

Dynamic Scheduling of Optical Data Bursts in Time-Domain Wavelength Interleaved Networks

Kevin Ross and Nicholas Bambos
Departments of MS&E and EE, Stanford University
Terman Engineering Center, Stanford, CA 94305-4026
kross, bambos @stanford.edu

Krishnan Kumaran, Iraj Saniee, Indra Widjaja
Bell Laboratories, Lucent Technologies
Murray Hill, NJ 07974
kumaran, iis, iwidjaja @lucent.com

Abstract

We consider the problem of scheduling bursts of data in an optical network with an ultra-fast tunable laser and a fixed receiver at each node. In [10] we considered the static scheduling problem of meeting demand in the minimal time. Here we substantially extend these results to the case of online, dynamic scheduling. Due to the high data rates employed on the optical links, the burst transmissions typically last for very short times compared to the round trip propagation times between source-destination pairs. A good schedule ensures that (i) there are no transmit/receive conflicts, (ii) throughput is maximized, and (iii) propagation delays are observed. We formulate the scheduling problem as a generalization of the well-known crossbar switch scheduling problem. We show that the algorithms presented in [10] can be implemented in dynamic form to give 100% throughput. Further, we show that one of the more intuitive solutions does not lead to maximal throughput. In particular, we show advantages of adaptive batch sizes rather than fixed batch sizes for both throughput and performance.

1 Introduction

We consider online scheduling of data bursts among a multiplicity of source and destination nodes that are geographically separated. This is a direct extension of the model analyzed in [10] to the case of dynamic demands.

This problem arises in a recently developed wavelength-division multiplexed (WDM) transport network, called *Time-domain Wavelength Interleaved Network* (TWIN) [13], that bridges the gap between the relatively high data

rate provided by the optical layer and the relatively low data rate required for a typical end-to-end service. TWIN performs efficient traffic grooming without resorting to electronic grooming, as is done today through SONET cross-connects [6, 7]. Briefly, TWIN defines an optical network with a simple core architecture consisting of wavelength-selective cross-connects capable of routing incoming wavelengths to the appropriate outgoing ports or fibers. The cross-connect configuration in the core is static once connections have been established among source and destination nodes. Fast switching of data in the core is emulated through the use of fast tunable lasers at the edge. In TWIN, a unique wavelength λ_j is assigned to each destination j , and sources that have data to transmit to a particular destination use the wavelength assigned to that destination. Thus the wavelength connections from various sources to a particular destination j can be viewed as an optical multipoint-to-point connection of wavelength j rooted at the destination. To ensure that bursts of the same wavelength do not arrive at the same destination simultaneously, sources must coordinate their transmissions in time domain. TWIN relies on scheduling to avoid such destination conflicts/collisions.

Figure 1 shows a simple TWIN architecture where multipoint-to-point optical connections rooted at two destinations (nodes 6 and 7) have been configured. Typically, such reconfigurations occur at a relatively long time scale (e.g., hours, days or weeks). The brief data transmission on a particular wavelength is called a burst. For example, in Figure 1, node 1 interleaves its transmissions to nodes 6 and 7 by tuning its laser to λ_6 when it wants to transmit a burst to node 6 and to λ_7 when it wants to transmit a burst to node 7. Each node in the network simply performs self-routing of bursts based on their colors (i.e., wavelengths).

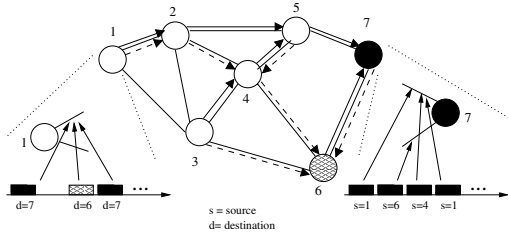


Figure 1. Logical multipoint-to-point trees, one for each of the two destinations, overlaid on top of a physical topology.

