN, as described in Section 5.3.5.1.5 and Section 5.3.5.2.4, is:

1. the time required to send the CREATE message for route optimised data traffic (MN to CN) from the MN to the MN-FW ($d_{mn-mnfw}$); plus

2. the time required to send the CREATE message for route optimised data traffic (MN to CN) from the MN-FW to the CN-FW ($d_{ANet-CNet}$); plus

3. the time required to send the CREATE message for route optimised data traffic (MN to CN) from the CN-FW to the CN ($d_{cn-cnfw}$); plus

4. the time required to send the RESPONSE message for route optimised data traffic (MN to CN) from the CN to the CN-FW ($d_{cn-cnfw}$); plus

5. the time required to send the RESPONSE message for route optimised data traffic (MN to CN) from the CN-FW to the MN-FW ($d_{ANet-CNet}$); plus

6. the time required to send the RESPONSE message for route optimised data traffic (MN to CN)from the MN-FW to the MN ($d_{mn-mnfw}$); plus

7. the time required to send the EXT message for route optimised data traffic (CN to MN) from the MN to the MN-FW ($d_{mn-mnfw}$); plus

8. the time required to send the RESPONSE message for route optimised data traffic (CN to MN) from the MN-FW to the MN ($d_{mn-mnfw}$).

As the NSIS firewall signaling for the route optimised data traffic takes place at the same time and the signaling delay for the EXT message is always lower than for the CREATE message ($2d_{mn-mnfw} < 2d_{mn-mnfw} + 2d_{ANet-CNet} + 2d_{cn-cnfw}$), the delay caused by the NSIS firewall signaling for the route optimised data traffic is:

$$l_{ro} = 2d_{mn-mnfw} + 2d_{ANet-CNet} + 2d_{cn-cnfw}$$

8.1.3.3.6 Bi-directional Tunneling

The NSIS firewall signaling for the firewall pinholes for the bi-directional tunneled data traffic, as described in Section 5.3.5.1.6, Section 5.3.5.2.5 and Section 5.3.5.3.5 is complicated. Therefore, the following analyses the signaling delay caused by the NSIS firewall signaling for the different bi-directional tunneled data traffic firewall pinholes individually.
The delay caused by the NSIS firewall signaling for the firewall pinholes for the bi-directional tunneled data traffic between the MN and the HA is:

1. the time required to send the CREATE message for the bi-directional tunneled data traffic from the MN to the MN-FW ($d_{mn-mnfw}$); plus
2. the time required to send the CREATE message for the bi-directional tunneled data traffic from the MN-FW to the HA-FW ($d_{ANet-HNet}$); plus
3. the time required to send the CREATE message for the bi-directional tunneled data traffic from the HA-FW to the HA ($d_{ha-hafw}$); plus
4. the time required to send the RESPONSE message for the bi-directional tunneled data traffic from the HA to the HA-FW ($d_{ha-hafw}$); plus
5. the time required to send the RESPONSE message for the bi-directional tunneled data traffic from the HA-FW to the MN-FW ($d_{ANet-HNet}$); plus
6. the time required to send the RESPONSE message for the bi-directional tunneled data traffic from the MN-FW to the MN ($d_{mn-mnfw}$).

The delay caused by the NSIS firewall signaling for the firewall pinholes for the bi-directional tunneled data traffic between the HA and the MN is:

7. the time required to send the EXT message for the bi-directional tunneled data traffic from the MN to the MN-FW ($d_{mn-mnfw}$); plus
8. the time required to send the RESPONSE message for the bi-directional tunneled data traffic from the MN-FW to the MN ($d_{mn-mnfw}$).

The delay caused by the NSIS firewall signaling for the firewall pinholes for the bi-directional tunneled data traffic between the HA and the CN is:

9. the time required to send the CREATE message for the bi-directional tunneled data traffic from the HA to the HA-FW ($d_{ha-hafw}$); plus
10. the time required to send the CREATE message for the bi-directional tunneled data traffic from the HA-FW to the CN-FW ($d_{HNet-CNet}$); plus
11. the time required to send the CREATE message for the bi-directional tunneled data traffic from the CN-FW to the CN ($d_{cn-cnfw}$); plus
12. the time required to send the RESPONSE message for the bi-directional tunneled data traffic from the CN to the CN-FW ($d_{cn-cnfw}$); plus
13. the time required to send the RESPONSE message for the bi-directional tunneled data traffic from the CN-FW to the HA-FW ($d_{HNet-CNet}$); plus
14. the time required to send the RESPONSE message for the bi-directional tunneled data traffic from the HA-FW to the HA ($d_{ha-hafw}$); plus

The delay caused by the NSIS firewall signaling for the firewall pinholes for the bi-directional tunneled data traffic between the CN and the HA is:

15. the time required to send the EXT message for the bi-directional tunneled data traffic from the HA to the HA-FW ($d_{ha-hafw}$); plus
16. the time required to send the RESPONSE message for the bi-directional tunneled data traffic from the HA-FW to the HA ($d_{ha-hafw}$).

The NSIS firewall signaling for the steps (1)-(6) and (7)-(8) takes place at the same time as the steps (9)-(14) and (15)-(16). However, the delay for steps (1)-(6) and (9)-(14) is always higher than for steps (7)-(8), respectively (15)-(16) ($2d_{mn-mnfw} + 2d_{ANet-HNet} + 2d_{ha-hafw} > 2d_{mn-mnfw}$). Therefore, the overall signaling caused by the NSIS firewall signaling for the bi-directional tunneled data traffic is:

$$l_{bt} = d_{mn-mnfw} + d_{ANet-HNet} + d_{ha-hafw} + max(d_{mn-mnfw} + d_{ANet-HNet} + d_{ha-hafw}, 2d_{ha-hafw} + 2d_{HNet-CNet} + 2d_{cn-cnfw})$$

8.1.3.3.7 Summary

The NSIS based Mobile IPv6 firewall traversal transmission delay can be summarised as depicted in Table 8.1.
The additional overall transmission delay introduced by the NSIS based Mobile IPv6 firewall traversal in the Mobile IPv6 procedure is:

$$l_{nsis} = l_{ike} + l_{buba} + l_{rrt} + l_{bubacn} + max(l_{ro}, l_{bt})$$

Mobile IPv6 Traffic			NSIS Signaling Transmission Delay
IKEv2 Signaling	l_{ike}	=	$2d_{mn-mnfw} + 2d_{ANet-HNet} + 2d_{ha-hafw}$
BU and BA	l_{buba}	=	$3d_{mn-mnfw} + 3d_{ANet-HNet} + 3d_{ha-hafw}$
Return Routability Test	l_{rrt}	=	$max((d_{mn-mnfw} + d_{ANet-HNet} + d_{ha-hafw} + max(2d_{mn-mnfw} + 2d_{ANet-HNet} + 2d_{ha-hafw}, 2d_{ha-hafw} + 2d_{HNet-CNet} + 2d_{cn-cnfw})), (2d_{mn-mnfw} + 2d_{ANet-CNet} + 2d_{cn-cnfw}))$
BU and BA CN	l_{bubacn}	=	$2d_{mn-mnfw} + 2d_{ANet-CNet} + 2d_{cn-cnfw}$
Route Optimisation	l_{ro}	=	$2d_{mn-mnfw} + 2d_{ANet-CNet} + 2d_{cn-cnfw}$
Bi-directional Tunneling	l_{bt}	=	$d_{mn-mnfw} + d_{ANet-HNet} + d_{ha-hafw} + max(d_{mn-mnfw} + d_{ANet-HNet} + d_{ha-hafw}, 2d_{ha-hafw} + 2d_{HNet-CNet} + 2d_{cn-cnfw})$

Table 8.1: NSIS Based Mobile IPv6 Firewall Traversal Transmission Delay

8.1.3.4 Analysis Results

Based on the theoretical model presented in the previous sections and the scenario earlier depicted in Figure 8.9, the NSIS based Mobile IPv6 firewall traversal transmission delay has been numerically analysed with various link delays in milliseconds (ms). According to discussions on link delays, for instance in [LMK04] and [SY06], we can assume:

- $d_{mn-mnfw}$: 15 ms, deviation: 15 ms,
- $d_{ANet-HNet}$: 10 ms, deviation: 50 ms,
- $d_{ANet-CNet}$: 10 ms, deviation: 50 ms,
- $d_{HNet-CNet}$: 10 ms, deviation: 50 ms,
- $d_{ha-hafw}$: 1 ms, no deviation,
- $d_{cn-cnfw}$: 1 ms, no deviation.

The following presents the NSIS based Mobile IPv6 firewall traversal transmission delay performance compared to the handover latency when utilising Mobile IPv6 as specified in [JPA04] without a firewall traversal approach.
Commonly, the handover latency is divided into four phases: movement detection, CoA configuration, home agent registration and route optimisation [XCZ+07, VPS+06]. The NSIS based Mobile IPv6 firewall traversal does not affect the first two phases, movement detection and CoA configuration, as it takes place afterwards.

Hence, the following analysis only compares the delays for the last two steps, home agent registration and route optimisation.
Therefore, the handover delay has been calculated which is required to perform the Mobile IPv6 procedure. More precisely, the delay to update the binding cache at the home agent, to run the return routability test and to update the binding cache at the correspondent node. At this point, the mobile node is able to send route optimised or bi-directional tunneled data traffic. In case of NSIS based Mobile IPv6 firewall traversal, the transmission delay also includes the NSIS firewall signaling to install the required firewall pinholes at the involved nodes.
However, the results of the normal Mobile IPv6 and the NSIS based Mobile IPv6 firewall traversal are not directly comparable as Mobile IPv6 would fail in a firewall scenario, e.g., the one shown in Figure 8.9. Nevertheless, it is the only possibility to evaluate the NSIS based Mobile IPv6 firewall traversal transmission delay performance as no Mobile IPv6 firewall traversal approach is currently implemented, nor are results available.

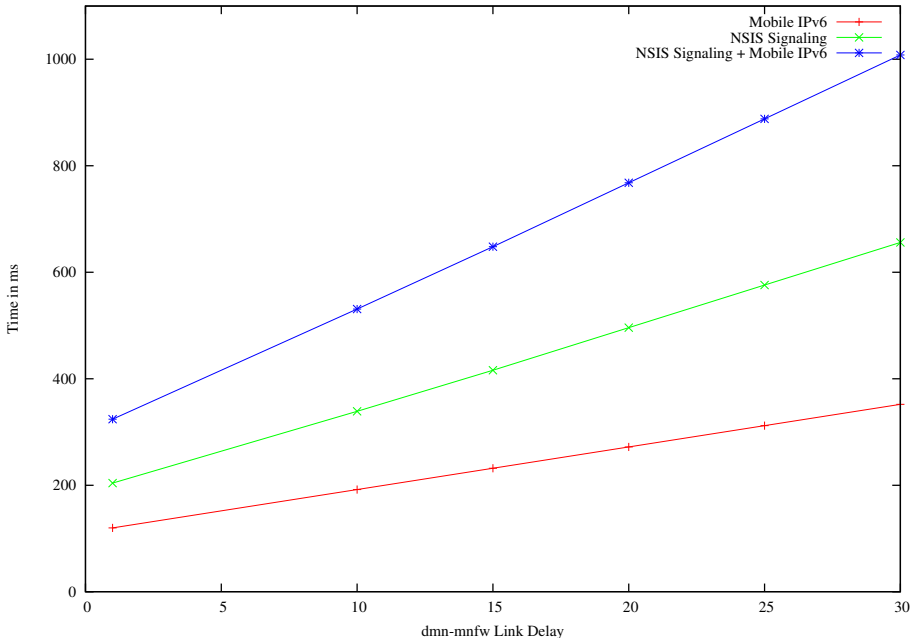


Figure 8.10: Varying Wireless Delay ($d_{mn-mnfw}$)

It is important to notice that several Mobile IPv6 performance evaluations and handover latency studies, for instance [XCZ+07, LSJP06, VPS+06] have verified that the excessive handover time of Mobile IPv6 is spent in the first two phases of the handover procedure, most of all in the CoA address configuration phase. The results

in [XCZ+07] show that the overall Mobile IPv6 handover latency in a local testbed, without long transmission delays to HA and CN, is still between 3,000 and 4,400 ms, with an average of 3,600 ms. Obviously, Mobile IPv6 can not fulfill the requirements of latency-sensitive applications such as VoIP. These overall Mobile IPv6 handover latency values must be considered in the following NSIS based Mobile IPv6 firewall traversal transmission delay study.

Firstly, the transmission delay performance has been analysed in the most important scenarios, namely with variances on the wireless link ($d_{mn-mnfw}$) and with variances on the link between the foreign and the home network of the mobile node ($d_{ANet-HNet}$).

Secondly, the transmission delay performance has also been analysed with variances on the link between the foreign network of the mobile node and the network of the correspondent node ($d_{ANet-CNet}$) and with variances on the link between the home network of the mobile node and the network of the correspondent node ($d_{HNet-CNet}$).

Figure 8.10 shows how variances on the wireless link ($d_{mn-mnfw}$) influence the NSIS based Mobile IPv6 firewall traversal transmission delay compared to the Mobile IPv6 transmission delay. It can be seen that the delay introduced by the NSIS firewall signaling is consistent compared to Mobile IPv6. The transmission delay of Mobile IPv6 together with NSIS based Mobile IPv6 firewall traversal is relatively stable, namely the transmission delay of the normal Mobile IPv6 2.75-times.

If the delay on the wireless link is less the 10 ms, the overall Mobile IPv6 with NSIS firewall signaling is quite acceptable with less than 500 ms in the scenario. However, if the wireless link delay is higher, the performance is getting spirally downward, up to 1,000 ms if the wireless link delay nearly 30 ms. Whereas a 500 ms delay is acceptable for some not-seamless application, a 1,000 ms delay is worse compared to the 350 ms for normal Mobile IPv6.

At first glance, the NSIS based Mobile IPv6 firewall traversal approach transmission delay performance is worse compared to the normal Mobile IPv6 transmission delay performance. However, as mention before, most of the handover time of Mobile IPv6 is spent in the first two phases of the handover procedure. [XCZ+07] have shown that in a local testbed – without long transmission delays – the average handover latency is 3,600 ms and nearly 60% of the overall handover latency is spent in the first two phases. Having this in mind, the transmission delay introduced by the NSIS based Mobile IPv6 firewall traversal approach is acceptable as Mobile IPv6 would fail in this scenario without any firewall traversal approach. Evidently, the NSIS based Mobile IPv6 firewall traversal approach does not meet the requirements of latency-sensitive applications like VoIP, but this is also the case for Mobile IPv6.

Figure 8.11 shows how variances on the home network link ($d_{ANet-HNet}$) influence the transmission delays. As a matter of fact the impact of the transmission delay on the home network link to NSIS Mobile IPv6 firewall signaling is higher compared to the impact of the wireless link delay (Figure 8.10) and to the Mobile IPv6 delay.

