Classical IP and ARP over ATM

Christoph L. Schuba [a,b], Berry Kercheval [b], and
Eugene H. Spafford [a]

[a] {schuba,spaf}@cs.purdue.edu
COAST Laboratory, Purdue University, 1398 Department of Computer Sciences, West Lafayette, IN 47907–1398

[b] {schuba,kerch}@parc.xerox.com
PARC – CSL, Xerox Corporation, 3333 Coyote Hill Road, Palo Alto, CA 94304

Abstract

This paper gives a self–contained description of classical IP (*internet protocol*) and ARP (*address resolution protocol*) over ATM (*asynchronous transfer mode*) and describes a model facilitating the implementation of the switched virtual circuit–based local area network ATM subnet model. Its contents are distilled from the design and implementation of a prototype of a device driver for this particular subnet model. The work was conducted at the *Computer Science Laboratory* (CSL) at the Xerox *Palo Alto Research Center* (PARC).

We outline the main features of the two technologies, establish their importance as data communication paradigms, motivate their integration, and sketch several scenarios of their interaction. We concentrate on one specific scenario: the utilization of ATM as a logical IP subnetwork. Building blocks of this approach are the ATM address resolution protocol as a unicast server based emulation of the classical broadcast address resolution protocol, and connection management functionality as provided by the Q.2931 protocol. The manipulation of transmitted network protocol data units between their submission to the network layer and their transmission by ATM are detailed in an appendix. These explanations are structured analogous to the lower layers in the IEEE local area network model.

The complexity and intricacy of the *logical IP subnetwork* (LIS) approach motivate the development of a model describing the interaction of all participants in an LIS. The devised model is based on a finite automaton. It instruments a structured implementation and it assists in convincing the implementor of the completeness and correctness of the implementation by facilitating its gradual testing and extension.

1 Introduction

The *internet protocol* (IP) is the standard protocol on the Internet that provides the basis for a connectionless, best–effort packet delivery service. IP uses an *address resolution protocol* (ARP) which is based on broadcast to dynamically bind high level protocol addresses to low level physical hardware addresses.

Asynchronous transfer mode (ATM) is rapidly becoming acknowledged as the base technology for the next generation of global communications. Both the technology and the standardization process receive extensive acceptance throughout the industry. Some of ATM's main characteristics are point–to–point or point–to–multipoint connection–oriented communications with small cells as data units and a good aggregate behavior. This particular form of cell networking provides the capability for high speed transmission in local area and wide area networks. Another strength of ATM are end–to–end *quality of service* (QoS) guarantees to support applications with special QoS demands.

The combination of the well established place of IP in the data communication world and the telephone companys' need for ATM as their new base technology establish the need for the integration of IP and ATM. In spite of some problems with the technology it promises a considerable service improvement for data communications.

This paper explains one architectural proposal for the interaction between IP and ATM and describes a prototype implementation. The physical topology of ATM networks and the logical structure imposed by the IP model are not easily mapped onto each other. To integrate the two different communication paradigms one can create *logical IP subnetworks* (LIS) in ATM that operate and communicate independently of other LISs on the same ATM network.

A modified form of ARP is used in an LIS to resolve IP to ATM addresses. Before data can be transmitted between two ATM endstations, a connection is established under the usage of a connection management module and data is packaged and fragmented into small size ATM cells by an adaptation layer. If other protocols besides IP utilize the ATM network an additional encapsulation of data becomes necessary to demultiplex received and reassembled ATM cells if the protocols are all multiplexed over a single virtual channel.

2 Integration of Classical IP and ARP with ATM

2.1 Classical IP and ARP

The *internet protocol* (IP) [RFC 791] defines a best–effort, connectionless delivery mechanism. It provides for the transmission of blocks of data (datagrams) from sources to destinations over an interconnected system of networks. IP imposes few requirements on utilized network technologies. Therefore, IP can only guarantee an earnest attempt to deliver packets. A *local area network* (LAN) protocol defines the physical and *data link layer* (DLL) in the ISO–OSI (*International Standards Organization – open systems interconnection*) reference model.

Concentrating on the IEEE approach of modeling networks (e.g., [IEEE 802.2]), the DLL is divided into *medium access control* (MAC) and *logical link control* (LLC) sublayers. The LLC sublayer provides a common interface for multiple network layer protocols to interoperate with different MAC sublayers and share the use of a data link. The MAC sublayer defines the mechanisms that are used to access, share, and manage a communication medium. That includes channel management, collision detection and resolution, priority handling, error detection and framing. Typical examples of LAN technologies (MAC sublayer and physical layer) are defined by the family of IEEE 802.x standards (e.g., CSMA/CD [IEEE 802.3], or DQDB [IEEE 802.6]). Figure 1 illustrates the correspondence between MAC & LLC and the DLL.

<table>
<tr><td>3</td><td>Network Layer</td><td></td><td></td></tr>
<tr><td rowspan="2">2</td><td rowspan="2">Data Link Layer</td><td>Logical Link Control (LLC)</td><td>IEEE 802.2</td></tr>
<tr><td>Medium Access Control (MAC)</td><td rowspan="2">IEEE 808.3-9</td></tr>
<tr><td>1</td><td>Physical Layer</td><td>Physical Layer (PHy)</td></tr>
<tr><td colspan="2">OSI Layering</td><td colspan="2">IEEE Layering</td></tr>
</table>

Fig. 1. Correspondence between ISO and IEEE layering, in particular the correspondence between data link layer and medium access control and logical link control

In this model two machines on a physical network can only communicate if they know each other's MAC addresses. Determining a machine's MAC address given its network layer address is known as the address resolution problem.

2.2 ATM

Asynchronous transfer mode (ATM) was developed for use in *broadband integrated services digital networks* (B–ISDN) to carry data, voice, images, and video traffic in an integrated manner. ATM is not limited to B–ISDN and contains physical layer and network layer functionality. Its architecture is based on switching small fixed–length packets called cells. Some aspects of ATM are currently defined by interim standards developed by a user and vendor group known as the ATM Forum [UNI 3.0]. ATM provides for point–to–point or point–to–multipoint, connection–oriented transmission of small data units. ATM gives quality of service guarantees (i.e., it handles resource reservation, offers bandwidth and latency guarantees), and exhibits a good aggregate behavior.

