On-line Distributed Traffic Grooming

R. Jordan Crouser
Depts. of Mathematics and Computer Science
Smith College
Northampton, MA

Brian Rice
Dept. of Mathematics
Harvey Mudd College
Claremont, CA

Adrian Sampson
Ran Libeskind-Hadas*
Dept. of Computer Science
Harvey Mudd College
Claremont, CA

***Abstract*—This paper addresses the problem of on-line traffic grooming in WDM paths. Each request consists of a source node, a destination node, and the desired bandwidth for the connection. Connections may be multi-hop, permitting the use of multiple lightpaths. We describe a new distributed on-line algorithm for this problem that is provably wide-sense non-blocking under certain assumptions. Moreover, we use simulations to demonstrate that the algorithm is extremely effective even when some of these assumptions are relaxed.**

I. Introduction

The combination of high speed and bandwidth have made optical networking increasingly attractive for applications ranging from local area to backbone networks. Because the bandwidth of an optical fiber far exceeds that of network nodes, *wavelength division multiplexing (WDM)* is used to partition the optical fiber into separate channels, allowing multiple simultaneous connections to be established over the same fiber.

Furthermore, connection requests typically require only a small fraction of the total capacity of a wavelength. Therefore, *traffic grooming* is used to pack or "groom" multiple connections onto a single wavelength, thereby further enhancing bandwidth utilization.

A *lightpath* comprises a sequence of physical links on which a message may be routed entirely optically on a single wavelength. In a multi-hop network, connections may comprise several contiguous lightpaths, and these lightpaths need not be on the same wavelength. At a node joining two lightpaths, the connection is converted from the optical domain into the electronic domain before being retransmitted onto the next lightpath. This optical-electronic-optical retransmission requires a receiver to terminate the incoming lightpath and a transmitter to initiate the outgoing lightpath. A collection of lightpaths on a physical topology, together with the associated tuning of the transceivers needed to realize these lightpaths, constitute a *virtual topology*.

This paper considers applications in which connection requests arrive on-line. Since requests must be established quickly, we assume that the virtual topology cannot be reconfigured on the fly; network reconfiguration will delay the establishment of the connection and potentially introduce delays to existing connections [1]. Therefore, we assume that the virtual topology must be configured in advance and remains static. When connection requests arrive on-line, they must be routed and groomed with a fast and simple distributed algorithm [2]. Thus, the problem considered in this paper comprises two parts: the construction of a static virtual topology and the design of a distributed on-line grooming algorithm on this topology.

We make several simplifying assumptions in the interest of obtaining fundamental results for this problem. Many of these assumptions can evidently be relaxed and our results can thus be generalized in various ways. For example, while we restrict our attention to bidirectional paths, our results extend naturally to rings and may also generalize to other topologies [3]. Furthermore, we assume that all nodes have the same upper bound on the amount of traffic that they may generate or receive. A heterogeneous network model, while more realistic, results in significant additional complexity and is therefore not considered in this initial study of the problem.

A sequence of connection requests is said to be k*-allowable* if, at any given time, every node is the source of at most k units of traffic and the destination of at most k units of traffic, an extension of the definition from [4]. In other words, k is an upper-bound on the amount of traffic originating from or destined for a node.

We describe a virtual topology for bidirectional paths, a distributed grooming algorithm, and a value k such that:

- *When connections are permanent, the algorithm provably satisfies every k-allowable sequence of requests and is wide-sense nonblocking (i.e. requires no reconfiguration of existing connections).*
- *When connections have finite duration, experimental results suggest that the algorithm has very low blocking probability or, alternatively, requires few reconfigurations of existing connections.*

A number of authors have considered the problem of on-line wavelength assignment and traffic grooming. Bartal and Leonardi [5] and Saengudomlert *et al.* [4] address dynamic wavelength assignment in all-optical networks, but assume that each connection request reserves an entire wavelength and do not allow wavelength conversion. Zhang and Yang [6] consider optimal on-line wavelength assignment in WDM networks with some wavelength conversion abilities, but again do not consider grooming. In [7], Xu, Li and Wang study dynamic routing in multi-fiber WDM networks without wavelength conversion. In [8], Srinivasan and Somani compare the per-

* This work was supported by the National Science Foundation under grant CNS 0451293 to Harvey Mudd College. Please address correspondences regarding this work to Ran Libeskind-Hadas at hadas@cs.hmc.edu.

formance of several dynamic algorithms on WDM networks that implement traffic grooming; however, none of them are distributed. Huang and Dutta's survey of dynamic traffic grooming reviews many centralized algorithms for routing dynamic groomed traffic [2].

