SCOMS demonstration project

The Interworking Trough SCOMS Switch

Concentrating on ATM and ISDN

Markus Mäkelä

TKK/TLM SCOMS Project

Index

Abbreviations

AAL	ATM Adaptation Layer
ABM	Asynchronous Balanced Mode
API	Application Program Interface
ARP	Address Resolution Protocol
ATM	Asynchronous Transfer Mode
B−Channel	Bearer Channel
B−ISDN	Broadband ISDN
BISUP	Broadband ISDN User Part
CATV	Community Antenna Television
CC	Call Control
CC−O	CC Originating Side
CC−T	CC Terminating Side
CLIP	Classical IP Over ATM
CLP	Cell Loss Priority
CORBA	Common Object Request Broker Architecture
CPCS	Common Part Of Convergence Sublayer
D−Channel	Delta Channel
DSS1	Digital Subscribe System 1
EDSS1	Euro ISDN
ETSI	European Telecommunications Standards Institute
FCF	Fabric Control Function
FSR	Frame Synchronized Ring
GFC	Generic Flow Control
HDLC	High−level Data Link Control
HEC	Header Error Control
ILMI	Interim Local Management Interface
IP	Internet Protocol

ISDN	Integrated Services Digital Network
ISUP	ISDN User Part
ITU−T	International Telecommunication Union Telecommunication Standardization Sector
LAPD	Link Access Protocol on the D−channel
MIB	Management Information Base
MPLS	Multi−Protocol Label Switching
MSN	Multiple Subscriber Number
MTP	Message Transfer Part
NNI	Network−Node Interface
OSI	Open Systems Interconnection
PPP	Point to Point Protocol
PSTN	Public Switched Telephone Network
PTI	Payload Type Identifier
PVC	Permanent Virtual Circuit
SAAL	Signaling AAL
SCC	Signaling CC
SCOMS	Software Configurable Multidiscipline Switch
SCOMS/FSR	FSR switch for SCOMS project
SNMP	Simple Network Management Protocol
SS7	Signaling System 7
SSCOP	Service Specific Connection Oriented Protocol
SVC	Switched Virtual Circuit
SW Module	Switch Module
TCP	Transmission Control Protocol
TKK/TLM	Teknillinen Korkeakoulu / TietoLiikenneohjelmistojen ja Multimedian laboratorio
UNI	User Network Interface
UNI−SSCF	UNI−Service Specific Coordination Function
VCI	Virtual Channel Identifier

VPI	Virtual Path Identifier
VTT	Valtion Teknillinen Tutkimuskeskus

1. Introduction

FSR is a switch produced by VTT. It has the ability to interconnect ATM, ISDN ja PSTN networks by signaling. I work with the SCOMS−project, whose job is to build signaling protocols for FSR switch and to make the switch really work in practice too. My employer is TKK/TLM.

My job in the project is to accomplish the practical part. So this document bases mostly in the practical part too. Practical part contains coding, configuring, buying hardware, installing, and a great amount of "tuning" like compiling the kernel many times and things like that. On this special assignment I concentrate on interworking of ISDN and ATM and forget the PSTN.

I created two IP subnets for special use of this demonstration. Subnets are ATM subnet (192.168.6.0) and ISDN subnet (192.168.7.0) and the addresses are 192.168.6.1 and 192.168.6.2 on ATM (Mikki and Hupu) and 192.168.7.1 and 192.168. 7.2 on ISDN (Roope and Aku).

Everything described on this paper is SCOMS/FSR switch related. If all protocols were fully described or ISDN/ATM completely introduced would this paper be like 500 pages which for sure is not the intention.

This whole paper (except 4) describes what I have done during the demonstration project. All the information, configurations, etc. on this paper is found out by me in a way or another. Without one single part of the information given below the project would have failed. I have been there implementing all of the parts more or less. But main idea of my work was to make all the protocols and the hardware work together. All the stuff in this paper concentrates on co−operation of all the parts coded by SCOMS and certain ready parts coded by someone else before.

2. Backgrounds on Why the Switch Was Made

There is large diversity in networks already existing and the diversity has lead to the heterogeneous networks. Of course for different kind of networks different kind of software solutions and services are made. There is no universal transport solution for different networks or it is hardly achieved. Services should be delivered over different kind of networks which should provide seamless integration of networks from perspective of the end users. In that case network elements have to support different kind of networks and call and connection control functions. Interworking capabilities are required to interconnect networks.

The usability of software and services in this case is ensured by using IP over ATM and ISDN on raw data channel after the data channel is opened by interworking of ISDN and ATM networks signaling. This makes most of the Linux software and services reachable.

The interworking between different network might also be done by software without SCOMS switch but then the interworking would be significantly slower and the idea making big networks work together would be wasted. Also, the ATM, ISDN and other cards still of course have to be physically attached somewhere to make the interworking work.

3. Introduction of the hardware used at demonstration project

The goal goal of the demonstration project is to demonstrate transferring data between ISDN and ATM networks trough FSR/SCOMS switch. Hardware combination used on the demo project consists of two ISDN Linux PC’s including one ISDN card each and three ATM Linux PC’s including one ATM card each.

Two of the ATM cards are connected to FSR as its clients and one is installed to a Linux PC which will control the FSR/SCOMS switch. The card installed on controlling Linux PC will be connected to FSR controlling card on FSR port one. All the signaling information is sent trough ATM controlling card to switch. After some huge testing I noted that Efficient´s (eni) ATM cards were the best available so I installed one on Mikki and one on Hupu. These two ATM cards are connected to FSR via normal ATM fiber. This time (too) I noticed that the ATM support is not very good yet. For instance using two ATM cards on same computer makes big problems. I noted that the other ATM card does not work at all with the current ATM Linux software.

On ISDN side Telewell´s passive card (Winbond support) turned out to be a very good choice. Software support is adequate on Linux and the card works fine. On FSR will be installed two VTT’s special production ISDN cards which will be directly connected to our normal passive ISDN cards via normal ISDN connection. Also ISDN phone may be connected to FSR.

Linux PC’s are all 450Mhz PentiumII except the switch control Linux PC, which is 200Mhz Pentium. Switch control Linux PC is only used for controlling the switch.

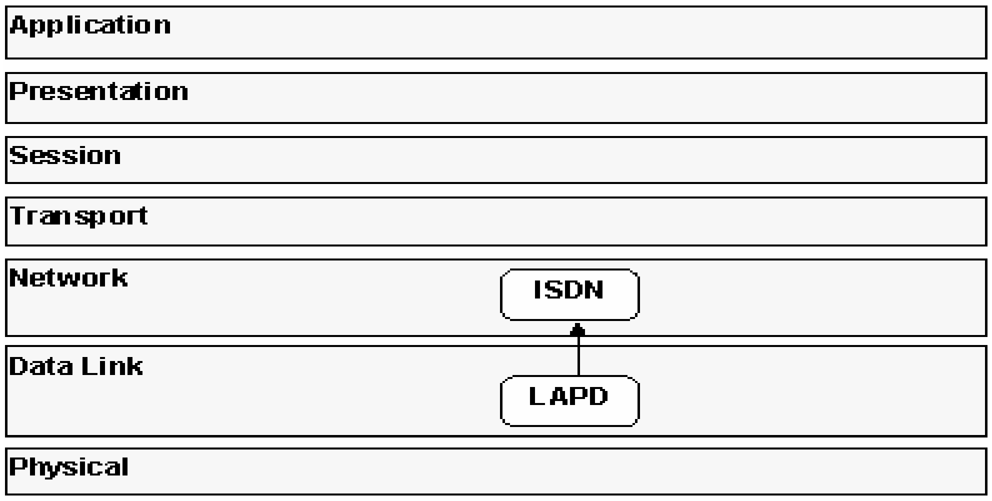
4. Basics of ATM and ISDN networking

To understand the interconnection between ISDN and ATM we must understand the basics of ISDN and ATM networking. The following chapters are dealing with that.

