22

DIVA: A DIstributed & dynamic VP management Algorithm

S. Srinivasan and M. Veeraraghavan
Bell Laboratories, Lucent Technologies
101 Crawfords Corner Road, Holmdel, NJ 07733
E-mail: {cheenu,mv}@bell-labs.com

Abstract

The concept of preestablishing Virtual Path Connections (VPCs) in ATM networks offers a number of advantages, such as simplified on-demand connection setup and fault management. However, preallocation of bandwidth resources to VPCs minimizes resource sharing and leads to poor resource utilization. This is especially true if the VPC bandwidth allocations are computed allowing for some uncertainty in traffic characterization. In this paper, we propose DIVA, an algorithm for distributed and dynamic VPC bandwidth management. DIVA alleviates the problem of poor VPC resource utilization by dynamically adjusting VPC bandwidth allocations, thus minimizing the effect of traffic uncertainty. This algorithm is proposed for hierarchical networks based on the ATM Forum's PNNI routing standard. Using hooks provided by PNNI routing, network nodes monitor VPC bandwidth usage, make dynamic VPC bandwidth/buffer modifications, and set up and remove VPCs dynamically. This algorithm also handles the additions and failures of network elements such as links and switches.

Keywords

ATM networks, virtual path management, dynamic/distributed algorithm, hierarchical networks, PNNI

1 INTRODUCTION

Provisioned Virtual Path Connections (VPCs) in ATM networks offer a number of advantages: (a) reduced on-demand connection setup time, (b) fast rerouting of bundles of Virtual Channel Connections (VCCs), useful in networks with high reliability requirements, (c) potential reduction in "switching" costs by using crossconnects without expensive call processing software in parts of the network instead of more expensive switches, and (d) in particular application areas, such as wireless ATM networks, where provisioning VPCs between adjacent pairs of base-stations allows mobile handoffs to be simplified (Srinivasan and Veeraraghavan, 1996).

On the other hand, a drawback of using provisioned VPCs with preallocated capacity is that network utilization (transmission efficiency) is reduced since link and node resources are partitioned (rather than shared) (Veeraraghavan et al., 1996). This effect is especially true if the resource allocations to VPCs are made allowing for uncertainty in traffic charac-

A. A. Lazar et al. (eds.), *Integrated Network Management V*

terization. The effect of varying traffic patterns can be sharply reduced by using a scheme which dynamically adjusts VPC resource allocations based on usage. The presence of such a scheme will reduce the transmission inefficiency factor of using provisioned VPCs with preallocated capacity. Besides being dynamic, the VPC management scheme needs to be distributed for the reason that a centralized solution does not scale well with the size of the network, and is also very poor from a fault tolerance perspective. In this paper, we propose a *dynamic and distributed algorithm for monitoring and managing VPCs.*

Dynamic management of VPC routes and resource allocations can be done by continuously monitoring the network and reacting to repeated congestion patterns and topological changes caused by failures and addition of network elements such as links and nodes. The standardization of the Simple Network Management Protocol (SNMP) (Stallings, 1993) and the ATM Management Information Base (MIB) (Ahmed and Tesink, 1994) provides one method to achieve this. However, constantly reading and writing MIB variables for this purpose is typically too burdensome to the switch agents and is also too slow. An alternative to gathering network state information by reading MIB variables is provided by the ATM Forum's PNNI routing protocol (ATM Forum, 1996), which enables information about the network to be gathered and disseminated in a scalable manner. Hence, DIVA is proposed for hierarchical networks based on the PNNI routing standard. Using hooks provided by PNNI routing, network nodes monitor VPC bandwidth usage, make dynamic VPC bandwidth/buffer modifications, and set up and remove VPCs dynamically. Note that, currently, there is no solution for dynamic VPC management in the PNNI standard. One approach to realize changes in VPC routes/allocations is by using the switch signaling software. However, the use of sequential node-by-node setup and configuration procedure can result in excessive VPC adjustment delays. This approach also requires additional software to monitor VPCs and initiate adjustments, on already overloaded switches. The Parallel Connection Control (PCC) scheme (Veeraraghavan et al., 1996), originally proposed for on-demand connection set up, provides solutions for both these drawbacks. First, PCC recognizes the inherent parallelism in the connection establishment procedure and executes actions at a number of switches in *parallel* rather than in sequence. This concept can be used for fast VPC setup/modification. A second aspect of PCC is that it separates some of the connection management functions into *connection servers*, distinct from switches. Hence, the additional software for monitoring VPCs and initiating adjustments can be moved to the connection servers.

