Report Number: WUCS-94-21

1994-01-01

CMAP

Ken Cox and John DeHart

This document specifies a Connection Management Access Protocol (CMAP) for call management in high-speed packet switched networks. We target CMAP to networks employing the Asynchronous Transfer Mode (ATM) communication standard. CMAP specifies the access procedues exercised by network clients to manipulate multipoint calls; it is thus a User-Network Interface (UNI) signalling protocol. We define a multipoint call as a group of multipoint connections. A multipoint connection is a communication channel between two or more clients or endpoints of the network, where all data sent by one client is received by all other clients who have elected to receive. A... **Read complete abstract on page 2.**

CMAP

Ken Cox and John DeHart

Complete Abstract:

This document specifies a Connection Management Access Protocol (CMAP) for call management in high-speed packet switched networks. We target CMAP to networks employing the Asynchronous Transfer Mode (ATM) communication standard. CMAP specifies the access procedues exercised by network clients to manipulate multipoint calls; it is thus a User-Network Interface (UNI) signalling protocol. We define a multipoint call as a group of multipoint connections. A multipoint connection is a communication channel between two or more clients or endpoints of the network, where all data sent by one client is received by all other clients who have elected to receive. A point-to-point connection is a special case of a multipoint connection involving only two clients. CMAP provides facilities to create, modify, and delete calls, connections, and endpoints. Once a connection is established, clients exchange data using ATM data-transfer protocols that are specified separately from CMAP.

CMAP

WUCS-94-21

Authors:

Version 3.0 (ARL-94-08):

Ken Cox

John DeHart

Previous Versions (WUCS-92-01, ARL-89-06) authored by:

John DeHart

Mike Gaddis

Rick Bubenik

July, 1994

Department of Computer Science

Washington Univeristy

Campus Box 1045

One Brookings Drive

St. Louis MO 63130-4899

Connection Management Access Protocol (CMAP) Specification

Ken Cox and John DeHart

Version 3.0
July 11, 1994

Applied Research Laboratory Working Note —**ARL-94-08**
Department of Computer Science Technical Report —**WUCS-94-21**

Applied Research Laboratory
Department of Computer Science
Washington University
Campus Box 1045
One Brookings Drive
St. Louis, Missouri 63130-4899
Telephone: 314-935-6160
FAX: 314-935-7302
Email: {jdd, kcc}@arl.wustl.edu

Versions Prior to 3.0 authored by:
John DeHart
Rick Bubenik
Mike Gaddis

Abstract

This document specifies a *Connection Management Access Protocol* (CMAP) for call management in high-speed packet switched networks. We target CMAP to networks employing the *Asynchronous Transfer Mode* (ATM) communication standard. CMAP specifies the access procedures exercised by network *clients* to manipulate multipoint calls; it is thus a *User-Network Interface* (UNI) signalling protocol. We define a *multipoint call* as a group of *multipoint connections*. A *multipoint connection* is a communication channel between two or more clients or *endpoints* of the network, where all data sent by one client is received by all other clients who have elected to receive. A *point-to-point connection* is a special case of a multipoint connection involving only two clients. CMAP provides facilities to create, modify, and delete calls, connections, and endpoints. Once a connection is established, clients exchange data using ATM data-transfer protocols that are specified separately from CMAP.

Hypertext Linkage

Previous Link

Master Index
Document Index

Title | Contents | Figures | Quit
Tables | Body | Index

Author: **Ken Cox and John DeHart**
Organization: **Applied Research Laboratory**
Project: **Zeus Project**
File: **/project/gbn/switch/sw/documentation/CM/CMAP/CMAP_hdr.frm.JDD**
Created: **February 26, 1992**
Modified: **July 11, 1994 10:15 am by kcc**

Version Information

Version	1.0:	Initial document containing CMAP message formats and little else.
Version	2.0:	Added introductory sections on ATM networks, the call model and the protocol stack (these sections were derived mostly from papers published by the authors). Changed many of the CMAP message formats and added some new messages. Added detailed explanations where deemed necessary to highlight obscure CMAP features. Added implementation and future direction sections (incomplete).
Version	2.1:	Major editing rewrite of Sections 1 through 8.5 from version 2.0 (now organized as Sections 1 through 6). More editing to come.
Version	2.1.1:	Minor editing changes from version 2.1. More editing to come (temporarily on hold).
Version	3.0:	Major editing involving restructuring of document, removal of redundacies, and general modifications/clarifications preparatory to implementation. More editing expected as implementation proceeds. Summary of major changes:

- Restructuring of document as a FrameMaker "book" with table of contents, lists of figures and tables, and index.
- Collection of several sections from earlier versions into a single chapter describing the environment in which CMAP operates.
- Expansion and clarification of the call model, including: separation of ownership and root functions; owner now need not be a participant in call; more complete listing of call, connection, *etc.* parameters; clarification of the distinction between clients and endpoints and between connections and UNI parameters, and more consistent use of the terms throughout document; modification of the system for connection and endpoint mappings.
- Collection of information about messages into a single chapter, including: explanation of use of messages; definitions of ACKs, NACKs, *etc.*; redefinition of message objects to support multi-drop/surrogate signalling and other changes to model; complete definitions of all message fields.
- Addition of commands to support client/network interfacing (*e.g.*, **status**, **alert**).
- Renaming of many commands to better reflect their use (*e.g.*, all notifications now begin with **announce_**).
- Rewriting of command explanations, including: explanation of message traffic produced by command; parameter negotiation in command; state machines modified to reflect separation of call ownership from participation in call and other changes to model.
- Expansion of examples, and linkage to actual applications.
- Wrote a preliminary version of the "Future Directions" section.
- Modification of appendices to reflect changes elsewhere in document.
- Unification of the state machines for the various operation types.

Copyright Notification

Table of Contents

List of Figures

List of Tables

1. Introduction

This document describes a *call* model for multipoint connections in a switched Asynchronous Transfer Mode (ATM) networks and specifies the Connection Management Access Protocol (CMAP) that allows network *clients* to create, manipulate and delete multipoint, multiconnection communication channels, which we term *calls*. A *multipoint call* is a call involving two or more clients; a *point-to-point* call is a special case of a multipoint call involving only two clients. Data sent over a connection by one participant in a call is received by all other participants electing to receive on this connection, although reliable delivery is not guaranteed by the network. Calls are allowed to change dynamically during their lifetime, in terms of the number of participants, the number of connections and the reserved bandwidth of the connections.

CMAP defines the interface between clients and the network used to create, manipulate and delete calls. As such, CMAP is an ATM *User Network Interface* (UNI) signalling protocol [3, 14]. It is layered over a reliable substrate, which we term CTL (for CMAP Transport Layer). We do not specify the CTL protocol. Rather, we list the requirements for CTL, which are generally met by several existing transport protocols (for example, TCP/IP over type 4 AAL). In this way, CMAP implementors can choose the most suitable CTL for their implementation environment.

Although CMAP and the associated call model provide a rich set of operations and capabilities, CMAP implementations are not required to be complete: CMAP implementations can omit certain operations and capabilities, while still adhering to the general message layouts and request/acknowledgment handshake procedures. A certain set of core capabilities are required, which we term *minimal* CMAP (Section 3.6). All other capabilities are optional.

CMAP also exhibits flexibility in the areas of addressing and data transfer. CMAP does not dictate any one addressing scheme. Rather, CMAP supports multiple addressing disciplines and multiple routing protocols. Currently-defined addressing schemes include IP (Internet Protocol) addressing [18], public network E.164 addressing [13], and OSI NSAP addressing [17]. An implementation of CMAP may support any or all of these schemes, as well as others. Similarly, CMAP does not require clients to use a particular data transfer protocol (*e.g.*, AAL-5) on the connections it creates. Instead, CMAP supplies raw ATM connections on which clients may layer any protocol they find appropriate.

The remainder of this document is organized as shown in Table 1.

Table 1. Document Structure

Section and Title	Description
Section 1: **Introduction**	Call model and CMAP overview.
Section 2: **Switched ATM Networks**	Introduction to switched, connection-oriented ATM networks.
Section 3: **CMAP Network Environment**	Description of CMAP layering over ATM networks.
Section 4: **Call Model**	Detailed description of the call model.
Section 5: **CMAP Messages**	General information on message use, formats and transmission.
Section 6: **CMAP Operations**	Detailed description of all CMAP operations.
Section 7: **Examples**	Examples of CMAP operations, including several types of call setup.
Section 8: **Future Directions in CMAP**	List of CMAP enhancements being considered.
Appendix A: **References**	List of related references.
Appendix B: **Acronym List**	List of acronyms used in document.
Appendix C: **CMAP Message Field Values**	Numerical values for CMAP message parameters.
Appendix D: **CMAP Status Codes**	Numerical values and actions for CMAP error codes.
Appendix E: **Endpoint Mappings**	Tabular list of all possible endpoint receive/transmit mappings.
Appendix F: **Parameter Negotiation**	Summary of parameter negotiation.

2. Switched ATM Networks

This section presents an overview of the switched ATM network architectures, the ATM standard, ATM network connections, and the two types of ATM connections (*Virtual Path* and *Virtual Channel*).