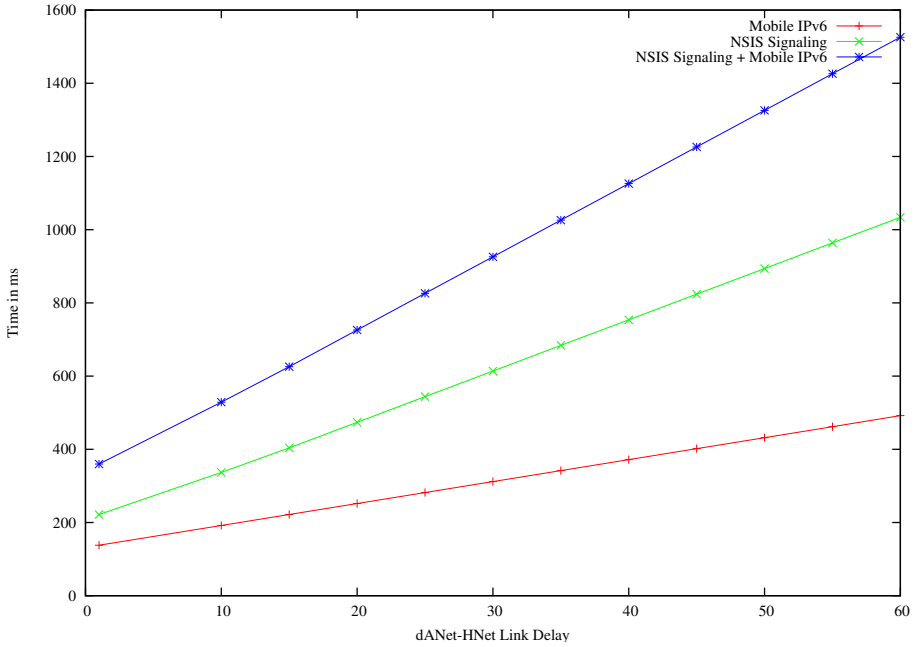


Figure 8.11: Varying Home Network Link Delay ($d_{ANet-HNet}$)

Even with a relatively low delay of 30 ms on the home network link, the Mobile IPv6 with NSIS firewall signaling transmission delay is still around 1,000 ms and reaches nearly 1,600 ms if $d_{ANet-HNet}$ is 60 ms. This indicates that an explicit firewall traversal solution like the NSIS based Mobile IPv6 firewall approach suffers from a high transmission delay to the home network. This mainly bases on the fact that the firewall traversal approach has to install firewall pinholes along the whole path between the mobile node and the home agent. As the NAT/FW NSLP currently does not support signaling for firewall pinholes different from the MRI, it has to signal the path also for the reverse direction. Additionally, it requires signaling the path several times as the NAT/FW NSLP currently does not support signaling for several firewall pinholes at the same time. The feasibility to signal for several firewall pinholes at the same time would rapidly decrease the NSIS firewall signaling transmission delay. Chapter 9 discusses these issues in detail.

These results denote that the NSIS based Mobile IPv6 firewall traversal is suitable to only a limited extent if the transmission delay to the home network is increased. However, when comparing these results to the average handover latency of 3,600 ms, the NSIS based Mobile IPv6 firewall traversal transmission delay performance might still be acceptable for non latency-sensitive applications. Although it suffers from the increasing transmission delay to the home network, it nevertheless allows to run

Mobile IPv6 in presence of firewalls.

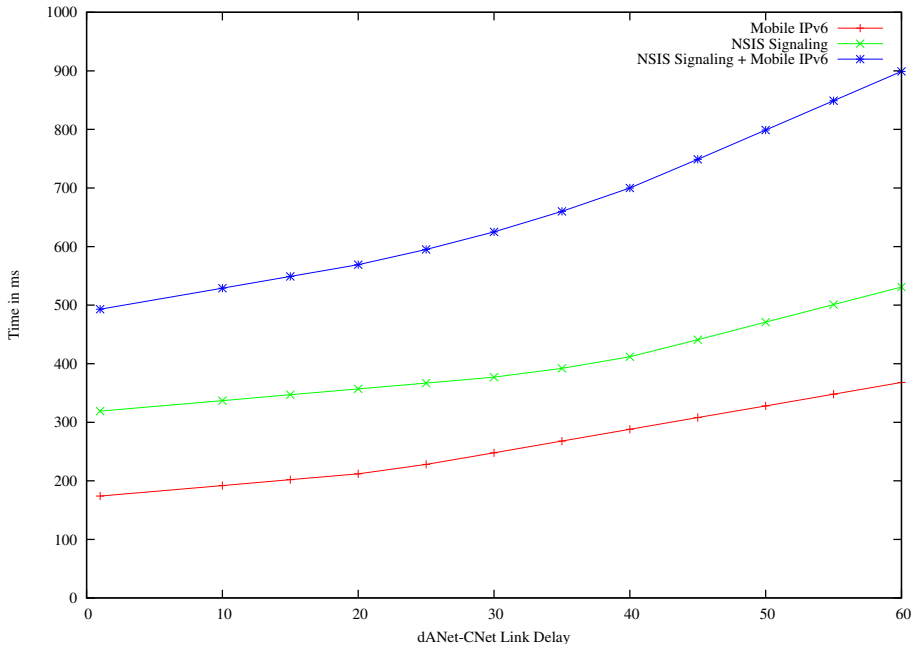


Figure 8.12: Varying Correspondent Network Link Delay ($d_{ANet-CNet}$)

Figure 8.12 shows the results with variances on the link between the foreign network of the mobile node and the network of the correspondent node ($d_{ANet-CNet}$). It can be seen that the impact of the transmission delay on this link on the NSIS based Mobile IPv6 firewall signaling latency is much lower compared to the impact of the home network link delay (Figure 8.11).
Even with a relatively high delay of 30 ms on the link, the Mobile IPv6 with NSIS firewall signaling transmission delay is still around 600 ms and reaches nearly 900 ms if $d_{ANet-CNet}$ is 60 ms. This shows that the influences of the $d_{ANet-CNet}$ link on the NSIS based Mobile IPv6 firewall signaling transmission delay is very limited. As mentioned before, the overall NSIS based Mobile IPv6 firewall signaling transmission delay suffers from an increasing home network link delay. Because the NSIS firewall signaling for the return routability test, more precisely, the NSIS firewall signaling for the firewall pinholes for the HoTI/HoT and CoTI/CoT messages takes place at the same time (as described in detail in Section 8.1.3.3.3), the NSIS firewall signaling for the HoTI/HoT firewall pinholes introduces more transmission delay than the CoTI/CoT firewall pinholes. Therefore, the increasing $d_{ANet-CNet}$ link delay only influences the NSIS based Mobile IPv6 firewall signaling transmission delay if it is higher than ∼35 ms.

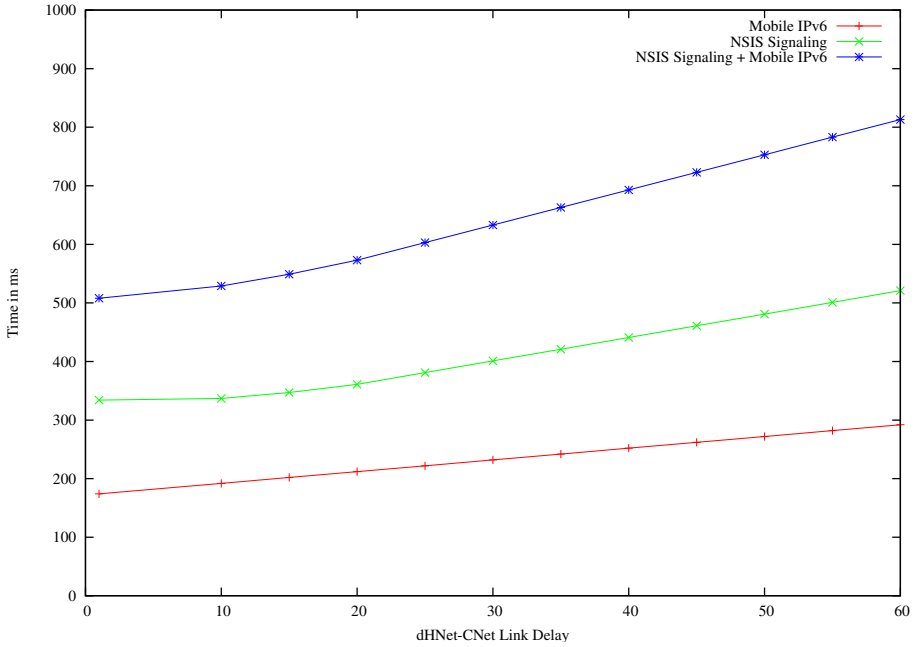


Figure 8.13: Varying Home to Correspondent Network Link Delay ($d_{HNet-CNet}$)

Figure 8.13 shows the results with variances on the link between the home network of the mobile node and the network of the correspondent node ($d_{HNet-CNet}$). The results demonstrate that the impact of the $d_{HNet-CNet}$ link delay on the NSIS based Mobile IPv6 firewall signaling transmission delay is very limited. This is due to the fact that this link is barely utilised in the NSIS based Mobile IPv6 firewall traversal approach. The differences of the NSIS firewall signaling transmission delay with a very low $d_{HNet-CNet}$ link delay of 1 ms and a high $d_{HNet-CNet}$ link delay of 60 ms are only 300 ms. These variances are insignificant with respect to the very high average handover latency of Mobile IPv6, figured out in [XCZ+07, LSJP06, VPS+06].

8.1.3.5 Summary

This section has presented an analytical study of the NSIS based Mobile IPv6 firewall traversal signaling performance compared to the handover latency when utilising Mobile IPv6 as specified in [JPA04] without a firewall traversal approach.
As the NSIS based Mobile IPv6 firewall traversal does not affect the first two phases of the Mobile IPv6 handover latency, the analysis only compares the delay of the last two steps, the home agent registration and the route optimisation.
The results of the NSIS based Mobile IPv6 firewall traversal and the normal Mobile IPv6 latency are not directly comparable as Mobile IPv6 would fail in a firewall

environment. Nevertheless, it is the only possibility to evaluate the NSIS based Mobile IPv6 firewall traversal transmission delay performance as no Mobile IPv6 firewall traversal approach is currently implemented, nor are results available.
The analysis study has identified that the delay introduced by the NSIS firewall signaling is consistent compared to Mobile IPv6 and that the approach mostly suffers from the transmission delay to the home network. The transmission delay of Mobile IPv6 together with the NSIS based Mobile IPv6 firewall traversal is relatively stable, namely the transmission delay of normal Mobile IPv6 2.75-times. Obviously, this does not meet the requirements of latency-sensitive applications like VoIP, but this is also the case for normal Mobile IPv6 as described in detail beforehand. Nevertheless, the transmission delay introduced by the NSIS based Mobile IPv6 firewall traversal should be acceptable for the most not-seamless applications.

8.2 Mobile IPv6 Application Layer Gateway

As mentioned in Section 7.2, the developed prototype implementation is still in an unstable condition and currently only supports the basic primitives. Therefore, it is not possible to gain performance evaluation results using this prototype implementation. In contrast to the *NSIS based Mobile IPv6 firewall traversal* approach, the firewall traversal delay introduced by the *Mobile IP6 Application Layer Gateway* is negligible, as it only inserts additional deep-packet inspection and processing time at the firewalls. Therefore, the performance of the *Mobile IP6 Application Layer Gateway* firewall traversal approach depends on the utilised firewall implementation, in this case the *netfilter/iptables* [netfilter] firewall implementation.
[KG04] and [HPS03] have investigated the performance of the iptables implementation and figured out that it lacks efficiency when handling a large amount of sessions. *"ipset"* [IPset] and *"nf-HiPAC"* [HiPAC] are promising candidates to overcome these limitations.
Nevertheless, any Mobile IPv6 firewall traversal approach would suffer from the firewall performance when it encounters a large amount of firewall sessions. Therefore, the delay introduced by the *Mobile IP6 Application Layer Gateway* firewall traversal approach is negligible compared to the Mobile IPv6 firewall traversal delay of any other approach as it only inserts additional deep-packet inspection and processing time on the firewalls. This is mainly based on the fact that an Application Layer Gateway based approach is performing the Mobile IPv6 firewall traversal *on-the-fly*.

9 Open Issues and Future Work

Today's infrastructure mostly supports Mobile IPv4, rarely Mobile IPv6. Therefore, it is necessary to investigate a Mobile IPv6/IPv4 dual stack solution for the two proposed Mobile IPv6 firewall traversal approaches in the future. Additionally, several new mobility proposal have been made in the past which extend or replace Mobile IPv6, for example, *Proxy Mobile IPv6* (PMIPv6) [GLD+08], *Mobile IPv6 Fast Handovers* (FMIPv6) [Koo08] or *Hierarchical Mobile IPv6 Mobility Management* (HMIPv6) [SCMB05]. The Mobile IPv6 firewall traversal solutions presented in this thesis potentially can be used to also enable firewall traversal for these new protocols. However, further study with respect to these mobility solutions is necessary.

The *NSIS based Mobile IPv6 firewall traversal* solution presented in Section 5.3 can deal with the problems of having firewalls in Mobile IPv6 environments. However, the approach as described might not be efficient enough for some environments. As a result, the optimisation of the signaling exchange and the reduction of the signaling delay are for further study. Overall, more work on performance optimisations and scalability investigations are necessary.

NSIS and NAT/FW NSLP based firewall traversal requires strong authentication and authorisation. An initial set of security mechanisms is proposed in Section 6.1 but further work is needed to study the security properties in detail, potentially including a formal analysis. Due to the security considerations, the improved NSIS based Mobile IPv6 firewall traversal utilises the pre-configured proxy mode as describes in Section 5.3.6 and therefore requires the firewall at the edge of the CN's ASP to be deployed. Further investigation of the authentication and authorisation aspects of the timeout based approach is necessary to deploy it for NSIS based Mobile IPv6 firewall traversal which would reduce the requirements of the NSIS based approach.

As described in Section 8.1.3.4, the approach of combined signaling (for control and data traffic) could be useful for the NSIS based Mobile IPv6 firewall traversal approach, but currently the NSIS NAT/FW NSLP protocol does not support installing multiple firewall pinhole rules at the same time. However, the feasibility to signal for several firewall pinholes at the same time would rapidly decrease the NSIS firewall signaling transmission delay. Further investigation of this issue is required as the NSIS and NAT/FW NSLP based Mobile IPv6 firewall traversal approach would highly benefit from such a support.

The *Mobile IPv6 Application Layer Gateway* approach presented in Section 5.4 can also deal with the problems of having firewalls in Mobile IPv6 environments. How-

ever, as described in Section 6.2, authentication and authorisation requirements are not yet investigated in detail. Further study on these issues is necessary for an Application Layer Gateway based Mobile IPv6 firewall traversal approach. [LC04] specifies authentication, authorisation and accounting requirements for the *Session Initiation Protocol* (SIP) [RSC+02]. An authentication and authorisation solution for the Application Layer Gateway based approach could be very similar, whereas the *SIP Proxies* have to be replaced by the *Mobile IPv6 Application Layer Gateway* aware firewalls.