For example, node 2 will route an incoming burst of λ_6 to node 4 and an incoming burst of λ_7 to node 5. In summary, the TWIN architecture allows route determination based on configuration of the wavelength-selective cross-connects to be performed at a relatively long time scale (hours), while the individual burst forwarding based on assignment of appropriate colors at the source to be performed at a relatively small time scale (microseconds). Further details of TWIN can be found in [10] and [13].

We assume that time is slotted at each receiver and each slot may contain exactly one burst. At each time slot, the scheduler needs to assign the appropriate wavelength to each tunable laser. The scheduler must avoid collisions at destinations and make efficient use of the time slots. Unlike the well-known problem of scheduling packets in a single node, scheduling in TWIN has to deal with an optical network of arbitrary topology and link propagation delays.

The propagation delay between a source node i and a destination node j is due to the length of the fiber on the transmission tree with j as its root (see [10]). We consider a dynamic approach with bursts arriving to an online scheduler. We show that a class of *batching* policies introduced in their static form in [10] achieve maximum throughput for admissible load.

This paper will proceed as follows. In section 2 we describe the model and formulate the scheduling problem. In section 3 we address the intuitive solution of fixed batch lengths, contrasting it with adaptive batch policies in section 4. In section 5 we demonstrate a particular dynamic adaptive batch policy and show a sample trace comparing adaptive and fixed batch policies. We discuss conclusions in section 6.

2 Dynamic Transmission Scheduling in TWIN: Problem Formulation

Consider a network of C nodes and denote their set by $\mathcal{C} = \{1, 2, 3, \dots, C\}$. For each communicating source-destination pair $(i, j) \in \mathcal{C} \times \mathcal{C}$ we assign a logical queue, where the data bursts originating in node i and destined to node j are queued up while waiting to be transmitted.

The system operates in slotted time and one data burst can be transmitted from each source in each time slot. Let $n \in \{1, 2, 3, \dots\}$ index the time slots.

In [10] we describe the physical network in more detail. We assume tree-based routing which means that internal conflicts in the network are avoided by scheduling only at the source and destination nodes. There is a fixed and known delay δ_{ij} from sending a burst from source i to its arrival at destination j .

The request traffic forms a dynamic arrival process, $A(n) = \{A_{ij}(n), (i, j) \in \mathcal{Q}\}$, where $A_{ij}(n)$ is the (integer) number of arriving bursts in queue (i, j) in time slot n . The scheduler chooses a service configuration matrix $S(n) = \{S_{ij}(n), (i, j) \in \mathcal{Q}\} \in \mathcal{S}$ at time slot n , where $S_{ij}(n)$ is 1 when a burst is sent from i to j at time slot n , and 0 otherwise. Hence, each service configuration $S(n)$ - in the set of all possible ones \mathcal{S} - describes the set of queues from which a burst can be transmitted if $S(n)$ is chosen. The system workload matrix is denoted $D(n) = \{D_{ij}(n), (i, j) \in \mathcal{Q}\}$, where $D_{ij}(n)$ is the number of waiting bursts in queue (i, j) at the end of time slot n . As easily seen, the evolution of each queue (i, j) is governed by

$$D_{ij}(n+1) = D_{ij}(n) + A_{ij}(n+1) - S_{ij}(n+1) \mathbf{1}_{\{D_{ij}(n) > 0\}} \quad (1)$$

The indicator function ensures that an empty queue is not served. We assume that each queue is *store-and-forward*, that is, a burst arriving in time slot will be available for transmission in the following time slot.

We assume that the traffic or arrival process to each queue (i, j) has a long-term average rate $\rho_{ij} > 0$, that is,

$$\lim_{n \rightarrow \infty} \frac{\sum_{k=1}^n A_{ij}(k)}{n} = \rho_{ij} \quad (2)$$

The average traffic load of the system is represented by the matrix $\rho = \{\rho_{ij}, (i, j) \in \mathcal{Q}\}$. The existence of long-term average traffic rates is the only assumption of the arrival process in the analysis that follows.

For each source-destination pair (i, j) we assume that there is a fixed, known propagation delay δ_{ij} for sending a data burst from node i to j . Hence, a burst sent at time n from i to j will arrive at its destination at time $n + \delta_{ij}$. We assume for simplicity in the exposition that the propagation delay is an integer multiple of time slots.

