Chicago Journal of Theoretical Computer Science

The MIT Press

Volume 1996, Article 3

31 October 1996

ISSN 1073–0486. MIT Press Journals, 55 Hayward St., Cambridge, MA 02142 USA; (617)253-2889; *journals-orders@mit.edu, journals-info@mit.edu*. Published one article at a time in LaTeX source form on the Internet. Pagination varies from copy to copy. For more information and other articles see:

- *http://www-mitpress.mit.edu/jrnls-catalog/chicago.html*
- *http://www.cs.uchicago.edu/publications/cjtcs/*
- *gopher.mit.edu*
- *gopher.cs.uchicago.edu*
- anonymous *ftp* at *mitpress.mit.edu*
- anonymous *ftp* at *cs.uchicago.edu*

The *Chicago Journal of Theoretical Computer Science* is abstracted or indexed in *Research Alert,® SciSearch,® Current Contents®/Engineering Computing & Technology*, and *CompuMath Citation Index.®*

The *Chicago Journal of Theoretical Computer Science* is a peer-reviewed scholarly journal in theoretical computer science. The journal is committed to providing a forum for significant results on theoretical aspects of all topics in computer science.

This article is included in the special issue, *Selected Papers from PODC 1994*, containing articles based on extended abstracts presented at the *13th Annual ACM Symposium on Principles of Distrituted Computing*, Los Angeles, California, August 1994. This special issue was edited by David Peleg.

Optimal Virtual Path Layout in ATM Networks with Shared Routing Table Switches

Ornan Gerstel Israel Cidon Shmuel Zaks

31 October, 1996

Abstract

Abstract-1 In this paper we present a new model for routing that occurs in high-speed ATM networks. Within this model we define a general routing problem, called a virtual path layout. Given a network of nodes (switches) and links, one must find a set of paths in the network, termed the virtual path layout, whereby each pair of nodes may connect using a route that is a concatenation of a small number of virtual paths and is also short in terms of the network links it traverses. Each such layout implies a utilization of the routing tables in the network's nodes. Our goal is to find a layout that minimizes this utilization, assuming that each such node has a central routing table. We prove that this problem is NP-complete, and consequently focus on a simpler problem, in which it is required to connect all nodes to a single switch. Next, we prove that even this problem is NP-complete, and restrict some of the assumptions to yield a practical subproblem, for which we present a polynomial-time greedy algorithm that produces an optimal solution. Finally, we use this restricted problem as a building block in finding a suboptimal solution to the original problem. The results exhibit a tradeoff between the performance of a routing scheme and its resource utilization.

This article contains results that were presented at the *13th Annual ACM Symposium on Principles of Distributed Computing* [GZ94].

1 Introduction

1.1 Background

1.1-1 The advent of fiber optic media has dramatically changed classical views on the role and structure of digital communication networks. Specifically, the sharp distinction (in terms of node hardware architectures, network management techniques, routing schemes, and various theoretical aspects) between telephone networks, cable television networks, and computer networks, has been replaced by a unified approach. In this integrated approach (termed B-ISDN), real-time, high-quality video and audio, and very large amounts of data are integrated into a single digital network that will support numerous applications—ranging from interactive television, video conferencing systems, and video-phone to distributed virtual reality systems.

1.1-2 While fiber optic cables are capable of transferring very large amounts of data (around 10^{10} bits per second), the nodes that connect these cables cannot accommodate this flood of data in software. For this reason, the routing of data packets, which is traditionally done by the network layer software, has been moved to special-purpose very fast hardware, a fact that requires very simple routing algorithms.

1.1-3 The most common solution for this new network architecture is called *Asynchronous Transfer Mode* (ATM for short), and is thoroughly described in the literature [ITU90, HH91, LB92, CS94] [1]. ATM is based on relatively small fixed-size packets called *cells*. Each cell is routed independently, based on two small routing fields at the cell header, called the *virtual channel index* (VCI) and the *virtual path index* (VPI).

1.1-4 At each intermediate node, these fields serve as indices to two routing tables (the VCI serves as an index to one table, and the VPI to the other), and the routing is done in accordance with predetermined information from the appropriate entries (which is entered into the tables during the setup of connections in the network).

1.1-5 Routing in ATM is hierarchical in the sense that the VCI of a cell is ignored as long as its VPI is not null. This algorithm effectively creates two types of predetermined simple routes in the network: routes that are based on VPIs (called *virtual paths* or VPs), and routes that are based on VCIs (called *virtual channels* or VCs). The latter may be viewed as a concatenation of complete VPs.

[1] A different approach to fast routing is based on automatic network routing (ANR) and was implemented in PARIS and plaNET [ACG$^+$90, CG88, CGG$^+$92].

1.1-6 VPs and VCs have different roles in the network. While VCs are used for creating connections between two users of the network that wish to communicate (e.g., a telephone call), VPs have an internal network role (decreasing the overhead caused by network management activities), and are used for bundling several VCs that share part of their route.

1.1-7 The VC and VP concepts have been extensively discussed in the communications literature; however, to the best of our knowledge, the problem of how VPs should be laid out in a given communication network has never been addressed analytically. A few heuristics have been proposed and experimented [ATTD94, HKIT94]. This problem is of practical importance, since the layout of VPs in a network determines many of the network's important performance characteristics, such as the overhead and delay of the setup of a new VC, the load on the VP routing tables, and various fault-tolerance qualities.

1.2 Our Work

1.2-1 All of the above calls for a new model—different from the classical model for routing in computer networks—for both describing the network and measuring the performance of algorithms on it. In Section 2 we define a model for routing in ATM networks, and determine the essential characteristics for measuring the optimality of a given layout. Using this model, we define the general layout problem for ATM. We then isolate a simpler case, which can be viewed as a variant of tree routing (analogous to the way that interval routing [SK85] is used in classical routing problems), suited for this model.

1.2-2 In Section 3 we prove that even the simpler case is NP-complete, and we therefore restrict ourselves to a subproblem that is general enough for practical purposes, and present a greedy algorithm that optimally solves the subproblem for any given tree network. In Section 4 we prove the general problem to be NP-complete as well, and demonstrate the use of the subproblem as a building block in the solution of the general problem on arbitrary networks. We conclude in Section 5 by summing up the results in the paper, and presenting open problems.

1.2-3 We expect our contribution to be three-fold:

1. This paper is among the first to analytically address the problem (together with [CGZ94]). For this reason, we focus on a relatively simple problem, with a low number of parameters. This is also a first attempt

to prove complexity results concerning the problem, and to present an optimal solution for a special case (the one-to-many layout problem on trees—see Definition 5).

2. We believe the problem we focus on is of practical importance in certain scenarios in ATM networks. In particular, our problem is most suitable for an environment with short-lived VCs without an a priori known bandwidth; such environments are expected to be typical for distributed applications over ATM networks [LPS91]. Even our restricted (polynomially solvable) problem seems to be useful for client-server applications over ATM.

3. We present basic techniques for deriving upper and lower bounds on such problems (exemplified by the tree-routing problem, often used as a benchmark case study). In particular, the crux of our greedy algorithm (i.e., the upper bound) is nonintuitive in nature, and was formulated by the needs of its optimality proof (lower bound).

1.2-4 The present paper is closely related to [CGZ94], in which a similar model is discussed, using a different VP-table load metric (on the edges of the graph rather than on its nodes). This difference stems from different node implementations, and seems to have an impact on the complexity of the problem. In particular, we present here NP-completeness results that we were unable to derive for the other load metric.

1.2-5 The simpler one-to-many problem on trees is addressed in [CGZ94] as well, using a different technique that is based on dividing the tree into small subtrees and solving the problem separately in each subtree. This technique yields a solution that is asymptotically optimal under certain assumptions. In contrast, the "greedy" technique presented herein provides an optimal solution for any given tree. Also, the resulting algorithm is much simpler to implement, and has lower time complexity. Furthermore, while many-to-many solutions in [CGZ94] yield high VP-table loads, the solution herein results in much smaller loads, by compromising the efficient utilization of the physical network (using the concept of a stretch factor).

1.2-6 A related problem is that of keeping small routing tables for routing in conventional computer networks. This problem has been widely studied [ABLP89, AP92, FJ86, FJ88, KK77, KK80, PU88] and has 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. The main difference stems from the fact that in our case there is no flexibility as to the routing scheme itself, since this is determined by the ATM standard [ITU90]. We therefore present a static structure in the network, while in the conventional model, the exact routing of a packet may be determined dynamically during the process of its routing (as in [AP92, ABLP89]), or by information that is based on the name of the destination (as in [FJ86, FJ88]).

1.2-7 Other differences stem from practical restrictions on the model, in which the routing tables of each node are restricted by the same size, while in some of the conventional solutions the total (or average) size of the routing tables is minimized (e.g., [FJ86, FJ88, PU88]).

1.2-8 Our approach resembles some of the above-mentioned techniques in that a simpler tree-routing problem is used as a building block in the general-case solution (analogous to the way that interval routing on trees [SK85] is used in some of the routing schemes for arbitrary graphs). We use the techniques of [AP92] for demonstrating the usage of our tree-routing problem in solving the general problem.

1.2-9 This paper is an extension of [GZ94].

2 The Model

2-1 We first define a graph-theoretic model for our problems (for basic terms and definitions, see [Eve79]). In our model, we have an underlying communication network that consists of nodes and links between nodes. 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.

2-2 Due to the routing mechanism in ATM, VPs are unidirectional paths in the network. However, for connection management reasons [CS94], VPs are coupled together in pairs of parallel routes in opposite directions, effectively creating bidirectional paths. For simplicity, we refer in the remainder of the paper to these bidirectional VPs exclusively.

2-3 A virtual path layout is a set of simple bidirectional paths on the given network, as formalized by the following definition:

Definition 1 *Let $\mathcal{P}(G)$ be the set of all simple paths in G. A* virtual path layout *or VPL is a subset of $\mathcal{P}(G)$. Formally, it is convenient to represent a VPL Ψ by a pair $\Psi = (G_\Psi, \mathcal{I})$, where $G_\Psi = (V, E_\Psi)$ is a "virtual" graph and*

$\mathcal{I}\colon E_\Psi \to \mathcal{P}(G)$ is a "mapping" function. A "virtual" edge $\psi = (a, b) \in E_\Psi$ represents a VP 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.*

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.

