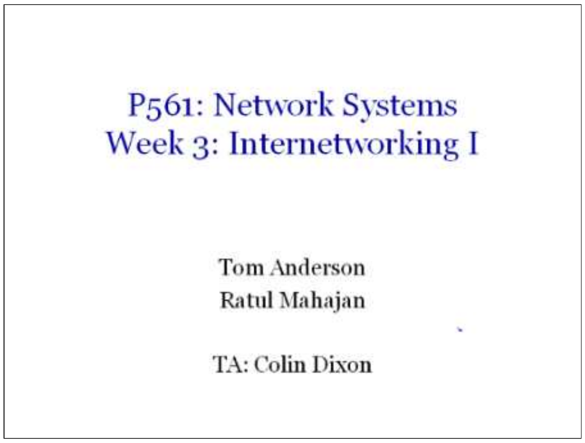
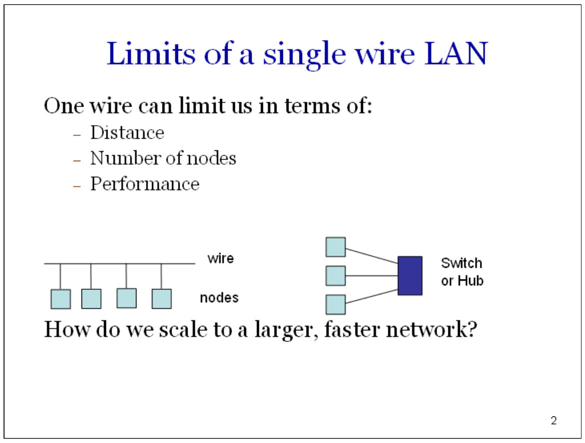


P561: Network Systems Week 3: Internetworking I

Tom Anderson
Ratul Mahajan

TA: Colin Dixon

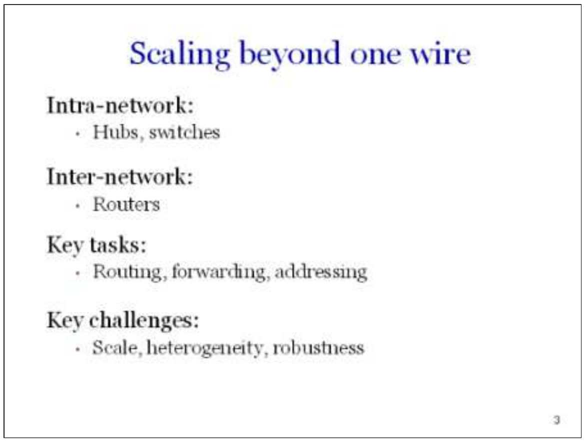


Limits of a single wire LAN

One wire can limit us in terms of:

- Distance
- Number of nodes
- Performance

How do we scale to a larger, faster network?



Scaling beyond one wire

Intra-network:

- Hubs, switches

Inter-network:

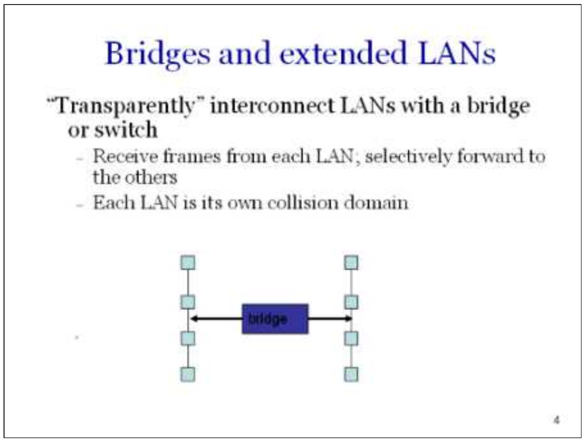
- Routers

Key tasks:

- Routing, forwarding, addressing

Key challenges:

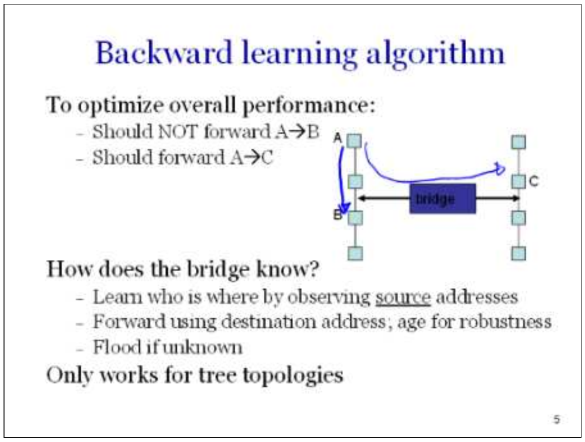
- Scale, heterogeneity, robustness



Bridges and extended LANs

"Transparently" interconnect LANs with a bridge or switch

- Receive frames from each LAN; selectively forward to the others
- Each LAN is its own collision domain



Backward learning algorithm

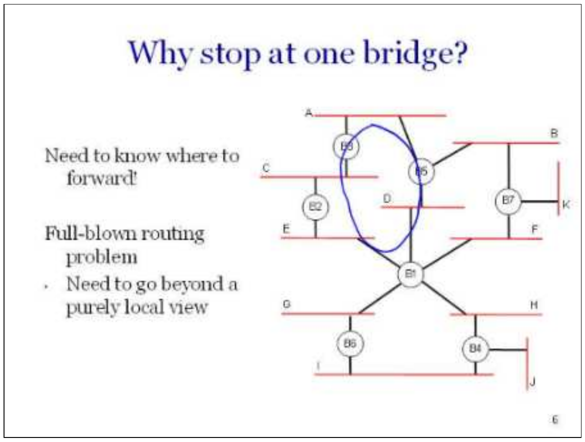
To optimize overall performance:

- Should NOT forward A→B
- Should forward A→C

How does the bridge know?

- Learn who is where by observing source addresses
- Forward using destination address; age for robustness
- Flood if unknown

Only works for tree topologies

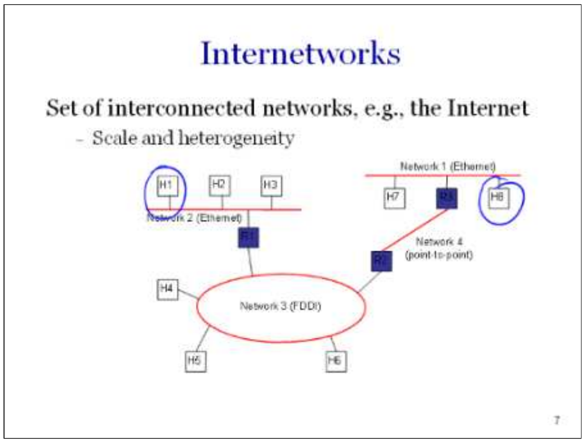


Why stop at one bridge?

Need to know where to forward!

