

List of Figures

1.1	The architecture of a centralized shared memory router.	4
1.2	The architecture of an output queued router.	7
1.3	The input-queued and CIOQ router architectures.	10
1.4	Achieving delay guarantees in a router.	15
1.5	Emulating an ideal output queued router.	20
1.6	The data-plane of an Internet router.	22
2.1	A comparison of the CIOQ and SB router architectures.	42
2.2	The pigeonhole principle	47
2.3	The parallel shared memory router.	49
2.4	A bad traffic pattern for the parallel shared memory router.	51
2.5	Maintaining a single PIFO queue in a parallel shared memory router.	55
2.6	Maintaining N PIFO queues in a parallel shared memory router.	56
3.1	Physical view of the distributed shared memory router.	68
3.2	Logical view of the distributed shared memory router.	68
3.3	A request graph and a request matrix for an $N \times N$ switch.	71
4.1	A router with different line cards.	91
4.2	The physical and logical view of a CIOQ router.	94
4.3	The constraint set as maintained by the input.	99
4.4	The constraint set as maintained by the output.	101
4.5	A stable marriage	105
4.6	Indicating the priority of cells to the scheduler.	106
5.1	Cross-sectional view of crossbar fabric.	123
5.2	The architecture of a buffered crossbar with crosspoint buffer.	124
5.3	SuperVOQ for a buffered crossbar	135
5.4	The architecture of a buffered crossbar with output buffers.	136
6.1	Using system-wide massive parallelism to build high-speed routers.	144
6.2	The architecture of a parallel packet switch.	149
6.3	Insertion of cells in a PIFO order in a parallel packet switch.	163
6.4	A parallel packet switch demultiplexor.	169

7.1	Packet buffering in Internet routers.	185
7.2	Memory hierarchy of packet buffer.	191
7.3	Detailed memory hierarchy of packet buffer.	192
7.4	The MDQFP buffer caching algorithm.	205
7.5	Buffer cache size as a function of pipeline delay.	213
7.6	The ECQF buffer caching algorithm.	215
8.1	Scheduling in Internet routers.	231
8.2	The architecture of a typical packet scheduler.	236
8.3	A caching hierarchy for a typical packet scheduler.	239
8.4	A scheduler that operates with a buffer cache hierarchy.	243
8.5	A scheduler hierarchy that operates with a buffer hierarchy.	245
8.6	A combined packet buffer and scheduler architecture.	248
8.7	The closed-loop feedback between the buffer and scheduler.	251
8.8	The worst-case pattern for a piggybacked packet scheduler.	253
9.1	Measurement infrastructure.	266
9.2	Memory hierarchy for the statistics counters.	270
9.3	Numeric notations	278
10.1	Maintaining state in Internet routers.	288
10.2	Maintaining state on a line card.	290
10.3	The read-modify-write architecture.	297
10.4	Tradeoff between update speedup and capacity using GPP-SMA.	299
A.1	Internal architecture of a typical memory.	324
B.1	Network traffic models.	328
D.1	The scheduling phases for the buffered crossbar.	336
F.1	An non-work-conserving traffic pattern for a PPS.	348
G.1	An example of the CPA algorithm.	354
G.2	An example of the CPA algorithm (continued).	355
H.1	A worst-case traffic pattern for the buffer cache.	358

List of Tables

1.1	Comparison of router architectures.	12
1.2	Organization of thesis.	24
1.3	Application of techniques.	28
2.1	Unification of the theory of router architectures.	44
3.1	Comparison between the DSM and PDSM router architectures.	79
4.1	An example of the extended pigeonhole principle.	109
5.1	Comparison of emulation options for buffered crossbars.	137
7.1	Tradeoffs for the size of the head cache.	218
7.2	Tradeoffs for the size of the tail cache.	218
8.1	Packet buffer and scheduler implementation options.	256
8.2	Packet buffer and scheduler implementation sizes.	258
8.3	Packet buffer and scheduler implementation examples.	259
10.1	Tradeoffs for memory access rate and memory capacity.	304