10 Conclusion

This thesis is one of the earliest attempts in the community to investigate the problems and impacts when middleboxes, especially firewalls, are placed in Mobile IPv6 environments. Several potential solutions have been identified to deal with the Mobile IPv6 firewall traversal problems and investigated in detail. The analysis of each technology included an overview of the technology, detailing the most important issues, showed how the individual solutions work, and evaluated them in terms of their applicability for Mobile IPv6 firewall traversal.

Two different solutions have been proposed to overcome the Mobile IPv6 firewall traversal problem; both have been designed as main contributions of this thesis. The first one bases on the NSIS framework and the NAT/FW NSLP protocol, namely the *NSIS based Mobile IPv6 firewall traversal*. For this solution, it introduced the NSIS and the NAT/FW protocols and showed how they can be utilised for Mobile IPv6 firewall traversal. Afterwards, it presented the second proposed solution, the *Application Layer Gateway based Mobile IP6 firewall traversal*.

Their applicability has also been verified with the help of two proof-of-concept implementations, developed as part of this thesis. Additionally, it has explained how authentication and authorisation can be accomplished for the proposed Mobile IPv6 firewall traversal solutions. Finally, it has evaluated and analysed the developed proof-of-concept implementation and the two proposed Mobile IPv6 firewall traversal approaches in general.

To summarise, this thesis has shown that both solutions can address the issues caused by stateful packet filter firewalls encountered in Mobile IPv6 networks. Besides, it has proven that firewall traversal in Mobile IPv6 environments is technically feasible, for both the *NSIS based Mobile IPv6 firewall traversal* and the *Application Layer Gateway based Mobile IP6 firewall traversal*.

The performance results show that the NSIS based Mobile IPv6 firewall traversal implementation is scalable and behaves well, even in a low-end hardware environment and that the selection of an appropriate firewall implementation has a big impact on the performance and more specifically the choice of *netfilter/iptables*. The results of the NSIS based Mobile IPv6 firewall traversal analytical study show that this solution introduces additional delay to signal for the NSIS firewall pinholes. However, compared to the overall Mobile IPv6 latency without handover latency optimisation like *FMIPv6* of *HMIPv6*, the NSIS based Mobile IPv6 firewall traversal caused delay is acceptable. Nevertheless, a seamless handover, as required for some applications, for

example VoIP, is only possible in certain scenarios.
In contrast, the *Application Layer Gateway based Mobile IP6 firewall traversal* approach caused firewall traversal delay is negligible, as it only inserts additional deep-packet inspection and processing time at the firewalls.
The *Application Layer Gateway based Mobile IP6 firewall traversal* approach benefits from its *on-the-fly* nature, the comparatively small requirements and the wide spread usage. Whereas the *NSIS based Mobile IPv6 firewall traversal* benefits from the NSIS two-layer signaling paradigm and the associated foreclosures of protocol knowledge on the firewall as well as the strong authentication and authorisation options, it suffers from the currently missing worldwide intra-community acceptance. Therefore, the long-term perspective of the *Application Layer Gateway based Mobile IP6 firewall traversal* is more promising than the *NSIS based Mobile IPv6 firewall traversal*. Another advantage of the *Mobile IPv6 Application Layer Gateway* solution is that it is adopted as MEXT working group draft.

A Appendix

A.1 Detailed Specification of each Firewall Pinhole Format

This specifies for all Mobile IPv6 signaling and data messages the format of the firewall pinhole which allows these messages to traverse the firewall. Besides the format of the firewall pinhole, it also describe the *netfilter* [netfilter] iptables-call to install this kind of firewall pinhole rule.

A.1.1 IKEv2 between MN and HA

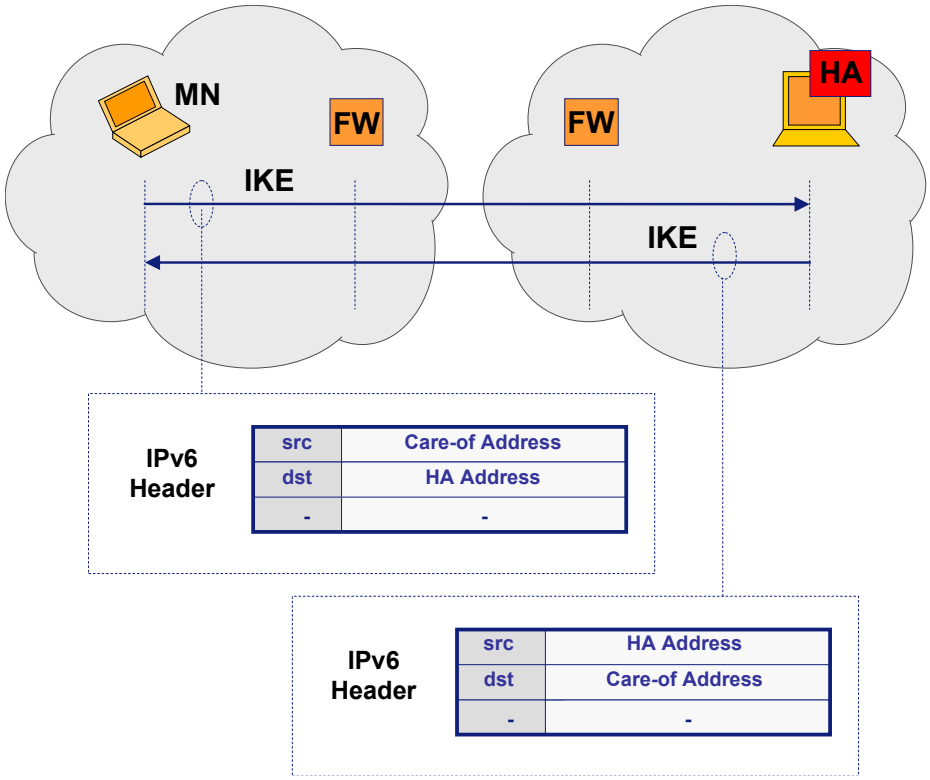


Figure A.1: IKEv2 between MN and HA

For the IKEv2 messages between the MN and the HA, the firewall pinhole has to match against:

- src address: Care-of Address
- dst address: HA Address
- protocol
- destination port

The iptables-call for installing this firewall pinhole is:

```
ip6tables -A FORWARD -p udp -s "Care-of Address" -d "HA Address"
--dport 500 -j ACCEPT
```

For the IKEv2 messages between the HA and the MN, the firewall pinhole has to match against:

- src address: HA Address
- dst address: Care-of Address
- protocol
- destination port

The iptables-call for installing this firewall pinhole is:

```
ip6tables -A FORWARD -p udp -s "HA Address" -d "Care-of Address"
--dport 500 -j ACCEPT
```

A.1.2 Binding Update

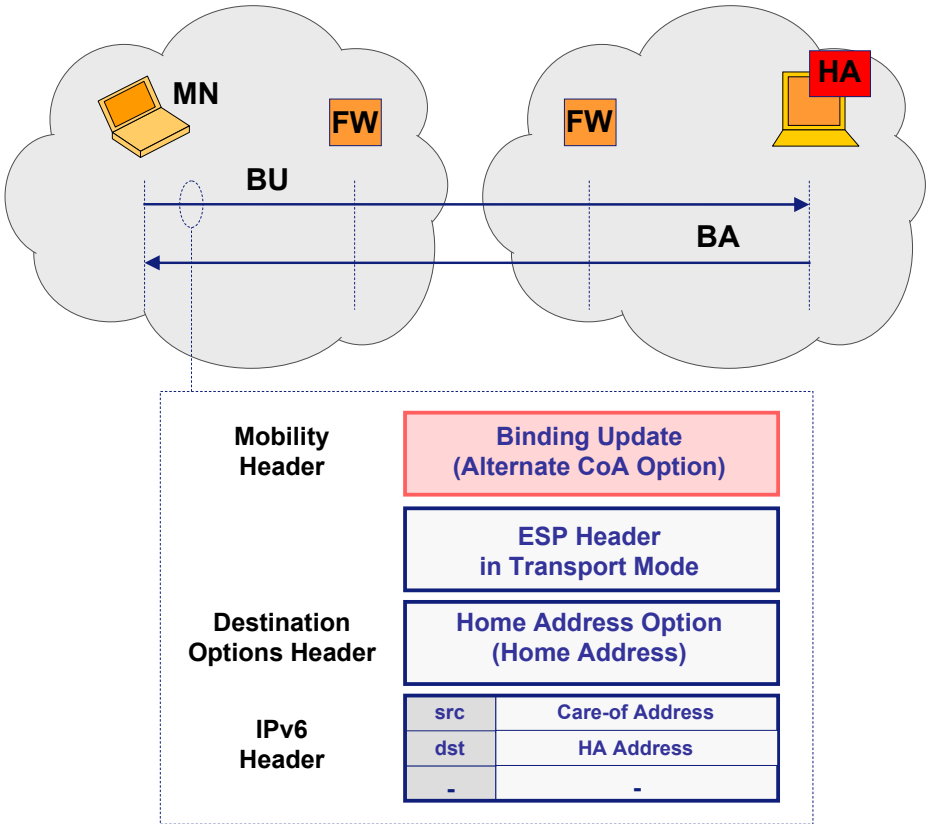


Figure A.2: Binding Update to HA

For the Binding Update message from the MN to the HA, the firewall pinhole has to match against:

- src address: Care-of Address
- dst address: HA Address
- ESP
- SPI

The iptables-call for installing this firewall pinhole is:

```
ip6tables -A FORWARD -p 50 -s "Care-of Address" -d "HA address"
-m esp --espspi X -j ACCEPT
```

A.1.3 Binding Acknowledgement

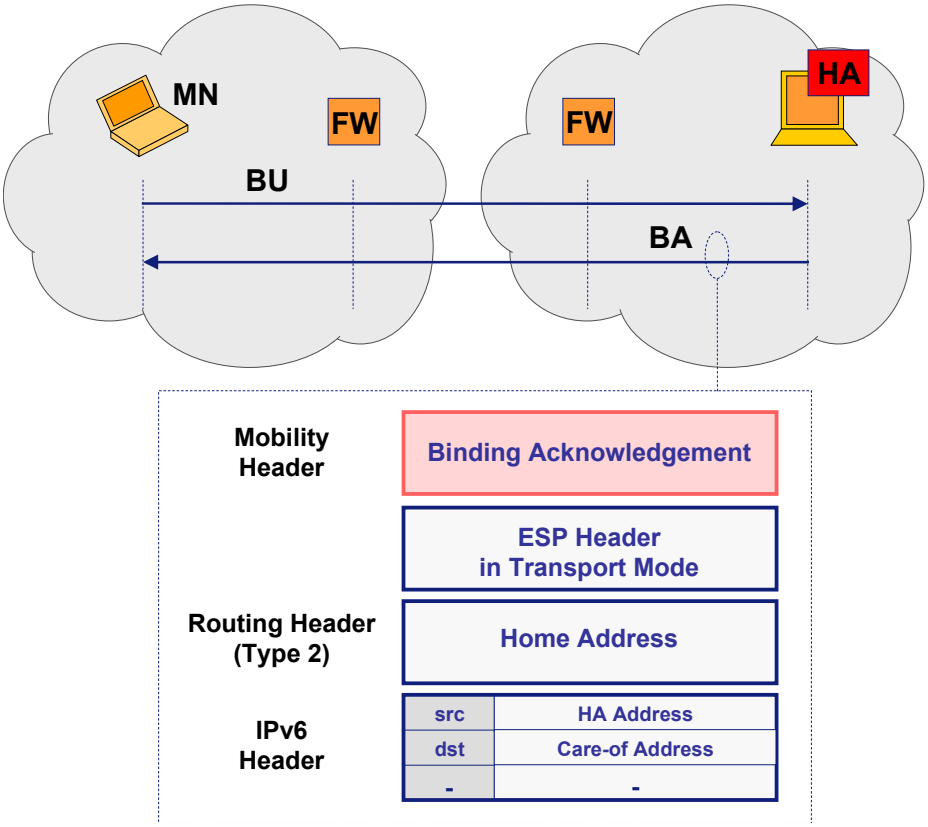


Figure A.3: Binding Acknowledgement from HA

For the Binding Acknowledgement message from the HA to the MN, the firewall pinhole has to match against:

- src address: HA Address
- dst address: Care-of Address
- ESP
- SPI

The iptables-call for installing this firewall pinhole is:

```
ip6tables -A FORWARD -p 50 -s "HA address" -d "Care-of Address"
-m esp --espspi X -j ACCEPT
```

A.1.4 Home Test Init

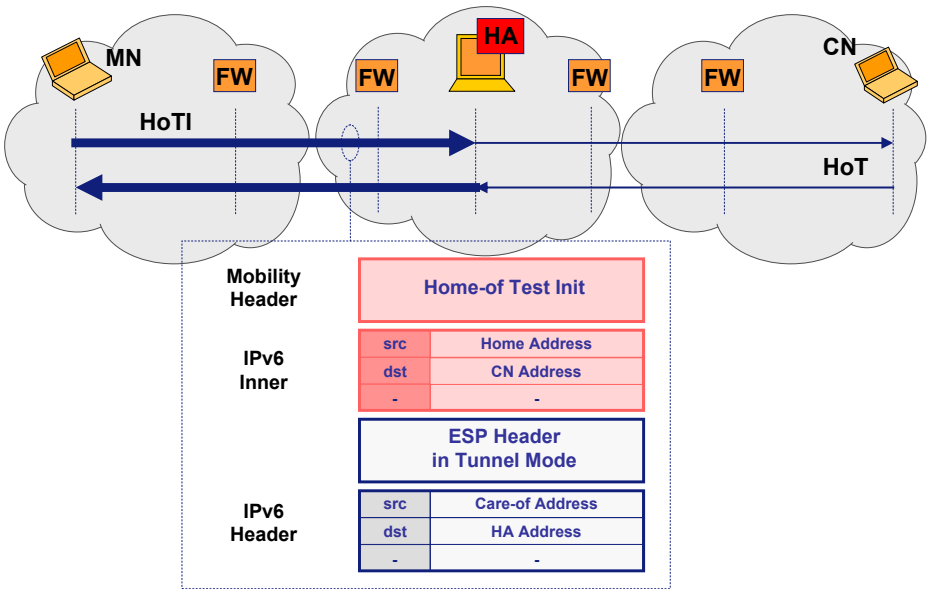


Figure A.4: Home Test Init (I)

For the Home Test Init message from the MN to the HA, the firewall pinhole has to match against:

- src address: Care-of Address
- dst address: HA Address
- ESP
- SPI

The iptables-call for installing this firewall pinhole is:

```
ip6tables -A FORWARD -p 50 -s "Care-of Address" -d "HA address"
-m esp --espspi X -j ACCEPT
```

If the SPI for the HoTI message is the same as for the Binding Update, the HoTI matches the previous installed firewall pinhole rule and no additional firewall pinhole would be required.