Since each source (and each destination) can send (or receive) a single burst at any time slot, we can immediately see that the throughput or stability region could not exceed

$$\left\{ \rho : \sum_j \rho_{ij} \leq 1, \forall i; \sum_i \rho_{ij} \leq 1, \forall j \right\} \quad (3)$$

This is actually known to be the stability region when the propagation delays are zero. We will show below that this

remains the stability/throughput region even in the presence of propagation delays.

In the case of zero propagation delays, the service $\{S(n)\}$ must satisfy the following constraints

$$\sum_{j \in \mathcal{C}} S_{ij}(n) \leq 1, \forall i \text{ and } \sum_{i \in \mathcal{C}} S_{ij}(n) \leq 1, \forall j \quad (4)$$

for each time slot n . At most one burst is sent by each source and one is received by each destination.

With a nonzero propagation delay matrix $\delta = \{\delta_{ij}, i, j \in \mathcal{C}\}$, the service configuration constraints must involve time delays. For example, two different sources i and i' may send to destination j at different time slots, but their propagation delays, δ_{ij} and $\delta_{i'j}$ respectively, may cause a conflict upon arrival at the destination. Hence, the service schedule $\{S(n), n = 1, 2, \dots\}$ must satisfy

$$\sum_j S_{ij}(n) \leq 1, \forall i \quad (5)$$

$$\sum_i S_{ij}(n - \delta_{ij}) \leq 1, \forall j \quad (6)$$

for all times slots n . We aim to construct a service policy $\{S(n), n \in \mathcal{N}\}$ that satisfies (5), (6) and maintains input-output flow balance (stability)

$$\rho_{ij}^{out} = \lim_{N \rightarrow \infty} \frac{\sum_{n=1}^N S_{ij}(n) \mathbf{1}_{\{D_{ij}(n-1) > 0\}}}{N} \quad (7)$$

must equal ρ_{ij} from equation 2 for each (i, j) pair. Such a policy is said to achieve 100% throughput if this is satisfied for every ρ matrix satisfying equation 3.

We assume a centralized scheduler receives demand requests from each node and signals bursts to be sent. These requests could come periodically or at the end of the batches described in the next sections. Nodes greater distance from the central scheduler could send more frequently than closer nodes to synchronize information. For simplicity in the exposition we ignore the additional delays caused by this except to note here that this additional bounded delay strengthens the case for adaptive batch sizes as it does not affect the stability proof.

3 Fixed Batch Lengths: The TDM equivalent

An intuitive solution attempt to the scheduling problem described in section 2 is analogous to time-division multiplexing (TDM) in a switch. Scheduling is divided into batches of fixed length B timeslots. The bursts waiting to be sent at a particular timeslot n are scheduled to be sent in the timeslots from $n + 1$ until $n + B$. The demand is considered at timeslots $\{0, B, 2B, \dots\}$ and bursts are scheduled at each batch to meet demand.

Bursts are scheduled to arrive within the same batch they were sent. This criteria necessitates some down-time in the schedule. In particular, no bursts can be sent in the last δ_{\min} timeslots of the batch, where $\delta_{\min} = \min_{i, j \in \mathcal{C} \times \mathcal{C}} \delta_{ij}$. Bursts sent after this time would arrive in the next batch, potentially causing further conflicts. This means that at most $B - \delta_{\min}$ of the B timeslots can be utilized, reducing the potential throughput to $\frac{B - \delta_{\min}}{B}$. Therefore if $\delta_{\min} > 0$, for any fixed, finite batch length B throughput is strictly less than 100%.

4 Adaptive Batching Policies and Rate-Stability

In this section we introduce a class of dynamic scheduling policies which do guarantee 100% throughput. This is done by adapting the length of the batches according to the waiting bursts, rather than fixing it in advance.

