In Proc. Infocom '95, 14th Annual Joint Conference of the IEEE Computer and Communications Societies, Boston, Mass. (April 1995), pp. 1222-1229.

A Model for Virtual Tree Bandwidth Allocation in ATM Networks

Adarshpal S. Sethi
Department of Computer and Information Sciences
University of Delaware
Newark, DE 19716
sethi@cis.udel.edu

Abstract

The technique of Virtual Paths is used in ATM networks to perform bandwidth allocation for virtual circuits and to simplify setting up of virtual circuits in response to connection requests. In this paper, we describe a new technique called Virtual Trees that can be used for bandwidth allocation in ATM networks. A Virtual Tree corresponds to pre-allocated bandwidth along a set of links in the network that form a tree rooted at a source node and leading to various destinations. The use of Virtual Trees provides more flexibility to a source node in setting up a virtual circuit and results in smaller rejection probabilities for connection requests. Virtual Trees retain all the advantages of Virtual Paths but have a better performance potential. We present an optimization model for Bandwidth Allocation for Virtual Tree configurations and present the results of a simulation study comparing the performance of Virtual Trees with that of Virtual Paths.

1 Introduction

The Asynchronous Transfer Mode (ATM) is a switching and multiplexing technique developed for Broadband ISDN (B-ISDN) networks that makes it possible to achieve the multiplexing of different kinds of traffic while providing individual quality of service guarantees for each traffic type [3, 12, 19]. The CCITT standard for the ATM Layer defines two types of connections: *Virtual Channel (VC) connections* and *Virtual Path (VP) connections* [3, 19]. The concept of a Virtual Channel (VC) is similar to that of a virtual circuit in traditional networks, namely, a logical unidirectional association that defines a connection between a source and its destination. To establish a VC, the source node sends a request that propagates through intermediate nodes to the destination requesting allocation of the required bandwidth. Because of the propagation delay (which assumes increasing importance as speeds increase), and the processing delay at each node of the network, VC establishment by this method can be slow and unacceptable for real-time needs, particularly when connections are established on a per-burst basis.

In ATM networks, Virtual Paths (VPs) solve this problem by providing a pre-defined route with pre-defined bandwidth between a source-destination pair. A Virtual Path is commonly defined as a bundle of virtual channels delimited by two Virtual Path terminators. Multiple VCs may exist simultaneously within a Virtual Path between a given source-destination pair. Virtual Channels are allowed to share the bandwidth pre-allocated to the Virtual Path they belong to, yet the sharing of the Virtual Path's bandwidth depends on the bandwidth allocation technique being used. In some cases, a Virtual Channel may need to use more than one Virtual Path to associate a source with its destination (i.e., to establish a connection); the VC may then be carried by a concatenation of Virtual Paths through a sequence of intermediate VP terminators.

The Virtual Path concept provides several advantages such as allowing simpler network architectures, eliminating the need for call-by-call routing and allowing easier implementation of dynamic bandwidth allocation schemes [4, 15, 16, 17]. Virtual Paths are conceptualized at the call layer as a direct link with fixed bandwidth between a source-destination pair. The idea of having a pre-defined path with pre-allocated bandwidth reduces the amount of routing information carried by each cell and eliminates the need for call-by-call routing which results in very short connection setup times.

The Virtual Paths used in a network must be peri-

odically modified to match the dynamically changing traffic patterns. A Virtual Path configuration refers to the set of Virtual Paths that are to be used during a period of time. There are several proposed algorithms that deal with the problem of finding such configurations [8, 9, 11, 17, 20] given predicted traffic patterns between source-destination pairs.

Virtual Channels are allowed to share the bandwidth pre-allocated to the VP they belong to. But the technique of Virtual Paths does not involve the policing of the source's traffic to enforce the amounts of bandwidth pre-allocated to each VP; therefore, each source must keep its traffic within the parameters negotiated at the call admission to ensure that the quality of service offered by the network will meet the requirements of all the sources using the network. Once a VP has been set up with a specified bandwidth, the source node must decide on each call request whether or not that call can be carried by the VP. In doing so, it is useful to refer to the term *equivalent bandwidth* which is the amount of bandwidth necessary to accommodate the aggregate traffic of a set of N bursty sources while complying with a specific quality of service [6, 7, 8]. The quality of service may be measured in terms of the buffer overflow probability or the cell loss probability. Other desirable quality of service measures are delay and delay jitter, but these are harder to use in computing equivalent bandwidth.

