Chapter 4
NETWORK LAYER

Distributed
Computing
Group

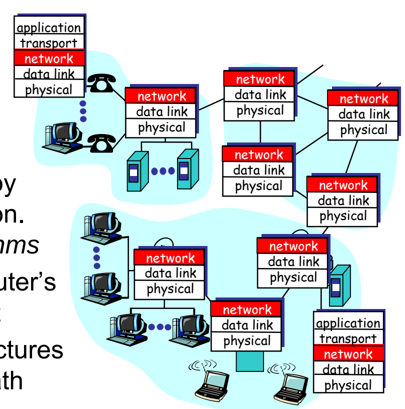
Computer Networks
Summer 2005

Overview

- Network layer services
- Routing principle: path selection
- Hierarchical routing, scalability
- IP, the Internet Protocol
  - Internet routing protocols reliable transfer
  - Intra-domain
  - Inter-domain
  - Routing convergence
- What's inside a router?
- Advanced Topics
  - IPv6

Network layer functions

- Transport packet from sending to receiving hosts
- Network layer protocols in *every* host, router



Three important functions:

- *path determination:* route taken by packets from source to destination. There are various *routing algorithms*
- *switching:* move packets from router's input to appropriate router output
- *call setup:* some network architectures require router call setup along path before data flows

Network service model

Q: What *service model* for "channel" transporting packets from sender to receiver?

service abstraction

- guaranteed bandwidth?
- preservation of inter-packet timing (no jitter)?
- loss-free delivery?
- in-order delivery?
- congestion feedback to sender?

The most important abstraction provided by network layer:

? virtual circuit ? or ? datagram?

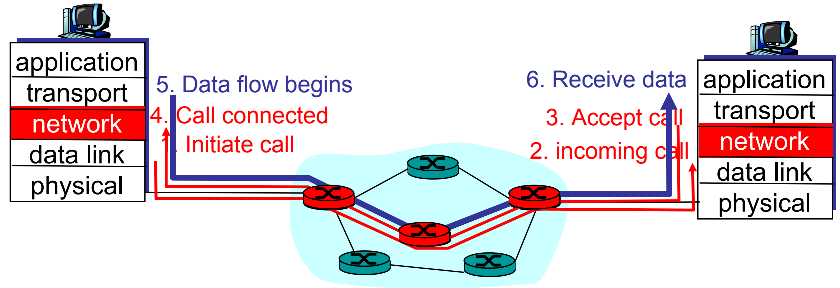
Virtual circuits

"source-to-destination path behaves much like telephone circuit"
- performance-wise
- network actions along source-to-destination path

- call setup, teardown for each call *before* data can flow
- each packet carries VC identifier (not destination host ID)
- *every* router on source-dest path maintains "state" for each passing connection
  - transport-layer connection only involved two end systems
- link, router resources (bandwidth, buffers) may be *allocated* to VC
  - to get circuit-like performance

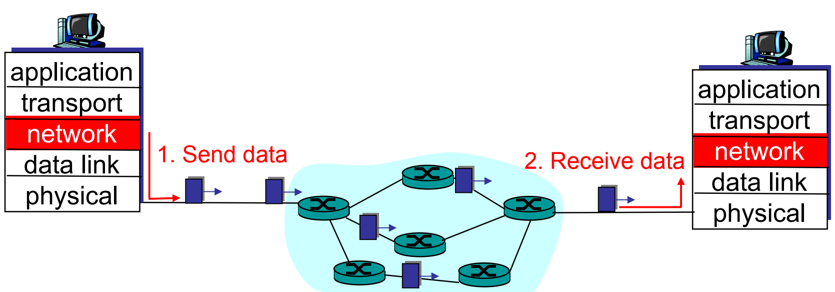
Virtual circuits: signaling protocols

- used to setup, maintain, and teardown VC
- used in ATM, frame-relay, X.25
- not used in today's Internet



Datagram networks: The Internet model

- no call setup at network layer
- routers: no state about end-to-end connections
  - no network-level concept of "connection"
- packets typically routed using destination host ID
  - packets between same source-dest pair may take different paths



Network layer service models

Network Architecture	Service Model	Guarantees ? Bandwidth	Loss	Order	Timing	Congestion feedback
Internet	best effort	none	no	no	no	no (inferred via loss)
ATM	CBR	constant rate	yes	yes	yes	no congestion
ATM	VBR	guaranteed rate	yes	yes	yes	no congestion
ATM	ABR	guaranteed minimum	no	yes	no	yes
ATM	UBR	none	no	yes	no	no

Datagram or VC network: why?

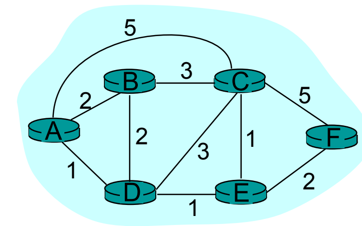
Internet
- data exchange among computers
  - "elastic" service, no strict timing req.
- "smart" end systems (computers)
  - can adapt, perform control, error recovery
  - simple inside network, complexity at "edge"
- many link types
  - different characteristics
  - uniform service difficult

ATM
- evolved from telephony
- human conversation
  - strict timing, reliability requirements
  - need for guaranteed service
- "dumb" end systems
  - telephones
  - complexity inside network

Routing

Routing protocol

Goal: determine "good" path (sequence of routers) through network from source to dest.



Graph abstraction for routing
- graph nodes are routers
- graph edges are physical links
  - link cost: delay, $ cost, or congestion level

"good" path:
- typically means minimum cost path
- other definitions possible

Routing Algorithm classification

Global or decentralized?

Global
- all routers have complete topology, link cost info
- "link state" algorithms

Decentralized
- router knows physically-connected neighbors, link costs to neighbors
- iterative process of computation, exchange of info with neighbors
- "distance vector" algorithms

Static or dynamic?

Static
- routes change slowly over time

Dynamic
- routes change more quickly
  - periodic update
  - in response to link cost changes

Single Source Shortest Paths

- For a given source we want the shortest path to all other nodes
- Optimality principle
  - For each shortest path $p = (v_0, v_1, \ldots, v_k)$, each subpath $p' = (v_i, \ldots, v_j)$ is also a shortest path. If this wasn't the case then there was a shorter path p'' from v_i to v_j which one could use to shortcut path p.
- For a given source s and a node v this means
  - There is a node u such that $c(sp(s,u)) + c(u,v) = c(sp(s,v))$, that is, in general $c(sp(s,u')) + c(u',v) \geq c(sp(s,v))$.
- The single source shortest path problem results in a tree

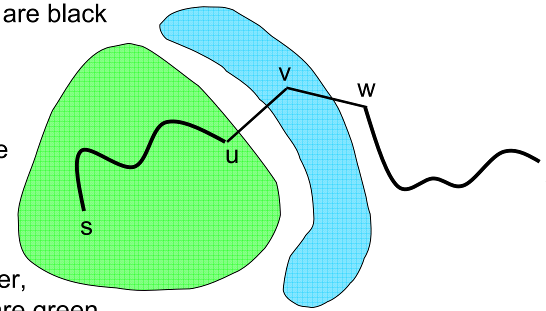
Single source shortest path: Intuition

- "Once upon a time, the Chinese Emperor wanted to know the distance and the best routes from Beijing to all the major cities in his country."
- "At the first day of the summer, a few scouts started in Beijing, taking all the roads leaving Beijing."
- "Whenever a scout arrives first in a city, he notes the current time and the path he took, and then immediately recruits new scouts that leave the city, taking all the possible roads and trails. Then he returns to Beijing."
- "Whenever a scout arrives second (or later) in a city, he does nothing and returns to Beijing."

- This "algorithm" solves the single source shortest path problem… How can one prove that it is correct? How efficient is the algorithm?

Algorithm idea

- There are 3 groups of nodes in the network
  - To the green nodes we know the shortest path
  - The blue nodes are directly reachable from the green nodes
  - All other nodes are black
- Idea
  - Start with source s as the only green node
  - Color the best* blue node green, one after another, until all nodes are green
    (*best = minimum distance to source s of all blue nodes)



A Link-State Routing Algorithm

Dijkstra's algorithm

- net topology, link costs known to all nodes
  - accomplished via "link state broadcast"
  - all nodes have same info
- computes single-source shortest path tree
  - gives routing table for source

Notation

- c(i,j): link cost from node i to j. Can be infinite if not direct neighbors, costs define adjacency matrix
- v.distance: current value of cost of path from source s to destination v
- v.visited: boolean variable that determines if optimal path to v was found
- v.pred: the predecessor node of v in the routing tree
- B: the set of blue nodes

Dijkstra's Algorithm (for source s and edge costs c)

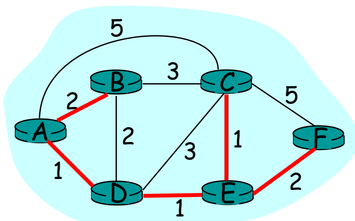
```
s.visited := true; s.distance := 0; s.pred := s; // init source s
for all nodes v ∈ V \ s do // init all other nodes
    v.visited := false; v.distance := ∞; v.pred := undefined;

B := {} // B is the set of blue nodes, initially all neighbors of s
for all nodes v ∈ V \ s that are direct neighbors of s
    B := B + {v}; v.distance := c(s,v); v.pred := s;

while B not empty do // always choose the best blue node v
    v := node in B with minimum v.distance;
    B := B – {v};
    v.visited := true;
    for all neighbors w of v with w.visited = false; // update neighbors of v
        if w not in B then
            B := B + {w}; w.distance := v.distance+c(v,w); w.pred := v;
        if w ∈ B then
            if (v.distance+c(v,w) < w.distance) then
                w.distance := v.distance+c(v,w); w.pred := v;
endwhile
```

Dijkstra’s algorithm: example

Step	visited	Set of blue nodes B (with distance)
0	A	D (1), B (2), C (5)
1	A, D (1)	E (2), B (2), C (4)
2	AD, E (2)	B (2), C (3), F(4)
3	ADE, B (2)	C (3), F(4)
4	ADEB, C (3)	F(4)
5	ADEBC, F (4)	-



Dijkstra’s algorithm, algorithm complexity

- n nodes, m (directed) edges
- Initialization costs O(n) operations
- Each round in the loop visits one unvisited node, that is, there are exactly n-1 rounds.
- In each round you have to find and remove the minimum node distance node v, and update the neighbors of node v.
- You can do both steps in O(n) time, thus $O(n^2)$ total time.

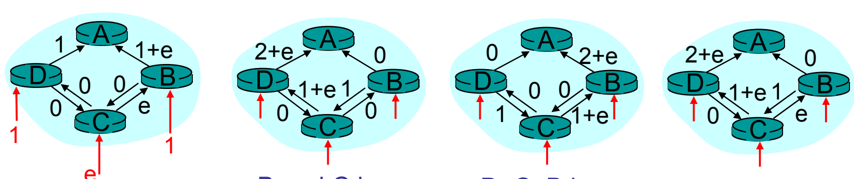
- Remark 1: With a Fibonacci-Heap, one can implement the whole algorithm in O(m + n log n) time.

- Remark 2: Some books claim that the algorithm complexity is O(n log n), which is clearly bogus since at least all the edges have to be examined…

Dijkstra’s algorithm, correctness

Oscillations possible

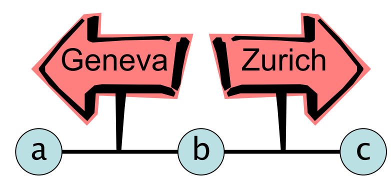
- For example if link costs depend on the amount of carried traffic. Example: three flows to node A, with traffic 1, 1, and e (<1)



initially | B and C have better routes | D, C, B have better routes | etc.

