ATM on Linux — The 3rd year

Werner Almesberger
werner.almesberger@lrc.di.epfl.ch

Laboratoire de Réseaux de Communication (LRC)
EPFL, CH-1015 Lausanne, Switzerland

March 2, 1997

Abstract

Since the beginning of 1995, ATM support is being developed for Linux. By now, Linux supports most functionality that is required for state of the art ATM networking. This article briefly introduces relevant ATM concepts, presents the current status of development on Linux, gives a configuration example, and outlines the future direction ATM on Linux will take.

1 Introduction

ATM (Asynchronous Transfer Mode, [1], see also [2] for a comprehensive overview of ATM technology) is a network technology for modern high-speed integrated services networks. It is not only very popular in WANs and for high-speed backbones interconnecting LANs, but ATM also offers a rich set of features to support guaranteed Quality of Service (QoS; bandwidth, end-to-end delay, etc.), which is necessary for many multimedia applications.

In order to create an ATM platform for research and education, the Laboratoire de Réseaux de Communication (LRC) of EPFL is developing ATM support for Linux.

The Web main page of the ATM on Linux project with pointers to the latest ATM on Linux distribution and related news is `http://lrcwww.epfl.ch/linux-atm/`

2 ATM basics

ATM is designed for demanding data and multimedia communication, such as audio and video transmission, and high-speed data transfer. The design of ATM has been strongly influenced by the telecommunication community, and therefore ATM differs in many ways from data network architectures like today's Internet. The probably most important differences are the following:

- ATM is connection-oriented
- ATM supports guaranteed QoS ("Quality of Service")
- ATM clearly distinguishes between end systems and "the network"

All these concepts have their counterparts in the telephony network: you have to establish a connection before you can communicate with the other party, the QoS (i.e. that you get reasonable bidirectional voice transmission) is guaranteed and doesn't depend on the network load, and your telephone is very different from, say, a PBX.

Another difference is that ATM sends data in tiny cells with a fixed size of only 53 bytes instead of in variable-size frames. While this difference is important at the lowest protocol layers, higher layers typically use larger units which are then transformed from/to cells by a so-called "ATM adaption layer" (AAL, [3]).

Figure 1 shows the structure of an ATM network. The network itself consists of interconnected switches. Two types of networks are distinguished: "private" networks are typically company or campus networks, and "public" networks correspond to what is offered by telephone carriers. The standardized interface between end systems ("hosts") and the ATM network is called the "user-network interface" (UNI). The UNI defines several types of physical media (i.e. multi-mode fiber, UTP-5, etc.), many bit rates (ranging from only a few Mbps to 155 Mbps and more), line codings, configuration and signaling protocols, etc.

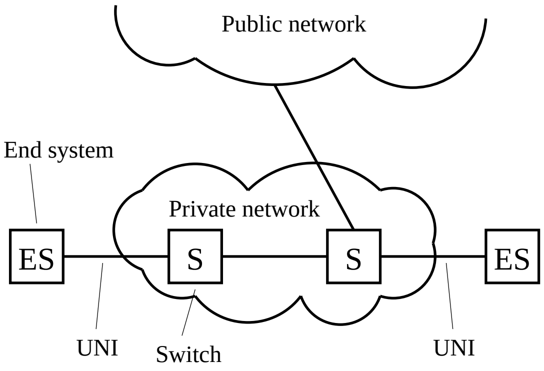

Figure 1: General structure of an ATM network

The most commonly used version of the UNI is 3.1 [4], but many providers of ATM stacks have already implemented the next version, 4.0 [5, 6], or are working on it. UNI 4.0 adds many new features to UNI 3.1, the most interesting ones are probably a scalable point-to-multipoint ("multicast") mechanism and support for ABR ("Available Bit Rate"), a traffic class with congestion control inside the network.

