A Scalable Architecture for Fair Leaky-Bucket Shaping

Jennifer Rexford
AT&T Labs Research
Murray Hill, NJ
jrex@research.att.com

Flavio Bonomi
ZeitNet, Cabletron
Santa Clara, CA
flavio@zeitnet.com

Albert Greenberg
AT&T Labs Research
Murray Hill, NJ
albert@research.att.com

Albert Wong
Lucent Technologies
Red Bank, NJ
akwong@lucent.com

Abstract

This paper presents a shaper architecture that scales to a large number of connections with diverse burstiness and bandwidth parameters. The architecture arbitrates fairly between connections with conforming cells by carefully integrating leaky-bucket traffic shaping with rate-based scheduling algorithms. Through a careful combination of per-connection queueing and approximate sorting, the shaper performs a small, bounded number of operations in response to each arrival and departure, independent of the number of connections and cells. To handle a wider range of rate parameters, a hierarchical arbitration scheme can reduce the implementation overheads and the interference between competing connections. Simulation experiments demonstrate that the architecture limits shaping delay and traffic distortions, even under heavy congestion.

1 Introduction

The advent of integrated networks for voice, data, and video applications introduces new challenges in supporting performance guarantees. With high-speed links and small cell sizes, modern ATM (asynchronous transfer mode) switches must process cell arrivals and departures every few microseconds, if not faster; in addition, these architectures should scale to a large number of connections with diverse traffic parameters and quality-of-service requirements. End-to-end guarantees for delay, throughput, and loss depend on the successful provisioning of buffer and bandwidth resources in the network, based on traffic contracts established during admission control. To regulate connections and avoid buffer overflow, broadband networks can employ traffic *shaping* to delay incoming cells until they conform to connection burst and bandwidth descriptors.

Many new networking applications, such as large-scale web or video servers, require traffic shaping for hundreds or even thousands of connections with different burst and bandwidth descriptors. ATM switches can also amortize implementation costs across multiple end systems by providing traffic shaping as a service at the network edge. In addition to shaping at the network end points, ATM available-bit-rate connections require traffic enforcement in the *interior* of the network at each virtual source node, where a connection's bandwidth allocation may change over time in response to feedback from the network [1]. Other connections may also be reshaped in the interior of the network to limit delay variation and buffer requirements at the downstream switches [2–5].

Most existing traffic shapers [6–11] employ some version of *leaky-bucket* control to enforce burstiness and bandwidth restrictions on each connection. Conceptually, a leaky-bucket controller generates tokens at rate ρ, where the token bucket holds at most σ credits [12]; an arriving cell must claim a token before receiving service. Given the status of the token bucket, the shaper can determine the conformance time of each arriving cell [13]. Conflicts arise when multiple cells, from different connections, become eligible for transmission during the same time slot. As a result, the shaper can develop a *backlog* of conforming cells, particularly when traffic arrives from multiple input links or a single high-speed link. Depending on how the switch arbitrates amongst conforming cells, collisions can distort connection leaky-bucket parameters and increase cell shaping delays, even for cells that are conforming on arrival. As a result, the outgoing cells may violate the traffic descriptors expected by downstream switches, possibly increasing delay and loss.

In contrast to first-in first-out scheduling, fair arbitration schemes [14–19] can limit these traffic distortions by guaranteeing that each connection receives its share of the link bandwidth on a small time scale. If the incoming traffic were *already* leaky-bucket compliant, weighted fair scheduling could ensure that multiplexing the connection with other traffic would never inflate σ by more than one cell [15]. However, a traffic shaper must handle a *mixture* of conforming and non-conforming cells, as shown in Figure 1. In fact, a spike in the non-conforming backlog can rapidly become a spike in the conforming backlog, if multiple cells reach

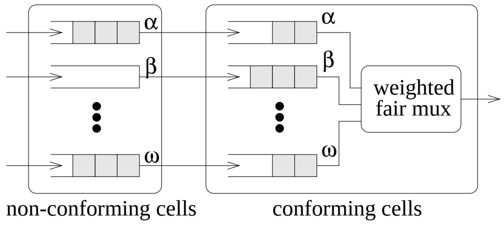


Figure 1: **Idealized Fair Traffic Shaping**

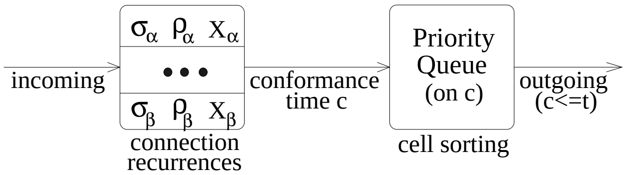


Figure 2: **Leaky-Bucket Traffic Shaper**

conformance in a small interval of time.

Instead of *concatenating* the traffic shaping and fair multiplexing operations, as in Figure 1, this paper presents an efficient architecture that *integrates* rate-based scheduling into the shaping logic. After a review of existing shaper designs in Section 2, Section 3 describes how the proposed architecture combines per-connection queueing, approximate sorting algorithms, and hierarchical arbitration to reduce implementation complexity. Section 4 extends the architecture to minimize traffic distortions through weighted fair link scheduling. The simulation experiments in Section 5 demonstrate that the shaper scales to a large number of connections with diverse traffic descriptors, even under heavy congestion. Section 6 concludes the paper with a discussion of future research directions.