- How would you prove that Dijkstra’s algorithm is optimal for constant (and positive!) link costs? (Not in this course.)

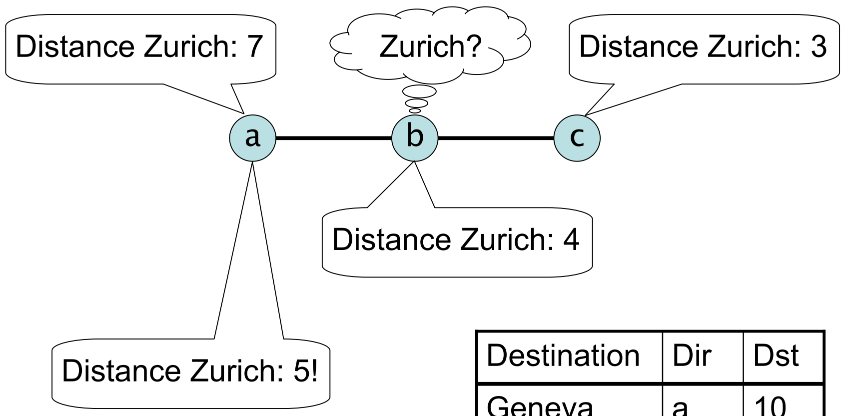
Distance Vector Routing: Intuition



Routing Table of b

Destination	Dir
Geneva	a
Zurich	c

Distance Vector Routing



Destination	Dir	Dst
Geneva	a	10
Zurich	c	4

Distance Vector Routing Algorithm

Algorithm is iterative

- continues until no nodes exchange info
- *self-terminating*: no "signal" to stop

asynchronous

- nodes need *not* to iterate in lock-step

distributed

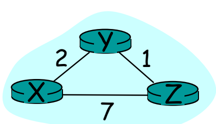
- each node communicates only with direct neighbors

Routing Table with distance info

- each node has one
- a node x has for each neighbor z an entry for each destination y (as in example before); $D^x(y,z)$ = distance from x to y through z
- the best route for a given destination is marked

$$\boxed{\begin{aligned} D^x(y) &= \min_z D^x(y,z) \\ D^x(y,z) &= c(x,z) + D^z(y) \end{aligned}}$$

Distance Vector Algorithm: example



D^X (dest)	cost via Y	cost via Z
Y	2	∞
Z	∞	7

D^Y (dest)	cost via X	cost via Z
X	2	∞
Z	∞	1

D^Z (dest)	cost via X	cost via Y
X	7	∞
Y	∞	1

D^X (dest)	cost via Y	cost via Z
Y	2	8
Z	3	7

D^Y (dest)	cost via X	cost via Z
X	2	8
Z	9	1

D^Z (dest)	cost via X	cost via Y
X	7	3
Y	9	1

D^X (dest)	cost via Y	cost via Z
Y		
Z		

D^Y (dest)	cost via X	cost via Z
X		
Z		

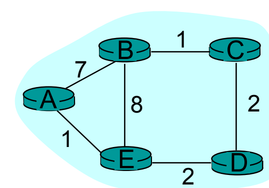
D^Z (dest)	cost via X	cost via Y
X		
Y		

Distance table gives routing table

cost to destination via

$D^E()$ (destination)	A	B	D
A	1	14	5
B	7	8	5
C	6	9	4
D	4	11	2

destination	Outgoing link to use, cost
A	A,1
B	D,5
C	D,4
D	D,2



Distance table → Routing table

Distance Vector Routing

Local iteration caused by
- local link cost change
- Neighbor sends a message saying that (at least) one of its least cost paths changed

Algorithm is distributed
- each node notifies neighbors *only* when its least cost path to any destination changes
  - neighbors then notify their neighbors if necessary, etc.

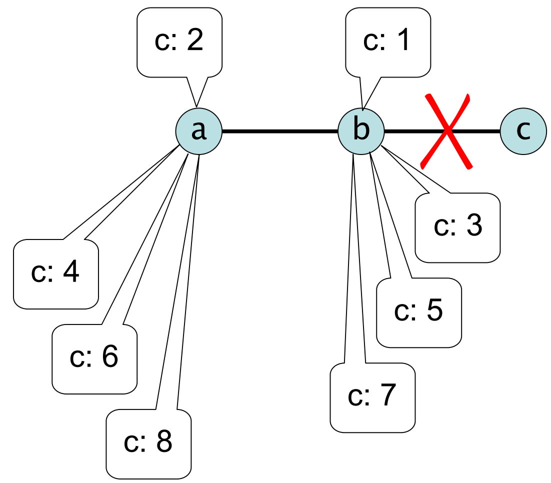
Each node executes a loop:

wait for (change in local link cost or msg from neighbor)

recompute distance table

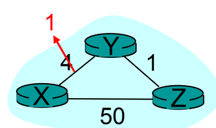
if least cost path to any dest has changed, *notify* all neighbors

Count to Infinity Problem

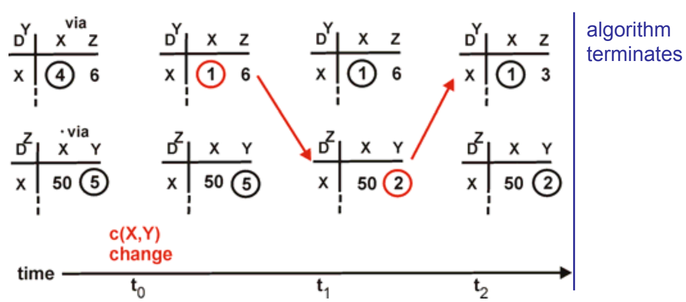


Distance Vector: link cost changes

Link cost changes
- node detects local link cost change
- updates distance table
- if cost change in least cost path, notify neighbors



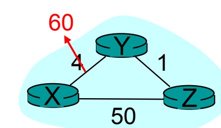
“good news travel fast”

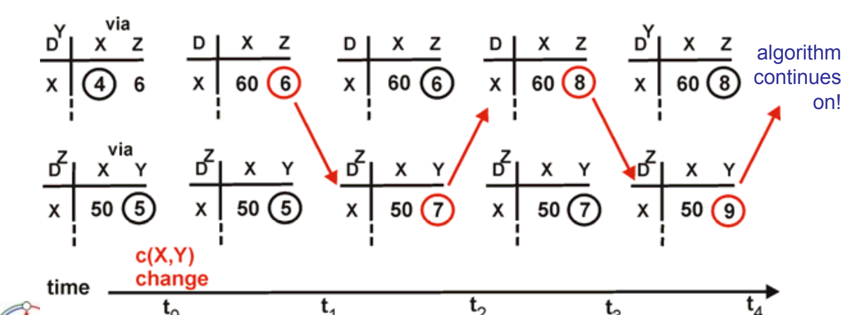


Distance Vector: link cost changes

- What if the cost of a link grows?
- Compare with the count to infinity problem

(More on this later)





Link-State vs. Distance-Vector Routing Algorithms

Message complexity

- LS: with n nodes, m links, network flooded with O(nm) messages
- DV: exchange between neighbors only
  - convergence time varies

Speed of Convergence

- LS: O(m + n log n)
  - may have oscillations
- DV: convergence time varies
  - count-to-infinity problem (later more)

Robustness

- what happens if router malfunctions?

LS:

- node can advertise incorrect *link* cost
- each node computes only its *own* table

DV:

- DV node can advertise incorrect *path* cost
- each node's table used by others → errors propagate thru network

Hierarchical Routing

So far we studied idealization

- all routers identical, "flat" graph

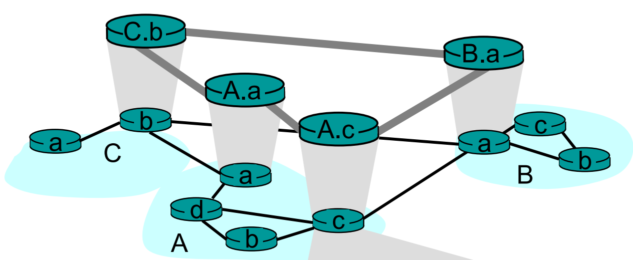
Reality

- Internet is network of networks
- Each network admin may want to control routing in own network
- You cannot store 200 million destinations in (all) routing tables; routing table exchange too massive…

Idea

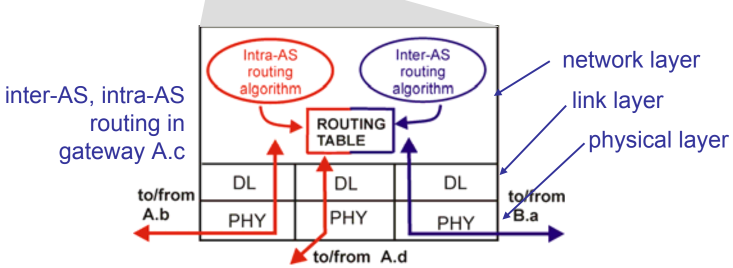
- aggregate routers into groups, "autonomous systems" (AS)
- routers in same AS run same routing protocol
  - "intra-AS" routing protocol
  - routers in a different AS can run a different intra-AS routing protocol
- Special gateway routers in AS's
  - run intra-AS routing protocol with all other routers in AS
  - run *inter-AS routing* protocol with other gateway routers

Intra-AS and Inter-AS routing

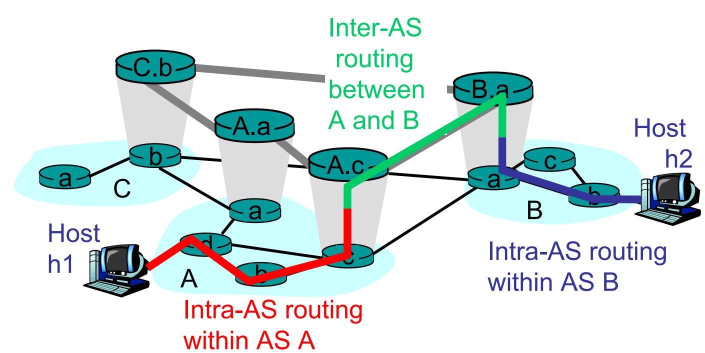


Gateways:

- perform inter-AS routing amongst themselves
- perform intra-AS routers with other routers in their AS



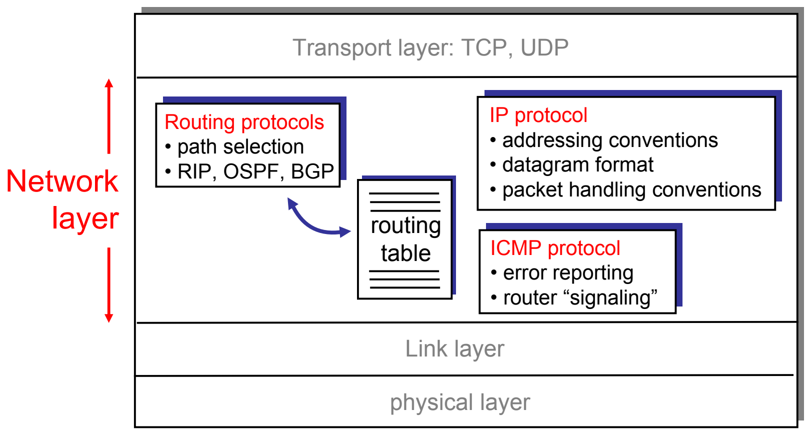
Intra-AS and Inter-AS routing



- We'll examine specific inter-AS and intra-AS Internet routing protocols shortly

