15-441 Computer Networking

14 - Router Design

Based on slides from Dave Andersen
and Nick Feamster

Overview

- The design of big, fast routers
- Partridge *et al.*, *A 50 Gb/s IP Router*
- Design constraints
  - Speed
  - Size
  - Power consumption
- Components
- Algorithms
  - Lookups and packet processing (classification, etc.)
  - Packet queuing
  - Switch arbitration

News of the Week

Oct 12, 2011, Silicon.com on BlackBerry:

"The outage has been caused by the failure of a core switch on the network that connects a major RIM datacentre in Slough to other RIM datacentres worldwide, O'Neill said. 'On Monday we thought we had identified the root cause - a core switch that connects our datacentres worldwide - and what we did yesterday was to identify that and change the components and the infrastructure, and brought the BlackBerry service back again overnight. ...'..."

Mentions 20PB/month.

BlackBerry says service is fully restored after 3-day outage.

More News

Oct 12, 2011. Gordon Peterson says:

"I found that the Internet is MISROUTING IP addresses... in this case, an entire block of 255 (at least) IP addresses belonging to the U.S. State Department (!!) (when tracerouted) never get delivered to the machine the DNS lookup gives, but instead are being misrouted to another machine that seems to be outside the USA (!)... and which (worse) it appears that is at least accessible to what appears to be a Russian-based organized crime organization."

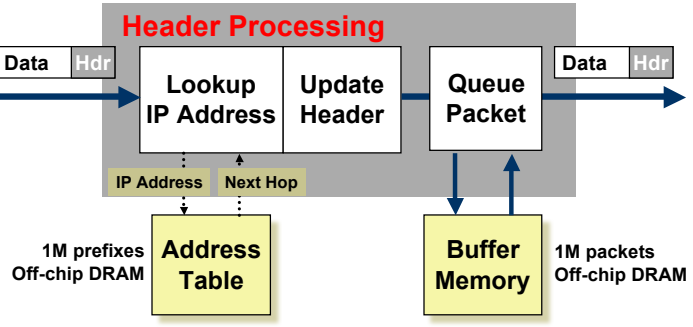
Summary of Routing Functionality

- Router gets packet
- Looks at packet header for destination
- Looks up routing table for output interface
- Modifies header (TTL, IP header checksum)

Why?

- Passes packet to output interface

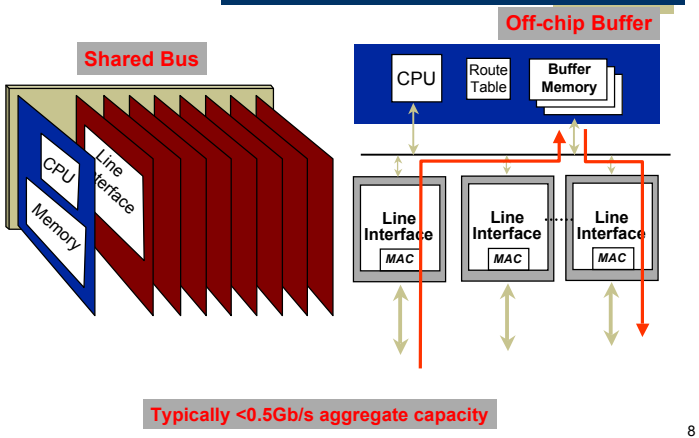
Generic Router Architecture



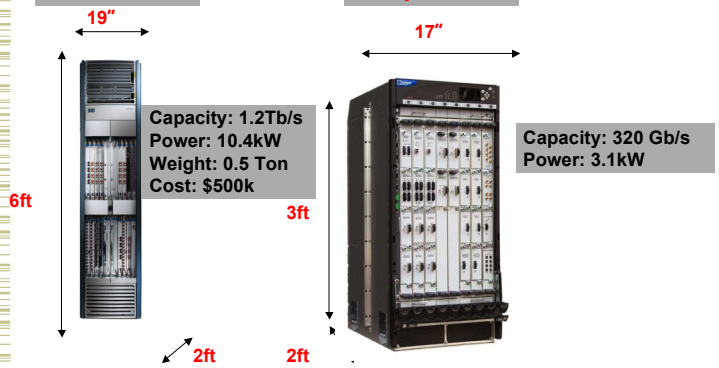
What's In A Router

- Interfaces
  - Input/output of packets
- Switching fabric
  - Moving packets from input to output
- Software
  - Routing
  - Packet processing
  - Scheduling
  - Etc.

