Technical Report UCSC-CRL-07-03:
Optimizing Capacity in Interconnection Networks with Finite Buffers

Kevin Ross[1] and Nicholas Bambos[2]

Abstract

A great deal of recent research attention has been given to the throughput maximization of interconnection networks. These networks connect computer processing and routing resources via both physical and wireless links, where the resources are reconfigurable in a dynamic fashion. Jobs move through several processing stations and can be buffered between each stage. Most theoretical work assumes that the capacity for buffering at each stage is infinite, and general classes of scheduling policies have shown how to maximize the capacity under those conditions. Here we consider the case when such buffers are finite, and analyze the effect on network stability.

We find that for very general arrival processes and an arbitrary fixed network topology, the stability region with finite buffers is a subset of that for the corresponding network with infinite buffers. As the capacity of buffers increases, the stability region eventually becomes equivalent to that for infinite buffer networks[3].

1 Introduction

In this paper, we develop a model for general interconnection networks, with service modes describing the routing options of the network. An interconnection network can be modeled as a globally controlled set of queues, and most analysis of the throughput properties of such networks has assumed infinite buffer sizes, which are not available in practice.

It has recently been proved [4, 5] that the throughput region of a certain class of networks maintain the same capacity under finite buffers as for the infinite case. There it was shown that for bernoulli iid traffic arrivals, a network with finite per-class buffers will guarantee to maintain maximal throughput for buffers of equal size per traffic class. It has remained an open question what effect finite buffers have on more general network topologies and traffic patterns.

We propose a new class of algorithms that is throughput maximizing for interconnection networks in the presence of finite internal buffers on a very general network model. We show that the throughput capacity of networks is nondecreasing in the size of any individual buffer, and that there exists a threshold above which the throughput capacity is equivalent to that for the same network topology with infinite buffers.

We consider the case where general arrival processes arrive following arbitrary distributions, but must enter the network through infinite-buffer ingress queues. Without such a restriction, there is always a nonzero probability of arrivals overflowing a finite ingress buffer. We permit different buffer sizes over the network.

We extend the class of *cone* or *max-pressure* policies that have previously been known to have favorable properties for throughput maximization, quality of service control and complexity reduction in switches [11] and networks [13] with infinite queue sizes. Cone algorithms are weighting-based policies in a similar style to [17] for networks and [8] for crossbar packet switches.

Previous work describing network stabilization has been presented, most notably [17]. Several extensions have shown related stability maximization results [1, 7, 9, 10, 16], however all of these assume that we

[1]kross@soe.ucsc.edu.edu; Baskin School of Engineering, University of California, Santa Cruz, CA 95064.

[2]bambos@stanford.edu; Department of Management Science & Engineering, and Department of Electrical Engineering, Stanford University, Stanford, CA 94305.

[3]Preliminary results were presented in [12]

have infinite buffers throughout the network. More detailed descriptions of interconnection networks and associated scheduling issues are addressed in [2, 3, 6].

Switches with finite internal buffers have recently been proposed [18] and their stability analyzed in [4, 5]. The related issue of allocating finite shared buffer capacity over a network is studied in [15] The case analyzed in this paper allows routing decisions, encoded in service modes, where arriving packets need not follow a pre-determined path through the network. Data packets may split or merge throughout the network.

The remainder of the paper proceeds as follows. In section 2, we describe the system model for interconnection networks with finite and infinite buffers. In section 3 we discuss network stability and throughput maximization, section 4 introduces batch-mode policies and in section 5 we describe the cone algorithms. We discuss conclusions and future work in section 6.

2 Model and Network Structure

In this paper, we model a fixed topology interconnection network with Q different job stations, each station includ ing a queue labeled $q \in \mathcal{Q} = \{1, 2, ..., Q\}$. Jobs arrive to ingress queues in the form of packets that can be divided into equal sized cells. Upon completion at one station they may be forwarded to another station, or exit the network. The networks we consider are feed-forward, where cells cannot return to the same queue. They may, however, return to the same processor via a different queue[4]. Figure 1 illustrates many of the important features of the queueing networks considered here, particularly

- different buffer sizes for different queues,
- arrivals only to the infinite-buffer queues,
- routing decisions at various nodes, eg. cells in queue 3 can be forwarded to queues 4, 5 or 6,
- both merging and splitting, where cells can enter and depart each buffer via multiple different paths,
- feed-forward topology, where cells do not return to the same queue and
- forwarding both to and from both finite and infinite buffer queues.

Time is slotted in discrete, even time increments, labeled $t \in \{0, 1, 2, ...\}$. At each timeslot, a global network scheduler can align the service resources and routing rules according to the backlog observed throughout the network. Cells arrive and depart from each queue according to external arrivals and internal forwarding between queues as controlled by a mode-selection algorithm.

Let $X_q(t)$ denote the number of cells stored in queue q at the start of timeslot t, for each queue q. The capacity of queue q is labeled B_q, so we will require the scheduling algorithm to ensure $X_q(t) \leq B_q$ for all timeslots t in each queue. For infinite-buffer queues, we set $B_q = \infty$ to allow for consistent notation.

