The Layout of Virtual Paths in ATM Networks

Ornan Gerstel, *Member, IEEE*, Israel Cidon, *Senior Member, IEEE*, and Shmuel Zaks

***Abstract*— We study the problem of designing a layout of virtual paths (VP's) on a given ATM network. We first define a mathematical model that captures the characteristics of virtual paths. In this model, we define the general VP layout problem, and a more restricted case; while the general case layout should cater connections between any pair of nodes in the network, the restricted case layout should only cater connections between a specific node to the other nodes. For the latter case, we present an algorithm that finds a layout by decomposing the network into subnetworks and operating on each subnetwork, recursively; we prove an upper bound on the optimality of the resulting layout and a matching lower bound for the problem, that are tight under certain realistic assumptions. Finally, we show how the solution for the restricted case is used as a building block in various solutions to more general cases (trees, meshes, K-separable networks, and general topology networks) and prove a lower bound for some of our results. The results exhibit a trade-off between the efficiency of the call setup and both the utilization of the VP routing tables and the overhead during recovery from link disconnections.**

I. Introduction

A. Background

ASYNCHRONOUS transfer mode (ATM) [9], [14], [15] is the transmission and multiplexing technique which is the emerging industry standard for B-ISDN. ATM was chosen by ITU (formerly CCITT), ANSI, and a large group of companies which are members of the ATM Forum. Due to the future importance of fast, broadband, integrated networks, ATM has been extensively discussed in recent years.

ATM is based on small fixed size packets, called *cells*. Due to the very high switching-rate requirements, the routing of cells must be carried at each network node by a dedicated hardware, implying simple routing algorithms. The routing scheme chosen in ATM is based on two fixed length fields in the header of each cell (VCI and VPI). These fields serve as indices into routing tables that reside at the nodes of the network, and they determine the route that a cell will take.

Routing in ATM is hierarchical in the sense that the VCI field of a cell is ignored in many nodes along the route, which perform the switching according to the VPI alone; only at a small number of nodes, is the VCI considered for switching the cell and determining the next VPI and VCI. This scheme effectively creates two types of predetermined unidirectional routes in the network: those based on VPI's (called *virtual path connections* or VP's) and those based on VCI's (called *virtual channel connections* or VC's).

These two route types have different roles in the network: while VC's are used for creating a connection between two users of the network (e.g., a telephone call), VP's are used for bundling together several VC's that share part of their route, thereby substantially reducing the magnitude of managed entities in the network. In particular, we are interested in two such management aspects.

1) Each VC requires a separate routing entry only at a small number of nodes, while in most of the nodes along its route, routing is performed in accordance to the VP in which it is contained and hence only a single routing entry is needed for that VP (rather than a separate VC entry at every node).
2) The time required for the setup of a new VC is proportional to the number of nodes in which routing tables must be updated. With VP's, the setup time depends on the number of VP's that are used by a VC (while without VP's, it depends on the total number of nodes in the path).

B. Motivation

The common view on the layout of VP's in an ATM network, is that VP's span through the entire network, connecting a pair of end nodes (or VP terminators) directly. In this view, a network of N nodes contains at least $N(N-1)$ VP's (typically much more, since multiple routes are desired between any pair of nodes, to overcome failures, to cater connections with different quality-of-service (QoS), and to enable better bandwidth allocation). Thus, it is plausible that the number of VP's at centrally located nodes will be $\Omega(N^2)$, overflowing the VP routing tables which are limited to 2^{12} entries—implying that this solution is impractical for large networks. Moreover, certain switch implementations further limit the number of bits actually used, because of hardware constraints and table size limitations. Such limitations are permitted by the UNI standard [9]. For example the Fujitsu ATM chip set [11] is limited to 10 bits per address.

A natural solution to this problem is a fragmentation of the network into domains, connected by VC switches (e.g., [13], [26]); however, this solution causes a VC to be routed through a relatively large number of VC switches (if it passes through many domains), directly influencing the time required for a VC setup.

Manuscript received August 22, 1994; revised November 15, 1996, March 11, 1996, and July 30, 1996; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor R. Watson. This work was done in part while O. Gerstel was with the Computer Science Department, Technion, Israel, and while I. Cidon was with Sun Microsystems Laboratories, Mountain View, CA.

O. Gerstel was with the Computer Science Department, Technion, Haifa, Israel. He is currently with the Optical Network Systems Group of the IBM T. J. Watson Research Center, Hawthorne, NY 10532 USA.

I. Cidon was with Sun Microsystems Laboratories, Mountain View, CA 94043 USA. He is currently with the Electrical Engineering Department of Technion, Haifa, Israel.

S. Zaks is with the Computer Science Department, Technion, Haifa, Israel.

Publisher Item Identifier S 1063-6692(96)08937-6.

In this paper, we propose a more integrated approach, in which the layout of VP's in a network is determined in a manner that allows to use small VP routing tables, and tune their size in accordance to the required call setup performance (thus exhibiting a tradeoff between the required time for call setup, and the size of the VP routing table).

Specifically, we study the problem of designing a virtual graph over a given ATM network, the nodes of which are the nodes of the network, and the edges of which represent VP's—we term this graph the *virtual path pair layout* (VPPL for short). Since network design is a complex task,[1] we separate the VPPL design from the design of the network itself; in other words, we design a VPPL for a given network, rather than changing the design of the network according to the layout considerations. In our layout design, we have the following assumptions.

A1) *Linear Connection Structure:* It is commonly assumed that VPs/VC's are coupled in pairs of unidirectional routes in opposite directions[2] since this structure improves connection management substantially [7]. This coupling defines *VP/VC pairs* (VPPs/VCP's), and in the sequel we refer exclusively to them rather than to VPs/VCs; thus, a VPP/VCP is a bidirectional, simple route in the network. In addition, a VCP may be viewed as composed of concatenated VPP's (refer to Fig. 1 and to example 1 for an example of these definitions).

A2) *Full Switching Capability:* We assume that each node can switch both VP's and VC's. This assumption is implied by an architecture in which VP and VC routing tables reside in every node/port-processor in the network. When a cell arrives at a node, its VPI is used to determine the next VPI (in the label swapping process), and the output port into which it is switched; during this process the cells' VCI is ignored as long as the new VPI is nonnull. Only when the VPI is null, is the VCI considered, to "demultiplex" the VC's that used the VP. The VC table determines the new VCI label and the output port (similar to the VP routing table), and also a new VPI label which matches the VP into which the VC is multiplexed (see [7] for a full description).

In realistic implementations, it is plausible that many nodes will switch VP's exclusively; however, incorporating this fact into the model complicates it (and hence the proposed solutions) with details that may damage the insight into the problem, which motivated this work. At the summary of this paper we propose a method for dealing with such "heterogeneous" networks.

A3) *Routing for Basic End-to-End Connectivity:* in this work we are not concerned with determining multiple routes between nodes, in accordance to bandwidth allocation or fault tolerance considerations. This attitude enables us to formulate the problem and solutions more clearly, and thus gain insight into them, while these additional parameters remain for future research. The proposed solutions are applicable with no modifications in certain scenarios, for instance for catering local-area network (LAN) emulation over ATM (i.e., relatively short messages which have no bandwidth guarantee).

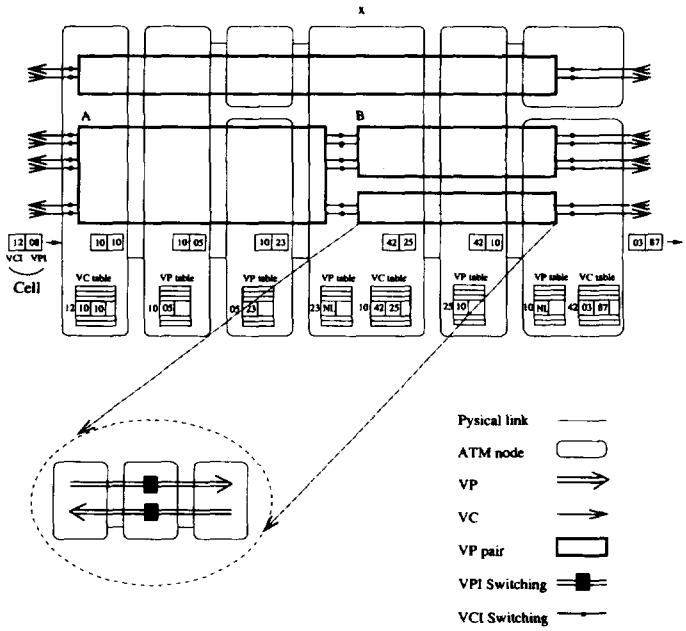

Fig. 1. The configuration of VP's and VC's in a simple network.

