1998 International Computer Symposium
Workshop on Computer Networks, Internet, and Multimedia
December 17-19, 1998, N.C.K.U., Tainan, Taiwan, R.O.C.

Connectivity Management in ATM Networks

K. Ravindran

Department of Computer Science
City University of New York (City College)
Convent Avenue at 138th Street, New York, NY 10031 (USA)
Contact e-mail address: *ravi@cs-mail.engr.ccny.cuny.edu*

Abstract

The paper describes a generalized model of logical connectivity among user entities through ATM based backbone networks. The model is based on tree-structured communication channels connecting data sources and destinations residing in 'access' network, with routing functions in one or more cell switching nodes and transport paths over the intervening VP links of the 'core' network. The model adopts a functional binding of channels to cell routing procedures and cell flow handlers installed at various nodes. The cell address, carried as VCI, may be mapped at each node traversed by the cell to a local flow handler and a route that defines the VP links incident on this node. The model also treats the multicast and unicast transport of cells in a uniform manner. The connectivity model will be useful for the evolving technologies of WAN/MANs to support multi-service applications (e.g., video conferencing, digital TV broadcast).

Key words: Routing control interface, multicast cell switching, associative address mapping, connection signalling protocols, packet-layer interworking, user-level reconfigurations.

1 Introduction

ATM (Asynchronous Transfer Mode) is a switching technology that is standardized for adoption over an interconnection of network nodes through fiber-optic links (such as SONET). ATM offers a viable solution for building high speed wide-area and/or metropolitan area backbone networks to support multi-service applications (e.g., digital TV, video conferencing, distributed computing) that exhibit high variability in the transport characteristics of data (such as data rates, acceptable data delays and data burstiness).

In an ATM network, data is transported across user entities of an application residing in 'access nodes' as small fixed size (53-byte long) packets called *cells*. The data transport takes place over an end-to-end virtual circuit (VC) connecting these entities through one or more 'core' switching nodes and the intervening fiber links. See Figure 1. Each switching node potentially maintains a large number of VCs to connect the user entities of a multitude of applications attached to the 'access nodes'. The ATM connection layer provides the low level functions for asynchronous transport of user data, namely, dispatch/fielding of cells belonging to a VC over a link and statistical multiplexing of cells belonging to various VCs over a link. Refer to [1] for a detailed description of these features.

An application level transport connection among a set of user entities over an ATM network may be viewed as embodying two types of functions in the network:

- Logical connectivity management for routing of cells from a source entity to one or more destination entities through various switching nodes and links;
- Data flow management for QOS and traffic handling across source-destination entities, for cell bandwidth allocations at various switching nodes and links.

The connectivity management functions and the data flow management functions fall along two different axes of the *control plane* of a target network architecture, realized on top of the ATM cell switching layer. See Figure 2. This exemplifies an object-oriented decomposition of the transport connection protocols that allows the above functions to directly interface with the applications.

Though the procedural aspects of connectivity management and data flow management are orthogonal to each other, these functions are 'glued' together at the application level transport interface to the ATM network. For this purpose, the address field carried in a cell (2-byte long) may be used as an index to the flow handling procedures in a switching node, in addition to being used for routing of cells over a VC. For instance, information on the number of cells discarded at a node due to traffic congestion on a given VC may be collected by the node which may be used by an appropriate flow handling procedure associated with the VC to modify the scheduling priority of cells. The allowance of separate flow handling procedures that are bindable to VCs allows extending the degrees of per-VC scheduling control offered by the ATM cell switching

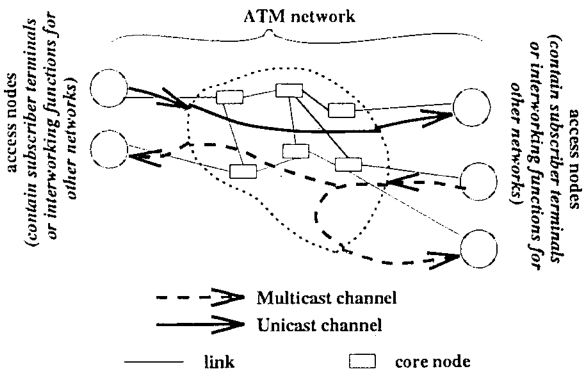
ATM network
access nodes (contain subscriber terminals or interworking functions for other networks)
access nodes (contain subscriber terminals or interworking functions for other networks)
Multicast channel
Unicast channel
link
core node

Figure 1: Communication channels in ATM networks

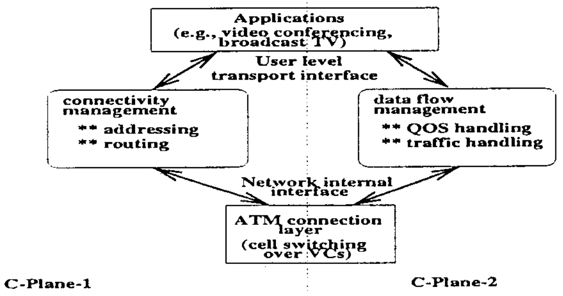
Applications (e.g., video conferencing, broadcast TV)
User level transport interface
connectivity management
** addressing
** routing
data flow management
** QOS handling
** traffic handling
Network internal interface
ATM connection layer (cell switching over VCs)
C-Plane-1
C-Plane-2

Figure 2: High level view of cell transport functions on ATM networks

layer (with 2-bit priority fields).

