Ubiquigeneous Networking

A Distributed Networking Application Over Mobile Embedded Devices.

Author: Rodolfo Kohn

Thesis Director: Luis Marrone

Thesis presented to obtain the grade of

Master in Data Networking

Faculty of Informatics - National University of La Plata

September 2004

ללאורה ולתינוק הראשון שלנו

A Laura y nuestro primer bebé.

באהבה.

Acknowledgments

This is the result of a four-year effort beginning with the first course at the Faculty of Informatics, at the University of La Plata, in August 2000. I feel the result of this effort is, in some way, reflected in this work. However, many people were involved in this effort and deserve to be especially thanked.

Luis Marrone, my professor and Thesis Director, taught me networking protocols, especially TCP/IP; he introduced me to the fundamentals and thus giving me the possibility to follow this way and to feel safe to go ahead and do more. During my thesis, he reviewed my work and gave me advice. He always did it with a humble attitude and outstanding knowledge and effort.

Alexandru Petrescu, from Edge Networking Research Lab of Motorola Labs, was my Advisor at Motorola. He kindly shared with me his knowledge on Mobility and Distributed Systems, especially on Mobile IPv6, through his advice, reviews and several discussions we had about cutting-edge technology topics. He gave me the opportunity to have some participation in the LIVSIX project and dedicated much of his time to provide me the technical support that was essential for the successful completion of this work.

My parents' support was always present. My managers at Motorola, Global Software Group, Argentina, gave me the freedom to use all the laboratory equipment to complete my Thesis. All my professors at the University of La Plata introduced me to the fundamentals of network security, distributed systems, networking protocols and operating systems.

Finally, I want to thank the most important person, Laura, who was always encouraging me with her love, patience and advice during this long process.

Table of contents

FIGURES

TABLES

Acronyms

Acronym	Description
AD	Administrative Domain
ARP	Address Resolution Protocol
BSD	Berkeley Software Distribution
BU	Binding Update
CN	Correspondent Node
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name Service
ESP	Encapsulated Security Payload
FA	Foreign Agent
GPL	General Public License
GPRS	GSM General Packet Radio Service
HA	Home Agent
ICMP	Internet Control Message Protocol
IETF	Internet Engineering Task Force
IP	Internet Protocol
LAN	Local Area Network
MAC	Medium Access Control
MIP	Mobile IP
MN	Mobile Node
MMU	Memory Management Unit
MTU	Maximum Transfer Unit
OOP	Object Oriented Programming
PAN	Personal Area Network
QoS	Quality of Service
RA	Routing Advertisement
RFC	Request For Comments
RTOS	Real Time Operating System
SA	Security Association

SGSN	Serving GPRS Support Node
SIP	Session Initiation Protocol
SOHO	Small Office Home Office
SPD	Security Policy Database
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UMTS	Universal Mobile Telecommunications System
UTRAN	UMTS Terrestrial Radio Area Network
VoIP	Voice over IP

1 Abstract

The objective of this work is to develop a chat application between systems embedding the ColdFire processor utilizing Mobile IPv6 and TCP protocols in order to demonstrate that TCP connections are maintained while a device roams between two Ethernet 802.3 networks and that a peer node can be located even though it is not attached to its home network. Furthermore, it will be demonstrated that mobility is transparent to upper layers and the benefits this fact brings up.

2 Motivation

Low cost and powerful processors are being used in an increasingly large number of devices embedded into objects surrounding us. While the challenge of creating meaningful services for these systems is constantly being addressed by various simple applications, the communication between these systems is most often scaled down to local exchanges of information, typically over a unique, homogeneous type of a wireless environment. It is expected that when every single system (whatever its size, power constraints or communication range) is enabled to use TCP/IP networking and to establish both client-server and end-to-end peer communications, a whole new range of distributed applications becomes possible. A tightly-related constraint is that such a large number of systems can be networked together only if the large addressing space of IPv6 is used. With IPv6, it is possible not only to ensure local communication, but also to assign a unique address to every single device; hence the ability to connect to the worldwide Internet, and over any wireless technology that is available at a given location.

From an application perspective, it is important to use the widespread BSD socket interface to the communications subsystem in a transparent manner with respect to mobility. Whenever a mobile system is forced to change its temporary address (because it is visiting a different network) this must happen in a transparent way to the upper-level application. Such a behavior is supported by the Mobile IPv6 protocol being developed by the Internet Engineering Task Force (IETF).

Another important aspect related to application-layer software is the always-present need to reduce maintenance costs. It cannot be imagined that every existing IPv6 software application could be ever changed such as to support mobility. Managing mobility entirely at the network layer with Mobile IPv6 helps creating persistent software code and thus staying competitive.

The new distributed applications created as a result of availability of wireless technologies involve highly mobile devices such as cell phones, PDA's, badges, sensors and various forms of robots. It has been identified though that while a data service is

being used, devices can move only as long as they remain attached to the same unique link-layer technology, and, in most cases, to the same access network, unless some new application-layer solution is provided. To improve mobility, the necessity of technology convergence appears and is addressed by some of these solutions. They could be implemented as part of the application - adding complexity and cost to it - or in a middleware layer over which the application would run unmodified. Currently, there are a number of different proposals for such middleware layer that can hardly interoperate. Moreover, any application layer solution over TCP would have the performance drawback of closing and opening new connections for every handover.

Unlike a pure middleware solution, the Mobile IP family of protocols localizes the changes due to mobility only within the IP stack. Being proposed in a standard manner by the IETF, it is intended for adoption by the worldwide Internet community. The Mobile IP solution for IPv4 contexts (RFC 3344) has a series of drawbacks that are being addressed by the IPv6 version. Mobile IPv6 takes advantage of various native IPv6 features (extension headers, address autoconfiguration, inherent IPSec features and others) to dramatically improve on performance and simplicity.

Among other possibilities, this will permit a cell phone connected to a GPRS network to switch to a Bluetooth home network and, from there, to the local Internet connection while all established IP connections are maintained.

Nevertheless, this brings up another challenge: how to locate a device that is not currently positioned at its home network. Traditional naming systems, such as DNS or X.500 directory, are not suitable for this dynamic environment: since a large number of mobile devices could move from one network to other many times in a short time, any server would be easily overwhelmed [1]. This becomes an unmanageable issue, especially if caching is considered. It is necessary to differentiate a naming service from a location service. Mobile IP involves a home-based location service [1] and its use along with DNS, to map from a human-friendly name to a home address, the identifier, is a powerful solution for this problem.

Highly mobile devices usually have low cost and powerful processors with the capacity of running sophisticated applications at the lowest possible price. Among the purposes

of this work is to describe the process of porting the LIVSIX Mobile IPv6 stack from a standard PC-like environment to a more constrained embedded system device using ColdFire MCF5272, executing uClinux (a Linux variant).

The shortly expressed concept of "ubiquigeneous networking" tries to capture in two words all the characteristics described above. Once small mobile devices are spread on a *ubiquitous* scale and use Internet communications protocols over a wide range of *heterogeneous* data and physical links, a great number of *ingenuous* communication and application paradigms will inevitably emerge.

3 Objective of the Thesis

The objective of this Thesis is two-fold: (1) develop a simple chat application using the standard BSD socket interface and (2) port the LIVSIX TCP/IPv6/Mobile IPv6 stack from a classic PC-like architecture to the MCF5272 "ColdFire" processor, under the uClinux open-source operating system.

Another important goal of the work will be to demonstrate the advantages of managing mobility within Mobile IPv6 on a highly mobile device. A demonstration will be performed showing the implementation and execution of these concepts.

Key aspects of this objective are, first, to demonstrate that distributed embedded applications can run unmodified while the device is moving between different networks and, second, that a device can be accessed from anywhere independently of its point of attachment to the Internet.

An important part of the Thesis report will be a description of current technologies and potentially competitor products. Differences and advantages of Mobile IPv6 compared to Mobile IPv4 will be explained. Differences and advantages compared to various existing middleware solutions will be identified. Similar market products will be considered, such as the Vocera communication devices and a short analysis will provide a comparison.

4 Thesis Presentation

This thesis is composed by:

- A document describing the work done.
- A CD containing the document, in word and PDF format, and the source code.

The source code is divided in three parts:

- LIVSIX for ColdFire/uClinux. In the directory ***livsix***.
- Utilities: includes a modified version of ping6, libraries and auxiliary applications. In the directory ***utils***.
- Chat Application. In the directory ***apps***.

The directory containing the other three directories also contains a *Makefile* file that will make the build and copy the binary files into the flash file system directory created by the main uClinux *Makefile*.

5 Mobility problems

An IP address identifies not only a node interface but also the network the interface is attached to.

If a node attaches to a different network it will have to change its IP address, otherwise it will not be able to receive any datagram delivered to it from another network and probably also in the same network:

- Any datagram generated outside of the network to which the node is attached to will be forwarded to the home network
- If the datagram has been generated in the same visiting network, the source host can either have a default gateway configured for destinations not belonging to the attached network or it can send an ARP broadcast asking for the corresponding MAC address. Both the destination node and one router will answer and depending on which frame has been received first, the datagram will be sent to one or the other.

Also datagrams sent out by the node may be dropped by routers implementing ingress filtering [19].
As an alternative, host-specific routes can be propagated by the routers throughout the Internet but this is not a scalable solution.

This feature has the following main consequences:

- The host cannot be accessed by another host that knows its old address.
- Any DNS server storing the host address must be updated and every cache entry for this host throughout Internet must be removed before the host can be accessed by other host having the old address stored in the cache or accessing to a DNS server.
- If the host moves from one network to the other, any existing connection that is based on the IP address, like a TCP connection, will be broken: a datagram destined to the new address will not be considered as belonging to the old socket; also a datagram delivered by the node, with the new address as source address, will not be considered as belonging to the same socket.

6 Mobile IP

6.1 Overview

Mobile IP is a protocol specified by the IETF Network Working Group in the RFC 3344 [11]. It brings up a mobility solution based on the IPv4 protocol.

Mobile IP consists of the following principal architectural entities:

- Mobile Node (MN): the node that is capable of moving away from the home network.
- Home Agent (HA): router on the MN's home network that has location information for a mobile node that is away from home and tunnels to the MN any datagram received in the home network destined to the MN.
- Foreign Agent (FA): a router in a visited network that assists MNs to achieve mobility. It provides routing services to mobile nodes registered with it.

A fourth entity must be considered:

- Correspondent Node (CN): a node communicating with the MN.

Every node has a fixed IP address, called *home address*, by which it can be identified.

When the node is attached to its home network any datagram addressed to it will reach it normally and any datagram sent by it will normally be routed to its destination. When the MN is visiting a foreign network it gets a new care-of address by some means and sends a registration request to the home agent which adds a mobility binding for the MN in its mobility binding list. When another node, called *correspondent node*, tries to access the MN, it sends a datagram that has the home address of the MN as destination address, the datagram is routed as in standard IPv4 to the home network of the MN where the home agent intercepts it, looks for the corresponding care-of address in the mobility binding list and tunnels it to the current care-of address registered for the MN. Finally the MN's IP layer will receive the packet and, if it is necessary, will relay the data to the higher layer.

The care-of address is the terminating point of the tunnel through which the HA delivers datagrams to the MN. Two types of care-of addresses exist:

- Foreign agent care-of address: the MN uses an address of the foreign agent, with which it registers, as the care-of address. Thus the home agent tunnels the datagrams to the foreign agent which de-tunnels the datagram, looks for the destination IP in its visitors list, obtains the corresponding MAC address and delivers the inner datagram to the MN through the link-layer to which both nodes are attached. By using this type of care-of address all visiting MN's can use the same foreign agent's address as care-of address.
- Co-located care-of address: the MN uses a care-of address obtained by other mechanism, such as DHCP. In this case the MN is the terminating point of the tunnel and it directly receives and de-tunnels the datagrams. This type is useful when FA's are not available in a visited network, though it may be not scalable for the IPv4 address space since it is necessary to have a pool of IP addresses reserved for possible visiting MN's.

While the IP home address is the identifier of the node's interface, its care-of address is the locator used to reach a node wherever it is attached to the Internet.

Every mobility agent, either home or foreign agent, sends out Agent Advertisements to publish its services on the link it is serving. An Agent Advertisement is an ICMP Router Advertisement used in IPv4 extended to include Mobility Agent Advertisement information and possibly a Prefix-Lengths extension. For a FA, the Mobility Agent Extension contains at least one care-of address that a MN can use as foreign agent care-of address. The Prefix-Lengths extension specifies the number of leading bits that define the network number of the corresponding Router Address listed in the ICMP Router Advertisement part of the message. More details about these extensions can be found in [11].

Figure 1 shows the typical scenario in Mobile IP. A host, called *Correspondent Node* in this scenario, is attached to an Ethernet LAN which is connected to the Internet through a router. This CN sends an IP packet to the MN using its home address as destination

address. In the home network, in this case a Bluetooth PAN, the Home Agent has, at least, one mobility binding for the MN in its mobility bindings list. The HA intercepts the packet addressed to the MN and tunnels it to the foreign agent, in this case the foreign network is located in a Radio Area Network of UMTS (UTRAN), the foreign agent gets the inner packet from the packet received and delivers it to the MN registered with it by some link layer means.

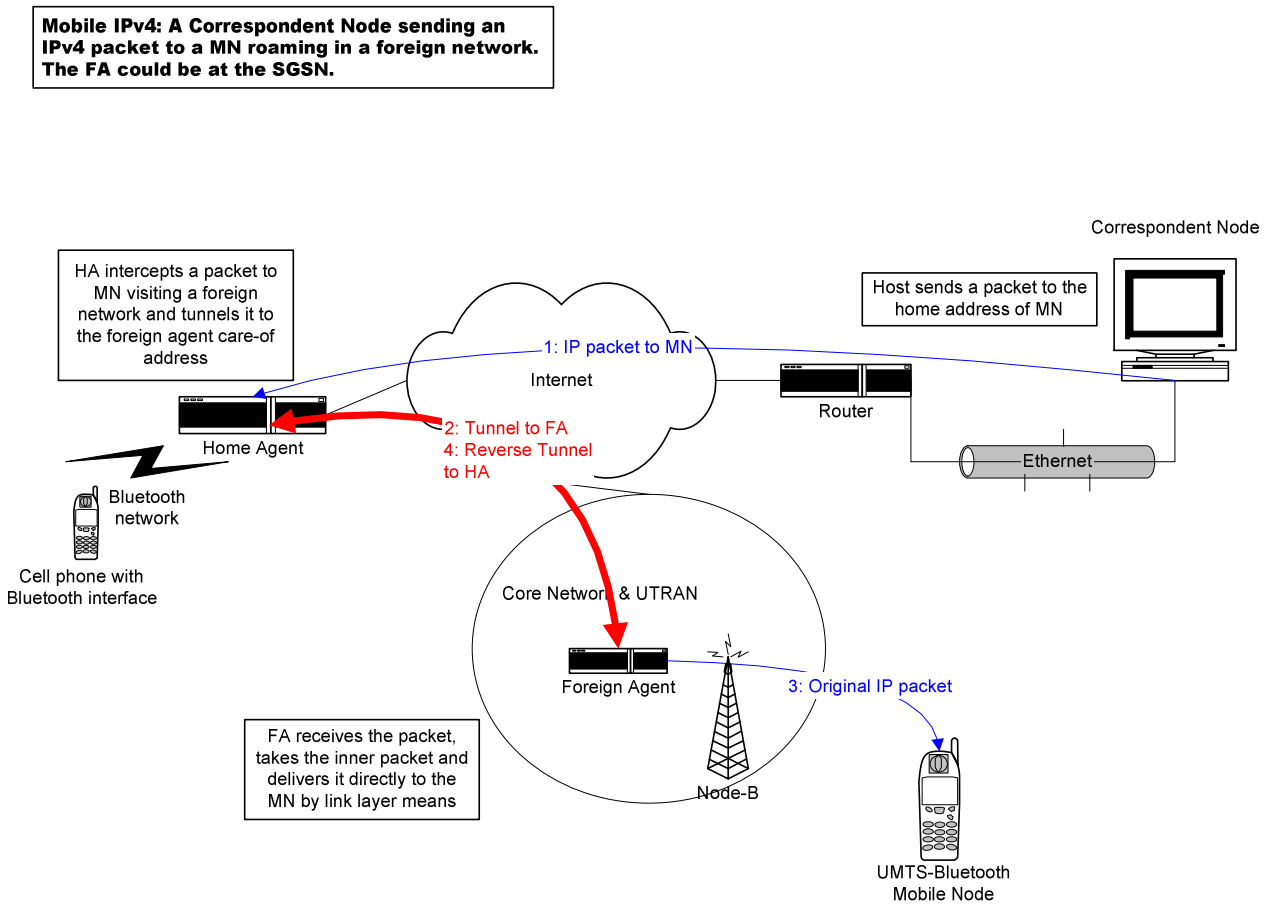

Figure 1 - Typical Mobile IPv4 scenario

RFC 3344 provides two primary algorithms for the MN to detect that it has moved to other network:

- A MN records the Lifetime value conveyed by the Agent Advertisement. If the MN hasn't received any other advertisement from the same agent within the specified

lifetime, it should attempt to discover a new agent with which to register.

- By comparing the router addresses and prefix-lengths received in the Agent Advertisements received, the MN can detect it has moved.

A MN can also use some link layer information to detect it has moved to other network, like in Cellular IP [4].

A Home Agent will use either gratuitous ARP or proxy ARP to intercept packets destined to a MN that is roaming out of its home network and is registered with it. After having intercepted the datagram, the HA will send it to the care-of address. In Mobile IPv4 a HA can have more than one Mobility Binding for one MN, in this case the packet will be sent to all the care-of addresses registered for the MN.

When a MN attached to a foreign network sends a datagram to a correspondent node, it must use its home address as the source address. It can send it directly to the correspondent node but the datagram could be filtered out by a router implementing ingress filtering [19]. A safer way is to send the datagram in a reverse tunnel to the Home Agent which, in turn, will take the inner datagram and will deliver it to the Correspondent Node.

Finally, security considerations are defined for Mobile IP in RFC 3344. Mobile Nodes and Mobility Agents must be able to perform authentication, especially Registration messages must be authenticated. The default algorithm is HMAC-MD5. Protection against replay attacks for registration requests is especially considered: a 64-bit identification field is present in a Registration Request sent from the MN to the HA; this field is used along with two methods – timestamps and nonces – for protection.

6.2 A Macromobility Solution

The terms macromobility and micromobility are widely used with a number of meanings. In this work the definition adopted in [4] will be used, and macromobility will be used to define mobility between different Autonomous Systems or Administrative Domains but it

will also be used to define mobility between different Access Networks and although usually there is a correspondence one-to-one between an Administrative Domain and an Access Network, this is not always true. An inter-AD handover implies the mobile node could need to be re-authenticated, QoS and charging policies could change, probably a different IP address must be used and there is no security of mobility support [4].

Mobile IP is considered a good solution for macromobility. It is not considered as a well suited solution for micromobility because of both the large amount of registration messages it could generate in the Internet and the fact that other mechanisms such as already implemented link layer solutions are faster to achieve handoffs [4][11]. In intra-access network handovers there is usually an L2 proprietary handover solution already implemented.

6.3 Mobile IPv4 Implementations

There are several Mobile IP implementations for different Operating Systems. Some of them are Cisco Mobile IP, for Cisco IOS; Dynamics, for Linux and Windows, from the Helsinki University of Technology; Sun Mobile IP, for Solaris; Treck Inc., for embedded systems and RTOS.

6.4 Drawbacks

Mobile IP has important drawbacks though:

- Every traffic must be delivered through the HA which must tunnel it to the corresponding care-of address. This puts a significant burden onto this node. Furthermore, a significant delay is added to the round-trip time, sometimes unnecessarily, e.g. when the correspondent node is in the same visiting network as the mobile node.
- The IP address is an identifier for a node. The lack of available addresses in IPv4 is an important hinder.

- It requires a costly infrastructure to be added: a home agent in the home network and a foreign agent in each network that can be visited. Besides, foreign agents must be installed if the scalable service of foreign agent care-of address is to be provided on a link.

7 Mobile IPv6

7.1 Overview and advantages over Mobile IPv4

Mobile IPv6 is a work in progress of the Mobile IP Working Group of Internet Engineering Task Force (IETF) [2].

Mobile IPv6 allows nodes to roam throughout the IPv6 Internet while still being reachable by any other node. Whereas the basic scheme is pretty similar to Mobile IP for IPv4, Mobile IPv6 solves all the drawbacks present in the former:

- By the utilization of new extension headers and destination options, it allows Route Optimization, thus the efficiency can be dramatically improved: HA load can be alleviated and the traffic can be faster.
- It uses the IPv6 address space which allows every device to have an IPv6 address as identifier.
- It gets rid of foreign agents thus diminishing the cost of infrastructure.

Mobile IPv6 defines new ICMPv6 packets, extension headers and destination options and makes use of the advantages of the Neighbor Discovery Protocols, as it will be commented below.

This new protocol defines the following principal elements:

- Mobile Node
- Home Agent
- Correspondent Node.

While the MN is attached to its home network, IPv6 datagrams sent to it (and sent by it) are routed normally as in IPv6.

In the typical scenario when a MN is visiting a foreign network, it listens to the Router Advertisements [4] sent out by the routers attached to the same link. By looking at the

new subnet prefixes, it can detect that it is away from the home network and, after checking with Duplicate Address Detection that its link-local address is unique within the link it is attached to, it uses stateless address auto-configuration [5] to form its own care-of address stemmed from its hardware address and a network prefix advertised by the routers. Afterwards, the device sends a Binding Update (BU) to its HA which in turn updates its binding cache in order to maintain the new locator, and then it sends BU's to any CN which has an entry for the MN in its binding cache.

Foreign Agents are not needed in Mobile IPv6 because of the existence of stateless address auto-configuration [5] specified by IPv6, in addition to stateful address auto-configuration, the modified IPv6 Router Advertisements and the stretched address space in IPv6. So, there are no foreign agent care-of addresses but only co-located care-of addresses.

When a node, not having the corresponding entry in its binding cache or not supporting Mobile IPv6, sends out a datagram destined to a MN that is away from home, the datagram is routed to its home address; there, the HA intercepts the datagram and obtains the current care-of address – the locator - of the MN from its binding cache, then the HA tunnels the datagram directly to the MN which de-tunnels it at the IP layer and passes it to the higher layers.