List of Theorems

Theorem 2.1 (Pigeonhole Principle) Given two natural numbers p and h with $p > h$, if p items (pigeons) are put into h pigeonholes, then at least one pigeonhole must contain more than one item (pigeon).	47
Theorem 2.2 (Sufficiency) A total memory bandwidth of $3NR$ is sufficient for a parallel shared memory router to emulate an FCFS output queued router.	51
Theorem 2.3 (Sufficiency) With a total memory bandwidth of $4NR$, a parallel shared memory router can emulate a PIFO output queued router within $k - 1$ time slots.	58
Theorem 3.1 (Sufficiency) A Crossbar-based DSM router can emulate an FCFS output queued router with a total memory bandwidth of $3NR$ and a crossbar bandwidth of $6NR$	70
Theorem 3.2 (Sufficiency) A Crossbar-based DSM router can emulate a PIFO output queued router with a total memory bandwidth of $4NR$ and a crossbar bandwidth of $8NR$ within a relative delay of $2N - 1$ time slots.	70
Theorem 3.3 (Sufficiency) A Crossbar-based DSM router can emulate an FCFS output queued router with a total memory bandwidth of $3NR$ and a crossbar bandwidth of $4NR$	71
Theorem 3.4 (Sufficiency) A Crossbar-based DSM router with a total memory bandwidth of $4NR$ and a crossbar bandwidth of $5NR$ can emulate a PIFO output queued router within a relative delay of $2N - 1$ time slots.	71
Theorem 3.5 (Sufficiency) A Crossbar-based DSM router can emulate an FCFS output queued router with a total memory bandwidth of $4NR$ and a crossbar bandwidth of $4NR$	72
Theorem 3.6 (Sufficiency) A Crossbar-based DSM router can emulate a PIFO output queued router within a relative delay of $N - 1$ time slots, with a total memory bandwidth of $6NR$ and a crossbar bandwidth of $6NR$	74
Theorem 3.7 If a request matrix S is ordered, then any maximal matching algorithm that gives strict priority to entries with lower indices, such as the WFA, can find a conflict-free schedule.	76
Theorem 4.1 (Sufficiency, by Reference) Any maximal algorithm with a speedup $S > 2$, which gives preference to cells that arrive earlier, ensures that any cell arriving at time t will be delivered to its output at a time no greater than $t + \lceil B/(S - 2) \rceil$, if the traffic is single leaky bucket B constrained.	96
Theorem 4.2 (Sufficiency) With a speedup $S > 2$, a crossbar can emulate an FCFS-OQ router if the traffic is single leaky bucket B constrained.	102

Theorem 4.3 (Sufficiency, by Citation) A crossbar CIOQ router can emulate a PIFO-OQ router with a crossbar bandwidth of $2NR$ and a total memory bandwidth of $6NR$	111
Theorem 4.4 A crossbar CIOQ router is work-conserving with a crossbar bandwidth of $2NR$ and a total memory bandwidth of $6NR$	115
Theorem 5.1 (Sufficiency, by Citation) A buffered crossbar can emulate an FCFS-OQ router with a crossbar bandwidth of $2NR$ and a memory bandwidth of $6NR$	130
Theorem 5.2 A buffered crossbar (with a simplified output scheduler) is work-conserving with a crossbar bandwidth of $2NR$ and a memory bandwidth of $6NR$	130
Theorem 5.3 (Sufficiency, by Reference) A buffered crossbar can emulate a PIFO-OQ router with a crossbar bandwidth of $3NR$ and a memory bandwidth of $6NR$	132
Corollary 5.1 (Sufficiency) A crossbar CIOQ router can emulate a PIFO-OQ router with a crossbar bandwidth of $2NR$ and a total memory bandwidth of $6NR$	133
Theorem 5.4 A buffered crossbar with randomized scheduling, can achieve 100% throughput for any admissible traffic with a crossbar bandwidth of $2NR$ and a memory bandwidth of $6NR$	135
Theorem 5.5 (Sufficiency, by Reference) A modified buffered crossbar can emulate a PIFO-OQ router with a crossbar bandwidth of $2NR$ and a memory bandwidth of $6NR$	136
Theorem 6.1 A PPS without speedup is not work-conserving.	152
Theorem 6.2 (Sufficiency) A PPS can emulate an FCFS-OQ switch with a speedup of $S \geq 2$	158
Theorem 6.3 (Digression) A PPS can be work-conserving if $S \geq 2$	159
Theorem 6.4 (Sufficiency) A PPS can emulate any OQ switch with a PIFO queuing discipline, with a speedup of $S \geq 3$	161
Theorem 6.5 (Sufficiency) A PPS with independent demultiplexors and multiplexors and no speedup, with each multiplexor and demultiplexor containing a co-ordination buffer cache of size Nk cells, can emulate an FCFS-OQ switch with a relative queuing delay bound of $2N$ internal time slots. . .	172
Theorem 7.1 (Necessity & Sufficiency) The number of bytes that a dynamically allocated tail cache must contain must be at least $Q(b - 1) + 1$ bytes. . .	197
Theorem 7.2 (Necessity - Traffic Pattern) To guarantee that a byte is always available in head cache when requested, the number of bytes that a head cache must contain must be at least $Qw > Q(b - 1)(2 + \ln Q)$	198

