**Massachusetts Institute of Technology**
**Department of Electrical Engineering and Computer Science**

6.892 Fall 1999 **Lecture 11: Router Architecture** October 18, 1999

---

**Overview.** The BBN 50 Gb/s router. Label switching and Multi-Protocol Label Switching (MPLS).

# 1 BBN 50 Gb/s router

## 1.1 Why bother?

- Trends
  1. Faster and faster link bandwidths.
  2. Larger and larger network size (hosts, routers, users).
  3. Users want more and more (15% increase in per-user demand in 1 year!).
  4. Not just because of Web: for example, Internet size has been growing at 80% per year since at least 1984!
  5. Conclusion: Unlike other areas in computer science that can rely on Moore's law to achieve progress, many aspects of networking cannot!
- *Conventional "wisdom":* IP routers are inherently slow. They cannot possibly forward packets fast enough, where "fast" refers to multi-gigabit (or higher) rates.
- This paper's motivation, at least in part, was to refute this belief.

## 1.2 How does it work?

- Routers do two things: participate in routing protocols with their neighbors, and forward packets. We are concerned primarily with the latter, since it's the data path.
- At first sight, forwarding is straightforward:
  1. Router gets packet.
  2. Looks at packet header for destination.
  3. Looks up routing table for output interface.
  4. Modifies header (ttl, IP header checksum).
  5. Passes packet to output interface.
- But there's a lot of stuff to take care of: RFC 1812 is 175 pages long!
- **Challenge.** How to do all this *fast.* In particular, in the common case.
- **Architecture.** Consists of:

1. Multiple line cards.
2. Multiple forwarding engines.
3. High-speed switch fabric and processor.
4. Network processor.

- Salient features of architecture:

  1. Separate forwarding engine from line card. From input line card, send only header to engine. Engine returns modified header to input line card, after which it gets vectored to output line card.
  2. Each forwarding engine has its own routing table (cache). Notice that only a subset of the complete routing table, which gives prefix-to-output-interface mapping is needed. This is therefore a *forwarding table* rather than complete routing table.
  3. Use a switched backplane rather than a shared bus. Custom-designed for IP rather than an ATM switch. The big advantage of this is parallelism.
  4. QoS processing in router (on output line card).
  5. An abstract link-layer to handle different link technologies.

- **Forwarding engines:**

  – Key design principle: *optimize the common case.*
  – Not ASIC but based on Alpha 21164 at 415 MHz.
  – 8 KB I and D caches: forwarding code.
  – 96 KB L2 cache (Scache): cache of routing table. Only relevant table entries. 12000 routes (95% hit rate).
  – 16 MB L3 cache (Bcache): complete forwarding table in 2 8 MB chunks. [Why is it banked like this?]
  – Why not an ASIC or embedded processor? Want software. And chip has a very high clock speed and large Icache and Scache.
  – Optimize common case in custom assembler. 85 instructions to forward a packet.
  – Note: Does not check IP header checksum. Q: is this a problematic drawback?
  – Not-so-common cases deferred to network processor. This includes route cache misses, multicast, IP options, packets requiring ICMP (uses templates), packets needing fragmentation.
  – Q: Do we need a route cache in the future? A: depends on what algorithm you choose to forward.

- **Switching fabric:** (Draw pic. of switch)

  – Advantage over bus? Parallelism.
  – Disadvantage? Multicast is hard. Need scheduling machinery. (They call this "allocation").
  – Input-queued switch. Q: what's the problem with standard input-queued switch? A: h-o-l blocking.

- How to fix this? Keep track of per-output port traffic. Each input port bids for different outputs. In each epoch, match an input to output port.
- Allocator arbitrates amongst bids to do this. Maximize switch throughput (rather than bounded latency).
- Scheduling is pipelined (4 stages): get bids, allocate schedule, notify pairings, ship bits.
- Q: What graph problem does this correspond to? A: bipartite matching.
- Flow control by destination ports. Prevents an input from hogging an output.

