Slide 0

Hardware-Efficient Fair Queueing Architectures for High-Speed Networks

Jennifer Rexford
University of Michigan
jrexford@eecs.umich.edu

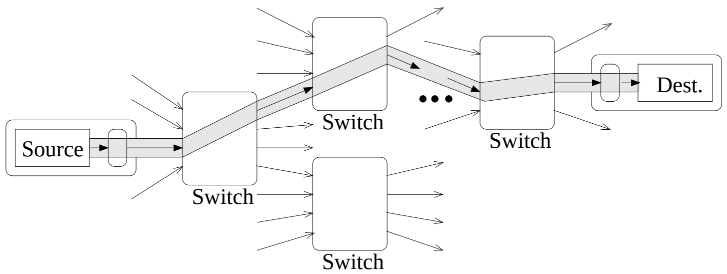
Albert Greenberg
AT&T Laboratories
albert@research.att.com

Flavio Bonomi
Zeitnet Incorporated
flavio@zeitnet.com

Real-Time Computing Laboratory
The University of Michigan

Slide 1

Mechanisms for End-to-End Quality of Service



- Route establishment
- Admission control
- Traffic policing/shaping
- Buffer management
- Link scheduling

Real-Time Computing Laboratory
The University of Michigan

Slide 2

Link Sharing Goals

- Minimum bandwidth r for each connection
- Guaranteed service for each connection on a small time scale
- High link utilization and connection admissability
- Preservation of connection traffic parameters
- "Fair" division of excess link bandwidth

→ rate-based link scheduling (e.g., fair queueing schemes)

Slide 3

Emerging High-Speed Networks

Network traffic multiplexed on a single link

- Thousands of connections/sessions
- Diverse burst and bandwidth parameters (video, audio, data)
- Diverse performance requirements (delay, loss, throughput)

→ powerful algorithms for managing link bandwidth

Network technology trends

- High-speed links (e.g., 155 megabits/second; 1 gigabit/second)
- Small data transfer unit (e.g., 53-byte ATM cell)

→ scalable and efficient hardware designs

Slide 4

Scalability Requirements

Scalable implementation:

- Small, bounded amount of work per cell arrival/departure
- Small, simple data structures

Scalable performance:

- Limited interference between competing connections
- Limited distortion of connection traffic parameters

Scheduling algorithms:

- First-in first out: implementation, but not performance
- Weighted fair queueing: performance, but not implementation
- Self-clocked fair queueing: $\sim$ performance and implementation (trouble with wide range of rates)

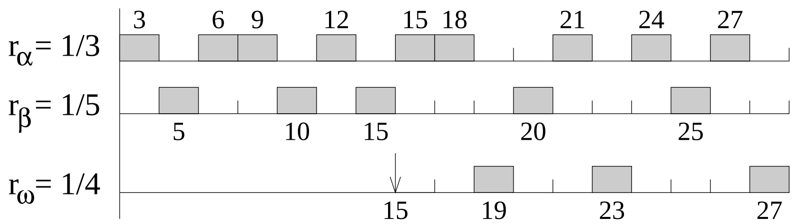
Real-Time Computing Laboratory
The University of Michigan

Slide 5

Self-Clocked Fair Queueing

[Golestani, Davin/Heybey, Roberts]

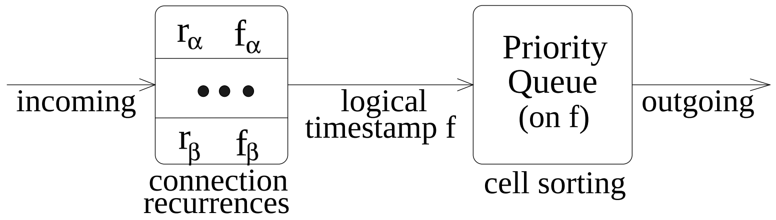
- Service busy connections α, β, ... in proportion to r_α, r_β, ...
- Compute logical timestamp f for each arriving cell
  $\rightarrow f_\alpha = \max\{f_\alpha, f_{serv}\} + 1/r_\alpha$
- Transmit cell with the smallest f value (new f_{serv})