There are two mechanisms for setting up ATM connections: the simple way is to configure each switch individually (a bit like it was done in the early days of telephony, where operators had to physically connect calls on switchboards). Such connections are called "permanent virtual circuits" (PVCs). A more convenient way of setting up connections it to "dial" them, which is called "signaling" in ATM terminology. "Switched virtual circuits" (SVCs) are set up using signaling. ATM signaling is based on the protocols DSS2 (see Q.2931 [7] for unicast and Q.2971 [8] for multicast), which in turn use the so-called SAAL [9, 10, 11] to transport signaling messages.

Figure 2 illustrates ATM signaling: first, the caller sends a SETUP message towards the destination (1). This message is processed at every single switch. If the destination accepts the call, it returns a CONNECT message (2). Again, this message is seen by all switches. When the CONNECT message reaches the destination, the data connection is established and data can be exchanged between both end systems (3).[1] Note that the switches don't have to interpret what is sent on the data connection.

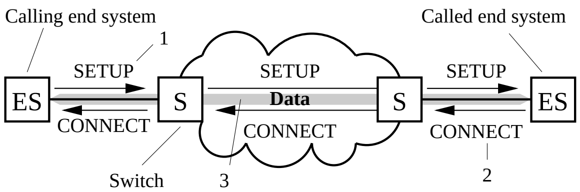

Figure 2: Signaling message flows

Two mechanisms that are closely related to signaling are address configuration and a directory service. Addresses are configured either manually or automatically, using the "interim local management interface" (ILMI, [4]), which is based on SNMP [12].

An ATM NSAP address (see section 5.1.3.1 of [4]) has a length of 20 bytes. Human beings therefore usually prefer to use names instead of numeric addresses. This can be accomplished either by using a hosts file or by using a distributed directory service. ATM Forum has specified such a directory service called ANS ("ATM Name Service", [13]), which is based on BIND ("named").

At the time of writing, ATM on Linux supports PVCs and SVCs with UNI 3.1 signaling. Support for UNI 4.0 signaling is being worked on. The ILMI demon was contributed by Scott Shumate. ANS support was contributed by Marko Kiiskilä.

3 ATM and the real world

ATM purists may dream of a world where all computers, TV sets, telephones, etc., are connected to

1. This is slightly simplified. ATM signaling also allows acknowledgements for the SETUP message and it requires an acknowledgement for CONNECT.

a big ATM cloud consisting of many interconnected ATM networks, but the real world is different: connectionless IP networks without QoS concepts play the dominant role, and "native" ATM applications are a minority.

The first step in running IP over ATM is to have a means to carry IP packets on ATM. This is mainly an encapsulation issue, defined in RFC1483 [14]. With this alone, IP can be run over ATM using PVCs.

For SVCs, also a way to resolve IP addresses to ATM addresses is needed. The IETF currently uses an approach called "classical IP over ATM" that is based on an extension of ARP, called ATMARP [15, 16]. ATMARP works like this (see also figure 3): each IP subnet has one ARP server (C). When a client (**A**, **B**) starts, it registers its own IP and ATM addresses at the ARP server (1). Now, if client **A** wants to send data to client **B**, but it only knows **B**'s IP address, it sends an ATMARP request (2) to the server. If the server knows **B**'s addresses, it responds with an ATMARP reply (3), containing **B**'s ATM address. **A** can now establish an SVC to **B** and send data (4).

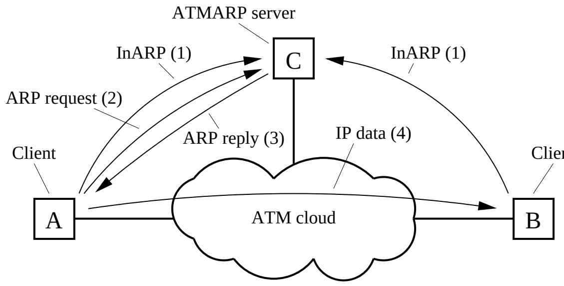

Figure 3: ATMARP message flows

ATM Forum has defined a similar service, called "LAN Emulation" (LANE) [17, 18]. LANE tries to provide exactly the functionality one would obtain from, say, an Ethernet. Therefore, is can also carry other protocols than just IP and it supports multicast (and even broadcast). One disadvantage of LANE with respect to classical IP over ATM is that the maximum IP packet size is limited to 1500 bytes, like on Ethernet, whereas the default maximum IP packet size for classical IP over ATM is 9180 bytes ([19]).