The following performance properties are affected by the design of VPPL. A "good" layout is characterized by achieving a good performance trade-off among them.

P1) *VCP Setup Complexity:* Low VCP setup complexity is important as it substantially reduces various overheads of connection management [6], [26], [27]. The setup complexity is proportional to the number of nodes in the VCP, in which the VCI is examined, since in these nodes the intervention of software is needed (to change the VC routing tables, to allocate bandwidth etc.). For this reason, the number of VPP's used for the routing of a VCP (termed *VP hop count*) should be small.

P2) *Length of Underlying Physical Route:* The chosen route for a VCP must also be short in terms of the number of physical links it uses, to efficiently utilize the communication network. In this work we restrict the discussion to shortest physical routes only. All the results are applicable if the shortest paths are based on shortest propagation delay rather than minimum number of links.

P3) *Utilization of VP Routing Tables:* The number of occupied entries in the VP routing tables (termed the *load* on the table) implied by the layout, should be low enough at any location in the network—see discussion in the beginning of this section.

[1] Network design typically involves multiple optimization criteria and a large number of input parameters, often resulting in a combination of automatic tools, heuristics, and human intervention during the process [16].

[2] Note, however, that this coupling is for routing purposes only, and other aspects of the unidirectional routes are managed separately, e.g., bandwidth allocation is not necessarily equal in both directions.

P4) *Recovery Overhead:* The resulting layout must overcome link disconnections with a low overhead. This is achieved by reducing the number of VPP's that share any link, so that if a link is disconnected, the number of VPP's that need to be rerouted, in order to bypass the faulty link, will be small (see [7] for a description of this recovery and reroute procedure). In the sequel we show that this property is closely related to P3.

Example 1: Consider the network in Fig. 1, in which a virtual path layout of 4 VPP's is depicted, and 4 VCP's that use this layout. The squares on the VP's denote the fact that at this point VP switching is done, while the circles denote VC switching. To demonstrate the label swapping process, consider the VPI and VCI fields of a cell in the lower part of the figure. This cell belongs to the lowest VC in the figure, going from left to right. The relevant VP and VC routing tables at each output port along the cell's path appear in the figure as well. Note that as long as the cell is routed through VPP A its VCI remains unchanged ($VCI = 10$). At node x, when the VPI is translated by the VP routing table to a null value (denoted "NL" in the figure), then VC switching is performed using the VCI of the cell and the VC is routed into VPP B (denoted by VPI 25 at x). The VCI is swapped to 42, a unique ID within VPP B.

C. Our Solution

We solve the VP layout problem in two stages. First, we define a more restricted form of the problem in which the VPPL is required to efficiently support VC's between a specific node (termed the root of the VPPL), and any other node. Such a layout is termed a "one-to-many" VP layout and is denoted by $\text{VPPL}^{1\to m}$. Next, we use the construction scheme for $\text{VPPL}^{1\to m}$ as a building block for constructing more general VP layouts, which are required to support VC's between any pair of nodes. Such layouts are termed "many-to-many" VP layouts and are denoted by $\text{VPPL}^{m\to m}$.

Our $\text{VPPL}^{m\to m}$ constructions are typically based on the following idea. Referring to Fig. 2, choose centrally located "hub" nodes in the network (depicted as dark squares in the figure). These nodes split the network into separated clusters, so that any VC between nodes in different clusters must be routed through these hubs (e.g., nodes x and y in the figure). For each hub construct a $\text{VPPL}^{1\to m}$ in each cluster, rooted at the hub. This $\text{VPPL}^{1\to m}$ will enable VC's to reach the hub with a small VP hop count. Thus, a VC between x and y (in the figure) can be set up using a small number of VPP's to the hub z and a small number of VPP's from z to y.

To enable the routing of VC's with both end points in the same cluster using short physical routes, each cluster is further split into subclusters with "second level" hubs and $\text{VPPL}^{1\to m}$s (depicted by dotted lines in the figure). This process is repeated a number of times to produce a "multilevel" solution which supports efficient routing of VC's both in terms of the VP hops (enabling efficient VC setup) and in terms of the shortest paths in the physical network. In the rest of the paper we present a precise formulation and analysis of the above ideas, tailored to increasingly general network topologies.

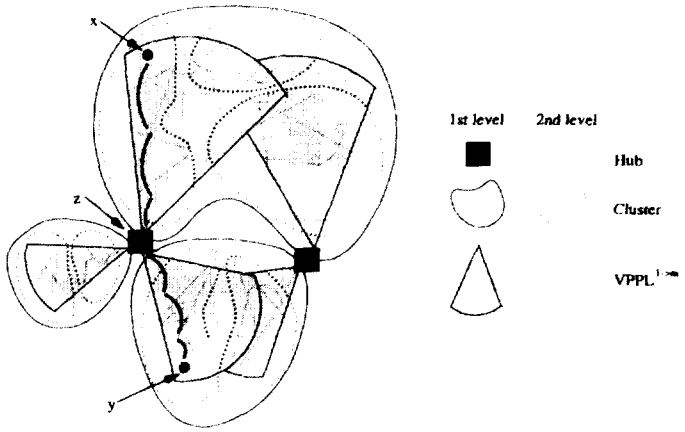

Fig. 2. General framework for the VP layout solution.

D. Related Works and Paper Structure

Most existing studies of the VP's in the network layout have considered only the case in which all VC's traverse a single VP through their entire end-to-end path (e.g., [13], [27]). As discussed above, such a design choice is not scalable and is suitable only for relatively small networks.

Several works have considered planning a VP layout so that a VC is routed through multiple VP's [1], [19]. In these works, many parameters are taken into account, which make the problem too complex for mathematical analysis. Consequently, an experimental approach is taken, based on heuristic optimization techniques. In contrast with these works, the present work (together with [12]) considers only few parameters, and is based on an analytical approach. We thus gain better insight into the problem, present simpler and more efficient algorithms, and base the performance analysis of the results on mathematical (rather than empirical) tools.

Our paper assumes a switch model in which VPI's (and VCI's) are looked at and swapped using link based tables. Thus the same VPI values can be received from multiple incoming links to indicate different VP's. In a closely related paper [12], we consider a different (and more restrictive) switch model in which a central VP table serves the whole switch (and hence the same VPI values cannot be reused by different incoming links). The "centralized" approach of [12] relies on the use of switch-wide shared memory which is not applicable for most space type switches. It is also a further constraint on the possible use of the already limited VPI space. In addition to the different switch models, the solution approaches taken in both papers are very different. While here we focus on the quantitative bounds that prove the practical feasibility of our solutions, in [12] we focus on more theoretical, qualitative results (such as proof of NP completeness and obtaining provably optimal results for rather restrictive cases).

A problem related to ours is that of keeping small routing tables for routing in traditional datagram networks. This problem was widely studied (for example, in [2], [3], [10], [17], [18], and [22]) and yielded interesting graph decompositions and structures, but it differs from ours in some major aspects which deemed most of these solutions impractical for our purposes. Some of these differences stem from the fact that in our case

there is no flexibility as to the nodal intermediate routing scheme itself since it is determined by the ATM standard [15], and by the requirement for very fast routing. For this reason we present a static structure in the network, while in the traditional model, the exact routing of a packet may be determined dynamically during its routing process (as in [2] and [3]), or by information based on the name of the destination (as in [10]).

Other differences stem from the ATM standard, in which the sizes of the routing tables at each node are fixed—implying a worst-case approach to the utilization of the tables, in contrast to the average-case approach, adopted by some of the solutions to traditional datagram routing (for example, in [10] and [22]). Many of the general solutions route packets in paths that are up to a multiplicative factor longer than the shortest path (this factor is termed *stretch factor*). These solutions are usually based on a large factor and are therefore not practical for our purposes.

The separator-based techniques for decomposing a graph that we have used in this work resemble those of [10]. Another related work is [5], in which a similar problem which arises in databases is studied for tree networks. The main difference between the problems (which thoroughly affects the solutions) is that here we are concerned with the load of VP's at any point in the network (a "local" property) while they are interested in their total number (a "global" property).

The paper is structured as follows: we start (in Section II) by formally defining our assumptions, the problem in its restricted and general form, and the essential characteristics of a "good" layout. In Section III, we propose a solution to the restricted problem that is based on a structural decomposition of a network into small subnetworks, and a recursive solution for each subnetwork. This solution is presented in stages, and is proven to yield a quantitative upper bound to the efficiency of the layout and an asymptotically matching lower bound (under realistic assumptions). In Section V, we show how the solution for the restricted problem is used for solving the general problem for various network topologies: trees, K-separable networks, meshes, and general topology networks. For the tree case, we also prove the asymptotic optimality of the result (under certain realistic assumptions). We conclude in Section VII by presenting both a qualitative and numerical comparison between the various schemes in the paper.