2.1 Network Architecture

An example ATM network is shown in Figure 1. The network consists of *clients, exterior nodes* (nodes that interface to clients), and *interior nodes* (nodes that interface to other nodes only), all interconnected by *fiber optic links*. Clients signal the network to set up connections to other clients by signalling exterior nodes using a *User Network Interface* (UNI) signalling protocol [3, 14]. The exterior nodes perform the requested operation, communicating if necessary with other nodes (both interior and exterior) using a *Network Node Interface* (NNI) signalling protocol.

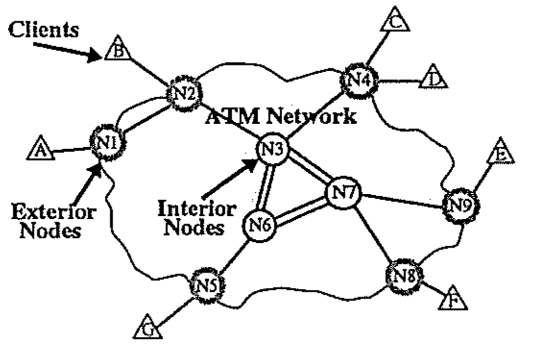

Figure 1. Example ATM Network

Clients may send ATM cells (packets) to the network and receive cells from the network along their fiber links. Each node in the network contains one or more ATM *switches* [29, 33, 34, 36, 37, 59, 62]. The switches route each ATM cell to the desired destination link(s) based upon header fields in the cell (Section 2.3). In order to keep up with line speeds, the switches perform all routing in hardware. Since the time interval within which each cell must be routed is very small, tables in the switches are preconfigured with routing information. This makes ATM networks more suitable for connection-oriented traffic, where the switch tables can be configured during connection setup. Connectionless traffic can be accommodated via *overlay* networks utilizing special-purpose routers or datagram processors [5, 9].

Figure 2 shows the architecture of one ATM switch, Turner's *Broadcast Packet Switch* [59, 62]. This switch contains a *Copy Network* (CN) concatenated to a *Routing Network* (RN). ATM cells (packets) enter the switch on the left. Multicast cells, destined for several locations, are replicated by the CN, then routed to the appropriate destination by the RN. Point-to-point cells follow an arbitrary path through the CN (following the path of "least resistance"), then are routed by the RN. Cells leave the switch on the right, where they traverse fiber optic links to other switches or clients.

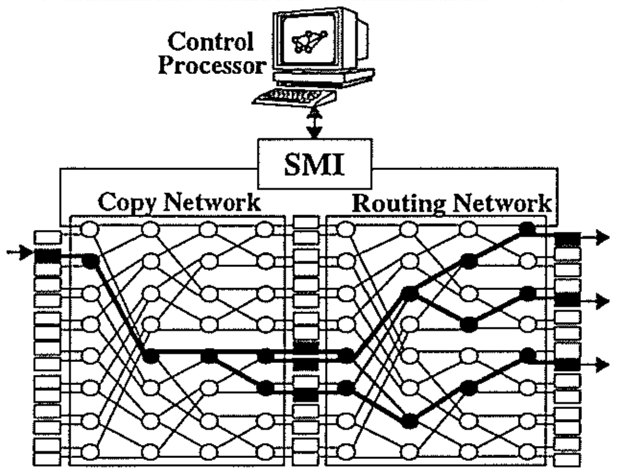

Figure 2. Architecture of Turner's Broadcast Packet Switch

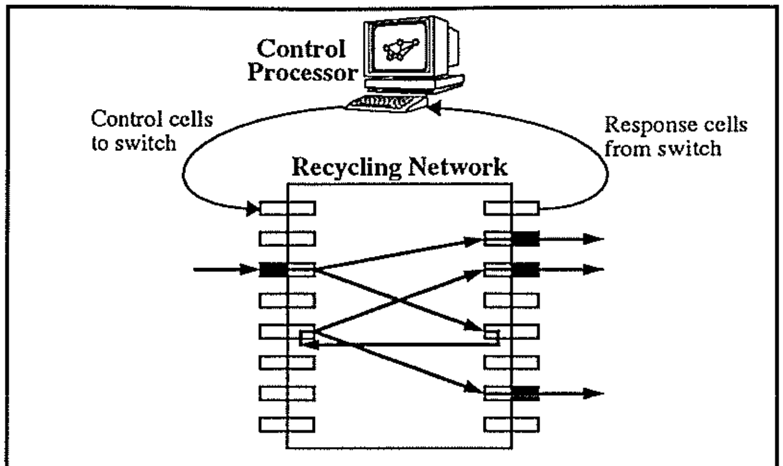

Figure 3. Architecture of Turner's Gigabit Recycling Switch

The broadcast packet switch is controlled by the *Control Processor* (CP) connected to the switch via an Ethernet-based *Switch Module Interface* (SMI). The CP configures the switch hardware to route incoming cells to the appropriate outgoing links by modifying tables within the switch, thus establishing connections.

Figure 3 shows the architecture of another ATM switch, Turner's *Gigabit Recycling Switch* [59, 62]. The internal structure of this switch differs considerably from that of the Broadcast Packet Switch—on each pass through the network at most two copies of a cell are produced, with larger numbers of copies obtained by recycling cells through the network—but the external behavior is the same. ATM cells enter the switch on the left, are replicated and routed as specified by internal tables, and leave the switch on the right. The recycling switch is also controlled by a CP, but instead of using an SMI the CP sends control information over the fiber links in the form of special control cells.

Figure 4 illustrates our concept of a *node*, where more than one ATM switch is under the control of a single CP. The CP manages routing for all the switches in the node, sending commands to update the tables in each switch. We assume that the CP is directly connected to (at least) one *main* switch. Its connections to the other, *satellite* switches may be indirect, for example, it may have to send commands in the form of ATM cells over the fiber links connecting the switches within the node. Note that, from the "outside", a node can be viewed as a single large switch through which connections may be routed, with the routing inside the node hidden.

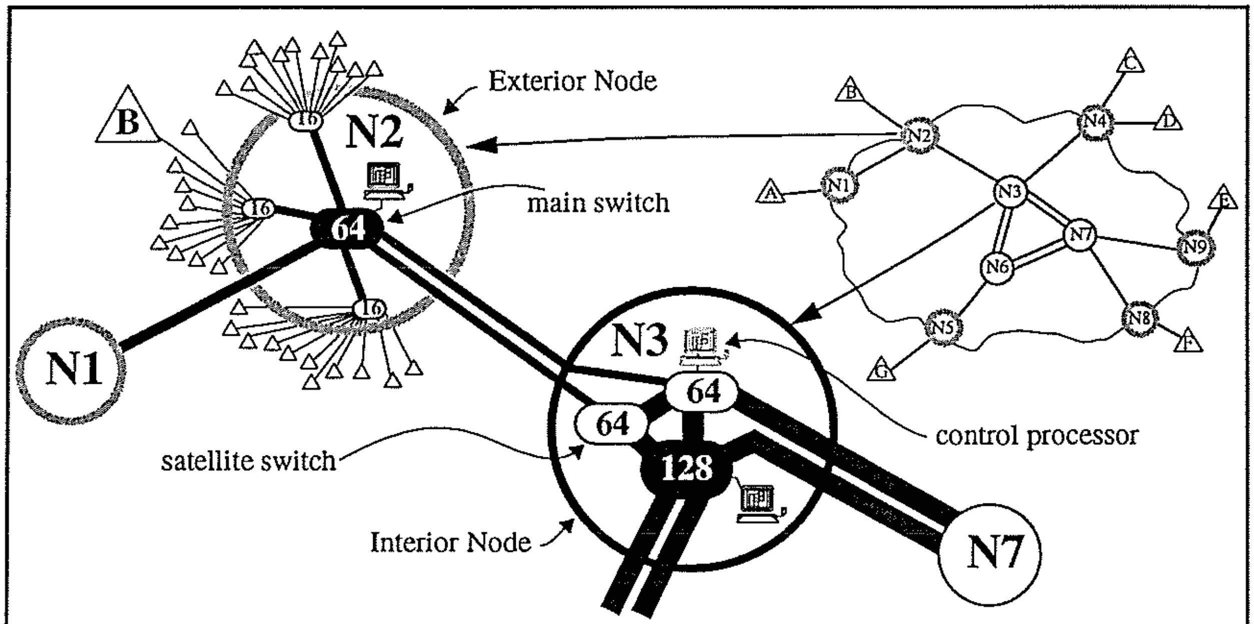

Figure 4. Generalized Node Architecture for Interior and Exterior Network Nodes

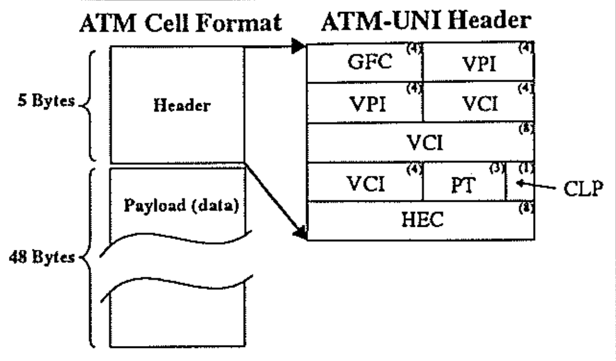

Figure 5. ATM UNI Cell Format.