Suppose a demand $D = \{D_{ij}, i, j \in \mathcal{C}\}$ is given (that is, some batched up burst transmission requests) and also some static feasible transmission schedule Σ (possibly suboptimal), which clears the demand D in $B(D)$ time slots. We examine such static batch schedules in the next section, but let us first see here how we would use such a schedule to process dynamically arriving data bursts. We leverage a technique introduced in [2, 3].

Starting at time $T_0 = 0$, we inductively define for $N = 0, 1, 2, 3, \dots$ the formation of batches and their processing intervals, as follows.

- At time T_N we schedule the batch demand $D(T_N) \neq 0$ of bursts waiting in the queues at T_N , according to the static schedule Σ , which clears it at time $T_{N+1} = T_N + B(D(T_N))$. Bursts in $D(T_N)$ are processed in the time interval $[T_N + 1, T_N + B(D(T_N))]$.
- New bursts arriving throughout the time interval $[T_N + 1, T_N + B(D(T_N))]$ are not processed, but stored until time $T_{N+1} = T_N + B(D(T_N))$. Thus, at time T_{N+1} the previous batch $D(T_N)$ has been completely cleared, and a new batch $D(T_{N+1})$ has been formed by accumulating the bursts that arrived after T_N and up to time T_{N+1} . The batch $D(T_{N+1})$ is then scheduled at T_{N+1} according to the static schedule Σ and the process repeats itself.
- If no new burst arrives while a batch is served, or $D(T_{N+1}) = 0$, we simply increment by default the time register $T_{N+2} = T_{N+1} + 1$ by one time slot.

According to the above scheme, the evolution of the increasing unbounded time sequence $\{T_N, N = 1, 2, 3, \dots\}$ is given by the following induction formula:

$$T_{N+1} = \begin{cases} T_N + B(D(T_N)), & \text{if } D(T_N) \neq 0 \\ T_N + 1, & \text{if } D(T_N) = 0 \end{cases} \quad (8)$$

Note that the N^{th} batch $D(T_N)$ is comprised of the jobs that arrived throughout the interval $[T_{N-1} + 1, T_N]$ and is processed throughout the interval $[T_N + 1, T_{N+1}]$.

We proceed by identifying some key necessary conditions for a service policy $\{S(n)\}_{n=1}^\infty$ to induce *rate-stability*. That is, to maintain the burst arrival rate to each queue ρ_{ij} is equal to the departure rate ρ_{ij}^{out} as in (7) and provide flow conservation through the system.

Proposition 4.1 (Rate Stability and Flow Conservation)

When the system operates under any service policy $\{S(n)\}_{n=1}^\infty$ such that

$$\lim_{n \rightarrow \infty} \frac{D(n)}{n} = 0, \quad (9)$$

where 0 is the zero matrix, it is rate-stable, so that $\rho_{ij}^{out} = \rho_{ij}$ from equations 2 and 7 for each $i, j \in \mathcal{C} = \{1, 2, 3, \dots, C\}$.

Proof: Note that $D_{ij}(n) = D_{ij}(0) + \sum_{k=0}^n A_{ij}(k) - \sum_{k=0}^n S_{ij}(k) \mathbf{1}_{\{D_{ij}(n-1) > 0\}}$. Dividing by n , letting $n \rightarrow \infty$, and using (9), we immediately obtain $\rho_{ij}^{out} = \rho_{ij}$. ■

Proposition 4.2 (Stability of Dynamic Batch Schedules)

When the system operates under any adaptive batch service policy $\{S(n)\}_{n=1}^\infty$ such that

$$\lim_{N \rightarrow \infty} \frac{T_N - T_{N-1}}{T_N} = 0, \quad (10)$$

then equation (9) holds, so it is rate-stable.