While much of the existing literature examines centralized routing, wavelength assignment, and grooming algorithms, distributed algorithms are needed in many contexts [2], [9]. To the best of our knowledge, this work is among the first to examine distributed on-line traffic grooming algorithms.

The remainder of this paper is organized as follows: Section II provides a description of the dynamic traffic grooming problem addressed in this paper, as well as terminology and notation. Section III describes the construction of a virtual topology on a given path network and presents the dynamic grooming algorithm. Section IV provides a proof that under certain conditions, our dynamic grooming algorithm will satisfy any k-allowable sequence of requests and then proves that this result cannot be significantly improved. Section V provides experimental results demonstrating that our algorithm is very effective. Section VI concludes with some directions for future research.

II. Preliminaries and Problem Formulation

In this section, we formally describe the dynamic grooming problem studied in this paper and introduce notation that will be used throughout. The problem can be characterized as follows:

Given:

- A bidirectionally connected path.
- N, the number of nodes in the path.
- T, the number of transceivers at each node.
- A set of W wavelengths denoted $\{\lambda_1, \ldots, \lambda_W\}$, each with capacity C.
- An upper bound, $k \leq C$, on the amount of bandwidth that can be sent by a node or received by a node. Any sequence of requests that satisfies the constraint that each node generates or receives at most k units of bandwidth is said to be *k-allowable.*

Objective:

- Construct a virtual topology over the physical topology by appropriately tuning transceivers.
- Provide a distributed algorithm for grooming any k-allowable configuration of on-line traffic across this virtual topology.

In order to satisfy arbitrary k-allowable connection request sequences, we restrict the values of N, T, k, and C such that

$$N \leq \frac{C}{k}(T)(T+1).$$

This constraint and its justification are further explained in Section IV.

We assume that a *connection request* consists of an ordered pair (i, j) with an associated bandwidth request β. We model a request for greater than unit bandwidth as a contiguous sequence of β unit requests. Thus, without loss of generality, we henceforth restrict our attention to on-line sequences of unit bandwidth requests.

Finally, note that a node v may be the source of up to k units of bandwidth and may be the destination of up to k units of bandwidth. In addition, some traffic may be routed through node v, since multi-hop communication permits a connection to undergo optical-electronic-optical (OEO) conversion at v. This additional traffic does not consume any of the bandwidth allotted to the node by the k-allowability constraint.

The notation used in the remainder of this paper is summarized in Table I.

TABLE I
Summary of Notation

Notation	Description
k	the maximum number of requests sourced by and the maximum number of requests destined for each node
N	the number of nodes in the network
T	the number of transceivers per node
W	the number of wavelengths available
λ_j	a wavelength
C	the capacity of each wavelength
(i, j)	a connection request from node i to node j

III. Virtual Topology and Dynamic Grooming

In this section, we describe a virtual topology on the path, and then introduce a dynamic grooming algorithm on this topology. Without loss of generality, we consider the part of the virtual topology oriented from left to right and restrict our attention to connection requests whose sources are to the left of their destinations. The analogous construction is used to facilitate connections from right to left.

A. Virtual Topology Construction

We construct a virtual topology where every node in the network is connected to each of its neighbors within a prespecified distance r (whose value is established below) by a lightpath. To this end, we tune the transceivers at each node as follows:

First, one transceiver at each node is tuned to the first wavelength, λ_1. This creates a lightpath with grooming capacity C between every pair of adjacent nodes (see Fig. 1).

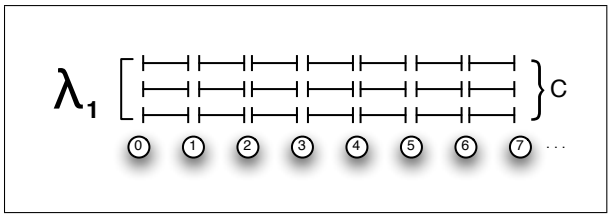

Fig. 1. Each node has a lightpath to the nodes immediately to its left and right.

Next, at each node in $\{0, 2, 4, ...\}$, a transceiver is tuned to λ_2, and at each node in $\{1, 3, 5, ...\}$, a transceiver is tuned to

λ_3, providing a lightpath with capacity C between all pairs of nodes that are distance 2 apart (see Fig. 2).

