DESIGN OF NONBLOCKING ATM NETWORKS

J. Andrew Fingerhut, Rob Jackson
Subhash Suri, Jonathan S. Turner

WUCS-96-03

January 31, 1996

Department of Computer Science
Campus Box 1045
Washington University
One Brookings Drive
St. Louis, MO 63130-4899

Abstract

This paper considers the problem of designing ATM networks that are nonblocking with respect to virtual circuit requests, subject to specified constraints on the traffic. In this paper, we focus on global traffic constraints that simply limit the total entering and exiting traffic at each switching system. After reviewing prior results for linear link costs, we introduce a more realistic link cost model, and develop a number of results using it. We also describe a technique for converting tree-structured networks to nonblocking hierarchical networks satisfying limits on the capacity of any single switch

[0]This work was supported by the National Science Foundation, Ascom Timeplex, Bay Networks, Bell Northern Research, NEC, NTT, Southwestern Bell and Tektronix.

DESIGN OF NONBLOCKING ATM NETWORKS

J. Andrew Fingerhut, Rob Jackson
Subhash Suri, Jonathan S. Turner

1. Introduction

As computer networks get larger and more complex, the need for careful planning in the design and configuration stage becomes more and more important. This has become an issue even for conventional shared-access LAN and router based networks, but is even more crucial for ATM. Because ATM networks support virtual circuit routing and must provide quality-of-service guarantees to real-time traffic (voice, video, etc.), connection requests can block if the network backbone does not have sufficient available bandwidth to satisfy a user's needs. To ensure that blocking occurs only rarely, the network backbone must be dimensioned with sufficient bandwidth to allow it to carry both the normal day-to-day traffic and peak traffic loads that may arise from time-to-time.

The network design problem is not a new one. References [8, 7, 12, 13] are representative of the recent published research in this area. However, ATM networks differ from telephone and classical data networks in several ways. First, they are multirate networks, meaning that their virtual circuits can operate at any bandwidth from a few bits per second to over one hundred megabits per second. They will be supporting a wide range of different applications with different bandwidth needs, different connection request rates and different holding times. Moreover, unlike traditional data networks they must be capable of providing connections with a guaranteed quality of service, requiring allocation of bandwidth to individual virtual circuits and raising the possibility of virtual circuit blocking. Second, ATM networks support not only point-to-point virtual circuits but also multipoint. Multipoint virtual circuits are essential for applications like video distribution or multimedia conferencing and include both one-to-many and many-to-many transmission patterns. The design of optimal networks supporting multipoint virtual circuits is largely uncharted territory. We have found nothing explicit in the technical literature that addresses this problem, although many point-to-point results can be usefully generalized to the multipoint environment. Finally, ATM networks are much less predictable than telephone networks or traditional low speed data networks. There is no reliable statistical data on application characteristics and connection request patterns. Indeed, the flexibility which is ATM's greatest strength makes it highly unpredictable, so classical network planning techniques which rely heavily

on statistical analysis become less relevant. In ATM networks, the whole notion of blocking probability for virtual circuit setup must be called into question, since there is no reasonable possibility of validating the probabilistic assumptions that must go into any analysis of blocking probability.

2. Network Design with Global Traffic Constraints

We start with the following formulation of the network design problem. We are given a complete graph, $G = (V, E)$, where each vertex represents a *switch* and each directed edge represents a *link group*, comprising one or more physical transmission links. Each vertex u has an integer *source capacity* $\alpha(u)$ and *sink capacity* $\omega(u)$, representing the maximum amount of traffic that can originate or terminate at u. For each vertex pair (u, v), we have a value $\gamma_\ell(u, v, x)$ that represents the cost of constructing a link of capacity x from u to v. We require that the costs satisfy the *triangle inequality*, meaning that the direct path of any given capacity between two vertices is never more expensive than an indirect path with the same capacity. We also have a switch cost function $\gamma_s(x)$ which gives the cost of a switch whose total capacity is x. If we assign a capacity $\kappa_\ell(u, v)$ to every edge (u, v) the resulting network cost is defined as

$$\sum_{(u,v)\in E} \gamma_\ell(u, v, \kappa_\ell(u, v)) + \sum_{u\in V} \gamma_s \left(\alpha(u) + \omega(u) + \sum_{v\in V, v\neq u} (\kappa_\ell(u, v) + \kappa_\ell(v, u)) \right)$$

The capacity of a vertex u in the network is denoted by $\kappa_s(u)$ and is given by the expression above within the large parentheses.

A *connection request* $R = (S, D, x)$ comprises a non-empty set of *sources* S, a non-empty set of *destinations* D and an integer *weight* $w \leq B$, where B is a maximum connection weight. If $S = D$ we say the request is *symmetric* and if $|S| + |D| = 2$, we refer to the request as *point-to-point*. A *route* T for a request R is a subgraph of G for which the underlying undirected graph is a tree and in which there is a directed path from every vertex in S to every vertex in D and no edge which does not belong to some simple directed path from a vertex in S to a vertex in T. A collection of routes C, places a weight $\lambda_C(u, v)$ on an edge u, v, where $\lambda_C(u, v)$ is defined as the sum of the weights of all routes that include the edge (u, v). We also let $\lambda_C(u)$ denote the weight on a switch u, which is equal to the sum of the weights of its incident edges.

A set of connection requests is *valid*, if for every vertex u, the sum of the weights of the requests containing u in their source set is $\leq \alpha(u)$ and the sum of the weghts of the requests containing u in their sink sets is $\leq \omega(u)$. A collection of routes C is *valid* if the connection requests they satisfy are valid and if for every edge (u, v), $\lambda_C(u, v) \leq \kappa_\ell(u, v)$ and if for every vertex u, $\lambda_C(u) \leq \kappa_s(u)$.

A *state* of a network is a valid set of routes. A *routing algorithm* is a procedure that maintains a valid set of routes under the following four operations: (1) add a new route satisfying a specified connection request; (2) remove an existing route; (3) add a new vertex to either the source set, the destination set, or both for some route in the current state; (4)

remove some vertex from either the source set, the destination set, or both for some route in the current state. A routing algorithm may fail to carry out operations of type (1) or (3), but will always carry out operations of type (2) or (4). We are only concerned with routing algorithms that are *incremental*, meaning that they only add, delete or modify a single route when carrying out a requested operation and that they cannot both add and remove edges from an existing route in a single operation.

