Scheduling Algorithms for Providing Flexible, Rate-Based, Quality of Service Guarantees for Packet-Switching in Banyan Networks

Constantine Caramanis[1], Michael Rosenblum[†], Michel X. Goemans[‡], Vahid Tarokh[§]
[1]Department of Electrical Engineering, Massachusetts Institute of Technology, Email: cmcaram@mit.edu
[†]Department of Mathematics, Massachusetts Institute of Technology, Email: mrosen@math.mit.edu
[‡]Department of Mathematics, Massachusetts Institute of Technology, Email: goemans@math.mit.edu
[§]Division of Engineering and Applied Sciences, Harvard University, Email: vahid@deas.harvard.edu

***Abstract*— We consider the problem of providing flexible, rate-based, quality of service guarantees for a particular class of multistage switch networks that includes Banyan networks. We focus on solving a type of on-line, traffic scheduling problem, whose input at each time step is a set of desired traffic rates through the switch network. These traffic rates in general cannot be exactly achieved since they treat the incoming data as fluid, that is, they assume arbitrarily small fractions of packets can be transmitted at each time step. The goal of the traffic scheduling problem is to closely approximate the given sequence of traffic rates by a sequence of switch uses throughout the network in which only whole packets are sent. The focus of this paper is bounding the costs incurred in using such an approximation, in terms of the additional buffer size, called *backlog*, required.**

Our contributions in this paper apply to a class of multistage switch networks that includes Banyan networks. We first prove that bounded backlog results of the type developed in Rosenblum et al. [1] do not exist for arbitrary switch networks or even for 4×4 or larger Banyan networks. However, if the switch network is allowed to run faster than the input line rates, it is possible to maintain bounded backlog. We prove that if speedup s is sufficient to maintain bounded backlog for any constant fluid policy, then speedup s is also sufficient to maintain bounded backlog for any time-varying fluid policy. We proceed to bound the speedup required to keep backlog bounded in terms of the number of stages in the multistage switch network. In particular, we give a polynomial time algorithm that for a Banyan network with N input ports keeps backlog bounded using speedup at most $\log_2 N + 1$.

I. Introduction and Related Work

Multistage switching fabrics, based on the interconnection of small switch components, allow for the efficient, modular construction of high-performance routers. We look at the resources, in terms of buffer size and switch speedup, required to provide flexible, rate-based, quality of service guarantees for multistage switch networks.

[1]This material is based upon research partly supported by a National Science Foundation Graduate Fellowship.

[†], [§] This material is based upon research supported by the National Science Foundation under the Alan T. Waterman Award, Grant No. CCR-0139398.

[†],[‡] This material is based upon research partly supported by the National Science Foundation under contracts ITR-0121495 and CCR-0098018.

[1],[†],[‡],[§] Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Numerous approaches to scheduling for switches exist in the literature both for crossbar switches and for general switch networks. In one approach, the designer first ignores the packet nature of traffic and constructs a schedule under the assumption that packets can be broken into arbitrarily small pieces and sent at different time slots (e.g. [1]–[14]). At each step in the schedule the designer specifies the number of fractional packets sent and received, such that the usage of each link in the network is at most 100%. This schedule is referred to as a *fluid policy*. Next, the designer constructs a *packetized policy*, which approximates the behavior of the fluid policy in order to send packet data.

In [12], Tabatabaee, Georgiadis, and Tassiulas consider the problem of packetizing time-varying fluid policies in an $N \times N$ crossbar switch. In [1], the additional buffer requirement due to approximating a fluid schedule with one that sends only whole packets is considered. *Backlog* is introduced as a metric for how much additional buffer space is used by a packetized policy than is used by the fluid policy it is emulating. A packetizing algorithm is presented that guarantees 100% throughput with a buffer requirement of at most $(N+1)^2/4$ packets per input port with no speedup.

It turns out that the design of scheduling algorithms for multistage switch networks is significantly more difficult than for crossbar switches, because of internal collisions: packets routed to the same outgoing link of an individual switch element compete for use of that link.

Our focus is constructing algorithms that approximate fluid schedules so as to provide quality of service guarantees. Specifically, we focus on bounding the costs incurred in using such an approximation, in terms of the additional buffer size, or *backlog*, required.

The analysis of packetizing fluid policies in this paper, as in most of the related work cited above, focuses on worst-case bounds rather than probabilistic bounds. Thus the bounds on backlog proved here are universal in that they do not depend on any statistical properties of the input traffic. Furthermore, these bounds on required, additional buffer size are not asymptotic; they apply to time increments of any finite duration.

While our results extend to any multistage switch network with the property that there is a unique-path between any input–output pair, here we focus on bounding backlog in Banyan networks.

II. Notations and Definitions

A. Structure of Banyan Networks

Banyan networks have been studied extensively in the literature due to their parallel capacity, modularity, expandability, and because they lend themselves to efficient implementation (see for instance, [15], [16], and references therein). Figure 1 depicts a 16×16 Banyan network.