In Mobile IPv6 the MN can communicate with a CN in two different modes:

- Bidirectional Tunneling: does not require Mobile IPv6 support from the CN, any datagram going to the MN goes to the home network, as in Mobile IPv4, where the HA intercepts it and sends it through a tunnel to the MN. Datagrams sent by the MN are reverse-tunneled to the HA which sends it to the CN.
- Route Optimization: after successfully running a procedure called Return Routability Procedure by which the MN receives two nonce indexes (home nonce and care-of nonce) and two tokens (home keygen token and care-of keygen token), the MN sends a Binding Update with the primary care-of address to the CN which creates an entry for the MN in its binding cache. When the CN sends a packet to the MN, it checks its binding cache looking for an entry for the MN's home address and obtains the care-of address which is used as the destination address of the IP packet, the

home address is conveyed in a new extension header called Routing Header Type 2. When the MN receives the packet it takes the home address as the destination address for IP and upper layers processing. The MN also sends packets directly to the CN using its care-of address as the source address and storing its home address in a new destination option called Home Address Destination Option, which is part of the Destination Options header. The CN will use the home address for IP and upper layers processing. This mode requires support of Mobile IPv6 in the correspondent node.

With Route Optimization the shortest communication path can be used. Nevertheless, when a CN supporting MIPv6 contacts the MN for the first time, it sends the first datagram to the home address of the MN and, after obtaining the MN's care-of address, subsequent packets can use Route Optimization. Figure 2 shows a typical Mobile IPv6 scenario using Route Optimization.

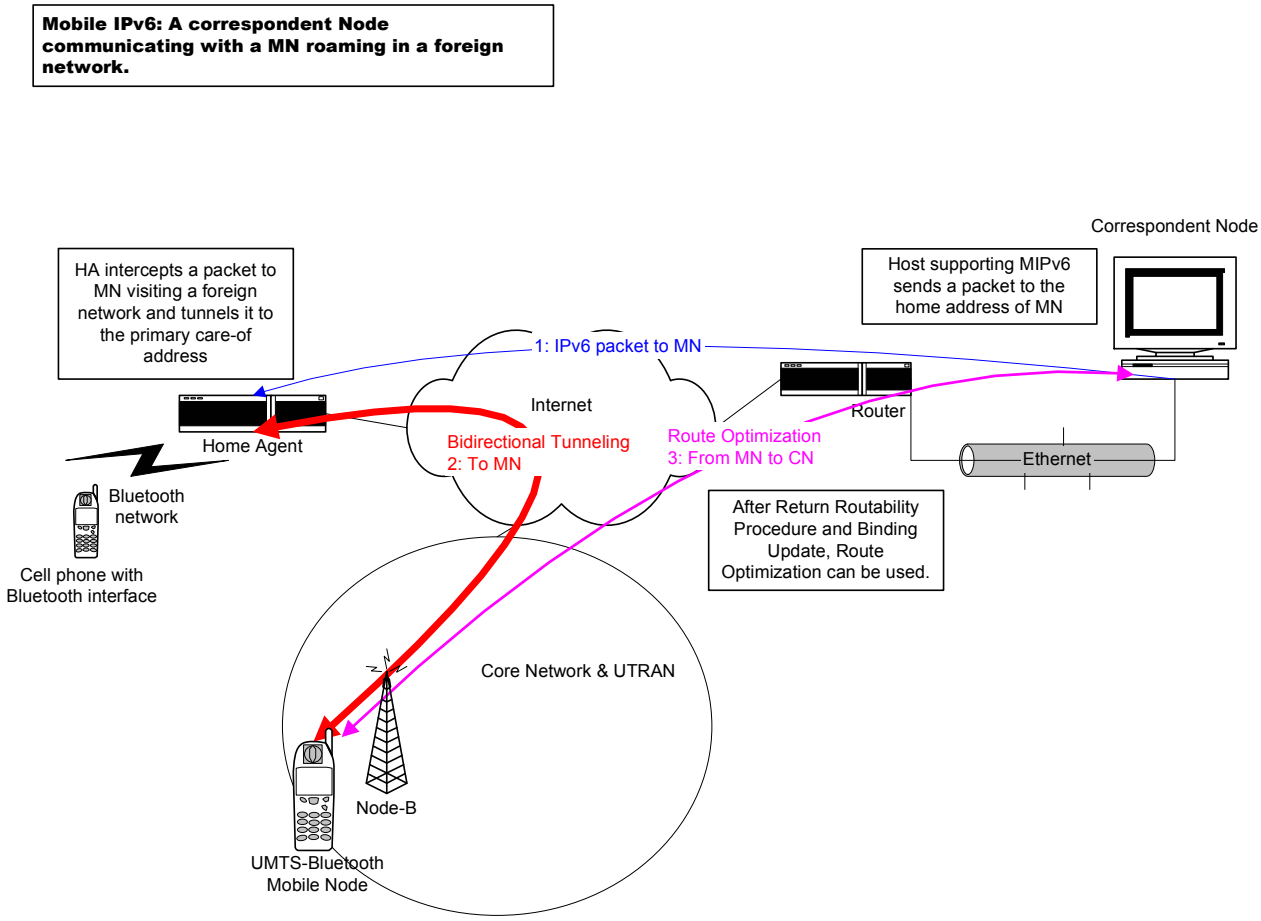

Figure 2 - Typical MIPv6 scenario

7.2 Return Routability Procedure and Binding Updates

As it was mentioned above when MN moves to a new foreign network obtaining a new primary care-of address it has to send a Binding Update to it HA and it should send BU's to the CN that appear in the MN' Binding Update List.

First of all, the MN must send the BU to the HA to register the new primary care-of address; upon receiving a successful Binding Acknowledgement the MN may initiate registration to the CN's.

When the MN registers a care-of address with a CN, it stores an entry in a data structure called Binding Update List for that CN. That entry will store data as the CN IPv6 address, the care-of address registered, the lifetime of the registration, a home nonce index, a care-of nonce index, a home keygen token, a care-of keygen token, etc.

Before sending a BU to a CN, for the sake of security, the MN must initiate a procedure, called Return Routability Procedure, with the CN. Thus, the CN is assured that the right MN is sending the BU. This procedure is made up of the following steps:

- The MN sends to the CN a Home Test Init message with its home address as the source address and a cookie, home init cookie. This message is reverse tunneled through the HA. Such tunneling should employ IPSec ESP in tunnel mode.
- The MN sends to the CN a care-of Init message with its care-of address as the source address and a care-of init cookie.
- The CN generates a home nonce and then it generates a home keygen token based on the home nonce, the home address of the MN and a secret key. Afterwards, it sends a Home Test message, containing the home keygen token and a home nonce index, to the MN's home address.
- The CN generates a care-of nonce and then it generates a care-of keygen token based on the care-of nonce, the care-of address of the MN and a secret key. Afterwards, it sends a Care-of Test message, containing the care-of keygen token and a care-of nonce index, to the MN's care-of address.

When the procedure is finished, the MN generates a binding management key, kbm, based on both keygen tokens and sends the BU with the following data:

- Source address is the care-of address.
- The home address within the Home Address Destination Option.
- A sequence number.
- A lifetime.
- A Mobility Header with the following options:
  - Home nonce index and Care-of nonce index.
  - A MAC (Message Authentication Code) generated with HMAC_SHA1 based on

the Binding Management Key, the care-of address, the correspondent address and the Mobility Header data. This MAC is conveyed in the Binding Authorization Data option of the Mobility Header.

When the BU is received by the CN, it validates the message and creates a new entry for the pair (home address, care-of address) in its Binding Cache, or updates the existing one for the home address. Then it sends a Binding Acknowledgement. When the acknowledgement is received by the MN, it updates or creates the corresponding entry in its Binding Update List.

In order to delete any existing BU for the MN in the HA or a CN, the MN sends a BU with a lifetime equal to 0 (zero) and the home address as the care-of address.

A MN can decide to send a BU to a CN which is not present in its Binding Update List. An open issue is how to detect that the CN supports Mobility or not, one possibility is by using the Home Address Destination Option in the Binding Update, if the CN does not recognize it, it will answer with an ICMPv6 error message.

A MN can decide not to publish its care-of address to certain CN's, so it keeps using bidirectional tunneling.

The MN also has entries in its Binding Update List for the HA bindings.

7.3 Modifications to IPv6 protocol.

Mobile IPv6 is a new protocol that adds two new extension headers and one new destination option to the IPv6 protocol. It also adds three new ICMPv6 messages.

A list of the new features follows:

- **Mobility Header.** This extension header is used by mobile nodes, correspondent nodes and home agents in all messages related to the creation and management of

bindings: Binding Refresh Request message, Home Test Init message, Care-of Test Init message, Home Test message, Care-of Test message, Binding Update message, Binding Acknowledgement message, and Binding Error message. The next header value for this header was not defined at the time of [8].

- **Home Address Option.** This option is carried by the Destination Option extension header and it contains the Home Address of the MN that sends a message when Route Optimization is used. The receiving node uses this address as the message's source address for IPv6 processing and for upper layers. This option cannot be used when the Mobility Header is present, except in the case of Binding Update messages. If the receiving host does not recognize this option, it must discard the packet and must send an ICMPv6 message to the sender.
- **Type 2 Routing Header.** This extension header is inserted in an IPv6 packet sent directly to the care-of address of the mobile node. It contains the home address of the MN. The receiving MN takes the home address from this header and uses it as the destination address for any IPv6 or upper layer processing.
- **ICMP Home Agent Address Discovery Request Message.** This message is sent by the mobile node to the Mobile IPv6 Home-Agents anycast address, for its own subnet prefix, in order to initiate the dynamic home agent address discovery mechanism.
- **ICMP Home Agent Address Discovery Reply Message.** This message is sent by a home agent to respond to the Home Agent Address Discovery Request, it conveys a list of home agents' addresses in the home link.
- **ICMP Mobile Prefix Solicitation Message.** This message is sent by the mobile node, while it is away from home, to the home agent to gather prefix information about its own network.
- **ICMP Mobile Prefix Advertisement Message.** This is the response to the last message above.

Mobile IPv6 also specifies modifications in the Router Advertisements defined by IPv6 Neighbor Discovery [10]. These modifications will be described in the description of this type of message.

7.4 Router Advertisements.

The Router Advertisement Message is a key element in Mobile IPv6 because it is essential for basic movement detection; sometimes they are not necessary though.

This is an ICMP message defined by IPv6 Neighbor Discovery [10]. Each router periodically multicasts this message, on the links to which it is attached, to advertise its presence along with various link and Internet parameters. This message is also sent as a response to a Router Solicitation Message.

The following information is included in the Router Advertisement Message:

- Cur Hop Limit. The default value that should be placed in the Hop Count field of the IP header for outgoing IP packets.
- Flag 'M' that indicates hosts in the link to use administered (stateful) protocol for address autoconfiguration.
- Flag 'O' that indicates hosts in the link to use administered protocol for autoconfiguration of other information.
- Router Lifetime as default router.
- Reachable Time.
- The time between retransmitted Neighbor Solicitation messages.
- The following options can be included:
    - Source link-layer address.
    - Link MTU.
    - Prefix Information. Prefixes that are on-link and/or are used for address autoconfiguration. All on-link prefixes should be included.

Based on router advertisements received, a host builds a list of default routers. It also can decide whether a destination address is on-link or not (even though an on-link prefix is not necessarily present in the Prefix Information option. Also, hosts use prefix information for stateless address autoconfiguration.

More information can be found in [10].

Mobile Nodes can use Router Advertisements to learn that they have moved from one network to another one. They can learn the presence of new routers and the fact that previous routers are no longer reachable. Thus, they can acquire a new care-of address and send the appropriate Binding Updates.

Since it is desirable to have faster movement detection, Mobile IPv6 relaxes the limits set by IPv6 Neighbor Discovery for the interval between Router Advertisements sent by routers that are expected to provide service to visiting mobile nodes or those that are set as home agents.

Home Agents must include the Source Link-Layer Address option in all RA's sent.

Other modifications specified by Mobile IPv6 are:

- A flag 'H' is added in the Router Advertisement Message to indicate that the router is a Home Agent on the link.
- A flag 'R', Router Address bit, is added to the Prefix Information option to indicate that the Prefix field contains a complete router global address. This is done because Neighbor Discovery only allows advertising the link-local address while Dynamic Home Agent Address Discovery mechanism requires the knowledge of the routers' global addresses. A HA must include at least one option with the 'R' bit set.
- A new Advertisement Interval Option is added to indicate the interval at which the sending router sends unsolicited multicast Router Advertisements. Routers may use this option or not.
- A new Home Agent Information Option to indicate the preference of a HA, useful for Home Agent Address Discovery mechanism, and the lifetime of the router availability as a HA.

More detailed information can be found in [8].

7.5 Movement Detection and Handover

L3 Handover is defined by [8] as the process by which a node detects a change in the on-link subnet prefix, possibly because of a change of the subnet to which it is attached; this requires a change in the care-of address and consequently the sending of binding updates to the HA and the CN's.

L2 Handover is the process by which the mobile node changes from one link-layer connection to another [8].

An L3 Handover can be a Horizontal Handover when the same interface is used and the link-layer connection changes or a Vertical Handover when the interface changes, for example when a device turns from a connection to a GPRS radio access network to a WLAN 802.11 connection.

According to [8], the primary goal of movement detection is to detect L3 handovers. The generic method described in [8] for detecting movement is based on Router Advertisements and Neighbor Unreachability Detection [10].

Basically, a node could detect that it has moved to a new L3 network when it receives Router Advertisements from a new router, with different subnet prefixes, and when it detects that the old router is no longer reachable by the use of Neighbor Unreachability Detection after having noticed that it hasn't received its RA's for a reasonable amount of time. However, a number of considerations that can be read in [8] make this detection pretty complicated and different methods could be necessary for different types of link-layers, applications and deployment scenarios [8]. Link-layer information can be needed and it can even be preferable to determine that there has been an L3 handover (for example in cellular radio access networks, which already use link-layer procedures to manage micromobility).

When a MN detects an L3 handover, it performs Duplicate Address Detection on its link-local address, selects a new default router, performs Prefix Discovery based on the RA's received, and creates the new care-of addresses. Registers its new primary care-of address with its HA and, afterwards, it can update associated mobility bindings in the

CN's it is performing route optimization with.

Mobile IPv6 can be preferably used for macromobility. Because the signaling load that Mobile IPv6 generates on L3 handovers could affect the Internet and could produce delays in movement management that could hinder the accomplishment of seamless handovers, for micromobility other protocols - such as link-layer protocols, Hierarchical Mobile IPv6 and per-host forwarding protocols - and variants of this protocol - such as Fast Handovers for Mobile IPv6 - could be preferred [4]. In [4] a number of alternatives for micromobility are described.

7.6 Implementations

There are several implementations of Mobile IPv6; one of them is LIVSIX, http://www.enrl.motlabs.com/livsix. Among others, the following can be mentioned: Cisco Mobile IP, for Cisco IOS, http://www.cisco.com/warp/public/732/Tech/mobile/ip; Monarch, for FreeBSD, from Rice University http://www.monarch.cs.cmu.edu/mobile_ipv6.html; MIPL, for Linux, from Helsinki University of Technology http://www.mipl.mediapoli.com; Treck Inc., for embedded systems and RTOS, http://www.treck.com. Some of the existent implementations have commercial license; others have GPL license or other type of Open Source license.

7.7 Transparency

Mobility management is transparent to the higher layers. Thus, an application can run regardless of the mobile node's being at the home network or the mobile node's roaming at a visiting network: a TCP connection towards the permanent Home Address can be maintained alive and a FTP client, for example, can download a large file while the node is roaming; also, a potentially mobile UDP server is always reachable at its Home Address.

8 LIVSIX: a Mobile IPv6 Stack

LIVSIX [17] is a Linux Mobile IPv6/TCP/UDP stack designed by Edge Networking Research Lab of Motorola Labs to be used in mobility environments under the Motorola LIVSIX Public License, an Open Source License.

This stack used to run in Linux systems, in i386 architecture processors. In the work described here it was run and tested on a variable-length RISC ColdFire architecture microprocessor, MCF5272.

LIVSIX stack is built into a module which must be loaded using the UNIX utility **insmod**.
At the time the tests were done, LIVSIX included the specifications of draft Draft-ietf-mobileip-ipv6-19 but work is in progress to apply Draft-ietf-mobileip-ipv6-24 modifications [8] and RFC 3775.

Bidirectional tunneling is fully implemented in LIVSIX. Route Optimization is in progress, during the test only messages sent from a correspondent node directly to a mobile node, using Routing Header Type 2, were tested; the binding was added manually to the binding cache.

IPSec is not fully implemented in LIVSIX yet. In the Security Policy Database, the destination and source addresses are used as SA selectors. During the tests done, no security was applied to IP packets. For every source and destination addresses security was bypassed.

LIVSIX allows the configuration of mobile nodes, correspondent nodes, home agents and routers with Mobile IPv6 support.

9 Processor platform

MCF5272 is a V2 core ColdFire family microprocessor, created by Motorola.

It has a variable-length RISC architecture where the instructions can be 16, 32 or 48 bits long. This allows code to be packed tighter in memory and thus, lowering memory and system costs.

The ColdFire core is easily integrated with memories, system modules and peripherals. This is a low-cost microprocessor especially designed for the cost-sensitive embedded systems market.

It is used in a large variety of products like industrial equipment, cameras, robots, small office-home office routers, Ethernet switches and VoIP phones.

It is a MMU less (no Memory Management Unit) processor so Virtual Memory is unavailable, though protected memory capability could be optionally added.

The processor used has no floating point support and this adds complexity to the programming tasks.

The architecture uses big endian byte order for integer numbers. This created some complications to the port work.

The main features are:

- Performance: 63 Dhrystone 2.1 MIPS @ 66MHZ
- 1KB I-cache
- 4KB SRAM
- Multiply-Accumulate Unit (MAC)
- Hardware integer divide unit
- Debug module - background and real time
- Doze mode Integrated processor:

- IEEE 802.3 compliant 10/100 Fast Ethernet Controller (FEC), with dedicated DMA
- USB 1.1 device controller and transceiver
- 4 2B+D TDM ports
- HDLC software module
- QSPI
- SDRAM controller
- 3 PWM outputs
- 2 UARTs
- 1-channel DMA
- 8 chip selects
- 16-bit general purpose I/Os
- 4 16-bit timers SW watchdog timer

The block diagram of the processor as shown in the Freescale site [18] can be seen in Figure 3 - M5272C3 block diagram.

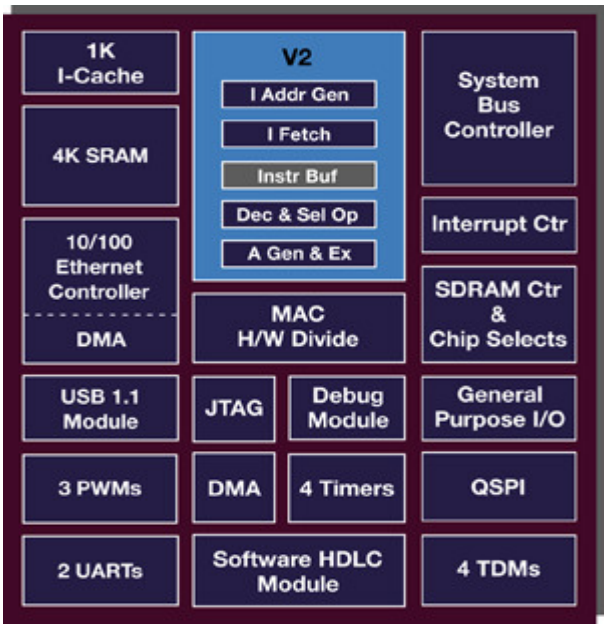

Figure 3 - M5272C3 block diagram[1]

In order to be able to use the processor and its different embedded peripherals an evaluation board was used: M5272C3.

M5272C3 has

- 1 MCF5272 microprocessor
- 2 MB Flash memory
- 16 MB SDRAM.
- 1 10/100 Ethernet with RJ-45 connector.
- BDM interface.
- 2 RS-232 interfaces.
- Other interfaces not used: USB 1.1, PWM, etc.

Figure 4 shows the two boards used during the tests done.

[1] This diagram was obtained from the Freescale web site [18].

Figure 4 - M5272C3 boards used in the tests.

The boards in the figure are connected through RS-232 to a PC; the output is shown in a serial terminal emulator. Both boards are attached to an Ethernet 802.3 network. One of the boards has a BDM device connected to download the image.

10 Operating System

uClinux (micro-controller Linux) [19] is a GNU open source Embedded Operating System evolved from the main Linux kernel; it is meant to run on small microprocessors with hardware constraints such as lack of MMU (Memory Management Unit), floating point arithmetic and others.

Its kernel is tightly related to the Linux kernel.
The kernel version used is 2.4.19.

uClinux source code can be compiled in Linux with the cross-compiler m68k-elf-gcc. The compilation process generates an image that includes, in this work, a flash file system.
The zipped image is downloaded to the MCF5272, in the M5272C3, by means of a BDM connector, to a specific address in flash memory.

In order to boot, a boot loader, called Colilo, has been used. This boot loader has the responsibility of loading the Operating System into RAM memory and passing the execution control to it.

uClinux distributions provide much of the possibilities Linux provides: different types of file systems, networking, etc.

Some distributions also include a number of applications. One of them, BusyBox [20] has been intensively used. BusyBox includes a set of UNIX utilities that are optimized for embedded environments. The utilities used during this work are: insmod, rmmod, lsmod; all of them are useful to manage device driver modules.

In order to get information and requesting help for problems, there are a number of mailing-lists. One very important list is the one hosted by the main site [19].

11 Application-layer Software

11.1 Introduction

A chat application has been developed for the purpose of demonstrating mobility support over different networks. Since mobility is managed in the Network Layer, these networks could be heterogeneous wireless or wired access systems (but this has not been tested for the current thesis). The typical scenario involves a person carrying a device and roaming between networks, where different wireless access technologies are available. One such environment could be a typical enterprise campus, where several buildings are linked by wired technologies but where "hotspot" areas offer wireless access to small devices. Once the application is running, changing the access system is managed by the IP stack, transparently to the application.

Besides Terminal Mobility, Personal Mobility can be achieved with Mobile IPv6. That is, the user could change the device and he could be still located if the same home address is used. This was not tested for this application.

A location service is needed in order to keep track of the current location of the user and his contact device. This service will most likely include the Home Agent capabilities of Mobile IPv6 that associates the permanent Home Address to any temporary Care-of Address.

In addition, the location framework may include a name service such as X.500-like directory in order to map user-friendly names to an identifier (an IPv6 Home Address, or a security certificate or a multicast group). Also the current DNS system could be considered to offer similar service. These types of naming services have not been used for this thesis; instead, a very simple Name Resolver was implemented to map user-friendly names to the home address if it were necessary.

11.2 Technical features

The application has the following technical characteristics:

- It is programmed with C++, with Object Oriented Programming (OOP)
- It was designed using an Object Oriented Design.
- UML was used for the design.
- The target object runs on the processor ColdFire MCF5272, M5272C3 evaluation board.
- The application is compiled in Linux with the cross-compiler GNU m68-elf-gcc.
- The application uses BSD sockets and IPv6.
- It is a one thread application. Non-blocking sockets is used.
- The application has a text user interface. Because of the use of de-coupled classes and a Model-View–Controller architectural pattern, the user interface could be changed without affecting the core of the application.
- The network interface used is Ethernet 802.3
- The application is unaware of any change in the network attachment. It only knows its home address and the home address of the devices with which a session is, or can be established.

