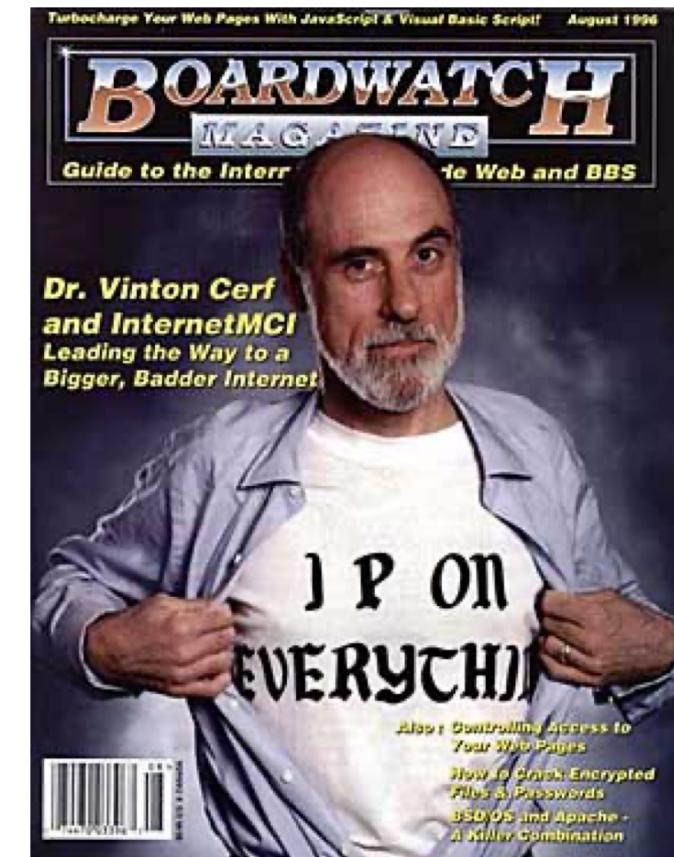




# Network layer: DHCP, NAT, Tunneling & Routing

Vivek Shah

Based on slides compiled by  
Marcos Vaz Salles



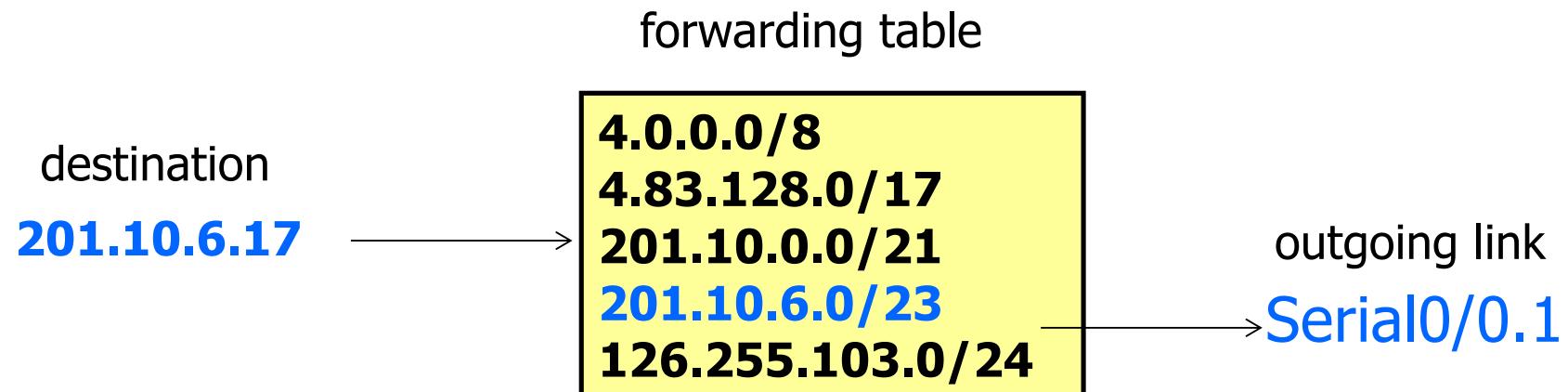
# Recap: Network layer

- What is the “best effort guarantee” of network layer ?
- Why can fragmentation happen at routers ?  
How does IPv4 handle it ? Why does IPv6 not handle it ?
- What do forwarding tables in routers contain ?  
Why is longest prefix match chosen ?
- Why is an IP address hierarchical ? What is a subnet mask ?
- How are IP addresses allocated ?