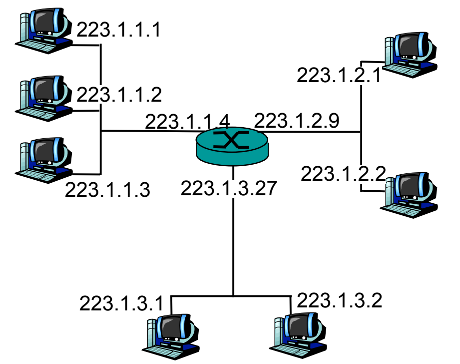
The Internet Network Layer

Host, router network layer functions:



IP Addressing: Introduction

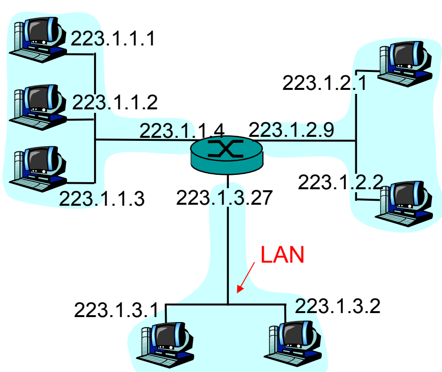
- IP address: 32-bit identifier for host, router *interface*
- Interface: connection between host, router and physical link
  - routers typically have multiple interfaces
  - host may have multiple interfaces
  - IP addresses associated with interface, not host or router



$$223.1.1.1 = \underbrace{11011111}_{223}\ \underbrace{00000001}_{1}\ \underbrace{00000001}_{1}\ \underbrace{00000001}_{1}$$

IP Addressing

- IP address
  - network part (high order bits)
  - host part (low order bits)
- What's a (local) network? (from IP address perspective)
  - device interfaces with same network part of IP address
  - can physically reach each other without intervening router



network consisting of 3 IP networks
(for IP addresses starting with 223,
first 24 bits are network address)

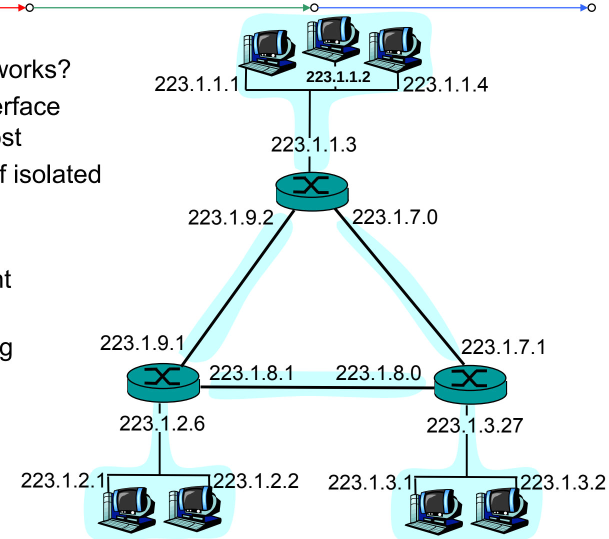
IP Addressing

How to find the networks?

- Detach each interface from router or host
- create "islands of isolated networks"

Example on the right

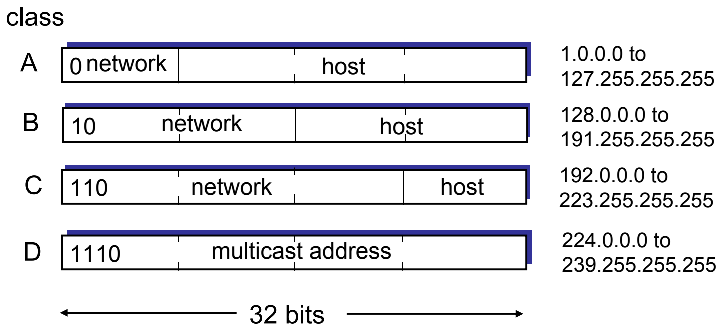
- Interconnected system consisting of six networks



IP Addresses

given notion of "network", let's re-examine IP addresses

"class-full" addressing



IP addressing: CIDR

- class-full addressing:
  - inefficient use of address space, address space exhaustion
  - e.g., class B net allocated enough addresses for 65K hosts, even if only 2K hosts in that network
- CIDR: Classless InterDomain Routing
  - network portion of address of arbitrary length
  - address format: a.b.c.d/x, where x is number of bits in network portion of address

200.23.16.0/23

IP addresses: how to get one?

How do hosts get one? (host portion)

- Either hard-coded by system admin in a file
  - Wintel: control-panel→network→configuration →tcp/ip→properties
  - UNIX: /etc/rc.config
- Or DHCP: Dynamic Host Configuration Protocol
  - dynamically get address: "plug-and-play"
  - host broadcasts "DHCP discover" message
  - DHCP server responds with "DHCP offer" message
  - host requests IP address: "DHCP request" message
  - DHCP server sends address: "DHCP ack" message

IP addresses: how to get one?

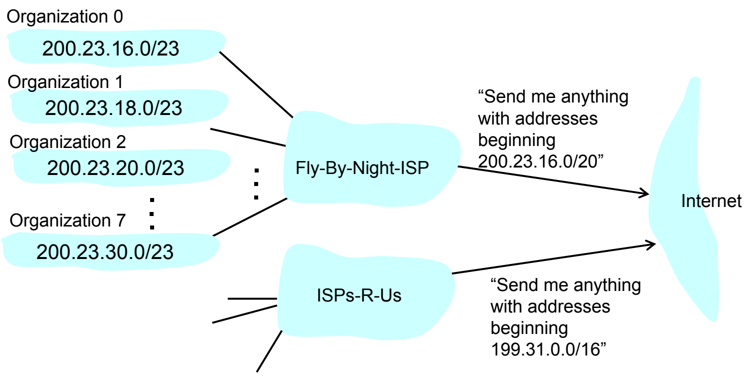
Network (network portion)

- get allocated portion of ISP's address space

ISP's block	11001000 00010111 00010000	00000000	200.23.16.0/20
Organization 0	11001000 00010111 00010000	00000000	200.23.16.0/23
Organization 1	11001000 00010111 00010010	00000000	200.23.18.0/23
Organization 2	11001000 00010111 00010100	00000000	200.23.20.0/23
...	.....	....	....
Organization 7	11001000 00010111 00011110	00000000	200.23.30.0/23

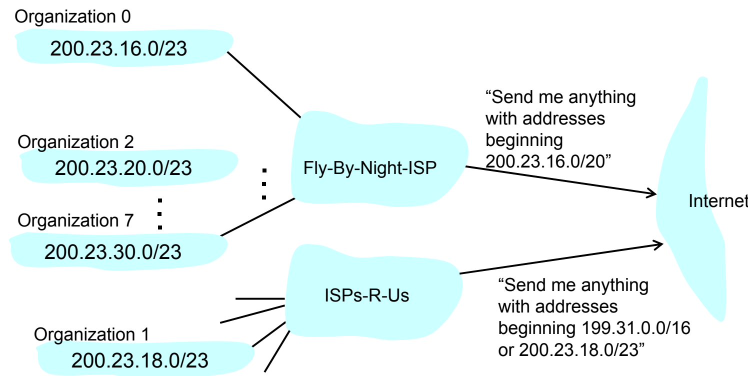
Hierarchical addressing: route aggregation

Hierarchical addressing allows efficient advertisement of routing information:



Hierarchical addressing: more specific routes

What if Organization 1 wants to change the provider?
ISPs-R-Us has a more specific route to Organization 1



IP addressing: the last word...

- How does an ISP get block of addresses?
  - from another (bigger) ISP or
  - with ICANN: Internet Corporation for Assigned Names and Numbers
    - allocates addresses
    - manages DNS
    - assigns domain names, resolves disputes
- Will there be enough IP addresses, ever?
  - No, there are some hacks around the corner (later)

Getting a datagram from source to destination

Known as "forwarding"

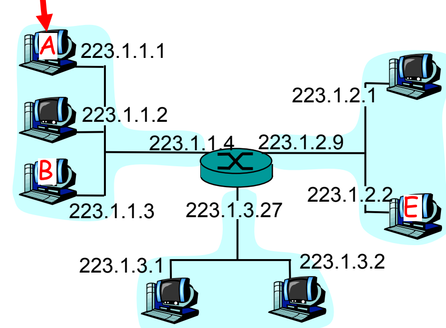
IP datagram:

misc fields	source IP addr	dest IP addr	data

- datagram remains unchanged, as it travels from source to destination
- addr fields of interest here

routing table in A

Dest. Net.	next router	#hops
223.1.1		1
223.1.2	223.1.1.4	2
223.1.3	223.1.1.4	2

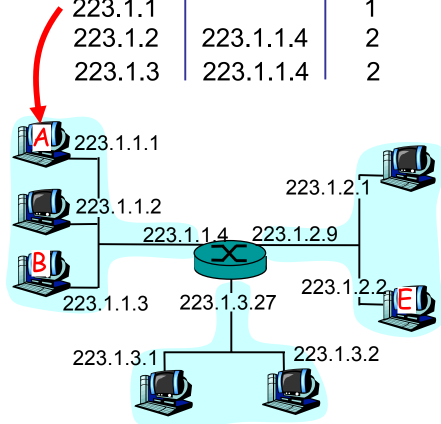


Getting a datagram from source to destination

misc fields	223.1.1.1	223.1.2.2	data

- Starting at A, given IP datagram addressed to B:
- look up net. address of B
- find B is on same net. as A
- link layer will send datagram directly to B inside link-layer frame
  - A and B are directly connected

Dest. Net.	next router	#hops
223.1.1		1
223.1.2	223.1.1.4	2
223.1.3	223.1.1.4	2

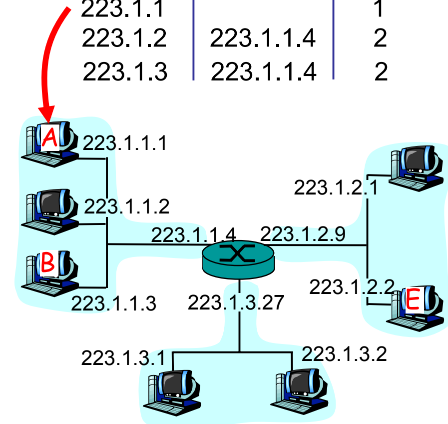


Getting a datagram from source to destination

misc fields	223.1.1.1	223.1.2.2	data

- Starting at A with destination E
- look up network address of E
- E on *different* network
  - A, E not directly attached
- routing table: next hop router to E is 223.1.1.4
- link layer sends datagram to router 223.1.1.4 inside link-layer frame
- datagram arrives at 223.1.1.4
- …