11.3 Functional Description

The chat application allows a user of a MCF5272 based device to connect to other users handling any device that runs the same application – or a compatible one - over IPv6. It allows the establishment of several concurrent chat sessions such that, in each of them, text messages are exchanged between two users.

As it was indicated above, a text user interface was implemented. This is so because the M5272C3 was connected through an RS-232 connection to a PC, in which output was displayed in a text mode serial terminal emulator. Input was inserted in the PC through this terminal emulator. Because a Model-View-Controller architectural pattern was used, the user interface is de-coupled from the core application, so a graphical user interface

could be implemented for a different output device, without modifying the main functionality classes.

The chat application uses IPv6 BSD sockets interface to connect to other applications over TCP and IPv6. It is not aware of Mobile IPv6.

Because at the moment of this work, LIVISIX has not been tested with multithreaded applications, this is a one-thread application using non-blocking sockets, in order to be able to manage many different chat sessions, besides listening to new connection requests. The application is constantly polling in order to detect new connections and to serve every established session. If blocking sockets had to be implemented in the future, the application, mainly the session layer implementation, would have to suffer important modifications since multithreading would be necessary and polling would not be useful.

The main actions that can be done in a chat application are:

- Select a contact to chat.
- Initiate a chat session with other user. A connect, or an invitation message, has to be sent to the intended user.
- Accept an invitation sent by other user to initiate a chat session.
- Reject an invitation sent by other user to initiate a chat session.
- Send a text message to other user.
- Finalize a chat session.

Figure 5 shows the corresponding Use Cases for the chat application.

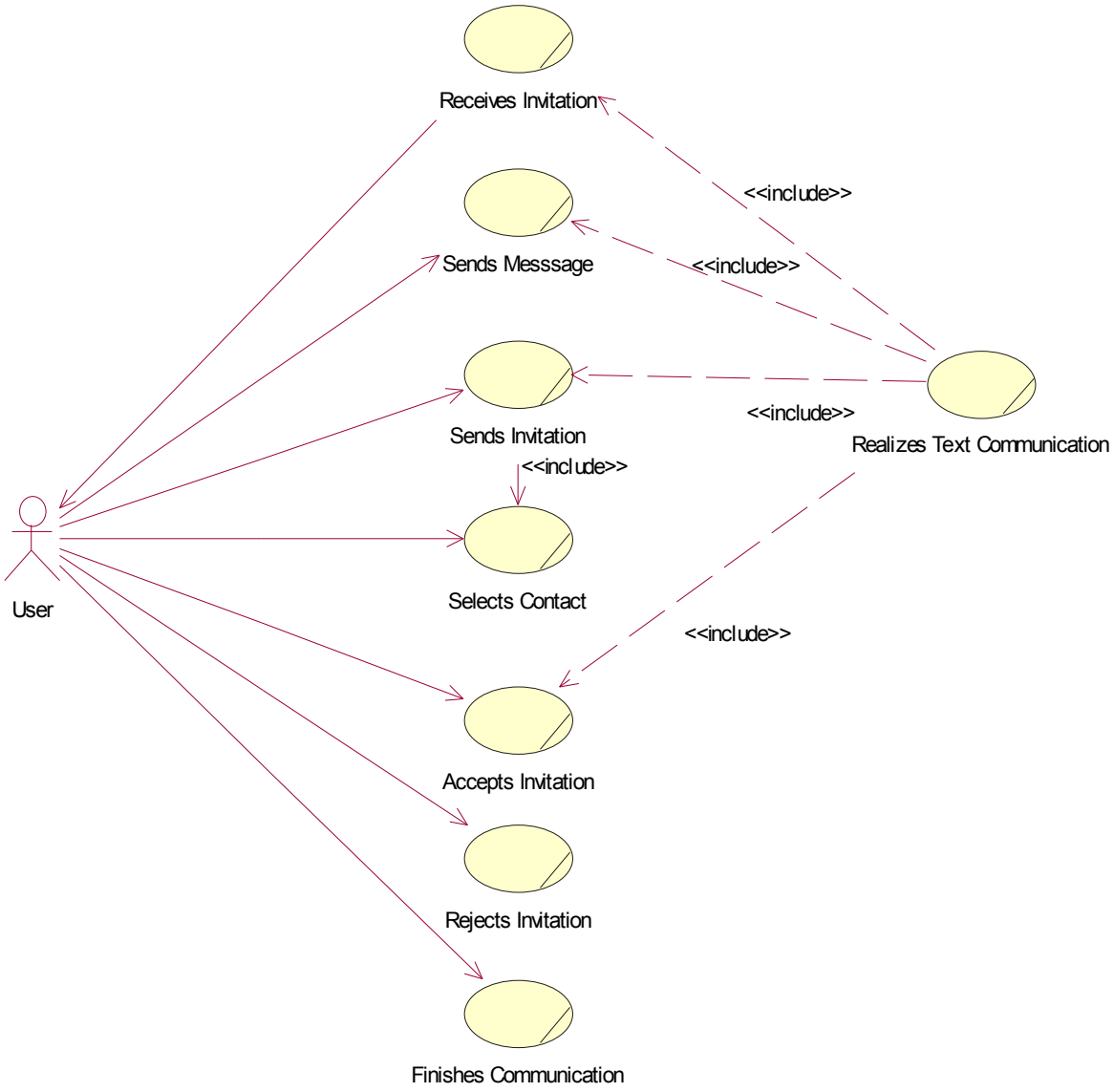

Figure 5 - Chat Use Cases

The application is made up of three main parts:

- User Interface: displays the application output, like command results and text received, and receives the user input, chat commands and text to send, to send to

the Main Application. This part encompasses the following class: UserInterface.

- Main Application: this is the part of the application that manages and connects all the components. It interprets user input and sends appropriate session commands; it receives session information and takes appropriate decisions, like generating output or sending session commands. This part encompasses the following classes: ChatMgr, Resolver.
- Session Layer: it deals with sockets, deals with and manages different sessions and waits for new connections. It sends information to the Main Application and receives commands from it. A session is implemented as an autonomous Finite State Machine. This part encompasses the following classes: SessionMgr, Session, SocketStream (Socket in the class diagram of Figure 6).

Figure 6 shows the class diagram of the system.

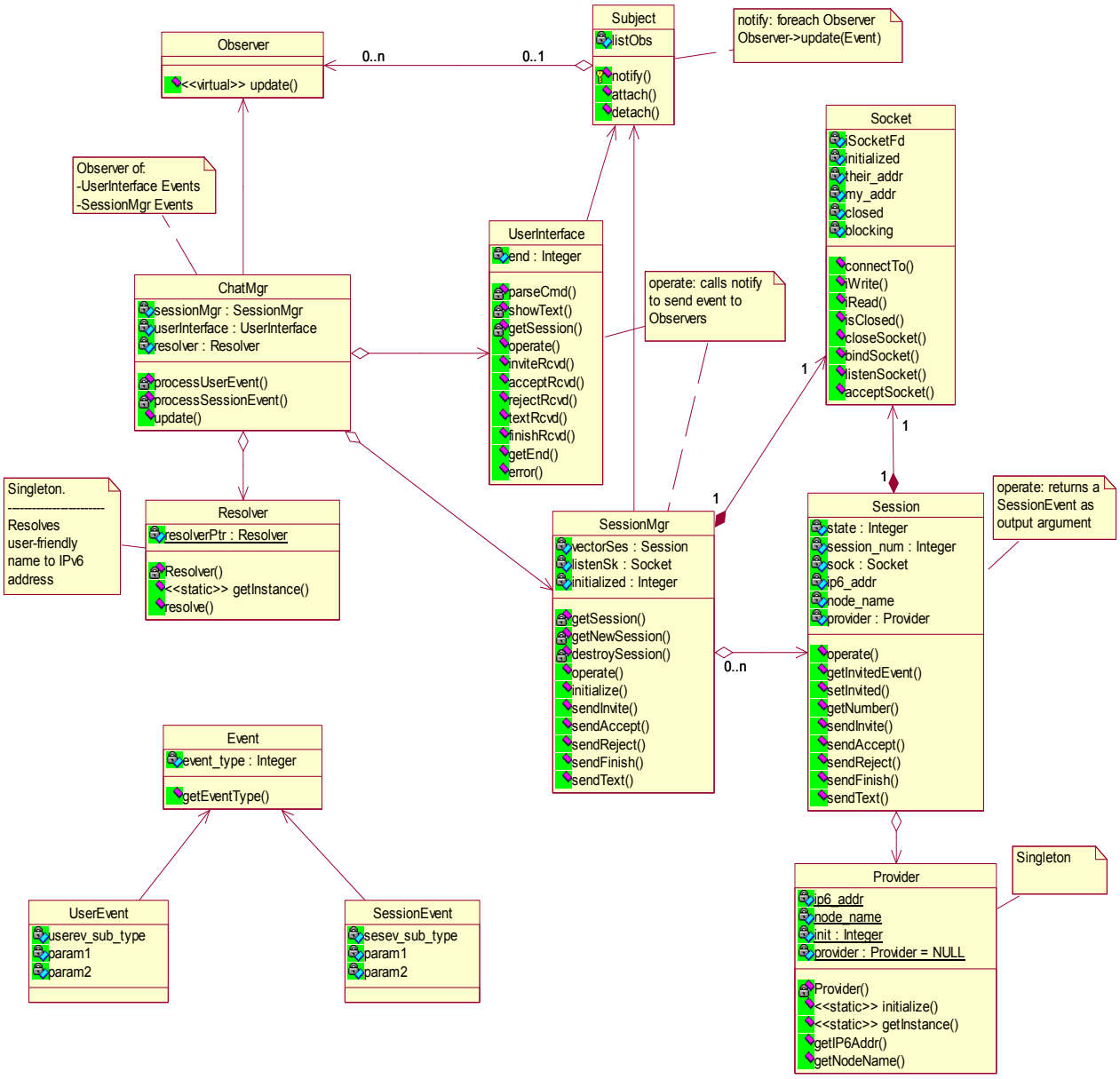

Figure 6 – Chat Class Diagram

A main function loops until the application is finished. In each loop it tells the SessionMgr object and the UserInterface object to operate.

ChatMgr implements the application layer behavior. It receives events from the Subject objects, UserInterface and SessionMgr and tells them to do something.

SessionMgr has a vector of Session objects and, being always told to operate, it loops over all the Session objects and tells each of them to operate. That is, to listen for messages and return new SessionEvent objects that are notified to the Observer objects, the ChatMgr object.

UserInterface is told to operate and it reads standard input to get user commands, it parses the commands and notifies UserEvent objects to the Observer objects, the ChatMgr object.

Sessions work as a Finite State Machine. So actions are performed depending on the current state and the event received. Figure 7 shows the different states that a chat session can have.

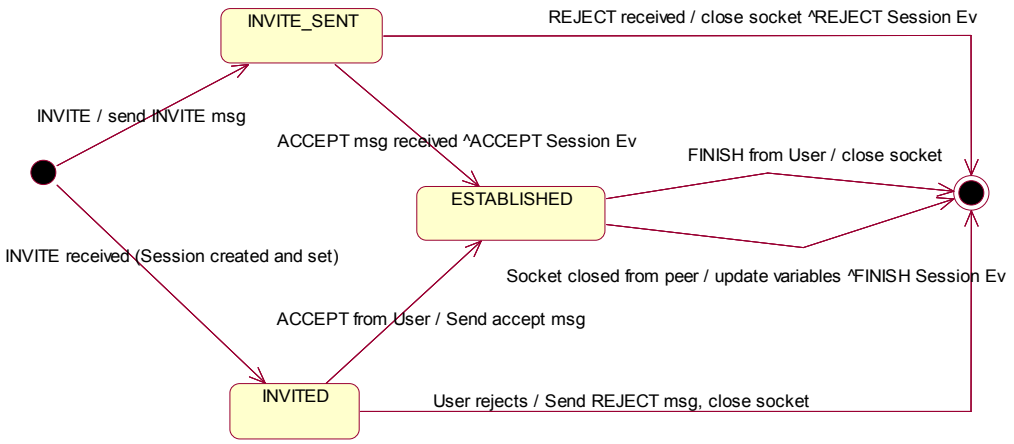

Figure 7 - Session States Diagram

Each established session has an integer identifier greater than zero.

The user can send two types of commands:

- Commands tied to a specific session. These are prefixed by the session identifier followed by a colon and the text to send.
- Commands tied to the application. These are prefixed by 0 (zero) followed by a colon and the application command.

In order to establish a new session, a user selects a contact and sends an invitation to that contact (locatable in a device with a fixed IPv6 home address). This is done with the application command INVITE. The invitation message can be sent to an IPv6 address or to a user-friendly name that is resolved to an IPv6 address.

```
0:invite 2002:C3D4:6FFD:1101:7232:4DFF:FECB:BE39
0:invite gondor
```

If the destination user accepts, it sends an ACCEPT message to the originator and a session establishment is finished; then, they can send chat messages to each other. An example of a user sending a message in session 1 is shown below.

```
1:Where are you?
```

Otherwise, the destination user sends a REJECT message.

In order to finish a chat session the user enters a FINISH command for that session. In the example, session 1 is finished.

```
0:finish 1
```

To finish the chat application the same command is used but the argument is 0 (zero).

11.4 Resolver

The Resolver is accessed by the ChatMgr if the destination of an INVITE message was entered as a name. The Resolver returns the IPv6 address corresponding to that name. The Resolver has a static mapping from certain names to IPv6 addresses; it does not access the DNS or any other name resolution system. Thus, whether the destination device is attached to the home network or to a visited network is transparent to the application.

12 Port of LIVSIX to uClinux over ColdFire

The steps followed to port the stack to ColdFire and to test it were:

1. Configuration and compilation of uClinux
2. Scripts and code modification and compilation.
3. Load of the stack module.
4. Ability to see and modify LIVSIX parameters.
5. Execution of modified ping6.
6. Reception of Routing Advertisements and stateless address auto-configuration.
7. Router settings on M5272C3.
8. Chat application.
9. Final Test.

12.1 Configuration and compilation of uClinux.

The first step was to set the environment in order to be able to compile the operating system with the appropriate configuration.

First of all, the GNU cross-compiler was installed: m68k-elf-gcc. All the other necessary tools, such as linker (m68k-elf-ld) and the library generator (m68k-elf-ar) were installed as well.

The next step was to configure uClinux so that the module could be compiled and built into the final image. The Kernel, applications and flash file system are built into a unique image that is finally uploaded in to the board flash memory.

The distribution of uClinux comes along with a set of applications found in the directory **user**; a directory called **livsix** was created here: **user/livsix**.

The file **user/Makefile** was modified in order to add an entry for the stack directory to compile the stack.

In order to add a help comment for configuration time the file **config/Configure.help** was updated with the following lines:

```
CONFIG_USER_LIVSIX
  Livsix MIPv6 stack.
  An open source mobile IPv6 stack from Motorola.
```

The file **config/config.in** was modified so that the LIVSIX stack could be selected during configuration:

```
comment 'Nice IPv6 stack'
bool 'LIVSIX'                   CONFIG_USER_LIVSIX
```

Into the directory user/livsix, three subdirectories were created:

- user/livsix/livsix: where the stack source code is located.
- user/livsix/utils: where utility functions and utilities library files are located.
- user/livsix/apps: where the chat application is located.

In order to configure uClinux the Makefile was executed with the xconfig argument according to what is indicated in the uClinux README file:

```
make config/xconfig/menuconfig
```

The following features were set:

- Loadable modules support enabled: to load the stack module.
- sysctl interface support enabled in the /proc file system: to be able to modify kernel settings.
- 16 MB RAM: this is the amount of memory the board M5272C3 has.
- LIVSIX selected.
- BusyBox applications selected: insmod, lsmod, rmmod. To be able of loading and stopping the module.
- Unnecessary applications were deselected: so that memory was not wasted

- Kernel 2.4.x selected.
- MCF5272 microprocessor selected.

By executing make in the top level directory the image was generated into a file: image/image.bin
This file could be loaded into the flash memory.

To be able to boot the embedded system and run the operating system a boot-loader had to be loaded into the flash memory: Colilo, ColdFire Linux Loader.

Colilo was installed at the beginning of the flash memory: 0xFFE00000
The image was compressed with *gzip* and loaded to the offset 0x40000: 0xFFE40000.

Having done this, the system can run with uClinux and the application.

12.2 Scripts and code modification and compilation

In order to be able to compile the stack for uClinux and ColdFire target, *autoconf* and *automake* scripts were modified. In each directory of the source code, the following files were modified:

- configure.in
- acinclude.m4
- Makefile.am

The modifications enabled:

- Use of the cross-compiler m68k-elf-gcc and other cross-compiling tools like m68k-elf-ar, m68k-elf-ld and others.
- Big Endian target compilation.

Also, to be able to automatically compile for ColdFire/uClinux target, the following scripts

were modified:

- config.sub
- config.guess

Some modifications were needed in the source code to adapt it to the new architecture.

The first problem to solve was the endianship; the stack had been tested in Pentium processors that use little-endian byte order. ColdFire uses big-endian byte order. The type of byte ordering the processor uses is called host type order. Furthermore, in networking protocols a network byte order must be specified. Internet protocols use big-endian order.

Little-endian byte order:

Address A	Address A+1
low-order byte	high-order

Big-endian byte order:

Address A	Address A+1
high-order	low-order byte

This is a complex issue since each line of code where integers of more than 1 Byte are managed must be checked.

When networking protocols are programmed and integer type fields are exchanged between nodes, there are certain functions that must be always used when the packets

are formed to send and when they are received and data and protocol defined fields are read.
The function prototypes are found in <netinet/in.h>

```
#include <netinet/in.h>
```

When setting integer packet fields the byte order must be converted from host order to network order:

uint16_t htons(uint16_t host16bitvalue);

uint32_t htonl(uint32_t host32bitvalue);

When retrieving integer packet fields the byte order must be converted from network order to host order:

uint16_t ntohs(uint16_t net16bitvalue);

uint32_t ntohl(uint32_t net32bitvalue);

In systems where the host order is equal to the network order, these functions are defined as null macros.

Luckily LIVSIX developers programmed the stack in the right way, using these functions whenever necessary. Only *autoconf* scripts had to be modified in order to support the correct byte-order type for ColdFire and a few modifications in the source code.

The main problems found regarding endianship were:

- Cases where an integer type pointer that points to the correct type value is type casted to a pointer of different size: this practice must be avoided in general. This usually works properly with little-endian systems while this is not correct in big-endian systems. A typical example of code that will work in little-endian but it will not execute correctly in big-endian follows:

```
uint8_t v1;
uint32_t *p1;

v1 = 2;
p1 = (uint32_t *)&v1;

if (p1 == 2) ...
```

- Cases where multi-byte integers must be managed one byte at a time. The bytes must not be accessed directly from memory if the code is meant to be portable. The following code will not execute correctly in a big-endian architecture.

```
uint16_t v1;
uint8_t  *a1;

a1 = &v1;

if (a1[0] == 0xFE)
...
```

The next point to consider was the fact that the microprocessor used has no support for floating-point numbers; it can be added optionally though. Places where floating-point is used had to be modified in order to use fixed-point numbers. Only one file had to be modified to solve this problem.

The fact that the microprocessor does not have Memory Management Unit caused some complications: there is no virtual memory; the system organizes the memory in blocks of different sizes. In order to be able to load the module other necessary applications were not run so that a suitable-size block of memory was available to allocate for the stack module. In case this does not work, the size of the memory blocks could have been modified.

TCP is working properly in LIVSIX but when the chat application was tested some bugs were found out and solved:

- While one connection was established the chat application was not able to accept a second connection: this was caused by a problem to access the connections related to a specific port and made the three way handshake to abort for connection requests.
- Non-blocking sockets are used. When accept is called in order to listen for new connections a structure for the new socket is allocated by the Linux and uClinux kernel, if there is no new connection established, i.e. with three-way handshake terminated, the kernel releases the structure calling the release function provided by the stack. When the last Byte of the structure address was equal to the last Byte of the listening socket address the listening socket was released.
- Linux uses 2 different constants to check whether the socket is non-blocking: 0x40 and 0400

12.3 Load of the stack module

The first step to make the stack work was to successfully load the LIVSIX module in uClinux.
In order to do it, one interface must be up. This can be achieved with the command **ifconfig**:

```
/> ifconfig eth0 192.168.0.12
eth0: config: auto-negotiation on, 100FDX, 100HDX, 10FDX, 10HDX.
/> eth0: status: link up, 100MBit Full Duplex, auto-negotiation
complete.
```

After that the module is loaded with **insmod**:

```
/> cd bin
/bin> insmod livsix.o
```

```
Using livsix.o
LIVSIXv0.3 Loaded
```

The only problem found was the one explained in `Scripts and code modification and compilation` regarding the lack of a suitable memory block to load the module.

12.4 Ability to see and modify LIVSIX parameters

There are a number of parameters to configure LIVSIX but the basic ones are managed with the LIVSIX command **livconfig**. The next step was to be able to execute this command to observe the parameter values and to modify some of them.

The following commands set the Security Policy Databases (-s), the home agent (-h) and route optimization (--ro, this feature is not fully implemented):

```
/bin> livconfig -s o ::/0 ::/0 0 0 0 0 0 1
/bin> livconfig -s i ::/0 ::/0 0 0 0 0 0 1
/bin> livconfig -h 2002:C3D4:6FFD:1101:2C0:26FF:FEB5:A24
Setting Home Agent to 2002:C3D4:6FFD:1101:2C0:26FF:FEB5:A24...
/bin> livconfig --ro enable
RO set
…
/bin> livconfig
eth0:
2002:C3D4:6FFD:1101:5258:1FF:FE9D:7EB8 ->(Home Address)
FE80::5258:1FF:FE9D:7EB8

lo:
::0.0.0.1

Inbound Security Policy Database:
1. SRC: * DST: *
Policy: BYPASS
```

```
Outbound Security Policy Database:
1. SRC: * DST: *
Policy: BYPASS
```

In the figure above the Aggregatable Global Unicast address, the Link-Local address and the loopback address can be seen.

12.5 Execution of modified ping6

One of the most important achievements was to run a modified version of the GNU file **ping6.c**.

After some work ping6 was tested successfully for global and link-local addresses, between a ColdFire processor and a PC and between two ColdFire processors.

The main problem was that the **inet_pton6** function, to convert a string address to a binary address, was not available because the Operating System must be configured without IPv6 support. The LIVSIX version of this function was used.

12.6 Reception of Routing Advertisements and stateless address auto-configuration

The reception of Routing Advertisements is important for movement detection and for stateless address auto-configuration. In the tests done, global addresses are auto-configured based on the prefixes advertised in Router Advertisements and the MAC (physical) addresses.

12.7 Router settings on M5272C3

When tests where done between only two MCF5272 boards connected by an Ethernet network, one of them had to be configured as Router so that it could send out Router Advertisements at variable intervals and the other board could create a global address.

