Concurrent ATM Connection Setup Reducing Need for VP Provisioning

Malathi Veeraraghavan, Madhu M. Kshirsagar, Gagan L. Choudhury
AT&T Bell Laboratories
101 Crawfords Corner Road, Room 4F-509
Holmdel, NJ 07733
Email: mv@research.att.com

Abstract
A common approach used for decreasing end-to-end connection setup delay in ATM networks is to provision partial segments a priori using virtual path connections (VPCs). In this paper, we present an analysis to study the effect of such provisioning. The analysis demonstrates that significant savings in bandwidth, and hence network costs, are achieved if the number of provisioned VPCs are minimized, especially if there is uncertainty in the traffic pattern characterization. For a given example, we show that to achieve a 0.1% blocking probability requirement, if link capacities are shared (i.e., not partitioned a priori to provisioned VPCs), resource savings of 14.4%, 47.9% or 80% are possible under exact, approximate and poor traffic characterizations, respectively. If the above bandwidth savings are to be achieved in networks that use traditional node-by-node connection setup approaches, end-to-end connection setup delay will increase. In this paper, we propose a new connection setup algorithm that allows the savings in bandwidth obtained through sharing while at the same time controlling the connection setup delay.

1 Introduction
ATM is a connection-oriented technology designed to support multimedia communication. Signaling procedures are needed to establish and release connections. The ATM signaling standards solutions [1] and [2] are based on a fundamental concept in virtual circuit networking, i.e., connections are established using a hop-by-hop circuit threading process prior to user data exchange. The per-node circuit threading (fabric configuration) process executed at each transit node adds to the end-to-end connection setup delay. This delay can be significant especially in wide-area networks. A solution offered to this problem is to provision (preestablish) partial segments of the connection, thus decreasing the number of transit nodes at which the per-circuit configuration or threading process is needed during on-demand connection setup. This scheme, used extensively in circuit-switched wide-area telephony networks, has been adopted in ATM networks through the use of virtual paths. However, the use of provisioned virtual path connections reduces sharing, the benefits of statistical multiplexing and hence network utilization. We present an analysis to quantify the effect of reduced sharing caused by using preestablished virtual path connections.

Having observed and quantified this effect, we propose an alternative means to reduce end-to-end connection setup delay using fewer preestablished virtual path connections than would be needed if the sequential hop-by-hop threading scheme is used. In this new approach, actions are executed in parallel at sets of switches on the route of the connection. Hence, our scheme is named the Parallel Connection Control (PCC) algorithm. In this paper, we describe the PCC algorithm for two-party connection setup.

Other research work [3][4] on the control of ATM connections only address new features such as the control of connections routed through special resources, while accepting the signaling standards solution for two-party connection control. References [5][6][7] suggest alternative mechanisms for two-party connection control, but since the standards solution has the most influence, we position our proposal relative to this scheme.

In Section 2, we provide background material describing the standards approach. In Section 3, we present an analysis to show the effects of preestablishing virtual path connections on network costs. This analysis motivates the need for alternative means to reduce end-to-end connection setup delay without relying solely on preestablished VPCs. Section 4 describes the PCC algorithm to set up two-party bidirectional ATM connections. We provide our conclusions in Section 5.

2 Background
There are two aspects to the problem of controlling switched ATM on-demand connections: routing and signaling. We describe their inter-relationship and the standards sequential approach for two-party connection setup.
2.1 Routing and signaling

Routing procedures in the management plane [8] constitute the periodic update of either routing data or topology and status data in switches. Signaling is a control plane procedure which handles on-demand connection setup and release requests. During connection setup, each switch consults the data updated by the routing procedure to determine the next switch or set of switches in the connection.

The optimal table-based routing scheme [9] uses a network manager to compute routes based on current loading conditions and predicted call arrival/departure patterns. This typically involves periodically solving a network-wide optimization problem for some objective function, such as call blocking probability, subject to a set of constraints. The computed routes are then downloaded to the switches. When a signaling request for a connection setup arrives at a switch, it simply consults its routing table (that was preloaded by the routing procedure) to determine the next switch in the route. This routing scheme is assumed by the ITU-T signaling standard for public ATM networks, B-ISUP (Broadband ISDN User Part) [2].

