Juho Paaso

Guaranteed access over consumer-level connections

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 3.11.2011

Thesis supervisor:

Prof. Jukka Manner

Thesis instructor:

M.Sc. (Tech.) Antti Mäkelä

Aalto University
School of Electrical
Engineering

AALTO UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING

ABSTRACT OF THE
MASTER'S THESIS

Author: Juho Paaso

Title: Guaranteed access over consumer-level connections

Date: 3.11.2011 Language: English Number of pages:7+50

Department of Communications and Networking

Professorship: Networking technology Code: S-38

Supervisor: Prof. Jukka Manner

Instructor: M.Sc. (Tech.) Antti Mäkelä

The scope of this thesis was providing guaranteed access over an array of unguaranteed and cheap consumer-grade connections. We tested how well high-availability access can be created with Redundant Array of Independent Internet Connections (RAIIC). In RAIIC, multiple unreliable connections are bundled together. Customer traffic is transferred on one connection at a time. State of the current connection is constantly monitored. If connectivity deteriorates, the system switches the traffic onto another unreliable connection. Connection switching should be invisible to the communicating nodes.

For this study we developed a Mobile IP based implementation. We were able to test the concept on running code. We measured how the connection switching affects the end-user experience and the results seemed quite promising. On TCP the switching corresponded to 1 - 1.5 second outage, which is considered to be well tolerable. VoIP quality remained "Fair" in Mean Opinion Score metrics.

Keywords: Networking, network programming, guaranteed access, Mobile IPv4, IP routing, UDP tunneling, Redundant Array of Independent Internet Connections (RAIIC), Service Level Agreement (SLA)

AALTO-YLIOPISTO
SÄHKÖTEKNIIKAN KORKEAKOULU

DIPLOMITYÖN
TIIVISTELMÄ

Tekijä: Juho Paaso

Työn nimi: Luotettava yhteys epäluotettavien liittymien yli

Päivämäärä: 3.11.2011 Kieli: Englanti Sivumäärä:7+50

Tietoliikenne- ja tietoverkkotekniikan laitos

Professuuri: Tietoverkkotekniikka Koodi: S-38

Valvoja: Prof. Jukka Manner

Ohjaaja: DI Antti Mäkelä

Tässä opinnäytetyössä tutkitaan konseptia Redundant Array of Independent Internet Connections (RAIIC), jossa ajatuksena on tarjota luotettava yhteys usean halvan ja epäluotettavan liittymän yli. Yhtä epäluotettavaa liittymää käytetään kerrallaan tämän tilaa jatkuvasti tarkkaillen. Jos yhteyden tila heikkenee, järjestelmä siirtää liikenteen toiselle liittymälle ilman että kommunikoivat osapuolet huomaavat muutosta. Näin ollen voidaan tarjota virtuaalinen, luotettava bittiputki halpojen yhteyksien yli.
Tutkimusta varten kehitimme Mobile IP -protokollaan pohjautuvan toteutuksen. Tällä pääsimme testaamaan konseptia oikeassa tietoverkossa. Mittasimme, miten liittymän vaihtaminen vaikuttaa loppukäyttäjän kokemaan palvelunlaatuun. TCP-protokollalla liittymän vaihto vastasi palvelussa 1 - 1.5 sekunnin katkoa, joka on vielä hyvinkin siedettävä. VoIP-palvelun laatu ei laskenut alle kohtalaisen tason (Mean Opinion Score -asteikolla ”Fair”).

Avainsanat: Tietoverkkotekniikka, luotettava yhteys, Mobile IPv4, IP-reititys, UDP-tunnelointi, Redundant Array of Independent Internet Connections (RAIIC), Service Level Agreement (SLA)

Preface

I want to thank my instructor Antti Mäkelä, supervisor Jukka Manner and co-worker, PhD student Nuutti Varis for all the support during the development process. It was very interesting and educational for me to work in a team with professionals of this level. Jukka is a professor with very nice and practical approach to research. Antti is an experienced networking expert who gave me assistance in issues concerning networking and debugging distributed systems. Nuutti is a very competent expert in computer science who helped me with problems concerning C libraries and Linux configurations. I found the project very challenging and it would have taken me twice as much time to finish it without this guidance.

I would also like to thank Elli Saarela for all the encouragement during my studies and work, and Pekka Pyysalo, Kaisa Korhonen, Lorenzo Pansana and Florian Dupuy for the peer support and good times at the university.

Otaniemi, 3.11.2011

Juho M. S. Paaso

Contents

Abbreviations

API	Application Programming Interface
AS	Autonomous system
BGP	Border Gateway Protocol
CN	Correspondent Node
CoA	Care-of Address
CoTI	Care-of Test Init message
CoT	Care-of Test message
CR	Correspondent Router
EDGE	Enhanced Data rates for Global Evolution
GPL	GNU General Public License
GRE	Generic Routing Encapsulation
HA	Home Agent
HAaRO	Home Agent assisted Route Optimization
HMAC	Hash-based Message Authentication Code
HoA	Home Address
HoT	Home Test message
HoTI	Home Test Init message
HUT	Helsinki University of Technology
ICMP	Internet Control Message Protocol
IETF	Internet Engineering Task Force
IP	Internet Protocol
ISP	Internet Service Provider
KRm	Registration Management Key
MD5	Message-Digest algorithm 5
MIP	Mobile Internet Protocol
MN	Mobile Node
MOS	Mean Opinion Score
MPLS	Multi Protocol Label Switching
MR	Mobile Router
NEMO	Network Mobility
NAI	Network Access Identifier
NAT	Network Address Translation
QoS	Quality of Service
RAIIC	Redundant Array of Independent Internet Connections
RFC	Request For Comments
RO	Route Optimization
RPF	Reverse Path Filtering
RR	Return Routability
SHA	Secure Hash Algorithm
SLA	Service Level Agreement
SLS	Service Level Specification
VoIP	Voice over IP
VPN	Virtual Private Network
WAN	Wide Area Network
WLAN	Wireless Local Area Network

1 Introduction

Many organizations need reliable intra-site connectivity. Connectivity guarantees are normally provided as Service Level Agreements (SLA) by service providers. Guaranteed access is traditionally implemented by dedicating lines with redundancy guarantees, which is expensive. In contrast, consumer-level connections are sold to households with low price. Even though they operate on highly overbooked lines they still work well most of the time. However, there are no guarantees concerning these connections.

Redundant Array of Independent Internet Connections (RAIIC) [1] is a concept for providing reliable and economical access by switching between several unreliable connections. One connection is used at a time. If the state of the connection deteriorates, the traffic is switched onto another connection. Connectivity remains as long as all the connections are not down at the same time. With independent connections using separate physical lines and links, the probability for a connection breakdown is almost negligible.

In this thesis, switching between the connections is conducted with Mobile IPv4 [3] handovers. Mobile IP enables a node to move to another network without changing the address visible to other nodes (Home Address). This enables switching between networks while maintaining the same IP address. All traffic in Mobile IP network goes via Home Agent which is an anchor node that associates Mobile Nodes' Home Addresses with their current locations: Care-of Addresses. Home Agent also accepts nodes to join the Mobile IP network and authenticates nodes to each other.

Mobile IPv4 protocol has been extended with many functionalities. A Mobile Node is able to act as Mobile Router with Network Mobility (NEMO) extension [8]. Mobile Router can provide connectivity to a whole set of networks. Mobile Routers are also able to create direct tunnels between each other instead of routing all traffic via Home Agent. This extension, Home Agent assisted Route Optimization (HAaRO) [6], enables distributing the traffic load. Also the state of a connection can be monitored via the keepalive mechanism specified in Mobile IP Traversal of Network Address Translation (NAT) Devices [7] extension. This mechanism enables noticing if quality of a connection deteriorates.

Reliable access over unreliable connections can be built using these Mobile IP features. Mobile Router provides a company site access using one consumer-level connection. Each Mobile Router is connected to multiple service providers. State of the connection via the current service provider is actively monitored via the keepalive mechanism. In case of path failure, the connection is handed over to another path, which is on another service provider's network. With this ability to switch between service providers, the Mobile Router is able to maintain connectivity even though the current path fails.

In order to be able to test the concept in real world, these functionalities were implemented following the tao of IETF [2]. This work was a part of the Future Internet programme of TIVIT (Finnish Strategic Centre for Science, Technology and Innovation in the field of ICT). The contribution of the author was reviewing the HAaRO specification draft and participating in the performance measurements

in addition to the design and implementation of the RAIIC software.

At the time we started developing the software Mobile IPv4 specification existed and there were implementations of the protocol available. NEMO specification also existed but there were no open source implementations of that. HAaRO specification was in progress and needed in order to make Mobile IPv4 a valid technology for RAIIC. Being able to measure performance with running code verified that the system works as well as in simulations.

The implementation was built by modifying Dynamics project (version 0.8.1) [17]. Dynamics is an implementation of the older version of Mobile IP protocol, described in RFC 2002 [5]. With the running implementation we measured how much the end-user experience deteriorates when the current unreliable connection breaks down and the system switches traffic onto another connection. Measurements were performed with TCP and VoIP traffic. On TCP the switching corresponded to 1 - 1.5 second outage, which is considered to be well tolerable. VoIP quality remained "Fair" in Mean Opinion Score metrics.

In this thesis we discuss the technologies, the implementation and the measurement results. Chapter 2 clarifies what guaranteed access means, how it can be implemented and where it can be used. Chapter 3 describes the technologies utilized to achieve the required functionalities. Chapter 4 describes how Dynamics was upgraded to implement the RAIIC-functionalities. Chapter 5 aims to help the following developers to get started with our implementation. Chapter 6 describes the testing conditions, test cases and measurement results achieved with the software.

2 Guaranteed Wide Area Network access

This chapter gives an overview to the scope of this thesis which is building guaranteed access over unreliable connections. In this project we researched how well a RAIIC-implementation based on Mobile IP would work as a business network where long lasting outages are not prohibited and quality of such services as Voice over IP (VoIP) should remain tolerable.

The first section of this chapter describes the differences between a business network and a normal consumer-grade network. The second section discusses possible techniques in implementing business network features. The final section considers other possible use cases for this technology. The concept behind this technology is described more thoroughly in article Concept for providing guaranteed service level over an array of unguaranteed commodity connections [9].

2.1 Differences between consumer-grade and business-grade connections

Many enterprises want guarantees that there are no breakdowns in the intranet stalling the business. Also such services as private site-to-site connectivity, good quality VoIP and authentication of users are often needed. Service providers offer such guarantees as Service Level Agreements (SLA). SLA is a formal contract between a service provider and a customer. It defines quality metrics and target values for provided services.

In consumer-grade Internet connections there are no Service Level Agreements. The service provider attempts to provide best possible service but furthermore, no guarantees whatsoever are offered. Consumer-grade connections have theoretical maximum bandwidths but the networking resources are typically highly overbooked under the operator's core network. Thus, the experienced bandwidth may occasionally be a lot lower and breakdowns may occur. A commodity connection still typically works satisfactorily most of the time.

SLA may contain one or more specifications concerning the service level. Service Level Specifications (SLS) may include both technical and procedural metrics. Here are some possible specifications listed:

- technical guarantees
  - Redundancy specifications (for specific nodes, links, paths)
  - Fault response/remediation time
  - Connectivity characteristics, such as
    - Uptime, can be expressed in percentages or expected downtimes during a course of a specific length of time (e.g. "maximum downtime: 15 minutes in a year")
    - Bandwidth
    - Delay, jitter

  - Traffic classification, specifying separate characteristics for different types of traffic, e.g. Voice over IP having strict quality requirements while other applications only have best-effort service.
  - Traffic source and destination, specifying separate characteristics for different peers, e.g. two primary offices running mission-critical systems having high requirements. Internet access has no SLS behind the service provider's network.

- procedural guarantees
  - Maximum technical support response time
  - Maximum time to get broken networking hardware replaced
  - Possible sanctions for service provider for failing to conform to the agreement, either in monetary or other terms

An SLA-covered business connection is typically about ten times more expensive [10] than a consumer-level connection with identical capabilities in normal situations. Adding connectivity guarantees increases the price considerably. Gartner's report [11] suggests that the price of a SLA specifying 99.9... % uptime increases 30 % with each additional nine stacked to the reliability factor. The massive difference between the prices makes it interesting to research whether it would be profitable to build a reliable connection using several consumer-grade connections without any SLAs. Even though each commodity connection may break down, they most likely wont be down at the same time. By relying on this assumption we are able to offer connectivity guarantees with a system that is able to switch to another connection in case the state of the current one deteriorates.

2.2 Possible technologies to implement high availability

High availability can be implemented using several different technologies. These technologies differ in quality of provided service level, price, complexity etc. A few solutions with their pros and cons are presented in Table 1 and discussed in this section.

Dedicated access with Service Level Agreements Traditionally high availability has been implemented with one single service provider using a Wide Area Network (WAN) technology, such as Multiprotocol Label Switching (MPLS). SLA guarantees are negotiated requiring the service provider to make custom configurations, dedicate links and reserve extra resources to ensure the quality of service. While the provider's core network is normally shared between multiple customers, at least the last-mile-links are typically dedicated for specific customers. This solution provides very good service level and it is well established but the price is normally very high compared to a commodity Internet access.

Table 1: Pros and cons of different connection technologies.

Dedicated access from one service provider	+ Very good service level + Well established method - Very expensive
Multi-homing via several providers with own AS number	+ Cheaper than dedicated lines + Service provider independence - Competence requirements - Breakdown recovery time
NAT with multiple links, Dynamic DNS, VPN	+ Cheap + Simple to deploy - Visible to end-users in case of outages - Inbound traffic reaction time - VPN requires at least one guaranteed connection
Redundant Array of Independent Internet Connections	+ Cheap + Simple to deploy + Invisible to end-users - Relies on uncontrollable infrastructure - All connections may be down at the same time

Multi-homing via several service providers Another possible approach is to implement multi-homing via several service providers. In this case, there is not necessarily any SLAs. Instead, the redundancy is implemented by connecting via multiple service providers concurrently. In case of path failures, a routing protocol - normally Border Gateway Protocol (BGP) - takes care of establishing the connection via another service provider. This approach has significantly slower breakdown recovery time; route modifications have to propagate throughout the network, and the propagation time can grow very large. In worst cases, the propagation can last a full hour [12]. The worst case is when the solution is used for accessing the Internet, as routing modifications have to propagate globally.

Multi-homing solution is normally cheaper than dedicated lines. However, the

customer still needs business-grade connectivity with dynamic routing protocol support from multiple service providers. In addition, managing the address space within the customer's own Autonomous System (AS) causes additional IT costs.

Multilink Network Address Translator Less costly approaches are also available for implementing redundancy at customer sites. These solutions are implemented by an overlay service over basic non-guaranteed access technologies. However, they are somewhat less flexible and less robust than the technologies described above.

The simplest solution is to use a Network Address Translator (NAT) with multiple outgoing links [13]. NAT provides redundant Internet access by sending outgoing packets to different links. The link is chosen by using some sort of heuristics. This system is simple to implement but it can typically accept inbound connections only from a specific link. This way there is no redundancy to services offered towards the Internet. Dynamic DNS [14] can be used for directing incoming traffic but in case of connection outages, it is not able to react quickly to changes or to change dynamic IP address assignments.

The multilink NAT approach can be combined with a Virtual Private Network (VPN) technology, such as IP Security Architecture (IPsec), in order to form site-to-site WAN connectivity. However, VPN normally has at least one fixed node. Therefore, at least one guaranteed connection is needed keeping the anchor node available. Also reconfiguration of VPN is complicated when the underlying topology changes as connections fail or IP allocations change. Redundancy and flexibility could be provided via multi-homing using routing protocols but service providers don't allow setting up BGP peering on a regular consumer-grade connection.