Proof: Note that condition (10) implies $\lim_{N \rightarrow \infty} \frac{T_{N-1}}{T_N} = 1$. Note also that, due to the structure of each batch, we have $D(T_N) = \sum_{k=T_{N-1}+1}^{T_N} A(k)$. Therefore, since $\{T_N\}_{N=1}^\infty$ is an increasing and unbounded sequence, using (2) we get

$$\begin{aligned} & \lim_{N \rightarrow \infty} \frac{D(T_N)}{T_N} \\ &= \lim_{N \rightarrow \infty} \frac{\sum_{k=T_{N-1}+1}^{T_N} A(k)}{T_N} \\ &= \lim_{N \rightarrow \infty} \frac{\sum_{k=1}^{T_N} A(k)}{T_N} - \lim_{N \rightarrow \infty} \frac{\sum_{k=1}^{T_{N-1}} A(k)}{T_{N-1}} \cdot \frac{T_{N-1}}{T_N} \\ &= \rho - \rho \cdot 1 = 0 \end{aligned} \quad (11)$$

where the above limit calculations hold for each individual component $i, j \in \mathcal{C}$ of the matrices D, A, ρ . Now, given any time slot $n \in \{1, 2, 3, \dots\}$, let it fall in the $N(n)$ batch interval, that is, $T_{N(n)} < n \leq T_{N(n)+1}$. Consider the bursts in the system at time slot n . Note that each such burst is included in one of the demands $D(T_{N(n)})$ or $D(T_{N(n)+1})$. Therefore, we have

$$\begin{aligned} & \lim_{n \rightarrow \infty} \frac{D(n)}{n} \\ & \leq \lim_{n \rightarrow \infty} \frac{D(T_{N(n)}) + D(T_{N(n)+1})}{n} \\ & \leq \lim_{n \rightarrow \infty} \frac{D(T_{N(n)})}{T_{N(n)}} + \lim_{n \rightarrow \infty} \frac{D(T_{N(n)+1})}{T_{N(n)+1}} \cdot \frac{T_{N(n)+1}}{T_{N(n)}} \\ & \leq 0 + 0.1 = 0 \end{aligned} \quad (12)$$

This completes the proof of proposition 4.2. ■

We are now ready to construct a key sufficient condition for stability, which is directly based on properties of the static batch schedule Σ that is used to schedule bursts for transmission within each individual batch. Recall that the demand $D(T_N)$ of the N^{th} batch is comprised of the bursts that arrived in the time interval $[T_{N-1} + 1, T_N]$; therefore, $D(T_N) = \sum_{k=T_{N-1}+1}^{T_N} A(k)$. This demand will be cleared by Σ at time $T_{N+1} = T_N + B(D(T_N))$.

Proposition 4.3 (The Role of the Static Batch Schedule)

If the static schedule Σ that is used to process the jobs within each batch has the asymptotic property

$$\lim_{N \rightarrow \infty} \frac{B(D(T_N))}{T_N - T_{N-1}} \leq 1, \quad (13)$$

then (10) holds, so the system is rate-stable.

Proof: Arguing by contradiction, suppose that 13 holds and $\limsup_{N \rightarrow \infty} \frac{T_N - T_{N-1}}{T_N} = \epsilon > 0$. Let $\{T_k\}_{k=1}^\infty$ be an increasing unbounded subsequence of $\{T_N\}_{N=1}^\infty$ such that $\lim_{k \rightarrow \infty} \frac{T_k - s(T_k)}{T_k} = \epsilon$ where $s(T_k)$ is the start of the previous batch to T_k .

Further, let $\phi = \limsup \frac{s(T_k) - s(s(T_k))}{s(T_k)}$ and let $\{T_m\}_{m=1}^\infty$ be an increasing subsequence of $\{T_k\}_{k=1}^\infty$ on which $\phi = \limsup \frac{s(T_m) - s(s(T_m))}{s(T_m)}$. Clearly $\phi \leq \epsilon$.

Note that $\frac{s(T_k)}{T_k} \rightarrow 1 - \epsilon$. Now we have two cases:

Case 1: $\phi > 0$.

$$\begin{aligned} \epsilon &= \lim_{m \rightarrow \infty} \frac{T_m - s(T_m)}{T_m} \\ &= \lim_{m \rightarrow \infty} \frac{T_m - s(T_m)}{s(T_m) - s(s(T_m))} \frac{s(T_m) - s(s(T_m))}{s(T_m)} \frac{s(T_m)}{T_m} \\ &\leq 1 \cdot \phi \cdot (1 - \epsilon) < \epsilon \end{aligned} \quad (14)$$

This is a contradiction.