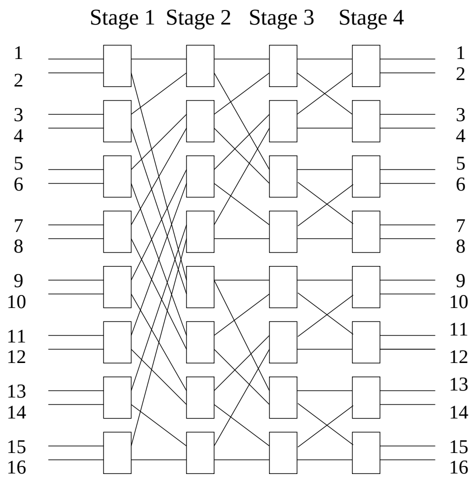

Fig. 1. A 16 by 16 Banyan Network built from 2 by 2 crossbar switch components.

In order to establish the notation, we consider an $N \times N$ Banyan network with $\log_2 N$ stages with N^2 input–output pairs. One of the defining properties of the Banyan switch fabric is that there is a unique path linking any input-output pair. Since we consider Banyan networks without buffers in the intermediate stages, this means that if input i is transmitting to output j, then any input-output pair (k, l) whose (unique) path shares at least one link with the path from i to j, is blocked. We refer to the set of input-output pairs (k, l) that are blocked by pair (i, j), as the neighborhood of (i, j). Therefore at any time, if (i, j) is utilized, then no other input-output pair in its neighborhood can be utilized.

We can visualize this blocking relationship in a graph-theoretic context. We define the *link graph* $G = (V, E)$ of a unique-path switch network (and in particular for the Banyan network) as follows: the link graph has a node for every input-output pair (i, j). Two nodes $(i, j), (k, l)$ are connected by an edge in the graph, if the unique path from input i to output j intersects the path from k to l. In Figures 2 and 3 we show the 4×4 Banyan network, and the associated link graph G. Consider link 2 in Figure 2. Link 2 is required for any transmission from input 2 to outputs 1,2,3 or 4. Therefore in a packetized model, at most one of these four transmissions can occur per time period. In the link graph, this constraint is represented

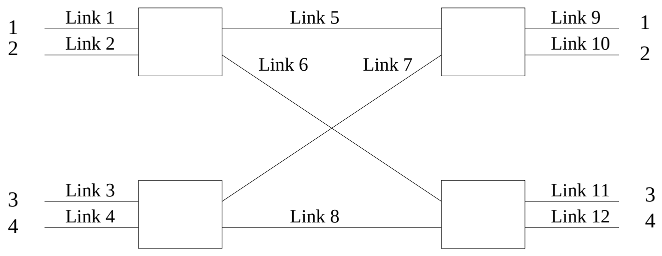

Fig. 2. A 4×4 Banyan Network.

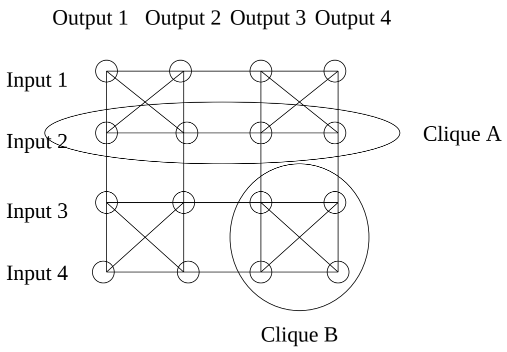

Fig. 3. The link graph corresponding to the above 4×4 Banyan Network. Note that all rows and columns actually form a clique. Edges are omitted to avoid cluttering.

by a clique[1] among nodes $\{(2,1), (2,2), (2,3), (2,4)\}$, i.e. those nodes are all interconnected as in clique A in Figure 3. Similarly, link 8 is required for transmission from 3 to 3, 3 to 4, 4 to 3, and 4 to 4, so among these input-output pairs, at most one transmission can take place. In the link graph, we have a clique among nodes $\{(3,3), (3,4), (4,3), (4,4)\}$. This corresponds to clique B in the figure. In general, each link in an $N \times N$ Banyan network corresponds to a constraint on N input-output pairs, which is represented by a clique in the link graph G (since the pairs that use a particular link all mutually block each other). A valid one-time transmission, therefore, is one which requires the use of each link at most once. In the case of fractional transmissions, or fluid policies, a valid policy is again one where at each step, the link usage totals to at most one for each link. Therefore, a one-time transmission schedule sending x_{ij} from input i to output j is valid if for each clique Q in the graph G, the total transmission corresponding to the nodes in the clique is at most one:

$$\sum_{(i,j) \in Q} x_{ij} \leq 1. \tag{1}$$

We see that the structure of link graph G (which is derived from the switch-network topology) has an intimate connection with the bounded backlog trackability of the switch network.

B. Traffic Model

We consider input queued Banyan networks with a total of N input ports at the first-stage, with N output ports at the last-

[1]A clique is a complete subgraph.

stage, and with $\log_2 N$ stages. See Fig. 1 for an example. The switch network operates in discrete time, and all packets are assumed to have the same size. The line rate (capacity) at the first-stage input ports is one packet per time step. The switch uses virtual output queueing to avoid head-of-line blocking.