The *reachable states* for a routing algorithm on a network with specified link capacities is the set of all states that can be reached by sequences of the four operations given above, starting from the empty state. We say that a given network is *routable* under a given algorithm, if for every valid set of connection requests, there is reachable state whose routes satisfy the given set of connection requests. We say that a network is *nonblocking* under a given routing algorithm if for every reachable state and every operation request whose completion would not exceed the source or sink capacity of any vertex in that state, the algorithm produces a new state satisfying the operation request.

The *routable network design problem* is to determine a set of link capacities that will yield a routable network of least cost under either a specified routing algorithm or some routing algorithm from a specified class of routing algorithms. In the latter case, the design problem is to produce both the link capacities and a specific routing algorithm from the given class, for which the network is routable.

Similarly, the *nonblocking network design problem* is to determine a set of link capacities that will yield a nonblocking network of least cost under either a specified routing algorithm or some routing algorithm from a specified class of routing algorithms. In the latter case, the design problem is to produce both the link capacities and a specific routing algorithm from the given class, for which the network is nonblocking.

Note that a solution to the nonblocking network design problem is also a solution to the routable network design problem, although the reverse does not hold. Also, a lower bound on the cost of a solution to the routable network design problem is also a lower bound on the cost of a solution to the nonblocking network design problem.

We say that a routing algorithm A is a *fixed path* routing algorithm if for all vertex set pairs (U, V), there is a route $A(U, V)$ which is used by A for all connection requests with source set U and sink set V. Some routing algorithms choose routes based on the cost of the fraction of the network's capacity that is used by those routes. We can define the cost a collection of routes C as

$$\sum_{(u,v)\in E} \frac{\lambda_C(u,v)}{\kappa_\ell(u,v)} \gamma_\ell(u,v,\kappa_\ell(u,v)) + \sum_{u\in V} \frac{\lambda_C(u)}{\kappa_s(u)} \gamma_s(\kappa_s(u))$$

A routing algorithm A is called a *cheapest available path* algorithm if when adding a new route or augmenting an existing route, the change in cost incurred by the operation is the smallest possible. Note that for fixed path routing algorithms, any routable network is also a nonblocking network.

There are some special cases of the network design problem that we often focus on. In the *linear* version, switch costs are zero and all link costs satisfy $\gamma_\ell(u,v,x) = c(u,v)x$, where

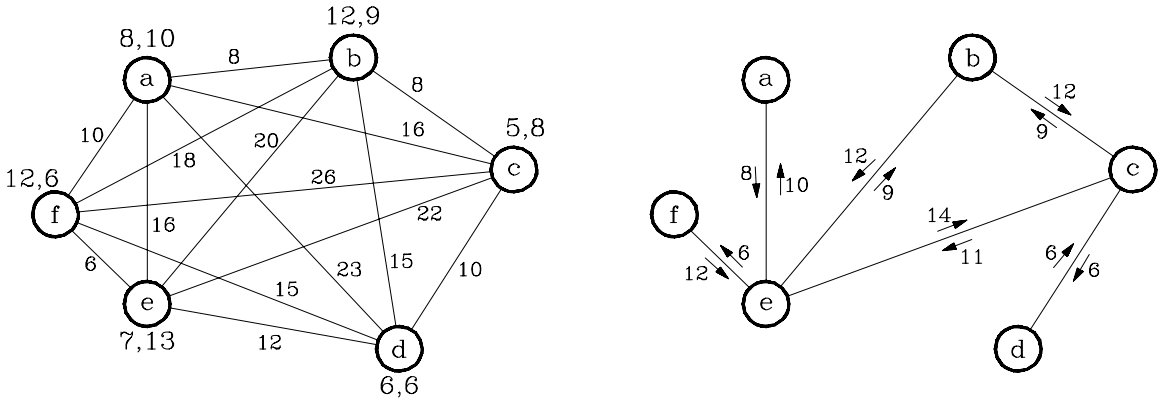

Figure 1: Example of Design Problem and Suboptimal Solution

$c(u, v)$ is a constant that depends only on u and v. The *symmetric* version of the problem has $\alpha(u) = \omega(u)$ for all vertices u, $\gamma_\ell(u, v, x) = \gamma_\ell(v, u, x)$ for all pairs u, v and restricts the choice of link capacities so that $\kappa_\ell(u, v) = \kappa_\ell(v, u)$. In the *balanced* version of the problem, $\sum_{u \in V} \alpha(u) = \sum_{u \in V} \omega(u)$. In the *point-to-point* version of the problem, connection requests are all point-to-point.

Figure 1 shows an instance of the network design problem on the left and a solution on the right. In the problem instance, each vertex u is labeled by the pair $\alpha(u), \omega(u)$ and the edge joining vertices u, v with the cost per unit capacity (assuming symmetric, linear link costs). In the network design shown on the right, the edges are labeled with the capacities (κ_ℓ), with the arrows indicating the capacity in each direction along a link. This particular network design is nonblocking if connections are always routed using least cost paths, meaning that any set of connection requests that does not exceed the source and sink capacities can be satisfied. Its cost is $18\times6+18\times16+21\times20+21\times8+25\times22+12\times10 = 1654$.

3. Results for Linear Link Costs

A number of results have been established for the linear version of the problem.

- Given any fixed path routing algorithm for point-to-point connections, there is an efficient algorithm for calculating the link capacities that yields a minimum cost nonblocking network. This can be used to calculate the link capacities in networks in which the links with non-zero capacities are restricted to form a specific tree.
- On the other hand, the network design problem is MAX-SNP-hard if we are required to produce a fixed path routing algorithm, along with the network design. This means that no polynomial time algorithm can always produce solutions that are guaranteed to be close to the cost of the optimal solution unless P=NP.
- One can efficiently compute a lower bound on the cost of an optimal solution by computing a maximum weighted matching.

- For balanced source/sink capacities, star networks are optimal among nonblocking tree networks (where a tree network is a network in which the underlying undirected graph induced by the edges of non-zero capacity is a tree, and a star network is a tree network in which there is one central vertex that is connected to all the other vertices).