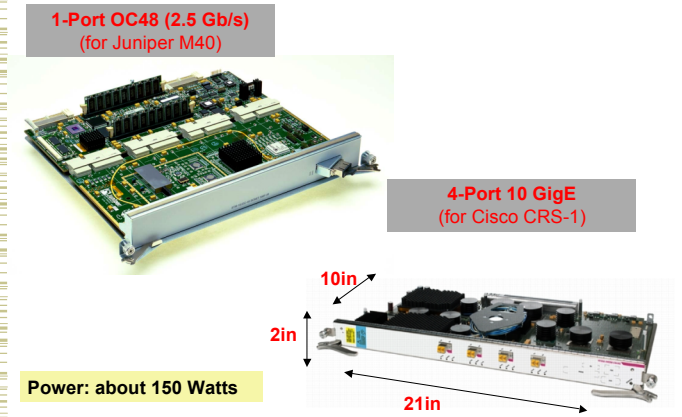
First Generation Routers



What a Router Chassis Looks Like



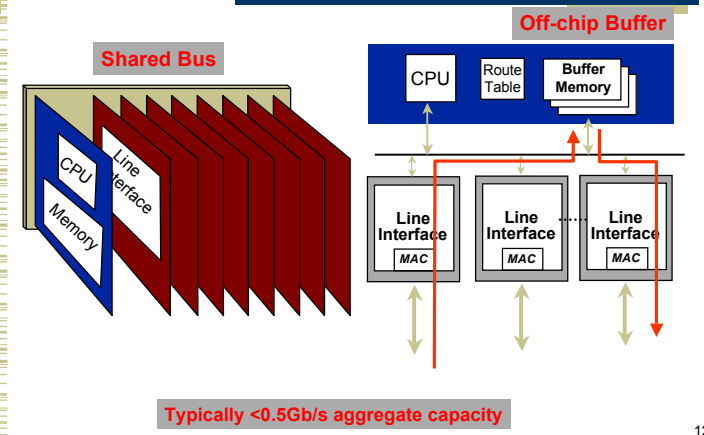
What a Router Line Card Looks Like



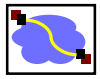
Big, Fast Routers: Why Bother?

- Faster link bandwidths
- Increasing demands
- Larger network size (hosts, routers, users)
- More cost effective

First Generation Routers



Innovation #1: Each Line Card Has the Routing Tables

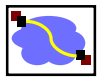


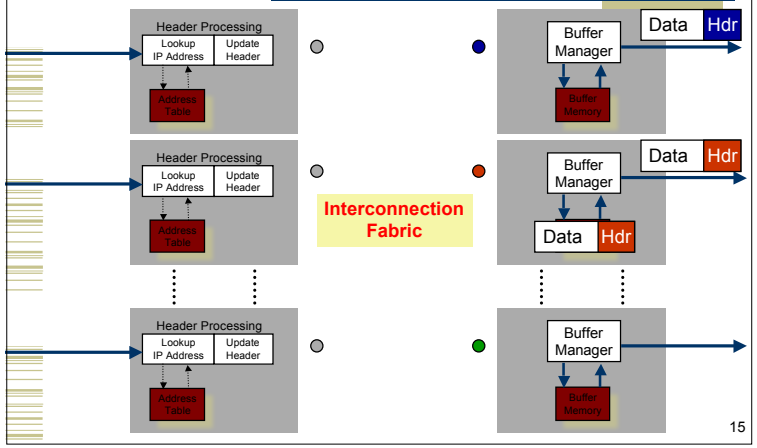
- Prevents central table from becoming a bottleneck at high speeds
- **Complication:** Must update forwarding tables on the fly.

Control Plane & Data Plane

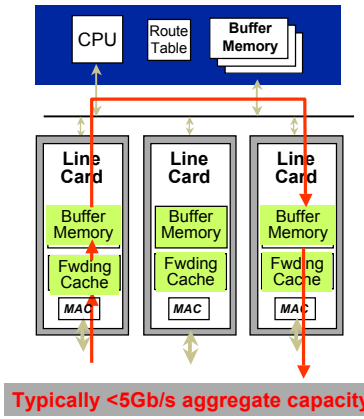
- Control plane must remember lots of routing info (BGP tables, etc.)
- Data plane only needs to know the "FIB" (Forwarding Information Base)
  - Smaller, less information, etc.
  - Simplifies line cards vs the network processor

