Burst Scheduling Networks: Flow Specification and Performance Guarantees*

Simon S. Lam and Geoffrey G. Xie

Department of Computer Sciences
The University of Texas at Austin
Austin, Texas 78712-1188

1 Introduction

We present a class of packet switching networks, called Burst Scheduling Networks, designed to provide throughput, delay, and delay jitter guarantees. These performance guarantees are derived from the delay guarantee of a VC server, and a new traffic model called Flow Specification [2, 3].

The delay guarantee of a VC server has several desirable properties, including the following *firewall* property: The guarantee to a flow is unaffected by the behavior of other flows sharing the same server. There is no assumption that sources are flow-controlled or well-behaved.

Each guaranteed flow is modeled as a sequence of bursts, each of which is a sequence of packets. Bursts are needed to specify two types of jitter bounds: over the delays of packets in a burst, and over the delays of bursts in a flow. For video flows, each encoded picture is naturally modeled by a burst. The model is also appropriate for audio and data flows that require delay and delay jitter guarantees.

With the new traffic model, a flow can be partitioned into intervals (bursts) that have substantially different average rates; the first packet of a burst carries information on the size and average rate of the burst. Switches are designed to process flows efficiently in bursts [2].

2 Delay guarantee and its properties

Consider a number of traffic sources and a service facility (a single server or a network). Each source generates a sequence of packets, called a *flow*. Prior to generating packets, the source of flow f requests for a reserved rate from the facility. Let $r(f)$ bits/second be the reserved rate allocated to flow f. For an arbitrary packet p, its length in bits is denoted by $l(p)$, its arrival time by $A(p)$, and its departure time by $L(p)$. The delay of packet p is $L(p) - A(p)$. The maximum packet size is l_{max}.

* Research sponsored in part by National Science Foundation grant no. NCR-9004464 and by the NSA INFOSEC University Research Program. Papers of the Networking Research Laboratory are available from http://www.cs.utexas.edu/users/lam/NRL.

Our concept of a delay guarantee is based upon the *virtual clock of a flow*. Let $priority(f)$ denote the virtual clock of flow f. It can be implemented as a variable, which is zero initially and updated as follows [4] whenever a flow f packet, say p, arrives to the facility:

$$priority(f) := \max\{priority(f), A(p)\} + \frac{l(p)}{r(f)} \tag{1}$$

The new value of $priority(f)$ in (1) is assigned to packet p as its virtual clock value, denoted by $P(p)$. Note that the virtual clock values of flow f are determined by the sequence of packet arrival times of f, and are independent of the design of the service facility. For example, if the service facility is a single server, we have not yet specified its service discipline.

A *VC server* is a priority server that uses the virtual clock value of a packet as its priority. The service discipline is work-conserving and nonpreemptive. Let C denote the capacity, in bits/second, of a VC server. The following delay guarantee is presented in [2] and proved in [3].

Theorem 1. If the capacity of a VC server has not been exceeded for a nonzero duration since the start of a busy period, then the following holds for every packet p that has been served during the busy period:

$$L(p) \leq P(p) + \frac{l_{max}}{C} \tag{2}$$

We next explain the condition of not exceeding the capacity of a VC server. At time t, a flow f is *active* iff the service facility (queue and server) is not empty and the flow's virtual clock is running faster than real time (i.e., $priority(f) > t$). The server capacity is not exceeded at time t iff $C \geq$ sum of the *reserved rates* of flows that are active at time t.

Note that the condition of not exceeding C can always be satisfied by the server because the allocation of reserved rates is under server control.

Source control not required. In proving Theorem 1, there is no assumption that sources are flow-controlled or well-behaved. While the source of a flow, say f, has a reserved rate of $r(f)$, the source can misbehave, i.e., generate traffic at a rate much larger than $r(f)$.[2] The delay guarantee in Theorem 1 holds even when sum of the *actual rates* of active flows is larger than C.

Conditional guarantee. Source control is not required because the delay guarantee is not a bound on $L(p) - A(p)$. A delay guarantee of the form, $L(p) \leq P(p) + \beta$ where β is a constant, is a conditional guarantee. Specifically, if a source is well-behaved, its packets incur a bounded delay. But if a source generates traffic much faster than its reserved rate, its packets may incur large delays.

Firewall property. The delay guarantee to packets in a flow is independent of the behavior of other traffic flows sharing the same server (obvious conclusion from (1)).

[2] It is assumed that each flow is allocated its own buffers, so that if a source misbehaves, it will fill up its own buffers but not those of other flows.

Role of source control. If a source is flow-controlled or known to be well-behaved such that for packet p in the flow, $P(p)-A(p)$ is bounded by a constant, then the delay guarantee becomes a delay bound. For example, $P(p) - A(p)$ is bounded if the source is leaky-bucket controlled.