Full-blown routing problem

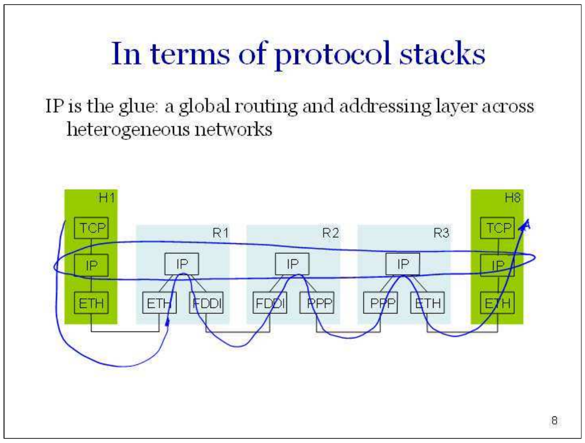
- Need to go beyond a purely local view



Internetworks

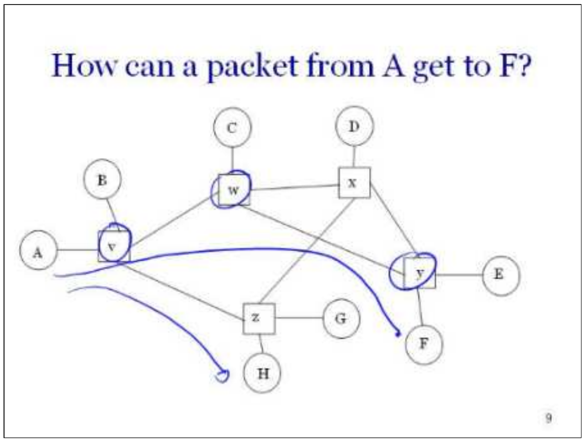
Set of interconnected networks, e.g., the Internet

- Scale and heterogeneity

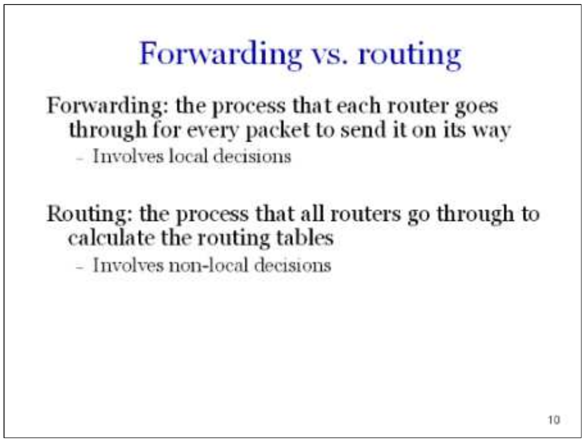


In terms of protocol stacks

IP is the glue: a global routing and addressing layer across heterogeneous networks



How can a packet from A get to F?



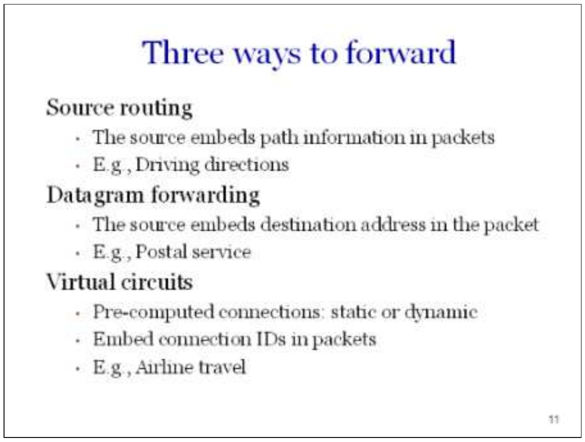
Forwarding vs. routing

Forwarding: the process that each router goes through for every packet to send it on its way

- Involves local decisions

Routing: the process that all routers go through to calculate the routing tables

- Involves non-local decisions



Three ways to forward

Source routing

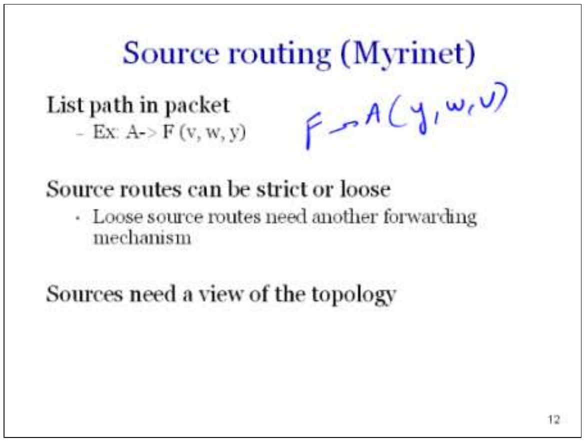
- The source embeds path information in packets
- E.g., Driving directions

Datagram forwarding

- The source embeds destination address in the packet
- E.g., Postal service

Virtual circuits

- Pre-computed connections: static or dynamic
- Embed connection IDs in packets
- E.g., Airline travel



Source routing (Myrinet)

List path in packet

- Ex: A-> F (v, w, y)

F -> A (y, w, v)

Source routes can be strict or loose

- Loose source routes need another forwarding mechanism

Sources need a view of the topology

Datagrams (Ethernet, IP)

Each packet has destination address

Each switch/router has forwarding table of destination -> next hop

- At v: F -> w
- At w: F-> y
- Forwarding decision made independently for each arriving packet

Distributed algorithm for calculating tables (routing)

Virtual circuits (ATM)

Each connection has destination address; each packet has virtual circuit ID (VCI)

Each switch has forwarding table of connection → next hop

- at connection setup, allocate virtual circuit ID (VCI) at each switch in path
- (input #, input VCI) -> (output #, output VCI)
  - At v: (A, 12) -> (w, 2)
  - At w: (v, 2) -> (y, 7)

Comparison of forwarding methods

	Src routing	Datagrams	Virtual circuits
Header size	worst	OK	best
Forwarding table size	none	# of hosts or networks	# of circuits
Forwarding overhead	best	Lookup	Lookup
Setup overhead	none	none	=~ datagram forwarding
Error recovery	Tell all sources	Tell all routers	Tear down circuit and reroute
QoS support	hard	hard	easier

Routing goals

Compute best path

- Defining "best" is slippery

Scale to billions of hosts

- Minimize control messages and routing table size

Quickly adapt to failures or changes

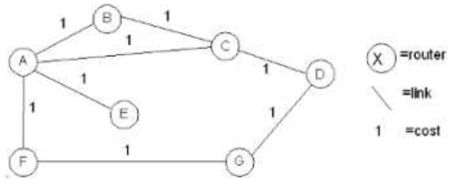
- Node and link failures, plus message loss

A network is a graph

Routing is essentially a problem in graph theory

- switches = nodes; links = edges; delay/hops = cost

Need dynamic computation to adapt to changes



Routing alternatives

Spanning Tree (Ethernet)

- Convert graph into a tree; route only along tree

Distance vector (RIP)

- exchange routing tables with neighbors
- no one knows complete topology

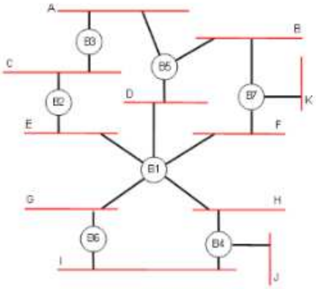
Link state (OSPF, IS-IS)

- send everyone your neighbors
- everyone computes shortest path

Spanning Tree Example

Convert graph into a tree; route only along the tree

Simple and avoids loops



Spanning tree algorithm overview

Distributed algorithm to compute spanning tree

- Robust against failures, needs no organization

Outline:

1. Elect a root node of the tree (lowest address)
2. Grow tree as shortest distances from the root (using lowest address to break distance ties)

Spanning tree algorithm in detail

Bridges periodically exchange config messages

- Contain: best root seen, distance to root, bridge address

Initially, each bridge thinks it is the root

- Each bridge tells its neighbors its address

On receiving a config message, update position in tree

- Pick smaller root address, then
- Shorter distance to root, then
- Bridge with smaller address

Periodically update neighbors

- Add one to distance to root, send downstream

Turn off forwarding on ports except those that send/receive "best"

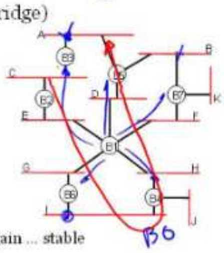
Algorithm Example

(B2, 0, B2)

Message format: (root, dist to root, bridge)

Messages sequence to and from B3:

1. B3 sends (B3, 0, B3) to B2 and B5
2. B3 receives (B2, 0, B2) and (B5, 0, B5) and accepts B2 as root
3. B3 sends (B2, 1, B3) to B5
4. B3 receives (B1, 1, B2) and (B1, 1, B5) and accepts B1 as root
5. B3 wants to send (B1, 2, B3) but doesn't as its nowhere "best"
6. B3 receives (B1, 1, B2) and (B1, 1, B5) again ... stable
7. Data forwarding is turned off to A



To bridge or not?