2.2 The ATM Standard

The emerging ATM standard [3, 14] specifies link-level cell formats for two interfaces: 1) the User Network Interface (UNI), for communication between the client and the network, and 2) the Network Node Interface (NNI), for communication between network nodes. The ATM UNI cell format is shown in Figure 5. It consists of a 48-byte payload (data) field and a five-byte header. The header has six fields: a Global Flow Control (GFC, 4 bits), a Virtual Path Identifier (VPI, 8 bits), a Virtual Channel Identifier (VCI, 16 bits), a Payload Type (PT, 3 bits), a Cell Loss Priority (CLP, 1 bit) and a Header Error Check (HEC, 8 bits).

Use of the GFC has not yet been standardized, although the intent is to use this field for arbitration on shared media access links (such as DQDB). The VPI and VCI fields are used to route cells, as described in the next section. The CLP bit is used to mark low priority cells, where CLP=1 indicates low priority. This bit may be set either by clients or the network. The last header field, the HEC, is a cyclic redundancy check (CRC) on the header.

The three-bit PT field is used to distinguish data cells from other cells. There are four types of client data cells, all with the most-significant PT bit set to 0. The other two bits are the network congestion bit (the middle bit) and the tagged data bits (the least-significant bit). The network congestion bit is used by the network to inform clients receiving a cell that congestion was encountered somewhere in the network. The tagged data PT marking can be used by the client to differentiate cells. This marking is preserved by the network. One potential use is in delineating segmented frames, where the cell containing the last fragment of the frame is tagged and all other frames are untagged [9, 42]. An example of this is shown in Figure 6 for IP datagram frames. The remaining four PT values (those with the most-significant bit equal to 1) are reserved for network control and resource management functions.

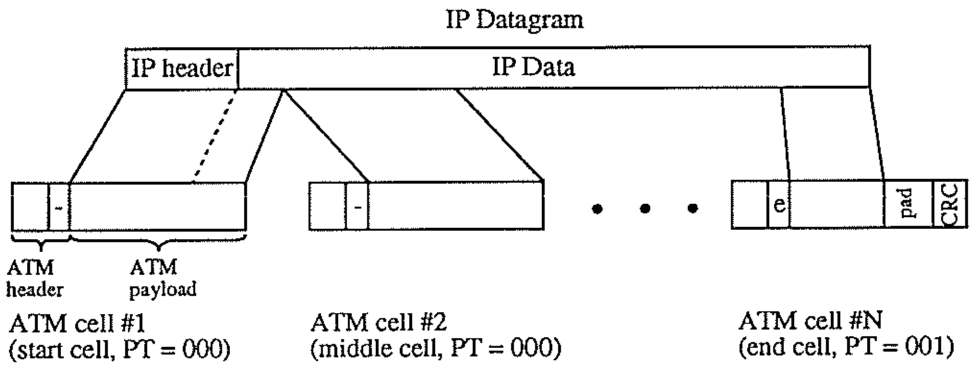

Figure 6. IP Segmentation/Reassembly Scheme.

2.3 Virtual Path and Virtual Channel Connections

The VPI and VCI header fields are used to route cells. The ATM standard provides for two types of routing: *Virtual Path* (VP) and *Virtual Channel* (VC). Both types of routing are based on an abstraction of ATM networks which we call *cell pipes* or *connections*. Clients of ATM networks communicate over cell pipes by sending and receiving

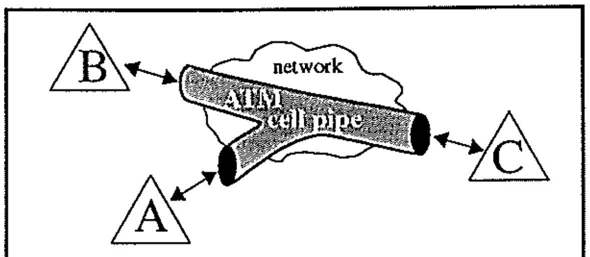

Figure 7. Three-way Cell Pipe

ATM cells. In the general case a cell pipe may be *n*-way (any number of clients), bidirectional (a client may both send and receive data on the same pipe), and multipoint-to-multipoint (all clients may send data, and all clients connected to the cell pipe will receive the data). Figure 7 shows the conceptual view of a three-endpoint cell pipe between clients A, B and C. All data sent by one client is received by all other clients who have elected to receive on this cell pipe.

Clients access cell pipes using the VPI and VCI fields of the header. To be more precise, clients place particular values in the VPI and VCI fields of each ATM cell that they send. The network uses these values to route the cells to the other clients that are using the same cell pipe. The routing tables in the network switches are configured with mapping information which, for each incoming VPI/VCI combination, indicates to which switch output line(s) the cell should be sent and what VPI/VCI pairs should be written into the headers of the output cells. The routing process thus could remap the VPI/VCI pair at each switch in the network.

In a virtual path connection, the network uses the VPI for routing, possibly remapping this field at every switch within the network. The VCI field is transmitted end-to-end unchanged by the network and is available for use by clients. Thus, in VP connections clients route the cell using the VPI field and may place any value in the VCI field. One anticipated use of the VCI can be used for source discrimination in multipoint connections, where each transmiting client places a unique VCI in all of its outgoing cells.

In a virtual channel connection, the network uses both the VPI and VCI for routing, possibly remapping both fields at every switch within the network. Clients route the cell using both the VPI and VCI fields; neither field is available for other purposes such as multiplexing. VC connections are desirable for connections that do not need source discrimination, and for connections that want to take advantage of rapid setup (where the network is able to reduce connection setup overhead by using preconfigured trunks).

Figure 8 shows an expanded view of the cell pipe from Figure 7. This cell pipe has been set up as a VP connection. Client A accesses the cell using VPI 9, client B uses VPI 3, and client C uses VPI 7. Any cell transmitted by one of these clients on the cell pipe is delivered to the other two with its VCI field unchanged. When a client is receiving from two or more transmitters, the cells from the two sources may be interleaved in an arbitrary fashion. However, the ATM standard guarantees that cells from each transmitter will be received in the order that they were sent.

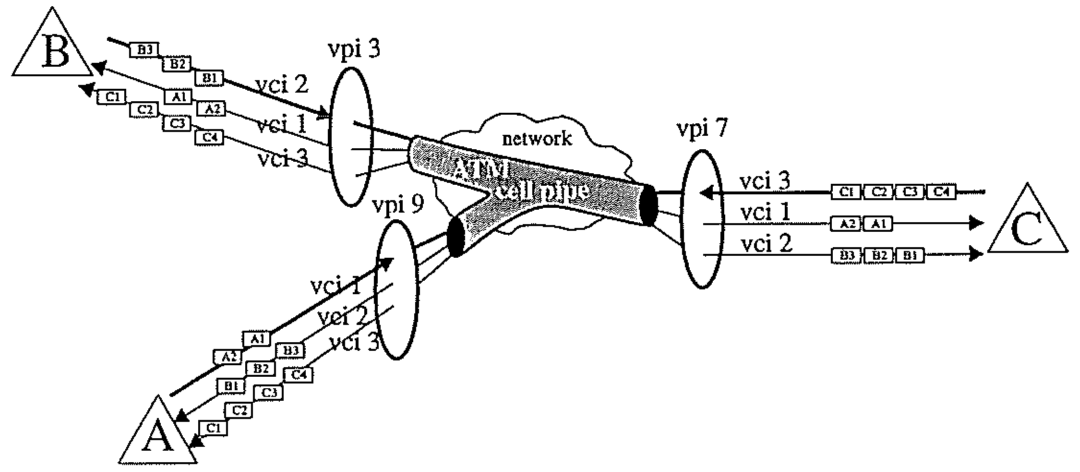

Figure 8. Multipoint VP Connection Using VCI for Source Discrimination

In this example, the clients are using the VCIs for source discrimination. Client A is setting the VCI of cells it transmits to 1, client B to 2, and client C to 3. The receivers can thus distinguish the source of each cell received, even though cells from two or more different transmitters may be interleaved on receipt. This technique would not be possible in a VC connection; therefore, if a VC connection is used for multipoint communication, higher-level protocol information must be embedded in the ATM cell payload for source discrimination.

ATM cell pipes in a network are set up, modified, and torn down under the control of software which arranges that appropriate values be stored in the routing tables of each switch involved in the connection. CMAP forms one component of this software. The next chapter examines CMAP's role in network routing in greater detail.

3. CMAP Network Environment

This section discusses how CMAP is used in a networking environment. Section 3.1 discusses the role of CMAP in network operations. The next three sections (3.2 through 3.5) describe the support provided by the network and by other software components. Finally, Section 3.6 defines complete and minimal CMAP implementations and indicates what facilities are necessary for a minimal implementation.

3.1 CMAP as a Session Management Protocol

Figure 9 shows the network process and protocol architecture used with CMAP. We believe that this abstract view of the network encompasses most actual and proposed network management implementations and is thus not unduely restrictive.

Management of ATM connections is encapsulated in a Connection Management Layer (CML), which provides facilities for building and destroying ATM connections. Clients have no knowledge of or direct access to the CML. Instead, clients interact with the Session Management Layer (SML), which provides a "higher-level" interface to the network resources. The Session Managers (processes of the SML) use the facilities provided by the CML to perform the network operations requested by clients. A network may support several different types of Session Manager, each providing a distinct client interface to the network. Contention for network resources by the Session Managers is resolved at the CML.