# Forwarding Revisited

- How to resolve multiple matches?
  - Router identifies most specific prefix:  
*longest prefix match (LPM)*
  - Cute algorithmic problem to achieve fast lookups

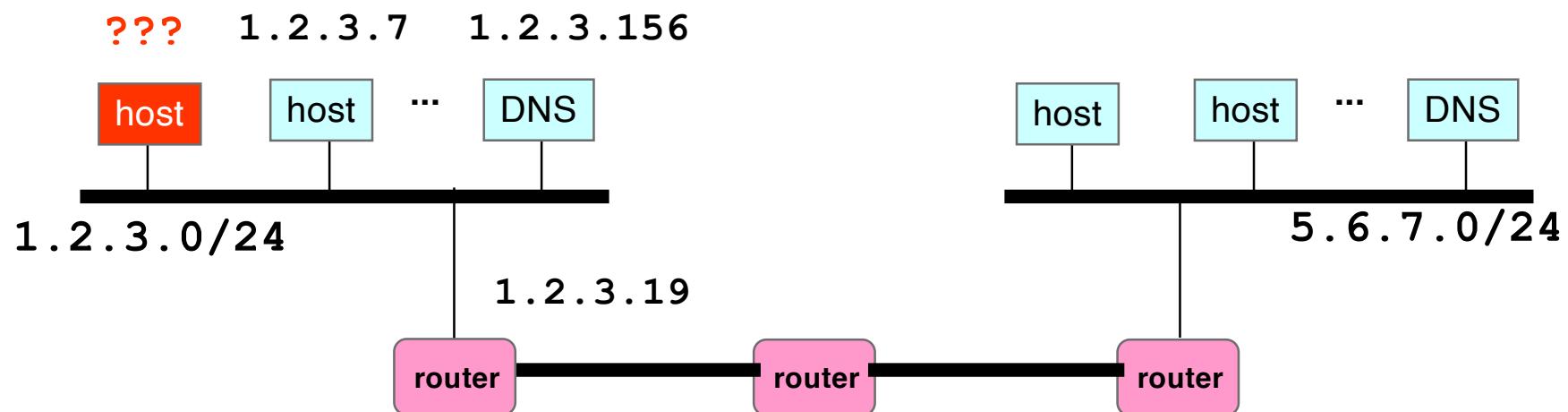


Source: Freedman (partial)



# How To Bootstrap an End Host?

- What local Domain Name System server to use?
- What IP address the host should use?
- How to send packets to remote destinations?
- How to ensure incoming packets arrive?