Yes:

- Simple (robust)
- No configuration required at end hosts or at bridges

No:

- Scalability
- Longer paths
- Minimal control

Research is fast eroding the difference with routing

- SmartBridge: A scalable bridge architecture, SIGCOMM 2000
- Floodless in SEATTLE: A scalable Ethernet architecture for large enterprises, SIGCOMM 2008

Distance vector routing

Each router periodically exchanges messages with neighbors

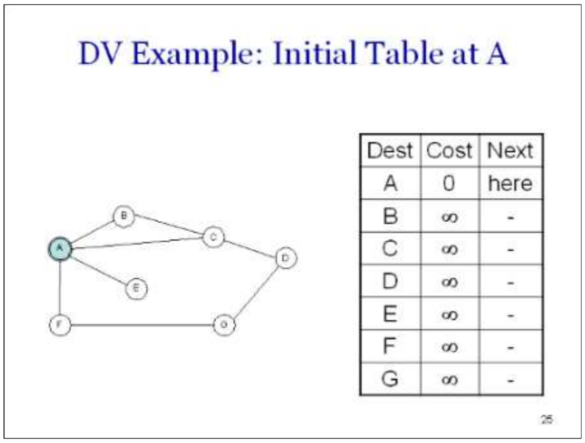
- best known distance to each destination ("distance vector")

Initially, can get to self with zero cost

On receipt of update from neighbor, for each destination

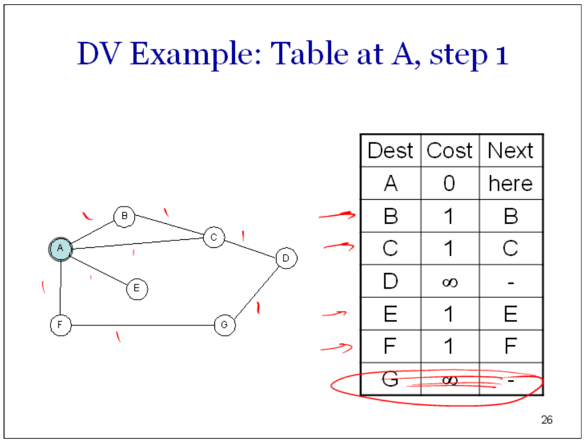
- switch forwarding tables to neighbor if it has cheaper route
- update best known distance
- tell neighbors of any changes

Absent topology changes, will converge to shortest path



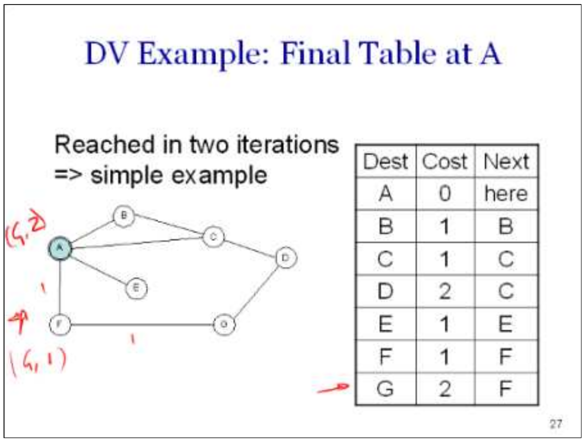
DV Example: Initial Table at A

Dest	Cost	Next
A	0	here
B	∞	-
C	∞	-
D	∞	-
E	∞	-
F	∞	-
G	∞	-



DV Example: Table at A, step 1

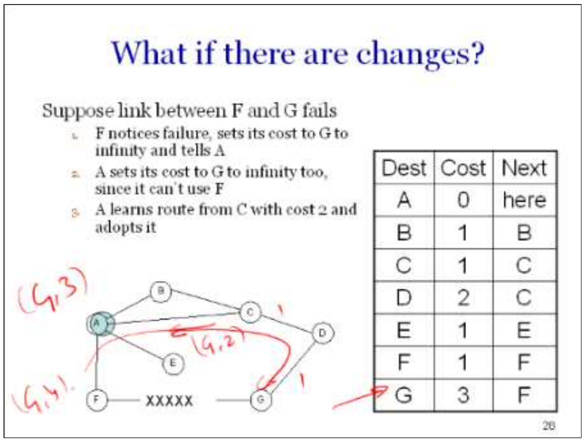
Dest	Cost	Next
A	0	here
B	1	B
C	1	C
D	∞	-
E	1	E
F	1	F
G	∞	-



DV Example: Final Table at A

Reached in two iterations
=> simple example

Dest	Cost	Next
A	0	here
B	1	B
C	1	C
D	2	C
E	1	E
F	1	F
G	2	F

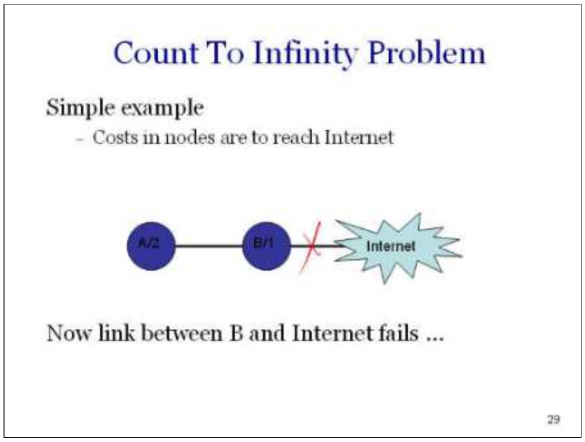


What if there are changes?