Real-Time Computing Laboratory
The University of Michigan

Slide 6

SCFQ Architecture: Sorting by f Values



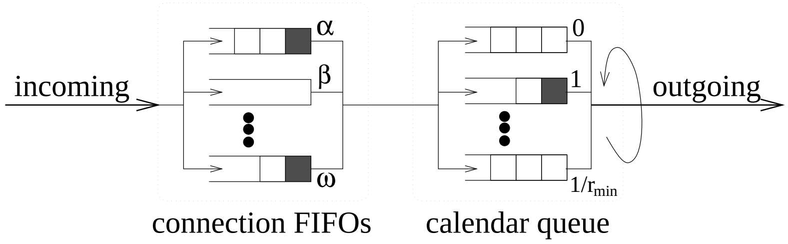
Simplifying the priority queue:

- Sort on just *head-of-line* cell from each busy connection
- Compute f_α only for *head-of-line* cells (no per-cell keys)
- Bound *range* of keys: $f_\alpha \in [f_{serv}, f_{serv} + 1/r_{\min}]$
- Bound *number* of keys: $f_\alpha \in \{f_{serv}, f_{serv} + 1, \ldots f_{serv} + 1/r_{\min}\}$

Real-Time Computing Laboratory
The University of Michigan

Slide 7

SCFQ Architecture



- Each sorting FIFO corresponds to a value of f (mod $1{+}1/r_{\min}$)
- Bin currently in service represents f_{serv}
- Link cycles through the non-empty FIFO bins (find min$\{f\}$)
- New head-of-line α cell joins bin $1/r_\alpha$ away from f_{serv}

$\rightarrow$ small, bounded work per cell arrival/departure

Real-Time Computing Laboratory
The University of Michigan

Slide 8

Implementation Complexity

Exact implementation of SCFQ requires

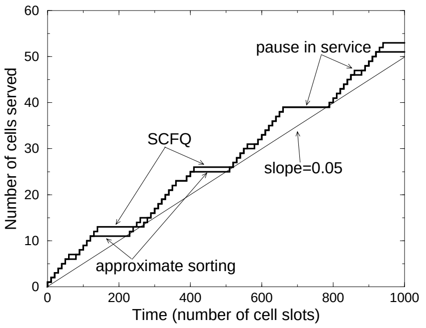
- Connection FIFOs and $1 + 1/r_{\min}$ sorting FIFOs
- Shared cell memory and idle-address FIFO (free list)
- Head and tail pointers for each FIFO
- Logic to cycle through non-empty sorting FIFOs

Approximate sorting to support finer *grain* of connection rates

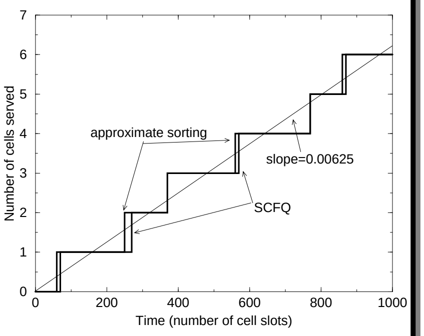
- Each sorting FIFO corresponds to a small *range* of f values
- Connection $1/r_\alpha$ value includes *integer* and *fractional* portions
- New cell inserts either $\lfloor 1/r_\alpha \rfloor$ *or* $\lceil 1/r_\alpha \rceil$ away from f_{serv}
- Good performance for *moderate* range of connection weights

Slide 9

Mixing $r = 1/3$ (100 VCs), $r = 1/24$ (20 VCs), and $r = 1/40$ (100 VCs)



High-bandwidth ($r = 1/3$)



Mid-bandwidth ($r = 1/24$)

(approximate SCFQ has sorting bins with grain = 7)

Slide 10

Wider Range of Connection Weights r_α

Accurate sorting for high-bandwidth connections requires:

- Small granularity for sorting bins (grain $\propto 1/r_{\max}$)
- Large number of sorting bins (range $\propto 1/r_{\min}$)
- Logic to locate non-empty sorting bins ($\propto \log(r_{\max}/r_{\min})$)