II. The Mathematical Model

In order to properly analyze the virtual path properties and layout, we first define a graph-theoretic model for it (for basic terms and definitions, see [8]). In our model, we have an underlying communication network, which consists of nodes and links between them. This network is modeled by an undirected graph $G = (V, E)$, where V corresponds to the set of nodes and E to the set of physical links between them.

Definition 1: Let $\mathcal{P}(G)$ be the set of all simple paths[3] in G. A *virtual path layout* is a subset of $\mathcal{P}(G)$. Formally, it is convenient to represent a VPPL Ψ by $\Psi = (G_\Psi, \mathcal{I})$, where $G_\Psi = (V, E_\Psi)$ is a "virtual" graph and $\mathcal{I}$: $E_\Psi \rightarrow \mathcal{P}(G)$ is a "mapping" function. A "virtual" edge $\psi = (a, b) \in E_\Psi$ represents a VPP between the nodes a and b. The function $\mathcal{I}(\psi)$ maps each virtual edge $\psi = (a, b)$ to its corresponding route in G. We term this path the *induced path* of ψ.

We extend the definition of $\mathcal{I}$ to paths in the virtual graph G_Ψ, as follows:

Definition 2: The *induced path* $\mathcal{I}(p)$ for a path $p \in \mathcal{P}(G_\Psi)$, $p = (\psi_1, \psi_2, \cdots, \psi_k)$, ($\psi_i \in E_\Psi$ for all i) is the path obtained by concatenating the induced paths of all ψ_is.

Next, we define the number of VP routing table entries consumed by the VPPL. Underlying this definition is the prevalent assumption that a VP routing table resides in each port processor. We also assume that each VPP is a bidirectional route (see assumption A1). Hence, each VPP that goes through a physical link contributes one to the utilization of the VP routing table at the port processors that are connected via that link (property P3).

Definition 3: The *load* $\mathcal{L}(e)$ on a link $e \in E$ is the number of VPP's $\psi \in E_\Psi$ that include e in their induced paths. Namely,

$$\mathcal{L}(e) = |\{\psi \in E_\Psi | e \in \mathcal{I}(\psi)\}|$$

the *load* $\mathcal{L}(\Psi)$ of a given VPPL Ψ is

$$\mathcal{L}(\Psi) = \max_{e \in E} \mathcal{L}(e).$$

Observation 1: The above load definition is also suited to express the fault tolerance overhead when a given link is disconnected [property P4].

Explanation: When a link fails, the VC's and VP's that are using it are broken. A simple and fast technique for recovering these connections is to reroute VP's that use the link, to other routes in the network, an operation which automatically heals VC's that are using these VP's [7]. Thus, the overhead of computing alternate routes for the VP's and for activating control protocols that set them up, is proportional to the number of VP's that may be affected by a link failure, which is equal to the load $\mathcal{L}(e)$ on the faulty link e. □

Definition 4: The *hop count* $\mathcal{H}(v, w)$ for $v, w \in V$ is the minimum number of VPP's that may be used to form a VCP between v and w, such that the VCP uses a shortest path in the physical network. Namely, it is the minimum k such that:

1) $\exists p = (\psi_1, \psi_2, \cdots, \psi_k) \in \mathcal{P}(G_\Psi)$,
2) $\exists x, y \in V, \psi_1 = (v, x), \psi_k = (y, w)$.
3) The induced path $\mathcal{I}(p)$ is a shortest path between v and w in G.

If no such k exists, define $\mathcal{H}(v, w) = \infty$.

Observation 2: There exist nodes v and w in the network with $\mathcal{H}(v, w) = \infty$ iff $E \not\subseteq E_\Psi$.

Explanation: Otherwise, there is always a path between v and w in the VPPL along a shortest path in the physical network G: a path that uses single link VP's. Furthermore, in this case there are always such v and w that are adjacent in G. □

If the paths should be shortest possible in their propagation delay rather than the number of links, the only change in the model should be performed in this definition. In the rest of the

[3] For the sake of notational convenience we refer to a path $p \in \mathcal{P}(G)$ either as a set of edges or as a set of nodes.

paper, each occurrence of the term "shortest path" should be replaced by "a path with shortest propagation delay."

We distinguish between two problems: 1) The general layout problem in which it is required to support the setup of VCP's between every pair of nodes; we term this case the "many-to-many" VP layout. 2) A more restricted case, when the VPPL should cater to VCP's from a single node (called the *root*) to all the other nodes; we term this case the "one-to-many" case. The VPPL for the many-to-many case is denoted by $\text{VPPL}^{m\to m}$, while a VPPL for the one-to-many case is denoted by $\text{VPPL}^{1\to m}$.

Definition 5: The following definitions capture the feasibility of the above-mentioned problems with respect to a given hop constraint $h > 0$:

1) A $\text{VPPL}^{m\to m}$ is *h-feasible* if $\max_{v,w\in V} \mathcal{H}(v, w) \le h$.
2) Let $r \in V$ be the root. A $\text{VPPL}^{1\to m}$ is *(h, r)-feasible* if $\max_{v\in V} \mathcal{H}(v, r) \le h$.

Notation: When the scheme for producing a feasible $\text{VPPL}^{m\to m}$ from G and h is understood from the context, the VPPL will be denoted by $\Psi^{m\to m}(G, h)$; for the $\text{VPPL}^{1\to m}$ case the notation will be $\Psi^{1\to m}(G, h, r)$. For notational convenience, when G belongs to a given family and $|V| = N$ we replace G by N.

The feasibility of a VPPL captures the notion of a VPPL in which the worst-case VPP hop count is bounded by h (property P1 above), and the chosen routes are minimal in the underlying physical network [property P2]. We now define an optimal solution as a solution that, in addition, minimizes the utilization of the VP routing tables [property P3], and enables efficient recovery from link disconnections [property P4].

Definition 6: A $\text{VPPL}^{m\to m}\Psi$ is *h-optimal* for a given h, if it is h-feasible and its load $\mathcal{L}(\Psi)$ is minimal amongst all other h-feasible VPPL's. This definition is extended in a straightforward manner to the (h, r)-optimality of a $\text{VPPL}^{1\to m}$ (for a given root r).

Observation 3: Finding a $\text{VPPL}^{1\to m}$ is easier than finding a $\text{VPPL}^{m\to m}$ as hinted by the following facts:

1) Given a network G, every h-feasible $\text{VPPL}^{m\to m}$ is also a (h, r)-feasible $\text{VPPL}^{1\to m}$ for any $r \in V$, but the reverse is not necessarily true.
2) The load $\mathcal{L}(\Psi)$ of an h-optimal $\text{VPPL}^{m\to m}$ is never less than the load of a (h, r)-optimal $\text{VPPL}^{1\to m}$. □

Besides its methodical value as an easier problem to be tackled first, $\text{VPPL}^{1\to m}$ has its own practical importance, as it may prove useful for server networks, where data is sent from a single source to multiple destinations and vice versa. An example for this is a video conferencing server—which has VCP's to all users who are currently engaged in a video conference[4] [21], [23].

III. The $\text{VPPL}^{1\to m}$ Problem for Tree Networks

In this section, we present results that concern the construction of a $\text{VPPL}^{1\to m}$. The results are presented for increasingly complex cases. We first present a method for finding a $\text{VPPL}^{1\to m}$ for the case when the network is a chain of nodes (an array of N nodes connected in a row) and then extend it to arbitrary tree networks. For the chain case, we start with $h = 2$, and extend the results to an arbitrary h.

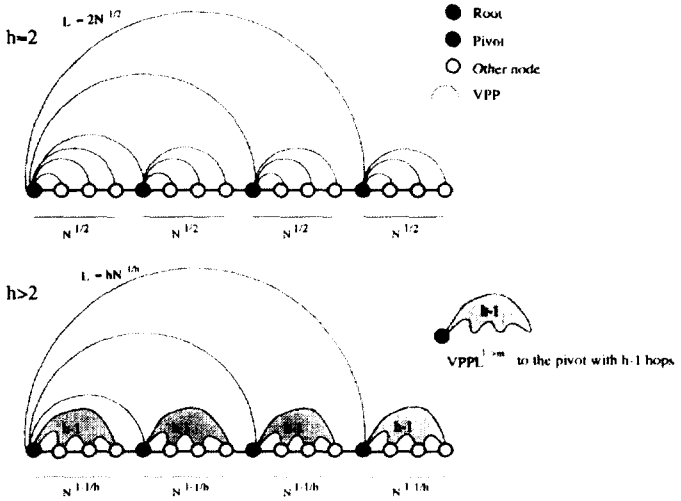

Fig. 3. $\text{VPPL}^{1\to m}$ on a chain of nodes.