We investigate loss-less networks where no packets are allowed to be dropped. Further, we will allow general arrival processes, and as such it is necessary to exclude external arrivals to any queue q for which with $B_q \neq \infty$[5]. Therefore we have in general a network of inifinite buffer ingress queues and finite-buffer internal queues. The internal queues only receive arrivals from forwarded jobs, hence we are able to restrict the arrivals to those queues via the chosen scheduling rules.

At the beginning of each timeslot, a scheduling algorithm selects the service *mode* for the network. The mode corresponds to a feasible allocation of network resources and is based in general on the backlog state encoded in the vector $X(t) = (X_1(t), X_2(t), ..., X_Q(t))$. When mode m is selected, cells are processed

[4]We utilize per-class or per-flow queues where each unique packet type is stored in a single virtual buffer.

[5]Under most arrival processes it is simple to create an adversarial arrival trace that would overflow an ingress buffer

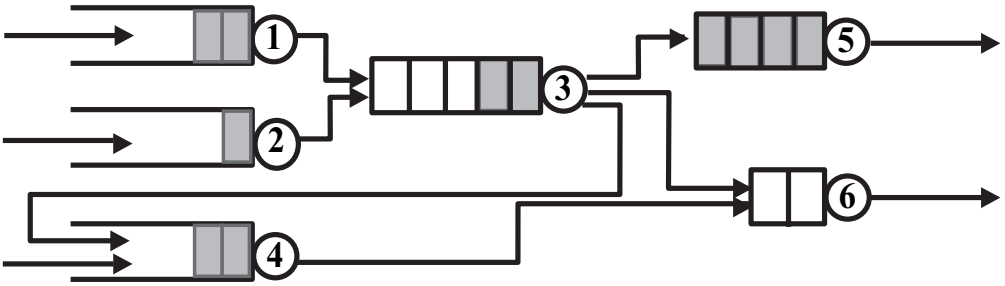

Figure 1: A sample network topology with $Q = 6$ different queues in the above network, where queues $q \in \{1, 2, 4\}$ each has an infinite buffer and may receive incoming requests from outside the network, and $q \in \{3, 5, 6\}$ each has a finite buffer, and can receive forwarded cells from the other queues. In this illustration, some of the cell slots are taken by the shaded cells, and the finite queues have a fixed number of available slots in the buffer space. The backlog illustrated above would be described as $X = (2, 1, 2, 2, 4, 0)$, with buffer limits $B = (\infty, \infty, 5, \infty, 4, 2)$. The service modes available would reflect the combinations of service and forwarding that could be applied in each timeslot. For example a mode m with $\mathbf{T}^m_{13} = 4, \mathbf{T}^m_{36} = 2, L^m_5 = 3$ and all other elements zero would forward 4 cells from queue 1 to queue 3, 2 cells from queue 3 to queue 6 and 3 cells from queue 5 out of the network. This would be reflected by the total departure and entering vectors $D^m = (4, 0, 2, 0, 3, 0)$ and $E^m = (0, 0, 4, 0, 0, 2)$.

at their stations and forwarded either to another queue or out of the network, depending on the mode. We denote $\mathbf{T}^m_{pq}$ to be the number of cells *transferred* from queue p into queue q under mode m. In addition to the transferred cells, L^m_q cells leave the network directly from queue q under mode m. When mode m is selected at time t, we use the notation $L(t) = L^{m(t)}, \mathbf{T}(t) = \mathbf{T}^{m(t)}$, and similarly for other mode-related statistics we define below.

Consider the total change in backlog in a given queue for each timeslot when mode m is selected. The total number of cells *entering* q from forwarding is given by $E^m_q = \sum_{p \in \mathcal{Q}} \mathbf{T}^m_{pq}$. Similarly, the total number of cells *departing* queue q is $D^m_q = \sum_{p \in \mathcal{Q}} \mathbf{T}^m_{qp} + L^m_q$.

There is a finite (but possibly large) set of modes $\mathcal{M}$ from which mode $m(t)$ must be selected at each timeslot t corresponding to all of the forwarding and exiting combinations. We assume that in each timeslot, the mode $m(t)$ is in fact selected from the subset of modes $\mathcal{M}(t)$ that are feasible for the current backlog level $X(t)$. In particular, if a mode corresponds to either more departures to a queue than are currently waiting or more arrivals to a queue than a receiving buffer will allow, then that mode is infeasible, and not included in the set $\mathcal{M}(t)$.

$$\mathcal{M}(t) = \{m \in \mathcal{M} : D^m_q \leq X_q(t), \text{ and } E^m_q - D^m_q \leq B_q - X_q(t) \text{ for all } q \in \mathcal{Q}\} \tag{2.1}$$

Notice that this corresponds to a restriction of arrivals and departures based on cells waiting at the beginning of a timeslot. This corresponds to a store-and-forward network, as opposed to cut-through where cells can both arrive and depart in the same timeslot. It does, however allow for cells departing a queue to be replaced in the same timeslot.