Before data can be transferred between machines, a connection must be established. These connections are called *virtual channels* (VC) and are identified hop–by–hop using a *virtual path* and *virtual channel identifier* pair (VPI/VCI). Data are transferred in the form of cells which are small fixed–size data packets with a 48 byte payload and a 5 byte header. Each switch contains mappings of input to output VC identifiers. These can be (semi–) permanently installed, (called *permanent virtual circuits*, or PVCs) or established at connection setup time by a signaling protocol (called *switched virtual circuits*, or SVCs). The switching of cells involves the appropriate change of VC information in the cell header and a forwarding of the cell over the associated physical link. A switch controller is associated with one or more ATM switches. It performs management functions such as enabling or disabling ports, polling for status information, or updating mapping information.

2.3 Motivation for the Integration of Paradigms

IP and ARP (*Address Resolution Protocol*) on the one hand and ATM on the other represent very different paradigms for data communication. What is the motivation to integrate these two technologies?

IP is the main building block of the Internet, the global network of interconnected networks. In the classical TCP/IP protocol hierarchy (see [Com91a]) IP is the central protocol. It serves a hierarchy of protocols (e.g., UDP [RFC 768] or TCP [RFC 793]), and integrates a wide variety of network technologies as its underlying data link and physical layers. Its main concepts have ripened to maturity and proven adequate for robust data delivery in interconnected heterogeneous environments. IP has established its place in today's data communication world. That is not expected to change in the foreseeable future.

ATM is rapidly becoming acknowledged as the base technology for the next generation of global communications. Telecommunication industries are planning on implementing ATM because it meets internal telephony needs. The integration of voice and data traffic into one medium saves resources. It is easier to multiplex separately addressed packets of voice and data than to multiplex mixes of individual bits. Moreover, new services can be offered to users, and the technology can be used to satisfy the telephone companys' own need for increased data communications because of increased computerization. ATM deployment in the local area promises performance advances in distributed computing with high performance and low latency data communication requirements, such as multimedia applications. Furthermore, ATM as a LAN technology promises easy connectivity of existing hubs, bridges, and routers to future wide–area ATM networks.

ATM is not without its problems or detractors, however. For example, it is not clear that classical telecommunication traffic and data communication will integrate well. Classical telecommunication can be described well by Poisson models, whereas the burstiness of self similar data traffic does not change over time (see [LTWW93]). There are difficulties with scaling ATM and maintaining reliability. Virtual circuits establish a *hard state* in the network for a call. The state of a connection is distributed over intermediate switching equipment. Once a VC is established, it is maintained until a message is received by one of the ends of the call requesting a change in the state of the connection. The product law of reliability, applicable to series systems of independent components, such as the hops in a virtual circuit, demonstrates how quickly system reliability degrades with an increase in complexity. On the other hand dynamic rerouting in a highly redundant system such as the IP network layer reaches a reliability approximating 100% (see [Tri82, Ch.1.10] for an analysis of the reliability in parallel redundant systems). Additionally, ATM does not yet support any dynamic routing functionality or network admission control.

For a more detailed discussion of benefits and drawbacks of ATM see [Par93, sections 4.2 and 4.10] and [BG92, section 2.10]

The combination of the well established place of IP in the data communication world and the telephone companys' need for ATM as their new base technology establish the need for the integration of IP and ATM. In spite of some problems with the technology it promises a considerable service improvement for data communications.

2.4 Scenarios of Interaction

The *IP over ATM* Working Group of the *Internet Engineering Task Force* (IETF) is chartered to develop standards for routing and forwarding IP packets over ATM subnets. In [CS94] the working group identifies and discusses several possible subnet models and end–to–end models.

The subnet models identified are an SVC–based *LAN ATM* (LATM) subnet, a PVC–based *WAN ATM* (WATM) subnet, and lightweight subnet models. End–to–end models of interest are the classical IP over ATM model, the SVC–based WATM, or *routing over large clouds* (ROLC) model, Ohta's "conventional" model, and peer models.

This paper focusses on the SVC–based LATM subnet model and the classical IP over ATM end–to–end model. We first give a brief description of the other models.

The PVC-based WATM is a non–broadcast multiple access subnetwork with possibly up to tens of thousands of endstations. These are all directly connected, share a common IP network prefix, support ATM PVC connections, do not have to support an efficient address resolution process, and are part of a single administrative group.

Lightweight subnet models are based on the observation that in single hop configurations some IP header information is implicitly bound to virtual circuits established between entities in two subnetwork endstations. The work to be done shifts between in–stream demultiplexing and call setup demultiplexing develops in several discrete steps. On one end lies LLC/SNAP encapsulation (see section A.2) and endstation demultiplexing based on source and destination address, protocol family, protocol, and port. On the other end lies a null encapsulation with the burden of demultiplexing placed upon the connection establishment capacity of the ATM subnet, with different forms of null encapsulation in the middle.

The SVC–based WATM supports SVCs as well a PVCs. Endstations do not necessarily share a common IP level network prefix, and multiple SNAP address formats and negotiation of parameters (such as MTU (*maximum transfer unit*) size or method of encapsulation) are supported. Further features are that the SVC–based WATM is not part of a single administrative group (which makes security an important consideration), and the need for a distributed address resolution protocol and billing scheme.

Ohta's "conventional" model has the same network layer architecture as the standard IP model, including its subnet architecture. However, IP level routers are assumed to be able to forward data cell by cell, yielding a possible latency advantage.

Peer models are motivated by the rationale that routing complexities of future ATM networks will be similar to routing over complex internets. The possibility to embed IP level routing within the ATM network fabric leads to peer models where IP routers are placed on a peer basis with corresponding entities in an ATM network.