A *Fluid Policy* for an $N \times N$ Banyan network is a sequence of transmissions specifying the number of fractional packets that ideally would be sent over the multistage switch. This is represented by a sequence of $N \times N$ non-negative-valued, *fluid matrices* $\{F^{(k)}\}_{k>0}$, where $F_{ij}^{(k)}$ represents the fraction of a packet sent from input i of the first stage to output j of the last stage at time step k. Each fluid matrix must satisfy the constraint that the total fluid traversing any link in the Banyan network is at most 1. A *constant* fluid policy is a fluid policy where for some fluid step F, $F^{(k)} = F$ for all k.

A *Packetized Policy* for an $N \times N$ Banyan network is a sequence of transmissions where only whole packets are sent and received at each time step. It is represented by a sequence of $N \times N$ *packetized matrices*, $\{P^{(k)}\}_{k>0}$ where $P_{ij}^{(k)}$ has value 1 if a packet is sent from input i of the first-stage to output j of the last-stage at time step k; otherwise it has value 0. Each packetized matrix must obey the constraint that for each link in the network, at most one packet traverses it; in other words, no packet collisions are tolerated in the packetized policy. Note that a $\{0,1\}$-valued matrix is a valid packetized matrix if and only if the entries with value 1 form a stable set[2] in the link graph of the switch network.

It is convenient to record, for each first-stage input port i and last-stage output port j, the difference between the cumulative number of fractional packets scheduled by the fluid policy up to and including time k, and the cumulative number of whole packets sent by the packetized policy up to and including time k. This information is stored in the $N \times N$ matrix $C^{(k)}$, for $k \geq 0$. In particular, $C^{(0)} := \mathbf{0}$, and $C^{(k)} := \sum_{l=1}^{k}(F^{(l)} - P^{(l)})$ for $k \geq 1$. For time step k, and for a set of pairs of first-stage input ports and last-stage output ports, we define their *backlog* to be the sum of corresponding entries in $(C^{(k)})^+$.[3] Following Tabatabaee et al. [12], we require that a packet can only be scheduled in a packetized matrix if some fluid for that packet has been scheduled at or before the current time step by the fluid policy. In other words, the packetizing algorithm can only schedule a packet at time t from first-stage input port i to last-stage output port j if $C_{ij}^{(t-1)} + F_{ij}^{(t)} > 0$.

We focus on solving the on-line, traffic scheduling problem, whose input at each time step is the corresponding fluid matrix from a fluid policy. The goal at each time step is to choose a packetized matrix so that backlog remains bounded for all time steps. If a packet-scheduling algorithm maintains bounded backlog for a given fluid policy, we say that it *tracks* the given fluid policy *with bounded backlog*.

In general, a switch is said to use speedup $s \geq 1$ if it processes packets s times as fast as the input line rate. In this paper, it will be more convenient to use an equivalent definition; we say that a scheduling algorithm uses *speedup* $s \geq 1$ if for each fluid policy, each of its fluid matrices is multiplied by $1/s$. We say that an algorithm uses *no speedup* if $s = 1$.

[2]A set of nodes in a graph is called a *stable set* if there are no edges between any of these nodes.

[3]The positive part of a matrix M is denoted M^+, where $M_{ij}^+ := \max\{M_{ij}, 0\}$.

III. Results and Contributions

Our contributions in this paper begin with a negative result. Specifically, we prove that bounded backlog results of the type developed in [1] do not exist for arbitrary switch networks. In fact, this is true even for Banyan networks, *for the simple case of a constant fluid policy*. This motivates us to develop the concept of *necessary speedup* for tracking with bounded backlog. Specifically, given a Banyan network (or any multistage unique-path switch network) we seek the minimum amount of speedup required so that any fluid policy can be tracked by a packetized policy, with bounded backlog. Universal bounds for this required speedup are established.

First, as concrete motivation for the sequel, in Section IV, we show that already for small Banyan networks, speedup is necessary for tracking with bounded backlog. For the 4×4 Banyan network, we show we need speedup at least $4/3$.

Section V contains the core of our methodology. We establish a natural connection between unique-path switch networks and their *link graphs*, as defined above. This graph formulation allows us to exactly characterize the convex hull of the set of valid packetized matrices, and the set of valid fluid matrices. These sets are polyhedral, and we refer to them as the *Packetized Polytope* of a network, and the *Fluid Polytope* of a network, respectively. These polytopes have been studied extensively in a more general setting in the combinatorics literature where they are referred to respectively as the *stable set polytope* and the *fractional stable set polytope* of a graph (see [17],[18] for details). It is the introduction of this machinery from the combinatorics literature that allows us to obtain some of our bounds.

Our first theorem is a polyhedral characterization of the speedup necessary and sufficient for tracking *constant* fluid policies with bounded backlog. Our second theorem strengthens the first, and proves that if speedup s is sufficient for tracking *constant* fluid policies, then in fact it is sufficient for tracking *arbitrary* fluid policies with bounded backlog.

In Section VI we revisit the 4×4 Banyan network, and show, using the machinery developed in Section V, that speedup $4/3$ is in fact necessary and sufficient for tracking arbitrary fluid policies with bounded backlog.

In Section VII we specialize our results for unique path networks, and show that for a Banyan network with N first-stage input ports, speedup $s = \log_2 N + 1$ is sufficient to track an arbitrary fluid policy with bounded backlog. Furthermore, in this case we show how to implement the packetizing algorithm of Section V to compute each packetized matrix in time polynomial in N.