CMAP is a client interface protocol based on the creation, manipulation, and deletion of *calls* (Section 4). CMAP clients use a reliable ATM transport protocol (generically called CTL, or CMAP Transport Layer) to send messages or *signals* to the CMAP Session Managers, which then perform the requested network operations. CMAP is designed to support call management in a network supporting dynamic n-way multipoint-to-multipoint bidirectional connection-oriented ATM communications. CMAP clients are allowed to select connection types (virtual channel or virtual path), connection bandwidth, and the VPI/VCI pairs that they will use to transmit and receive. A complete CMAP implementation requires that all these facilities be available either from the network or from the other software components. The next three sections describe the support required from each component.

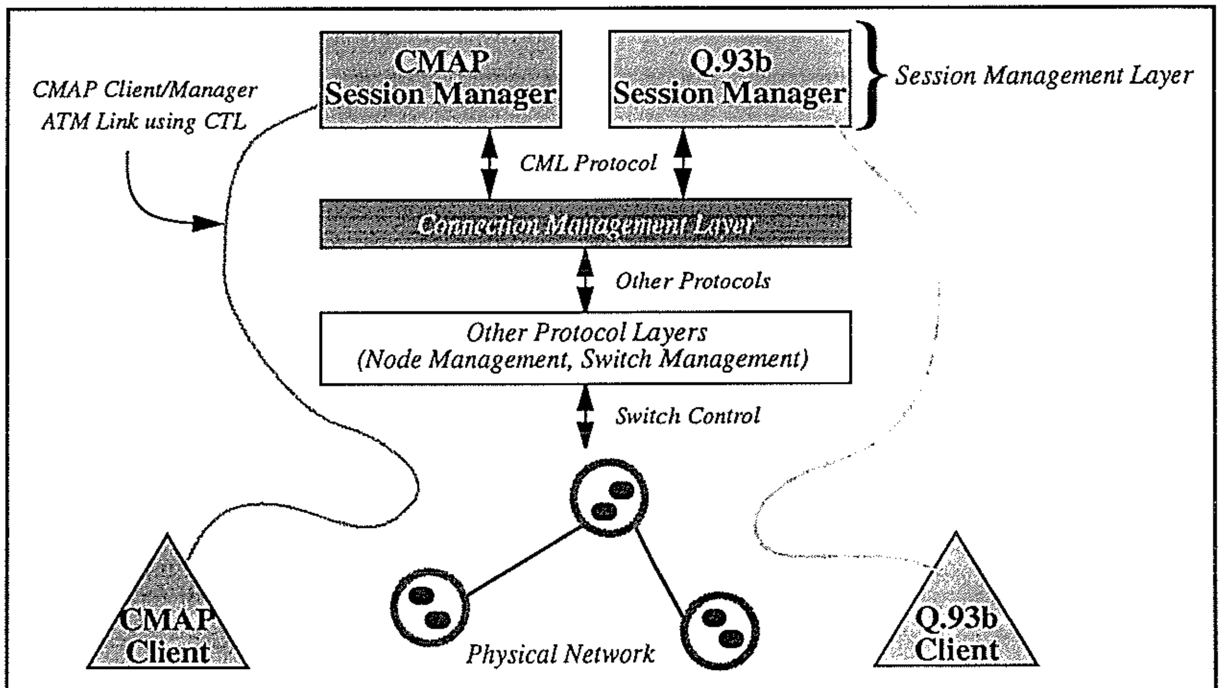

Figure 9. Network Configuration Used with CMAP

3.2 Network Functionality

Despite its central role, the physical network actually has to supply very little direct support for CMAP. This is primarily due to the fact that CMAP is situated at a high level on the protocol stack and is thoroughly insulated from the network by the other software components. The network facilities are primarily those needed to support the types of calls and connections that CMAP allows. In addition, many of the network requirements (*e.g.*, sequenced delivery of cells with no cell duplication or extraneous cells) are already required by the ATM standard and so need not be repeated here.

The network requirements are summarized in Table 2. The third column indicates whether the facility is required for a minimal CMAP implementation or optional. Most of the entries are self-explanatory, with the possible exception of the last four. By a static connection we mean one that is configured exactly once, when it is created, and cannot be changed thereafter. A dynamic connection can be changed while it is in use, for example to add or remove clients or to change bandwidth. An endpoint mapping (Section 4.5.3) determines whether a client receives and/or transmits data on a connection. A static mapping is one that is configured exactly once and not changed thereafter, while a dynamic mapping may change. As the table indicates, we require that the network allow clients to modify their mappings—for example, a client may indicate that it does not wish to receive data from a connection, and the network must stop delivering data to that client.

Table 2. Required and Optional Network Functionality

Network Functionality	Description	Required/ Optional
VP	Virtual Path connections	Optional
VC	Virtual Channel connections	Required
Point-to-point	Connections between exactly two clients	Required
Point-to-multipoint	Connections between two or more clients, only one transmitter	Optional
Multipoint-to-multipoint	Connections between two or more clients, any number of transmitters	Optional
Unidirectional	One-way data flow in connections	Required
Bidirectional	Two- or more-way data flow in connections	Optional
Static connections	Connection paths may not be reconfigured once set up	Required
Dynamic connections	Connection paths may be reconfigured once set up	Optional
Static mappings	Endpoint mappings may not be reconfigured once set up	Required
Dynamic mappings	Endpoint mappings may be reconfigured once set up	Optional

3.3 CTL Functionality

Table 3 lists the functions required of the CMAP Transport Layer (CTL), which is the protocol used to transmit CMAP signals over the ATM links between the CMAP clients and Session Managers. We do not specify a particular protocol but allow any ATM-compatible reliable transport protocol to be used. The table lists both required and optional CTL functions. The required functions are needed for correct message transmission and CMAP operation. The optional functions can be used to augment client interfaces and applications. We do not describe these augmentations in this document.

The first five functions deal with the mechanism whereby signals are broken down into ATM cells, transmitted, and reassembled into CMAP signals. As the table indicates, we require that this process guarantee that messages (CTL frames) be delivered to the remote peer without error. The remaining, optional functions may require further explanation. *Flow control* indicates that the CTL protocol will voluntarily refrain from sending messages when the network or host is congested, thus allowing time for the congestion to clear. *Internal ping* provides a mechanism to determine whether the remote peer is still alive. This would allow clients and CMAP Session Managers to periodically check the status of their peer and (if appropriate) attempt to restart the peer when it dies. *Auto synchronization* performs the internal ping automatically within the CTL layer, with notifications to the higher layers when synchronization is lost—again, the higher layer (CMAP client or Session Manager) might attempt a restart of its peer under these conditions.

Multiple connections on a single access link allow for multiple signalling connections to a single client and for multiple clients on an access link. The use of such connections is described in greater detail below (Section 3.4). *Query* allows for client agents and Session Managers to learn the capabilities of the network and adapt to them. For example, an application could query to learn whether the network supports multipoint connections and, if not, it could emulate multipoint connectivity using several point-to-point connections, transparently to the user.

Table 3. Required and Optional CTL Functionality

CTL Functionality	**Description**	**Required/ Optional**
SAR	Segmentation and Reassembly of CMAP frames to/from ATM cells	Required
Sequenced	Sequenced delivery of frames	Required
Lossless	No lost frames	Required
Duplicate suppression	No duplicate frames	Required
Error-free	Frames are delivered to higher level without bit errors	Required
Flow control	Throttling mechanism to handle network or host congestion	Optional
Internal ping	Internal CTL mechanism for learning if remote peer is still alive	Optional
Auto synchronization	CTL maintains synchronization and can report loss to CMAP layer	Optional
Multiple connections	Support for multiple CTL connections on a single access link	Optional
Query	Single ATM cell query to report SAR, transport protocol, *etc.*	Optional

3.4 Signalling Connections

At the lowest level, the ATM cells comprising CMAP messages are sent over a bidirectional signalling connection between the CMAP Client and the CMAP Session Manager (Figure 9), using the CTL protocol described above. The setup of these ATM signalling connections from clients to Session Managers, and the setup of any required ATM signalling paths between CMAP Session Managers, falls into the area of network management. We require that facilities for setting up such one connection between each client and its Session Manager be available. The exact mechanisms and protocols used to establish these connections are outside the scope of this document.

Clients may request the creation of additional signalling connections, but are not required to do so (in this way our signalling connection functions as an ATM standard *meta-signalling connection*). As Table 3 indicates, we do not require that the network or CTL support such additional signalling connections; a CMAP implementation must be able to function with a single signalling connection between each client and the network.

3.4.1 Multidrop Signalling

Our model allows for multiple clients on the same access link, as shown in Figure 10. This is one case where additional signalling connections may be desirable, so as to differentiate the multiple clients on the access link. To allow for the case where additional signalling connections cannot be allocated, CMAP messages contain the address of the client to which they are directed. This allows several clients to share a single signalling link, provided the CTL delivers a copy of the data to each of the clients. As noted above, we do not require support for either additional signalling connections nor multiple connections on a single link.

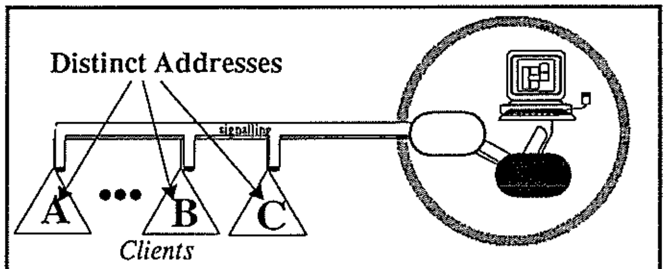

Figure 10. Multidrop Signalling

3.4.2 Surrogate Signalling

Another signalling capability that we allow for is *surrogate signalling*, where one client (called the *surrogate* client) is designated as the signalling entity for another client (called the *mute* client), as shown in Figure 11. The surrogate client originates all signalling messages for the mute client and receives all signalling messages from the network that would otherwise be sent to the mute client. To handle this, CMAP messages contain the address of the client at which the signal is directed. The *surrogate* client can be anywhere in the network and does not have to be local to the *mute* client's node. The configuration of mute and surrogate clients falls into the area of network management, but we do not require the network to provide these facilities.