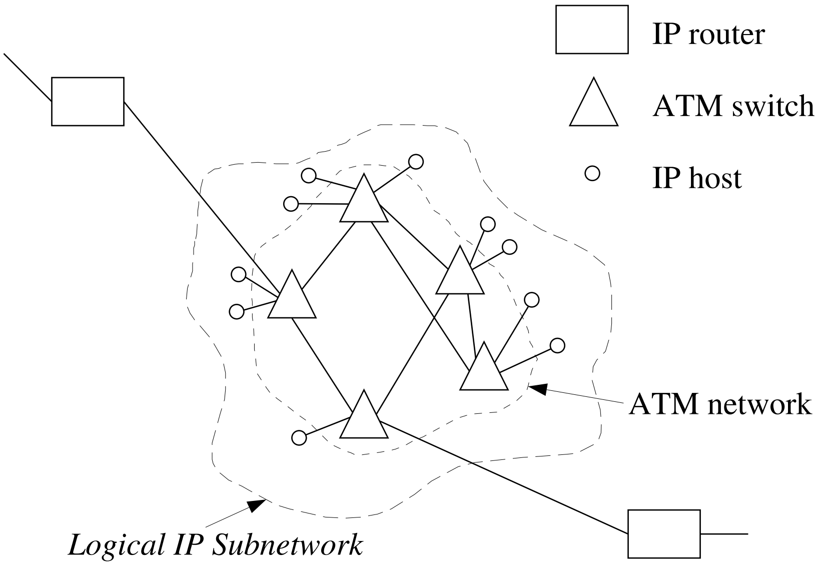

Fig. 2. A logical IP subnetwork (LIS or LATM) appears to the IP layer as just another LAN technology

The SVC-based LATM model takes advantage of the fact that IP can make use of any networking technology that provides connectivity as a delivery subsystem, including ATM. The goal of this approach is to transmit IP datagrams over an ATM LAN (see Figure 2). A *logical IP subnetwork* (LIS or LATM) is a single ATM network, that is a direct replacement for the local area network segment connecting IP routers and endstations in the classical IP and ARP paradigm. All endstations are directly connected and they share a common IP level network prefix.

One LIS can grow to replace several LAN segments. Several LISs can operate on the same ATM network. At least one IP router must be attached to each LIS to integrate it with the rest of the internetwork. It is configured as a member of one or several LISs and handles inter domain routing of datagrams transparently. IP hosts on the same ATM network that are configured to belong to different LISs must not establish a connection to each other directly through ATM, but must communicate with each other via an intermediate IP router.

The previously mentioned necessity for the network layer to resolve its addresses to MAC addresses (see section 3) is not the only problem that has to be solved to establish an LIS as a new subnetwork technology. In appendix A we will explain the conversion between network layer protocol data units (e.g., IP datagram) and ATM cells and the need for adaptation and encapsulation layers and a convergence sublayer. Section 4 will outline our approach to an integrated address resolution and call establishment in the presence of switched virtual circuits.

3 The Protocols

Address resolution in an LIS, and ATM connection management are nontrivial protocols. In this section we are concerned with their description.

3.1 ARP and InARP

The address resolution problem, e.g., determining a machine's MAC address given its network layer address, has been solved in many different ways. One well known solution in the case of Ethernet as the LAN technology is ARP ([RFC 826]). A machine S determines the MAC address of another machine D by broadcasting an ARP request containing D's network protocol address. The machine that matches the network protocol address of an ARP request responds with an ARP reply containing D's MAC address.

Inverse ARP[1] (InARP) is defined in [RFC 1293] and addresses a different problem. Given an established connection between two machines, they do not have to know one another's higher level protocol addresses if the call establishment was based on lower layer addresses. InARP is a protocol that enables either endpoint of a connection to query the other one for its protocol addresses.

3.1.1 Why Classical ARP is not sufficient

Classical ARP relies on an efficient broadcast capability of the physical layer[2]. Even if ATM multicast were already widely implemented, it is not advisable to simulate broadcast by ATM's multicast connections, because of its poor scaling. Point–to–multipoint connections offer bidirectional communication only between the originator and all destinations, but not among the destinations.

[1] Note that we are talking about InARP, not *reverse ARP.* (RARP) [RFC 903].

[2] We are not considering a static address binding for ATM networks, because ATM physical addresses are longer than IP addresses and cannot be freely chosen.

To simulate broadcast, each endstation would have to establish a point–to–multipoint connection to all other endstations.

A different approach, described in [RFC 1577], is to use a dedicated service on the network that maintains a cache of address mappings. The mappings are established through registration by each host at its boot time and are periodically updated. The service replies to address resolution requests with the desired mapping, or a negative acknowledgement, if the mapping is not available.

ATM physical addresses are longer than IP addresses. Therefore IP cannot use static address binding for ATM networks.

3.2 ATMARP and InATMARP

The *ATM address resolution protocol* (ATMARP) is basically the same protocol as ARP extended with support in a unicast server environment. ATMARP answers the question: "Given a network protocol address, what is the associated ATM address?".

Inverse ATMARP (InATMARP) is the same protocol as InARP applied to ATM networks. Given a virtual circuit identifier between two endpoints, InATMARP answers the question: "What are the other endpoint's network layer and ATM address?" All hosts participating in an LIS must support these extended protocols as defined in [RFC 1577]. Each LIS must have one ATMARP server.

The ATMARP service contains the following elements which we will outline in successive sections: address mapping registration, queries and positive or negative replies, and mapping table maintenance.

3.2.1 Registration

ATMARP clients must start the ATMARP registration process with the server before they can participate as a member of an LIS. Each client must initiate a SVC connection to the server. Upon ATM call acceptance from any other ATM endpoint, the server generates and transmits an `InATMARP_REQUEST` to the client asking it for its network layer to ATM address binding. Clients must reply to each `InATMARP_REQUEST` with an `InATMARP_REPLY`. The server will update his ATMARP table with the received mapping and a timestamp unless a mapping for the same network layer address and an established VC already exist.

3.2.2 Query and Reply

Once hosts in an LIS are registered they can request address resolutions for other hosts in the same LIS via `ATMARP_REQUESTs`. Clients generate and transmit `ATMARP_REQUEST` packets to the server. An `ATMARP_REPLY` packet from the server then contains the mapping which is used to build/refresh the clients cache. An `ATMARP_NAK`[3] packet from the server tells the client that the requested mapping could not be resolved. It is an extension to classical ARP adding to the robustness of the protocol by giving clients the ability to distinguish a cache miss from a fatal server failure.