Overall, logical connectivity among user entities over an ATM network determines the path interconnecting the 'access' nodes containing these entities through 'core' switching nodes and links. The VC abstraction supports the interconnection with procedures to route cells over the path and with one or more cell flow handling procedures, instantiated in the layer above 'virtual path' (VP) switching. For instance, one VC may carry delay-sensitive but somewhat loss-tolerant audio data, while another VC may carry loss-sensitive but delay-tolerant file data. The flow handling procedures for these VCs are quite different, and need to be be indexed by distinct VC identifiers (VCI). In general, the various nodes in the path of a VC employ each an appropriate cell flow handling protocol instance. See Figure 3.

Though connectivity related functions are part of VC protocols used in many wide-area networks (e.g., ARPANET) [2], these functions are often monolithically blended into the VC protocols with no direct user level control (in particular, bandwidth allocation on links). In the evolving ATM networks for multi-service applications how-

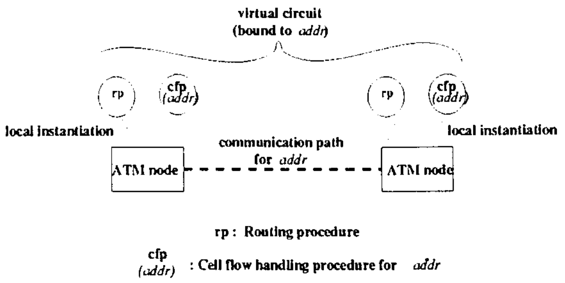
virtual circuit (bound to addr)
rp
cfp (addr)
rp
cfp (addr)
local instantiation
local instantiation
communication path for addr
ATM node
ATM node
rp : Routing procedure
cfp (addr) : Cell flow handling procedure for addr

Figure 3: VC and its binding to cell path

ever, the connectivity functions assume importance due to the need to support: i) multicasting of cells over a VC, and ii) multiplexing of cells belonging to a large number of different VCs over a shared VP link, with each VP link being bandwidth controllable. The purpose of this paper is to analyze how such a connectivity among user entities can be provided over an ATM network.

The paper focusses on the internals of the connectivity functions and how these functions *externally* interface with the data flow related functions. The connectivity related aspects discussed are: i) topology of communication channels that determine the multicast paths underlying the VCs, ii) assignment of addresses to VCs, iii) mapping of a VC address to the underlying path for cell routing, and iv) functional binding of VCs to data flow handling protocol instances. The internal details of flow handling procedures (such as channel bandwidth reservation and transport buffering of cells) are outside the scope of this paper.

2 Our model of logical connectivity

The ATM network consists of a set of nodes (or switches) $\mathcal{V}$, interconnected with one another through high speed fiber links. The user entities implementing an application are the end-points of communication, with the 'access nodes' $\mathcal{U} \subseteq \mathcal{V}$ containing the user entities forming a *configuration set.* Each user entity can be a source generating data and/or a destination consuming data. In an example of conference, the source is the initiator of a conversation and the destinations are the other participants in the conference browsing the conversation. The generation and consumption of data by applications, which we refer to as 'user-level transport functions' (UTL), involves transporting the data cells from sources over a VC set up through the network to connect to various destinations. We refer to the functions dealing with VC set up between source and destination entities and the addressing of these VCs as 'routing control layer' (RCL).

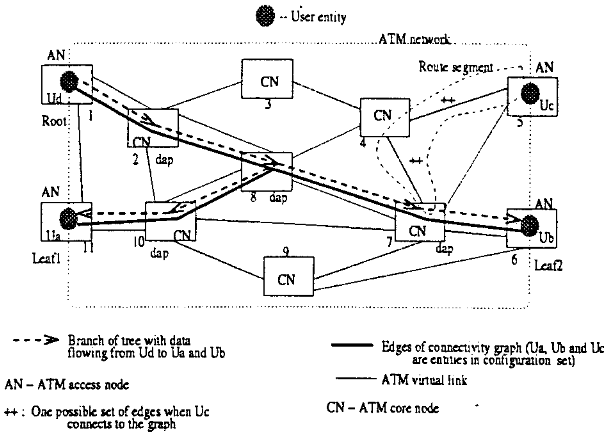
-- User entity
ATM network
AN
Ud
Root
1
CN
3
Route segment
AN
Uc
5
CN
4
++
CN
2
dap
8
dap
AN
Ua
Leaf1
11
10
CN
dap
9
CN
7
CN
dap
AN
Ub
6
Leaf2
Branch of tree with data flowing from Ud to Ua and Ub
AN – ATM access node
++ : One possible set of edges when Uc connects to the graph
Edges of connectivity graph (Ua, Ub and Uc are entities in configuration set)
ATM virtual link
CN – ATM core node