We make an assertion on the set of modes that allows us to always select an exact mode corresponding to real arrivals and departures. We assume that the set of modes is *complete* in the sense that any forwarding or departing cell can be canceled to give a new mode. Formally, if $m \in \mathcal{M}$, then $\hat{m} \in \mathcal{M}$, where

$$\mathbf{T}^{\hat{m}}_{pq} = \mathbf{T}^m_{pq} - 1 \geq 0, \text{ or } L^{\hat{m}}_q = L^m_q - 1 \geq 0 \tag{2.2}$$

for any $p, q \in \mathcal{Q}$. In general it is possible to make any initial mode set complete by adding the appropriate modes to the set. These additional modes correspond to actual cell transfers, avoiding the potential of applying service to empty queues[6].

Correspondingly, for any mode m and backlog sate X such that mode m would be infeasible for backlog X, we define the function $m_f(m, X)$ to be one which selects a feasible mode $m_f \in \mathcal{M}$ such that

$$D_q^{m_f} \leq X_q \leq B_q - E_q^{m_f} \tag{2.3}$$

with

$$T_{pq}^{m_f} \leq T_{pq}^{m} \text{ and } L_q^{m_f} \leq L^m \tag{2.4}$$

for every $p, q \in \mathcal{Q}$. Further, we require that it select a maximal feasible such mode in the sense that if any other m' satisfies (2.3) then $T_{pq}^{m_f} \geq T_{pq}^{m'}$ and $L_q^{m_f} \geq L^{m'}$. Notice that the function $m_f(m, X)$ is equivalent to canceling transmissions until the mode becomes feasible. If more than one maximal mode satisfies (2.3) then one can be selected arbitrarily.

We also differentiate between modes which forward into finite-buffer queues and those which do not.

$$\mathcal{M}^+ = \{m \in \mathcal{M} : E_q = 0 \text{ for all } q \text{ with } B_q < \infty\} \tag{2.5}$$

In addition to transferred cells, let $A_q(t)$ be the number of cells arriving to queue q from outside the network. All of the key terms have been described with subscripts q according to each queue. We will omit the subscripts except where necessary, and consider the Q-length *vector* of components corresponding to the queues in $\mathcal{Q}$. The backlog evolution in the system is described by

$$X(t) = X(t-1) + A(t-1) + E(t-1) - D(t-1) \tag{2.6}$$

for $t \in \{1, 2, ...\}$, where $X(0)$ is the initial backlog, $A(t)$ the vector of external arrivals and both $E(t)$ and $D(t)$ are determined by the mode selected via the scheduling algorithm. We are particularly interested in the behavior of backlog $X(t)$ as modes are selected and arrivals processed.

Consider for example a network that is a simple series of connected feed-forward stations, each with capacity to store one cell. Suppose that in a single timeslot exactly one cell can be forwarded from its queue to the next, or out of the network at the final station. In this case, the modes correspond to each queue-forwarding option, and are of the form $\mathbf{T}_{pq}^m = 1$ for the mode m forwarding queue p to q, and zero for all other (p, q) pairs. For the mode m processing a cell from a queue q with no down downstream one, $L_q^m = 1$ for that queue and zero for all other queues. Each mode is available in $\mathcal{M}(t)$ whenever that buffer contains a cell, and its downstream buffer is empty.

A more general interconnection would be one with P processors to be allocated over all Q queues. If each processor still forwards a single cell per slot, the modes available would be those satisfying $\sum_p \mathbf{T}_{qp}^m + L_q^m \leq P$, with $\mathbf{T}_{pq}^m, L_q^m \in (0, 1)$ and $L_q^m = 1$ only in the most down-stream queues.

Far more general processor sharing and routing situations can be captured in this model by developing the appropriate set of service modes. In the context of this model, finite buffers have the primary effect of making certain modes infeasible for a given backlog state, and we will concentrate on how this can influence the capacity of an interconnected network. We will assume that every mode in $\mathcal{M}$ is feasible for some backlog state X, since otherwise it could be dropped from the set $\mathcal{M}$.

[6]In doing this, we may require a large number of modes to be in $\mathcal{M}(t)$. However, it is not necessary to consider all modes available at a given timeslot, since in practice we can ignore modes that are strictly dominated by others.

3 Stability: The Infinite Buffer Case

Let

$$\rho_q = \lim_{T\to\infty} \frac{\sum_{t=0}^{T-1} A_q(t)}{T} \geq 0 \tag{3.1}$$

be the long-term average external arrival rate to queue q for all $q \in \mathcal{Q}$. We assume that this is finite but perhaps unknown to the network scheduler. Arrivals can follow arbitrary dynamics, including dependence between queues. For queues with finite buffers, we note that $\rho_q = 0$ (since $A_q(t) = 0$), and consider the vector $\rho \geq 0$ of the arrival rates to all queues.