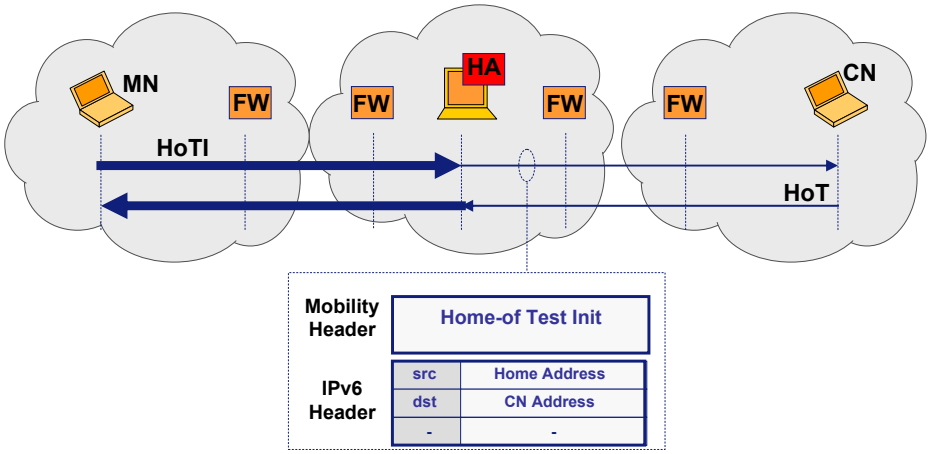


Figure A.5: Home Test Init (II)

For the Home Test Init message from the HA to the CN, the firewall pinhole has to match against:

- src address: Home Address
- dst address: CN Address
- Mobility Header Type 1

The iptables-call for installing this firewall pinhole is:

```
ip6tables -A FORWARD -s "Home Address" -d "CN Address"
-p mh --mh-type 1 -j ACCEPT
```

A.1.5 Home Test

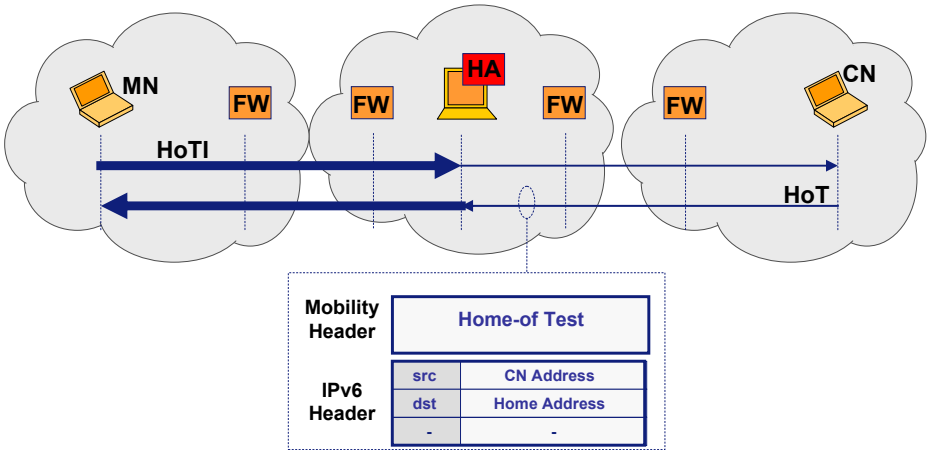


Figure A.6: Home Test (I)

For the Home Test message from the CN to the HA, the firewall pinhole has to match against:

- src address: CN Address
- dst address: Home Address
- Mobility Header Type 3

The iptables-call for installing this firewall pinhole is:

```
ip6tables -A FORWARD -s "CN Address" -d "Home Address"
-p mh --mh-type 3 -j ACCEPT
```

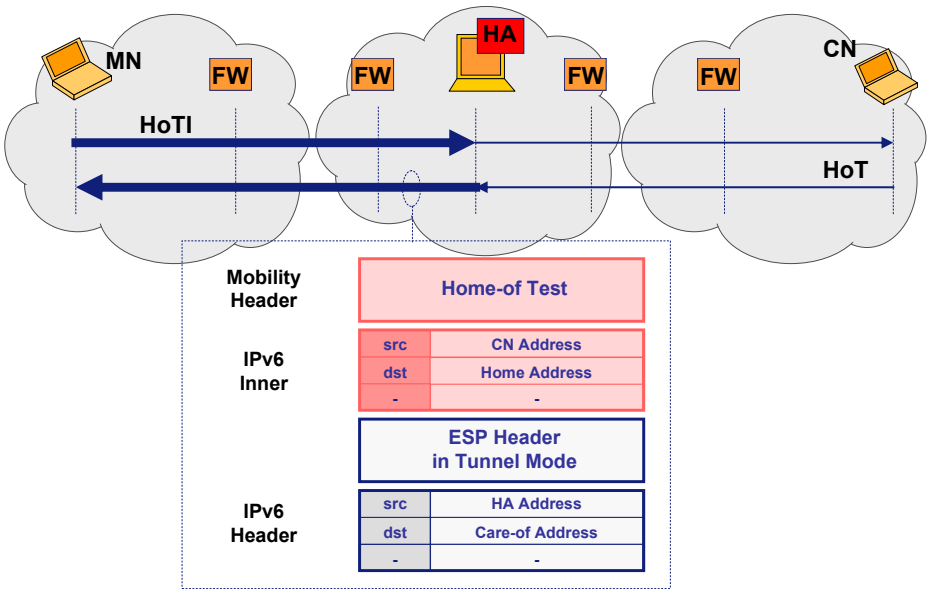


Figure A.7: Home Test (II)

For the Home Test message from the HA to the MN, the firewall pinhole has to match against:

- src address: HA Address
- dst address: Care-of Address
- ESP
- SPI

The iptables-call for installing this firewall pinhole is:

```
ip6tables -A FORWARD -p 50 -s "HA Address" -d "Care-of Address"
-m esp --espspi X -j ACCEPT
```

If the SPI for the HoT message is the same as for the Binding Acknowledgement, the HoT matches the previous installed firewall pinhole rule and no additional firewall pinhole would be required.

A.1.6 Care-of Test Init

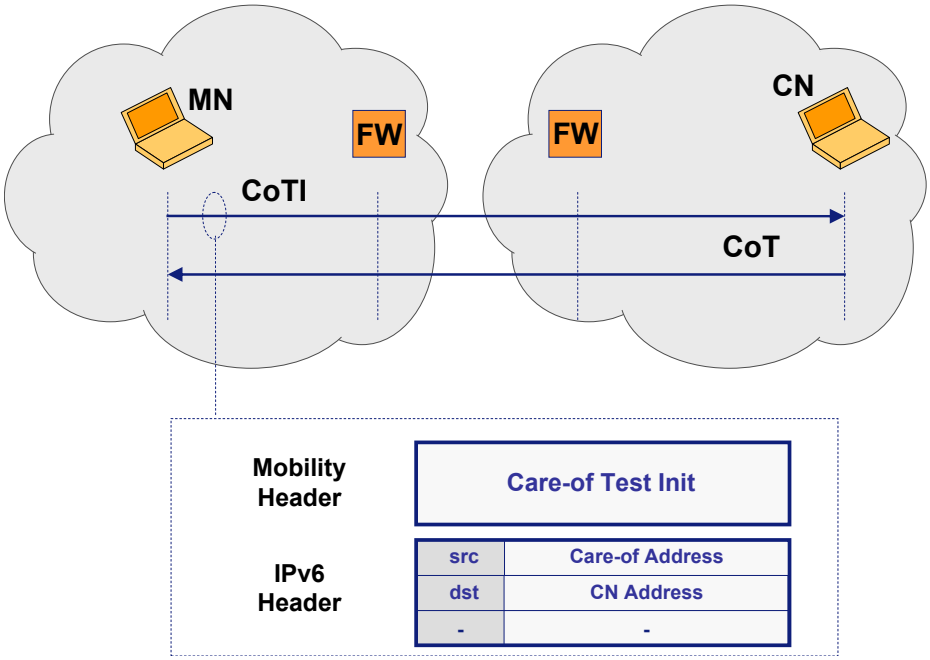


Figure A.8: Care-of Test Init

For the Care-of Test Init message from the MN to the CN, the firewall pinhole has to match against:

- src address: Care-of Address
- dst address: CN Address
- Mobility Header Type 2

The iptables-call for installing this firewall pinhole is:

```
ip6tables -A FORWARD -s "Care-of Address" -d "CN Address"
-p mh --mh-type 2 -j ACCEPT
```

A.1.7 Care-of Test

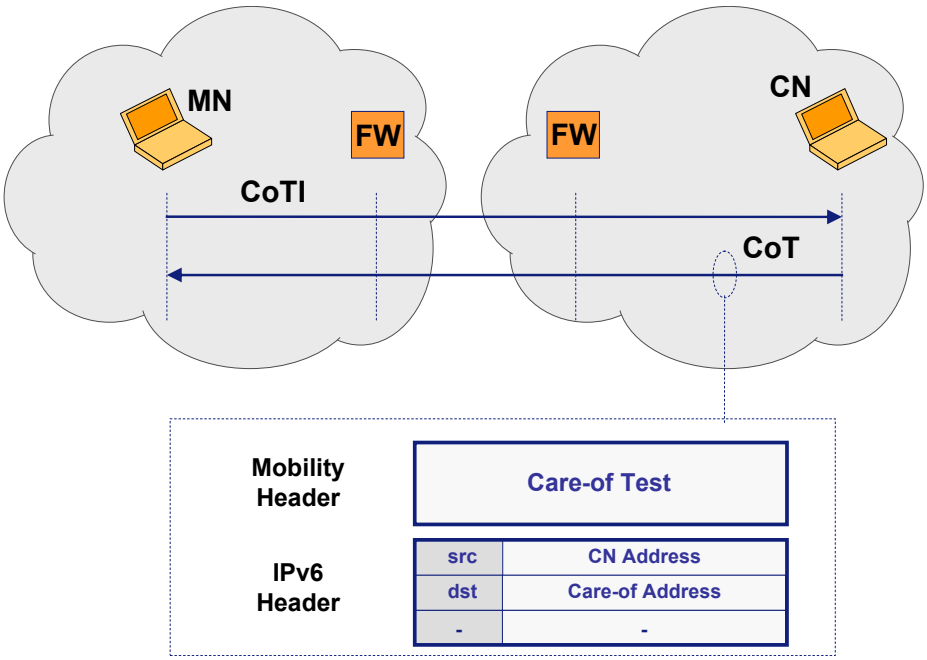


Figure A.9: Care-of Test

For the Care-of Test message from the CN to the MN, the firewall pinhole has to match against:

- src address: CN Address
- dst address: Care-of Address
- Mobility Header Type 4

The iptables-call for installing this firewall pinhole is:

```
ip6tables -A FORWARD -s "CN Address" -d "Care-of Address"
-p mh --mh-type 4 -j ACCEPT
```

A.1.8 Binding Update/Binding Acknowledgement CN

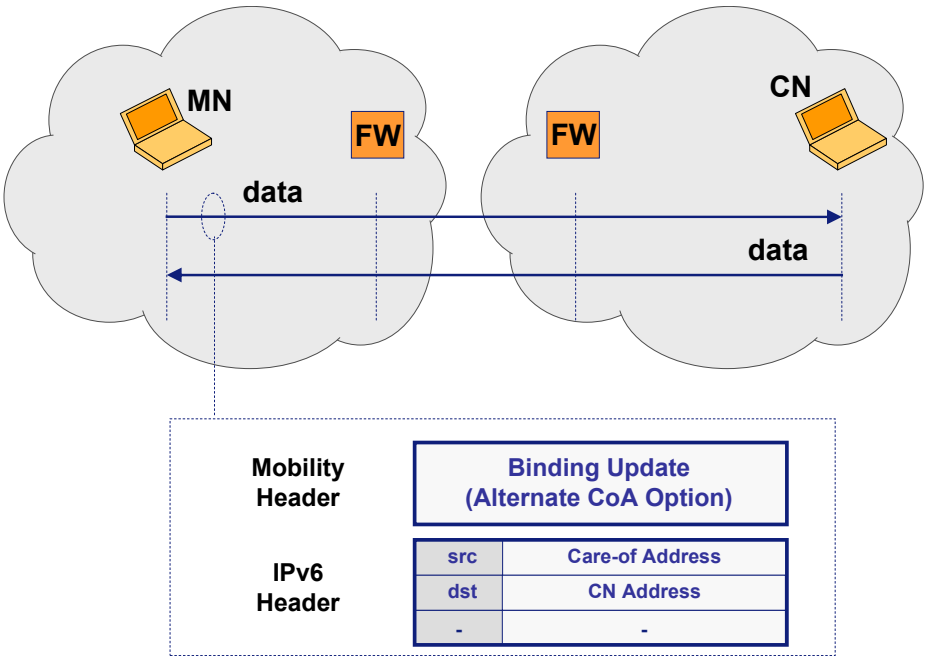


Figure A.10: Binding Update CN

For the Binding Update message from the MN to the CN, the firewall pinhole has to match against:

- src address: Care-of Address
- dst address: CN Address
- Mobility Header Type 5
- IPv6 Destination Options Header

The iptables-call for installing this firewall pinhole is:

```
ip6tables -A FORWARD -s "CN Address" -d "Care-of Address" -p mh
--mh-type 5 -m ipv6header --header dst -j ACCEPT
```

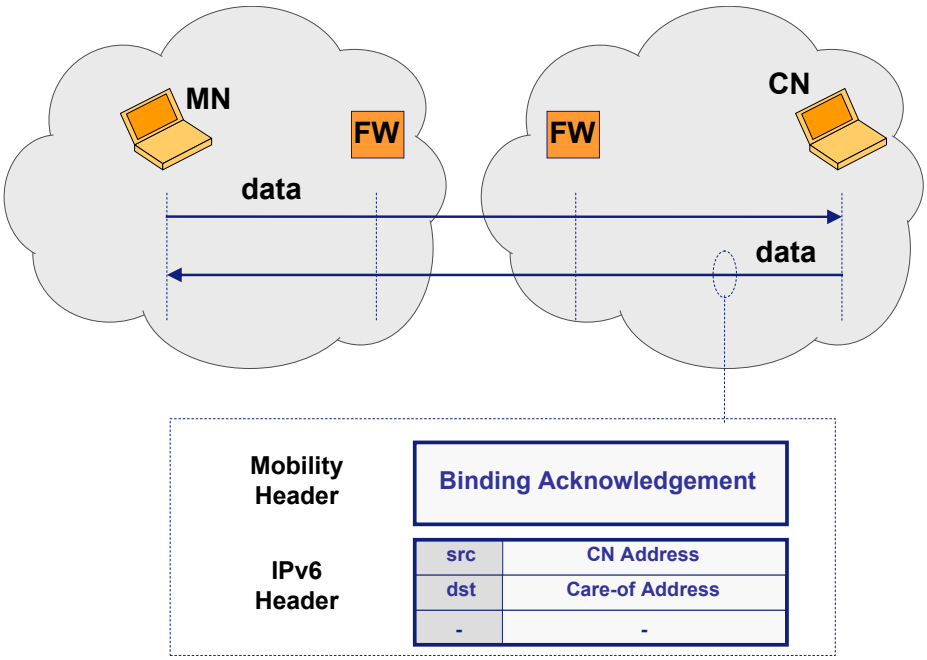


Figure A.11: Binding Acknowledgement CN

For the Binding Acknowledgement message from the CN to the MN, the firewall pinhole has to match against:

