Tampere University of Technology
Department of Information Technology

Heikki Vatiainen

Implementation and testing of Multi-Protocol Over ATM client on Linux

Master of Science Thesis

Subject approved by the department council on 14.4.1999

Supervisors: Prof. Jarmo Harju
Dr. Tech Mika Grundström

Foreword

This Master of Science thesis was done at the Tampere University of Technology during the spring of 1999. The thesis is part of the Faster Pro project which includes a range of topics on areas such as IP over ATM and QoS for IP networks.

I would like to thank Prof. Jarmo Harju for guidance, my cow-orkers Juha Laine and especially Sampo Saaristo with whom I did the coding. I would also like to thank Richard Parke for introducing me to the world of networks and computing. Thanks for support go to my girlfriend Johanna and KPIG 107.5 radio for providing the tunes.

Tampere, 30th August 1999

Heikki Vatiainen
Vaajakatu 5 K 210
33720 Tampere
Finland
Heikki.Vatiainen@iki.fi

Abstract

TAMPERE UNIVERSITY OF TECHNOLOGY

Department of Information Technology

Telecommunications Laboratory

Heikki Vatiainen: Implementation and testing of Multi-Protocol Over ATM client on Linux

Master of Science thesis: 57 pages

Supervisors: Prof. Jarmo Harju and Dr. Tech Mika Grundström

August 1999

This master's thesis describes the implementation of Multi-Protocol Over ATM (MPOA) Client for the Linux operating system and the results acquired from testing and measuring a multivendor MPOA network. The thesis was done as a part of Faster Pro project at Tampere University of Technology.

MPOA is a solution for routed networks to benefit more from the underlying ATM network and its QoS properties. MPOA offers more scalability and performance, as well as lower latency than it is possible to get from a router based network.

MPOA builds on many existing protocols and combines them to a scalable layer 3 switched routing solution.

This thesis gives a brief introduction to the existing protocols MPOA builds on, introduces the MPOA protocol and then proceeds to discuss the MPOA client implementation for the Linux operating system. The MPOA client implementation was part of the interoperability and performance testing which was done with the University's multivendor ATM/MPOA network. Results from those tests follow the discussion about the implementation. Finally, some measurements and thoughts about the QoS properties of MPOA are presented.

The MPOA client implementation is currently distributed as a part of the ATM on Linux package.

Tiivistelmä

TAMPEREEN TEKNILLINEN KORKEAKOULU

Tietotekniikan osasto

Tietoliikennetekniikan laitos

Heikki Vatiainen: Implementation and testing of Multi-Protocol Over ATM client on Linux

Diplomityö: 57 sivua

Tarkastajat: Prof. Jarmo Harju ja TkT Mika Grundström

Elokuu 1999

Tässä diplomityössä esitellään Multi-Protocol Over ATM (MPOA) -asiakassovelluksen toteutus Linux-käyttöjärjestelmään. Tehtyä sovellusta käytettiin osana usean valmistajan laitteista koostuvan MPOA-verkon toimivuuden, suorituskyvyn sekä käytettävyyden testauksessa. Työ tehtiin osana Faster Pro -tutkimusprojektia Tampereen teknillisessä korkeakoulussa.

MPOA tarjoaa ATM:n päällä tapahtuvalle reititykselle mahdollisuuden hyötyä tehokkaammin alla olevan ATM-verkon ominaisuuksista. MPOA:n etuja reititinverkkoon verrattuna ovat parempi laajennettavuus, pienempi viive ja suurempi suorituskyky.

MPOA-protokolla pohjautuu useisiin olemassa oleviin protokolliin yhdistäen ne skaalautuvaksi 3-tason kytkentäratkaisuksi.

Diplomityön alussa esitellään protokollat, joiden varaan MPOA:n toiminta perustuu, käydään lävitse MPOA-protokolla sekä Linux-käyttöjärjestelmään tehdyn MPOA-asiakkaan toteutus. Toteutuksen jälkeen kuvataan TTKK:n usean eri toimittajan laitteista koostuvaa ATM/MPOA-verkkoa, ja siinä tehtyjä yhteensopivuus- ja suorituskykykokeiluja. Työn loppuosassa esitellään palvelunlaatuun (QoS) perustuvia mittaustuloksia sekä pohditaan hieman MPOA:n palvelunlaatuun liittyviä ominaisuuksia.

Työssä kuvattu MPOA-asiakas on tällä hetkellä osa ATM-on-Linux -pakettia.

Contents

List of Acronyms

API	Application Programming Interface
ARP	Address Resolution Protocol
ARPA	Advanced Research Projects Agency
ATM	Asynchronous Transfer Mode
BUS	Broadcast and Unknown Server
CBR	Constant Bit Rate
CDV	Cell Delay Variation
DLL	Data Link Layer
ELAN	Emulated LAN
ICMP	Internet Control Message Protocol
IEEE	The Institute of Electrical and Electronics Engineers
INET	ARPA Internet protocols
I/O	Input/Output
IP	Internet Protocol
ITU	International Telecommunication Union
ITU-T	ITU - Telecommunication Standardization Sector
LAN	Local Area Network
LANE	LAN Emulation
LE_ARP	LANE ARP
LEC	LAN Emulation Client
LECS	LAN Emulation Configuration Server
LES	LAN Emulation Server
LLC	Logical Link Control
LNNI	LANE Network Network Interface
LUNI	LANE User Network Interface
MAC	Medium Access

MPOA	Multi-Protocol Over ATM
MPC	MPOA Client
MPS	MPOA Server
NHC	Next Hop Resolution Protocol Client
NHRP	Next Hop Resolution Protocol
NHS	NHRP Server
NNI	Network Node Interface or Network – Network Interface
NTP	Network Time Protocol
PNNI	Private Network to Network Protocol
PVC	Permanent Virtual Channel
QoS	Quality of Service
RFC	Request for Comments
RSVP	Resource reSerVation Protocol
SDU	Service Data Unit
SNAP	SubNetwork Access Protocol
SVC	Switched Virtual Channel
TCP	Transmission Control Protocol
TLV	Type – Length – Value
TUT	Tampere University of Technology
UBR	Unspecified Bit Rate
UDP	User Datagram Protocol
UNI	User – Network Interface
VC	Virtual Channel
VCC	Virtual Channel Connection
VCI	Virtual Channel Identifier
VLAN	Virtual LAN
VPI	Virtual Path Identifier

List of Figures

List of Tables

Chapter 1

Introduction

This master's thesis describes the implementation of Multi-Protocol Over ATM (MPOA) Client for the Linux operating system and the results from testing and measuring a multivendor MPOA network. The thesis was done as a part of Faster Pro project at Tampere University of Technology.

The goal of this thesis was to examine an MPOA network by implementing an MPOA client and testing it as a part of an MPOA enabled network. Starting from the basic interoperability testing, the performance, QoS and other properties were tested and measured. One of the goals was also to add MPOA client support for the existing ATM on Linux package, which already supports a great number of ATM protocols and applications.

MPOA is based on existing technologies so a brief introduction to ATM, LAN Emulation and ATM on Linux is given in Chapter 2.

Chapter 3 describes MPOA and gives an introduction to MPOA basics and the problems it was designed to solve. The MPOA protocol and components are introduced with some examples of what an MPOA network might look like.

The implementation of Linux MPOA client is described in Chapter 4. The kernel and user space parts are discussed as well as their interfaces to other Linux components.

The tests and test results from the TUT multivendor MPOA network are discussed in Chapter 5. Various test were done to verify the interoperability between implementations done by different vendors, to measure the possible increase in network performance and to find out how MPOA might help solving problems with the jitter caused by routers.

MPOA is usually touted as a technique that helps to increase network performance. Chapter 6 discusses what other possibilities and applications an MPOA network might have.

Finally, Chapter 7 discusses the project conclusions.

Chapter 2

The Foundation – ATM, LANE and ATM on Linux

2.1 ATM – Asynchronous Transfer Mode

ATM is a network technology in which the data transfer is done using small, fixed size packets called cells. The transfer is done by switching cells between devices called ATM switches. The fixed size of the ATM cells makes fast, hardware based switching easy and efficient.

Before the cells can be switched, a connection must be established across the network to set up the switching tables in the ATM switches. The switching information in the cell headers is only valid between two switches, so all the switches between the connection end points must know about the new flow of cells that is about to enter the network.

Since a path across the network must be established before the data transfer can start, it is possible to define and set the Quality of Service characteristics a connection requires from the network. Cells for which no connection is available, can not enter the network and compete for resources with the existing connections, so the QoS reservation is guaranteed

to be available during the lifetime of the connection.

There are two types of ATM connections. Permanent Virtual Connections (PVCs) are static and always available while dynamic Switched Virtual Connections (SVCs) are created on demand and utilize ATM signalling which establishes the connections based on ATM addresses. PVCs are created by using network management tools or manually by some other means. One manual configuration method is using `telnet(1)` to connect the switch and then configuring the connections by hand. Both connection types have their own advantages in terms of ease of implementation, ease of management and other factors.

ATM network makes a clear distinction between ATM switches and ATM edge devices such as routers, hosts and Ethernet switches. The interface between two ATM switches is called NNI or Network Node Interface and interface between an ATM switch and edge device is called UNI or User Network Interface. Both interfaces share common functions such as ATM signalling, but some functions, such as call routing, are only defined for NNI interface.

The ATM standardization work is done by two organizations, ITU-T and ATM Forum. Many ATM Forum documents, such as ATM signalling, are based on the work done by ITU-T.

2.2 LANE – LAN Emulation Over ATM

The LAN Emulation Over ATM service allows end systems such as ATM attached hosts, routers and LAN bridges access ATM network as if they were attached to a traditional LAN. LANE is specified by the ATM Forum and is currently at its second revision, LANEv2. The LANEv2 specification is backwards compatible with LANEv1 and allows LANEv1 clients to co-operate with LANEv2 service. The LANEv2 specification is divided into two documents, LANE User Network Interface (LUNI) and LANE Network Network Interface (LNNI). [LANE, LUNIv2]

The main characteristics of legacy LANs, such as Ethernet/IEEE 802.3 and Token Ring (IEEE 802.5), are connectionless data transfer, multicast and broadcast service through shared medium and network independent addresses. ATM by contrast is connection oriented, non-broadcast and uses addresses that are closely connected to the network topology. [802.3, 802.5]

The LANE specification defines a method in which the ATM network uses software emulation to emulate IEEE 802.x LANs on Medium Access (MAC) level. The MAC frames are encapsulated by a software layer and then transferred across the ATM network by LANE clients. By encapsulating MAC frames and not for example IP packets, support for most of the existing applications is achieved.

2.2.1 LAN Emulation basics

An emulated LAN is composed of one or more LAN Emulation clients (LECs) and single LANE service. The LANE service includes LAN Emulation Configuration Server (LECS) LAN Emulation Server (LES) and Broadcast and Unknown Server (BUS).

LANE clients are responsible for encapsulating MAC frames into LANE frames and bridging them across ATM VCCs. The VCCs are created by first resolving LAN destinations (Ethernet MAC addresses or Token Ring MAC addresses and route descriptors) to ATM addresses and then initiating VCCs using the resolved ATM address. The resolving of LAN destinations to ATM addresses is done with the help of LAN Emulation Server.

The distribution of unicast frames is done between two LECs across a point-to-point VCC. The distribution of broadcast and multicast frames is done with the help of Broadcast and Unknown Server. When a LEC needs to send a broadcast or multicast frame, it sends the frame to the BUS which then distributes it to all the LECs that have joined the ELAN.

The LANEv1 specification had support for PVC connections but the new LANEv2 specification only supports SVC connections. The LANEv2 specification also requires support

for point-to-multipoint connections.

2.2.2 Strengths and weaknesses of LANE

Before the IEEE 802.1Q VLAN standard was approved, LANE was the only method for creating multivendor VLANs. If the underlying ATM network was supporting SVCs and the LAN Emulation Configuration Server was reachable using the Well-Known LECS address, adding new LANE clients to existing ELANs was essentially a plug-and-play operation. To add a new client to an existing ELAN, only the name of the ELAN was required for successful configuration. [802.1Q]

Even if the IEEE 802.1Q is now a standard, LANE is still perfectly usable for creating multivendor VLANs. LANE based networks have been in production use for years now, so the initial problems have already been solved. The current LANE products are well tested and have proved to be reliable.

The flexibility offered by LANE does not come without drawbacks. The efficient end-to-end VCCs are specified only within a network layer subnet and routers are still needed for communication across subnet boundaries.

With LANEv1, the LANE service was a single point of failure where loosing the LES or BUS would bring down the whole emulated LAN. This weakness has been addressed in the upcoming LANEv2 LUNI specification which defines redundant LANE service.

The LANE specification is quite complex and complexity usually results in software bugs. The inclusion of resolving and mapping MAC addresses to ATM addresses and VCCs adds yet another layer in the existing software stacks. This results in lower performance compared to non-emulated LANs.