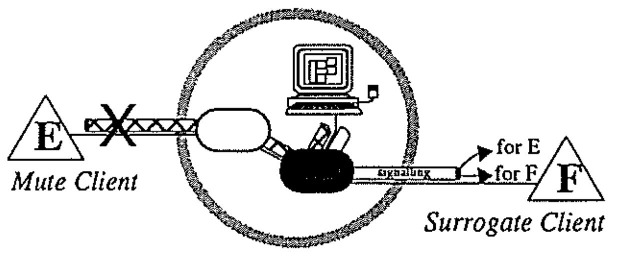

Figure 11. Surrogate Client Signalling at the UNI

An example of where surrogate signalling could be used is shown in Figure 12, where an array of high-resolution displays (used for digitized X-rays or other medical diagnostic displays) are controlled from a nearby workstation. The physician uses the workstation to select which images should be shown on each display and all of the signalling is performed by the workstation, even though each of the displays is a separate CMAP client.

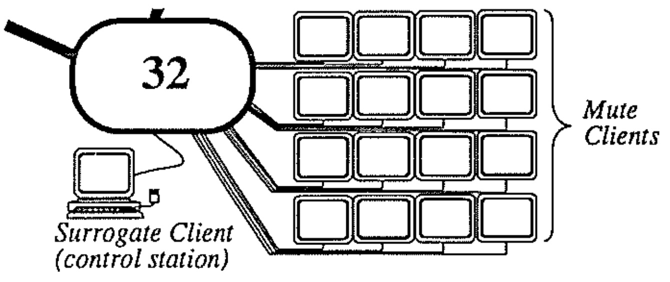

Figure 12. Surrogate Signalling for Medical Applications

3.5 Connection Management Layer Functionality

The Connection Management Layer provides the facilities whereby the CMAP Session Managers build, maintain, and destroy ATM connections, and its requirements thus lie in the area of providing the types of connections that the CMAP model supports. These requirements are summarized in Table 4. Some of these facilities (indicated by "Network" in the third column of the table) overlap with, and depend upon, the facilities made available by the physical network—obviously the CML cannot provide VP connections if the network does not.

The CML also includes a number of resource management facilities which may need explanation. *Prioritization* refers to the ability to assign calls and connections different levels of priority, with mechanisms whereby high-priority calls can take resources from lower-priority calls. *Quality of service* refers to the ability to support different levels of quality—high-quality connections might be able to guarantee no loss of ATM cells, minimal delivery times, and so forth. *Peak bandwidth reservation* refers to the bandwidth management mechanism whereby a client reserves the maximum connection bandwidth that it will ever require. *Bandwidth management* refers to more general schemes, for example involving statistical multiplexing based on average bandwidth and traffic burstiness. *Best-effort connections* are ones without reserved bandwidth. Cells on such connections will be discarded if any of the links are saturated. *Connection holding* refers to the ability to reserve connection resources (bandwidth, VPI/VCI pairs, *etc.*) without actually using the connection. One case where this might arise is when a client wishes to turn off cell reception for a period but still have the connection available; the client would then direct that the connection be held to turn off the cell flow, and when the client later directed that the cell flow be resumed the connection could be rapidly reestablished. Finally, *VPI/VCI pair allocation* refers to the ability of clients to choose which VPI/VCI pairs they will use for receiving and transmitting data. This might be particularly important where specialized hardware is used in data sources or sinks; the hardware might be able to use only a single VPI/VCI pair in the cells it sends or receives.

Table 4. Required and Optional CML Functionality

CML Functionality	Description	Required/ Optional
VP and VC	Virtual Path and Virtual Channel connections	Network
Connection types	Point-to-point, point-to-multipoint, and multipoint-to-multipoint	Network
Directionality	Unidirectional and bidirectional	Network
Connection configuration	Static and dynamic	Network
Endpoint configuration	Static and dynamic	Network
Prioritization	Ability to select among levels of priority	Optional
Quality of service	Ability to select among levels of quality	Optional
Peak bandwidth reservation	Reservation and allocation by peak bandwidth	Required
Bandwidth management	Other types of bandwidth reservation and allocation	Required
Best-effort connections	Connections without reserved bandwidth	Required
Connection holding	Ability to reserve connections without actually using them	Optional
VPI/VCI allocation	Ability to direct use of particular VPI/VCI pairs	Optional

The *Connection Management Network Protocol* (CMNP) is one possible connection management layer. CMNP provides a uniform, connection-oriented view of the ATM network and provides all the facitilities listed in Table 4 (subject, of course, to the network's capabilities). CMNP is described in a separate technical report [*REFERENCE*].

3.6 Minimal CMAP

An implementation of CMAP is distinguished from the general CMAP specification. The design of CMAP (described in Sections 4 through 6) is based on a general model of the ATM protocol and fast packet switching networks and of the supporting software such as the Connection Management Layer. A particular network environment may not be able to support all of the CMAP capabilities described. For this reason we allow an implementation of CMAP to vary from the generalized model. The network is still considered a CMAP network as long as it conforms to at least the minimum specification described here. By making CMAP as general as possible but allowing it to have implementation subsets, we provide a single call management protocol that can be layered over a diverse set of networks.

If all CMAP options are implemented and all abstractions of the call model are supported in a CMAP implementation, then we refer to this as *complete CMAP*. We also defined a *minimal CMAP* that specifies the minimum capabilities that the network must support in order to be considered a CMAP network. The preceding sections (particularly Tables 2, 3, and 4) describe the required support for a minimal CMAP. Table 5 summarizes these sections and describes minimal and complete CMAP. The first column lists each of the functions. The next column indicates whether the function is required or optional in CMAP. Required functions must be provided in both minimal and complete CMAP. Optional functions need not be provided in a minimal CMAP implementation but are provided by complete CMAP. The remaining two columns contrast the minimal and complete CMAP implementations—of course, for required functions the two implementations are identical. In some cases (*e.g.*, bidirectional connections) the function may not be required by the supporting layers but is still required in CMAP. In these cases CMAP is required to emulate the function, possibly using the method suggested in the table.

Table 5. Minimal CMAP Functionality

Function	Required/ Optional	Effect on Minimal CMAP	Effect on Complete CMAP
VP connections	Optional	Clients requesting a VP connection are told the connection could not be created	All VP's supported by the network are available to clients
VC connections	Required	All VC's supported by the network are available to clients	
Point-to-point	Required	Two-client point-to-point calls must be fully supported	
Point-to-multipoint	Optional	Clients requesting point-to-multipoint are told the connection could not be created	Fully supported
Multipoint-to-multipoint	Optional	Clients requesting multipoint-to-multipoint are told the connection could not be created[1]	Fully supported
Unidirectional	Required	Fully supported	
Bidirectional	Required	If the network does not support bidirectional connections, we require that CMAP emulate the connections by establishing two (or more) unidirectional connections	
Static connections	Required	Fully supported	
Dynamic connections	Optional	Clients attempting to modify a connection are informed that the operation could not be performed	Fully supported
Static mappings	Required	Fully supported	
Dynamic mappings	Required	If dynamic mappings are not supported by the network, the CMAP clients and session managers must set up all connections as receive/transmit. Requests to change mappings affect software data but not hardware. Cells received on a connection that is not set to receive should be silently discarded.	
Error-free CTL	Required	Used for message transmission as described in Section 3.3	
Flow control	Optional	Not used	Used, but invisible to client
Internal ping	Optional	Not used	May be used to restart clients and/or session managers
Auto synchronization	Optional	Not used	May be used to restart clients and/or session managers
Multiple connections	Optional	Not used	May be used to support multi-drop and surrogate signalling
Query	Optional	Not used	May be used to determine network capabilities and assist with emulation activities
Single signalling channel	Required	Configured by network management	
Additional signalling channels	Optional	No facility for allocating additional channels provided to clients	Clients may request any number of additional private signalling channels

Table 5. Minimal CMAP Functionality

Function	Required/ Optional	Effect on Minimal CMAP	Effect on Complete CMAP
Multidrop signalling	Optional	Multidrop configurations may not be used	Where clients share a single link, must ensure each gets a copy of all signals sent
Surrogate signalling	Optional	Surrogate signalling may not be used	Fully supported
Prioritization	Required	CMAP Session Managers keep track of call priorities and abort lower-level calls when their resources are needed by a higher-level call.	
Quality of Service	Optional	All CMAP QOS levels map to the same network QOS	Each CMAP QOS level maps to a different network QOS
Peak bandwidth reservation	Required	Fully supported; sum of peak bandwidths on any link does not exceed the link's capacity	
Bandwidth management	Optional	Session managers do not monitor traffic or attempt to do statistical multiplexing	Session managers may monitor traffic to collect statistics, set up hardware to enforce the connection BW behavior, and perform statistical multiplexing
Best-effort connections	Required	No bandwidth reserved for best-effort connections; cells transmitted on connections have CLP = 1, thus indicating that they should be discarded when congestion is encountered	
Connection holding	Required	Connection holding may be emulated by keeping the connection open and discarding cells (see dynamic mappings above)	
VPI/VCI allocation	Optional	Clients attempting to allocate a specific pair are informed that the pair is unavailable	Fully supported

[1] If the network supports point-to-multipoint but not multipoint-to-multipoint, we strongly encourage the emulation of multipoint-to-multipoint by the use of multiple point-to-multipoint connections, one for each transmitter.