In this paper, we describe a new technique, proposed previously by us ([18]), called Virtual Trees (VTs) to be used in a manner similar to the way Virtual Paths are currently used. In VTs, there is predefined bandwidth available for use by a given source node, just as in VPs. Whereas in VPs, the path and the corresponding bandwidth is defined separately for each destination, in VTs some of the links and their bandwidths could be shared by VCs going to different destinations. This takes the form of a tree rooted at the source node whence the name Virtual Tree.

In a Virtual Path, different Virtual Channels may share the bandwidth of the VP resulting in better multiplexing of the traffic than if the network allocated bandwidth individually for each VC [14]. Because of this multiplexing effect, a smaller amount of bandwidth may be allocated to the VP than the sum of the bandwidths of all VCs traversing it. Put another way, the Equivalent Bandwidth required by each VC decreases as more traffic is multiplexed on the same VP. However, multiplexing in VPs is limited to traffic flowing between the same source-destination pair.

In Virtual Trees, the multiplexing effect of VPs is enhanced by permitting sharing of bandwidth among traffic flowing to different destinations. It is still limited to traffic originating at a single source, as that is what allows the source node to decide on call admission without sending out a call setup request to the destination. But a Virtual Tree allows the bandwidth allocated to a particular source node to be used flexibly for various destinations. The effect is a reduced probability of blocking (rejecting connection requests), better utilization of allocated capacity, and flexibility in determining routes when the VT is set up. These effects are obtained by preserving all the major advantages of VPs with only the slight additional cost of a somewhat more complex call admission algorithm at the source node.

In the following sections, we first describe the concept of a Virtual Tree and contrast it with a Virtual Path. We also briefly describe how call admission with Virtual Trees may be handled by a source node. The main contribution of this paper is a model for Bandwidth Allocation for Virtual Tree configurations which is presented in Section 3. This model allows us to allocate bandwidth to various links in a VT configuration (the set of VTs to be used at any time) while satisfying constraints of physical link capacity and optimizing a specified objective function. Finally, we present the results of a preliminary simulation study comparing the performance of Virtual Trees with that of Virtual Paths.

2 Virtual Trees

Consider a network $G = (V, E)$ of nodes $n \in V$ interconnected by links $l \in E$. A source-destination pair is defined as a pair of nodes (n_1, n_2) where n_1, $n_2 \in V$ in which n_1 is the source of traffic and n_2 the traffic destination of the pair. Consider a source $\gamma \in V$ of traffic and a set of destinations $D_\gamma \subseteq V$ defining a set $\{(\gamma, d) | d \in D_\gamma\}$ of source-destination pairs with a common source γ. A *Virtual Tree* $VT_\gamma = (\gamma, V', E')$ is defined as a rooted tree in G, where V' is the set of nodes of the Virtual Tree such that $\gamma \in V'$ and $D_\gamma \subseteq V' \subseteq V$, and $E' \subseteq E$ is the set of links of the Virtual Tree. Each link in E' has a pre-allocated amount of bandwidth which is shared by all the paths that use the link. In every Virtual Tree, VT_γ, there is only one source of traffic located at the root γ. Every leaf node is a destination of traffic while every intermediate node is a transit node. In addition, some (possibly all) intermediate nodes may also be destinations. Note that a VT need not include all nodes in the network in which case no path exists from the source to the excluded nodes, just as a VP need