In order to configure the router a program, **setrouter.c**, was created to update the following parameters through the *sysctl* interface:

- Prefix to advertise: in file /proc/sys/net/livsix/conf/eth0/ra_prefix_00/ra_pfl_prefix.
- Prefix length: in file /proc/sys/net/livsix/conf/eth0/ra_prefix_00/ra_pfl_prefixlen.
- Maximum interval between RA's: in file "/proc/sys/net/livsix/conf/eth0/ra_maxra_interval.
- Minimum interval between RA's: in file "/proc/sys/net/livsix/conf/eth0/ra_minra_interval.
- Enabling RA sending: in file /proc/sys/net/livsix/conf/eth0/ra_send_ras
- Setting the node as Router: in file /proc/sys/net/livsix/isrouter.

Other programs were created to set other parameters in different type of nodes:

- **setdefint.c**: to set the default interface to send out packets.
- **setval.c**: to set a value into a specific file.

12.8 Chat application over TCP

The chat application has been described above.

12.9 Final Test

The final demonstrated the chat application worked correctly even though the device

was attached to network different than the home network. No new sockets had been opened and the existing sessions kept working correctly. The socket to which the application was listening also kept working correctly and continued accepting new TCP connections.

The final testbed and test are described in the following section.

13 Application and stack test

13.1 Testbed

The final testbed involved the following configuration:

- 2 Networks:
    - **network1** prefix: 2002:c3d4:6ffd:1101::/64
    - **network2** prefix: 2002:c3d4:6ffd:2202::/64
- 1 PC, with Linux kernel 2.4.21, called **shire**, running LIVSIX set as HA in network1. Global address (eth0): 2002:C3D4:6FFD:1101:2C0:26FF:FEB5:A24/64
- 1 PC, with Linux kernel 2.4.21, called **gondor**, running LIVSIX set as router with 2 interfaces:
    - eth0 to network1. Global address: 2002:C3D4:6FFD:1101:2E0:7DFF:FEE1:FBC1/64
    - eth1 to network2. Global address: 2002:C3D4:6FFD:2202:208:54FF:FE03:FFF1/64
- 2 M5272C3 boards, with uClinux kernel 2.4.19, set as MN, initially connected to network1. Home addresses set:
    - EB1, eth0:
        - Global address: 2002:C3D4:6FFD:1101:5258:1FF:FE9D:7EB8
        - Link-local address: FE80::5258:1FF:FE9D:7EB8
    - EB2, eth0:
        - Global address: 2002:C3D4:6FFD:1101:7232:4DFF:FECB:BE39
        - Link-local address: FE80::7232:4DFF:FECB:BE39

Details to configure each network element can be found at [8] [9] [10].

During the initial configuration both boards were connected to **network1**, EB1 was connected through the hub HUB1 while EB2 was connected to the hub HUB2 which, in turn, was connected to HUB1. This was done in this way because the way of simulating movement between **network1** and **network2** was by disconnecting the board from **network1** and connecting it to **network2**. The problem was that when the board is disconnected the situation was detected by the board Ethernet device driver and this affected LIVSIX. Then the board to be moved, EB2, was connected to HUB2 thus it was HUB2 which was disconnected from network1 and connected to network2 while from EB2's standpoint the link was always up. Figure 8 shows the initial configuration.

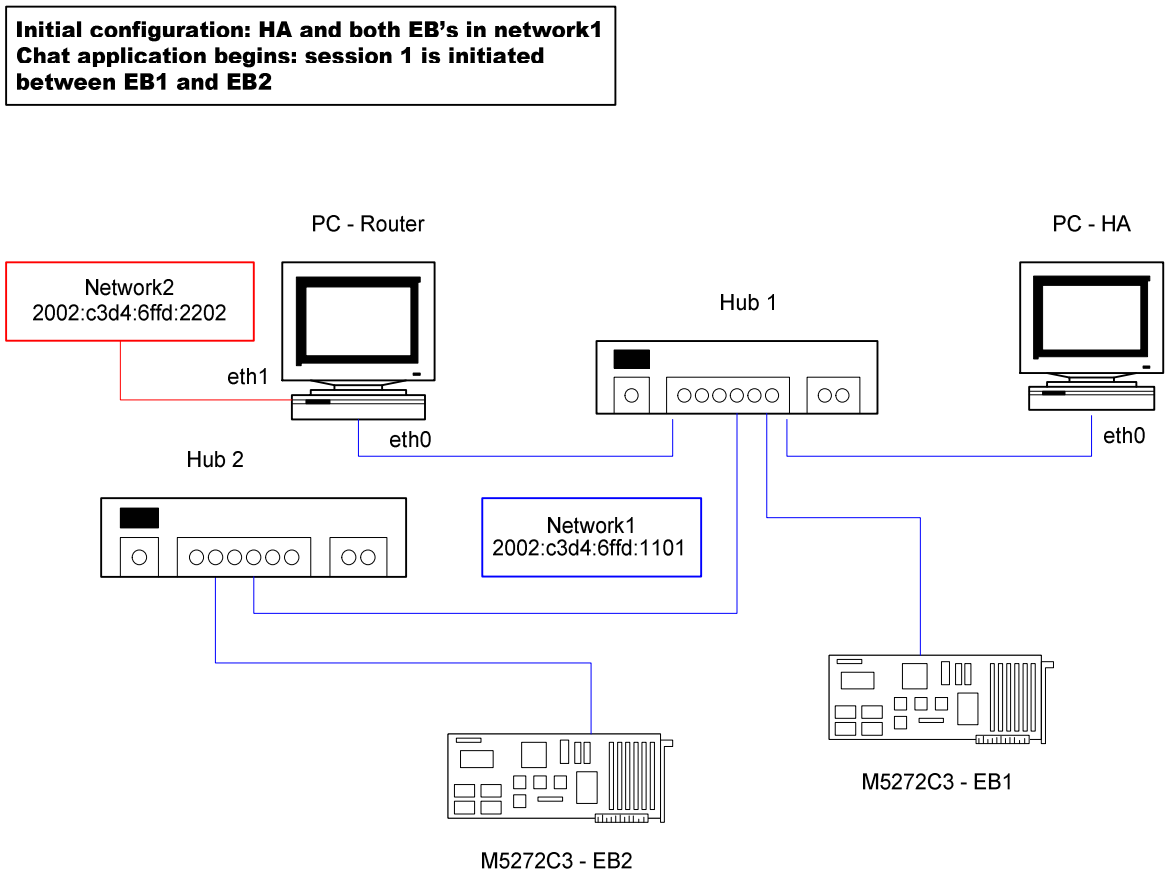

Figure 8 - Initial configuration

The chat application began in each M5272C3, EB1 sent an INVITE (application layer) message to EB2, when the message was received by EB2, a new session was initiated, EB2 accepted the invitation and sent an accept (application) message to the other one. They began to chat. In the middle of the chat, EB2 –actually HUB2 - was disconnected from network1 and was connected to network2, it detected the movement after having received a Router Advertisement sent out by the Router's eth1 interface with a prefix different than that of the home network. Figure 9 shows the configuration after the network change has been realized.

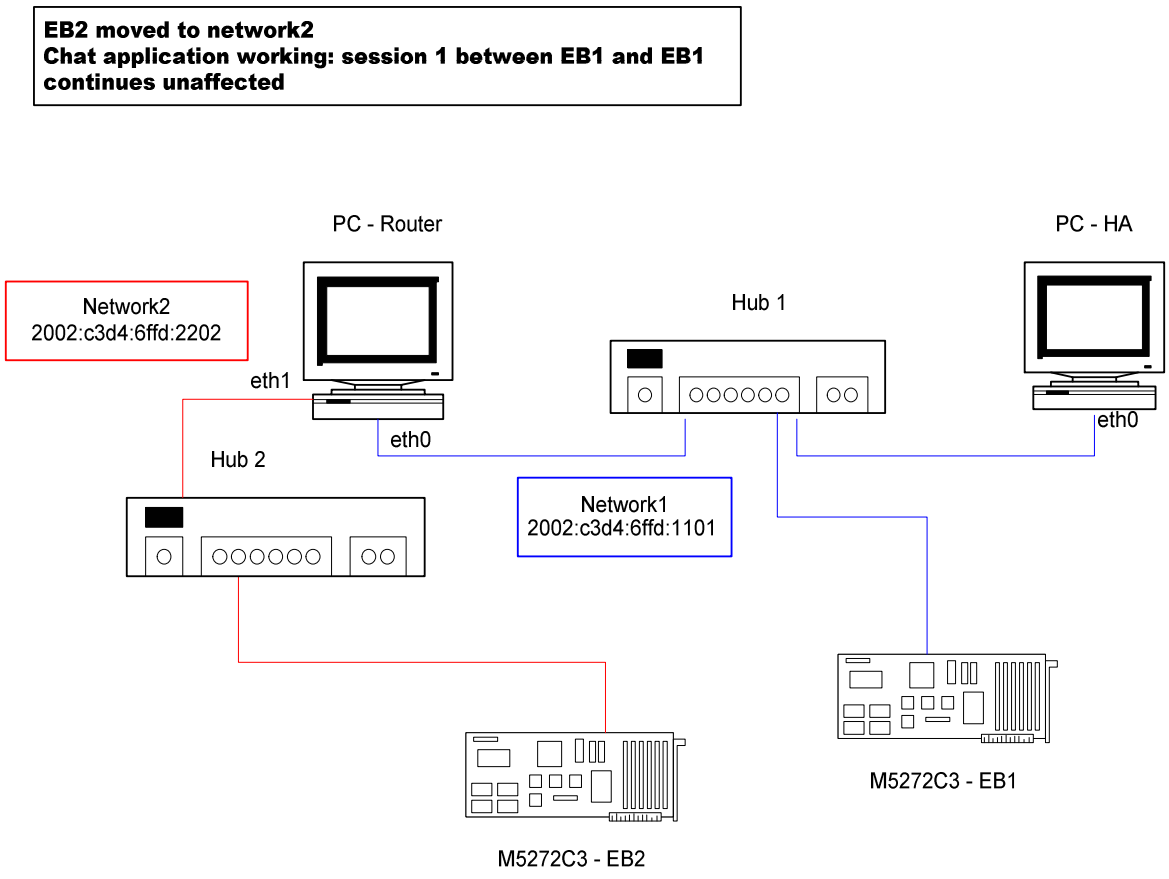

Figure 9 - EB2 is connected to network2

This was transparent for the TCP layer. The connection and the session were kept and the chat continued normally.

Figure 10 shows the hardware set initially in the place where the test was executed.

Figure 10 - The Hardware initially set.

The board at the left is EB1; it is connected to the hub on the PC case, HUB1. The other board is EB2 and it is connected to HUB2, the one at the left, next to the monitor. HUB2 is initially connected to HUB1 with the grey cable.

The PC on the right is the one used as HA. It is also connected through RS-232 with both boards. The PC on the left is the one used as router. In the router, eth0 is connected to HUB1 with a blue cable while eth1 has the yellow cable connected to it. The other point of the yellow cable is not connected. In order to get to the configuration shown in Figure 9, HUB2's grey cable will be disconnected and the yellow cable will be connected to it. This is shown in Figure 11.

Figure 11 - HUB2 connections after handover.

13.2 Tests description

With the testbed described above, two tests have been successfully performed:

- Transparent Movement from one network to other.
- One-way MIPv6 Route Optimization (from CN to MN).

Both boards were connected to the Linux PC set as HA through RS-232 serial connections. EB1 was connected to COM1 whereas EB2 was connected to COM2. A serial terminal emulator called **microcom** was used and was configured as:

- COM interface as 19200 bauds.
- Log session to a file.

Two Ethereal instances were executing from the beginning: one in **shire**, sniffing on eht0, the second one in **gondor**, sniffing on eth0 and after the “handover”, on eth1.

In the first test, initially, the layout of Figure 8 was set. Both boards were turned on, uClinux initialized and the steps below were followed in the shown order, in each board:

1. Interface eth0 was set up with ifconfig: IPv4 addresses 192.168.0.11 and 192.168.0.12 for EB1 and EB2 respectively.
2. LIVSIX module **livsix.o** was loaded.
3. Interface eth0 was set as default by updating file /proc/sys/net/livsix/conf/eth0/defint with setdefint utility.
4. Outbound SPD was configured with “livconfig –s”.
5. Inbound SPD was configured.
6. Home agent was to 2002:C3D4:6FFD:1101:2C0:26FF:FEB5:A24 set with “livconfig –h”.
7. Route Optimization was set, though this feature is not fully implemented.
8. livconfig utility was run to show the initial configuration.

After that, the home agent was run in the HA Linux PC, **shire**. This is shown in Figure

12.

```
[root@shire userspace]# livsix.sh start
Starting LIVSIX:                                           [  OK  ]
Homeaddress set to 2002:C3D4:6FFD:1101:2C0:26FF:FEB5:A24
Default Interface set to eth0
LIVSIX box configured as Home Agent
eth0:
FE80::2C0:26FF:FEB5:A24

lo:
::0.0.0.1

Inbound Security Policy Database:
1. SRC: * DST: *
Policy: BYPASS

Outbound Security Policy Database:
1. SRC: * DST: *
Policy: BYPASS

[root@shire userspace]# ifconfig eth0 add 2002:C3D4:6FFD:1101:2C0:26FF:FEB5:A24
[root@shire userspace]# livconfig -b
livconfig: Binding Cache:
HOME ADDRESS                            CARE-OF ADDRESS
lt
```

Figure 12 - HA was initiated.

The router was initialized in the Router PC, **gondor**. This is shown in Figure 13.

```
[root@gondor userspace]# ./livsix.sh start
Starting LIVSIX:                                           [  OK  ]
LIVSIX box configured as Router
eth1:
FE80::208:54FF:FE03:FFF1

eth0:
FE80::2E0:7DFF:FEE1:FBC1

lo:
::0.0.0.1

Inbound Security Policy Database:
1. SRC: * DST: *
Policy: BYPASS

Outbound Security Policy Database:
1. SRC: * DST: *
Policy: BYPASS

[root@gondor userspace]# ifconfig eth0 add 2002:c3d4:6ffd:1101:2E0:7DFF:FEE1:FBC1
[root@gondor userspace]# ifconfig eth1 add 2002:c3d4:6ffd:2202:208:54FF:FE03:FFF1
[root@gondor userspace]# ./addroutes
[root@gondor userspace]# ./setrouter
```

Figure 13 - Router was initiated.

After having called **setrouter** the following configuration was set in the Kernel IPv6 routing table.

```
[root@gondor userspace]# route -A inet6
Kernel IPv6 routing table
Destination                                 Next Hop    Flags Metric Ref    Use Iface
2002:c3d4:6ffd:1101::/64                    ::    U     1      1        0 eth0
2002:c3d4:6ffd:2202::/64                    ::    U     1      1        0 eth1
```

Figure 14 - Kernel IPv6 routing table.

Once the application **setrouter** was run, the router began to send out Router Advertisements in both networks.

Looking at 18.6 (frames 173, 202, 439, and 552), it is possible to see that each node, including the router, sent out a Router Solicitation ICMPv6 message. In the router PC, LIVSIX sent this type of message before having been configured as router. Frame 775 is the first Router Advertisement sent by the router. After all nodes received it, they began to send IP packets with the auto-configured global address as source address (from frame 964 up).

The chat application was initiated in both boards. As it can be seen in 18.2 , from EB1 an INVITE message was sent to the home IPv6 address of EB2.

```
>>0:invite 2002:C3D4:6FFD:1101:7232:4DFF:FECB:BE39
```

Then a Neighbor Solicitation ICMPv6 packet was sent, this is frame 964 in 18.6 , and until frame 1018, several Neighbor Solicitation and Neighbor Advertisement packets are logged (in 18.4 , frames 1422-1474). Finally, EB1 got the MAC address of EB2, this frame cannot be seen because it was sent directly from EB2 to EB1 and the hubs did not deliver the frame to all the networks.

Then the first TCP message over MIPv6, a SYN, was sent from EB1 to EB2, the 3-way handshake was produced and the INVITE message was sent. EB2 received the

notification and sent the ACCEPT (18.3).

```
*** 1:INVITE RECEIVED FROM IP 2002:c3d4:6ffd:1101:5258:01ff:fe9d:7eb8
NODE: uClinux***
>>0:accept 1
```

When EB1 received the ACCEPT message it sent the first text message (18.2).

```
*** 1:ACCEPT RECEIVED FROM IP 2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
NODE: ***
>>1:HELLO!
```

All these frames cannot be seen in the Ethereal logs.

After a while, HUB2, and EB2 with it, was disconnected from **network1** and was connected to **network2**.

In **gondor**, before connecting HUB2 to **network2**, Ethereal was set to sniff on eth1.

Once EB2 was connected to **network2**, from this action all the frames, until the chat session is closed, are shown in Table 1. **gondor** detected the link on eth1 as active and a Router Solicitation was sent to the all routers multicast address {1} and it delivered a Router Advertisement to the all hosts multicast address {2}.

N	Time	Source	Destination	Prot.	Protocol Info
1	0.000000	fe80::208:54ff:fe03:fff1	ff02::2	ICMPv6	ICMPv6 Router solicitation
2	0.000098	fe80::208:54ff:fe03:fff1	ff02::1	ICMPv6	ICMPv6 Router advertisement
3	0.001911	2002:c3d4:6ffd:2202:7232:4dff:fecb:be39	2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24	IPv6	Unknown (0x3e)
4	0.005327	2002:c3d4:6ffd:2202:208:54ff:fe03:fff1	2002:c3d4:6ffd:2202:7232:4dff:fecb:be39	ICMPv6	Neighbor solicitation
5	0.006563	2002:c3d4:6ffd:2202:7232:4dff:fecb:be39	2002:c3d4:6ffd:2202:208:54ff:fe03:fff1	ICMPv6	Neighbor solicitation
6	0.007477	2002:c3d4:6ffd:2202:208:54ff:fe03:fff1	2002:c3d4:6ffd:2202:7232:4dff:fecb:be39	ICMPv6	Neighbor advertisement
7	0.008312	2002:c3d4:6ffd:2202:7232:4dff:fecb:be39	2002:c3d4:6ffd:2202:208:54ff:fe03:fff1	ICMPv6	Neighbor advertisement
8	0.009096	2002:c3d4:6ffd:1101:	2002:c3d4:6ffd:2202:	IPv6	Unknown (0x3e)

		2c0:26ff:feb5:a24	7232:4dff:fecb:be39		
9	23.447565	2002:c3d4:6ffd:1101: 7232:4dff:fecb:be39	2002:c3d4:6ffd:1101: 5258:1ff:fe9d:7eb8	TCP	1514 > 49152 [ACK] Seq=12345741 Ack=12345734 Win=37782 Len=46
10	23.585459	2002:c3d4:6ffd:1101: 5258:1ff:fe9d:7eb8	2002:c3d4:6ffd:1101: 7232:4dff:fecb:be39	TCP	49152 > 1514 [ACK] Seq=12345734 Ack=12345787 Win=29954 Len=0
11	33.106548	2002:c3d4:6ffd:1101: 5258:1ff:fe9d:7eb8	2002:c3d4:6ffd:1101: 7232:4dff:fecb:be39	TCP	49152 > 1514 [ACK] Seq=12345734 Ack=12345787 Win=29954 Len=17
12	33.166377	2002:c3d4:6ffd:1101: 7232:4dff:fecb:be39	2002:c3d4:6ffd:1101: 5258:1ff:fe9d:7eb8	TCP	1514 > 49152 [ACK] Seq=12345787 Ack=12345751 Win=37782 Len=0
13	68.897671	2002:c3d4:6ffd:1101: 7232:4dff:fecb:be39	2002:c3d4:6ffd:1101: 5258:1ff:fe9d:7eb8	TCP	1514 > 49152 [ACK] Seq=12345787 Ack=12345751 Win=37782 Len=10
14	68.985326	2002:c3d4:6ffd:1101: 5258:1ff:fe9d:7eb8	2002:c3d4:6ffd:1101: 7232:4dff:fecb:be39	TCP	49152 > 1514 [ACK] Seq=12345751 Ack=12345797 Win=29954 Len=0
15	81.017736	2002:c3d4:6ffd:1101: 7232:4dff:fecb:be39	2002:c3d4:6ffd:1101: 5258:1ff:fe9d:7eb8	TCP	1514 > 49152 [FIN, ACK] Seq=12345797 Ack=12345751 Win=37782 Len=0
16	81.185318	2002:c3d4:6ffd:1101: 5258:1ff:fe9d:7eb8	2002:c3d4:6ffd:1101: 7232:4dff:fecb:be39	TCP	49152 > 1514 [ACK] Seq=12345751 Ack=12345798 Win=29954 Len=0
17	81.585708	2002:c3d4:6ffd:1101: 5258:1ff:fe9d:7eb8	2002:c3d4:6ffd:1101: 7232:4dff:fecb:be39	TCP	49152 > 1514 [FIN, ACK] Seq=12345751 Ack=12345798 Win=29954 Len=0

Table 1 - Frames on eth1 of the Router

After that, EB2 detected it was in a foreign network and sent a Binding Update to the HA {3}. The Next Header type for the Mobility Header is 0x3E (Ethereal did not recognize it). The reply is {8}. Once the successful reply was received, the TCP communication continued normally. Frame {9} can be seen below, containing the chat message that appears in 18.3 . Messages between EB1 and EB2 are conveyed through a bidirectional tunnel.

```
Frame 9 (172 bytes on wire, 172 bytes captured)
    Arrival Time: Apr 19, 2004 01:02:05.251106000
    Time delta from previous packet: 23.438469000 seconds
    Time relative to first packet: 23.447565000 seconds
    Frame Number: 9
    Packet Length: 172 bytes
    Capture Length: 172 bytes
Ethernet II, Src: 70:32:4d:cb:be:39, Dst: 00:08:54:03:ff:f1
    Destination: 00:08:54:03:ff:f1 (Netronix_03:ff:f1)
    Source: 70:32:4d:cb:be:39 (70:32:4d:cb:be:39)
    Type: IPv6 (0x86dd)
Internet Protocol Version 6
    Version: 6
    Traffic class: 0x00
    Flowlabel: 0x00000
    Payload length: 118
    Next header: IPv6 (0x29)
    Hop limit: 255
    Source address: 2002:c3d4:6ffd:2202:7232:4dff:fecb:be39
    Destination address: 2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24
Internet Protocol Version 6
    Version: 6
    Traffic class: 0x00
    Flowlabel: 0x00000
    Payload length: 78
    Next header: TCP (0x06)
    Hop limit: 255
    Source address: 2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
    Destination address: 2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8
Transmission Control Protocol, Src Port: 1514 (1514), Dst Port: 49152 (49152), Seq:
12345741, Ack: 12345734, Len: 46
    Source port: 1514 (1514)
    Destination port: 49152 (49152)
    Sequence number: 12345741
    Next sequence number: 12345787
    Acknowledgement number: 12345734
    Header length: 32 bytes
    Flags: 0x0010 (ACK)
        0... .... = Congestion Window Reduced (CWR): Not set
        .0.. .... = ECN-Echo: Not set
        ..0. .... = Urgent: Not set
        ...1 .... = Acknowledgment: Set
        .... 0... = Push: Not set
        .... .0.. = Reset: Not set
        .... ..0. = Syn: Not set
        .... ...0 = Fin: Not set
    Window size: 37782
    Checksum: 0x276d (correct)
    Options: (12 bytes)
        NOP
        NOP
        Time stamp: tsval 964, tsecr 464
Data (46 bytes)

0000  00 03 00 2e 49 27 76 65 20 6d 6f 76 65 64 20 74   ....I've moved t
0010  6f 20 6e 65 74 77 6f 72 6b 32 2e 20 49 27 6d 20   o network2. I'm
0020  69 6e 20 6e 65 74 77 6f 72 6b 32 21 21 21         in network2!!!
```

Figure 15 - Frame 9. Segment sent from EB2 in network2.