Redundant Array of Independent Internet Connections The approach studied in this thesis is Redundant Array of Independent Internet Connections (RAIIC). This solution is based on low costs and it is immediately deployable on the Internet without any changes on the end-hosts. High-quality access is provided by switching between several low-quality connections. The access is visible to the end-hosts as a single, high-reliability bit-pipe.

RAIIC enables connecting intranet address spaces over the Internet. It is not possible to control routing over commodity connections so switching between paths is enabled using tunnels. The state of the current tunnel is actively monitored and the traffic is handed over to another tunnel in case of connectivity issues. The handovers need to be fast enough to achieve transparent response of failures.

In this approach, the infrastructure under the connections is completely controlled by external entities. The wires and links connecting the sites are maintained by the service providers or operators so changes in connectivity are unpredictable. It is also possible that occasionally none of the connections is operational. In this case there is no way to offer access. Therefore, it is important to choose connections that do not share hardware resources. Independence between the connections decreases the possibility of all paths being non-operational at the same time.

2.3 Other use cases for reliable access over unreliable connections

The ability to switch between connections could basically be useful in any device having multiple networking interfaces connected to independent networks. For example smartphones normally have 3G, EDGE (Enhanced Data rates for Global Evolution) and WLAN antennas. A smartphone could switch between its wireless networks depending on the state of the connection. Wireless connections can be very unstable and for example switching between 3G and WLAN is usually done manually. Using RAIIC implementation could increase the experienced service level. A user could be using a WLAN without even knowing that a WLAN hot-spot is present. However, the keepalive mechanism could be quite consuming for the device's battery so the power efficiency would require more research.

Connections are also less stable in vehicles. Quality of access could be increased for example on a train providing Internet access to its travelers. A Mobile Router running RAIIC-implementation could connect the LAN on the train to the Internet by using several antennas simultaneously. Also, during a military crisis or a natural disaster, mobile command centers could be set up using RAIIC.

3 Mobile IPv4 with extensions

The RAIIC implementation developed for this thesis is based on Mobile IPv4 protocol [3]. Mobile IP offers a simple signaling mechanism with fast handovers. The protocol has also been extended with many functionalities. This chapter describes the technology utilized in implementing guaranteed access.

Mobile IP enables a node to move to another network without changing the address visible to the nodes it is communicating with. Nodes communicating with a Mobile Node are called Correspondent Nodes. Mobile IPv4 addressing is explained in Section 3.1. When moving to a new network a handover is required. Mobile IP has a very simple signaling mechanism based on requests and responses. The signaling mechanism is illustrated in Section 3.2. Mobile IP includes a UDP tunneling extension, which is discussed in Section 3.3. This extension also contains connection monitoring mechanism. Network Mobility extension, explained in Section 3.4, turns Mobile Node into Mobile Router able to provide access to a whole set of networks. Home Agent assisted Route Optimization extension helps distributing traffic load more equally on Mobile IP network. This extension is described in Section 3.5. Authentication in direct tunneling is discussed in Section 3.6.

3.1 Mobile IPv4 addressing

Mobile IP enables a node to switch between networks without changing the IP address visible outside the network: the Home Address (HoA). A Mobile Node is always available via its Home Address which points to the node's Home Network (HN). If a Mobile Node moves away from its Home Network, it informs its Home Agent about its current location: the Care-of Address. Home Agent is a router which keeps track of the Mobile Nodes' locations. If a Mobile Node is away from its Home Network, all traffic it sends or receives goes via the Home Agent which forwards packets to the node's Care-of Address.

Mobile Node's Care-of Address may be represented by the node itself or by a Foreign Agent. The former mode is called co-located Care-of Addressing. In this case, the Care-of Address is the IP-address of the node on the current Foreign Network. In the latter addressing mode the node's Care-of Address is the address of a Foreign Agent. Foreign Agent is a router on a Foreign Network that takes care of tunneling the traffic of the Mobile Nodes connected to its network. Traffic is tunneled to the Home Agent which forwards it to the Correspondent Nodes. In our case, co-located Care-of Addressing was used so only this mode is discussed in this thesis.

Care-of Address is only used as the tunnel end-point between the Mobile Node and the Home Agent. Correspondent Nodes are never aware of the Care-of Addresses. They communicate using Home Addresses, which point to the nodes' Home Network and end up to the Home Agent in case the node's are away. Home Agent identifies Mobile Nodes by the Home Address so two nodes never have equal Home Addresses. Home Agent maintains information about the nodes' current locations in a Binding Table. Binding Table associates Mobile Nodes' Home Addresses with

their current Care-of Addresses.

Home Network may also be virtual. This means that a Mobile Node is never on its Home Network. Home Agent allocates Home Addresses from a virtual address pool. All addresses in the pool point to the Home Agent so all traffic is still forwarded by the Home Agent. In this thesis we only discuss the scenario of virtual Home Network so the Mobile Nodes never visit their Home Network.

3.2 Mobile IPv4 signaling

Mobile IP has a very simple signaling mechanism based on requests and responses. If a Mobile Node moves to another network, it sends a *Registration Request* (RegReq) to its Home Agent informing of its new IP address. Home Agent accepts or rejects the Registration Request with a *Registration Reply* (RegRep). For example, when a Mobile Node with Home Address 10.10.10.10 moves to another network and its Care-of Address changes from 80.80.70.70 to 80.80.80.80, it sends its Home Agent Registration Request having the information "Me, Home Address 10.10.10.10, can now be reached via Care-of Address 80.80.80.80". If the request is accepted, the Home Agent sends the Mobile Node Registration Reply with accepting code and modifies its Binding Table to contain the node's current location.

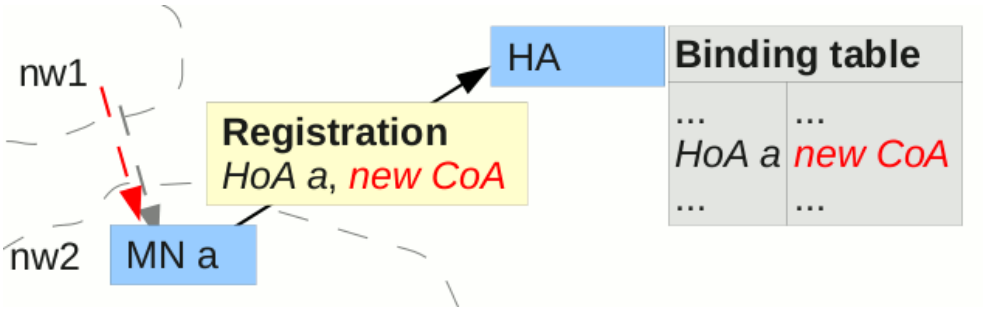

Figure 1: Mobile node handover.

Mobile IP handover process is illustrated in Figure 1. In the figure a Mobile Node moves to another network, informs its Home Agent of its new Care-of Address and the Home Agent updates its binding table. Here is the handover of the figure in detail:

1. Mobile Node *MN a* with Home Address *HoA a* moves from network *nw1* to network *nw2*, and its Care-of Address changes to *new CoA*.

2. *MN a* informs the new Care-of Address to its Home Agent, *HA*.

3. The Home Agent updates the entry of *HoA a* in its Binding Table to contain the new Care-of Address *new CoA*.

Each HoA-CoA binding has a lifetime. If a Mobile Node is connected to the same Foreign Network long enough, it is required to send Registration Request to its Home Agent in order to maintain the binding. Registration Request with

unchanged Care-of Address is called Reregistration. If no Reregistration is received during the expiration interval, Home Agent removes the node from its Binding Table. If a Mobile Node wishes to leave the Mobile IP network (or returns to its Home Network), it sends its Home Agent Registration Request with lifetime field set to 0. This packet is called Deregistration.

Mobile IP protocol has been extended new functionalities. The extensions are included in the registration messages. For example, if a Mobile Node needs to inform the Home Agent of its Network Access Identifier (NAI), it adds Mobile Node NAI Extension [15] into its Registration Request. Home Agent acknowledges extensions in Registration Reply by adding a replying extension or by setting the reply code to a corresponding value.

3.3 UDP in IP tunneling

The original Mobile IP specification [3] defines two different tunneling mechanisms: IP-in-IP and Generic Routing Encapsulation (GRE). Both of them are based on encapsulating packets with new IP packets with the tunnel endpoint as the destination address. In Mobile IP, traffic is tunneled between the Mobile Node and the Home Agent. In order to forward packets to a Mobile Node, Home Agent adds an extra layer of encapsulation pointed to the Mobile Node's Care-of Address. Also Mobile Node adds an IP packet destined for the Home Agent outside each outgoing packet. This way all traffic to any Correspondent gets transferred via the Home Agent.

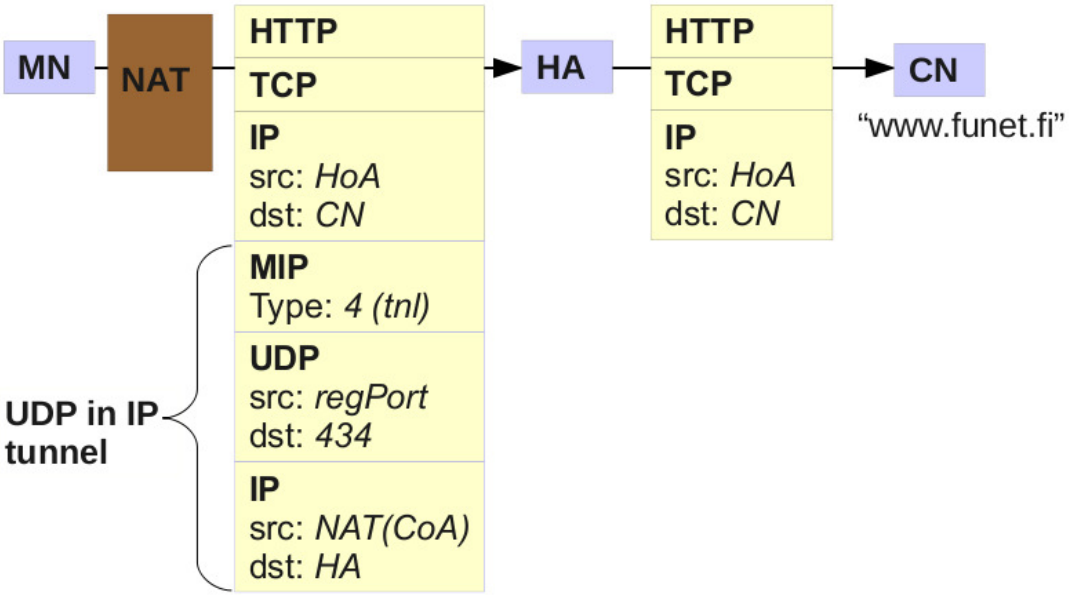

Figure 2: MN behind a NAT sending a HTTP GET message via Home Agent to *www.funet.fi*.

While this kind of tunneling works in networks with full end-to-end connectivity, commonly IP-networks are surrounded by Network Address Translators (NAT) and firewalls. Some firewalls discard all traffic that is not TCP or UDP. In order to

enable tunneling in the modern Internet, a new tunneling scheme was introduced to Mobile IPv4. This extension is Mobile IP Traversal of Network Address Translation (NAT) Devices [7].

A Mobile Node supporting UDP tunneling registers with a UDP Tunnel Request extension in its Registration Request. If the Home Agent replies with a UDP Tunnel Reply extension having an accepting code, the Mobile Node starts sending tunnel keepalive packets to the Home Agent's Mobile IP port (434). The tunnel keepalive packets are ICMP Echo Requests. Keepalives are sent to ensure that the Home Agent is able to send packets to the Mobile Node's even though the Mobile Node is currently behind a NAT or a firewall. When there is constantly traffic from the Mobile Node, NAT does not remove the session from its Session Mapping Table and close the connectivity from outside the network. The UDP tunnel remains open.

The keepalive mechanism can also be utilized when monitoring the state of the connection. If ICMP Echo Reply packets come with increased delay or if they are never received, we can conclude that the connectivity has deteriorated. In this case we perform a handover so that the traffic is switched onto another connection.

The NAT Traversal extension introduces a new message type to the Mobile IPv4 protocol. In addition to the signaling messages (Registration Requests and Replies) there is a third MIP message type for the data transfer: MIP Tunnel Data Message. This header is added on the outer UDP header (see Figure 2) to indicate that the tunneled UDP traffic is MIP Tunnel Data traffic. When sending a packet to the Home Agent, Mobile Router encapsulates it with MIP Tunnel Data Message header and sends it as UDP to the Home Agent's Mobile IP port.

Protocol stack used in UDP in IP tunneling is illustrated in Figure 2. The Mobile Node "MN" sends HTTP GET to a node outside the Mobile IPv4 network. The IP datagram is first tunneled to the Home Agent by encapsulating it with MIP Tunnel Data Message header and sending the MIP packet on UDP from the Mobile Node's Care-of Address to the Home Agent's MIP signaling port. When receiving UDP packet containing MIP Tunnel Data Message, the Home Agent removes the MIP in UDP in IP stack and forwards the remaining IP packet to the Correspondent Node "CN".

The Correspondent replies to the Mobile Node's Home Address, which is routed to the Home Agent. The Home Agent has the Care-of Address corresponding to the Home Address in its Binding Table, so it is able to forward the packet to the Mobile Node's current network. The Home Agent adds MIP Tunnel Data Message header to the packet and sends it on UDP from its MIP signaling port to the port the Mobile Node registered from. When receiving the UDP packet, the Mobile Node removes the MIP Tunnel Data Message header and forwards the remaining IP packet to the corresponding process.

3.4 Network Mobility

Network Mobility (NEMO) extension turns the Mobile Node into a Mobile Router which is able to make a whole set of networks mobile. The nodes on these networks are not aware of the mobility of the router. All traffic from hosts in the Mobile

Networks end up to the Mobile Router which tunnels it to its Home Agent. The Home Agent forwards the traffic just like in the regular Mobile IP.

Mobile Router registers a network to the Home Agent with Mobile Network Request extension. One Mobile Network Request extension per network is added to the Registration Request. Home Agent accepts or rejects these networks by adding a Mobile Network Acknowledgement extension for each Mobile Network Request extension. The acknowledgement has a code that informs if the Mobile Network was accepted or not.

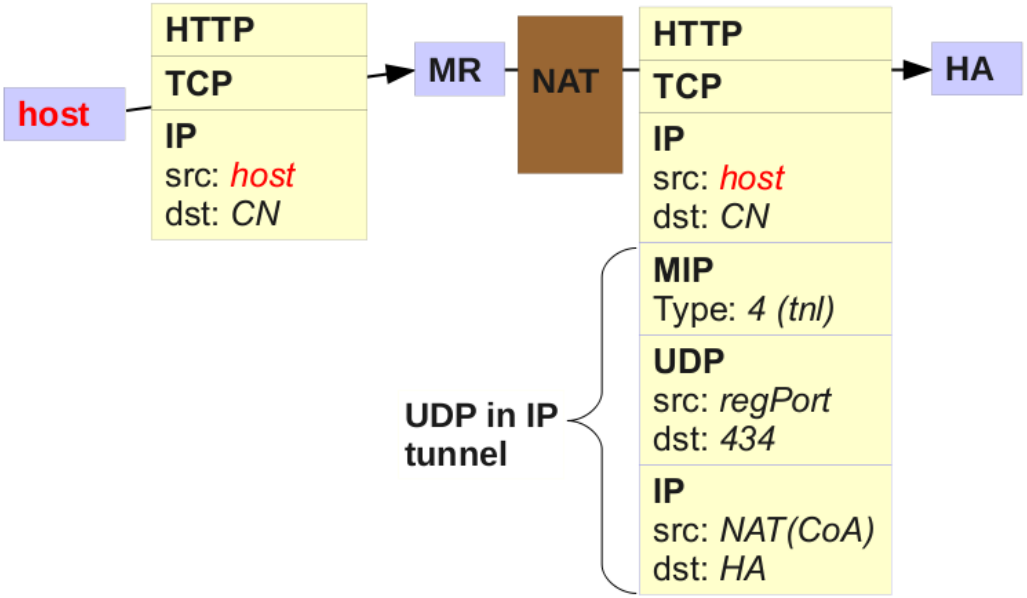

Figure 3: HTTP packet sent by a host in the Mobile Network.