A second approach for routing is a shortest-path table-based routing scheme [9] which is used in the PNNI (Private Network Node/Network Interface) routing protocol defined by the ATM Forum [10]. Periodic topology state packets are exchanged between ATM switches. When a signaling request for connection setup arrives at a switch, it uses the network topology and state information (updated by the PNNI routing protocol) to select a set of switches through which to route the connection using an on-the-fly path determination algorithm, such as Dijkstra’s shortest path algorithm. Hierarchical ordering of ATM switches (peer groups) is used to scale this procedure for large networks. This routing scheme is assumed by the PNNI signaling standard [1] for private ATM networks.

2.2 Switch-by-switch sequential setup approach

We now describe the standards signaling procedure for two-party connection setup. Upon receiving a connection setup request, the first step is to compute the route for the connection. As described in Section 2.1, in the public network approach, route computation is simply a matter of consulting a prestored data table which indicates the next switch toward which the connection must be routed in order to reach the destination. In the private network approach, a shortest-path route is computed by the switch receiving the setup request. It forms a Designated Transit List (DTL) of the set of switches on the route of the connection. This “source routing” approach frees resources in subsequent switches on the route from performing any route computations. As shown in Fig. 1, upon receiving a SETUP request, using either route computation approach, switch $SW_1$ determines the route as passing through $SW_2$. $SW_1$ then performs the following actions:

1. Step 1: CAC (Connection Admission Control) function to determine if the new connection can be admitted without disrupting any QoS (Quality-of-Service) guarantees given to preexisting connections [11].

2. Step 2: Selection of VPCI/VCIs (Virtual Path Connection Identifier/Virtual Channel Identifier) or VPI/VCIs (Virtual Path-Identifier/Virtual Channel Identifier) to the next switch on one of the interconnecting ATM interfaces.

3. Step 3: Making translation table entries (configuring the fabric) to map VPI/VCI pairs from the incoming to outgoing ports in one or both directions. This is the key “threading” step in connection setup. During data exchange, ATM cells arriving on the connection are routed through the switch by matching the VPI and VCI fields in the cell header with an entry in this translation data table.

4. Step 4: Setting of control parameters for run-time algorithms, such as scheduling, rate and/or priority control, usage parameter control, fault handling, etc., in the switch.

Upon completion of these activities, a signaling message is sent to the next switch on the route, $SW_2$. At each switch on the route, the above listed 4 steps are performed. The last switch in the connection may be required to perform an additional function, end-to-end QoS computation, for measures, such as end-to-end delay.

In order to minimize the number of switches at which the PNNI and B-ISUP signaling procedures are performed sequentially, partial segments of connections are preestablished using virtual path connections (VPCs). As shown in Fig. 1, during on-demand connection setup, no signaling procedures are executed at the VP cross-connect. Switch $SW_1$ selects a VCI within the VPC shown terminating at switch $SW_2$ and sends the next-hop signaling message directly to switch $SW_2$. This technique of using VPCs reduces the number of nodes at which connection setup messages are processed, and hence, the end-to-end connection setup delay. However, it sacrifices network utilization, as shown by the analysis of the next section.

1. In the above description, we assumed that all 4 steps are performed as signaling proceeds in the forward direction. However, steps 3 and 4 may be performed when the response signaling messages traverse in the backwards direction.
3 Analysis

In this section, we analyze the effects of preestablishing virtual path connections (VPCs) on network resources, such as link bandwidth. We compare the link capacities required in the two network configurations shown in Fig. 2 needed to satisfy the same blocking probability criterion. The main difference between these two configurations is that link $l_3$ bandwidth is partitioned amongst $K$ preestablished VPCs in configuration 1, while, in configuration 2, link $l_3$ bandwidth is completely shared. The goal of this analysis is to study the effect of partitioning vs. sharing.