IV. Speedup is Required

In this section, we exhibit a behavior of the Banyan network that is fundamentally different from the crossbar switch. We exhibit a constant fluid policy that, using no speedup, is impossible to track with bounded backlog. Recall the 4×4 Banyan network in Figure 2. Consider the constant fluid policy, with each fluid step $F^{(k)}$ equal to the matrix

$$F = (x_{ij}) = \begin{bmatrix} \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} \end{bmatrix}.$$

Note that each link in the network has total demand equal to unity. In other words, for every clique Q in the graph G (see Figure 3 above) we have

$$\sum_{(i,j) \in Q} x_{ij} = 1,$$

and the above is indeed a valid fluid step. Suppose that at each time step, this fluid step is requested, and knowing this stationary policy in advance, we wish to choose a valid packetized policy so that the total backlog after M time steps, is minimized. While we cannot transmit fractional values with packetized policies, if we could transmit unit value along four of the eight pairs of positive entries in the fluid step at one time step, and then transmit the remaining four at the next time step, then the backlog would remain bounded. However, one can verify that a packetized policy cannot transmit any more than three of the eight pairs of positive entries of the fluid policy, at any given step. For instance, if $(1,1)$ is transmitted, this rules out $(1,4), (3,1), (2,2)$. Then if, say, $(2,3)$ is transmitted, $(3,3)$ is ruled out, and of the two that remain, $\{(4,2), (4,4)\}$, only one can be transmitted. The same can be seen to be true for any possible set of choices. Therefore any packetized policy can only transmit 3 units per time step, while the fluid policy transmits 4 units each time step. Thus regardless of what packetized policy we choose, the backlog becomes unbounded. In fact, we have proved that the minimum speedup required for a 4×4 Banyan network to track any constant fluid policy with bounded backlog is at least $4/3$. In Section VI we show that this result is tight.

V. Characterization of Required Speedup

In this section we give a characterization of the required speedup for a general unique-path switch network. In addition, we develop the essential elements of our polyhedral and combinatorial methodology which we use throughout. We define $\mathcal{P}$, the *Packetized Polytope of a Switch Network*, to be the convex hull of the set of valid packetized steps, and $\mathcal{F}$, the *Fluid Polytope of a Switch Network*, to be the set of valid fluid steps. In general, we always have that $\mathcal{P} \subseteq \mathcal{F}$ (since a valid packetized step is also a valid fluid step and $\mathcal{F}$, like all polytopes, is convex). The example of the 4×4 Banyan network in Section IV above shows that this inclusion can be strict.

If $\mathcal{P} = \mathcal{F}$ then every fluid step can be seen as a convex combination of packetized matrices and any constant fluid policy can be packetized with bounded backlog and no speedup (by simply scheduling the packetized matrices in the decomposition at the right frequencies). In graph theoretic terms, $\mathcal{P} = \mathcal{F}$ is equivalent to the link graph being *perfect* (see [17]). Many classes of perfect graphs are known, and in particular, the link graph of a crossbar switch is perfect, as it can be seen to be the line graph of a complete bipartite graph. This is a graph-theoretic explanation of the fact that no speedup is required for tracking crossbar switches with bounded backlog.

Theorem 1: All *constant* fluid policies can be tracked with bounded backlog and speedup s if and only if $\mathcal{F} \subseteq s\mathcal{P}$.

Proof: If $F \in \mathcal{F} \subseteq s\mathcal{P}$ then $\frac{1}{s}F$ can be decomposed into a convex combination of packetized matrices. On the other hand, if $\mathcal{F} \not\subseteq s\mathcal{P}$ then there exists a constant fluid policy which cannot be tracked with bounded backlog and speedup s. Consider a matrix $F \in \mathcal{F}$, such that $F \notin s\mathcal{P}$. Then

$$\delta := \inf_{P \in \mathcal{P}} \sum_{i,j} (\frac{F}{s} - P)^{+}_{ij},$$

is strictly positive since $\mathcal{P}$, like all polytopes, is closed. Now, for any packetized policy $\{P^{(k)}\}_{k>0}$, consider the cumulative difference matrix at time step k. We have

$$C^{(k)} = \frac{kF}{s} - \sum_{i=1}^{k} P^{(i)}.$$

Since $(\sum_{i=1}^{k} P^{(i)})/k \in \mathcal{P}$, we have that the sum of positive entries of $\left(\frac{F}{s} - (\sum_{i=1}^{k} P^{(i)})/k\right)$ is at least δ. Thus, the sum of positive entries of $C^{(k)}$ is at least $k\delta$, which grows unbounded as $k \to \infty$. Therefore, the constant fluid policy that schedules F at each time step cannot be tracked with bounded backlog. This proves Theorem 1.

Next, we use ideas similar to those in [1], to show that the necessary speedup s for tracking constant fluid policies is the same as that required for tracking arbitrary fluid policies with bounded backlog. This implies, for any switch network operating strictly slower than the minimum required speedup, *even constant fluid policies are not trackable with bounded backlog*; as soon as the switch runs at least as fast as the minimum required speedup, then *any fluid policy is trackable with bounded backlog*.