4.1 ISDN

ISDN (Integrated Services Digital Network) is an all digital communications line that allows the transmission of voice, data, video and graphics, at high speeds, over standard communication lines. ISDN provides a single, common interface with which to access digital communications services that are required by varying devices, while remaining transparent to the user. ISDN is not restricted to public telephone networks alone; it may be transmitted via packet switched networks, telex, CATV networks, etc.

The following diagram shows ISDN and LAPD in relation to the OSI model:



PICTURE 1: ISDN and LAPD in relation to the OSI model

There are three logical digital communication channels in ISDN called B, D and H. D channel and two B channels are used in this demonstration. The channels perform the following functions:

B−Channel
: Bearer channel used to transfer raw data. Carries user service information including: digital data, video and voice

D−Channel
: Carries signals and data packets between the user and the network

H−Channel
: Performs the same function as B−Channels, but operates at rates exceeding DS−0 (64 Kbps)

LAPD Link Access Protocol on the D−channel (defined in CCITT Q.920/921). LAPD is a bit oriented protocol on the data link layer of the OSI reference model (layer 2). Its prime function is ensuring the error free transmission of bits on the physical layer (layer 1). LAPD works in the Asynchronous Balanced Mode (ABM). This mode is totally balanced (i.e., no master/slave relationship). Each station may initialize, supervise, recover from errors, and send frames at any time.

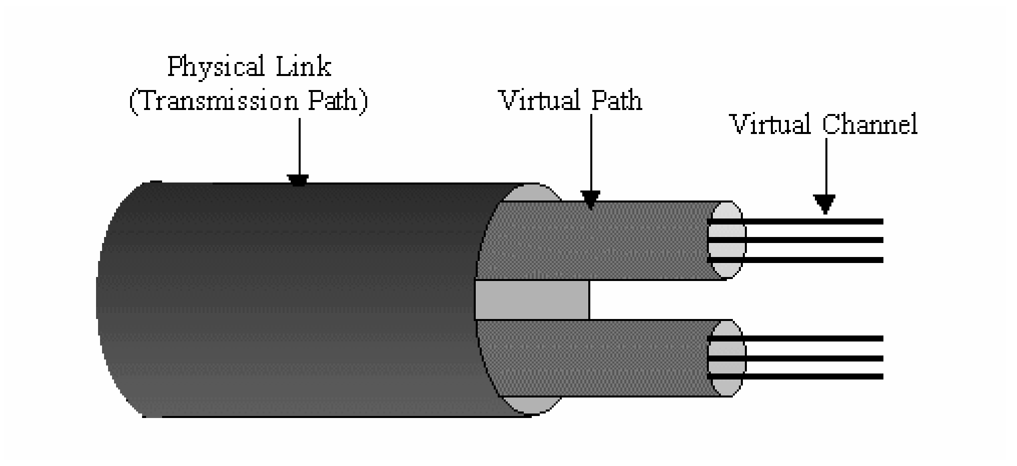
ISDN protocols used on this demonstration on different layers :

Layer 3: DSS1 and EDSS1 (Euro ISDN)
Layer 2: LAPD and HDLC

The structure of ISDN number is described on standard E.164 defined by ITU−T. The basis of the number comes from the more traditional E.163 number which is used on telephone networks. Number consists of country number, network prefix and personal number. The routing is made on the order described below.

4.2 ATM

ATM stands for Asynchronous Transfer Mode and it is a type of network protocol. ATM is the foundation technology for Broadband Integrated Services Digital Network (B−ISDN). ATM uses short, fixed length data units called "cells" (53 bytes per cell). ATM network moves cells with low delay and low delay variation with no fixed timing relationship between the 2 communicating hosts. ATM is designed for all kinds of data i.e. voice, video and data.



PICTURE 2: Relationship between Transmission Path, Virtual Path & Virtual Channel

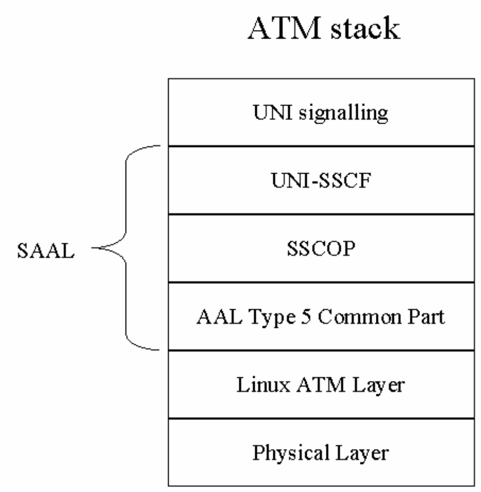
A physical link contains several virtual paths and each virtual path consists of a number of virtual channels (circuits). Bundles of virtual channels are switched via virtual paths.

ATM cells are transmitted via cells switching among virtual paths and virtual channels. A cell will be switched into different virtual channel and path as it moves toward its destination by ATM switches in the network by changing the VPI/VCI value in the cell header. All cells within a virtual channel have the same VPI/VCI value.

Methods used in traffic management to ensure high performance in ATM networks:

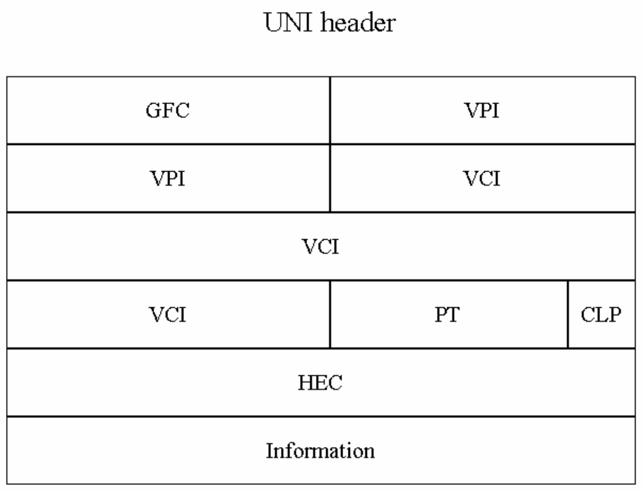
1. Specify traffic "contract" on each virtual channel/path
2. Route cells via virtual path/channel with adequate resources
3. Mark priority (by Cell Loss Priority bit) for discarding cells that violate traffic policy

Transmission on ATM is connection oriented, cell switching, cell discarded with no retransmission, unidirectional or bi−directional, symmetric or asymmetric bandwidth. The signaling stack is pictured below:



PICTURE 3: The ATM/UNI Stack

UNI signaling protocol is roughly analogous to Q.931, and depends upon a lower layer called SAAL (composed of SSCF and SSCOP) for reliable transport of the signaling messages, much as Q.931 depends on Q.921. For those unfamiliar with signaling, the protocol defines a vocabulary of messages and agreed−upon procedures which allow an endpoint to request that the network dynamically set−up and tear−down communication connections with specified characteristics (e.g., bandwidth and "quality−of−service" desired) to specified destinations.