- When n vertices are distributed at random in the unit disk and link costs are proportional to the Euclidean distances between vertices, and source/sink capacities are all equal to one, then with probability approaching one as $n \to \infty$, there is a nonblocking star network for which the cost is arbitrarily close to that of an optimal network.

- Any network that is nonblocking for point-to-point connection requests under a fixed path routing algorithm A is also nonblocking for general requests under routing algorithm A.

- For symmetric capacities and links, linear link costs and and $\gamma(u, v, x)/x$ equal to a constant for all choices of u, v and x, a star network is optimal. While this is a somewhat specialized case, it applies to practical situations where the link costs are dominated by the cost of the terminating electronics and where there is a single type of link from which larger link groups must be constructed.

- For symmetric capacities and links and linear link costs there is a star network whose cost is never more than twice the cost of the optimal solution to the routable network design problem.

- For balanced source/sink capacities and linear link costs, there is a star network whose cost is never more than three times the cost of the optimal solution to the routable network design problem.

- For unbalanced source/sink capacities and linear link costs, there is a star network whose cost is never more than

$$2 + \frac{\max\ \{\sum_u \alpha(u), \sum_u \omega(u)\} - 1}{\min\ \{\sum_u \alpha(u), \sum_u \omega(u)\}}$$

  times the cost of the optimal solution to the routable network design problem. Thus, the more imbalanced the source/sink capacities become the further from optimal the star network becomes.

These results are described in detail in [5, 6].

4. Realistic Link Models

The linear link cost model ignores the fact that communication links of different capacities must be constructed from a limited set of link types, each with its own capacity. For example, in ATM networks, some common link capacities are 50 Mb/s, 150 Mb/s, 600 Mb/s and 2.4 Gb/s. If we need a certain amount of capacity, say 11 Gb/s, we must provide

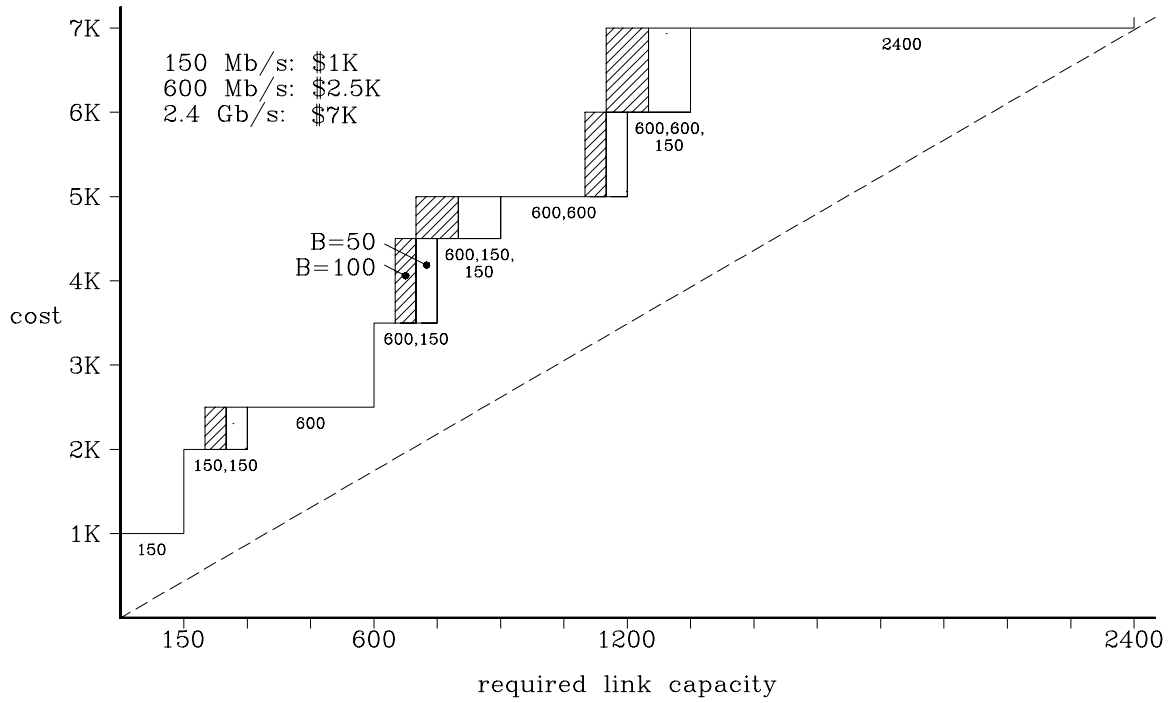

Figure 2: Link Group Costs Based on Discrete Links

this by using some combination of links of the available types, and we'd like to accomplish this for the least possible cost. To model the cost properly, we need to take into account the fact that the cost of a link is determined by a combination of two factors, one of which is independent of the length of the link and another which is linear in the length of the link. For example, the electronics associated with the two endpoints of a 150 Mb/s bidirectional ATM link has a cost of between $1,000 and $2,000 in typical products today, while the cost of the pair of fibers is directly proportional to distance and has a cost of between five cents per foot and fifteen cents per foot. Note that for short enough distances (say a few hundred feet), the fiber cost is negligible compared to the electronics cost, while for long enough links (say twenty miles), the fiber costs are dominant.

Another issue arises from the fact that virtual circuits in ATM networks must be routed over a specific link in the set making up a link group, so while there may be sufficient capacity over the set to accommodate a given virtual circuit, there may not be enough capacity on any single link to handle a new virtual circuit. Such fragmentation effects reduce the usable capacity of a link under worst-case assumptions. In general, the usable capacity of a link group containing m distinct physical links is equal to its total nominal capacity minus $(m-1)B$, where B is the maximum weight associated with any single virtual circuit. The implications of discrete link sizes and fragmentation due to virtual circuit blocking are illustrated in Figure 2.

Assume we have t different link types with capacities $k_1, \ldots, k_t$, termination costs $g_1^0, \ldots, g_t^0$ and cost per unit distance of $g_1^*, \ldots, g_t^*$. For arbitrary vertices u, v in a network and capacity x, we want to be able to compute $\gamma(u, v, x)$, the cost of a link joining u

and v with the given capacity. To do this, we need to define distances between all pairs of vertices, which reflect the length of the path through which fibers must be routed to join the vertices in question. Let $d(u, v)$ denote the distance from u to v. We can now define $g_i = g_i^0 + g_i^* d(u, v)$ to be the cost of a single link of type i between u and v.