2-4 We extend the definition of $\mathcal{I}$ to simple paths in G_Ψ as follows:

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

2-5 Each VP utilizes an entry[2] in the routing table of each node that is part of its path (be it an endpoint or an intermediate node). Since the routing tables are bounded by the ATM standard to 4,096 entries, this resource should be carefully allocated in large networks. The following load measure expresses this utilization, assuming that each node has a central, shared routing table:[3]

Definition 3 *The* load $\mathcal{L}(v)$ *on a node $v \in V$ is the number of VPs $\psi \in E_\Psi$ that include v in their induced paths, namely, $\mathcal{L}(v) = |\{\, \psi \in E_\Psi \mid v \in \mathcal{I}(\psi) \,\}|$. The load definition is extended to a VPL Ψ by $\mathcal{L}(\Psi) = \max_{v \in V} \mathcal{L}(v)$.*

Definition 4 *Let $\mu \in \mathbb{R}$ be a real number called the* stretch factor. *The* hop count $\mathcal{H}_\mu(v, w)$ *for a pair of nodes $v, w \in V$ is the minimum k such that:*

1. *$\exists p = (\Psi_1, \Psi_2, \ldots, \Psi_k) \in \mathcal{P}(G_\Psi)$, ($\Psi_i \in E_\Psi$ for every i),*
2. *$\exists x, y \in V, \psi_1 = (v, x), \psi_k = (y, w)$, and*
3. *the induced path $\mathcal{I}(p)$ is at most μ times longer than the shortest path between v and w in G.*

If no such path exists, define $\mathcal{H}_\mu(v, w) = \infty$ (this does not necessarily mean that G_Ψ is not connected).

[2]A VP actually uses two entries in each such routing table, for the two unidirectional VPs from which it is composed, but we neglect this fact, as it exactly doubles the utilization at any table.

[3]This alternative is applicable mainly for nodes that are based on shared memory. Other node architectures are based on a separate processor for each link in the node, which dictates a separate routing table for each link, and a different load measure.

2-6 We distinguish between two problems:

- the general layout problem in which it is required to connect every pair of nodes in V—we term this access pattern "many-to-many" (m-m for short), and
- a more restricted access pattern in which it is required to connect all nodes to a single node (called the *root*)—we term this access pattern "one-to-many" (1-m).

The VPL for m-m access is denoted by $\mathrm{VPL}^{\text{m-m}}$, while a VPL for 1-m access is denoted by $\mathrm{VPL}^{\text{1-m}}$. Note that $\mathrm{VPL}^{\text{1-m}}$ differs from broadcasting in that the data is not distributed from the root to all other nodes in the network, but rather, it enables separate connections (VCs) from the root to all the other nodes.

2-7 The two problems are formalized by the following definition.

Definition 5 *Let $h > 0$; $\mu \geq 1$; let $G = (V, E)$ be a given network; and let $r \in V$. The following two cases correspond to the above-mentioned problems.*

- *A $VPL^{m\text{-}m}$ in G is* h-feasible *if* $\max_{v,w \in V} \mathcal{H}_\mu(v, w) \leq h$.
- *A $VPL^{1\text{-}m}$ in G is* (h, r)-feasible *if* $\max_{v \in V} \mathcal{H}_\mu(v, r) \leq h$.

2-8 The feasibility of a VPL captures the notion of a network that has good worst-case performance (since h determines the maximum number of VPs that are needed for the compositions of any VC, and this number is proportional to the time that the creation of that VC takes). We now define an optimal solution as a solution that does not require much of the network resources, while maintaining this good performance (the load on a node is proportional to the required capacity of the VP routing tables, as well as other resources).

Definition 6 *Let $h > 0$ and $\mu \geq 1$. A $VPL^{m\text{-}m}$ Ψ is* h-optimal *if it is h-feasible and its load $\mathcal{L}(\Psi)$ is minimal among all other h-feasible VPLs. This definition is extended in a straightforward manner to the (h, r)-optimality of a $VPL^{1\text{-}m}$ with a root r.*

Example 1-1 **Example 1** Consider the ATM network with VPs in Figure 1, in which there are three VCs, one between user $x1$ and user $x2$ (we denote it by $VC(x1, x2)$), one between $y1$ and $y2$, and one between $z1$ and $z2$. All VCs are composed

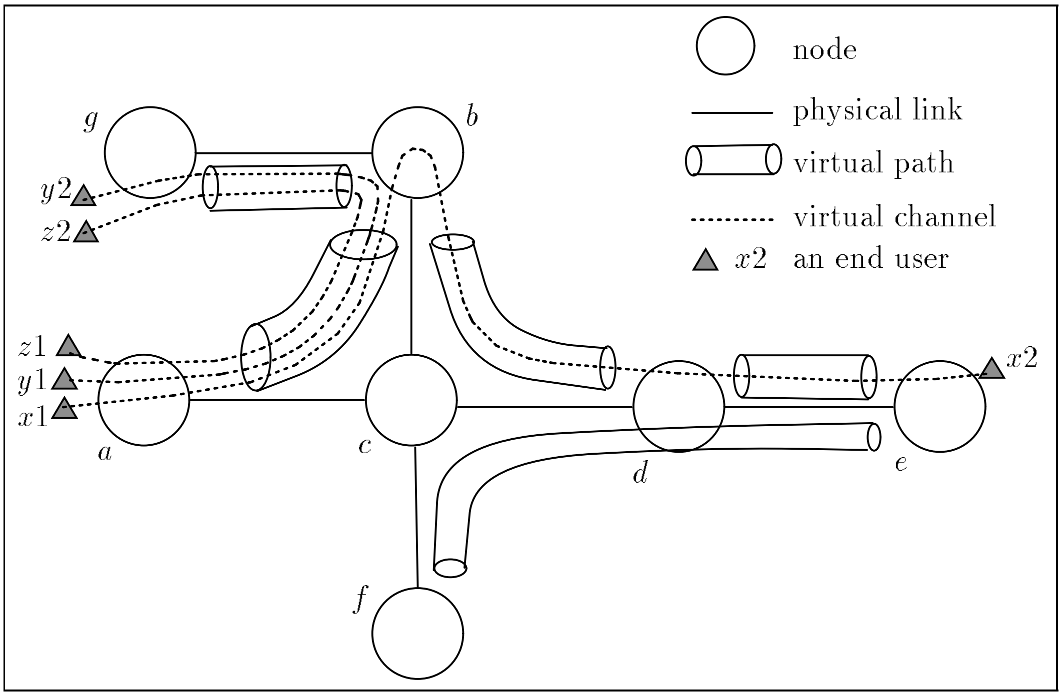

Figure 1: An example for VP/VC layout

of a concatenation of VPs (e.g., $VC(y1, y2)$ is composed of $VP(a, b)$ and $VP(b, g)$).

Example 1-2 Since the routing of cells from a to b is done on $VP(a, b)$, there is no need for a separate entry in the VC routing table of c for each such VC, and this remains true even if there are many VCs that use $VP(a, b)$ as part of their route. Only at b there is an entry for each of the VCs, since the $VP(a, b)$ ends there, to determine if the VC continues into $VP(b, d)$ (for $VP(x1, x2)$) or into $VP(b, g)$ (for $VC(y1, y2)$ and $VC(z1, z2)$).

Example 1-3 Since each VC is composed of complete VPs, there is no way to create a VC between a and c, despite the fact that the physical network itself is connected (hence $\mathcal{H}_\mu(a, c) = \infty$).

Example 1-4 Note that if $\mu < \frac{5}{3}$, then $\mathcal{H}_\mu(a, e) = \infty$, since the shortest path $d_G(a, e) = 3$ and the path induced by the VPs $(a, b), (b, d), (d, e) \in E_\Psi$ is of length 5 (since $p = ((a, b), (b, d), (d, e))$ and $\mathcal{I}(p) = \mathcal{I}((a, b)), \mathcal{I}((b, d)), \mathcal{I}((d, e)) = (a, c, b, c, d, e)$); If, however, $\mu > \frac{5}{3}$, then $\mathcal{H}_\mu(a, e) = 3$. As to the load measure, $\mathcal{L}(c) = |\{(a, b), (b, d), (e, f)\}| = 3$ and $\mathcal{L}(d) = |\{(b, d), (e, d), (e, f)\}| = 3$.

3 Tools

3-1 Our formulation of the optimal VPL$^{\text{1-m}}$ algorithm and its optimality proof are based on some vector notations, definitions, and properties, which are presented in this section.

Notation 1 *Let $V, W \in \mathbb{N}^h$, $c \in \mathbb{N}$, and $i, j \in \{1, \ldots, h\}$. We use the following notations:*

- $V[i]$*—the ith element in the vector* V*,*
- $\|V\|$*—the sum of the elements* $V[i]$ *(since all elements are positive, this is the* $\mathcal{L}_1$ *norm of the vector, often written* $\|V\|_1$*),*
- $V[i : j]$*—the part of the vector from the ith to the jth position (inclusive),*
- $\vec{c}[1 : i]$*—the vector* $(c, \ldots, c)$ *of* i *elements that are equal to* c*,*
- $V \bullet W$*—the concatenation of the vectors* V *and* W*,*
- $V + W$*—the vector sum of* V *and* W*, and*

- $V < W$—*the lexicographic comparison between vectors, using an order in which* $V[1]$ *is the least-significant component, e.g.,* $(9, 11, 1) < (2, 12, 1) < (0, 0, 2)$. *The same applies for* $>$, $\leq$, *and* $\geq$.

3-2 We shall need the following properties of vectors in $\mathbb{N}^h$.

Lemma 1 *Let* L, M *be vectors in* $\mathbb{N}^h$ *for some* $h > 1$, *such that* $\|L[1\colon h-1]\| \leq 1$ *and* $M \geq L$. *Then* $\|M\| \geq \|L\|$.

Proof of Lemma 1 If $M \geq L$, then one of the following holds:

Case 1 $M[1\colon h-1] \geq L[1\colon h-1]$ and $M[h] = L[h]$**:** Clearly

$$\|M[1\colon h-1]\| \geq \|L[1\colon h-1]\|$$

hence

$$\|M\| = \|M[1\colon h-1]\| + M[h] \geq \|L[1\colon h-1]\| + M[h] = \|L\|$$

Case 2 $M[h] > L[h]$**:** Then

$$\|M\| \geq M[h] \geq L[h] + 1 \geq L[h] + \|L[1\colon h-1]\| = \|L\|$$

Proof of Lemma 1 □

3-3

Definition 7 *A vector* L *is called* k-nontrivial *if* $\|L[1\colon k-1]\| \leq 1$ *and* $\|L[1\colon k]\| > 1$. *In particular, if* $L[1] > 1$ *it is* 1*-nontrivial, and if* $\|L\| \leq 1$, *then* L *is* ∞*-nontrivial. A vector is* $k^{(>)}$*-nontrivial if it is* x*-nontrivial for some* $x > k$ *(similar definitions apply for* $<$, $\leq$, *and* $\geq$*).*

Definition 8 *Let* $\mathcal{S}$ *be a sequence of vector pairs in* $\mathbb{N}^h$, $\mathcal{S} = ((L_i, M_i))_{i=1}^{s}$. $\mathcal{S}$ *is* proper *if the following properties hold:*

P1: $L_i \leq M_i$ *for every* $i \in \{1, \ldots, s\}$, *and*

P2: if $i > j$, *then there exists* $k > 0$ *such that* $L_j[1\colon k] \geq L_i[1\colon k]$, L_j *is* k*-nontrivial, and* L_i *is* $k^{(\geq)}$*-nontrivial.*

Notation 2 *We shall need the following additional vector notations, where L is an arbitrary vector of length h, $\mathcal{S} = ((L_i, M_i))_{i=1}^{s}$.*

