15-441 Computer Networking
Lecture 10 – Routers and Routing

Peter Steenkiste

Departments of Computer Science and

Electrical and Computer Engineering

15-441 Networking, Spring 2008

http://www.cs.cmu.edu/~dga/15-441/S08

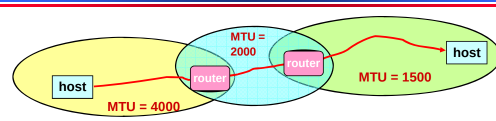
Outline

- ICMP
- How do Routers Works?
- Routing
- Distance vector

Internet Control Message Protocol (ICMP)

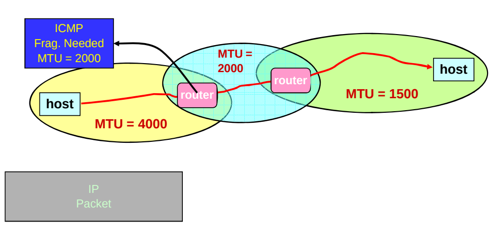
- Short messages used to send error & other control information
- Examples
  - » Ping request / response
    - – Can use to check whether remote host reachable
  - » Destination unreachable
    - – Indicates how packet got & why couldn't go further
  - » Flow control
    - – Slow down packet delivery rate
  - » Redirect
    - – Suggest alternate routing path for future messages
  - » Router solicitation / advertisement
    - – Helps newly connected host discover local router
  - » Timeout
    - – Packet exceeded maximum hop limit

IP MTU Discovery with ICMP

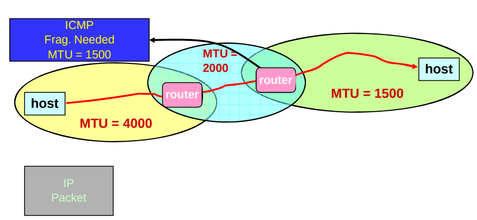


- Typically send series of packets from one host to another
- Typically, all will follow same route
  - » Routes remain stable for minutes at a time
- Makes sense to determine path MTU before sending real packets
- Operation
  - » Send max-sized packet with "do not fragment" flag set
  - » If encounters problem, ICMP message will be returned
    - – "Destination unreachable: Fragmentation needed"
    - – Usually indicates MTU encountered

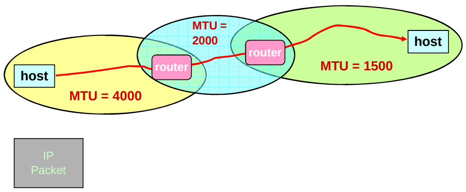
IP MTU Discovery with ICMP



IP MTU Discovery with ICMP



IP MTU Discovery with ICMP



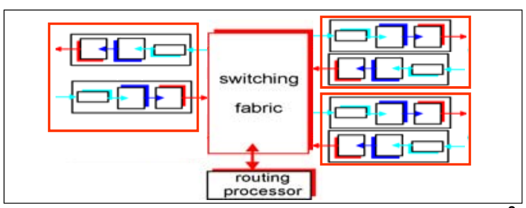
- » When successful, no reply at IP level
  - – "No news is good news"
- » Higher level protocol might have some form of acknowledgement

Outline

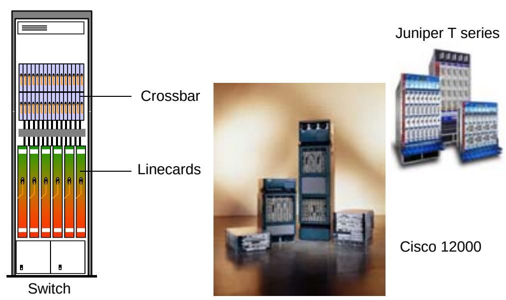
- ICMP
- How do Routers Works?
- Routing
- Distance vector

Router Architecture: Key Functions

- Run routing algorithms/protocol (RIP, OSPF, BGP)
  - » Done by routing processor
- *Switching* datagrams from incoming to outgoing link
  - » Common case handled by line cards



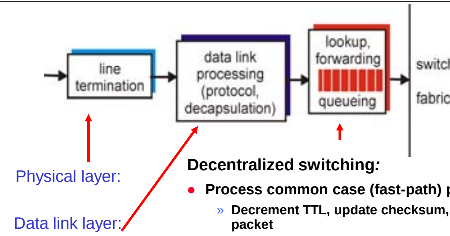
Router Physical Layout



Line Cards

- Often uses special purpose hardware (e.g. ASICs)
- Network interface cards
- Fast path (common-case) processing
  - » Decrement TTL
  - » Recompute checksum
  - » Forward to next hop line card
    - – Forwarding engine