- src address: CN Address
- dst address: Care-of Address
- Mobility Header Type 6
- Routing Header for IPv6 Type 2

The iptables-call for installing this firewall pinhole is:

```
ip6tables -A FORWARD -s "CN Address" -d "Care-of Address" -p mh
--mh-type 6 -m rt --rt-type 2 -j ACCEPT
```

A.1.9 Route Optimisation Data Traffic

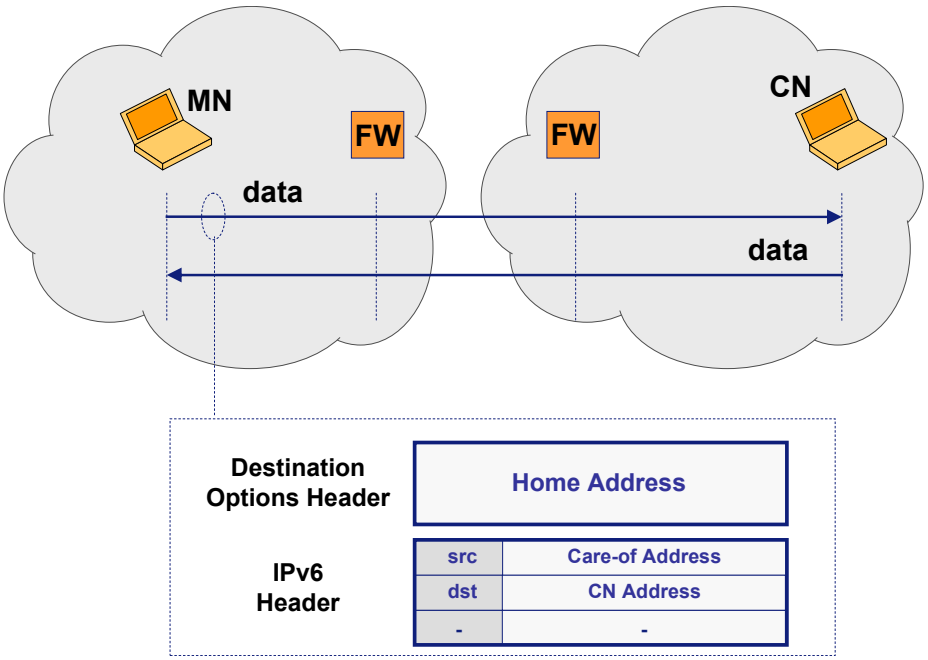


Figure A.12: Route Optimisation Data Traffic (I)

For the route optimisation data traffic from the MN to the CN, the firewall pinhole has to match against:

- src address: Care-of Address
- dst address: CN Address
- IPv6 Destination Options Header

The iptables-call for installing this firewall pinhole is:

```
ip6tables -A FORWARD -s "Care-of Address" -d "CN Address"
-m ipv6header --header dst -j ACCEPT
```

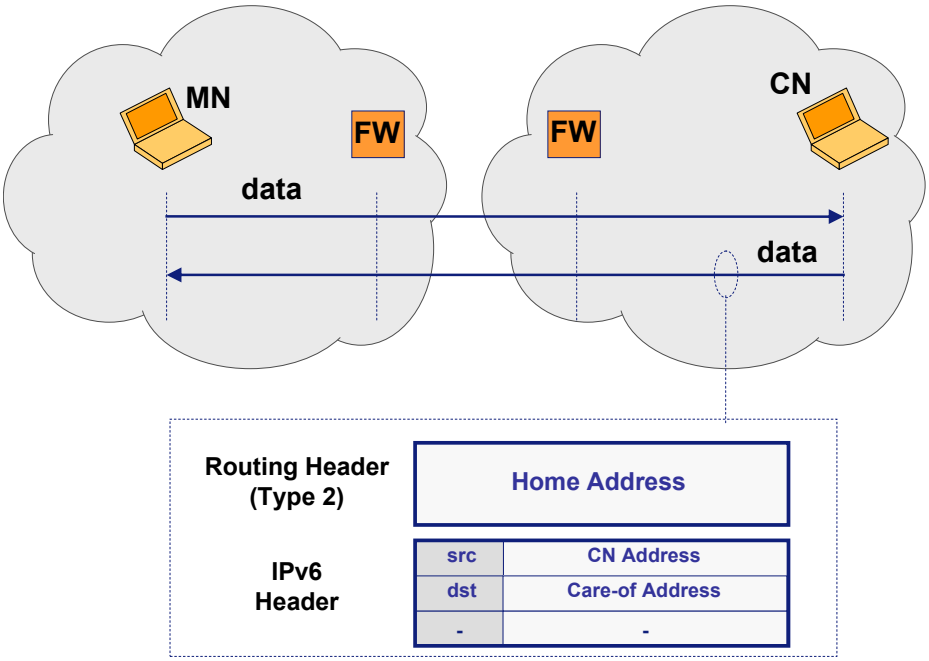


Figure A.13: Route Optimisation Data Traffic (II)

For the route optimisation data traffic from the CN to the MN, the firewall pinhole has to match against:

- src address: CN Address
- dst address: Care-of Address
- Routing Header for IPv6 Type 2

The iptables-call for installing this firewall pinhole is:

```
ip6tables -A FORWARD -s "CN Address" -d "Care-of Address"
-m rt --rt-type 2 -j ACCEPT
```

A.1.10 Bi-directional Tunneled Data Traffic

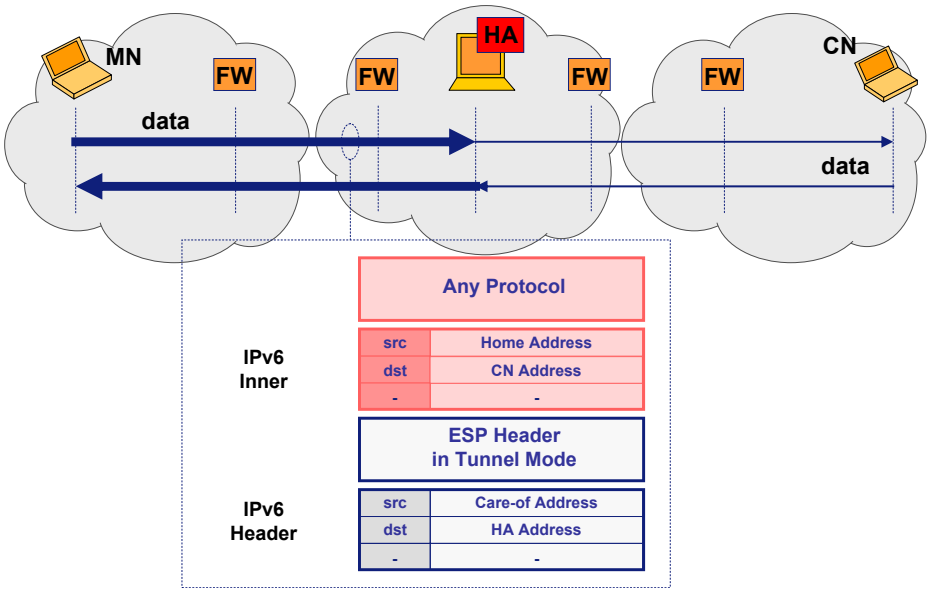


Figure A.14: Bi-directional Tunneled Data Traffic (I)

For the bi-directional tunneled data traffic from the MN to the HA, the firewall pinhole has to match against:

- src address: Care-of Address
- dst address: HA Address
- ESP
- SPI

As this matches the previous installed firewall pinhole rules as described in A.1.2 and A.1.4, no additional firewall pinhole rules are required.

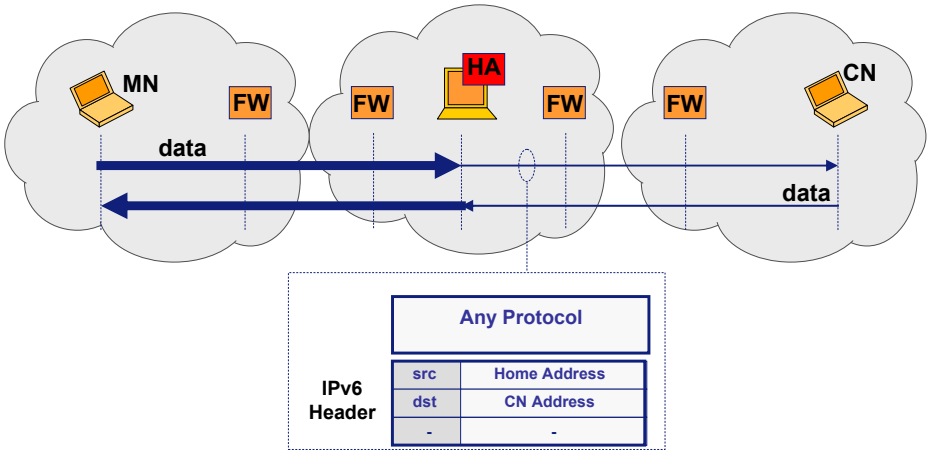


Figure A.15: Bi-directional Tunneled Data Traffic (II)

For the bi-directional tunneled data traffic from the HA to the CN, the firewall pinhole has to match against:

- src address: Home Address
- dst address: CN Address
- src port
- dst port

The iptables-call for installing this firewall pinhole is:

```
ip6tables -A FORWARD -s "Home Address" -d "CN Address"
--sport X --dport X -j ACCEPT
```

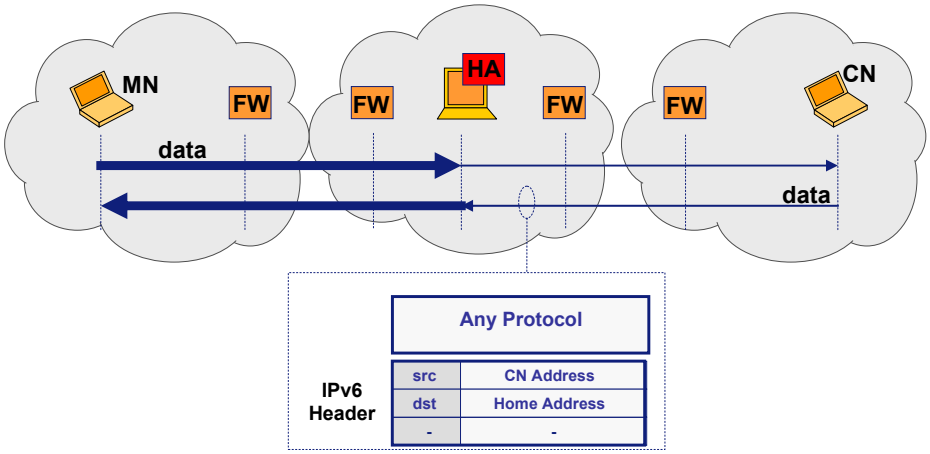


Figure A.16: Bi-directional Tunneled Data Traffic (III)

For the bi-directional tunneled data traffic from the CN to the HA, the firewall pinhole has to match against:

- src address: CN Address
- dst address: Home Address
- src port
- dst port

As this matches the previous installed pinhole rules, no additional firewall pinhole rules are required.

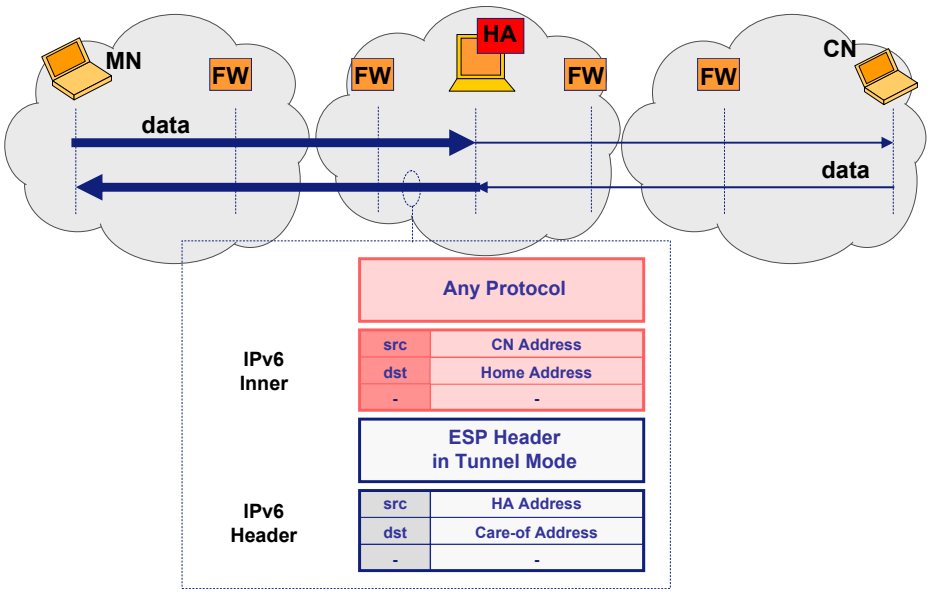


Figure A.17: Bi-directional Tunneled Data Traffic (IV)

For the bi-directional tunneled data traffic from the HA to the MN, the firewall pinhole has to match against:

- src address: HA Address
- dst address: Care-of Address
- ESP
- SPI

As this matches the previous installed firewall pinhole rules as described in A.1.3 and A.1.5, no additional firewall pinhole rules are required.

A.2 NAT/FW NSLP Packet Format Overview

This describes which NAT/FW NSLP packet must respectively can carry which NAT/FW NSLP objects.

- CREATE
  - Signaling Session Lifetime Object (Mandatory)
  - Extended Flow Information Object (Mandatory)
  - Message Sequence Number Object (Mandatory)
  - Nonce Object (Mandatory) if *P Flag* set to "1" in the NSLP Header, otherwise (Optional)
  - ICMP Types Object (Optional)
- EXTERNAL
  - Signaling Session Lifetime Object (Mandatory)
  - Extended Flow Information Object (Mandatory)
  - Message Sequence Number Object (Mandatory)
  - Data Terminal Information Object (Mandatory)
  - Nonce Object (Mandatory) if *P Flag* set to "1" in the NSLP Header, otherwise (Optional)
  - ICMP Types Object (Optional)
- RESPONSE

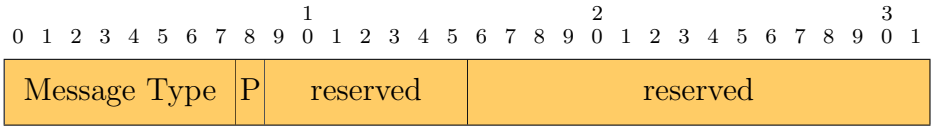
  The RESPONSE message for the class "Success" (0x2) carries these objects:
  - Signaling Session Lifetime Object (Mandatory)
  - Message Sequence Number Object (Mandatory)
  - Information Code Object (Mandatory)
  - External Address Object (Optional)

  The RESPONSE message for other classes than "Success" carries these objects:
  - Message Sequence Number Object (Mandatory)
  - Information Code Object (Mandatory)
- NOTIFY
  - Information Code Object (Mandatory)

A.3 NAT/FW NSLP Objects Overview

Defined NAT/FW NSLP objects are:

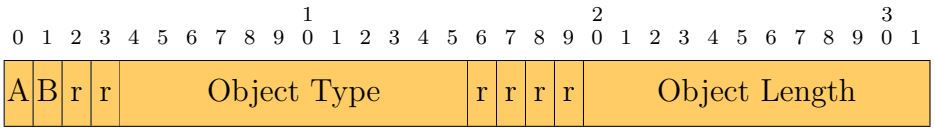
- Common NSLP Header, Length 1.