Generic Router Architecture





Second Generation Routers

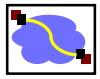


Bypasses memory bus with direct transfer over bus between line cards

Moves forwarding decisions local to card to reduce CPU pain

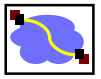
Punt to CPU for "slow" operations

Bus-based

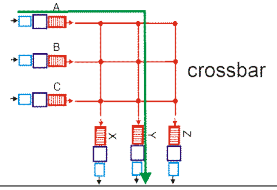


- Improvement over first generation:
  - Cache bits of forwarding table in line cards,
  - Send directly over bus to outbound line card
- But shared bus was big bottleneck
  - E.g., *modern* PCI bus (PCIx16) is only 32Gbit/sec (in theory)
  - Almost-modern Cisco (XR 12416) is 320Gbit/sec.
  - Ow! How do we get there?

Innovation #2: Switched Backplane

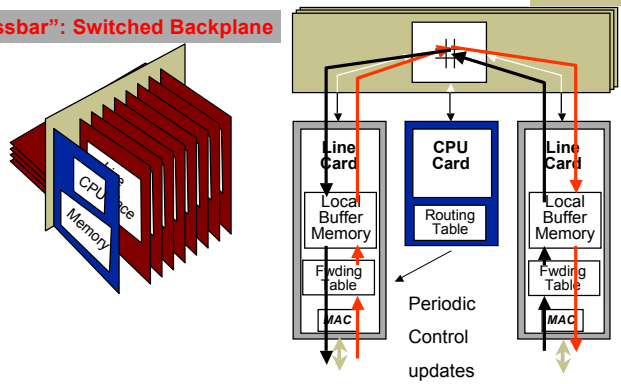


- Every input port has a connection to every output port
- During each timeslot, each input connected to zero or one outputs
- **Advantage:** Exploits parallelism
- **Disadvantage:** Need scheduling algorithm



Third Generation Routers

"Crossbar": Switched Backplane



Typically <50Gb/s aggregate capacity

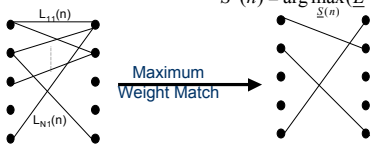
What's so hard here?

- Back-of-the-envelope numbers
  - Line cards can be 40 Gbit/sec today (OC-768)
    - Undoubtedly faster in a few more years, so scale these numbers appropriately!
  - To handle minimum-sized packets (~40b)
    - 125 Mpps, or 8ns per packet
    - But note that this can be deeply pipelined, at the cost of buffering and complexity. Some lookup chips do this, though still with SRAM, not DRAM. Good lookup algos needed still.
- For every packet, you must:
  - Do a routing lookup (where to send it)
  - Schedule the crossbar
  - Maybe buffer, maybe QoS, maybe filtering by ACLs

Crossbar Switching

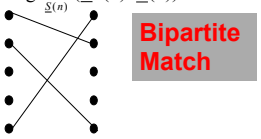
- **Conceptually:** *N* inputs, *N outputs*
  - Actually, inputs are also outputs
- In each timeslot, one-to-one mapping between inputs and outputs.
- Crossbar constraint: If input I is connected to output j, no other input connected to j, no other output connected to input I
- **Goal:** Maximal matching

Traffic Demands



Maximum Weight Match

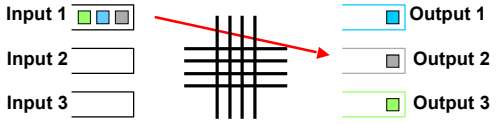
$$S^*(n) = \arg\max_{\underline{S}(n)} (\underline{L}^T(n) \cdot \underline{S}(n))$$



Bipartite Match

Head-of-Line Blocking

Problem: The packet at the front of the queue experiences contention for the output queue, blocking all packets behind it.



Maximum throughput in such a switch: 2 – sqrt(2)

What can we do about it?

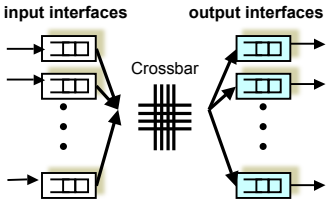
Put queues on output side.

Separate queues for each output.

M.J. Karol, M. G. Hluchyj, and S. P. Morgan, "Input Versus Output Queuing on a Space-Division Packet Switch," *IEEE Transactions On Communications*, Vol. Com-35, No. 12, December 1987, pp. 1347-1356.