2.3 ATM on Linux

ATM support for the Linux operating system has been under active development since 1995. The project was started by MSc Werner Almesberger and is currently at its 55th release. The project still continues with steady pace and the ATM support will probably be included in the Linux distribution kernels in the near future. [Linux-ATM]

The ATM on Linux distribution supports a wide variety of ATM related protocols and utilities. The features include support for a number of ATM network cards, UNI 3.0, 3.1 and 4.0 unicast signalling, UBR and CBR traffic categories, support for PVCs and SVCs through a Berkeley sockets based API, Classical IP, LANE and MPOA.

Chapter 3

Multi-Protocol Over ATM

3.1 What is MPOA?

Multi-Protocol Over ATM is a method for efficient transfer of inter-subnet unicast data in LANE environment. The existing functionality of LANE is preserved while allowing direct communication over ATM VCCs across network layer subnet boundaries.

These ATM VCCs, often called as MPOA shortcuts, bypass the routers decreasing the router load and lowering the possibility for network bottlenecks resulting from traffic congestion on heavily used links.

MPOA introduces MPOA Clients (MPCs) and MPOA Servers (MPSs) which co-operate with the existing LANE clients and LANE service. The MPCs query their MPSs for shortcut information and with that information create and utilize the shortcut connections.

MPOA is specified by the ATM Forum. The version 1.0 of the MPOA specification was released in July 1997. [MPOA]

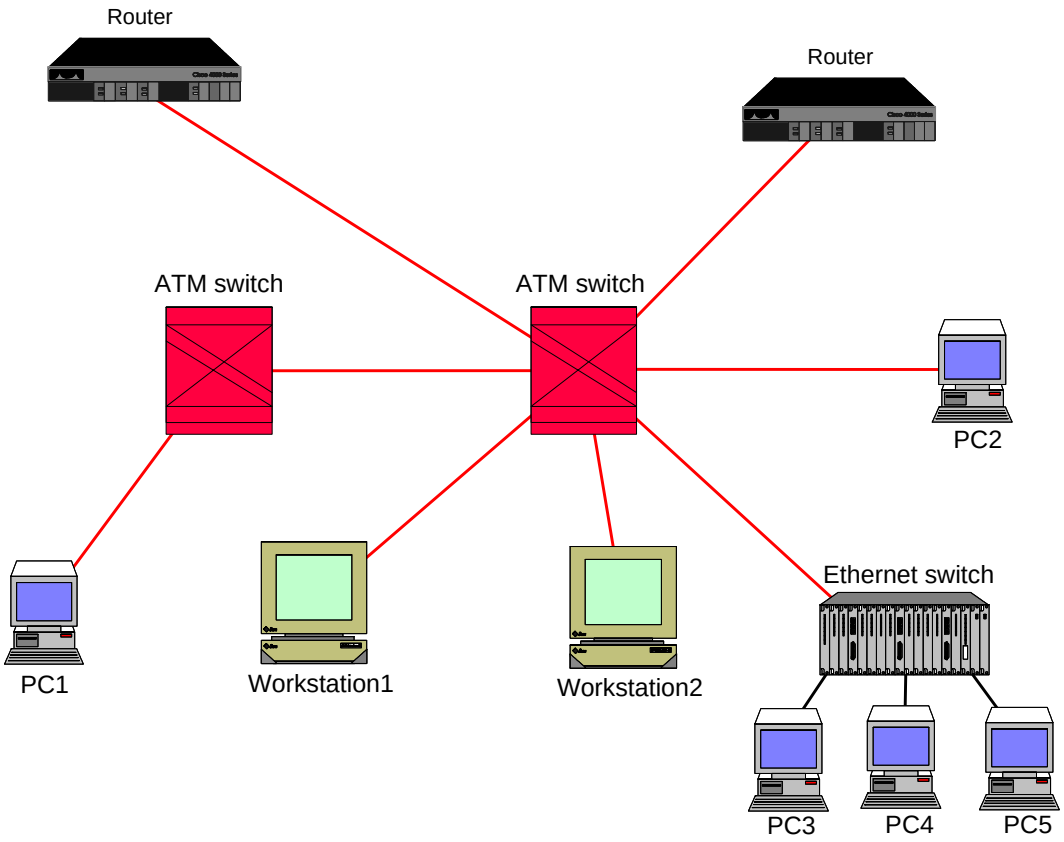


Figure 3.1: Network layout at the physical layer

3.2 Problems with the LANE method

One of the goals of Multi-Protocol Over ATM is to resolve the problems with the current LAN Emulation model. The LANE model, which is effective within a network layer subnet, creates bottlenecks when packets need to be routed between subnets. The problems arise from the fact that the physical network topology needs not to correspond to the logical network layer topology which is used in routing.

Figure 3.1 shows what an ATM based network might physically look like. It is worth noting that the routers are attached to the rest of the network with a single link. This implies that all the routed packets have to pass twice through the same link.

Figure 3.2 shows the same network seen from the network layer. The ATM and Ethernet switches are not visible and the routers have a central position in the network.

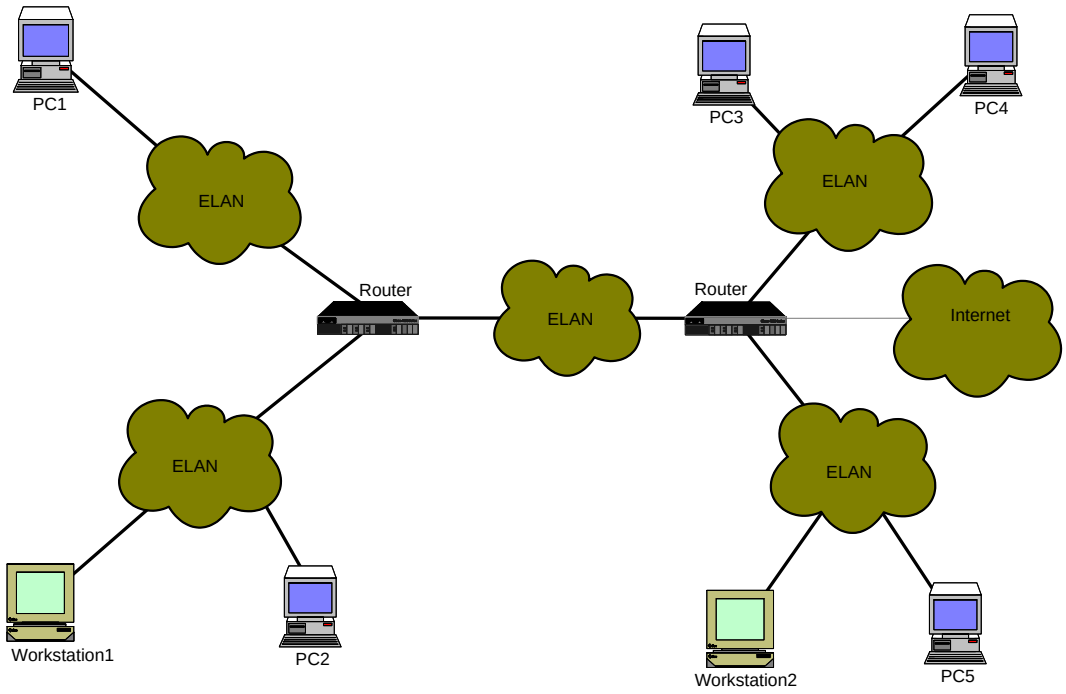


Figure 3.2: Network layout at the network layer

Even if the physical topology is hidden from the network layer, it still dictates how the actual flow of packets passes through the network. By comparing Figure 3.1 and Figure 3.2, it is clear that a high amount of traffic is passing through the routers and the routers' connections to the network can start hindering the network performance.

3.3 Virtual routing

The solution that MPOA provides to the LANE performance problems is the better usage of the underlying ATM network. Two LANE clients, which reside in the same network layer subnet, can establish direct VCCs between each other. MPOA extends direct VCCs across network layer subnets by distributing the routing functions across the MPOA enabled ATM network.

Since the routers are not needed on the data path between subnets, MPOA effectively separates the two functions of router: route calculation and packet forwarding. The route

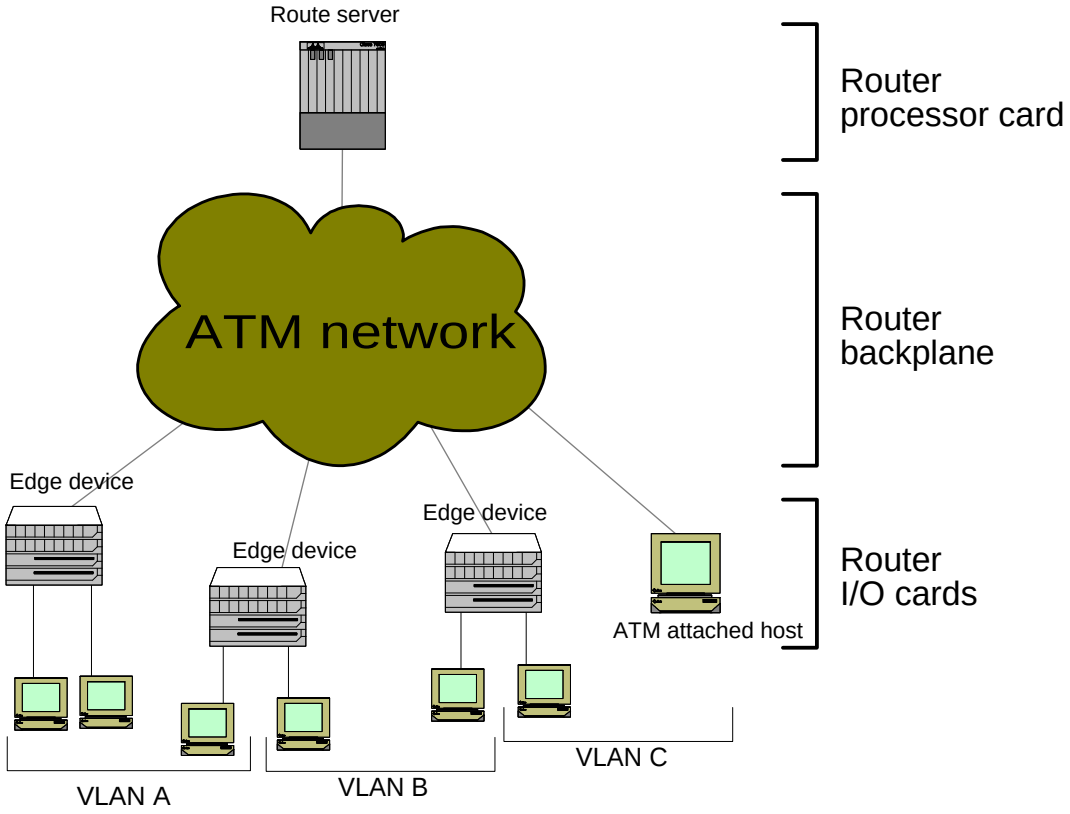


Figure 3.3: Virtual router

calculation in the MPOA model is done by the MPOA servers, also called route servers, and the packet forwarding is the responsibility of the MPOA clients which reside in ATM edge devices.

This technique, in which routing functions are distributed across the network, is often called virtual routing. The benefits virtual routing has over legacy router based networks include *(a)* effective communication across subnet boundaries and *(b)* increased scalability and manageability since the number of routers can be reduced.

Figure 3.3 illustrates how a traditional router can be divided to components and how these components correspond to the components of an MPOA system.

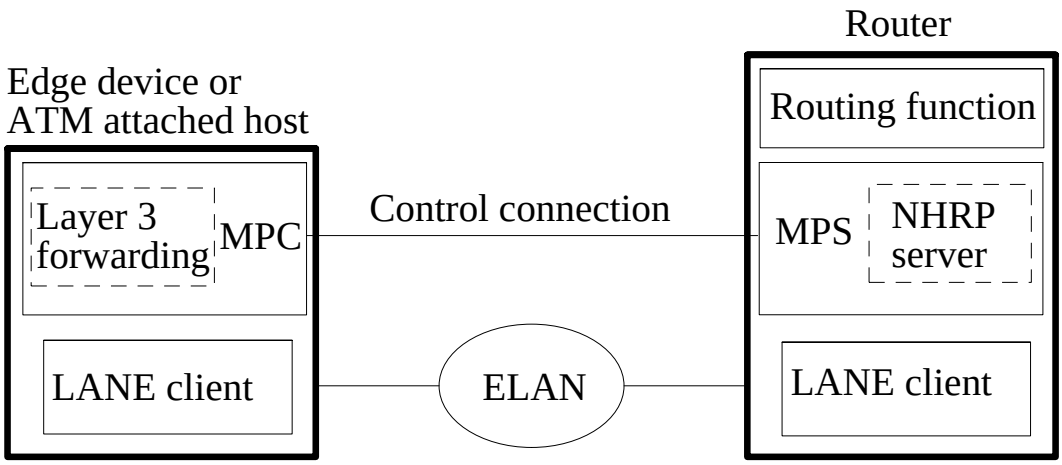


Figure 3.4: The Components in an MPOA system

3.4 MPOA components

MPOA has client – server architecture in which MPOA clients and MPOA servers are connected via LANE. Figure 3.4 shows how MPCs and MPSs relate to other devices in the network.