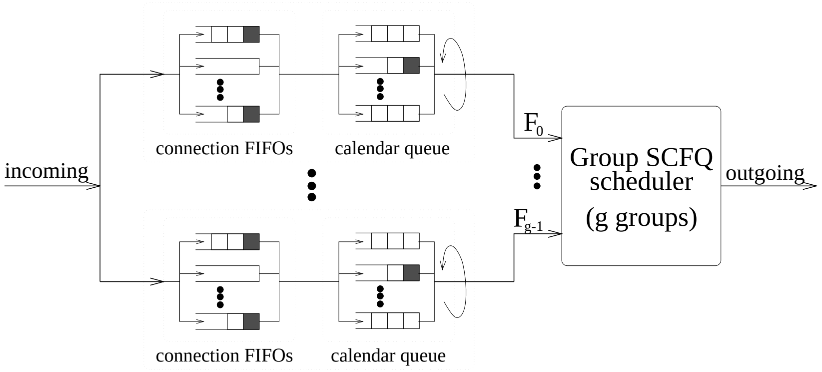
Performance degradation for high-bandwidth connections:

- Multiple low-rate connections can have similar f values
  $\rightarrow$ deny service to high-rate connections
- Degradation of average and worst-case performance
  $\rightarrow$ cell delay and traffic distortions (burstiness)
- Analytical model (Golestani) and simulation results
  $\rightarrow$ distortion and delay grow $\propto n(r_\alpha/r_{\min})$

Real-Time Computing Laboratory
The University of Michigan

Slide 11

Hierarchical SCFQ Architecture



- Each group: large number of connections with similar rates
- Second stage: small number of groups with diverse rates

Real-Time Computing Laboratory
The University of Michigan

Slide 12

Dynamic Group Weights

For connection-level fairness, group i's weight ϕ_i should reflect the *aggregate* bandwidth requirements of the group's *busy* connections

- Connection α_i becomes backlogged (cell arrival):
  $\rightarrow \phi_i = \phi_i + r_{\alpha_i}$
- Connection α_i becomes idle (cell departure):
  $\rightarrow \phi_i = \phi_i - r_{\alpha_i}$

Assign group F_i with new ϕ_i when

- Cell departs from group (new F_i)
- Cell arrives to idle connection in *idle* group (new F_i)
- Cell arrives to idle connection in *backlogged* group (adjust F_i)

Real-Time Computing Laboratory
The University of Michigan

Slide 13

Performance of Dynamic Hierarchical Arbitration

ON/OFF traffic with mean rate ρ and burst parameter σ

	N	r	ρ	σ
1	1	1/3	0.20000	0
2	4	1/4	0.15000	0
3	500	1/3000	0.00020	100
4	1000	1/15000	0.00004	100

- Exact SCFQ: 15001 sorting bins
- Hierarchical SCFQ: 2 groups with 5 and 6 bins, respectively
- Approximate SCFQ: 11 bins with width 1500

Real-Time Computing Laboratory
The University of Michigan

Slide 14

Performance Comparison

	FIFO		SCFQ		Approx.		Hierarch.	
	σ^{out}	$\overline{\sigma}^{out}$	σ^{out}	$\overline{\sigma}^{out}$	σ^{out}	$\overline{\sigma}^{out}$	σ^{out}	$\overline{\sigma}^{out}$
1	17851.0	252.1	55.7	49.1	217.7	198.0	2.0	0.3
2	12851.0	302.6	35.7	30.6	96.2	88.1	2.0	0.2
3	113.5	0.6	102.0	0.0	101.9	0.0	102.0	0.0
4	101.0	0.0	76.6	0.0	76.5	0.0	76.5	0.0

- Classes 1, 2, and 3 have connection start times uniformly distributed throughout their ON/OFF periods
- Class 4 has connection start times uniformly distributed throughout 1/10 of its ON/OFF periods

Slide 15

Conclusions

Efficient self-clocked fair queueing architectures, with a small, bounded number of operations on each cell arrival and departure

- Small range and set of connection rates
  → exact implementation of SCFQ
- Small range and fine grain of connection rates
  → approximate sorting with calendar queues
- Wide range of connection rates and hierarchical fairness
  → hierarchical SCFQ with static group weights
- Wide range of connection rates and connection-level fairness
  → hierarchical SCFQ with dynamic group weights
  (often *outperforms* exact SCFQ)

→ scalable implementation and performance properties

Slide 16

Ongoing and Future Work

- Simulation experiments (e.g., multi-hop nets; traffic models)
- Analysis of the approximate SCFQ algorithm
- Dynamic connection weights r_α for available-bit-rate service
- Hierarchical models for other link-scheduling algorithms
- Architectures for integrated traffic shaping and link scheduling
  (Rexford, Bonomi, Greenberg, and Wong)

Papers and slides available at
`http://www.eecs.umich.edu/~jrexford/papers.html`