DIVA utilizes these advantages of PCC and the hooks provided by PNNI routing to provide a scalable solution for the VPC management problem. In other words, DIVA is targeted at networks that use PNNI and PCC for switched connection management. Specifically, it addresses the following issues.

- Where should VPCs be laid and what are the parameters needed?
- How should VPCs be realized? To answer this, we need to answer the following two questions: (a) How should VPCs be routed? (b) How much bandwidth and buffer resources should be allocated to each VPC?
- When should adjustments be made to VPC routes/allocations?
- How are changes to VPC configurations accomplished in reaction to link and node (a) failures, and (b) additions and restorals?

Past approaches for the creation and maintenance of VPCs are the following.

- Centralized optimization based schemes (Siebenhaar, 1994, Cheng and Lin, 1994, Faragó et al., 1995, Logothetis and Shioda, 1995): These are based on the solution of a non-linear optimization with non-linear constraints. Such optimizations are computationally intractable and approximate models are used which retract from the optimality while still not simplifying the schemes sufficiently to allow them to be run often enough, further compromising on the optimality of the assigned bandwidths. The centralized nature of these approaches has drawbacks from the points of view of scalability and fault tolerance.
- Distributed heuristic (Shioda and Uose, 1991): The drawbacks of this scheme are the following. (a) Only one VPC is allowed between each pair of switches which is unreasonable in practice since the required bandwidth to accommodate all the traffic between a pair of switches may not be available on a single route. It is also not acceptable from a fault tolerance perspective. (b) The route of each VPC is precomputed and fixed. In practice, network state is constantly changing and so does the "best" route between a pair of nodes. Hence, a VPC management scheme should allow for VPCs to be rerouted periodically.

None of the above schemes addresses the issue of dynamically handling network state changes such as link and node *failures* and *additions*. Lastly, and perhaps most importantly, any solution to the VPC management problem needs to address the practical issue of interworking with existing standards and/or systems. There is no attempt in this direction in (Siebenhaar, 1994, Cheng and Lin, 1994, Faragó et al., 1995, Logothetis and Shioda, 1995, Shioda and Uose, 1991), which detracts from their practicality.

In our work, we aim for a solution to the VPC bandwidth/buffer/route management problem that has the following properties.

- The solution should be scalable with respect to the size of the network. This precludes any centralized approaches.
- The solution should be robust and be capable of handling network state changes such as network element additions and failures.
- It should not assume fixed precomputed routing of the VPCs. VPCs should be routed according to the "best" paths, which could change with time as the network state changes.
- It should be able to take advantage of existing standards and interwork with them.

The rest of the paper is organized as follows. Section 2 defines some terms used in the rest of the paper. Section 3 describes the details of DIVA, our algorithm for VPC management, and Section 4 summarizes the contributions of the paper.

2 TERMINOLOGY AND DEFINITIONS

We model the call arrival processes as *Markov Modulated* processes (Shroff and Schwartz, 1996, Choudhury et al., 1996). Each source is characterized by the tuple $[M, R]$, where M is the K-state transition matrix, and R is a $K \times 1$ vector of source intensities.

When cells from multiple sources arrive at a switch, the cells are stored in a *buffer* till the destination port/link is free. Cells may be lost due to *buffer overflow* of finite

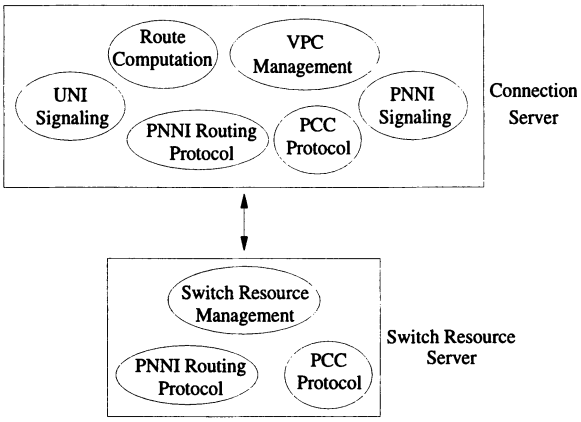

Figure 1 Distribution of functions between connection server and switch resource server.