3.2.3 Cache Maintenance

The ATMARP server maintains an ATMARP table that ideally contains the network layer to ATM address mapping for all hosts participating in its LIS. It is initially built by the `InATMARP_REPLY` packets received from hosts at registration time. Clients maintain a similar table that contains mappings that they previously requested from the server. The entries in the tables are aged periodically. [RFC 1577] requires that client table entries remain valid for at most 15 minutes and server table entries for at least 20 minutes. Before invalidating an entry that still has an open VC associated, a server must attempt to revalidate the mapping by generating and transmitting an `InATMARP_REQUEST`. Entries without an associated open VC are deleted. ATMARP table entries exist until they are aged or invalidated. Teardown of an SVC does not remove ATMARP table entries.

There are other occasions when the server can update his table. When the server receives an `ATMARP_REQUEST` over a VC, whose associated network layer and ATM address match the table entry, the server can update the timeout on that entry. To add robustness, the server examines the source information of received `ATMARP_REQUESTs`. If there is no table entry present for the source network address, the server adds the source information mapping to its table associated with the VC.

3.2.4 Unresolved Issues

Several issues remain unresolved by [RFC 1577]. If only one server per LIS assumes the task of address resolution, the service depends on the overall reliability of that server machine. A protocol that embodies a replicated ATMARP service is desirable. Currently the ATMARP service is reached via a well known address. A mechanism should exist for determining the ATMARP service access point dynamically.

[3] *nak* is the abbreviation of *negative acknowledgement*.

3.3 Connection Management

We mentioned before that communications in ATM are based on a connection–oriented model. Before user data can be transmitted, a virtual circuit has to be established. VCs come in two forms: permanent VCs, and switched VCs. Permanent VCs are statically configured and installed into the switching tables of the ATM switches. Switched point–to–point and point–to–multipoint connections over an ATM network are dynamically established, maintained, and cleared on a need basis utilizing the Q.2931 protocol (see [UNI 3.0]), a broadband signaling standard.

The initiator of a connection specifies desired characteristics for the connection and relies on Q.2931 to establish or to report the failure of establishment of a call. Q.2931 neither supports any routing functionality nor provides for call acceptance or forwarding policies.

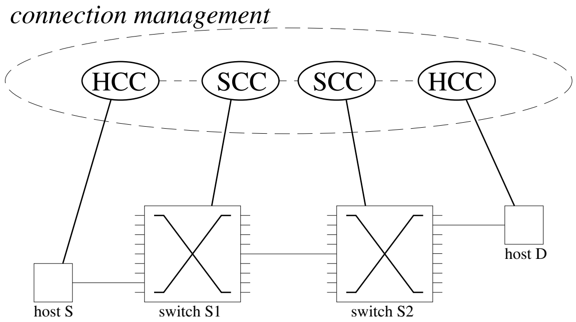

Fig. 3. ATM connection management

Each entity participating in connection management must have an associated process that handles Q.2931 protocol messages. [Qport94b] distinguishes between *host call control* (HCC) and *switch call control* (SCC) modules. Figure 3 depicts the interrelation among the switching processes and their relationship to either ATM switches or endsystems. All HCC and SCC processes initially read configuration files which describe local network configuration, such as port identification, routing information, and address information for neighboring HCC and SCC processes. It is the responsibility of HCC processes to coordinate the establishment and release of calls. A HCC process communicates with the application and the Q.2931 module to coordinate their interaction regarding offering, acceptance, and release of each call. A SCC process is responsible for interfacing peer Q.2931 instances on the ATM switches. It accepts incoming call requests from other SCC or HCC processes and determines the outgoing port which should be used when routing the call. Routing information is accessed through a route manager on the ATM switch.

Figure 4 sketches the integration of the connection management into a host. A HCC process runs in user space, interacts with the device drivers for ATM host

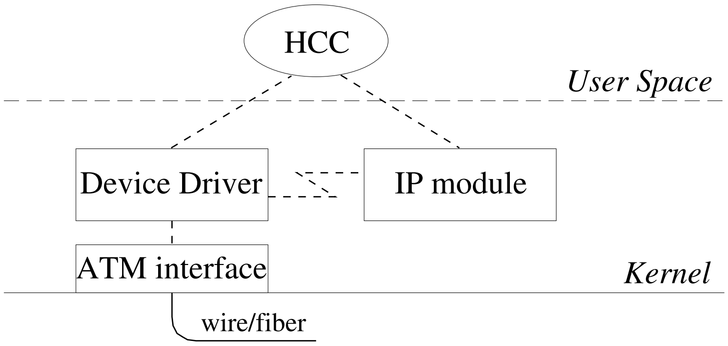

Fig. 4. Integration of connection management into a host

interfaces directly and communicates with its peer SCC and HCC processes on different machines through the regular TCP/IP protocol stack. Both HCC and SCC processes must, at least initially, transmit messages over PVCs to participate in connection management, or they could not communicate among each other. ATM reserves a well known VC for connection management.

4 Model of Integration

Now that we have established the complexity and some of the details of the classical IP and ARP over ATM in an SVC–based LATM, we need to devise a model for the implementation of the proposed architecture. Clearly, Figure 2 hides much of the complexity and shows only an idealized, transparent view of the newly proposed subnetwork architecture.

The system has to integrate connection management, ATM address resolution, and LLC/SNAP encapsulation. In the absence of ATM host interfaces that provide adaptation layer encapsulation and segmentation and reassembly, the system also has to manage a reassembly cache at cell level. Packets that are submitted by the network layer have to be buffered if no SVC to the destination address has already been established. Timers are associated with each reassembly in progress and all buffered network layer packets.

Communication between all involved entities is asynchronous and non–blocking. That requires the recording of all sent messages that expect a reply or an acknowledgement, associated timers, and possibly retransmissions in order to ensure the reliability of the message transport.

4.1 Integrated Model

The previous sections have established the functionality of the modules that participate in ATM as a LAN technology. They all contribute actions as well as pieces of information. It is possible to describe actions (e.g., call establishment, or address resolution) that have taken place with the amount and relationship of address information that has been gathered in the process. That means that the complex interaction between signaling protocol and address resolution protocol in an LIS can be reduced to a conceptually much simpler problem, which is described by a finite state automaton. We want to stress that Figure 5 does not show the whole automaton that is necessary to describe the system. We will not present a complete specification of the automaton, because it is not crucial for the understanding of our approach.