In network1, the frames in Table 2 were seen in eth0 of shire (HA) since the Binding Update was received until the chat session was closed.

N	Time	Source	Destination	Prot.	Protocol Info
600	36.059687	2002:c3d4:6ffd:2202:7232:4dff:fecb:be39	2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24	IPv6	Unknown (0x3e)
601	36.059988	2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24	ff02::1	ICMPv6	ICMPv6 Neighbor advertisement
602	36.060193	2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24	ff02::2	ICMPv6	ICMPv6 Neighbor advertisement
603	36.060483	2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24	2002:c3d4:6ffd:2202:7232:4dff:fecb:be39	IPv6	Unknown (0x3e)
634	59.508966	2002:c3d4:6ffd:1101:7232:4dff:fecb:be39	2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8	TCP	1514 > 49152 [ACK] Seq=12345741 Ack=12345734 Win=37782 Len=46
635	59.509124	2002:c3d4:6ffd:1101:7232:4dff:fecb:be39	2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8	TCP	1514 > 49152 [ACK] Seq=12345741 Ack=12345734 Win=37782 Len=46
636	59.644921	2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8	2002:c3d4:6ffd:1101:7232:4dff:fecb:be39	TCP	49152 > 1514 [ACK] Seq=12345734 Ack=12345787 Win=29954 Len=0
637	59.645169	2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8	2002:c3d4:6ffd:1101:7232:4dff:fecb:be39	TCP	49152 > 1514 [ACK] Seq=12345734 Ack=12345787 Win=29954 Len=0
640	69.167487	2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8	2002:c3d4:6ffd:1101:7232:4dff:fecb:be39	TCP	49152 > 1514 [ACK] Seq=12345734 Ack=12345787 Win=29954 Len=17
641	69.167754	2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8	2002:c3d4:6ffd:1101:7232:4dff:fecb:be39	TCP	49152 > 1514 [ACK] Seq=12345734 Ack=12345787 Win=29954 Len=17
642	69.229518	2002:c3d4:6ffd:1101:7232:4dff:fecb:be39	2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8	TCP	1514 > 49152 [ACK] Seq=12345787 Ack=12345751 Win=37782 Len=0
643	69.229655	2002:c3d4:6ffd:1101:7232:4dff:fecb:be39	2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8	TCP	1514 > 49152 [ACK] Seq=12345787 Ack=12345751 Win=37782 Len=0
646	104.966228	2002:c3d4:6ffd:1101:7232:4dff:fecb:be39	2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8	TCP	1514 > 49152 [ACK] Seq=12345787 Ack=12345751 Win=37782 Len=10
647	104.966388	2002:c3d4:6ffd:1101:7232:4dff:fecb:be39	2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8	TCP	1514 > 49152 [ACK] Seq=12345787 Ack=12345751 Win=37782 Len=10
648	105.051984	2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8	2002:c3d4:6ffd:1101:7232:4dff:fecb:be39	TCP	49152 > 1514 [ACK] Seq=12345751 Ack=12345797 Win=29954 Len=0
649	105.052233	2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8	2002:c3d4:6ffd:1101:7232:4dff:fecb:be39	TCP	49152 > 1514 [ACK] Seq=12345751 Ack=12345797 Win=29954 Len=0
654	117.088198	2002:c3d4:6ffd:1101:7232:4dff:fecb:be39	2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8	TCP	1514 > 49152 [FIN, ACK] Seq=12345797 Ack=12345751 Win=37782 Len=0
655	117.088354	2002:c3d4:6ffd:1101:7232:4dff:fecb:be39	2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8	TCP	1514 > 49152 [FIN, ACK] Seq=12345797 Ack=12345751 Win=37782 Len=0
656	117.253923	2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8	2002:c3d4:6ffd:1101:7232:4dff:fecb:be39	TCP	49152 > 1514 [ACK] Seq=12345751 Ack=12345798 Win=29954 Len=0
657	117.254172	2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8	2002:c3d4:6ffd:1101:7232:4dff:fecb:be39	TCP	49152 > 1514 [ACK] Seq=12345751 Ack=12345798 Win=29954 Len=0
660	117.654363	2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8	2002:c3d4:6ffd:1101:7232:4dff:fecb:be39	TCP	49152 > 1514 [FIN, ACK] Seq=12345751 Ack=12345798 Win=29954 Len=0
661	117.654615	2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8	2002:c3d4:6ffd:1101:7232:4dff:fecb:be39	TCP	49152 > 1514 [FIN, ACK] Seq=12345751 Ack=12345798 Win=29954 Len=0

Table 2 - Frames on eth0 of the HA

After having received the Binding Update {600}, the HA sent out two Neighbor

Advertisement ICMPv6 messages, one to all nodes multicast address {601} and another one to all routers multicast address {602}, to indicate that its link-layer address corresponds to EB2's IPv6 address in order to act as a proxy for EB2. Then the registration was acknowledged {604} and, from that moment, every message sent to EB2 was intercepted in **network1**, the home network, by the HA {636, 640, 648, 656, 660}, which tunneled the IP packet in a new packet destined to the care-of address {637, 641,649, 657, 661}, when EB2 received the packet, it obtained the packet inside.

Every packet from EB2 to EB1 was reverse-tunneled to the HA {634, 642, 646, 654} which sent the final packet to EB1 {635, 643, 647, 655}.

As it can be seen in Table 1 and Table 2, the session was closed by EB2 {654}. This is the log of EB2, since it sent the last message before moving, until it closed the session.

```
>>1:I'll move to network2
>>1:I've moved to network2. I'm in network2!!!
>>
*** 1:That's great!***
>>ok bye

*** PARSE ERROR: 109 ***
>>1:ok bye
>>0:finish 1
```

Figure 16 - Chat display while EB2 was moving.

It is possible to see that frame {9}, a TCP segment, in Table 1, was sent without performing a new 3-way handshake before and the communication continued normally. All TCP segments have the prefix 2002:c3d4:6ffd:1101::/64 in the source address because IPv6 packets were sent using bidirectional tunneling and Ethereal shows, in the summary, the source address of the inner packet. In the details of frame {9} the external packet has the prefix 2002:c3d4:6ffd:2202::/64. This segment belonged to a connection set when EB2 was attached to **network1**, otherwise, after having received it, EB1 would have sent a RESET segment, which, at least, would have been received by the HA and

shown in Table 2. Also the session continued with further segments being exchanged.

Thus, the movement of EB2 from **network1** to **network2** was transparent for the TCP layer. As a consequence, the application did not have to be aware of that situation.

Route optimization was partially tested since it is still not completely implemented in LIVSIX: after the first test, a binding was manually added in EB1's binding cache. The application **livconfig** [12] was used to do this.

Figure 17 shows the way it was done.

```
/bin>          livconfig          -a          2002:C3D4:6FFD:1101:7232:4DFF:FECB:BE39
2002:C3D4:6FFD:2202:7232:4DFF:FECB:BE 39 30000
Entry added to the Binding Cache
/bin> livconfig -b
livconfig: Binding Cache:
HOME ADDRESS                              CARE-OF ADDRESS                          lt
2002:C3D4:6FFD:1101:7232:4DFF:FECB:BE39 2002:C3D4:6FFD:2202:7232:4DFF:FECB:BE39 29994
```

Figure 17 – Setting a binding update in EB1

After doing this, the chat application was run again, a new session was initiated by EB1, it was established and some messages were exchanged. This is shown in Figure 18.

```
/bin> chat uc1 2002:C3D4:6FFD:1101:5258:1FF:FE9D:7EB8
HOST: uClinux
IP: 2002:C3D4:6FFD:1101:5258:1FF:FE9D:7EB8

>>0:invite 2002:C3D4:6FFD:1101:7232:4DFF:FECB:BE39
>>
*** 1:ACCEPT RECEIVED FROM IP 2002:c3d4:6ffd:1101:7232:4dff:fecb:be39 NODE: ***
>>1:Hello
>>
*** 1:Hi, it's you again!***
>>yep, bye

*** PARSE ERROR: 109 ***
>>1:yep bye
>>0:finish 1
>>0:finish 0
>>
End
/bin> livconfig -b
livconfig: Binding Cache:
HOME ADDRESS                              CARE-OF ADDRESS                           lt
2002:C3D4:6FFD:1101:7232:4DFF:FECB:BE39 2002:C3D4:6FFD:2202:7232:4DFF:FECB:BE39 29726
/bin>
```

Figure 18 - Chat on EB1 with RO.

Every message from the CN to the MN was destined to address 2002:C3D4:6FFD:2202:7232:4DFF:FECB:BE39 while every message from the MN to the CN was sent using reverse tunneling through the HA. Table 3 shows the TCP segments summary on eth1 of the Router, since a new session was initiated by EB1. Frames {19, 20, 21} correspond to the 3-way handshake and {21} conveys the INVITE message. It is possible to note the difference between Table 1 and Table 3 on the destination address of packets directed to EB2.

N	Time	Source	Destination	Prot.	Protocol Info
19	383.126189	2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8	2002:c3d4:6ffd:2202:7232:4dff:fecb:be39	TCP	49152 > 1514 [SYN] Seq=12345678 Ack=0 Win=30000 Len=0
20	383.221545	2002:c3d4:6ffd:1101:7232:4dff:fecb:be39	2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8	TCP	1514 > 49152 [SYN, ACK] Seq=12345678 Ack=12345679 Win=37800 Len=0
21	383.225012	2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8	2002:c3d4:6ffd:2202:7232:4dff:fecb:be39	TCP	49152 > 1514 [ACK] Seq=12345679 Ack=12345679 Win=37800 Len=27
22	384.220113	2002:c3d4:6ffd:1101:	2002:c3d4:6ffd:1101:	TCP	1514 > 49152 [ACK] Seq=12345679

		7232:4dff:fecb:be39	5258:1ff:fe9d:7eb8		Ack=12345706 Win=37800 Len=0
23	399.271254	2002:c3d4:6ffd:1101: 7232:4dff:fecb:be39	2002:c3d4:6ffd:1101: 5258:1ff:fe9d:7eb8	TCP	1514 > 49152 [ACK] Seq=12345679 Ack=12345706 Win=37800 Len=4
24	399.384050	2002:c3d4:6ffd:1101: 5258:1ff:fe9d:7eb8	2002:c3d4:6ffd:2202: 7232:4dff:fecb:be39	TCP	49152 > 1514 [ACK] Seq=12345706 Ack=12345683 Win=29996 Len=0
25	419.584996	2002:c3d4:6ffd:1101: 5258:1ff:fe9d:7eb8	2002:c3d4:6ffd:2202: 7232:4dff:fecb:be39	TCP	49152 > 1514 [ACK] Seq=12345706 Ack=12345683 Win=29996 Len=9
26	419.654114	2002:c3d4:6ffd:1101: 7232:4dff:fecb:be39	2002:c3d4:6ffd:1101: 5258:1ff:fe9d:7eb8	TCP	1514 > 49152 [ACK] Seq=12345683 Ack=12345715 Win=37791 Len=0
27	432.601719	2002:c3d4:6ffd:1101: 7232:4dff:fecb:be39	2002:c3d4:6ffd:1101: 5258:1ff:fe9d:7eb8	TCP	1514 > 49152 [ACK] Seq=12345683 Ack=12345715 Win=37791 Len=23
28	432.783956	2002:c3d4:6ffd:1101: 5258:1ff:fe9d:7eb8	2002:c3d4:6ffd:2202: 7232:4dff:fecb:be39	TCP	49152 > 1514 [ACK] Seq=12345715 Ack=12345706 Win=29996 Len=0
29	446.854926	2002:c3d4:6ffd:1101: 5258:1ff:fe9d:7eb8	2002:c3d4:6ffd:2202: 7232:4dff:fecb:be39	TCP	49152 > 1514 [ACK] Seq=12345715 Ack=12345706 Win=29996 Len=11
30	447.050742	2002:c3d4:6ffd:1101: 7232:4dff:fecb:be39	2002:c3d4:6ffd:1101: 5258:1ff:fe9d:7eb8	TCP	1514 > 49152 [ACK] Seq=12345706 Ack=12345726 Win=37789 Len=0
31	453.924838	2002:c3d4:6ffd:1101: 5258:1ff:fe9d:7eb8	2002:c3d4:6ffd:2202: 7232:4dff:fecb:be39	TCP	49152 > 1514 [FIN, ACK] Seq=12345726 Ack=12345706 Win=29996 Len=0
32	454.050839	2002:c3d4:6ffd:1101: 7232:4dff:fecb:be39	2002:c3d4:6ffd:1101: 5258:1ff:fe9d:7eb8	TCP	1514 > 49152 [ACK] Seq=12345706 Ack=12345727 Win=37789 Len=0
33	454.821201	2002:c3d4:6ffd:1101: 7232:4dff:fecb:be39	2002:c3d4:6ffd:1101: 5258:1ff:fe9d:7eb8	TCP	1514 > 49152 [FIN, ACK] Seq=12345706 Ack=12345727 Win=37789 Len=0

Table 3 - Using RO. Frames on eth1 of the Router.

When Route Optimization was used, before sending any packet, EB1 looked into its binding cache for an existing binding between the home address 2002:C3D4:6FFD:1101:7232:4DFF:FECB:BE39 and a care-of address. It found the binding to the care-of address 2002:C3D4:6FFD:2202:7232:4DFF:FECB:BE39 so the destination address of the packet was set to the care-of address and a Routing Header type 2 was added to convey the destination home address. Figure 19 shows the details of frame {21}.

```
Frame 21 (137 bytes on wire, 137 bytes captured)
    Arrival Time: Apr 19, 2004 01:08:05.028553000
    Time delta from previous packet: 0.003467000 seconds
    Time relative to first packet: 383.225012000 seconds
    Frame Number: 21
    Packet Length: 137 bytes
    Capture Length: 137 bytes
Ethernet II, Src: 00:08:54:03:ff:f1, Dst: 70:32:4d:cb:be:39
    Destination: 70:32:4d:cb:be:39 (70:32:4d:cb:be:39)
    Source: 00:08:54:03:ff:f1 (Netronix_03:ff:f1)
    Type: IPv6 (0x86dd)
Internet Protocol Version 6
    Version: 6
    Traffic class: 0x00
    Flowlabel: 0x00000
    Payload length: 83
    Next header: IPv6 routing (0x2b)
    Hop limit: 254
    Source address: 2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8
    Destination address: 2002:c3d4:6ffd:2202:7232:4dff:fecb:be39
Routing Header, Type 2
    Next header: TCP (0x06)
    Length: 2 (24 bytes)
    Type: 2
    Segments left: 1
Transmission Control Protocol, Src Port: 49152 (49152), Dst Port: 1514 (1514), Seq:
12345679, Ack: 12345679, Len: 27
    Source port: 49152 (49152)
    Destination port: 1514 (1514)
    Sequence number: 12345679
    Next sequence number: 12345706
    Acknowledgement number: 12345679
    Header length: 32 bytes
    Flags: 0x0010 (ACK)
        0... .... = Congestion Window Reduced (CWR): Not set
        .0.. .... = ECN-Echo: Not set
        ..0. .... = Urgent: Not set
        ...1 .... = Acknowledgment: Set
        .... 0... = Push: Not set
        .... .0.. = Reset: Not set
        .... ..0. = Syn: Not set
        .... ...0 = Fin: Not set
    Window size: 37800
    Checksum: 0x2680 (correct)
    Options: (12 bytes)
        NOP
        NOP
        Time stamp: tsval 1316, tsecr 1283
Data (27 bytes)

0000  00 00 00 1b 20 02 c3 d4 6f fd 11 01 52 58 01 ff   .... ...o...RX..
0010  fe 9d 7e b8 75 43 6c 69 6e 75 78                  ..~.uClinux
```

Figure 19 - Frame 21. RO is used in messages sent by EB1.

Thus, the second test demonstrated that Route Optimization successfully works for packets sent from a CN to a MN as long as the binding is previously set with the **iivconfig** application. Moreover, the fact of the mobile node's being attached to foreign network was transparent to the TCP layer and the chat application in both boards.

14 An Application proprietary solution for mobility

14.1 What if

What would have happened if Mobile IPv6 had not been present in the MCF5272 processor? One alternative had been to implement mobility support over the Transport Layer, as part of the application or as part of a Middleware Layer.

The functionalities explained below must be provided in order to apply this kind of solution.

14.2 Location service, Identifier and Locator

Having understood that DNS, alone, does not provide a solution to mobility, it is necessary to implement a location service that will provide a locator based on an identifier.

Since the IP address will change depending on the attached network, it is not a possible identifier. A sequence number, or a token, could be used to identify each device. It must be assigned by some authority organization. An IP address will be used as Locator.

As location services, [1] proposes the following solutions:

1. Broadcasting and Multicasting.
2. Forwarding Pointers.
3. Home-Based Approaches.
4. Hierarchical Approaches.

Solution 1 is not scalable for Internet: a broadcast or multicast message must be spread through the Internet if a device is being searched [1].

Solution 4 is more complicated than solutions 2 and 3.

A Home Based approach implementation will be described further.

It is necessary to have a server to which every device will access when looking for other device based on it identifier.

In the server side it is necessary to implement:

- A database with the assigned identifiers and state data, including the current location.
- Listening to and managing new connections to the service: for a device indicating that it is joining the service and it is available on a specific address.
- Listening to and managing search requests to return the corresponding address: for a device searching another device owning the provided identifier.

On the device/client side it is necessary to implement:

- A connection service and interface to be called by the application as soon as it pretends to be accessed.
- A search service to be invoked when a device pretends to communicate with another device whose identifier is known.
- Assuming connection oriented communication is implemented it is necessary to create the implementation of the main functionalities: connection identifier (like sockets), connect, send, receive, close, listen, bind, accept.
- The new interface must be defined.

14.3 Mobility support

A solution must be implemented in both server and devices sides for mobility.

On the server side, the following functionalities must be implemented:

- Listening to and managing locator updates. If the server knows the current

connections for every identifier it will send the updates to the corresponding devices; otherwise, the corresponding devices will need to implement a method to detect the other peer has moved.

On the device side, the following functionalities must be implemented:

- A way to detect that the device has moved to another network.
- Obtaining and update of the new address/locator, once movement has been detected. The address update must be sent to the server and to the correspondent nodes.
- Update of the peer address in the correspondent device. If a server update or a peer update is not received, it is necessary to implement another way to detect the peer is not reachable in the currently available address and to search the new address in the server.

14.4 Disadvantages

The main disadvantages of a solution implemented as part of the application are:

- Extra effort to implement functionalities that are not part of the core application. It can be seen that almost all Mobile IPv6 functionalities would have to be implemented as part of the application.
- Lack of Portability: any already existing application must be ported to the new interface.
- If a reliable connection like TCP is utilized, it will be necessary to close the old TCP connection and to create a new TCP connection for every handover. This adds some overhead.
- Not a standard solution. As Mobile IPv6 is.

If the solution were implemented in a Middleware layer, only the first point would be overcome.

15 Related Devices

15.1 A mobile chat device

The tested application along with MIPv6 implementation on a MCF5272 processor and the addition of a small LCD screen, a small keyboard, or speech recognition software and hardware, and wireless connectivity, bring up the possibility of creating a mobile chat device that can roam over different networks with different link/physical layer technologies and different carrier providers. This is a very simple service but other services could be provided by such device: messaging, voice over IP, video and so on.

15.2 The Vocera Communications Badge

This is a device developed by Vocera Communications [22]. It provides instant two-way communications in a campus environment using wireless LAN technology for mobile workers.

This device combines the advantages of 802.11b wireless LAN technology, speech recognition and VoIP to allow in-building mobile workers to instantly communicate with one another while roaming and away from wired telephones and other hardwired communications systems.

The system consists of Vocera Communications Server Software for a Windows 2000-based server system and the Vocera Communications Badge. This badge is a small wearable device that permits one user to communicate with another one, or to connect to other phones through PBX integration. It is hands-free since voice commands can be understood by the use of speech recognition.

It is mainly intended for mobile workers within a campus or a building environment, like doctors or nurses in a hospital.

The size of this device is: 4.2” tall x 1.4” wide.

Figure 20 - Vocera Communications Badge[2]

15.3 Advantages of using Mobile IPv6 in such device

By the use of MIPv6, a device like this could be used while a worker is moving from one level 3 network to another. This situation could be common in a factory building where a number of contiguous small range wireless LANs must be deployed because of the interference caused by the factory equipment. Thus an initiated communication could be maintained while the person moves from one network to other. Also it could be possible to do vertical handover between different link technologies. All this would allow for less code to be developed for handling mobility and would leverage mobility capabilities.

[2] The image was obtained from the Vocera web site [22]

16 Future possible applications

16.1 Possibilities

As well as the applications already described, other applications can be created based on the existence of Mobile IPv6.

16.2 Mobile MP3 player with home server

Some ColdFire processors have already been used to develop an MP3 player. The implementation of Mobile IPv6 on ColdFire could lead to the implementation of a mobile MP3 player, possibly running in the car. The MP3 player can download and play MP3 files from a server running at home. Instead of having all the files in a flash file system, the mobile MP3 player could be connected to the server all the time, through different technologies, like GPRS or 802.11 depending on the type of link available at any moment.

16.3 Car Router

ColdFire is usually used for SOHO routers. So, it is possible to use it as a car mobile router to bring Internet connectivity to a number of different devices inside the car. The fact of having a MIPv6 stack in this router adds the possibility of vertical mobility between different technologies like UMTS and 802.11, for example.

16.4 Anywhere Internet Device

There are places where cell phone services are not available, for example in underground buildings, some manufacturing plants, elevators or special places where some frequency bands are just banned. Sometimes, even though cell phones services are available other type of access networks, like 802.11G, and the intranet, possibly including a VPN, can provide the necessary communications service at a lower cost or

taking the advantages of low-power consumption technology like Bluetooth. A two-way radio system is an alternative that must be left behind if text messages, images, voice or video are to be exchanged.