PICTURE 4: The UNI Header

Together, the VPI and VCI comprise the VPCI. These fields represent the routing information within the ATM cell.

ATM systems use both E.164 and NSAP numbers (Network service access point). E.164 is used mainly on public ATM networks and NSAP on private ones. E.164 is also used to make it easier to interconnect with N−ISDN networks. NSAP address structure consists of IDP and DSP parts. IDP defines the domain defined the structure. It is divided in two parts: AFI and IDI. DSP is the address defined by the domain. It is divided in three parts: HO−DSP, ESI and SEL. No more introduction on these protocols is required on this particular paper.

5. Introduction And Configuration of Software Used in Demonstration Project

This chapter describes the software used in demonstration project. Every software used in this project also has exact description how the software is configured to work properly with the SCOMS/FSR switch.

5.1 ISDN Linux software

On ISDN side the only software which turned out to be usable on this project was isdn4linux (ISDN for Linux). Other software are not supported at all or they are very randomly and rarely updated and fixed. The documentation on isdn4linux turned out to be superior too. On these facts the choice between different software was clear: There was no choice. The software itself, information on it and the FAQ are available on many sites, for instance at http://www.isdn4linux.de.

The first step installing ISDN support on Linux PC is to configure the kernel and modules so that ISDN features needed are added. ISDN support is working as a module and all ISDN stuff is under the module. The following configuration on kernel (kernel 2.2.14) works fine with our Telewell ISDN cards:

```
ISDN subsystem --->   <M> ISDN support
            (*)   Support synchronous PPP
            (*)   Support audio via ISDN
            (*)   Support isdn diversion services
            <M> isdnloop support
            <M> HiSax SiemensChipSet driver support
                        (*)     HiSax Support for EURO/DSS1
                        (*)     HiSax Support for Winbond W6692 based cards
```

Modules configuration file is located at following place:

```
/etc/conf.modules
```

After configuring the kernel and the modules they are to be compiled by the following way:

```
/usr/src/linux/> make menuconfig
/usr/src/linux/> make dep clean
/usr/src/linux/> make bzImage
/usr/src/linux/> make modules
/usr/src/linux/> make modules_install
```

Now that the kernel is configured we can concentrate on configuring the ISDN software for our needs. Of course for first the software has to be installed. It is done by the following way:

```
rpm −Uvh isdn4k−utils−x.xx−xxx.i386.rpm
```

Linux RedHat6.1 is used by SCOMS project and the best rpm packet for it is isdn4k−utils−3.0−5.i386.rpm. After the kernel is configured and rpm packet installed we only have to make some scripts and settings. I wrote a script that does all the things needed to rise ISDN even if the Linux PC was booted before executing the script. The scripts are located at:

```
roope:/etc/rc.d/rawip−start.roope
aku:/etc/rc.d/rawip−start.aku
```

Isdnctrl is the command used on most configuring and it was included in isdn4linux packet. With /usr/sbin/isdnctrl list you can check whether the configurations are what you wanted.

```
#!/bin/sh
# Modules install
# Markus Mäkelä 13.06.2000
# ISDN subnet:
# Roope's         IP:                    192.168.7.1
# Aku's           IP:                    192.168.7.2

# MSN numbers.
# OWNMSN is the number you are calling from
# MSNOUT is the number you are calling to
OWNMSN=4125384
MSNOUT=054125385

# Type is the card identification number. Protocol is the protocol used (DSS1).
# IRQ is cards interrupt, io it's IO address and id is the binding identification.
# Slhc, isdn and ppp are the modules loaded.
modprobe hisax type=36 protocol=2 irq=10 io=0xec00 id=hisaxID
modprobe slhc
modprobe isdn
modprobe ppp

# Adding new ISDN interface called isdn0 to the kernel
/usr/sbin/isdnctrl addif isdn0

# Setting dialmode auto
/usr/sbin/isdnctrl dialmode isdn0 auto

# Setting incoming and outcoming numbers for calls on isdn0
# "*" accepts all the incoming numbers.
/usr/sbin/isdnctrl addphone isdn0 out $MSNOUT
/usr/sbin/isdnctrl addphone isdn0 in "*"

# EuroISDN (DSS1) own number
/usr/sbin/isdnctrl eaz isdn0 $OWNMSN

# Layer-2/3 protocols for interface name
# HDLC and LAPD are much the same so using HDLC
/usr/sbin/isdnctrl l2_prot isdn0 hdlc
/usr/sbin/isdnctrl l3_prot isdn0 trans

# Setting security options off
/usr/sbin/isdnctrl secure isdn0 off

# RawIP as encapsulation. Direct handshaking!
/usr/sbin/isdnctrl encap isdn0 rawip

# Timeouts
# /usr/sbin/isdnctrl huptimeout isdn0 300

# The callback delay
/usr/sbin/isdnctrl cbdelay isdn0 0

# Binding the card to interface
/usr/sbin/isdnctrl bind isdn0 hisaxID,1

# Ifconfig sets up the kernel-resident network
ifconfig isdn0 192.168.7.1 pointopoint 192.168.7.2 netmask 255.255.255.0 up

# Let's set  the route
```

```
/usr/sbin/route add -host 192.168.7.2 isdn0
```

The most positive thing using RawIP encapsulation is that RawIP does without the use of a protocol such as X.75, HDLC or PPP. So we do not have to use pppd or ipppd demons which would have made big problems on ATM side since they send their frames with the data. No demons are used at ISDN side at all! When using RawIP encapsulation are TCP/IP packets directly exchanged. This fact makes it possible to transfer data between ISDN and ATM trough the switch using normal IP addresses. Other positive things about RawIP are: No handshaking (faster connections), authorization by Caller ID (fast, safe, no password) and fixed IP address (a broken connection can be continued by redialing).