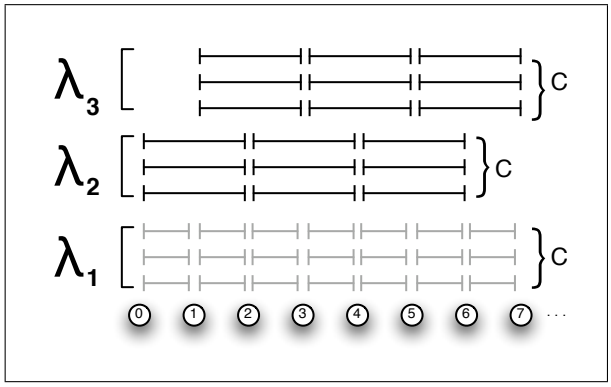

Fig. 2. Each node now has a lightpath to its neighbors at distance 2, using the next pair of wavelengths.

In general, we tune each subset of nodes

$$\{a + mb : m \in \mathbb{Z} \text{ and } 0 \leq a + mb \leq N - 1\}$$

where $b \leq r$ and $a \leq b$ to the next available wavelength, which constructs a lightpath with capacity C between each pair of nodes separated by distance b. Because there are r possible values for b, it follows that we will require r transceivers at each node, and thus we consider $r = T$. From the perspective of an individual node, the virtual topology appears as in Fig. 3.

B. Dynamic Grooming Algorithm

A *connection request* $\mathbf{c}$ is represented as an interval $(L(\mathbf{c}), R(\mathbf{c}))$, where $0 \leq L(\mathbf{c}) < R(\mathbf{c}) \leq N-1$, and signifies a request for one unit of directed bandwidth from node $L(\mathbf{c})$ to node $R(\mathbf{c})$.

A *segment* s can be written as the pair $(L(s), R(s))$ and corresponds to the lightpath from $L(s)$ to $R(s)$ reserved by the virtual topology construction in Section III-A. We define a length function $d(s) = R(s) - L(s)$; recall that all segments have the property $d(s) \leq r$. To satisfy a request, we designate

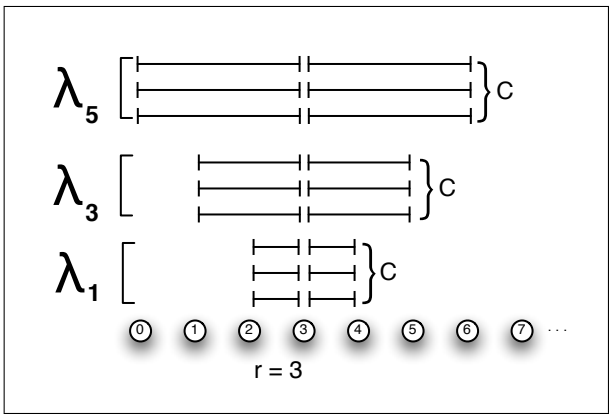

Fig. 3. A node has direct connections to all nodes within local radius, r. In this example, $r = 3$.

Dynamic Grooming Algorithm (DGA)

Input: A connection request $\mathbf{c}$ and a network state function U.
Output: Returns $true$ if $\mathbf{c}$ was satisfied and $false$ otherwise. If it was satisfied, U and $S(\mathbf{c})$ reflect the new network state.

$S(\mathbf{c}) \leftarrow \emptyset$
$s \leftarrow (L(\mathbf{c}), \min\{R(\mathbf{c}), L(\mathbf{c}) + r\})$
while $L(s) \neq R(s)$ **do**
  if $U(s) < C$ **then**
    // s is available to be used by $\mathbf{c}$
    $U(s)$++
    let $S(\mathbf{c})$ contain s
    $s \leftarrow (R(s), \min\{R(\mathbf{c}), R(s) + r\})$
  else
    // a conflict occurs with s
    $s \leftarrow (L(s), R(s) - 1)$
  end
end
return $R(s) == R(\mathbf{c})$ // true iff connection is successfully satisfied

a sequence of segments $S(\mathbf{c})$ such that their concatenation equals the entire interval $\mathbf{c}$. We say that a satisfied connection *uses* each segment in the sequence $S(\mathbf{c})$.

The state of the network is represented by a function U mapping segments to integers. $U(s) = u$ indicates that s is used by u satisfied connections. Before any connection requests are satisfied, C units of bandwidth are reserved for communication between any two nodes whose distance is at most r by the construction in Section III-A. This is signified by letting $U(s) = 0$ initially for every s. In any valid network state, $U(s) \leq C$ for all segments s because no segment may be used by more than C connections.