Figure 4: Tree-structured logical connectivity

2.1 Tree-structured channels

An ATM VC is modelled as a *channel* that allows users to communicate with one another through a set of 'core nodes' $\widehat{\mathcal{U}}$, where $\mathcal{U} \subseteq \widehat{\mathcal{U}} \subseteq \mathcal{V}$. The network views the channel as an *acyclic graph* $\mathcal{G}(\widehat{\mathcal{U}}, \widehat{\mathcal{E}})$ providing logical connectivity among users, with vertices residing in $\widehat{\mathcal{U}}$ and edges $\widehat{\mathcal{E}}$ mapping to the intervening links. See Figure 4. The flow of cells from a node x to a neighboring node y over the link between these nodes, where $x, y \in \widehat{\mathcal{U}}$, manifests as traversal of the edge $e(x,y) \in \widehat{\mathcal{E}}$ connecting the adjacent vertices in x and y. The edge $e(x,y)$ represents the cell routing and flow handling peer procedures installed in x and y. Figure 4 shows a channel that connects user entities U_a, U_b and U_d through a network with $\mathcal{V} = \{1, 2, \cdots, 10, 11\}$, $\mathcal{U} = \{1, 6, 11\}$, $\widehat{\mathcal{U}} = \{1, 2, 8, 7, 6, 10, 11\}$ and $\widehat{\mathcal{E}} = \{e(1,2), e(2,8), e(8,10), e(10,11), e(8,7), e(7,6)\}$.

The network may support one or more channels, with a switch capable of supporting a path segment each for these channels. Typically, a channel may define connectivity among users in a single application. For example, all participants in a video conference may be connected by a single channel.

2.2 Data access points

Each node $u \in \widehat{\mathcal{U}}$ supports a certain level of data rate by allocating link resources (e.g., buffers) to handle the cells arriving at and/or sent by u. This allows cells to be 'injected' into and/or 'tapped' from $\mathcal{G}$ through u by user entities, so u can be considered as a *data access point* (DAP). When a user U requests to be connected to $\mathcal{G}$, a routing algorithm in the network locates a node u that can serve as DAP for U and sets up a path from U to u. Here, u acts as a junction point and/or a branch point for data access through the path segment $e(u_l, u)$, based on whether U is a source and/or destination.

The tree-structured channel can be used as building block for constructing both multicast and unicast cell transport.

2.3 'Logical connectivity' versus 'data connections'

The notion of tree-structured 'connectivity' deals with protocols and mechanisms that allow various user entities to be connected over unique paths through the network. It *does not* refer to any particular level of reliability in cell transport (such as 'acknowledged cell delivery'). In comparison, the notion of 'connection', as used in many networks [3, 4], is architecturally at a higher level, embodying: i) cell routing protocols over the path determined by the 'connectivity', and ii) cell flow handling protocols to sustain data transport over this path (say, allocation of cell buffers at nodes for a specified data rate). The distinction between the notions of 'connectivity' and 'connection' has become sharper in the context of multi-service networks since the flow handling functions are parameterizable by 'user-oriented' specification of the transport characteristics of data (viz., QOS), while the data routing functions are parameterizable by 'network-oriented' configuration information concerning the placement of communicating user entities relative to one another in the physical topology of the backbone network. Such a separation of concerns permeates across all levels of the ATM node architecture (refer to Figure 2).

This paper deals with how bindings are set up from VCs to cell routes and flow handlers, but does not deal with route setup protocols or flow handling protocols as such.

We now describe the support functions required in an ATM network to realize our channel abstraction.

3 Support functions in ATM backbone networks

The ATM backbone connection layer consists of two native sublayers: the VC layer on top of the VP layer, whereby multiple VC links can be multiplexed on a given VP link. The embedding of connectivity functions on an ATM network requires providing the RCL functions partly in the lower layers of the 'access' component (ASL) and partly in the VC layer in the 'core' component of the network. See Figure 5. This section presents an extended view of ATM connection layer that allows incorporating the RCL functions.

3.1 VP level cell transport

The VP layer in the 'core' network may be viewed as providing physical transport of cells between switching nodes, with the following primitives invokable by a node:

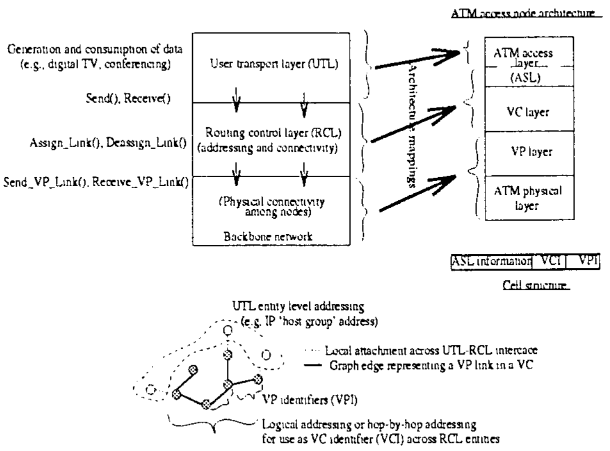
ATM access node architecture
Generation and consumption of data (e.g., digital TV, conferencing)
Send(), Receive()
Assign_Link(), Deassign_Link()
Send_VP_Link(), Receive_VP_Link()
User transport layer (UTL)
Routing control layer (RCL) (addressing and connectivity)
(Physical connectivity among nodes)
Backbone network
Architecture mappings
ATM access layer (ASL)
VC layer
VP layer
ATM physical layer
ASL information | VCI | VPI
Cell structure
UTL entity level addressing (e.g. IP 'host group' address)
Local attachment across UTL-RCL intercace
Graph edge representing a VP link in a VC
VP identifiers (VPI)
Logical addressing or hop-by-hop addressing for use as VC identifier (VCI) across RCL entities