Just as Figure 3.4 shows, the MPOA components are not stand-alone, but embedded in other network devices.

3.4.1 MPOA Client

The MPOA client is responsible for sourcing and sinking internetwork shortcuts. Because the MPC must both source and sink the shortcuts, it operates in two roles, ingress and egress. Even if the roles are separate from each other, the MPC must be able to support them simultaneously.

An ingress MPC does the flow detection by monitoring LANE frames destined to the default router. If the default router has an MPS in it, the MPC can query the MPS for the ATM address corresponding to the destination IP address. If the query is successful, the

ingress MPC can establish a shortcut VCC to the egress MPC and start directing the packet flow over the new shortcut. The packet format used over shortcuts is described later in Section 3.5.5, which talks about data transfer in more detail.

An egress MPC is the terminating point of a shortcut and receives the packets the ingress MPC forwards over the shortcut. The egress MPC also removes the encapsulation used on the shortcut and adds the DLL encapsulation imposed by the MPS serving the egress MPC. The packets are then passed to the upper layers of the network stack.

The DLL encapsulation is required to make MPOA transparent to the upper layers. The re-encapsulation ensures that the interface presented by MPOA is similar to LANE and the upper layers in the software stack need not to be modified.

3.4.2 MPOA Server

Since the MPOA clients do all the data forwarding in an MPOA system, the role of the MPOA server is to provide the MPOA clients the information they need for creating shortcuts. An MPS needs to interact with the co-located routing function and the Next Hop Server to respond to the queries from ingress MPCs and provide DLL encapsulation information to egress MPCs.

3.5 MPOA in action

Both MPOA components, MPSs and MPCs, perform the following operations when operational:

Configuration Obtaining configuration parameters needed during the following operations

Discovery Learning about other MPSs and MPCs in the same subnet

Target Resolution Determining the ATM address and other information needed when establishing shortcuts to forward packets across subnets

Connection Management Creating, maintaining and terminating control and shortcut VCCs

Data Transfer Forwarding network layer packets across shortcut VCCs.

3.5.1 Configuration

MPOA components need a number of configuration parameters during their operation. One example parameter is the flow threshold needed in flow detection. The components can be configured by using LECS or by some local methods which include the use of predefined configuration parameters from the MPOA specification.

3.5.2 Discovery

Before MPCs can issue MPOA Resolution Requests or MPSs can impose egress cache entries, they need to discover each others presence. The discovery process is embedded in the LE_ARP protocol and happens when LANE clients resolve MAC addresses to ATM addresses.

LANE clients serving MPOA components add MPOA Device Type TLV in the LANE LE_ARP frames they send and inform their MPOA component about any MPOA Device Type TLVs they receive. The MPOA Device Type TLV indicates the type of the MPOA component and its control ATM address. If the component is MPS, the TLV can also include the MAC address(es) the MPS has.

The extended LE_ARP frames are not specified in LANEv1, so MPOA needs LANEv2 support to work. The discovery protocol is designed so that LANEv1 compliant devices can exist in the same ELAN with LANEv2 and MPOA enabled LANEv2 clients.

3.5.3 Target Resolution

The target resolution allows an MPC to determine the ATM address of the endpoint of a shortcut. The MPC does the resolution in cooperation with its MPS. The MPS propagates the resolution request along the network layer routed path to the MPC at the end point of the future shortcut.

Figure 3.5 illustrates the MPOA resolution process. The protocol messages used during the resolution are shown as well as the resulted shortcut. The MPOA Resolution Request carries the destination network layer address, requestor's ATM address and optionally requestor's network layer address. This information is returned with the destination ATM address in the corresponding MPOA Resolution Reply. MPOA shortcuts are unidirectional so the resolution process must be done separately in both directions if the flow is bidirectional.

IETF's Next Hop Resolution Protocol (NHRP) is used to carry MPOA Resolution Requests and Responses between MPOA servers. NHRP allows NHRP Clients (NHCs) to resolve network layer addresses to ATM addresses in ATM networks. Since MPOA uses NHRP between MPSs, there was no need to define a new protocol between MPOA servers. The MPOA Request/Response packets are based on NHRP protocol packets and only a simple conversion is needed to translate MPOA Resolution/Request packets to the respective NHRP Resolution/Request packets. [RFC 2332]

Another advantage of using NHRP between MPOA servers is the possibility to use routers that support NHRP but do not support MPOA as relays for MPOA resolution packets. Even if some of the routers along the routed path do not support MPOA, they can still forward MPS to MPS NHRP requests and responses.

The resolution process is described below from the perspective of the ingress MPC, ingress MPS, egress MPS and egress MPC.

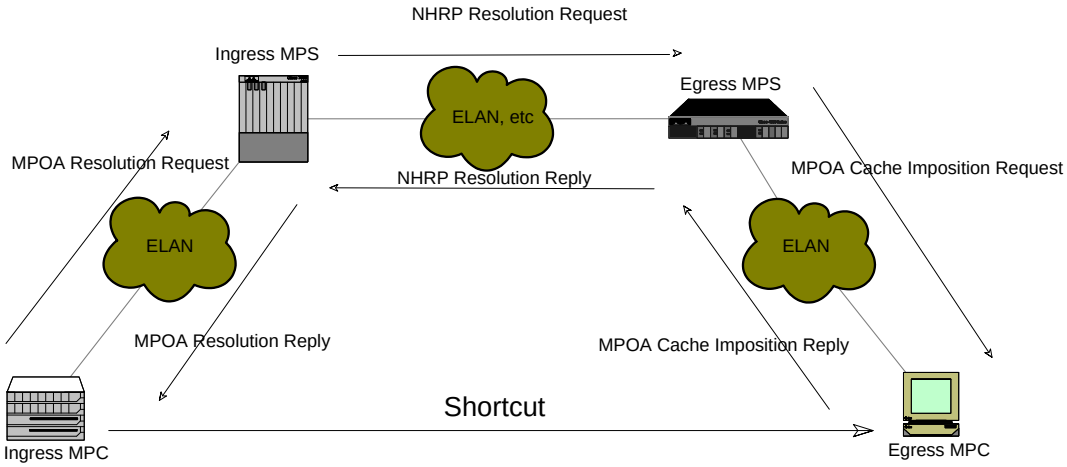


Figure 3.5: The MPOA resolution protocol

Ingress MPC perspective

Initially all the network layer packets destined outside of the MPC's subnet are forwarded to the default router. When the MPC detects a flow of network layer packets to a destination network layer address, it sends an MPOA Resolution Request to its MPS. The control address of the MPS was previously learnt during the device discovery.

It is also possible that the MPC receives an MPOA Trigger message. In that case the MPC behaves like the flow was detected by the MPC itself.

When the ingress MPS receives the requested information about the egress MPC, an MPOA Resolution Reply is returned to the ingress MPC. Resolved entries are aged using a timer called holding time. The holding time is refreshed with a new MPOA Resolution Request as long as the traffic to the destination network layer address exceeds the flow threshold.

Ingress MPS perspective

When the ingress MPS receives an MPOA Resolution Reply from the ingress MPC, it will either have to answer the MPC if the destination is local to the MPS or re-originate the request along the routed path.

When the MPS re-originates the request, it will use its own network layer address as the source protocol address. This ensures that the MPS will receive the respective reply. The protocol between MPOA servers is NHRP and the re-originated MPOA Resolution Replies are sent and received through the co-located NHS.

Egress MPS perspective

When the egress MPS receives an NHRP Resolution Request targeted for an MPC which the MPS serves, the MPS sources an MPOA Cache Imposition Request to the egress MPC.

The MPOA Cache Imposition request contains state and other information the egress MPC needs during the data transfer over the future shortcut. The MPOA Cache Imposition Reply returned by the egress MPC contains the information the egress MPS needs in order to return an NHRP Resolution Reply back to the original requester.

Egress MPC perspective

When the egress MPC receives an MPOA Cache Imposition Request, it has to decide if it can accept the new egress cache entry and the potential shortcut. In any case, the egress MPC must return an MPOA Cache Imposition Reply. If the egress MPC is unable or unwilling to receive the entry or the future shortcut, it must return the appropriate error status with the reply.

3.5.4 Connection management

MPOA components establish VCCs between each other to transfer control messages and data. ATM addresses for control connections are learnt with the device discovery protocol and the addresses for data connections are resolved with target resolution as described above.

The default traffic category for VCCs is UBR but the MPOA specification allows the use of other traffic categories, too.

3.5.5 Data Transfer

Unicast data flows through an MPOA system have two modes: the default flow and the shortcut flow. After a shortcut for a flow has been established between two MPOA clients, the packets belonging to that flow will be sent over the shortcut.

All the other data is sent along the normal routed path. The other data includes packets for which no flow has been detected yet, packets carrying data for a protocol for which flow detection is not enabled and the rest of the traffic such as IP multicast.

The default encapsulation for packets forwarded over a shortcut is the RFC 1483 Encapsulation for Routed Protocols. Alternate encapsulations are also possible but their use is optional. The MPOA specification also defines MPOA Tagged Encapsulation format which seems to be common according to tests done with the TUT MPOA test network. [RFC 1483]

Figure 3.6 shows two shortcuts with their respective default data paths. The dashed arrows show the default routed path the packets were traversing before the shortcut was established.

Shortcut 2 illustrates the situation in which an MPS decides to terminate a shortcut. Host PC1 has issued an MPOA Resolution Request for an IP address that is not within the direct reach of an MPOA enabled router. The router, which is at the edge of the MPOA enabled network, can respond to resolution requests with its own ATM address if it wishes to do so. Even if it was not possible to establish a direct shortcut between two MPOA clients, the optimal exit point from the MPOA enabled network was found.

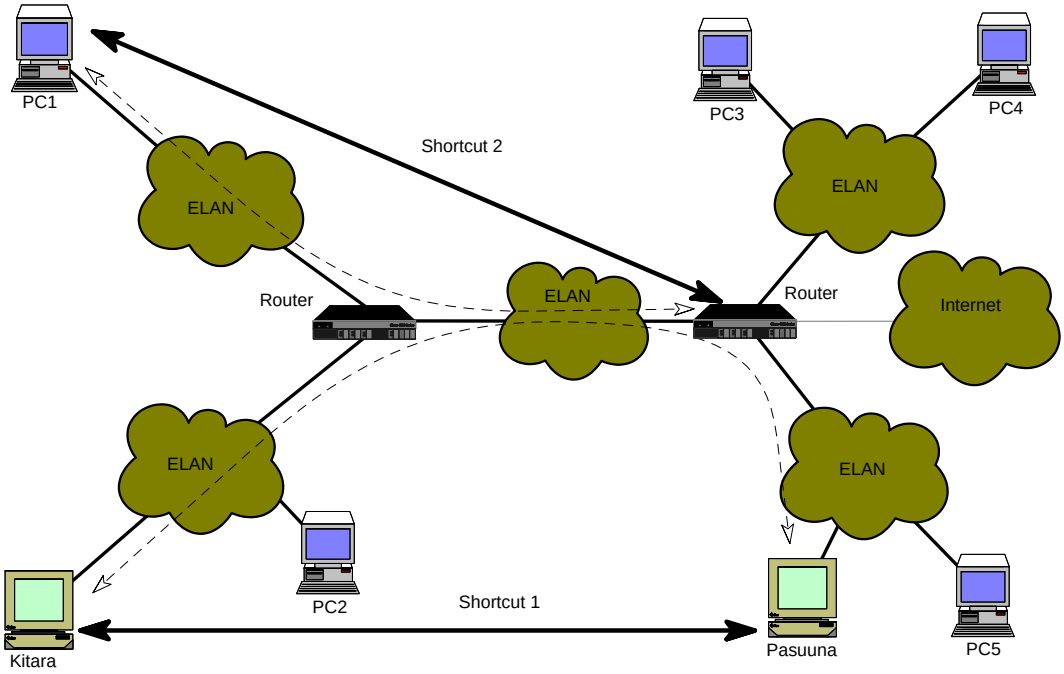


Figure 3.6: Two shortcuts

Chapter 4

Implementation of the Linux MPOA client

4.1 System Design

Like most of the Linux-ATM services, the Linux MPOA client is divided into two parts. The flow detection, packet forwarding and most of the ingress and egress cache maintenance is done by the kernel, while the configuration, connection management and target resolution is done by the user space daemon program. The MPOA device discovery is done in co-operation with LED Zeppelin, the Linux LAN Emulation Daemon.

One of the benefits of a daemon program handling the MPOA control protocol is the smaller size of the kernel component. The kernel code has more privileges than user space programs, so smaller amount of kernel code results in smaller probability of fatal bugs. All the validation of incoming messages is done by the unprivileged user space code where a fatal bug will only cause the program to terminate not risking the whole system integrity.