We chose this model because it has several nice features. Its simple structure is easy to understand. It instruments a structured implementation and assists the implementor in convincing himself of its completeness and correctness. Furthermore, the model facilitates the stepwise testing and extension of the implementation. For a functionally restricted prototype, a partial implementation of the automaton is sufficient. A stepwise extension is generally possible without the modification of the already implemented portion of the automaton. We experienced that the implementation became quickly very complex without the guidance of the model.

The states in our model are the possible combinations of network layer address, medium access control sublayer address, switched virtual circuit identifier, and signaling code specific call handle in regard to one network layer address. The set of possible states is identical for the ATMARP server and its clients (i.e., all participants of an LIS).

The automaton exists concurrently in many different instances in a host: once for each destination network layer address. Ideally this automaton has to be augmented to note which encapsulation type is associated with each connection. As explained in appendix A.1, our approach assumes that LLC/SNAP encapsulation is present. State (0) as in Figure 5 is the initial state. The set of final states is empty because this protocol does not have a natural termination.

Figure 5 depicts a scenario where a network layer packet is passed to our module and neither a known ATM address mapping, nor an established switched virtual circuit to the destination address is present. The first transition results in buffering the packet, sending an `ATMARP_REQUEST` to the ATMARP server, and recording that this request was sent. If the `ATMARP_REPLY` contains the desired mapping, it moves the automaton into a state where the mapping network layer address to ATM address is complete. A fresh timer value will be

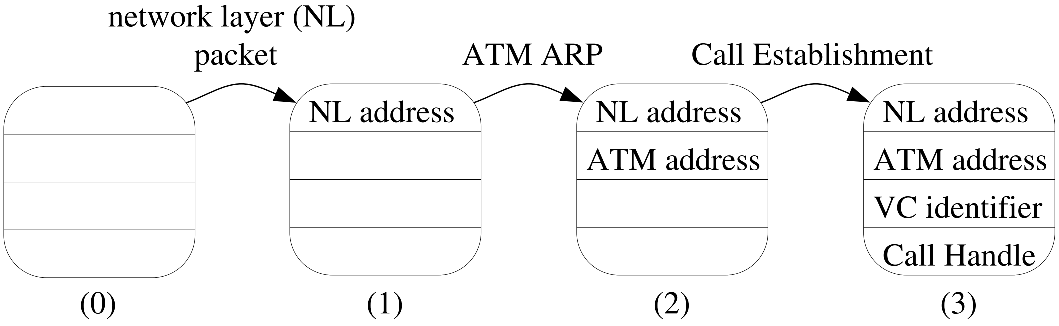

Fig. 5. Example for a complete resolution process. This graph is only a small subset of the real automaton

associated with this mapping (not shown in Figure 5) and the recorded request will be cleared. The next action at this point is to send an `attempt` message to the HCC and record this attempt. Once an `originated--call--connected` message from the HCC is received, we move into the fully resolved state, store the associated VC identifier and call handle, clear the recorded attempt, and send all network layer packets that were enqueued up to that point. All successive packets to the same address will be sent over the associated SVC directly.

Obviously, this was a best case scenario, where no special cases occurred. The following list details all possible events that must be considered in the process:

– data packets from the network layer
– `ATMARP_REPLY` and `ATMARP_NAK` messages
– `InATMARP_REPLY` messages
– connection management messages:
  · `originated--call--connected`
  · `released`
  · `terminated--call--connected`
  · `error`
– expiration of timers for any of the recorded requests or attempts
– expiration of timer for the validity of an ATMARP mapping

The complete automaton can be produced by applying all possible events to an initially empty state (see state (0) in Figure 5 and all resulting states, until a complete transition table is created. New states and actions associated with the state transitions are determined by careful study of the involved protocol specifications.

5 Conclusions and Future Work

The previous sections of the paper are a self–contained description of classical IP and ARP over ATM in an SVC–based LATM in the form of a logical IP subnetwork.

We have implemented a prototype of this model under SunOS 4.1.3 with ForeRunner SBA–100 host adapters and therefore proven the feasibility of [RFC 1577] and [RFC 1483]. The connection management code used was a prototype implementation of the Q.2931 protocol by Bellcore (see [Qport94a,Qport94b,Qport94c]).

To facilitate our implementation we have devised a simple model in section 4 that describes the complex interaction of these protocols. The model instrumented a cleanly structured implementation and assisted us in convincing ourselves of its completeness and correctness. Furthermore, the model facilitated the stepwise testing and extension of the implementation.

An idea for future work is the development and study of a different model in which states are represented by data entries in a relational fashion. The relations must describe the functional dependencies among the different address and identifier entities that are mapped onto each other. The basic idea here is that the mappings can be described without loss of information in 5NF[4], not in 1NF such as in our model. The advantages are clear: a reduction of the degree of redundancy in the state information, a smaller automaton and therefore easier model, and the possibility of a simplified enforcement of integrity constraints. Further work includes the porting of the driver software to further hardware interfaces, as well as interoperability tests with other signaling implementations.

Acknowledgements

This work was sponsored in part by the US Advanced Research Projects Agency under contract DABT63–92–C–0034 and by a gift from SUN Microsystems to the COAST laboratory.

We would like to thank Sandeep Kumar and John Lin for their valuable feedback on early drafts of this paper. Bryan Lyles provided much technical insight and encouragement. Ron Frederick provided much valuable help (and no small amount of code) with the driver prototype.

[4] NF stands for *normal form*. See [Dat91, chapter 21] for a description of normalization theory in relational database design.

References

[BCS93] Edoardo Biagioni, Eric Cooper, and Robert Sansom. Designing a Practical ATM LAN. *IEEE Network*, pages 32–39, March 1993.

A description of Fore Systems' ATM LAN switch and host interface design.

[BG92] Dimitri Bertsekas and Robert Gallager. *Data Networks.* Prentice–Hall, Englewood Cliffs, New Jersey, second edition, 1992.

Textbook in computer networking.

[Bou92] Jean-Yves Le Boudec. The Asynchronous Transfer Mode: A Tutorial. *Computer Networks and ISDN Systems*, 24:279–309, 1992.

A Tutorial for ATM.

[CGST94] H. Jonathan Chao, Dipak Ghosal, Debanjan Saha, and Satish K. Tripathi. IP on ATM Local Area Networks. *IEEE Communications Magazine*, pages 52–59, August 1994.