5.2 ATM Linux software

When finding out a proper software on ATM side the situation was much the same as on ISDN side. There was just one good software available, ATMonLinux. ATMonLinux and valuable information and help on it can be found for instance at: http://icawww1.epfl.ch/linux−atm/ . On ATM side of the demonstration too the first step is to configure the kernel. The following kernel configuration works fine with our Efficient (eni) ATM cards, kernel (2.2.14), Linux Redhat6.1 and ATMonLinux version 0.59.

```
[*] Asynchronous Transfer Mode (ATM, EXPERIMENTAL)
        [*]   Use "new" skb structure
        [*]   Classical IP over ATM
        [*]     Do NOT send ICMP if no neighbour
        <M> LAN Emulation (LANE) support
        <M> Multi-Protocol Over ATM (MPOA) support
```

After configuring the kernel and the modules they are to be compiled by the following way:

```
/usr/src/linux/> make menuconfig
/usr/src/linux/> make dep clean
/usr/src/linux/> make bzImage
/usr/src/linux/> make modules
/usr/src/linux/> make modules_install
```

There is one thing to mention about kernels and glibc versions: Our UNI3.1 and ATMonLinux's Linux UNI3.1 (running as atmsigd = ATM signaling demon) could not communicate if the kernel version was 2.2.14 but glibc was version 2.1.2−11 which came with the kernel 2.2.14 packet. The glibc version had to be updated to 2.1.3−15 to make signaling work properly. The version number of ATMonLinux version is as low as is because if newer version would have been used a kernel update should have been done. Version 0.59 works fine so there no need to update everything just because of one software.

The ATMonLinux software is easy to install. Just download the software from internet, for instance from ftp://icaftp.epfl.ch/pub/linux/atm/dist/atm−0.59.tar.gz and normally extract it by shell command in the atm directory:

```
/usr/src/atm/> tar xfz atm−0.59.tar.gz
```

On /usr/src/linux directory then exract the ATM related patches. If ATM signaling demons work fine this step may be skipped.

```
/usr/src/linux/> patch −s  −p1 <  /usr/src/atm.patch
```

Now compile /usr/src/atm/ and all the directories under it if makefile under atm/ does not make it itself and you have an ATM support on your Linux PC. Now we just have to do some scripting. The ATM interface for the ATM cards should be set 0. It is set atm0 automatically if just on ATM card is installed. But because atm daemon (atmsigd) and linux kernel does not seem to be able to handle 2 atm cards on same Linux PC there should be just one ATM card on one Linux PC.

Now that we have the software we can start running it. First we should configure the ATM address used by our Linux PC. ATM addresses should be configured in /etc/hosts.atm file and should look like this:

```
47.0005.80FFE1000000F20F4A15.0020EA003D62.00    mikki.tcm-atm
47.0005.80FFE1000000F20F4A15.0020EA002AEF.00    hupu.tcm-atm
```

After address configuration the address used on your Linux PC must be told to your ATM software so it can be used later:

```
/usr/local/sbin/> ./atmaddr -a mikki.tcm-atm
```

At this time we should start running our switch signaling UNI. There is switch control software (sw) included in our UNI so starting it now makes it possible to connect trough switch on later parts. Our UNI also makes communication between two atmsigd’s on different Linux PC’s possible by handling the signaling between them trough the switch. As seen UNI loads its ORB configuration from file orbacus.cfg.

```
~/scoms/bin/iut/uni> ./uniiut -ORBconfig orbacus.cfg
```

UNI and switch related configuration files are found at:

```
~/scoms/bin/iut/uni/address.cfg
~/scoms/bin/iut/uni/switch.cfg
```

The most important demon on Linux ATM is atmsigd which communicates with our switch related UNI3.1. ATM signaling demon is started the following way. The option −d means debugging and 0.0.50 are interface number and last are VPI (Virtual Path Identifier) and VCI (Virtual Channel Identifier). VPI and VCI are explained in chapter 4.

```
/usr/local/sbin/> ./atmsigd –d 0.0.50
```

To make it possible for ATM and ISDN to contact with IP addresses must Classical IP over ATM (CLIP, defined in RFC1577) be configured. It is done by starting the ATM ARP demon and defining the ATM ARP options:

```
/usr/local/sbin/> ./atmarpd –d
```

Next we create atm0 interface with help of ATM ARP:

```
/usr/local/sbin/> atmarp –c atm0
```

And ifconfig is used to raise the interface just created for certain IP address:

```
ifconfig atm0 192.168.6.1 netmask 255.255.255.0 up
```

Next we have to configure atmarp addresses. The first one configures Mikki (192.168.6.1) as ARP server and the second one just tells Mikki there´s Hupu on IP 192.168.6.2 and ATM address 47.0005.80FFE1000000F20F4A15.0020EA002AEF.00.

```
hupu:/usr/local/sbin> ./atmarp –s 192.168.6.1
47.0005.80FFE1000000F20F4A15.0020EA003D62.00 arpsrv
mikki:/usr/local/sbin> ./atmarp –s 192.168.6.2
47.0005.80FFE1000000F20F4A15.0020EA002AEF.00
```

Usable ATM ARP information can be found in following files:

```
/var/run/atmarpd.table
/proc/atm/arp
```

5.3 SCOMS and Switch Software

The SCOMS and switch softwares handle the signaling trough the switch and the switch configuration. The handling of signallation, the switch control and what happens when the connection is taken is explained later on chapter 4: The Principles of Connecting ATM and ISDN trough the SCOMS switch. The protocols themselves are not configured anywhere after coding because it is not needed and the protocols fulfil the specifications requirements given by ITU−T and ETSI and ATM Forum. The SCOMS protocols used handle the signaling between ATM and ISDN on any cases. Addresses and switch must be configured though. Linux ATM (ATMonLinux) and ISDN (isdn4linux) softwares are configured before in chapters 3.1 ISDN Linux Software and 3.2 ATM Linux Software. ATM and ISDN addresses used with the switch are configured in ~/scoms/bin/iut/uni/address.cfg and should look like this:

```
# ~/scoms/bin/iut/uni/address.cfg
# Manually registered addresses
# Configured by Markus Mäkelä @ SCOMS
# Switch Control on       port 1
# Mikki on                port 2
# Hupu on                 port 3
# Roope on                port 4
# Aku on                  port 5

BEGIN ADDRESSES
# port   link    address
  2      1       47.0005.80FFE1000000F20F4A15.0020EA003D62.00
  3      1       47.0005.80FFE1000000F20F4A15.0020EA002AEF.00
  4      1       47.0005.80FFE1000000F20F4A15.0020EA002AF8.00
  5      1       47.0005.80FFE1000000F20F4A15.0020EA002AF5.00

END
```

```
# Mikki          (ATM)
# Hupu           (ATM)
# Roope          (ISDN)
# Aku            (ISDN)
```

Ports are switch ports. ATM cards are on ports 1, 2 and 3 and ISDN cards are on ports 4 and 5. Switch control is on port 1, Mikki is on port 2, Hupu on port 3, Roope on port 4 and Aku on port 5. Certain addresses are bound to certain ports. So if an address is called the switch knows to which port it makes the connection and recognizes the card type (ISDN/ATM) on that port.