- *The transformation $L^{trans(j)}$ on a vector L is defined by: $L^{trans(0)} = L$, and $L^{trans(j)} = \vec{0}[1\colon j] \bullet (L[j+1]+1) \bullet L[j+2\colon h]$ for $j > 0$.*
- *$\mathcal{S}_L = (L_i)_{i=1}^{s}$, and $\mathcal{S}_R = (M_i)_{i=1}^{s}$.*
- *$\Sigma((L_i)_{i=1}^{s}) = \sum_{i=1}^{s} L_i$.*
- *Let $P = (p_1, p_2, \ldots, p_s)$ where $0 \leq p_i \leq h-1$. Then $\mathcal{S}^{trans(P)} = (L_i^{trans(p_i)})_{i=1}^{s}$.*

3-4 Consider a proper sequence $\mathcal{S} = ((L_i, M_i))_{i=1}^{s}$. Clearly $\Sigma(\mathcal{S}_L) \leq \Sigma(\mathcal{S}_R)$. Note that the transformation $L_i^{\mathrm{trans}(j)}$ of a vector in $\mathcal{S}$ increases L_i lexicographically. So, the sum of the L_i vectors, after transforming $L_i^{\mathrm{trans}(j)}$, may become lexicographically larger than the sum of the M_is. That is, in some cases $\Sigma(S_L^{\mathrm{trans}((0,\ldots,0,j,0,\ldots,0))}) > \Sigma(\mathcal{S}_R)$, even though $\mathcal{S}$ is proper. To solve this problem, we first show that if the total sum of values in the M_i vectors ($\|\Sigma(\mathcal{S}_R)\|$) is less than that of the L_i vectors, *one* of the L_is can indeed be transformed without violating the overall lexicographic order, that is: $\Sigma(\mathcal{S}_L^{\mathrm{trans}((0,\ldots,0,j,0,\ldots,0))}) \leq \Sigma(\mathcal{S}_R)$.

Lemma 2 *Let $\mathcal{S} = (L_i, M_i)_{i=1}^{s}$ be a proper sequence, and let $\|\Sigma(\mathcal{S}_R)\| < \|\Sigma(\mathcal{S}_L)\|$. Then there exist integers $0 < p \leq h-1$ and $0 < k \leq s$ such that $\Sigma(\mathcal{S}_L) \leq \Sigma(\mathcal{S}_R)$, where $P = 0[k-1] \bullet p \bullet \vec{0}[1\colon s-k]$.*

Proof of Lemma 2-1 **Proof of Lemma 2** Let L_k be the first vector in $\mathcal{S}_L$ that is p-nontrivial for $p < \infty$. Such a choice of k exists, since otherwise $\|\Sigma(\mathcal{S}_L)\| \leq \|\Sigma(\mathcal{S}_R)\|$ by Lemma 1.

Proof of Lemma 2-2 Clearly $\|L_k^{\mathrm{trans}(p)}\| < \|L_k\|$. If $L_k[p+1\colon h] < M_k[p+1\colon h]$, then $L_k^{\mathrm{trans}(p)} \leq M_k$ and $\Sigma(\mathcal{S}_L) \leq \Sigma(\mathcal{S}_R)$, as required by the lemma. Otherwise, if $L_k[p+1\colon h] = M_k[p+1\colon h]$ (and therefore $L_k^{\mathrm{trans}(p)} > M_k$), then two cases are possible:

Case 1 For every $j > k$, $L_j[p+1\colon h] = M_j[p+1\colon h]$:
Then $\|L_j\| \leq \|M_j\|$ since L_j is $p^{(\geq)}$-nontrivial, and hence $L[1\colon p], M[1\colon p]$ satisfy the conditions of Lemma 1. But, since $\|L_i\| \leq \|M_i\|$ for every $i < k$, we have $\|\Sigma(\mathcal{S}_L)\| \leq \|\Sigma(\mathcal{S}_R)\|$, contradicting the conditions of the lemma.

$$\begin{pmatrix} L_1 & \leq & M_1 \\ & \vdots & \\ L_k & \leq & M_k \\ & \vdots & \\ L_j & < & M_j \\ & \vdots & \\ L_s & \leq & M_s \end{pmatrix} \longrightarrow \begin{pmatrix} L_1 & \leq & M_1 \\ & \vdots & \\ L_k^{\text{trans}(p)} & ? & M_k \\ & \vdots & \\ L_j & < & M_j \\ & \vdots & \\ L_s & \leq & M_s \end{pmatrix} \longrightarrow \begin{pmatrix} L_1 & \leq & M_1 \\ & \vdots & \\ L_k^{\text{trans}(p)} & \leq & M_k^{\text{dir}} \\ & \vdots & \\ L_j & \leq & M_j^{\text{indir}} \\ & \vdots & \\ L_s & \leq & M_s \end{pmatrix}$$

(a) (b) (c)

Figure 2: Transformations in Case 2 of Lemma 2

Case 2 There exists $j > k$ such that $L_j[p+1\colon h] < M_j[p+1\colon h]$**:**
We transform the M_k and M_j vectors so as to keep the lexicographic order in the kth and jth pairs, without changing the sum $\Sigma(\mathcal{S}_R)$, thereby proving the lemma (refer to Figure 2). The transformations are as follows.

$$M_k^{\text{dir}} = M_j[1\colon p] \bullet (M_k[p+1]+1) \bullet M_k[p+2\colon h]$$

$$M_j^{\text{indir}} = M_k[1\colon p] \bullet (M_j[p+1]-1) \bullet M_j[p+2\colon h]$$

Clearly $M_k + M_j = M_k^{\text{dir}} + M_j^{\text{indir}}$ and therefore $\sum_{j=1}^{s} M_j = \Sigma(\mathcal{S}_R)$. It is also clear that $L_k^{\text{trans}(p)} \leq M_k^{\text{dir}}$. The lexicographic order still holds for the jth pair ($L_j \leq M_j^{\text{indir}}$), because $L_j[p+1\colon h] \leq M_j^{\text{indir}}[p+1\colon h]$ and $L_j[1\colon p] \leq L_k[1\colon p] \leq M_k[1\colon p] = M_j^{\text{indir}}[1\colon p]$.

Proof of Lemma 2 □

3-5 We now use Lemma 2 repeatedly to show that it is possible to decrease the total sum of values in all the L vectors ($\|\Sigma(\mathcal{S}_L)\|$) without increasing their total lexicographic order ($\Sigma(\mathcal{S}_L)$) too much.

Lemma 3 *Let $\mathcal{S} = ((L_i, M_i))_{i=1}^{s}$ be a proper sequence. Then there exists a vector of integers $P = (p_1, p_2, \ldots, p_s)$ such that the following conditions hold:*

1. $0 \leq p_i \leq h-1$ *for every* i,
2. $\Sigma(\mathcal{S}_L^{trans(P)}) \leq \Sigma(\mathcal{S}_R)$, *and*
3. $\|\Sigma(\mathcal{S}_L^{trans(P)})\| \leq \|\Sigma(\mathcal{S}_R)\|$.

Proof of Lemma 3-1 **Proof of Lemma 3** Starting with $P = (0,\ldots,0)$, we gradually increase the coordinates of P until Condition 3 holds, while preserving Condition 2. Note that for the initial $P = (0,\ldots,0)$, Condition 2 holds: $\Sigma(\mathcal{S}_L^{\mathbf{trans}(P)}) = \Sigma(\mathcal{S}_L) \leq \Sigma(\mathcal{S}_R)$ (the left equality holds by definition of $L_i^{\mathbf{trans}(0)}$, and the right inequality is due to Property P1 for each pair in $\mathcal{S}$). The transformations implied by the changes in P may, however, violate P1 (i.e., $L_i^{\mathbf{trans}(p_i)}$ may become lexicographically greater than M_i). In order to avoid such violations (and thereby preserve Condition 2), we use Lemma 2.

Proof of Lemma 3-2 In this process, we have advanced toward satisfying Condition 3, since now $\|L_i^{\mathbf{trans}(p_i)}\| \leq \|M_i\|$, while this did not necessarily hold for $\|L_i\|$ and $\|M_i\|$. Before the next change in P, $\mathcal{S}$ must be rearranged to remain proper. To this end, the vectors in $\mathcal{S}$ need to be transformed as in Lemma 2 to maintain Property P1. In addition, they may need to be reordered so as to keep Property P2. However, these changes do not alter $\Sigma(\mathcal{S}_R)$.

Proof of Lemma 3-3 Since $\|\Sigma(\mathcal{S}_L)\|$ decreases each time Lemma 2 is applied, after a finite number of applications of Lemma 2, it will become smaller than $\|\Sigma(\mathcal{S}_R)\|$.

Proof of Lemma 3 □

4 The VPL$^{\text{1-m}}$ Problem

4-1 In this section we discuss the VPL$^{\text{1-m}}$ problem on tree networks. Our main result is an optimal polynomial algorithm for this case and its proof of optimality, based on tools from the previous section. Before going into the details, we mention that this problem is hard for general topologies, assuming an unbounded stretch factor (i.e., $\mu = \infty$). In other words, given the following decision problem:

PROBLEM: The Optimal VPL$^{\text{1-m}}$ problem (OPT-VPL$^{\text{1-m}}$).

INSTANCE: An undirected graph $G = (V, E)$, a node $r \in V$, integers $L, h > 0$.

QUESTION: Assuming $\mu = \infty$, does there exist Ψ, an (h, r)-feasible VPL$^{\text{1-m}}$ on G such that $\mathcal{L}(\Psi) \leq L$?

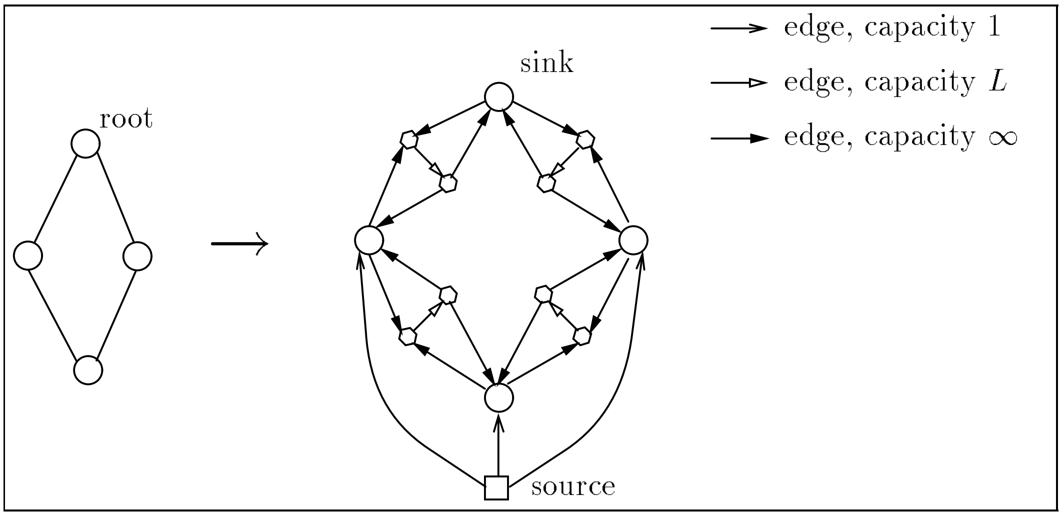

Figure 3: Transforming the VPL$^{\text{1-m}}$ problem into a flow problem

4-2 We prove that OPT-VPL$^{\text{1-m}}$ is NP-complete. The details of the proof may be found in Appendix A.1.