Case 2: $\phi = 0$.

In this case we have $\lim_{k \rightarrow \infty} \frac{s(T_k) - s(s(T_k))}{s(T_k)} = 0$. Then $\lim_{k \rightarrow \infty} \frac{D_{ij}(s(T_k)) + \delta_{ij}}{s(T_k)} = 0$ for all $(i, j) \in \mathcal{C} \times \mathcal{C}$ since $D_{ij}(s(T_k)) + \delta_{ij} \leq s(T_k) - s(s(T_k))$ whenever $D_{ij}(s(T_k)) > 0$.

$$\begin{aligned} \epsilon &= \lim_{k \rightarrow \infty} \frac{T_k - s(T_k)}{T_k} \\ &\leq \lim_{k \rightarrow \infty} \frac{\sum_{ij} (D_{ij}(s(T_k)) + \delta_{ij}) \frac{s(T_k)}{T_k}}{s(T_k)} \\ &= 0 \end{aligned} \quad (15)$$

This completes the proof. ■

Thus we establish conditions which lead to 100% throughput.

5 The Simple-Batch Policy

In [10] we described the simple-batch policy¹. It uses an extension of the 0-delay solution to the problem with nonzero propagation delay.

We note that the Birkhoff-von Neumann theorem [4, 12] permits us to construct a 0-delay schedule using a fixed number of permutation matrices. More precisely, dropping the time variable for notational ease, given a demand matrix D , the following lemma holds as a direct consequence of this theorem:

Lemma 5.1 *Let $\max\{\max_i \sum_j D_{ij}, \max_j \sum_i D_{ij}\}$ be denoted B^* . It is then true that $D = \sum_{k=1}^K \alpha^k S^k$, where $K \leq C^2 - 2C + 2$ for some choice of permutation matrices and their derivatives, $\alpha^k > 0$, and $\sum_{k=1}^K \alpha^k = B^*$.*

The lemma expresses the fact that, among the set of all possible permutation matrices and their derivatives ($\sim 2C!$), it is possible to construct a schedule for D consisting of less than C^2 permutation matrices, with matrix S^k being repeated exactly α^k times. Each matrix has unit entries ($S_{ij}^k = 1$) iff the source-destination pair (i, j) is picked for transmission, and 0 otherwise. The theorem also guarantees that, if the D_{ij} are all integral, so are the α^k . A valid 0-delay schedule can then be obtained by successively applying the S^k in any chosen sequence so that each S^k appears α^k times. Decomposing the demand matrix into permutation matrices efficiently is discussed in [5].

Assume now that one has performed such a decomposition and obtained S^k and α^k for a 0-delay schedule. How does one construct a valid δ -delay schedule by utilizing the 0-delay schedule? The main idea is to transmit a batch of bursts with α^k copies of permutation S^k , wait until all transmissions have reached their destinations, and repeat the process with the next batch of α^{k+1} copies of S^{k+1} . More formally,

Theorem 5.1 *Let $\delta_{\max}^k \triangleq \max_{i,j} (S_{ij}^k \delta_{ij})$ and $\delta_{\min}^k \triangleq \min_{i,j} (S_{ij}^k \delta_{ij})$, where δ_{ij} is the delay from source i to destination j . Then the following procedure gives a valid δ -delay schedule: Order the S^k, α^k in any fixed manner, say as $\alpha_1 \geq \alpha_2 \geq \dots$, and repeat the following steps for $k = 1, 2, \dots, K$*

- Apply the schedule S^k exactly α^k times in successive time slots; i.e., as a batch
- Wait $\delta_{\max}^k - \delta_{\min}^{k+1}$ time slots.

¹In [10] we also proposed the Twin Iterative Scheduler (TIS), a static batching policy based on a simple heuristic introduced by Kahale and Wright [8]. This is effective on lower congestion networks but does not guarantee 100% throughput for all admissible traffic loads.