Given a chain of N nodes and a root r at one of its ends[5] construct $\text{VPPL}^{1\to m}$ in the following way: First split the chain into $\sqrt{N}$ equal sections of size $\sqrt{N}$ each.[6] Call the node that is closest to r in section i the "pivot of section i." Connect r to all pivots by VPP's and connect the pivot of section i to all the nodes in its section (see Fig. 3). This construction for $h = 2$ can be extended to any h by the following recursive scheme (see Fig. 3 for a demonstration).

Scheme 1—A $\text{VPPL}^{1\to m}$ on a Chain:

1) Divide the chain into $N^{1/h}$ sections of size $N^{1-1/h}$ each.
2) Connect each pivot to the root r by a direct VPP.
3) Connect each pivot to its section by a $\text{VPPL}^{1\to m}$ with up to $h - 1$ VP-hops [i.e., let S_i be the subgraph of section i, p_i the pivot of the section then construct $\Psi^{1\to m}(S_i, h - 1, p_i)$].
4) The resulting VPPL is a union of the partial VPPL's constructed in earlier steps.

Correctness: The underlying route from any node to the root is the shortest possible, since the path advances only in the direction of the root. Also, each node can reach r using no more than h hops: $h - 1$ hops to the pivot of its section and one more hop to r. □

Load Analysis: Let S the subgraph of a given section, whose pivot is p. The load on each link in S is affected only by VPP's that connect pivots of sections farther away from r, to r, and by the next level $\text{VPPL}^{1\to m}$ in S. Thus, the load of

[4] This is not to be confused with a multicast service, where all destinations receive the same data from a given source, while here we discuss separate streams of data from a service center.

[5] It is easy to see that if the root is not at an end of the array, then the problem may be decomposed into two independent subproblems with r at the end of each subchain.

[6] If N is not a perfect square then the sections are of size $\lfloor\sqrt{N}\rfloor$ and there also exists a shorter section of the remaining nodes. However, for the sake of simplicity, we shall ignore such cases henceforth, since it is easy to infer from the techniques presented in the paper how they may be dealt with.

the VPPL$^{1\to m}$ satisfies the following recurrence formula:

$$\mathcal{L}[\Psi^{1\to m}(G, h, r)] \le N^{1/h}\mathcal{L}[\Psi^{1\to m}(S, h-1, p)]$$

or (together with the boundary condition)

$$\mathcal{L}[\Psi^{1\to m}(N, h)] \le \begin{cases} N-1, & h=1, \\ N^{1/h} + \mathcal{L}[\Psi^{1\to m}(N^{1-1/h}, h-1)], & h>1. \end{cases}$$

It can be shown by induction that

$$\mathcal{L}[\Psi^{1\to m}(N,h)] \le hN^{1/h}$$

since

$$\begin{aligned}\mathcal{L}[\Psi^{1\to m}(N, 1)] &= N-1 < 1\cdot N^{1/1} \\ &= N\end{aligned}$$

and

$$\begin{aligned}hN^{1/h} &\le 1\cdot N^{1/h} + (h-1)(N^{1-1/h})^{1/(h-1)} \\ &= N^{1/h} + (h-1)N^{(h-1)/h(h-1)} \\ &= hN^{1/h}.\end{aligned}$$

□

We now extend Scheme 1 from chains to arbitrary tree networks (with an arbitrary h). We shall need the following graph-theoretic result on trees:

Definition 7: Given a graph G, an (α, β)-*separator* is a set of nodes S whose size does not exceed α and whose removal (together with adjacent edges) separates the graph into subgraphs, each with size not greater than β. These subgraphs are termed the *separated components* of S.

Lemma 1 [5]: Given a tree T with N nodes and an integer $k>0$, there exists a $[k, \lceil N/(k+1)\rceil]$-separator for T.

Furthermore, the algorithm that is implied by the proof of the lemma is linear in the tree size N. Using Lemma 1, we can now construct a layout for arbitrary trees by the following scheme (see Fig. 4 for a graphic demonstration).

Scheme 2—A VPPL$^{1\to m}$ on a Tree:

1) If $h=1$—connect each node to the root by a direct VPP.
2) Given $h>1$, choose $k = N^{1/h}$.
3) Find a $[k, N/(k+1)]$-separator P for T (as shown in Lemma 1) and define the nodes of the separator P to be the pivots. Let C denote the set of separated components of P.
4) Connect the root r to all the pivots $p \in P$ by direct VPP's.
5) Connect each pivot $p \in P$ to each separated component $c \in C$ which contains a node adjacent to p, unless the path from p to the root goes through c, using a VPPL$^{1\to m}(c, h-1, p)$.
6) If the root is not a pivot, connect it to the separated component $c \in C$ it belongs to, by a VPPL$^{1\to m}(c, h-1, r)$
7) The resulting VPPL is a union of the partial VPPL's constructed in earlier steps.

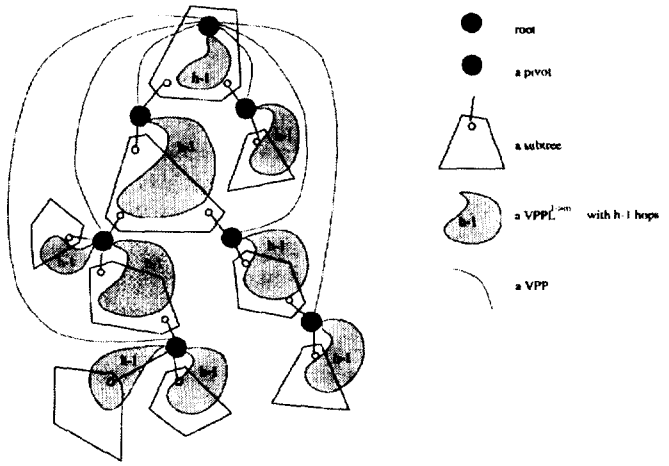

Fig. 4. VPPL$^{1\to m}$ on a tree.

Correctness: To reach the root, each node v must first reach a pivot (by no more than $h-1$ hops, using the shortest path—as guaranteed by the recursive application of the scheme), and then one additional hop to the root using a shortest path. If the root is not adjacent to the separated component in which v resides, then there exists a pivot which is on the shortest path from v to the root of the tree. □

Load Analysis: Each link may be used by no more than k VPP's that connect the pivots $p \in \mathcal{P}$ to the root r; it may also be used by a VPPL$^{1\to m}$ with $h-1$ VP-hops which connects the separated component that contains the link, to the pivot (the same argument applies to links which connect pivots to components, and we ignore them in what follows for the sake of simplicity). Since each link belongs to a single component c, and there is a single pivot p adjacent to c such that c is not on the path between p and the root—there is only one such VPPL$^{1\to m}$. As for the boundary condition, for $h=1$ there are direct VPP's from every node to the root, yielding a load of $N-1$ in the worst case. The load function thus satisfies

$$\mathcal{L}[\Psi^{1\to m}(N, h, r)] \le \begin{cases} N-1, & h=1 \\ \mathcal{L}[\Psi^{1\to m}(k, 1, r)] & \\ \quad + \mathcal{L}\left[\Psi^{1\to m}\left(\dfrac{N}{k+1}, h-1, p\right)\right], & h>1. \end{cases}$$

By the choice of k, we get $\mathcal{L}[\Psi^{1\to m}(k, 1, r)] \le N^{1/h}$. Combining these (and the fact that $\mathcal{L}[\Psi(x, y, z)]$ increases as x increases), we get

$$\mathcal{L}[\Psi^{1\to m}(N, h, r)] \le N^{1/h} + \mathcal{L}[\Psi^{1\to m}(N^{1-1/h}, h-1, p)].$$

Which identifies with the recurrence for the chain case. Thus

$$\mathcal{L}[\Psi^{1\to m}(N, h, r)] \le h\cdot N^{1/h}.$$

□

This result is asymptotically optimal for most real-life cases (in which Δ and h are small), as proven by the following theorem.