In configuration 1, $K$ virtual path connections are shown preestablished between the set of switches $SW_{11}, SW_{12}, \ldots, SW_{1K}$ and $SW_2$ passing through VP cross-connects. The term cross-connect is used while referring to a node that does not participate in the on-demand connection setup procedure (as shown in Fig. 1) due to the existence of a preestablished virtual path connection passing through it. The term switch is used to refer to nodes which participate in the on-demand connection setup procedure. In configuration 2, the VP cross-connect $CC_3$ is replaced by switch $SW_3$ since, in this configuration, signaling messages are processed by this node during on-demand connection setup, unlike in configuration 1. Thus, in configuration 2, the $K$ virtual path connections (VPCs) extend only between each switch in the set of switches, $SW_{11}, SW_{12}, \ldots, SW_{1K}$, and $SW_3$.

Let $p_i$ represent the product of call arrival rate and mean holding time for class $i$ calls, where class $i$ represents the calls routed on virtual path connection $i$, $1 \leq i \leq K$. The arrival process is assumed to be Poisson for each class. Let $b_i$ represent the bandwidth (an integral multiple of a basic bandwidth unit, such as 64Kbps or 1.544Mbps) required for a class $i$ call on each link of the route. Let $N_i$ represent the bandwidth (also, an integral number of the same basic bandwidth unit) on links $l_{1i}$ and $l_{2i}$ preassigned to VPC $i$ in both configurations (Fig. 2), and $M$ represent the total preassigned bandwidth on link $l_3$. Clearly, in configuration 1, $M = \sum_{j=1}^{K} N_j$. However, in configuration 2, $M$ is not directly related to the $N_j$'s, since this bandwidth is shared among the $K$ classes.

Since the model has a product-form structure, exact formulations for the blocking probabilities in the two configurations can be stated in terms of one-dimensional and $(K+1)$-dimensional normalization constants, $g_1(\cdot)$ and $g_2(\cdot)$, respectively. Let $p_i^{(1)}$ and $p_i^{(2)}$ represent the class $i$ blocking probabilities under configurations 1 and 2 respectively. Then,

$$p_i^{(1)} = 1 - \frac{g_1(N_i)}{g_1(N_j)}$$

$$p_i^{(2)} = 1 - \frac{g_2(N_j, \ldots, N_{i-1}, N_j - b_i, N_{i+1}, \ldots, N_{K}, M - b_i)}{g_2(N_j, \ldots, N_{i-1}, N_{i+1}, \ldots, N_{K}, M)}$$

These formulations with the definitions of $g_1$ and $g_2$ are derived in the Appendix. See [12] for further details.

Next, assuming the mean call holding times to be the same for all classes, we get the overall blocking probabilities for a randomly chosen call in the two configurations as:

$$p^{(1)} = \frac{\sum_{i=1}^{K} p_i p_i^{(1)}}{\sum_{i=1}^{K} p_i}$$

and

$$p^{(2)} = \frac{\sum_{i=1}^{K} p_i p_i^{(2)}}{\sum_{i=1}^{K} p_i}$$

We now consider three numerical examples to study the effect of using provisioned virtual path connections (VPCs) in ATM networks. In all of these examples, we compute the bandwidth assignment required in order to satisfy a given blocking probability criterion. In other words, $p^{(1)}$ and $p^{(2)}$ are assumed to be given, and the link capacities, $N_j$ and $M$, are computed using EQ1, EQ2, and EQ3. An input required in the VPC provisioning process is the expected traffic pattern characterization. We thus study the effect of provisioning under varying degrees of uncertainty in the expected traffic. The first example illustrates the bandwidth assignment assuming that we have an exact characterization of traffic patterns. In the second example, we determine bandwidth assignments assuming that only an approximate characterization of traffic patterns is available. Finally, the third example repeats the second example, except, that we assume the availability of a very poor characterization of the traffic pattern.

In all three examples, we illustrate the difference in bandwidth requirements for the two configurations needed to satisfy a given blocking probability criterion. We only concentrate on the bandwidth difference in the common
link, $I_3$, between VP cross-connect $CC_2$ and SW, in configuration 1, and between $SW_3$ and $SW_2$ in configuration 2. In the rest of the network (i.e., links $I_{1i}$ and $I_{2i}$), there are no significant bandwidth differences since we assume complete partitioning in both configurations to keep the model simple.