The Home Agent keeps track of the Mobile Routers and the networks they represent in the Binding Table. Home Agent won't allow several routers to represent networks with the same address. Correspondents communicate using the addresses of the hosts inside the Mobile Networks so all the Mobile Networks have to have distinct addresses. The IP address inside Mobile IP the tunnel is not anymore the Mobile Router's Home Address but the address of the host currently communicating via the Mobile Router. Therefore the Mobile Network addresses can not overlap.

Figure 3 illustrates the protocol stack of NEMO traffic. A packet is sent by a host in a Mobile Network of a Mobile Router ("MR"). The packet is a HTTP message sent to some Correspondent Node (CN) on the Internet. All traffic from any host in the Mobile Network ends up to the network's Mobile Router which tunnels it as UDP towards the Home Agent (HA). The address of the host has replaced the HoA of the Mobile Router inside the tunnel.

3.5 Home Agent assisted Route Optimization

Home Agent assisted Route Optimization (HAaRO) extension enables Mobile Routers to create direct tunnels between each other. Instead of sending all traffic via the

Home Agent, Mobile Routers can communicate with each other directly. Home Agent is the trusted signaling anchor that coordinates the connecting and authenticates the Mobile Routers to each other. Mobile Routers create direct tunnels by sending Registration Requests to each other just like in the Home Agent's case. Registraion is optionally preceded by Return Routability procedure which is used for calculating an authentication key for the session: Registration Management Key (KRm).

Only the Home Agent has a list of all the Mobile Nodes on the Mobile IP network. If a Mobile Router receives an inter Mobile Router Registration Request (direct tunneling request) from an unknown Mobile Router, it has to verify the existence of the router from the Home Agent. Without this solicitation basically anyone could request traffic from the Mobile Networks of the router. The Home Agent grants the authorization if it finds the solicited Mobile Router from its Binding Table. Mobile Router maintains its own Route Optimization (RO) Cache which is similar to the Binding Table of the Home Agent. Correspondents that are verified are stored in the RO Cache so the solicitation needs to be done only once per Correspondent.

This chapter describes step-by-step how Route Optimization is initiated in Mobile IP network. Mobile Routers register to each other like they register to the Home Agent. Only some extra procedures are performed in order to verify the routers and test the connectivity. Packets used to set up a direct HAaRO-connection are illustrated in Figure 4.

Advertising HAaRO-support

A Mobile Router supporting HAaRO adds RO Capability extension to the Registration Request when registering to its Home Agent. Home Agent acknowledges the RO Capability extension with RO Reply. Home Agent also advertises (with RO Prefix Advertisement extension) all the other Mobile Routers in the Mobile IP network that support HAaRO. Also these routers' Mobile Networks behind them are advertised. The Mobile Router adds these Correspondent Routers into its RO Cache. Direct tunnel initialization begins with Return Routability process towards the Correspondent Router in order to create Registration Management Key (KRm). This signaling is illustrated in Figure 4, stages 1 - 3.

Return Routability procedure

A Mobile Router tests connectivity towards a Correspondent Router by sending two initialization messages: Home-Test Init (HoTI) and Care-of-Test Init (CoTI). HoTI is sent from the Mobile Router's Home Address. CoTI is sent from the Care-of Address which is to be the source address of the direct tunnel. Both initialization messages are sent to the Correspondent Router's Home Address so they get routed via Home Agent. The Correspondent replies to HoTI with Home-Test (HoT) message and to CoTI with Care-of-Test (CoT) message. Mobile Router calculates Registration Management Key (KRm) from the token information in those messages. The router is able to send Registration Request with an authenticator hash

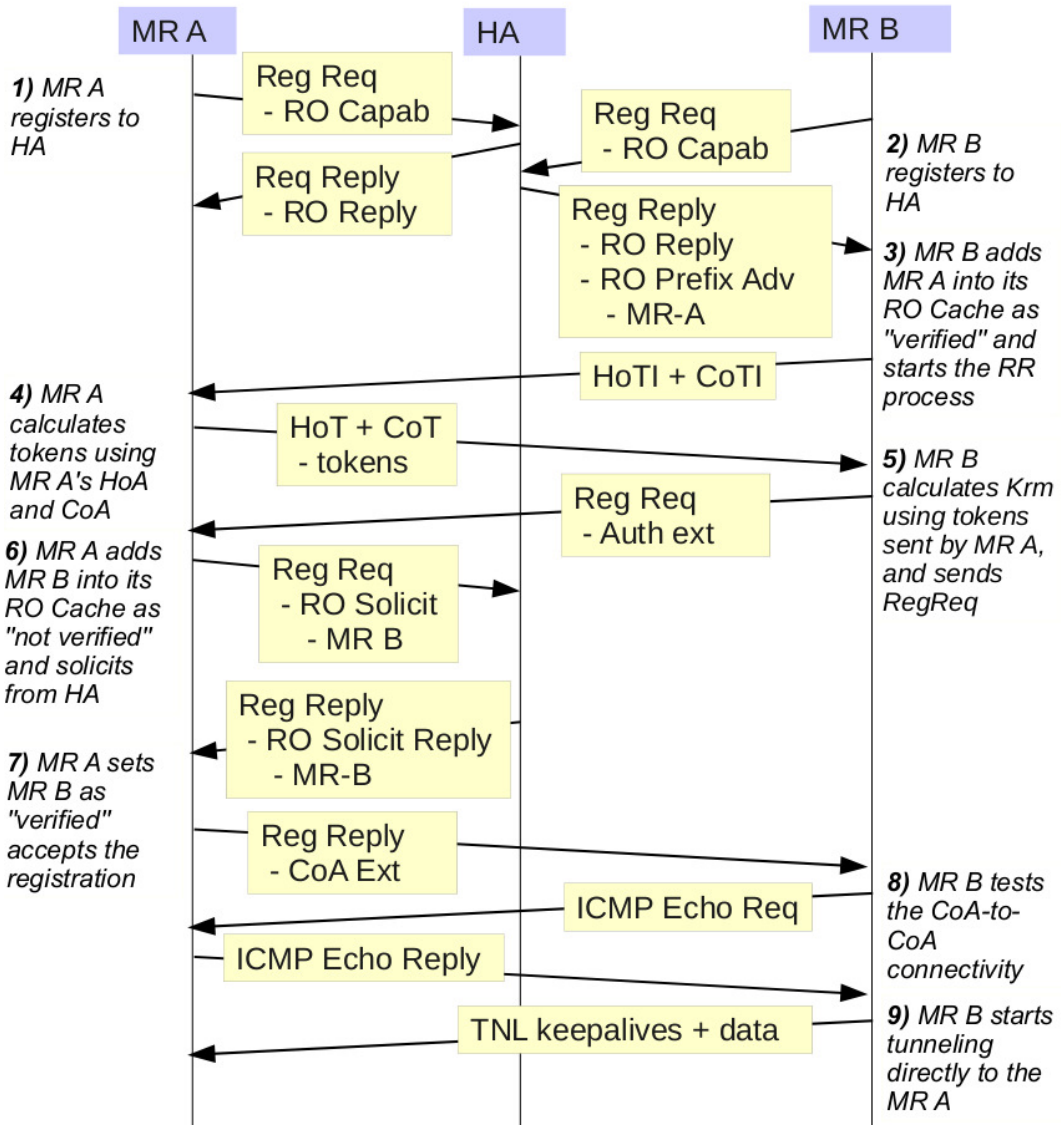

Figure 4: Mobile Routers A and B join the Mobile IP network and the router B initializes a direct tunnel towards the router A.

encrypted with this key. Authentication in route optimization is discussed more in Section 3.6. Return Routability procedure is illustrated in Figure 4, stages 3 - 5.

Solicitation of an unknown router

When receiving a registration from the Mobile Router, the Correspondent first checks if the Mobile Router exists in its RO Cache. If the Home Agent has advertised this router, it has been added into the RO Cache and it can be trusted. If the registering Mobile Router is unknown, the Correspondent has to verify its existence from the Home Agent. The solicitation message is a Registration Request including RO Solicitation extension. This extension includes Home Address of the Mobile Router and all networks the router claims to represent. If the Home Agent

finds the corresponding information from its Binding Table, it verifies the information with RO Solicit Reply. This router solicitation process is illustrated in Figure 4, stages 6 - 7. Registration can be accepted as the Mobile Router is verified by the Home Agent.

Setting up direct inter Mobile Router tunnel

If the Correspondent finds the registration valid, it replies with the code Registration Accept. If both nodes support UDP tunneling, the Correspondent adds at least one Care-of Address (CoA) extension to the reply. By this extension the Mobile Router is informed of the Correspondent's Care-of Addresses usable for HAaRO-tunneling.

Mobile Router starts testing the CoA-to-CoA connection by sending an ICMP Echo Request to the first CoA in the CoA extension list. If the first one does not reply the Mobile Router goes through the list until one replies. Source of the tunnel will be the Care-of Address the Mobile Router registered from. Destination will be the Care-of Address from which the ICMP Echo Reply was received. This direct CoA-to-CoA connection testing process is illustrated in stages 6 and 7 of Figure 4.

The Mobile Router creates a tunneling device for the new tunnel. Routes for each network represented by the Correspondent are set to this device. Now traffic sent from one of the Mobile Router's networks to one of the Correspondent's networks is sent to the direct tunnel.

If the Mobile Router notices that the Correspondent no longer replies to tunnel keepalives, it removes the tunneling device and all routes towards it. This way all traffic is again routed via Home Agent. Setting up a new tunnel requires a new registration possibly preceded by a Return Routability routine.

Deciding the tunneling roles: Mobile Router and Correspondent Router

There are two roles in direct tunneling. One router acts as the Mobile Router and the other router is the Correspondent, whose role is similar with the Home Agent. A direct tunnel is attached to Mobile Router's registration port and Correspondent's Mobile IP port (434). When setting up the tunnel, Mobile Router is the active side which takes care of finding a working CoA-to-CoA pair while Correspondent remains passive and waits for the Mobile Router to get an ICMP Echo Request through to one of its CoAs. Both nodes monitor the tunnel's state by sending keepalives. The one noticing a problem in the connection first, sends a registration from a new CoA.

The roles are decided by the Home Addresses. The router with smaller Home Address acts as the Mobile Router. Before Mobile Router accepts a registration from another Mobile Router, it compares the Home Addresses. In case its Home Address is smaller than the router's registering, it replies with code PRE_ACCEPT. This indicates to the other router, the Correspondent, that it should wait for the Mobile Router to register. The Mobile Router performs RR procedure, calculates KRm and sends registration to the Correspondent. If the registration is valid, the Correspondent replies to it with the normal ACCEPT code and starts waiting for an incoming CoA-to-CoA ICMP Request.

3.6 Authentication in Route Optimized Mobile IPv4 network

There are two security matters concerning direct tunnels. The first one is to verify whether a Correspondent Router can trust that an arbitrary Mobile Router is indeed the proper route to reach an arbitrary Mobile Network. Routers trust information provided by Home Agent so in this case a solicitation is performed.

The second matter to verify is that a specific Care-of Address is indeed managed by a specific Home Address. This is done via Return Routability routine where the Correspondent calculates Registration Management Key (KRm) from Home Address and Care-of Address of the node requesting registration. The Correspondent does not establish any state information during the Return Routability procedure, mitigating denial of service attacks. The cryptography scheme is very similar to the one used in Mobile IPv6 [16].

3.6.1 Authentication keys

Registration Management Key (KRm) authenticates a registration. This key is calculated using both private and public authentication keys. Each Mobile Router has a private Correspondent Router Key (Kcr) and private Nonces ("Number used only once"). Cookies and Tokens are used as public keys.

Private keys: Correspondent Router Key and Nonce Each router has a private Correspondent Router Key (Kcr), which is used when the router is acting as a Correspondent. With this key the Correspondent is able to verify that the keygen tokens, sent by the registering Mobile Router, are its own. The Correspondent Router Key is a 96 bit random number.

Routers also maintain indexed Nonces. Nonces are random numbers generated at regular intervals (e.g. 10 minutes). Only the Nonce Index is sent online. The Correspondent may use the same Nonce with all Mobile Routers. In addition to the current Nonce, routers keep in memory a small set of previous nonces whose lifetime have not expired yet.

Cookies and Tokens Cookies and Keygen Tokens are used in identifying communication initialization requests (HoTI, CoTI) and their replies (HoT, CoT). Home Init Cookie and Care-of Init Cookie are 64 bit random values sent within HoTI and CoTI messages, and returned in HoT and CoT messages. MR generates a new pair of cookies for each HoTI and CoTI message pair it sends. Cookies verify that the HoT or CoT message matches to the HoTI or CoTI message sent. These Cookies also ensure that someone who has not seen the request cannot spoof responses.

HoT and CoT messages each contain a cookie, a nonce index and a token. Tokens are 64 bit values generated using the current Correspondent Router Key and Nonces as well as the Home or Care-of Address. A token is valid until the Correspondent Router Key or the Nonce expires.

3.6.2 Encryption

Correspondent Router generates Registration Management Key (KRm) from Home Keygen Token and Care-of Keygen Token which are calculated from the Mobile Router's Home Address and Care-of Address, respectively. HAaRO specification defined HMAC_MD5 as the encryption algorithm until version 4. In version 5 the algorithm was changed to HMAC_SHA1 because MD5 is currently very vulnerable to collision attacks [19]. SHA1 produces 160 bit code instead of 128 so it is slightly heavier than MD5.

3.6.3 Keys generated during Return Routability procedure

Return Routability (RR) procedure begins when a MR sends a CR HoTI and CoTI in order to acquire the Keygen Tokens that are received in HoT and CoT. When the Correspondent receives the HoTI message, it generates a Home Keygen Token as follows:

Home Keygen Token = getFirst64Bits(HMAC_SHA1(Kcr, (HoA | Nonce | 0)))

where | denotes concatenation (joining end-to-end). The final "0" inside the HMAC_SHA1 function is a single zero octet, used to distinguish Home and Care-of Tokens from each other. Care-of Token is generated with CoA and ending octet "1":

Care-of Keygen Token = getFirst64Bits(HMAC_SHA1(Kcr, (CoA | Nonce | 1)))

In this calculation Correspondent Router uses the source address of the CoTI packet as the Care-of Address. The address may have changed in transit due to a NAT between the Mobile Router and the Correspondent. This does not effect the authentication because the Registration Request following this Return Routability procedure will contain the same Care-of Address so the Token is equal.

When the Mobile Router has received both Home and Care-of Keygen Token (in HoT and CoT) it is able to calculate Regisration Management Key and register with this key. The Mobile Router will establish a Registration Management Key KRm by default using SHA1 hash algorithm:

KRm = SHA1(Home Keygen Token | Care-of Keygen Token)

In deregistration (Registration Request with time set to zero), Care-of Keygen Token is not used. The Registration Management Key is generated with Home Keygen Token only:

KRm = SHA1(Home Keygen Token)

3.6.4 Registration management key updating

CR creates new a new Nonce once in every 30 seconds and it keeps the 8 newest Nonces in memory. This means that if MR gets Nonce Index 0 via HoT or CoT message, and it reregisters after more than 240 seconds, the CR does not have the nonce with index 0 in memory anymore. In this case CR sends Registration Reply with code Expired Home Nonce Index, Expired Care-of nonce Index or Expired nonces. In this case the MR has to calculate a new KRm via Return Routability procedure.