4. Call Model

CMAP is a protocol whereby network *clients* request the creation, modification, and deletion of *calls*, which are distributed entities containing *connections* and *endpoints*. This section defines the attributes of CMAP clients, calls, connections, and endpoints. The attributes of CMAP entities and the operations that may be performed on them (described in Section 6) together define the CMAP *call model*. Section 4.1 first describes the basic concepts of CMAP. Sections 4.2 through 4.5 then describe the attributes of the various entities. Finally, Section 4.6 summarizes the identifiers that serve to uniquely identify each entity in the CMAP context.

4.1 Basic Concepts

A CMAP *client* is a user of the network, in the most general sense. Our model views all clients as equal. We draw no distinction, for example, between network gateways, multimedia workstations, or video phones. All clients are constrained to signal the network through our CMAP protocol and communicate through ATM cell pipes. We believe that our call model provides the necessary core functionality so that other services can be layered over CMAP, perhaps presenting a more sophisticated network model to users. Additionally, we feel that CMAP's interconnection services are suitable for both local and wide area networks.

Network clients signal other clients by issuing CMAP requests that create and manipulate *calls*. A call is a distributed object maintained by the network that describes the communication paths that interconnect clients. Two of the call's parameters are its *owner* and *root*. The owner is a CMAP client which is responsible for managing the call. It need not be a participant in the call (for example, it might be managing a mute video server). The root is another CMAP client which is participating in the call. The root provides a known point in the network toward which routing can be directed. In most cases, the root and the owner will be the same client.

A call has one or more communication channels, which we term *connections*. A connection is simply an ATM end-to-end cell pipe (Section 2.3). Participants in a call can add connections to the call, subject to approval by the owner. Multiple connections within a call are useful for applications such as video conferences, where one connection carries video and another audio. Figure 13 shows an example of such a multimedia call at a client, with separate connections within the call carrying video, voice, and data. The protocols used by clients to send data over connections are outside the scope of CMAP.

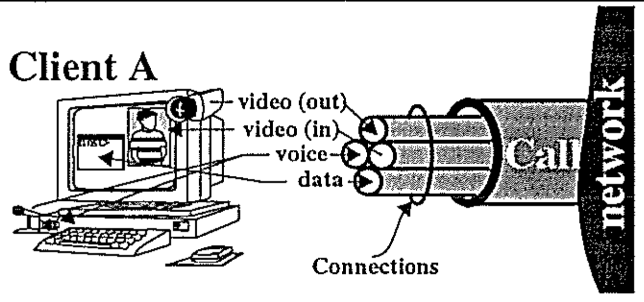

Figure 13. Example Multiple-Connection Multimedia Call

A call also has one or more *endpoints*, which are the interfaces between clients and calls. A client may join in a call several times, resulting in several distinct endpoints associated with that client. When a call is first created, an endpoint for the root and optionally one additional endpoint (for either the root or for another client) are added to the call. Additional endpoints may be added by: 1) invitation from the owner, where the invited party has the option of refusing the invitation, 2) request from a client not currently in the call to be added, where the owner has the option of denying the request, or 3) request from a third party, not necessarily in the call, to add a client, where both the owner and the client being added have the option to refuse.

Each endpoint has separate parameters for each connection in the call. We use the term *UNI* (user network interface) to refer to these per-connection endpoint properties. One of the most important of the UNI properties is the endpoint *mapping*, which indicates whether the endpoint receives, transmits, and/or echoes data on the connection. Figure 14 shows six common ways (of the twelve possible) that an endpoint may map a connection to itself at its access point (refer to Table 12 on page 148 for a complete list of possible mappings). During the lifetime of the call, the client

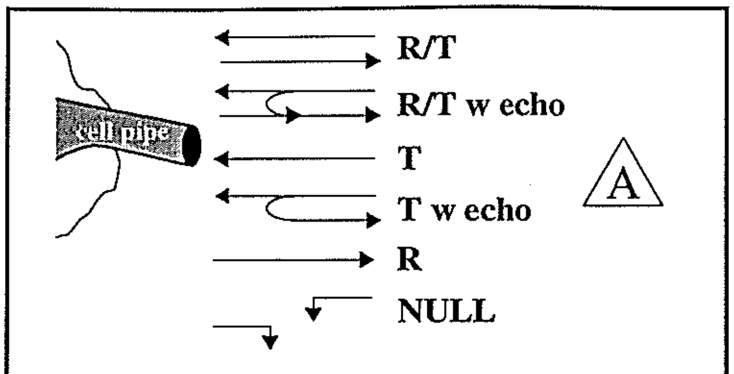

Figure 14. Sample Connection Mappings at the UNI

may dynamically change its mappings for each connection in the call. Such a feature might, for example, be useful in a video teleconferencing system in which only one person can transmit at a time; clients would normally be mapped to the video connection as receivers, but would change to transmit with echo when the client is the speaker.

4.2 Clients

A CMAP client is any user of the network. Clients have only one parameter, an address, which serves to uniquely identify the client for the entire network.

4.2.1 Client Address

Each client of the network is identified by a unique address. These addresses are used in CMAP operations to indicate the client to be added, modified or dropped from a call or connection, and to indicate the client performing the operation. Internally, the network may use the addresses for signal routing and connection setup.

The CMAP protocol does not specify a particular addressing scheme. Rather, CMAP implementations can choose to support one or more addressing schemes (for example, an implementation might support several well-known addressing schemes, such as IP or NSAP, and one or two experimental addressing schemes). The format of a CMAP client address is shown in Figure 15. The address size is a constant 24 bytes, regardless of the addressing scheme used. The first four bytes comprise a *type* field that tells how the remainder of the address should be interpreted. The remainder is partitioned into three additional fields whose sizes vary depending on the type of addressing used: a *network address field, local address field* and an *unused* field. The network address field contains the only portion of the address that the network uses in client identification and routing. The local address field is passed end-to-end in CMAP operations for use by the clients and is not interpreted by the network. The remainder of the address is unused.

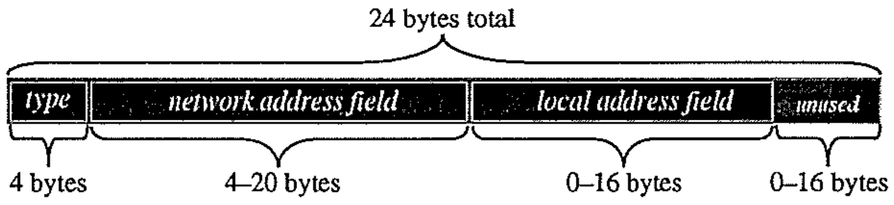

Figure 15. CMAP Client Address Format

Figure 16 shows several example partitioning schemes. The upper portion of the figure shows the partitioning used for IP addresses [18]. The network address field is 4 bytes and holds the IP address. The local address field is 2 bytes and contains the higher level port number (for example, the TCP or UDP port). The remaining 14 bytes of the address field are unused. The middle portion of the figure shows the partitioning for the CCITT E.164 addresses [13] used in the public telecommunications networks. The network address field is 8 bytes and encodes the 15 decimal digit E.164 address. The local address field is 4 bytes and holds the E.164 subaddress field. The remaining 8 bytes are unused. The bottom portion of the figure shows the partitioning for OSI NSAP (Network Service Access Point) addresses

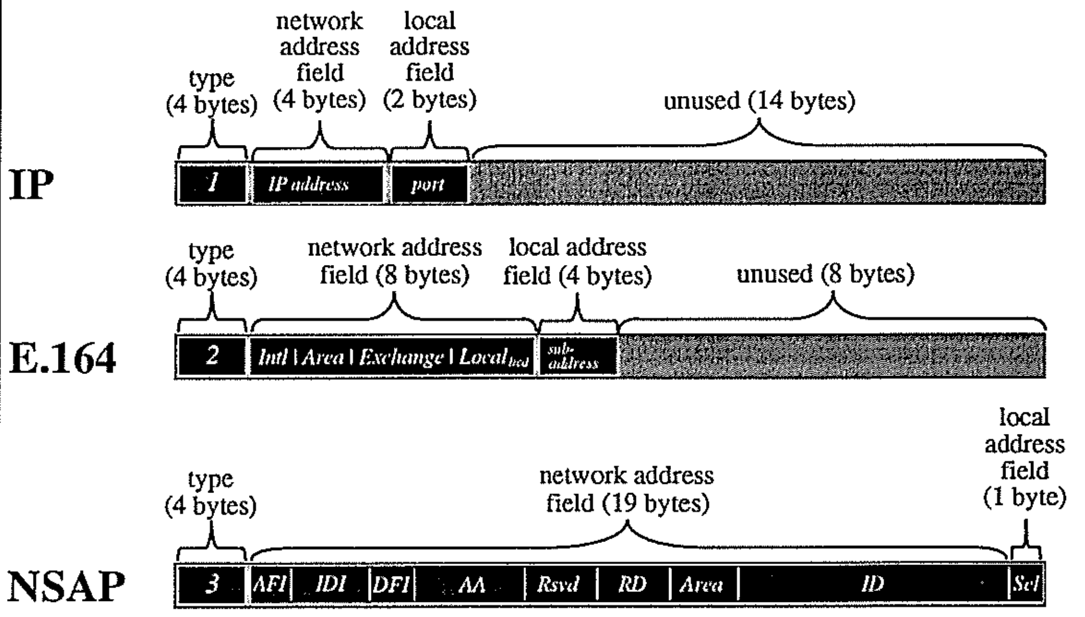
IP
type (4 bytes)
network address field (4 bytes)
local address field (2 bytes)
unused (14 bytes)
1
IP address
port
E.164
type (4 bytes)
network address field (8 bytes)
local address field (4 bytes)
unused (8 bytes)
2
Intl | Area | Exchange | Local
sub-address
NSAP
type (4 bytes)
network address field (19 bytes)
local address field (1 byte)
3
AFI
IDI
DFI
AA
Rsvd
RD
Area
ID
Sel

Figure 16. Examples of Client Address Partitioning

[17]. The network address field is 19 bytes and contains all fields of the NSAP address except for the SEL (selector) field, which is contained in the 1-byte local address field. Since NSAP addresses are 20 bytes in total length, there is no unused portion.

