Switching Technology: Outline

- Networking review
- Digital hardware issues
  - Router architecture
  - Switch fabrics
  - Packet classification

Overview

Packet switched network—ensemble of hosts, links, and routers

- Kleinrock: MIT PhD early 1960s, UCLA early 1970s
  - efficient
  - reliability
  - simplicity (end-to-end argument)

Networking hierarchy

- Physical layer—lasers, fibers, RF, antennas, modulation, demodulation
- Datalink layer—Ethernet, SONET
- Network layer—IP
- Transport layer—TCP, UDP

Internet architecture

- LANs (ACES building)
- enterprise (UT system)
- access (MCI connection)
- core (Sprint network)

Current problems

Formerly—Internet routers were general purpose computers connected via bus to Tx/Rx hardware

- link bandwidth bottleneck ⇒ conventional processor could implement entire router: check-summing, packet encapsulation, routing, billing, firewalling, security

Today—high-speed optical fiber technology, added functionality ⇒ routers are bottleneck

Solution

Many operations not performance-critical ⇒ can still performed with general purpose computers

- Implement high performance router as combination of workstation-class CPU and collection of specialized ICs
  - Processor for "control functions"—compute best routes, analyzing traffic statistics, etc.
  - Chips for line speed operations—packet encapsulation/decapsulation, longest prefix matching, scheduling the switching fabric and output queues, etc.

Hardware: ASICs, network processors, FPGAs

What do routers do?

A lot

- Compute routing tables
- Forward packets
- Switching
- Manage different classes of service
- Congestion control
- Security (encryption, DOS prevention, AV)
- Fragmentation/reassembly
- SNTP, ICMP
- NAT, Load balancing, Firewalling

- Accounting, Peering
- Multicast
- Multiprotocol operation

Router Architecture: low-end

Use general-purpose workstation

- equip with a number of network interfaces

Possible bottlenecks: CPU, memory, I/O bus

- assuming DMA: each packet crosses I/O bus twice, written to and read from main memory once

⇒ upper bound on throughput is $\min$(half main memory BW, half I/O bus BW)

- P&D: usually I/O bus BW is bottleneck

Case study: Alpha workstation

- I/O bus advertises 100MBps (32 bit bus at 25 Mhz)
  - 8 clock cycles of overhead in bus capture ⇒ 48 byte ATM cell transfer takes 20 cycles
- memory-CPU is 114MBps ⇒ if we have n reads/writes, throughput goes to 114 MBps/n
  - $n = 5$ is not uncommon (copy from one buffer to another) ⇒ 22MBps (176Mbps)

Integrate buffers used by device drivers, OS, application to minimize copies, exploit cache

Design considerations

- Design cost
  - time to design, number of designers
- Manufacturing cost
  - technology, amount of memory, ASIC/FPGA/μP, pin count
- Deployment cost
  - administrator time, flexibility, ease of upgrades, downtime

Switching Technology: Outline

- Networking review
- Digital hardware issues
  - ⇒ **Router architecture**
  - Switch fabrics
  - Packet classification

Generic architecture: input ports, output ports, switching fabric, routing processor

- input port—point of attachment for physical link, point of entry for incoming packets
- output port—stores packets, schedules them for service on output link
  - reside on line cards (possibly many ports)
- switch fabric—interconnects input & output ports
- routing processor—participates in routing protocols, creates forwarding table

Input port/linecard functions:

- datalink decapsulation, datalink protocols
- route lookup, classification

Switching fabrics

- bus — overhead, scales poorly
- shared memory — switch pointers, still limited by memory bandwidth
- crossbar — connect any of N inputs to N outputs (simple N^2 complexity implementation in VLSI)

Packet processing

Previously assuming that moving data is only problem

- not true when lots of short packets (parsing header, computing next-hop)
- 100,000 pps on a fast workstation, 64 byte packets $\Rightarrow$ overal throughput is $10^5 \times 64 \times 8$ = 51.2Mbps
  - shared by all users