3.7 Summary of technologies

Mobile IP protocol with a few extensions provides all functionalities needed to implement RAIIC system. With Mobile IP signaling (Registration Requests and Replies) the node is able to switch the traffic onto another path. One Mobile Node is connected to several service providers' networks. If one connection fails, the node launches a handover by sending a registration via another service provider's network. Care-of Addresses are never visible outside the Mobile IP network so the Correspondent Nodes don't have to care which service provider the Mobile Node is currently connected to. They can always reach a Mobile Node via its Home Address.

NEMO extension turns the Mobile Node into a router. It enables the Mobile Router to offer connection to a whole set of networks. The hosts connected to the Mobile Networks only see a single, high-availability bit-pipe and are not required to support Mobile IP.

NAT traversal extension provides the ability maintain tunnels through NAT-devices and firewalls. It also contains a keepalive mechanism which makes the Mobile Router able to monitor the connections' states. If quality of the current connection deteriorates under a specified threshold, a handover is requested.

HAaRO extension offers Mobile Routers the ability to establish direct tunnels. Data traffic between Mobile Routers is transferred without the Home Agent forwarding. This distributes the traffic load more equally and prevents the Home Agent node becoming a bottleneck.

4 Implementation

This chapter describes our RAIIC implementation. We created the software by upgrading Dynamics project which is an open source Mobile IP implementation. We implemented a MIP stack that supports NEMO, NAT traversal and HAaRO extensions.

Section 4.1 describes how we ended up using Dynamics instead of creating the whole software from scratch. Section 4.2 goes through the functional modifications made on the original Dynamics software. Section 4.3 illustrates the most essential containers in the software. Section 4.4 sketches the socket interface implemented for the connections. Section 4.5 illustrates step-by-step what a router does when it gets launched.

4.1 From scratch or using existing code

The first task was to decide whether to build the functionalities on an existing Mobile IP implementation or to do the whole software from scratch. The goal was to implement Mobile IPv4 supporting co-located Care-of Addressing and extensions NEMO, NAT traversal and HAaRO.

4.1.1 Observing existing implementations

Route optimization has been included in Mobile IPv6 [16] from the start. One possibility was to take a Mobile IPv6 implementation and make it to run on IPv4 protocol. Some promising open source Mobile IPv6 implementations were found of which the most interesting one was MIPL - Mobile IPv6 for Linux [20]. The project was developed in Helsinki University of Technology in 2006. However building functionalities on an IPv4 software seemed more attractive than changing the whole core protocol from IPv6 to IPv4.

Only a few open source Mobile IPv4 implementations were found. The first option was a Java implementation of Mobile IPv4. The project was developed as a student's master's thesis in Bremen University. It is named UoB JMIP - Mobile IP implementation in Java [21]. Since the implementation does not support Care-of Addresses but only Foreign Agents, it was not our choice either. Also the version number of the software, 0.1, did not imply that the program was very thoroughly tested.

An older Mobile IPv4 implementation, Dynamics - HUT Mobile IP, supports Care-of Addresses. Dynamics was developed in Helsinki University of Technology in 2001. The source code is written in C language and it is available under GPL license on sourceforge [17]. We decided to build on this code because it included many useful functionalities and it seemed to be quite well tested. Dynamics supports co-located Care-of Addressing and message authentication. It also includes an API implementation for sending instructions to running routerÂ instances. The API is described more in Section 4.2.5.

4.1.2 Dynamics as a software to build on

Dynamics is a massive and complex package that includes no documentation of the software architecture. It has a lot of functionalities that are not needed in our implementation. Luckily the source code of some unwanted features, like Foreign Agent and Surrogate Home Agent, is conveniently isolated into separate sub-folders.

Figure 5 illustrates the sub-modules of Dynamics package. Also dependencies between the modules are shown. Dynamics consists of nine sub-modules of which three have been disabled during our development. Some of the major modifications implemented on the sub-modules are also listed in the figure. These modifications are discussed more in Section 4.2.

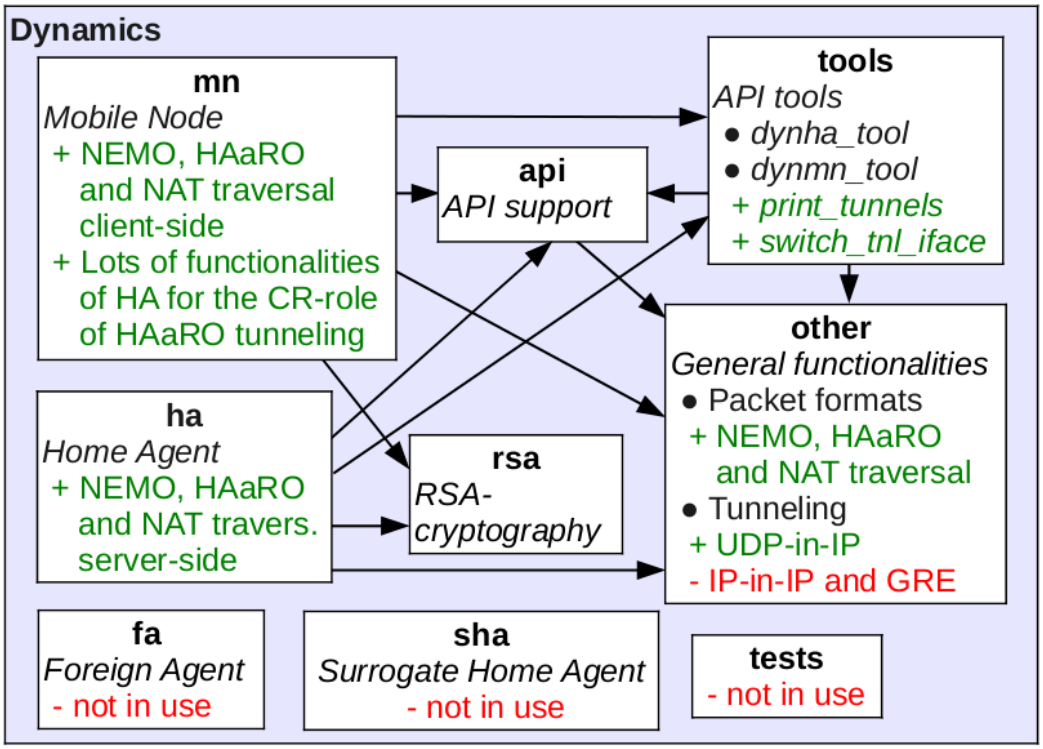

Figure 5: Dynamics software with its sub-modules, their dependencies and the main modifications implemented.

It took us quite some time to understand the implementation and learn how to develop it. It was challenging to debug such a massive software that had a lot of code we were not familiar with. However, it would have taken a lot more time to implement the whole Mobile IP protocol from scratch so it was clearly a better solution to use Dynamics.

4.1.3 Drawbacks of Dynamics' single thread design

The main task of such router process as Mobile Router or Home Agent is basically to wait for either a packet to arrive or a timer to expire, and to execute the corresponding tasks. In Dynamics, this has been implemented by passing all timers

and file descriptors of all listener sockets to *select* function. The function waits until a message is received or a timer is expired. When the function returns, the corresponding task is executed and the function is called again. So if a message is received during the execution of a task, the message is not handled until the task execution is finished. This way the router may become slow if lots of packets are transmitted.

Another possible way to implement a router is based on multi-threading. The most common action, which is handling bulk data, should happen as quickly as possible. All kind of background processing such as signaling could be forked to a new thread. This way the node could handle user data as quickly as possible and the user would experience the best possible service level. However, functionality testing was our main goal so this kind of performance optimization was not implemented.

4.2 Functional modifications implemented on Dynamics

Our implementation was built by upgrading Dynamics version 0.8.1 [17]. Some functional modifications have been made on the original software. This section describes the modifications implemented, the features added and the parts disabled.

4.2.1 Tunneling mode changed

Dynamics supports only the native Mobile IP tunneling modes which are IP-in-IP and GRE. Both of these are supported by the linux kernel. In our implementation they have been disabled and replaced with UDP-in-IP tunneling. This tunneling mode is not supported by the kernel so it is done in the user-space. The headers launching UDP tunneling mode are described in RFC 3519 - Mobile IP Traversal of Network Address Translation Devices [8].

UDP-in-IP tunneling is implemented using linux kernel's TUN interfaces [18]. TUN stands for network TUNnel. It is a virtual interface that simulates a network layer device. One TUN device is created for each tunnel. Packets sent by an operating system via a TUN device are delivered to the user-space program that created the device. The program can also send packets to the TUN device. The kernel handles these packets like any packets in its network stack. TUN devices appear as normal networking interfaces so they can also be added in routing tables. TUN technology is discussed more in Section 4.4.1.

4.2.2 Foreign agent disabled

Foreign Agent is implemented in Dynamics. The user is able to choose whether to use Foreign Agents or co-located Care-of Addressing. Network Mobility extension [8] works only with co-located Care-of Addressing, so we naturally used only that during the development. Therefore the Foreign Agent execution mode is no longer supported in our implementation.

4.2.3 Added support for Network Mobility and Route Optimization

Support for Network Mobility (NEMO) and Home Agent assisted Route Optimization (HAaRO) extensions was implemented. Mobile Node supporting NEMO is able to operate as Mobile Router. Now Correspondent Nodes no longer communicate with the Mobile Node's Home Address but with the hosts on the Mobile Networks. This requires Home Agent to have all networks of a Mobile Router in its routing table, pointing to the tunnel towards this router. Without NEMO, only Mobile Nodes' Home Addresses are pointing to the tunnel devices.

Nodes supporting HAaRO can create direct tunnels between each other. This required implementing inter Mobile Router registration process, so some routines from the Home Agent side had to be included also on the Mobile Router side. Such routines are Registration Request handling, Registration acknowledging, Authentication extension handling, NEMO Request handling, NEMO acknowledging etc. Also Route Optimization Cache had to be implemented on the Mobile Router side. Mobile Router's Route Optimization Cache is similar to Home Agent's Binding Table. Mobile Router maintains information about its direct tunnels in this container.

4.2.4 Virtual Home Addresses instead of Home Network

In our case, Mobile Routers are not really mobile. They only switch between certain service providers' networks. Home Agent is never on the same network so no Mobile Router is ever on its Home Network.

Therefore our implementation uses virtual Home Addresses. When a Mobile Router gets launched, it immediately registers to its Home Agent with Home Address set to 0.0.0.0. The Home Agent allocates the next available virtual Home Address from its virtual Home Address pool. When the router performs deregistration, the Home Agent frees the address so that it can be allocated for another router.

4.2.5 Added two API commands to dynmn_tool

Dynamics package has an API tool named *dynmn_tool* for managing the locally running Mobile Router process. With this tool the user is able to perform manual location updates, use wireless protocol extensions, make the node to rescan networking interfaces of the system etc. The rest of the features are mostly related to Dynamics' own tunneling system and the Foreign Agent.

We added two commands to the API: *print_tunnels* and *switch_tnl_iface*. The former prints information about the tunnels created towards the Home Agent and other Mobile Routers. Information about both the active and the passive tunnels is printed. This command can be executed for example by giving it as a parameter to the *dynmn_tool* process:
src/tools/dynmn_tool print_tunnels.

The other instruction implemented - *switch_tnl_iface* - binds a tunnel to a new networking device. This is handy in testing load balancing which will be implemented as a feature of the software in a future release. The *switch_tnl_iface* instruction takes

two additional parameters: Home Address of the tunnel's destination node and the name of the new networking device name. For example tunnel from a Mobile Router to a Correspondent Router with Home Address *172.31.0.3* would be bound to networking device named *eth1* by executing:
src/tools/dynmn_tool switch_tnl_iface 172.31.0.3 eth1

4.3 Containers

This section describes the implementation of the main containers. These are Home Address pool, Binding Table, Route Optimization Cache and the containers for packet handling.

4.3.1 Home Agent's Home Address pool

Home Agent allocates Home Addresses (HoA) for Mobile Routers from its virtual Home Address pool. A virtual Home Address is sent in Registration Reply to each Mobile Router registering with a valid request and the header's Home Address field set to 0.

Home Address pool is implemented in *src/ha/ha.c* with an array named *allocated_hoas*. All the Home Addresses that are no longer free to allocate are stacked in this array. At first this array is initialized with function *init_hoa_pool*. This function sets all Home Address slots in the array to *0.0.0.0*, which in this case means "free to allocate". An address is allocated from the HoA pool by calling function *get_hoa_from_pool*. If the requested address is not found from *allocated_hoas*, it is allocated for the Mobile Router and added to this array to indicate that it is taken. An address is freed by calling function *free_hoa* which sets the address to *0.0.0.0* in the array.

4.3.2 Home Agent's Binding Table

Home Agent caches information about all the (successfully) registered Mobile Routers in its Binding Table. This table associates Mobile Routers' Home Addresses with their current locations: Care-of Addresses. When a Mobile Router registers, a new entry is created in this table. When the node performs a handover and registers from the new Foreign Network, the entry of the node is modified. When the node deregisters or its binding expires, its entry is removed from the table.

Binding Table is implemented as a list of structure instances. The structure is defined in file *src/other/binding.h*. The Binding Table list is initialized and handled in *src/other/binding.c*. The table handling interface includes functions for example for adding, removing and fetching entries. Entries can be fetched by Care-of Address, Network Access Identifier etc.

4.3.3 Route Optimization Cache

Route Optimization Cache (RO Cache) is an array maintained by the Mobile Router. It contains an entry for each known Correspondent Router which supports Route

Optimization. RO Cache (*ro_cache*) is an array of type *struct ro_entry* defined in file *src/mn/mn.c*. Type RO entry (*struct ro_entry*) is defined in file *src/mn/mn.h* and it contains the following information:

- HoA of the CR (*cr_hoa*)
  - identifies the entry
- TUN device (*tun_dev_name*) of the tunnel towards the CR
- tunnel destination CoA (*tnl_dst*)
- all known CoAs of this CR (*cr_coas*)
- Mobile Networks reachable via this CR (*cr_nws*)
- local networking devices (for handovers) available for tunneling to the CR (*mr_coas*)
- etc.

RO Cache entry states and triggers Entries in the RO Cache are associated with different states during the execution. These states are represented by flags (integer values) switched on and off in different situations. For example after sending Registration Request to a Correspondent, the RO Cache entry of this CR is put to state "waiting for Registration Reply from this CR". This state is associated with a timer so the reply is not waited forever.

RO Cache entries also contain triggers. If a trigger is set, an action is done as soon as the trigger checking function gets called in the main loop. For example when new Registration Management Key is needed for a CR, "start Return Routability" trigger is set in its entry.

Let's illustrate the states and triggers with an example where a router needs to be solicited before accepting its registration. When Registration Request is received from an unknown Correspondent Router, a new RO Cache entry is created. As the CR has not been confirmed by the Home Agent, flag *confirmed_by_ha* is set to 0. The request is parsed and Solicitation Request is sent to the Home Agent. In case the Home Agent verifies the Mobile Router, flag *confirmed_by_ha* and trigger *send_regrep* are set. Triggers are monitored in every iteration of the main loop before entering the select function. State of the trigger launches Registration Reply creation.

4.3.4 Headers and packets

Message header formats are defined as *structs* in file *src/other/message.h*. Unsigned data types of different lengths represent the fields of the headers. For example *__u8* is one byte data block.

When a packet gets received, it is parsed into a variable of type *struct msg_extensions*. This structure (defined in *src/other/msgparser.h*) contains mostly pointers that are set to point to different parts of the message buffer. Observe that a variable of this

type does not contain the data itself. When handling a packet via these pointers, the data buffer needs to be in the memory.