sized buffers and the stochastic nature of the sources. An important QoS criterion for a connection is the *Cell Loss Ratio* (CLR), which is the ratio of the total number of cells that are dropped due to buffer overflow to the total number of cells that arrive at the switch. Cells also experience a *Cell Delay Variation* (CDV), which is the time spent in a buffer, before being switched out. We make a distinction between the end-to-end CLR and CDV constraints for a connection and a VPC. We refer to the former simply as the end-to-end CLR/CDV constraints and to the latter as the end-to-end VPC CLR/CDV constraints.

The *equivalent bandwidth* of a set S_n, of n sources, is the constant rate $c_{eq}(b, \theta, S_n)$ at which a buffer of size b cells should be emptied so that the CLR experienced by the source is less than θ. Expressions for $c_{eq}(.)$ exist in the literature, for instance in (Guérin et al., 1991).

Calls arriving at a VPC are modeled as Poisson processes with a mean rate λ. The call departure process is also Poisson with mean μ. The ratio $\rho = \frac{\lambda}{\mu}$ is called the *offered traffic*. A call gets blocked on the VPC when the resources available on the link are inadequate to support the call. The probability that this happens, is termed as the *Call Blocking Probability* (CBP).

VPCs are of two types, *homogeneous* if they only support calls of a single traffic class, or *heterogeneous* if they support multiple classes of traffic. For a homogeneous VPC capable of supporting ν calls, the CBP is a function of ν and ρ. For a heterogeneous VPC with k classes capable of supporting ν_j calls of class j with offered traffic ρ_j, $j \in \{1, \cdots, k\}$, the CBP is a function of $\nu_1, \cdots, \nu_k$ and $\rho_1, \cdots, \rho_k$ (Kaufman, 1981, Labourdette and Hart, 1992, Choudhury et al., 1995).

3 DYNAMIC AND DISTRIBUTED VPC MANAGEMENT

In this section, we describe the network architecture assumed, the procedure for exchange and dissemination of routing information inside the network, and our algorithm, DIVA, for managing VPCs in a hierarchically organized network.

3.1 Network architecture

DIVA is designed for networks that use PNNI and Parallel Connection Control (PCC) (Veeraraghavan et al., 1996) for switched connection management. The network nodes are gathered into hierarchical *Peer Groups* (PGs) for scalability reasons. Nodes are of two types, switches and *connection servers* (CSs). CSs collect and disseminate routing information, compute routes, and coordinate parallel connection setup inside a PG. Switches run *switch resource servers* (SRSs) which perform switch resource management functions, consisting of (a) Connection Admission Control (CAC) for buffer and bandwidth resource allocations, (b) selecting incoming and outgoing VPIs/VCIs for VCCs, or incoming and outgoing VPIs for VPCs, (c) configuring the switch fabric by setting Port/VPI/VCI translation tables for VCCs or Port/VPI translation tables for VPCs, and (d) programming parameters for various runtime (user-plane) algorithms, such as cell scheduling, per VC queuing or per-VP queuing, priority control, rate control, etc. Software implementing UNI (User Network Interface) signaling (The ATM Forum, 1994), PNNI signaling (for PG to PG connection setup), PNNI routing (for communicating current topology state information), and PCC (for intra-PG connection setup) are distributed between the SRSs and CSs, as shown in Figure 1.

The function of VPC monitoring and adjustment initiation, required for VPC management, is also assigned to connection servers, which initiate VPC modifications/setups in response to topological changes reported via the PNNI routing protocol messages. For this added functionality, a *primary connection server* is assigned to each switch. Any of the other connection servers in the same PG act as *alternate connection servers* in case of failures. (These details are described in Section 3.3).

Each PG in the network has a CS designated as the Peer Group Leader (PGL). A PGL of a PG at level L represents this PG at level $L+1$ as a *logical group node* (LGN). Two LGNs may be connected to each other by a *logical link* which is an aggregate representation of one or more links between the corresponding PGs at the lower level. PGLs obtain routing information about the switches in their peer groups and propagate a condensed version of this information to its peers in the higher-level peer group.

Figure 2 shows the PNNI/PCC network architecture assumed for our VPC management algorithm.