When a request $\mathbf{c}$ arrives on-line, it is handled by the Dynamic Grooming Algorithm (DGA), shown above. The DGA is a greedy algorithm that attempts to use the longest possible segments first. Recall that all segments have length between 1 and r. In general, the algorithm first tries to use a segment of length r. If no such segment is available, the algorithm next tries a segment of length $r-1$ and so forth until the first available segment is found or, if no available segment is found, the algorithm fails to satisfy the connection. Recall that since each segment may be used by up to C connections, a segment is unavailable if and only if $U(s) = C$.

Once the algorithm finds the longest available segment, it greedily commits to using that segment and never backtracks to replace that segment with one of different length. Although such backtracking might appear be advantageous, it can result in exponential worst-case running time. Moreover, we prove below that our greedy algorithm is essentially optimal.

The algorithm repeats this process starting from the right endpoint of the previously used segment. When the algorithm reaches a node that is within distance r or less from the

destination, the algorithm begins by considering the segment of length equal to this distance.

Henceforth, when the DGA is invoked on a connection $\mathbf{c}$, we say that it *attempts to satisfy* $\mathbf{c}$. When the algorithm reaches the top of the loop on line 4, we say that it *considers* the segment s. If the conditional statement branches to line 8, we say that a *conflict* occurs with segment s. Note that the occurrence of a conflict with s implies that $U(s) \geq C$; furthermore, because no segment may be used by more than C connections, $U(s) = C$ in the case of a conflict.

Note that the DGA does not explicitly assign wavelengths to segments or connections. By the virtual topology construction in Section III-A, bandwidth for each segment is reserved on a specific wavelength; that is, the selection of wavelengths is implied by the algorithm's selection of segments.

The DGA can be implemented in a distributed fashion. The only network state required by the algorithm is the function U. Because the domain of U is the set of segments in the network, its values for a given segment s may be stored at its endpoint nodes $L(s)$ and $R(s)$. A connection request propagates from source to destination, possibly traversing multiple lightpaths; at every node that terminates a lightpath, only local state information is required. Finally, note that the total running time of the DGA, per connection request, is bounded by $O(rN) = O(TN)$ since each of the N nodes in the network examines at most $r = T$ possible segments.

IV. Analysis of the DGA

In this section, we prove that the DGA, when run on a path such that $N \leq \frac{C}{k}(r)(r+1)$, can satisfy any k-allowable sequence of connection requests assuming that the requests have infinite duration. Additionally, we demonstrate that no algorithm can satisfy arbitrary k-allowable connection request sequences if $N \geq \frac{C}{k}(r)(r+1)+2$ given the proposed virtual topology. Finally, we propose strategies for grooming traffic with finite-duration connections.

A. Traffic Satisfaction Guarantee

We first prove that if the algorithm considers a segment of length less than r (that is, tests whether the segment may be used in satisfying a connection), then a conflict (an inability to use a segment) must have previously occurred.

Lemma 1. *If the DGA considers segment s while attempting to satisfy connection $\mathbf{c}$ where $R(s) \neq R(\mathbf{c})$ and $d(s) < r$, the algorithm previously encountered a conflict with segment $(L(s), R(s)+1)$.*

Proof: Assume, by way of contradiction, that s is the result of an assignment in line 2 or 7 of the algorithm. In either case, $R(s) = R(\mathbf{c})$ or $R(s) = L(s) + r$. If $R(s) = R(\mathbf{c})$, we contradict the condition that $R(s) \neq R(\mathbf{c})$. If $R(s) = L(s)+r$, then $d(s) = r$, another contradiction. Our initial assumption must therefore be false and we conclude that s is the result of the only remaining assignment in the algorithm, that on line 9, and $s = (L(t), R(t) - 1)$ for another segment t. It follows that $t = (L(s), R(s)+1)$. Line 9 is only executed in the case that a conflict occurs with this segment t. ■

We can also prove the result of Lemma 1 given an alternate hypothesis. Rather than assuming that the segment considered is not at the destination of a connection, we base our proof on the hypothesis that a conflict occurs at the segment.

Corollary 1. *If the DGA encounters a conflict with segment s, and $d(s) < r$, then a conflict occurred previously with segment $(L(s), R(s)+1)$.*

Proof: Because a conflict occurred at s, it follows that $U(s) = C$. Consider the C previously-satisfied connections that use s as well as the connection the algorithm was attempting to satisfy when the conflict occurred. Of these $C+1$ connections, at most k may have $R(s)$ as their destination. Because $k \leq C < C+1$, at least one connection's destination is not $R(s)$; call this connection $\mathbf{c}'$. It follows that $R(\mathbf{c}') > R(s)$.