Another benefit is the possibility to control the MPOA client with normal process control tools such as `kill(1)`. The kill program can be used to signal the MPOA client to terminate

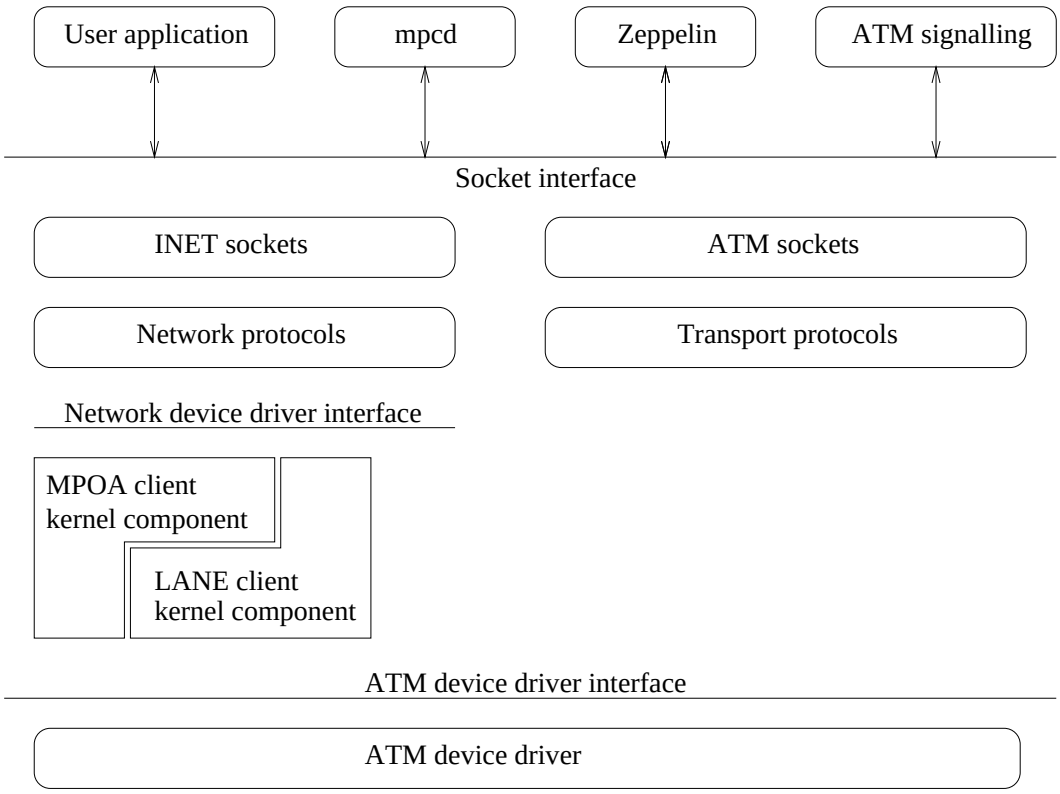


Figure 4.1: Components of the MPOA Client in Linux

or reload itself. Also, the addition of a new MPOA client is just a matter of starting a new daemon process.

Figure 4.1 shows what an MPOA enhanced Linux system looks like.

The kernel provides `/proc` file system interface for monitoring and changing kernel data structures. Through a file in `/proc`, the caches and states of the shortcuts can be easily monitored. Using the `/proc` file system also goes along with the existing Linux conventions.

The separation of duties between the kernel component and the user space program was easy to do by following the example set by the Linux LANE client. Packet forwarding and ingress and egress caches were natural to have in the kernel because of the great overhead which would have resulted from copying data packets and doing the cache lookups between

the kernel and daemon. Events like reception and sending of MPOA control messages can be done by passing messages between the kernel and daemon, since they occur very seldom compared to the receipt and transfer of data messages.

Adding, resetting and removing MPOA clients from a running system was designed from the beginning to be as flexible as possible. The idea of extending the existing LANE components with MPOA functionality was discarded early and the LANE and MPOA components were decoupled from each other as completely as possible.

The MPOA implementation also uses loadable module support provided by the kernel. The module support makes it possible to upgrade parts of the running kernel without the need to reboot the machine.

4.2 Extensions to the existing Linux LANE client

The existing Linux LANE client was extended to support the LANEv2 functionality required by MPOA. The most important feature was the support for LE_ARP based TLVs needed by the MPOA device discovery protocol. Closely related to ARP based TLVs is the LANEv2 Extended API to Higher Layers which is used to associate TLVs with LE_ARP responses and notify higher layers of TLVs received with LE_ARP frames.

4.2.1 MPOA client — LANEv2 client interface

Since the MPOA client is the only user of LANEv2 Extended API to Higher Layers, the API was chosen as the interface between the LANE and MPOA client kernel components. The interface was implemented with a C language structure that has function pointers corresponding to the primitives in the extended API definition. This structure is also the only connection shared by the LANE and MPOA components. The structure is shown in Figure 4.2.

```
static struct lane2_ops lane2_ops = {
        lane2_resolve,          /* resolve,              spec 3.1.3 */
        lane2_associate_req,    /* associate_req,        spec 3.1.4 */
        NULL                    /* associate indicator, spec 3.1.5 */
};
```

Figure 4.2: MPOA client — LANEv2 client interface

The NULL pointer will later be filled in by the MPOA kernel component. This pointer is used by the LANE component to notify the MPOA component about TLVs seen in the LE_ARP frames. The two non-NULL functions are called by the MPOA component to perform requests specified in the LANEv2 spec, section 3.1. The MPOA component can access this structure via a pointer in the LANE device driver.

Using function pointers, which are only valid when the component providing the actual functionality is present, was important for the modular and independent design of Linux LANE and MPOA components. Since both components check the pointers for validity before using them, each one of the components can be loaded and unloaded from kernel at any time. This was especially useful during the implementation and debugging phase when the kernel component was updated gradually without the need of reboots.

The MPOA client flow monitoring is started and stopped with two functions in the kernel part. The idea is to substitute the LANE client's send function with the flow monitoring function from the MPOA client's kernel part. When the upper layers call the send function in the LANE driver, the MPOA flow monitoring function is called instead. The two control functions are shown in Figure 4.3.

The function `mpc_send_packet` does the flow monitoring and packet forwarding over MPOA shortcuts. It also calls the LANE driver's send function (`mpc->old_hard_start_xmit`) for packets that are not forwarded over a shortcut. When the user space daemon program is killed, `stop_mpc` restores the LANE driver's send function back to its original value. The contents of `struct mpoa_client` are introduced in section 4.3.2.

```
static void start_mpc(struct mpoa_client *mpc, struct device *dev)
{
        mpc->old_hard_start_xmit = dev->hard_start_xmit;
        dev->hard_start_xmit = mpc_send_packet;
}
static void stop_mpc(struct mpoa_client *mpc)
{
        mpc->dev->hard_start_xmit = mpc->old_hard_start_xmit;
        mpc->old_hard_start_xmit = NULL;
}
```

Figure 4.3: MPOA flow monitoring controls

4.3 Implementation of MPOA client

The implementation of the user space daemon program, `mpcd(8)`, and the kernel component are discussed in more detail below.

4.3.1 MPOA Client daemon: mpcd

The MPOA client daemon is divided in parts as shown in Figure 4.4.

The I/O component is responsible for accepting and creating ATM VCCs to MPOA server and other MPOA clients. The connections are done in non-blocking fashion to keep the daemon operational while active connections are proceeding. The I/O component also takes care of receiving and sending packets from the kernel and network and dispatching handlers for received packets.

The LECS configuration component retrieves the MPOA client configuration parameters from the LECS. After receiving the parameters, the component notifies the kernel about the new parameters. In MPOAv1, all the parameters affect only the behavior of the kernel part. It is also possible to completely bypass the LECS configuration phase and use the default parameters. The strict decoupling of LANE and MPOA components is the reason why `mpcd(8)` retrieves the configuration information directly without the help of `zeppelin(8)`.

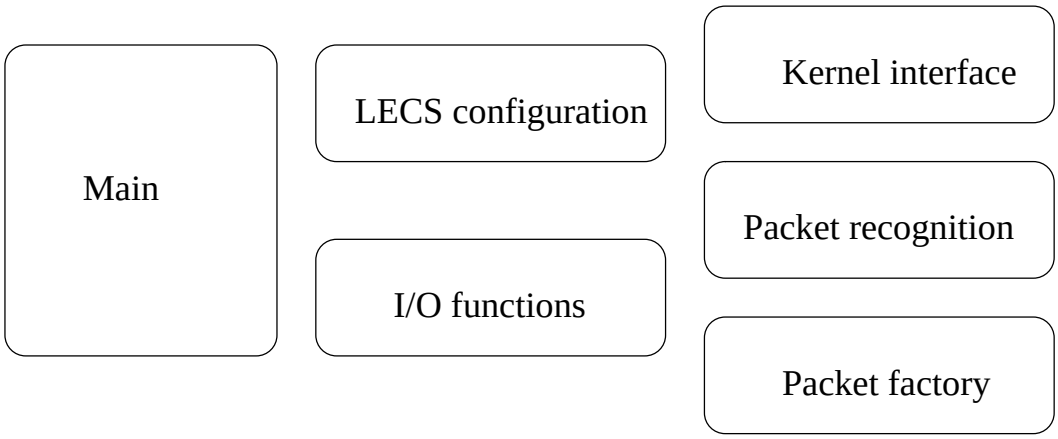


Figure 4.4: mpcd components

All the outgoing MPOA control messages are created by the packet factory component. The packet factory component has a number of functions such as `send_resolution_request()`, which requests the transmission of an MPOA Resolution Request control message with the parameters supplied by the caller.

All the incoming MPOA control packets are passed by the I/O component to the packet recognition component. This component validates the packets and calls the appropriate function based on the packet type and contents. Most of the control messages cause a message to be sent to the kernel, but some messages, such as MPOA Keep Alive, are handled by the component itself.

Kernel interface component receives and sends messages to and from the MPOA client kernel part. Besides requesting the transmission of MPOA Resolution Request, the kernel interface also controls the MPOA Keep Alive state machine.

The implementation of MPOA Keep Alive protocol does not have its own component but is controlled mutually by all the components.

When the daemon receives a signal that would normally cause the termination of the process, it requests the kernel to first remove all the egress cache entries. The MPS will not remove the egress cache entries when the connection to MPC is closed but instead uses nor-

mal time out mechanisms to age out the entries. If the MPOA client is restarted before the entries are timed out, the cache coherency will be lost. The only signal treated specially is SIGHUP, which also clears the egress cache but instead of terminating the daemon causes it re-initialize itself.

4.3.2 MPOA Client kernel component

The Linux MPOA client kernel component is composed of three files which are linked together into one object file. Instead of being linked statically in the kernel, the files can also be compiled to a loadable kernel module. The files correspond to the three parts the whole kernel component is divided to. The parts are the main part, the caches part and the `/proc` file system part.

The main part

The main part initializes the other two parts, performs the flow detection and subsequent packet forwarding and provides the interface to the user space daemon program. The main part also utilizes the LANEv2 extended API to higher layers to receive the TLVs learnt by the LANE client and to associate the MPOA Device Type TLV with its LANE client.

The main part also has pointers to the `struct mpoa_client` list and `struct atm_mpoa_qos` list. The purpose of `struct atm_mpoa_qos` is to hold user requested QoS requirements for the shortcuts. Section 6.1 discusses the QoS support in more detail.

The control structure representing one MPOA client, `struct mpoa_client`, is shown in Figure 4.5. The kernel component is written so that it can handle multiple MPOA clients as opposed to the daemon which only represents one MPOA client.

The `struct mpoa_client` is a self contained unit which consists of a pointer to the LANE device it does flow monitoring for, pointer to LANE device's send function, pointer to the internal VCC connected to the daemon program and pointers to ingress and egress cache

```
struct mpoa_client {
        struct mpoa_client *next;
        struct device *dev;          /* lec in question                    */
        int dev_num;                 /* e.g. 2 for lec2                    */
        int (*old_hard_start_xmit)(struct sk_buff *skb, struct device *dev);
        struct atm_vcc *mpoad_vcc;   /* control channel to mpoad           */
        uint8_t mps_ctrl_addr[ATM_ESA_LEN];  /* MPS control ATM address    */
        uint8_t our_ctrl_addr[ATM_ESA_LEN];  /* MPC's control ATM address  */

        rwlock_t ingress_lock;
        struct in_cache_ops *in_ops; /* ingress cache operations           */
        in_cache_entry *in_cache;    /* the ingress cache of this MPC      */

        rwlock_t egress_lock;
        struct eg_cache_ops *eg_ops; /* egress cache operations            */
        eg_cache_entry *eg_cache;    /* the egress  cache of this MPC      */

        uint8_t *mps_macs;           /* array of MPS MAC addresses, >=1    */
        int number_of_mps_macs;      /* number of the above MAC addresses  */
        struct mpc_parameters parameters;  /* parameters for this client   */
};
```

Figure 4.5: `struct mpoa_client`

entries and their respective operations. The struct also holds ATM and MAC addresses for this MPC and its MPS.