**Example 1:** We assume $K = 5$, the traffic load for each class to be known exactly, and that the traffic load is the same for all classes. Specifically, $p_j = 60$ for $i = 1, 2, ..., 5$. We also assume $b_j = 1$ for $i = 1, 2, ..., 5$. Therefore, the best static bandwidth assignment in configuration 1 would be to assume $N_1 = N_2 = ... = N_5 = N$ and $M = 5N$. We assume $N_1 = N_2 = ... = N_5 = N$ for configuration 2 also, but compute $M$ for this configuration. For several overall blocking requirements (the quantities $k(1)$ and $P(2)$ in EQ3), we first compute the minimum required $N$ for configuration 1. For this configuration, we can immediately determine $M$ since $M = 5N$. Next, assuming the same $N$, we compute the minimum required $M$ in configuration 2. Fig. 3 shows the minimum required bandwidth on the common link $I_3$ for the two configurations (plots marked Example 1).

We see that even though an exact characterization of the traffic pattern is available, there is some saving of link bandwidth resources in configuration 2. For example, at a blocking probability requirement of 0.001 (0.1%), the saving is about 14.4%.

**Example 2:** We assume that only an approximate traffic characterization is available. The sum of the traffic loads on the five VPCs is the same as in Example 1 but that the exact individual loads are not known and may be unbalanced. A load variation of 20% is assumed to be possible for each class. The worst case imbalance occurs when the traffic on four VPCs are 20% less compared to Example 1 (i.e., the corresponding $p_j$'s are equal to 48 units) and the fifth VPC carries this extra traffic (i.e., the corresponding $p_j$ is 108 units). We further assume that it is not known a priori which particular path carries the extra traffic (may be different paths carried at different times) and hence we still have to use $N_1 = N_2 = ... = N_5 = N$ in both configurations. Link $I_3$ bandwidth requirement $M = 5N$ in configuration 1, but is computed for configuration 2 as before. Fig. 3 shows the minimum required bandwidth on the common link $I_3$ for the two configurations (plots marked Example 2). The bandwidth savings in configuration 2 relative to configuration 1 are substantially higher when there is some uncertainty in the expected traffic characterization. For instance, at a blocking probability requirement of 0.001 (0.1%), the saving is about 47.9%. It is also to be noted that as we go from Example 1 to 2, the bandwidth requirement in configuration 1 goes up substantially. By contrast, in configuration 2, the bandwidth requirement goes up only slightly at lax blocking probability requirements and remains almost the same (in fact, even goes down a bit) at stringent blocking probability requirements. Thus, configuration 2 is more robust with respect to variation in traffic patterns. This is an important advantage since in broadband networks (particularly in the initial deployment phase), the traffic patterns are likely to be highly variable and unpredictable.

**Example 3:** This is a more extreme form of Example 2 in which we start with a very poor traffic characterization. We assume that blocking requirements should be satisfied under maximal possible traffic imbalance, i.e., if all the traffic goes over just one VPC. As before, we assume that it is not known which particular VPC experiences this worst case traffic and so the same bandwidth needs to be assigned on each VPC. Fig. 4 shows the minimum required bandwidth on the common link for the two configurations. The effects observed in Example 2 are further enhanced here. Configuration 2 requires 80% lower bandwidth on the common link $I_3$ relative to configuration 1 in order to meet a 0.1% blocking probability criterion.

Our bandwidth calculation is appropriate for constant-bit-rate (CBR) traffic but conservative for variable-bit-rate (VBR) traffic since we do not take into account possible statistical multiplexing gains in that case (we assume bandwidths of multiple connections to be additive). However, if we could take into account the statistical multiplexing effect for VBR traffic [13], the advantage of configuration 2 over configuration 1 would be even higher since the statistical multiplexing gain would be bigger in the shared system compared to that in the partitioned system.

In the above analysis, we demonstrated that, while

2. The total bandwidth requirement on links $I_{1i}$, $I_{2i}$, and $I_{3i}$ is more for example 2 compared to example 1 even for configuration 2. To meet a stringent blocking probability criterion, the bandwidth requirement on links $I_{1i}$, $I_{2i}$ increase significantly as in configuration 1 since these resources are not shared.
planning end-to-end routes for multiple streams of the same service class, by using more common links that are shared than a completely partitioned set of routes, fewer network resources are needed to achieve a specified call blocking probability. Alternatively, one can achieve lower call blocking probability for a given set of resources in networks that use fewer provisioned virtual path connections.