The fragmentation problem described above can be factored out by transforming the problem slightly. Let $k_i^B = k_i - B$ and consider a set S of links with capacities in $k_1, \ldots, k_t$ and the corresponding set S' with capacities in $k_1^B, \ldots, k_t^B$. If the usable capacity of S is x when the maximum weight is B, then the usable capacity of S' is $x - B$ when the maximum weight is some arbitrarily small $\epsilon > 0$. The converse also holds. Thus by modifying the capacities of the individual links and the desired link group capacity by B, we can proceed without explicitly considering the fragmentation issue.

Determining the best set of links to construct a link group of a given capacity is a variant of the well-known knapsack problem, which is NP-complete. However, because of the limited number of link types that arise in practice, there is a dynamic programming algorithm that can produce optimal solutions efficiently enough for most practical purposes. Define $\mathit{cheapest}(i, z)$ to be the cost of the cheapest set of links using only types $1, \ldots, i$ that has capacity $\geq z$. For $i > 0$,

$$\mathit{cheapest}(i, z) = \min \{\mathit{cheapest}(i-1, z), \mathit{cheapest}(i, z - k_i^B) + g_i\}$$

where $\mathit{cheapest}(i, z) = 0$ if $z \leq 0$. Using the additional fact that $\mathit{cheapest}(1, z) = \lceil z/k_1^B \rceil g_1$ we can construct a dynamic programming algorithm that fills in a table whose entries are values of $\mathit{cheapest}(i, z)$ for $1 \leq i \leq t$ and z ranging from 0 to the desired link capacity. If r is the greatest common divisor of the rates k_i^B, then we need only compute $\mathit{cheapest}(i, z)$ for values of z that are multiples of r, making the number of entries in the table equal to t times the smallest integer that is at least equal to the ratio of the desired capacity to r. So for example with three link types whose raw capacities are 150 Mb/s, 600 Mb/s and 2.4 Gb/s and a maximum virtual circuit weight of $B = 50$ Mb/s, the required r is 50 Mb/s and a table with 6000 entries is sufficient to compute the best configurations for link capacities of up to 10 Gb/s. Filling in such a table can be done in a few milliseconds on a modern processor. If one is willing to sacrifice optimality of the computed set of links in order to reduce the time then one can do so using a variant of an algorithm by Ibarra and Kim [9] for the knapsack problem. This algorithm can produce solutions which approximate the optimal as closely as possible, allowing a smooth trade-off between the desired accuracy and the time required for the computation.

There is an alternative dynamic programming algorithm that can be preferable in some situations. In particular, if we let $\mathit{biggest}(i, z)$ be the capacity of the largest capacity link group with cost $\leq z$ using only links of type $1, \ldots, i$ then for $i > 1$

$$\mathit{biggest}(i, z) = \begin{cases} \max \left\{\mathit{biggest}(i-1, z), \mathit{biggest}(i, z - g_i) + k_i^B\right\} & \text{if } z \geq g_i \\ \mathit{biggest}(i-1, z) & \text{if } z < g_i \end{cases}$$

Using the boundary condition, $\mathit{biggest}(1, z) = \lfloor z/g_1 \rfloor k_1^B$, we can construct a dynamic programming algorithm that fills in a table whose entries are values of $\mathit{biggest}(i, z)$.

5. Network Design Algorithms for Multiple Link Types

When there are several different link types available, the higher capacity links generally offer a lower cost per unit capacity than do smaller capacity links. This advantage is magnified by fragmentation, which has a more severe impact on small capacity links and by long distances, since the cost of the fiber required for high capacity links differs little from that required for low capacity links. Consequently, whenever there is sufficient traffic available to make good use of high capacity links, it makes sense to use them. This suggests that in situations where multiple discrete link types are available, star networks will not offer the most cost-effective solution. Rather, we would expect the best networks to be tree-structured, with lower capacity links used near the leaves and higher capacity links toward the center.

In this section we consider network design algorithms designed to handle multiple link types. We start by describing a simple algorithm for two link types and prove a probabilistic result showing that for random distributions of switches in large planar regions, this algorithm produces solutions whose costs are near optimal, with probability approaching 1 as the number of switches grows. This is used to motivate another class of algorithms that can handle an arbitrary number of link types and can be expected to perform well for more irregular distributions of switches.

Consider an instance of the network design problem obtained by distributing n switches at random in a rectangular region of the plane with width X and height Y with $X \geq Y$. There are two types of links with capacities k_1, k_2, termination costs g_1^0, g_2^0 and cost per unit distance of g_1^*, g_2^*. Assume that $g_1^0 \leq g_2^0$, $g_1^* \leq g_2^*$ but $(g_1^0/k_1) \geq (g_2^0/k_2)$ and $(g_1^*/k_1) \geq (g_2^*/k_2)$. Euclidean distances define the length of links between switch pairs. For simplicity, we assume $\alpha(u) = \omega(u) = \alpha$ for all switches u.

Let $0 < \epsilon < 1$ be a constant. The *uniform parititioning algorithm* subdivides the given region into $R = 4r^2$ subregions of width x and height y, where $x/y = X/Y$, such that r is an integer and $n/(\ln n)^2 \leq R \leq n/(\ln n)^{(1+\epsilon)}$. Then for each region, it connects every switch in the region to a regional center, where the link groups all have capacity α. It then connects all the regional centers to a global center, which is positioned at the geographical center of the area. The link groups joining the global centers to the regional centers are assigned a capacity equal to the total termination capacity of their regions. It's easy to see that if no region has greater termination capacity than all the other regions combined, this results in a nonblocking network.

We evaluate the uniform partitioning algorithm by comparing the cost of the networks it produces to a lower bound on the cost of any nonblocking network as the number of switches gets large. First, we prove an elementary lemma on the number of switches per region. We call a distribution of switches *even* if every region has between $(1-\epsilon)n/R$ and $(1+\epsilon)n/R$ switches. The proof of lemma requires the following proposition due to Angluin and Valiant [1]) on the tail probabilities of the binomial distribution. Let $B(n,p)$ denote the binomial distribution. If x is a random variable with distribution $B(n,p)$ then by definition $P(x=k) = \binom{n}{k} p^k (1-p)^{n-k}$.