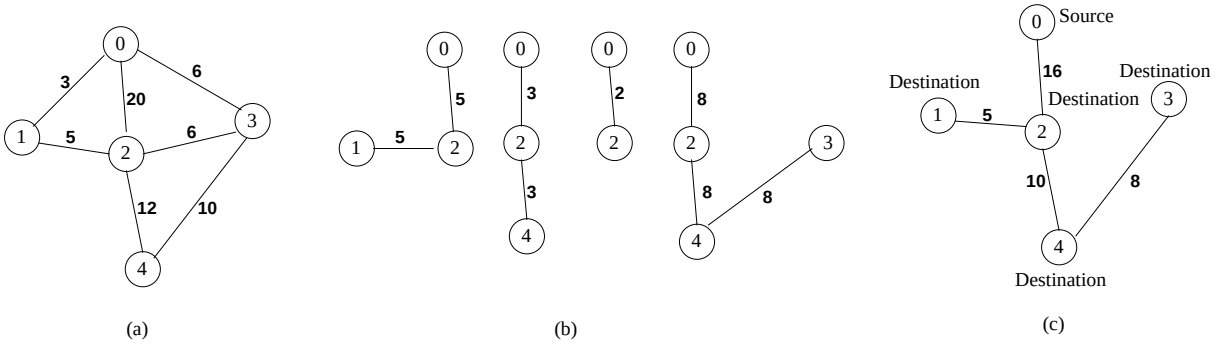

Figure 1: (a) A computer network G. Numbers along lines show link capacity. (b) A set of Virtual Paths in G originating at source 0 with numbers indicating bandwidth allocated to VP. (c) A Virtual Tree VT_0 from G rooted at the node 0. In a Virtual Tree, the amount of bandwidth allocated to a given link does not have to match the sum of equivalent bandwidths from the paths that use that link.

not exist between all pairs of nodes in the network. Also, it is possible to have multiple VTs rooted at the same source coexist with each other so as to provide multiple paths between a given pair of nodes, just as multiple VPs can exist between the same node pair.

Virtual Trees are more tolerant of inadequacies in the computation of the equivalent bandwidth and of sub-optimal bandwidth allocations to paths because such inadequacies are compensated by the sharing of the bandwidth between different source-destination pairs. With Virtual Paths, the sharing of bandwidth among different VPs is not possible, whereas in Virtual Trees, bandwidth can be shared by source-destination pairs with a common source that pass through a given link. Because of this, we should expect to obtain lower blocking probabilities with VTs than with VPs. In addition, VTs will be more bandwidth efficient for carrying multicast traffic.

The concept of a Virtual Tree and its relationship with the paths for source-destination pairs is depicted in Figure 1. Part (a) of this Figure shows a network with five nodes connected by links with specified bandwidths. The traffic from each source-destination pair is transmitted along the route selected for each one of them. Part (b) shows a set of Virtual Paths for the source-destination pairs (0-1), (0-2), (0-3), and (0-4) with a bandwidth allocated to each VP based on predicted traffic patterns. Part (c) shows a Virtual Tree rooted at the source node 0 for the same set of source-destination pairs. Bandwidth is now allocated to each link of the VT as shown. The total bandwidth allocated to a Virtual Tree's link is the equivalent bandwidth of all the traffic multiplexed in it; therefore, it might happen that the sum of the individual equivalent bandwidths used by the VPs at a given link will not match the equivalent bandwidth allocated to the same link in the VT.

Virtual Trees require a slightly more complex procedure for call admission at the source node (see below); nevertheless, the extra complexity added to the access node allows us to use Virtual Trees while mantaining the same complexity at the switching (transit) nodes as that needed by VPs.

At the call layer, a set of connection requests still needs the computation of its equivalent bandwidth in order to do bandwidth allocation and call admission with Virtual Trees. The techniques used for this purpose are similar to the ones used with Virtual Paths; however, at the path level, the computation of the equivalent bandwidth for a VT has to be done on a link-by-link basis given that there is no one-to-one relation between a Virtual Tree, VT_γ and a set of Virtual Paths with a common source γ.

In a normal call admission procedure with VPs, the access node proceeds with the bandwidth allocation after the equivalent bandwidth is determined. During the allocation of bandwidth, the access node determines whether or not there is enough available bandwidth in a Virtual Path to accommodate the incoming connection request. If there is enough bandwidth, the connection is granted (call is admitted), and a corresponding amount of bandwidth is subtracted from that available; otherwise, the call is rejected and no bandwidth is allocated. In certain cases, alternate Virtual Paths may be available to the same destination, or an attempt may be made to route the call using a concatenation of Virtual Paths.