4 The Parallel Connection Control (PCC) algorithm

In the above analysis, we demonstrated the advantages of sharing link resources. Given these results, a network designer should minimize the usage of preestablished virtual path connections (VPCs) that reduce sharing. However, if the usage of preestablished VPCs in minimized, and the switch-by-switch sequential approach described in Section 2.2 is used for on-demand connection setup, the end-to-end connection setup delay will be high. This is because switch configurations have to be made at each transit node that is not preconfigured with a virtual path connection. For example, in Fig. 2, in configuration 1, while setting up a connection, per-switch setup functions are needed only at one of the switches, SW_{11}, SW_{12}, ..., SW_{1K}, and at switch SW_{2}, while in configuration 2, these functions need to be performed at switch SW_{3} also.

In this section, we describe an alternative approach to connection setup. Our approach exploits parallelism to achieve reductions in end-to-end connection setup delay thus minimizing the number of preestablished virtual path connections needed. Instead of performing functions sequentially at each switch, groups of per-switch functions are executed concurrently at multiple switches. We are not recommending a solution in which provisioned virtual path connections are completely abandoned. Instead by employing a mixture of virtual path connections and VC routing, one can design a network in which utilization is not sacrificed in favor of end-to-end connection setup delay.

Before presenting the PCC algorithm, we describe the functional network architecture assumed.

4.1 Network architecture

Fig. 5 shows that the functionality needed to setup/modify/release a connection is split between two types of servers, connection servers and switch resource servers. Connection servers perform route determination, and end-to-end or segment quality of service computation. A switch resource server manages switch resources, such as VPI/VCI's on all the ATM interfaces to the switch with which it is associated, link bandwidth, buffer space, etc., and performs fabric control functions to make and clear VPI/VCI translation table entries in the switch. Switch resource servers also set parameters for the various run-time algorithms, such as priority control, usage parameter control, rate control, scheduling, fault handling, etc., based on the QoS parameters specified for the connection. We assume one switch resource server per ATM switch.

Fig. 5 shows that the network is subdivided into "domains". Each domain consists of a set of connection servers, a set of switches (with switch resource servers) and a set of end hosts. Each connection server in a domain is logically connected to all the switches in the domain and to some subset of end hosts which are physically connected to this set of switches. For example, domain 1 shows p con-

---

3. A segment is defined as a concatenation of channels on a connection.
nection servers, $CS_{11}, ..., CS_{1p}$, $k$ switches, $SW_{11}, ..., SW_{1k}$ with associated switch resource servers $SR_S_{11}, ..., SR_S_{1k}$, and a group of end hosts. Domains may be heterogeneous, in that different domains may have different numbers of switches, connection servers, and end hosts. Multiple connection servers within a domain serve to improve scalability and reliability. Further, the usage of multiple domains allows for the design of large-scale networks based on this architecture.

In Fig. 5, the signaling links shown between the end hosts and the connection servers, between connection servers and switch resource servers, and between connection servers from one domain to another, are assumed to be ATM logical links used only to carry signaling messages. The signaling channel on the user-network interface from each end host terminates on one of the connection servers in the domain of the end host. In general, an end host may have direct signaling links to multiple connection servers as required from a performance and reliability standpoint. For signaling communication between any two nodes that do not have a direct ATM logical link, datagram communication through routers, such as IP routers, can be used.

We now present the PCC algorithm for establishing two-party bidirectional connections. We describe the more general procedure for connection setup spanning multiple domains, such as from end host $A$ to $C$ in Fig. 5. The intradomain connection setup procedure is a subset of this general procedure.

### 4.2 Connection setup procedure

We first provide an overview of the procedure and discuss its implications. Details are then provided.