Work from IETF and ATM Forum is being merged in MPOA ("MultiProtocol Over ATM", [20]), which aims to overcome many of the limitations of classical IP and LANE. In particular, MPOA allows the use of "shortcut" connections which bypass intermediate routers.

Furthermore, there is work being done on integrating IP mechanisms for negotiating QoS parameters (e.g. RSVP [21]) with ATM [22].

ATM on Linux supports IP over ATM for PVCs and SVCs as defined by RFC1577 [15] and others. Comprehensive support for LANE, including complete LANE server functionality, has been contributed by Marko Kiiskilä. At the time of writing, work has started to also support MPOA.

Because standard IP currently supports neither direct ATM end-to-end connectivity beyond subnet boundaries nor negotiation of QoS aspects, LRC has designed an extension of ATMARP called Arequipa ("Application Requested IP over ATM", [23]). Arequipa allows applications to request a direct ATM connection for their exclusive use with TCP/IP protocols. The applications can also determine exactly what QoS will be available to them.

4 The implementation

An overview of implementation history and implementation principles can be found in [24]. The design principle of putting as much functionality as possible into user mode processes instead of bloating the kernel has been very successful and continues to be applied.

Further implementation details can be found in the following documents: [25] describes the socket interface for native ATM applications, [26] defines the interface between the ATM stack and device drivers, [27] specifies the protocol used between the kernel and the signaling demon, [28] describes LANE and its implementation on Linux, and [29] details the design and implementation of Arequipa.

5 Configuration example

This section illustrates how two Linux PCs are configured for classical IP over ATM. For further details, please see [30].

The first step is to download the ATM on Linux distribution,[2] to patch, rebuild, and boot the kernel, and to compile and install the ATM tools. All this is described in [30]. In this example, we have two PCs called **alice** and **bob**. They are equipped with ATM adapters and we assume that they are connected by an ATM switch.[3] Figure 4 illustrates the example configuration.

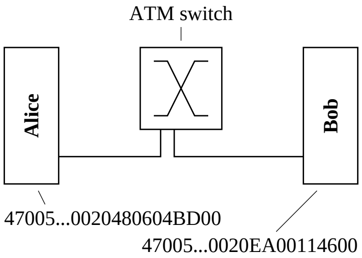

Figure 4: Example ATM network

In order to use signaling, the signaling demon must be started on both machines. Also, their ATM addresses need to be configured. This is normally handled by the ILMI demon. So the next commands on both machines are (the `-b` option starts the demons in the background):

```
# atmsigd -b
# ilmid -b
```

After that, the ATM addresses can be queried with the `atmaddr` command (the `-n` option yields numeric output):

```
alice% atmaddr -n
47000580FFE1000000F21A26D80020480604BD00

bob% atmaddr -n
47000580FFE1000000F21A26D80020EA00114600
```

Now, classical IP over ATM can be set up. Both machines connect to the same Logical IP Subnet (LIS) with the address 193.8.232.0 (netmask 255.255.255.0), see figure 5.

First, the ATMARP demon needs to be started on each machine. After that, the IP over ATM interface can be created with `atmarp -c`, brought up (`ifconfig`), and the route to the LIS can be added to the routing table:

```
alice# atmarpd -b
```

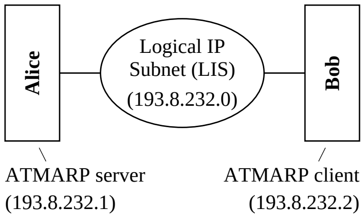

Figure 5: Example LIS (Logical IP Subnet) configuration

```
alice# ifconfig ‘atmarp -c‘ 193.8.232.1 up
alice# route add -net 193.8.232.0

bob# atmarpd -b
bob# ifconfig ‘atmarp -c‘ 193.8.232.2 up
bob# route add -net 193.8.232.0
```

So far the configuration of both machines has been almost identical. The final step is to define who acts as the ATMARP server (see section 3) and who acts as an ATMARP client. In our example, **alice** is the ATMARP server. Because `atmarpd` assumes by default to be an ATMARP server, no further configuration is needed on **alice**.