For call admission with Virtual Trees, the access node makes use of a Virtual Tree link table which indicates the amount of bandwidth available for each link

of the Virtual Tree. Each entry of the Virtual Tree link table consists of two columns: a link identifier and the amount of bandwidth available. During a connection request, the access node determines whether or not there is enough bandwidth available at each link along the unique path defined in the VT for the source-destination pair. If there is enough bandwidth in all the examined links, the amount of bandwidth requested is subtracted from the bandwidth available for each link, and the call is admitted; otherwise, the call is rejected and no bandwidth is allocated.

The algorithm for call admission with Virtual Trees is described in more detail in [13]. It requires the access node to maintain the above information in appropriate data structures. All of the information needed to set up a VC is thus available locally at the access node, just as in VPs. The time complexity of the algorithm is $O(l)$ where l is the number of links along the VT from the source to the destination, as contrasted with $O(1)$ needed for Virtual Paths. Also, just as in VPs, if the algorithm fails to find enough bandwidth along this path, it may try an alternate VT if one exists rooted at the same source, or may try to concatenate paths from two or more VTs rooted at different nodes.

We have also described in [18] how the existing ATM cell structure with header fields for the Virtual Channel Identifier (VCI) and Virtual Path Identifier (VPI) can be used with no modification to use Virtual Trees. The routing tables and switching functions of the intermediate switching nodes are also unchanged. Thus VTs can be implemented with no changes in the ATM standards; the only differences from VPs arise in the allocation of bandwidth to the links of the VTs and in the call admission algorithm.

3 Bandwidth Allocation for Virtual Trees

One of the important problems that needs to be addressed is: Given an ATM network topology with various physical link capacities, and given a predicted traffic pattern between various source-destination pairs, how do we determine a Virtual Tree configuration that will carry the given traffic? A VT configuration is a set of Virtual Trees, with one or more VTs rooted at each potential traffic source. In this paper, we only explore configurations with exactly one VT rooted at each source. A number of algorithms exist in the literature for determining Virtual Path configurations [1, 2, 9, 10, 11, 17]. We discuss here possible methods

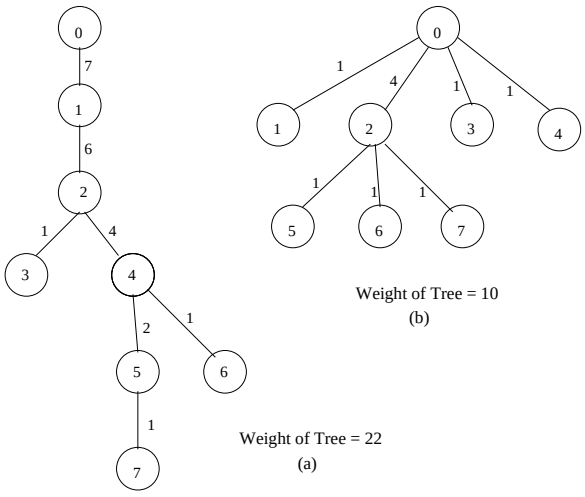

Figure 2: (a) A spindly tree and its weight. (b) A bushy tree and its weight.

to be used for VTs and explore some of the tradeoffs involved in choosing a VT configuration.

The main advantage of using a VT over using a set of VPs is the additional multiplexing afforded by the traffic from the common source flowing through the same link to different destinations. One way to conceptualise this multiplexing effect is to consider two types of Virtual Trees shown in Figure 2 that we call "Spindly" VTs and "Bushy" VTs respectively. It is intuitive that a spindly VT would result in a larger sharing of bandwidth between traffic flowing to different destinations than a bushy VT. Thus, spindly VTs should be expected to show a larger reduction in the blocking probabilities over VPs. However, using a spindly VT may also result in traffic to some destinations flowing over more links (i.e, a larger hop count) resulting in larger delays. We thus need to explore methods to achieve a balance between these opposing factors.

We now define two attributes of a VT called the *Weight* and the *Cost* that help us to arrive at such a balance.

The *Weight of a Virtual Tree link* is defined as the number of destinations that can be reached from the source through that link. The *Weight of a Virtual Tree* is defined as the sum of the weights of all its links. The weight of a Virtual Tree gives us a good indication of the shape of the tree. Spindly trees are characterized by large weights, whereas bushy trees are characterized by small weights. Figure 2 shows a spindly tree and a bushy tree with their respective weights. Note that the numbers next to each link rep-

resent the weight of the link.