Suppose link between F and G fails

1. F notices failure, sets its cost to G to infinity and tells A
2. A sets its cost to G to infinity too, since it can't use F
3. A learns route from C with cost 2 and adopts it

Dest	Cost	Next
A	0	here
B	1	B
C	1	C
D	2	C
E	1	E
F	1	F
G	3	F

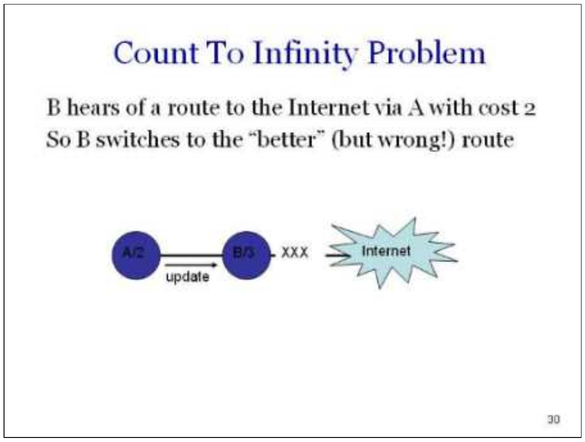


Count To Infinity Problem

Simple example

- Costs in nodes are to reach Internet

Now link between B and Internet fails …



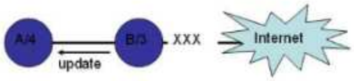
Count To Infinity Problem

B hears of a route to the Internet via A with cost 2

So B switches to the "better" (but wrong!) route

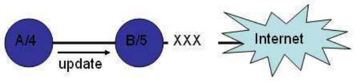
Count To Infinity Problem

A hears from B and increases its cost



Count To Infinity Problem

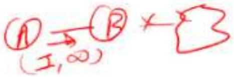
B hears from A and (surprise) increases its cost
Cycle continues and we "count to infinity"



Packets caught in a loop between A and B

Solutions to count to infinity

Lower infinity ☺



Split horizon

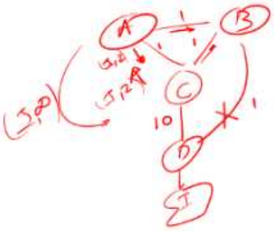
- Do not advertises the destination back to its next hop – that's where it learned it from!
- Solves trivial count-to-infinity problem

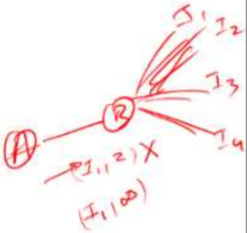
Poisoned reverse (RIP)

- Go farther: advertise infinity back to next hop

Question

Why does poisoned reverse bring additional benefit over split horizon?





Link state routing

Every router learns complete topology and then runs shortest-path

Two phases:

- Topology dissemination -- each node gets complete topology via reliable flooding
- Shortest-path calculation (Dijkstra's algorithm)

As long as every router uses the same information, will reach consistent tables

Topology flooding

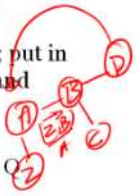
Each router identifies direct neighbors; put in numbered link state packets (LSPs) and periodically send to neighbors

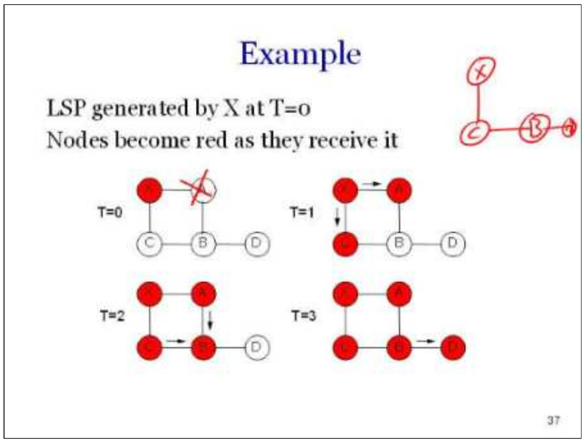
- LSPs contain [router, neighbors, costs]

If get a link state packet from neighbor Q

- drop if seen before
- else add to database and forward everywhere but Q

Each LSP will travel over the same link at most once in each direction

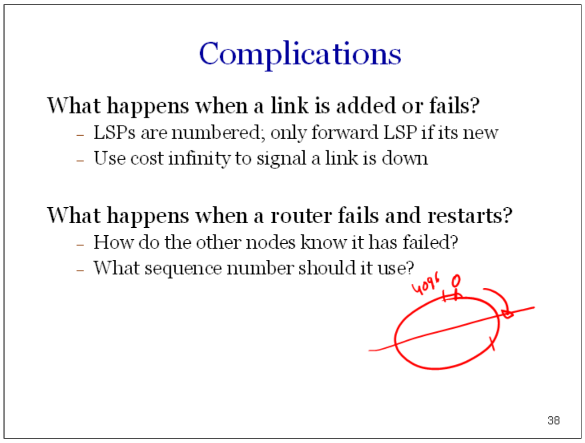




Example

LSP generated by X at T=0
Nodes become red as they receive it

37



Complications

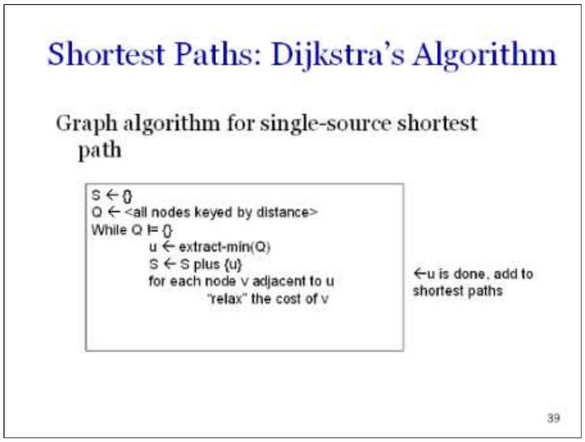
What happens when a link is added or fails?

- LSPs are numbered; only forward LSP if its new
- Use cost infinity to signal a link is down

What happens when a router fails and restarts?

- How do the other nodes know it has failed?
- What sequence number should it use?

38



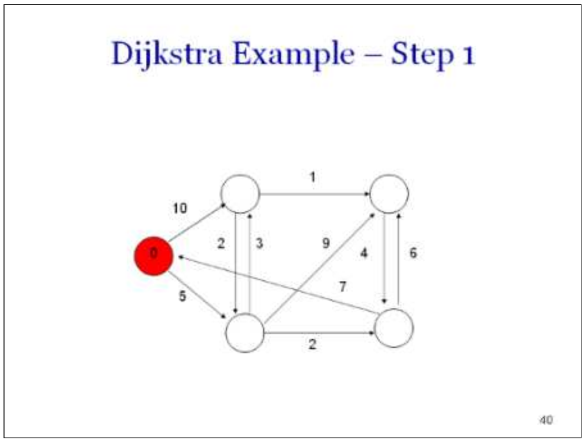
Shortest Paths: Dijkstra's Algorithm

Graph algorithm for single-source shortest path

```
S ← {}
Q ← <all nodes keyed by distance>
While Q != {}
        u ← extract-min(Q)
        S ← S plus {u}
        for each node v adjacent to u
                "relax" the cost of v
```