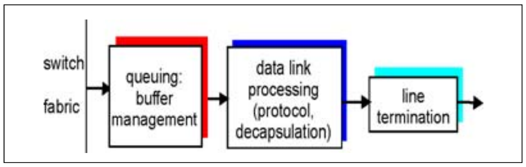
Line Card: Input Port



Decentralized switching*:*

- Process common case (fast-path) packets
  - » Decrement TTL, update checksum, forward packet
- Given datagram dest., lookup output port using routing table in input port memory
- Queue needed if datagrams arrive faster than forwarding rate into switch fabric

Line Card: Output Port

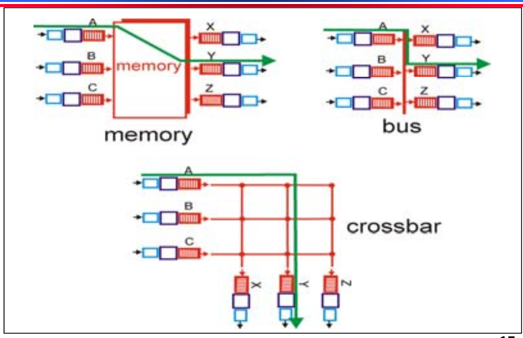


- Queuing required when datagrams arrive from fabric faster than the line transmission rate

Router Processor

- Runs routing protocol and downloads forwarding table to forwarding engines
- Performs “slow” path processing
  - » ICMP error messages
  - » IP option processing
  - » Fragmentation
  - » Packets destined to router

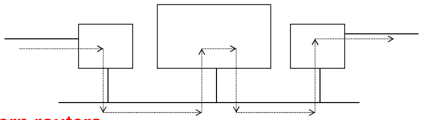
Three Types of Switching Fabrics



Switching Via a Memory

First generation routers → looked like PCs

- Packet copied by system’s (single) CPU
- Speed limited by memory bandwidth (2 bus crossings per datagram)

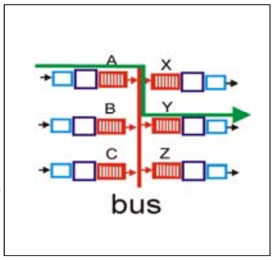


Modern routers

- Input port processor performs lookup, copy into memory
- Cisco Catalyst 8500

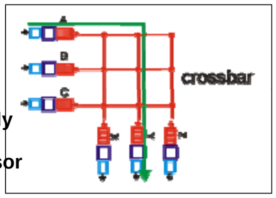
Switching Via a Bus

- Datagram from input port memory to output port memory via a shared bus
- Bus contention: switching speed limited by bus bandwidth
- 1 Gbps bus, Cisco 1900: sufficient speed for access and enterprise routers (not regional or backbone)



Switching Via an Interconnection Network

- Overcome bus bandwidth limitations
- Crossbar provides full NxN interconnect
  - » Expensive
- Banyan networks & other interconnection nets initially developed to connect processors in multiprocessor
  - » Typically less capable than complete crossbar
- Cisco 12000: switches Gbps through the interconnection network



Buffering

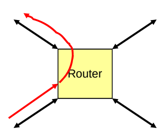
- Suppose we have N inputs and M outputs
  - » Multiple packets for same output → output contention
  - » Switching fabric may force different inputs to wait → Switch contention
- Solution – buffer packets when/where needed: input, switch, or output
- What happens when these buffers fill up?
  - » Packets are THROWN AWAY!! This is where packet loss comes from

Outline

- ICMP
- How do Routers Works?
- Routing
- Distance vector

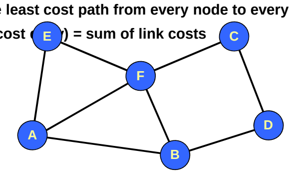
IP Forwarding versus Routing

- The Story So Far…
  - » IP addresses are structure to reflect Internet structure
  - » IP packet headers carry these addresses
  - » When Packet Arrives at Router
    - – Examine header to determine intended destination
    - – *Look up in table to determine next hop in path*
    - – Send packet out appropriate port
- How do we generate the forwarding table?