By this datatype it is easy to see in which parts of the code a certain header is handled. You can check which field in *struct msg_extensions* represents the header and find all references of that field. This can be done for example with *grep* string finder or with the *references* tool of Eclipse.

4.4 Socket interface

Sockets are used as communication endpoints. Mobile Router communicates with other MIP-routers (CR, HA), hosts in Mobile Networks, API-tools and the kernel's TUN devices. Connection-oriented sockets (TCP) are not used in this implementation so we only discuss connectionless sockets in this chapter.

There are two types of connectionless sockets: outbound sockets (for sending) and listener sockets (for receiving). Listener is bound to a certain port of a certain IP address with *bind* function. Messages sent via a socket bound to *10.10.10.10:55555* have source address *10.10.10.10* and source port *55555*. Also, if this socket is monitored in the main loop, all packets destined for *10.10.10.10:55555* are received in the software via this socket. Outbound sockets are not required to be bound to any port. Random ports are allocated for the unbound sockets.

There is also a possibility to bind a socket to a networking device. The socket option enabling this is *SO_BINDTODEVICE*. This option is handy with handovers as you can strictly control where the packets are routed. If a socket is bound to the IP address of for example *eth0* but a packet is sent to a destination where the kernel's best route goes out *eth1*, the packet will go out *eth1* with the source IP address of *eth0*. This would mess up the routers' ISP switching system so *SO_BINDTODEVICE* is used in binding Mobile Routers' CoA sockets to the routers' networking devices.

Main sockets of the routers are described in the following sections. Section 4.4.1 discusses the file descriptor TUN FD which is used for communicating with the TUN device. Section 4.4.2 illustrates how Home Agent handles connections. Section 4.4.3 lists the sockets utilized on Mobile Router's side.

Sockets discussed in these sections are shown in Table 2 (TUN FD), Table 3 (HA) and Table 4 (MR). These tables illustrate the purpose, the interface and the port of each socket. Some sockets are not bound to any port but to the interface with socket option *SO_BINDTODEVICE*.

4.4.1 Tunneling file descriptors (tun_fd)

Tunneling file descriptor (TUN FD) is a channel between a software and the kernel's virtual tunneling (TUN) device created by the software. TUN FD is received from the kernel when creating a TUN device. The software sends messages to a TUN device via its TUN FD and vice versa. One TUN device is created for each tunnel and each TUN device has a TUN FD. Traffic received from the tunnel is written

Table 2: TUN device socket: TUN FD.

Purpose	**Interface**	**Port/Device**
Communicate with the TUN device	TUN device of the corresponding tunnel	*SO_BINDTODEVICE*

into the tunnel's TUN FD. Traffic going out to the tunnel is received from the TUN FD.

Incoming tunnel traffic A tunnel message is first received into a Care-of Address socket bound to a physical networking device, for example *eth0*. A tunnel message is recognized by its outer protocol stack: MIP TNL on UDP. MIP TNL header is removed and the following IP datagram is sent to the corresponding TUN FD. The corresponding TUN FD can be solved from the address that the UDP packet was received from (the outer IP packet). This address is the Care-of Address of one of the known routers (HA or MR). The kernel forwards the IP packet to the corresponding device.

Figure 6 illustrates how Mobile Router forwards a tunnel message it received to the corresponding TUN device. In this case a message was received on a socket bound to a networking device. At first, it is checked if it is a signaling packet or a tunnel message. If the packet came from a tunnel, the sender needs to be identified. The outermost source IP address is compared with the Home Agent and all the Correspondents in the Route Optimization Cache. If a match is found, the outer header stack (MIP in UDP in IP) is removed and the remaining IP packet (inside the tunnel) is sent to the corresponding TUN FD.

Outgoing tunnel traffic The TUN FDs also need to be actively monitored in the *select* function of the main loop. If a message is received in a TUN FD (sent by the kernel), MIP TNL header is added and the packet is sent as UDP to the corresponding router's Care-of Address (tunnel endpoint). More detailed information about the TUN/TAP technology can be found from the TUN/TAP interface tutorial [18].

Figure 7 illustrates how a packet sent out to the tunnel is handled by Mobile Router. Mobile Router monitors all TUN FDs it has created. If a packet is routed to a TUN device, it ends up in the TUN FD of that device. Mobile Router reads the packet from the TUN FD, adds MIP TNL header outside the packet and sends it on UDP to the Care-of Address of the CR the tunnel was created for.

4.4.2 Home Agent side connection sockets

Home Agent has a socket bound to each of its interfaces. Sockets are bound to the MIP signaling port 434. In our case all Mobile Routers register to the same interface (same IP address) of the Home Agent.

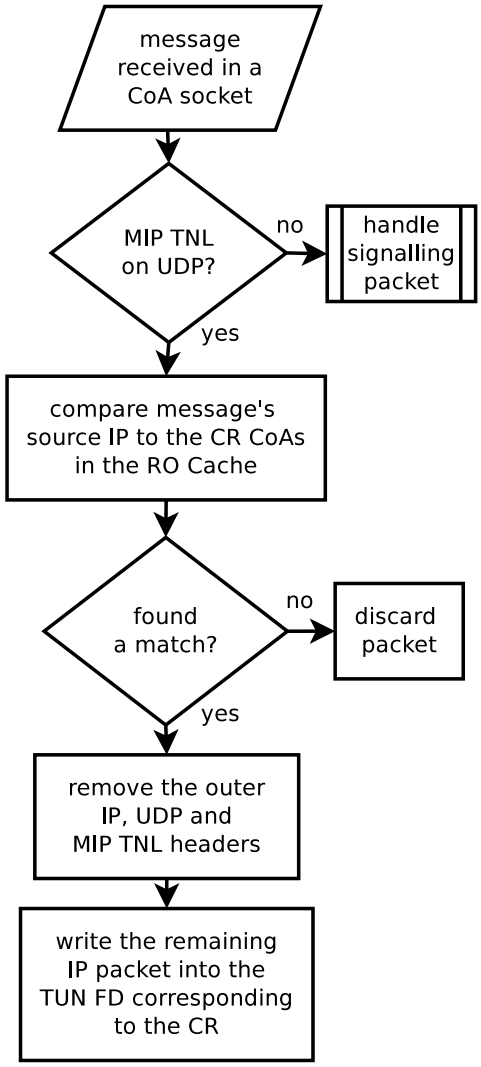

Figure 6: Mobile Router forwards a tunnel message to a TUN device.

Table 3: Home Agent's MIP sockets.

Purpose	**Interface**	**Port/Device**
Communicate with the Mobile Routers	One per each CoA	port 434

Home Agent handles all communication with a Mobile Router via the socket the initial registration was received from. Communication includes receiving Registration Requests and incoming tunnel messages and sending Registration Replies and outgoing tunnel messages.

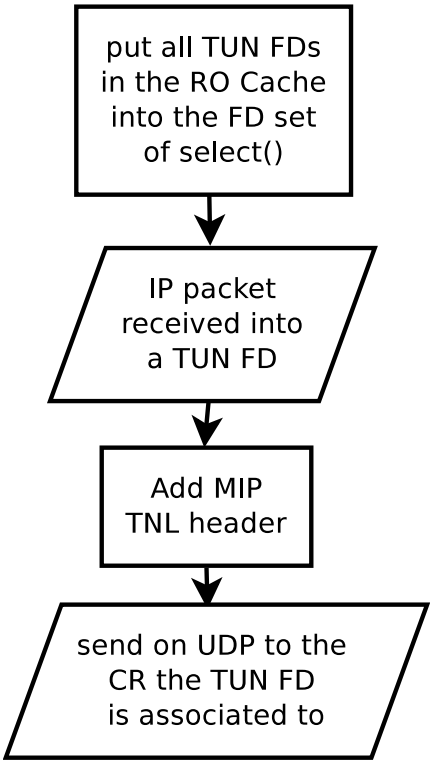

Figure 7: Mobile Router sends a packet from the TUN device to the tunnel.

Table 4: Mobile Router's connection sockets.

Purpose	**Interface**	**Port/Device**
Registration to HA	CoA	*SO_BINDTODEVICE* (random port)
Registration to CR	CoA	*SO_BINDTODEVICE* (random port)
Send CoTI and receive CoT	CoA	*SO_BINDTODEVICE* (random port)
Tunneling as "CR"	CoA	port 434
Send HoTI and receive HoT	HoA	random port
Receive HoTI and send HoT	HoA	port 434

4.4.3 Mobile Router side connection sockets

Mobile Router has a socket towards the Home Agent and several sockets to use for communication with Correspondent Routers.

Socket towards Home Agent Mobile Router registers to its Home Agent and sends outgoing tunneling traffic via *registration_socket* defined in *struct mn_data*. Registration replies and incoming tunneling traffic are also received into this socket. If the Home Agent connection fails, this socket is bound to the next networking device and a new Registration Request is sent with the new CoA.

Sockets for registering to Correspondent Routers Mobile Router creates one registration socket for each entry of Route Optimization Cache. Sockets are bound to the networking devices. One of these sockets is used when registering to a Mobile Router (Route Optimization). Tunneling traffic is also transfered via the registration socket in case the tunneling role in the connection is "MR".

Sockets for Care-of Test Initializations (CoTI) There is also a CoTI socket for each entry of the Route Optimization Cache. CoTI messages are sent and CoT messages received via this socket. The socket is bound to a networking device the CoTI is to be sent from.

Tunnel traffic when tunneling as "CR" Mobile Router's each networking device (CoA) has a socket bound to the MIP signaling port (434). Tunneling traffic is handled via these sockets if the tunneling role of an inter Mobile Router connection is "CR".

Sockets bound to the Home Address HoTI messages are sent and HoT messages are received on a socket named *hoa_socket* defined in *struct mn_data*. This socket is bound to the Mobile Router's Home Address, to a random port.

A socket named *hoa_listener_socket* is bound to the MIP signaling port (434) of the Home Address. HoTI messages are received and HoT messages are sent on this socket.

4.5 Execution states

This section illustrates the execution of the Home Agent and Mobile Router processes. The execution of a Mobile IP router can be divided in three different states: initialization, packet handling and timer expiration handling. In the first state the node is initialized by loading configuration, reading command line arguments, setting timer expiration times etc. Mobile Router directly connects to the Home Agent it finds from its configuration file by sending Registration Request.

After the initialization the execution jumps into *select* function to wait either for a packet to arrive or for a timer to expire. The packet arrival state is entered when a packet arrives in a socket that is listened to in the select function. In this state the incoming packet is handled. The timer expiration state is entered when a timer - given to select as parameter - expires. The expiration of a timer triggers an action such as sending a keepalive ping, updating a binding etc.

4.5.1 Initialization

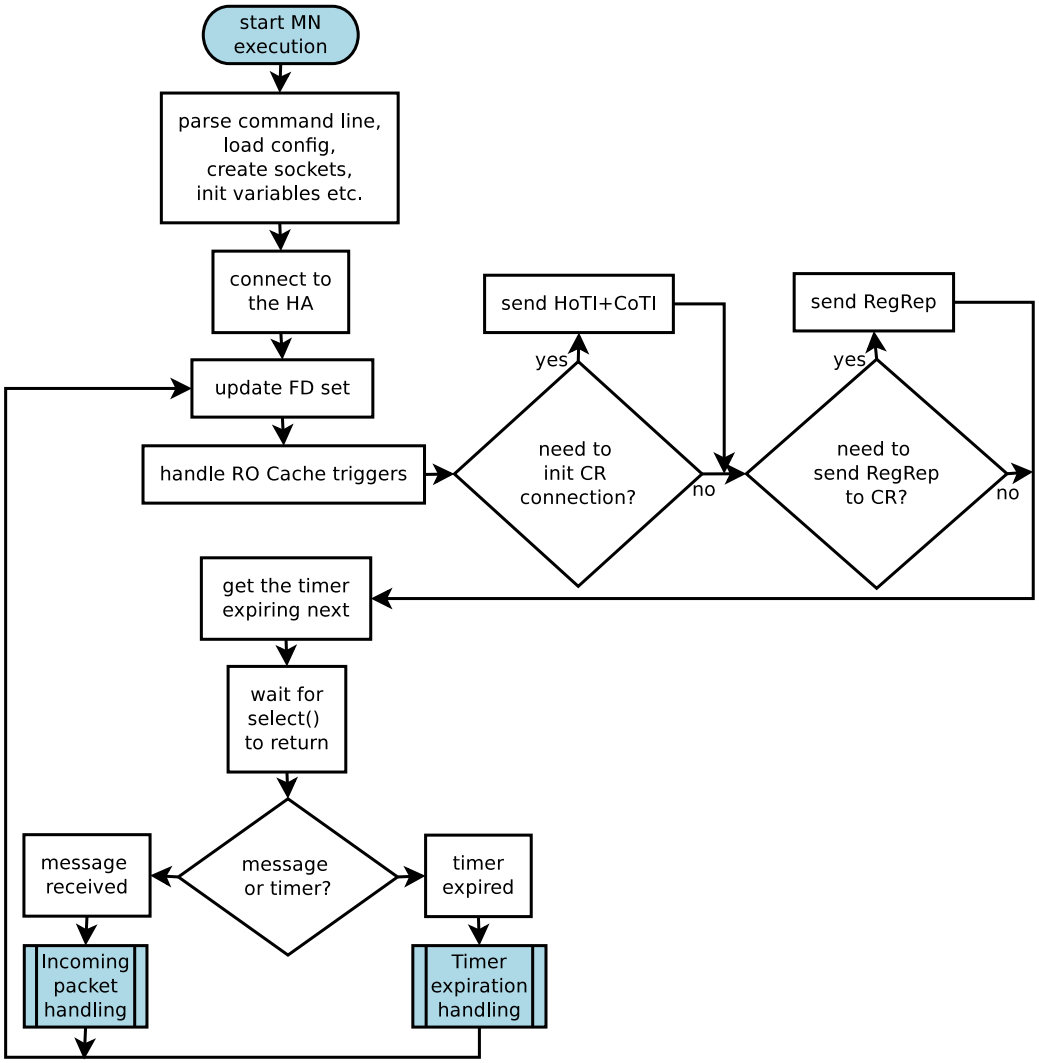

Figure 8: Flowchart of the initialization state of the Mobile Router process.

Execution of a router starts by going through the first state, initialization. In this state the node is initialized with values given by the user. At first, the command line and the configuration file are parsed and global run-time variables are set to obey the current configuration. Also the socket interface is created to enable communication between the routers.

In Mobile Router's case, the execution of the node starts by connecting to the Home Agent. Its IP address is read from the configuration file, and Registration Request is sent towards this address. When the registration is successful, the main loop is entered.

Execution of the loop starts by updating the file descriptor set. Also Route Optimization Cache triggers are checked. Each entry (Correspondent) in the RO Cache has integer variables which trigger certain actions if they are set to a non-

zero value. For example sending HoTI, CoTI and Registration Reply messages to Correspondents are launched via these triggers.

In the main loop there is a *select* function call. Two parameters are passed to the select function: the file descriptor set and the timer that will expire next. Before calling select function, the timer expiring next is fetched from the timer array.

Select returns when either there is a message in a file descriptor or the timer expired. If a message is received, the execution steps into the *incoming packet handling* routine. If the timer expired, the execution jumps into the *timer expiration handling* routine. Tasks and conditions in the initialization state are illustrated as flowchart in Figure 8.

Home Agent process launch is similar with the Mobile Router, except that Mobile Router starts directly requesting registration from the Home Agent and Home Agent simply remains waiting for routers to register. Home Agent also maintains Binding Table instead of RO Cache so there are no triggers on the Home Agent side.