In this work we consider a general notion of stability, known as rate-stability. A queue is rate-stable if the average rate of arrivals is equal to the average rate of departures. Formally, the network of queues is rate-stable if

$$\lim_{T\to\infty} \frac{\sum_{t=0}^{T-1}[A(t) + E(t)]}{T} = \lim_{T\to\infty} \frac{\sum_{t=0}^{T-1} D(t)}{T} \tag{3.2}$$

A natural question is to evaluate the arrival rates ρ for which it is possible to maintain rate-stability given a set $\mathcal{M}$ of feasible modes. We refer to this set as the *stability region*. For infinite buffer networks, this is know to be

$$\mathcal{R}^\infty = \{\rho \in \Re_{0+}^Q : \rho \leq \sum_{m\in\mathcal{M}} \phi^m (D^m - E^m) \text{ for some } \phi^m \geq 0, \sum_{m\in\mathcal{M}} \phi^m = 1\} \tag{3.3}$$

where $\Re_{0+}^Q$ denotes the non-negative real Q-vectors, and the inequality holds componentwise for all $q \in \mathcal{Q}$. Note that $\mathcal{R}^\infty$ is defined only over modes which are available for some feasible backlog, but it is not necessarily true that each of these modes can be utilized whenever desired, as noted by the possibility of empty and full buffers. The *geometry* of the stability region is illustrated in Fig. 2. The stability region can be seen as the positive quadrant of the convex hull of possible $(D^m - E^m)$ vectors in Q dimensions.

Consider any arrival process with $\rho \notin \mathcal{R}$ according to (4.6). It is clear that rate stability (3.2) can not hold, and at least one queue will have linear growth in backlog level. Therefore this characterizes an upper bound on the maximum possible set of arrival vectors ρ for which rate-stability may *conceivably be* achieved. It has been known that this region can be achieved for infinite internal buffers, but with finite internal buffers it remains to show that some algorithms do necessarily guarantee rate stability as in (3.2). We say that a mode-selection algorithm is *throughput maximizing* if it guarantees rate stability (3.2) for any $\rho \in \mathcal{R}$ without prior knowledge of ρ.

In [4, 5], the scheduling policy itself ensured that forwarding to full buffers was never selected. Here, we take a different approach, explicitly defining batch-modes to forward cells through the network through sequences of feasible modes.

It has been known that a class of policies known as max-pressure. maximum weighted matching or cone scheduling policies will achieve the maximum possible throughput in these networks if $B_q = \infty$ for all queues $q \in \mathcal{Q}$ (see for example [1, 14, 17]). The policies select a mode at time t as follows:

$$m(t) \in \arg\max_{m\in\mathcal{M}(t)} \{\sum_{q=1}^{Q} (D_q^m - E_q^m) X_q(t)\} \tag{3.4}$$

These policies, and weighted versions with prioritized queues can be shown to maximize throughput to the network. The difficulty for mode selection in interconnection networks with finite buffers is that the *best* modes may become unavailable. For example, if the backlog in one queue becomes large, the scheduler

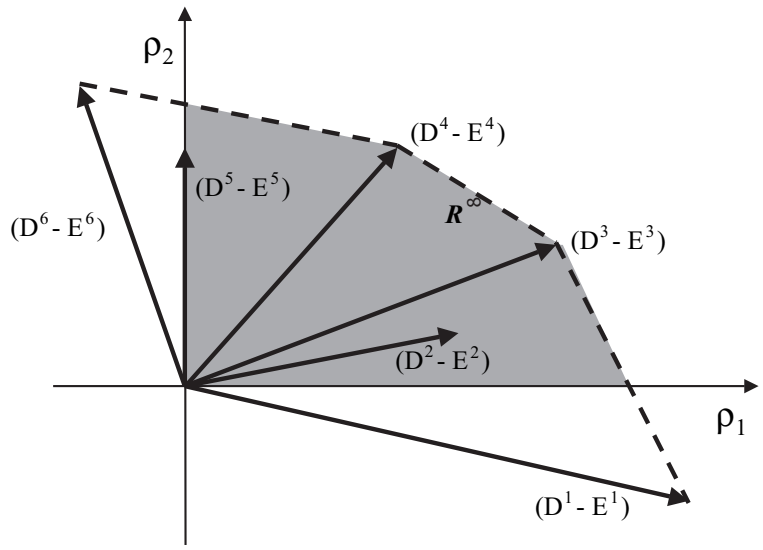

Figure 2: **The stability region.** The stability region with infinite buffers $\mathcal{R}^\infty$ can be illustrated geometrically as the positive quadrant of the convex hull of change vectors $(D^m - E^m)$ for all modes m. We will see in this work that the extreme points of $\mathcal{R}^\infty$ are important for the analysis, and one can see above that these correspond to both true service modes and convex combinations of actual service rates from chosen modes.

would *like* to allocate service to that queue at a high rate. However, if that queue forwards to another queue that is full, the modes serving the large queue are not feasible. The scheduler should then remove cells from the downstream queue before serving the large queue. More generally there may be a whole *sequence* of modes required to clear the path for service to the large queue, which we call a batch-mode, as developed in the next section.