3.2 Inter-node routing information exchange

As described in Section 3.1, DIVA utilizes hooks in the PNNI routing protocol to gather periodic updates about network topology and state. This section describes how the PNNI routing protocol is executed in the network architecture shown in Figure 2 with switches and connection servers.

PNNI routing messages are of two types, *Hello* packets and PNNI Topology State Packets (PTSPs) (ATM Forum, 1996). Nodes in the network run a modified version of the *Hello protocol* (ATM Forum, 1996). *Hello* packets are exchanged across each link that comes up and the nodes attached to the link exchange information to establish their identity as well as their PG membership. Each switch also informs each of its neighbors belonging to a different PG the identity of its primary connection server. This piece of information propagates to the connection servers in the adjacent PG and is used for setting up connections spanning multiple PGs (as will be explained later). Neighboring

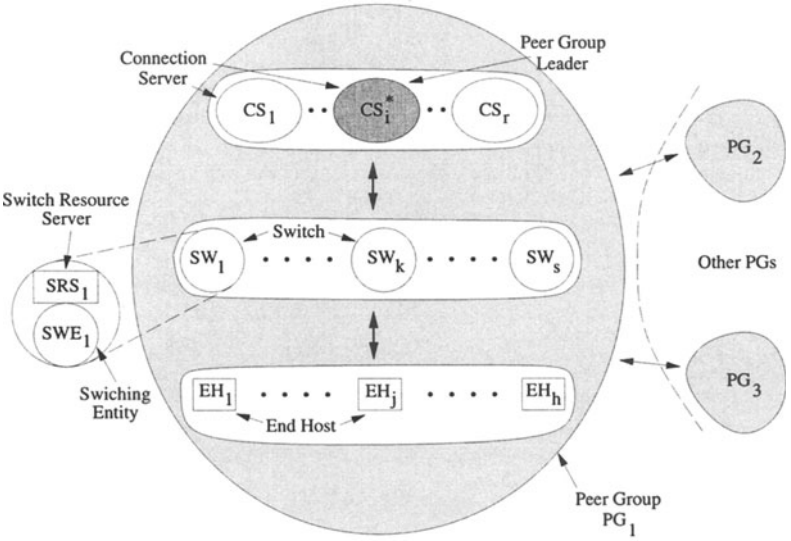

Figure 2 Architecture of a peer group.

nodes also exchange periodic *Hello* packets which serve as "heartbeats" to monitor the health of the neighboring node.

Switches in the network create PNNI Topology State Elements (PTSEs) about their *local* state, namely about their own identity and capabilities. Switches *do not* maintain topology state information and hence do not exchange such information with each other. Instead these PTSEs are encapsulated into PTSPs and sent to the primary connection server of the switch. These PTSEs also contain topology state parameters describing the state of links associated with the node in question as well as all the VPCs originating at the nodes. This information is used by the connection server to perform VPC bandwidth management functions.

The connection servers in a PG exchange topology state and reachability information with each other in order to synchronize their topology databases. Each connection server reliably floods the connection servers in its PG with detailed information about the state of the nodes for which it is the primary connection server as well as links associated with these nodes. This information is used by the connection servers to perform detailed route computation inside the PG. Also, the PGL condenses this information and floods it in its PG. The PGL also receives such condensed information about other PGs and floods this information among all the nodes in its child PG if this is not a lowest level PG and among the different connection servers in a lowest level PG.

PTSEs in the topology databases of the connection servers are subject to aging rules and deletion just as in PNNI based networks.

3.3 DIVA: Algorithm for VPC management

In this section, we present the details of our VPC management algorithm, DIVA.

Where should VPCs be laid and what are the parameters needed?

The following parameters have to be determined by or specified to the connection server which sets up the VPC. (a) VPC type, namely heterogeneous or homogeneous, (b) the source characteristics, as the set of $[M, R]$ tuples for each source, (c) the end-to-end CLR and CDV constraints τ and θ, for each class of traffic supported by the VPC, and (e) the number of calls of each class of traffic, either directly specified or indirectly specified by giving the CBP constraints.

The pairs of switches i and j to be connected by VPCs can be determined by one of the following methods.