The algorithm considered s while attempting to satisfy $\mathbf{c}'$ where $R(s) \neq R(\mathbf{c}')$, so the hypotheses of Lemma 1 are satisfied, and therefore the DGA encountered a conflict with $(L(s), R(s)+1)$. ■

Under the hypothesis of either Lemma 1 or of Corollary 1, we apply induction to prove the existence of a series of conflicts.

Lemma 2. *Consider a segment s where $d(s) < r$. If the DGA encountered a conflict with s, or $s \in S(\mathbf{c})$ where $R(s) \neq R(\mathbf{c})$, then, for every ℓ where $1 \leq \ell \leq r - d(s)$, a conflict occurred previously with a segment $(L(s), R(s)+\ell)$, denoted s_ℓ.*

Proof: The proof is by induction on ℓ.

Basis: $\ell = 1$. If a conflict occurred at s, Corollary 1 applies. Otherwise, if $s \in S(\mathbf{c})$, line 6 executed when the DGA was run with input $\mathbf{c}$. The algorithm considered s while attempting to satisfy $\mathbf{c}$ and Lemma 1 applies. In either case, a conflict occurred with s_ℓ.

Inductive Step: Assume that a conflict occurred at s_ℓ where $1 \leq \ell < r - d(s)$. It follows that $d(s_\ell) = (R(s)+\ell) - L(s)$ is less than $(R(s)+r-d(s)) - L(s)$, which equals r. Corollary 1 applies and a conflict occurred with $s_{\ell+1}$. ■

We now apply these lemmata to obtain our main result, which guarantees the satisfaction of any k-allowable traffic sequence under the aforementioned constraints.

Theorem 1. *Consider a unidirectional path network with N nodes such that the grooming capacity for each wavelength is C. If $k \leq C$ and*

$$N \leq \frac{C}{k}(r)(r+1),$$

then executing the DGA satisfies any k-allowable sequence of connection requests with infinite duration.

Proof: Assume, by way of contradiction, that the DGA returns false (*fails*) when invoked on $\mathbf{c}$ where $\mathbf{c}$ is an element of a k-allowable connection request sequence on a network satisfying the given constraints.

The DGA fails only when it encounters a conflict with an interval s where $d(s) = 1$. By Lemma 2, conflicts previously occurred with each of the $r-1$ other segments whose left endpoint is $L(s)$. Each of the r segments (including s) with left endpoint $L(s)$ must therefore be used by C previously-satisfied connections. No single connection may use two segments with equal left endpoints, so there must be $C \cdot r$ distinct connections that use a segment with left endpoint $L(s)$. Let D be the set containing these $C \cdot r$ connections along with the connection the algorithm was attempting to satisfy when the DGA failed.

Let $E = \{\mathbf{e} \in D \mid L(\mathbf{e}) \neq L(s)\}$, the subset of existing connections (including the current connection request) that do not begin at L(s), but use a segment that does. Note that $|E| = (Cr+1) - |D-E|$. The number of remaining connections $|D-E|$, all of whose sources are $L(s)$, is at most k. Because $k \leq C$, it follows that $|D-E| \leq C$, and therefore $|E| \leq Cr+1-C = C(r-1)+1$.

The connections in E each use a segment with right endpoint $L(s)$. There are r such segments: $(L(s)-i, L(s))$, $1 \leq i \leq r$. It follows that each segment with right endpoint $L(s)$ is used by at least one connection in E. Thus, Lemma 2 applies for each segment $(L(s)-i, L(s))$, proving the existence of conflicts with every segment in

$$I = \{(L(s)-i, L(s)+j) \mid 1 \leq i \leq r \text{ and } 1 \leq j \leq r-i\}.$$

Let F be the set of connections that use a segment in I. Because every segment in I contains the interval $(L(s)-1, L(s))$, each segment in I must be used by C distinct connections in F. It therefore follows that $|F| = C|I| = C\sum_{i=1}^{r}(r-i) = C\frac{(r)(r-1)}{2}$.

By definition, no connection in F uses a segment whose left or right endpoint is $L(s)$, and therefore F and D are disjoint. Thus, $|D \cup F| = |D| + |F| = (C \cdot r + 1) + C\frac{(r)(r-1)}{2} = C\frac{(r)(r+1)}{2} + 1$.