Theorem 1: Let T be a tree network with N nodes, let Δ be the maximum degree of a node, and $h > 1$. For every VPPL$^{1\to m}$ with h hops, there exists a link $e \in E$ with load

$$\mathcal{L}(e) \geq \frac{1}{2^{1/h}} \cdot \frac{1}{\Delta} N^{1/h}.$$

Proof: Let L be an upper bound on the load in a given VPPL with h hops. Start at the root r and count the maximum number of nodes reachable from r. Fewer than $L\Delta$ nodes are one hop away from r (at most L VPP's on each link adjacent to r, and up to $\Delta - 1$ such links). For each such node v, at most $L\Delta$ nodes are one hop away from v, and thus are two hops from r—a total of $(L\Delta)^2$ such nodes. By repeating this argument h times, we get up to $(L\Delta)^h$ nodes that are h hops from r. So the number of nodes which are up to h hops from r does not exceed

$$\begin{aligned}\sum_{i=1}^{h}(L\Delta)^i &= \frac{(L\Delta)^{h+1}-1}{L\Delta - 1} \\ &\leq \frac{(L\Delta)^{h+1}}{0.5L\Delta} \\ &= 2(L\Delta)^h.\end{aligned}$$

On the other hand, the size of the network N must not exceed this number (since every network node must be reached), so we get

$$N \leq 2(L\Delta)^h$$

or

$$L \geq \frac{1}{2^{1/h}} \cdot \frac{1}{\Delta} N^{1/h}.$$

□

IV. The VPPL$^{m\to m}$ Problem for Tree Networks

The main motivation for studying the VPPL$^{1\to m}$ problem is that its solutions may be used as a building block in the solution of the general VPPL$^{m\to m}$ problem. We now demonstrate this by building a VPPL$^{m\to m}$ for arbitrary tree networks. The recursive construction scheme follows (see also Fig. 5).

Scheme 3—A VPPL$^{m\to m}$ for Trees:

1) Recalling Lemma 1, choose a $(1, N/2)$-separator s of T as a pivot,
2) Construct a VPPL$^{1\to m}$ with up to $h/2$ VP-hops in each subtree, with s as its root,
3) Recursively, build a VPPL$^{m\to m}$ with up to h VP-hops in each subtree,
4) The resulting VPPL$^{m\to m}$ is a union of the partial VPPL$^{1\to m}$s constructed in earlier steps.

Correctness: Note that two nodes that reside in different subtrees may be connected using no more than h VPP's, by going from one node to the pivot (no more than $h/2$ hops), and from the pivot to the other node (again, no more than $h/2$ hops). This is achieved by steps 1) and 2) in the scheme. Step 3) is necessary to enable such a connection on a shortest route in the tree. Thus, if the nodes reside in the same subtree, then they are catered to by the recursive application of the scheme in that subtree. □

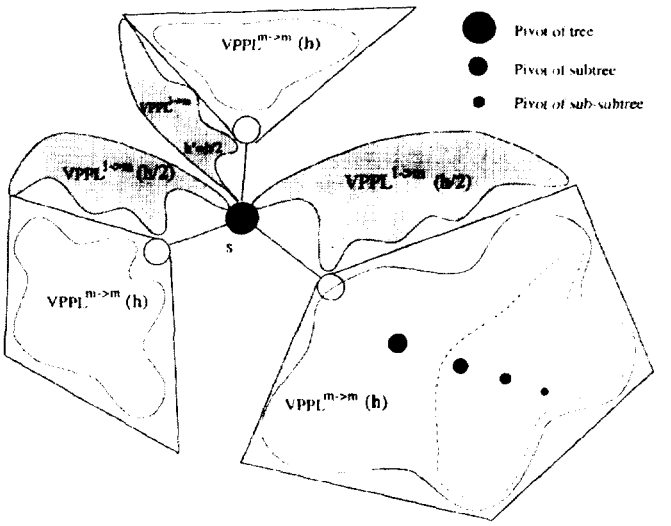

Fig. 5. VPPL$^{m\to m}$ on a tree.

Load Analysis: The load on each link is composed of the load of the VPPL$^{1\to m}$ to the pivot and the load of the recursive application of VPPL$^{m\to m}$ in the subtree. Since a link participates in one such VPPL$^{1\to m}$ and one smaller VPPL$^{m\to m}$ (of its subtree), and since by Definition 7, a subtree of the $(1, N/2)$-separator contains no more than $N/2$ nodes, its load is bounded by the following recurrence equation.

$$\mathcal{L}[\Psi^{m\to m}(h, N)] \leq \begin{cases} 1, & N \leq h, \\ \mathcal{L}\left[\Psi^{m\to m}\left(h, \frac{N}{2}\right)\right] & \\ \quad + \mathcal{L}\left[\Psi^{1\to m}\left(\frac{h}{2}, \frac{N}{2}\right)\right], & N > h. \end{cases}$$

Let $\mathcal{L}(N) = \mathcal{L}[\Psi^{m\to m}(h, N)]$, then by using the result obtained for VPPL$^{1\to m}$ on trees we get

$$\mathcal{L}(N) \leq \mathcal{L}\left(\frac{N}{2}\right) + \frac{h}{2}\left(\frac{N}{2}\right)^{2/h}.$$

So,

$$\begin{aligned}\mathcal{L}(N) &\leq \mathcal{L}\left(\frac{N}{2}\right) + \frac{h}{2}\left(\frac{N}{2}\right)^{2/h} \\ \mathcal{L}\left(\frac{N}{2}\right) &\leq \mathcal{L}\left(\frac{N}{4}\right) + \frac{h}{2}\left(\frac{N}{4}\right)^{2/h} \\ &\vdots \\ \mathcal{L}(2h) &\leq \mathcal{L}(h) + \frac{h}{2}(h)^{2/h} \\ \mathcal{L}(h) &\leq 1.\end{aligned}$$

By summing the above equations we get

$$\begin{aligned}\mathcal{L}(N) &\leq 1 + \frac{h}{2}\left[\left(\frac{N}{2}\right)^{2/h} + \left(\frac{N}{4}\right)^{2/h} + \cdots + h^{2/h}\right] \\ &\leq \frac{h}{2}\frac{\left(\frac{N}{2}\right)^{2/h}}{1 - \frac{1}{2^{2/h}}} \\ &= \frac{h}{2(2^{2/h}-1)} N^{2/h}.\end{aligned}$$

Based on the following two lemmas, we prove a lower bound for this case as well.

Lemma 2: Given a tree and a $(1, N/2)$-separator s, the number P^s of pairs of nodes, whose route goes through s it is at least $\frac{1}{4}N^2$.

Proof: Let δ be the degree of s, let t_i, $1 \le i \le \delta$ be the sizes of the subtrees adjacent to s. Without loss of generality, assume s is in t_1. Clearly $P^s = \sum_{1 \le i < j \le \delta} t_i \cdot t_j$. By changing the order of the sum we get

$$P^s = \tfrac{1}{2} \sum_{1 \le i \le \delta} t_i (N - t_i)$$

or

$$P^s = \tfrac{1}{2} \left[\sum_{1 \le i \le \delta} N t_i - \sum_{1 \le i \le \delta} t_i^2 \right] = \tfrac{1}{2} \left[N^2 - \sum_{1 \le i \le \delta} t_i^2 \right].$$

A lower bound on P^s is obtained using an upper bound on the sum $\sum_{1 \le i \le \delta} t_i^2$, which is maximal if δ is minimal and each t_i is maximal. However for a $(1, N/2)$-separator $t_i \le N/2$, so we get

$$P^s \ge \frac{1}{2} \left[N^2 - \frac{N^2}{2} \right] = \tfrac{1}{4} N^2 .$$

□

Lemma 3: The number P_h of pairs of nodes that may use a given VPP, in an up to h hop route does not exceed $2h(L\Delta)^{h-1}$.

Proof: Given a VPP (a, b), the number of pairs of nodes that may use it in a one hop route is 1 (only a and b). The number of VPP's for which a (or b) is an end point does not exceed $L\Delta$, thus at most $2L\Delta$ pairs use (a, b) in a two hop route (half of which are pairs that include a, the other half include b). The number of pairs that include (a, b) in a three hop route does not exceed $3(L\Delta)^2$, since they may 1) include a as one end point, and a node which is two hops away from a in b's direction as the other; 2) include b and a node which is two hops from b in a's direction; or 3) include a node which is one hop away from a, and a node which is one hop away from b. By repeating the argument h times, we get

$$P_h \le 1 + 2(L\Delta) + \ldots + h(L\Delta)^{h-1} = \sum_{i=1}^{h} i(L\Delta)^{i-1}$$

or

$$\begin{aligned} P_h &= \frac{d}{d(L\Delta)} \sum_{i=1}^{h} (L\Delta)^i \\ &= \frac{d}{d(L\Delta)} \left[\frac{(L\Delta)^{h+1} - 1}{L\Delta - 1} \right] \\ &= \frac{(h+1)(L\Delta)^h (L\Delta - 1) - [(L\Delta)^{h+1} - 1]}{(L\Delta - 1)^2} \\ &= \frac{h(L\Delta - 1)(L\Delta)^h - (L\Delta)^h + 1}{(L\Delta - 1)^2} \\ &\le \frac{h(L\Delta)^h}{L\Delta - 1} \\ &\le 2(L\Delta)^h \\ &= 2h(L\Delta)^{h-1} . \end{aligned}$$

□

Theorem 2: Let T be a tree network with N nodes, let Δ be the maximum degree of a node, and $h > 1$. For every VPPL$^{m \to m}$ with h hops, there exists a link $e \in E$ with load

$$\mathcal{L}(e) \ge \frac{1}{(8h)^{1/h}} \frac{1}{\Delta} N^{2/h}.$$

Proof: The number of pairs that use an up-to-h hop route, which passes through a given node v, does not exceed $L\Delta \cdot P_h$, since each VPP that passes through v can support up to P_h pairs.