Dest. Net.	next router	#hops
223.1.1		1
223.1.2	223.1.1.4	2
223.1.3	223.1.1.4	2

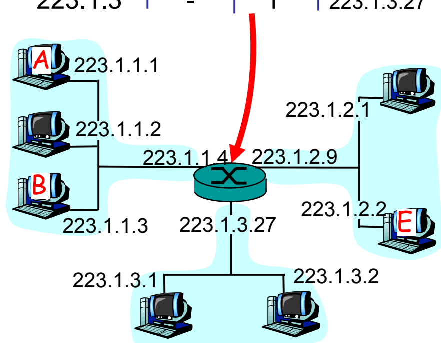


Getting a datagram from source to destination

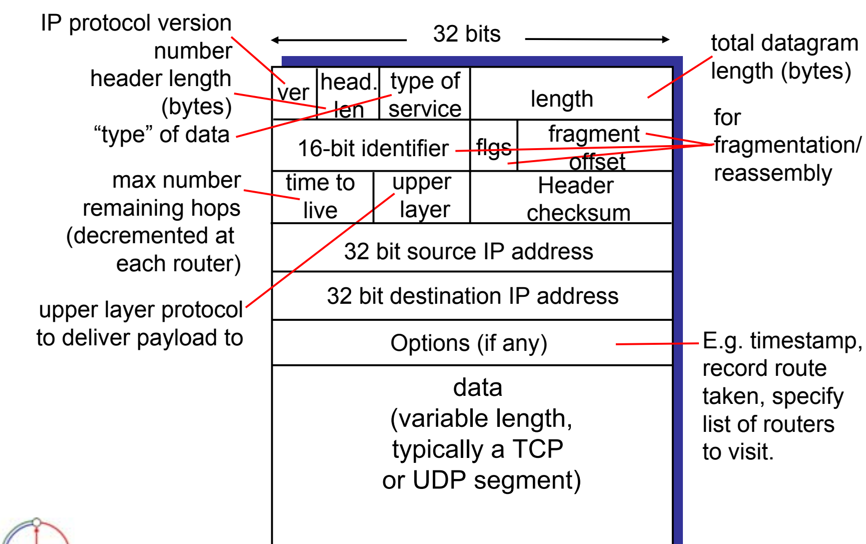
misc fields	223.1.1.1	223.1.2.2	data

- Arriving at 223.1.4, destined for 223.1.2.2
- look up network address of E
- E on *same* network as router's interface 223.1.2.9
  - router, E directly attached
- link layer sends datagram to 223.1.2.2 inside link-layer frame via interface 223.1.2.9
- datagram arrives at 223.1.2.2
- (hooray!)

Dest. network	next router	#hops	interface
223.1.1	-	1	223.1.1.4
223.1.2	-	1	223.1.2.9
223.1.3	-	1	223.1.3.27

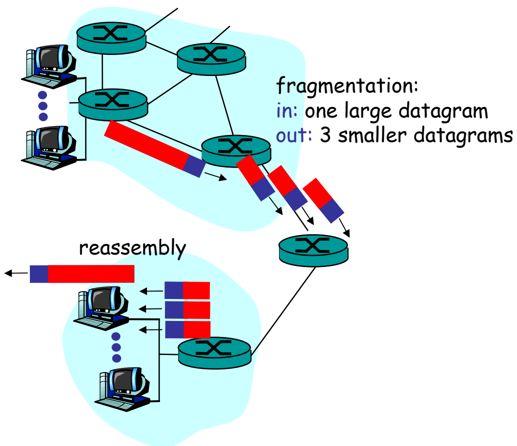


IP datagram format

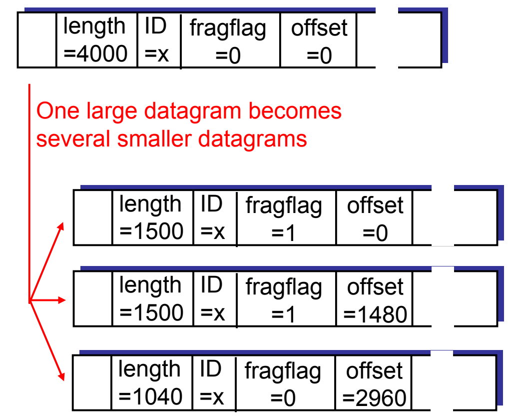


IP Fragmentation and Reassembly

- network links have MTU
  - max. transmission unit
  - largest possible link-level frame
- large IP datagram divided ("fragmented") within net
  - one datagram becomes several datagrams
  - "reassembled" only at *final* destination
  - IP header bits used to identify, order related fragments



IP Fragmentation and Reassembly



ICMP: Internet Control Message Protocol

- used by hosts, routers, gateways to communication network-level information
  - error reporting: unreachable host, network, port, protocol
  - echo request/reply (used by ping)
- network-layer "above" IP:
  - ICMP msgs carried in IP datagrams
- ICMP message: type, code plus first 8 bytes of IP datagram causing error

Some typical types/codes

Type	Code	description
0	0	echo reply (ping)
3	0	dest. network unreachable
3	1	dest host unreachable
3	2	dest protocol unreachable
3	3	dest port unreachable
3	6	dest network unknown
3	7	dest host unknown
4	0	source quench (congestion control - not used)
8	0	echo request (ping)
9	0	route advertisement
10	0	router discovery
11	0	TTL expired
12	0	bad IP header

DHCP: Dynamic Host Configuration Protocol

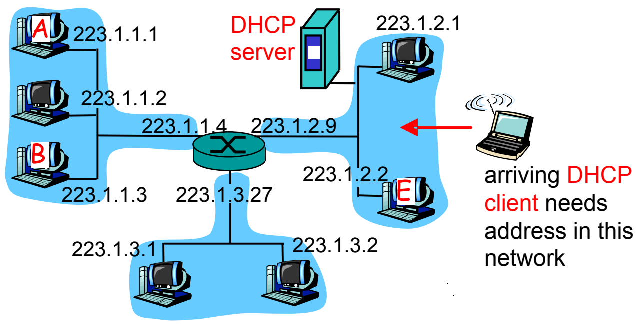
Goals

- allow host to *dynamically* obtain its IP address from network server when it joins network
- Can renew its lease on address in use
- Allows reuse of addresses (only hold address while connected and "on")
- Support for mobile users who want to join network (more shortly)

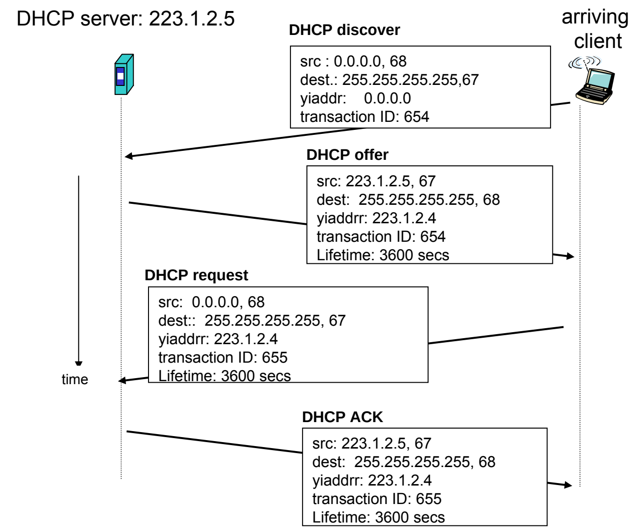
DHCP review

- host broadcasts "DHCP discover" message
- DHCP server responds with "DHCP offer" message
- host requests IP address: "DHCP request" message
- DHCP server sends address: "DHCP ack" message

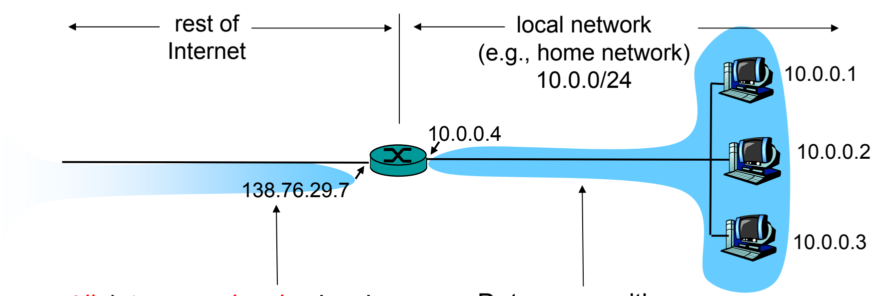
DHCP client-server scenario



DHCP client-server scenario



NAT: Network Address Translation



All datagrams *leaving* local network have same single source NAT IP address: 138.76.29.7, different source port numbers

Datagrams with source or destination in this network have 10.0.0/24 address for source, destination (as usual)

NAT: Network Address Translation

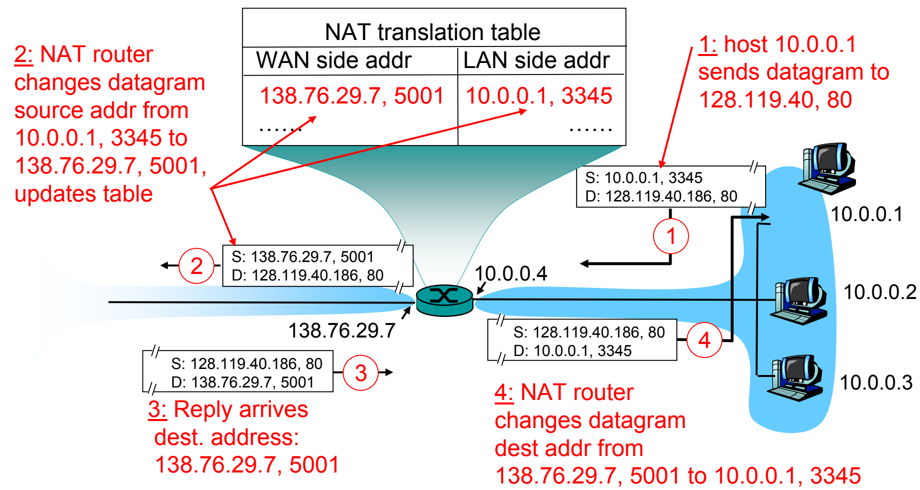
- Motivation
  - local network uses just one IP address as far as outside world is concerned
  - no need to be allocated range of addresses from ISP
  - just one IP address is used for all devices
  - can change addresses of devices in local network without notifying outside world
  - can change ISP without changing addresses of devices in local network
  - devices inside local net not explicitly addressable, visible by outside world (a security plus).
  - BUT: machines cannot be servers!