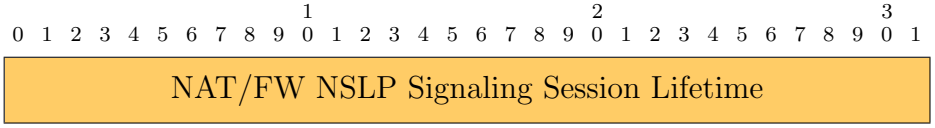
  NAT/FW NSLP Message Types:

  - **0x0101**: CREATE
  - **0x0102**: EXTERNAL
  - **0x0201**: RESPONSE
  - **0x0301**: NOTIFY

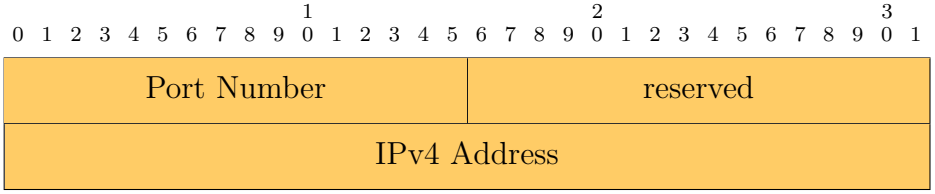
- Common NSLP Object Header, Length 1.



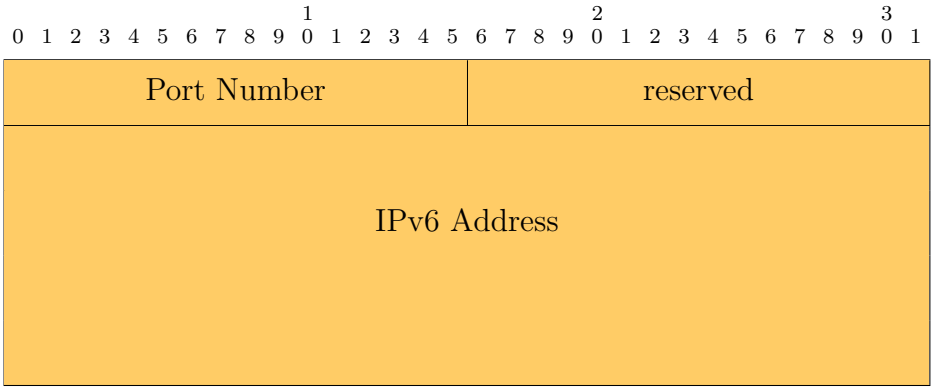
- Signaling Session Lifetime Object, Length 1.



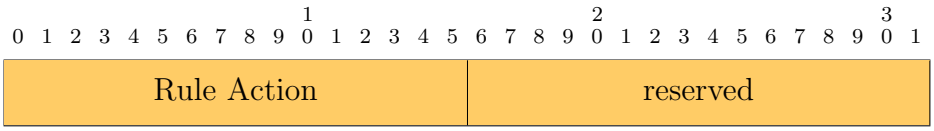
- External Address Object IPv4, Length 2.



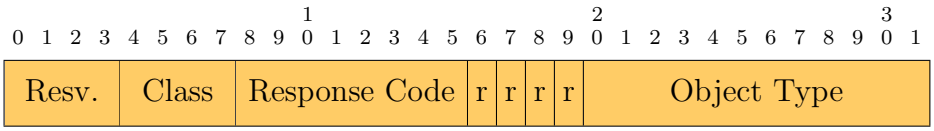
- External Address Object IPv6, Length 5.



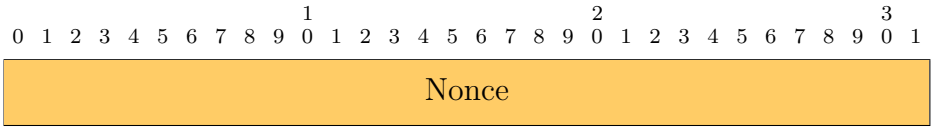
- Extended Flow Information Object, Length 1.



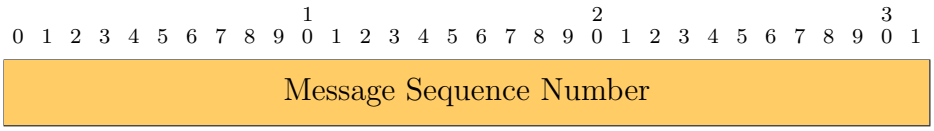
- Information Code Object, Length 1.



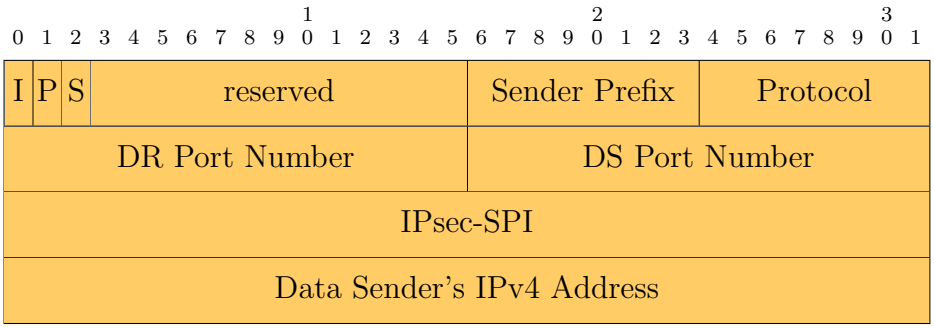
- Nonce Object, Length 1.



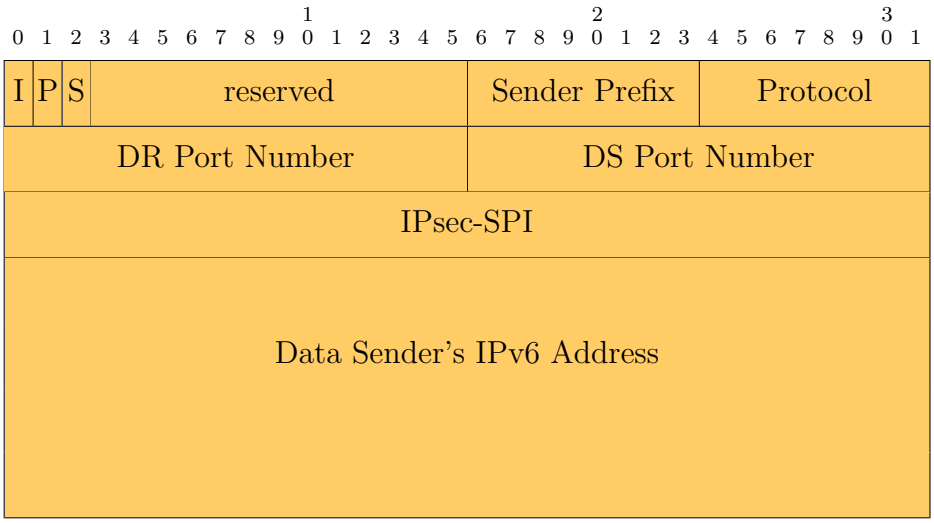
- Message Sequence Number Object, Length 1.



- Data Terminal Information Object IPv4, Length variable, Maximum 3.



- Data Terminal Information Object IPv6, Length variable, Maximum 6.



Acronyms

AAA	Authentication, Authorisation and Accounting
ALG	Application Layer Gateway
API	Application Programming Interface
ASA	Access Service Agent
ASP	Access Service Provider
ASP-FW	Access Service Provider Firewall
BA	Binding Acknowledgement
BA	Bootstrapping Agent
BAA	Bootstrapping Authorisation Agent
BCA	Bootstrapping Configuration Agent
BT	Bootstrapping Target
BU	Binding Update
CN	Correspondent Node
CoA	Care-of Address
COPS	Common Open Policy Service
CoT	Care-of Test
CoTI	Care-of Test Init
DoS	Denial of Service
EAP	Extensible Authentication Protocol
ESP	Encapsulation Security Protocol
EXT	External
DEA	Diameter-EAP-Answer
DNS	Domain Name System
FQDN	Fully Qualified Domain Name
FSM	Finite State Machine
FTP	File Transfer Protocol
FW	Firewall
GBA	Generic Bootstrapping Architecture
GIST	General Internet Signaling Transport
GSABA	Generic Service Authorisation and Bootstrapping Architecture

HA	Home Agent
HoA	Home Address
HoT	Home of Test
HoTI	Home of Test Init
HTTP	Hypertext Transfer Protocol
ICE	Interactive Connectivity Establishment
ICMP	Internet Control Message Protocol
IdP	Identity Provider
IETF	Internet Engineering Task Force
IKE	Internet Key Exchange
IP	Internet Protocol
IPF	IP-Filter
IPsec	Internet Protocol Security
IPv4	Internet Protocol Version 4
IPv6	Internet Protocol Version 6
ISP	Internet Service Provider
M-ICE	Mobile IP Interactive Connectivity Establishment
MH	Mobility Header
MIB	Management Information Base
MIDCOM	Middlebox Communication
MIPL	Mobile IPv6 for Linux
MIPv6	Mobile IPv6
MN	Mobile Node
MRI	Message Routing Information
MSA	Mobile Service Agent
MSP	Mobile Service Provider
MSP-FW	Mobile Service Provider Firewall
NAF	Network Application Function
NAI	Network Access Identifier
NAT	Network Address Translators
NAT-PT	Network Address Translators-Protocol Translation
NAT/FW NSLP	NAT/Firewall NSIS Signaling Layer Protocol
NE	NSIS Entity
NF	NSIS Forwarder
NI	NSIS Initiator
NR	NSIS Responder
NSIS	Next Steps in Signaling
NSLP	NSIS Signaling Layer Protocol
NTLP	NSIS Transport Layer Protocol

PBN	Policy-Based Networks
PDA	Personal Digital Assistant
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PI	Peer Information
PKI	Public Key Infrastructure
PGP	Pretty Good Privacy
RO	Route Optimisation
RRT	Return Routability Test
RSVP	Resource ReSerVation Protocol
RTT	Round Trip Time
SA	Security Association
SAML	Security Assertion Markup Language
SCTP	Stream Control Transmission Protocol
SDA	Signaling Destination Address
SDP	Session Description Protocol
SIMCO	Simple Middlebox Control
SIP	Session Initiation Protocol
SNMP	Simple Network Management Protocol
SOHO	Small Office Home Office
SP	Service Protocol
SPI	Security Parameter Index
SPF	Stateful Packet Filter
STUN	Simple Traversal of UDP Through Network Address Translators
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TLS/IA	TLS Inner Application
TLV	Type-Length-Value
TURN	Traversal Using Relay NAT
UE	User Equipment
UDP	User Datagram Protocol
UPnP	Universal Plug and Play
VoIP	Voice of IP
VPN	Virtual Private Network

List of Figures

List of Tables

Bibliography

[3GP08a] 3GPP. *Generic Authentication Architecture (GAA); Access to network application functions using Hypertext Transfer Protocol over Transport Layer Security (HTTPS).* TS 33.222, 3rd Generation Partnership Project (3GPP), June 2008. http://www.3gpp.org/ftp/Specs/html-info/33222.htm.

[3GP08b] 3GPP. *Generic Authentication Architecture (GAA); Generic bootstrapping architecture.* TS 33.220, 3rd Generation Partnership Project (3GPP), October 2008. http://www.3gpp.org/ftp/Specs/html-info/33220.htm.

[ABV+04] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, and H. Levkowetz. *Extensible Authentication Protocol (EAP).* RFC 3748, Internet Engineering Task Force, June 2004. http://www.rfc-editor.org/rfc/rfc3748.txt.

[ADD04] J. Arkko, V. Devarapalli, and F. Dupont. *Using IPsec to Protect Mobile IPv6 Signaling Between Mobile Nodes and Home Agents.* RFC 3776, Internet Engineering Task Force, June 2004. http://www.rfc-editor.org/rfc/rfc3776.txt.

[AJ07] F. Audet and C. Jennings. *Network Address Translation (NAT) Behavioral Requirements for Unicast UDP.* RFC 4787, Internet Engineering Task Force, January 2007. http://www.rfc-editor.org/rfc/rfc4787.txt.

[Baj08] G. Bajko. *Firewall friendly Return-Routability Test (RRT) for Mobile IPv6.* Internet-Draft draft-bajko-mip6-rrtfw-03, Internet Engineering Task Force, Work in progress, February 2008. http://www.ietf.org/internet-drafts/draft-bajko-mip6-rrtfw-03.txt.

[BEHAVE] *The BEHAVE Working Group*, Internet Engineering Task Force. Website. http://www.ietf.org/html.charters/behave-charter.html.

[BZB+97] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. *Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification.* RFC 2205, Internet Engineering Task Force, September 1997. http://www.rfc-editor.org/rfc/rfc2205.txt.

[CB94] W. Cheswick and S. Bellovin. *Firewalls and Internet Security: Repelling the Wily Hacker*. Addison-Wesley, 1994. ISBN 0-201-63357-4.

[CB02] B. Carpenter and S. Brim. *Middleboxes: Taxonomy and Issues*. RFC 3234, Internet Engineering Task Force, February 2002. http://www.rfc-editor.org/rfc/rfc3234.txt.

[CD98] A. Conta and S. Deering. *Generic Packet Tunneling in IPv6 Specification*. RFC 2473, Internet Engineering Task Force, December 1998. http://www.rfc-editor.org/rfc/rfc2473.txt.

[CDFT98] J. Callas, L. Donnerhacke, H. Finney, and R. Thayer. *OpenPGP Message Format*. RFC 2440, Internet Engineering Task Force, November 1998. http://www.rfc-editor.org/rfc/rfc2440.txt.

[CLG+03] P. Calhoun, J. Loughney, E. Guttman, G. Zorn, and J. Arkko. *Diameter Base Protocol*. RFC 3588, Internet Engineering Task Force, September 2003. http://www.rfc-editor.org/rfc/rfc3588.txt.

[CSF+08] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. RFC 5280, Internet Engineering Task Force, May 2008. http://www.rfc-editor.org/rfc/rfc5280.txt.

[DA99] T. Dierks and C. Allen. *The TLS Protocol Version 1.0*. RFC 2246, Internet Engineering Task Force, January 1999. http://www.rfc-editor.org/rfc/rfc2246.txt.