PROPOSITION 5.1. *If x is a random variable with distribution $B(n,p)$ then for all δ, $0 < \delta < 1, P(x \leq (1-\delta)np) < e^{-\delta^2 np/2}$ and $P(x \geq (1+\delta)np) < e^{-\delta^2 np/3}$.*

LEMMA 5.1. *If $\epsilon > 0$ is fixed and $R \leq n/(\ln n)^{(1+\epsilon)}$, the probability that the switch distribution is even approaches 1 as $n \to \infty$.*

proof. If x is a random variable representing the number of switches in a given region, then x has distribution $B(n, 1/R)$. Using Proposition 5.1, the probability that there is any region with fewer than $(1-\epsilon)n/R$ or greater than $(1+\epsilon)n/R$ switches is

$$\begin{aligned} &< R\left(e^{-\epsilon^2 n/2R} + e^{-\epsilon^2 n/3R}\right) \\ &= \exp\left[\ln R - \epsilon^2 n/2R\right] + \exp\left[\ln R - \epsilon^2 n/3R\right] \\ &< \exp\left[\ln n(1-(\epsilon^2/2)(\ln n)^{\epsilon})\right] + \exp\left[\ln n(1-(\epsilon^2/3)(\ln n)^{\epsilon})\right] \\ &= n^{1-(\epsilon^2/2)(\ln n)^{\epsilon}} + n^{1-(\epsilon^2/3)(\ln n)^{\epsilon}} \to 0 \end{aligned}$$

□

Define a Cartesian coordinate system with origin at the center of the rectangular area. Let $d = \sqrt{x^2+y^2}$ and $A = \sum_{i=0}^{r-1}\sum_{j=0}^{r-1}\sqrt{(ix)^2+(jy)^2}$. Note that A equals the sum of the minimum distances from the origin to each region in the upper right hand quadrant. Observe that if $i \geq j$, $\sqrt{(ix)^2+(jy)^2} \geq jd$ and if $i \leq j$, $\sqrt{(ix)^2+(jy)^2} \geq id$. This can be seen most easily by noting that the distance from a region to the origin is at least equal to the length of the line segment with slope y/x joining the region's point closest to the origin with either the horizontal or vertical axis. This yields

$$\begin{aligned} \sum_{i=0}^{r-1}\sum_{j=0}^{r-1}\sqrt{(ix)^2+(jy)^2} &\geq d\sum_{j=1}^{r-1} j + 2d\sum_{i=1}^{r-2}\sum_{j=1}^{i} j \\ &= d\left[\binom{r}{2} + \sum_{i=1}^{r-1} i(i-1)\right] \\ &= (1/6)dr(r-1)(2r-1) \end{aligned}$$

If $n \to \infty$, so does r and $A \to dr^3/3$. The following lemma provides a lower bound on the cost of a nonblocking network for an even distribution of switches.

LEMMA 5.2. *Given an even distribution of switches, any nonblocking network has a cost that is at least.*

$$\frac{(1-\epsilon)\alpha(n/R)}{k_2}\left[(1/2)g_2^0 R + 4g_2^* A\right]$$

proof. Any nonblocking network must satisfy the set of connection requests in which for every i, j pair for which $1 \leq i \leq r$, $1 \leq j \leq r$, there are requests from the region with center $((i-(1/2))r, (j-(1/2))r)$ to the region with center $(-(i-(1/2))r, -(j-(1/2))r)$ that

exhaust the termination capacity within one of the regions, plus requests from the region with center $(-(i-(1/2))r, (j-(1/2))r)$ to the region with center $((i-(1/2))r, -(j-(1/2))r)$ that exhaust the termination capacity within one of those regions. For each of these pairings there must be a link group whose capacity is at least $(1-\epsilon)\alpha(n/R)$ and whose length is at least equal to the minimum distance between the regions, which by choice of the regions is equal to twice their distance from the origin. Since the type two links have the lowest cost per unit capacity, the least expensive way to connect the regions is using type two links. Hence the cost of a nonblocking network must be at least

$$\frac{(1-\epsilon)\alpha(n/R)}{k_2}\left[(1/2)g_2^0 R + 4g_2^* A\right]$$

□

The following lemma provides a similar upper bound on the cost of a solution produced by the uniform partitioning algorithm.

LEMMA 5.3. *Given an even distribution of switches, the uniform partitioning algorithm produces a solution whose cost is at most*

$$\lceil \alpha/k_1 \rceil \left[g_1^0 + g_1^* d\right] n + \left\lceil \frac{(1+\epsilon)\alpha(n/R)}{k_2} \right\rceil \left[(1/2)g_2^0 R + 4g_2^*(A + 2r^2(x+y))\right]$$

proof. The first term is an upper bound on the cost of the links within the R regions. Each switch within a region is connected to the regional center by a link group of capacity α. Since this capacity can be provided by $\lceil \alpha/k_1 \rceil$ links of capacity k_1 and the cost per link is at most $g_1^0 + g_1^* d$, the cost of providing all the local link groups is bounded by the first term.

The second term is an upper bound on the cost of the links joining the global center to each of the regional centers. Observe that the cost of these links is at most

$$\left\lceil \frac{(1+\epsilon)\alpha(n/R)}{k_2} \right\rceil \left[(1/2)g_2^0 R + 4g_2^* \sum_{i=1}^{r}\sum_{j=1}^{r} \sqrt{(ix)^2 + (jy)^2}\right]$$

The double-sum is only slightly larger than A.

$$\begin{aligned}\sum_{i=1}^{r}\sum_{j=1}^{r} \sqrt{(ix)^2 + (jy)^2} &< A + \sum_{i=1}^{r} \sqrt{(ix)^2 + (ry)^2} + \sum_{j=1}^{r} \sqrt{(rx)^2 + (jy)^2} \\ &\leq A + \sum_{i=1}^{r}(ix + ry) + \sum_{j=1}^{r}(rx + jy) \leq A + 2r^2(x+y)\end{aligned}$$