3 New traffic model

In the balance of this paper, we consider networks with a fixed packet size (such as ATM networks). Consider a flow that requires performance guarantees from a network. Such a flow, which may be video, audio, or data, is called a guaranteed flow. Each guaranteed flow is assumed to satisfy the following Flow Specification when entering the network. For burst i in the flow, let n_i denote its size, in number of packets, and δ_i the maximum duration of its packet arrivals. The average rate of burst i is $\lambda_i = n_i/\delta_i$. (The actual rate is not constrained, e.g., all n_i packets can arrive at the same time.) The jth packet in burst i is denoted by (i,j) and its arrival time at a service facility is denoted by $A(i,j)$.

Flow Specification:

- Each flow is a sequence of bursts, each of which is a sequence of packets. The first packet of burst i carries information on λ_i and n_i.
- Packets in burst i satisfy a *jitter* timing constraint, namely: for $j = 1, 2, \ldots, n_i$,

$$0 \leq A(i,j) - A(i,1) \leq \frac{j-1}{\lambda_i} \tag{3}$$

- Bursts in the flow satisfy a *separation* timing constraint, namely: for $i \geq 1$,

$$A(i+1,1) - A(i,1) \geq \frac{n_i}{\lambda_i} \tag{4}$$

Adaptive rate allocation. In Burst Scheduling networks, the reserved rate allocated to a flow at a channel adapts to the average rate λ_i of burst i when its first packet arrives. The separation timing constraint in (4) ensures that each active flow contains at most one active burst (property used by the server to check that its capacity is not exceeded). (4) also ensures that Theorem 1 holds for a VC server that performs adaptive rate allocation.[3]

Efficiency. In Burst Scheduling networks, the virtual clock of a flow is updated very efficiently. Specifically, the algorithm in (1) is executed only for the first packet of a burst. When any other packet in the burst arrives, the virtual clock is updated by a single increment operation. Furthermore, only one virtual clock value is stored per flow. For a sorted priority queue implemented as a heap, updating a priority value incurs $O(\log N)$ worst-case time where N is the number of active flows (not the number of queued packets).

[3] With adaptive rate allocation, (4) is a sufficient condition which can be relaxed.

4 End-to-end delay bounds

Consider a sequence of nodes, indexed by $0, 1, 2, \ldots, K+1$, where node 0 denotes the source, node $K+1$ the destination, and the other nodes packet switches. The following end-to-end delay bounds provided to a guaranteed flow are based upon several assumptions, all of which can be relaxed [2]. Consider an arbitrary packet (i, j) of the flow. Let $D(i, j)$ denote its end-to-end delay, which is measured from the time the packet leaves node 0 to the time it arrives at node $K + 1$. Define

τ_s propagation time from node s to $s + 1$, in seconds, $s = 0, 1, \ldots, K$

γ_s channel capacity from node s to $s + 1$, in packets/second, $s = 1, 2, \ldots, K$

Theorem 2. The end-to-end delay of the first packet of burst i, for $i = 1, 2, \ldots$, has the following lower and upper bounds:

$$D(i,1) \geq \frac{K-1}{\lambda_i} + \sum_{s=1}^{K} \frac{1}{\gamma_s} + \sum_{s=0}^{K} \tau_s$$

$$D(i,1) \leq \frac{1}{\lambda_i} + (K-1) \max_{1 \leq h \leq i} \{\frac{1}{\lambda_h}\} + \sum_{s=1}^{K} \frac{1}{\gamma_s} + \sum_{s=0}^{K} \tau_s \tag{5}$$

A proof of Theorem 2 can be found in [2]. Because the jitter timing constraint in (3) is preserved by every switch [2], the delay D_i of burst i is bounded as follows

$$D_i \leq D(i,1) + \frac{n_i}{\lambda_i} = D(i,1) + \delta_i \tag{6}$$

Concluding remarks. (i) The concept of delay guarantee to a packet based upon its virtual clock value was proposed in [2, 3] and recently extended to an end-to-end path [1]. (ii) VC servers and PGPS servers provide the same end-to-end delay bound [1]. But computing virtual clock values is substantially more efficient than computing virtual-time finishing times for PGPS. (iii) With the firewall property, the impact of a source-controller malfunction is limited. This is a significant advantage not found in FIFO and static-priority service disciplines.

References

1. Pawan Goyal, Simon S. Lam, and Harrick M. Vin. Determining end-to-end delay bounds in heterogeneous networks. In *Proceedings NOSSDAV*, April 1995.
2. Simon S. Lam and Geoffrey G. Xie. Burst Scheduling: architecture and algorithm for switching packet video. Technical Report TR-94-20, July 1994. An abbreviated version in *Proceedings INFOCOM '95*, April 1995.
3. Geoffrey G. Xie and Simon S. Lam. Delay guarantee of Virtual Clock server. Technical Report TR-94-24, October 1994. Presented at 9th IEEE Workshop on Computer Communications, October 1994.
4. Lixia Zhang. VirtualClock: A new traffic control algorithm for packet switching networks. In *Proceedings of ACM SIGCOMM '90*, pages 19–29, August 1990.