Ingress and egress caches

The caches part contains the implementation of MPOA client's ingress and egress cache. The cache is accessed through function pointers in `struct mpoa_client`. For example, a search by IP address `uint32_t ipaddr` from the ingress cache of an MPOA client is done as `mpc->in_ops->search(ipaddr,mpc);`. This calls function `in_cache_search()` in file `mpoa_caches.c` via a function pointer. When the cache related functions are called via function pointers in structures, only the function which fills in the `in_ops` and `eg_ops` members in `struct mpoa_client` needs to be visible from the caches part. All the other functions can be declared `static` reducing the kernel name space pollution.

```
urku:~# cat /proc/atm/mpc
QoS entries for shortcuts:
IP address
  TX:max_pcr pcr     min_pcr max_cdv max_sdu
  RX:max_pcr pcr     min_pcr max_cdv max_sdu

Interface 2:

Ingress Entries:
IP address      State      Holding time  Packets fwded  VPI  VCI
130.230.54.146  resolved   593           151            0    85

Egress Entries:
Ingress MPC ATM addr
Cache-id        State      Holding time  Packets recvd  Latest IP addr   VPI VCI
47002300000003030001000206002048065 2de00
13              resolved   1193          151            130.230.54.146   0   85
```

Figure 4.6: Two shortcuts

`/proc` file system

The `/proc` file system part lets user space programs access the kernel data structures. With `/proc/atm/mpc`, the state of the ingress and egress cache entries can be viewed. An example of this is shown in Figure 4.6.

The purpose of `QoS entries for shortcuts` field is discussed later in section 6.1.1. Field `Interface 2` indicates that the ingress and egress entries that follow belong to the MPOA client which is associated with interface `lec2`. Currently there is only one entry in both the ingress and egress caches and both entries have an active VCC associated with them.

The field `IP address` in `Ingress Entries:` part displays the IP address this entry was created for. The `Holding time` field shows the current holding time in seconds. When the holding time reaches zero, the entry is removed from the ingress cache. If the entry has an active VCC associated with it, the `VPI` and `VCI` fields show its connection identifiers. Finally, `Packets fwded` displays the number of packets forwarded over the shortcut.

The different values for field `State` are shown below. An entry goes to state `refreshing`

when 2/3 of the holding time has elapsed. Note that the state of the shortcut VCC does not affect the state of an ingress entry. An entry can be valid even if establishing a shortcut did not succeed.

`invalid` The flow threshold for this entry has not yet been exceeded

`resolving` An MPOA Resolution Reply has been sent, but no successful response has been received

`resolved` A successful MPOA Resolution Reply was received. If a shortcut has been opened, the VPI and VCI fields show the connection identifiers

`refreshing` The entry is valid and will be refreshed to `resolved` state if the flow threshold is exceeded.

The `Egress Entries:` part displays entries imposed by the egress MPS. Field `Ingress MPC ATM addr` shows the ATM address of the ingress MPC which caused the imposition of this entry. The `State` field is currently always `resolved` which means the entry is valid and has holding time left. `Packets recvd` is the number of packets received over the shortcut and `Latest IP addr` displays the source IP address from the most recent packet received over the shortcut. The reason the field is called `Latest IP addr` comes from the fact that the LEC associated with the ingress MPC might be a proxy LEC e.g. Ethernet switch, which is sending packets from multiple end stations over the same shortcut.

Figure 4.6 also shows how the same VCC (VPI 0, VCI 48) is used by both ingress MPC and egress MPC. The Linux implementation first tries to find an already open VCC from the caches before initiating signalling for a new VCC. The same optimization was noticed in other MPOA implementations too.

4.3.3 Communication between kernel and `mpcd`

The communication between the kernel and `mpcd(8)` is based on messages sent over a special ATM VCC. The order of messages does not matter i.e. the message protocol itself is stateless. However, there is some state information shared between the kernel and `mpcd(8)`. One example is the state of the MPOA Keep Alive protocol. When an MPOA Resolution Reply or MPOA Cache Imposition Request is received, `mpcd(8)` starts the keep alive state machine. When the kernel notices there are no more ingress or egress cache entries left, it requests `mpcd(8)` to stop the keep alive state machine.

The interface between kernel and `mpcd(8)` is discussed below with the descriptions of the messages that both entities pass to each other.

Kernel-`mpcd(8)` interface

The communication socket between the kernel and `mpcd(8)` is a normal ATM socket which is converted to a special MPOA control socket with `ioctl(ATMMPC_CTRL)`. The similar method is also used for shortcut sockets which are attached to the kernel with `ioctl(ATMMPC_DATA)`.

The kernel-`mpcd(8)` interface is represented by an ATM device driver in the kernel. The ATM device driver method lets `mpcd(8)` to communicate with the kernel using the same system calls, such as `read()` and `write()`, as it uses with other MPOA components.

The kernel never processes MPOA control messages even if they are received over shortcut VCCs. The control messages are passed to `mpcd(8)` which then processes them.

The structure of messages passed between kernel and `mpcd(8)` is shown in Figure 4.7.

The contents of `union content` differ depending on the message type. The `in_info` and `eg_info` structure members are the control information fields needed in ingress and egress cache entries. When e.g. an MPOA Cache Imposition Request is received, `mpcd(8)` can

```
struct k_message{
        uint16_t type;
        uint32_t ip_mask;
        uint8_t  MPS_ctrl[ATM_ESA_LEN];
        union {
                in_ctrl_info in_info;
                eg_ctrl_info eg_info;
                struct mpc_parameters params;
        } content;
        struct atm_qos qos;
};
```

Figure 4.7: Structure of kernel-`mpcd(8)` messages

Message	Purpose
`SND_MPOA_RES_RQST`	Requests send of an MPOA Resolution Request
`SET_MPS_CTRL_ADDR`	Notifies `mpcd(8)` about the control ATM address of MPS
`SND_MPOA_RES_RTRY`	Requests send of an MPOA Resolution Request for refreshing an ingress cache entry
`STOP_KEEP_ALIVE_SM`	Both caches are empty, stop keep alive state machine
`SND_EGRESS_PURGE`	Requests send of an MPOA Egress Cache Purge Request
`DIE`	Requests `mpcd(8)` to terminate itself
`DATA_PLANE_PURGE`	Results from an egress cache miss or MPS death. In case of an egress cache miss, received by `mpcd(8)` over a shortcut VCC. Results in Ingress Purge over dataplane
`OPEN_INGRESS_SVC`	Requests to create a shortcut VCC

Table 4.1: Messages from kernel to `mpcd(8)`

directly compose and pass the kernel the control information of the new egress cache entry.

Message types

The message types between kernel and daemon are presented in Tables 4.1 and 4.2. The usage of the fields in `struct k_message` depends on the message type.

Message	Purpose
MPOA_TRIGGER_RCVD	Carries information received with MPOA Trigger
MPOA_RES_REPLY_RCVD	An MPOA Resolution Reply was received
INGRESS_PURGE_RCVD	An NHRP Purge Request was received. Remove the invalid ingress entry
EGRESS_PURGE_RCVD	An MPOA Egress Cache Purge Request was received. Remove the associated egress entry
MPS_DEATH	The MPS failed to send MPOA Keep-Alive messages
CACHE_IMPOS_RCVD	An MPOA Cache Imposition Request was received (and accepted)
SET_MPC_CTRL_ADDRESS	Associate this ATM address with LANE client as our MPC control ATM address
SET_MPS_MAC_ADDR	Use this MAC for router's MAC address when detecting flows
CLEAN_UP_AND_EXIT	Clear both caches and notify `mpcd(8)` when done
SET_MPC_PARAMS	Use these MPC parameters instead of default ones

Table 4.2: Messages from `mpcd(8)` to kernel

Chapter 5

Tests and results from a working MPOA system

5.1 The MPOA test network

A number of different tests and measurements were done with an MPOA test network at TUT. The network consisted of three ATM switches, Fore ASX-200BX, FORE ASX-1000 and Cisco LS1010. The first tests with MPOA enabled equipment were done during the summer of 1998 with one router, Fore ASN-9000, and four ATM enabled hosts. The hosts were two Sun workstations and two Linux PCs. The network was expanded with a Cisco router and additional Linux PCs before the end of 1998.

Both routers and most of the edge devices were connected to the Cisco ATM switch. The ATM switches had multiple connections between them and were using PNNI protocol for call routing. Figure 5.1 shows the layout of the laboratory network. All the connections between the devices were 155Mbits fiber or twisted pair links.

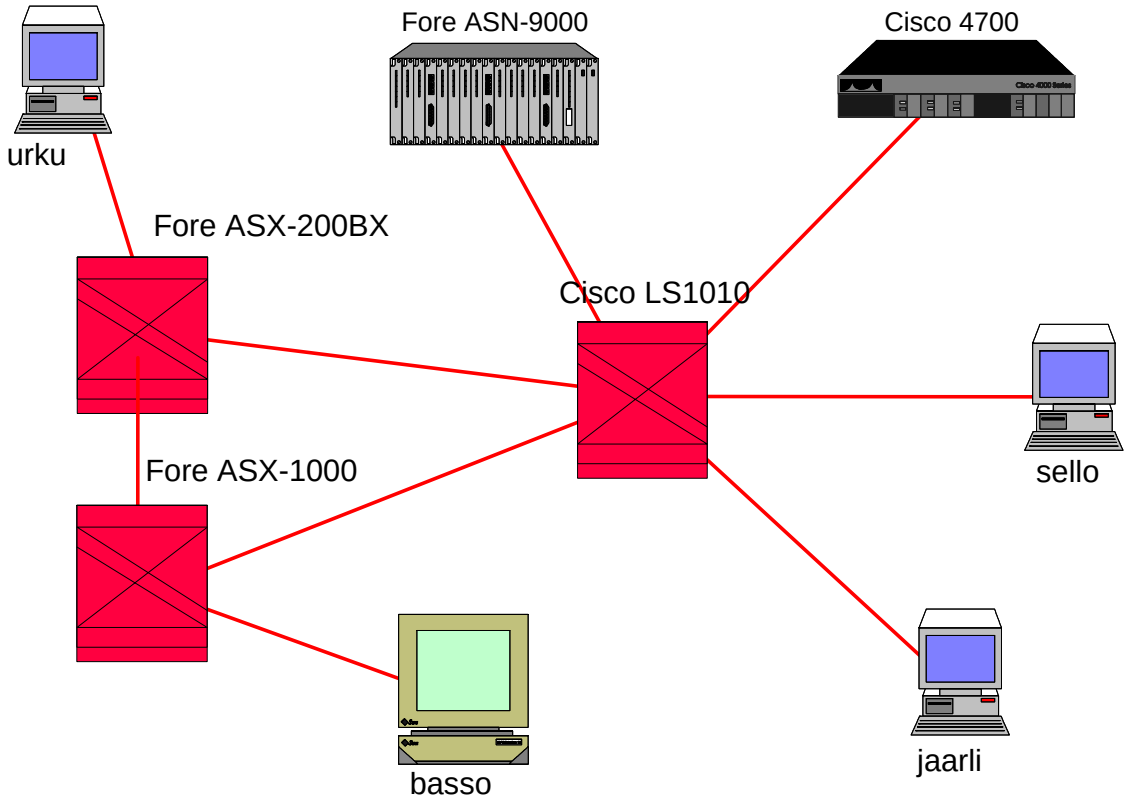


Figure 5.1: Laboratory test network physical topology

The MPOA test network is a part of the TUT core campus network but only the devices shown in the Figure 5.1 were used in the testing.

The hosts were assigned into different IP subnetworks as shown in Figure 5.2. All the ELANs had LANEv2 service and MPOA enabled LANE clients in them. The ELAN between the two routers was dedicated for routing only.

In the beginning of 1999 the Pori School of Technology and Economics, which is part of TUT, joined the MPOA test network. The network connection to Pori is an ATM virtual path tunnel with end points in Pori and TUT. The tunnel peak cell rate was unlimited and the total number of ATM switches between Pori and the test network MPOA servers was at least seven. Five switches are under TUT control and at least two are in the operator network. The geographical distance between Pori and Tampere is about 180 kilometers (110 miles).