- **Allocation algorithm:**
  - Heavily studied problem. Approaches include Parallel Iterative Matching (from DEC), wavefront scheduling, etc.
  - Use *randomization* to alleviate unfairness issues.
  - Essentially a greedy algorithm that progresses along diagonal of allocation matrix.
  - [Is this algorithm optimal?] No!
  - Prioritize forwarding engine inputs over line cards (by skewing shuffling of ports).

- **Line cards:**
  - Input: simple, except for ATM (small cells) and multicast (copies have to be made).
  - Output: QoS processor for scheduling and queue management. A VLIW FPGA.

- **Network processor:**
  - Alpha processor running NetBSD. (Another nice example in this architecture of using off-the-shelf components.)
  - Handles "not-so-common" cases of packet processing.
  - Handles routing protocols (gated).
  - Builds forwarding tables and populates table for each forwarding engine using a subset of each entry.
  - Using 2 banks in forwarding engines reduces disruptions and flapping effects; allows network processor to upload a forwarding table while forwarding is still in progress.

### 1.3 Fast lookups

- Two approaches from Sigcomm 97 papers (current state-of-the-art)
  1. Make better use of caching via compression (Degermark et al.).
  2. Constant-time lookups via clever data structures (Waldvogel et al.).
- Problem: Best-matching prefix.
- Basic idea: use hash tables for each prefix length. Then, apply binary search techniques to get to the longest matching prefix to find output interface.

### 1.4 Performance

- Peak performance of about 50 Gbps, assuming 15 simultaneous transfers take place through the switch in a single transfer cycle (called an *epoch*, equal to 16 clock ticks). (Notice that it includes the line card to forwarding engine transfers too!)
- Note: this is only an estimate, not much deployment experience/measurements reported in paper.

### 1.5 Critique & summary

- Sub-optimal switch allocation may end up affecting average case throughput.
- Actually, does not exclude forwarding engine to line card transfers in calculations of peak throughput estimates. I.e, the calculation is done by looking at spare capacity, rather than by looking at how the allocation would work. This is esp. important because of the higher priority given to forwarding engine input ports.
- Impressive design with persuasive arguments to motivate most of the design choices. It's the most detailed account of a modern (late 90s) and real router design.

## 2 Label switching

**References.** Davie et al., Switching in IP networks, MPLS working group RFCs.

### 2.1 The idea

- Forward based on fixed-length labels (or tags) rather than longest prefix. This is more efficient.
- Motivations include (i) efficiency, (ii) richer forwarding classes than simply destination-based forwarding, and (iii) the desire to leverage hardware support to forward based on labels (e.g., ATM VCI/VPI pairs), and
- Label switching enables forwarding equivalence classes (FEC) to be richly defined. An FEC is a group of packets that are all forwarded identically through a Label Switching Router (LSR).
- FEC examples. Use IP TOS fields. Use (gasp!) TCP port numbers.
- *Data path: forwarding.* Hash label into table to decide where to forward.
- *Control path: routing.* Consistent routing is main goal. Needs (i) consistent dissemination of information (ii) same algorithm at each router.

### 2.2 Forwarding using labels

- Label switching forwarding table: incoming label hashes into a list of subentries. Each subentry has outgoing label, outgoing interface and next-hop address.
- Before forwarding, *swap* label.
- Can maintain one forwarding table, or one per incoming interface.
- Over link technologies that don't support label fields (e.g., Ethernet), introduce "shim" header between network-layer and link headers.
- Label can also encapsulate resource usage information, stored in entry in forwarding table.
- Single forwarding algorithm for unicast, multicast and QoS routing. LSR behavior identical in all these cases.
- *Multiprotocol.* Doesn't matter what the network layer is, doesn't matter what the link layer is.

### 2.3 Routing/control of labels

- Local and remote binding.
- Upstream vs downstream dissemination of binding.
- Free labels.
- Control-driven vs data-driven binding.

  For control-driven, there are 2 options:

  1. Piggyback with routing protocol.
  2. Separate label distribution protocol (LDP).

### 2.4 Proposals and implementations

- Toshiba cell switching router (CSR) (1993-94).
- Ipsilon IP switching (1996).
- Cisco tag switching (led to MPLS group formation).
- IBM ARIS (aggregate route-based IP switching).
- MPLS. IETF standardization effort. (Current state-of-the-art).