A person roaming in a factory between the offices, the campus and the production plant with an Anywhere Internet Device can be always reachable as long as some type of access network is provided and the device has the appropriate interface. The difference with any other type of service and products is that the device is always connected through different technologies of access networks. Thus the already existing infrastructure can be used; thus, saving costs.

This device can be built with a ColdFire processor or other type of cheap processor having uClinux, a Mobile IPv6 stack and the appropriate interfaces.

The device could run any type of application from messaging to video.

In order to provide this service the appropriate Mobile IPv6 routers must be deployed in the places where connectivity is to be provided, if they are not already there. An X.500 directory service or a DNS could provide the mapping from human-friendly names or some specific attributes to device home addresses.

A person having this device could move around in a factory while holding a data session with its Anywhere Internet Device, an FTP download, for example. The person moves from one office, where the device is connected through Bluetooth, Figure 21-A, to the company's campus, where the device makes a handover to a GPRS connection, Figure 21-B. When the person arrives to the manufacturing plant the device makes a handover to an 802.11G connection, Figure 21-C. The data session is maintained during all the handovers.

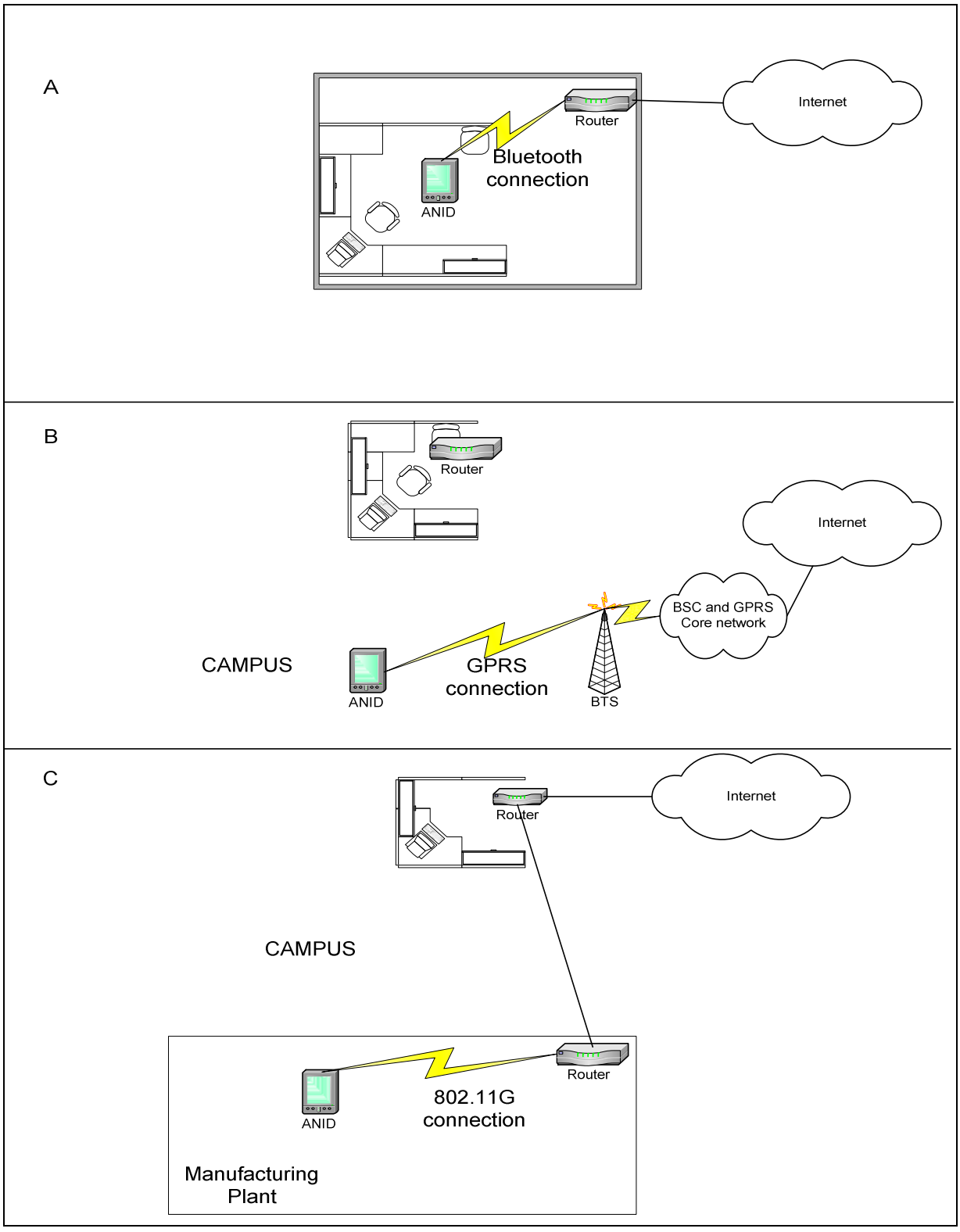

Figure 21 - Different link technologies during the same session

This device is similar to the Vocera Communications Badge described in 15.2 with the advantages brought by MIPv6 and, as a consequence, with a different infrastructure.

The advantages are:

- There is no need of middleware, the Software Engineer will be focused on the business case without worrying about reaching the aimed device, or group of devices.
- In a near future we could obtain personal mobility with mobile IPv6. Reaching not a specific device but a specific person in any mobile IPv6 device, including the one described here, with little software development effort.
- Since mobility is implemented at network layer, no matter what the underlying technology is.
- The advantages stated above allow the implementation of this service in different buildings or campus, one far away from the other as long as they are connected.
- A device can be benefited with the advantages of low-power consumption brought by Bluetooth.
- If there are different IP LANs a device will not be bothered by broadcasts on other LANs.
- IPv6 brings the security provided by IPSec.

16.5 Easy Push To Talk

Push To Talk is an application first developed for IDEN technology but now used for all cellular technologies. This is just a simple instance of the more general Anywhere Internet Device described above. The fact of having Mobile IPv6 on a ColdFire processor opens the possibility for having Push To Talk devices that can be located on any type of access networks, like 802.11, Bluetooth, Infrared and even a wired network like 802.3.

17 Conclusions

The port of LIVSIX to ColdFire worked correctly regarding MIPv6, ICMPv6 with the specific messages for MIPv6 and TCP for the messages used during the chat running and mobility. One-way Route optimization was tested: the binding between home address and the care-of address had to be manually set on the correspondent node; when this was done, a packet destined to the home address at the application layer was directly sent to the care-of address by the network layer and the routing header type 2 was added to the extension headers.

The type of handover tested here is called “horizontal handover”; further tests should be done for wireless links, like Wi-Fi 802.11 and Bluetooth, and for vertical handovers by changing the interface through which the board is connected.

The tests showed that an application programmed to communicate with IPv6 without any consideration about mobility, can be run in a device over MIPv6 so that when the mobile roams among different networks, this is transparent to TCP, or UDP, and the application.

Mobile devices are no longer limited to a specific access network. They can seamlessly move from one provider network to another provider network as long as Mobile IPv6 is supported. In the future, from a technical point of view, a user will be able to dynamically change to the best, or cheapest, carrier provider without affecting a data - including Voice over IP, video, etc. – communication.

A data communication session will no longer be limited to only one physical technology as long as the device has the appropriate interfaces and service providers. A handover between networks of different physical and data link layer technology is possible and this is also transparent for Transport Layer protocols. Thus, a user coming home will be able to change from a GPRS connection, through which he is maintaining a voice over IP session and a file transfer session, for example, to a home Bluetooth connection to a SOHO router connected to the Internet through a DSL connection, without affecting the current sessions and, as a consequence, lowering the communications costs.

From the paragraphs above it can be concluded that full device mobility can be achieved. The final advantages will be noted in the communications costs and in the full seamless mobility possibilities.

The transparency property was demonstrated during this work. Thus, distributed applications programmed to run in small mobile devices have the mobility problem already solved and this is a considerable save in software development costs according to what has been seen in “An Application proprietary solution for mobility”.

Furthermore, transparency brings up portability for already implemented applications. This means that no extra effort, nor costs, will be incurred to add mobility capabilities to an already existent device in the market.

The conclusion is that, by the use of Mobile IPv6, development and maintenance software costs, for mobile devices, will be lowered.

This work has been based on a ColdFire microprocessor with uClinux operating system implementation. Different devices and applications, for which a ColdFire microprocessor is used, have been described in “Processor platform”. The operating system uClinux is currently used in many other microprocessors. Thus, LIVSIX could also be ported to other different platforms and all the Mobile IPv6 advantages can be brought.

As well as the chat application developed for this test, many new distributed embedded applications, to provide new mobile services, can be developed to run on small mobile devices, the Vocera Communications Badge is just one example. Also, in “Future possible applications”, some “ingenious” networking applications, that take advantage of Mobile IPv6 characteristics to provide mobile and ubiquitous services, have been described during this work. The applications will not be limited to cell phones or PDA Internet navigation, e-mail or simple games. All the possible future mobile and ubiquitous applications will open a wide market for new services and, looking at many telecommunications companies, it can be seen that new markets are actually being opened.

The fact of Mobile IPv6 being defined as an open standard by the IETF is no less

important. Every provider developing MIPv6-capable devices will be sure that their devices will correctly interwork with any different provider's MIPv6 infrastructure devices. This opens the way to competitiveness and price reduction.

Finally, implementations on cheap and small microprocessors such as the one successfully realized during this work, the demonstrated capabilities brought by Mobile IPv6 and the new possible applications based on this type of implementation, will lead to the concept introduced at the beginning of this work: an almost unlimited number of highly mobile devices with full connectivity, wherever the device moves to, and providing many new kind of services. This is **Ubiquigeneous Networking**.

18 Appendix A

18.1 Displays and Ethereal Output

In this appendix the display output of both boards and the Ethereal IPv6 frames summaries are copied. Ethereal format files with all the frames (with or without IPv6 packets) are included in the companion CD.

18.2 EB1 Output

```
Motorola 5272 C3 boot

Uncompressing...done.
Linux version 2.4.21-uc0 (root@shire.middleearth) (gcc version 2.95.3
20010315 (release)(ColdFire patches - 20010318 from
http://fiddes.net/coldfire/)(-msep-data patches)) #112 Sun Apr 18
23:48:14 ART 2004
 uClinux/COLDFIRE(m5272)
COLDFIRE port done by Greg Ungerer, gerg@snapgear.com
Flat model support (C) 1998,1999 Kenneth Albanowski, D. Jeff Dionne
On node 0 totalpages: 4096
zone(0): 0 pages.
zone(1): 4096 pages.
zone(2): 0 pages.
Kernel command line:
Calibrating delay loop... 43.62 BogoMIPS
Memory available: 14376k/16384k RAM, 0k/0k ROM (648k kernel code, 208k
data)
kmem_create: Forcing size word alignment - vm_area_struct
kmem_create: Forcing size word alignment - mm_struct
kmem_create: Forcing size word alignment - filp
Dentry cache hash table entries: 2048 (order: 2, 16384 bytes)
Inode cache hash table entries: 1024 (order: 1, 8192 bytes)
kmem_create: Forcing size word alignment - inode_cache
Mount cache hash table entries: 512 (order: 0, 4096 bytes)
kmem_create: Forcing size word alignment - bdev_cache
kmem_create: Forcing size word alignment - cdev_cache
kmem_create: Forcing size word alignment - kiobuf
Buffer-cache hash table entries: 1024 (order: 0, 4096 bytes)
Page-cache hash table entries: 4096 (order: 2, 16384 bytes)
POSIX conformance testing by UNIFIX
Linux NET4.0 for Linux 2.4
Based upon Swansea University Computer Society NET3.039
kmem_create: Forcing size word alignment - sock
Initializing RT netlink socket
Starting kswapd
kmem_create: Forcing size word alignment - file_lock_cache
```

```
ColdFire internal UART serial driver version 1.00
ttyS0 at 0x10000100 (irq = 73) is a builtin ColdFire UART
ttyS1 at 0x10000140 (irq = 74) is a builtin ColdFire UART
kmem_create: Forcing size word alignment - blkdev_requests
fec.c: Probe number 0 with 0x0000
eth0: FEC ENET Version 0.2, 50:58:01:9d:7e:b8
fec: PHY @ 0x1, ID 0x0022561b -- AM79C874
SLIP: version 0.8.4-NET3.019-NEWTTY (dynamic channels, max=256).
CSLIP: code copyright 1989 Regents of the University of California.
Blkmem copyright 1998,1999 D. Jeff Dionne
Blkmem copyright 1998 Kenneth Albanowski
Blkmem 7 disk images:
0: F6290-1C628F [VIRTUAL F6290-1C628F] (RO)
1: FFE00000-FFE3FFFF [VIRTUAL FFE00000-FFE3FFFF] (RW)
2: FFE00000-FFE07FFF [VIRTUAL FFE00000-FFE07FFF] (RW)
3: FFE08000-FFE3FFFF [VIRTUAL FFE08000-FFE3FFFF] (RW)
4: FFE40000-FFFFFFFF [VIRTUAL FFE40000-FFFFFFFF] (RW)
5: FFF00000-FFFFFFFF [VIRTUAL FFF00000-FFFFFFFF] (RW)
6: FFE00000-FFFFFFFF [VIRTUAL FFE00000-FFFFFFFF] (RW)
RAMDISK driver initialized: 16 RAM disks of 4096K size 1024 blocksize
PPP generic driver version 2.4.2
NET4: Linux TCP/IP 1.0 for NET4.0
IP Protocols: ICMP, UDP, TCP
kmem_create: Forcing size word alignment - ip_dst_cache
IP: routing cache hash table of 512 buckets, 4Kbytes
TCP: Hash tables configured (established 1024 bind 1024)
NET4: Unix domain sockets 1.0/SMP for Linux NET4.0.
VFS: Mounted root (romfs filesystem) readonly.
Freeing unused kernel memory: 24k freed (0xde000 - 0xe3000)
Shell invoked to run file: /etc/rc
Command: hostname uClinux
Command: /bin/expand /etc/ramfs.img /dev/ram0
Command: mount -t proc proc /proc
Command: mount -t ext2 /dev/ram0 /var
Command: mkdir /var/tmp
Command: mkdir /var/log
Command: mkdir /var/run
Command: mkdir /var/lock
Command: ifconfig lo 127.0.0.1
Command: cat /etc/motd
Welcome to
          ____ _  _
         /  __| ||_|
      _  _| |  | | _ ____  _   _  _  _
   | | | | |  | || |  _ \| | | |\ \/ /
   | |_| | |__| || | | | | |_| |/    \
   |  ___\____|_||_|_| |_|\____|\_/\_/
   | |
   |_|

For further information check:
http://www.uclinux.org/

Execution Finished, Exiting

Sash command shell (version 1.1.1)
```

```
/> ifconfig 192.168.0.11_ __ __ __ __ __ __ __ __ __ __ __ _eth0
192.168.0.12_ _1
eth0: config: auto-negotiation on, 100FDX, 100HDX, 10FDX, 10HDX.
/> eth0: status: link up, 100MBit Full Duplex, auto-negotiation
complete.

/> insmod livsix.o
insmod: /lib/modules/2.4.21-uc0: No such file or directory
insmod: livsix.o: no module by that name found
pid 12: failed 256
/> cd bin
/bin> insmod livsix.o
Using livsix.o
LIVSIXv0.3 Loaded
/bin> setdefint eth0
Interface eth0 is the default
File /proc/sys/net/livsix/conf/eth0/defint
/bin> livconfig -s o ::/0 ::/0 0 0 0 0 0 1
/bin> livconfig -s i ::/0 ::/0 0 0 0 0 0 1
/bin> livconfig -h 2002:C3D4:6FFD:1101:2C0:26FF:FEB5:A24
Setting Home Agent to 2002:C3D4:6FFD:1101:2C0:26FF:FEB5:A24...
/bin> livconfig --ro enable
RO set
/bin> livconfig
eth0:
FE80::5258:1FF:FE9D:7EB8

lo:
::0.0.0.1

Inbound Security Policy Database:
1. SRC: * DST: *
Policy: BYPASS

Outbound Security Policy Database:
1. SRC: * DST: *
Policy: BYPASS

/bin> livconfig
eth0:
2002:C3D4:6FFD:1101:5258:1FF:FE9D:7EB8 ->(Home Address)
FE80::5258:1FF:FE9D:7EB8

lo:
::0.0.0.1

Inbound Security Policy Database:
1. SRC: * DST: *
Policy: BYPASS

Outbound Security Policy Database:
1. SRC: * DST: *
Policy: BYPASS

/bin> chat uc1 2002:C3D4:6FFD:1101:5258:1FF:FE9D:7EB8
HOST: uClinux
```

```
IP: 2002:C3D4:6FFD:1101:5258:1FF:FE9D:7EB8

>>0:invite 2002:C3D4:6FFD:1101:7232:4DFF:FECB:BE39
>>ERROR: nbuf_dequeue: nbuf queue list is null
ERROR: nbuf_dequeue: nbuf queue list is null
ERROR: nbuf_dequeue: nbuf queue list is null

*** 1:ACCEPT RECEIVED FROM IP 2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
NODE: ***
>>1:He_ _ELLO!
>>
*** 1:HI I'm EB2***
>>Wh_ __ _1:Where are you_ __ __ __ __ __ __ __ __ __ __ __ __ __ __
_Where are you?

*** PARSE ERROR: 109 ***
>>1:Where are you?
>>
*** 1:I'm in network1***
>>
*** 1:I'll move to network2***
>>ERROR: nbuf_dequeue: nbuf queue list is null

*** 1:I've moved to network2. I'm in network2!!!***
>>1:Thats_ _'s great!
>>
*** 1:ok bye***
>>
*** 1:FINISH RECEIVED***
>>0:finish 0
>>
End
/bin> livconfig -a 2002:C3D4:6FFD:1101:7232:4DFF:FECB:BE39
2002:C3D4:6FFD:2202:7232:4DFF:FECB:BE39 30000
Entry added to the Binding Cache
/bin> livconfig -b
livconfig: Binding Cache:
HOME ADDRESS                              CARE-OF ADDRESS
lt
2002:C3D4:6FFD:1101:7232:4DFF:FECB:BE39
2002:C3D4:6FFD:2202:7232:4DFF:FECB:BE39 29994
/bin> chat _ __ __ __ __ _livconfig
eth0:
2002:C3D4:6FFD:1101:5258:1FF:FE9D:7EB8 ->(Home Address)
FE80::5258:1FF:FE9D:7EB8

lo:
::0.0.0.1

Inbound Security Policy Database:
1. SRC: * DST: *
Policy: BYPASS

Outbound Security Policy Database:
1. SRC: * DST: *
Policy: BYPASS
```

```
/bin> chat uc1 2002:C3D4:6FFD:1101:5258:1FF:FE9D:7EB8
HOST: uClinux
IP: 2002:C3D4:6FFD:1101:5258:1FF:FE9D:7EB8

>>0:invite 2002:C3D4:6FFD:1101:7232:4DFF:FECB:BE39
>>
*** 1:ACCEPT RECEIVED FROM IP 2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
NODE: ***
>>1:Hello
>>
*** 1:Hi, it's you again!***
>>yep, bye

*** PARSE ERROR: 109 ***
>>1:yep bye
>>0:fui_ __ _inish 1
>>0:finish 0
>>
End
/bin> livconfig -b
livconfig: Binding Cache:
HOME ADDRESS                              CARE-OF ADDRESS
lt
2002:C3D4:6FFD:1101:7232:4DFF:FECB:BE39
2002:C3D4:6FFD:2202:7232:4DFF:FECB:BE39 29726
/bin>
```