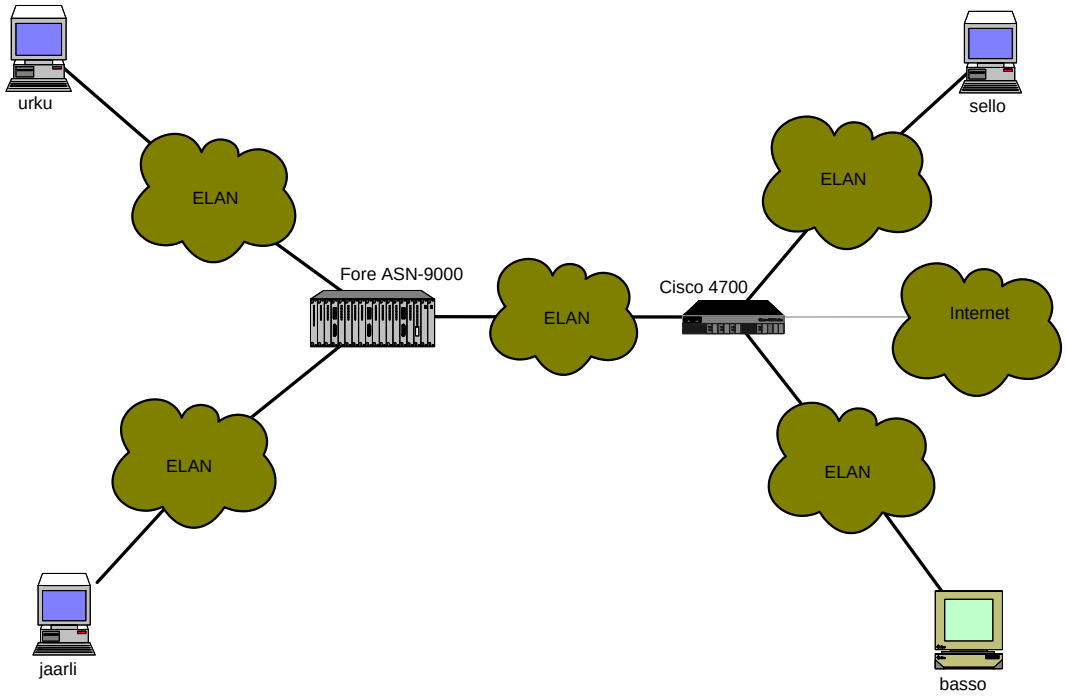


Figure 5.2: Laboratory test network IP level topology

The equipment in Pori consisted of two MPOA clients which were using the test network's MPOA servers. This configuration made it possible to experiment how well MPOA functions with geographically wide VLANs.

5.2 MPOA interoperability and initial testing

Internetworking between Fore, Cisco and Linux implementations was surprisingly smooth. All three implementations had initially small problems which were reported and fixed during the testing period.

The experiments indicate that multivendor testing in heterogeneous environment is beneficial to testing the software for hidden bugs. Both Fore and Cisco had fatal bugs and some of the bugs were only seen when the other vendor's software did something unexpected.

The Linux implementation also had its problems but the decision to validate and parse pack-

ets in user space did not cause the whole system to become unstable even if the `mpcd(8)` daemon dies. The design of Linux MPOA client also ensures that the system will still function as a normal LANE client even if the MPOA client parts is not operational or dies.

5.3 Results of performance testing

The performance tests clearly show that MPOA can increase network throughput by bypassing routers. The Cisco 4700 router was a real bottleneck since its ATM module was only capable of routing about 35Mbits per second. When the 4700 was bypassed, the high end Linux PCs (urku, PentiumPro200 MHz and jaarli, PentiumII 350MHz) got much higher readings from `netperf(1)` benchmarking program.

The Fore ASN-9000 router is capable of routing much higher amount of traffic but its 155Mbits link can easily be saturated with the current PCs and workstations. The problem is common with any single armed router configuration including Fast Ethernet.

All the tests were done using the `netperf(1)` program. Netperf is freely available, comes with extensive documentation, supports a number of different tests and benchmarks and can be compiled on many different platforms. [Netperf]

The netperf runs were done between two hosts using TCP streams. Each stream lasted 60 seconds. Each test was repeated five times in a row and the average TCP throughput and average difference from the average was recorded.

5.3.1 Machine configurations and loopback performance

Table 5.1 lists host configurations and the results from running `netperf(1)` over the loopback interface. The loopback results serve at most as comparisons between machine performance since *(a)* loopback interface is usually optimized to not to go through the whole network stack and *(b)* the host functions both as a server and client during the test.

Host	CPU	Memory	ATM NIC	Loopback Mbit/s
urku	PentiumPro 200MHz	64MB	Efficient	271.66 ± 17.22
jaarli	PentiumII 350MHz	128MB	Fore LE 155	593.47 ± 15.53
sello	Pentium 133MHz	64MB	Fore PCA200E	154.85 ± 1.95
basso	UltraSparc 143MHz	64MB	Fore PCA200E	306.25 ± 6.05

Table 5.1: Machine configurations

The Linux PCs were running Linux 2.2.3 and ATM on Linux 0.55. The Sun was running Solaris 2.6 and version 5.0.0.7 of Fore ATM software. Fore ASN-9000 was running ForeThought 5.0.2, patchlevel 11453 and Cisco 4700 was running IOS 12.0.3. The host configurations are shown in Table 5.1.

5.3.2 Single ELAN performance

The test was conducted by running a series of `netperf(1)` tests between all the hosts. The hosts were in the same ELAN and there was only one active stream per host at a time. The test results are shown in Table 5.2

The TCP implementation in Linux 2.2.3 has problems with high speed LANs. These problems are causing the relatively low results the host jaarli has in the single ELAN testing. The problems are visible in the later tests too.

5.3.3 One stream over one router

The test was conducted by running a series of `netperf(1)` tests across one router. Only one stream was active at a time. The test results are shown in Table 5.3. Hosts see much better performance over the Cisco router but little lower over ASN-9000. The possible reasons for the lower performance of MPOA are discussed later in Section 5.3.7.

Hosts	LANE Mbit/s
urku – jaarli	41.08 ± 5.16
jaarli – urku	124.21 ± 0.73
sello – basso	70.99 ± 0.88
basso – sello	62.24 ± 0.16
urku – basso	69.36 ± 0.36
basso – urku	108.43 ± 0.34
jaarli – sello	57.67 ± 0.12
sello – jaarli	36.25 ± 1.94

Table 5.2: Single ELAN performance

Hosts	LANE Mbit/s	MPOA Mbit/s
urku – jaarli	114.57 ± 1.58	48.06 ± 4.96
jaarli – urku	119.12 ± 0.30	98.20 ± 0.58
sello – basso	30.59 ± 0.08	68.72 ± 0.30
basso – sello	20.19 ± 0.61	49.34 ± 0.17

Table 5.3: One stream over one router

Hosts	LANE Mbit/s	MPOA Mbit/s
urku – sello	22.98 ± 1.34	47.39 ± 0.19
sello – urku	25.00 ± 1.07	57.64 ± 0.97
jaarli – sello	24.16 ± 1.11	48.03 ± 0.22
sello – jaarli	16.99 ± 0.73	37.39 ± 5.29
urku – basso	32.95 ± 0.27	83.05 ± 0.36
basso – urku	26.37 ± 1.34	106.41 ± 0.41
jaarli – basso	32.36 ± 0.39	81.49 ± 0.21
basso – jaarli	12.03 ± 0.44	81.21 ± 2.96

Table 5.4: One stream over two routers

5.3.4 One stream over two routers

The test was conducted by running a series of `netperf(1)` tests across two routers. Only one stream was active at a time. The test results are shown in Table 5.4. In this test the Cisco router is hindering the overall LANE performance. MPOA shortcut streams see much better throughput than the routed LANE streams.

5.3.5 Two streams over two routers

The test was conducted by running two series of `netperf(1)` tests across two routers. Two streams were active at a time. The test results are shown in Table 5.5. When the number of concurrent streams increases, the benefits of MPOA become more visible. The shortcut streams see about 100Mbit/s higher throughput than the routed LANE streams.

Hosts	LANE Mbit/s	MPOA Mbit/s
urku – sello	14.70 ± 0.43	47.32 ± 0.12
jaarli – basso	14.65 ± 0.45	81.58 ± 0.08
Total	29.35	128.90
sello – urku	23.75 ± 0.37	55.80 ± 1.26
basso – jaarli	5.52 ± 0.40	82.83 ± 2.77
Total	29.27	138.63
sello – jaarli	7.62 ± 0.22	82.56 ± 0.14
urku – basso	21.01 ± 0.53	47.97 ± 2.00
Total	23.63	130.53

Table 5.5: Two streams over two routers

Hosts	LANE Mbit/s	MPOA Mbit/s
urku – basso	5.33 ± 0.15	36.89 ± 0.64
basso – urku	15.62 ± 0.21	65.60 ± 0.40
sello – jaarli	4.35 ± 0.19	24.42 ± 0.28
jaarli – sello	11.11 ± 0.58	34.14 ± 0.45
Total	36.14	161.05

Table 5.6: Four streams over two routers

5.3.6 Four streams over two routers

The test was conducted by running four series of `netperf(1)` tests across two routers. Four streams were active at the same time. The test results are shown in Table 5.6. The four stream test is the first case where the combined rate of the shortcut streams is more than the theoretical rate of an 155 Mbit/s ATM link. Even if the routers had been capable of routing at wire speed, they would have been outperformed by MPOA shortcuts.

5.3.7 Comparison between LANE and MPOA performance

The test results show that almost all the tests involving a router saw better network throughput when MPOA was used. The only exception was in Table 5.3 where the tests over Fore ASN-9000 saw better throughput over LANE than MPOA shortcut. Some of the possible and real reasons for this are given below:

- The Fore router is capable of routing close to wire speed and causes no bottleneck for single streams
- MPOA must do egress cache lookup when receiving packets over shortcuts. LANE has no egress cache and no cache lookup is required when receiving packets
- Even the longest LLC/SNAP header (12 bytes) is shorter than data link layer header (usually 14 bytes). The incoming packet must be copied to a bigger buffer before the shortcut encapsulation can be replaced with the data link layer header.

However, the benefits of MPOA are obvious when there are more than one high speed streams involved or routers are not capable of routing at wire speed. Just as Section 5.3.6 shows, the use of MPOA boosts the TCP throughput by almost 125Mbit/s when compared to throughput seen by the routed LANE streams.

5.4 Results from delay variation measurements

A number of experiments were done to measure the delay variation, also known as “jitter”, experienced by packets when using LANE or MPOA. The tests were done by using two streams where one stream was used to generate load on the routers while the other stream was measured for jitter experienced by the packets.

The program used for generating the streams was `mgen` which is part of the MGEN-3.0 package. The accompanying program is called `drec`, which receives the flows generated by

`mgen`. With the programs in the MGEN package it is possible to create and measure UDP/IP flows which use predefined amounts of bandwidth have different traffic patterns. [MGEN]

5.4.1 Test methods

Hosts jaarli and basso were used for generating load traffic across the two routers. The traffic was adjusted so that the bandwidth at the ATM level was about 23.3Mbit/s. The traffic pattern was either periodic (constant) or Poisson distributed with peak variation of about 0.5Mbit/s.

Hosts sello and urku were used to generate and receive the test streams. The streams were periodic, using 3.6Mbit/s and 7.2Mbit/s of bandwidth. The respective packet rates were 450 and 900 packets per second and the size of packets was 1000 bytes. These streams were measured for jitter generated by routers and MPOA shortcuts while the network was loaded with the 23.3Mbit/s stream.

Even if the clocks were synchronized with NTP, it was noticed that measuring transmit delay with the time stamps recorded by `mgen` and `drec` was not feasible due to small resolution needed for the task. For this reason it was decided that the jitter detection is done by comparing the time difference of adjacent packets in transmit and receipt.

Fireberd ATM protocol analyzer was used to make sure the Linux network stack does not cause random delays after the packet has been time stamped and sent by `mgen`. The results with the analyzer placed between host urku and the Fore ASX-200BX switch, confirm that the delay caused by the network stack is almost constant. Similar test confirms that this applies also for receipt.

The analyzer was used in the jitter testing to capture packets just before they were received by the destination host. The time difference of received packets was calculated from the analyzer capture file. The time difference of transmitted packets was calculated from the time stamps recorded by `mgen`.

5.4.2 Test results

Common to both test streams, 7.2Mbit/s and 3.6Mbit/s, were the periodic delays seen in transmit. For the 7.2Mbit/s stream the test data revealed that about every ninth packet was sent more than nine milliseconds after the preceding packet. The rest of the packets were sent in average of 76 microseconds after the preceding packet.

This behavior – packets were periodically sent after a long delay – results from the periodic transmission pattern `mgen` that uses when sending packets. For the 3.6Mbit/s stream, about 2/9 of all packets were delayed longer to achieve the 450 packets/second transmit rate.

7.2Mbit/s streams

An example of a 7.2Mbit/s packet flow generated by `mgen` is shown in Figure 5.3. The x-axis shows the packet sequence number and the y-axis shows the time delay (delta time) between this and the preceding packet.

Figure 5.4 shows what the packet stream from Figure 5.3 looks like just before received by `drec` over an MPOA shortcut. The delta times in send and receipt have been plotted together and the plots overlap almost perfectly. Thus the delay variation introduced by MPOA is almost non-existent even if the shortcut VCC crossed two ATM switches.

Figure 5.5 shows the transmit and receive delta time plots for the packets which were routed over both routers while the 23.3Mbit/s Poisson distributed or periodic load was active. The delta times plots of sent and received packets overlap less and the amount of jitter has increased.