Theorem 7.3 (Sufficiency) For MDQF to guarantee that a requested byte is in the head cache (and therefore available immediately), the number of bytes that are sufficient in the head cache is $Qw = Qb(3 + \ln Q)$ 203

Theorem 7.4 (Sufficiency) With MDQFP and a pipeline delay of x (where $x > b$), the number of bytes that are sufficient to be held in the head cache is $Qw = Q(C + b)$ 210

Theorem 7.5 (Necessity) For a finite pipeline, the head cache must contain at least $Q(b - 1)$ bytes for any algorithm. 213

Theorem 7.6 (Sufficiency) If the head cache has $Q(b - 1)$ bytes and a lookahead buffer of $Q(b - 1) + 1$ bytes (and hence a pipeline of $Q(b - 1) + 1$ slots), then ECQF will make sure that no queue ever under-runs. 217

Corollary 8.1 (Sufficiency) A packet scheduler requires no more than $Qb(4 + \ln Q) \frac{|D|}{P_{min}}$ bytes in its cache, where P_{min} is the minimum packet size supported by the scheduler, and $|D|$ is the descriptor size. 242

Corollary 8.2 (Sufficiency) A scheduler (which only stores packet lengths) requires no more than $Qb(4 + \ln Q) \frac{\log_2 P_{max}}{P_{min}}$ bytes in its cache, where P_{min} , P_{max} are the minimum and maximum packet sizes supported by the scheduler. 246

Theorem 8.1 (Sufficiency) The MDQF packet buffer requires no more than $Q[b(4 + \ln Q) + RL]$ bytes in its cache in order for the scheduler to piggy-back on it, where L is the total closed-loop latency between the scheduler and the MDQF buffer cache. 255

Theorem 8.2 (Sufficiency) A length-based packet scheduler that piggy-backs on the MDQF packet buffer cache requires no more than $Q[b(3 + \ln Q) + RL] \frac{\log_2 P_{max}}{P_{min}}$ bytes in its head cache, where L is the total closed-loop latency between the scheduler and the MDQF buffer cache. 255

Theorem 9.1 (Necessity) Under any CMA, a counter can reach a count $C(i, t)$ of $\frac{\ln [(N - 1)(b/(b - 1))^{b-1}]}{\ln(b/(b - 1))}$ 273

Theorem 9.2 (Optimality) Under all arriving traffic patterns, LCF-CMA is optimal, in the sense that it minimizes the count of the counter required. 275

Theorem 9.3 (Sufficiency) Under the LCF-CMA policy, the count $C(i, t)$ of every counter is no more than $S \equiv \frac{\ln bN}{\ln(b/(b - 1))}$ 276

Theorem 9.4 (Sufficiency) Under the LCF policy, the number of bits that are sufficient per counter, to ensure that no counter overflows, is given by $\log_2 \frac{\ln bN}{\ln d}$ 277

- Corollary 9.1 (Sufficiency) A counter of size $S \equiv \log_2[Nb(3 + \ln N)]$ bits is sufficient for LCF to guarantee that no counter overflows in the head cache. 278
- Theorem 10.1 (Sufficiency) Using GPP-SMA, a memory subsystem with h independent banks running at the line rate can emulate a memory that can be updated at C times the line rate (C reads and C writes), if $C \leq \frac{\lfloor \sqrt{4h+1} - 1 \rfloor}{2}$. 298
- Theorem J.1 (Sufficiency) A PPS, which has a maximum fanout of m , can mimic an FCFS-OQ switch with a speedup of $S \geq 2\sqrt{m} + 1$ 377
- Theorem J.2 (Sufficiency) A PPS that has a maximum fanout of m can mimic a PIFO-OQ switch with a speedup of $S \geq 2\sqrt{2m} + 2$ 379
- Theorem J.3 (Sufficiency, by Reference) A PPS can emulate a multicast FCFS-OQ router with a speedup of $S \geq 2\sqrt{N} + 1$ and a multicast PIFO-OQ router with a speedup of $S \geq 2\sqrt{2N} + 2$ 379
- Theorem J.4 (Sufficiency) A single-buffered router can emulate a multicast FCFS-OQ router with a speedup of $S \equiv \theta(\sqrt{N})$ and a multicast PIFO-OQ router with a speedup of $S \equiv \theta(\sqrt{2N})$ 379
- Theorem J.5 (Sufficiency) For MDQF to guarantee that a requested byte is in the head cache (for a copy multicast router with maximum fanout m), the number of bytes that are sufficient in the head cache is $Qw = Q(m + 1)\frac{b}{2}(3 + \ln Q)$ 380