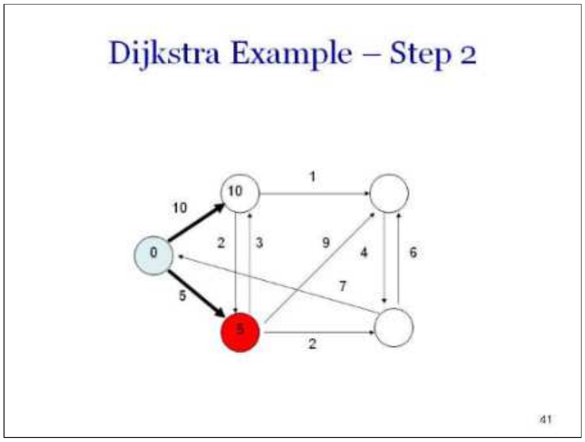
←u is done, add to shortest paths

39



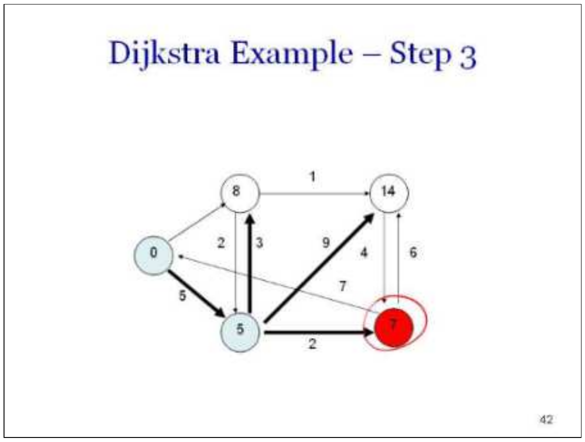
Dijkstra Example – Step 1

40



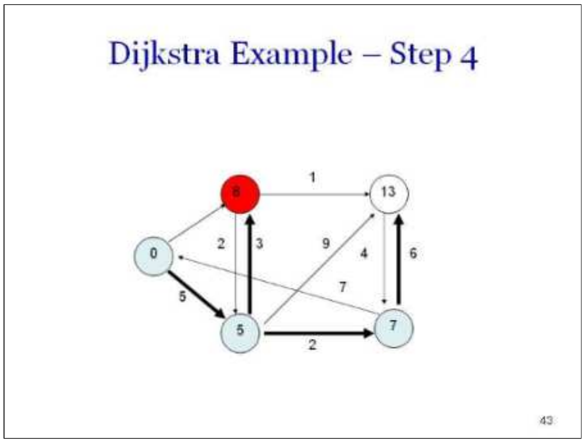
Dijkstra Example – Step 2

41

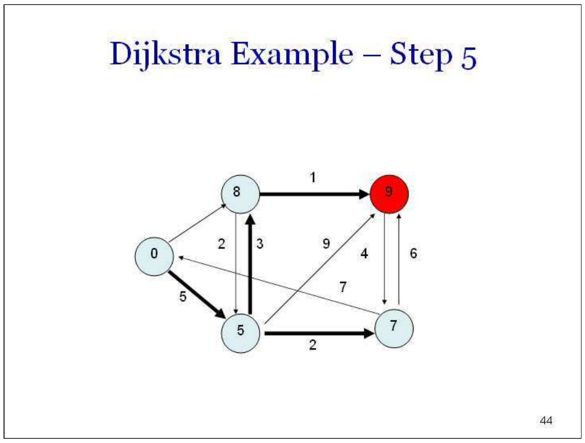


Dijkstra Example – Step 3

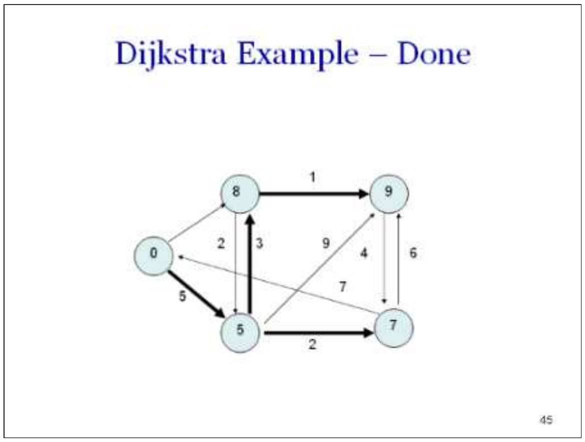
42



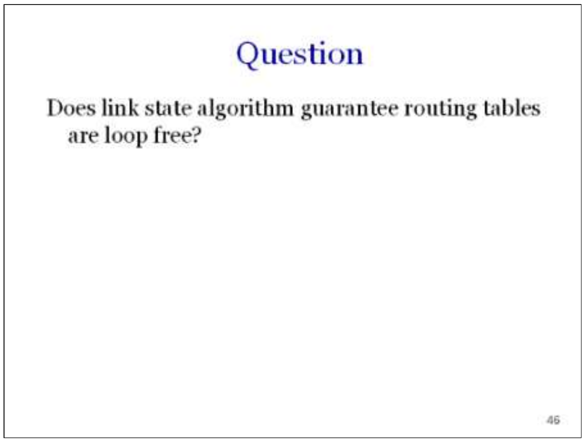
Dijkstra Example – Step 4



Dijkstra Example – Step 5

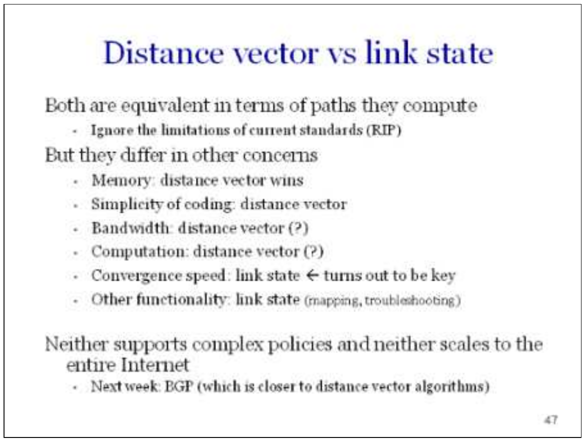


Dijkstra Example – Done



Question

Does link state algorithm guarantee routing tables are loop free?



Distance vector vs link state

Both are equivalent in terms of paths they compute

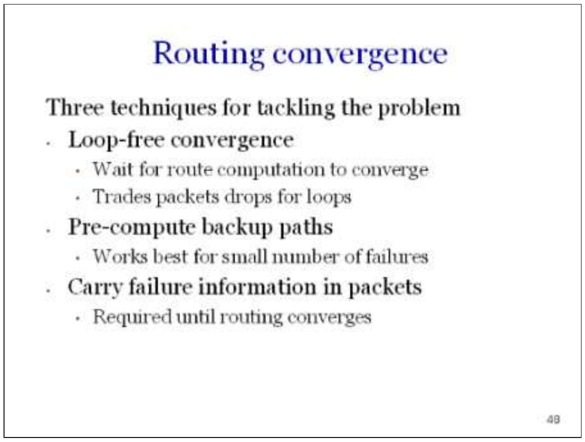
- Ignore the limitations of current standards (RIP)

But they differ in other concerns

- Memory: distance vector wins
- Simplicity of coding: distance vector
- Bandwidth: distance vector (?)
- Computation: distance vector (?)
- Convergence speed: link state ← turns out to be key
- Other functionality: link state (mapping, troubleshooting)

Neither supports complex policies and neither scales to the entire Internet

- Next week: BGP (which is closer to distance vector algorithms)

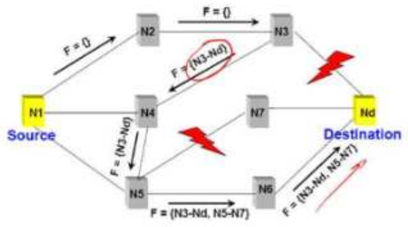


Routing convergence

Three techniques for tackling the problem

- Loop-free convergence
  - Wait for route computation to converge
  - Trades packets drops for loops
- Pre-compute backup paths
  - Works best for small number of failures
- Carry failure information in packets
  - Required until routing converges

Failure carrying packets



Route flapping

Constant churn in routes

- E.g., due to faulty equipment
- Can overload routers

Flap damping sometimes used

- Suppress frequent updates
- Slows convergence

Skeptics

- Spread bad news quickly, good news slowly

On Routing Cost Metrics

How should we choose cost?

- To get high bandwidth, low delay or low loss?
- Do costs depend on the load?