The switch itself is configured in file ~/scoms/bin/iut/uni/switch.cfg and it should look like this:

```
#
# Switch configuration file
# Configured by Markus Mäkelä @ SCOMS
#

# Control port indicates controller's (PC) control port (default 0)
CONTROL_PORT 0

# Fabric control port indicates fabric (switch hardware) control port
# where controller is connected to (default 1)
FABRIC_CONTROL_PORT 2

#  VPI/VCI values (default 0/5)
SIGNALING_VPI 0
SIGNALING_VCI 50

# Signaling VPI/VCI values (default 0/15)
API_VPI 0
API_VCI 64

# Minimum and maximum values for VPI/VCI values of data channel.
# This VCI range can be restricted with set methods in sw module
# (e.g. called by switching fabric), but not extended.
# Only VPI = 0 is supported in this version, following VPI values
# have no effect (default VCI range 32–65535).
MIN_VPI 0
MAX_VPI 0
MIN_VCI 32
MAX_VCI 65535

# Base values for special VCI reservation
# (default base for signaling vci is 40 and for ILMI vci 60)
SIGNALING_VCI_BASE 40
```

```
# Select used modules (default values for all is 'used' (1))
ILMI_USED 0
SS7_USED 0

# Select fabric control function (GSMP, API, NOT_USED)
FABRIC API

# Routing (SIMPLE, TRS)
# Default routing is SIMPLE routing.
ROUTING SIMPLE

# MANDATORY config variable POINT_CODE (no default value available)
# This SS7 point code is used also for CORBA Nameservice to identify
# a switch instance
POINT_CODE 1

# Network prefix used in address registration
# (no default value set (empty string))
NET_PREFIX 47.0005.80FFE1000000F20F4A15

# Debug trace NULL, COUT, FILE, CORBA (default is off (NULL))
# Note: Only COUT is noticed, other values are interpreted like NULL
# Debug levels are 1,2,3 (default is most quiet (1))
DEBUG_TRACE COUT
DEBUG_LEVEL 3

# Ports to be installed. Lists port numbers (uses FSR155 port type).
# NOTE: If FABRIC is NOT_USED, this variable has no effect, because
# switching hardware installs ports automatically.

BEGIN PORTS
# type port numbers
  ATM 1 2 3
  ISDN 4 5
END

# Links to be installed. List link type, port number and link number.
# NNI links need also point codes.
# NOTE: If corresponding ports are not installed, exception handling
# prevents to install links.

BEGIN LINKS
# port link type (point code)
# ATM (UNI31, UNI40, DSS2, BISUP)
  2 1 UNI31
  3 1 UNI31
  4 1 DSS1
  5 1 DSS1
END

BEGIN SS7_USERS
  TCAP
  BISUP
  ISUP
END
```

```
BEGIN MTP3_ROUTES
# start point code, end point code, port number, link number
  2 2 2 1
  3 3 3 1
  4 4 4 1
  5 5 5 1
END

# If ILMI_USED is set (1), this variable has no effect, because
ADDRESS_FILE_NAME address.cfg
```

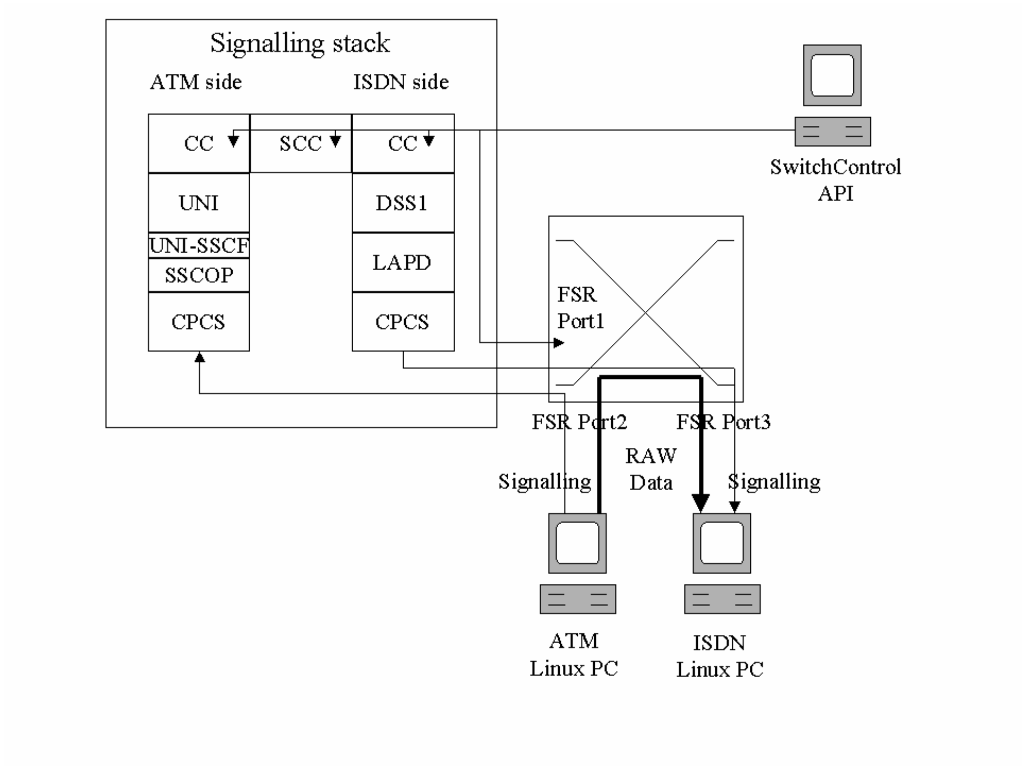
Each part of the configuration file is explained above in corresponding place.

6. Connecting ATM and ISDN Trough the SCOMS Switch

On common ISDN connection and in our system there are two Bearer (B) and one signaling (D) channels. Protocols used on ISDN B/D channels on our demonstration are on D Channel: EDSS1/DSS1 and HDLC/LAPD. Linux EDSS (Euro ISDN) corresponds DSS1 (specifications: ITU−T Q.931) and Linux HDLC corresponds LAPD (specifications: ITU−T Q.921).

B channel is for transferring raw data. On ATM there are no B or D channels but the data transmission is handled trough the Virtual Channels and signaling is handled by signaling to certain VPI/VCI. Two virtual ATM channels with own VPI/VCI values both make much the same as B channels on ISDN side. When the signaling is done is the data transferred as is through the switch via data channel. For instance MP3 sent from ISDN arrives as sent at ATM side. The connections may be taken by IP address. These facts make this system very usable.

The following picture shows the main idea of interworking of ISDN and ATM trough the FSR/SCOMS switch. The explanation of what is happening in the picture follows the pic.



PICTURE 5: Interworking of ATM and ISDN, point of view 1.