1. **Approach based on traffic monitoring:** As explained in Section 3.1, one of the functions performed by the connection servers is to receive UNI signaling messages requesting SVC setup and determine routes within its PG to set up SVCs. Thus each connection server CS knows the number of on-demand connections set up between each switch SW for which it is the primary connection server and other switches in the PG. If there is no VPC between switches SW and SW' and the number of connection setup requests over a certain window of time $\tau_{monitor}$ exceeds a threshold r_{max}, CS decides that SW and SW' need to be connected by one or more VPC.
   Besides determining the node pairs that should be connected by a VPC, some of the parameters of the VPC can also be determined from the SVC setup and release data, namely, source characteristics, end-to-end VPC CLR and CDV constraints, the CBP constraint on the VPC and the offered traffic. The details of how these parameters are determined are deferred to a later paper. The type of VPC to set up as well as the CBP constraint, either system wide or for each node pair, have to specified to the connection servers.
2. **Rule based VPC approach:** For certain applications, we may decide to have VPCs between certain pairs of nodes from the very beginning, based on a certain set of prespecified rules. For example, in IP-over-ATM networks we can avoid connection setup to transfer connectionless IP packets by connecting each pair of IP routers by a VPC.
3. **VPCs requests from an administrator/manager:** Each such request has to specify all the parameters enumerated above, for every pair of switches, i and m, that have to be connected by one or more VPCs.

The above three approaches are not mutually exclusive. While a given approach may work better for certain types of VPCs, it may not be applicable for others. The complete answer to the question of where to lay the VPCs is a combination of all three approaches.

How are VPCs realized?

The two questions related to the realization of VPCs are: (a) How should VPCs be routed?, and (b) How much bandwidth and buffer resources should be allocated to each VPC? Let us assume that a connection server needs to realize a sufficient number of VPCs between two switches i and m with the given set of parameters defined above. Connection servers and switch resource servers are involved in the VPC-realization phase. The following steps are involved in setting up a VPC.

Step 1: First, the connection server needs to *determine the number of calls* that must

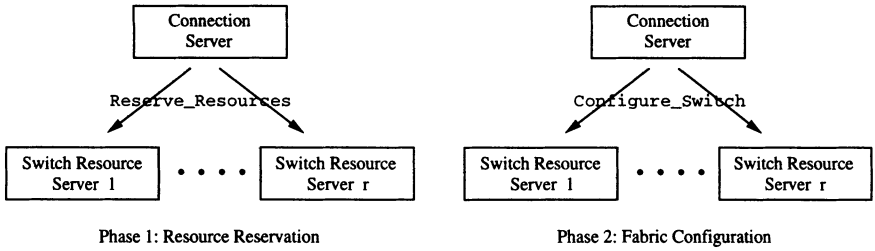

Figure 3 VPC setup within a PG.

be accommodated on these VPCs. This data may be directly provided in the request or it may need to be inferred by the connection server from the specified call blocking probabilities and offered traffic. The number of calls can be computed using the Erlang-B formula (Jagerman, 1974) for homogeneous VPCs and using a multiservice CBP method (Kaufman, 1981, Labourdette and Hart, 1992, Choudhury et al., 1995) for heterogeneous VPCs.

Step 2: Next, the connection server performs *route computation* using routing information (reachability and the state of different elements) previously collected via PNNI routing protocol messages. It uses an estimate of the required bandwidth derived from the source descriptions to determine the "best" (most likely to succeed) "shortest path" routes for the VPCs. It then attempts to set up the VPC along one of these routes.

Step 3: For each selected route, the connection server needs to *apportion the end-to-end VPC CLR and CDV*, for traffic class j, θ_j^{tot} and τ_j^{tot} respectively, among the switches on the route. This is a complex problem (Nagarajan, 1993), especially if switches from multiple vendors are to be accommodated. One simple solution, based on estimating the number of switches n_s the VPC passes through, is to apportion the end-to-end class j VPC CDV constraint τ_j^{tot} equally among all the switches giving a per-switch constraint of $\frac{\tau_j^{tot}}{n_s}$. Similarly, if the end-to-end class j VPC CLR constraint θ_j^{tot} is equally distributed among the n_s switches, the CLR constraint on each switch is $1-(1-\theta_j^{tot})^{\frac{1}{n_s}}$.