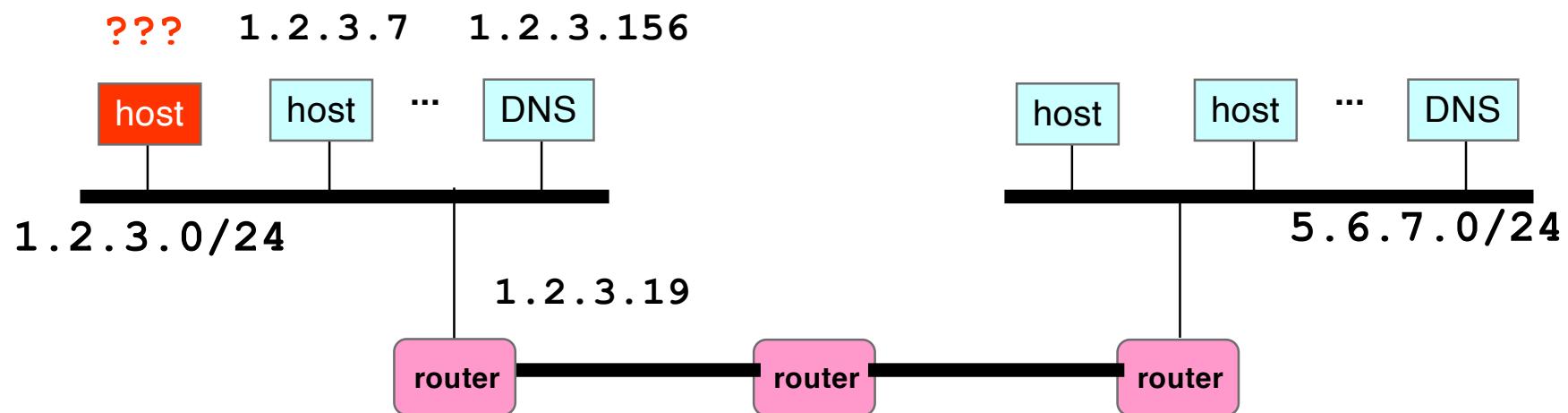


Source: Freedman



# Avoiding Manual Configuration

- Dynamic Host Configuration Protocol (DHCP)
  - End host learns how to send packets
  - Learn IP address, DNS servers, and gateway
- Address Resolution Protocol (ARP)
  - Others learn how to send packets to the end host
  - Learn mapping between IP address & interface address