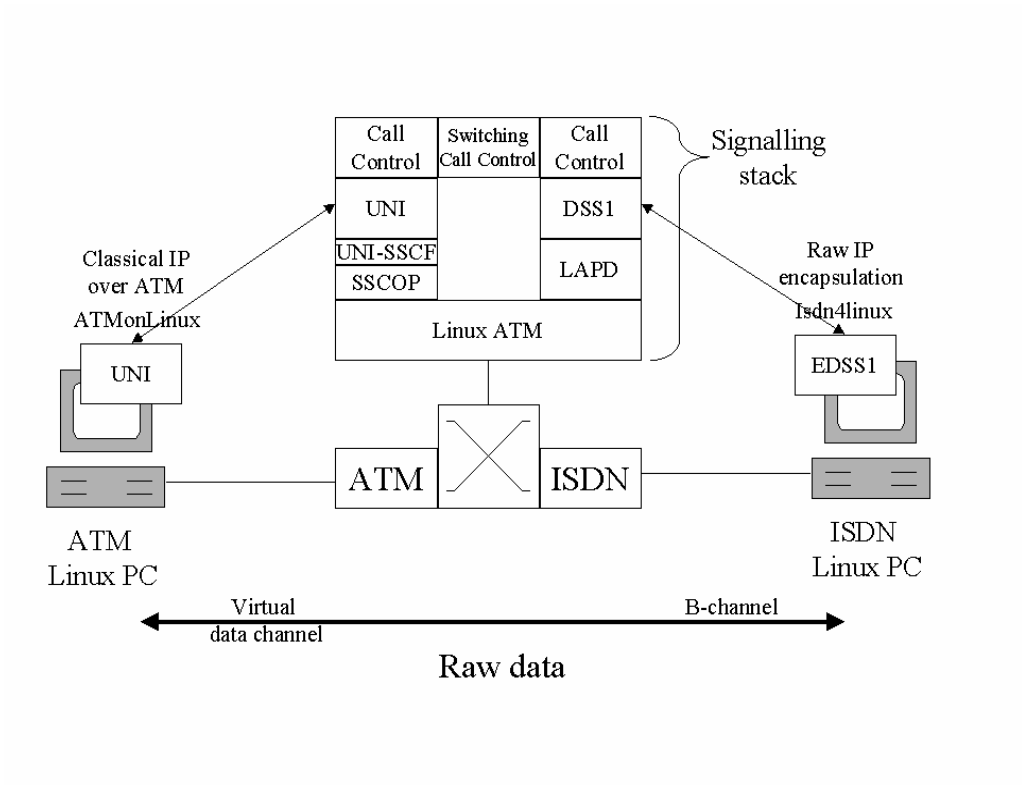
ATM Linux PC (from ip 192.168.6.1) is trying to contact the ISDN Linux PC with IP address, say 192.168.7.1. Linux PC ATM card is connected to one of the ATM ports on switch. ATM card sends signaling information to switch by VPI/VCI 0.50. Switch is controlled by switch control Linux PC which is running API (Application Program Interface). API is the software controlling the switch. API runs on a Linux PC with ATM card and it controls the switch by signaling via permanent VPI/VCI. Right when the API is started are the protocol stacks established from CPCS to UNI on ATM side and from CPCS to LAPD on ISDN side. When connection establishment request is received from the incoming link (UNI at this case) sends the coordination protocol a SETUP message to call control (CC). After the setup arrives at CC it recognizes itself as originating side (CC−O). It reserves the resources from incoming link and demands a route to be established with help of switch and switch (sw) software (includes FCF). SwSwitch has an STL map which includes link interworking possibilities. The state of SCC is controlled by swSwitch. Different states of SCC contain different connection possibilities. Routing returns the identification to an outgoing link (DSS1) which CC−O uses to to send an establishment command to an outgoing link trough SCC. SCC part switches the signaling to a form understandable on ISDN side. When establishment command arrives at ISDN side an ISDN stack is established and the whole signaling stack is now established.

Now signaling reaches the with help of routing made by switch Linux PC ISDN card and is in understandable form to it. The signaling route is now established and the data channel will be opened to transfer data right trough the switch. Since the data channel is opened it stays permanent until closed by signaling. Data transferred trough the switch is raw and stays the unchanged on its way to the Linux ISDN PC. The connection just made acts like a normal IP connection.

ISDN side's IP address (at this case 192.168.7.1) and ISDN cards MSN number are configured with help of isdnctrl software to isdn0 interface. The same MSN number just like all other MSN numbers and ATM addresses (on atm0) are configured in address.cfg file with corresponding FSR/SCOMS switch port numbers. Classical IP Over ATM addresses are configured as explained before. Signaling channel must be opened before connecting with IP. It is done from ATM Linux PC by calling to a certain ATM address which is bound to a known ISDN port. When calling the certain number the switch and its control software knows between witch cards on switch to open the connection. The first of two connected cards of course is the card that makes the call. Switch and address configuration files are explained better on chapter 3: Introduction And Configuration of Software Used at Demonstration Project.

After the connection is opened by calling to a certain ATM address or MSN number may IP features be fully used.

The same thing is explained from a little different point of view using a little different combination in the following picture. The explanation of what is happening in the picture follows the pic.



PICTURE 6: Interworking of ISDN and ATM, point of view 2.

On the left side there is Linux ATM PC running ATM signaling demon called atmsigd and it handles the Linux UNI signaling. On the same Linux PC we run UNI coded in SCOMS software (specifications made by ATM forum). Our UNI and Linux UNI signal trough the lower ATM layers UNI−SSCF and SSCOP. Linux UNI and our UNI both are able to handle the signaling trough UNI−SSCF and SSCOP The signaling on ISDN (right) side is handled much the same away. Isdn4linux EDSS1 (Euro ISDN) contacts our DSS1 (specifications made by ITU−T) trough the lower layer LAPD. The signaling trough LAPD is handled by starting HDLC layer 2 protocol on ISDN Linux PC. HDLC signals with LAPD so that signaling between DSS1 and EDSS1 is possible. Switch and SCC handle the interconnection of ISDN and ATM just like described in the chapter before. Both ATM and ISDN sides have their own subnets. When the data channel is opened by signaling may the both sides connect to each other's IP addresses and the network just created works just like a normal IP network and any IP software can be used such as MP3 server being able to be connected from side to side.

As seen on the picture the switch is controlled by ATM and ATM network is designed to work as a backbone of the networks connected to switch. The connections taken via switch base on ATM SVC connections. That is why the switch control Linux PC is connected to ATM port too. Switched Virtual Circuits (SVCs) are connections that are established dynamically, and are then torn down when the connection is no longer needed. When using SVCs, a host must pass information to the ATM switch, declaring its intent to set up a connection with another host. The term for connection setup is signaling. The ATM protocol used between a host and a switch is the user−network−interface (UNI) signaling standard.

7. Maybe the Most Important Part of Project: SW Module

The SW module provides the framework for ATM switch software. It provides facilities to access the cards, physical interfaces, logical signaling interfaces and other resources in the switch. The SW module implements a main program for ATM switch software and it is implemented to delegate operations to proper modules. We have a runtime access to configure switch parameter and resources via CORBA interface which is very clearly seen in PICTURE 8: The Structure of the SCOMS stack too.

I myself have coded a big part of SW switch control module. In the following chapters I will describe the architecture and some implementation details about the SW module.

The PICTURE 7 shows the switch architecture. Management protocol implementations, SNMP and ILMI and configuring interface implementation are CORBA adapters to support access to ATM MIB and other system variables. Routing resources are implemented in other module and it is distributed also with CORBA. The switch class has different prototypes and common objects as attributes, e.g. SS7 stack, and cloneable prototypes of signaling protocols. The architecture of the switch framework consists of following classes or collections of classes: system, switch, routing, card, physical interface, logical signaling interface, subscription, call, virtual path and virtual channel. More about these entities is described further on further chapters.