Observation 1 *It is interesting to note that the variant of OPT-VPL$^{1\text{-}m}$ problem with fixed $h = 1$ can be solved polynomially. This is done by the transformation depicted in Figure 3, which transforms the (undirected) graph of the VPL$^{1\text{-}m}$ problem into a (directed) flow problem. Each undirected edge is replaced by five edges and two new nodes, where the edge between these new nodes is restricted to a capacity of L, and the other edges do not have capacity restrictions. A new source node is added and connected to all the original nodes except the root r by edges of capacity 1. The original root node is defined as the sink.*

4-3 We now present a polynomial-time optimal solution for VPL$^{\text{1-m}}$, assuming $\mu = 1$ and a tree network. These restrictions still allow a variety of applications for the problem. In particular, this restricted VPL$^{\text{1-m}}$ problem is used as a building block for solving the VPL$^{\text{m-m}}$ problem. In addition, it has practical applications of its own, in the field of client-server applications over ATM [LPS91]. The problem also serves as a sufficiently complex case to allow for a presentation of nontrivial upper and lower bound results, using techniques which we expect to be basic in similar future studies.

4.1 The Algorithm

4.1-1 We devise a greedy algorithm that produces an optimal $\mathrm{VPL}^{\text{1-m}}$ for any tree. This algorithm is an iterative process in which the minimal load is guessed by the *main* procedure, which calls the *find_layout* procedure with the guessed load as a parameter. The *find_layout* procedure tries to construct a layout under the given load constraint, and informs the *main* if the guess enabled the construction of the layout. Using this information, *main* increases or decreases its guess, until it achieves a minimal guess for which *find_layout* succeeds. Let *vectarray* represent the type **array**$[V(T)]$ **of** $\mathbb{N}^h$.

procedure $main(h\colon \mathbb{N},\ T\colon \text{tree},\ root\colon V(T))$
   var $L1, L2\colon vectarray$
   find $\mathcal{L}_{\max} \in \{1, \ldots, |V(T)|\}$ **such that**
      $\mathit{find\_layout}(h, T, root, \mathcal{L}_{\max}, L1) = \text{SUCCESS}$ **and**
      $\mathit{find\_layout}(h, T, root, \mathcal{L}_{\max} - 1, L2) = \text{FAILURE}$
   $\mathit{construct\_layout}(root, L1)$
end procedure *main*

4.1-2 In the *find_layout* procedure, we view the tree T as rooted at *root*. The procedure starts from the leaves of the tree and advances toward the root. For each node v it maintains the number of VPs that go through v (and contribute to its load). This number is kept in a vector $L_v \in \mathbb{N}^h$, where $L_v[1]$ holds the number of VPs that serve only as a first hop from some descendant, and $L_v[i]$ holds the number of VPs that are an ith hop[4] for some descendant x, but not an $(i+1)$th hop for any descendant.

4.1-3 Specifically, *find_layout* creates a first-hop VP for each leaf (at line 4). At an internal node v, the vector L_v, of VPs is equal to the vector sum of all the VPs that arrive at it from all its sons, plus one additional first-hop VP from v toward the root (at line 14). If this vector is too big (i.e., $\|L_v\|$ exceeds the current load constraint at line 15), then the VPs that go through v are reduced by stopping some of them at the sons of v (hence turning the VP that starts at that son from a first-hop VP to an $(i+1)$th hop VP), by applying the transformation in Figure 4 (at lines 16–21).

[4]The shortest path, in terms of VPs, from x toward the root includes the VP as the ith VP from it.

function *find_layout*$(h\colon \mathbb{N},\ root\colon V(T),\ \mathcal{L}_{\max}\colon \mathbb{N},$ **var** $L\colon vectarray)$
  returns $\{\textsc{success}, \textsc{failure}\}$
  (L_v indicates the entry of L indexed by v)
  for all $w \in V(T)$
    if w is a leaf **then** $L_w \leftarrow (1, 0, \ldots, 0)$
      else $L_w \leftarrow$ *UNDEFINED*
    end if
  end for
  loop *forever outer*
    if $v = root$ **then return** SUCCESS **end if**
    find $v \in V(T)$ **such that**
      $L_v =$ *UNDEFINED* **and**
      for all $x \in Sons(v)$, $L_x \neq$ *UNDEFINED*
    loop *forever inner*
      $L_v \leftarrow \sum_{s \in Sons(v)} L_s + (1, 0, \ldots, 0)$
      if $\|L_v\| \leq \mathcal{L}_{\max}$ **then exit loop** *inner* **end if**
      find $s \in Sons(v)$, $i \in \{1, \ldots, h-1\}$ **such that**
        $\|L_s[1\colon i]\| > 1$ **and**
        for all $s' \in Sons(v) \setminus \{s\}$, $L_s[1\colon i] \geq L_{s'}[1\colon i]$ (lexicographic)
        for all $s' \in Sons(v)$, $\|L_{s'}[1\colon i-1]\| \leq 1$
      if no such i, s exist **then return** FAILURE **end if**
      $L_s \leftarrow \vec{0}[1\colon i] \bullet (L_s[i+1] + 1) \bullet L_s[i+2\colon h]$
        (transform L_s to $L_s^{\mathrm{trans}(i+1)}$)
    end loop *inner*
  end loop *outer*
end function *find_layout*

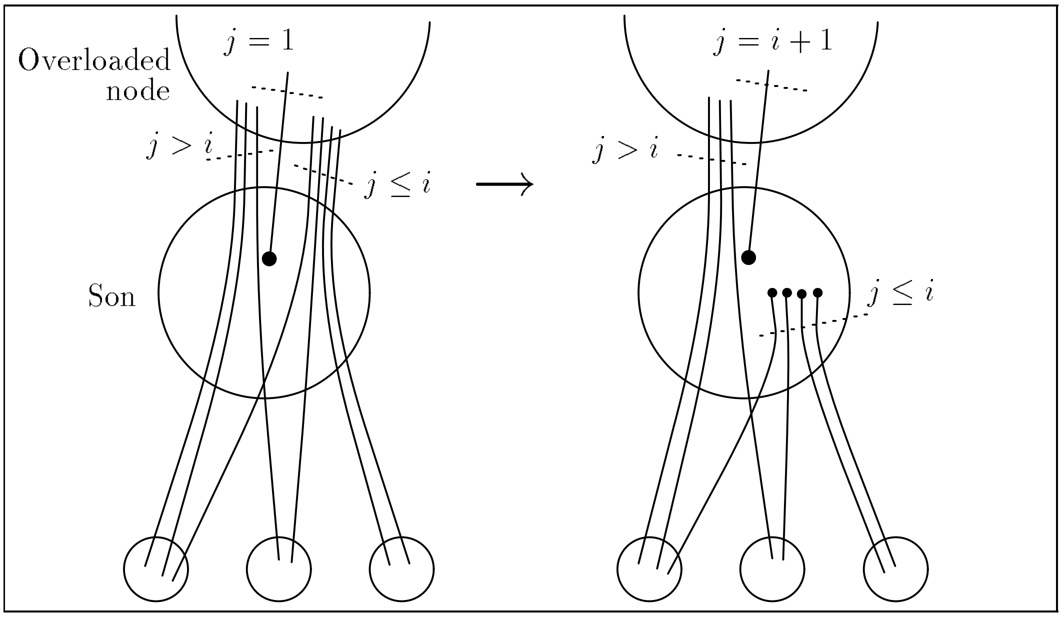

Figure 4: The transformation on an overloaded node v: $L_v \rightarrow L_v^{\mathbf{trans}(i+1)}$

4.1-4 A partial intuition behind this transformation is that it is better to stop VPs that serve as a jth hop for the lowest possible j, as this does not increase the vector L_v too much in its lexicographic order. A part of this transformation is, however, nonintuitive, and stems directly from needs of the proof of optimality.

4.1-5 The *main* procedure relies on *construct_layout* for the actual creation of the VPL$^{\text{1-m}}$ from the set of vectors in $\bar{L}$. This procedure is activated for a node v and recursively creates VPs for all the nodes in the subtree rooted at v. The code of *construct_layout* is straightforward, yet lengthy and involved, and is omitted in this paper. For similar reasons, we also omit the proof of correctness of the algorithm. Similar code and proofs may be found in [Ger95].

4.2 Proof of Optimality

4.2-1 We prove the solution that *find_layout* produces is optimal for every given tree and h, thereby implicitly proving a lower bound for the VPL$^{\text{1-m}}$ problem. The main lemma is Lemma 6, in which the solution $\{\, L_v \mid v \in V \,\}$

of *find_layout* is compared to an arbitrary minimal solution with a vector representation $\{ M_v \mid v \in V \}$. To this end, we first prove (in Lemma 5) that every minimal solution indeed has a vector representation with some desired properties. In this representation, $\|W_v\|$ is the load on node v. Using the tools in Section 3, in particular Lemma 3, we prove (by induction) that at every node v, $L_v \leq M_v$ and $\|L_v\| \leq \|M_v\|$, thereby proving that the solution of *find_layout* has minimal load (and is thus optimal). The simple case for the proof is when v is not overloaded, in which case $L_v = \sum_{s \in Sons(v)} L_s + (1, 0, \ldots, 0)$, and the inductive claim easily follows, assuming it holds for each son of v. For an overloaded node, the load vectors of the sons are arranged in the order that they are chosen by *find_layout* at lines 15–16. This order is reflected by Property P2 of a proper sequence (see Definition 8). Then, they are lexicographically compared to the respective M vectors.

4.2-2 Thus, before applying transformations on any son of v, we have $L_i \leq M_i$, $i = 1, \ldots, s$. Using Lemma 3, we show that after several transformations the total load at v, $\|\Sigma(\mathcal{S}_L)\|$, decreases at least to the level of the given optimal solution ($\|\Sigma(\mathcal{S}_R)\|$), while maintaining its property of being lexicographically minimal.

4.2-3 The above sketch is formalized as follows.