4 Batch-Modes Under Full Buffers

The intuition for these batch modes can be seen from the simplest network of two tandem queues in figure 3. There are two modes available, m_1 forwards a single cell from queue 1 to 2, and m_2 forwards from queue 2 out of the network. Whenever queue 2 is full, the selection algorithm should ideally utilize the two modes consecutively. The net effect of this is serving queue 1 at rate 0.5 and queue 2 at rate 0. We call this a *batch-mode* $\hat{m} = 0.5m_1 + 0.5m_2$. This concept of batch-modes is developed more generally in this section.

Notice that the definition (4.6) of the stability region $\mathcal{R}^\infty$ can be interpreted as the positive quadrant of the convex hull of a set of vectors. Those vectors, $D^m - E^m$, reflect the total change of backlog under each mode. Consider the *extreme points* of the stability region $\mathcal{R}^\infty$. From the stability region definition, the boundary must be a convex combination of $D^m - E^m$ vectors, and the extreme points either equal to $D^m - E^m$ for some mode m, or the intersection of convex combination of these vectors with the boundaries of the positive quadrant. For any extreme point of the stability region that is not an original mode, we define a *batch mode* to be the convex combination of modes that gives rise to that combination or original modes.

A batch-mode $\hat{m}$ is an ordered sequence $\{\hat{m}_1, \hat{m}_2, ..., \hat{m}_{t^{\hat{m}}}\}$ of $t^{\hat{m}}$ (perhaps repeated) regular modes $m \in \mathcal{M}$. Associated with $\hat{m}$ we define

$$\mathbf{T}^{\hat{m}} = \frac{1}{t^{\hat{m}}} \sum_{i=1}^{t^{\hat{m}}} T^{\hat{m}_i} \text{ and } L^{\hat{m}} = \frac{1}{t^{\hat{m}}} \sum_{i=1}^{t^{\hat{m}}} L^{\hat{m}_i} \tag{4.1}$$

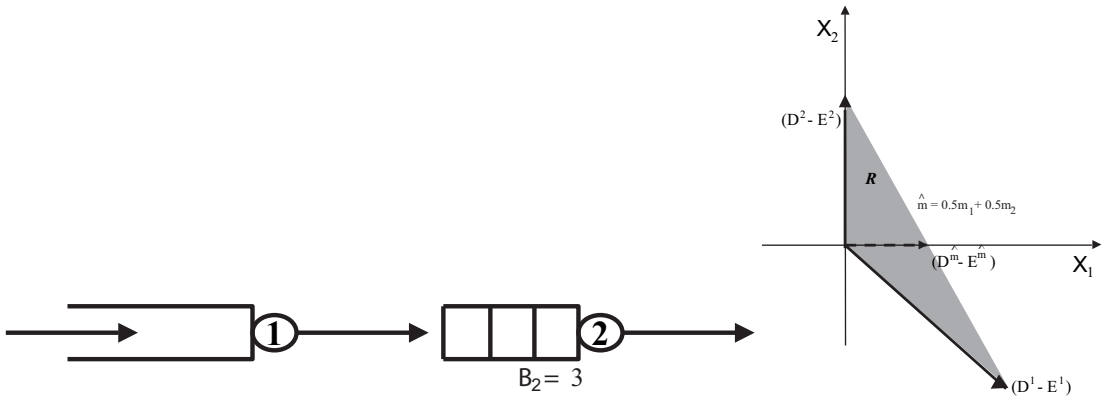

Figure 3: Two tandem queues with 2nd finite buffer. The figure above depicts a two-queue tandem network with finite second buffer, in this case $B_2 = 3$. The batch-mode corresponding to the extreme point of the feasibility region is indicated by the mode $\hat{m}$. This corresponds to a combination of true modes being utilized.

This corresponds to the effective service rates that would correspond to using the sequence of modes defined in the batch as long as all modes remained feasible. Batch modes are only defined for sequences on which

$$D^{\hat{m}} - E^{\hat{m}} = \frac{1}{t^{\hat{m}}} \sum_{i=1}^{t^{\hat{m}}} E^{\hat{m}_i} - \frac{1}{t^{\hat{m}}} \sum_{i=1}^{t^{\hat{m}}} D^{\hat{m}_i} \geq 0 \tag{4.2}$$

to be the corresponding departure vector under the rates in batch mode $\hat{m}$.

Batch-modes are utilized as follows. If at time t_0 batch mode $\hat{m}$ is selected to begin, then for timeslots t_0 until $t_0 + t^{\hat{m}} - 1$ the mode selected at time t is $\hat{m}_i$ where $i = t - t_0 + 1$. Since it is possible that some of the nodes are infeasible, we use $m_f(X(t), \hat{m}_i)$.

We impose the following condition on the modes in the batch, which we later show leads to feasible sequences of modes.

$$-B_q \leq \sum_{i=1}^{k} D_q^{m_{i(modt^{\hat{m}})}} - E_q^{m_{i(modt^{\hat{m}})}} \leq B_q \tag{4.3}$$

for all $q \in \mathcal{Q}$ with $B_q < \infty$, and for all $1 \leq k \leq t^{\hat{m}}$. These conditions imply that if the batch mode repeated in cycles, the queue would never more than completely fill or completely empty a buffer. We denote the set of all batch-modes satisfying (4.2) and (4.3) to be $\hat{\mathcal{M}}$.