However, **bob** needs to be told where is has to look for its ATMARP server. This is configured with the `atmarp` command:

```
bob# atmarp -s 193.8.232.0 \
  47000580FFE1000000F21A26D80020480604BD00 \
  arpsrv
```

That's all ! Now, both machines can talk to each other using TCP/IP, e.g.

```
bob% ping 193.8.232.1
PING 193.8.232.1 (193.8.232.1): 56 data bytes
64 bytes from 193.8.232.1: seq=0 time=3.1 ms
64 bytes from 193.8.232.1: seq=1 time=0.9 ms
64 bytes from 193.8.232.1: seq=2 time=0.9 ms
```

6 The future

The core functionality of ATM on Linux has stabilized in '96. Drivers for several new ATM adapters

2. See `http://lrcwww.epfl.ch/linux-atm/`

3. Configurations without an ATM switch are possible, but they are typically only used for certain experimental setups.

have been contributed and people are continuing to work on drivers for additional adapters.

At the time of writing, two major changes are happening: signaling is extended to support several features of UNI 4.0, and the kernel code is upgraded for use in 2.1 kernels.

ATM on Linux is expected to be ready for integration into the next stable release of the "mainstream" Linux kernel in the middle of 1997.

7 Conclusion

A brief introduction to the most important concepts of ATM in today's networking world was given and it was illustrated that ATM on Linux supports all the respective mechanisms.

A configuration example for setting up a classical IP over ATM subnet was given.

At the end of this paper, plans for future development were described.

References

[1] Le Boudec, Jean-Yves. *The Asynchronous Transfer Mode: a tutorial*, Computer Networks and ISDN Systems, Volume 24, Number 4, 1992.

[2] Alles, Anthony. *Internetworking with ATM*, `http://cell-relay.indiana.edu/cell-relay/docs/cisco.html`, Cisco Systems, May 1995.

[3] ITU-T Recommendation I.363. *B-ISDN ATM adaptation layer (AAL) specification*, ITU, March 1993.

[4] The ATM Forum. *ATM User-Network Interface (UNI) Specification, Version 3.1*, `ftp://ftp.atmforum.com/pub/UNI/ver3.1`, Prentice Hall, 1994.

[5] The ATM Forum, Technical Committee. *ATM User-Network Interface (UNI) Signalling Specification, Version 4.0*, `ftp://ftp.atmforum.com/pub/approved-specs/af-sig-0061.000.ps`, The ATM Forum, July 1996.

[6] The ATM Forum, Technical Committee. *ATM Forum Traffic Management Specification, Version 4.0*, `ftp://ftp.atmforum.com/pub/approved-specs/af-tm-0056.000.ps`, April 1996.

[7] ITU-T Recommendation Q.2931. *Broadband Integrated Services Digital Network (B-ISDN) – Digital subscriber signalling system no. 2 (DSS 2) – User-network interface (UNI) – Layer 3 specification for basic call/connection control*, ITU, February 1995.

[8] ITU-T Recommendation Q.2971. *B-ISDN – DSS 2 – User-network interface layer 3 specification for point-to-multipoint call/connection control*, ITU, October 1995.

[9] ITU-T Recommendation Q.2100. *B-ISDN signalling ATM adaptation layer (SAAL) overview description*, ITU, July 1994.

[10] ITU-T Recommendation Q.2110. *B-ISDN ATM adaptation layer – service specific connection oriented protocol (SSCOP)*, ITU, July 1994.