Definition 9 *Let $\hbar(\psi)$ denote the maximal hop count for which ψ serves from some node to the root (this is the value $j = i + 1$ assigned to the VP starting at the son in Figure 4.*

Definition 10 *Given $h > 0$, a VPL$^{1\text{-}m}$ is (h, r)-*minimal *if it is (h, r)-feasible and the deletion of a VP $\psi \in E_\Psi$ yields a VPL$^{1\text{-}m}$ that is not (h, r)-feasible.*

Lemma 4 *In a minimal VPL$^{1\text{-}m}$, there exists a unique VP ψ_v for every node v that starts at v toward the root.*

Proof of Lemma 4 Assuming two such VPs ψ_1 and ψ_2 exist in a minimal VPL$^{\text{1-m}}$, consider the paths p_i $(i = 1, 2)$ from v to the root that start with ψ_i. Without loss of generality, p_1 is not longer than p_2 (in terms of VP-hops). Therefore, we can remove ψ_2 from the VPL$^{\text{1-m}}$ without damaging its feasibility, contradicting the minimality assumption.

Proof of Lemma 4 □

4.2-4 Note that the previous lemma implies that the virtual graph G_Ψ of a minimal VPL$^{\text{1-m}}$ on a tree, is a tree as well.

Lemma 5 *Given $h > 0$, let T be a tree rooted at r. Every (h, r)-minimal VPL$^{1\text{-}m}$ on T can be represented by a set of vectors $\left\{ M_v \in \mathbb{N}^h \mid v \in V(T) \right\}$, such that for every node v, the following conditions hold:*

1. $\mathcal{L}(v) = \|M_v\|$,
2. *if v is a leaf then $M_v = (1, 0, \ldots, 0)$, and*
3. $M_v \geq \sum_{s \in Sons(v)} M_s + (1, 0, \ldots, 0)$.

Proof of Lemma 5 Let T_v denote the set of nodes in the subtree below v. Define M_v for a node v by

$$M_v[i] = |\{ \psi \in E_\Psi \mid v \in \psi \wedge \hbar(\psi) = i \}|$$

Condition 2 is trivially satisfied. Condition 1 is satisfied since each VP that includes the node v contributes one to a single $M_v[i]$ (according to the value of $\hbar(\psi)$), hence $\mathcal{L}(v) = \|M_v\|$. To prove Condition 3, let ψ_v be the VP starting at v. Note that if $\mathcal{H}(\psi_v) = 1$ then no VP stops at v, and every VP that loads a son of v loads v as well, which has an additional load incurred by ψ_v, hence $M_v = \sum_{s \in Sons(v)} M_s + (1, 0, \ldots, 0)$. If $\mathcal{H}(\psi_v) = i > 1$ then a VP ψ that stops at v has $\mathcal{H}(\psi) \leq i - 1$, hence $M_v[i\colon h] = \sum_{s \in Sons(v)} M_s[i\colon h] + (1, 0, \ldots, 0)$ and $M_v > \sum_{s \in Sons(v)} M_s + (1, 0, \ldots, 0)$.

Proof of Lemma 5 □

Lemma 6 *Given $h > 0$, if there exists an (h, r)-feasible VPL$^{1\text{-}m}$ Ψ with $\mathcal{L}(\Psi) \leq X$ for some $X > 0$ with a vector representation $\{ M_v \mid v \in V(T) \}$, then for every node $v \neq r$ the following holds:*

- *If $\mathcal{L}_{max} = X$ then find_layout does not return* FAILURE *while handling v, and*
- *the vector L_v produced by find_layout satisfies $L_v \leq M_v$.*

Proof of Lemma 6-1 **Proof of Lemma 6** We use induction on the height H of the subtrees of T.

Proof of Lemma 6-2 **Basis.** For a tree of height $H = 0$ (i.e., a leaf), it is clear that $L_v = M_v = (1, 0, \ldots, 0)$.

Proof of Lemma 6-3 **Induction step.** Assume that the claim holds for subtrees of height at most $H-1$, and let v be a node such that T_v is of height H. By the induction hypothesis, *find_layout* does not return FAILURE for every son s of v and $L_s \leq M_s$. Let $L'_v = \sum_{s \in Sons(v)} L_s + (1, 0, \ldots, 0)$.

Proof of Lemma 6-4 If $\|L'_v\| \leq X$ then *find_layout* will continue all VPs through v (no transformations are performed), in which case, (1) it does not return FAILURE while handling v, and (2) the vector $L_v = L'_v \leq \sum_{s \in Sons(v)} M_s + (1, 0, \ldots, 0) \leq M_v$.

Proof of Lemma 6-5 The case when $\|L'_v\| > X$ is proven as follows. Arrange the load vectors of the sons of v in the order they are chosen at lines 16–19 (which is the order determined by Property P2 of Definition 8), and add a pair of $(1, 0, \ldots, 0)$ vectors to this sequence. The sequence that includes these load vectors and the corresponding M vectors is proper.

$$\begin{array}{rcl}
(1, 0, \ldots, 0) & \leq & (1, 0, \ldots, 0) \\
L_1 & \leq & M_1 \\
L_2 & \leq & M_2 \\
& \vdots & \\
L_s & \leq & M_s
\end{array}$$

Proof of Lemma 6-6 By Lemma 3, there exists a vector $P = (p_1, \ldots, p_s)$ of integers $0 \leq p_i \leq h-1$ such that $\Sigma(\mathcal{S}_L^{\mathrm{trans}(P)}) \leq \Sigma(\mathcal{S}_R)$ and $\|\Sigma(\mathcal{S}_L^{\mathrm{trans}(P)})\| \leq \|\Sigma(\mathcal{S}_R)\|$. Note that $\Sigma(\mathcal{S}_L^{\mathrm{trans}(P)})$ is the value of the vector L_v after some finite number of iterations in the internal loop of *find_layout* for node v. The only way that *find_layout* may return FAILURE is if no son that satisfies the conditions of lines 16–19 is found. In this case, all sons s have $\|L_s[1\colon h-1]\| \leq 1$. By Lemma 1, $\|L_s\| \leq \|M_s\|$ and hence $X < \|L'_v\| \leq \|\Sigma(\mathcal{S}_R)\|$—which is a contradiction to the feasibility of the given solution, since by Lemma 5 $\|\Sigma(({}_R\mathcal{S}))\| \leq \|M_v\|$. Therefore we get $\Sigma(({}_L\mathcal{S})^{\mathrm{trans}(P)}) \leq X$ and *find_layout* finishes handling v without returning FAILURE with $L_v \leq M_v$.

Proof of Lemma 6 □

4.2-5 The theorem that concludes the proof is:

Theorem 1 *Given $h > 0$, the main procedure finds an (h, r)-optimal VPL$^{1\text{-}m}$ for any given tree T with N nodes rooted at r, in $O(hN \log_2 N)$ time.*

Proof of Theorem 1-1 **Proof of Theorem 1** By Lemma 6, *find_layout* will successfully handle each son of r if $\mathcal{L}_{\max}$ implies a feasible solution.

Proof of Theorem 1-2 It remains to show that *find_layout* does not fail for the root r. This proof is identical to the induction step of Lemma 6, except that the $(1, 0, \ldots, 0)$ vectors are not added as part of the list (since no new VP starts at the root r). Hence, Lemma 3 is applied on the load vectors L_i produced by *find_layout* for the sons of r and on the load vectors M_i for those sons in some other $\mathrm{VPL}^{\text{1-m}}$.

Proof of Theorem 1-3 Therefore, *find_layout* will return SUCCESS after handling the root (at line 9). It also returns a feasible layout that obeys $\mathcal{L}_{\max}$. Since *main* finds the minimal value for $\mathcal{L}_{\max}$ for which *find_layout* returns SUCCESS, it returns the optimal $\mathrm{VPL}^{\text{1-m}}$.

Proof of Theorem 1-4 As to the time complexity of this algorithm, the search for the optimal $\mathcal{L}_{\max}$, done by *main*, involves $\log_2 N$ steps (by means of a binary search in the range $1, \ldots, N$ for $\mathcal{L}_{\max}$). In each such step, *find_layout* is activated and scans the tree bottom up, during which time each node v is chosen once (at line 10) as the current node. Each node may also be chosen up to h times (at line 16) as a son to be transformed, since each transformation of L_v increases the number of zeros in it by at least one, and L_v has h values.

Proof of Theorem 1 □

4.2-6 The above algorithm does not imply a numerical bound of the load incurred by the layout; however, using a result from [CGZ94], it is easy to derive such a bound.

Lemma 7 *For every tree network T, the $\mathrm{VPL}^{\text{1-m}}$ produced by main obeys $\mathcal{L}(\Psi) \leq hN^{\frac{1}{h}}$.*

Proof of Lemma 7 Since *main* produces an optimal layout, it produces a better layout than the one suggested by [CGZ94] for which the load obeys $\mathcal{L}(\Psi) \leq hN^{\frac{1}{h}}$.

Proof of Lemma 7 □

5 The $\mathrm{VPL}^{\text{m-m}}$ Problem

5-1 In this section we use the algorithm for $\mathrm{VPL}^{\text{1-m}}$ to solve $\mathrm{VPL}^{\text{m-m}}$, exemplifying our motivation for the definition of $\mathrm{VPL}^{\text{1-m}}$. But first we note that the

$\mathrm{VPL}^{\text{m-m}}$ problem is a hard problem as well, by proving it is NP-complete for any odd hop count. The details of the proof may be found in Appendix A.2.

5.1 A Solution for General Graphs

5.1-1 We now present a suboptimal construction scheme, based on a technique of [AP92], for the construction of "regional routing schemes" (used for regular routing problems).

5.1-2 The scheme is based on a clustering technique, which yields connected overlapping subgraphs of a given graph, such that each pair of nodes with distance less than a given integer δ may be found together in at least one cluster. In each cluster we choose a center (i.e., the node that is closest to all the other nodes of the cluster), construct a breadth-first search tree to the center, and connect all the other nodes of the cluster to it by a $\mathrm{VPL}^{\text{1-m}}$ with $\mu = 1$ and $h' = \frac{h}{2}$ (using the greedy algorithm of the previous section). Clearly, every pair of nodes whose distance is less than δ may be connected using no more than h hops via the pivot of their common cluster.

5.1-3 We repeat this scheme for increasing parameter δ (until the algorithm yields one cluster that includes the whole network). To minimize the stretch factor μ, each pair is connected using the $\mathrm{VPL}^{\text{1-m}}$ of the smallest common cluster. Refer to Figure 5 for a graphic demonstration of this scheme.

5.1-4 We now present a precise definition of the above scheme. These definitions are based on [AP92].

Definition 11 *The* j-neighborhood *of a node* $v \in V$ *is defined as*

$$\mathcal{N}_j(v) = \{\, w \mid d_G(v, w) \le j \,\}$$

The radius *of a node wrt a graph is defined by*

$$Rad(v, G) = \max_{w \in V} d_G(v, w)$$

The radius *of a graph is defined by*

$$Rad(G) = \min_{v \in V} Rad(v, G)$$

A center *of a graph is a node for which* $Rad(v, G) = Rad(G)$.