NAT: Network Address Translation

Implementation: NAT router must

- *outgoing datagrams: replace* (source IP address, port #) of every outgoing datagram to (NAT IP address, new port #)
  - remote clients/servers will respond using (NAT IP address, new port #) as destination addr.
- *remember (in NAT translation table)* every (source IP address, port #) to (NAT IP address, new port #) translation pair
- *incoming datagrams: replace* (NAT IP address, new port #) in dest fields of every incoming datagram with corresponding (source IP address, port #) stored in NAT table

NAT: Network Address Translation



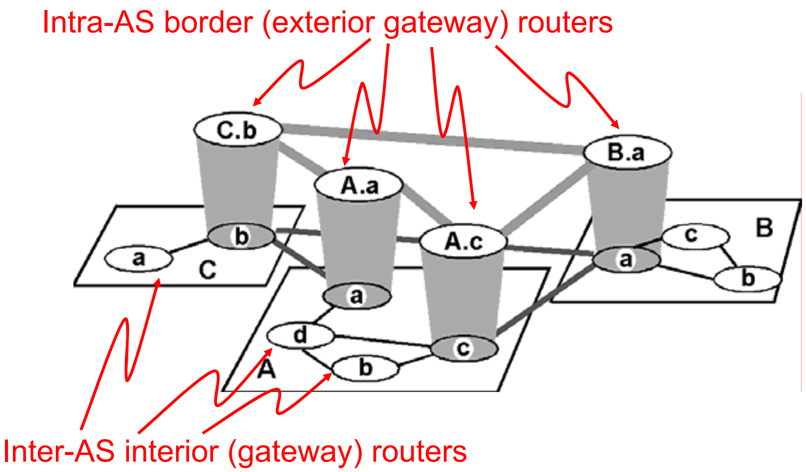
NAT: Network Address Translation

- 16-bit port-number field
  - 60,000 simultaneous connections with a single LAN-side address!
- NAT is controversial
  - routers should only process up to layer 3
  - violates end-to-end argument
    - NAT possibility must be taken into account by app designers, e.g., P2P applications
  - address shortage should instead be solved by IPv6
    - delays deployment of IPv6

Routing in the Internet

- The Global Internet consists of Autonomous Systems (AS) interconnected with each other. There are several “types”
  - Stub AS: small corporation
  - Multihomed AS: large corporation (no transit)
  - Transit AS: provider
- Two-level routing
  - Intra-AS: administrator is responsible for choice
  - Inter-AS: unique standard

Internet AS Hierarchy



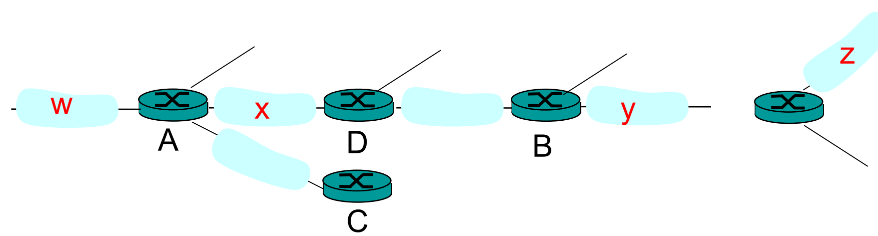
Intra-AS Routing

- Also known as Interior Gateway Protocols (IGP)
- Most common IGPs:
  - RIP: Routing Information Protocol
  - OSPF: Open Shortest Path First
  - IGRP: Interior Gateway Routing Protocol (Cisco proprietary)

RIP (Routing Information Protocol)

- Distance vector algorithm
- Included in BSD-UNIX Distribution in 1982
- Distance metric: number of hops (max = 15 hops)
  - *Can you guess why?*
- Distance vectors: exchanged every 30 sec via Response Message (also called “advertisement”)
- Each advertisement: route to up to 25 destination networks within AS

RIP (Routing Information Protocol)



Destination Network	Next Router	Num. of hops to dest.
w	A	2
y	B	2
z	B	7
x	--	1
….	….	….

Routing table in D

RIP: Link Failure and Recovery

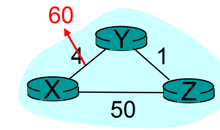
If no advertisement heard after 180 sec → neighbor/link declared dead

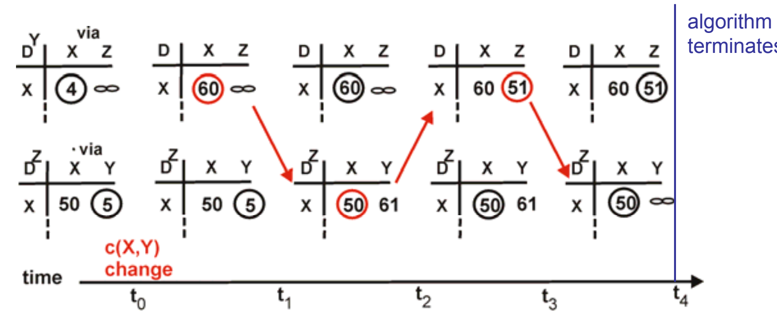
- routes via neighbor invalidated
- new advertisements sent to neighbors
- neighbors in turn send out new advertisements (if tables changed)
- link failure info quickly propagates to entire net
- poison reverse (next slide) used to prevent ping-pong loops (infinite distance = 16 hops)

Distance Vector: poisoned reverse

If Z routes through Y to get to X :

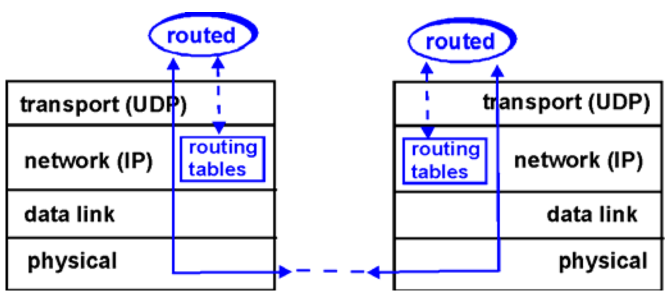
- Z tells Y its (Z's) distance to X is infinite (so Y won't route to X via Z)
- will this completely solve count to infinity problem…?





RIP Table processing

- RIP routing tables managed by *application-level* process called route-d (daemon)
- advertisements sent in UDP packets, periodically repeated



RIP Table example (continued)

Router: *giroflee.eurocom.fr*

Destination	Gateway	Flags	Ref	Use	Interface
127.0.0.1	127.0.0.1	UH	0	26492	lo0
192.168.2.	192.168.2.5	U	2	13	fa0
193.55.114.	193.55.114.6	U	3	58503	le0
192.168.3.	192.168.3.5	U	2	25	qaa0
224.0.0.0	193.55.114.6	U	3	0	le0
default	193.55.114.129	UG	0	143454	

- Three attached class C networks (LANs)
- Router only knows routes to attached LANs
- Default router used to "go up"
- Route multicast address: 224.0.0.0
- Loopback interface (for debugging)

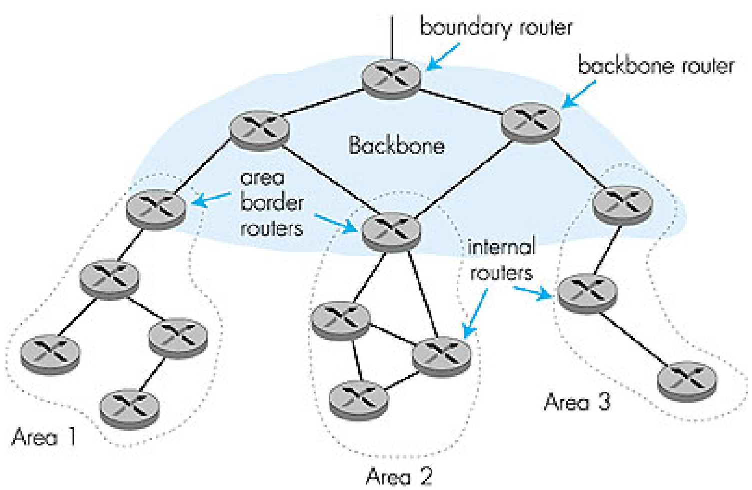
OSPF (Open Shortest Path First)

- "open": publicly available
- Uses Link State algorithm
  - LS packet dissemination
  - Topology map at each node
  - Route computation using Dijkstra's algorithm
- OSPF advertisement carries one entry per neighbor router
- Advertisements disseminated to entire AS (via flooding)

OSPF "advanced" features (not in RIP)

- Security
  - all OSPF messages authenticated
  - therefore no malicious intrusion
  - TCP connections used
- Multiple same-cost paths allowed (only one path in RIP)
- For each link, multiple cost metrics for different TOS
  - e.g., satellite link cost set "low" for best effort; high for real time
- Integrated uni- and multicast support:
  - Multicast OSPF (MOSPF) uses same topology data base as OSPF
- Hierarchical OSPF in large domains

Hierarchical OSPF



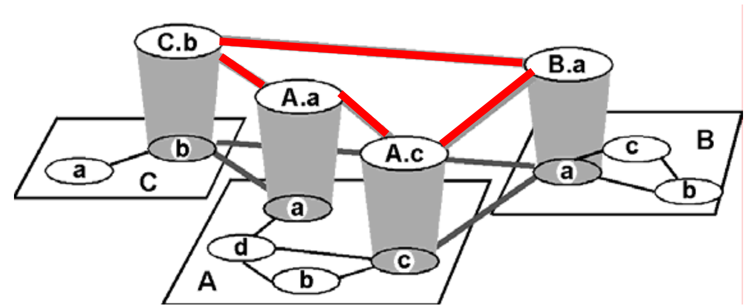
Hierarchical OSPF

- Two-level hierarchy: local area or backbone
  - Link-state advertisements only in area
  - each node has detailed area topology but only knows direction (shortest path) to nets in other areas.
- Area border routers
  - "summarize" distances to networks in own area
  - advertise to other area border routers.
- Backbone routers
  - run OSPF routing limited to backbone.
- Boundary routers
  - connect to other ASs.

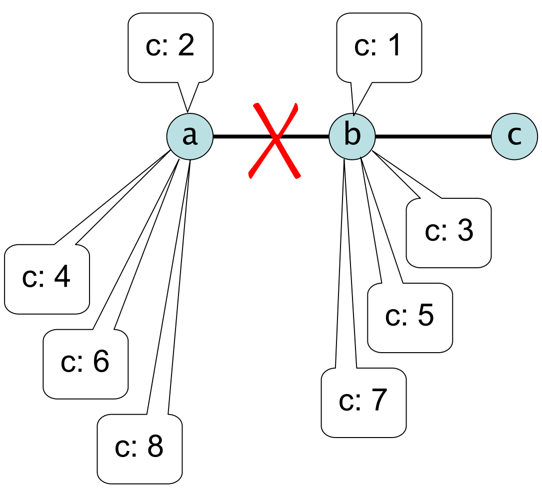
[E]IGRP: [Enhanced] Interior Gateway Routing Protocol

- CISCO proprietary; successor of RIP (mid 80s)
- Distance Vector, like RIP
- several cost metrics (delay, bandwidth, reliability, load etc)
- uses TCP to exchange routing updates
- Loop-free routing via Distributed Updating Algorithm (DUAL) based on *diffused computation*