Router architecture: high-end

Keep in mind different applications have different needs

- core — reliability & speed
  - replication: dual datapaths, power supplies, etc.
  - custom hardware for classification and scheduling
- enterprise — cheap, QOS, firewalling, multicast
- access — formerly modem pool attached to concentrator
  - DSL/cable, need for complex protocols

Switching Technology: Outline

- Networking review
- Digital hardware issues
  - Router architecture
  - Switch fabrics
  - Packet classification

Switching fabrics

- Bus — limited by loading, arbitration
- Memory — limited by memory access speeds
- Crossbar — limited by scheduler
- Low degree fabrics: Benes, Torsu, Clos, etc. — limited by scheduler, configuration

  Fixed width crossbars are the primary choice today

  - break variable sized IP packets into fixed width "cells"

Packet classification

Longest prefix-matching for IP forwarding

- Software based approaches — memory space-time tradeoffs
  - algorithm using 8 lookups to 60ns access time DRAM ⇒ 2 Mpps
    - use 10ns SRAM ⇒ 12.5 Mpps

Table updates are relatively slow ⇒ can use complex data structures

- tree variants, hash tables

Hardware based approaches

- CAM + priority encoder — scales poorly
- expand table, use DRAM — updates complex

Generalizations: flow identification, classification on multiple fields, stateful classification

- all open problems

Reducing port cost

Port cost is a function of

- amount/type of memory: SRAM/DRAM, amount?
- processing power: ASIC, μP?
- communication between routing processor and port: use switching fabric?

Does ϕ hardware cost dominate?

Miscellaneous

Network level management: compute routes, determine policies, etc.

Operating system: abstract hardware, build services on top of an API

Output queuing vs. input queuing

Output queuing — very natural, easy implement QOS, congestion control

- big disadvantage — all input ports may have packets headed to same output
  - partially overcome by using cell-wide memories

Input queuing — advance only one cell under contention

- HOL blocking: can overcome with multiple queues at input
  - yields a nontrivial scheduling problem

Hybrid approach — emulate output queuing using memories at inputs and outputs

- scheduler becomes bottleneck

Link scheduling

- Output queued router: input and output line rates same $\Rightarrow$ buffering needed
  - how to serve packets in buffer?

Obvious — first-come first-served (FCFS)

- no preferential service, protection

Fair Queuing

- each source sharing output link given a weight
  - focus on sources that are backlogged: source k gets $R \cdot w_k / \sum w_i$
  - discrete packets makes life more complicated
- need added state, hardware to implemented FQ

Fabrics

References:

- Architectural choices in large scale ATM switches. J. Turner and N. Yamanaka. IEICE Transactions, 1998
- An Engineering Approach to Computer Networking. S. Keshav. Addison-Wesley, 1997.
- Parallel Algorithms and Architectures. T. Leighton. Morgan Kaufmann, 1992.

ATM

ATM vision: combine flexibility of Internet with per-user QOS of telephone network

- virtual circuits, VCI "virtual channel identifier"
- fixed size cells
- statistical muxing
- integrated services

First two were thought key to building big switches in hardware

- signaling for set up and tear down (significant delay/overhead for short connections)
- each switch needs state for each VC
- small identifier reduces overhead (fast classification algorithms make this insignificant)
- all cells must follow same route — failure recovery difficult
- VC does not automatically guarantee QOS

Switching fabrics

Use to move packets from inputs to outputs: appear in L2 and L3 devices

- Single-stage — simple but fewer ports, less throughput
  - Buses, rings, crossbars
- Multi-stage
  - Banyan, Benes, Clos

Issues: scalable, reliable, cost-effective

Queueing issues

Peak data rate much higher than average — use statistical multiplexing

Buses/rings with bandwidth equal to sum of input link bandwidths: queuing primarily at OPPs

- memory needed dependent on ratio of link rate to peak transmission rate of individual streams
  - high ratio (20:1) — small buffers suffice
  - small ratio — 10$\times$ average sustained data burst
    - 1 MB image $\Rightarrow$ 10 MB buffer
    - delay becomes an issue — use multiple priority levels/per VC queues