Figure 5: Connectivity functions in ATM network architecture

Send_VP_Link(b_c, VPI), and
(b_c, VPI) = **Receive_VP_Link**

to send and receive respectively a cell b_c over the VP link referred to by VPI. The **Send_VP_Link** involves transmitting b_c as a bit stream over the physical medium on which the VP link is realized; the **Receive_VP_Link** involves extracting the bit stream seen on a physical medium as the contents of b_c and determine the VP link therefrom.

The 'core' network may support VP level multicasting, whereby a node S maps the $\text{VPI}_{S,in,n}$ carried in the header of an incoming cell over the n^{th} link to one or more outgoing links across which the cell is to be replicated for transmission, with a new $\text{VPI}_{S,n,l}$ affixed in the header of a cell replica forwarded over l^{th} outgoing link. This mapping may be represented as

$$\text{VPI}_{S,in,n} \rightarrow \{\text{VPI}_{S,n,l}\}_{l=1,2,\ldots,r_n}$$
$$\text{for } 1 \leq r_n \leq (K_S - 1) \text{ and } n = 1, 2, \ldots, K_S,$$

where '$\rightarrow$' denotes a binding relation on identifiers, K_S is the number of VP links supported by S. Note that the incoming VPI and the outgoing VPI on a link need not be the same (this is because the switches at either end of a link may assign their outgoing VPIs independently). The above form of VP level cell forwarding may be used in defining physical connectivity in the network.

There have been recent works on ATM network implementations that make use of VP level multicasting in the 'core' network to support multi-destination data delivery in applications such as conferencing and information distribution [5, 6, 7].

3.2 Assignment of VCI/VPI

The VCI is a distinct address assigned to each segment of a VC, and may be viewed as an index to the protocol

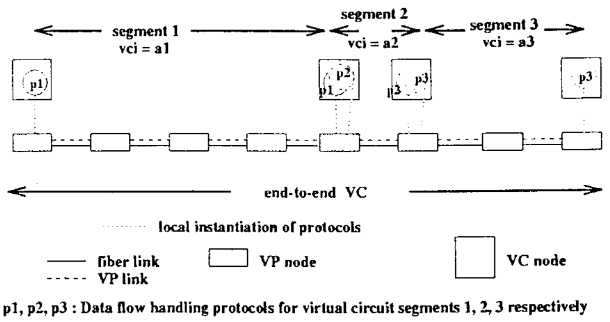
segment 1
vci = a1
segment 2
vci = a2
segment 3
vci = a3
p1
p2
p1
p3
p3
p3
end-to-end VC
local instantiation of protocols
fiber link
VP link
VP node
VC node
p1, p2, p3 : Data flow handling protocols for virtual circuit segments 1, 2, 3 respectively

Figure 6: VC segments and their protocol bindings in an ATM network

instance used for cell transport over a given VC. So a VC is bound to an ordered set of VCIs, with each VCI in turn binding to an ordered set of VPIs. For instance, the segments 1, 2 and 3 of the VC in Figure 6 may be assigned the VCIs a_1, a_2 and a_3 respectively; so the end-to-end VC is bound to $\{a_1, a_2, a_3\}$. We consider only the two most common cases, as given below, in this paper.

When a single instance of flow handling protocol is needed end-to-end in a VC, a networkwide logical address *addr* may be used to bind to each node and its incident links. In this case, the end-to-end VC may in its entirety be referenced by a single VCI, viz., *addr*. Accordingly, the same VCI is used at every 'core' node in the cell path. The cell flow handling procedures are installed in the ASL along C-plane-2. When a different flow handling protocol is needed in each 'core' node in the path of a VC, a hop-by-hop addressing may be used whereby each hop in the path is assigned a locally unique address ***hid***. In this case, the end-to-end VC may be referenced by a set of VCIs, viz., $\{$***hid***$\}$. So a VCI has a local scope meaningful only to the particular hop. The cell flow handling procedures are installed in the VC layer along C-plane-2, with the VCI in a given hop indexing to the local instance of flow handler.

The above addressing methods lead to logical address based cell routing and hop-by-hop cell routing respectively in the RCL.