Proof It is easy to see that this procedure produces a valid δ -delay schedule. Repeated successive applications of the same S^k does not produce any conflicts within batches, since successively transmitted bursts only arrive successively for any given (i, j) pair. The waiting time between distinct permutations is chosen in such a way that the last arrival from the previous batch occurs before the first arrival of the next batch, which avoids a conflict across successive batches.

This simple batched schedule with appropriately chosen waiting times between batches then has a total length given

$$B_\delta = B^* + \sum_{k=1}^{K-1} (\delta_{\max}^k - \delta_{\min}^{k+1}) \equiv B^* + \Delta \quad (16)$$

Lemma 5.2 *The algorithm described above achieves*

$$\lim_{N \rightarrow \infty} \frac{B(D(T_N))}{T_N - s(T_N)} \leq 1 \quad (17)$$

on any increasing time sequence $\{T_N\}_{N=1}^\infty$ on which $\lim_{N \rightarrow \infty} \frac{T_N - s(T_N)}{T_N} = \theta > 0$.

Proof: First observe that

$$\begin{aligned} & \lim_{N \rightarrow \infty} \frac{\sum_{n=s(T_N)}^{s(T_N)} A(n)}{T_N - s(T_N)} \\ &= \lim_{N \rightarrow \infty} \frac{\sum_{n=0}^{T_N} A(n) \frac{T_N}{T_N - s(T_N)}}{T_N} \\ & \quad - \lim_{N \rightarrow \infty} \frac{\sum_{n=0}^{s(T_N)} A(n) \frac{s(T_N)}{T_N - s(T_N)}}{s(T_N)} \\ &= \rho\left(\frac{1}{\theta}\right) - \rho\left(\frac{1}{\theta} - 1\right) \\ &= \rho \end{aligned} \quad (18)$$

since $\theta > 0$. Also observe from equation 16 and lemma 5.1 that the total delay within each batch under this policy is bounded by $\Delta \leq C^2 \delta_{\max}$ where δ_{\max} is over all (i, j) .

Now according to the policy,

$$\begin{aligned} & \lim_{N \rightarrow \infty} \frac{B(D(T_N))}{T_N - s(T_N)} \\ & \leq \lim_{N \rightarrow \infty} \frac{\max_{i,j} \left\{ \sum_{i,j} D_{ij}(T_N) \right\} + C^2 \delta_{\max}}{T_N - s(T_N)} \\ & = \lim_{N \rightarrow \infty} \frac{\max_{i,j} \left\{ \sum_{j,i} \sum_{n=s(T_N)}^{T_N} A_{ij}(n) \right\} + C^2 \delta_{\max}}{T_N - s(T_N)} \\ & = \lim_{N \rightarrow \infty} \max_{i,j} \left\{ \frac{\sum_{j,i} \sum_{n=s(T_N)}^{T_N} A_{ij}(n)}{T_N - s(T_N)} \right\} + \lim_{N \rightarrow \infty} \frac{C^2 \delta_{\max}}{T_N - s(T_N)} \\ & = \max \left\{ \max_i \sum_j \rho_{ij}, \max_j \sum_i \rho_{ij} \right\} + 0 \\ & \leq 1 \end{aligned} \quad (19)$$

This completes the proof. \blacksquare

5.1 Sample Trace

We compare the performance of the simple adaptive batch policy described here with its corresponding fixed batch version. Figure 2 shows the trace of the workload for three different policies with the same arrival sequence. The three policies compared are adaptive batch, and fixed batch with small (10 timeslots) and large (50 timeslot) batches.

The example is of the simplest possible TWIN network, with two sending nodes and one receiving node. The bursts from each sending node arrive according to a poisson distribution of rate 0.2 and 0.79 to the two nodes respectively. There are corresponding propagation delays of one timeslot of one and three timeslots.

We observe that the adaptive batch performs better than both fixed batch policies. For the smaller batch size, the limited throughput leads to a growing total workload in the system. With the larger batch, the effect of lost throughput is reduced, but the initial lag before the first batch begins serving increases workload to the system from the beginning. Initially, bursts experience a shorter delay with the smaller batch size but the growing workload leads to growing delay to each burst.