Addresses issues that are involved in implementing IP in ATM LANs: LAN emulation, ATM as a link layer protocol, and connection management.

[Com91a] Douglas E. Comer. *Internetworking with TCP/IP.* Prentice–Hall, Englewood Cliffs, New Jersey, second edition, 1991.

Description of the specification of the TCP/IP protocols and their functionality.

[Com91b] Douglas E. Comer. *Internetworking with TCP/IP*, Volume II. Prentice–Hall, Englewood Cliffs, New Jersey, 1991.

Detailed specification of one implementation of the TCP/IP protocol.

[CS94] Robert G. Cole and David Shur. *IP over ATM: A Summary of Framework Discussions.* Internet Engineering Task Force, Reston, VA., November 1994. (working draft).

Summary of the discussion of the IP over ATM working group over the last several years, in particular subnet models, and end-to-end models are being discussed.

[Dat91] C. J. Date. *An Introduction to Database Systems*, Volume I. Addison–Wesley Publishing Company, Inc., fifth edition, 1991.

Textbook providing the basis for a solid education in the fundamentals of database technology.

[Ker94] Berry Kercheval. ATMARP: An Architecture Proposal. IP over ATM working group, March 1994.

A design proposal for RFC 1577.

[LTWW93] Will E. Leland, Murad S. Taqq, Walter Willinger, and Daniel V. Wilson. On the self-similar nature of Ethernet traffic. In *Proceedings of ACM SIGCOMM '93*, pages 183–193, September 1993. also in *Computer Communication Review* 23 (4), Oct. 1992.

Demonstrates that Ethernet local area network (LAN) traffic is statistically self-similar and that none of the commonly used traffic models is able to capture this fractal behavior.

[Par93] Craig Partridge. *Gigabit Networking*. Addison–Wesley Publishing Company, Inc., 1993.

A practical look at the advances that are making high–speed networking a reality.

[Qport94a] Bellcore. *Q.port Installation and User's Guide*. Bellcore, Bell Communications Research, Piscataway, NJ, January 1994.

Bellcore's proprietary ATM signaling code. – Installation and User's Guide.

[Qport94b] Bellcore. *Q.port Module Descriptions*. Bellcore, Bell Communications Research, Piscataway, NJ, January 1994.

Bellcore's proprietary ATM signaling code. – Module Descriptions.

[Qport94c] Bellcore. *Q.port Implementation Notes*. Bellcore, Bell Communications Research, Piscataway, NJ, January 1994.

Bellcore's proprietary ATM signaling code. – Implementation Notes.

[RFC 768] Jon Postel, editor. *RFC–768 User Datagram Protocol*. Network Information Center, August 1980.

The UDP specification.

[RFC 791] Jon Postel. *RFC–791 Internet Protocol*. Information Science Institute, University of Southern California, CA, September 1981.

The IP specification.

[RFC 793] Jon Postel, editor. *RFC–793 Transmission Datagram Protocol*. Information Sciences Institute, USC, CA, September 1981.

The TCP specification.

[RFC 826] David C. Plummer. *RFC–826 An Ethernet Address Resolution Protocol*. Network Working Group, November 1982.

The ARP specification. Question: Given a network protocol address: What is the associated MAC address?

[RFC 903] Ross Finlayson, Timothy Mann, Jeffrey Mogul, and Marvin Theimer. *RFC–903 A Reverse Address Resolution Protocol*. Network Working Group, June 1984.

The RARP specification. Question: Given a MAC address: What is the associated network protocol address?

[RFC 1042] Jon Postel and Joyce K. Reynolds. *RFC–1042 A Standard for the Transmission of IP Datagrams over IEEE 802 Networks.* Network Working Group, February 1988.

Standard method of encapsulating IP datagrams and ARP requests and replies on IEEE 802 networks.

[RFC 1293] Terry Bradley. *RFC–1293 Inverse Address Resolution Protocol.* Network Working Group, January 1992.

The InARP specification. Given an established connection, ask the peer entity for its network layer and MAC address.

[RFC 1483] Juha Heinanen. *RFC–1483 Multiprotocol Encapsulation over ATM Adaptation Layer 5.* Network Working Group, July 1993.

Describes two encapsulation methods for carrying network interconnect traffic over ATM AAL5.

[RFC 1577] Mark Laubach. *RFC–1577 Classical IP and ARP over ATM.* Network Working Group, January 1994.

Initial application of classical IP and ARP in ATM network environment configured as a Logical IP Subnetwork (LIS).

[RFC 1626] Randall J. Atkinson. *RFC–1626 Default IP MTU for use over ATM AAL5.* Network Working Group, May 1994.

Specifies the default value for IP MTU over AAL5 to be 9180 octets.

[RFC 1700] Jon Postel and Joyce K. Reynolds. *RFC–1700 Assigned Numbers.* Network Working Group, October 1994.

Status report on the parameters used in the Internet community.

[Suz94] Toshikazu Suzuki. ATM Adaptation Layer Protocol. *IEEE Communications Magazine*, pages 80–83, April 1994.

The paper provides a background for and describes ATM Adaptation Layer 5 in comparison to AAL 3/4.

[Tri82] Kishor S. Trived. *Probability & Statistics with Reliability, Queuing, and Computer Science Applications.* Prentice–Hall, Englewood Cliffs, New Jersey, 1982.

Textbook on Statistics.

[UNI 3.0] ATM Forum. *ATM User–Network Interface Specification, Version 3.0.* Prentice–Hall, Englewood Cliffs, New Jersey, 1993.

Interim standard of ATM defined by the user and vendor group known as the ATM Forum.

A From PDUs to Cells

In this section we explain the necessary encapsulations and modifications for the conversion of IP datagrams to ATM cells. Figure A.1 sketches the different stages of this process: LLC encapsulation (sections A.1 and A.2), adaptation layer convergence (section A.3.1), and segmentation and reassembly (section A.3.2). It is important to indicate that we concentrate on the adaptation of the TCP/IP protocol suite to ATM. Multiprotocol encapsulation and convergence are defined for a variety of other protocols and types of applications, which is beyond the scope of this paper.