□

THEOREM 5.1. *For large enough n, the ratio of the cost of the solutions produced by the uniform partioning algorithm to the cost of an optimal solution is*

$$\leq \frac{1+\epsilon}{1-\epsilon} + \frac{6}{1-\epsilon}\left(\frac{k_2}{\alpha} + \frac{k_2}{k_1}\right)\frac{g_1^0}{g_2^* X}$$

with probability approaching 1.

proof. By Lemma 5.1 the switches are evenly distributed with probability approaching 1 as $n \to \infty$. The ratio of the costs is given by the ratio of the expressions in Lemmas 5.3 and 5.2, when the switches are evenly distributed. Taking the ratio of the first term in the upper bound to the lower bound gives

$$\begin{aligned}
\frac{\lceil \alpha/k_1 \rceil \, [g_1^0 + g_1^* d] \, n}{((1-\epsilon)\alpha(n/R)/k_2) \, [(1/2)g_2^0 R + 4g_2^* A]} &\leq \frac{k_2(1+\alpha/k_1)(g_1^0 + g_1^* d)}{(1-\epsilon)\alpha((1/2)g_2^0 + 4g_2^* A/R)} \\
&\to \frac{k_2(1+\alpha/k_1)(g_1^0 + g_1^* d)}{(1-\epsilon)\alpha((1/2)g_2^0 + (1/3)g_2^* dr)} \\
&\to \frac{k_2(1+\alpha/k_1) g_1^0}{(1-\epsilon)\alpha((1/2)g_2^0 + (1/3)g_2^* dr)} \\
&< \frac{6}{1-\epsilon}\left(\frac{k_2}{\alpha} + \frac{k_2}{k_1}\right)\frac{g_1^0}{g_2^* X}
\end{aligned}$$

Taking the ratio of the second term in the upper bound to the lower bound gives

$$\begin{aligned}
&\frac{\lceil (1+\epsilon)\alpha(n/R)/k_2 \rceil \, [(1/2)g_2^0 R + 4g_2^*(A + 2r^2(x+y))]}{((1-\epsilon)\alpha(n/R)/k_2) \, [(1/2)g_2^0 R + 4g_2^* A]} \\
&\quad \leq \frac{(k_2 + (1+\epsilon)\alpha(n/R)) \, [(1/2)g_2^0 + 4g_2^*(A + 2r^2(x+y))/R]}{((1-\epsilon)\alpha(n/R)) \, [(1/2)g_2^0 + 4g_2^* A/R]} \\
&\quad \leq \frac{k_2 + (1+\epsilon)\alpha(n/R)}{(1-\epsilon)\alpha(n/R)} \cdot \frac{A + 2r^2(x+y)}{A} \to \frac{k_2 R}{(1-\epsilon)\alpha n} + \frac{1+\epsilon}{1-\epsilon} \to \frac{1+\epsilon}{1-\epsilon}
\end{aligned}$$

Hence, for large enough n the ratio of the cost of the solution produced by the uniform partitioning algorithm to the cost of an optimal solution is

$$\leq \frac{1+\epsilon}{1-\epsilon} + \frac{6}{1-\epsilon}\left(\frac{k_2}{\alpha} + \frac{k_2}{k_1}\right)\frac{g_1^0}{g_2^* X}$$

with probability approaching 1. □

Note that for large geographic areas $g_2^* X >> g_1^0$ and so the second term in the performance ratio will typically be small. The uniform partitioning algorithm can clearly be generalized to have partitions at successive levels and one would expect than under appropriate conditions, similar asymptotic results could be obtained. Rather than explore this direction, we now introduce a class of algorithms called *hierarchical partitioning* algorithms which are intended to handle more realistic situations and which should perform similarly to uniform partitioning for uniform switch distributions. The algorithms in this class follow the outline shown below.

- *Initialization.* Choose a center vertex C and compute the capacities of direct link groups from C to all other vertices needed to produce a nonblocking star network. Call this star network the *current solution*, and designate the entire rectangle containing the switches an *active region with center* C Define the *core* of the current solution to be the set $\{C\}$.

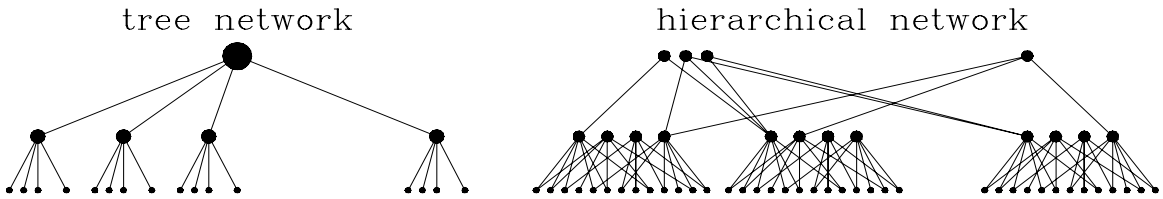

Figure 3: Converting Trees to Hierarchies

- *Refinement Step.* Select any active region R with center c and divide it into two subregions R_1, R_2 using a dividing line perpendicular to the long side of the active region. Attempt to improve the current active region by selecting vertices c_1 and c_2 in R_1 and R_2 and locally modifying the current solution in either of two ways: (1) make c_1 and c_2 children of c, add them to the core and connect all non-core vertices to some core vertex using the cheapest link group that yields a nonblocking tree; (2) make c_1 and c_2 children of the parent of c (assuming c is not the root of the tree), add them to the core and connect all non-core vertices to some core vertex using the cheapest link group that yields a nonblocking tree. Retain the best improvement obtained and call it the current solution, make R_1 and R_2 active and R inactive.

There are many variations that fit the outline, depending on how one selects the original center, the subregions, the subregion centers and how one pairs up non-core vertices to core vertices. A simulation study is now underway to evaluate the performance of hierarchical partitioning. We believe that for networks with a large enough geographic span, the hierarchical partitioning algorithms can be expected to produce solutions close to optimal.