Step 4: VPC setup is done using parallel setup of switches within a PG and sequential setup from one PG to the next. Connection setup within a PG consists of two phases: a *resource reservation phase* and a *fabric configuration phase*, as shown in Figure 3. In the *resource reservation phase*, the connection server sends a `Reserve_Resources` message to the SRSs (Switch Resource Servers) on the switches of the selected route, *in parallel*, requesting them to set aside the necessary resources. Each SRS that receives such requests pushes them into a queue and processes them sequentially. This allows multiple connection servers to simultaneously contact an SRS without causing any deadlocks (Veeraraghavan et al., 1996). Each `Reserve_Resources` message specifies the parameters $[M_j, R_j]$, θ_j, τ_j, and ν_j for each service class j. Each SRS then performs CAC (Connection Admission Control) functions to determine if it can satisfy the request. To do this, it must determine the required bandwidth and buffer resources from the parameters specified in the `Reserve_Resource` message. Buffer and bandwidth allocations are performed simultaneously by the SRS on each switch within the constraints of (a) the available switch buffer and bandwidth resources, and (b) the cell loss and delay variation constraints on the connection. If the SRS determines that it has sufficient resources to meet these re-

quirements, it sets aside these resources and marks them as unavailable. It also selects VPIs to be used for this VPC. It then sends a `Resources_Available` message back to the requesting connection server. If the requested resources are not available, it responds with a `Resources_Unavailable` message.

Step 5: If the first phase is successful, responses carrying actual delay guarantees are received by the connection server which uses these to compute the residual constraints to be satisfied by switches in PGs further down the route. Then the connection server (A.2.5 in the example) enters the second phase by sending a `Configure_Switch` message in parallel to all the switches on the route, as shown in Figure 3. An SRS receiving this message sets port/VPI translation table entries. In other words, it configures the switch fabric to realize the VPC. It also sets buffer allocations for the VPCs (assuming per-VP queuing in the nodes) and provides the allocated bandwidth to the switch scheduler for enforcement.

For multi-PG VPCs, a connection server in each PG performs this two phase setup approach for the switches in its PG. PG to PG setup proceeds sequentially using PNNI signaling messages for communication between connection servers in different PGs.

Once the above 5 steps are completed successfully in the last PG along the route of the VPC, appropriate data tables are created at the VPC terminating switches, by the connection servers in the PGs of these switches, to enable them to treat each other as "logically connected" neighbors. □

When and how should adjustments be made to VPC routes and resource allocations?

This determination is done using one of two approaches: *by monitoring call blocking probabilities* or *by monitoring VPC cell usage.* Using the periodic updates received from each switch SW for which it is the primary connection server, each connection server CS monitors the state of each VPC associated with the switches. If, in the previous time-frame, either (a) the observed CBP β_{meas} on a VPC is not within a specified range $[(1-\rho_l)\times\beta, (1+\rho_u)\times\beta]$ where the target CBP for the VPC is β, or the average occupied bandwidth c_{occ} is not in the range $[min \times c_{actual}, max \times c_{actual}]$, the primary connection server of one of the terminating switches of the VPC takes corrective action.

When $\beta_{meas} < (1-\rho_l)\times\beta$ or $c_{occ} < min \times c_{actual}$, it attempts to reduce the allocated bandwidth. The connection server uses the two-phase approach described in Section 3.3 to realize this adjustment of lowering the assigned bandwidth (expressed as a percentage in the `Reserve_Resources` message). The exception to this bandwidth reduction step is when the occupied VPC bandwidth is smaller than a minimum c_{min}. In such a case the connection server marks the VPC as "frozen". No more connections are admitted on this VPC and it is allowed to empty out. When the last call completes, the VPC is torn down and the resources recovered.

On the other hand, when the bandwidth is inadequate to support the required CBP and $\beta_{meas} > (1+\rho_u)\times\beta$ or $c_{occ} > max \times c_{actual}$, the connection server attempts to increase the bandwidth assigned to the VPC. This is also done using the two-phase approach described in Section 3.3. In the first phase, the connection server contacts all the switches in parallel requesting an increase in bandwidth (again, expressed as a percentage). In the second phase, it sends a commit request. If the attempt to increase bandwidth fails, the connection server attempts to set up a new VPC between this pair of switches on a new route.

How do we accommodate link and node additions and failures?

Lastly, we briefly describe the actions that happen in the network when its state changes in any of the following ways: addition of a switch, connection server or link, and failure of a switch, connection server or link.