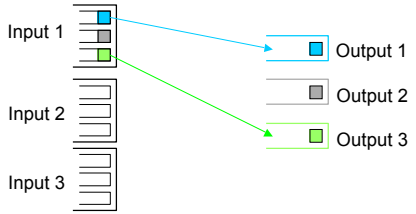
Combined Input-Output Queuing

- **Advantages**
  - Easy to build
  - Better throughput
- **Disadvantages**
  - Harder to design algorithms
  - Two congestion points



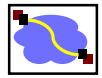
Solution: Virtual Output Queues

- Maintain N virtual queues at each input
  - one per output

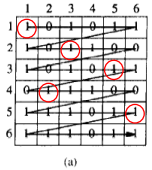


N. McKeown, A. Mekkittikul, V. Anantharam, and J. Walrand, "Achieving 100% Throughput in an Input-Queued Switch," IEEE Transactions on Communications, Vol. 47, No. 8, August 1999, pp. 1260-1267.

Early Crossbar Scheduling Algorithm



- Wavefront algorithm



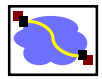
A_{ij} = 1 indicates that card i has a packet to send to card j

Problems: Fairness, speed, …

Alternatives to the Wavefront Scheduler

- PIM: Parallel Iterative Matching
  - **Request:** Each input sends requests to all outputs for which it has packets
  - **Grant:** Output selects an input at random and grants
  - **Accept:** Input selects from its received grants
- **Problem:** Matching may not be maximal
- **Solution:** Run several times
- **Problem:** Matching may not be "fair"
- **Solution:** Grant/accept in round robin instead of random

Scheduling and Fairness



- What is an appropriate definition of fairness?
  - One notion: Max-min fairness
  - Disadvantage: Compromises throughput
- Max-min fairness gives priority to low data rates/small values

Max-Min Fairness

- A flow rate x is **max-min fair** if any rate x cannot be increased without decreasing some y which is smaller than or equal to x.
- How to share equally with different resource demands
  - small users will get all they want
  - large users will evenly split the rest
- More formally, perform this procedure:
  - Split resources among all customers with unsatisfied demands
  - No customer receives more than requested
  - Iterate as long as there are more resources and unsatisfied demands

Example

- Demands: 1.9, 2.6, 4, 5; capacity: 10
  - 10/4 = 2.5
  - **Problem:** 1st user needs only 1.9; excess of 0.6,
- Distribute among 3, so 0.6/3=0.2
  - now we have allocs of [2, 2.7, 2.7, 2.7],
  - leaving an excess of 0.1 for cust #2
  - divide that in two, gets [2, 2.6, 2.75, 2.75]
- *Maximizes* the *minimum* share to each customer whose demand is not fully serviced

How to Achieve Max-Min Fairness

- **Take 1:** Round-Robin
  - Problem: Packets may have different sizes
- **Take 2:** Bit-by-Bit Round Robin
  - Problem: Feasibility
- **Take 3:** Fair Queuing
  - Service packets according to soonest "finishing time"

Adding QoS: Add weights to the queues…

Why QoS?

- Internet currently provides one single class of **"best-effort" service**
  - No assurances about delivery
- Existing applications are ***elastic***
  - Tolerate delays and losses
  - Can adapt to congestion
- Future "real-time" applications may be ***inelastic***

Router Components and Functions

- Route processor
  - Routing
  - Installing forwarding tables
  - Management
- Line cards
  - **Packet processing and classification**
  - Packet forwarding
- Switched bus ("Crossbar")
  - Scheduling

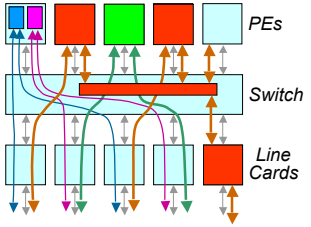
Processing: Fast Path vs. Slow Path

- **Optimize for common case**
  - BBN router: 85 instructions for fast-path code
  - Fits entirely in L1 cache
- Non-common cases handled on slow path
  - Route cache misses
  - Errors (*e.g.*, ICMP time exceeded)
  - IP options
  - Fragmented packets
  - Mullticast packets

Recent Trends: Programmability

- NetFPGA: 4-port interface card, plugs into PCI bus (Stanford)
  - Customizable forwarding
  - Appearance of many virtual interfaces (with VLAN tags)

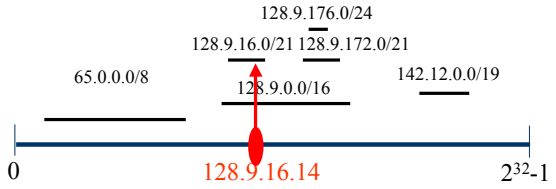
- Programmability with Network processors (Washington U.)