This paper complements existing work on traffic shaper architectures [6–11] by emphasizing implementation and performance scalability. Ongoing research on rate-based link scheduling [14, 15, 17–19] motivates the use of weighted fair arbitration to improve the shaper's performance. Recent work in this area considers schemes that improve fairness by temporarily "stalling" connections that have received link bandwidth ahead of schedule [20–22]. In contrast to our traffic shaper, these link-scheduling schemes define cell eligibility in terms of the underlying fair queueing discipline, without necessarily enforcing burst and bandwidth parameters for each connection. The hierarchical arbitration scheme extends our earlier work on efficient link-scheduling algorithms [23].

2 Review of Shaper Architectures

High-speed shaper designs typically employ per-connection recurrences to assign a conformance time to each arriving cell, followed by a sorting unit that schedules cells for departure, as shown in Figure 2.

2.1 Connection Recurrences

Instead of dropping or marking non-conforming cells, a traffic *shaper* delays incoming cells until they conform to the connection's burst and bandwidth descriptors. Although a traffic shaper could conceivably have a separate leaky-bucket controller and cell buffer for each connection, a more scalable solution maintains a table that stores the leaky-bucket status of each connection [6]. The shaper uses this information to compute the time that an arriving cell first conforms to the connection traffic descriptors. To limit the amount of non-conforming traffic, the shaper can discard cells with conformance times far in the future.

Figure 3 shows a leaky-bucket shaping scheme based on the *virtual scheduling algorithm* [13]. The shaper assigns a conformance time c to each incoming cell, using a state variable X that represents the cell's *estimated* arrival time, based on the connection's shaping rate ρ. A cell arriving at most σ/ρ time units ahead of X can still claim a token; otherwise, the cell must wait until the controller generates an additional token at time $X - \sigma/\rho$. For a connection with no burst tolerance (i.e., $\sigma = 0$), the algorithm reduces to $c = X = \max\{X + 1/\rho_{peak}, t\}$. The shaper can enforce both peak and sustainable cell rates by using a dual leaky-bucket algorithm to compute c, resulting in a $(\sigma, \rho, \rho_{peak})$ specification for each connection.

2.2 Cell Sorting

The shaper uses the cell conformance times c to schedule traffic for transmission. One approach employs a priority queue to rank cells in order of increasing c values; during each time slot, the link transmits the queue's head cell if its conformance time has been reached (i.e., $t \geq c$). High-speed operation typically requires hardware support for cell scheduling [7]. To avoid the complexity of an exact priority queue implementation, a shaper could dynamically construct a link transmission schedule as each cell arrives. In this approach, the schedule consists of a large collection of bins, where each bin corresponds to a single transmission slot [8], as shown in Figure 4(a).

The shaper sequences through the bins at the link rate, transmitting a cell in each time slot unless the current bin is empty. An arriving cell enters the first empty location at or after c, modulo the number of bins; this requires complex logic for locating empty slots in the schedule. The shaper drops the incoming cell if $c - t$ exceeds the range of the bins. When

```
X = X + 1/ρ;                // Estimated time
if (X ≤ t)                  // Full bucket
   c = X = t;
else if (X ≤ t + σ/ρ)       // Partially full
   c = t;
else                        // Empty bucket
   c = X − σ/ρ;
```

Figure 3: **Computing Cell Conformance Times**

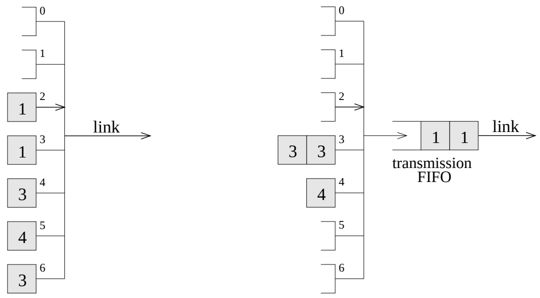
the scheduler services a large number of connections, under heavy load, the shaper may have to schedule a cell for transmission much later than its conformance time c. The insertion mechanism favors earlier arrivals, possibly delaying a well-behaved connection to serve other cells with larger conformance times.

To transmit cells in order of increasing conformance times, the shaper can employ a *calendar queue* [24], as shown in Figure 4(b). Instead of storing a single cell in each bin, this architecture maintains a logical *linked list* of cells at each conformance time [9]. Although this requires additional memory to store a pointer field with each cell, an arriving cell can join the linked list that corresponds to its c value without searching for an empty slot in the schedule. Since the link cannot transmit more than one cell in each time slot, the shaper maintains a separate FIFO for holding any cells that have reached their conformance times; in each time slot, the shaper appends this transmission FIFO with the contents of the current sorting bin. Hence, under a backlog of conforming traffic, the shaper transmits cells in order of increasing c values.