4 Support functions required in RCL

A channel $\mathcal{G}$ is associated with a unique service name *srvc* of the application that is bound to either a network-wide logical address *addr* or a set of local hop addresses $\{hid\}$, denoted as

$$srvc \rightarrow addr \quad \text{or} \quad srvc \rightarrow \{hid\}.$$

The above binding exists across the UTL-RCL interface, with *addr*/*hid* used as VCI for cell routing.

4.1 UTL-RCL interface

The service name *srvc* may be specified at the time $\mathcal{G}$ is created by a user level management entity in an 'access node', and may be inherited by every node of $\mathcal{G}$. A user level data entity U may request path set up to $\mathcal{G}$ by specifying *srvc* in its request. A routing algorithm uses *srvc* as a key to locate a DAP in $\mathcal{G}$ for creating a path segment between U and the DAP. U may also specify a QOS in its request that binds to cell flow handling procedures, denoted as cfp, to be installed in the nodes of the path segment.

For U to send or receive data cells through a channel, the following primitives may be used:

$$\textbf{Send}(d\_c, srvc) \quad \text{and} \quad \textbf{Receive}(d\_c, srvc).$$

U generates *srvc* as an 'Interface Control Information' (ICI) associated with each cell sent/received across the UTL-RCL interface, to enable the routing functions identify the tree through which the cell needs to flow. The **Send** causes the cell *d_c* to be affixed with an ***addr*** or ***hid*** as the VCI, as the case may be, and sent over the tree. The **Receive** causes a cell *d_c* to be received over the channel.

4.2 RCL-VP layer interface

A function **Assign_Link**$(addr/hid, \mathrm{VPI}_j, cfp)$ is provided that creates a directed path from a switch S originating through the VP link VPI_j and incident on the neighbor S^j and installing the protocol cfp as the cell flow handler peer instances in S and S^j. The function causes S to forward an incoming cell with $addr/hid$ as the VCI by processing it through cfp and sending it over VPI_j (cfp may allocate transport resources, viz., buffers and link bandwidth, for S to receive cells and send them over $lnk(\mathrm{VPI}_j)$). To remove this path, a function **Deassign_Link**$(addr/hid, \mathrm{VPI}_j, cfp)$ is provided. The function causes S to stop forwarding cells over VPI_j and remove the cfp instances in S and S^j. The **Assign_Link** and **Deassign_Link** functions are executed during creation and deletion of path segments respectively.

The routing functions in the RCL maintains forwarding tables to decide on the VP links through which a cell arriving on a VP link is to be dispatched, as described below.

4.3 Routing tables in nodes

Consider the channels which are routed through the RCL entity S in a switch. For each of these channels, the routing point may prescribe a path to one or more neighbors of S. This information is maintained by S in a routing table, with each entry in the table corresponding to a channel. For a channel bound to address *addr/hid*, the corresponding table entry provides a mapping of the form

$$(addr \text{ or } hid) \rightarrow \{(x_1, \mathrm{VPI}_1, cfp_1), (x_2, \mathrm{VPI}_2, cfp_2), \ldots, (x_r, \mathrm{VPI}_r, cfp_r)\}$$

with $1 \leq r \leq K_S$, where VPI_r refers to the VP link for a path segment from S to a neighbor and x_r refers to the corresponding VC link (***addr*** in the case of logical address based routing and a new hop address $\boldsymbol{hid_r}$ in the case of hop-by-hop routing). We expect that $\{cfp_r\}_{\forall r}$ are instances of a single cell flow handler procedure cfp. The RCL functions **Assign_Link**$(addr/hid, \mathrm{VPI}_{r'}, cfp_{r'})$ and **Deassign_Link**$(addr/hid, \mathrm{VPI}_{r'}, cfp_{r'})$ executed by S allow a $(\mathrm{VPI}_{r'}, cfp_{r'})$ to be added to and removed from the mapping information respectively. The mapping information may be extracted on a cell-by-cell basis to bind ***addr/hid*** carried as the VCI of cells to the VPIs and to the functional instances of cell flow handlers. The lookup function to extract $\{(\mathrm{VPI}_r, cfp_r)\}$ from routing table is critical to the efficiency of cell routing through S.

We now describe how the RCL realizes end-to-end connectivity for channels set up over an ATM network.

5 Realization of connectivity functions

The RCL entities (which reside in the lower part of ASL and in the VC layer) communicate with one another to coordinate their actions for routing table updates. The communication is achieved by exchange of control cells over 'signalling VC' using the **Send_VP_Link** and **Receive_VP_Link** primitives.

To utilize the 'core' network facilities, the connectivity architecture may be grafted at the interface between the VC and VP sublayers, with the RCL functions embedded in the VC layer and the UTL functions in the layers above. This architectural realization is described below.

5.1 VP level connectivity

Let $\{A_i\}_{i=1,2,\ldots,R \text{ for } R\geq 2}$ be the 'access' nodes of an ATM network. Since only the VC/VP links are distinctly identifiable in the ATM connection layer but not the switches, the physical connectivity among the A_i's, as seen by a routing protocol, needs to be realized using the VC/VP abstraction. For this purpose, each switch in the physical configuration allocates a set of VPs, each of which maps to a distinct transmission medium $lnk(\mathrm{VP})$ connected to the switch. These VPs, which we refer to as P_CONN VPs, are set up each time when the ATM network is initialized and/or reconfigured.