IP Address Lookup

- **Challenges:**
  - **Longest-prefix match (LPM).**
  - Tables are large and growing.
  - Lookups must be fast.

IP Lookups find Longest Prefixes

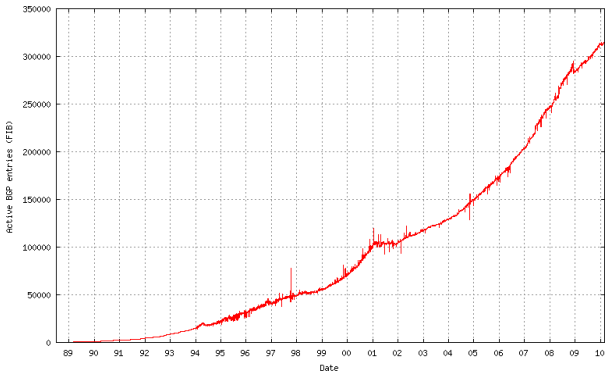


Routing lookup: Find the longest matching prefix (aka the most specific route) among all prefixes that match the destination address.

IP Address Lookup

- **Challenges:**
  - Longest-prefix match (LPM)
  - **Tables are large and growing.**
  - Lookups must be fast.

Address Tables are Large

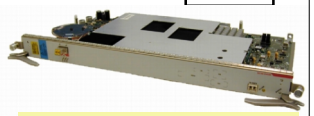


Note: FIB = Forwarding Information Base

IP Address Lookup

- **Challenges:**
  - Longest-prefix match (LPM)
  - Tables are large and growing.
  - **Lookups must be fast.**

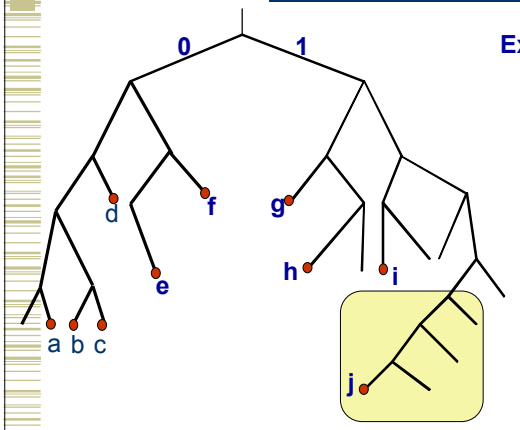
Lookups Must be Fast



Cisco CRS-1 1-Port OC-768C (Line rate: 42.1 Gb/s)

Year	Line	40B packets (Mpkt/s)	
1997	622Mb/s	1.94	**OC-12**
1999	2.5Gb/s	7.81	**OC-48**
2001	10Gb/s	31.25	**OC-192**
2003	40Gb/s	125	**OC-768**

IP Address Lookup: Binary Tries



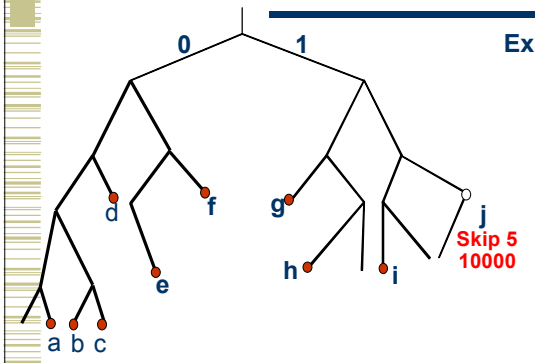
Example Prefixes:

a) 00001
b) 00010
c) 00011
d) 001
e) 0101
f) 011
g) 100
h) 1010
i) 1100
j) 11110000

Patricia Trie or Radix Trie

- Each node with only one child is merged with its child
- Edges can be labeled with sequences of bits rather than a single bit

IP Address Lookup: Patricia Trie



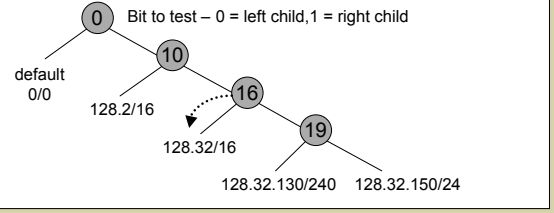
Example Prefixes

a) 00001
b) 00010
c) 00011
d) 001
e) 0101
f) 011
g) 100
h) 1010
i) 1100
j) 11110000