Although the sorting bins obviate the need for a complex priority queue, cells may have c values ranging far into the future, particularly when a low-bandwidth connection has a backlog of non-conforming traffic. To reduce the size of the shaper, each bin can represent a range of g consecutive conformance times. However, shaping with a large bin granularity can introduce excessive jitter that distorts the leaky-bucket parameters of the outgoing traffic, particularly for high-bandwidth connections. In addition, existing shaper architectures do not control the interference between competing connections when multiple conforming cells await service.

3 Scalable Traffic Shaper

This section introduces a new shaper architecture that limits implementation complexity by combining per-connection queueing, approximate sorting, and hierarchical arbitration.



(a) One cell per bin (b) Linked list per bin

Figure 4: **Ranking Cells in Sorting Bins**

3.1 Approximate Sorting

Although a connection α may have a large backlog of cells, with a wide range of conformance times, the connection's *head-of-line* cell has a c value at most $1/\rho_\alpha$ time slots into the future; otherwise, the shaper would not have already transmitted the connection's previous cell. To reduce the number of sorting bins, a shaper can consider just the head cell from each backlogged connection, while holding the connection's remaining cells in a FIFO list. In addition to limiting shaper complexity, per-connection queueing enhances architectural flexibility by facilitating effective buffer-management and link-scheduling policies, as well as rapid responses to changes in connection rate parameters. Although per-connection queueing can complicate the implementation of exact sorting, approximate schemes can significantly reduce complexity.

To illustrate the difficulty of implementing exact sorting, consider the situation when a cell enters the sorting unit upon the departure of the connection's previous cell. If the shaper has accumulated a backlog of conforming traffic, this new head cell may have *already* reached its conformance time (i.e., $c < t$). To transmit cells in order of their c values, the shaper would have to insert this cell *in between* other conforming cells awaiting service. To handle these cells, the sorting unit could include extra data structures to hold conforming traffic, in addition to the bins for the non-conforming head-of-line cells [11]. However, this requires a large number of extra sorting bins, as well as more complicated sequencing logic.

Approximate sorting algorithms may be necessary to avoid these overheads in high-speed implementations. Instead of exact sorting, a shaper can reduce complexity by placing all conforming head-of-line cells in a single transmission FIFO, as shown in Figure 5. Since the sorting bins hold only *non-conforming* head-of-line cells, the shaper requires at most $b = 1/(g\rho_{\min})$

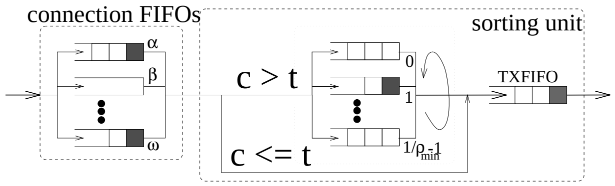


Figure 5: **Scalable Shaper Architecture**

bins, independent of the number of connections or the amount of backlogged traffic, where g is the bin granularity. Every g time slots, the shaper concatenates a single sorting bin with the transmission FIFO, as in Figure 4(b). After transmitting a cell, the connection's next cell proceeds to a sorting bin (if $c > t$) or directly to the transmission FIFO (if $c \leq t$), as shown in Figure 6. If the new head-of-line cell has already reached its conformance time, this policy sacrifices accuracy in order to reduce implementation complexity.

To quantify these overheads, consider a shaper with s cells, v connections, and b sorting bins. The shaper consists of $v + b + 1$ logical linked lists that represent the connection FIFOs, the sorting bins, and the transmission FIFO, where each list consists of a head pointer, a tail pointer, and an empty flag; each connection includes an additional flag to indicate whether or not the connection has a head-of-line cell in the sorting unit. To connect cells in the various linked lists, each buffered cell includes a small pointer field that consists of $\lceil \log_2 s \rceil$ bits to denote a location in the cell memory. As shown in Figure 6, each cell arrival or departure introduces a small, bounded number of pointer manipulations on these linked lists, independent of s and v. The shaper also includes a free list for assigning unused memory locations to arriving cells; initially, this list includes every element in the memory. Upon cell departure, the shaper returns the free memory location to the head of the free list.

3.2 Bandwidth Groups

Although the algorithm in Figure 6 has $\Theta(1)$ enqueue and dequeue complexity, handling a wide range of rate parameters would require a large number of sorting bins. For a low-rate connection, even the head-of-line cell could have a conformance time far into the future. Although low-bandwidth traffic requires the sorting unit to handle a wide *range* of c values, these connections can tolerate some inaccuracy in cell scheduling, permitting coarse-grain sorting bins. On the other hand, a large bin granularity g can introduce significant shaping delay and jitter to high-rate connections. To reconcile these conflicting requirements, the shaper can group connections based on their bandwidth requirements, allowing each sorting unit to select a different grain (g) and range (bg) for its bins. This results in a two-level architecture, where each group consists of a sorting unit that handles a large number of connections with similar rates.