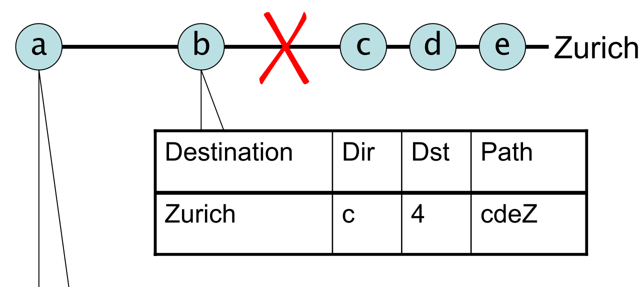
Inter-AS routing



Remember: Count to Infinity Problem



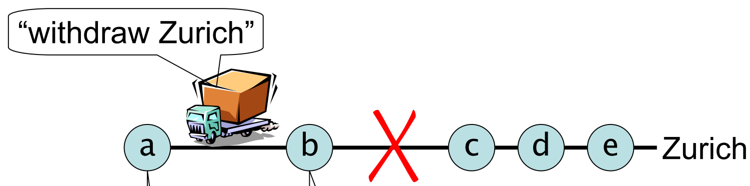
BGP does not count to infinity



Destination	Dir	Dst	Path
Zurich	c	4	cdeZ

Destination	Dir	Dst	Path
Zurich	b	5	bcdeZ

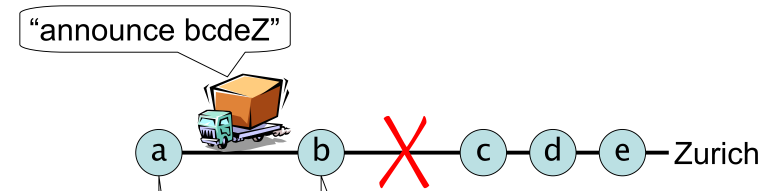
BGP does not count to infinity



Destination	Dir	Dst	Path
~~Zurich~~	~~c~~	~~4~~	~~cdeZ~~

Destination	Dir	Dst	Path
~~Zurich~~	~~b~~	~~5~~	~~bcdeZ~~

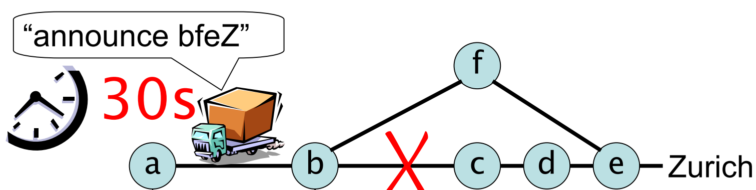
BGP Basics Continued



Destination	Dir	Dst	Path
~~Zurich~~	~~c~~	~~4~~	~~cdeZ~~

Destination	Dir	Dst	Path
~~Zurich~~	~~b~~	~~5~~	~~bcdeZ~~

BGP Basics Continued

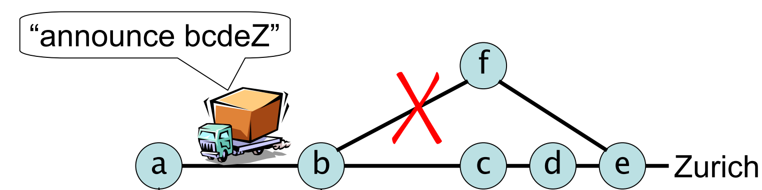


Destination	Dir	Dst	Path
~~Zurich~~	~~c~~	~~4~~	~~cdeZ~~
Zurich	f	3	feZ

backup
active

Destination	Dir	Dst	Path
Zurich	b	4	bfeZ

BGP Basics Continued



Destination	Dir	Dst	Path
Zurich	c	4	cdeZ
~~Zurich~~	~~f~~	~~3~~	~~feZ~~

backup
active

Destination	Dir	Dst	Path
Zurich	b	~~4~~	~~bfeZ~~

BGP (Border Gateway Protocol)

- BGP is *the* Internet de-facto standard
- *Path* Vector protocol

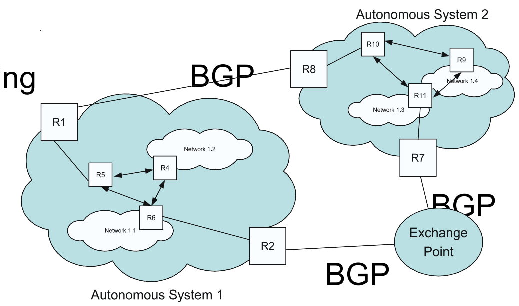
1) Receive BGP update (announce or withdrawal) from a neighbor.
2) Update routing table.
3) Does update affect active route? (Loop detection, policy, etc.) If yes, send update to all neighbors that are allowed by policy.

- MinRouteAdver: At most 1 announce per neighbor per 30+jitter seconds.
- Store the active routes of the neighbors.

Internet Architecture

Destination	Dir	Dst	Path
Zurich	c	4	cdeZ
172.30.160/19	R1	4	1239 1 3561

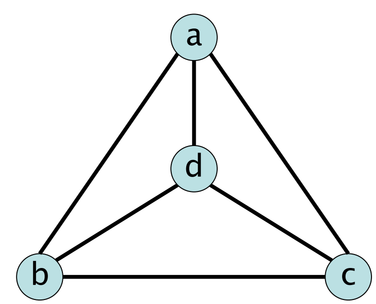
- iBGP
- Route flap dampening
- Multipath
- Soft configuration
- …



Internet inter-AS routing: BGP

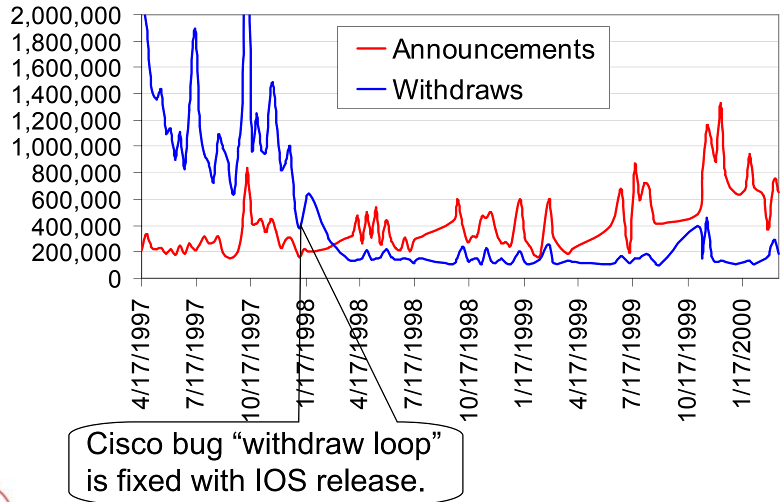
- BGP messages exchanged using TCP.
- BGP messages
  - OPEN: opens TCP connection to peer and authenticates sender
  - UPDATE: advertises new path (or withdraws old)
  - KEEPALIVE keeps connection alive in absence of UPDATES; also ACKs OPEN request
  - NOTIFICATION: reports errors in previous msg; also used to close connection
- Policy
  - Even if two BGP routers are connected they may not announce all their routes or use all the routes of the other
  - Example: if AS A does not want to route traffic of AS B, then A should simply not announce anything to B.

Robustness of BGP

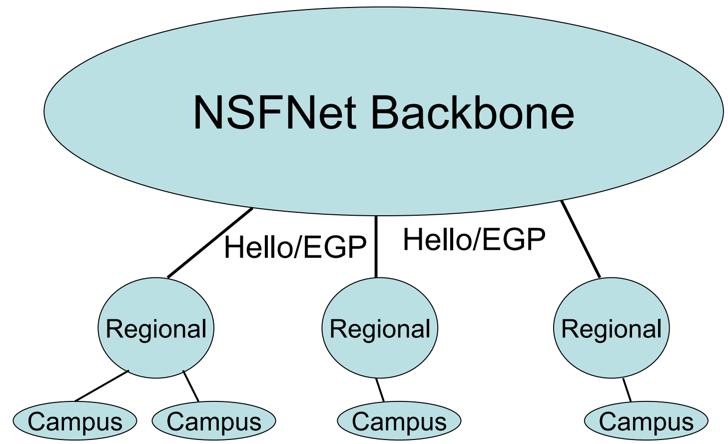


- We are interested in routes to destination d.
- Nodes a,b,c all have the policy to prefer a 2-hop route through their clockwise neighbor over a direct 1-hop route to destination d.

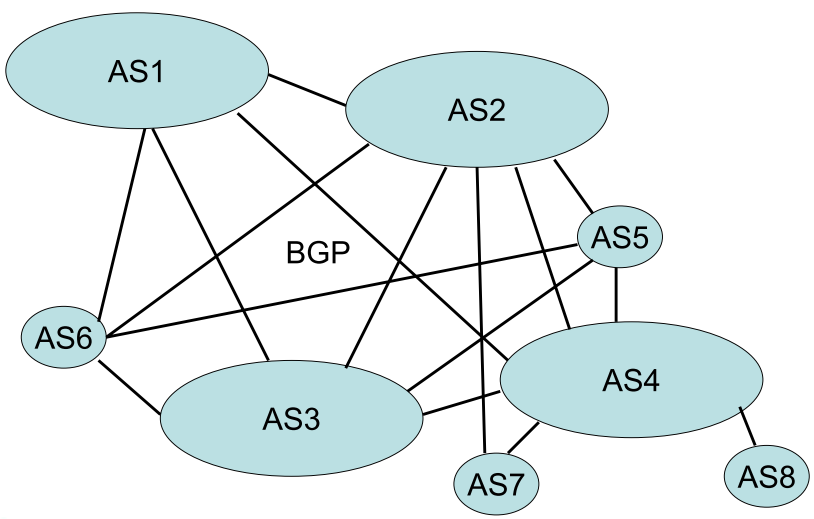
BGP Update Traffic (Mae-East)



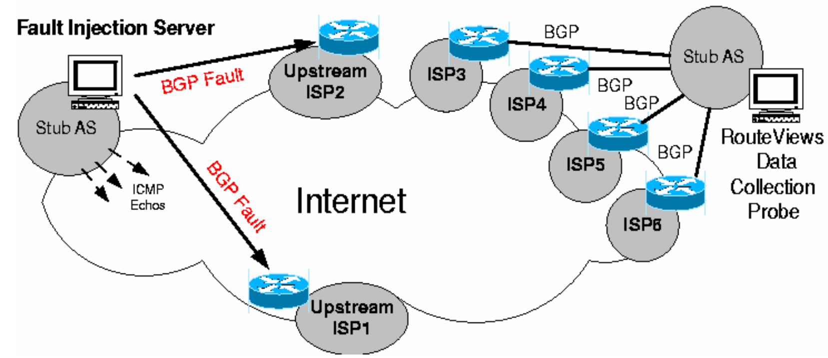
Internet Evolution: NSFNet (1995)



Internet Evolution: Today

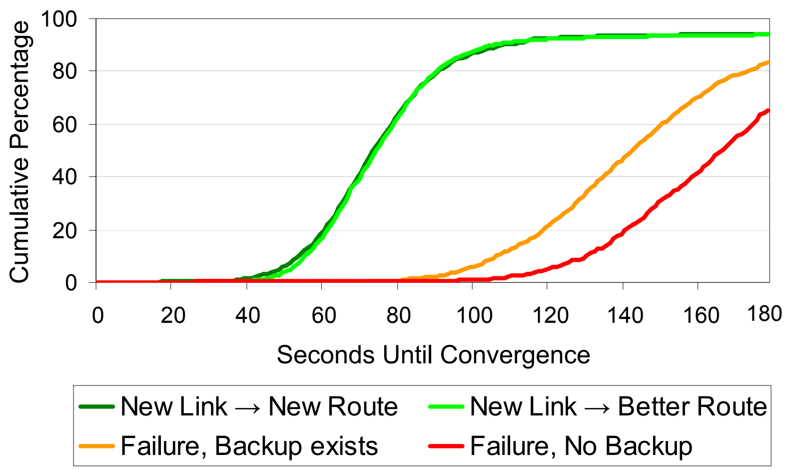


Experimental Setup



- Analyzed secondary paths of 20x20 AS pairs:
  - Inject and monitor BGP faults.
  - Survey providers on policies.