Since connectivity among switches can change asynchronously due to dynamic reconfigurations in the network, it is necessary that the various P_CONN VPs share a well-known networkwide broadcast VC link, say, with VCI vc'. Accordingly, the physical connectivity table (PCT) of a switch S may consist of information to bind vc' to

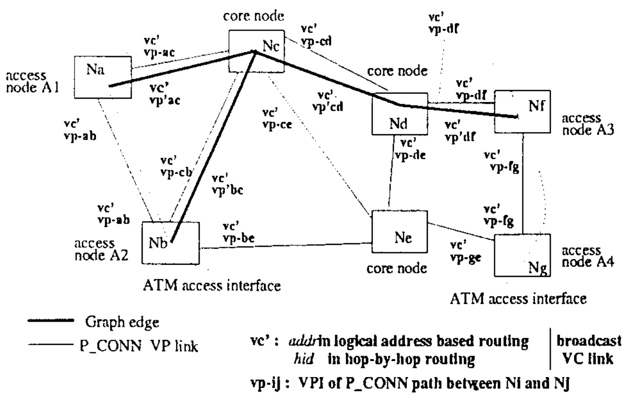

Figure 7: VP level connectivity in ATM networks

the locally generated VP links connecting to neighboring switches in the VP level connectivity protocols. This information may be represented as:

$$(S,\ (vc', VPI'_{S,in,n}) \rightarrow vc'|_{n=1,2,\ldots,K_S},$$
$$VPI'_{S,in,n} \rightarrow \{VPI'_{S,n,l}\}_{l=1,2,\ldots,K_S-1}|_{n=1,2,\ldots,K_S}),$$

where $vc'/VPI'_{S,in,n}$ is the VC/VP level header of an incoming cell over the n^{th} P_CONN VP link and $VPI'_{S,n,l}$ is a P_CONN VP link for outgoing cells from S to the l^{th} neighbor relative to the n^{th} link. Accordingly, a cell received by S with vc' and $VPI'_{S,in,n}$ as the VC and VP header respectively will be sent over each of the $(K_S - 1)$ outgoing links, with vc' and $VPI'_{S,n,l}$ as the VC and VP headers respectively over the l^{th} outgoing link. It may be seen that the PCT of S is a specific subset of the VCI/VPI tables in S.

Using the above form of physical connectivity, a routing protocol may have a node exchange control messages (in the form of cells) with another node to create a VP/VC link for data transport between these nodes. Typically, the incoming and outgoing cells at a node S may be part of a physical connectivity management protocol or a routing protocol executed by S. If S is an 'access' node executing the protocol, it binds a UTL address *srvc* to vc' for use as VC level header of cells exchanged across the UTL-RCL interface.

Consider the interconnection topology shown in Figure 7 as an illustration. The 'access' nodes A_1, A_2, A_3, A_4 can reach one another through the 'core' nodes $N_a, N_b, N_c, N_d, N_e, N_f, N_g$. The PCT of, say, A_2 residing in 'core' node N_b will be based on the VC link vc' and a set of VP links $\{vp_{ba}, vp_{bc}, vp_{be}\}$ (in addition to the VPI assigned for the UTL-RCL interface in A_2), where vp_{ba} is the P_CONN VP link between N_b and N_a, vp_{bc} is the P_CONN VP link between N_b and N_c, and so on. When the RCL function in A_2 broadcasts a cell, say, to locate a DAP for path set up between A_2 and the channel, the cell is sent over all the VP links $vp_{ba}, vp_{bc}, vp_{be}$; when A_2 receives such a cell over a VP link vp_{ab}, it may send the cell over the VP links vp_{bc}, vp_{be} (we assume that the same VPI is used in either direction of a VP link); in both cases, the cell VC header carries vc'.

5.2 Embedding RCL functions in VC layer

A path segment is a subtree consisting of 'access' nodes $\{A_i\}_{i=1,2,\ldots,R}$ and intermediate 'core' nodes, with the path between two adjacent nodes realized over the VP link between them; so a VP link may be shared by the cells generated by various A_i. The VP links provide the logical connectivity among A_i to overlay the data routing related functions. See Figure 8.

When the 'core' network needs only to provide a single instance of protocol for cell flow handling at various switches, a single VC link across the entire 'core' network, i.e., with no intermediate 'transfer point' along the data path, suffices to embed our RCL model on ATM networks. The embedding requires implementing an access protocol to a shared VC link. To send a cell on a shared VC, the RCL in an 'access' node A_i initializes the VCI and VPI fields of the cell respectively with the logical address *addr* of the channel and the VPI through which A_i is connected to the data path (i.e., this VPI is A_i's local reference to the path). Accordingly, a channel that connects A_1, A_2, A_3, A_4 through the 'core' nodes may be represented as:

$$([vp'_{ac}, vp'_{bc}], vp'_{cd}, [vp'_{df}, (vp'_{de}, vp'_{eg})]),$$

where $vp'_{ac}, vp'_{bc}, \ldots$ refer to VP links between N_a and N_c, between N_b and N_c, ... respectively that contain a path segment each. The VP link vp'_{cd} is shared by the cells sent from A_1 and A_2 (or from A_3 and A_4). The *addr*+VPI indexes to the routing information for the tree rooted at A_i. For this purpose, the VPI space at a switch is partitioned into disjoint subspaces, each bound to a distinct logical address *addr* of channels routed through the switch.