Theorem 2: All *arbitrary* fluid policies can be tracked with bounded backlog and speedup s if and only if $\mathcal{F} \subseteq s\mathcal{P}$.

Proof: Assume $\mathcal{F} \subseteq s\mathcal{P}$ holds, so that for each fluid step (which is multiplied by $1/s$ before being packetized) we have $F \in \mathcal{P}$. We present a packetizing algorithm using speedup s that tracks any fluid policy with backlog at most N^2 per input port. The algorithm maintains the invariant $(C^{(k)})^{+} \in N^2\mathcal{P}$, which in turn means that no input port can have backlog more than N^2.

Algorithm 1: Given a fluid policy, this packetizing algorithm computes the packetized policy as follows:

First, calculate $C^{(k-1)} := \sum_{i=1}^{k-1}(F^{(i)} - P^{(i)})$. By our assumption above that $F^{(k)} \in \mathcal{P}$, we have $(F^{(k)} + C^{(k-1)})^{+} \in (N^2 + 1)\mathcal{P}$. Since $\mathcal{P}$ is an N^2-dimensional polytope, Caratheodory's theorem gives that any point in $\mathcal{P}$

can be written as a convex combination of at most $N^2 + 1$ vertices, which in this case are packetized matrices. Thus, we can decompose $(F^{(k)}+C^{(k-1)})^+$ into a convex combination of at most N^2+1 vertices of $(N^2+1)\mathcal{P}$. Augment the weight on the first of these matrices until the sum of the weights equals (N^2+1), and denote the resulting weighted sum of matrices by D. At least one matrix in the decomposition must now have weight at least 1. Let π be one such matrix. We set the packetized step $P^{(k)}$ to be π restricted[4] to the set of entries that have positive values in $F^{(k)} + C^{(k-1)}$.

It follows that $F^{(k)}+C^{(k-1)}-P^{(k)}$ is dominated by $D-\pi$, and $D-\pi \in N^2\mathcal{P}$, and thus the invariant is maintained at time step k, that is $(C^{(k)})^+ = (C^{(k-1)} + F^{(k)} - P^{(k)})^+ \in N^2\mathcal{P}$. In other words, Algorithm 1 tracks any fluid policy with at most N^2 backlog per input port.

In the proof of the above result, we use Caratheodory's Theorem to decompose $(F^{(k)} + C^{(k-1)})^+$ into a convex combination of packetized matrices. Caratheodory's Theorem, however, is not constructive (unless one has a description of the Packetized Polytope in terms of linear inequalities). We give a modified packetizing algorithm below that only requires decomposing each fluid matrix F_k into a convex combination of packetized matrices, and that tracks any fluid policy with bounded backlog. We show in Section VII that for the case of a Banyan Network with N first-stage input ports and speedup $s = \log_2 N+1$, such a decomposition can be computed in time polynomial in N; thus, in this case the packetizing algorithm below runs in time polynomial in N and tracks any fluid policy with bounded backlog.

Again we assume that $\mathcal{F} \subseteq s\mathcal{P}$, so if we take each step $F^{(i)}$ of the fluid policy to be in $\frac{1}{s}\mathcal{F}$, then each step satisfies $F^{(i)} \in \mathcal{P}$. The algorithm maintains the invariant for all time steps k that we have stored matrices $Q^{(i)}$, which are vertices of $\mathcal{P}$, and non-negative constants λ_i such that

$$C^{(k)} \leq \sum_{i=1}^{N^2+1} \lambda_i Q^{(i)},$$

and

$$\sum_{i=1}^{N^2+1} \lambda_i = N^2.$$

The invariant implies $(C^{(k)})^+ \in N^2\mathcal{P}$ for all time steps k. In other words, the packetizing algorithm tracks any fluid policy with backlog at most N^2 per input port at all time steps.

Algorithm 2: Given $C^{(k)}$, and fluid step $F^{(k+1)}$, construct packetized step $P^{(k+1)}$ as follows:

Since $F^{(k+1)} \in \frac{1}{s}\mathcal{F} \subseteq \mathcal{P}$, it can be decomposed as follows,

$$F^{(k+1)} = \sum_{j=1}^{N^2+1} \gamma_j R^{(j)},$$

where for all j, $\gamma_j \geq 0$, the γ_j sum to 1, and $R^{(j)}$ are vertices of $\mathcal{P}$. It is this decomposition that we compute in polynomial time for Banyan networks in Section VII. Now, since $C^{(k)} \leq \sum_{i=1}^{N^2+1} \lambda_i Q^{(i)}$ holds by the invariant above, we have

$$C^{(k)} + F^{(k+1)} \leq \sum_{i=1}^{N^2+1} \lambda_i Q^{(i)} + \sum_{j=1}^{N^2+1} \gamma_j R^{(j)}.$$

The sum of coefficients λ_i and γ_j is $N^2 + 1$. Let

$$T := \sum_{i=1}^{N^2+1} \lambda_i Q^{(i)} + \sum_{j=1}^{N^2+1} \gamma_j R^{(j)}.$$