With this hierarchical approach, the shaper can select a small sorting granularity for high-rate connections to reduce delay and jitter, relative to the existing architectures in Section 2. To formalize these trade-offs, consider a shaper with bandwidth parameters that can vary from $\rho_{\min}$ to $\rho_{\max}$, where each group handles rates within a factor $m > 1$ of each other. As a result, group i includes any connections α_i with $\rho_{\alpha_i} \in [m^i \rho_{\min}, m^{i+1}\rho_{\min})$, where $i = 0, 1, \ldots, n-1$ and the shaper consists of $n = \log_m\{\rho_{\max}/\rho_{\min}\}$ groups; for example, if $m = 16$, the shaper can support rates ranging from 1 kilobit/second to 1 gigabit/second with just *five* groups (since $\log_{16}\{2^{30}/2^{10}\} = 5$). In each group, the *low*-bandwidth connections dictate the range of the sorting bins; for group i, a head-of-line cell can have a c value at most $1/(m^i \rho_{\min})$ time units into the future. To prevent delay and traffic distortions for the *high*-bandwidth connections, each group should select a bin granularity g_i based on the requirements of its largest possible rate; i.e., $g_i = 1/u(m^{i+1}\rho_{\min})$, where a larger value of u corresponds to more precise sorting.

As a result, each group i requires at least

$$b = \frac{\text{range of the bins}}{\text{grain of each bin}} = \frac{1/(m^i \rho_{\min})}{1/u(m^{i+1}\rho_{\min})} = mu$$

sorting bins, for a total of mnu bins in the shaper. The number of bins increases under more precise scheduling (larger u) or a wider range of connection rates (larger n or m). However, if the shaper did *not* divide connections into groups, a single sorting unit would have to handle the maximum range of $1/\rho_{\min}$ and the minimum granularity of $1/(u\rho_{\max})$, for a total of $m^n u$ bins. With rates ranging from 1 kilobit/second to 1 gigabit/second, this shaper would require over $13,000$ times more sorting bins than a hierarchical architecture with $m = 16$, for the same value of u. Alternatively, for the same number of sorting bins, the hierarchical architecture can support much more precise scheduling (with u over $13,000$ times larger).

4 Integrating Fair Link Scheduling

The hierarchical architecture in Section 3 scales well with the number of connections, the number of cells, and the range of connection rate parameters. However, good performance depends on how well the switch arbitrates between connections when multiple conforming cells await service. By carefully multiplexing cells from different connections, the shaper can en-

Cell Arrival
`compute` c `for the new cell;`
`if (connection is currently idle) {`
`// New head-of-line cell`
`if (`c `==` t`) // Conforming cell`
`enqueue cell onto transmission FIFO;`
`else // Non-conforming cell`
`enqueue cell onto bin` $\lfloor(c \bmod bg)/g\rfloor$`;`
`}`
`else // backlogged connection`
`enqueue cell onto connection FIFO;`

Cell Departure
`if (connection is still backlogged) {`
`dequeue cell from connection FIFO;`
`// New head-of-line cell`
`if (`$c \leq t$`) // Conforming cell`
`enqueue cell onto transmission FIFO;`
`else // Non-conforming cell`
`enqueue cell onto bin` $\lfloor(c \bmod bg)/g\rfloor$`;`
`}`
`else`
`// No action for idle connection`

Conforming Bin
`if (`t `mod` g `==` $g-1$`)`
`append bin` $\lfloor(t \bmod bg)/g\rfloor$ `onto transmission FIFO;`

Figure 6: **Shaping Algorithm with Per-Connection Queueing**

sure that each connection receives a fair share of the link bandwidth on a small time scale.

4.1 Fair Arbitration Within a Group

Ideally, a traffic shaper schedules an arriving cell for transmission at its conformance time c, forwarding conforming cells directly to the outgoing link. However, since the link carries traffic for multiple connections, several cells may become eligible for transmission at the same time; as a result, the shaper can develop a backlog of conforming traffic, especially during periods of heavy congestion; this is particularly likely in switches with a large number of input ports and connections. As discussed in Section 2, most existing traffic shapers transmit conforming cells in order of increasing conformance times. Under a backlog of conforming cells, this "exact sorting" favors connections with larger σ values since these connections can have multiple cells in a small range of c values. These bursts restrict link access for connections with smaller σ values; by the time this transient congestion begins to dissipate, a low-σ connection can have a large backlog of conforming traffic, which may generate an unexpected burst on the output link.

The shaper can mitigate these collision effects by *interleaving* the conforming cells from competing connections based on their rate parameters. The approximate architecture in Figure 5 suggests an efficient mechanism for fair link scheduling when the shaper consists of a single group. In this architecture, when connections have a backlog of conforming traffic, each cell transmission triggers the insertion of the connection's next cell into the transmission FIFO, behind the head-of-line cells from each of the other conforming connections. As a result, the shaper implicitly sequences through these connections in a *round-robin* fashion, guaranteeing a minimum bandwidth to each connection on a relatively small time scale.