**Overview of the procedure.** Fig. 6 shows the message exchange needed to set up a connection between two end hosts that involves multiple connection servers, for example, from end host $A$ to end host $C$ in the network of Fig. 5. The first connection server $CS_{11}$ that receives the `Setup-connection` message determines the route of the connection as passing through switches in its domain and through domain 2. It establishes the first segment of the connection through switches in its domain using a `two-phase` message exchange as shown in Fig. 6. In the `first phase`, it requests the switch resource server of the switches on the route within its domain to reserve resources for the connection. This phase is executed `in parallel` at all the switches within this domain through which the connection is routed. Each switch performs connection admission control (CAC) procedures and selects VPI/VCI as described in steps 1 and 2 of Section 2.2.

4. Standard UNI (User-Network Interface) [14] signaling messages can be used for the end host-connection server communication.
end hosts, then 15 of these should be designated ATM VP cross-connects. Resources need to be preassigned for virtual path connections passing through these 15 VP cross-connects, which greatly reduces sharing.

Using the PCC approach, we assume that a connection server can set up a segment passing through 5 switches in 150ms. This time is only slightly larger than the per-switch processing time of 100ms assumed for the switch-by-switch sequential approach because the per-switch functions are performed in parallel at the 5 switches in the segment. We added an extra 50ms for the additional processing in the connection servers and signaling message communication times. Using this 150ms assumption for the PCC approach, the connection can be routed through a maximum of 3 domains to meet the end-to-end connection setup delay requirement of 500ms. Assuming each domain has the same number of switches (five), the total number of switches at which per-node configuration occurs during on-demand connection setup is 15, which implies that only 5 VP cross-connects are needed. Thus, resources need to be preassigned for virtual path connections passing through only these 5 VP cross-connects instead of the 15 cross-connects that we needed using the switch-by-switch sequential approach. This leads to increased sharing of resources in the PCC approach and hence better network utilization or lower network costs.

Details of the procedure. We now provide details of this procedure. The flow in Fig. 6 assumes that the end-to-end connection between end hosts A and C is routed through switches $SW_{11}, ..., SW_{1k}$ in domain 1 and then through $SW_{21}, ..., SW_{2m}$ in domain 2. Since $SW_{1k}$ is assumed to be physically connected to $SW_{21}$, only two domains and the corresponding connection servers $CS_{11}$ and $CS_{21}$ are involved in this connection setup procedure. In general, the connection may pass through several domains, and hence several connection servers may be involved.

When connection server $CS_{11}$ receives a Setup-connections message from end host $A$ as shown in Fig. 6, it forms the route determination function. The PCC setup scheme can coexist with both the optimal and shortest-path table-based routing schemes described in Section 2.1. In the optimal approach, the routing table at the connection server $CS_{11}$ maps the combination of the source and destination addresses to a sequence of switches within its domain and the identity of a connection server in the next domain through which the connection must be routed by consulting prestored data tables. The shortest-path PNNI routing protocol scheme is especially synergistic with the PCC connection setup scheme. A PCC domain is comparable to a PNNI peer group. $CS_{11}$ computes the DTL (Section 2.2) in the same manner as does the first switch in the PNNI signaling model. The DTL comprises the detailed set of switches within domain 1 and the identifiers of the connection servers in other domains through which the end-to-end route will pass.

The detail with respect to the next phase of operation, the Reserve-resources phase, is with regards to the selection of the VP/VCIs. This is illustrated in Fig. 7.

We assume that end host $A$ selects VP/VCIs on its user-network interface and specifies these in the Setup-connections message. In the Reserve-resources message, $CS_{11}$ requests:

- the first switch resource server in domain 1, $SRS_{11}$, to select VP/VCIs in the direction from its switch $SW_{11}$ to the next switch $SW_{12}$;
- intermediate switch resource servers, $SRS_{12}, ..., SRS_{(k-1)}$, to select outgoing VP/VCIs in both directions from their associated switches; and
- the last switch resource server in domain 1, $SRS_{1k}$, to select a VP/FCI from $SW_{1k}$ to $SW_{2(k-1)}$ and VP/VCIs in both directions on the ATM interface between $SW_{1k}$ and the switch in the next domain, $SW_{21}$.