Because $C\frac{(r)(r+1)}{2}+1$ connection requests have been made, there are $2(C\frac{(r)(r+1)}{2}+1) = C \cdot r(r+1) + 2$ connection endpoints on the network. Note that the source of every connection in $D \cup F$ is at most $L(s)$ and the destination of every such connection is at least $L(s)+1$. Thus, each node is exclusively a source or exclusively a destination of connections in $D \cup F$ and, because the request sequence is k-allowable, every node is an endpoint of at most k connections in $D \cup F$. The network must contain at least $\lceil \frac{1}{k}(C \cdot r(r+1)+2) \rceil = \lceil \frac{C}{k}(r)(r+1) + \frac{2}{k} \rceil$ nodes.

This contradicts the assumption that $N \leq \frac{C}{k}(r)(r+1)$. Thus, the initial hypothesis must be false, and therefore no failure can occur under any k-allowable request sequence. ■

B. Tightness

We now show that it is impossible to significantly improve on the satisfying capacity of the DGA. We have shown that any k-allowable sequence of requests can be satisfied by our algorithm provided that $N \leq \frac{C}{k}(r)(r+1)$. In fact, for any $N \geq \frac{C}{k}(r)(r+1)+2$, there is a k-allowable sequence of requests that is unsatisfiable regardless of the algorithm used.

Such a sequence can be constructed by requesting all connections such that, for every connection $\mathbf{c}$ in the sequence, $L(\mathbf{c}) \leq \lfloor \frac{N}{2} \rfloor - 1$ and $R(\mathbf{c}) \geq \lfloor \frac{N}{2} \rfloor$, and the sequence remains k-allowable. The order in which these connections are requested is insignificant. Because there are $\lfloor \frac{N}{2} \rfloor$ possible sources for connections, and at least as many possible destinations for those connections, $f = \lfloor \frac{N}{2} \rfloor k$ requests are made.

Assume, by way of contradiction, that this sequence of f connection requests is satisfiable by some traffic grooming algorithm given the virtual topology constructed in Section III-A. Because of our bound on N,

$$\begin{aligned} f &\geq \left\lfloor \frac{1}{2}\left(\frac{C}{k}(r)(r+1)+2\right)\right\rfloor k \\ &= \left\lfloor \frac{C}{2k}(r)(r+1)+1 \right\rfloor k \\ &= \left\lfloor \frac{C}{2k}(r)(r+1) \right\rfloor k + k \end{aligned}$$

The fractional part of the floored quantity is strictly less than 1. We rewrite $\lfloor \frac{C}{2k}(r)(r+1) \rfloor$ as $\frac{C}{2k}(r)(r+1) - \alpha$ where $\alpha < 1$. Thus, the inequality above can be written as

$$\begin{aligned} f &\geq \left(\frac{C}{2k}(r)(r+1) - \alpha\right)k + k \\ &= \frac{C}{2}(r)(r+1) - \alpha k + k \end{aligned}$$

Because $\alpha < 1$, $\alpha k < k$, and

$$\begin{aligned} f &> \frac{C}{2}(r)(r+1) - k + k \\ &= \frac{C}{2}(r)(r+1) \end{aligned}$$

Our k-allowable connection sequence thus consists of more than $\frac{C}{2}(r)(r+1)$ connections containing the interval $\left(\lfloor \frac{N}{2} \rfloor - 1, \lfloor \frac{N}{2} \rfloor\right)$. Each of these connections must use a segment whose left endpoint is at most $\lfloor \frac{N}{2} \rfloor - 1$ and whose right endpoint is at least $\lfloor \frac{N}{2} \rfloor$.

For a given length ℓ, there are ℓ segments s where $d(s) = \ell$ that satisfy the above property: $\left(\lfloor \frac{N}{2} \rfloor - i, \lfloor \frac{N}{2} \rfloor + \ell - i\right)$ where $1 \leq i \leq \ell$. Over all $1 \leq \ell \leq r$, then, there are $\sum_{i=1}^{r} i = \frac{r(r-1)}{2}$ segments that contain the interval $\left(\lfloor \frac{N}{2} \rfloor - 1, \lfloor \frac{N}{2} \rfloor\right)$. Because each segment may be used by at most C connections, at most $\frac{C}{2}r(r-1)$ connections contain this interval, contradicting the assertion that $f > \frac{C}{2}(r)(r+1)$. This contradicts our assumption that this k-allowable connection request sequence is satisfiable given our virtual topology.

We have thus shown that our virtual topology construction cannot satisfy arbitrary k-allowable request sequences when $N \geq \frac{C}{k}(r)(r+1)+2$. Because it can satisfy arbitrary k-allowable request sequences when $N \leq \frac{C}{k}(r)(r+1)$, the performance of the DGA is close to optimal for this virtual topology.