Addition of a link: The switches on either side of the newly added link inform their respective primary connection servers about the new link. These connection servers in turn flood this information among other connection servers in the PG. This information also passes up the hierarchy if the link is between PGs. Thus the new link is added to the topology databases of all the necessary connection servers in the routing domain. These connection servers then start using the link for routing future connections.

Addition of a switch: As soon as a switch S fires up (it may be a new switch or a failed switch being restored after repair), it starts the *Hello* protocol over each of its outgoing links. Simultaneously it requests a list of connection servers in its PG from its neighbors. Once it receives this information S sends a `Register` message to a chosen connection server CS_c informing it of its status as a newly added node and requesting that it be assigned a primary connection server. CS_c chooses the least loaded connection server*, in its estimation, CS_p in the PG. It then forwards the `Register` message to CS_p. CS_p then establishes a direct logical signaling link with S and informs S about it. Once this link is established, S starts sending periodic PTSPs on this link to CS_p. The other connection servers in the PG know about S and topology state information associated with S from the PTSPs flooded periodically by CS_p in the PG.

Addition of connection server: When a connection server, CS_n comes up, it contacts its neighbors (which may be switches or other connection servers) to determine the address of other connection servers in the PG. It then proceeds to contact them and establish logical links to some or all of them and performs a topology database copy. It now starts receiving PTSPs from other connection servers and proceeds to do the same. Other connection servers in the PGs also use a "handshake" protocol to off-load some of the switches they are currently responsible for to the newly added connection server which is lightly loaded.

Failure of a link: Failures, both *intermittent* and *permanent*, may be detected by Operation and Maintenance (OAM) functions executed at different levels of the network hierarchy, namely, the physical layer OAM, ATM VP and VC layer OAM (ITU-T, 1993). *Hello* packets running at the PNNI routing protocol layer also aid in detecting failures. For example, when a link goes down, the nodes on either side of the link stop receiving *Hello* "heartbeat" packets from the node across the link and pass this information up to their primary connection servers. Correlating the alarms and failure indication messages received from different levels of the hierarchy and determining their cause is a complex problem. The connection servers could execute one of the algorithms from the literature (see (Katzela and Schwartz, 1995) for one such algorithm) to determine the failed network element. For example, using the information about missing *Hello* packets, the connection servers could determine which link has failed. This information is then flooded inside the network and eventually reaches the the topology databases of necessary connection servers which then do not use the failed element in computing future routes. Also, when a connection server CS determines that a link l has failed, it reroutes any

*Real-time loading information is assumed to be exchanged between the CSs in a PG in the PTSPs.

VPCs that it is monitoring, which use this link. The reuse of signaling software for VPC setup/modification and PCC for fast connection setup are the two aspects of DIVA that enable fast restoration of VPCs.

Failure of a switch: The failure of a switch is equivalent to the failure of all the links around it. The node is marked unreachable by the connection servers and is not used for routing connections.

Failure of a connection server: When a connection server CS_1 fails, each switch SW for which it was the primary connection server goes through the procedure followed by a switch that is newly added to the network (as explained above). Switch SW chooses another connection server CS_2 as its primary connection server, and starts sending their PTSPs to CS_2.

4 SUMMARY

Virtual Path Connections (VPCs) provide numerous advantages in ATM networks. They however suffer from the disadvantage of poor resource allocation due to preallocation of bandwidth and lack of sharing. In this paper, we addressed the issue of making VPCs resource efficient using dynamic VPC bandwidth management. We proposed **DIVA**, a distributed and dynamic algorithm for VPC monitoring and maintenance for the ATM Forum's Private Network-Network Interface (PNNI) standards based hierarchical networks. We described the approach used by DIVA to answer the following questions about VPC management.

- Where should VPCs be laid and how are the parameters to set up VPCs determined?
- How should VPCs be realized, *i.e.*, (a) how should VPCs be routed, and (b) how much bandwidth and buffer resources should be allocated to each VPC?
- When and how should adjustments be made to VPC routes/allocations?
- How are changes to VPC configurations accomplished in reaction to link and node (a) failures, and (b) additions and restorals?

5 REFERENCES

Ahmed, M. and Tesink, K. (1994). *RFC-1695: Definition of managed objects for ATM management using SMIv2.*

ATM Forum (1996). Private Network-Network Specification Interface v1.0.