4.1 Stability/Feasibility of Batch Modes

Having defined batch modes we turn our attention to the implications for stability. We argue that if a batch mode is buffer-feasible, in the sense that it satisfies (4.3), then if queue q is nonempty from time t_0 and batch mode $\hat{m}$ is selected repeatedly at times $t_0, t_0 + t^{\hat{m}}, t_0 + 2t^{\hat{m}}, ...$ then

$$\lim_{T \to \infty} \frac{\sum_{t=t_0}^{T} T_{pq}(t)}{T - t_0} = T_{pq}^{\hat{m}} \tag{4.4}$$

for all p, q and

$$\lim_{T\to\infty} \frac{\sum_{t=t_0}^{T} L_{pq}(t)}{T - t_0} = L_{pq}^{\hat{m}} \tag{4.5}$$

for all q. This means that the effective long term service rates to every queue are precisely given by $T^{\hat{m}}, L^{\hat{m}}, E^{\hat{m}}$ and $D^{\hat{m}}$.

This leads to the following definition of the stability region for finite-buffer networks.

$$\mathcal{R} = \{\rho \in \Re_{0+}^{Q} : \rho \leq \sum_{m\in\mathcal{M}^+\cup\hat{\mathcal{M}}} \phi^m (D^m - E^m) \text{ for some } \phi^m \geq 0, \sum_{m\in\mathcal{M}} \phi^m = 1\} \tag{4.6}$$

Observe that $\mathcal{R} \subseteq \mathcal{R}^\infty$. This follows from the fact that $\mathcal{M}^+$ is a subset of $\mathcal{M}$ and $\hat{\mathcal{M}}$ is defined by vectors which are convex combinations of ones in $\mathcal{M}$.

We also observe that $\mathcal{R}$ is dependent on B_q values through (4.3), and is in fact nondecreasing in each B_q value. That is, if $B_q^1 \leq B_q^2$ for all q, then the associated $\mathcal{R}^1$ (based on buffer sizes B_q^1) and $\mathcal{R}^2$ (based on buffer sizes B_q^2) will satisfy $\mathcal{R}^1 \subseteq \mathcal{R}^2 \subseteq R^\infty$. This means that the stability region may grow as the buffers grow. Finally, we show that there is a sufficiently large buffer such that in fact $\mathcal{R} = \mathcal{R}^\infty$.

The core concept can be illustrated by returning to the earlier tandem example. In this case, mode $\hat{m}$ corresponds to spending half the time in each of the two available modes. If the batch mode uses a strictly alternating sequence, (4.3)implies that the 2nd queue must satisfy $B_2\ geq 1$. However, if the batch-mode were to repeat each mode r times before switching, it would require $B_2 \geq r$. Therefore we see that (4.3) is critical in evaluating the necessary buffer levels.

This follows from theorem 5.1 (detailed later), and gives a sufficient condition for the buffer sizes B_q in each finite queue.

An important observation, reflected in proposition 4.1 is that *if a batch-mode is utilized long enough, eventually the entire sequence of original modes corresponds to a feasible sequence of modes*.

Proposition 4.1 *Consider any batch-mode $\hat{m}$ corresponding to a sequence of actual modes satisfying (4.3) . Then there exists a finite number N the entire sequence of original modes must become feasible after N repeated batch-modes, as long as X_q remains nonempty for any queue with $D_q^{\hat{m}} - E^{\hat{m}} > 0$.*

Proposition is important since it assures that if one batch-mode is used long enough, the actual effective service to each of the queues is reflected by the rates $E^{\hat{m}}$ and $D^{\hat{m}}$ from (4.1), even if initial batches have to cancel some cell transfers.

Proof: (of proposition 4.1) This proof relies on the topology of the network we have described. Notice that $\hat{m}$ corresponds to a sequence of forwarding between queues in $\mathcal{Q}$, corresponding to a network of feed-forward queues. That is, we can label the queues $\mathcal{Q} = \{1, 2, ..., Q\}$ whereby queue p forwards to q only if $p < q$.

There are two cases when mode m may be selected in $\hat{m}$ but not fully utilized: (i) mode m is trying to forward to a full buffer (corresponding to $E_q^m - D_q^m > B_q - X_q$), or $(ii)m$ is trying to serve cells out of an empty queue (corresponding to $D_q > X_q$). This proof argues that after batch-mode $\hat{m}$ has been used for $t^{\hat{m}}$ batches, (i) cannot be the case, and after a further $t^{\hat{m}}$ batches, (ii) cannot be the case. Hence after $N = 2t^{\hat{m}}$ batches, the actual service modes are all feasible.

Consider the most downstream queue $q \in \mathcal{Q}$ for which $D_q^m > 0$ for one of the modes in $\hat{m}$. Since it is the most downstream queue, in the first batch it cannot be blocked by a full buffer downstream from it. Therefore if there is positive backlog in q then it will all be forwarded in the first batch. Further, it will be able to receive up to B_q cells in the next batch.