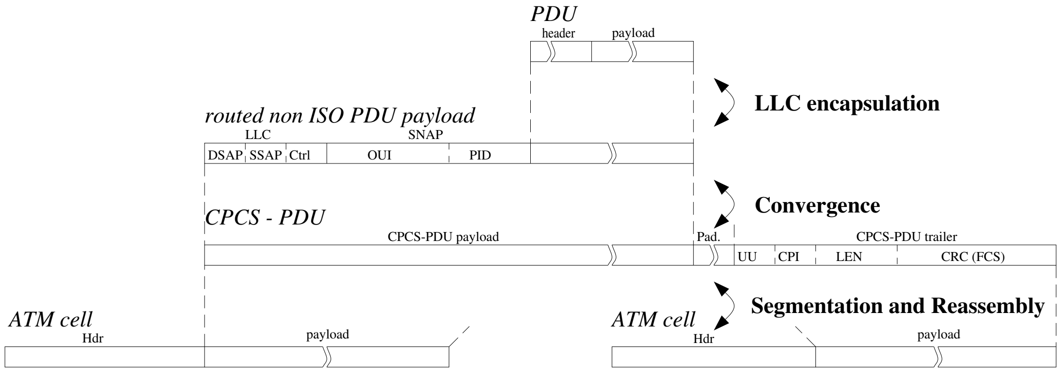

Fig. A.1. The transformation from a network layer PDU to ATM cells

As mentioned above, the basic unit of data transmission in the TCP/IP protocol suite is called a datagram. [RFC 791, section 3.1] defines the contents of datagram headers (20 – 60 octets long) and the possible length of its protocol data unit (up to $2^{16} - 1$ octets incl. header). [RFC 791] also standardizes the fragmentation and reassembly of datagrams within the IP layer.

ATM cells are 53 bytes long. The first 5 bytes are header information. They contain the VC identifier that is used in ATM switches, across *user network interfaces* (UNI), and *network network interfaces* (NNI) for routing. Additionally, cell headers contain a payload type identifier, a priority bit, and a checksum, called *header error control* (HEC). Cells that cross the UNI also contain generic flow control information. The remaining 48 bytes are called the data portion or payload. Cells are rather small compared to IP datagrams. ATM cell layout is defined in [UNI 3.0].

A.1 Multiprotocol Encapsulation

[RFC 1483] describes two methods for carrying routed and bridged *protocol data units* (PDU) over an ATM network. In contrast to [RFC 1483] we will not concentrate on the aspects of bridging. PDUs can be carried over a single ATM VC if they contain additional protocol identifiers. This first approach is called *LLC encapsulation*, because the respective PDUs are prefixed by an IEEE 802.2 LLC header and its possible extensions. The second approach is called *VC based multiplexing*. It does higher–layer protocol multiplexing by using a separate VC for each different protocol.

We chose LLC encapsulation, because we preferred not to establish several different VCs between the same pair of hosts. However, we acknowledge the advantage of VC based multiplexing if the dynamic creation of large numbers of ATM VCs is economical and the applied charging model does not excessively depend on the number of simultaneous VCs.

A.2 LLC Encapsulation

LLC encapsulation applies to different types of data communication services: Type 1, an unacknowledged connectionless mode, Type 2, a connection–oriented mode, and a hybrid Type 3, a semi–reliable mode. We consider network layer protocols that operate over LLC Type 1.

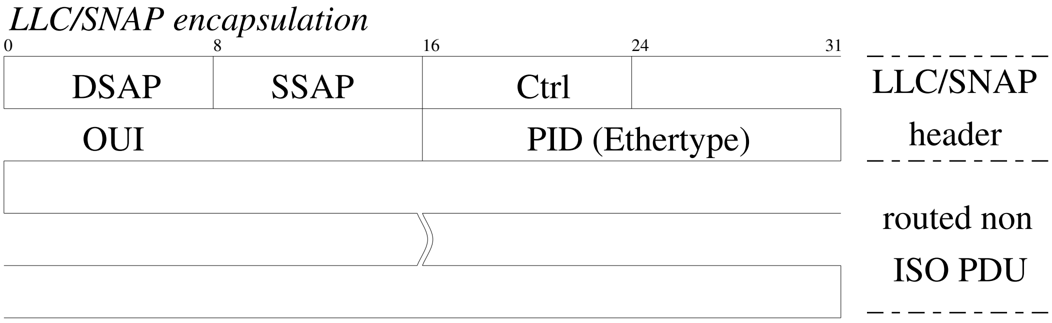

Fig. A.2. LLC/SNAP encapsulation of routed non ISO PDU

The prefixing IEEE 802.2 LLC header consists of three one octet fields. It is possibly followed by an IEEE 802.1a *subnetwork attachment point* (SNAP) header. Figure A.2 depicts the layout of an IP datagram (routed, non ISO PDU) prefixed by its LLC/SNAP header. This format is compatible with the header format specified in [RFC 1042], the specification of encapsulating IP datagrams and ARP requests and replies on IEEE 802 networks. The value `0x03` in the LLC Control field specifies an unnumbered information command PDU. A LLC header value of `0xAA-AA-03` indicates the presence of a SNAP header. A SNAP header contains 2 fields: A three octet *organizationally unique identifier* (OUI) and a two octet *protocol identifier* (PID). A value of `0x00-00-00` for the OUI specifies that the following PID is an ethertype. Valid values for the PID are defined in [RFC 1700, pp.168]. We will use only the values for IP datagrams (`0x0800`) and for ARP frames (`0x0806`).

The use of IP and ARP must be only consistent within its LAN type. It is not necessary that it be consistent across all other types of subnetwork technologies.

A.3 The Adaptation Layer

ATM networks are not designed to exclusively transmit connectionless data packets such as IP datagrams. Several kinds of higher level data (such as datagrams, voice samples, or video frames) must be packaged into ATM cells efficiently. This process is referred to as adaptation. It is performed by an adaptation layer that is conceptually located between IP and ATM. It consists of a convergence sublayer, and a segmentation and reassembly sublayer.

A.3.1 Convergence

Four classes of applications were determined by the former CCITT that would require different types of service. *ATM adaptation layers* (AALs) are optimized for their associated class of applications. Depending on which class of application is used, the convergence sublayer performs functions such as multiplexing, cell loss and error detection, or timing recovery. AAL5 is the *ATM adaptation layer 5*, designed for connectionless data applications. It provides the most efficient and functional convergence layer for this class of applications: it is optimized to introduce little header overhead, to require little cell handling cost in host interfaces, and to resemble common data communications interfaces, such as those for Ethernet.