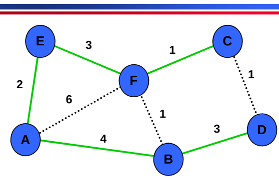
Graph Model

- Represent each router as node
- Direct link between routers represented by edge
  - » Symmetric links ⇒ undirected graph
- Edge "cost" c(x,y) denotes measure of difficulty of using link
  - » delay, $ cost, or congestion level
- Task
  - » Determine least cost path from every node to every other node
    - – Path cost ( ) = sum of link costs



Routes from Node A

Forwarding Table for A		
Dest	Cost	Next Hop
A	0	A
B	4	B
C	6	E
D	7	B
E	2	E
F	5	E



- Properties
  - » Some set of shortest paths forms tree
    - – Shortest path spanning tree
  - » Solution not unique
    - – E.g., A-E-F-C-D also has cost 7

Ways to Compute Shortest Paths

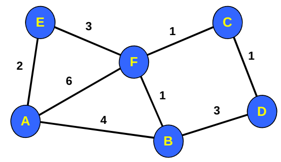
- Centralized
  - » Collect graph structure in one place
  - » Use standard graph algorithm
  - » Disseminate routing tables
- Link-state
  - » Every node collects complete graph structure
  - » Each computes shortest paths from it
  - » Each generates own routing table
- Distance-vector
  - » No one has copy of graph
  - » Nodes construct their own tables iteratively
  - » Each sends information about its table to neighbors

Outline

- ICMP
- How do Routers Works?
- Routing
- Distance vector

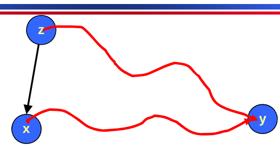
Distance-Vector Method

Initial Table for A		
Dest	Cost	Next Hop
A	0	A
B	4	B
C	∞	–
D	∞	–
E	2	E
F	6	F



- **Idea:** At any time, have cost/next hop of best known path to destination
  - » Use cost ∞ when no path is known
- **Initially**
  - » Only have entries for directly connected nodes

Distance-Vector Update



- Update(x,y,z)

d ← c(x,z) + d(z,y)   # Cost of path from x to y with first hop z
if d < d(x,y)
  # Found better path
  return d,z   # Updated cost / next hop
else
  return d(x,y), nexthop(x,y)   # Existing cost / next hop

Algorithm

- Bellman-Ford algorithm
- Repeat
  - For every node x
    - For every neighbor z
      - For every destination y
        - d(x,y) ← Update(x,y,z)
- Until converge

Start

Table for A		
Dst	Cst	Hop
A	0	A
B	4	B
C	∞	–
D	∞	–
E	2	E
F	6	F

Table for B		
Dst	Cst	Hop
A	4	A
B	0	B
C	∞	–
D	3	D
E	∞	–
F	1	F

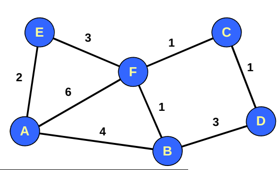


Table for C		
Dst	Cst	Hop
A	∞	–
B	∞	–
C	0	C
D	1	D
E	∞	–
F	1	F

Table for D		
Dst	Cst	Hop
A	∞	–
B	3	B
C	1	C
D	0	D
E	∞	–
F	∞	–

Table for E		
Dst	Cst	Hop
A	2	A
B	∞	–
C	∞	–
D	∞	–
E	0	E
F	3	F

Table for F		
Dst	Cst	Hop
A	6	A
B	1	B
C	1	C
D	∞	–
E	3	E
F	0	F

Iteration #1

Table for A		
Dst	Cst	Hop
A	0	A
B	4	B
C	7	F
D	7	B
E	2	E
F	5	E

Table for B		
Dst	Cst	Hop
A	4	A
B	0	B
C	2	F
D	3	D
E	4	F
F	1	F

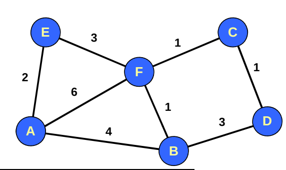


Table for C		
Dst	Cst	Hop
A	7	F
B	2	F
C	0	C
D	1	D
E	4	F
F	1	F

Table for D		
Dst	Cst	Hop
A	7	B
B	3	B
C	1	C
D	0	D
E	∞	–
F	2	C

Table for E		
Dst	Cst	Hop
A	2	A
B	4	F
C	4	F
D	∞	–
E	0	E
F	3	F

Table for F		
Dst	Cst	Hop
A	5	B
B	1	B
C	1	C
D	2	C
E	3	E
F	0	F

Iteration #2

Table for A			Table for B		
Dst	Cst	Hop	Dst	Cst	Hop
A	0	A	A	4	A
B	4	B	B	0	B
C	6	E	C	2	F
D	7	B	D	3	D
E	2	E	E	4	F
F	5	E	F	1	F

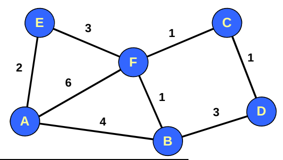
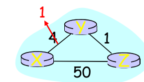