However, round-robin arbitration does not provide truly "fair" service when connections have different ρ values, since low-rate and high-rate connections receive the same fraction of the bandwidth. Instead, the shaper should employ *weighted* round-robin scheduling. With a small extension, the architecture in Figure 5 can allow connections to insert *multiple* cells into the sorting unit, in proportion to their rates. In particular, the shaper should allow connection α to have up to $\lfloor\rho_\alpha/\rho_{\min}\rfloor$ consecutive cells in the sorting unit. This allows a high-bandwidth connection to have several conforming cells in the transmission FIFO, ahead of conforming traffic from low-bandwidth connections, even if the low-bandwidth connections have cells with *smaller* c values. When the shaper has a backlog of conforming traffic, this ensures that connections are served in proportion to their rates.

Even though some connections have multiple cells in the sorting unit, the weighted fair arbitration does not require additional sorting bins, since these cells have conformance times at most $1/\rho_\alpha \cdot \lfloor\rho_\alpha/\rho_{\min}\rfloor$ time units into the future, which is within the range ($1/\rho_{\min}$) of sorting bins. Relative to the architecture in Section 3, the weighted fair shaper includes a count of the number of cells in the sorting unit for each connection. The shaper increments (decrements) this *counting semaphore* as the connection's cells enter (leave) the sorting unit; when the count reaches $\lfloor\rho_\alpha/\rho_{\min}\rfloor$, the connection cannot insert another cell into the sorting unit until a cell departs. This results in a very efficient architecture, with $\Theta(1)$ enqueue and dequeue complexity, that gives fair service to conforming connections on a small time scale.

4.2 Fair Arbitration Between Groups

The weighted round-robin arbitration can provide fair service to connections in a single group, but the hierarchical shaper architecture requires an effective mechanism for interleaving conforming cells from *different* groups. For an efficient implementation, the shaper can also apply weighted fair scheduling algorithms to divide link bandwidth between groups. To coordinate bandwidth sharing, the weights ϕ_i represent the *aggregate* rate requirements of each group i; in the simplest case, $\phi_i = \sum \rho_{\alpha_i}$. A variety of rate-based link-scheduling algorithms [14–17, 25–27] can guarantee that group i receives a fair portion $\phi_i / \sum_{j=0}^{n-1} \phi_j$ of the link bandwidth on a small time scale.

With these static weights ϕ_i, an idle connection divides its bandwidth amongst the busy connections in the same group, instead of sharing with other connections in different groups. This type of hierarchical link sharing is particularly useful when groups correspond to different institutions, traffic classes, or protocol families [20, 28, 29], and can ensure that the link guarantees a minimum bandwidth to each connection. As a heuristic extension [23], the shaper could share excess bandwidth between individual connections by dynamically adjusting the ϕ_i values to reflect the aggregate throughput requirements of the *backlogged* connections in each group. That is, a group could assign

$$\phi_i(t) = \sum_{\alpha_i \in B_i(t)} \rho_{\alpha_i}$$

where $B_i(t)$ is the set of backlogged connections in group i at time t.

Changes to $B_i(t)$ occur upon cell arrivals and departures, facilitating efficient updates to ϕ_i by adding (subtracting) ρ_{α_i} when connection α_i becomes backlogged (idle). To demonstrate the use of dynamic weights, Section 5.4 evaluates an extension to the *self-clocked fair queueing* [25–27] algorithm that adjusts to changes in the ϕ_i values. Although the dynamic arbitration scheme can sometimes deviate from connection-level fairness [23], the shaping function in each group ensures that each connection makes forward progress based on the leaky-bucket conformance times of its incoming cells; a group continues to compete for link access until it clears its temporary backlog of conforming cells. Ultimately, static weights provide a minimum rate to each connection on a small time scale, while dynamic weights provide a fairer division of excess link bandwidth. Both approaches provide an efficient, event-driven way to multiplex a large number of connections with diverse bandwidth requirements.

5 Performance Evaluation

This section focuses on how the various shaper architectures affect the burstiness of the outgoing cell streams. Inherently, the architectures have similar performance under low traffic loads, since the shaper does not develop a significant backlog of conforming traffic. Hence, most of the experiments consider the effects of congestion, particularly when the shaper mixes connections with different traffic parameters.

5.1 Simulation Environment

The simulation experiments evaluate connections sharing access to a single link that can transmit one cell in each time slot; for stability, the link bandwidth must exceed the sum of the connections' sustainable cell rates. Each connection generates periodic bursts of cells according to an on/off model with leaky-bucket parameters (σ_{in}, ρ_{in}), with the peak rate equal to the link bandwidth; the burst length σ_{in} could correspond to the packet or message size in a data transmission. The simulation experiments generate temporary backlogs of non-conforming cells by allowing σ_{in} to exceed the burstiness parameter σ enforced by the shaper.

To control the interaction between cell streams, each connection has an independent starting time, uniformly distributed in an interval $[0, x]$; the length of this interval can vary from 0 to the connection's on/off period. Smaller values of x generate periods of heavier congestion and can capture the effects of multiple input links (or one high-speed input link) entering the switch. With this parameterized model of incoming traffic, the simulation experiments can evaluate the performance scalability of the shaper architectures under different arrival patterns (σ_{in}, ρ_{in}), shaping parameters (σ, ρ), and degrees of congestion x.