4.5.2 Incoming packet handling

Incoming packet handling routine starts when a message is received in a file descriptor. Different sockets handle different types of messages. The first byte of the message indicates the type. Figure 9 illustrates the execution paths of the incoming packet handling routine on the Mobile Router side. The path executed depends on the socket the message was received in along with the type of the message. The boxes with blue frames represent the listener sockets of the Mobile Node.

Mobile Node sends Registration Requests to the Home Agent via *HA registration socket*. This socket is also used for receiving Registration Replies or Tunnel Data Messages sent by the Home Agent. The first byte of the MIP-message indicates whether the message is Registration Reply or Tunnel Data Message. The message is parsed according to the packet format. Data in the message is either forwarded to a customer-host (tunnel message) or parsed locally (registration message signaling). After receiving, parsing and handling the message, the execution is taken back to the beginning of the main loop.

If a message is received in *API socket*, the user is requesting for some information about the node via the API interface. It may concern the status of the node, its tunnels, configuration etc. The request is replied and the execution is continued back to the beginning of the loop.

RT Netlink monitor socket (*RT Netlink socket*) being active indicates that the networking interface configuration of the system has changed. In this case the router checks if networking interfaces have been added or removed. The networking device list of the software is updated to correspond to the configuration of the operating system and the execution is returned to the loop.

Dynamics has an ICMP socket listening to agent advertisements. In our implementation Agent Advertisements are never received because the Mobile Routers have virtual Home Networks and there are no Foreign Agents. The Mobile Routers still have the ability to handle Agent Advertisements and there is a socket (*ICMP socket*) listening to these messages. In further development this feature can be

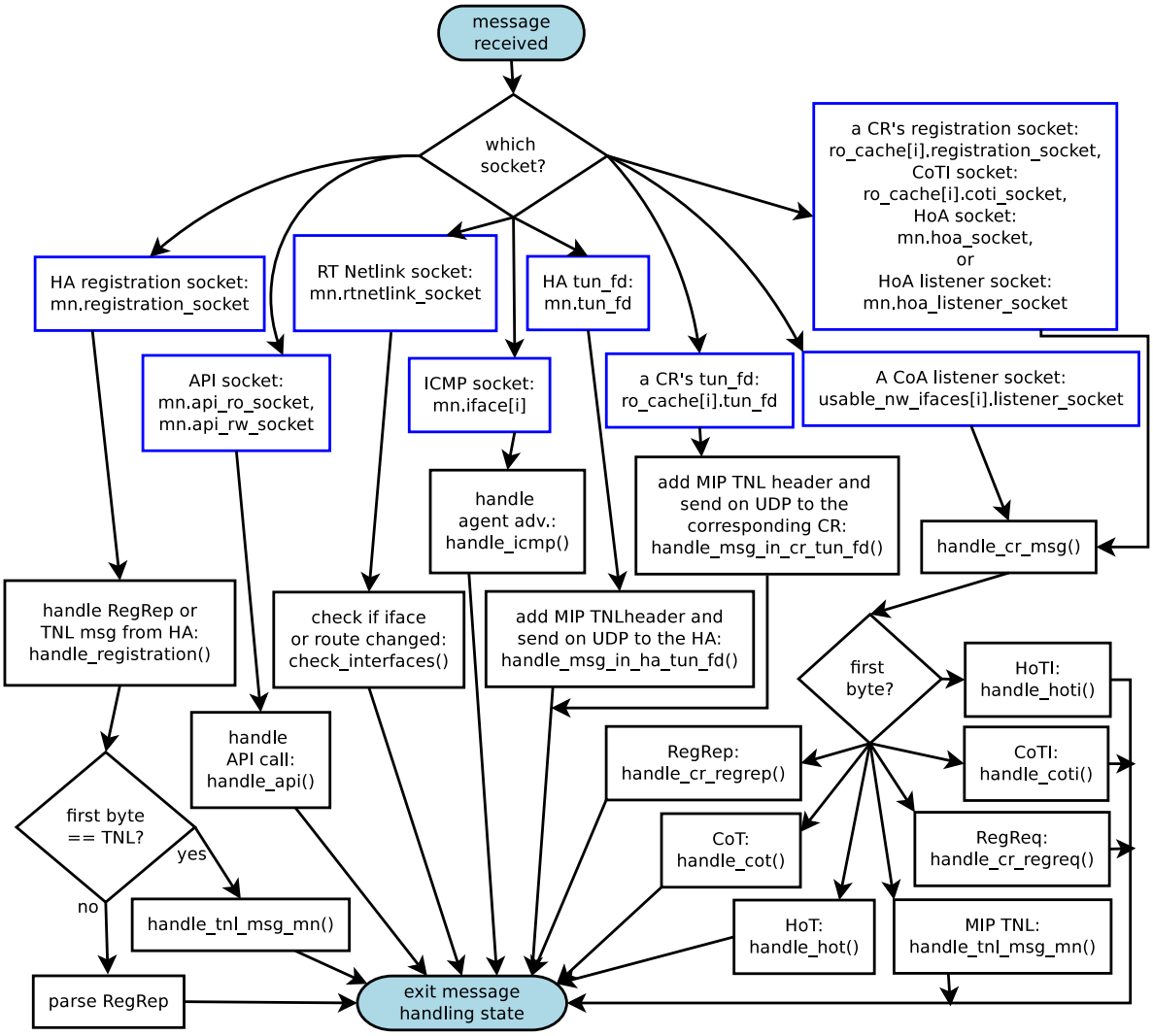

Figure 9: Flowchart of the incoming packet handling state of the Mobile Router process.

disabled in order to clean the code, improve performance etc.

Mobile Router establishes a tunnel to its Home Agent and possibly some direct tunnels towards Correspondent Routers. There is a tunneling socket for each tunnel (*HA tun_fd*, *a CR's tun_fd*). When a message is received in one of the tunneling sockets, the packet is encapsulated with MIP TNL header and sent to the tunnel endpoint. Functionality of the tunneling file descriptors is discussed more in Section 4.4.1

Mobile Router has one socket per each Care-of Address (*CoA listener socket*) bound to the MIP port (434). These sockets serve as direct tunnel endpoints when the node is tunneling as Correspondent (tunneling roles explained in Section 3.5). Keepalive pings and Tunnel Data Messages are received from Mobile Routers via these sockets.

Mobile Router performs Return Routability process before registering to a Cor-

respondent. It sends CoTI and receives CoT via *CoTI socket*. HoTI is sent and HoT is received via *HoA socket*. When Registration Management Key has been calculated, Registration Request is sent and Reply is received via one of the *CR's registration sockets*.

4.5.3 Timer expiration handling

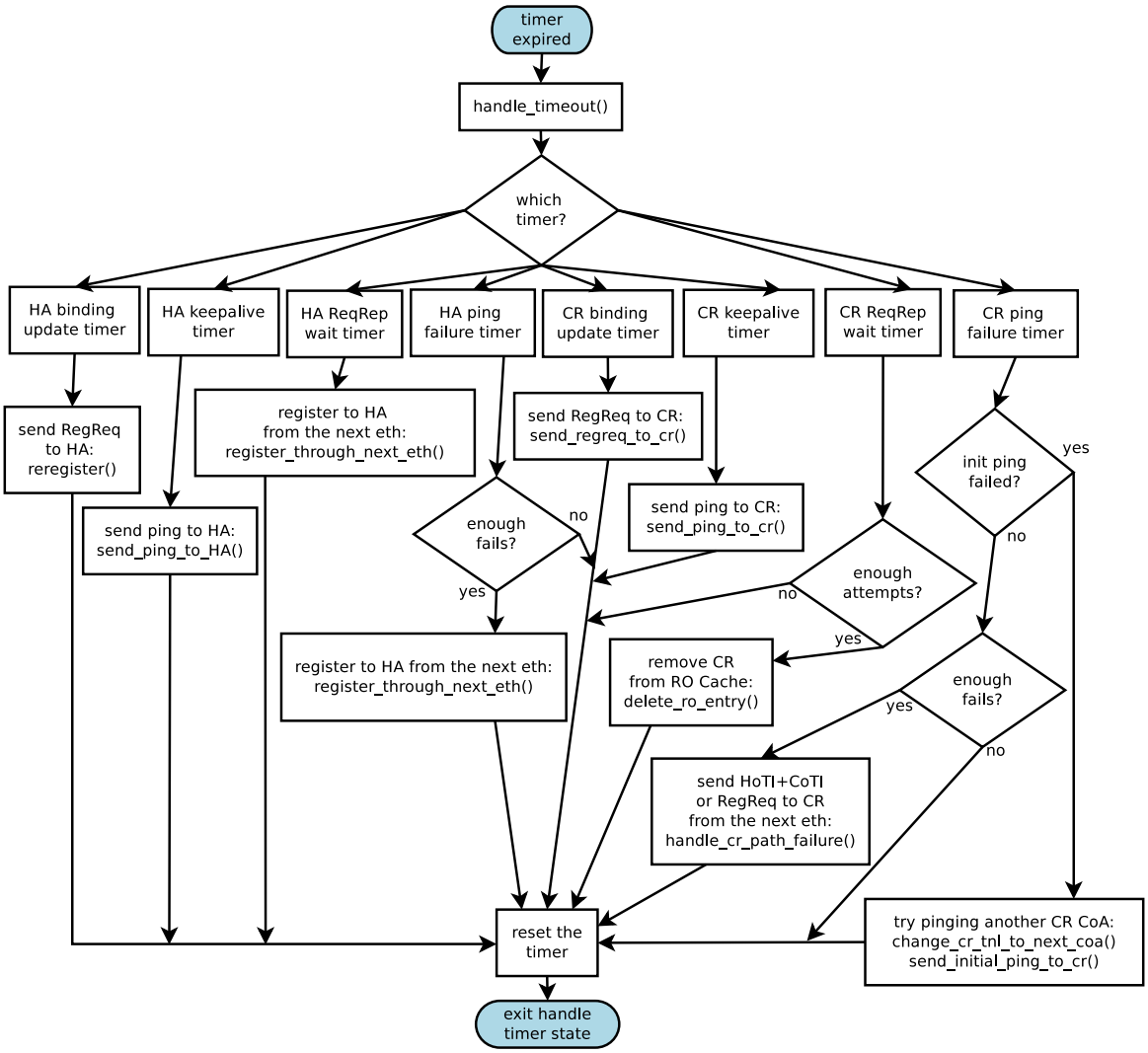

Figure 10: Flowchart of the timer expiration handling state of the Mobile Router process.

Timer expiration handling begins if *select* function returns 0. This value indicates that the timer given to select as parameter got expired before any message got received. At first it is examined, which timer it was that got expired and the corresponding routine is executed. Finally the timer is reset to expire after the time interval associated with the timer. Figure 10 illustrates the execution paths of the timer expiration handling routine. The path executed depends on which timer got

expired. These timers are discussed in this section.

Home Agent informs Mobile Router about the amount of time that the registration is valid for. Mobile Router sets *HA binding update timer* to expire when a new Registration Request needs to be sent. In direct tunnel bindings there is similar logic. There is one *CR binding update timer* for each Correspondent tunnel where the tunneling role is Mobile Router.

When Registration Request is sent there is a timer waiting for the reply to arrive. If the request gets lost in the way, this timer expires and a new Registration Request is sent. There is *HA RegRep wait timer* and one *CR RegRep wait timer* for each Correspondent. If several requests to a Correspondent get lost, the corresponding entry is removed from the RO Cache.

Each tunnel needs to be kept alive so that firewall or NAT possibly on the way does not close the connection. There is *HA keepalive timer* to ensure that keepalives are sent frequently enough to the tunnel towards the Home Agent. There is also *CR keepalive timer* for each direct tunnel where the tunneling role is Mobile Router.

ICMP Echo Requests are used in direct inter Mobile Router tunneling for two purposes: as initial MR CoA to CR CoA connectivity test and as tunnel keepalive packets. When an ICMP Echo Request is sent, a timer is set to wait for the ICMP Reply. For this purpose there is one *CR ping failure timer* for each Correspondent. There is also *HA ping failure timer* for the keepalives of the HA tunnel.

The action following a lost ICMP Echo Request depends on the purpose of the packet. If it was a CoA to CoA connectivity test, a new ICMP Echo Request is sent to another CR CoA. If it was a tunnel keepalive, a new keepalive is sent to the tunnel. The user has configured the maximum amount of keepalive failures tolerated. In case this amount of keepalives have not been replied, the path is interpreted to be broken and a handover to another networking device is performed.

5 Usage and maintenance

This chapter describes how to use and develop the software. The software consists of preparation scripts, source code, compilation scripts and configuration files (text format). Section 5.1 includes very brief instructions how to get the program running. Section 5.2 describes how the software execution can be configured. Section 5.3 describes debugging routines used during the development process.

5.1 Getting started with the program

After decompressing the package it needs is prepared for the environment, compiled and executed. The commands in this chapter are to be run in the root folder of the package.

5.1.1 Preparation and compilation

Start by executing the configuration script. Surrogate Home Agent and Foreign Agent nodes are not used in this project so they can be left out:
./configure –without-sha –without-fa

The *configure* script checks if the environment is missing some components. For example GMP library is required. This can be installed on Debian with the following command:
apt-get install libgmp3-dev

After installing the missing software and running configure without errors, the software can be compiled by executing:
make

...or it can be installed with command:
make install

5.1.2 Execution

The program needs to be run as root because it modifies routing tables and uses ports with number less than 1024. For example the Mobile IP signaling port is 434. Compilation of the software package creates multiple executables into different folders. The following executable files got modified during this project:

- *src/ha/dynhad* (Home Agent executable)
- *src/mn/dynmnd* (Mobile Router executable)
- *src/tools/dynmn_tool* (a tool to control the Mobile Router currently running)

Executables dynhad and dynmnd Executable *dynhad* is the Home Agent and *dynmnd* is the Mobile Router. In normal Mobile IP network there is one Home Agent and multiple Mobile Routers. Bash script *run_haaro.sh* was added in the package in order to make the it easier to run the executables in different development modes. It's a simple switch-case script that takes one of the following arguments: *ha*, *ha-debug*, *mn*, *mn-debug* or *help*. All the execution modes do the following:

- run in the terminal without making the process a daemon
- read configuration from the local configuration file:
  - *src/ha/dynhad.conf* (Home Agent)
  - *src/mn/dynmnd.conf* (Mobile Router)
- print the execution traces into two locations:
  - *stdout*
  - file named *ha_traces.txt* (Home Agent) or *mn_traces.txt* (Mobile Router)

If a mode is run with *-debug* extension (*ha-debug* or *mn-debug*) - in addition to the traces - also debug information is printed into a file named *ha_debug.txt* (Home Agent) or *mn_debug.txt* (Mobile Router). This execution mode runs a bit slower so it should not be used when performing measurements.

For example Mobile Router in debug mode executes *src/mn/dynmnd* by reading configuration from *src/mn/dynmnd.conf*. Debug information is printed into *mn_debugs.txt* and execution traces into both *mn_traces.txt* and *stdout*. The command is the following:
sh run_haaro.sh mn-debug

Executable dynmn_tool When Mobile Router (*dynmnd*) is running, it can be monitored and controlled by sending it commands with the executable *dynmn_tool*. For example, it is possible to print its current tunnels by executing:
src/tools/dynmn_tool print_tunnels

A tunnel can also be binded to another networking device. For example the following command would bind the tunnel going towards the Correspondent having Home Address *172.31.0.3* to interface *eth1*:
src/tools/dynmn_tool switch_tnl_iface 172.31.0.3 eth1

5.2 Configuration

This section describes how to configure the execution. The execution can be adjusted by modifying command-line parameters, values in the configuration files or defined preprocessor variables (*#define*).