Now consider the nth batch, and assume that the last $n-1$ queues with $\alpha^m D_q^m > 0$ are not blocked by a full downstream buffer, and are all able to receive up to B_q cells in that batch. Then the nth queue from

the end must be able to forward all cells in that batch. Further, if there are cells waiting in that queue, then they will not block any entering cells in the following batch.

By induction, it follows that after at most $t^{\hat{m}}$ batches of $\hat{m}$, there can be no blocking due to full down-stream buffers.

We now consider the next Q batches, by considering first the most upstream queue to receive service. Since this corresponds to a queue with only departures in the batch, it must correspond to a queue with $D_q^{\hat{m}} > 0$, and $E_q^{\hat{m}} = 0$. We have asserted that X_q is large enough that there will always be waiting cells in such a queue q.

In the $(Q+1)$st batch, the first queue must have at least $D_q^{\hat{m}} - E_q^{\hat{m}}$ cells waiting, and not be blocked from forwarding. Hence that queue can always forward all cells that are scheduled in the batch. Further, the next down-stream queue now has received the total arrivals for the batch.

Generally, suppose that after $t^{\hat{m}} + n$ batches, the first $n-1$ queues in the feed-forward network have forwarded $D_q^{\hat{m}}$ and the first n have received at least $E_q^{\hat{m}}$. Then in the nth queue must be able to forward all $D_q^{\hat{m}}$ cells. It further follows that the $n+1$st queue must have received all $E_q^{\hat{m}}$ entering cells in that batch.

We now see by induction, that after at most $2t^{\hat{m}}$ batches, all cell forwarding and receiving exactly follows the schedule according to the batch-mode $\hat{m}$, and this completes the proof of proposition 4.1. ∎

In our example, the batch-mode uses modes m_1 and m_2 by cycling from one to the other. Hence, the first time the batch-mode is used, it may try to do m_1 before m_2, and forward a cell from queue 1 to queue 2. If the second queue is full, this is canceled by selecting a mode dominated by m_1 that is feasible, in this case the zero service mode. However, if the batch-mode is repeated again, the cell transfers become feasible since the first cycle clears the 2nd buffer, and after that the sequence of m_1 followed by m_2 could be used repeatedly. Proposition 4.1 generalizes this property for feed-forward networks. Operationally, when mode $\hat{m}$ is selected, the mode is utilized for (at least) $t^{\hat{m}}$ consecutive timeslots. We denote $\hat{\mathcal{M}}$ to be the be the set of batch-modes.

5 Stable Policies

We show here that the class of cone policies will maximize throughput in the interconnection networks described. These algorithms select either a mode from $\mathcal{M}(t)$ or a batch-mode from $\hat{\mathcal{M}}$ based the backlog vector $X(t)$ at time t, and effective service to the set of queues, as reflected in the vector $(D^m - E^m)$ for each mode m.

We restrict the modes considered in time t to those which do not forward any cells to finite buffer queues. Instead we capture them in the batch-modes. Let

$$\hat{\mathcal{M}}(t) = \{m \in \mathcal{M}(t) : (D_q^m - E_q^m) \geq 0 \text{ for all } q \text{ with } B_q < \infty\} \tag{5.1}$$

be the original modes available in time t that have no net forwarding to finite buffer queues.

If the backlog at time t is given by $X(t)$, and there is no batch-mode already operating, the cone algorithm with positive weight parameters $\{w_1, w_2, ..., w_Q\}$ selects the mode m from the union of sets $\mathcal{M}(t)$ and $\hat{\mathcal{M}}$ so as to maximize the weighted sum of backlog multiplied by effective change to queues. That is, the cone algorithms select $m(t)$ to satisfy

$$\max_{m \in \hat{\mathcal{M}}(t) \cup \hat{\mathcal{M}}} \sum_{q \in \mathcal{Q}} w_q X_q(t)(D_q^m - E_q^m) \tag{5.2}$$

If more than one mode maximizes (5.2), then ties are broken by using the same mode as in the previous timeslot (or same batch-mode as previous batch). Beyond that, one can be chosen arbitrarily.

Cone algorithms correspond to backlog differential policies [17, 1] in that they favor the modes which forward from larger upstream to smaller downstream buffers. As the difference between a served queue and its immediate downstream neighbor increases, the weighting to that mode increases.

The backlog differential policy has the effect of load balancing the network, and the w_q weights correspond to relative balancing levels. For example, increasing a particular w_q value while keeping others fixed will give greater weight to modes that forward from that queue, and lower weight to those forwarding to that queue.

The geometric intuition of backlog evolution also exposes a structure to the mode selection algorithm. In particular, there is a neighborhood relationship between modes in that certain modes share boundaries backlog space division described in detail in [11]. As the backlog changes, it moves from one region to another, utilizing different modes for each region. In particular, for large backlog values, one could consider only nearby regions rather than all available modes in each timeslot. This could lead to lower complexity and distributed implementations of cone algorithms, as discussed further for the case of single-pass networks in [11].