As expected, the delay caused by routers is well visible. However, with the periodic load the router behavior differs considerably from the case where the Poisson distributed load was used. Since the testing was done when the network was almost free from other traffic, the probable explanation is some form of flow detection or buffering mechanism in one of the routers. This kind of buffering might be advantageous with non-delay sensitive applications

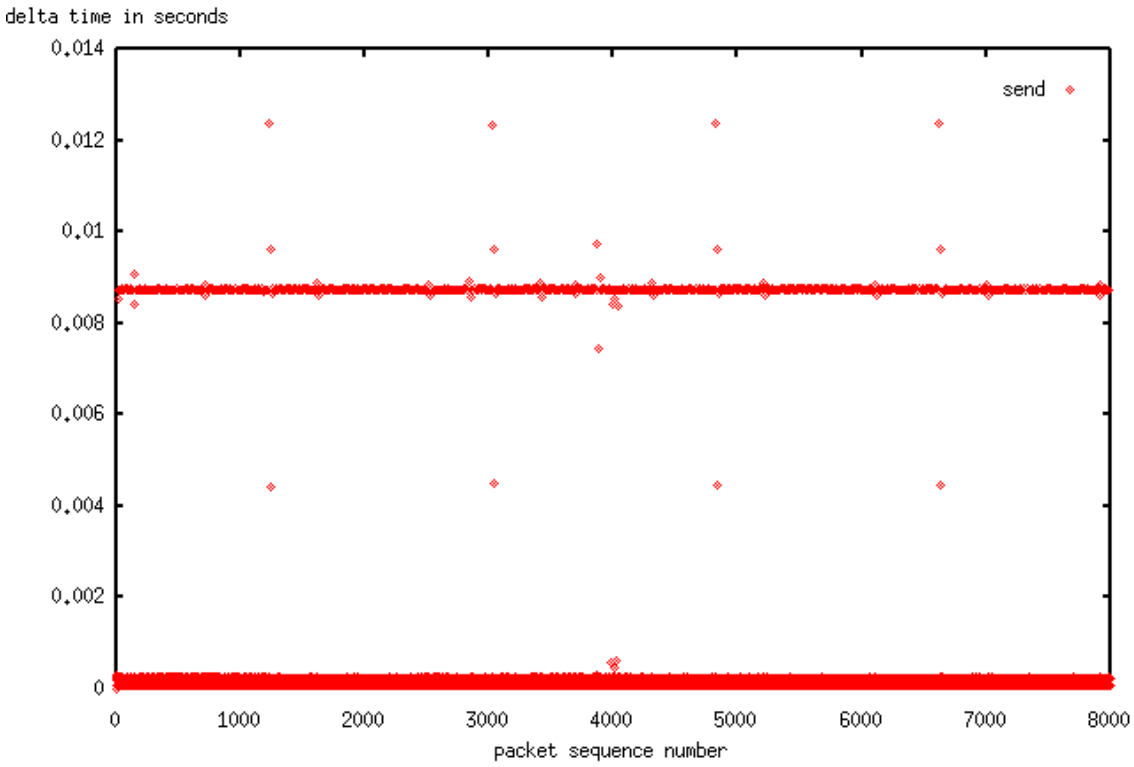


Figure 5.3: 7.2Mbit/s `mgen` packet stream

but disadvantageous with applications that require constant delay from the network.

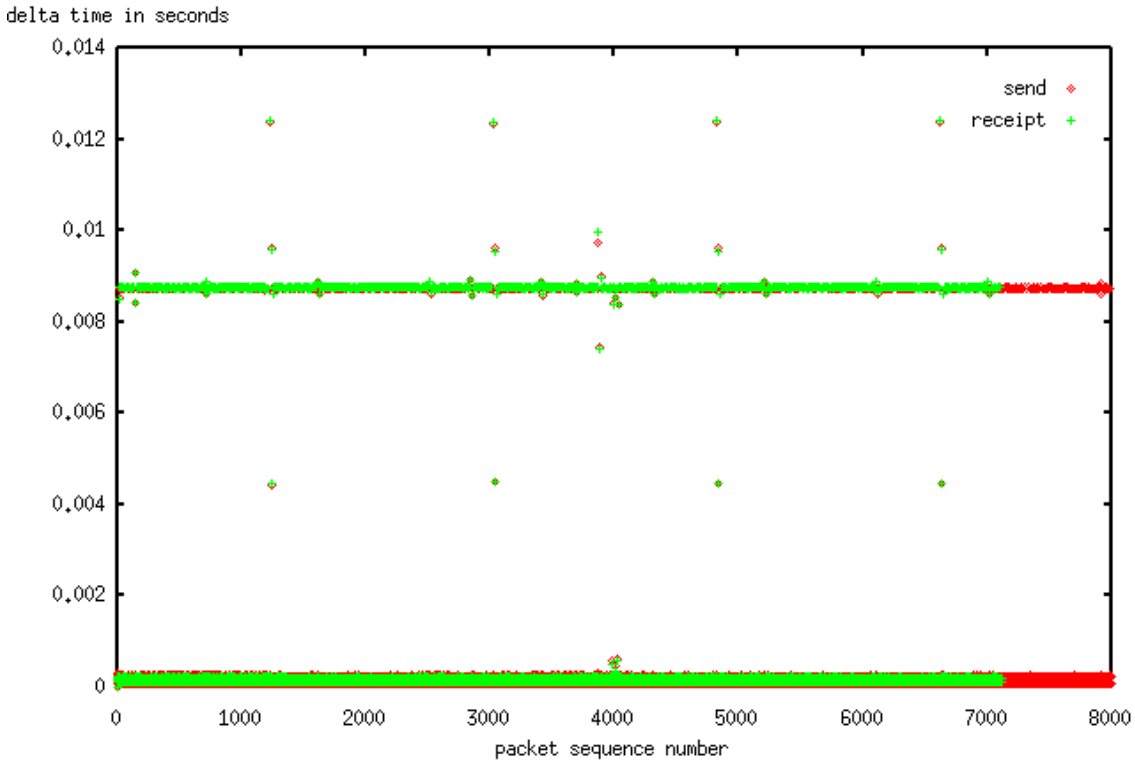
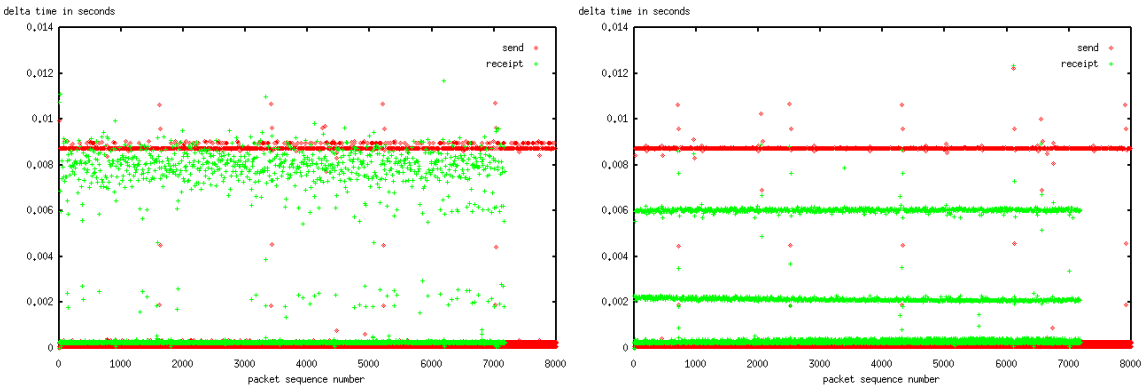


Figure 5.4: 7.2Mbit/s `mgen` packet stream received over a shortcut



(a) Poisson (b) Periodic

Figure 5.5: 7.2Mbit/s routed `mgen` packet streams

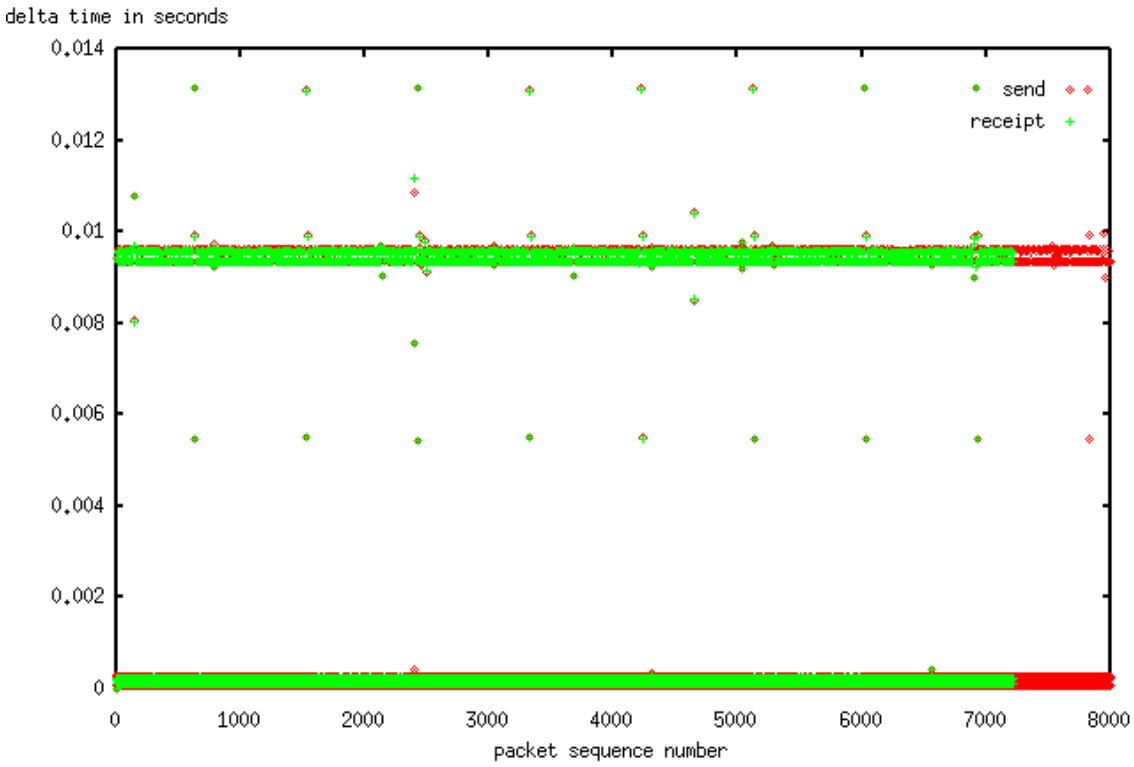
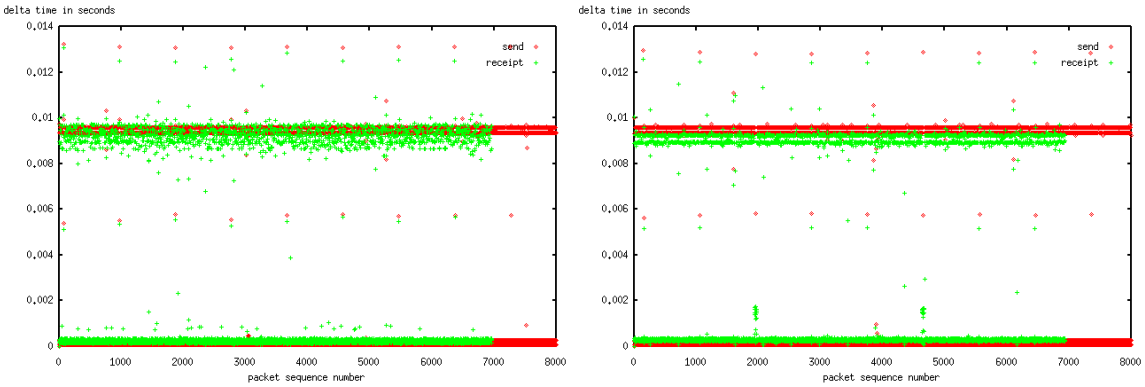


Figure 5.6: 3.6Mbit/s `mgen` packet stream received over a shortcut

3.6Mbit/s streams

Figure 5.6 shows the plot for the 3.6Mbit/s stream which was received over an MPOA shortcut. Like in Figure 5.4, the transmit and received delta times have been plotted together.

Analogous to Figure 5.5, Figure 5.7 shows the transmit and receive delta time plots for the packets which were routed over both routers while the 23.3Mbit/s Poisson distributed or periodic load was active. The amount of jitter is smaller than with 7.2Mbit/s streams, but still visible.



(a) Poisson (b) Periodic

Figure 5.7: 3.6Mbit/s routed `mgen` packet streams

Stream	Load	Mean	Variance	Std deviation	Max	Lost	Out of seq
3.6Mbit/s LANE	Periodic	0.000314	0.000004886	0.002211	0.050111	4	37
3.6Mbit/s LANE	Poisson	0.000162	0.000000041	0.000202	0.005507	2	0
3.6Mbit/s MPOA		0.000035	0.000000000	0.000013	0.000315	0	0
7.2Mbit/s LANE	Periodic	0.000596	0.000000866	0.000930	0.003105	0	5
7.2Mbit/s LANE	Poisson	0.000206	0.000000152	0.000389	0.003438	226	0
7.2Mbit/s MPOA		0.000038	0.000000000	0.000013	0.000304	0	0

Table 5.7: Statistics for 3.6Mbit/s and 7.2Mbit/s streams

Conclusions

Table 5.7 lists some statistical key values for the measured data with counts for lost and out of sequence packets. The values for columns Mean and Max indicate the absolute values for mean and maximum delay variation introduced by LANE and MPOA. For example, the maximum change of delay between transmit and receipt times for two adjacent packets caused by MPOA packet forwarding was 0.000315 seconds for the 3.6Mbit/s stream. In average the routers and use of LANE modified the delay between packets 0.000596 seconds for the 7.2Mbit/s stream while the periodic 23.3Mbit/s load was active.