Shared memory

Large queuing memories at output ports for statistical muxing $\Rightarrow$ average memory utilization is low

Bus/ring architecture — add large shared memory

- contains per-link /per-VC queues
- bus/ring bandwidth needs to be doubled, high memory bandwidth
- typical — 2-3$\times$ memory for single output port
- 16 port OC-3 switch 5–8$\times$ reduction in memory, 64 $\rightarrow$ 128 bit bus, memory subsystem handling 600 MB/s

Buses

Bus supporting n ports operating at R bps must provide at least Rn bps bandwidth

- if bus operates at r Hz, width must be Rn/r
  - 16 OC-3 links with internal clock of 40 MHz need 64 bit bus
- poor scaling
  - increase ports $\Rightarrow$ increase bandwidth **and** RC loading width increases as n^3
- common for smaller switches (few Gbps)

Rings

IPP and OPP have ring interfaces which insert data onto ring, remove data from ring

- simplest — time slotted approach
  - cells sent synchronously on specified time slots, busy bit indicates availability
- point-to-point communication $\Rightarrow$ decoupling
  - rings have better growth characteristics; added latency is minimal

Crossbars

Matrix of N^2 crosspoints:

- can be much less expensive than bus/ring of same performance, since multiple data transfers can take place simultaneously

If queuing primarily at outputs need a “speed advantage”

- run interface from IPP/OPP to crossbar higher than link rate
- worst case — speedup of N needed
- bursty traffic with low peak rates — factor of two suffices

Avoid speedup by queuing at inputs:

- cannot use single FIFO queue because of HOL blocking
- maintain separate queues for each output port
- control is much more complex
  - IPPs signal to central crossbar controller
  - controller maximizes number of cells that can be switched
  - matching problem in bipartite graphs — need heuristics (e.g., iSLIP)

Avoid speedup by buffering at each crosspoint:

- avoids “head-of-line” blocking, buffers storing cells for any specific output all compete for that output
- huge increase in complexity of crosspoints

Multistage switching fabrics

Overcome quadratic growth complexity by parallelism

- constituent switch elements can store (small) number of cells
- routing through switch — static/dynamic

— internal links need not be much faster than line speed

Requires a resequencing buffer, since cells can arrive along different paths

- 1000 ports — resequencing buffer of size 50–100 cells statistically acceptable

Static routing

Statically routed switch — all cells in a VC follow same path

- IPP inserts path specification into the cell headers
- can perform static routing with many interconnection topologies, including Benes network

3-stage Clos network similar to Benes network, except that switch elements in stage 1 are $d \times r$, stage 2 are $d \times d$, and stage 3 are $r \times d$

- when new VC added, control processor must find some path with enough unused bandwidth on each link
  - checks r pairs of links

Banyan network

Dynamically routed multistage fabric built from 2×2 switches

- Banyan on 2^n outputs: n stages each with 2^{n-1} elements
- element in i-th stage examines i-th bit of output-id

Collision: two packets want to go to same output port,

- introduce packet buffers
- uniform random traffic single packet buffer ⇒ throughput is 45%

Benes networks

$B_{n,d}$ — n is number of input/output ports, d is number of inputs and outputs of constituent switch elements

- recursive construction $n = d$, single $d \times d$ switch element
- $n = d^2$ consists of 3 stages with d elements in each stage
  - first stage distributes incoming cells across all switches in middle to balance load
  - second and third stages route cells to outputs
    * output port id: pair of base d digits
- $n = d^k$: construct d networks with $n = d^{k-1}$, precede and antecede by d^{k-1} switch elements each linked to all central subnetworks

Because of load balancing, achieve excellent performance

Packet Classification

References:

- P. Gupta. Algorithms for routing lookups and packet classification. PhD Thesis, Stanford University, 2001.
  - Excellent survey, pages 106–128; recursive flow classification