Caratheodory's theorem now tells us that T can also be expressed as a weighted sum of just N^2+1 matrices from the set $\{Q^{(i)}\} \cup \{R^{(j)}\}$, with the weights summing to N^2+1. This follows from the fact that we have N^2 equalities defining T and one equality constraining the sum of the λ_i's and the γ_j's, all these equalities being defined on the variables $\{\lambda_i\} \cup \{\gamma_j\}$. Here, the decomposition can be done in polynomial time: as long as there are more than $N^2 + 1$ matrices with positive weights, we can reduce this number by taking any nonzero vector y in the null space of the matrix defining the $N^2 + 1$ equalities and modifying the variables $\{\lambda_i\} \cup \{\gamma_j\}$ in the direction y until one more variable becomes 0. This variable is then removed from consideration and the process is repeated until we are down to at most $N^2 + 1$ matrices with positive weights. Rename these matrices to be $\{Q^{(i)}\}$ and the corresponding weights to be $\{\lambda_i\}$. One of these coefficients must be at least 1, so we can set $P^{(k+1)}$ to be the corresponding matrix $Q_l \in \mathcal{P}$, restricted to the set of entries that have positive values in $F^{(k+1)} + C^{(k)}$. For this particular l, decrement λ_l by 1. Then,

$$C^{(k+1)} = C^{(k)} + F^{(k+1)} - P^{(k+1)} \leq \sum_{i=1}^{N^2+1} \lambda_i Q^{(i)},$$

with $\sum_{i=1}^{N^2+1} \lambda_i \leq N^2$ as desired, maintaining the invariant. Thus, the packetizing algorithm tracks any fluid policy with backlog at most N^2 per input port.

VI. Speedup Required for 4×4

In Section IV we exhibited a constant fluid policy for a 4×4 Banyan network, that required speedup $4/3$ to be tracked with bounded backlog. Using the results of Section V above, we show that in fact speedup $s = 4/3$ is necessary and sufficient for tracking arbitrary fluid policies on the 4×4 Banyan network.

From the above discussion, it is sufficient to show that $\mathcal{F} \subseteq \frac{4}{3}\mathcal{P}$ for the 4×4 Banyan network. To show this, decompose any fluid step F into a linear combination of four matrices, each with the four entries in one corner set to 0, as shown in Figure 4. The weight of each matrix is $1/3$. We then use the fact that one can further decompose any of these four matrices into a convex combination of packetized steps. This follows since the subgraph corresponding to one of these matrices with a corner deleted, is the complement of a so-called *comparability graph*. A comparability graph is

[4]We call the matrix D formed by replacing a set of elements E with 0's the matrix D *restricted* to E^c, the complement of E.

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \\ x_{41} & x_{42} & x_{43} & x_{44} \end{bmatrix} = \frac{1}{3}\begin{bmatrix} 0 & 0 & x_{13} & x_{14} \\ 0 & 0 & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \\ x_{41} & x_{42} & x_{43} & x_{44} \end{bmatrix} + \frac{1}{3}\begin{bmatrix} x_{11} & x_{12} & 0 & 0 \\ x_{21} & x_{22} & 0 & 0 \\ x_{31} & x_{32} & x_{33} & x_{34} \\ x_{41} & x_{42} & x_{43} & x_{44} \end{bmatrix}$$
$$+\frac{1}{3}\begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ 0 & 0 & x_{33} & x_{34} \\ 0 & 0 & x_{43} & x_{44} \end{bmatrix} + \frac{1}{3}\begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & 0 & 0 \\ x_{41} & x_{42} & 0 & 0 \end{bmatrix}$$

Fig. 4. 4×4 Banyan network: We obtain the upper bound by decomposing any 4×4 fluid policy into four parts.

such that its edges can be oriented so they form a directed, acyclic, transitive graph $D = (V, A)$. Here *transitive* means that for any nodes u, v, w, if $(u, v) \in A$ and $(v, w) \in A$, then also $(u, w) \in A$. In figure 5 we exhibit such an orientation of the complement of the subgraph obtained when the bottom right corner of the link graph of the 4×4 Banyan network is removed. It is well known (see e.g. [17]) that complements

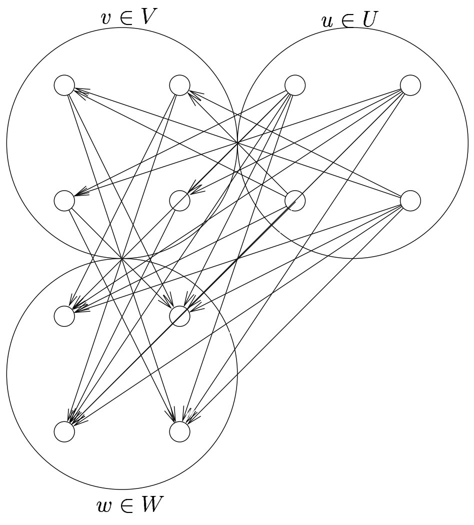

Fig. 5. This is the complement of a 4×4 Banyan network link graph with the corner removed. The edges are oriented so that edges between nodes in U and nodes in V are directed towards V, edges between U and W are directed towards W, and edges between V and W are directed towards W. Therefore this is a directed, acyclic, transitive graph.

of comparability graphs are perfect. Thus any fluid step can be written as a convex combination of packetized steps. Doing this for all four quadrants results in a nonnegative combination of incidence matrices of packetized steps, with the sum of weights at most $4/3$. This shows that $\mathcal{F} \subseteq \frac{4}{3}\mathcal{P}$. We can then use Packetizing Algorithm 1 above with $4/3$ speedup to build, for any given fluid policy, a packetized policy that is at most 16-backlogged per input port.