[DBC+00] D. Durham, J. Boyle, R. Cohen, S. Herzog, R. Rajan, and A. Sastry. *The COPS (Common Open Policy Service) Protocol*. RFC 2748, Internet Engineering Task Force, January 2000. http://www.rfc-editor.org/rfc/rfc2748.txt.

[DFT+06] C. Dickmann, X. Fu, H. Tschofenig, H. Schulzrinne, and D. Hogrefe. *Overhead and Performance Study of the General Internet Signaling Transport (GIST) Protocol*. In *INFOCOM 2006* (April 2006), IEEE.

[DH98] S. Deering and R. Hinden. *Internet Protocol, Version 6 (IPv6) Specification*. RFC 2460, Internet Engineering Task Force, December 1998. http://www.rfc-editor.org/rfc/rfc2460.txt.

[EB97] R. Elz and R. Bush. *Clarifications to the DNS Specification*. RFC 2181, Internet Engineering Task Force, July 1997. http://www.rfc-editor.org/rfc/rfc2181.txt.

[EF94] K. Egevang and P. Francis. *The IP Network Address Translator (NAT).* RFC 1631, Internet Engineering Task Force, May 1994. http://www.rfc-editor.org/rfc/rfc1631.txt.

[EHZ05] P. Eronen, T. Hiller, and G. Zorn. *Diameter Extensible Authentication Protocol (EAP) Application.* RFC 4072, Internet Engineering Task Force, August 2005. http://www.rfc-editor.org/rfc/rfc4072.txt.

[ENABLE] *Enabling Efficent and Operational Mobility in Large Heterogeneous IP Networks.* Website. http://www.ist-enable.org/.

[ET05] P. Eronen and H. Tschofenig. *Pre-Shared Key Ciphersuites for Transport Layer Security (TLS).* RFC 4279, Internet Engineering Task Force, December 2005. http://www.rfc-editor.org/rfc/rfc4279.txt.

[EUFP6] *The European Union (EU) Framework Programme 6.* Website. http://cordis.europa.eu/fp6.

[FBWS+06] P. Funk, S. Blake-Wilson, N. Smith, H. Tschofenig, and T. Hardjono. *TLS Inner Application Extension (TLS/IA).* Internet-Draft draft-funk-tls-inner-application-extension-03, Internet Engineering Task Force, Work in progress, June 2006. http://www.ietf.org/internet-drafts/draft-funk-tls-inner-application-extension-03.txt.

[FDC08] X. Fu, C. Dickmann, and J. Crowcroft. *General Internet Signaling Transport (GIST) over SCTP.* Internet-Draft draft-ietf-nsis-ntlp-sctp-05, Internet Engineering Task Force, Work in progress, October 2008. http://www.ietf.org/internet-drafts/draft-ietf-nsis-ntlp-sctp-05.txt.

[FGM+99] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. *Hypertext Transfer Protocol – HTTP/1.1.* RFC 2616, Internet Engineering Task Force, June 1999. http://www.rfc-editor.org/rfc/rfc2616.txt.

[FreeNSIS] *The Free Next Steps In Signaling (FreeNSIS) Implementation by University of Göttingen.* Website. http://user.informatik.uni-goettingen.de/∼nsis/.

[GBF+08] S. Guha, K. Biswas, B. Ford, S. Sivakumar, and P. Srisuresh. *NAT Behavioral Requirements for TCP.* RFC 5382, Internet Engineering Task Force, October 2008. http://www.rfc-editor.org/rfc/rfc5382.txt.

[GF07] Saikat Guha and Paul Francis. *An end-middle-end approach to connection establishment.* In *SIGCOMM '07: Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications* (New York, NY, USA, 2007), ACM, pp. 193–204. ISBN 978-1-59593-713-1.

[GHJP00] S. Glass, T. Hiller, S. Jacobs, and C. Perkins. *Mobile IP Authentication, Authorization, and Accounting Requirements.* RFC 2977, Internet Engineering Task Force, October 2000. http://www.rfc-editor.org/rfc/rfc2977.txt.

[GLD+08] S. Gundavelli, K. Leung, V. Devarapalli, K. Chowdhury, and B. Patil. *Proxy Mobile IPv6.* RFC 5213, Internet Engineering Task Force, August 2008. http://www.rfc-editor.org/rfc/rfc5213.txt.

[GLH+00] B. Gleeson, A. Lin, J. Heinanen, G. Armitage, and A. Malis. *A Framework for IP Based Virtual Private Networks.* RFC 2764, Internet Engineering Task Force, February 2000. http://www.rfc-editor.org/rfc/rfc2764.txt.

[Gnup] *Gnuplot.* Website. http://www.gnuplot.info/.

[HC98] D. Harkins and D. Carrel. *The Internet Key Exchange (IKE).* RFC 2409, Internet Engineering Task Force, November 1998. http://www.rfc-editor.org/rfc/rfc2409.txt.

[HiPAC] *nf-HiPAC: High Performance Firewall for Linux Netfilter.* Website. http://www.hipac.org/.

[HJP06] M. Handley, V. Jacobson, and C. Perkins. *SDP: Session Description Protocol.* RFC 4566, Internet Engineering Task Force, July 2006. http://www.rfc-editor.org/rfc/rfc4566.txt.

[HKLdB05] R. Hancock, G. Karagiannis, J. Loughney, and S. Van den Bosch. *Next Steps in Signaling (NSIS): Framework.* RFC 4080, Internet Engineering Task Force, June 2005. http://www.rfc-editor.org/rfc/rfc4080.txt.

[HPS03] D. Hoffmann, D. Prabhakar, and P. Strooper. *Testing iptables.* In *Proceedings of the 2003 Conference of the Centre for Advandced Studies on Collaborative Research* (Toronto, Ontario, Canada, 2003), IBM Press, pp. 80–91.

[IETF] *The Internet Engineering Task Force.* Website. http://www.ietf.org/.

[IPset] *IP sets*. Website. http://www.ipset.netfilter.org/.

[IPv6] *The IP Version 6 Working Group*, Internet Engineering Task Force. Website. http://www.ietf.org/html.charters/OLD/ipv6-charter.html.

[JPA04] D. Johnson, C. Perkins, and J. Arkko. *Mobility Support in IPv6*. RFC 3775, Internet Engineering Task Force, June 2004. http://www.rfc-editor.org/rfc/rfc3775.txt.

[JTS08] Dilip Antony Joseph, Arsalan Tavakoli, and Ion Stoica. *A policy-aware switching layer for data centers*. In *SIGCOMM* (Seattle, WA, USA, 2008), ACM, pp. 51–62. ISBN 978-1-60558-175-0.

[Kau05] C. Kaufman. *Internet Key Exchange (IKEv2) Protocol*. RFC 4306, Internet Engineering Task Force, December 2005. http://www.rfc-editor.org/rfc/rfc4306.txt.

[Ken05] S. Kent. *IP Encapsulating Security Payload (ESP)*. RFC 4303, Internet Engineering Task Force, December 2005. http://www.rfc-editor.org/rfc/rfc4303.txt.

[KG04] J. Kadlecsik and G.Pasztor. *Netfilter Performance Testing*.

[Kit01] H. Kitamura. *A SOCKS-based IPv6/IPv4 Gateway Mechanism*. RFC 3089, Internet Engineering Task Force, April 2001. http://www.rfc-editor.org/rfc/rfc3089.txt.

[Koo08] R. Koodli. *Mobile IPv6 Fast Handovers*. RFC 5268, Internet Engineering Task Force, June 2008. http://www.rfc-editor.org/rfc/rfc5268.txt.

[KS05] S. Kent and K. Seo. *Security Architecture for the Internet Protocol*. RFC 4301, Internet Engineering Task Force, December 2005. http://www.rfc-editor.org/rfc/rfc4301.txt.

[KSSB08] S. Krishnan, Y. Sheffer, N. Steinleitner, and G. Bajko. *Guidelines for firewall vendors regarding MIPv6 traffic*. Internet-Draft draft-ietf-mext-firewall-vendor-00, Internet Engineering Task Force, Work in progress, October 2008. http://www.ietf.org/internet-drafts/draft-ietf-mext-firewall-vendor-00.txt.

[KSYB08] S. Krishnan, N. Steinleitner, Q. Ying, and G. Bajko. *Guidelines for firewall administrators regarding MIPv6 traffic*. Internet-Draft draft-ietf-mext-firewall-admin-00, Internet Engineering Task Force, Work in progress, October 2008. http://www.ietf.org/internet-drafts/draft-ietf-mext-firewall-admin-00.txt% fi.

[KTF+06] F. Kohlmayer, H. Tschofenig, R. Falk, R. Marin Lopez, S. Zapata Hernandez, P. GarcíaSegura, and A. F. Gómez Skarmeta. *GSABA: A Generic Service Authorization Architecture.* In *MobiArch '06: Proceedings of first ACM/IEEE International Workshop on Mobility in the evolving Internet Architecture* (New York, NY, USA, 2006), ACM, pp. 51–56. ISBN 1-59593-566-5.

[LC04] J. Loughney and G. Camarillo. *Authentication, Authorization, and Accounting Requirements for the Session Initiation Protocol (SIP).* RFC 3702, Internet Engineering Task Force, February 2004. http://www.rfc-editor.org/rfc/rfc3702.txt.

[LFPT06] F. Le, S. Faccin, B. Patil, and H. Tschofenig. *Mobile IPv6 and Firewalls: Problem Statement.* RFC 4487, Internet Engineering Task Force, May 2006. http://www.rfc-editor.org/rfc/rfc4487.txt.

[LGL+96] M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas, and L. Jones. *SOCKS Protocol Version 5.* RFC 1928, Internet Engineering Task Force, March 1996. http://www.rfc-editor.org/rfc/rfc1928.txt.

[LMK04] J. Lee, J. Min, and S. Kim. *Considerations for designing fast Handoff Mechanisms in Mobile IPv6.* In *The 6th International Conference on Advanced Communication Technology* (2004), vol. 1, pp. 21–24. ISBN 89-5519-119-7.

[LMS02] D. Levi, P. Meyer, and B. Stewart. *Simple Network Management Protocol (SNMP) Applications.* RFC 3413, Internet Engineering Task Force, December 2002. http://www.rfc-editor.org/rfc/rfc3413.txt.

[LSJP06] J. Lai, Y. A. Sekercioglu, N. Jordan, and A. Pitsillides. *Performance Evaluation of Mobile IPv6 Handover Extensions in an IEEE 802.11b Wireless Network Environment.* In *ISCC '06: Proceedings of the 11th IEEE Symposium on Computers and Communications* (Washington, DC, USA, 2006), IEEE Computer Society, pp. 161–166. ISBN 0-7695-2588-1.

[MEXT] *The MEXT Working Group*, Internet Engineering Task Force. Website. http://www.ietf.org/html.charters/mext-charter.html.

[MH99] A. Medvinsky and M. Hur. *Addition of Kerberos Cipher Suites to Transport Layer Security (TLS).* RFC 2712, Internet Engineering Task Force, October 1999. http://www.rfc-editor.org/rfc/rfc2712.txt.

[Micro] *MICROSOFT.* Website. http://www.microsoft.com.

[MIDCOM] *The MIDCOM Working Group*, Internet Engineering Task Force. Website. http://www.ietf.org/html.charters/midcom-charter.html.

[MIP6] *Mobility for IPv6 Working Group*, Internet Engineering Task Force. Website. http://www.ietf.org/html.charters/OLD/mip6-charter.html.

[MIP6ALG] *Mobile IPv6 Application Layer Gateway - Prototype Implementation.* Website. http://user.informatik.uni-goettingen.de/~nsteinle/software/mip6alg/.

[MIPL] *MIPL - Mobile IPv6 for Linux.* Website. http://www.mobile-ipv6.org.

[MKM08] J. Manner, G. Karagiannis, and A. McDonald. *NSLP for Quality-of-Service Signaling.* Internet-Draft draft-ietf-nsis-qos-nslp-16, Internet Engineering Task Force, Work in progress, February 2008. http://www.ietf.org/internet-drafts/draft-ietf-nsis-qos-nslp-16.txt.

[MMUSIC] *The MMUSIC Working Group*, Internet Engineering Task Force. Website. http://www.ietf.org/html.charters/mmusic-charter.html.

[Moc87a] P.V. Mockapetris. *Domain names - concepts and facilities.* RFC 1034, Internet Engineering Task Force, November 1987. http://www.rfc-editor.org/rfc/rfc1034.txt.

[Moc87b] P.V. Mockapetris. *Domain names - implementation and specification.* RFC 1035, Internet Engineering Task Force, November 1987. http://www.rfc-editor.org/rfc/rfc1035.txt.

[Mog89] J. C. Mogul. *Simple and flexible datagram access controls for Unix-based Gateways.* Website, 1989. ftp://ftp.digital.com/pub/Digital/WRL/research-reports/WRL-TR-89.4.pdf.

[MSTB08] J. Manner, M. Stiemerling, H. Tschofenig, and R. Bless. *Authorization for NSIS Signaling Layer Protocols.* Internet-Draft draft-manner-nsis-nslp-auth-04, Internet Engineering Task Force, Work in progress, July 2008. http://www.ietf.org/internet-drafts/draft-manner-nsis-nslp-auth-04.txt.

[netfilter] *Netfilter, Firewalling, NAT, and Packet Mangling for Linux.* Website. http://www.netfilter.org/.

[NSIS] *The NSIS Working Group*, Internet Engineering Task Force. Website. http://www.ietf.org/html.charters/nsis-charter.html.

[NSISMIP6] *NSIS Based Mobile IPv6 Firewall Traversal - Prototype Implementation by University of Göttingen*. Website. http://user.informatik.uni-goettingen.de/~nsteinle/software/nsismip6fwtraversal/.

[Per96] C. Perkins. *IP Mobility Support*. RFC 2002, Internet Engineering Task Force, October 1996. http://www.rfc-editor.org/rfc/rfc2002.txt.

[PLK+05] A. Patel, K. Leung, M. Khalil, H. Akhtar, and K. Chowdhury. *Mobile Node Identifier Option for Mobile IPv6 (MIPv6)*. RFC 4283, Internet Engineering Task Force, November 2005. http://www.rfc-editor.org/rfc/rfc4283.txt.

[PLK+06] A. Patel, K. Leung, M. Khalil, H. Akhtar, and K. Chowdhury. *Authentication Protocol for Mobile IPv6*. RFC 4285, Internet Engineering Task Force, January 2006. http://www.rfc-editor.org/rfc/rfc4285.txt.

[Pos80] J. Postel. *User Datagram Protocol*. RFC 0768, Internet Engineering Task Force, August 1980. http://www.rfc-editor.org/rfc/rfc768.txt.

[Pos81a] J. Postel. *Internet Control Message Protocol*. RFC 0792, Internet Engineering Task Force, September 1981. http://www.rfc-editor.org/rfc/rfc792.txt.

[Pos81b] J. Postel. *Internet Protocol*. RFC 0791, Internet Engineering Task Force, September 1981. http://www.rfc-editor.org/rfc/rfc791.txt.