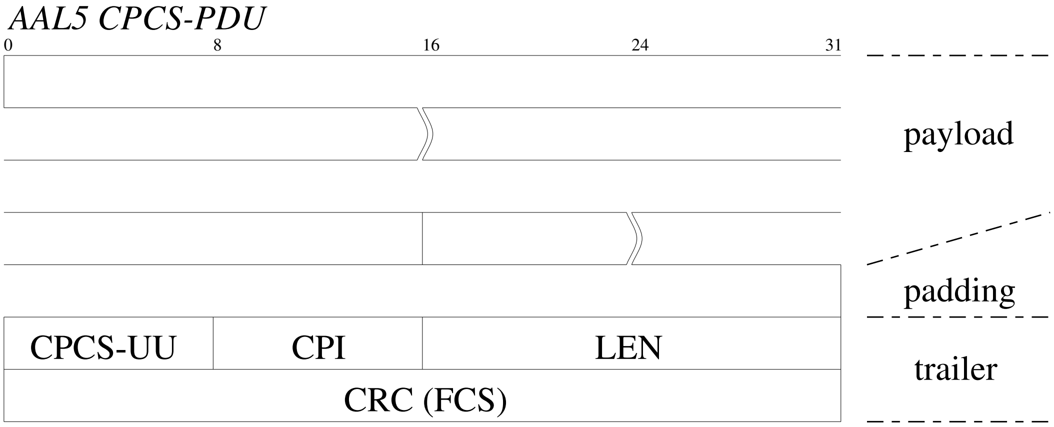

Fig. A.3. AAL5 CPCS–PDU

The convergence sublayer is further subdivided into an upper and lower half, the *service specific convergence sublayer* (SSCS), and the *common part convergence sublayer* (CPCS), respectively. We assume a null SSCS, because the

CPCS functions provide sufficient processing for network and transport layers such as TCP/IP.

The AAL5 frame format is specified in [RFC 1483, section 3] and depicted in Figure A.3. AAL5 appends a trailer to the CPCS *protocol data unit* (PDU). See [RFC 1626] for a discussion on the default IP *maximum transfer unit* (MTU) for use over ATM AAL5. The carried PDU is padded such that the total number of octets of the frame is a multiple of the number of octets in the ATM payload (48 octets).

The fields CPCS–UU (*user–to–user* indication) and *common part indicator* (CPI) are both one octet long. Users can transparently transfer user to user information in the CPCS–UU field. It can contain any value. The CPI field has currently no function except for 64–bit alignment of the AAL5 trailer. It contains a null value. The length field determines the length of the payload field in octets. The *cyclic redundancy check* (CRC), also referred to as *frame check sum* (FCS), protects the entire PDU except the CRC field itself.

A.3.2 Segmentation and Reassembly

We mentioned above that the carried PDU is padded by null characters. Let $|name|$ denote the length of entity *name* in octets. The number of octets P required for padding ($P \in [0, |\text{ATM payload}| - 1]$) is determined by formula (A.1).

$$\begin{aligned}\underline{|\text{CPCS}-\text{PDU payload}| + P + |\text{CPCS}-\text{PDU trailer}|} &= \\ |\text{LLC/SNAP hdr}| + |\text{PDU}| + P + 8 &= \\ 8 + |\text{PDU}| + P + 8 &= k \cdot 48 \\ &= \underline{k \cdot |\text{ATM payload}|}\end{aligned}$$

$$\Longrightarrow \quad P = ((|\text{PDU}| + 16 + 47) \text{ div } 48) \cdot 48 - |\text{PDU}| - 16 \tag{A.1}$$

The *segmentation and reassembly* (SAR) sublayer is responsible for breaking the CPCS–PDU into cells at the sender side and reassembling cells into frames at the receiver side. One of the features of connection–oriented transmission technologies is that data is delivered in sequence. Therefore it is sufficient to flag the final cell of each frame to signal its completion of reassembly. This flag is the user signaling bit, the last bit of the payload type identifier.

B Glossary

AAL5	ATM Adaptation Layer 5
ARP	Address Resolution Protocol
ATM	Asynchronous Transfer Mode
ATMARP	ATM Address Resolution Protocol
B–ISDN	Broadband Integrated Services Digital Network
CCITT	International Telegraph and Telephone Consultative Committee
COAST	Computer Operations, Audit and Security Technology
CPCS	Common Part Convergence Sublayer
CPCS–UU	CPCS User–to–User indication
CPI	Common Part Indicator
CRC	Cyclic Redundancy Check
CSL	Computer Science Laboratory
CSMA/CD	Carrier Sense Multiple Access w/ Collision Detection
DLL	Data Link Layer
DQDB	Distributed Queue Dual Bus
FCS	Frame Check Sum
HCC	Host Call Control
HEC	Header Error Control
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IP	Internet Protocol
ISDN	Integrated Services Digital Network
ISO	International Standards Organization
InARP	Inverse Address Resolution Protocol
InATMARP	Inverse ATM Address Resolution Protocol
LAN	Local Area Network
LATM	Local Area Network ATM
LIS	Logical IP Subnetwork
LLC	Logical Link Control
MAC	Medium Access Control
MTU	Maximum Transfer Unit
NF	Normal Form
NNI	Network Network Interface
OSI	Open Systems Interconnection
OUI	Organizationally Unique Identifier
PARC	Palo Alto Research Center
PDU	Protocol Data Unit
PID	Protocol Identifier
PVC	Permanent Virtual Circuit
QoS	Quality of Service
RARP	Reverse Address Resolution Protocol
ROLC	Routing Over Large Clouds

SAR	Segmentation and Reassembly
SCC	Switch Call Control
SNAP	Subnetwork Attachment Point
SSCS	Service Specific Convergence Sublayer
SVC	Switched Virtual Circuit
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UNI	User Network Interface
VC	Virtual Circuit
VCI	Virtual Circuit Identifier
VPI	Virtual Path Identifier
WAN	Wide Area Network
WATM	Wide Area Network ATM