Upon completion of the CAC and VP/FCI selection functions in the reserve-resources phase, each switch resource server asynchronously sends its response to $CS_{11}$ as shown in Fig. 6. Parameters of this response include switch-level QoS guarantees, such as cross-switch delay, and the requested VP/VCIs. The connection server computes any segment-level measures that contribute toward end-to-end measures, such as end-to-end delay, using the returned switch-level QoS parameters. If user QoS specifications for the connection are not violated, it forms the Configure-switch messages and sends them in parallel to the switch resource servers as shown in Fig. 6. The detail here is in the parameters of these messages. The connection server specifies the input and output VP/VCIs (with port numbers) needed to configure the switch fabric in each Configure-switch message. This VP/FCI data is obtained from the Reserve-resources response that it received in the previous phase. For example, the VP/FCI from $SW_{1(k-1)}$ to $SW_{1k}$ (see Fig. 7) returned in the Reserve-resources response message from $SRS_{1(k-1)}$ is the input channel, and the VP/FCI from $SW_{1k}$ to $SW_{21}$, which was
returned in the Reserve-resources response from $SRS_{1k}$, is the output channel specified in the Configure-switch message to $SRS_{1k}$ for one direction of the fabric configuration (translation table entry).

The next detail is with regards to the Setup-segment message shown in Fig. 6. The VPI/VCI IDs selected for the two directions on the inter-domain ATM link between switches $SW_{1k}$ and $SW_{21}$ (Fig. 7) are passed as parameters in this message. Upon receiving the Setup-segment message, $CS_{21}$ determines the route for its segment of the connection. If the PNNI routing scheme is used, this action is equivalent to the detailed DTL computation in a new peer group. If the optimal table-based routing scheme is used, this action consists of consulting a prestored data table. In this example, the segment route consists of switches $SW_{21},...,SW_{2m}$. Upon completion of the two phases of segment setup (each executed in parallel), $CS_{21}$ offers the connection to the end host $C$ with the VPI/VCIs for the two directions on its user-network interface.

If the end host $C$ accepts the connection, it notifies its application to enable reception of data on the incoming channel. Upon receiving the Offer-connections response, $CS_{21}$ can send the Start-traffic message to end host $C$, as shown in Fig. 6, since the entire connection from end host $A$ is completely configured, including end host application configuration for data reception. $CS_{21}$ also sends a Setup-segment response to connection server $CS_{11}$ which can then send a Start-traffic message to end host $A$ allowing it to start data transfer. Thus, data transfer from end host $C$ to $A$ may begin before data transfer from end host $A$ to $C$. However, there is no race condition, in that data transfer will not precede switch fabric configurations. We ensure this by allowing segment setup to proceed to the next domain only after the segment is completely established in the current domain.

As mentioned above, an important part of connection setup is application configuration at the end hosts. In the above description of the PCC scheme, we assumed that the Setup-connections message carries VPI/VCI assignments for the two directions of the interface from end host $A$ to switch $SW_{11}$ (Fig. 7). In this case, the application at end host $A$ is configured to receive data on the incoming ATM channel. If the Setup-connections message does not specify the VPI/VCIs from/to the calling end host $A$, $SRS_{11}$ needs to select these channels as part of the reserve-resources phase. In this case, the identifiers of these channels are communicated to end host $A$ during the configure-switch phase of the first segment setup so that upon completion of the latter procedure, the first segment of the connection (through domain 1) is fully established including the calling end host $A$ application configuration. Thus, when the last connection server ($CS_{21}$) receives the Setup-segment message, it is assured that previous segments are fully established, permitting it to send the Start-traffic message to its end host $C$.

Only the “sunny-day” cases have been presented here. However, we have worked out the exception situations in our detailed design. Connection release follows a similar procedure to the above-described setup procedure.

5 Conclusions

Two main contributions were made in this paper. First, we carried out an analysis to study the effect of VP provisioning. A priori provisioning, typically done to reduce the number of transit nodes at which on-demand connection setup requests are handled, requires a predicted characterization of traffic patterns. The analysis demonstrates that significant savings in bandwidth, and hence network costs, are achieved if the number of provisioned virtual path connections are minimized, especially if there is uncertainty in the traffic pattern characterization. For a given example, we show that to achieve a 0.1% blocking probability requirement, if link capacities are shared (i.e., not partitioned a priori to provisioned VPCs), resource savings of 14.4%, 47.9% or 80% are possible under exact, approximate and poor traffic characterizations, respectively.