[11] ITU-T Recommendation Q.2130. *B-ISDN signalling ATM adaptation layer – service specific coordination function for support of signalling at the user network interface (SSFC at UNI)*, ITU, July 1994.

[12] RFC1157: Schoffstall, Martin Lee; Fedor, Mark; Davin, James R.; Case, Jeffrey D. *A Simple Network Management Protocol (SNMP)*, IETF, May 1990.

[13] The ATM Forum, SAA/Directory Work group. *ATM Name Service (ANS) Specification Version 1.0*, `ftp://ftp.atmforum.com/pub/approved-specs/af-saa-0069.000.ps`, The ATM Forum, November 1996.

[14] RFC1483; Heinanen, Juha. *Multiprotocol Encapsulation over ATM Adaptation Layer 5*, IETF, 1993.

[15] RFC1577; Laubach, Mark. *Classical IP and ARP over ATM*, IETF, 1994.

[16] RFC1755; Perez, Maryann; Liaw, Fong-Ching; Mankin, Allison; Hoffman, Eric; Grossman,

Dan; Malis, Andrew. *ATM Signaling Support for IP over ATM*, IETF, 1995.

[17] Truong, H. L.; Ellington, W. W. Jr.; Le Boudec, J.-Y.; Meier, A. X.; Pace, J. W. *LAN Emulation on an ATM Network*, IEEE Communications Magazine, May 1995, pp. 70–85.

[18] The ATM Forum, Technical Committee. *LAN Emulation Over ATM, Version 1.0*, `ftp://ftp.atmforum.com/pub/approved-specs/af-lane-0021.000.ps`, The ATM Forum, January 1995.

[19] RFC1626; Atkinson, Randall J. *Default IP MTU for use over ATM AAL5*, IETF, 1994.

[20] The ATM Forum, Multiprotocol Sub-Working Group. *MPOA Baseline Version 1*, `ftp://ftp.atmforum.com/pub/mpoa/baseline.ps`, September 1996.

[21] Braden, Bob; Zhang, Lixia; Berson, Steve; Herzog, Shai; Jamin, Sugih. *Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification* (work in progress), Internet Draft `draft-ietf-rsvp-spec-14.ps`, November 1996.

[22] RFC1821; Borden, Marty; Crawley, Eric S.; Davie, Bruce S.; Batsell, Stephen G.. *Integration of Real-time Services in an IP-ATM Network Architecture*, IETF, August 1995.

[23] Almesberger, Werner; Le Boudec, Jean-Yves; Oechslin, Philippe. *Application Requested IP over ATM (AREQUIPA) and its Use in the Web*, Global Information Infrastructure (GII) Evolution, pp. 252–260, IOS Press, 1996.

[24] Almesberger, Werner. *ATM on Linux*, `ftp://lrcftp.epfl.ch/pub/linux/atm/papers/atm_on_linux.ps.gz`, EPFL, March 1996.

[25] Almesberger, Werner. *Linux ATM API*, `ftp://lrcftp.epfl.ch/pub/linux/atm/api/`, EPFL, July 1996.

[26] Almesberger, Werner. *Linux ATM device driver interface*, `ftp://lrcftp.epfl.ch/pub/linux/atm/docs/`, January 1996.

[27] Almesberger, Werner. *Linux ATM internal signaling protocol*, `ftp://lrcftp.epfl.ch/pub/linux/atm/docs/`, September 1996.

[28] Kiiskilä, Marko. *Implementation of LAN Emulation Over ATM in Linux*, `ftp://viulu.atm.tut.fi/pub/misc/linux-lane.ps.gz`, Tampere University of Technology, October 1996.

[29] Almesberger, Werner. *Arequipa: Design and Implementation*, `ftp://lrcwww.epfl.ch/pub/arequipa/aq_di-1.tar.gz`, Technical Report 96/213, DI-EPFL, November 1996.

[30] Almesberger, Werner. *ATM on Linux User's guide*, `ftp://lrcftp.epfl.ch/pub/linux/atm/dist/`, EPFL, July 1996.