These results apply only to the virtual topology proposed in Section III-A. However, we conjecture that in order to satisfy arbitrary k-allowable request sequences, the maximum number of nodes is within a constant factor of $\frac{C}{k}(r)(r+1)$ regardless of the static virtual topology used.

C. Finite-Duration Traffic

As shown in Section IV-A, the DGA is guaranteed to satisfy any k-allowable request sequence if the connections are permanent. However, if connections have finite duration, the proof does not apply. Even if the set of active connections remains k-allowable, we have no satisfaction guarantee (as in Theorem 1) if connections can depart. In fact, it is possible to construct such a sequence of connection requests that the DGA is unable to satisfy if connections terminate in finite time. This necessitates a means of dealing with such failures when they occur.

We propose two methods for coping with failures that may occur when handling finite-duration traffic. The first method is simply to block any requests which cause a failure. Experimental results provided in the next section show the blocking probability to be extremely small in general (at most 1 in 10^8 for sufficiently large values of the network parameters). Because blocking has relatively negligible overhead, does not impact existing connections, and occurs infrequently using the DGA, we believe that compromising the non-blocking behavior is suitable for many practical applications.

Alternatively, in situations where we must maintain the ability to satisfy any k-allowable request sequence without blocking, an approach that reconfigures connections may be more desirable. When a failure occurs, all existing connections on the network are temporarily taken down and re-groomed using the DGA from left to right. To accomplish this, a message is broadcast to all nodes to tear down their connections. Then, the leftmost node invokes the DGA to reestablish its connections. When those connections are completed, the next node reestablishes its connections. This process is repeated until all nodes have completed reconfiguration. Because the set of active connections is k-allowable, the proof in Section IV-A guarantees that they will all be successfully reestablished.

The drawback of this method is that it requires brief network downtime whenever a failure occurs. However, experimental data suggests that the probability of incurring such a connection reestablishment event is extremely small.

V. Experimental Results

We constructed a path network simulator capable of modeling sequences of connection arrivals and departures, employing an implementation of the DGA to satisfy requests. Connections arrived randomly such that the length of time between each pair of connections followed a negative exponential distribution. Arriving connections were chosen randomly such that the set of active connections remained k-allowable. The duration of each connection was also chosen from a negative exponential distribution.

We assume that the activity on the network is proportional to the number of nodes, N, and to the bandwidth constraint, k. Thus, the mean of the probability distribution used to select connection duration is proportional to Nk times that of the distribution selecting the delay between connection arrivals; that is, $\mu_{\text{duration}} = \rho N k \cdot \mu_{\text{delay}}$ for some proportionality constant, ρ. In our simulations, ρ took the values $\frac{1}{2}$, $\frac{3}{8}$, and $\frac{1}{4}$ to model three different load patterns. Letting $\rho = \frac{1}{2}$ induces an average of k live connections per node, an appropriate full-load model, but other values of ρ are useful when considering lighter loads.

TABLE II
Summary of Experimental Results

T	C	k	N	ρ	blocks / million	reconfigurations / million
3	2	1	24	1/2	0.64	0.48
3	2	1	24	3/8	0.24	0.16
3	2	1	24	1/4	0.04	0.04
3	2	2	12	1/2	399.2	252.28
3	2	2	12	3/8	195.44	147.56
3	2	2	12	1/4	38.12	30.24
3	4	4	12	1/2	9.702	6.34
3	4	4	12	3/8	2.63	2.238
3	4	4	12	1/4	0.118	0.082
3	8	8	12	1/2	0.022	0.018
4	2	2	20	1/2	24.835	18.027
4	2	2	20	3/8	7.13	5.806
4	2	2	20	1/4	0.416	0.378
4	4	4	20	1/2	0.037	0.026
5	2	2	30	1/2	1.041	0.885
5	2	2	30	3/8	0.146	0.132
6	2	2	42	1/2	0.03	0.022

Simulation parameters and frequencies of failures (blocks or reconfigurations) per million connection requests are given. Results are only shown for simulations where the failure frequencies are at least 1 in 10^8.

Both of the methods described above for dealing with failures were implemented. Under the first policy, connections that could not be satisfied were blocked and we recorded the number of blockings; under the second, all live connections were removed and re-established when the DGA encountered unresolvable conflicts and number of reconfigurations was recorded. We tested $T \in \{3, \ldots, 8\}$ with the following pairs (C, k): $(2,1); (2,2); (4,1); (4,2); (4,4); (8,4); (8,8); (16,8); (16,16)$. In every case, we set $r = T$ and let N be the maximum value guaranteed by Theorem 1.