The tests verify that routing can introduce non-predictable behavior which can cause prob-

lems for applications that are delay sensitive or demand that packets arrive in correct sequence. The direct shortcut VCCs ensure that the delay is as low as the ATM network can provide and the higher layer packets will arrive in the correct order.

With all the routed LANE streams there were a number of anomalities introduced by the routers. Packets were sometimes lost and received out of sequence. The small variance and mean jitter experienced by shortcut streams indicate that the packets were received with almost constant delay. The constant delay makes MPOA advantageous for jitter sensitive applications.

MPOA can also help to extend the usable life time of existing routers. Cisco 4700 serves as an example of a router that does not offer very good routing performance even if the router has a high speed ATM module installed. However, the routing performance does not matter very much if it can function as a route server and let the MPOA clients do the data forwarding.

Chapter 6

Other applications for MPOA

6.1 QoS support for shortcuts

QoS support for shortcuts was implemented as an extension to the ATM Forum MPOA specification. The QoS aware MPOA client supports CBR traffic category VCCs in addition of the default UBR traffic category. The Linux MPC can detect both layer 3 and layer 4 flows. The layer 4 flows are application to application flows for which QoS support gives a possibility that one application can have e.g. 5Mbits CBR VCC reserved for it while the other applications share the bandwidth that is left over from the total capasity.

The version currently included in the Linux-ATM distribution has QoS support only for layer 3 flows. Source code patches have been made available which add QoS support for layer 4 flows, too.

6.1.1 QoS for layer 3, CBR shortcuts for IP flows

Figure 6.1 shows an example of how to create CBR shortcuts instead of default UBR. The `echo` command creates a new QoS entry in the kernel using the `/proc` file system. When the threshold for a new shortcut is exceeded, the QoS entries are checked for a matching IP

```
urku:~# echo add 130.230.54.146 tx=80000,0,1600 rx=tx > /proc/atm/mpc
urku:~#              # generate enough traffic to trigger a shortcut
urku:~# cat /proc/atm/mpc
QoS entries for shortcuts:
IP address
  TX:max_pcr pcr     min_pcr max_cdv max_sdu
  RX:max_pcr pcr     min_pcr max_cdv max_sdu
130.230.54.146
     80000   0       0       0       1600
     80000   0       0       0       1600

Interface 2:

Ingress Entries:
IP address      State      Holding time  Packets fwded  VPI  VCI
130.230.4.3     invalid    1160          0
130.230.54.146  resolved   542           151            0    109

Egress Entries:
Ingress MPC ATM addr
Cache-id        State      Holding time  Packets recvd  Latest IP addr   VPI VCI
47002300000003030001000206002048065 2de00
15              resolved   1143          151            130.230.54.146   0   109
```

Figure 6.1: Examples of layer 3 CBR VCC shortcuts

address. If a matching entry is found, the QoS description associated with the IP address is used when initiating a new shortcut VCC.

In Figure 6.1 the shortcut VCC for IP address `130.230.54.146` uses CBR traffic category with the maximum peak cell rate of 80000 cells/second, default CDV (0) and maximum SDU size of 1600 bytes. The backward values are the same as forward values because `tx=rx` was given on the command line.

The example host `130.230.54.146` was a Sun Ultra1 workstation running Solaris 2.6 equipped with Fore PCA200E ATM network card. The VPI and VCI values indicate that only one SVC was signalled open even though two shortcuts (one to `130.230.54.146` and one from `130.230.54.146`) were established between the two hosts.

6.1.2 QoS for layer 4, CBR shortcuts for application flows

To support QoS for layer 4 (application to application) flows, the flow detection was extended from destination IP address only to monitor also all the identifiers in Figure 6.2.

```
source IP address, destination IP address,
protocol,           (UDP/TCP/ICMP)
source port, destination port (only valid for UDP/TCP)
```

Figure 6.2: Identifiers for application-to-application flows

After the necessary modifications to shortcut detection were complete, a number of test were done. One test case had the following QoS requirements:

- For all UDP shortcuts to any hosts, PCR of 40000 cells per second will be requested.
- For a shortcut to host 130.230.54.209 triggered by telnet (port 23), PCR of 20000 cells per second will be requested
- For all TCP shortcuts to host 130.230.54.209 (not including telnet), PCR of 50000 cells per second will be requested
- For all ICMP shortcuts to any hosts, PCR of 10000 cells per second will be requested

After the QoS requirements were set in the kernel via `/proc/atm/mpc`, enough traffic was created to trigger the shortcut CBR SVCs. The results of experiment are shown in Figure 6.3.

The `/proc/atm/mpc` display has changed slightly from Figure 6.1 to reflect the changes needed for layer 4 flow detection. IP addresses in the QoS entry definitions now have the port number attached to them. A special address `0.0.0.0` is used to indicate any IP address and special port `0` indicates any port. The ingress entries have additional fields `Proto` and `Port` which are needed to differentiate the entries.

```
jaarli:~# cat /proc/atm/mpc
QoS entries for shortcuts:
Src IP address:port   Dst IP address:port  Protocol
  TX:max_pcr pcr      min_pcr max_cdv max_sdu
  RX:max_pcr pcr      min_pcr max_cdv max_sdu
    0.0.0.0:0        0.0.0.0:0          udp
     40000   0        0       0       1600
     40000   0        0       0       1600
    0.0.0.0:0        130.230.54.209:23  tcp
     20000   0        0       0       1600
     20000   0        0       0       1600
    0.0.0.0:0        130.230.54.209:0  tcp
     50000   0        0       0       1600
     50000   0        0       0       1600
    0.0.0.0:0        0.0.0.0:0          icmp
     10000   0        0       0       1600
     10000   0        0       0       1600

Interface 2:

Ingress Entries:
IP address      Proto Port State      Holding time  Packets fwded  VPI  VCI
130.230.54.209  tcp   23   resolved   519           14             0    136
130.230.54.209  icmp  0    resolved   453           6              0    135
130.230.54.209  tcp   0    resolved   417           11             0    134
130.230.54.209  udp   0    resolved   381           12             0    133
130.230.4.5     udp   0    invalid    921           0

Egress Entries:
Ingress MPC ATM addr
Cache-id        State      Holding time  Packets recvd  Latest IP addr  VPI VCI
47002300000003030001000201002 0ea00192d02
5               resolved   973           175            130.230.54.209  0   132
```

Figure 6.3: Examples of layer 4 CBR VCC shortcuts

6.1.3 Problems with MPOAv1.0 and QoS

The MPOA specification recommends the use of ATM Service Category Extension when sending out MPOA Resolution Requests, but other than that there are few words written about QoS. The purpose of ATM Service Category is to give MPCs a possibility to advertise any non-UBR service categories they support. The rest is left implementation dependent. MPOA specification Chapter 4.8.9.1 reads:

> The mechanisms by which an MPOA component decides to initiate a VCC that uses anything other than the UBR service category, are outside the scope of this specification. Different mechanisms are possible such as monitoring the data flow, or monitoring or participation in a resource reservation mechanism like RSVP[RSVP].

The specification does not address e.g. cases where there are multiple VCCs which all map to the same IP address. This can happen when an ingress MPC creates shortcut SVCs to the same IP address for different applications which all require different QoS characteristics from the network. There should also be a way for the ingress MPC to notify the egress MPC that a bi-directional flow across network is about to enter the network and all the return packets that belong to this flow will also need a non-UBR VCC. Otherwise there is the possibility that the other direction of the flow uses an UBR VCC as a return channel. One possible application that needs bi-directional and QoS aware VCCs is IP telephony.

The Linux version of MPOA QoS support is based on the fact that an egress MPC does not care from which VCC the packet arrives, as long as the packet has a tag that can be found from the egress cache. If QoS characteristics are defined for some layer 4 flows, shortcuts for those flows may require multiple VCCs to the same IP address which was not defined in the MPOA specification. Our solution was to issue one MPOA Resolution Request per flow which, except for the first one, end up refreshing the existing entry in the ingress/egress MPSs and egress MPC.

6.2 MPOA shortcuts and WAN links

One of the experiments with the test network had two MPOA enabled LANE clients from Pori joining different ELANs in the test network. The traffic between the clients was traveling all the way to Tampere even though the clients were connected to the same ATM switch in Pori.

Once the traffic between the two clients exceeded the flow threshold, a shortcut was opened and the subsequent packets were forwarded directly across the ATM switch ports in Pori.

Even if the Pori example is very advantageous for MPOA, there might be other similar configurations in which the routed path contains links which have lower bandwidth than the direct path would have. Many examples can be probably found from big corporations and other entities which have geographically wide networks.

6.3 IP telephony over MPOA shortcuts

During the implementation of the MPOA client, the Linux LANE client was extended to support bridging between the ATM and legacy Ethernet parts of the ELAN. Since the bridging function lies on top of the LANE layer, the Linux MPOA client can also initiate and receive MPOA shortcuts on behalf of the Ethernet devices in the normal Ethernet segment. This enables a Linux box to operate as an MPOA enabled Ethernet switch.

The MPOA enabled bridging support was used when making IP telephone calls to Pori School of Technology and Economics. IP phones were attached directly to two Linux PCs (one in Pori and the other in Tampere) which were configured to bridge packets between ATM and an emulated LAN. When a call was made, the MPOA flow threshold was exceeded and the packets were sent and received over an MPOA shortcut.

Given the possibility that Linux MPOA client can also establish shortcuts using the CBR traffic category, it is possible to create CBR VCCs for IP phone calls. This can be done by

using two Linux PCs which either bridge or route the IP phone packets over MPOA enabled LANE interfaces. Even if the reserved bandwidth is low, the CBR connection ensures that it is always available.

Chapter 7

Conclusions

ATM Forum's MPOA is a technique which can help to reduce the network bottlenecks introduced by the current LANE method. MPOA expands ATM SVCs across internetwork layer subnet boundaries making the ATM network topology available for the routed internetwork traffic. MPOA does not replace existing protocols but builds on existing LANE and routing technologies. MPOA is backwards compatible with existing protocols and the migration to MPOA can happen gradually instead of completely reorganizing the existing network and its protocols. MPOA is aimed at campus networks which have problems created by single armed VLAN routing.

The Linux MPOA client implemented during this project is now a part of the popular ATM on Linux package and is available to all those who are interested in running ATM on the Linux operating system.

The performance test with the University's network indicate that MPOA can increase significally the network throughput. Especially networks with relatively low end routers can have very good performance increase by using the routers as MPOA servers and letting the MPOA clients do the internetwork layer packet forwarding.

MPOA can also help the jitter introduced by the routers since the traffic is now switched on

the ATM level instead of routed by the, often overloaded, routers. The amount of jitter is much smaller when the packets are sent over the shortcut SVCs as compared to the amount of jitter introduced by the routers.

Even though the positive results from interoperability, performance and jitter testing demonstrate the benefits of MPOA, it is uncertain how much MPOA will be deployed in production networks. MPOA based products from major vendors have been out for a while now (summer 1999), so the next year will probably show how well MPOA is accepted by the networking professionals.

Bibliography

[Kiiskilä] Marko Kiiskilä. *Implementation of LAN Emulation Over ATM in Linux.* Master's Thesis. Tampere University of Technology. Tampere, Finland October 1996.

[LANE] ATM Forum Technical Committee. *Lan Emulation Over ATM Version 1.0.* January 1997. 4

[Linux-ATM] Werner Almesberger. *ATM on Linux homepage.* http://icawww1.epfl.ch/linux-atm/ 7

[LUNIv2] ATM Forum Technical Committee. *Lan Emulation Over ATM Version 2 – LUNI Specification.* July 1997. 4

[MGEN] Brian Adamson. *The MGEN Toolset.* ftp://manimac.itd.nrl.navy.mil/Pub/MGEN/dist/ 42

[MPOA] ATM Forum Technical Committee. *Multi-Protocol Over ATM – Version 1.0.* July 1997. 8

[Netperf] Rick Jones. *The Public Netperf Homepage.* http://www.netperf.org/ 36

[RFC 1483] J Heinänen. *Multiprotocol Encapsulation over ATM Adaptation Layer 5.* July 1993. 18

[RFC 2332] J. Luciani, D. Katz, D. Piscitello, B. Cole, N. Doraswamy. *NBMA Next Hop Resolution Protocol (NHRP).* April 1998. 15

[802.1Q] IEEE. *IEEE Std 802.1Q – Standards for Local and Metropolitan Area networks: Virtual Bridged Local Area Networks.* July 1998 6

[802.3] IEEE. *IEEE Standards for Local Area Networks: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications.* IEEE, New York, New York. 1985 5

[802.5] IEEE. *IEEE Standards for Local Area Networks: Token Ring Access Method and Physical Layer Specifications.* IEEE, New York, New York. 1985 5