Motivation

IP networks have very limited support for QoS, security, accounting

- customers unwilling to commit to IP-based infrastructure

In principle, packet classification could help solve these problem

- differentiate between customers, type of traffic

In practise, performance drops dramatically

- number of algorithms & architectures for fast packet classification

Theorem: blocking can alway be avoided if $r > 2\lceil(\beta d - B)/(1 - B)\rceil - 2$

- β: reciprocal of speed advantage
- B: ratio of maximum rate of any single VC to internal link bandwidth

Switching Technology: Outline

- Networking review
- Digital hardware issues
  - Router architecture
  - Switch fabrics
  - ⇒ **Packet classification**

Computational complexity

First implementation of firewalls: linear scan, with some tricks

- works fine for 3 Mb Ethernet, 10 rules in rule-set

Innate complexity: performing classification is akin to searching geometrical structures

- recall from IP forwarding: prefix corresponds to interval of $[0, 2^{32} - 1]$

Given set of N disjoint ranges $\{[l_1, u_1], \ldots, [l_N, u_N]\}$ partitioning $[0, 2^{32} - 1]$ range lookup problem is to find range corresponding to point P

- assumption: lots of lookups, can spend time preprocessing set
- various solutions: binary search on endpoints, prefix matching (convert range to set of prefixes)
  - $[4, 7]$ — 01$**$; Interval $[3, 8]$ — 0011, 01$**$, 1000; Interval $[1, 14]$ — 0001, 001$*$, 01$**$, 10$**$, 110$*$, 1100
  - worst case: range on W-dimensions becomes $2W - 2$ prefixes

Syntax and semantics of rules

Will study classification based on header fields

- IP src/dest addresses, protocol, TCP src/dest ports

Syntactically, rule is *(predicate, action)* pair

- predicate is sequence of prefixes, one per header field
  - in some formulations, use intervals for succinctness
- action is an integer in some range $[0, A - 1]$

Semantically, packet satisfies predicate iff each header field from packet matches corresponding prefix

- rule-set — collection of rules
- each rule is given a priority, action taken is that of (any) highest priority rule satisfied by packet

Working example

Rule	F1	F2
R1	00$*$	00$*$
R2	0$*$	01$*$
R3	1$*$	0$*$
R4	00$*$	0$*$
R5	0$*$	1$*$
R6	$*$	1$*$

Rules ordered from top to bottom

How to lookup $(v_1, v_2, \ldots, v_d)$?

- proceed recursively in each dimension
- at **each** F1-trie node, follow next-trie pointer recursively
- encounter a rule iff rule matches $\Rightarrow$ keep record of highest priority rule

Complexity:

- space — $\Theta(N \cdot d \cdot W)$ "rake"
- time — $\Theta(W^d)$

Set-pruning tries

Improve search time by replication

- construction similar to that of hiearchical trie
  - for **each** prefix p in $F1$, recursively construct $(d-1)$-dimensional trie T_p on those rules which specify p **or** a prefix of p in first dimension
  - conceptually "pushing" down prefixes

Hardware solution — TCAMs

Very much in spirit of IP forwarding. TCAM essential, cannot do with CAMs (great diversity in lengths)

Very much as in IP forwarding

- select highest priority matching rule with priority encoder
- need to pipeline for performance
- various architectures trading off area/time, lookup/update
- same problems (density/cost, power)

Hierachical tries

Natural extension of tries; construct recursively

- $d = 1$, very similar to usual trie
- $d > 1$, construct $F1$ trie on prefixes from first dimension
  - use next-trie pointer to link p to T_p

Convince yourself that the hierarchical trie actually captures rule-set

Search for $(v_1, v_2, \ldots, v_d)$ by finding longest matching prefix of v_1, follow its next trie pointer, recurse

- guaranteed every matching rule encountered

Complexity:

- time — $\Theta(W \cdot d)$ (no backtrack!)
- space — $\Theta(N^d \cdot d \cdot W)$