The simulation trace illustrates the case where the small batch size is quickly leading to instability, the large batch size is *just* unstable (arriving at rate 0.99 with maximum service of 0.98) and the adaptive batch policy is stable.

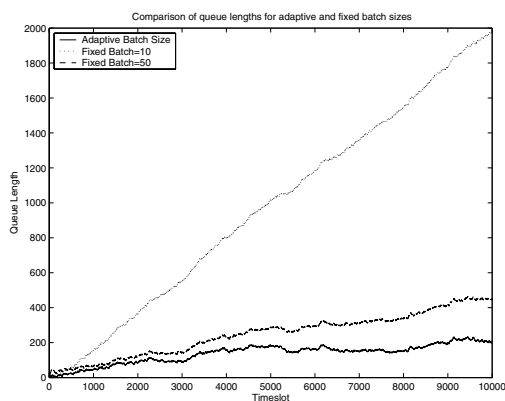


Figure 2. The trace of a simple TWIN network with adaptive and fixed batching policies

6 Conclusions

We have introduced adaptive batching policies for scheduling optical data bursts in time-domain wavelength interleaved networks. We have extended previous results in this scheduling domain from static to dynamic scheduling. We have shown a simple adaptive batching policy to guarantee 100% throughput and demonstrated that the intuitive fixed batch policy does not.

References

- [1] S. Alexander *et al.* "A Precompetitive Consortium on Wide-Band All-Optical Networks," *J. Lightwave Technology*, vol. 11, pp. 714-735, May. 1993.
- [2] M. Armony and N. Bambos, "Queueing networks with interacting service resources," *Proc. of 1999 Allerton Conference*, Urbana, IL, pp. 42-51, 1999.
- [3] M. Armony and N. Bambos, "Queueing dynamics and maximal throughput scheduling in switched processing systems," *Technical Report SU NETLAB-2001-09/01*, Stanford University, 2001.
- [4] G. Birkhoff, "Tres observaciones sobre el algebra lineal," *Univ. Nac. Tucumán Rev. Ser. A*, vol. 5, pp. 147-151, 1946.
- [5] C. Chang, W. Chen and H. Huang, "Birkhoff-von Neumann input buffered crossbar switches," in *Proc. IEEE INFOCOM'00*, 2000.
- [6] A. Chiu and E. Modiano, "Traffic grooming algorithms for reducing electronic multiplexing costs in WDM ring networks," *J. Lightwave Technology*, vol. 18, pp. 2-12, Jan. 2000.
- [7] O. Gertsel, G. Sasaki and R. Rawaswami, "Combined WDM and SONET network design," *Proc. IEEE INFOCOM'99*, 1999.
- [8] N. Kahale and P. Wright, "Dynamic global packet routing in wireless networks", in *Proc. IEEE INFOCOM'97*, pp. 1414-1421, 1997.
- [9] N. McKeown, "The iSLIP scheduling algorithm for input-queue switches," *IEEE Trans. on Networking*, vol. 7, pp. 188-201, Apr. 1999.
- [10] Ross, K., Bambos, N., Kumaran, K., Saniee, I., Widjaja, I., Scheduling Bursts in Time-Domain Wavelength Interleaved Networks, Technical Report SU NETLAB-2002-12/1, Engineering Library, Stanford University, Stanford, CA 94305; December 2002. Submitted for publication.
- [11] R. E. Tarjan, "Data structures and network algorithms," *Soc. Ind. Appl. Mathematics*, PA, Nov. 1983.
- [12] J. von Neumann, "A certain zero-sum two-person game equivalent to the optimal assignment problem," *Contribution to the Theory of Games*, vol. 2, pp. 5-12, Princeton University Press, New Jersey, 1953.
- [13] I. Widjaja, I. Saniee, R. Giles, and D. Mitra, "Light core and intelligent edge for a flexible, thin-layered and cost-effective optical transport network," to appear in *IEEE Communications Magazine*, May 2003.