Table for C			Table for D			Table for E			Table for F		
Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop
A	6	F	A	7	B	A	2	A	A	5	B
B	2	F	B	3	B	B	4	F	B	1	B
C	0	C	C	1	C	C	4	F	C	1	C
D	1	D	D	0	D	D	5	F	D	2	C
E	4	F	E	5	C	E	0	E	E	3	E
F	1	F	F	2	C	F	3	F	F	0	F

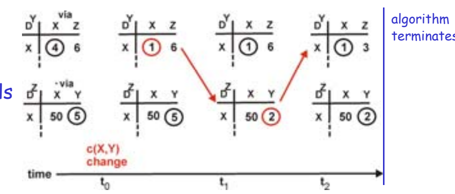
Distance Vector: Link Cost Changes

Link cost changes:

- Node detects local link cost change
- Updates distance table
- If cost change in least cost path, notify neighbors



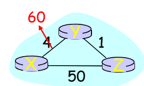
"good news travels fast"

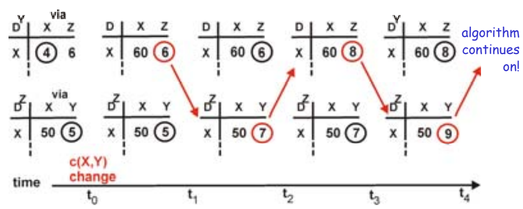


Distance Vector: Link Cost Changes

Link cost changes:

- Good news travels fast
- Bad news travels slow - "count to infinity" problem!

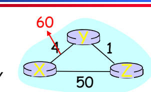


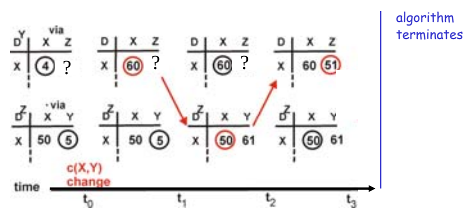


Distance Vector: Split Horizon

If Z routes through Y to get to X :

- No need for Z to tell Y about its route to X
- Z does not advertise its route to X back to Y

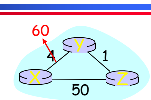


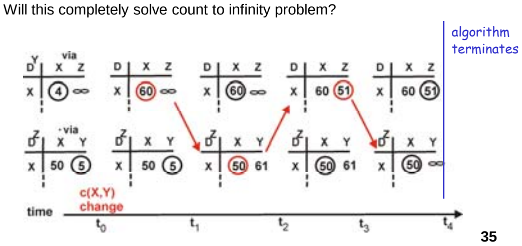


Distance Vector: Poison Reverse

If Z routes through Y to get to X :

- Z tells Y its (Z's) distance to X is infinite (so Y won't route to X via Z)
- Eliminates some possible timeouts with split horizon
- Will this completely solve count to infinity problem?





Poison Reverse Failures

Table for A			Table for B			Table for D			Table for F		
Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop
C	7	F	C	8	A	C	9	B	C	1	C

Table for A		
Dst	Cst	Hop
C	∞	–

Table for F		
Dst	Cst	Hop
C	∞	–

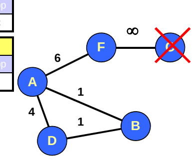


Table for A		
Dst	Cst	Hop
C	13	D

Table for B		
Dst	Cst	Hop
C	14	A

Table for D		
Dst	Cst	Hop
C	15	B

Table for A		
Dst	Cst	Hop
C	19	D

- **Iterations don't converge**
- **"Count to infinity"**
- **Solution**
  - **Make "infinity" smaller**
  - **What is upper bound on maximum path length?**

Routing Information Protocol (RIP)

- Earliest IP routing protocol (1982 BSD)
  - » Current standard is version 2 (RFC 1723)
- Features
  - » Every link has cost 1
  - » "Infinity" = 16
    - – Limits to networks where everything reachable within 15 hops
- Sending Updates
  - » Every router listens for updates on UDP port 520
  - » RIP message can contain entries for up to 25 table entries

RIP Updates

- Initial
  - » When router first starts, asks for copy of table for every neighbor
  - » Uses it to iteratively generate own table
- Periodic
  - » Every 30 seconds, router sends copy of its table to each neighbor
  - » Neighbors use to iteratively update their tables
- Triggered
  - » When every entry changes, send copy of entry to neighbors
    - – Except for one causing update (split horizon rule)
  - » Neighbors use to update their tables

RIP Staleness / Oscillation Control

- Small Infinity
  - » Count to infinity doesn't take very long
- Route Timer
  - » Every route has timeout limit of 180 seconds
    - – Reached when haven't received update from next hop for 6 periods
  - » If not updated, set to infinity
  - » Soft-state refresh → important concept!!!
- Behavior
  - » When router or link fails, can take minutes to stabilize