When the shaper accumulates a backlog of conforming cells, link arbitration policies can affect the leaky-bucket parameters of the outgoing streams. Multiplexing a connection with other network traffic results in an outgoing stream that is (σ_{out}, ρ)-compliant, where σ_{out} may exceed the shaping parameter σ. To evaluate the burstiness of the output stream, the simulator feeds a connection's outgoing cells to a hypothetical link of rate ρ; σ_{out} $(\overline{\sigma}_{out})$ represents the worst-case (average) queue length encountered by the connection's cells. Ideally, the switch produces a well-shaped, (σ, ρ)-compliant stream, resulting in $\sigma_{out} \leq \sigma$. The experiments compare the proposed shaper architecture against a traditional shaper that transmits conforming traffic in order of increasing conformance times.

5.2 Different Burst Parameters

The architecture in Figure 5 reduces implementation complexity through per-connection queueing and

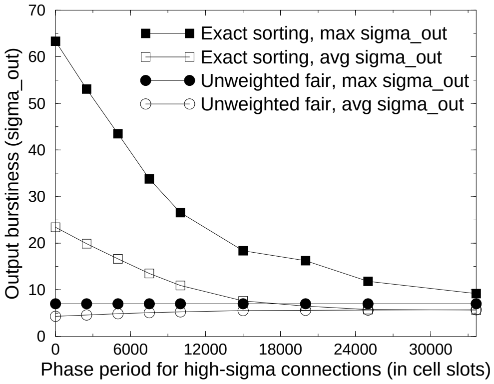


Figure 7: **Mixing Different Burst Parameters**

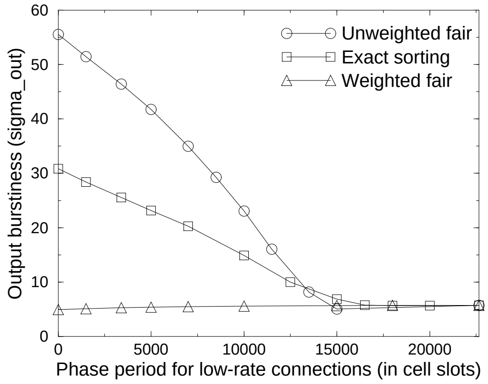


Figure 8: **Mixing Different Rate Parameters**

approximate sorting of backlogged conforming cells. Figure 7 compares the architecture to a shaper that always transmits cells in order of their conformance times. In this experiment, the input traffic consists of 80 connections with high burstiness ($\sigma_{in} = 300$ and $\sigma = 100$) and 10 connections with lower burstiness ($\sigma_{in} = 50$ and $\sigma = 5$); all connections have $\rho_{in} = 0.009$ and $\rho = 0.010$ for a total link utilization of 81%. To generate periods of congestion in the shaper, the experiment varies the phase of the 80 high-σ connections from 0 to their on/off period, while the low-σ connections have uniform random start times during their on/off period. With an on period of length $\sigma_{in}/(1 - \rho_{in})$ and an off period of length σ_{in}/ρ_{in}, the high-σ connections have an on/off period of 33636 time slots, as shown by the x-axis in Figure 7.

Under exact sorting, this congestion has an adverse effect on the leaky-bucket parameters of the low-σ connections; the high-σ connections do not experience significant traffic distortion under either shaper architecture. Exact sorting favors connections with larger σ values, which can inject a burst of cells with a small range of conformance times. In times of congestion, these bursts can restrict link access for connections with smaller σ values; by the time this transient congestion begins to dissipate, a low-σ connection can have a large backlog of conforming traffic, which can then generate an unexpected burst on the output link. This can introduce significant traffic distortions in the outgoing cell streams, particularly when multiple high-σ connections are active simultaneously. In contrast, the fair arbitration scheme effectively interleaves backlogged connections in a round-robin fashion, resulting in virtually no inflation in σ_{out}, even under heavy load. Since this experiment assigns the same ρ value to all connections, the weighted and unweighted fair architectures have identical performance.

5.3 Different Bandwidth Parameters

Although the unweighted fair architecture guarantees a minimum bandwidth to each backlogged connection, on a small time scale, round-robin arbitration does not provide fair service when connections have different bandwidth parameters. To demonstrate this effect, Figure 8 shows the performance of 10 high-rate connections ($\rho_{in} = 0.0090$ and $\rho = 0.0100$) in the presence of 320 low-rate connections ($\rho_{in} = 0.0022$ and $\rho = 0.0025$), for a total link utilization of 79%. The high-rate traffic has $\sigma = 5$ and the low-rate traffic has $\sigma = 25$, with $\sigma_{in} = 50$ for all connections. The experiment varies the phase of the 320 low-rate connections from 0 to their on/off period (22644), while the high-rate connections have uniform random start times during their on/off period.

Although all three shaper architectures perform well under low levels of congestion, unweighted fair arbitration significantly inflates the burstiness of the high-rate connections during periods of heavy load, as shown in Figure 8; the low-rate connections have $\sigma_{out} \approx \sigma$ throughout the experiment. Exact sorting also introduces traffic distortions, since the shaper services connections with different burstiness parameters. The high-rate connections also experience high cell shaping delays (not shown) under these two architectures. In contrast, weighted fair arbitration successfully preserves the leaky-bucket descriptors and limits cell shaping delay, even under extremely heavy congestion, by guaranteeing sufficient bandwidth to each connection on a small time scale.