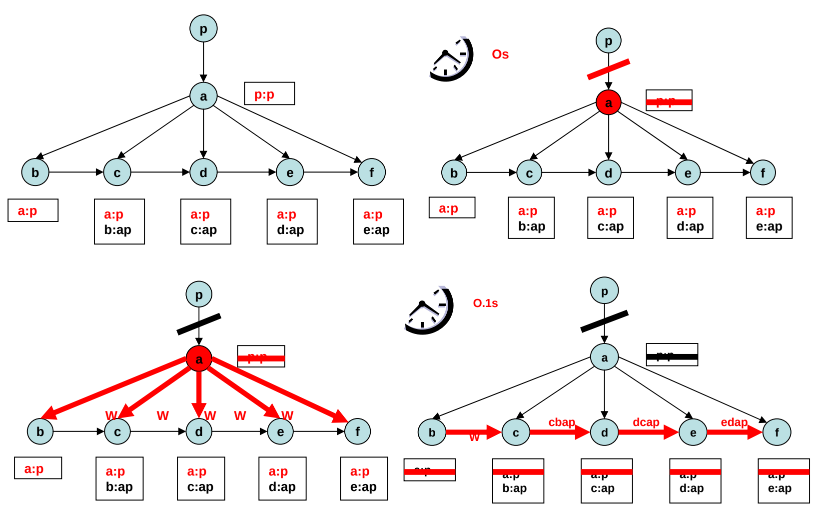
BGP Convergence Times



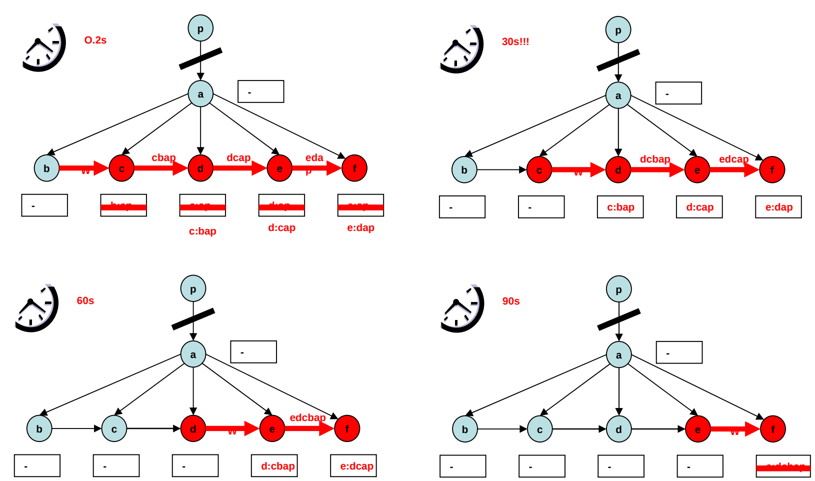
BGP Convergence Results

- If a link comes up, the convergence time is in the order of time to forward a message on the shortest path.
- If a link goes down, the convergence time is in the order of time to forward a message on the longest path.

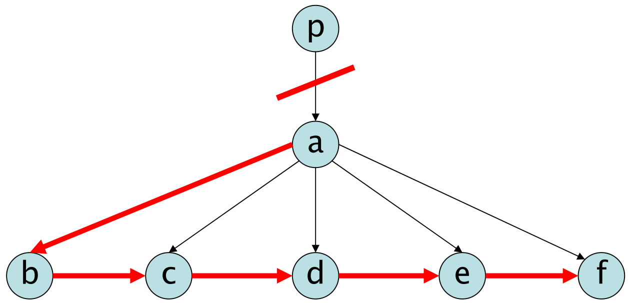
Intuition for Slow Convergence



Intuition for Slow Convergence



Intuition for Slow Convergence

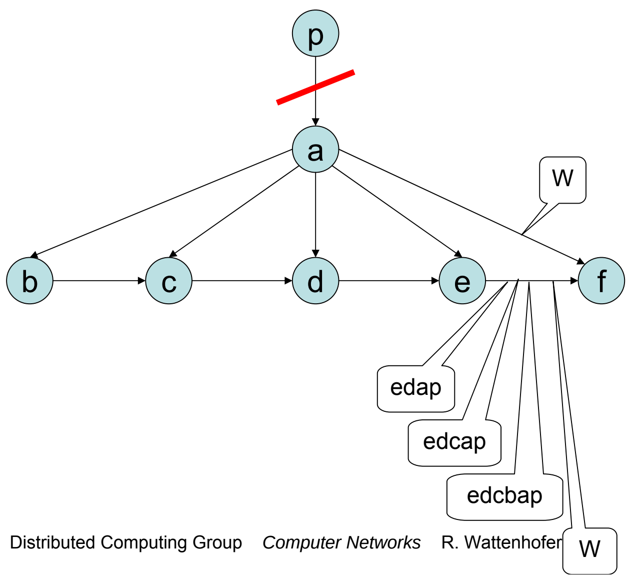


Convergence in the time to forward a message on the longest path.

Example of BGP Convergence

Time	BGP Message/Event
10:40:30	*2129 withdraws p*
10:41:08	*2117 announces 5696 2129 p*
10:41:32	*2117 announces 1 5696 2129 p*
10:41:50	*2117 announces 2041 3508 3508 4540 7037 1239 5696 2129 p*
10:42:17	*2117 announces 1 2041 3508 3508 4540 7037 1239 5696 2129 p*
10:43:05	*2117 announces 2041 3508 3508 4540 7037 1239 6113 5696 2129 p*
10:43:35	*2117 announces 1 2041 3508 3508 4540 7037 1239 6113 5696 2129 p*
10:43:59	*2117 withdraws p*

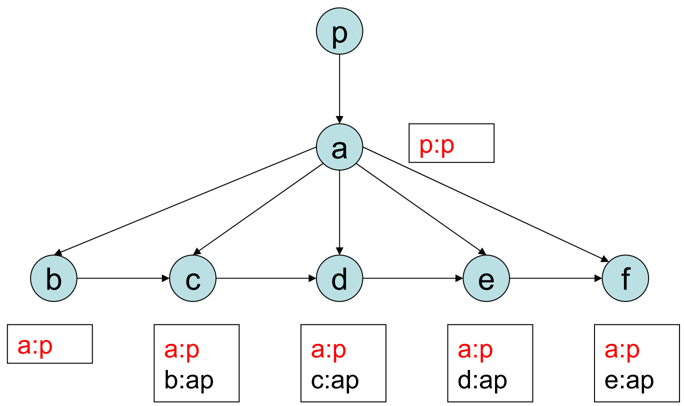
Remember the Example



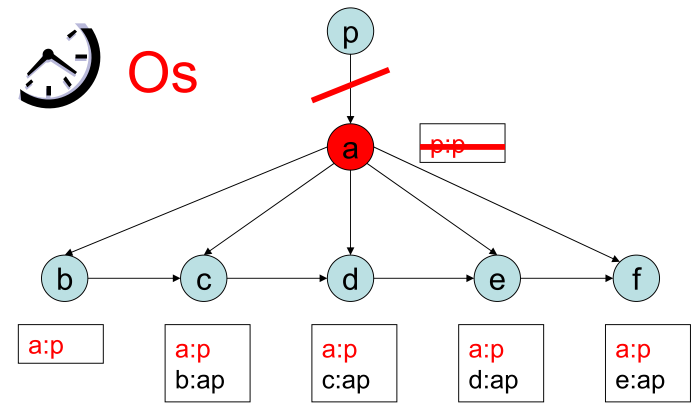
What might help?

- Idea: Attach a “cause tag” to the withdrawal message identifying the failed link/node (for a given prefix).
- It can be shown that a cause tag reduces the convergence time to the shortest path
- Problems
  - Since BGP is widely deployed, it cannot be changed easily
  - ISP’s (AS’s) don’t like the world to know that it is their link that is not stable, and cause tags do exactly that.
  - Race conditions make the cause tags protocol intricate

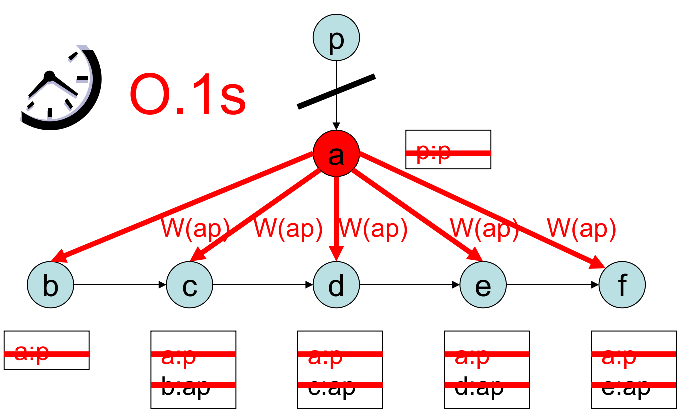
Example with BGP-CT (Cause Tags)



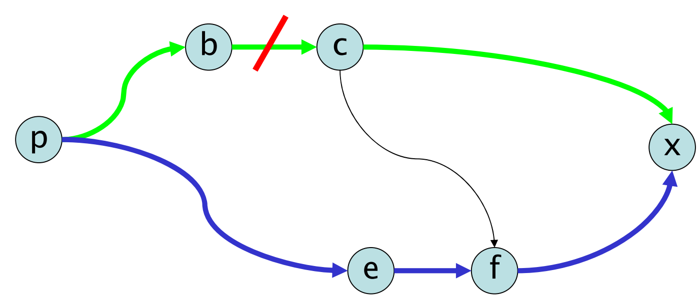
Example with BGP-CT



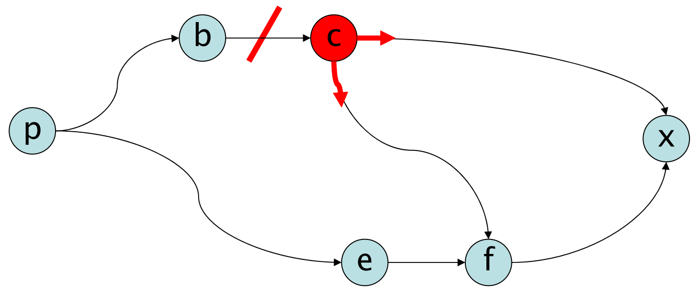
Example with BGP-CT



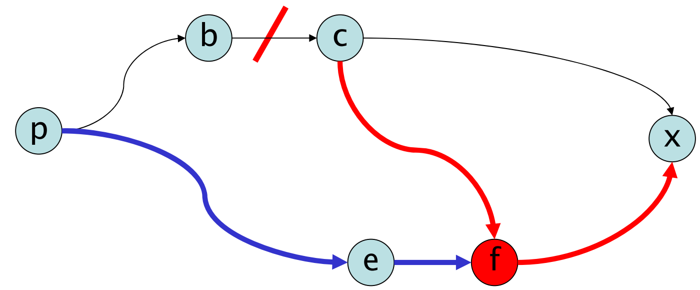
Convergence Time using Cause Tags



Convergence Time using Cause Tags



Convergence Time using Cause Tags



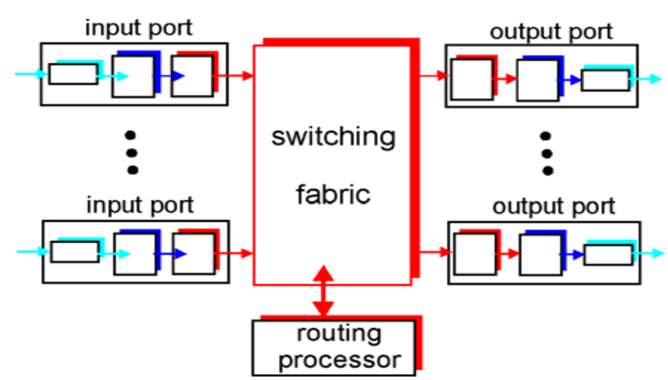
Convergence in the time to forward a message on the new shortest path (instead of the longest).

Why are Intra- and Inter-AS routing different?