5.2.1 Command line parameters

Home Agent and Mobile Router executables take command line arguments. During the development, the nodes were run with the following parameters:

- *fg*: runs the process in the terminal without making it a daemon
- *traces*: prints execution traces which describe the state of the execution
- *config*: reads configuration from the file defined

Here is the full command used when running the Home Agent:
./src/ha/dynhad –fg –traces –config src/ha/dynhad.conf 2>&1 | tee ha_traces.txt

5.2.2 Configuration files

When a node gets launched it reads a configuration file in addition to the command line. The program reads the file and looks for values following certain keywords. Lines starting with hash character (#) are ignored.

If for example the Mobile Router switches to another link too easily, the amount of lost keepalives tolerated can be changed by modifying the number in the following line:
Max_lost_keepalives 3

In this line the Mobile Router is instructed about how many keepalives can get lost before the connection is switched onto another link.

5.2.3 Defined constants

In the code, certain constants are defined as preprocessor variables (*#define*) in order to make them easy to modify. For instance, HAaRO message type numbers were not yet confirmed when this software was developed. They need to be updated in the code when they get confirmed.

Message types are defined in *src/other/message.h*. Mobile Router related constants such as nonce creation interval and Registration Reply waiting time are defined in *src/mn/mn.h*. Home Agent related constants like maximum amount of bindings and default listener port are defined in *src/ha/ha_config.h* Maximum message size is defined in *src/mn/mn.h* (MR) and in *src/ha/ha_config.h* (HA).

5.3 Debugging tools and routines

This section describes how the software was debugged during the development. The aim is to help the next developers with their issues. The system being distributed makes the development more complicated. In addition to basic bugs in the software,

an error can lie in several different locations such as Linux configuration or format of a packet generated.

Software execution logic and memory accessing was monitored with Gnome Debugger (GDB). Section 5.3.1 discusses this kind of debugging. Section 5.3.2 describes how certain attributes in the Linux configuration were verified to be suitable for the software execution. Section 5.3.3 illustrates how packet formats and contents were monitored. Section 5.3.4 describes how outages were caused manually on the infrastructure network. Breaking down paths enables testing if the software detects breakdown and finds a working path.

5.3.1 Monitoring execution of the software with Gnome Debugger

The easiest fault to locate is an error in execution of the local router instance. When the node acts strangely or crashes for a segmentation fault, the situation should be repeated under Gnome Debugger (GDB).

Before executing on GDB, limitations on shell processes should be removed by executing:
ulimit -c unlimited

Also compilation should be done in debug mode. Compilation mode can be changed by modifying the *Makefile* in the root folder of the project. In the Makefile, *-g* flag should be added to the *CFLAGS* variable:
CFLAGS = -O2 -g

GDB can be launched by giving it the executable as command line parameter. For example the Home Agent executable was debugged in folder *src/ha/* with command:
sudo gdb dynhad

In GDB a breakpoint can be set at a certain line with command (following the *(gdb)* label):
(gdb) break ha.c:2267

Or at a function:
(gdb) break ha.c:handle_reg_msg

The program is launched with *run* command and parameters:
(gdb) run –fg –debug –config dynhad.conf

On GDB the process can be stopped by sending it signal SIGINT (ctrl+c). It may also crash. The last function calls of a stopped or crashed execution can be back-traced with command *bt*. The execution can be followed line by line with command *n* (next) and jump inside a function with command *s* (step). Variables' current values can be printed with *p <variable name>*. You can jump into the caller function with command *up* and back to the callee with *down*.

5.3.2 Checking Linux routing configuration

Incorrectly configured routing table or Reverse Path Filtering option may prevent the software from working properly. This can for instance cause a situation where no packet leaves the interface even though the software seems to be sending packets correctly.

Routing table Mobile Router has to have at least one interface for its Mobile Network(s) and one interface for each wide-area network (ISP). When the router is running, there are also devices for each tunnel. Routes to a Correspondent's Mobile Networks point to the tunnel towards the Correspondent. You can see network interface statuses by executing:
ifconfig

You can create a new networking interface *eth2* with IP address *10.10.10.10*:
ifconfig eth2 10.10.10.10 netmask 255.255.255.0 up

Routes can be monitored and managed with command *ip*. The following prints the routing table:
ip route

Current route to host *11.11.11.11* is shown with:
ip route get 11.11.11.11

Route to host *11.11.11.11* is added via device *TUN1* with command:
ip route add 11.11.11.11 via TUN1

Default route via *20.20.20.20* is deleted with command:
ip route del default via 20.20.20.20

See the full manual:
man ip

Reverse Path Filtering One parameter in Linux configuration which caused us confusion was Reverse Path Filtering (RPF). This feature is used for example to prevent IP source address spoofing. RP-filter checks that source IP addresses of packets received on a networking interface match the rules set in the routing table.

Let's take an example. A router has two networking interfaces:

- *eth1*
  - address: *10.10.10.10*
  - routes: *10.10.10.0/24*
- *eth2*

- address: *20.20.20.20*
- routes: *20.20.20.0/24*

The router has RP-filtering enabled for *eth1*. A packet on *eth1* from source address *20.20.20.5* will be blocked because its route belongs to *eth2*.

In this project RP-filter should be disabled for all interfaces utilized by the router. The status of the filters on networking interfaces can be checked by executing command:
sysctl -a | grep rp_filter

RP-filtering can be disabled for *eth0* with command:
sudo sysctl -w net.ipv4.conf.eth0.rp_filter=0

5.3.3 Monitoring packet formats with Wireshark

An outgoing packet can contain invalid fields causing the kernel discarding it. Packets received by a networking device can be monitored with Wireshark. With this graphical tool it is easy to follow connection initialization, connection maintenance and disconnection packets.

One confusing situation we came up against was a tunnel inside of a tunnel. Tunnel packet was routed to a tunneling device two times instead of one. These kind of bugs would be hard to notice without observing the actual bits online.

Figure 11 illustrates how packets are shown in Wireshark. There is a tunnel keepalive (ICMP Request) sent from a Mobile Router with HoA *172.31.0.2* to a router with HoA *172.31.0.3*. The keepalive packet is in UDP tunnel so it is encapsulated with MIP in UDP in IP stack.

5.3.4 Testing how the node finds a working path

Path fails if a link on the way is down or some part of the infrastructure is broken. Mobile Router is supposed to notice a problem in connectivity and find a working path fast enough in order to maintain tolerable service level. Router's ability to notice breakdowns was tested by breaking down links and routes inside the infrastructure so that packets won't get through. Noticing a local link being down is quick and easy because the status of the link changes. Problem in the infrastructure gets noticed only when sent keepalives are no longer replied by the destination router.

In our tests, the service provider infrastructure was emulated with a networking switch (described more in Section 6.1). We caused breakdowns in the infrastructure by adjusting this switch. The most rudimentary method we used to disconnect a path through the infrastructure was to manually pull out a networking cable. However it was faster and more flexible to do this at software level. This required adjusting the operating system of switch.

Packets can be blocked by creating filtering rules with *iptables*. Iptables is an interface to the firewall inside the linux kernel. You can block specific IP addresses, ports etc.

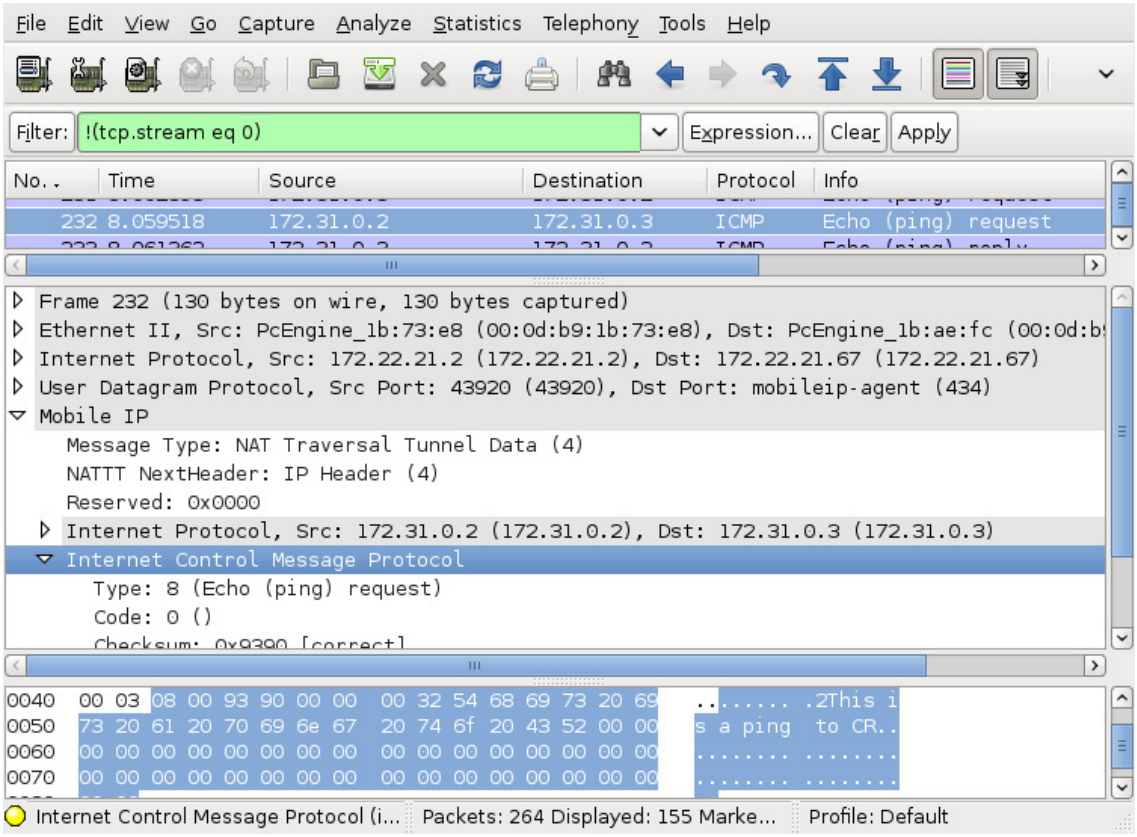

Figure 11: Tunneled ICMP Request viewed with Wireshark.

The method we found the most convenient was configuring Virtual Local Area Network (VLAN) inside the infrastructure switch. VLAN can be used to manage routing inside the switch. The user can define which interfaces are interconnected. Breakdowns can be caused simply by reconnecting one end of a working path to a different interface.

6 Performance measurements

This section describes the measurement conditions and performance results achieved with the software. A physical test network was set up to run the network on. Performance was measured by testing how fast the implementation restores connectivity in case of a path failure. We also measured how an outage affects TCP download rate and VoIP voice quality.

Section 6.1 describes the hardware used to build the test network. Section 6.2 discusses the logical topology of the test network. Section 6.3 describes the test cases executed. Section 6.4 illustrates the results achieved with the running code. These measurements and results are described more thoroughly in Antti Mäkelä's article [22].

6.1 Hardware

Routers and infrastructure The software was tested on a network that consists of a Home Agent, three virtual ISP routers, three Mobile Routers and hosts on the routers' Mobile Networks. Each Mobile Router was running on a PC-Engine's Alix Geode system board, running Debian GNU/Linux. The board has 256 megabytes of memory, variable number of network interfaces and AMD Geode LX CPU. Since Home Agent has to process large amount of the signaling, a more powerful Intel i7 Core-based PC was used as the Home Agent.

Figure 12: A router in the HAaRO test network.

The last board was a switch working as an "infrastructure" node. The board emulated the three service provider routers and the WAN connections from each router to each service provider. In addition to the switching, the infrastructure

system could be set up to provide variable delays, capture traffic for debugging, set up traffic failures and so on.

Figure 12 shows the piece of hardware running the Mobile Router process. There are three Ethernet wires, each connected to a service provider router (the WAN connections). In the USB-Ethernet adapter there is the "customer" connection (host in the Mobile Network). Traffic from the customer was forwarded by the Mobile Router to a tunnel on one of the WAN-connections. When the WAN was disconnected, the router created a new tunnel on a different WAN.

Traffic and measurement The measurements were performed using Spirent Testcenter, which is a well-known industry standard testing platform. Spirent consists of two components: the "Avalanche" component emulates end-users and the "Reflector" component emulates servers. The platform's primary use cases are running stress tests either on network services by emulating end-users, or on the network infrastructure by emulating both the end-users and the services they are accessing. Our case resembles the latter as we want to observe effects to end-user experience. The end-users can be instructed to load a website, conduct a VoIP call and so on. In a single test run an increasing number of users starts to use the network services, until a desired number of users is reached. Then the "steady state" is kept for desired length before the users leave the network.

The Mobile Routers have some performance limitations, mainly due to the tunneling technology. UDP tunneling was implemented using Linux TUN/TAP technology [18] which operates in user-space. Since our primary interest is consumer-level connections, our packet rate was limited to 10 Mbps, which our chosen hardware can handle easily. Processing the signaling messages also creates some processing overhead.

6.2 Topology

The test network consists of a Home Agent, three Mobile Routers, three service provider routers and some end-user hosts, the "customers". Figure 13 illustrates the topology of the test network. Service providers are emulated by the "infra router". Each Mobile Router and the Home Agent are all connected to each service provider. A Mobile Router can register to the Home Agent via any service provider. Each customer host is connected to one Mobile Router. All traffic a customer sends is routed to its Mobile Router. Mobile Router tunnels all traffic via the current service provider. The tunnel endpoint is either the Home Agent or a Correspondent Router (in HAaRO). The Home Agent has access to the Internet.

Each MIP node has three networking interfaces. Each interface is connected to a different service provider network with a different metric value. When Mobile Router process starts, it first connects to the Home Agent via the interface having the lowest metric. When the router receives a UDP Tunneling Accept, it creates a new networking device for the HA tunnel: *TUNLMNA*. The router sets its default route to this device and starts forwarding packets from the virtual tunneling device to the physical networking device the registration was sent from. UDP Data Message

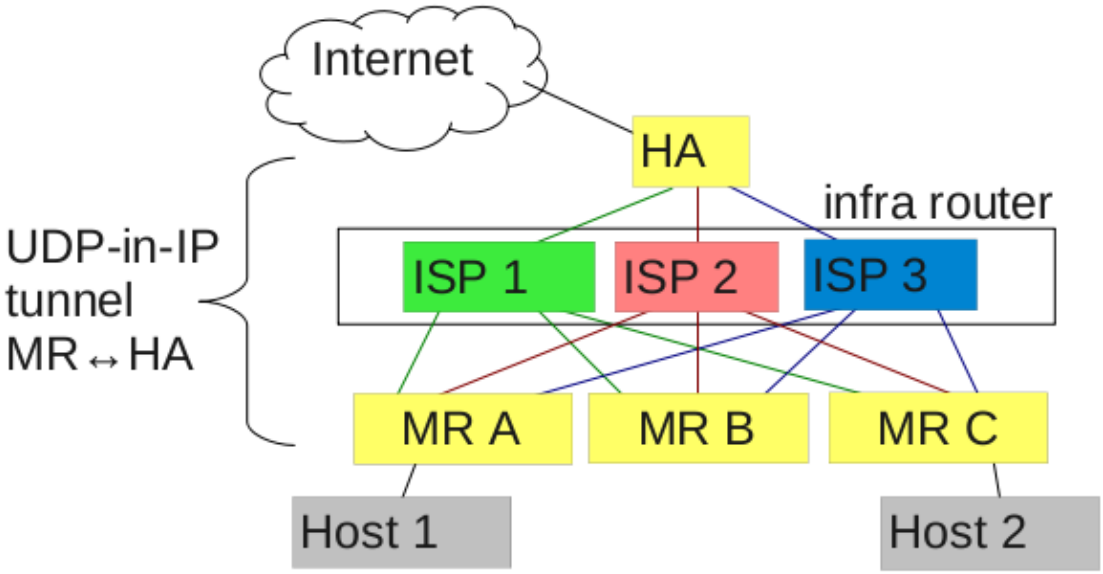

Figure 13: HAaRO test network topology.

in UDP in IP header stack is added to each packet forwarded. This way all packets sent from the TUN device are "in the tunnel" being encapsulated with the UDP tunneling stack

6.3 Test cases

We were interested in testing how much end-user experience on our network deteriorates when an unreliable commodity-connection fails. User experience consists of availability and quality of network services. In practice, availability would be responsiveness to an HTTP-request initiated by an end-user when clicking on a link in a web browser. Another example of quality is download speed of a web site, or in case of VoIP, perceived quality of speech.