When the 'core' network needs to provide separate protocol instances for cell flow handling at various switches, a distinct VC link needs to be associated with each path segment in a channel. This is a case of hop-by-hop addressing of cells for routing through the channel. Accordingly, the channel shown in Figure 8 may be represented as:

$$([(vc'_{ac}, vp'_{ac}), (vc'_{bc}, vp'_{bc})], (vc'_{cd}, vp'_{cd}),$$
$$[(vc'_{df}, vp'_{df}), ((vc'_{de}, vp'_{de}), (vc'_{eg}, vp'_{eg}))]),$$

where each segment is now bound to a (VCI,VPI) pair, with a VCI $vc'_{..}$ being the hop address for the path segment realized over the corresponding VP link.

In general, we expect that the use of a single VC to provide connectivity across the entire network will be the most common case. This because a single protocol instance may suffice for end-to-end cell flow handling in many applications.

We now discuss the connectivity management protocols that may be implemented in various layers.

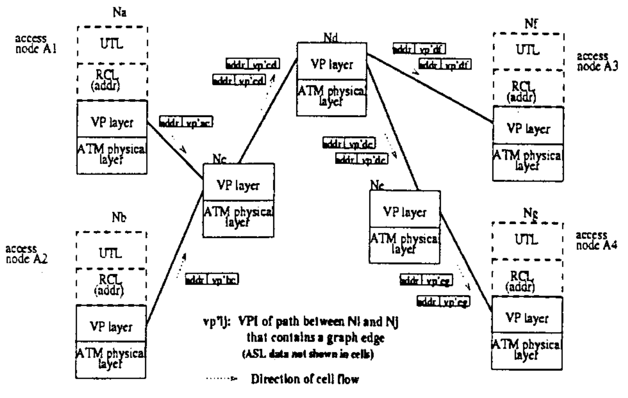

Figure 8: A sample scenario for routing of ATM cells (with logical addressing)

6 Connectivity management protocols

The embedding of our connectivity architecture on an ATM network is along two different projections of the VP-VC layer interface (refer to Figure 2) consisting of: i) transport functions that deal with cell switching and cell flow handling over the VP/VC links, and ii) routing functions that deal with path setup over the VP/VC links. These functions are described below (we assume, without loss of generality, the use of a networkwide logical addressing of channels).

6.1 VP level protocols in data plane

The VPI vp_i' affixed in cells sent by an 'access' node A_i gets mapped to different VPIs as the cells are forwarded at various intermediate VP switches. A VP switch examines the VPI_{in} available in the header of an incoming cell to determine the links and the VPs on which the cell is to be sent out, indicated as a list $VPI_{in} \rightarrow \{(lnk(VPI_{out}), VPI_{out})\}$; the VP switch sends out the cell by replicating the cell data on each $lnk(VPI_{out})$ and affixing the cell header with VPI_{out}. The mapping can be many-to-one when the VP switch is a junction point in the channel. The VPI tables for the $VPI_{in} \rightarrow \{(VPI_{out}, lnk(VPI_{out}))\}$ mappings appear flat in the data plane since the bindings between various hops of a path segment for *addr* and its corresponding VPIs are implicitly established when the VPI tables are set up. The VCI field of each cell, which carries the logical address *addr*, however does not change along the entire path between A_i's. Refer to Figure 8 for an illustration of cell forwarding over the channel.

6.2 VP level protocols in control plane

In the control plane, the VPI tables describing the path segment through the ATM network are indexed by the logical address *addr* associated with the channel. A path setup protocol may be executed on a 'signalling channel' of the core network to load VPI tables of various switches with routing information that map the VPI_{in} of incoming cells to $\{VPI_{out}, lnk(VPI_{out})\}$. Suppose, for instance, a search based algorithm is used to locate a DAP node for *addr* in the network and set up route to this node [8]. The search messages are broadcast by an 'access' node A_i over its P_CONN VP links and VC link vc' for propagation through the network. Upon locating a switch S that has a VPI table indexed by *addr*, a path is set up between A_i and S which involves creating an entry in the VPI table of S indexed on *addr* such that: i) the new path gets included in $\{VPI_{out}, lnk(VPI_{out})\}$ for each VPI_{in} at S — as part of an Assign_Link($addr, VPI_{out}, \ldots$) invocation, and ii) cells arriving at S over the new path get forwarded over various existing $\{VPI_{out}, lnk(VPI_{out})\}$. Referring to Figure 7, the channel is $([vp'_{ac}, vp'_{bc}], vp'_{cd}, vp'_{df})$ with RCL functions in N_a, N_b, N_c, N_d, N_f. A_4 residing in 'core' node N_g creates a path segment to a DAP at N_d through N_e. This involves N_d updating its VPI table entry indexed on *addr* from $(vp'_{cd} \rightarrow \{vp'_{df}\})$ to $(vp'_{cd} \rightarrow \{vp'_{df}, vp'_{de}\})$, N_e creating a new VPI table entry indexed on *addr* in the form $(vp'_{de} \rightarrow \{vp'_{eg}\})$, and A_4 creating a VCI table entry $(addr \rightarrow vp'_{ge})$. Similarly, a Deassign_Link($addr, VPI_{out}, \ldots$) invocation causes the deletion of the reference VPI_{out} for the VC bound to *addr* and the underlying VPs.