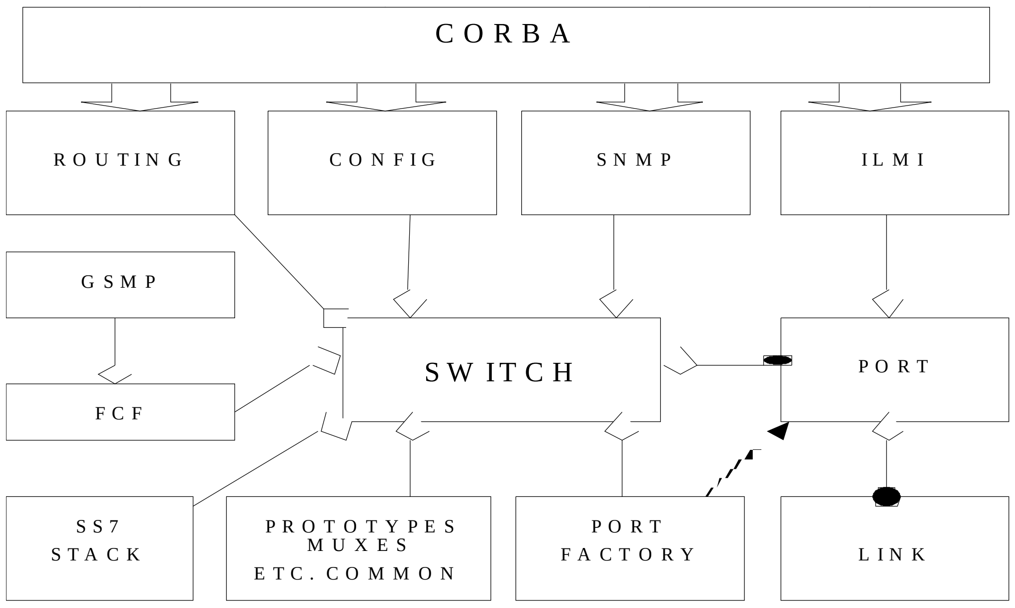


Figure 1. Switch architecture.

PICTURE 7: SW

System provides a singleton for a global access point for the switch framework. The system can contain several switches that can be accessed through the system. The system may be implemented using several separate singletons if needed.

Switch is an object that provides an access point to a switch and resources of the switch. The interface of the switch provides access to line interface cards, physical interfaces in those cards, logical signaling interfaces in those physical interfaces and virtual paths and channels in physical interfaces in addition to other physical or logical resources. Routing is an object that makes routing decisions based on the information given from the switch.

The routing contains a static route database. The switch informs routing about dynamic changes in routes such as congestion, blocking and route failures.

Card is an object that models a physical line interface card attached to the switch. One card can contain one or several physical interfaces. Activation or deactivation of a card activates or deactivates physical interfaces contained in the card. Other operations performed on the card can be propagated to physical interfaces as well.

Physical interface (port) is an object that models a physical network interface that is connected to the switching fabric. The physical interface has several properties that describe characteristics of the interface such as line speed and the number of virtual paths and channels supported.

Logical signaling interface (link) is an object that models the signaling capabilities of the whole physical interface or one logical channel inside the physical interface. The logical signaling channel supports one of various signaling protocol stacks such as UNI signaling stack. The UNI signaling stack contains ATM, CPCS, UNI−SSCF and UNI 3.1. I coded links to ISDN and fixed ATM links.

Subscription is an object that models subscribed services in local interfaces. It contains a user service profile.

Call is an object that models the ongoing call and resources associated to it. The call can be SVC or PVC.

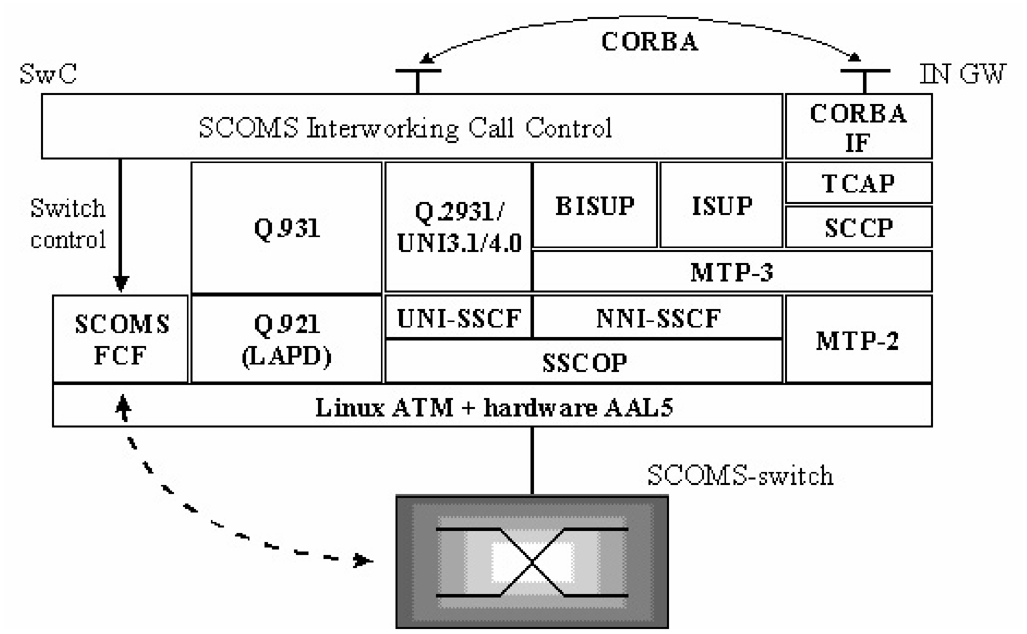
Virtual path is an object that models virtual path connection in a physical interface. This object is used for unallocated and allocated virtual paths. Virtual paths can be switched if that is supported by the hardware.

Virtual channel is an object that models virtual channel connection in a physical interface. This object is used for unallocated and allocated virtual channels. Virtual channels can be switched if that is supported by the hardware.

A port factory creates ports using registered port prototypes (identified with a string). A fabric control function (FCF) is module that includes a generic interface, which hides the concrete switching protocol or API, e.g. ATM FCF implementation uses API to make fabric connections in a switching hardware.

8. The Structure of the SCOMS Stack

To understand more about SCOMS software and its future must the whole SCOMS stack be introduced. In the following picture there are all the protocols used at SCOMS project in stack.



PICTURE 8: The Structure of the SCOMS Stack

In the picture, the ISDN and ATM stacks described better before are very clearly seen. In the future the stack will change so that MTP−2 reaches the bottom of MTP−3 under ISUP. We can here also see the importance of Linux ATM when communicating with the SCOMS switch. The Call Control’s meaning as a highest level protocol is now also clearly seen. The signaling protocols UNI, DSS1, BISUP and ISUP are right under SCOMS interworking Call Control. The protocols on the right side are for future to be used with intelligent networks.