The *Cost of a Link* is a function used to determine the cost of using a certain amount of bandwidth offered by a particular link. Most often, the cost function is defined as a factor σ (unit cost per $Mbps$) multiplied by the amount of bandwidth B that is being used (i.e., σB). Such a definition yields a linear cost function. However, the cost of the link may as well be defined as a non-linear function of the bandwidth and may also include other factors such as distance or type of physical medium used. The *Cost of a Virtual Tree* is defined as the sum of the costs of all its links. The cost of a Virtual Tree is also related to its shape. Trees that result in larger hop counts for paths and thus larger traffic delays will usually require more bandwidth and will have higher costs.

We can use the definitions of Weight and Cost above to formulate an optimization model to determine a Virtual Tree configuration that provides a good sharing of bandwidth at the lowest cost possible within the given constraints. Such a balance between weight and cost will also help reduce the delays between source-destination pairs. It should be noted, however, that such a VT will not necessarily yield the best performance measured solely in terms of blocking probabilities. It is also not appropriate to call this an "optimal VT" as there may be many different ways of defining the objective function each yielding a different definition of "optimality".

Consider a network $G = (V, E)$ with N nodes and L links, where $N = |V|$, and $L = |E|$. Each node is capable of being a source of traffic or a destination, and the network can have as many as $N(N-1)$ source-destination pairs.

We assume that an equivalent bandwidth E_ω has been obtained for the source-destination pair ω, and for all ω's, we will try to allocate E_ω for use by the pair ω to a Virtual Tree. Traffic need not actually exist between all source-destination pairs, in which case the appropriate value of E_ω will be 0.

For a link l, let C_l be its capacity and σ_l the unitary cost of using l (measured in units of $\frac{\$}{Mbps}$). We estimate that a linear cost function is good enough to make the analysis simpler and to provide a good abstraction of more realistic situations.

In a network G, we can enumerate $J = |B_0 \cdot B_0^t|$ spanning trees [5], where B_0 is the incidence matrix of the network G minus a row (i.e., reduced incidence matrix). A spanning tree from G is denoted by T_j, $j = 1..J$, which represents the j^{th} tree from the set of trees enumerated from G. We define $T_{j,n}$ ($j = 1..J$ and $n = 0..N-1$) as a rooted tree T_j with root n.

A Tree $T_{j,n}$ defines a set of links $L_{j,n} \subseteq E$, and the amount of bandwidth allocated to each link $l \in L_{j,n}$ is denoted by $B_{j,n,l}$. The value $W_{j,n}$ represents the weight of the tree $T_{j,n}$.

We introduce the multiplicative binary variable,

$$m_{j,n} = \begin{cases} 0 & : \quad \text{if } T_{j,n} \text{ is absent} \\ 1 & : \quad \text{if } T_{j,n} \text{ is present} \end{cases},$$

to indicate the absence or presence of the Virtual Tree $T_{j,n}$ in the optimized configuration. We define the normalized cost $\varphi_{j,n}$ of using a tree $T_{j,n}$ and the normalized weight $\kappa_{j,n}$ of a tree $T_{j,n}$ as

$$\varphi_{j,n} = \sum_{\forall l \in L_{j,n}} \frac{B_{j,n,l}\sigma_l}{G_1} \quad , \tag{1}$$

$$\kappa_{j,n} = \frac{W_{j,n}}{G_2}, \tag{2}$$

where G_1 and G_2 are normalizing constants that make comparable the cost of a Virtual Tree and its weight. The complete mathematical model is defined as follows:

Optimize

$$\text{Min} \qquad \sum_{j=1}^{J} \sum_{n=0}^{N-1} (\varphi_{j,n} - \kappa_{j,n}) m_{j,n} \tag{3}$$

Subject to

$$\sum_{j=1}^{J} m_{j,n} = 1 \quad \text{, for } n = 0..N-1 \tag{4}$$

$$\sum_{j=1}^{J} \sum_{n=0}^{N-1} B_{j,n,l} m_{j,n} \leq C_l \quad ,\forall (l \in L_{j,n}) \tag{5}$$

$$m_{j,n} = 0|1 \qquad ,\forall (j,n) \tag{6}$$

With the objective function (3), we optimize the algebraic difference of the normalized values of cost and weight from the final configuration of Virtual Trees. Our objective is to minimize the cost while maximizing the weight. The resultant configuration of Virtual Trees corresponds to the minimum value of (3) which satisfies the constraints in (4), (5) and (6).