VII. A Greedy Algorithm for Decomposing Fluid Matrices

We now prove our main result about Banyan networks.

Theorem 3: For a Banyan network with N first-stage input ports, $\mathcal{F} \subseteq (\log_2 N+1)\mathcal{P}$. In particular, speedup $s = \log_2 N + 1$ is sufficient to track an arbitrary fluid policy with bounded backlog. Moreover, the tracking algorithm runs in polynomial time.

Proof: By Algorithm 2 of Section V, it suffices to show for any $F \in \mathcal{F}$, that one can decompose F into a linear combination $\sum \lambda_l A^{(l)}$, with $\forall l, \lambda_l \geq 0$, $A^{(l)}$ representing a valid packetized matrix, and $\sum \lambda_l \leq \log_2 N + 1$. The greedy algorithm below obtains such a decomposition, where for a stable set S of the link graph G of a Banyan network, χ^S denotes the $N \times N$ packetized matrix with value 1 at entries corresponding to elements of S and 0 otherwise.

Algorithm 3: Let F be a given constant fluid matrix.

```
1) Set $M \leftarrow F$ and set $l \leftarrow 1$.
2) Repeat while $M \neq \mathbf{0}$:
     • Find a maximal stable set in the link graph $G$
       restricted to the set of nodes with positive value
       in $M$. Call it $S_l$.
     • Set $\lambda_l \leftarrow \min_{(i,j)\in S_l} M_{ij}$.
     • Set $M \leftarrow M - \lambda_l \chi^{S_l}$ and then increment $l$ by 1.
```

Since at each iteration, at least one entry of M is set to 0, the algorithm terminates after at most N^2 iterations. For a fixed node (i, j) in G, we denote the neighborhood of (i, j) as $\mathcal{N}((i, j)) :=$

$$\{(k, l) : (k, l) = (i, j) \text{ or an edge connects } (k, l) \text{ to } (i, j)\}.$$

Denote the last iteration of the algorithm by t. For any node (i, j) in stable set S_t, for any iteration $t' \leq t$ we have that $S_{t'}$ either contains (i, j) or contains a node (k, l) that is connected to (i, j) by an edge in the link graph. Thus we have

$$\sum_{t'=1}^{t} \lambda_{t'} \leq \sum_{(k,l)\in\mathcal{N}((i,j))} F_{kl}.$$

The sum on the right hand side is the sum of fluid whose path through the switch network includes at least some link in the (unique) path from input i of the first stage to output j of the last stage. Since for any valid fluid step, the total fluid traversing any link in the switch network is at most 1, the total fluid sharing at least one link with the path from i to j is at most the number of links on this path. For a Banyan network with N first-stage input ports and last-stage output ports (so with $\log_2 N$ stages), all path lengths are one more than the number of stages in the network. Thus, we have bounded $\sum_{t'=1}^{t} \lambda_{t'}$ by $\log_2 N + 1$ as desired. By a similar argument, it can be shown for a general, unique-path switch network that speedup equal to the longest path length in the network is sufficient to track with bounded backlog.

We now bound the running time of Algorithm 3. First, we show that it is possible to construct all the maximal stable sets S_l from step 2 in total time $O(N^3 \log^2 N)$. As a subroutine to the construction, we check whether adding a new node to a stable set of the link graph G of a Banyan network results in a stable set. In order to do this efficiently, we keep track of which of the $N(\log_2 N + 1)$ constraints in the link graph G are *covered* by the stable set S under consideration, where we say that a set S of nodes *covers* a constraint if at least one node in S is in the corresponding clique. Then we can check, for any node (i, j) in G, whether $\{(i, j)\} \cup S$ is a stable set by checking if none of the $\log_2 N + 1$ constraints whose corresponding cliques contain (i, j) are covered by S.

We construct maximal stable set S_1 by sequentially considering each of the nodes of G with positive corresponding entries in M, keeping those with no constraints currently covered and then setting all their constraints to "covered." This takes time $O(N^2 \log N)$. For $l \geq 2$, we construct the maximal stable set S_l by starting with the stable set S_{l-1} and removing the set of nodes (call it R_{l-1}) whose corresponding entries in M were set to 0 during iteration $l - 1$. Also, all constraints covered by a node in R_{l-1} are set to "not covered." We then augment $S_{l-1} \setminus R_{l-1}$ to form a maximal stable set in G restricted to the nodes with positive value in M, by considering in turn each node in $\cup_{(i,j)\in R_{l-1}} \mathcal{N}((i, j))$ with positive value in M; for each such node, if all its constraints are not covered, we add it to the stable set and cover all its constraints. The resulting stable set S_l is maximal since adding any node with positive value in M and not in $\cup_{(i,j)\in R_{l-1}} \mathcal{N}((i, j))$ to $S_{l-1} \setminus R_{l-1}$ would result in a set that is not stable (otherwise S_{l-1} is not maximal).