6. Converting Tree Networks to Hierarchies

Tree-structured networks often provide the best solution to a network design problem. Unfortunately, as networks get large, the capacity of the switches required near the root of tree-structured networks may be unrealistically high. There are different approaches one can take to this. One is to replace the over-sized switches in the original design with a collection of smaller switches interconnected in such a way as to yield a nonblocking network which is functionally equivalent to the original switch. Classical techniques can be applied to carry out this strategy. See [11, 15, 16, 20, 21]. Another approach is to replace the tree with a hierarchical interconnection of switches with a similar interconnection structure. This process is illustrated in Figure 3. In this section, we describe how to carry out such a conversion so as to obtain a network that is nonblocking for point-to-point connections.

Let T be an arbitrary tree network with symmetric capacities and links. If T has any internal vertices z with $\alpha(z) > 0$, we augment the tree by adding a new leaf z' connected to z, make $\alpha(z) = 0$ and let $\alpha(z')$ be equal to the original source capacity of z. From this point on, we refer to this augmented tree as T. Now we convert this tree (in which only leaves have non-zero source capacity) to a hierarchical equivalent H of T by replacing each vertex u by a set of vertices $u_0, \ldots, u_{n_u-1}$. We require that if v is the parent of u in T, then

n_v must be an integer multiple of n_u and let $d_u = n_v/n_u$. Also, if v is the parent of u in T and u is not leaf, we join v_i to $v_{\lfloor i/d_u \rfloor}$ for $0 \leq i \leq n_v$. If u is a leaf we join every u_i to every v_j. Finally, if u has termination capacity $\alpha(u)$ in T, we require that $\sum_i \alpha(u_i) = \alpha(u)$. When routing connections in H, we restrict ourselves to paths that correspond to simple paths in T.

We say a vertex in H as depth j in H if the vertex it was derived from in T was at distance j from the root and we call a vertex in H a leaf if it was derived from a vertex that was a leaf in T. Similarly, any vertex derived from a proper ancestor of u in T is referred to as an ancestor of u_i in H. Also, each vertex in H is considered an ancestor of itself. For any switch u_i at depth k in H, let $S_{u_i,j}$ be the set of leaves that are descendants of some depth j ancestor of u_i. Let $d_{u_i,j}$ be the number of edges joining a depth j ancestor of u_i to depth $j-1$ ancestors of u_i and let $d_{u_i,j!} = d_{u_i,j} d_{u_i,j-1} \cdots d_{u_i,1}$. Finally, let $\kappa_{u_i,j}$ be the capacity of the links joining u_i's depth j ancestors to its depth $j-1$ ancestors (they're assumed to be equal). With these definitions, we can show that H is nonblocking with respect to point-to-point connections if for all leaves u_i in H at depth k (for all k),

$$\frac{\alpha(S_{u_i,k}) - B}{\kappa_{u_i,k} - B} d_{u_i,(k-1)!} + \frac{\alpha(S_{u_i,k-1}) - \alpha(S_{u_i,k})}{\kappa_{u_i,k-1} - B} d_{u_i,(k-2)!} + \cdots + \frac{\alpha(S_{u_i,1}) - \alpha(S_{u_i,2})}{\kappa_{u_i,1} - B} < m/2$$

where m is the number of vertices in H corresponding to the root of T.

For cases where all the leaves are at the same depth in H, there is another condition based on purely local considerations. Consider a vertex u in H with edges of capacity $k_0, \ldots, k_{d-1}$ joining it to its parents in H, where $k_0 \leq \cdots \leq k_{d-1}$. Let A be any upper bound on the amount of traffic that can potentially be routed through the edges linking u to its parents (this is bounded by the source capacity of u, if u is a leaf and by the capacity of the edges joining u to its children if u is an internal node). Let r be the largest integer for which

$$\sum_{i=0}^{r} \min\{0, k_i - B\} < A - B$$

(where B is the maximum virtual circuit weight). If $r + 1 < d/2$ we say that u is *locally nonblocking*. If all vertices are locally nonblocking then the entire network is nonblocking for point-to-point connections. If $k_0 = \cdots = k_{d-1}$, the requirement for u to be locally nonblocking simplifies to

$$\left\lfloor \frac{A - B}{k_0 - B} \right\rfloor_{-} < d/2$$

where $\lfloor x \rfloor_{-}$ is the largest integer that is strictly less than x and is called the *subfloor* of x.

7. Closing Remarks

Global traffic constraints are appealing because they require that a network manager provide a minimum of information about user traffic patterns. Since detailed traffic information is often not readily available, this is a significant advantage. Unfortunately global constraints can be so loose that they lead to overly expensive network designs. There is a trade-off

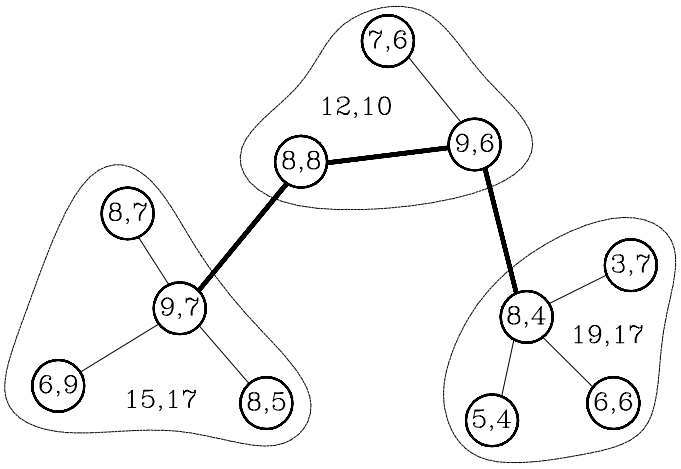

Figure 4: Hierarchical Clustering

here between minimizing the amount of information required of the network manager and providing enough information to yield the most cost-effective designs. We are currently exploring additional constraints that can improve the quality of network designs without placing an undue burden on the network manager.