The constraint (4) ensures the selection of only one Virtual Tree $T_{j,n}$, in the optimal configuration for each source n. The constraint (5) imposes the network bandwidth limitations to the Virtual Tree configuration. The amount of bandwidth pre-allocated to all the Virtual Trees that use a link $l \in L$ should at most equal the capacity of the link.

Finally, (6) enforces zero/one programming. With this model, because of the constraint (6), we only get one tree rooted at each source; however, when the network has very limited resources, it might happen that the solution for the configuration of single Virtual Trees becomes infeasible due to the lack of bandwidth. In such cases, we may relax the constraint (6) to allow more than one Virtual Tree rooted at a source (linear programming). The values of $m_{j,n}$ will then represent the fraction of bandwidth $B_{j,n,l}$ to be pre-allocated to each link l of each Virtual Tree $T_{j,n}$ in the optimal configuration.

4 Simulation Results

At the University of Delaware, we are in the process of designing and implementing a set of Simulation Testbeds for studying and evaluating the performance of Virtual Paths and Virtual Trees. The primary aim of these testbeds is to implement various algorithms for allocating bandwidth in an ATM network and simulate their behavior with a view to comparing the resulting performance.

We now present the results of a few preliminary experiments carried out by us using these simulation testbeds. In each case, the inputs to the testbed consist of the network configuration to be simulated and a profile of the traffic to be carried on it. This input is used to construct an optimization model for Virtual Trees as described in Section 3; for Virtual Paths, an optimization model similar to the one proposed by Hui et al [11] is used which generates an optimal VP configuration. The GAMS (General Algebraic Modeling System) package is used to solve these models. The VP and VT configurations are then used in a call-level simulation of the network to determine the average blocking probabilities and other performance metrics.

We ran tests on a small set of networks with 5, 6, and 7 nodes with link capacities of 2 Gbps and unit cost \$10 per Mbps. The equivalent bandwidth requested by a call was chosen randomly with a uniform distribution in the interval 1 Mbps to 10 Mbps. The call arrival process for each source-destination pair was modeled as a Poisson process, and the connection duration was exponentially distributed. We ran every simulation for a period of 100,000 *secs*. This period was determined by running a series of simulations intended to find the amount of time each configuration required to reach steady-state.

In Figure 3, we show the average blocking probability vs. Mean Load per source-destination pair for

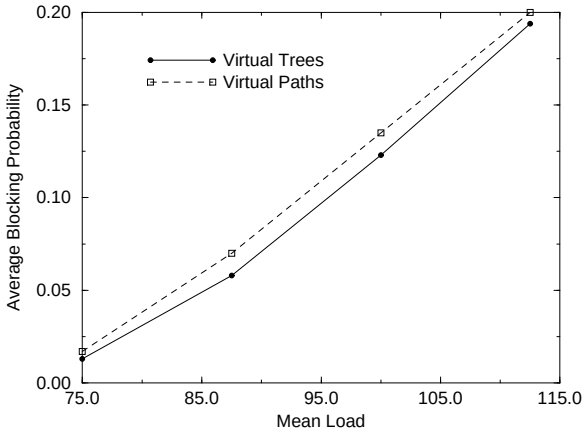

Figure 3: Average Blocking Probability (fraction of connections rejected) vs. Mean load for a fixed configuration of Virtual Trees and Virtual Paths.

an experiment where we used a fixed VP configuration and a fixed VT configuration. For these fixed configurations, the mean load was increased from 80 to 125 (measured as the ratio of mean call duration and the mean call interarrival time). As expected, the blocking probability increases with load since the VP and VT configurations have fixed capacities, but the VT configurations always gave lower blocking probabilities. The two curves tend to converge as the load increases since the traffic in the end far exceeds the capacity of the VPs and the VTs.

Figure 4 shows the results of an experiment on a different network where we changed the VP and VT configurations for each load such that we used VPs and VTs configured to carry the predicted load. In this case, the blocking probability actually decreases with increasing load because more capacity becomes available in the VPs and VTs leading to an increased multiplexing effect.