5.4 Hierarchical Arbitration

Although the weighted fair architecture performs well under diverse leaky-bucket parameters, handling a wide range of connection rates requires a large number of sorting bins. To quantify this cost-performance trade-off, Figure 9 compares four different configurations of the proposed shaper, serving a mixture of connections that employ weighted fair arbitration, with a total link utilization of nearly 90%. The experiment mixes five high-rate, low-burstiness connections ($\sigma_{in} = 10$, $\rho_{in} = 0.0909$, $\sigma = 0$, $\rho = 0.1000$) with ten medium-rate connections ($\sigma_{in} = 50$, $\rho_{in} = 0.0094$, $\sigma = 25$, $\rho = 0.0100$) and seventy low-rate connections ($\sigma_{in} = 40$, $\rho_{in} = 0.0048$, $\sigma = 20$, $\rho = 0.0050$). The experiment varies the phase period of the 70 low-rate connections from 0 to their on/off period (8528) to study the effects of congestion in the shaper; the other 15 connections have uniform random start times.

The low-rate connections dictate the range of the sorting bins, since a head-of-line cell can have a c value up to 200 time slots into the future. With coarse-grain sorting, the shaper can limit the number of bins:

Key	Configuration
○	Single group with 201 bins of grain 1
□	Single group with 10 bins of grain 21
◇	High-rate group with 11 bins of grain 1; low-rate group with 201 bins of grain 1
×	High-rate group with 5 bins of grain 3; low-rate group with 5 bins of grain 41

at the expense of the burstiness parameters for the high-bandwidth traffic, as shown by comparing the top two curves in Figure 9. To highlight the differences in the four configurations, the graph omits the "exact sorting" configuration, which performs dramatically worse, particularly for small values of x. For example, when $x = 4000$, exact sorting has an output burstiness of $\overline{\sigma}_{out} = 60$, in contrast to values of less than 8 for each of the configurations in Figure 9. The low-rate connections do not experience significant traffic distortions under any of the shaper architectures. By dividing connections into two groups, the hierarchical architecture can reduce the number of sorting bins without distorting the high-bandwidth traffic, even under heavy congestion. In this configuration, one group services the five high-rate connections, while the second group handles the remaining traffic. Even with just ten sorting bins, the coarse-grain hierarchical architecture has small $\overline{\sigma}_{out}$ values, since each group tailors its bin granularity to the connection rate parameters.

Interestingly, the two hierarchical architectures even outperform the more expensive, fine-grain sorting scheme. This occurs because, even with weighted fair arbitration, a group can have multiple cells, from different connections, within a small range of sorting bins. Hence, when all connections share a single group, a high-rate connection may wait behind a long FIFO of low-bandwidth traffic before receiving service. In contrast, hierarchical arbitration limits the number of low-rate connections that can receive service between visits to the high-rate group, providing a minimum bandwidth to the high-rate connections on a smaller time scale.

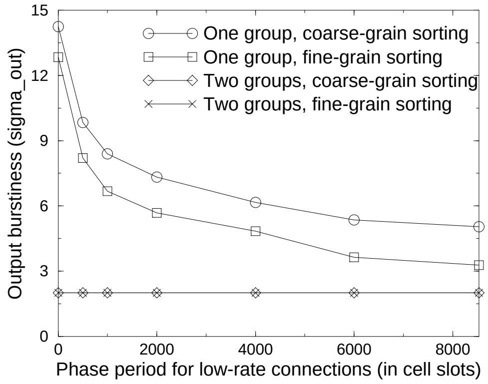


Figure 9: **Hierarchical Fair Arbitration**

6 Conclusion

Modern high-speed networks require efficient traffic shapers that can service a large number of connections with a wide range of bandwidth and burstiness parameters. To reduce complexity, traffic shaper designs can incorporate per-connection queueing, approximate sorting algorithms, and hierarchical arbitration policies. With a careful selection of sorting and arbitration schemes, these designs can also limit the traffic distortions that arise when multiple conforming cells await service. This is particularly important for shaping at the egress of a network, where the incoming traffic rate may temporarily exceed the outgoing link capacity, and in switches with multiple input links.

Ultimately, traffic shaping and link scheduling require a careful balance between implementation complexity and accuracy in approximating an idealized scheme. In contrast to the model in Figure 1, the proposed architecture avoids both the replication of complex sorting logic and the expensive movement of cells between two separate priority queues. As future work, we are further analyzing the properties of the integrated architecture, through additional simulation experiments with more realistic traffic patterns. Finally, we are considering extensions to the hierarchical traffic-shaping and link-scheduling framework, including arbitration schemes that differentiate between ATM traffic classes, as well as support for connections that do not require reshaping at every switch.

References

[1] F. Bonomi and K. W. Fendick, "The rate-based flow control framework for the available bit rate ATM service," *IEEE Network Magazine*, pp. 25–39, March/April 1995.