Combining this with Lemmas 2 and 3, we get

$$L\Delta \cdot 2h(L\Delta)^{h-1} \ge \tfrac{1}{4} N^2$$

or

$$\mathcal{L}(e) \ge \frac{1}{(8h)^{1/h}} \cdot \frac{1}{\Delta} N^{2/h} .$$

□

Note that for most real-life cases (with a small Δ, h), the load of the construction and the lower bound of Theorem 2 are asymptotically tight.

V. Beyond Tree Networks

A. K-Separable Networks

So far, we have concentrated on ATM networks with a tree topology. The above technique can be easily extended for wider families of networks as well—namely K-separable networks.

Definition 8: Given $K > 0$, a graph G is *K-separable* if it contains a $(K, \frac{2}{3}N)$-separator, and each remaining subgraph is K-separable as well.

This family includes many known graph families, in particular graphs with "bounded treewidth" (see [24]) which include e.g., rings, chordal rings, interval graphs, circular arc graphs, series-parallel graphs, outer planar graphs, and cographs [4]; planar networks are $\sqrt{N}$-separable [20].

Finding a $(K, \frac{3}{4}N)$-separator in a graph in which such a separator exists can be done in linear time if K is small enough [24]. Furthermore, for graphs with treewidth $\le K$, a $(K, \frac{2}{3}N)$-separator may be found in linear time using flow techniques [25]. Based on such algorithms, a construction scheme for such networks follows.

Scheme 4—A VPPL$^{m \to m}$ for K-Separable Networks:

1) Find a $(K, \frac{2}{3}N)$-separator of G,
2) Construct a BFS spanning tree T_v for every node v of the separator,
3) Construct a VPPL$^{1 \to m}$ with up to $h/2$ VP-hops on each T_v with v as its root,

4) Recursively build a VPPL$^{m\to m}$ with up to h VP-hops in each separated part of G,
5) The resulting VPPL$^{m\to m}$ is a union of the partial VPPL$^{1\to m}$s constructed in earlier steps.

Correctness: The correctness is proven along the lines of the proof for trees. The only difference is that a shortest path between a pair of nodes in separated parts of G, must go through a node v of the separator, hence there is a path with h hops or less in VPPL, that is a shortest path in G (use the VPPL$^{1\to m}$ on T_v). □

Load Analysis: Each link participates in at most K VPPL$^{1\to m}$s at the first stage of the recursion (namely in T_v for every v in the separator). Therefore, the load function of the scheme satisfies

$$\mathcal{L}[\Psi^{m\to m}(N,h)] \le K\cdot\mathcal{L}\left[\Psi^{1\to m}\left(N,\frac{h}{2},v\right)\right] + \mathcal{L}\left[\Psi^{m\to m}\left(\frac{2}{3}N,h\right)\right]$$

or

$$\mathcal{L}[\Psi^{m\to m}(N,h)] \le \frac{Kh}{2(1.5^{2/h}-1)}\cdot N^{2/h}$$

Using similar considerations as for trees. □

B. Meshes

For square mesh networks, we use a different technique which exemplifies the construction scheme for composite networks. Recall that an $\sqrt{N}\times\sqrt{N}$ mesh network of size N is comprised of horizontal and vertical chains of size $\sqrt{N}$. We construct a VPPL$^{m\to m}$ for these networks by merging chain layouts for each row and column.

Scheme 5—A VPPL$^{m\to m}$ for Square Meshes:

1) Build a VPPL$^{m\to m}$ with $h/2$ hops for each horizontal/vertical chain,
2) The union of all these chain layouts is the resulting mesh layout.

Correctness: The hop count is restricted by h, since any switch can be reached by no more than $h/2$ hops (to get to the correct vertical position) and no more than $h/2$ hops in the vertical VPPL. This adds up to no more than h hops. The stretch factor is preserved since one of the shortest routes in a mesh is composed of one horizontal segment and one vertical segment. □

Load Analysis: The links of different row/column components are distinct, and hence the load on a link is determined only by the layout of the component it belongs to. The load of the scheme is thus $\mathcal{L}\Psi \le C(h/2)N^{2/h}$, where $C(h)$ is the h dependent factor of the VPPL$^{m\to m}$ for trees (Scheme 3).□

C. General Topology Networks

For networks with arbitrary topology, it was proven [12] that there probably exists no efficient algorithm which yields an optimal solution (i.e., that the problem is NP-hard). A simple solution for such topologies is given by the foowing sceheme.

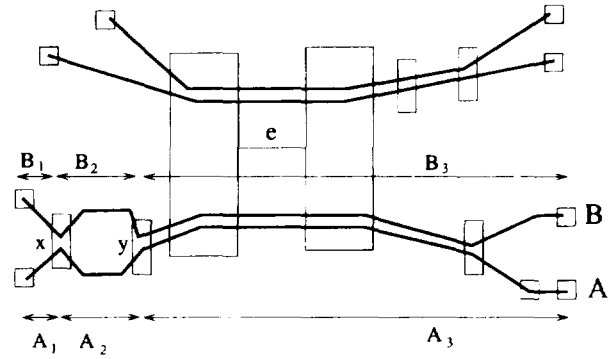

Fig. 6. $\mathcal{S}[e]$ with an induced subgraph which is not a tree.

Scheme 6—Simple Scheme for General Networks:

1) Find a shortest-path spanning tree (produced by a BFS algorithm) from each node to the rest of the network,
2) Construct a separate VPPL$^{1\to m}$ on each tree, with h hops.
3) The resulting VPPL$^{m\to m}$ is a union of the tree VPPL$^{1\to m}$s constructed in earlier steps.

This layout is easily proven to be h-feasible, and has a load of at most $\mathcal{L}(\Psi) \le 4hN^{1+1/h}$. Note that this upper bound is typically much higher than the real bound, since it assumes a worst case, for which all BFS trees share a link. A better approach is based on a greedy scheme, for which we need the following lemma:

Lemma 4: Given a link $e\in E$, a set of pairs of nodes X and a set $\mathcal{S}[e]$ of shortest paths between each pair in X such that all paths in $\mathcal{S}[e]$ include e, there exists a set $\mathcal{S}'[e]$ of shortest paths between the pairs in X that include e as well, such that their induced subgraph[7] is a tree.

Proof: Assuming that the induced subgraph of $\mathcal{S}[e]$ is not a tree, construct a new set of paths $\mathcal{S}'[e]$ whose induced subgraph is a tree, as follows. Add paths from $\mathcal{S}[e]$ to $\mathcal{S}'[e]$ one by one. If a new path to be added, $B\in\mathcal{S}[e]$, creates a cycle in the induced subgraph of $\mathcal{S}'[e]$, then it is modified to eliminate this cycle. To this end, assume without loss of generality, that there is a path $A\in\mathcal{S}'[e]$ which meets and separates from B before reaching e (see Fig. 6). Let x, y be the closest pair of nodes on both paths A and B such that the segment of path B between x and y (B_2 in the figure) is disjoint from the segment A_2 of path A between them. Let $A=(A_1,x,A_2,y,A_3)$ and $B=(B_1,x,B_2,y,B_3)$ where A_i, B_i are segments of A and B.

We first show that $e\notin A_2$ and $e\notin B_2$: If (by contradiction) $e\in A_2$ then if $e\in B_2$, A_2 is not shortest possible (there exist x', y' closer to each other in A and B), and if $e\notin B_2$ then B is not a shortest path—it can be shortened by not including e in B.