In case of a path failure, the software has to find a working connection before the user notices critical decrease of service quality. Studies [23], [24], [25] suggest that web users generally tolerate pauses with maximum length of 2 seconds. For VoIP, a commonly accepted methodology is Mean Opinion Score (MOS) which measures call quality on a scale from 1 to 5, where 1 is "Bad", 2 is "Poor", 3 is "Fair", 4 is "Good" and 5 is "Excellent".

In each test case, 5 test runs were conducted and the results were averaged. MR to MR round-trip time was set to 50 $\pm$ 6 ms to emulate higher geographical distance between each site. Before running the application, we performed baseline measurements with the infrastructure. We caused connection breakdowns of different lengths and measured how these outages affected TCP download speed and VoIP quality. With the results we could later approximate how long of a connection breakdown does the implementation's connection restoration correspond to.

Performance of the implementation was tested by introducing connection failures

and measuring how quickly the connection is reinitialized on another path. Paths were disconnected inside the infrastructure so that the Mobile Router does not detect the current link's status being down. Instead, the router notices tunnel keepalives not being replied. This is the breakdown case that takes more time to notice. The time it takes for the router to recover depends on the router configuration (minimum keepalve interval, failure detection threshold) so it is more interesting in this research.

The configuration was adjusted to provide failure detection and connection recovery as fast as possible. Return Routability was set to be performed on all potential paths before any KRm gets expired. This way the Mobile Router is able to switch to any link directly without consuming extra round-trips for calculating registration keys.

The two configuration parameters that require tuning are minimum keepalve interval (keepalive sending frequency) and failure detection threshold (the amount of lost pings that is interpreted to be a path failure). Larger keepalive interval delays failure detection but too frequent pinging will flood the link. Too large failure detection threshold also slackens failure detection but too small threshold - specially with small keepalive interval - causes incorrectly interpreted path failures, so called false positives. The existing practice for the threshold is 3 pings and for the interval 200 ms. Linux's Priority Queuing was utilized to give the registration and keepalive messages priority over end-user data.

6.4 Measurement results

The effects of different length breakdowns on TCP and VoIP are listed in Table 5. These are the baseline measurement results. In the TCP column, the first subcolumn illustrates how many seconds it took for the download rate to rise to half of the maximum. The second subcolumn is the amount of time the throughput was below 10 % of the maximum. Measured MOS values are listed in the VoIP column.

Table 5: Baseline measurement of TCP download rate recovery time and VoIP quality on breakdowns of different lengths.

	TCP		**VoIP**
	< 50 %	< 10 %	MOS
100 ms	2.6 s	0.2 s	3.94
200 ms	2.6 s	0.6 s	3.66
400 ms	3.4 s	0.8 s	3.18
600 ms	3.8 s	0.6 s	2.76
800 ms	4.0 s	0.8 s	2.60
1000 ms	4.4 s	2.0 s	2.29
1200 ms	4.6 s	2.8 s	2.08

Two different configurations were used during the performance measurements:

200 ms interval with failure threshold 3 and 150 ms interval with failure threshold 2. Measurement results achieved with the implementation are listed in Table 6. VoIP is simply UDP-stream so comparing the results with the baseline measurements is quite straightforward. The first MOS value match with the baseline value of 800 ms and the second MOS value match close to average of 400 and 600 ms. This makes sense because the response time of the latter is 300 ms faster (200 ms * 3 vs. 150 ms * 2). Both of these MOS-values are rounded to "Fair" with the metrics introduced in Section 6.3.

It takes TCP more than two seconds in both cases before transmission rate starts increasing. This indicates that TCP is on "slow start" mode. Due to the congestion control, the measured values are not that easy to compare with the baseline breakdown values. Connection recovery of the implementation during a TCP session corresponds to a breakdown of 1 - 1.5 s. As mentioned in Section 6.3, pauses under 2 seconds are considered tolerable.

Table 6: Measurements with the implementation

	TCP		**VoIP**
	< 50 %	< 10 %	MOS
interval 200 ms, threshold 3	5.8 s	2.2 s	2.61
interval 150 ms, threshold 2	5.6 s	2.8 s	3.09

Keepalive intervals and timers affect the failure detection time and sensitivity to false positives. On the other hand, end-to-end delay on the network affects the connectivity restoration time. This has significant effect on end-user application behavior, especially on TCP which uses congestion control.

The measurements were performed with end-to-end delay set to 25 ms, causing 50 ms round-trip time. Figure 14 illustrates the effect of different end-to-end delays on recovery speed of TCP. Test runs were conducted with 25 ms, 50 ms and 75 ms delays. In all cases the throughput starts increasing immediately when the connection is restored, but the time spent in getting the transmission rate to the maximum increases dramatically when the delay is increased.

6.5 Performance summary

In these measurements we tested end-user experience during a connection outage. The implementation has to notice the connection being down and switch to another connection fast enough in order to maintain tolerable service level. End-user experience was measured with TCP and VoIP traffic. On TCP, the user experience comes from responsiveness of for instance a link on a web page. On VoIP, the experience depends on the voice quality.

Connection switching during a TCP session corresponded to an outage of 1 - 1.5 seconds. This response time is considered to be well tolerable. In VoIP case, an

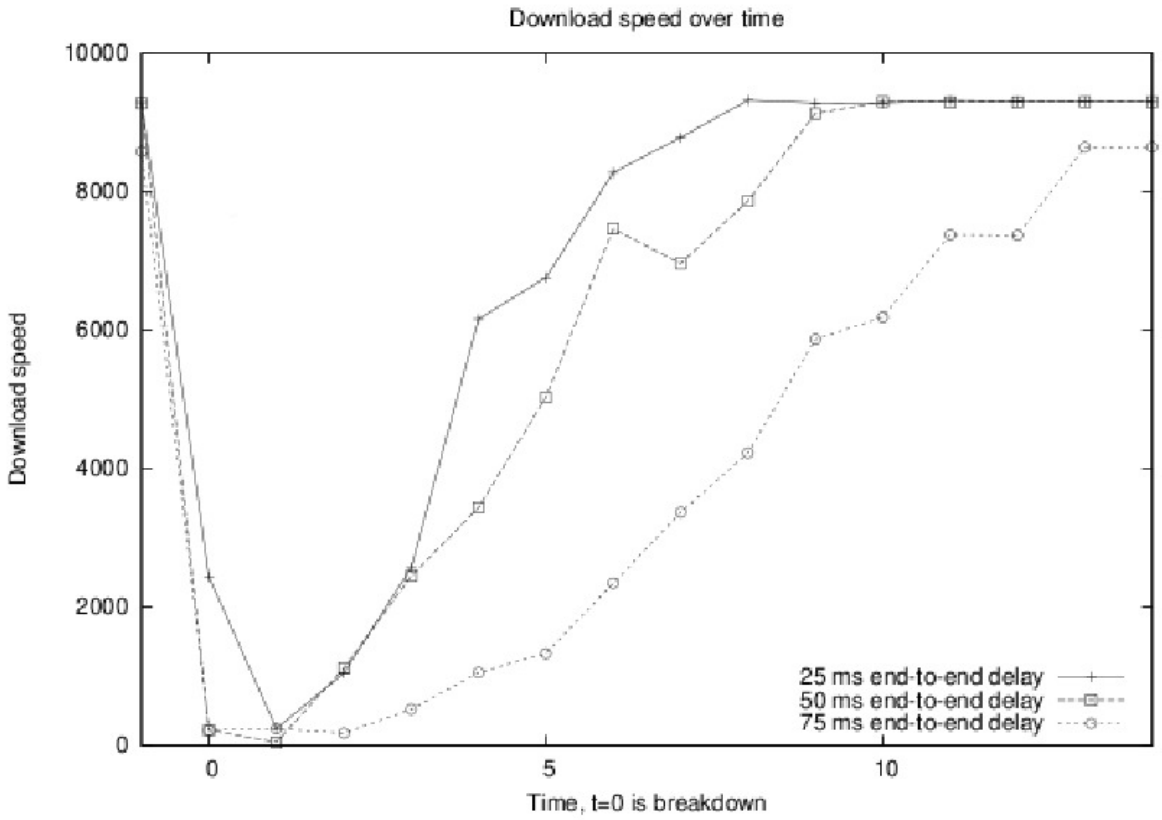

Figure 14: Effect of delay on throughput in HTTP case [22]

outage during a call did not lower the voice quality below "Fair" in Mean Opinion Score metrics.

The implementation was tested on real hardware and with real traffic. However, we did not have a proper network. Our infrastructure switch was very idealistic. In the future the implementation needs to be tested in various network conditions. We need to use networks of real-world service providers in order to reveal all possible networking issues.

7 Summary and future work

The purpose of this study was to explore how well Redundant Array of Independent Internet Connections (RAIIC) works in real world. RAIIC is a concept for providing guaranteed access with low cost. A set of cheap, unreliable consumer-level connections is utilized to provide guaranteed service level. One connection is used at a time and the state of the connection is actively monitored. In case the quality deteriorates, the traffic is switched onto another unreliable connection. Hence the system is able to provide access as long as all the connections are not down at the same time. If the connections are truly independent with no shared infrastructure, the probability of a full outage is almost negligible.

We implemented a Mobile IP based solution in order to be able to test our concept on real hardware and running code. We wanted to verify that the concept is sound and the user experience remains tolerable during a connection breakdown. The system was required to perceive the connection quality deterioration and switch the traffic onto another connection fast enough in order to maintain satisfactory end-user experience. This was tested with TCP and VoIP traffic and in both cases the implementation was able to maintain adequate quality of service. However, our infrastructure network was simulated with a networking switch. In order to achieve more realistic results, the implementation needs to be tested using networks of real-world service providers.

The reason to provide high-availability with multiple commodity-level connections instead of one guaranteed connection is to save on costs. Connectivity guarantees are normally expensive. It may be considerably cheaper to buy multiple commodity-grade connections than to buy SLA guarantees. However, more business model analysis is needed. We still need to study whether you can actually make enough money on network that you don't control.

Also the implementation will be further upgraded. A router will be able to utilize all available paths via load balancing. With this feature, the traffic is distributed more equally on the operational connections instead of just using one at a time. Support for load balancing has already been specified as Mobile IPv4 extension [26]. We also have implemented functions that can be utilized when developing this sort of functionality.

References

[1] A. Mäkelä, H. Warma, K. Kilkki, A. Decros, J. Manner *Economic feasibility analysis of seamless multi-homing WAN solution*, June 2011, Next Generation Internet (NGI), 2011 7th EURO-NGI Conference

[2] Tao of Internet Engineering Task Force. `http://www.ietf.org/tao.html`

[3] C. Perkins, *IP Mobility Support for IPv4, Revised*, RFC 5944, November 2010, Internet Engineering Task Force. `http://tools.ietf.org/html/rfc5944`

[4] C. Perkins, *IP Mobility Support for IPv4*, RFC 3344, August 2002, Internet Engineering Task Force. `http://tools.ietf.org/html/rfc3344`

[5] C. Perkins, *IP Mobility Support*, RFC 2002, October 1996, Internet Engineering Task Force. `http://tools.ietf.org/html/rfc2002`

[6] A. Mäkelä and J. Korhonen, *Home Agent assisted Route Optimization between Mobile IPv4 Networks*, draft-makela-mip4-nemo-haaro-05, April 2010, Internet Engineering Task Force. `http://tools.ietf.org/html/draft-ietf-mip4-nemo-haaro-05`

[7] H. Levkowetz, S. Vaarala, *Mobile IP Traversal of Network Address Translation (NAT) Devices*, RFC 3519, April 2003, Internet Engineering Task Force. `http://tools.ietf.org/html/rfc3519`

[8] K. Leung, G. Dommety, V. Narayanan, A. Petrescu, *Network Mobility (NEMO) Extensions for Mobile IPv4*, RFC 5177, April 2008, Internet Engineering Task Force. `http://tools.ietf.org/html/rfc5177`

[9] Mäkelä, A. *Concept for providing guaranteed service level over an array of unguaranteed commodity connections*, in proceedings of The 25th Symposium On Applied Computing (ACM SAC 2010), March 2010.

[10] Comparison Guide: *Bandwidth Suppliers for Small- to Medium-sized Businesses*, October 2007. `www.voip-news.com`

[11] N. Rickard, *Cost Cutting by Rightsizing Network Reliability*, Gartner research report G00155940, April 2008.

[12] R. Teixeira, S. Uhlig, C. Diot, *BGP Route Propagation between Neighboring Domains*, in Passive and Active Network Measurement, Lecture Notes in Computer Science, 2007.

[13] *IOS IOS NAT Load-Balancing for Two ISP Connections*, Cisco Document ID 100658, February 2008.

[14] P. Vixie et al, *Dynamic Updates in the Domain Name System (DNS UPDATE)*, IETF RFC 2136, April 1997, Internet Engineering Task Force.

[15] P. Calhoun, C. Perkins *Mobile IP Network Access Identifier Extension for IPv4*, RFC 2794, March 2000, Internet Engineering Task Force. `http://www.ietf.org/rfc/rfc2794.txt`

[16] D. Johnson, C. Perkins, and J. Arkko *Mobility Support in IPv6*, RFC 3775, June 2004, Internet Engineering Task Force. `http://tools.ietf.org/html/rfc3775`

[17] B. Andersson, D. Forsberg, J. Hautio, J. Malinen, K. Mustonen and K. Weckström *Dynamics software package in sourceforge*, October 2001. `http://sourceforge.net/projects/dynamics/`

[18] Tun/Tap interface tutorial, March 2010. `http://backreference.org/2010/03/26/tuntap-interface-tutorial/`

[19] M. Stevens, *Fast Collision Attack on MD5*, August 2009, `http://crppit.epfl.ch/documentation/Hash_Function/Examples/Code_Project/Documentation/104.pdf`

[20] Mobile IPv6 for Linux (MIPL) source code, March 2006. `http://linux.wareseeker.com/download/mipl-mobile-ipv6-for-linux-2.0.1.rar/334567`

[21] Mobile IP Implementation in Java (JMIP) source code, March 2005. `http://www.mobileip.org/`

[22] A. Mäkelä, J. Manner *Performance of a economical, redundant system for Intranet connectivity*, IEEE ISCC 2011.

[23] D. Miras et al, *A Survey on Network QoS Needs of Advanced Internet Applications*, Internet 2, QoS Working Group, 2002.

[24] D. Galletta, R. Henry, S. McCoy, P. Polak, *Web Site Delays: How tolerant are users?*, Journal of the Association for Information Systems Vol. 5 Issue 1, pages 1-28, January 2004.

[25] J. Jacko, A. Sears, M. Borella, *The effect of network delay and media on user perceptions of web resources*, Behavior and Information Technology, Volume 19, Issue 6, November 2000, pages 427 - 439.

[26] S. Gundavelli and K. Leung, *Multiple Tunnel Support for Mobile IPv4*, draft-gundavelli-mip4-multiple-tunnel-support-01, January 2010, Internet Engineering Task Force. `http://tools.ietf.org/html/draft-gundavelli-mip4-multiple-tunnel-support-01`