Source: Freedman



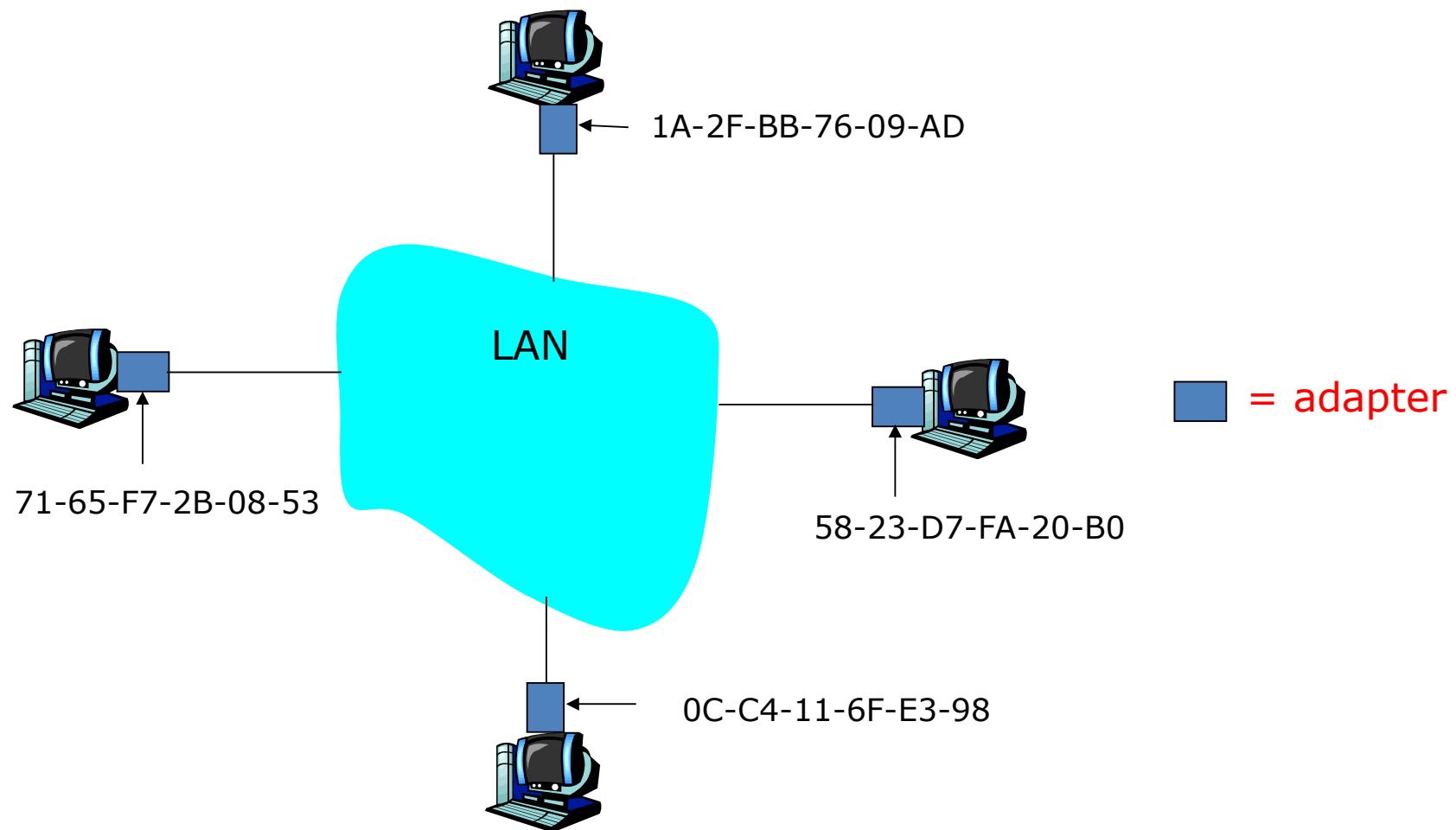
# Key Ideas in Both Protocols

- **Broadcasting:** when in doubt, shout!
  - Broadcast query to all hosts in the local-area-network
  - ... when you don't know how to identify the right one
- **Caching:** remember the past for a while
  - Store the information you learn to reduce overhead
  - Remember your own address & other host's addresses
- **Soft state:** ... but eventually forget the past
  - Associate a time-to-live field with the information
  - ... and either refresh or discard the information
  - Key for robustness in the face of unpredictable change

Source: Freedman



# Media Access Control (MAC) Addresses

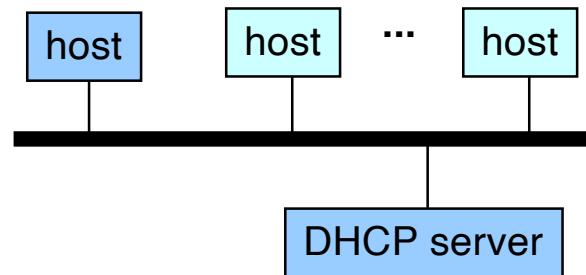


Source: Freedman



# Bootstrapping Problem

- Host doesn't have an IP address yet
  - So, host doesn't know what source address to use
- Host doesn't know who to ask for an IP address
  - So, host doesn't know what destination addr to use
- Solution: shout to discover a server who can help
  - Broadcast a DHCP server-discovery message
  - Server sends a DHCP “offer” offering an address

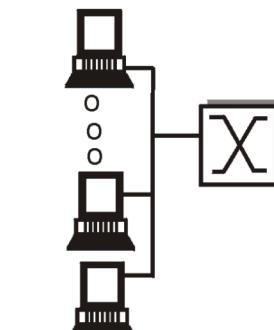


Source: Freedman

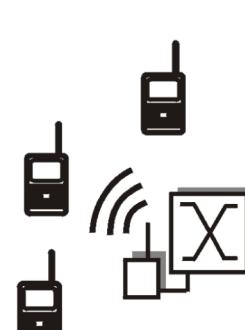


# Broadcasting

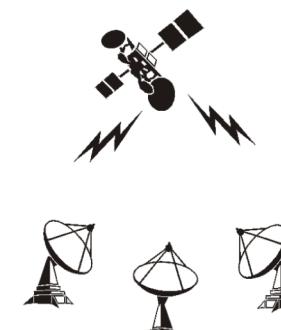
- Broadcasting: sending to everyone
  - Special destination address: FF-FF-FF-FF-FF-FF
  - All adapters on the LAN receive the packet
- Delivering a broadcast packet
  - Easy on a “shared media”
  - Like shouting in a room – everyone can hear you