Static Metrics

- Unit cost? Treats OC48 same as ISDN
- Inverse bandwidth? Typical default
- Manually tweak to yield desired goal? ← state of art

Dynamic Metrics

- Depend on load; try to avoid hotspots (congestion)
- But can lead to oscillations (damping needed)

Internet Protocol (IP)

To connect diverse networks together

Service model:

- Best effort datagram forwarding

Addressing:

- Routing scalability
  - Each IP address has "network #" and "host #"
  - Routing uses network #
  - Immense pressure on scalability today
- Every host gets a globally reachable address
  - Oops: NATs (private host addresses)
  - Retrofitting: sub- and super-nets
  - Redesign: IPv6

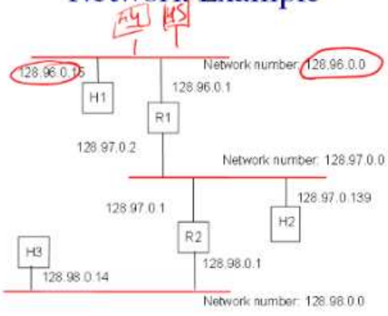
IPv4 Address Formats

		7	24
Class A	0	Network	Host
		14	16
Class B	1 0	Network	Host
		21	8
Class C	1 1 0	Network	Host
		27	
Class D	1 1 1 0	Multicast Group #	

32 bits written in "dotted quad" notation

- Example: 18.31.0.135

Network Example



Problems with IPv4 Addresses

Only 4B possible addresses
- 20B+ microprocessors fabricated in 2001

Rigid class structure makes it worse
- Internal fragmentation: cannot use all addresses
- Class B disproportionately popular (only ~16K nets)

Router tables still too large
- 2M class C networks!
- Need better aggregation

Flexible IP Address Allocation

Subnets
- split net addresses between multiple sites

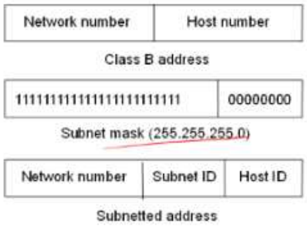
Supernets
- assign adjacent net addresses to same org
- classless routing (CIDR)
  - combine routing table entries whenever all nodes with same prefix share same hop

Subnetting – More Hierarchy

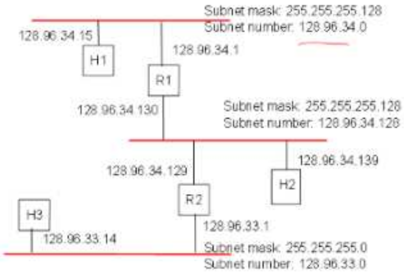
Split one network # into multiple physical networks

Internal structure isn't propagated

Helps allocation efficiency



Subnet Example


Subnet mask: 255.255.255.128
Subnet number: 128.96.34.0
128.96.34.15
H1
128.96.34.1
R1
128.96.34.130
Subnet mask: 255.255.255.128
Subnet number: 128.96.34.128
128.96.34.129
128.96.34.139
H2
R2
H3
128.96.33.1
128.96.33.14
Subnet mask: 255.255.255.0
Subnet number: 128.96.33.0

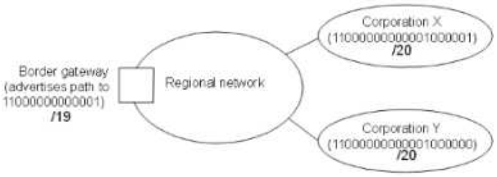
CIDR (Supernetting)

CIDR = Classless Inter-Domain Routing

Aggregate adjacent advertised network routes
- Ex: ISP has class C addresses 192.4.16 through 192.4.31
- Really like one larger 20 bit address class ...
- Advertise as such (network number, prefix length)
- Reduces size of routing tables

CIDR Example

X and Y routes can be aggregated because they form a bigger contiguous range.



IP Forwarding Revisited

IP address still has network #, host #

- With class A/B/C, split was obvious from first few bits
- Now split varies as you traverse the network!

Routing table contains variable length "prefixes"

- IP address and length indicating what bits are fixed
- Next hop to use for each prefix

To find the next hop:

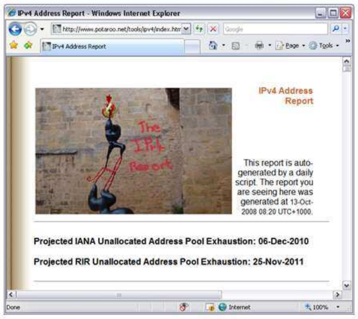
- There can be multiple matches
- Take the longest matching prefix

IPAddr Pfx

3.0.0.0/8
3.1.0.0/16

3.1.1.1
3.0.0.1

The sky is falling!



IPv4 Address Report - Windows Internet Explorer

http://www.potaroo.net/tools/ipv4/index.htm

IPv4 Address Report

This report is auto-generated by a daily script. The report you are seeing here was generated at 13-Oct-2008 08:20 UTC+1000.

Projected IANA Unallocated Address Pool Exhaustion: 06-Dec-2010

Projected RIR Unallocated Address Pool Exhaustion: 25-Nov-2011

IPv6 addressing

16 byte addresses (4x IPv4)

- 1.5K per sq. foot of earth's surface
- Written in hexadecimal as 8 groups of 2-bytes
  - E.g., 1234:5678:9abc:def1:2345:6789:abcd

Prefix	Use
00…0 (128 bits)	Unspecified
00…1 (128 bits)	Loopback
1111 1111	Multicast
1111 1110 10	Link local unicast
1111 1110 11	Site local unicast
Everything else	Global unicast

IPv6 vs. IPv4

Pretty similar overall

Except that the address length of v6 offers some unique flexibilities

- Stateless autoconfiguration of hosts (in a few slides)
- Deeper hierarchy and more efficient aggregation (e.g., geographical)

Two ways to map an IPv4 address to IPv6

Network Address Translators (NATs)

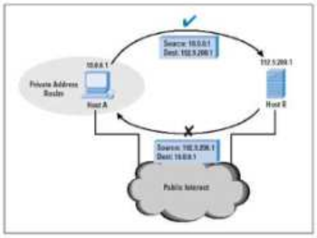
Middle-boxes that change IP addresses or ports for packets that traverse network edge

Original goal: enable internal hosts to use private addresses while still being able to communicate with external hosts

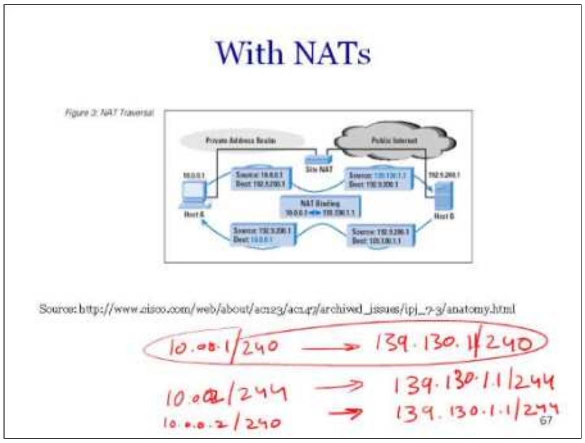
Side-effect: Limit allowed communication patterns

Without NATs

Figure 2: Public/Private Communication



Source: http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_7-3/anatomy.html