Hierarchical Clustering. Real networks often form natural hierarchies in which communication tends to be concentrated in subtrees of the hierarchy. This observation leads to a natural generalization of global constraints in which switches are grouped into clusters, clusters are grouped into super-clusters and so forth, with the number of levels of clustering determined by the size of the network and the natural "communities of interest" formed by its users. Given a cluster C at any level in this hierarchical decomposition, a network manager can specify a source capacity $\alpha(C)$ and a sink capacity $\omega(C)$, where the source capacity is an upper bound on the traffic that can leave the cluster and the sink capacity is an upper bound on the traffic that can enter the cluster. This provides a natural way to express traffic constraints among groups of users whose need to communicate is more limited than what would be allowed by global constraints alone. An example of a two level hierarchy is shown in Figure 4 (the numbers within the vertices are their individual source/sink capacities, while the numbers in the clusters represent cluster source/sink capacities). Preliminary simulation studies show that tree-structured networks that reflect the hierarchical clustering can be configured to be nonblocking with a cost that approaches the lower bound. We conjecture that in large networks, this result will hold true with probability approaching one, for random distributions of clusters and switches in the plane.

Distance Bounds. While it is often possible to decompose traffic in a network in a hierarchical fashion, this is not always possible. Often, we have overlapping groupings based on distance, rather than strict clusters. To formalize this, we need a way to express distances among switches. A natural choice is based on physical location in the plane, with Euclidean distances between points being used to define traffic constraints. Given a way to determine distances between vertices, we can generalize

our source and sink constraints by adding a distance parameter. In particular, we let $\alpha_d(u)$ be an upper bound on the traffic from u to vertices whose distance from u is greater than d. The original source capacity can then be written as $\alpha_0(u)$ and the network manager can specify different source capacities associated with longer and longer distances. Sink capacities are generalized in the same way. The natural network designs that arise when traffic capacities are expressed in this way are not tree-structured, but tend to have a mesh interconnection pattern.

Node-Set Pair Constraints. While hierarchical clustering and distance bounds capture important special cases for expressing traffic requirements, it's clear that neither provides a fully general method for expressing traffic constraints. This leads to another form of traffic constraint that generalizes the hierarchical and distance constraints, as well as allowing expression of additional constraints. For any pairs of vertex sets A, B let $\mu(A, B)$ be an upper bound on the traffic from A to B. If we define sets representing clusters and superclusters, then we can represent the hierarchical constraints by node-set pair constraints of the form $\mu(S, V - S)$, where S is the set of vertices in a particular cluster or supercluster and V is the set of all switches. Distance-based constraints can also be expressed naturally. For each switch u, we can let $S_d(u)$ be the set of switches at distance greater than d from u. Then $\mu(\{u\}, S_d(u))$ is equivalent to the distance-based constraint $\alpha_d(u)$. Node-set pairs can also express more general distance-based constraints. For example, in some situations there may be switches that while not separated by large geographic distances are separated by some natural or cultural obstacle (like a mountain range or language difference), that causes less need for communication than could be easily expressed with a strict distance-bound model. In addition, for longer distances, it's most natural to express distance constraints relative to a set of switches rather than for all the individual switches in the set.

References

[1] Angluin, D., L. G. Valiant. "Fast Probabilistic Algorithms for Hamiltonian Circuits and Matchings." In *Journal of Computer and System Sciences 18* , 155–193, 1979.

[2] Fingerhut, J. Andrew. "Designing Communication Networks with Fixed or Nonblocking Traffic Requirements," Washington University Computer Science Department, WUCS-91-55.

[3] Fingerhut, J. Andrew. "Algorithms for Designing Nonblocking Communication Networks with General Topologies," Washington University Computer Science Department, WUCS-MF-92-05

[4] Fingerhut, J. Andrew. "Approximation Algorithms for Hierarchical Nonblocking Communication Networks," Washington University Computer Science Department technical report, WUCS-93-19, 8/93.

[5] Fingerhut, J. Andrew. "Approximation Algorithms for Configuring Nonblocking Communication Networks," Washington University Computer Science Department doctoral dissertation, 5/94.

[6] Fingerhut, J. Andrew, Subhash Suri, Jonathan Turner. "Designing Least-Cost Nonblocking Broadband Networks," In preparation for submission to *SIAM Symposium on Discrete Algorithms*, 7/95.

[7] Gibbens, R. J. and F. P. Kelley. "Dynamic Routing in Fully Connected Networks," *IMA Journal of Mathematical Control and Information*, 1990.

[8] Gersht, Alexander and Robert Weihmayer. "Joint Optimization of Data Network Design and Facility Selection," *IEEE Journal on Selected Areas in Communications*, 12/90.

[9] Ibarra, Oscar and C. E. Kim. "Fast Approximation Algorithms for the Knapsack and Sum of Subset Problems," *Journal of the ACM*, 463–468, 1975.

[10] Imase, Makoto and Bernard Waxman. "The Dynamic Steiner Tree Problem," *SIAM Journal on Discrete Mathematics*, August 1991.

[11] Jordan, B. W., Jr. and G. M. Masson. "Generalized Multistage Connection Networks," *Networks 2*, 1972, 191–209.

[12] Kershenbaum, Aaron, Parviz Kermani and George Grover. "MENTOR: An Algorithm for Mesh Network Topological Optimization and Routing," *IEEE Transactions on Communications*, April 1991.

[13] Minoux, Michel. "Network Synthesis and Optimum Network Design Problems: Models, Solution Methods and Applications," *Networks*, 1989.

[14] Melen, Riccardo and Jonathan S. Turner. "Nonblocking Multirate Networks," *SIAM Journal on Computing*, 4/89.

[15] Melen, Riccardo and Jonathan S. Turner. "Nonblocking Networks for Fast Packet Switching," *Proceedings of Infocom*, 4/89.

[16] Melen, Riccardo and Jonathan S. Turner. "Nonblocking Multirate Distribution Networks," *IEEE Transactions on Communications*, 2/93.

[17] Papadimitriou, C. H. and M. Yannakakis, "Optimization, Approximation and Complexity Classes," *Journal of Computer and System Sciences*, 1991, 425–440.

[18] Turner, Jonathan S. "Design of a Broadcast Packet Network," *IEEE Transactions on Communications,* 6/88.

[19] Waxman, Bernard. "Routing of Multipoint Connections," *IEEE Journal on Selected Areas of Communications*, 12/88.

[20] Witte, E., "The Clos Network as a Multirate Distributor with a Greedy Routing Algorithm," Washington University Computer Science Department, WUCS-92-13.

[21] Yang, Yuanyuan and Gerald Masson. "Nonblocking Broadcast Switching Networks," *IEEE Transactions on Computers*, 1991.