List of Algorithms

2.1	The constraint set technique for emulation of OQ routers.	40
4.1	Constraint set-based time reservation algorithm.	98
4.2	Extended constraint sets for emulation of OQ routers.	109
4.3	Extended constraint sets for work conservation.	115
5.1	A deterministic buffered crossbar scheduling algorithm.	128
5.2	A randomized buffered crossbar scheduling algorithm.	135
6.1	Centralized parallel packet switch algorithm for FCFS-OQ emulation. .	157
6.2	Modified CPA for PIFO emulation on a PPS.	162
6.3	Distributed parallel packet switch algorithm for FCFS-OQ emulation. .	171
7.1	The most deficated queue first algorithm.	199
7.2	Most deficated queue first algorithm with pipelining.	208
7.3	The earliest critical queue first algorithm.	216
8.1	The most deficated linked list first algorithm.	241
9.1	The longest counter first counter management algorithm.	275
10.1	The generalized ping-pong state management algorithm.	298

List of Examples

1.1	The largest available commodity SRAM.	6
1.2	The largest available commodity DRAM.	6
1.3	A 100 Gb/s input queued router.	12
1.4	A push in first out queue.	15
2.1	Examples of shared memory routers.	48
2.2	Traffic pattern for a PSM router.	50
2.3	Maintaining PIFO order in a PSM router.	54
2.4	Violation of PIFO order in a PSM router.	55
3.1	Bus-based distributed shared memory router.	67
3.2	A frame-based DSM router	82
4.1	A router with different line cards.	90
4.2	Calculating the size of the input link constraint for a CIOQ router.	100
4.3	Maintaining input priority queues for a CIOQ router.	107
4.4	Opportunity and contentions sets for cells in a CIOQ router.	108
5.1	Cross section of a crossbar ASIC.	122
5.2	Implementation considerations for a buffered crossbar.	126
6.1	Using system-wide massive parallelism to build high-speed routers.	143
6.2	Limits of parallelism in a monolithic system.	147
6.3	Designing a parallel packet switch.	152
6.4	The centralized PPS scheduling algorithm	157
6.5	QoS guarantees for a PPS router.	164
6.6	Designing a parallel packet switch.	173
7.1	Numbers of queues in typical Internet routers.	187
7.2	The random cycle time of a DRAM.	188
7.3	Uses of a queue caching hierarchy.	193
7.4	Size of head cache using MDQF.	204
7.5	Size of head cache with and without pipeline delay.	211
7.6	Example of earliest critical queue first algorithm.	214
7.7	Size of packet buffer cache on a typical Ethernet switch.	222

8.1	Application and protocols on the Internet.	228
8.2	Packet scheduler operation rate.	232
8.3	Architecture of a typical packet scheduler.	235
8.4	Implementation of a typical packet scheduler.	237
8.5	A caching hierarchy for a packet scheduler.	242
8.6	A scheduler that operates with a buffer cache hierarchy.	244
8.7	A scheduler hierarchy that operates with a buffer hierarchy.	246
8.8	Counter-example pattern for a scheduler.	252
8.9	A scheduler that piggybacks on the buffer cache hierarchy.	256
9.1	Examples of measurement counters.	265
9.2	Memory access rate for statistics counters.	269
9.3	Counter design for an 100 Gb/s line card.	271
9.4	Implementation considerations for counter cache design.	280
10.1	Examples of state maintenance.	289
10.2	Maintaining state for a 100 Gb/s line card.	293
10.3	State design for an OC192 line card.	301
10.4	GPP-SMA with other caching and load balancing techniques.	302
11.1	Design and verification complexity of our techniques.	311