If the above bandwidth savings are to be achieved in networks that use traditional node-by-node connection setup approaches, end-to-end connection setup delay will increase. Our second contribution is a new ATM connection setup algorithm. In this scheme, by using Parallel Connection Control (PCC) within segments of the connection, end-to-end connection setup delay is reduced without requiring a large number of preestablished virtual path connections. Thus, lower cost networks can be implemented using our PCC approach. Our signaling procedure for on-demand connection setup works in conjunction with both the optimal and shortest path table-based routing schemes. This solution is scalable and provides for robust implementations. We have implemented this algorithm for connection setup (and release) to demonstrate its feasibility in a prototype test bed consisting of three types of ATM switches and two types of multimedia end hosts.

6 Acknowledgment

The authors sincerely thank Richard Buskens and Kai Y. Eng of AT&T Bell Labs. for their valuable input to this work.
Appendix

We derive the normalization constants, \( g_1(\bullet) \) and \( g_2(\bullet) \) in EQ1 and EQ2. The normalization constants are obtained as

\[
g_1(N) = \sum_{n = 0}^{N} \frac{N!}{k!} \frac{b_j^n}{n!} \tag{A.1}
\]

where \( \lfloor x \rfloor \) indicates the largest integer less than or equal to \( x \), and,

\[
g_2(N) = \sum_{n \in S} \left( \prod_{j=1}^{K} \frac{\rho_j^n}{n_j!} \right) \tag{A.2}
\]

where \( N = (N_1, ..., N_K, N_{K+1}) \), \( n = (n_1, ..., n_K) \), and the state space \( S \) is given by:

\[
S = \left\{ n \in \mathbb{Z}^K : n_j b_j \leq N_j \text{ for } j = 1, 2, ..., K \right\} \tag{A.3}
\]

and \( \sum_{j=1}^{K} n_j b_j \leq N_{K+1} \)

Normalization constant \( g_1(N) \) is easy to compute and indeed as we plug A.1 into EQ1, we obtain the classical Erlang-B blocking probability formula with offered traffic \( \rho_j \) and \( N/b_j \) servers. Computation of \( g_2(N) \) is more involved but still may be computed by forming a \((K+1)\)-dimensional generating function as in [12]. Define

\[
G_2(z) = \sum_{N=0}^{\infty} \prod_{i=1}^{K+1} \frac{N_i!}{z_i^{N_i}} \tag{A.4}
\]

where \( z = (z_1, z_2, ..., z_{K+1}) \). It can be shown that \( G_2(z) \) has the compact expression

\[
G_2(z) = \frac{e^{-\sum_{j=1}^{K+1} \rho_j b_j}}{\prod_{i=1}^{K+1} (1 - z_i)} \tag{A.5}
\]

In general, \( g_2(N) \) may be obtained from \( G_2(z) \) via \((K+1)\)-dimensional inversion which has a computational complexity growing exponentially with \((K+1)\). Fortunately, however, due to the special structure of the underlying network, \( K \) of the dimensions may be inverted explicitly thereby greatly reducing the computational complexity. Specifically, let

\[
G_2(N_1, N_2, ..., N_K, z_{K+1}) = \sum_{N_{K+1}=0}^{\infty} g_2(N) \frac{N_{K+1}!}{z_{K+1}^{N_{K+1}}} \tag{A.6}
\]

Then, \( G_2(\bullet) \) may be expressed as:

\[
G_2(N_1, N_2, ..., N_K, z_{K+1}) = \prod_{j=1}^{K} \left( \frac{\rho_j b_j}{n_j!} \right) \sum_{n_j=0}^{N_j} \left( \prod_{j=1}^{K} \left( \frac{\rho_j b_j}{n_j!} \right) \right) \tag{A.7}
\]

Normalization constant \( g_2(N) \) is obtained exactly by performing a fast one-dimensional inversion on the expression in A.7 as in [12].

References


3b.1.9