It is important to notice that cone algorithms will only select a batch-mode if some 'better' true node has become infeasible. This can be seen from the definition of a batch mode.

If ties are broken by selecting a true mode over a batch-mode, this means that the reason m^* isn't selected must be due to $m^* \notin \mathcal{M}(t)$. For example, this corresponds to the case where the algorithm would *like to* forward a cell between two queues, but the downstream buffer is full.

Theorem 5.1 *For an interconnection network with external arrival rate $\rho \in \mathcal{R}$ from (4.6) and buffer sizes satisfying (4.3) defined over all feasible modes, the cone algorithms (5.2) guarantee maximal throughput for any set of weighting factors $\{w_1, w_2, ..., w_Q\}$ with $w_q > 0$.*

Proof: Theorem theorem 5.1 follows from theorem 7.1 in [14]. There, it is shown that K-delayed cone schedules guarantee maximal throughput. The argument in proposition 4.1 shows that eventually the service vectors $D^{\hat{m}}$ and $E^{\hat{m}}$ fully respresent the service rates. This will happen after a finite number of timeslots selecting the same batch-mode, which is sufficient to show that the policy is delayed PCS. The argument in [14] actually applies only to single stage switches, but the extension to feed-forward networks is found in [12].

6 Conclusions and Future Research

We have established a definition of the stability capacity for interconnection networks with finite internal buffers. These buffers can reduce the potential throughput region of such a network, but as buffers grow, the stability region approaches and eventually reaches the same as that for the related infinite-buffer network.

In order to achieve the maximum possible throughput, we have shown that cone schedules can be integrated with batch-modes, which apply a sequence of modes and manage the buffer levels throughout the network. This work establishes an analytical framework for allocating buffer sizes throughout a network, and indicates that the finite internal buffers do not need to be large to maintain the same throughput rate as the infinite-buffer analysis would indicate.

It remains for further consideration how one can efficiently calculate the best batch-mode sequences. We have seen that ordering indeed matters for stability, and this is the subject of ongoing investigation. This also allows one to consider many extensions, such as the optimal distribution of finite buffer capacity in shared networks, and the value added of increasing buffer size at different network stations.

References

[1] J. G. Dai and W. Lin. Maximum pressure policies in stochastic processing networks. *Operations Research*, 53(2):197–218, 2005.

[2] W.J. Dally and C.L. Seitz. Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Transactions on Computers*, 36(5):547–553, 1987.

[3] W.J. Dally and B. Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann, 2003.

[4] P. Giaccone, E. Leonardi, and D. Shah. On the maximal throughput of networks with finite buffers and its application to buffered crossbars. *IEEE INFOCOM*, 2005.

[5] P. Giaccone, E. Leonardi, and D. Shah. Throughput region of finite-buffered networks. *IEEE Transactions on Parallel and Distributed Systems*, 18(2):251–263, 2007.

[6] N. Kahale and P. Wright. Dynamic global packet routing in wireless networks. *IEEE INFOCOM*, pages 1414–1421, 1997.

[7] E. Leonardi, M. Mellia, M.A. Marsan, and F. Neri. On the throughput achievable by isolated and interconnected input-queueing switches under multiclass traffic. In *IEEE INFOCOM*, 2002.

[8] N. McKeown, A. Mekkittikul, V. Anantharam, and J. Walrand. Achieving 100% throughput in an input-queued switch. *IEEE Transactions on Communications*, 47(8):1260–1267, 1999.

[9] M.J. Neely, E. Modiano, and C.E. Rohrs. Power allocation and routing in multibeam sattelites with time-varying channels. *IEEE/ACM Transactions on Networking*, 11(1):138–152, 2003.

[10] M.J. Neely, E. Modiano, and C.E. Rohrs. Dynamic power allocation and routing for time varying wireless networks. *IEEE JSAC*, 2005.

[11] K. Ross and N. Bambos. Local search scheduling algorithms for maximal throughput in packet switches. In *IEEE INFOCOM*, 2004.

[12] K. Ross and N. Bambos. Capacity maximizing packet scheduling algorithms for interconnection networks with finite buffers. In *IEEE GLOBECOM*, 2006.

[13] K. Ross and N. Bambos. Job scheduling for maximal throughput in autonomic computing systems. In *International Workshop on Self-Organizing Systems*, 2006.

[14] K. Ross and N. Bambos. Capacity maximizing packet scheduling algorithms for interconnection networks with finite buffers. *To appear in IEEE Transactions on Networking*, 2007.

[15] J. MacGregor Smith and F.R.B.Cruz. The buffer allocation problem for general finite buffer queueing networks. *IIE Transactions*, 37:343–365, 2005.

[16] L. Tassiulas and P.P. Bhattacharya. Allocation of interdependent resources for maximal throughput. *Stochastic Models*, 16(1), 1999.

[17] L. Tassiulas and A. Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Transactions on Automatic Control*, 37(12):1936–1948, 1992.

[18] K. Yoshigoe and K.J. Christensen. An evolution to crossbar switches with virtual output queueing and buffered crosspoint. *IEEE Network*, pages 48–56, September 2003.