Figure 5 depicts a summary of results for a variety of networks with different numbers of nodes and links. All points in this figure were generated for the same predicted load and the same call parameters during the simulations. As a result, the blocking probability for Virtual Paths is the same for all networks whereas that for Virtual Trees varies with network configuration. For each network, the reduction in blocking probability from VPs to VTs lies in the range of 11-18%. This is typical of most of our observations.

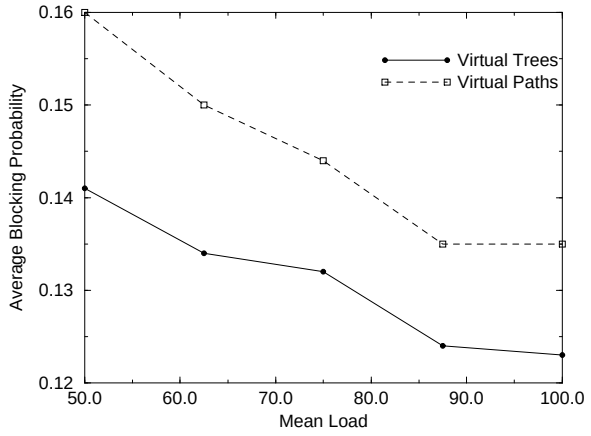

Figure 4: Average Blocking Probability (fraction of connections rejected) vs. Mean load for Virtual Trees and Virtual Paths reconfigured for each load level.

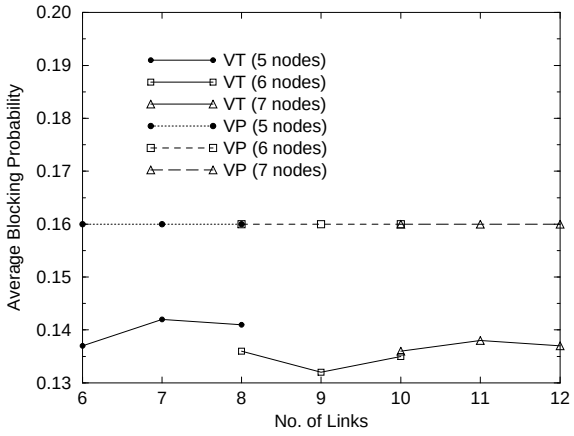

Figure 5: Average Blocking Probability (fraction of connections rejected) for Virtual Trees and Virtual Paths for fixed Load in networks with different numbers of nodes and links.

5 Conclusions

Virtual Paths are currently used in ATM networks as a technique for allocating and managing bandwidth. However, they only permit sharing of bandwidth among Virtual Channels that have the same source and destination. We have proposed the new concept of Virtual Trees that will enhance the sharing of bandwidth among VCs originating at the same source but going to different destinations. VTs provide performance improvements over VPs by reducing call blocking probabilities because of the increased multiplexing of traffic over common links. These advantages are achieved by preserving the biggest advantage of Virtual Paths, namely, all call admission decisions are made at the source.

In this paper, we proposed a model for bandwidth allocation for Virtual Trees that allocates capacity to each link of the VT under various constraints. We presented the results of some simulation experiments that contrasted the performance of a network under both VPs and VTs. We saw from these results that the use of VTs can yield significant reductions in blocking probabilities as compared with VPs. The model used by us to derive VT configurations needs further study to see how these results can be improved upon. In particular, it should be interesting to explore the tradeoffs between the use of spindly and bushy trees as well as the incorporation of other heuristics and cost functions into the model. We are continuing further studies along these lines using our simulation testbeds. We are also studying a generalization of Virtual Trees to Virtual Networks which will not have the limitation of a tree structure and should offer increased multiplexing benefits over both Virtual Paths and Virtual Trees.

Acknowledgements

The author would like to thank Antonio Mock who participated in the development of the simulation testbed and helped with some of the simulation experiments. The author would also like to thank the anonymous reviewers who provided valuable comments.

References

[1] R. Bolla, F. Danovaro, F. Davoli, and M. Marchese. Integrated dynamic resource allocation scheme for ATM networks. In *Proc. Infocom '93, 12th Annual Joint Conference of the IEEE Compu ter and Communications Societies*, volume 3, pages 1288–1292, San Francisco,CA, 1993.