4.3 Calls

The call is the primary object manipulated by clients. When a client creates a call it becomes the owner of the call. When a client is added to a call it becomes an endpoint in the call and gains access to all the connections of the call. Calls have a number of parameters that describe the call and indicate how clients may access and modify the call. These parameters are described in the following subsections and summarized in Table 6.

Table 6. Call Parameters

Parameter	Description
Owner	The client that manages this call
Root	The client toward which routing is done
Local Identifier	Root-wide unique identifier for this call
Type	Multipoint or point-to-point
Accessibility	Ability of clients to join the call
Modifiability	Whether a nonowner client may add connections
Traceability	Whether a nonowner client may obtain parameters
Monitoring	Level of notification of client joins and drops
Priority	Call level priorities, for call preemption
User Type	User description of call
Connection List	Current connections in call
Endpoint List	Current endpoints in call

4.3.1 Call Owner

The owner is the address of the client which manages the call. Management operations include activities such as modifying call, connection, or endpoint parameters, closing the call, and approving the addition of endpoints. The owner is initially the client who creates the call.

4.3.2 Call Root

The root is the address of a client which is participating in the call. The root may be used by the network routing algorithms when adding new endpoints to the call. The root is initially chosen by the creator of the call.

4.3.3 Call Local Identifier

The local identifier is a small integer which is unique for all calls having the same root. Together, the root address and the local call identifier form the *call identifier*, a network-wide unique identifier for the call. The local identifier is initially chosen by the creator of the call (with the approval of the network, which ensures that it is not already in use) and subsequently changes only if the root of the call changes.

4.3.4 Call Type

The type designates whether the call is *multipoint* or restricted to *point-to-point*:

Type ∈ { **MULTIPOINT, POINT-TO-POINT** }

We use **MULTIPOINT** to refer to both point-to-multipoint and multipoint-to-multipoint calls. While the call model supports multicast calls as the general object, some calls are intrinsically point-to-point and, if designated as such, can be serviced more efficiently by the network (for example, by routing the call over internal trunks).

4.3.5 Call Accessibility

The accessibility parameter is a 2-tuple that controls whether clients are allowed to add endpoints:

Accessibility ∈ { **OPEN, VERIFY, CLOSED** }

If the call is **OPEN**, any client in the network may join the call without the owner's permission. A **CLOSED** call restricts the call such that only the owner may add clients. If the call has the **VERIFY** accessibility, then any client may attempt to join the call, but he operation will be verified with the owner before it is allowed to complete.

4.3.6 Call Modifiability

The modifiability parameter controls whether non-owners have the permission to add connections to the call

Modifiability ∈ { **ON, OFF** }

If this parameter is **ON**, any participant in the call may add connections. If it is **OFF**, only the owner may add connections. Non-participants are never allowed to add connections. One example of where this option is useful is in multiparty conference calls, where each endpoint adds a one-to-all connection for his feed when joining the call.

4.3.7 Call Traceability

The traceability parameter is a 2-tuple that controls whether clients have the permission to obtain information about the call):

Traceability ∈ { **OPEN, MEMBERS, CLOSED** }

A value of **OPEN** indicates any client may perform traces. A value of **MEMBERS** indicates that only clients who are participating in the call may perform traces. **CLOSED** indicates only the owner may perform a trace. The owner of the call is always allowed to perform traces.

4.3.8 Call Monitoring

The monitoring parameter is a 3-tuple designating whether endpoints of the call are notified when endpoints join or drop out of the call or modify their parameters:

```
Monitor = < owner ∈ { ON, OFF },
            transmitters ∈ { ON, OFF },
            all ∈ { ON, OFF } >
```

If the *owner* field is set to ON, all endpoint joins and drops are reported to the owner. Likewise, if the *transmitters* field is set to ON, all transmitters are notified when these changes occur, and if the *all* field is set to ON, all participants in the call are notified. When used with an accessibility of **OPEN**, an owner modifiability setting of **ON** allows the owner to keep track of endpoint joins and drops without being burdened with explicitly verifying every such change.

4.3.9 Call Priority

The priority parameter is a 1-tuple that allows some calls to take precedence over others:

$$\textbf{Priority} \in \{\ \textbf{NORMAL, PREEMPT, OVERRIDE}\ \}$$

NORMAL is the lowest priority, followed by **PREEMPT**, followed by **OVERRIDE**. If the network does not have the resources to support a call and the required resources are currently allocated to lower-priority calls, those resources are reclaimed (aborting the lower priority calls) so that the higher priority call can be supported. The priority affects how the CMAP Session Managers process the call and whether the call should be blocked if resources are unavailable. ATM cell-level priorities are implemented separately as parameters of connections (Sections 4.4.2 and 4.4.3).

4.3.10 Call User Type

The user type is a value chosen by the call creator. This parameter is not manipulated by the network but is simply passed end-to-end so that clients may examine it.

4.3.11 Call's Connection List

The connection list describes the connections of the call. The parameters of connections are described below. The list is dynamic since connections can be added to, modified, or removed from the call at any time.

4.3.12 Call's Endpoint List

The endpoint list describes the endpoints participating in the call. The parameters of endpoints are described below. The list is dynamic since endpoints can be added to, modified, or removed from the call at any time.

4.4 Connections

The connection is the primary information-carrying component of a call. The parameters associated with connections are described in the following subsections and summarized in Table 7.

Table 7. Connection Parameters

Parameter	Description
Identifier	Call-wide unique identifier for this connection
Type	Three tuple of VP/VC, dynamic/static bandwidth and QOS
Bandwidth	Reserved bandwidth for the connection
Defaults	Defaults to be offered to endpoints joining the connection
Permissions	Permissions to be offered to endpoints joining the connection
User Type	User description of connection

4.4.1 Connection Identifier

The identifier is a small integer which uniquely identifies the connection within the call. This value may be determined by the client which adds the connection to the call, with the network ensuring that it is unique, or the client may leave the field blank and permit the network to select a unique identifier.

4.4.2 Connection Type

The type is a 3-tuple:

$$\begin{aligned}\textbf{Type} = <\ & \textbf{channel\_type} \in \{\ \textbf{VP, VC}\ \},\\ & \textbf{BW\_type} \in \{\ \textbf{STATIC, DYNAMIC}\ \},\\ & \textbf{QOS} \in \{\ \textbf{HIGH, MEDIUM, LOW}\ \} >\end{aligned}$$

The first component specifies the type of channel that the connection requires and must be one of two values: *virtual path* (VP) or *virtual channel* (VC) (Section 2.3). The second component specifies whether the connection is **DYNAMIC** or **STATIC**. If a connection is static then the bandwidth is fixed throughout the life of the call, which makes the connection more predictable and allows for more efficient use of network resources (such as routing over internal trunks). Dynamic connections can have their bandwidth modified during the life of the call. The last component specifies the desired *quality-of-service* (QOS) and may have the values of **HIGH, MEDIUM** or **LOW.** QOS relates to options for cell loss behavior that may vary from network to network. QOS, therefore, is left intentionally vague. If the network can implement different cell loss behavior strategies then the network control software will group these into the categories of **HIGH, MEDIUM** and **LOW.**

4.4.3 Connection Bandwidth

We support two types of bandwidth specification for connections: 1) *reserved,* where the network guarantees and enforces the requested bandwidth, and 2) *best-effort,* where the network neither guarantees nor enforces. Reserved bandwidth connections are treated preferentially by the network, whereas best-effort connections compete with one another for the remaining bandwidth not currently used by reserved connections.

The connection's bandwidth is specified as a 3-tuple:

Type = < peak, average, peak_burst_length >

The *peak* and *average* parameters are expressed in cells per second. The *peak burst length* is measured in cells and indicates the maximum number of cells that the endpoint can send during a burst at peak rate. Best-effort connections are indicated by all zeros in the bandwidth specification.