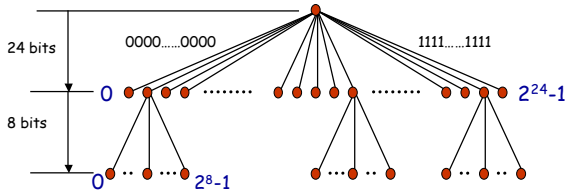
Problem: Lots of (slow) memory lookups

LPM with PATRICIA Tries

- Traditional method – Patricia Trie
  - Arrange route entries into a series of bit tests
- Worst case = 32 bit tests
  - Problem: memory speed, even w/SRAM!



Address Lookup: Direct Trie



- When pipelined, one lookup per memory access
- **Inefficient use of memory**

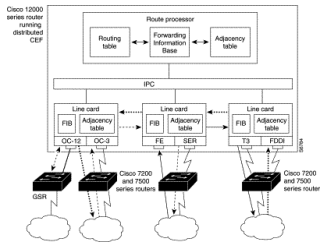
Faster LPM: Alternatives

- Content addressable memory (CAM)
  - **Hardware-based** route lookup
  - Input = tag, output = value
  - Requires exact match with tag
    - Multiple cycles (1 per prefix) with single CAM
    - Multiple CAMs (1 per prefix) searched in parallel
  - Ternary CAM
    - (0,1,don't care) values in tag match
    - Priority (*i.e.*, longest prefix) by order of entries

Historically, this approach has not been very economical.

Faster Lookup: Alternatives

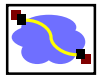
- Caching
  - Packet trains exhibit temporal locality
  - Many packets to same destination
- Cisco Express Forwarding



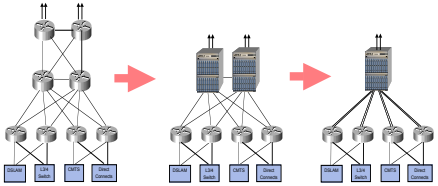
IP Address Lookup: Summary

- Lookup limited by memory bandwidth.
- Lookup uses high-degree trie.
- State of the art: 10Gb/s line rate.
- Scales to: 40Gb/s line rate.

Fourth-Generation: Collapse the POP

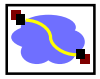


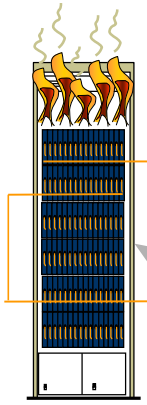
High Reliability and Scalability enable “vertical” POP simplification



Reduces CapEx, Operational cost
Increases network stability

Fourth-Generation Routers

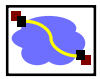


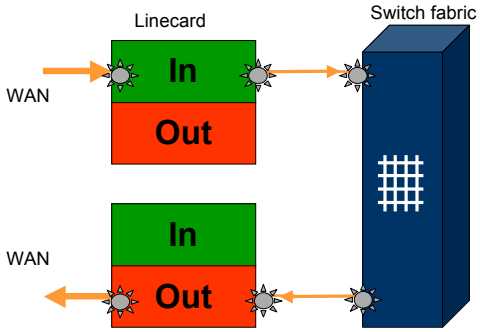


Switch

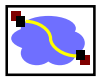
Limit today ~2.5Tb/s
- Electronics
- Scheduler scales <2x every 18 months
- Opto-electronic conversion

Multi-rack routers





Router Design



- Many trade-offs: power, $$$, throughput, reliability, flexibility
- Move towards distributed architectures
  - Line-cards have forwarding tables
  - Switched fabric between cards
  - Separate Network processor for “slow path” & control
- Important bottlenecks on fast path
  - Longest prefix match
  - Cross-bar scheduling
- Beware: lots of feature creep

Summary

- Modern network data rates are *much* too fast for conventional processors and multiple interface cards on a bus
- Partition work into:
  - Examine header, make decisions
  - Move the bits
- Parallelism is essential
  - Line cards process incoming packets on each input
  - Switch fabric complex but avoids bottleneck
- Caching and other optimizations needed
  - and possible for common cases
  - "slow" cases handled out of the main, "fast" path