Computing the maximal stable set at iteration $l \geq 2$ takes time $O(|R_l| N \log^2 N)$, since the algorithm checks, for each node (i, j) in R_l, for each node (k, m) in the neighborhood of (i, j), whether the $\log_2 N + 1$ constraints of (k, m) are covered. Since for each node (i, j), there is at most a single $l \geq 2$ such that $(i, j) \in R_l$, we have $\sum_{l\geq 2} |R_l| \leq N^2$. Because the maximum size of a stable set of G is N, the other parts of step 2 can be computed in time $O(N)$ per iteration. Thus, the total run time of Algorithm 3 is $O(N^3 \log^2 N)$.

VIII. Conclusions

In this paper, we have considered packetized policies in order to track with bounded backlog arbitrary time varying fluid policies for an $N \times N$ Banyan network. First, we showed that in contrast to the crossbar switch, Banyan networks require speedup in order to track arbitrary fluid policies with bounded backlog. Next, we developed the concept of required speedup, and computed the exact speedup required for the 4×4 Banyan network, and obtained logarithmic bounds on the speedup required for a general $N \times N$ Banyan network. Computing the exact speedup required for general Banyan networks, and other networks of interest, remains an interesting and stimulating open problem.

In addition, we introduced a natural relationship to graph theory and combinatorics, allowing us to develop significant machinery. We believe that many more benefits can be reaped from this relationship between switch networks and polyhedral combinatorics.

IX. Acknowledgments

We would like to thank Samuel Fiorini for helpful discussions and specific comments on improving this paper.

References

[1] M. Rosenblum, M. X. Goemans, and V. Tarokh, "Universal bounds on buffer size for packetizing fluid policies in input queued, crossbar switches," in *Proc. IEEE Infocom (in press)*, 2004.
[2] Abhay K. Parekh and Robert G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: The single-node case," *ACM/IEEE Transactions on Networking*, vol. 1, no. 3, June 1993.
[3] Abhay K. Parekh and Robert G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: The multiple node case," *ACM/IEEE Transactions on Networking*, vol. 2, no. 2, April 1994.
[4] R. Cruz, "A calculus for network delay.I. network elements in isolation.," *IEEE Transactions on Information Theory*, vol. 37, no. 1, pp. 114–131, January 1991.
[5] R. Cruz, "A calculus for network delay. II. network analysis.," *IEEE Transactions on Information Theory*, vol. 37, no. 1, pp. 132–141, January 1991.
[6] Jon C. R. Bennett and Hui Zhang, "WF^2Q: worst-case fair weighted fair queueing," in *INFOCOM (1)*, 1996, pp. 120–128.
[7] A. Stamoulis and J. Liebherr, "S^2GPS: slow-start generalized processor sharing," in *Proc. of Workshop on Resource Allocation Problems in Multimedia Systems (held in conjunction with IEEE Real-Time Systems Symposium.)*, University of North Carolina at Chapel Hill, December 1996.
[8] Nick G. Duffield, T. V. Lakshman, and Dimitrios Stiliadis, "On adaptive bandwidth sharing with rate guarantees," in *INFOCOM (3)*, 1998, pp. 1122–1130.
[9] Anastasios Stamoulis and Georgios B. Giannakis, "Deterministic time-varying packet fair queueing for integrated services networks," in *IEEE Global Telecommunications Conference*, 2000, vol. 1, pp. 621–625.
[10] Michael John Girone, "Tracking switch fluid policies: Bounding lookahead," Master's project, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, February 2002.
[11] Cheng-Shang Chang, Wen-Jyh Chen, and Hsiang-Yi Huang, "Birkhoff-von Neumann input buffered crossbar switches," in *INFOCOM (3)*, 2000, pp. 1614–1623.
[12] Vahid Tabatabaee, Leonidas Georgiadis, and Leandros Tassiulas, "QoS provisioning and tracking fluid policies in input queueing switches," in *INFOCOM*, 2000, pp. 1624–1633.
[13] M.A. Bonuccelli and M.C. Clo, "EDD algorithm performance guarantee for periodic hard-real-time scheduling in distributed systems," *IEEE IPPS/SPDP*, pp. 668–677, April 1999.
[14] Micah Adler, Petra Berenbrink, Tom Friedetzky, Leslie Ann Goldberg, Paul W. Goldberg, and Mike Paterson, "A proportionate fair scheduling rule with good worst-case performance," in *ACM Symposium on Parallel Algorithms and Architectures*, San Diego, CA, USA, June 2003.
[15] A. Pattavina, *Switching Theory, Architectures and Performance in Broadband ATM Networks* John Wiley and Sons, New York, NY, 1998.
[16] S. Keshav, *An Engineering Approach to Computer Networking* Addison–Wesley Pub Co, Boston, MA, 1997.
[17] M. Grötschel, L. Lovász, and A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*, Springer–Verlag, Berlin Heidelberg, 1993.
[18] A. Schrijver, *Combinatorial Optimization: Polyhedra and Efficiency*, Springer, New York, 2003.