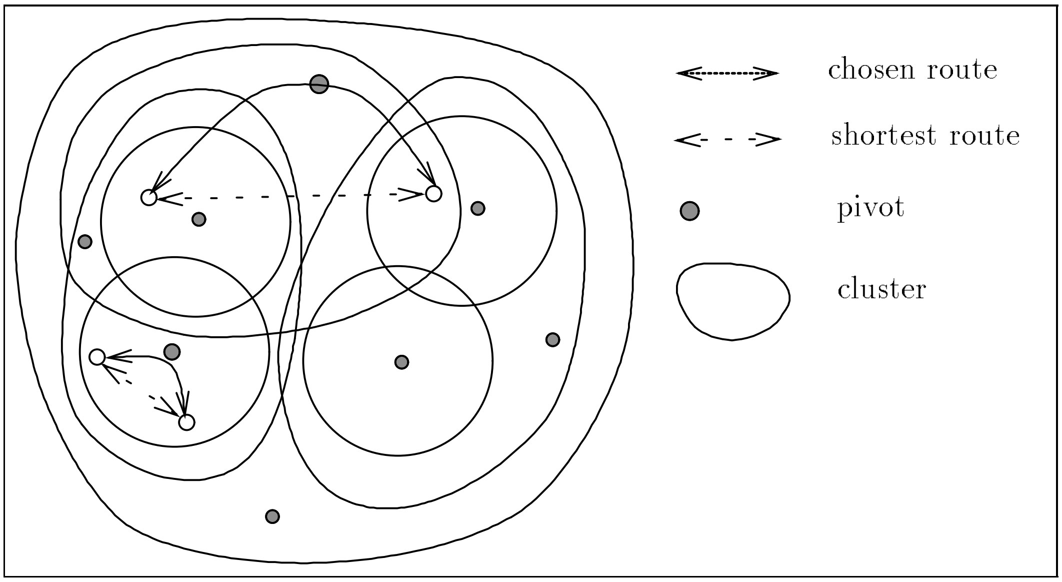

Figure 5: VPL$^{\text{m-m}}$ on a general graph

Definition 12 *Given a set of nodes $C \subseteq V$, let $G(C)$ denote the subgraph induced by C in G. A* cluster *is a subset C of nodes such that $G(C)$ is connected. We use $Rad(v, C)$ as a shorthand for $Rad(v, G(C))$. A* cover *is a collection of clusters $\mathcal{C} = \{ C_i \mid 1 \le i \le \delta \}$ such that $\bigcup_i C_i = V$. Extend the definitions of radius by $Rad(\mathcal{C}) = \max_i Rad(C_i)$. For a node v let $\Delta(\mathcal{C}, v)$ denote the number of occurrences of v in the clusters C_i. The* maximum degree *of a cover $\mathcal{C}$ is defined as $\Delta(\mathcal{C}) = \max_{v \in V} \Delta(\mathcal{C}, v)$. Given two covers $\mathcal{C} = \{ C_i \mid 1 \le i \le \delta \}$ and $\mathcal{T} = \{ T_i \mid 1 \le i \le k \}$ we say that $\mathcal{T}$* subsumes *$\mathcal{C}$ if for every $C_i \in \mathcal{C}$ there exists a $T_j \in \mathcal{T}$ such that $C_i \subseteq T_j$.*

5.1-5 We rely on the following theorem:

Theorem 2 ([AP92]) *Given a graph G, a cover $\mathcal{C}$, and an integer $k \ge 1$, it is possible to construct a cover $\mathcal{T}$ that satisfies the following properties:*

1. *$\mathcal{T}$ subsumes $\mathcal{C}$,*
2. *$Rad(\mathcal{T}) \le (2k-1)Rad(\mathcal{C})$, and*
3. *$\Delta(\mathcal{T}) = O(k\,|\mathcal{C}|^{1/k})$.*

Intuitively, this theorem states that the graph can be divided into overlapping clusters, with three important properties:

1. For each node v there is a cluster that includes v, and all its δ-neighborhood (recall that δ is a parameter of the algorithm).

2. The distance between every pair of nodes in each cluster is not more than $(2k-1)\delta$.

3. Each node is included in not-too-many clusters (i.e., $O(kN^{1/k})$).

5.1-6 Our construction is based on the routing scheme from [AP92], combined with the usage of VPL$^{\text{1-m}}$ for trees. We first present a scheme that depends on a parameter δ, and then tune the scheme by choosing δ. The scheme connects pairs of nodes with a distance not exceeding δ.

function *general_vpl*(G: network, δ, k: $\mathbb{N}$) $(\delta, k > 0)$
    returns VPL$^{\text{m-m}}$
    $\mathcal{C} \leftarrow \{\mathcal{N}_\delta(v) \mid v \in V(G)\}$
    Construct $\mathcal{T}$ as in Theorem 2
    Select a center c in each T_i
    Construct a VPL$^{\text{1-m}}$ in each T_i to the center with $h' = \frac{h}{2}$
    return the union of all the above VPL$^{\text{1-m}}$s
end function *general_vpl*

Lemma 8 *Let v, w be a pair of nodes such that $d_G(v,w) \leq \delta$ and let the stretch factor be $\mu = 8k$. Then the number of hops between v and w is $\mathcal{H}_\mu(v,w) \leq h$ in the VPL$^{m\text{-}m}$ produced by algorithm general_vpl.*

Proof of Lemma 8-1 **Proof of Lemma 8** Clearly, $v \in \mathcal{N}_\delta(w)$, and there exists a cluster $T_i \in \mathcal{T}$ that includes $\mathcal{N}_\delta(w)$. The route chosen for v, w will use the VPL$^{\text{1-m}}$ in T_i by going from v to the root of the VPL$^{\text{1-m}}$ (using a shortest route) and from there to w; denote the total length of it by X. This route is composed of no more than h VPs. Note that the distance to the center in $\mathcal{N}_\delta(w)$ is no more than δ, and by Property 2 of $\mathcal{T}$ we have:

$$X \leq 2Rad(\mathcal{T}) \leq 2(2k-1)Rad(\mathcal{C}) = 2(2k-1)\delta < 4k\delta$$

Proof of Lemma 8-2

Now suppose that $d_G(v,w) \geq \frac{\delta}{2}$, then the stretch factor satisfies $\mu \leq \frac{4k\delta}{\delta/2} = 8k$.

Proof of Lemma 8 □

Lemma 9 *Let Ψ be the VPL$^{\text{m-m}}$ produced by algorithm general_vpl. Then, its maximum load satisfies*

$$\mathcal{L}(\Psi) = O(khN^{\frac{1}{k}+\frac{2}{h}})$$

Proof of Lemma 9-1

Proof of Lemma 9 By Theorem 2, each node is shared by at most $\Delta(\mathcal{T}) = O(k\,|\mathcal{C}|^{1/k})$ clusters. On the other hand, using Lemma 7, the load on a node caused by a single VPL$^{\text{1-m}}$ is $\mathcal{L}(\Psi^{1-m}) \leq \frac{h}{2}N^{2/h}$, and since $|\mathcal{C}| \leq N$ we get

$$\mathcal{L}(\Psi) = O(hN^{\frac{2}{h}}kN^{\frac{1}{k}}) = O(khN^{\frac{1}{k}+\frac{2}{h}})$$

Proof of Lemma 9 □

Proof of Lemma 9-2

So far, we have assumed that the distance between v and w is not less than $\frac{\delta}{2}$. To achieve this, we apply the entire scheme for increasing values of δ. We start with $\delta = 2$ and at the ith round take $\delta = 2^i$. Clearly, this process completes in $\log_2 N$ rounds. To ensure that the assumption used for the calculation of the stretch factor holds, we choose the VPL$^{\text{1-m}}$ to connect v and w in the smallest cluster that includes both nodes. If we take $\delta = 2^i$ for an i that satisfies $2^{i-1} \leq d(v,w) \leq 2^i$, then we have $\frac{\delta}{2} \leq d(v,w)$ as assumed earlier. This process multiplies the load by a factor of $\log_2 N$, so we have

$$\mathcal{L}(\Psi) = O(kh\log NN^{\frac{1}{k}+\frac{2}{h}})$$

The following theorem summarizes the characteristics of the above description:

Theorem 3 *Given an arbitrary network G with N nodes, and $k,h > 1$, the above scheme yields an h-feasible VPL$^{\text{m-m}}$ with stretch factor $\mu \leq 8k$ and load $\mathcal{L}(\Psi) = O(kh\log NN^{\frac{1}{k}+\frac{2}{h}})$.*

6 Summary

6-1 In this paper we presented a model and a class of problems for designing the layout of virtual paths in a given ATM network to be used for general-purpose connection routing. We first defined the characteristics that are important for such a layout, the VPL$^{\text{m-m}}$ problem, and the simpler VPL$^{\text{1-m}}$ problem. Next, we showed that both problems are NP-complete, but presented an optimal construction scheme for a more limited variant of the VPL$^{\text{1-m}}$ (namely, with stretch factor $\mu = 1$ and assuming a tree network). We then demonstrated how this limited problem can serve as a building block for a suboptimal solution of a VPL$^{\text{m-m}}$ (in which $\mu > 1$).

6-2 We believe that our approach quite accurately models certain routing problems in ATM networks, including the optimality criterion. We expect this work to motivate further work in finding better schemes for general networks, in extending the existing schemes for supporting fault tolerance in the network, and in extending the characteristics of a good layout to the case when some statistics regarding the volume and bandwidth of connections between every pair of nodes is known.

7 Acknowledgment

We would like to thank the managing editor of this journal, Mike O'Donnell, and the copy editor, Lisa Clark, for their very thorough and involved work in improving the presentation of this paper.

A Appendix

A.1 NP-Completeness Result for the VPL$^{\text{1-m}}$ Problem

A.1-1 Recalling the problem statement in Section 4, we now prove that the OPT-VPL$^{\text{1-m}}$ problem is NP-complete (if the stretch factor is unbounded).

Theorem 4 *OPT-VPL$^{1\text{-}m}$ is NP-complete.*

Proof of Theorem 4-1 **Proof of Theorem 4** Clearly OPT-VPL$^{\text{1-m}}$ is in NP. To show NP-completeness, we reduce it to:

Problem: The disjoint connecting paths problem (ND40) [GJ79].

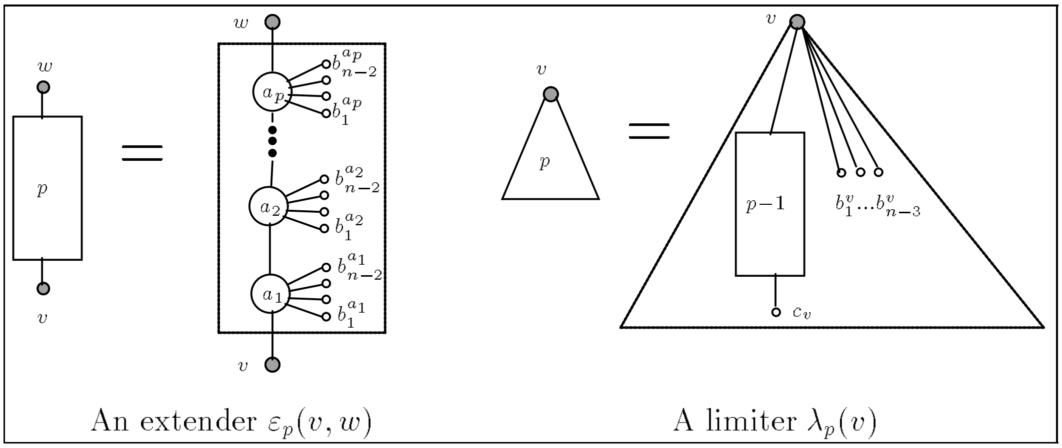

Figure 6: The limiter and extender components

INSTANCE: An undirected graph $G = (V, E)$, collection of disjoint pairs of nodes $(s_1, t_1), (s_2, t_2), \ldots, (s_k, t_k)$.