So, without loss of generality, $e\in A_3, e\in B_3$. Note that $|A_2|=|B_2|$ since if $|A_2|<|B_2|$, then B is not a shortest path [$B'=(B_1,x,A_2,y,B_3)$ is a shorter path between the same end points]. Therefore, this violation of the tree structure can be fixed by diverting B to use A_2 instead of B_2 [take $B'=(B_1,x,A_2,y,B_3)$ instead of B]. Note that B' may

[7]The union of all links and nodes that participate in these paths.

still cause cycles if added to $\mathcal{S}'[e]$, but such cycles occur in other parts of B' (not between x and y), which may be dealt with using the same method for further modifying B'. Clearly, the paths in $\mathcal{S}'[e]$ are still shortest paths between the same end points, all of which include e. □

A greedy scheme based on the above lemma is the following.

Scheme 7—Greedy Scheme for General Networks:

1) Find a set $\mathcal{S}$ of $\binom{N}{2}$ paths, one shortest path for each pair of nodes in the network.
2) If the maximum number of paths in $\mathcal{S}$ that share a link e is less than some threshold X—define the remaining paths in $\mathcal{S}$ as independent, single hop VPP's (we shall determine X in the sequel).
3) Otherwise, refer to the set $\mathcal{S}[e]$ of paths that includes all paths in $\mathcal{S}$ that share the above link e. If the induced subgraph of $\mathcal{S}[e]$ is not a tree, modify it into a tree by changing each of the paths in $S[e]$ that violates the tree structure into another shortest path between the same pair of nodes (see Lemma 4 for the feasibility proof),
4) Assuming the induced subgraph of $\mathcal{S}[e]$ is a tree, choose an end point of e as a root and construct a VPPL$^{1\to m}$ with $h/2$ hops to it, on the induced subgraph,
5) Return to step 2) after removing $\mathcal{S}[e]$ from S,
6) The resulting VPPL is a union of the partial VPPL's constructed in earlier steps, and the remaining paths in $\mathcal{S}$.

Correctness: The scheme yields a feasible VPPL$^{m\to m}$ since for every pair of nodes, there exists a shortest path in the network, which is either a single hop VPP between the nodes [defined as a VPP in step 2], or is part of a tree for which a VPPL$^{1\to m}$ was constructed in step 4. In this case, it may follow this shortest path, using up to $h/2$ VPP's to the root of the VPPL$^{1\to m}$ and up to $h/2$ hops to the destination. □

Load Analysis: Assume the greedy scheme constructed Y VPPL$^{1\to m}$s. Then the load of the greedy scheme is bounded by the load incurred by these VPPL$^{1\to m}$s, plus at most X, since that was the remaining load when the scheme ended:

$$\mathcal{L}(\Psi) \le Y \cdot \mathcal{L}\left[\Psi^{1\to m}\left(N, \frac{h}{2}\right)\right] + X.$$

Recall that the initial size of $\mathcal{S}$ is less than $N^2/2$, and that each choice of a subset $\mathcal{S}[e]$ in step 3) involves at least X pairs. Therefore, we get $Y \le N^2/2X$. Substituting these in the above load conditions we get

$$\mathcal{L}(\Psi) \le \frac{N^2}{2X} \cdot \frac{h}{2} N^{2/h} + X = \frac{h}{4} N^{2+2/h} \cdot \frac{1}{X} + X.$$

To achieve the minimum $\mathcal{L}(\Psi)$ we choose

$$X = \frac{\sqrt{h}}{2} \cdot N^{1+1/h}$$

Yielding the upper bound

$$\mathcal{L}(\Psi) \le \sqrt{h} \cdot N^{1+1/h}.$$

□

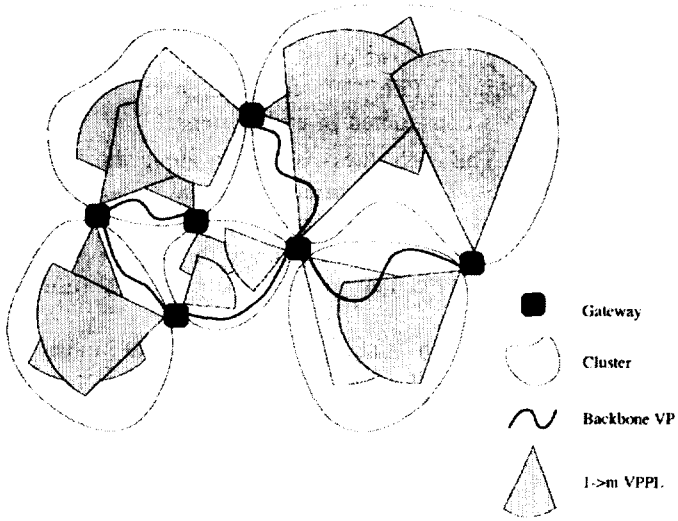

Fig. 7. An example to an *ad-hoc* VPPL$^{m\to m}$.

A better "*ad-hoc*" heuristic approach may be devised for specific real-world network topologies, in which a set of centrally located nodes (such as gateways between subnetworks) naturally separate the network. This "three stage" scheme is based on connecting all the nodes in each subnet to the adjacent gateways by VPPL$^{1\to m}$s, and interconnecting the gateways by a "backbone" VPPL$^{m\to m}$—see Fig. 7 for a pictorial demonstration. In addition to the VPP's that are drawn in the figure, the nodes in each subnet are directly connected by VPP's.

When a VC is established between nodes in remote subnets, it is first routed to a gateway via a VPPL$^{1\to m}$, it then uses the backbone VPPL$^{m\to m}$ to reach the remote subnet, and finally it is routed to the destination via a VPPL$^{1\to m}$ in the remote subnet.

VI. Discussion and Summary

In the paper we studied the problem of the construction of a virtual path layout (VPPL) on a given ATM network. Our results exhibit a trade-off between the hop count h (which effects the call-setup time), and the number of VPP's that use a physical link, $\mathcal{L}(\Psi)$ (which effects the utilization of the VP routing tables and the recovery overhead at link disconnection).

We presented a technique for building a VPPL$^{1\to m}$ on tree networks [which yielded $\mathcal{L}(\Psi) = O(N^{1/h})$], and proved its correctness and optimality. We also demonstrated how the VPPL$^{1\to m}$ problem may be used in solving the VPPL$^{m\to m}$ problem efficiently.

While all these results were presented for tree networks, we showed how to extend them for larger classes of networks, including rings, chordal rings, and meshes. Finally, we discussed how to utilize these techniques for general topologies. These results are summarized in Table I.

A numerical demonstration of efficiency and scalability of the various schemes is illustrated in Fig. 8 (for a reasonably low hop constraint $h = 3$). These schemes are compared with the trivial option of having a direct VPP between any

TABLE I
SUMMARY OF RESULTS

Problem	Graph family	Method	Upper bound	Lower bound
$\text{VPPL}^{1\to m}$	Tree	Recursive Decomposition	$C_1(h)N^{\frac{1}{h}}$	$\frac{C_2(h)}{\Delta}N^{\frac{1}{h}}$
$\text{VPPL}^{m\to m}$	Tree	Recursive Decomposition	$C_3(h)N^{\frac{2}{h}}$	$\frac{C_4(h)}{\Delta}N^{\frac{2}{h}}$
	K-Separable	Recursive Decomposition	$C_5(h,K)N^{\frac{1}{h}}$	—
	$\sqrt{N}\times\sqrt{N}$ Mesh	Union of chain $\text{VPPL}^{m\to m}$s	$C_3\left(\frac{h}{2}\right)N^{\frac{2}{h}}$	—
	General	A $\text{VPPL}^{1\to m}$ for every node	$C_1(h)N^{1+\frac{1}{h}}$	—
		A $\text{VPPL}^{1\to m}$ for the loaded links (greedy)	$C_6(h)N^{1+\frac{1}{h}}$	—

$$C_1(h)=h\,,C_2(h)=\left(\frac{1}{2}\right)^{\frac{1}{h}}\,,C_3(h)=\frac{h}{2(2^{\frac{2}{h}}-1)}\,,C_4(h)=\left(\frac{1}{8h}\right)^{\frac{1}{h}}\,,C_5(h,K)=\frac{Kh}{2(1.5^{\frac{2}{h}}-1)}\,,C_6(h)=\sqrt{h}\,.$$