As can be seen, the native support for multicasting provided by the 'core' network can be exploited by the embedding of our architecture at the VC-VP layer interface[1].

7 Conclusions

In future, it is likely that ATM-based cell switching technology will be used in fiber-optic backbone networks to support high speed multi-service applications. In these networks, maintaining logical connectivity among subscriber terminals through cell switching nodes and VP links becomes a major requirement. In comparison to the well-known 'virtual circuit' routing techniques used in current networks (e.g., ARPANET), maintaining logical connectivity in an ATM network needs to tackle two additional issues: support for multicasting of cells and efficient multiplexing of cells over a link. In the context of these issues, the paper described a generalized model of logical connec-

[1]The work on connection management over ATM networks [9] deals only with the ATM 'access' level functions but not the 'core' functions. In contrast, our work deals with extending the 'core' multicast facility to an 'access' level data routing functionality.

tivity among user entities through ATM-based backbone networks.

Our model is based on tree-structured communication channels connecting data sources and destinations residing in 'access' nodes, with routing functions in one or more cell switching nodes and transport functions over the intervening VP links of the 'core' network. Each node employs a cell routing protocol and an appropriate cell flow handling protocol to support the incident VP links. The address associated with a cell (carried as VCI) is mapped at each node traversed by the cell to the set of links constituting the path through which the cell is routed. The functional binding of addresses to cell flow handlers at various nodes in a path, as underscored by our model, allows flexible implementations of application configurations on ATM networks. In addition, the model treats multicast and unicast transport of cells in a uniform manner, as advocated in P-NNI [10]. In comparison to our signaling-oriented modeling of VP level connections in a multicast context, [11] deals with a mathematical characterization of the extent of connection load on various VP links in a path.

The architectural solutions provided in the paper offer potential for extensible, scalable and efficient implementations of data connections on target networks by allowing: i) the 'access' nodes to insulate from the 'core' network, where necessary, for cell transport (this feature is useful for mobile networks), ii) the 'core' nodes to establish their local bindings to a logical address (or hop-by-hop address) carried in a cell as VCI, and iii) the 'core' network to choose an efficient implementation of the binding of a cell address to a path. Our model of connectivity can also be adapted to other types of networks such as interconnected LANs and high speed 'data service' networks (e.g., SMDS).

References

[1] J. Y. Le Boudec. **The Asynchronous Transfer Mode: A Tutorial.** In *Computer Networks and ISDN Systems*, vol.24, pp.279-309, 1992.

[2] U. Black. **The Network Layer: Internetworking.** In chapter 6, *OSI: A Model for Computer Communication Standards*, Prentice-Hall Publ. Co., 1991.

[3] C. Topolcic. **Experimental Internet Stream Protocol, version 2 (ST-II).** In *RFC 1190*, CIP Working Group, Oct. 1990.

[4] L. Zhang, S. E. Deering, D. Estrin. S. Shenker and D. Zappala. **RSVP: A New Resource ReSerVation Protocol** In *Network*, IEEE Communications Society, pp.8-18, Sept. 1993.

[5] I. Gopal, R. Guerin, J. Janniello and V. Theoharakis. **ATM Support in a Transparent Network.** In *IEEE Network*, pp.62-68, Nov. 1992.

[6] E. Biagioni, E. Cooper and R. Sansom. **Designing a Practical ATM LAN.** In *IEEE Network*, pp.32-39, March 1993.

[7] J. R. Cox, M. E. Gaddis and J. S. Turner. **Project Zeus.** In *IEEE Network*, pp.20-30, March 1993.

[8] K. Ravindran and M. Sankhla. **Multicast Models and Routing Algorithms for High Speed Multi-service Networks.** In *International Conf. on Distributed Computing Systems*, IEEE-CS, Yokohama (Japan), June 1992.

[9] L. A. Crutcher and A. G. Waters. **Connection Management for an ATM Network.** In *IEEE Network*, pp.42-55, Nov. 1992.

[10] ATM Forum Technical Committee. **Private Network-Network Interface Specification.** In PNNI SWG, Version 1.0, March 1996.

[11] O. Gerstel, I. Cidon, and S. Zaks. **The Layout of Virtual Paths in ATM Networks.** In *IEEE/ACM Transactions on Networking*, vol.4, no.6, pp.873-884, Dec. 1996.