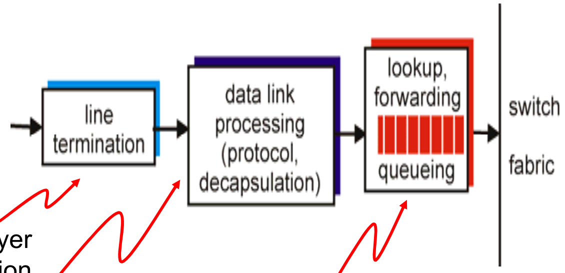
- Policy
  - Inter-AS: admin wants control over how its traffic routed, and who routes through its net.
  - Intra-AS: single admin, so no policy decisions needed
- Scale
  - hierarchical routing saves table size, reduced update traffic
- Performance
  - Intra-AS: can focus on performance
  - Inter-AS: policy may dominate over performance

Router Architecture Overview

Two key router functions
- run routing algorithms/protocols (RIP, OSPF, BGP)
- *switch* datagrams from incoming to outgoing link



Input Port Functions



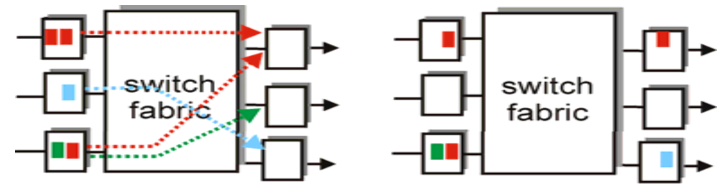
Physical layer
bit-level reception

Data link layer
e.g., Ethernet
see chapter 5

Decentralized switching
- given datagram dest., lookup output port using routing table in input port memory
- goal: complete input port processing at “line speed”
- queuing: if datagrams arrive faster than forwarding rate into switch fabric

Input Port Queuing

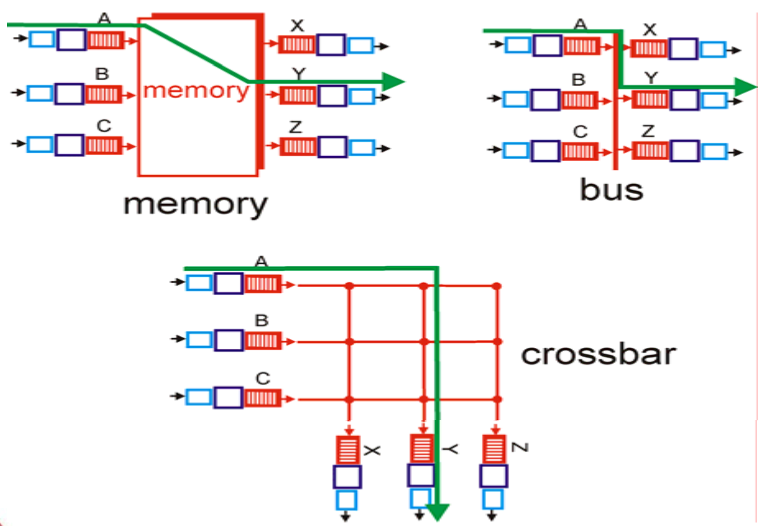
- Fabric slower that input ports combined
  - queueing may occur at input queues
- Head-of-the-Line (HOL) blocking
  - queued datagram at front of queue prevents others in queue from moving forward
- queuing delay and loss due to input buffer overflow



output port contention at time t - only one red packet can be transferred

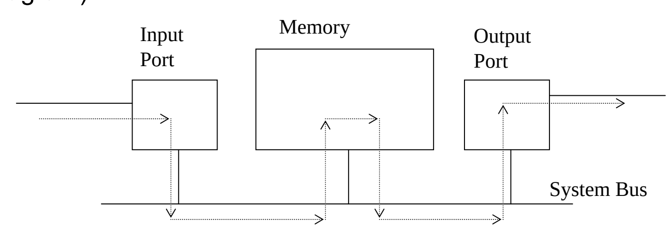
green packet experiences HOL blocking

Three types of switching fabrics



Switching Via Memory

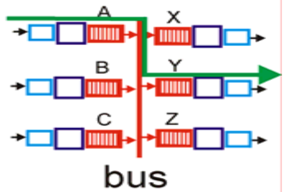
First generation routers
- packet copied by system’s (single) CPU
- speed limited by memory bandwidth (2 bus crossings per datagram)



Modern routers
- input port processor performs lookup, copy into memory
- Cisco Catalyst 8500

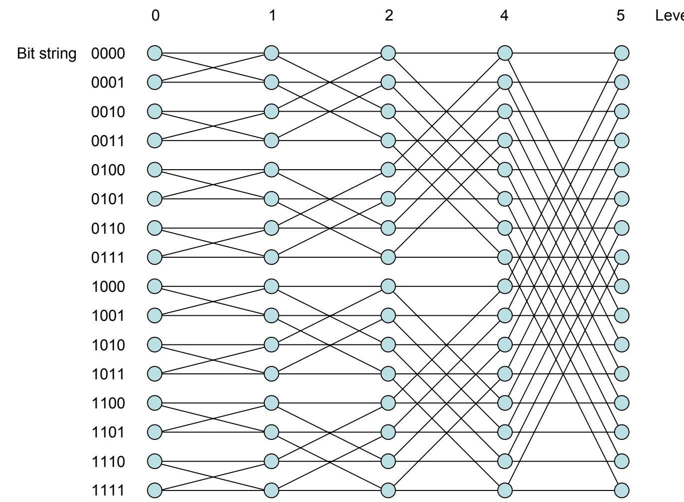
Switching Via Bus or Interconnection Network

- 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)



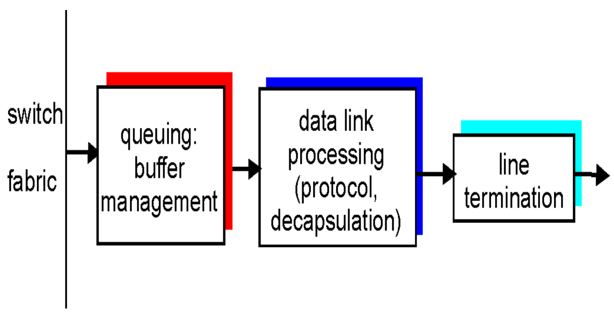
- Interconnection Network: overcome bus bandwidth limitations
- Banyan networks, other interconnection nets initially developed to connect processors in multiprocessor
- Advanced design: fragmenting datagram into fixed length cells, switch cells through the fabric.
- Cisco 12000: switches Gbps through the interconnection network

Butterfly Network



Butterfly with Dimension d=4

Output ports



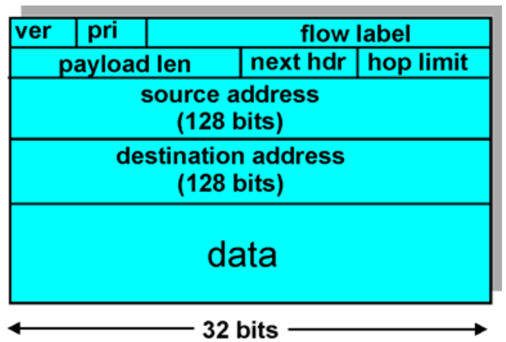
- *Buffering* required when datagrams arrive from fabric faster than the transmission rate
- *Scheduling discipline* chooses among queued datagrams for transmission

IPv6

- Initial motivation: 32-bit address space completely allocated by 2008.
- Additional motivation
  - header format helps speed processing/forwarding
  - header changes to facilitate QoS (quality of service)
  - new “anycast” address: route to “best” of several replicated servers
- IPv6 datagram format:
  - fixed-length 40 byte header
  - no fragmentation allowed

IPv6 Header

- Priority
  - identify priority among datagrams in flow
- Flow Label
  - identify datagrams in same “flow” (concept of“flow” not well defined)
- Next header
  - identify upper layer protocol for data



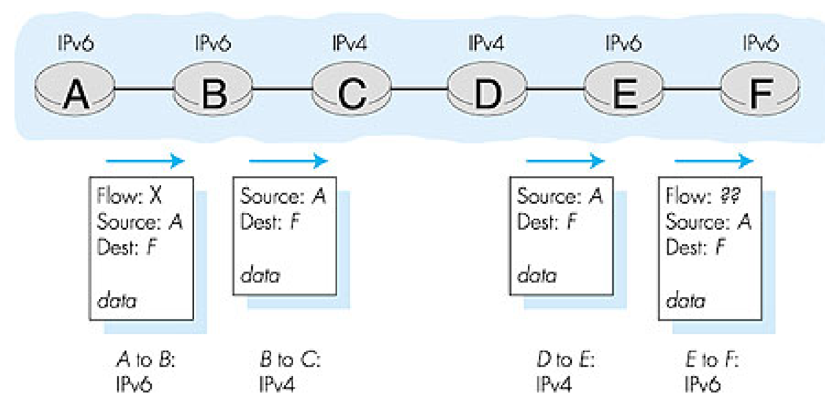
Other Changes from IPv4

- Checksum
  - removed entirely to reduce processing time at each hop
- Options
  - allowed, but outside of header
  - indicated by “Next Header” field
- ICMPv6: new version of ICMP
  - additional message types, e.g. “Packet Too Big”
  - multicast group management functions

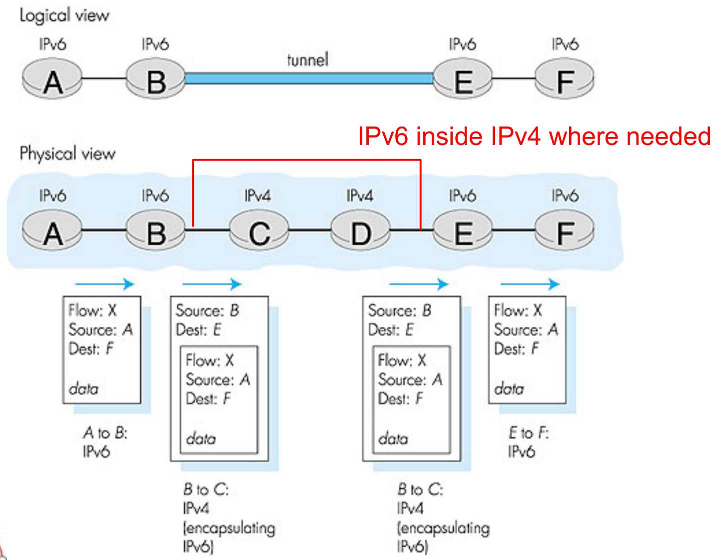
Transition From IPv4 To IPv6

- Not all routers can be upgraded simultaneously
  - no “flag days”
  - How will the network operate with mixed IPv4 and IPv6 routers?
- Two proposed approaches
  - Dual Stack
    - some routers with dual stack (v6, v4) can “translate” between formats
  - Tunneling
    - IPv6 carried as payload in IPv4 datagram among IPv4 routers

Dual Stack Approach



Tunneling


Logical view
IPv6
IPv6
tunnel
IPv6
IPv6
A
B
E
F
IPv6 inside IPv4 where needed
Physical view
IPv6
IPv6
IPv4
IPv4
IPv6
IPv6
A
B
C
D
E
F
Flow: X
Source: A
Dest: F
data
Source: B
Dest: E
Flow: X
Source: A
Dest: F
data
Source: B
Dest: E
Flow: X
Source: A
Dest: F
data
Flow: X
Source: A
Dest: F
data
A to B:
IPv6
B to C:
IPv4
(encapsulating
IPv6)
B to C:
IPv4
(encapsulating
IPv6)
E to F:
IPv6