[2] H. Zhang and D. Ferrari, "Rate-controlled service disciplines," *Journal of High Speed Networks*, vol. 3, no. 4, pp. 389–412, 1994.

[3] H. Zhang, "Providing end-to-end performance guarantees using non-work-conserving disciplines," *Computer Communications*, vol. 18, pp. 769–781, October 1995.

[4] L. Georgiadis, R. Guerin, V. Peris, and K. N. Sivarajan, "Efficient network QoS provisioning based on per node traffic shaping," *IEEE/ACM Trans. Networking*, vol. 4, pp. 482–501, August 1996.

[5] J. Rexford, J. Hall, and K. G. Shin, "A router architecture for real-time point-to-point networks," in *Proc. Inter. Symposium on Computer Architecture*, pp. 237–246, May 1996.

[6] H. J. Chao, "Design of leaky bucket access control schemes in ATM networks," in *Proc. Inter. Conference on Communications*, pp. 180–187, June 1991.

[7] H. J. Chao and N. Uzun, "A VLSI sequencer chip for ATM traffic shaper and queue manager," *IEEE J. of Solid-State Circuits*, vol. 27, pp. 1634–1643, November 1992.

[8] P. E. Boyer, F. M. Guillemin, M. J. Servel, and J.-P. Coudreuse, "Spacing cells protects and enhances utilization of ATM network links," *IEEE Network Magazine*, pp. 38–49, September 1992.

[9] E. Wallmeier and T. Worster, "The Spacing Policer, an algorithm for efficient peak bit rate control in ATM networks," in *Proc. Inter. Switching Symposium*, pp. 22–26, October 1992.

[10] T. Moors, N. Clarke, and G. Mercankosk, "Implementing traffic shaping," in *Proc. Conf. on Local Computer Networks*, pp. 307–314, October 1994.

[11] G. Mercankosk, T. Moors, and A. Cantoni, "Multiplexing spacer outputs on cell emissions," in *Proc. IEEE INFOCOM*, pp. 49–55, April 1995.

[12] J. Turner, "New directions in communications, or which way to the information age?," *IEEE Communication Magazine*, vol. 24, pp. 8–15, October 1986.

[13] ATM Forum, "Traffic management specification, version 4.0." ATM Forum/95-0013R10, February 1996.

[14] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," *J. Internetworking: Research and Experience*, pp. 3–26, September 1990.

[15] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: The single node case," *IEEE/ACM Trans. Networking*, vol. 1, pp. 344–357, June 1993.

[16] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: The multiple node case," *IEEE/ACM Trans. Networking*, vol. 2, pp. 137–150, April 1994.

[17] H. Zhang and S. Keshav, "Comparison of rate-based service disciplines," in *Proc. ACM SIGCOMM*, pp. 113–121, September 1991.

[18] H. Zhang, "Service disciplines for guaranteed performance service in packet-switching networks," *Proc. of the IEEE*, vol. 83, pp. 1374–1396, October 1995.

[19] C. M. Aras, J. F. Kurose, D. S. Reeves, and H. Schulzrinne, "Real-time communication in packet-switched networks," *Proc. of the IEEE*, vol. 82, pp. 122–139, January 1994.

[20] J. C. R. Bennett and H. Zhang, "Hierarchical packet fair queueing algorithms," in *Proc. ACM SIGCOMM*, pp. 143–156, August 1996.

[21] D. Stiliadis and A. Varma, "A general methodology for designing efficient traffic scheduling and shaping algorithms." To appear in *Proc. IEEE INFOCOM*, April 1997.

[22] S. Suri, G. Varghese, and G. Chandranmenon, "Leap forward virtual clock: A new fair queuing scheme with guaranteed delays and throughput fairness." To appear in *Proc. IEEE INFOCOM*, April 1997.

[23] J. Rexford, A. Greenberg, and F. Bonomi, "Hardware-efficient fair queueing architectures for high-speed networks," in *Proc. IEEE INFOCOM*, pp. 638–646, March 1996.

[24] R. Brown, "Calendar queues: A fast O(1) priority queue implementation for the simulation event set problem," *Communications of the ACM*, vol. 31, pp. 1220–1227, October 1988.

[25] J. R. Davin and A. T. Heybey, "A simulation study of fair queueing and policy enforcement," *Computer Communication Review*, pp. 23–29, October 1990.

[26] J. W. Roberts, "Virtual spacing for flexible traffic control," *Inter. J. on Communication Systems*, vol. 7, pp. 307–318, October–December 1994.

[27] S. J. Golestani, "A self-clocked fair queueing scheme for broadband applications," in *Proc. IEEE INFOCOM*, pp. 636–646, 1994.

[28] S. Shenker, D. D. Clark, and L. Zhang, "A scheduling service model and a scheduling architecture for an integrated services packet network." Working paper, Xerox PARC, August 1993.

[29] S. Floyd and V. Jacobson, "Link-sharing and resource management models for packet networks," *IEEE/ACM Trans. Networking*, vol. 3, pp. 365–386, August 1995.