QUESTION: Does G contain k mutually node-disjoint paths, one connecting s_i and t_i for each $i \in \{1, \ldots, k\}$?

Given an instance of ND40, let $n = |V|$. Define the following components (see Figure 6):

Extender ($\varepsilon_p(v, w)$): This component *extends* a path that goes through v and w in V to include p additional hops. It is composed of a path of p nodes $a_1, \ldots, a_p$, each of which is connected to $n - 2$ new nodes $b^{a_i}_1, \ldots, b^{a_i}_{n-2}$. In addition, v is connected to a_1 and w to a_p. In the case when $p = 0$, the extender reduces to an edge (v, w).

Limiter ($\lambda_p(v)$): This component *limits* the path from its head v to the root, to include no more than $h - p$ hops. It also limits the number of VPs that may go through v. The limiter is constructed by connecting $n - 3$ new nodes $(b^v_1, \ldots, b^v_{n-3})$ to v, adding yet another new node c_v, and connecting v and c_v by $\varepsilon_{p-1}(v, c_v)$.

Let $U = V \setminus \{t_1, t_2, \ldots, t_k\}$, let $L = n$, and $h = k + 2$. Consider the following transformation of G to G' (see Figure 7, in which the extender is drawn as a rectangle and the limiter as a triangle):

- Initially define G' as G.
- Add a new node r to G'.
- Add a limiter $\lambda_1(v)$ to every node $v \in U$.
- Add an extender $\varepsilon_{h-1}(v, r)$ for every node $v \in U$.
- Add an extender $\varepsilon_{h-i+1}(s_i, r)$ to every $s_i, i \in \{1, \ldots, k\}$.
- Add a limiter $\lambda_i(t_i)$ to every $t_i, i \in \{1, \ldots, k\}$.
- Add a node d_i and an edge (d_i, t_i) to every $t_i, i \in \{1, \ldots, k\}$.

Proof of Theorem 4-2

We prove that OPT-VPL$^{\text{1-m}}$ replies *True* on G' with r as the root iff ND40 replies *True* on G. In the proofs we use the following terminology:

- A node v with $\mathcal{L}(v) = L$ is called *saturated*.
- A route from v to r is said to go *via* a node y if the route includes two VPs that are concatenated in y.
- A route from v to r is said to go *through* y if y is included in one of the VPs that form the route.
- In a minimal VPL, the (single) VP that starts at a node v and is used (as a first VP) in the path to r is called the VP *from* v.

Lemma 10 *The limiter and extender have the following properties:*

1. *Given a node v with a limiter $\lambda_p(v)$, the route from v to r is no more than $h - p$ hops.*
2. *Given a node v with a limiter $\lambda_p(v)$, then the VPs that include v are from v itself and from its limiter, plus at most one additional VP.*
3. *Given two nodes v, w with a limiter $\lambda_q(v)$ and an extender $\varepsilon_p(v, w)$, then a route through v and w includes at least $p - 1$ complete VPs (excluding the VPs that go through v and w).*

Proof of Lemma 10 We divide the proof into the three cases:

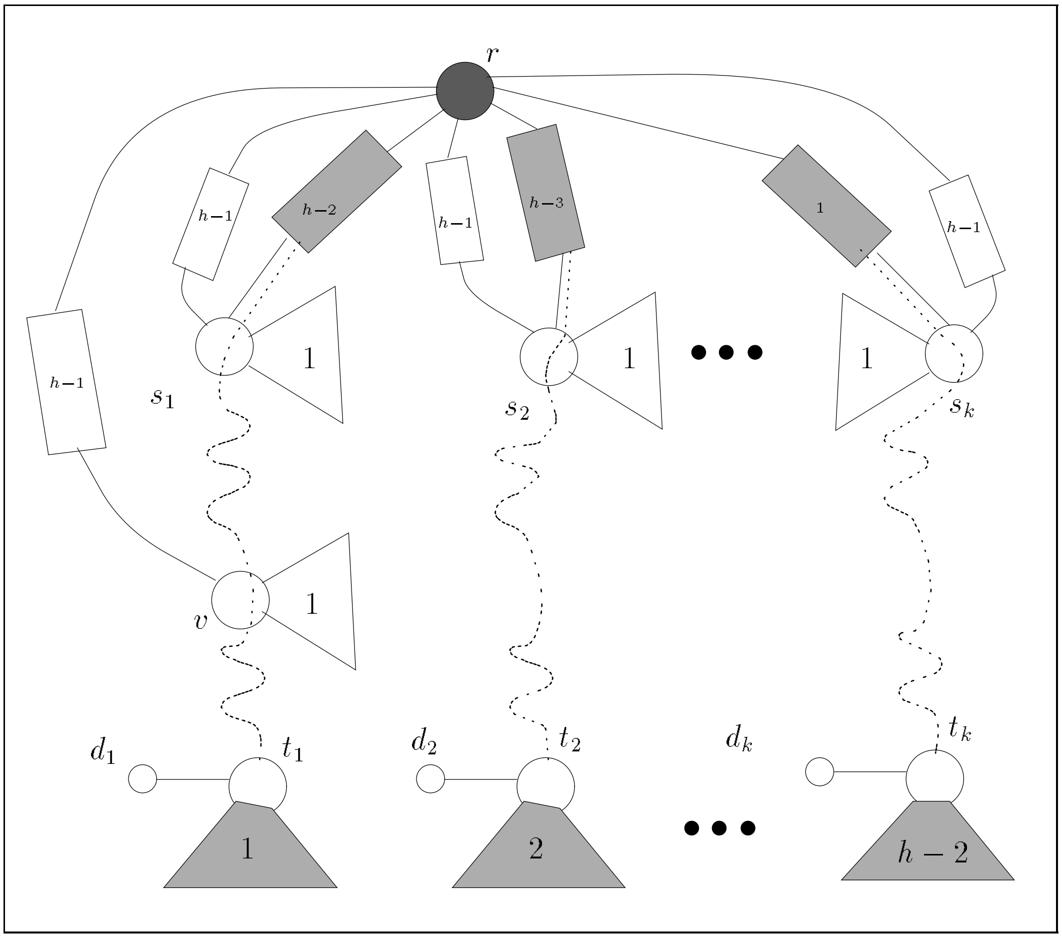

Figure 7: The transformed graph G'

Case 1 Since c_v is connected to a node a_1 of an extender in $\lambda_p(v)$, its VP can either end at a_1 or continue (toward v). If it continues, then the VP from a_1 must also go through a_2 (there is no other way to connect r), and $\mathcal{L}(a_2) > L$. Therefore it stops at a_1. From similar considerations, the VP from a_2 stops at a_3 and finally, the VP from a_p stops at v, otherwise, there are two VPs that must go through some node $u \in U$ and hence more than k VPs that go through some s_i, causing $\mathcal{L}(s_i) > L$ for some s_i. Therefore, v must connect r using $h - p$ hops or less (the case when $p = 1$ is trivial).

Case 2 Every VP from b_v in $\lambda_p(v)$ goes through v, plus one VP from v and one from a_p, a total of $L-3+1+1 = L-1$, leaving one extra VP through v.

Case 3 The proof resembles the one of Case 1, with the difference that the VP from a_1 does not go through v because if it does, v is saturated, and no other VP can go through v.

Proof of Lemma 10 □

Proof of Theorem 4-3

Proposition 1 *In every VPL$^{1\text{-}m}$ on G', the load on every node $v \in U$ is composed of one VP from v, one VP that includes v from the $L-2$ nodes in $\varepsilon_1(v)$, and no more than one additional VP, which we term the* extra VP *of v. In every $t_f i$, the load is composed of the same VPs and one VP from d_i, hence with no extra VP.*

Lemma 11 *If there exists a solution for OPT-VPL$^{1\text{-}m}$ on G', then there are k disjoint paths in G between every s_i and t_i.*

Proof of Lemma 11-1 **Proof of Lemma 11** By the previous Lemma, the route from t_k includes no more than two VPs. This route cannot go through any $\varepsilon_{h-1}(v, r)$, and it cannot go through s_i, $i \neq k$ because of the $\varepsilon_{h-i+1}(s_i, r)$ which adds at least $k + 2 - (k + 1) + 1 - 1 = 1$ VPs in the route from s_i to r (by Lemma 10), excluding the VP to r and the VP from t_k through s_i (a total of three VPs). Therefore, the route from t_k goes through s_k.

Proof of Lemma 11-2 From similar considerations, the route from t_{k-1} cannot go through s_i, for $i < k - 1$. Also, it cannot go through s_k, since the extra VP of s_k was used by the route from t_k. And by repeating this argument, the route from t_i goes through s_i. Note that this route is composed of a single VP (that includes both t_i and s_i), because $\lambda_i(t_i)$ and $\varepsilon_{h-i+1}(s_i, r)$ use $h-1$ VPs. Also note that each of these paths is node disjoint (since every intermediate node has only one extra VP), and that a path from t_i does not go through t_j, since every t_j is already saturated (because of d_j).

Proof of Lemma 11 □

Lemma 12 *If there are k disjoint paths between s_i and t_i respectively, then there exists a solution for OPT-VPL$^{\text{1-m}}$ on G' with $L = n$, $h = k + 2$.*

Proof of Lemma 12 Build a VPL$^{\text{1-m}}$ in the following manner:

- Create a VP between every b_i^v and v.
- Create a VP between every $v \in U$ to a_1 in $\varepsilon_{h-1}(v, r)$, and a VP between every a_p in $\varepsilon_p(x, r)$ to r (for every x for which such an extender exists).
- Connect c_v to a_1 of the extender in every $\lambda_p(v)$ by a VP, and a_p to v.
- Create a VP between every a_i to a_{i+1} in every extender (if $p = 0$ in $\varepsilon_p(u, v)$ then connect u and v by a VP).
- Connect d_i to t_i for every $i \in \{1, \ldots, k\}$.
- Connect every t_i to a_1 in $\varepsilon_p(s_i, r)$ using the path obtained by ND40 through s_i (as indicated by the dotted lines in Figure 7).

It is easy to verify that $\mathcal{L}(\Psi) \leq L$.

Proof of Lemma 12 □

Proof of Theorem 4-4 By Lemma 11, if there exists a VPL$^{\text{1-m}}$ on G', then there exist k disjoint paths from s_i to t_i in G. By Lemma 12, if there are k such paths, then there is a VPL$^{\text{1-m}}$ on G', hence our transformation is correct.