For each configuration of the above parameters, several request sequences were generated. In general, for every set of parameter values, we simulated 10 request sequences of 100 million connections each. For $\rho = \frac{3}{8}$ and $\rho = \frac{1}{4}$, we executed only 5 runs of the same size. In the cases where $T = 3$ and $C = 2$, each run consisted only of 5 million connections because of computational resource constraints.

The results of our tests are summarized in Table II. For brevity, we omit configurations in which blocking or reconfiguration occurred less often than once per 100 million connections. In the vast majority of the omitted configurations, our simulations generated zero blocked connections or reconfigurations.

The probabilities of blocking and reconfiguration rates quickly become very small as T and C increase. In particular, in cases where $C > k$, nearly every set of parameter values yielded negligible failure rates. Lighter average traffic loads (smaller values of ρ) also yield vastly reduced failure rates, in some cases by two orders of magnitude.

While unresolvable conflicts are consistently less common under the reconfiguration model than under the blocking model, we note that the greater cost involved in reestablishing the entire set of active connections partially counteracts this advantage.

For configurations with $T = 7$ or $T = 8$, only one blocked connection was observed in billions of simulated connection requests, indicating that, for larger networks, the non-blocking property of the DGA is nearly preserved under a finite-duration traffic model.

VI. Conclusion

In this paper, we have described a distributed on-line traffic grooming algorithm for paths and a static virtual topology to support it. We have proven that this algorithm can satisfy arbitrary k-allowable sequences of permanent connection requests under reasonable conditions. When connections have finite duration, experimental results have shown that our algorithm very rarely encounters unresolvable conflicts; furthermore, simple policies exist for coping with these conflicts if they do occur.

Our analytical results have shown that our algorithm is nearly optimal given the proposed virtual topology, but we conjecture that it is asymptotically optimal for any fixed virtual topology. Future work might attempt to produce theoretical bounds on the number of satisfied connections that may be guaranteed on a path, independent of virtual topology.

While we have restricted our analysis to paths, the DGA extends naturally to bidirectional ring networks with minor adaptation [3]. Similar results for other physical topologies are interesting open problems. Other generalizations, including heterogeneity in the traffic matrix (i.e. extending from a fixed k bandwidth upper-bound to one that varies from node to node), merit further study.

Finally, while our experimental results regarding finite-duration requests are very promising, we have not provided theoretical bounds on the probability of blocking or reconfiguration. Future research may also explore this area.

Acknowledgments

We would like to thank Prof. Rudra Dutta and Prof. Biswanath Mukherjee for their useful comments and suggestions.

References

[1] S. Huang and R. Dutta, "Research problems in dynamic traffic grooming in optical networks," *Proceedings of First International Workshop on Traffic Grooming, San Jose, October, 2004*, 2004.

[2] ——, "Dynamic traffic grooming: The changing role of traffic grooming," *Communications Surveys & Tutorials, IEEE*, vol. 9, no. 1, pp. 32–50, 2007.

[3] J. Crouser, B. Rice, A. Sampson, and R. Libeskind-Hadas, "On-line distributed traffic grooming in paths and rings," Harvey Mudd College, Department of Computer Science, Tech. Rep., 2007.

[4] P. Saengudomlert, E. Modiano, and R. Gallager, "Dynamic wavelength assignment for WDM all-optical tree networks," *IEEE/ACM Transactions on Networking (TON)*, vol. 13, no. 4, pp. 895–905, 2005.

[5] Y. Bartal and S. Leonardi, "Online routing in all-optical networks," *Proceedings of 24th International Colloquium on Automata, Languages and Programming (ICALP97)*, pp. 516–526, 1997.

[6] Z. Zhang and Y. Yang, "On-line optimal wavelength assignment in WDM networks with shared wavelength converter pool," *IEEE/ACM Transactions on Networking (TON)*, vol. 15, no. 1, pp. 234–245, 2007.

[7] S. Xu, L. Li, and S. Wang, "Dynamic routing and assignment of wavelength algorithms in multifiber wavelength division multiplexing networks," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 10, pp. 2130–2137, 2000.

[8] R. Srinivasan and A. Somani, "Dynamic routing in WDM grooming networks," *Photonic Network Communications*, vol. 5, no. 2, pp. 123–135, 2003.

[9] R. Melhem, S. Li, and T. Znati, "Minimizing wavelength conversions in WDM path establishment," *Photonic Network Communications*, vol. 3, no. 3, pp. 197–211, 2001.