With NATs

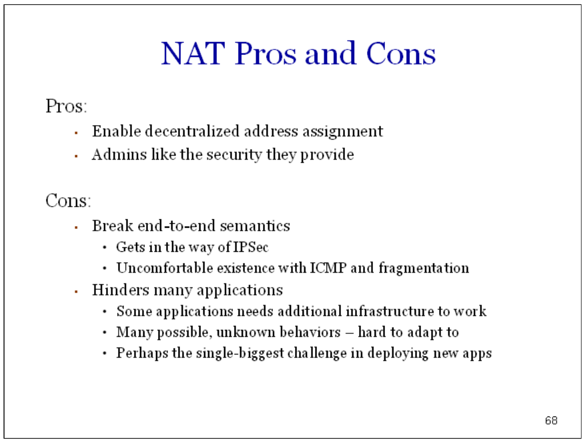
Figure 3: NAT Traversal

Source: http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_7-3/anatomy.html

10.0.0.1/240 → 139.130.1.1/240

10.0.0.2/244 → 139.130.1.1/244

10.0.0.2/240 → 139.130.1.1/244



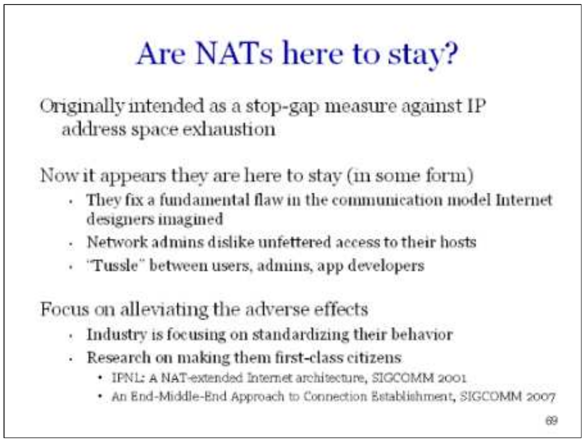
NAT Pros and Cons

Pros:

- Enable decentralized address assignment
- Admins like the security they provide

Cons:

- Break end-to-end semantics
  - Gets in the way of IPSec
  - Uncomfortable existence with ICMP and fragmentation
- Hinders many applications
  - Some applications needs additional infrastructure to work
  - Many possible, unknown behaviors – hard to adapt to
  - Perhaps the single-biggest challenge in deploying new apps



Are NATs here to stay?

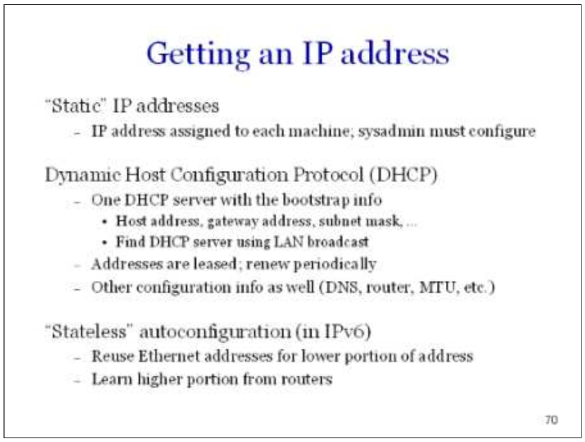
Originally intended as a stop-gap measure against IP address space exhaustion

Now it appears they are here to stay (in some form)

- They fix a fundamental flaw in the communication model Internet designers imagined
- Network admins dislike unfettered access to their hosts
- "Tussle" between users, admins, app developers

Focus on alleviating the adverse effects

- Industry is focusing on standardizing their behavior
- Research on making them first-class citizens
  - IPNL: A NAT-extended Internet architecture, SIGCOMM 2001
  - An End-Middle-End Approach to Connection Establishment, SIGCOMM 2007



Getting an IP address

"Static" IP addresses

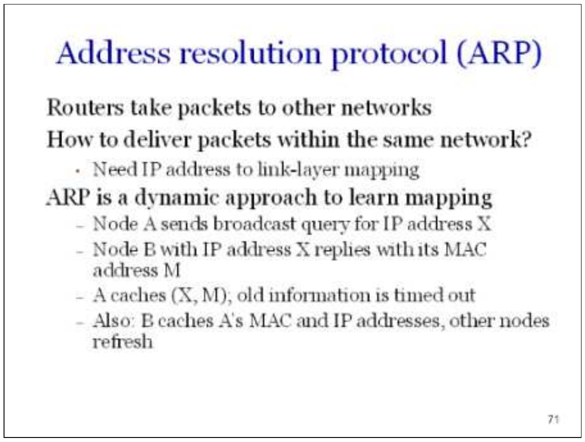
- IP address assigned to each machine; sysadmin must configure

Dynamic Host Configuration Protocol (DHCP)

- One DHCP server with the bootstrap info
  - Host address, gateway address, subnet mask, …
  - Find DHCP server using LAN broadcast
- Addresses are leased; renew periodically
- Other configuration info as well (DNS, router, MTU, etc.)

"Stateless" autoconfiguration (in IPv6)

- Reuse Ethernet addresses for lower portion of address
- Learn higher portion from routers



Address resolution protocol (ARP)

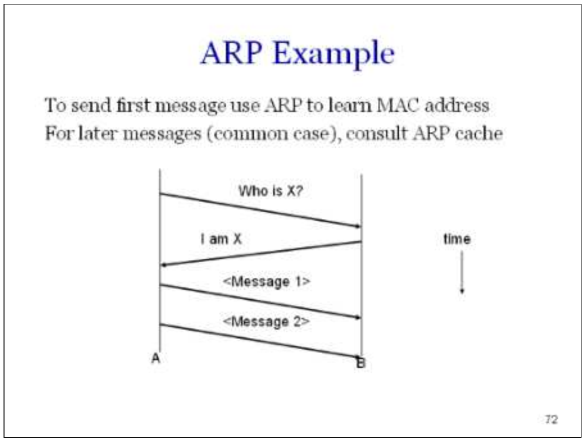
Routers take packets to other networks

How to deliver packets within the same network?

- Need IP address to link-layer mapping

ARP is a dynamic approach to learn mapping

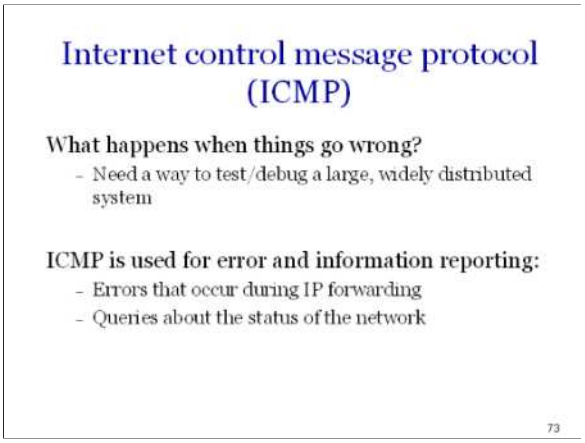
- Node A sends broadcast query for IP address X
- Node B with IP address X replies with its MAC address M
- A caches (X, M); old information is timed out
- Also: B caches A's MAC and IP addresses, other nodes refresh



ARP Example

To send first message use ARP to learn MAC address

For later messages (common case), consult ARP cache



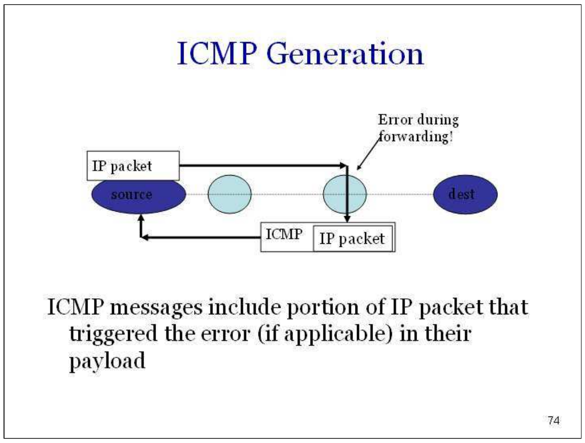
Internet control message protocol (ICMP)

What happens when things go wrong?