Proof of Theorem 4 □

A.2 NP-Completeness Result for the VPL$^{\text{m-m}}$ Problem

A.2-1 The proof that the VPL$^{\text{m-m}}$ problem is NP-complete is based on the following NP-complete problem.

PROBLEM: The Symmetric Forwarding Index Problem (SFI) [Saa93].

INSTANCE: An undirected graph $G = (V, E)$, an integer $k > 0$.

QUESTION: Does there exist a set R of $\binom{|V|}{2}$ undirected paths in G such that for every pair $x, y \in V$ there exists a path in R whose endpoints are x and y, and the number of paths in R that go through any node v, for which v is not an endpoint, does not exceed k?

A.2-2 The VPL$^{\text{m-m}}$ problem is formalized as the following decision problem.

PROBLEM: The optimal h-hop VPL$^{\text{m-m}}$ problem (OPT-VPL$^{\text{m-m}}(h)$).

INSTANCE: An undirected graph $G = (V, E)$, an integer $L > 0$.

QUESTION: Assuming $\mu = \infty$, does there exist an h-feasible VPL$^{\text{m-m}}$, Ψ, on G such that $\mathcal{L}(\Psi) \leq L$?

A.2-3 We first reduce the OPT-VPL$^{\text{m-m}}$ (1) problem to SFI, and then reduce OPT-VPL$^{\text{m-m}}$ $(h + 2)$ to OPT-VPL$^{\text{m-m}}$ (h) for every h, by this proving that OPT-VPL$^{\text{m-m}}(h)$ is NP-complete for every odd h. A proof that OPT-VPL$^{\text{m-m}}(2)$ is NP-complete would suffice to extend this result to every $h > 0$.

Lemma 13 *OPT-VPL$^{m\text{-}m}(1)$ is NP-complete.*

Proof of Lemma 13 Given an instance of SFI, we define an instance of OPT-VPL$^{\text{m-m}}$ (1) with the same graph G and $L = k + |V| - 1$. This proof is simple, since the only difference between the problems is that in SFI a path is not considered part of the "load" on the endpoint nodes, while in OPT-VPL$^{\text{m-m}}(1)$ it is considered as part of the load. However, since there are $n - 1$ paths in R for which a given node is an endpoint, the load limit must be increased by $|V| - 1$.

Proof of Lemma 13 □

Lemma 14 *OPT-VPL$^{m\text{-}m}(h + 2)$ is NP-complete if OPT-VPL$^{m\text{-}m}(h)$ is NP-complete.*

Proof of Lemma 14-1 **Proof of Lemma 14** Let a graph $G = (V, E)$ and $L > 0$ be an instance of OPT-VPL$^{\text{m-m}}$ (h), with $|V| = n$. Construct a graph $G' = (V', E')$ by initially taking G' as G and then connecting $M = nL$ new nodes $\{w_v^i\}_{i=1}^{M}$ to every node $v \in V$.

Proof of Lemma 14-2 We show that there exists a solution Ψ to OPT-VPL$^{\text{m-m}}(h)$ on G with load $\mathcal{L}(\Psi) \leq L$ iff there is a solution Ψ' to OPT-VPL$^{\text{m-m}}(h+2)$ on G' with load $\mathcal{L}(\Psi) \leq L' = L + M$. Assume Ψ is a solution Ψ to OPT-VPL$^{\text{m-m}}(h)$, and add VPs from every w_v^i to v. The resulting VPL Ψ' is a solution to OPT-VPL$^{\text{m-m}}$ $(h+2)$ since the distance in Ψ between every pair of nodes $a, b \in V$ satisfies $\mathcal{H}_\mu(a, b) \leq h$, and therefore the distance in Ψ' between every w_a^i and w_b^j obeys $\mathcal{H}_\mu(w_a^i, w_b^j) \leq h + 2$, and $\mathcal{H}_\mu(a, w_b^i) \leq h + 1$. Since the load in Ψ is bounded by L at each node, the load in Ψ' is clearly bounded by L'.

Proof of Lemma 14-3 On the other hand, if there exists a solution Ψ' to OPT-VPL$^{\text{m-m}}(h+2)$ on G' then two cases are possible:

Proof of Lemma 14-4 **Case 1** For every $a, b \in V$ in Ψ' $\mathcal{H}_\mu(a, b) \leq h$. In this case, we may take the VPs whose endpoints are in V as a solution Ψ to the OPT-VPL$^{\text{m-m}}(h)$ problem. It is clear that Ψ uses only nodes from G (a simple path from a to b cannot include any w_x^i), and that the solution is h-feasible. To show that $\mathcal{L}(\Psi) \leq L$, note that each node in $x \in V$ has at least load M from its neighbors w_x^i. This leaves only L for VPs between nodes in V.

Proof of Lemma 14-5 **Case 2** Otherwise, there exists a pair of nodes $a, b \in V$ for which $\mathcal{H}_\mu(a, b) > h$ in Ψ'. Regard the sets of nodes $W_a = \{\, w_a \mid 1 \leq i \leq M \,\}$ and $W_b = \{\, w_b^i \mid 1 \leq i \leq M \,\}$. If there exists a pair of nodes $x_a \in W_a$, $x_b \in W_b$ which are included only in the VPs $(x_a, a), (x_b, b) \in E_\Psi$, then the path between x_a and x_b must include the path between a and b, and hence $\mathcal{H}_\mu(x_a, x_b) = \mathcal{H}_\mu(a, b) + 2 > h + 2$—a contradiction. For this reason, either all nodes in W_a or all nodes in W_b (without loss of generality, assume it is W_a) are included in VPs that include also a neighbor of a. By the pigeonhole principle, there exists some neighbor $c \in V$ that is included in at least $\frac{M}{n}$ such VPs. Now, since every node w_c^j adds to the load on c as well, we get $\mathcal{L}(c) \geq M + \frac{M}{n} > M + L = L'$—a contradiction.

Proof of Lemma 14-6 Note that if there exists a VP (w_a^i, w_c^j) in Ψ', which is considered in the second component of the above sum (i.e., as a VP from w_c^j), there must be another VP from w_c^j that contributes to the first component of this sum as well, since (w_a^i, w_c^j) is part of the path from w_a^i to some node x_b in W_b.

Proof of Lemma 14 □

Theorem 5 *OPT-VPL$^{m\text{-}m}$ (h) is NP-complete for every odd $h > 0$.*

Proof of Theorem 5 The proof follows directly from the previous two lemmata by induction on h.

Proof of Theorem 5 □

Acknowledgement of support: Part of Ornan Gerstel's work was done while he was with the Computer Science Department, Technion, Haifa 32000, Israel. Part of Israel Cidon's work was done while he was with Sun Microsystems Labs, 2550 Garcia Avenue, Mountain View, CA 94043.

References

[ABLP89] B. Awerbuch, A. Bar-Noy, N. Linial, and D. Peleg. Compact distributed data structures for adaptive routing. In *21st Symposium on Theory of Computing*, pages 479–489, 1989.

[ACG+90] B. Awerbuch, I. Cidon, I. Gopal, M. Kaplan, and S. Kutten. Distributed control for PARIS. In *9th Annual ACM Symposium on Principles of Distributed Computing*, pages 145–160, 1990.

[AP92] B. Awerbuch and D. Peleg. Routing with polynomial communication-space tradeoff. *SIAM Journal on Discrete Math*, 5(2):151–162, May 1992.

[ATTD94] S. Ahn, R. P. Tsang, S. R. Tong, and D. H. C. Du. Virtual path layout design on ATM networks. In *IEEE Infocom '94*, pages 192–200, 1994.

[CG88] I. Cidon and I. Gopal. PARIS: An approach to integrated high-speed networks. *International Journal of Digital and Analog Cabled Systems*, 1(2):77–86, April–June 1988.

[CGG+92] I. Cidon, I. Gopal, P. M. Gopal, J. Janniello, and M. Kaplan. The plaNET/ORBIT high speed network. IBM Research Report RC 18270, IBM Research Division, Watson Research Center, August 1992.

[CGZ94] I. Cidon, O. Gerstel, and S. Zaks. A scalable approach to routing in ATM networks. In G. Tel and P. M. B. Vitányi, editors, *The 8th International Workshop on Distributed Algorithms (LNCS 857)*, pages 209–222, Terschelling, The Netherlands, October 1994. Springer-Verlag. To appear in *IEEE/ACM Transactions on Networking*.

[CS94] R. Cohen and A. Segall. Connection management and rerouting in ATM networks. In *IEEE Infocom'94*, pages 184–191, 1994.

[Eve79] S. Even. *Graph Algorithms*. Computer Science Press, 1979.

[FJ86] G. N. Frederickson and R. Janardan. Separator-based strategies for efficient message routing. In *27th Symposium on Foundations of Computer Science*, pages 428–437, 1986.

[FJ88] G. N. Frederickson and R. Janardan. Designing networks with compact routing tables. *Algorithmica*, 3:171–190, 1988.

[Ger95] O. Gerstel. *Virtual Path Design in ATM Networks*. PhD thesis, Technion, Israel Institute of Technology, December 1995.

[GJ79] M. R. Garey and D. S Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Co., 1979.

[GZ94] O. Gerstel and S. Zaks. The virtual path layout problem in fast networks. In *The 13th Annual ACM Symposium on Principles of Distributed Computing*, pages 235–243, Los Angeles, CA, August 1994.

[HH91] R. Händler and M. N. Huber. *Integrated Broadband Networks: An Introduction to ATM-Based Networks*. Addison-Wesley, 1991.

[HKIT94] H. Hadama, R. Kawamura, T. Izaki, and I. Tokizawa. Direct virtual path configuration in large-scale ATM networks. In *IEEE Infocom'94*, pages 201–207, 1994.

[ITU90] ITU recommendation. I series (B-ISDN), Blue Book, November 1990.

[KK77] L. Kleinrock and F. Kamoun. Hierarchical routing for large networks; preformance evaluation and optimization. *Computer Networks*, 1:155–174, 1977.

[KK80] L. Kleinrock and F. Kamoun. Optimal clustering structures for hierarchical topological design of large computer networks. *Networks*, 10:221–248, 1980.

[LB92] J. Y. Le Boudec. The asynchronous transfer mode: a tutorial. *Computer Networks and ISDN Systems*, 24:279–309, 1992.

[LPS91] T. F. La Porta and M. Schwarz. Architectures, features, and implementation of high-speed transport protocols. *IEEE Communications Magazine*, pages 14–22, May 1991.

[PU88] D. Peleg and E. Upfal. A tradeoff between space and efficiency for routing tables. In *20th Symposium on Theory of Computing*, pages 43–52, 1988.

[Saa93] R. Saad. Complexity of the forwarding index problem. *SIAM Journal on Discrete Math*, 6(3):418–427, 1993.

[SK85] N. Santoro and R. Khatib. Labelling and implicit routing in networks. *The Computer Journal*, 28:5–8, 1985.