For reserved bandwidth connections, the figures given are for aggregate bandwidth, as seen at any receiver in the connection (this takes into account the increased loading that results when the transmissions from multiple transmitters combine and flow over links traversed by the connection). Figure 17 illustrates how we use the bandwidth specification to characterize the endpoint's transmit behavior on a connection at the UNI. The *peak* value tells us a percentage of the bandwidth used by the endpoint when transmitting a burst of data. The user may not arbitrarily send cells back-to-back for the duration of the *peak burst length,* but must pace them according to the *peak* value as shown. The *peak burst length* tells the network how long the connection will remain transmitting at *peak* rates. The example in the figure shows a *peak* value of 87,000 cells per second (in this case, roughly 25% of a 155Mbps link) and a *peak burst length* of 1000 cells. The *average* value is used over a longer period of time (for example, several seconds) to enforce the long term average utilization of the connection.

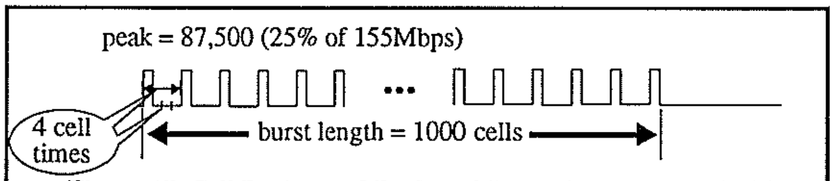

Figure 17. Cell Pacing and Peak and Burst Length at the UNI

For best-effort connections, the endpoint is not restricted in any way. Instead, all cells are forcibly marked at the UNI by setting the header's CLP bit. Inside the network, as buffers fill, the network discards marked cells if there is no room in the buffers for unmarked (reserved) cells.

We chose to provide both reserved and best-effort bandwidth connections for several reasons. Some applications clearly need bandwidth guarantees, such as live video, where cell loss can result in incomprehensible transmissions. However, many other applications do not need these guarantees. For example, data transfer applications can usually tolerate lossy networks and network congestion using schemes such as retransmission and backoff, as in TCP/IP [18]. Additionally, many applications cannot easily predict their bandwidth requirements, and if forced to do so would produce specifications that are too low or too high, each with negative consequences. Finally, when all applications are forced to use reserved, guaranteed connections, network utilization is sometimes low (depending on the connection mix) since the level of allowed multiplexing must be limited to provide the guarantees. By introducing the best-effort bandwidth class, we accommodate those connections that do not need the guarantees and cannot easily estimate requirements, and we provide a means whereby the excess capacity of the network can be utilized.

4.4.4 Connection Defaults

The defaults is the value of the endpoint defaults field (and also of the initial endpoint mapping) offered to endpoints joining the call. The defaults may be overridden by the call owner so that individual endpoints can be assigned different defaults. The defaults is a 3-tuple:

$$\textbf{Defaults} = < \textbf{receive} \in \{ \text{ON, HOLD, OFF} \}, \; \textbf{transmit} \in \{ \text{ON, HOLD, OFF} \}, \; \textbf{echo} \in \{ \text{ON, OFF} \} >$$

Description of these fields is deferred until the discussion of the endpoint mapping (Section 4.5.3).

4.4.5 Connection Permissions

The permissions is the permissions given to endpoints joining the call. The permission may be overridden by the call owner so that individual endpoints can be assigned different permissions. The permissions is a 3-tuple:

$$\textbf{Permissions} = < \textbf{receive} \in \{ \text{ON, VERIFY, OFF} \}, \; \textbf{transmit} \in \{ \text{ON, VERIFY, OFF} \}, \; \textbf{echo} \in \{ \text{ON, OFF} \} >$$

Description of these fields is deferred until the discussion of the endpoint permissions (Section 4.5.3).

4.4.6 Connection User Type

The user type is a value chosen by the connection creator. This parameter is not manipulated by the network but is simply passed end-to-end so that clients may examine it.

4.5 Endpoints

The *endpoint* is the interface of a client with a call. The parameters associated with endpoints are described in the following subsections and summarized in Table 8. Except for the endpoint's address and local identifier, all of these parameters are UNI parameters assigned on a per-connection basis — that is, the endpoint has a separate mapping for each connection, a separate defaults, and so forth.

Table 8. Endpoint Parameters

Parameter	Description
Address	Client with which this endpoint is associated
Local Identifier	Client-wide unique identifier for this endpoint
Mapping	Current access of endpoint to connection
Defaults	Default value of the mapping
Permissions	Whether the endpoint may modify its mapping
Transmit Pair	VPI/VCI pair for ATM transmit
Receive Pair	VPI/VCI pair for ATM receive

4.5.1 Endpoint Address

The address is the address of the client with which the endpoint is associated.

4.5.2 Endpoint Local Identifier

The local identifier is a small integer which is unique for the endpoint with respect to the client. Together, the endpoint address and local identifier form the *endpoint identifier* which uniquely identifies the endpoint within the client. The local identifier may be determined by the client which adds the endpoint to the call, with the network ensuring that it is unique, or the client may leave the field blank and permit the network to select a unique identifier.

4.5.3 Endpoint Mapping

Endpoints have separate receive, transmit, and echo mappings for each connection in the call. The mapping is a dynamic field which may be changed by the call owner or by the endpoint itself. Initially the mapping is set to the endpoint *defaults*. For each connection in a call, the associated endpoint mapping is a 3-tuple:

$$\textbf{Mapping} = < \textbf{receive} \in \{ \text{ON, HOLD, OFF} \}, \textbf{transmit} \in \{ \text{ON, HOLD, OFF} \}, \textbf{echo} \in \{ \text{ON, OFF} \} >$$

For the *receive* mapping, a value of **ON** indicates that the endpoint is receiving on the connection, a value of **OFF** indicates that the endpoint is not receiving on the connection, and a value of **HOLD** indicates that the endpoint is not currently receiving but that bandwidth should be reserved within the network up to the endpoint's access node (for use when the endpoint's receive permission is changed to **ON** at some later time). The *transmit* mapping is defined analogously to the receive mapping.

The *echo* mapping is used in conjunction with transmit to allow endpoints to view their own transmissions. When set to **ON**, the endpoint's transmissions will be echoed, and when set to **OFF** they will not be echoed.

4.5.4 Endpoint Defaults

The defaults indicates the default value of the endpoint's mapping, as distinct from the endpoint mapping itself which is the current value of the mapping. The defaults is a 3-tuple whose components and values are the same as those listed above for the mapping. Again, the endpoint has a separate default for each connection in the call. When an endpoint is added its defaults are taken from the connection defaults (Section 4.4.4), or the call owner may override the connection defaults and assign any desired endpoint defaults.

4.5.5 Endpoint Permissions

The permissions indicates the manner in which individual endpoints may change their mappings. Endpoints have separate permissions for each connection in the call and, as with mappings, the permissions for each connection may be completely unrelated to those of the other connections. For each connection in a call, the associated endpoint permission is a 3-tuple:

$$\textbf{Permissions} = < \textbf{receive} \in \{ \text{ON, VERIFY, OFF} \}, \textbf{transmit} \in \{ \text{ON, VERIFY, OFF} \}, \textbf{echo} \in \{ \text{ON, OFF} \} >$$

Each field affects the corresponding field of the mapping. If the permission field is **OFF**, the endpoint is not permitted to change its endpoint mapping and it must equal the endpoint's default mapping for the connection (with one exception: if the default field is **ON** the endpoint may also use a mapping of **HOLD**). If the permission field is **ON**, the endpoint may choose any mapping. In this case, the default mapping is not enforced and is merely viewed as a suggestion to the endpoint. If the permission is **VERIFY**, the endpoint can alter his mapping, but the owner will first be queried to see if the new mapping is acceptable (with one exception, related to the previous: if the default field is **ON** the endpoint may change its mapping from **ON** to **HOLD** or vice-versa without verification). When an endpoint is added its permissions are taken from the connection permissions (Section 4.4.5), or the call owner may override the connection permissions and assign any desired endpoint permissions.

4.5.6 Endpoint Transmit Pair

The transmit pair is the ATM VPI/VCI pair which the endpoint uses to transmit data. When the endpoint is added, it may suggest a pair which (if it is available) will be used by the network. The pair may also be left blank, in which case the network will choose an available pair.

4.5.7 Endpoint Receive Pair

The receive pair is the ATM VPI/VCI pair which the endpoint uses to receive data. When the endpoint is added, it may suggest a pair which (if it is available) will be used by the network. The pair may also be left blank, in which case the network will choose an available pair.

4.6 Summary of Identifiers

Clients, calls, connections and the endpoints participating in calls all require some form of identification at the CMAP level so that these entities can be referenced by clients.

Client addresses are unique labels assigned to clients so that clients can identify one another. Client addresses are also used by the network to locate clients and route messages internally.

Call identifiers are unique labels assigned to calls so that each call can be referenced by clients (for example, for joining and modifying calls). They are a combination of the call's *root address* and a *local identifier* (a small integer). Clients can assign well-known call identifiers to calls that are to be globally known, for example, for broadcast video distributions.

Connection identifiers are unique labels for the connections within a call so that clients can individually reference the connections (for example, for changing ones receive/transmit mapping on a given connection). They are small integers. Clients can assign well-known connection identifiers to connections within a call, allowing the clients to easily distinguish among them.

Endpoint identifiers are labels assigned to each interface between a client and a call. They are a combination of the client's *address* and a *local identifier* (a small integer). A client is allowed to join a call more than once and the local endpoint identifier is used to distinguish the separate instances of client participation.