- Need a way to test/debug a large, widely distributed system

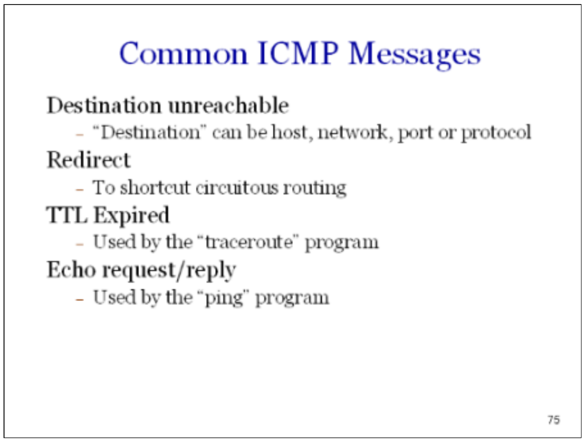
ICMP is used for error and information reporting:

- Errors that occur during IP forwarding
- Queries about the status of the network



ICMP Generation

ICMP messages include portion of IP packet that triggered the error (if applicable) in their payload



Common ICMP Messages

Destination unreachable

- "Destination" can be host, network, port or protocol

Redirect

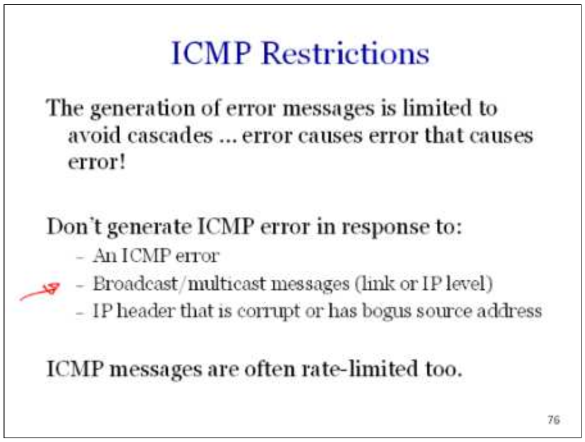
- To shortcut circuitous routing

TTL Expired

- Used by the "traceroute" program

Echo request/reply

- Used by the "ping" program



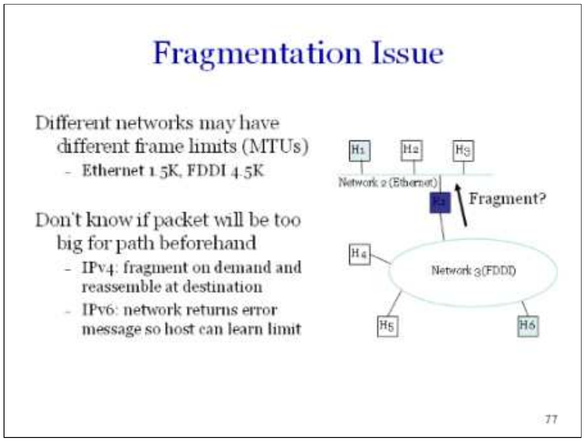
ICMP Restrictions

The generation of error messages is limited to avoid cascades … error causes error that causes error!

Don't generate ICMP error in response to:

- An ICMP error
- Broadcast/multicast messages (link or IP level)
- IP header that is corrupt or has bogus source address

ICMP messages are often rate-limited too.



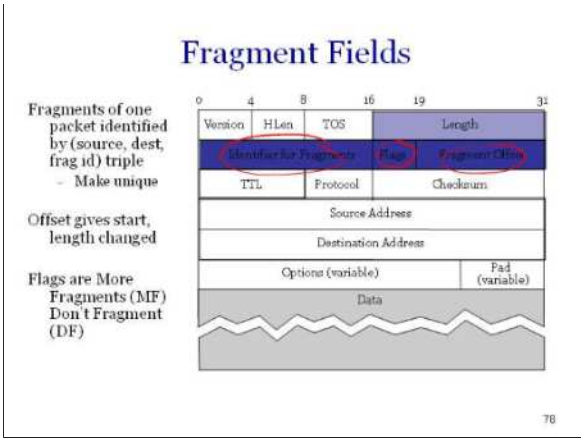
Fragmentation Issue

Different networks may have different frame limits (MTUs)

- Ethernet 1.5K, FDDI 4.5K

Don't know if packet will be too big for path beforehand

- IPv4: fragment on demand and reassemble at destination
- IPv6: network returns error message so host can learn limit



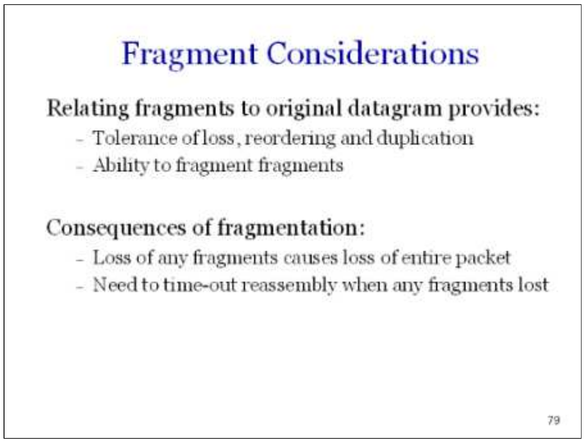
Fragment Fields

Fragments of one packet identified by (source, dest, frag id) triple

- Make unique

Offset gives start, length changed

Flags are More Fragments (MF) Don't Fragment (DF)



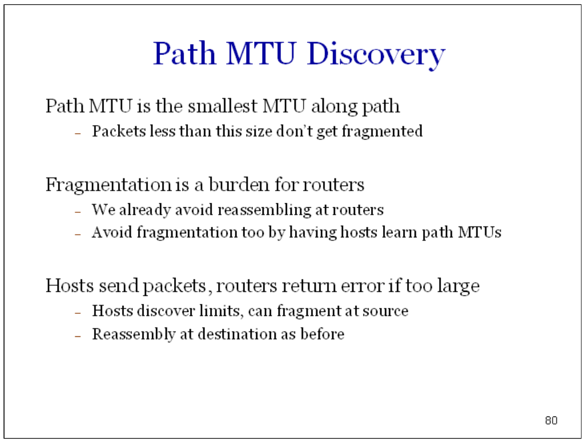
Fragment Considerations

Relating fragments to original datagram provides:

- Tolerance of loss, reordering and duplication
- Ability to fragment fragments

Consequences of fragmentation:

- Loss of any fragments causes loss of entire packet
- Need to time-out reassembly when any fragments lost



Path MTU Discovery

Path MTU is the smallest MTU along path

- Packets less than this size don't get fragmented

Fragmentation is a burden for routers

- We already avoid reassembling at routers
- Avoid fragmentation too by having hosts learn path MTUs

Hosts send packets, routers return error if too large

- Hosts discover limits, can fragment at source
- Reassembly at destination as before