18.3 EB2 Output

```
Motorola 5272 C3 boot

Uncompressing...done.
Linux version 2.4.21-uc0 (root@shire.middleearth) (gcc version 2.95.3
20010315 (release)(ColdFire patches - 20010318 from
http://fiddes.net/coldfire/)(-msep-data patches)) #112 Sun Apr 18
23:48:14 ART 2004


uClinux/COLDFIRE(m5272)
COLDFIRE port done by Greg Ungerer, gerg@snapgear.com
Flat model support (C) 1998,1999 Kenneth Albanowski, D. Jeff Dionne
On node 0 totalpages: 4096
zone(0): 0 pages.
zone(1): 4096 pages.
zone(2): 0 pages.
Kernel command line:
Calibrating delay loop... 43.62 BogoMIPS
Memory available: 14376k/16384k RAM, 0k/0k ROM (648k kernel code, 208k
data)
kmem_create: Forcing size word alignment - vm_area_struct
kmem_create: Forcing size word alignment - mm_struct
kmem_create: Forcing size word alignment - filp
Dentry cache hash table entries: 2048 (order: 2, 16384 bytes)
```

```
Inode cache hash table entries: 1024 (order: 1, 8192 bytes)
kmem_create: Forcing size word alignment - inode_cache
Mount cache hash table entries: 512 (order: 0, 4096 bytes)
kmem_create: Forcing size word alignment - bdev_cache
kmem_create: Forcing size word alignment - cdev_cache
kmem_create: Forcing size word alignment - kiobuf
Buffer-cache hash table entries: 1024 (order: 0, 4096 bytes)
Page-cache hash table entries: 4096 (order: 2, 16384 bytes)
POSIX conformance testing by UNIFIX
Linux NET4.0 for Linux 2.4
Based upon Swansea University Computer Society NET3.039
kmem_create: Forcing size word alignment - sock
Initializing RT netlink socket
Starting kswapd
kmem_create: Forcing size word alignment - file_lock_cache
ColdFire internal UART serial driver version 1.00
ttyS0 at 0x10000100 (irq = 73) is a builtin ColdFire UART
ttyS1 at 0x10000140 (irq = 74) is a builtin ColdFire UART
kmem_create: Forcing size word alignment - blkdev_requests
fec.c: Probe number 0 with 0x0000
eth0: FEC ENET Version 0.2, 70:32:4d:cb:be:39
fec: PHY @ 0x1, ID 0x0022561b -- AM79C874
SLIP: version 0.8.4-NET3.019-NEWTTY (dynamic channels, max=256).
CSLIP: code copyright 1989 Regents of the University of California.
Blkmem copyright 1998,1999 D. Jeff Dionne
Blkmem copyright 1998 Kenneth Albanowski
Blkmem 7 disk images:
0: F6290-1C628F [VIRTUAL F6290-1C628F] (RO)
1: FFE00000-FFE3FFFF [VIRTUAL FFE00000-FFE3FFFF] (RW)
2: FFE00000-FFE07FFF [VIRTUAL FFE00000-FFE07FFF] (RW)
3: FFE08000-FFE3FFFF [VIRTUAL FFE08000-FFE3FFFF] (RW)
4: FFE40000-FFFFFFFF [VIRTUAL FFE40000-FFFFFFFF] (RW)
5: FFF00000-FFFFFFFF [VIRTUAL FFF00000-FFFFFFFF] (RW)
6: FFE00000-FFFFFFFF [VIRTUAL FFE00000-FFFFFFFF] (RW)
RAMDISK driver initialized: 16 RAM disks of 4096K size 1024 blocksize
PPP generic driver version 2.4.2
NET4: Linux TCP/IP 1.0 for NET4.0
IP Protocols: ICMP, UDP, TCP
kmem_create: Forcing size word alignment - ip_dst_cache
IP: routing cache hash table of 512 buckets, 4Kbytes
TCP: Hash tables configured (established 1024 bind 1024)
NET4: Unix domain sockets 1.0/SMP for Linux NET4.0.
VFS: Mounted root (romfs filesystem) readonly.
Freeing unused kernel memory: 24k freed (0xde000 - 0xe3000)
Shell invoked to run file: /etc/rc
Command: hostname uClinux
Command: /bin/expand /etc/ramfs.img /dev/ram0
Command: mount -t proc proc /proc
Command: mount -t ext2 /dev/ram0 /var
Command: mkdir /var/tmp
Command: mkdir /var/log
Command: mkdir /var/run
Command: mkdir /var/lock
Command: ifconfig lo 127.0.0.1
Command: cat /etc/motd
Welcome to
```

```
          ____ _  _
         /  __| ||_|
      _  | |  | | _ ____  _   _  _  _
     | | | | |  | || |  _ \| | | |\ \/ /
     | |_| | |__| || | | | | |_| |/    \
     |  ___\____|_||_|_| |_|\____|\_/\_/
     | |
     |_|

For further information check:
http://www.uclinux.org/

Execution Finished, Exiting

Sash command shell (version 1.1.1)
/> ifconfig eth0 192.168.0.12
eth0: config: auto-negotiation on, 100FDX, 100HDX, 10FDX, 10HDX.
/> eth0: status: link up, 100MBit Full Duplex, auto-negotiation
complete.

/> cd bin
/bin> insmod livsix.o
Using livsix.o
LIVSIXv0.3 Loaded
/bin> setdefint eth0
Interface eth0 is the default
File /proc/sys/net/livsix/conf/eth0/defint
/bin> livconfig -s o ::/0 ::/0 0 0 0 0 0 1
/bin> livconfig -s i ::/0 ::/0 0 0 0 0 0 1
/bin> livconfig -h 2002:C3D4:6FFD:1101:2C0:26FF:FEB5:A24
Setting Home Agent to 2002:C3D4:6FFD:1101:2C0:26FF:FEB5:A24...
/bin> livconfig --ro enable
RO set
/bin> livconfig
eth0:
FE80::7232:4DFF:FECB:BE39

lo:
::0.0.0.1

Inbound Security Policy Database:
1. SRC: * DST: *
Policy: BYPASS

Outbound Security Policy Database:
1. SRC: * DST: *
Policy: BYPASS

/bin> livconfig
eth0:
2002:C3D4:6FFD:1101:7232:4DFF:FECB:BE39 ->(Home Address)
FE80::7232:4DFF:FECB:BE39

lo:
::0.0.0.1
```

```
Inbound Security Policy Database:
1. SRC: * DST: *
Policy: BYPASS

Outbound Security Policy Database:
1. SRC: * DST: *
Policy: BYPASS

/bin> chat uc2 2002:C3D4:6FFD:1101:7232:4DFF:FECB:BE39
HOST: uClinux
IP: 2002:C3D4:6FFD:1101:7232:4DFF:FECB:BE39

>>ERROR: nbuf_dequeue: nbuf queue list is null
ERROR: tcp_listen_sock_lookup: connection request
ERROR: nbuf_dequeue: nbuf queue list is null
ERROR: nbuf_dequeue: nbuf queue list is null
ERROR: nbuf_dequeue: nbuf queue list is null

*** 1:INVITE RECEIVED FROM IP 2002:c3d4:6ffd:1101:5258:01ff:fe9d:7eb8
NODE: uClinux***
>>0:acceptr_ _ 1
>>
*** 1:HELLO!***
>>1:HI I'm EB2
>>
*** 1:Where are you?***
>>1:Im_ _'m in networl_ _k1
>>1:I'll move to network2
>>1:I've moved to network2. I'm in network2!!!
>>
*** 1:That's great!***
>>ok by2_ _e_ __ __ __ __ __ _1:ok b_ __ __ __ __ __ _ok bye

*** PARSE ERROR: 109 ***
>>1:ok bye
>>0:finish 1
>>0:finish 0
>>
End
/bin> livconfig --bulist
livconfig: Binding Update List:
Hoa\Coa\HA\lifetime

2002:C3D4:6FFD:1101:7232:4DFF:FECB:BE39
2002:C3D4:6FFD:2202:7232:4DFF:FECB:BE39
2002:C3D4:6FFD:1101:2C0:26FF:FEB5:A24 4294967295
/bin> livconfig --bulist
livconfig: Binding Update List:
Hoa\Coa\HA\lifetime

2002:C3D4:6FFD:1101:7232:4DFF:FECB:BE39
2002:C3D4:6FFD:2202:7232:4DFF:FECB:BE39
2002:C3D4:6FFD:1101:2C0:26FF:FEB5:A24 4294967295
/bin> chat_ __ __ __ _livconfig
eth0:
2002:C3D4:6FFD:2202:7232:4DFF:FECB:BE39
```

```
2002:C3D4:6FFD:1101:7232:4DFF:FECB:BE39 ->(Home Address)
FE80::7232:4DFF:FECB:BE39

lo:
::0.0.0.1

Inbound Security Policy Database:
1. SRC: * DST: *
Policy: BYPASS

Outbound Security Policy Database:
1. SRC: * DST: *
Policy: BYPASS

/bin> chat uc2 2002:C3D4:6FFD:1101:7232:4DFF:FECB:BE39
HOST: uClinux
IP: 2002:C3D4:6FFD:1101:7232:4DFF:FECB:BE39

>>ERROR: ip6_rh_process: new dst =
2002:C3D4:6FFD:1101:7232:4DFF:FECB:BE39
ERROR: ip6_unpack: Routing Header type not supported
ERROR: tcp_listen_sock_lookup: connection request
ERROR: ip6_rh_process: new dst =
2002:C3D4:6FFD:1101:7232:4DFF:FECB:BE39
ERROR: ip6_unpack: Routing Header type not supported

*** 1:INVITE RECEIVED FROM IP 2002:c3d4:6ffd:1101:5258:01ff:fe9d:7eb8
NODE: uClinux***
>>
>>1:acept _ __ __ __ __ __ __ __ _0:accept 1
>>ERROR: ip6_rh_process: new dst =
2002:C3D4:6FFD:1101:7232:4DFF:FECB:BE39
ERROR: ip6_unpack: Routing Header type not supported

>>ERROR: ip6_rh_process: new dst =
2002:C3D4:6FFD:1101:7232:4DFF:FECB:BE39
ERROR: ip6_unpack: Routing Header type not supported

*** 1:Hello***
>>1:Hi, it's you again!
>>ERROR: ip6_rh_process: new dst =
2002:C3D4:6FFD:1101:7232:4DFF:FECB:BE39
ERROR: ip6_unpack: Routing Header type not supported

>>ERROR: ip6_rh_process: new dst =
2002:C3D4:6FFD:1101:7232:4DFF:FECB:BE39
ERROR: ip6_unpack: Routing Header type not supported

*** 1:yep bye***
>>ERROR: ip6_rh_process: new dst =
2002:C3D4:6FFD:1101:7232:4DFF:FECB:BE39
ERROR: ip6_unpack: Routing Header type not supported

*** 1:FINISH RECEIVED***
>>0:finish 0
>>
```

```
End
/bin>
```

18.4 Ethereal IPv6 packets summary: on HA's eth0 before handover

```
No.  Time           Source                Destination           Protocol  Info
DestPort
 1  0.000000      fe80::7232:4dff:fecb:be39  ff02::2                ICMPv6    Router
solicitation
 2  14.020907     fe80::5258:1ff:fe9d:7eb8  ff02::2                ICMPv6    Router
solicitation
 3 126.668528  fe80::2c0:26ff:feb5:a24 ff02::2          ICMPv6   Router solicitation
 4  162.432315   fe80::2e0:7dff:fee1:fbc1  ff02::2                ICMPv6    Router
solicitation
 5  193.193778   fe80::2e0:7dff:fee1:fbc1  ff02::1                ICMPv6    Router
advertisement
 6             284.222978                    2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8
2002:c3d4:6ffd:1101:7232:4dff:fecb:be39 ICMPv6   Neighbor solicitation
 7             284.223252                    2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24
2002:c3d4:6ffd:1101:7232:4dff:fecb:be39 ICMPv6   Neighbor solicitation
 8             284.224157                    2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1
2002:c3d4:6ffd:1101:7232:4dff:fecb:be39 ICMPv6   Neighbor solicitation
 9             284.224425                    2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24
2002:c3d4:6ffd:1101:7232:4dff:fecb:be39 ICMPv6   Neighbor solicitation
10            284.224306                   2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 ICMPv6   Neighbor solicitation
11             284.224713                    2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 ICMPv6   Neighbor solicitation
12            284.225327                    2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1
2002:c3d4:6ffd:1101:7232:4dff:fecb:be39 ICMPv6   Neighbor solicitation
13             284.225557                    2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24
2002:c3d4:6ffd:1101:7232:4dff:fecb:be39 ICMPv6   Neighbor solicitation
14            284.225824                   2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24 ICMPv6   Neighbor solicitation
15             284.226078                    2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24
2002:c3d4:6ffd:1101:7232:4dff:fecb:be39 ICMPv6   Neighbor advertisement
16            284.226606                    2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 ICMPv6   Neighbor solicitation
17             284.226815                    2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 ICMPv6   Neighbor solicitation
18            284.227094                    2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8
2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24 ICMPv6   Neighbor solicitation
19             284.227346                    2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 ICMPv6   Neighbor advertisement
20            284.227491                   2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1 ICMPv6   Neighbor solicitation
21             284.227734                    2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24
2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1 ICMPv6   Neighbor solicitation
22            284.227819                    2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1
2002:c3d4:6ffd:1101:7232:4dff:fecb:be39 ICMPv6   Neighbor solicitation
23            284.227983                    2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1
2002:c3d4:6ffd:1101:7232:4dff:fecb:be39 ICMPv6   Neighbor solicitation
24            284.228603                   2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24 ICMPv6   Neighbor advertisement
25            284.228816                    2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8
2002:c3d4:6ffd:1101:7232:4dff:fecb:be39 ICMPv6   Neighbor solicitation
26            284.229053                    2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1
2002:c3d4:6ffd:1101:7232:4dff:fecb:be39 ICMPv6   Neighbor solicitation
27            284.229243                    2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1
2002:c3d4:6ffd:1101:7232:4dff:fecb:be39 ICMPv6   Neighbor solicitation
28            284.228899                    2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 ICMPv6   Neighbor solicitation
```

```
29              284.229479                              2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 ICMPv6   Neighbor solicitation
30              284.229317                              2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8
2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1 ICMPv6   Neighbor solicitation
31               284.229729                               2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24
2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1 ICMPv6   Neighbor solicitation
32              284.230085                              2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1
2002:c3d4:6ffd:1101:7232:4dff:fecb:be39 ICMPv6   Neighbor solicitation
33              284.230197                              2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1
2002:c3d4:6ffd:1101:7232:4dff:fecb:be39 ICMPv6   Neighbor solicitation
34              284.230499                              2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8
2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24 ICMPv6   Neighbor advertisement
35             284.230705                             2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 ICMPv6   Neighbor solicitation
36              284.230876                              2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 ICMPv6   Neighbor solicitation
37              284.231399                              2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1
2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24 ICMPv6   Neighbor solicitation
38               284.231654                               2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24
2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1 ICMPv6   Neighbor advertisement
39              284.231745                              2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8
2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24 ICMPv6   Neighbor advertisement
40             284.232057                             2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24 ICMPv6   Neighbor advertisement
41              284.232463                              2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1
2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24 ICMPv6   Neighbor solicitation
42               284.232715                               2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24
2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1 ICMPv6   Neighbor advertisement
43             284.232857                             2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24 ICMPv6   Neighbor advertisement
44              284.233620                              2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 ICMPv6   Neighbor solicitation
45              284.233741                              2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 ICMPv6   Neighbor solicitation
46              284.235974                              2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1
2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24 ICMPv6   Neighbor advertisement
47             284.236194                             2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1 ICMPv6   Neighbor solicitation
48              284.236366                              2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8
2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1 ICMPv6   Neighbor solicitation
49              284.238243                              2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1
2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24 ICMPv6   Neighbor advertisement
50             284.242854                             2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24 ICMPv6   Neighbor solicitation
51               284.243130                               2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24
2002:c3d4:6ffd:1101:7232:4dff:fecb:be39 ICMPv6   Neighbor advertisement
52              284.243393                              2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8
2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24 ICMPv6   Neighbor solicitation
53               284.243666                               2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 ICMPv6   Neighbor advertisement
54              284.270636                              2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8
2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24 ICMPv6   Neighbor advertisement
55              284.271503                              2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8
2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24 ICMPv6   Neighbor advertisement
56             284.296099                             2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24 ICMPv6   Neighbor advertisement
57             284.296906                             2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24 ICMPv6   Neighbor advertisement
58             284.297742                             2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24 ICMPv6   Neighbor advertisement
```

18.5 Ethereal IPv6 packets summary: on HA's eth0 after handover

```
No. Time        Source                Destination           Protocol Info
DestPort
 1 0.000000    2002:c3d4:6ffd:2202:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24 IPv6     cftp (0x3e)
 2 0.000301    2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24 ff02::1               ICMPv6
Neighbor advertisement
 3 0.000506    2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24 ff02::2               ICMPv6
Neighbor advertisement
 4 0.000796    2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24
2002:c3d4:6ffd:2202:7232:4dff:fecb:be39 IPv6     cftp (0x3e)
 5 23.449279   2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 TCP      1514 > 49152 [ACK] Seq=12345741
Ack=12345734 Win=37782 Len=46   49152
 6 23.449437   2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 TCP      1514 > 49152 [ACK] Seq=12345741
Ack=12345734 Win=37782 Len=46   49152
 7 23.585234   2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8
2002:c3d4:6ffd:1101:7232:4dff:fecb:be39 TCP      49152 > 1514 [ACK] Seq=12345734
Ack=12345787 Win=29954 Len=0    1514
 8 23.585482   2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8
2002:c3d4:6ffd:1101:7232:4dff:fecb:be39 TCP      49152 > 1514 [ACK] Seq=12345734
Ack=12345787 Win=29954 Len=0    1514
 9 33.107800   2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8
2002:c3d4:6ffd:1101:7232:4dff:fecb:be39 TCP      49152 > 1514 [ACK] Seq=12345734
Ack=12345787 Win=29954 Len=17   1514
10 33.108067   2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8
2002:c3d4:6ffd:1101:7232:4dff:fecb:be39 TCP      49152 > 1514 [ACK] Seq=12345734
Ack=12345787 Win=29954 Len=17   1514
11 33.169831   2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 TCP      1514 > 49152 [ACK] Seq=12345787
Ack=12345751 Win=37782 Len=0    49152
12 33.169968   2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 TCP      1514 > 49152 [ACK] Seq=12345787
Ack=12345751 Win=37782 Len=0    49152
13 68.906541   2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 TCP      1514 > 49152 [ACK] Seq=12345787
Ack=12345751 Win=37782 Len=10   49152
14 68.906701   2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 TCP      1514 > 49152 [ACK] Seq=12345787
Ack=12345751 Win=37782 Len=10   49152
15 68.992297   2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8
2002:c3d4:6ffd:1101:7232:4dff:fecb:be39 TCP      49152 > 1514 [ACK] Seq=12345751
Ack=12345797 Win=29954 Len=0    1514
16 68.992546   2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8
2002:c3d4:6ffd:1101:7232:4dff:fecb:be39 TCP      49152 > 1514 [ACK] Seq=12345751
Ack=12345797 Win=29954 Len=0    1514
17 81.028511   2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 TCP      1514 > 49152 [FIN, ACK] Seq=12345797
Ack=12345751 Win=37782 Len=0 49152
18 81.028667   2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 TCP      1514 > 49152 [FIN, ACK] Seq=12345797
Ack=12345751 Win=37782 Len=0 49152
19 81.194236   2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8
2002:c3d4:6ffd:1101:7232:4dff:fecb:be39 TCP      49152 > 1514 [ACK] Seq=12345751
Ack=12345798 Win=29954 Len=0    1514
20 81.194485   2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8
2002:c3d4:6ffd:1101:7232:4dff:fecb:be39 TCP      49152 > 1514 [ACK] Seq=12345751
Ack=12345798 Win=29954 Len=0    1514
21 81.594676   2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8
2002:c3d4:6ffd:1101:7232:4dff:fecb:be39 TCP      49152 > 1514 [FIN, ACK] Seq=12345751
Ack=12345798 Win=29954 Len=0 1514
22 81.594928   2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8
2002:c3d4:6ffd:1101:7232:4dff:fecb:be39 TCP      49152 > 1514 [FIN, ACK] Seq=12345751
```

```
Ack=12345798 Win=29954 Len=0 1514
23 160.396859  fe80::2e0:7dff:fee1:fbc1 ff02::1               ICMPv6   Router
advertisement
24 176.629426  fe80::2e0:7dff:fee1:fbc1 ff02::1               ICMPv6   Router
advertisement
25 191.301747  fe80::2e0:7dff:fee1:fbc1 ff02::1               ICMPv6   Router
advertisement
26 206.144085  fe80::2e0:7dff:fee1:fbc1 ff02::1               ICMPv6   Router
advertisement
27 218.045975  fe80::2e0:7dff:fee1:fbc1 ff02::1               ICMPv6   Router
advertisement
28 226.197265  fe80::2e0:7dff:fee1:fbc1 ff02::1               ICMPv6   Router
advertisement
29 235.078666  fe80::2e0:7dff:fee1:fbc1 ff02::1               ICMPv6   Router
advertisement
30 245.120256  fe80::2e0:7dff:fee1:fbc1 ff02::1               ICMPv6   Router
advertisement
31 256.812108  fe80::2e0:7dff:fee1:fbc1 ff02::1               ICMPv6   Router
advertisement
32 269.174059  fe80::2e0:7dff:fee1:fbc1 ff02::1               ICMPv6   Router
advertisement
33 277.765421  fe80::2e0:7dff:fee1:fbc1 ff02::1               ICMPv6   Router
advertisement
34 287.716994  fe80::2e0:7dff:fee1:fbc1 ff02::1               ICMPv6   Router
advertisement
35 296.148331  fe80::2e0:7dff:fee1:fbc1 ff02::1               ICMPv6   Router
advertisement
36 305.879868  fe80::2e0:7dff:fee1:fbc1 ff02::1               ICMPv6   Router
advertisement
37 314.751275  fe80::2e0:7dff:fee1:fbc1 ff02::1               ICMPv6   Router
advertisement
38 329.523619  fe80::2e0:7dff:fee1:fbc1 ff02::1               ICMPv6   Router
advertisement
39 340.935420  fe80::2e0:7dff:fee1:fbc1 ff02::1               ICMPv6   Router
advertisement
40 356.607896  fe80::2e0:7dff:fee1:fbc1 ff02::1               ICMPv6   Router
advertisement
41 372.510412  fe80::2e0:7dff:fee1:fbc1 ff02::1               ICMPv6   Router
advertisement
42 383.280153  2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 TCP      1514 > 49152 [SYN, ACK] Seq=12345678
Ack=12345679 Win=37800 Len=0 49152
43 383.280310  2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 TCP      1514 > 49152 [SYN, ACK] Seq=12345678
Ack=12345679 Win=37800 Len=0 49152
44 384.278859  2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 TCP      1514 > 49152 [ACK] Seq=12345679
Ack=12345706 Win=37800 Len=0    49152
45 384.279011  2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 TCP      1514 > 49152 [ACK] Seq=12345679
Ack=12345706 Win=37800 Len=0    49152
46 384.892368  fe80::2e0:7dff:fee1:fbc1 ff02::1               ICMPv6   Router
advertisement
47 393.723770  fe80::2e0:7dff:fee1:fbc1 ff02::1               ICMPv6   Router
advertisement
48 399.332394  2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 TCP      1514 > 49152 [ACK] Seq=12345679
Ack=12345706 Win=37800 Len=4    49152
49 399.332550  2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 TCP      1514 > 49152 [ACK] Seq=12345679
Ack=12345706 Win=37800 Len=4    49152
50 404.505477  fe80::2e0:7dff:fee1:fbc1 ff02::1               ICMPv6   Router
advertisement
51 415.927285  fe80::2e0:7dff:fee1:fbc1 ff02::1               ICMPv6   Router
advertisement
52 419.718482  2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 TCP      1514 > 49152 [ACK] Seq=12345683
Ack=12345715 Win=37791 Len=0    49152
53 419.718639  2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 TCP      1514 > 49152 [ACK] Seq=12345683
Ack=12345715 Win=37791 Len=0    49152
```

```
54 425.368779  fe80::2e0:7dff:fee1:fbc1 ff02::1               ICMPv6   Router
advertisement
55 432.668156  2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 TCP      1514 > 49152 [ACK] Seq=12345683
Ack=12345715 Win=37791 Len=23   49152
56 432.668311  2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 TCP      1514 > 49152 [ACK] Seq=12345683
Ack=12345715 Win=37791 Len=23   49152
57 440.631193  fe80::2e0:7dff:fee1:fbc1 ff02::1               ICMPv6   Router
advertisement
58 447.119454  2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 TCP      1514 > 49152 [ACK] Seq=12345706
Ack=12345726 Win=37789 Len=0    49152
59 447.119610  2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 TCP      1514 > 49152 [ACK] Seq=12345706
Ack=12345726 Win=37789 Len=0    49152
60 450.772802  fe80::2e0:7dff:fee1:fbc1 ff02::1               ICMPv6   Router
advertisement
61 454.120952  2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 TCP      1514 > 49152 [ACK] Seq=12345706
Ack=12345727 Win=37789 Len=0    49152
62 454.121108  2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 TCP      1514 > 49152 [ACK] Seq=12345706
Ack=12345727 Win=37789 Len=0    49152
63 454.891117  2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 TCP      1514 > 49152 [FIN, ACK] Seq=12345706
Ack=12345727 Win=37789 Len=0 49152
64 454.891263  2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 TCP      1514 > 49152 [FIN, ACK] Seq=12345706
Ack=12345727 Win=37789 Len=0 49152
65 459.374155  fe80::2e0:7dff:fee1:fbc1 ff02::1               ICMPv6   Router
advertisement
66 473.466389  fe80::2e0:7dff:fee1:fbc1 ff02::1               ICMPv6   Router
advertisement
67 488.428766  fe80::2e0:7dff:fee1:fbc1 ff02::1               ICMPv6   Router
advertisement
68 500.990769  fe80::2e0:7dff:fee1:fbc1 ff02::1               ICMPv6   Router
advertisement
69 518.983587  fe80::2e0:7dff:fee1:fbc1 ff02::1               ICMPv6   Router
advertisement
70 535.256163  fe80::2e0:7dff:fee1:fbc1 ff02::1               ICMPv6   Router
advertisement
71 551.268693  fe80::2e0:7dff:fee1:fbc1 ff02::1               ICMPv6   Router
advertisement
```