[Pos81c] J. Postel. *Transmission Control Protocol*. RFC 0793, Internet Engineering Task Force, September 1981. http://www.rfc-editor.org/rfc/rfc793.txt.

[PPST06] J. Peterson, J. Polk, D. Sicker, and H. Tschofenig. *Trait-Based Authorization Requirements for the Session Initiation Protocol (SIP)*. RFC 4484, Internet Engineering Task Force, August 2006. http://www.rfc-editor.org/rfc/rfc4484.txt.

[PR85] J. Postel and J. Reynolds. *File Transfer Protocol*. RFC 0959, Internet Engineering Task Force, October 1985. http://www.rfc-editor.org/rfc/rfc959.txt.

[Pre02] R. Presuhn. *Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP)*. RFC 3416, Internet Engineering Task Force, December 2002. http://www.rfc-editor.org/rfc/rfc3416.txt.

[QSS08] J. Quittek, M. Stiemerling, and P. Srisuresh. *Definitions of Managed Objects for Middlebox Communication.* RFC 5190, Internet Engineering Task Force, March 2008. http://www.rfc-editor.org/rfc/rfc5190.txt.

[Ree96] D. Reed. *IP-Filter (IPF) 3.0.* Website, 1996. http://coombs.anu.edu.au/∼avalon/.

[RM06] E. Rescorla and N. Modadugu. *Datagram Transport Layer Security.* RFC 4347, Internet Engineering Task Force, April 2006. http://www.rfc-editor.org/rfc/rfc4347.txt.

[RMM08] J. Rosenberg, R. Mahy, and P. Matthews. *Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN).* Internet-Draft draft-ietf-behave-turn-11, Internet Engineering Task Force, Work in progress, October 2008. http://www.ietf.org/internet-drafts/draft-ietf-behave-turn-11.txt.

[RMMW08] J. Rosenberg, R. Mahy, P. Matthews, and D. Wing. *Session Traversal Utilities for NAT (STUN).* RFC 5389, Internet Engineering Task Force, October 2008. http://www.rfc-editor.org/rfc/rfc5389.txt.

[Roe06] U. Roedig. *Firewall-Architekturen für Multimedia-Applikationen.* PhD thesis, University of Darmstadt, 2006.

[Ros07] J. Rosenberg. *Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols.* Internet-Draft draft-ietf-mmusic-ice-19, Internet Engineering Task Force, Work in progress, October 2007. http://www.ietf.org/internet-drafts/draft-ietf-mmusic-ice-19.txt.

[Ros08] J. Rosenberg. *TCP Candidates with Interactive Connectivity Establishment (ICE).* Internet-Draft draft-ietf-mmusic-ice-tcp-07, Internet Engineering Task Force, Work in progress, July 2008. http://www.ietf.org/internet-drafts/draft-ietf-mmusic-ice-tcp-07.txt.

[RS02] J. Rosenberg and H. Schulzrinne. *An Offer/Answer Model with Session Description Protocol (SDP).* RFC 3264, Internet Engineering Task Force, June 2002. http://www.rfc-editor.org/rfc/rfc3264.txt.

[RSC+02] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. *SIP: Session Initiation Protocol.* RFC 3261, Internet Engineering Task Force, June 2002. http://www.rfc-editor.org/rfc/rfc3261.txt.

[RWC00] C. Rigney, W. Willats, and P. Calhoun. *RADIUS Extensions*. RFC 2869, Internet Engineering Task Force, June 2000. http://www.rfc-editor.org/rfc/rfc2869.txt.

[RWHM03] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy. *STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)*. RFC 3489, Internet Engineering Task Force, March 2003. http://www.rfc-editor.org/rfc/rfc3489.txt.

[RWRS00] C. Rigney, S. Willens, A. Rubens, and W. Simpson. *Remote Authentication Dial In User Service (RADIUS)*. RFC 2865, Internet Engineering Task Force, June 2000. http://www.rfc-editor.org/rfc/rfc2865.txt.

[SCMB05] H. Soliman, C. Castelluccia, K. El Malki, and L. Bellier. *Hierarchical Mobile IPv6 Mobility Management (HMIPv6)*. RFC 4140, Internet Engineering Task Force, August 2005. http://www.rfc-editor.org/rfc/rfc4140.txt.

[sfFW96] *sf Firewall*. Website, 1996. http://www.ifi.unizh.ch/ikm/SINUS/sf-doc/impl.htm.

[SFH+07] N. Steinleitner, X. Fu, D. Hogrefe, T. Schreck, and H. Tschofenig. *An NSIS-based Approach for Firewall Traversal in Mobile IPv6 Networks*. In *The Third Annual International Wireless Internet Conference (WICON 2007)* (Austin, Texas, USA, October 2007), ACM Press.

[SFJ+08] T. Sanda, X. Fu, S. Jeong, J. Manner, and H. Tschofenig. *Applicability Statement of NSIS Protocols in Mobile Environments*. Internet-Draft draft-ietf-nsis-applicability-mobility-signaling-11, Internet Engineering Task Force, Work in progress, November 2008. http://www.ietf.org/internet-drafts/draft-ietf-nsis-applicability-mobility-signaling-11.txt.

[SFL07] N. Steinleitner, X. Fu, and F. Le. *Mobile IPv6 - NSIS Interaction for Firewall traversal*. Internet-Draft draft-thiruvengadam-nsis-mip6-fw-08, Internet Engineering Task Force, Work in progress, November 2007. http://www.ietf.org/internet-drafts/draft-thiruvengadam-nsis-mip6-fw-08.txt.

[SFSG08] P. Srisuresh, B. Ford, S. Sivakumar, and S. Guha. *NAT Behavioral Requirements for ICMP protocol*. Internet-Draft draft-ietf-behave-nat-icmp-11, Internet Engineering Task Force, Work in progress,

November 2008. http://www.ietf.org/internet-drafts/draft-ietf-behave-nat-icmp-11.txt.

[SH99] P. Srisuresh and M. Holdrege. *IP Network Address Translator (NAT) Terminology and Considerations.* RFC 2663, Internet Engineering Task Force, August 1999. http://www.rfc-editor.org/rfc/rfc2663.txt.

[SH08] H. Schulzrinne and R. Hancock. *GIST: General Internet Signalling Transport.* Internet-Draft draft-ietf-nsis-ntlp-17, Internet Engineering Task Force, Work in progress, October 2008. http://www.ietf.org/internet-drafts/draft-ietf-nsis-ntlp-17.txt.

[Sim95] W. Simpson. *IP in IP Tunneling.* RFC 1853, Internet Engineering Task Force, October 1995. http://www.rfc-editor.org/rfc/rfc1853.txt.

[Simco] *Open Source Implementation of SIMCO by University of Stuttgart.* Website. http://sourceforge.net/projects/simco-firewall/.

[SKR+02] P. Srisuresh, J. Kuthan, J. Rosenberg, A. Molitor, and A. Rayhan. *Middlebox communication architecture and framework.* RFC 3303, Internet Engineering Task Force, August 2002. http://www.rfc-editor.org/rfc/rfc3303.txt.

[SMS+02] R. P. Swale, P. A. Mart, P. Sijben, S. Brim, and M. Shore. *Middlebox Communications (midcom) Protocol Requirements.* RFC 3304, Internet Engineering Task Force, August 2002. http://www.rfc-editor.org/rfc/rfc3304.txt.

[Spe07] R. Spenneberg. *Linux-Firewalls mit iptables & Co.* Addison-Wesley, 2007. ISBN 3-8273-2670-2.

[SPG97] S. Shenker, C. Partridge, and R. Guerin. *Specification of Guaranteed Quality of Service.* RFC 2212, Internet Engineering Task Force, September 1997. http://www.rfc-editor.org/rfc/rfc2212.txt.

[SPTF06] N. Steinleitner, H. Peters, H. Tschofenig, and X. Fu. *Implementation and Performance Study of a New NAT/Firewall Signaling Protocol.* In *Proceedings of the 5th International Workshop on Assurance in Distributed Systems and Networks (ADSN2006), in conjunction with the 26th International Conference on Distributed Computing Systems (ICDCS 2006)* (July 2006), IEEE Computer Society. ISBN 0-7695-2541-5. ISSN 1545-0678.

[SQC06] M. Stiemerling, J. Quittek, and C. Cadar. *NEC's Simple Middlebox Configuration (SIMCO) Protocol Version 3.0.* RFC 4540, Internet Engineering Task Force, May 2006. http://www.rfc-editor.org/rfc/rfc4540.txt.

[SQT05] M. Stiemerling, J. Quittek, and T. Taylor. *Middlebox Communications (MIDCOM) Protocol Semantics.* RFC 3989, Internet Engineering Task Force, February 2005. http://www.rfc-editor.org/rfc/rfc3989.txt.

[SQT08] M. Stiemerling, J. Quittek, and T. Taylor. *Middlebox Communication (MIDCOM) Protocol Semantics.* RFC 5189, Internet Engineering Task Force, March 2008. http://www.rfc-editor.org/rfc/rfc5189.txt.

[STAD08] M. Stiemerling, H. Tschofenig, C. Aoun, and E. Davies. *NAT/Firewall NSIS Signaling Layer Protocol (NSLP).* Internet-Draft draft-ietf-nsis-nslp-natfw-20, Internet Engineering Task Force, Work in progress, November 2008. http://www.ietf.org/internet-drafts/draft-ietf-nsis-nslp-natfw-20.txt.

[SXM+00] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, and V. Paxson. *Stream Control Transmission Protocol.* RFC 2960, Internet Engineering Task Force, October 2000. http://www.rfc-editor.org/rfc/rfc2960.txt.

[SY06] D. Su and S.-J. Yoo. *Fast Handover Failure-Case Analysis in Hierarchical Mobile IPv6 Networks. IEICE Transactions 89-B*, 6 (2006), 1892–1895.

[TE06] H. Tschofenig and P. Eronen. *Analysis of Options for Securing the Generic Internet Signaling Transport (GIST).* Internet-Draft draft-tschofenig-nsis-gist-security-01, Internet Engineering Task Force, Work in progress, June 2006. http://www.ietf.org/internet-drafts/draft-tschofenig-nsis-gist-security-01.txt.

[Top] *Top, Procps - The /proc file System Utilities.* Website. http://www.procps.sourceforge.net/.

[TS00] G. Tsirtsis and P. Srisuresh. *Network Address Translation - Protocol Translation (NAT-PT).* RFC 2766, Internet Engineering Task Force, February 2000. http://www.rfc-editor.org/rfc/rfc2766.txt.

[Tsc08] H. Tschofenig. *Mobile IP Interactive Connectivity Establishment (M-ICE).* Internet-Draft draft-tschofenig-mip6-ice-02, Internet Engineering Task Force, Work in progress, February 2008. http://www.ietf.org/internet-drafts/draft-tschofenig-mip6-ice-02.txt.

[TTF+08] T. Tsenov, H. Tschofenig, X. Fu, C. Aoun, E. Davies, and I. Property. *GIST State Machine.* Internet-Draft draft-ietf-nsis-ntlp-statemachine-06.txt,, Internet Engineering Task Force, Work in progress, November 2008. http://www.ietf.org/internet-drafts/draft-ietf-nsis-ntlp-statemachine-06.txt,.pdf.

[UPnP] *The UPnP Forum.* Website. http://www.upnp.org/.

[USAGI] *UniverSAl playGround for IPv6.* Website. http://www.linux-ipv6.org/.

[v6St] *IPv6 to Standard.* Website. http://www.ipv6-to-standard.org/.

[VCF+00] J. Vollbrecht, P. Calhoun, S. Farrell, L. Gommans, G. Gross, B. de Bruijn, C. de Laat, M. Holdrege, and D. Spence. *AAA Authorization Framework.* RFC 2904, Internet Engineering Task Force, August 2000. http://www.rfc-editor.org/rfc/rfc2904.txt.

[VPS+06] A. Viinikainen, J. Puttonen, M. Sulander, T. ämäläinen, T. Ylönen, and H. Suutarinen. *Flow-based fast handover for mobile IPv6 environment - implementation and analysis. Computer Communications 29*, 16 (2006), 3051–3065.

[Win06] D. Wing. *Media Session Authorization.* Internet-Draft draft-wing-session-auth-00, Internet Engineering Task Force, Work in progress, January 2006. http://ietf.org/internet-drafts/draft-wing-session-auth-00.txt.

[Wro97] J. Wroclawski. *Specification of the Controlled-Load Network Element Service.* RFC 2211, Internet Engineering Task Force, September 1997. http://www.rfc-editor.org/rfc/rfc2211.txt.

[WRT07] D. Wing, J. Rosenberg, and H. Tschofenig. *Discovering, Querying, and Controlling Firewalls and NATs using STUN.* Internet-Draft draft-wing-behave-nat-control-stun-usage-03, Internet Engineering Task Force, Work in progress, July 2007. http://www.ietf.org/internet-drafts/draft-wing-behave-nat-control-stun-usage-03.txt.

[WSF+07] C. Werner, N. Steinleitner, X. Fu, H. Tschofenig, and C. Aoun. *NAT/FW NSLP State Machine*. Internet-Draft draft-werner-nsis-natfw-nslp-statemachine-06, Internet Engineering Task Force, Work in progress, November 2007. http://www.ietf.org/internet-drafts/draft-werner-nsis-natfw-nslp-statemachine-06.txt.

[XCZ+07] G. Xie, J. Chen, H. Zheng, J. Yang, and Y. Zhan. *Handover Latency of MIPv6 Implementation in Linux*. In *Global Telecommunications Conference, 2007. GLOBECOM '07* (Washington, DC, USA, nov 2007), IEEE Computer Society, pp. 1780–1785. ISBN 978-1-4244-1043-9.

[XORP] *XORP eXtensible Open Router Platform*. Website. http://www.xorp.org.

[YPG00] R. Yavatkar, D. Pendarakis, and R. Guerin. *A Framework for Policy-based Admission Control*. RFC 2753, Internet Engineering Task Force, January 2000. http://www.rfc-editor.org/rfc/rfc2753.txt.

[ZCCR06] E. D. Zwicky, S. Cooper, D. B. Chapman, and D. Russell. *Building Internet Firewalls (2nd Edition)*. O'Reilly, 2006. ISBN 1-56592-871-7.

Curriculum Vitae

Niklas Steinleitner

Persönliche Daten

Geburt 1980 in Marburg

Staatsangehörigkeit deutsch

Wissenschaftlicher Werdegang

1986-1992	Hermann-Schaft Schule, Fuldabrück
1992-1996	Integrierte Gesamtschule, Guxhagen
1996-1999	Friedrich-List Schule, Kassel
	Abschluss: Allgemeine Hochschulreife
2000-2005	Georg-August-Universität Göttingen
	Studium der Angewandten Informatik
	Abschlüsse:
	10/2004 Bachelor of Science (gut)
	12/2005 Master of Science (sehr gut)
seit 2006	Georg-August-Universität Göttingen
	Wissenschaftlicher Mitarbeiter am Institut für Informatik
	Lehrstuhl für Telematik