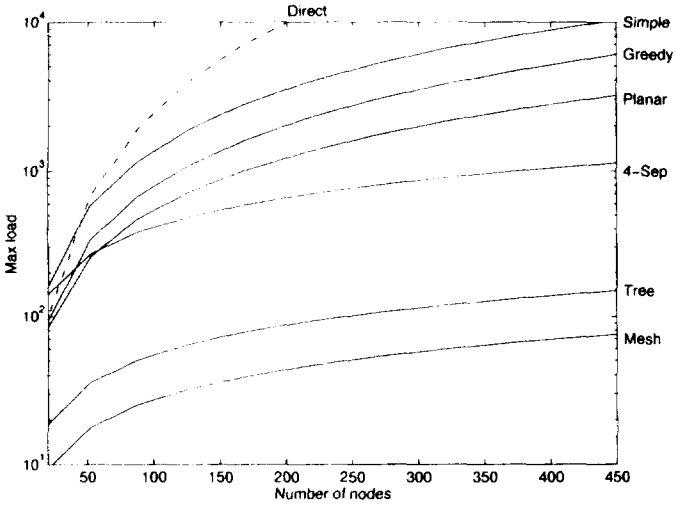

Fig. 8. Upper bounds on the link load for $h = 3$.

pair of nodes (labeled "Direct" in the figure). The dotted line represents the upper bound on the load, as defined by the ATM standard (i.e., 2^{12}). The graph labeled "4-Sep" represents the worst-case for four-separable networks. It is evident from the figure that our techniques for these networks generally yield much more scalable solutions than the trivial solution, even for the "simple" scheme for general networks. Note, however, that for small networks the trivial solution is superior to our solutions for general topology networks. The results for our schemes are even better for higher hop count constraints.

Throughout this paper we assumed that every node has both VP and VC switching capabilities. This assumption may be eliminated by building a $\text{VPPL}^{m\to m}$ as described above, and allocating as such "dual function" nodes, only nodes in which VC switching is done in the VPPL. It is easy to see that if the recursive construction is not carried out for subnetworks which are small enough (in which direct VPP's are used), the number of "dual function" nodes need not be too large.

We believe that the approaches in this paper improve the insight into the VP layout problem, and form a basis for extending the techniques to general topologies, to multiple routes between pairs of nodes, and to many other realistic extensions of the problem.

REFERENCES

[1] S. Ahn, R. P. Tsang, S. R. Tong, and D. H. C. Du, "Virtual path layout design on ATM networks," in *IEEE INFOCOM '94*, 1994, pp. 192–200.
[2] B. Awerbuch, A. Bar-Noy, N. Linial, and D. Peleg, "Compact distributed data structures for adaptive routing," in *21st Symp. Theory Comp.*, 1989, pp. 479–489.
[3] B. Awerbuch and D. Peleg, "Routing with polynomial communication-space trade-off," *SIAM J. Discrete Math.*, vol. 5, no. 2, pp. 151–162, May 1992.
[4] H. L. Bodlaender, "A tourist guide through treewidth," Utrecht University, Dept. of Computer Science, Netherlands, Tech. Rep. RUU-CS-92-12, Mar. 1992.
[5] H. L. Bodlaender, G. Tel, and N. Santoro, "Trade-offs in nonreversing diameter," *Nordic J. Comp.*, vol. 1, pp. 111–134, 1994.
[6] J. Burgin and D. Dorman, "Broadband ISDN resource management: The role of virtual paths," *IEEE Commun. Mag.*, vol. 29, 1991.
[7] R. Cohen and A. Segall, "Connection management and rerouting in ATM networks," in *IEEE INFOCOM'94*, 1994, pp. 184–191.
[8] S. Even, *Graph Algorithms*. Rockville, MD: Computer Science Press, 1979.
[9] The ATM Forum, *ATM User-Network Interface Specification, Ver. 3.0*. Englewood Cliffs: Prentice-Hall, 1993.
[10] G. N. Frederickson and R. Janardan, "Separator-based strategies for efficient message routing," in *27th Symp. Foundations Comp. Sci.*, 1986, pp. 428–437.
[11] "MB86689 address translation controller (ATC)," Fujitsu data sheet, draft ed. 1.0, Oct. 1993.
[12] O. Gerstel, I. Cidon, and S. Zaks, "Optimal virtual path in ATM networks with shared routing table switches," *Chicago J. Theoretical Computer Sci.*, Oct. 1996.
[13] H. Hadama, R. Kawamura, T. Izaki, and I. Tokizawa, "Direct virtual path configuration in large-scale ATM networks," in *IEEE INFOCOM'94*, 1994, pp. 201–207.
[14] R. Händler and M. N. Huber, *Integrated Broadband Networks: An Introduction to ATM-Based Networks*. New York: Addison-Wesley, 1991.
[15] ITU recommendation, I series (B-ISDN), Blue Book, Nov. 1990.
[16] A. Kershenbaum, *Telecommunications Network Design Algorithms*, New York: McGraw-Hill, 1993.
[17] L. Kleinrock and F. Kamoun, "Hierarchical routing for large networks, preformance evaluation and optimization," *Comp. Networks*, vol. 1, pp. 155–174, 1977.
[18] ——, "Optimal clustering structures for hierarchical topological design of large computer networks," *Networks*, vol. 10, pp. 221–248, 1980.
[19] F. Y. S. Lin and K. T. Cheng, "Virtual path assignment and virtual circuit routing in ATM networks," in *IEEE GLOBECOM'93*, 1993, pp. 436–441.
[20] R. J. Lipton and R. E. Tarjan, "A separator theorem for planar graphs," *SIAM J. Appl. Math.*, vol. 36, no. 2, pp. 177–189, Apr. 1979.

[21] P. Lougher and D. Sheperd, "The design of a storage server for continuous media," *Comp. J.*, vol. 36, no. 1, 1993.
[22] D. Peleg and E. Upfal, "A trade-off between space and efficiency for routing tables," in *20th Symp. Theory Comp.*, 1988, pp. 43–52.
[23] S. Ramanathan and P. Venkat Rangan, "Feedback techniques for intra-media continuity and inter-media synchronization in distributed multimedia systems," *Comp. J.*, vol. 36, no. 1, pp. 19–31, 1993.
[24] B. A. Reed, "Finding approximate separators and computing tree width quickly," in *24th Symp. Theory Comp.*, 1992, pp. 221–228.
[25] N. Robertson and P. D. Seymour, "Graph minors {XIII}. The disjoint path problem," Sept. 1986, preprint.
[26] K. I. Sato, S. Ohta, and I. Tokizawa, "Broad-band ATM network architecture based on virtual paths," *IEEE Trans. Commun.*, vol. 38, no. 8, pp. 1212–1222, Aug. 1990.
[27] Y. Sato and K. I. Sato, "Virtual path and link capacity design for ATM networks," *IEEE J. Select. Areas Commun.*, vol. 9, 1991.

Ornan Gerstel (M'95) received the B.A., M.S., and D.S. degrees from the Technion, Haifa, Israel.

He is a Research Staff Member at the IBM T. J. Watson Research Center where he works in the Optical Network Systems Group. His research interests include routing and related network design problems in ATM and WDM networks.

Israel Cidon (M'85–SM'90) received the B.Sc. (summa cum laude) and the D.Sc. degrees from the Technion—Israel Institute of Technology, Haifa, in 1980 and 1984, respectively, both in electrical engineering.

From 1984 to 1985, he was with the faculty of the Electrical Engineering Department at the Technion. In 1985, he joined the IBM T. J. Watson Research Center, NY, where he was a Research Staff Member and the Manager of the Network Architectures and Algorithms Group involved in various broadband networking projects such as the Paris/Planet Gigabit testbeds, the Metaring/Orbit Gigabit LAN and the IBM BroadBand Networking architecture. In 1994 and 1995, he was with Sun Microsystems Labs in Mountains View, CA, as Manager of High-Speed Networking working in various ATM projects including Openet—an open and efficient ATM network control platform. Since 1990, he has been with the Department of Electrical Engineering at the Technion. His research interests are in high-speed wide and local area networks, distributed network algorithms, network performance and mobile networks. He is a founding editor for the IEEE/ACM TRANSACTIONS ON NETWORKING. Previously, he served as the Editor for Network Algorithms for the IEEE TRANSACTIONS ON COMMUNICATIONS and as a Guest Editor for *Algorithmica*.

Dr. Cidon received the IBM Outstanding Innovation Award for his work on the PARIS high-speed network and topology update algorithms respectively, in 1989 and 1993, respectively.

Shmuel Zaks received the B.Sc. and M.Sc. degrees from the Technion, Haifa, Israel, both in mathematics, in 1971 and 1972, respectively, and the Ph.D. degree in computer science from the University of Illinois at Urbana-Champaign, in 1979.

He is with the Department of Computer Science, the Technion, Haifa, Israel. His main research is in the area of distributed computing. Others areas of interest include communication networks (especially ATM-related models), combinatorial and graph algorithms.