18.6 Ethereal IPv6 packets summary: on Router's eth0 before handover

```
No. Time        Source                Destination           Protocol Info
173 43.639309   fe80::7232:4dff:fecb:be39 ff02::2               ICMPv6   Router
solicitation
202 57.657999   fe80::5258:1ff:fe9d:7eb8 ff02::2               ICMPv6   Router
solicitation
439 170.288045  fe80::2c0:26ff:feb5:a24 ff02::2               ICMPv6   Router
solicitation
552 206.045841  fe80::2e0:7dff:fee1:fbc1 ff02::2               ICMPv6   Router
solicitation
775 236.802396  fe80::2e0:7dff:fee1:fbc1 ff02::1               ICMPv6   Router
advertisement
964 327.817308  2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8
2002:c3d4:6ffd:1101:7232:4dff:fecb:be39 ICMPv6   Neighbor solicitation
965 327.818378  2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1
```

```
2002:c3d4:6ffd:1101:7232:4dff:fecb:be39 ICMPv6   Neighbor solicitation
966 327.817850  2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24
2002:c3d4:6ffd:1101:7232:4dff:fecb:be39 ICMPv6   Neighbor solicitation
967 327.819546  2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1
2002:c3d4:6ffd:1101:7232:4dff:fecb:be39 ICMPv6   Neighbor solicitation
968 327.818634  2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 ICMPv6   Neighbor solicitation
969 327.820825  2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 ICMPv6   Neighbor solicitation
970 327.819013  2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24
2002:c3d4:6ffd:1101:7232:4dff:fecb:be39 ICMPv6   Neighbor solicitation
971 327.822040  2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1
2002:c3d4:6ffd:1101:7232:4dff:fecb:be39 ICMPv6   Neighbor solicitation
972 327.819306  2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 ICMPv6   Neighbor solicitation
973 327.823102  2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 ICMPv6   Neighbor solicitation
974 327.820142  2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24
2002:c3d4:6ffd:1101:7232:4dff:fecb:be39 ICMPv6   Neighbor solicitation
975 327.824261  2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1
2002:c3d4:6ffd:1101:7232:4dff:fecb:be39 ICMPv6   Neighbor solicitation
976 327.820156  2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24 ICMPv6   Neighbor solicitation
977 327.825618  2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1
2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24 ICMPv6   Neighbor solicitation
978 327.821422  2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8
2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24 ICMPv6   Neighbor solicitation
979 327.826683  2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1
2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24 ICMPv6   Neighbor solicitation
980 327.821438  2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 ICMPv6   Neighbor solicitation
981 327.827836  2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 ICMPv6   Neighbor solicitation
982 327.821833  2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1 ICMPv6   Neighbor solicitation
983 327.829129  2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1
2002:c3d4:6ffd:1101:7232:4dff:fecb:be39 ICMPv6   Neighbor advertisement
984 327.822328  2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24
2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1 ICMPv6   Neighbor solicitation
985 327.830217  2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1
2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24 ICMPv6   Neighbor advertisement
986 327.823615  2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8
2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1 ICMPv6   Neighbor solicitation
987 327.831441  2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 ICMPv6   Neighbor advertisement
988 327.824320  2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24
2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1 ICMPv6   Neighbor solicitation
989 327.832479  2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1
2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24 ICMPv6   Neighbor advertisement
990 327.824573  2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1 ICMPv6   Neighbor advertisement
991 327.833648  2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8
2002:c3d4:6ffd:1101:7232:4dff:fecb:be39 ICMPv6   Neighbor solicitation
992 327.834406  2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24
2002:c3d4:6ffd:1101:7232:4dff:fecb:be39 ICMPv6   Neighbor solicitation
993 327.835232  2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24
2002:c3d4:6ffd:1101:7232:4dff:fecb:be39 ICMPv6   Neighbor solicitation
994 327.835948  2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24
2002:c3d4:6ffd:1101:7232:4dff:fecb:be39 ICMPv6   Neighbor solicitation
995 327.826260  2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24
2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1 ICMPv6   Neighbor advertisement
996 327.837087  2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24 ICMPv6   Neighbor solicitation
997 327.837617  2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8
2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24 ICMPv6   Neighbor solicitation
998 327.827347  2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24
2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1 ICMPv6   Neighbor advertisement
999 327.827379  2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8
2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1 ICMPv6   Neighbor advertisement
1000 327.838983  2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 ICMPv6   Neighbor solicitation
```

```
1001 327.839579  2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 ICMPv6   Neighbor solicitation
1002 327.840174  2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 ICMPv6   Neighbor solicitation
1003 327.828348  2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8
2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1 ICMPv6   Neighbor advertisement
1004 327.828650  2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1 ICMPv6   Neighbor advertisement
1005 327.829862  2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1 ICMPv6   Neighbor advertisement
1006 327.830776  2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1 ICMPv6   Neighbor solicitation
1007 327.842238  2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1
2002:c3d4:6ffd:1101:7232:4dff:fecb:be39 ICMPv6   Neighbor advertisement
1008 327.830942  2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8
2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1 ICMPv6   Neighbor solicitation
1009 327.843315  2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 ICMPv6   Neighbor advertisement
1010 327.831310  2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8
2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1 ICMPv6   Neighbor advertisement
1011 327.832015  2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1 ICMPv6   Neighbor advertisement
1012 327.832587  2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8
2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1 ICMPv6   Neighbor advertisement
1013 327.832878  2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1 ICMPv6   Neighbor advertisement
1014 327.833450  2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8
2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1 ICMPv6   Neighbor advertisement
1015 327.834075  2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1 ICMPv6   Neighbor advertisement
1016 327.834540  2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8
2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1 ICMPv6   Neighbor advertisement
1017 327.835027  2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1 ICMPv6   Neighbor advertisement
1018 327.836427  2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:2e0:7dff:fee1:fbc1 ICMPv6   Neighbor advertisement
```

18.7 Ethereal IPv6 packets summary: on Router's eth1 after handover

```
No. Time        Source                Destination           Protocol Info
  1 0.000000    fe80::208:54ff:fe03:fff1 ff02::2              ICMPv6   Router
solicitation
  2 0.000098    fe80::208:54ff:fe03:fff1 ff02::1              ICMPv6   Router
advertisement
  3 0.001911    2002:c3d4:6ffd:2202:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24 IPv6     Unknown (0x3e)
  4 0.005327    2002:c3d4:6ffd:2202:208:54ff:fe03:fff1
2002:c3d4:6ffd:2202:7232:4dff:fecb:be39 ICMPv6   Neighbor solicitation
  5 0.006563    2002:c3d4:6ffd:2202:7232:4dff:fecb:be39
2002:c3d4:6ffd:2202:208:54ff:fe03:fff1 ICMPv6   Neighbor solicitation
  6 0.007477    2002:c3d4:6ffd:2202:208:54ff:fe03:fff1
2002:c3d4:6ffd:2202:7232:4dff:fecb:be39 ICMPv6   Neighbor advertisement
  7 0.008312    2002:c3d4:6ffd:2202:7232:4dff:fecb:be39
2002:c3d4:6ffd:2202:208:54ff:fe03:fff1 ICMPv6   Neighbor advertisement
  8 0.009096    2002:c3d4:6ffd:1101:2c0:26ff:feb5:a24
2002:c3d4:6ffd:2202:7232:4dff:fecb:be39 IPv6     Unknown (0x3e)
  9 23.447565   2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 TCP      1514 > 49152 [ACK] Seq=12345741
Ack=12345734 Win=37782 Len=46
 10 23.585459   2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8
2002:c3d4:6ffd:1101:7232:4dff:fecb:be39 TCP      49152 > 1514 [ACK] Seq=12345734
```

```
Ack=12345787 Win=29954 Len=0
 11 33.106548   2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8
2002:c3d4:6ffd:1101:7232:4dff:fecb:be39 TCP      49152 > 1514 [ACK] Seq=12345734
Ack=12345787 Win=29954 Len=17
 12 33.166377   2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 TCP      1514 > 49152 [ACK] Seq=12345787
Ack=12345751 Win=37782 Len=0
 13 68.897671   2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 TCP      1514 > 49152 [ACK] Seq=12345787
Ack=12345751 Win=37782 Len=10
 14 68.985326   2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8
2002:c3d4:6ffd:1101:7232:4dff:fecb:be39 TCP      49152 > 1514 [ACK] Seq=12345751
Ack=12345797 Win=29954 Len=0
 15 81.017736   2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 TCP      1514 > 49152 [FIN, ACK] Seq=12345797
Ack=12345751 Win=37782 Len=0
 16 81.185318   2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8
2002:c3d4:6ffd:1101:7232:4dff:fecb:be39 TCP      49152 > 1514 [ACK] Seq=12345751
Ack=12345798 Win=29954 Len=0
 17 81.585708   2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8
2002:c3d4:6ffd:1101:7232:4dff:fecb:be39 TCP      49152 > 1514 [FIN, ACK] Seq=12345751
Ack=12345798 Win=29954 Len=0
 18 137.834238  fe80::208:54ff:fe03:fff1 ff02::1              ICMPv6   Router
advertisement
 19 383.126189  2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8
2002:c3d4:6ffd:2202:7232:4dff:fecb:be39 TCP      49152 > 1514 [SYN] Seq=12345678 Ack=0
Win=30000 Len=0
 20 383.221545  2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 TCP      1514 > 49152 [SYN, ACK] Seq=12345678
Ack=12345679 Win=37800 Len=0
 21 383.225012  2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8
2002:c3d4:6ffd:2202:7232:4dff:fecb:be39 TCP      49152 > 1514 [ACK] Seq=12345679
Ack=12345679 Win=37800 Len=27
 22 384.220113  2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 TCP      1514 > 49152 [ACK] Seq=12345679
Ack=12345706 Win=37800 Len=0
 23 399.271254  2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 TCP      1514 > 49152 [ACK] Seq=12345679
Ack=12345706 Win=37800 Len=4
 24 399.384050  2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8
2002:c3d4:6ffd:2202:7232:4dff:fecb:be39 TCP      49152 > 1514 [ACK] Seq=12345706
Ack=12345683 Win=29996 Len=0
 25 419.584996  2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8
2002:c3d4:6ffd:2202:7232:4dff:fecb:be39 TCP      49152 > 1514 [ACK] Seq=12345706
Ack=12345683 Win=29996 Len=9
 26 419.654114  2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 TCP      1514 > 49152 [ACK] Seq=12345683
Ack=12345715 Win=37791 Len=0
 27 432.601719  2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 TCP      1514 > 49152 [ACK] Seq=12345683
Ack=12345715 Win=37791 Len=23
 28 432.783956  2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8
2002:c3d4:6ffd:2202:7232:4dff:fecb:be39 TCP      49152 > 1514 [ACK] Seq=12345715
Ack=12345706 Win=29996 Len=0
 29 446.854926  2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8
2002:c3d4:6ffd:2202:7232:4dff:fecb:be39 TCP      49152 > 1514 [ACK] Seq=12345715
Ack=12345706 Win=29996 Len=11
 30 447.050742  2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 TCP      1514 > 49152 [ACK] Seq=12345706
Ack=12345726 Win=37789 Len=0
 31 453.924838  2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8
2002:c3d4:6ffd:2202:7232:4dff:fecb:be39 TCP      49152 > 1514 [FIN, ACK] Seq=12345726
Ack=12345706 Win=29996 Len=0
 32 454.050839  2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 TCP      1514 > 49152 [ACK] Seq=12345706
Ack=12345727 Win=37789 Len=0
 33 454.821201  2002:c3d4:6ffd:1101:7232:4dff:fecb:be39
2002:c3d4:6ffd:1101:5258:1ff:fe9d:7eb8 TCP      1514 > 49152 [FIN, ACK] Seq=12345706
Ack=12345727 Win=37789 Len=0
```

19 Appendix B

19.1 Utility files

Some files were created in order to set the configuration of the mobile boards and the Router PC.

19.2 File addroutes

This is a shell script file to set the routes in kernel IPv6 routing table, in the Router.

```
#/bin/sh
route -A inet6 add 2002:C3D4:6FFD:1101::/64 eth0
route -A inet6 add 2002:C3D4:6FFD:2202::/64 eth1
```

19.3 File setrouter.c

This file is compiled and executed in the Router to configure it according to [14].

```
#include <stdio.h>


#define SYSCTL_SPD              "/proc/sys/net/livsix/spd"
#define SYSCTL_SAD              "/proc/sys/net/livsix/sad"
#define SYSCTL_PREFIX_0
"/proc/sys/net/livsix/conf/eth0/ra_prefix_00/ra_pfl_prefix"
#define SYSCTL_PREFIX_1
"/proc/sys/net/livsix/conf/eth1/ra_prefix_00/ra_pfl_prefix"
#define SYSCTL_PREFIX_0_LEN
"/proc/sys/net/livsix/conf/eth0/ra_prefix_00/ra_pfl_prefixlen"
#define SYSCTL_PREFIX_1_LEN
"/proc/sys/net/livsix/conf/eth1/ra_prefix_00/ra_pfl_prefixlen"
#define SYSCTL_RAS_SEND0
"/proc/sys/net/livsix/conf/eth0/ra_send_ras"
#define SYSCTL_RAS_SEND1
"/proc/sys/net/livsix/conf/eth1/ra_send_ras"
#define SYSCTL_ISRO             "/proc/sys/net/livsix/isrouter"
#define SYSCTL_MAX_RA0
"/proc/sys/net/livsix/conf/eth0/ra_maxra_interval"
#define SYSCTL_MAX_RA1
"/proc/sys/net/livsix/conf/eth1/ra_maxra_interval"
#define SYSCTL_MIN_RA0
```

```
"/proc/sys/net/livsix/conf/eth0/ra_minra_interval"
#define SYSCTL_MIN_RA1
"/proc/sys/net/livsix/conf/eth1/ra_minra_interval"
#define LEN_TEXT 255


int main()
{
   FILE *sysctl_entry;
   char text[LEN_TEXT + 1];

/*eth0*/
   if ((sysctl_entry = fopen(SYSCTL_PREFIX_0, "r+")) == NULL)
   {
      sprintf(text, "Failed to open file %s\n", SYSCTL_PREFIX_0);
      perror (text);
      return -1;
   }

   fputs ("2002:c3d4:6ffd:1101::", sysctl_entry);
   fclose (sysctl_entry);

   if ((sysctl_entry = fopen (SYSCTL_PREFIX_0_LEN, "r+")) == NULL)
   {
      sprintf(text, "Failed to open file %s\n", SYSCTL_PREFIX_0_LEN);
      perror (text);
      return -1;
   }

   fputs ("64", sysctl_entry);
   fclose (sysctl_entry);


/*eth1*/

   if ((sysctl_entry = fopen(SYSCTL_PREFIX_1, "r+")) == NULL)
   {
      sprintf(text, "Failed to open file %s\n", SYSCTL_PREFIX_1);
      perror (text);
      return -1;
   }

   fputs ("2002:c3d4:6ffd:2202::", sysctl_entry);
   fclose (sysctl_entry);

   if ((sysctl_entry = fopen (SYSCTL_PREFIX_1_LEN, "r+")) == NULL)
   {
      sprintf(text, "Failed to open file %s\n", SYSCTL_PREFIX_1_LEN);
      perror (text);
      return -1;
   }

   fputs ("64", sysctl_entry);
   fclose (sysctl_entry);
```

```
/*eth0*/

   if ((sysctl_entry = fopen (SYSCTL_MAX_RA0, "r+")) == NULL)
   {
      sprintf(text, "Failed to open file %s\n", SYSCTL_MAX_RA0);
      perror (text);
      return -1;
   }

   fputs("1800", sysctl_entry);
   fclose (sysctl_entry);

   if ((sysctl_entry = fopen (SYSCTL_MIN_RA0, "r+")) == NULL)
   {
      sprintf(text, "Failed to open file %s\n", SYSCTL_MIN_RA0);
      perror (text);
      return -1;
   }

   fputs("800", sysctl_entry);
   fclose (sysctl_entry);

   if ( ( sysctl_entry = fopen (SYSCTL_RAS_SEND0, "r+") ) == NULL )
   {
      sprintf(text, "Failed to open file %s\n", SYSCTL_RAS_SEND0);
      perror (text);
      return -1;
   }

   fputc('1', sysctl_entry);
   fclose (sysctl_entry);


/*eth1*/

   if ((sysctl_entry = fopen (SYSCTL_MAX_RA1, "r+")) == NULL)
   {
      sprintf(text, "Failed to open file %s\n", SYSCTL_MAX_RA1);
      perror (text);
      return -1;
   }

   fputs("1800", sysctl_entry);
   fclose (sysctl_entry);

   if ((sysctl_entry = fopen (SYSCTL_MIN_RA1, "r+")) == NULL)
   {
      sprintf(text, "Failed to open file %s\n", SYSCTL_MIN_RA1);
      perror (text);
      return -1;
   }

   fputs("800", sysctl_entry);
   fclose (sysctl_entry);

   if ( ( sysctl_entry = fopen (SYSCTL_RAS_SEND1, "r+") ) == NULL )
```

```
   {
      sprintf(text, "Failed to open file %s\n", SYSCTL_RAS_SEND1);
      perror (text);
      return -1;
   }

   fputc('1', sysctl_entry);
   fclose (sysctl_entry);

   if ((sysctl_entry = fopen (SYSCTL_ISRO, "r+")) == NULL)
   {
      sprintf(text, "Failed to open file %s\n", SYSCTL_ISRO);
      perror (text);
      return -1;
   }

   fputc ('1', sysctl_entry);
   fclose (sysctl_entry);

   return 0;
}
```

19.4 File setval.c

This file is compiled and then executed in the shell to insert any value in a “/proc” file.

```
#include <stdio.h>

#define LEN_FILENAME 255
#define LEN_VALUE    150



int main(int argc, char **argv)
{
   FILE    *sysctl_entry;
   char value[LEN_VALUE + 1];
   char filename[LEN_FILENAME + 1];

   if (argc == 3)
   {
      sprintf(filename, "%s", argv[2]);
```

```
      printf("File %s\n", filename);
      sprintf(value, "%s", argv[1]);
      printf("Value %s\n", value);
   }
   else
   {
      perror("Two arguments must be entered: value file\n");
      exit(1);
   }


   if ((sysctl_entry = fopen(filename, "r+")) == NULL)
   {
      perror ("Cannot open file\n");
      exit(1);
   }

   fputs(value, sysctl_entry);
   fclose (sysctl_entry);

   return 0;
}
```

19.5 File setdefint.c

This file is compiled and then executed in the shell to set the default interface.

```
#include <stdio.h>

#define SYSCTL_INTERFACE_CONF "/proc/sys/net/livsix/conf"
#define LEN_FILENAME 255
```

```
int main(int argc, char **argv)
{
   FILE    *sysctl_entry;
   char interface[LEN_FILENAME + 1];

   if (argc == 2)
   {
      sprintf(interface,  "%s/%s/defint",  SYSCTL_INTERFACE_CONF,
argv[1]);
      printf("Interface %s is the default\n", argv[1]);
      printf("File %s\n", interface);
   }
   else
   {
      perror("One argument must be entered: interface\n");
      exit(1);
   }

   if ((sysctl_entry = fopen(interface, "r+")) == NULL)
   {
      perror ("Cannot open file\n");
      exit(1);
   }

   fputc ('1', sysctl_entry);
   fclose (sysctl_entry);

   return 0;
}
```

20 Bibliography

[1] Andrew S. Tanenbaum, Maarten van Steen. “Distributed Systems, Principles and Paradigms”. Prentice Hall. 2002.

[2] William Stallings. “Network Security Essentials, Applications and Standards”. Prentice Hall. 2000.

[3] Douglas E. Comer. “Internetworking with TCP/IP, Principles, Protocols and Architectures. Volume 1”. Prentice Hall. Fourth Edition. 2000.

[4] Dave Wisely, Philip Eardley, Louise Burness. “IP for 3G, Networking Technologies For Mobile Communications”. WILEY. 2002.

[5] Alessandro Rubini, Jonathan Corbet. “LINUX DEVICE DRIVERS”. O’Reilly. Second Edition. 2001.

[6] W. Richard Stevens. “UNIX NETWORKING PROGRAMMING, Networking APIs: Sockets and XTI. Volume 1”. Prentice Hall. Second Edition. 1998.

[7] C. Perkins. Network Working Group. “IP Mobility Support for IPv4”. RFC 3344. August 2002.

[8] D. Johnson, C. Perkins, J. Arkko. IETF Mobile IP Working Group. “Mobility Support in IPv6”. Draft-ietf-mobileip-ipv6-24.txt . Internet-Draft. June 2003.

[9] S. Deering, R. Hinden. Network Working Group. “Internet Protocol, Version 6 (IPv6), Specification”. RFC 2460. December 1998.

[10] T. Narten, E. Nordmark, W. Simpson. Network Working Group. “Neighbor Discovery for IP Version 6 (IPv6)”. RFC 2461. December 1998.

[11] S. Thomson, T. Narten. Network Working Group. “IPv6 Stateless Address

Autoconfiguration". RFC 2462. December 1998.

[12] M. Crawford. Network Working Group. "Transmission of IPv6 Packets over Ethernet Networks". RFC 2464. December 1998.

[13] R. Hinden, S. Deering. Network Working Group. "IP Version 6 Addressing Architecture". RFC 2373. July 1998.

[14] Alexandru Petrescu, Emmanuel Riou. "HOWTO use a LIVSIX box as a router. A guide to LIVSIX routing configuration". http://www.enrl.motlabs.com/livsix . May 2003.

[15] Alexandru Petrescu, Emmanuel Riou. "IPv6 Applications over LIVSIX. A guide to the IPv6 applications which are running over LIVSIX". http://www.enrl.motlabs.com/livsix . February 2003.

[16] Alexandru Petrescu, Emmanuel Riou. "The LIVSIX mobility HOWTO. A detailed guide to LIVSIX mobility configuration and use". http://www.enrl.motlabs.com/livsix. January 2003.

21 Web Sites

[17] LIVSIX web site. http://www.enrl.motlabs.com/livsix - main contact for the author: Alexandru Petrescu – Authors can be found at: http://www.enrl.motlabs.com/livsix/livsix/AUTHORS

[18] 68K/ColdFire web site: http://www.freescale.com/webapp/sps/site/homepage.jsp?nodeId=018rH3YTLC

[19] uClinux web site: www.uclinux.org

[20] BusyBox web site: www.busybox.net

[21] microcom development page in sourceforge.net: http://sourceforge.net/projects/microcomste

[22] Vocera Communications web site: www.vocera.com . All the information regarding the device, as well as the picture, was obtained from this site documentation.