Cheng, K.-T. and Lin, F. Y.-S. (1994). On the joint virtual path assignment and virtual circuit routing problem in ATM networks. In *Proc. IEEE Globecom*, pages 777–782.

Choudhury, G. L., Leung, K. K., and Whitt, W. (1995). An inversion algorithm for computing blocking probabilities in loss networks with state-dependent rates. In *Proc. IEEE Infocom*, pages 513–521.

Choudhury, G. L., Lucantoni, D. M., and Whitt, W. (1996). Squeezing the most out of ATM. *IEEE Trans. Comm.*, 44(2):203–216.

Faragó, A., Blaabjerg, S., Ast, L., Gordos, G., and Henk, T. (1995). A new degree of freedom in ATM network dimensioning: Optimizing the logical configuration. *IEEE J. Selected Areas Comm.*, 13(7):1199–1206.

Guérin, R., Ahmadi, H., and Naghshineh, M. (1991). Equivalent capacity and its application to bandwidth allocation in high-speed networks. *IEEE J. Selected Areas Comm.*, 9(7):968–981.

ITU-T (1993). *B-ISDN Operation and Maintenance Principles and Functions.* Rev. 1, Geneva.

Jagerman, D. L. (1974). Some properties of the Erlang loss function. *Bell System Technical J.*, 53(3):525–551.

Katzela, I. and Schwartz, M. (1995). Fault identification schemes in communication networks. *IEEE Trans. Networking.*

Kaufman, J. S. (1981). Blocking in a shared resource environment. *IEEE Trans. Comm.*, 29(10):1474–1481.

Labourdette, J.-F. and Hart, G. W. (1992). Blocking probabilities in multi-traffic loss systems: Insensitivities, Asymptotic behavior, and Approximations. *IEEE Trans. Comm.*, 40(8):1355–1366.

Logothetis, M. and Shioda, S. (1995). Medium-term centralized virtual-path bandwidth control based on traffic measurements. *IEEE Trans. Comm.*, 43(10):2630–2640.

Nagarajan, R. (1993). *Quality-of-Service Issues in High-Speed Networks.* PhD thesis, Department of ECE, University of Massachusetts, Amherst.

Shioda, S. and Uose, H. (1991). Virtual path bandwidth control method for ATM networks: successive modification method. *IEICE Trans.*, E74(12):4061–4068.

Shroff, N. and Schwartz, M. (1996). Improved loss calculations at an ATM multiplexer. In *Proc. IEEE Infocom*, pages 561–568.

Siebenhaar, R. (1994). Optimized ATM virtual path bandwidth management under fairness constraints. In *Proc. IEEE Globecom*, pages 321–329.

Srinivasan, S. and Veeraraghavan, M. (1996). Virtual paths in hierarchical wireless ATM networks. In *Proc. 3rd Int. Workshop Mobile Multimedia Comm.*

Stallings, W. (1993). *SNMP, SNMPv2, and CMIP: The practical guide to network-management standards.* Addison-Wesley Publishing Company.

The ATM Forum (1994). *ATM User-Network Interface Specification v3.1.*

Veeraraghavan, M., Kshirsagar, M., and Choudhury, G. L. (1996). Concurrent ATM connection setup reducing need for VP provisioning. In *Proc. IEEE Infocom*, pages 303–311.

6 BIOGRAPHY

SANTHANAM SRINIVASAN received his B.Tech. degree in Electronics & Communication Engineering from the Indian Institute of Technology (Madras) in 1991 and M.A. and Ph.D. degrees in Electrical Engineering from Princeton University in 1993 and 1995, respectively. He is currently at Bell Laboratories in the Networking Research Laboratory, where he is engaged in research in network management for IP and ATM networks.

MALATHI VEERARAGHAVAN is currently a Distinguished Member of Technical Staff at Bell Laboratories in the Networking Research Laboratory. Her research interests include mobility management, signaling and control of networks, and network management. Dr. Veeraraghavan received her B.Tech. degree in Electrical Engineering from the Indian Institute of Technology (Madras) in 1984, and M.S. and Ph.D. degrees in Electrical Engineering from Duke University in 1985 and 1988, respectively. She served as an Associate Editor of the IEEE Transactions on Reliability from 1992-1994.