9. The Future

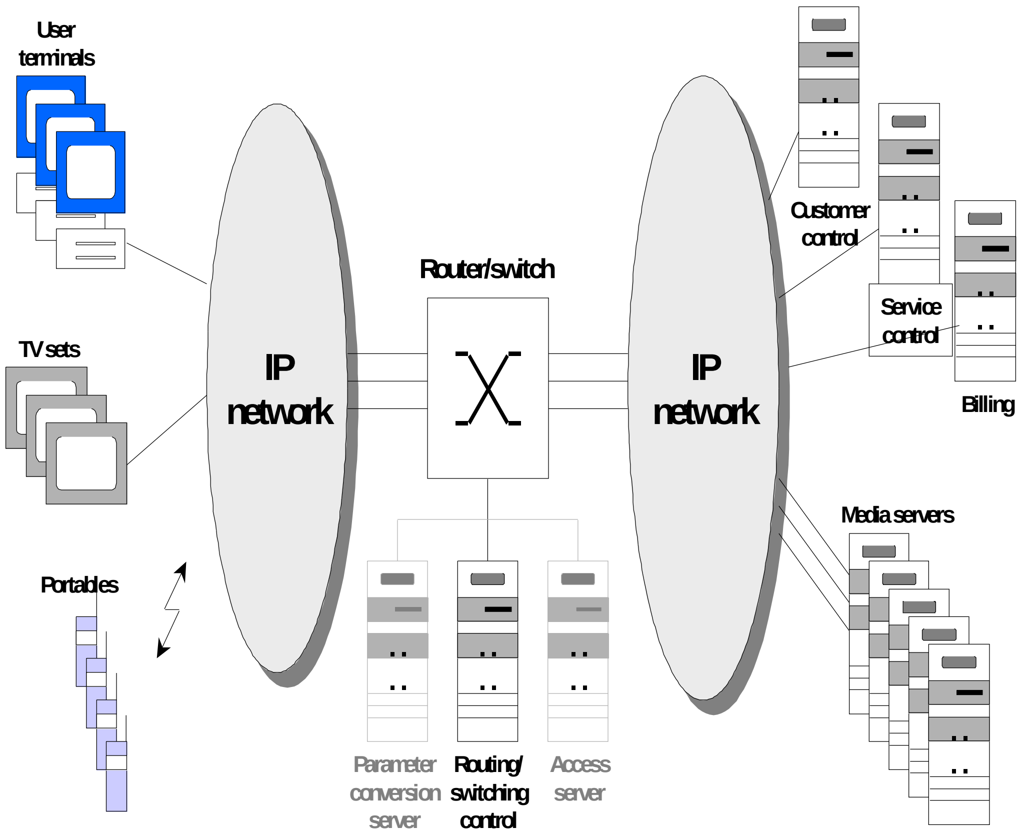
At the moment only ATM and ISDN interworking works on SCOMS/FSR switch but in the future, more different networks will be available to be connected to SCOMS/FSR switch. The future of the switch lies on that fact that it is constantly updated. Next step will be connecting Ethernet IP and MPLS IP networks to the switch. MPLS is a protocol that uses ATM hardware in physical transmission and IP technology in routing. So it completely acts like an IP network which once again makes big things possible.

With the hardware, Linux and IP knowledge gained during the SCOMS project it should be relatively easy to start coding new network protocols to be used by switch. Making the new hardware work with Linux PC's should not be a problem either. Of course, when making big things big problems may occur but the point is that now that the backbone is set and the biggest problems are solved all the problems that occur are solvable.

All the connection oriented networking protocols have the same main idea of signaling and data transfer, there is always a signaling channel in a form or another and a data channel in a form or another. The switch has been designed by that fact also. That fact makes it even easier to code new protocols and attach new networks to the switch.

The word of the future of all kind of networks is service so on service must we concentrate too. The visions of future include Media Servers, Service Control, Billing and Customer Control on MPLS IP networks which will be directly connected to the switch and the services that way brought to other networks connected to switch. About Ethernet IP networks we have a vision too. Normal user terminals will off course be connected to switch via Ethernet but also many other kind of things like TV sets and portables for instance.

The following picture describes two of the future networks to be connected to switch. On left side is ethernet IP and on the right is MPLS based IP network.



PICTURE 8. Two future networks to be connected to switch: Ethernet on the left and MPLS on the right side.

In the future MPLS, ATM and Ethernet networks will all use Classical IP features to make IP access over the switch even easier. I coded the CLIP OVOPS++ interface to MPLS, ATM and ethernet. The interface bases on ATMonLinux CLIP.

There are many visions in the future and many will be implemented but the next sentence describes my feelings about the future quite well. The world of visions is infinite on possibilities. Sadly, the world of reality is not infinite on funding.

10.Problems I Met

One of the problems was that UNI3.1 specified by ATM Forum did not run with glibc−2.1.2−11. Our UNI3.1 and ATMonLinux’s Linux UNI3.1 (running as atmsigd = ATM signaling demon) could not communicate if the kernel version was 2.2.14 but glibc was version 2.1.2−11 which came with the kernel 2.2.14 packet. The glibc version had to be updated to 2.1.3−15 to make signaling work properly. The version number of ATMonLinux version is as low as is because if newer version would have been used the kernel update should have been done. Version 0.59 works fine so is there no need to update everything just because of one software.

One big problem was making the ATM cards work. It was finally found out that ATM drivers and ATM software on Linux still have many bugs. You just cannot add two ATM cards on same machine unless you fix the kernel sources. The other ATM related problem was that the interfaces created with atmarp and bound to IP with ifconfig never were fully removed on reset. This problem was found out to be a driver bug.

Of course, in this kind of work I met dozens of software configuration problems. Everything had to work with our specific hardware in our specific network environment. Configurations finally found out working and explanations to them can be found on chapter 5. This really was not the easiest part though someone who have not struckled with configurations might easily think so. The point here is that if every part is not all right the whole thing just will not work.

ATM sure was not the only thing that made me sit long days at work. There were problems with ISDN too. The first ISDN cards and their drivers were so buggy I brought the cards back to the store right away. The second ones, Telewell´s work fine. Of course, bugs on Telewell drivers are found too but they do not make networking impossible.

The main problem so to say was making the IP features work on ISDN, ATM and then finally over the switch and the whole system. The configurations on this are found in chapter 5 and the explanation on the whole system is found in chapter 6 so it is not necessary to explain it all again here.

Making sw module work also made some problems. The one to mention is ISDN and ATM links. All the links were not easily implemented. Due to lack o f time ISDN stack did not end up such as planned. We had to make it support only one connection from one device because coding the support to many devices would have taken too much time. This feature will be implemented soon in future.

11.What Did I Actually Do?

- SW Links
- Parts of SW Protocol
- Parts of UNI3.1 Protocol
- Parts of DSS1 Protocol
- Parts of LAPD Protocol
- Switch SVC and PVC testing softwares
- ATM CLIP interface to OVOPS++
- Hardware configuration
- Software configuration
- Fixed dozens of things
- Endless hours of testing and then again configuring
- This paper

12.Final Words

Puuh! It is done at last, the job is done and this paper is written. I must admit that I have not always had fun doing this project. There were situations nothing seemed to work and the project seemed to be impossible to accomplish. Hard work and a strong belief in own capabilites made writing this final chapter finally possible. The positive feeling take the advance now that all is done but of course, during the project there were plenty of good times too. Always when you get something radical work, something you see with your own eyes it makes you happy.

There were many problems but they were solvable and here we are. What else can I say? I am happy it is done.

Special thanks to Antti M. Nissinen (TKK/TLM maintenance) for helping me a lot with Linux and hardware stuff.