[2] R. Bolla, F. Davoli, A. Lombardo, S. palazzo, and D. Panno. Adaptive bandwidth allocation by hierarchical control of multiple ATM traffic classes. In *Proc. Infocom '93, 12th Annual Joint Conference of the IEEE Compu ter and Communications Societies*, volume 1, pages 30–38, San Francisco,CA, 1993.

[3] J. Le Boudec. The Asynchronous Transfer Mode: A tutorial. *Computer Networks and ISDN Systems*, 24(4):279–309, May 1992.

[4] J. Burgin and D. Dorman. Broadband ISDN resource management: The role of virtual paths. *IEEE Communications Magazine*, 29(9):44–48, September 1991.

[5] N. Christofides. *Graph Theory - an Algorithmic Approach*. Academic Press Inc., 1978.

[6] M. Decina and T. Toniatti. On bandwidth allocation to bursty virtual connections in ATM networks. In *Proc. ICC '90, IEEE International Conference on Communications*, pages 844–851, Atlanta, GA, 1990.

[7] M. Decina, T. Toniatti, P. Vaccari, and L. Verri. Bandwidth assignment and virtual call blocking in ATM networks. In *Proc. Infocom '90, Ninth Annual Joint Conference of the IEEE Computer and Communications Societies*, pages 881–888, San Francisco, CA, 1990.

[8] S. Evans. Optimal resource management and capacity allocation in broadband integrated services network. In *Proc. Performance '90, 14th IFIP WG 7.3 International Symposium on Computer Performance Modelling, Measurement, and Evaluation*, pages 159–173, Edinburgh, Scotland, September 1990.

[9] M. Gerla, J.A.S. Monteiro, and R. Pazos. Topology design and bandwidth allocation in ATM nets. *IEEE Journal on Selected Areas in Communications*, 7(8):1253–1262, October 1989.

[10] J. Hui. Layered required bandwidth for heterogeneous traffic. In *Proc. Infocom '92, 11th Annual Joint Conference of the IEEE Computer and Communication Societies*, pages 13–20, Florence, Italy, 1992.

[11] J.Y. Hui, M.B. Gursoy, N. Moayeri, and R.D. Yates. A layered broadband switching architecture with physical or virtual path configurations. *IEEE Journal on Selected Areas in Communications*, 9(9):1416–1426, December 1991.

[12] M. Kawarasaki and B. Jabbari. B-ISDN architecture and protocol. *IEEE Journal on Selected Areas in Communications*, 9(9):1405–1415, December 1991.

[13] A. Mock. Virtual trees - a new technique for bandwidth allocation in ATM networks. Master's thesis, Dept. of Computer & Information Sciences, University of Delaware, Newark, DE, August 1993.

[14] J.A.S. Monteiro, M. Gerla, and L. Fratta. Statistical multiplexing in ATM networks. *Performance Evaluation*, 12(3):157–167, June 1991.

[15] K. Sato, S. Ohta, and I. Tokizawa. Broadband ATM network architecture based on virtual paths. *IEEE Transactions on Communications*, 38(8):1212–1222, August 1990.

[16] Y. Sato and K. Sato. Virtual path and link capacity design for ATM networks. *IEEE Journal on Selected Areas in Communications*, 9(1):104–111, January 1991.

[17] Y. Sato, N. Yamanaka, K.I. Sato, and N. Tokura. Experimental ATM transport system and virtual path management techniques. In *Proc. Globecom '91, IEEE Global Telecommunications Conference and Exhibition*, volume 3, pages 2110–2116, Phoenix, AZ, 1991.

[18] A.S. Sethi and A. Mock. Virtual trees - a new technique for bandwidth allocation in ATM networks. In *Proc. Second International Conference on Telecommunication Systems, Modeling and Analysis*, pages 113–119, Nashville,TN, March 1994.

[19] E. Sykas, K. Vlakos, and M. Hillyard. Overview of ATM networks: Functions and procedures. *Computer Communications*, 14(10):615–626, December 1991.

[20] W. Wang and T. Saadawi. Bandwidth allocation for ATM networks. In *Proc. ICC '90, IEEE International Conference on Communications*, pages 439–442, Atlanta, GA, 1990.