shared wire  
(e.g. Ethernet)



shared wireless  
(e.g. Wavelan)



satellite



cocktail party

Source: Freedman



# Response from the DHCP Server

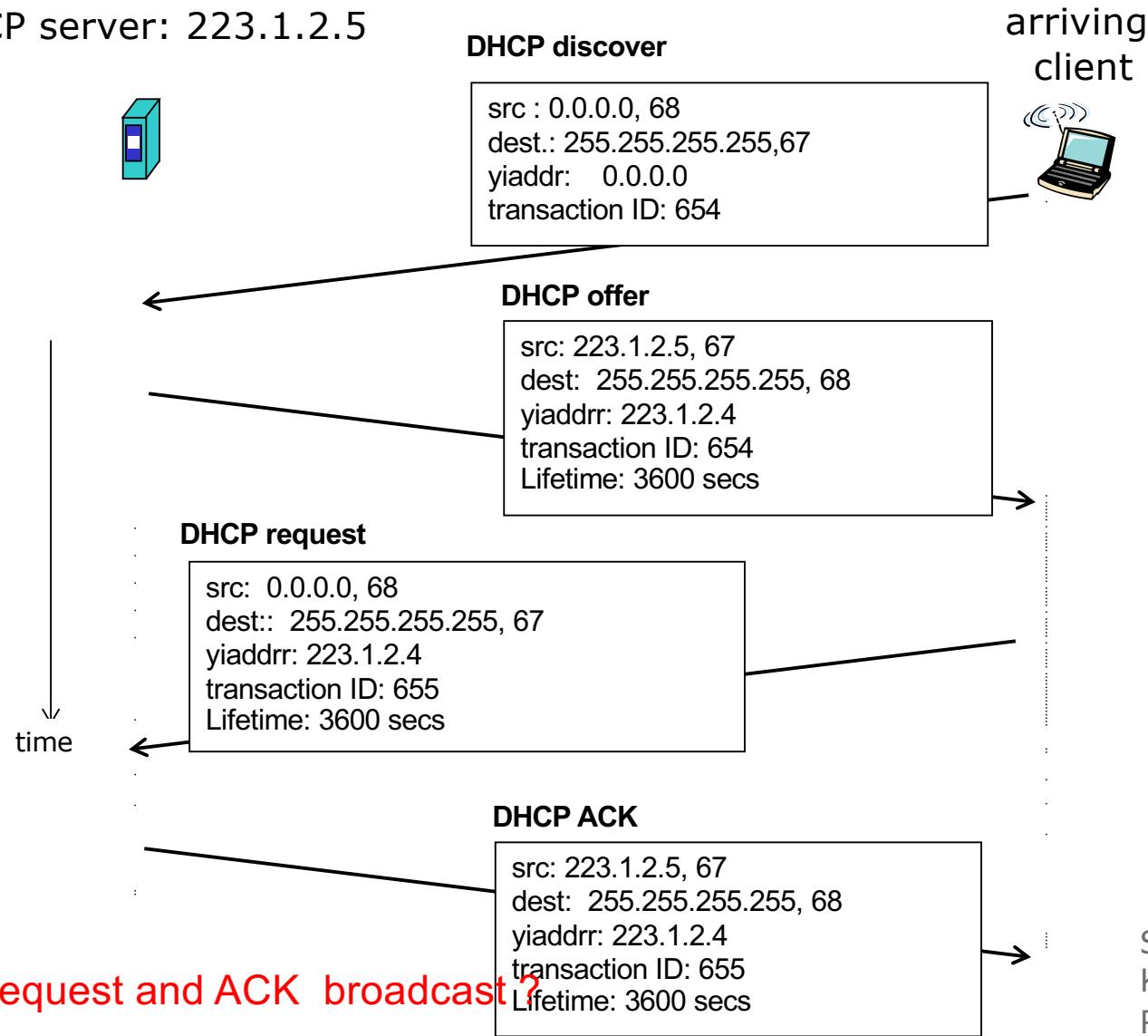
- DHCP “offer message” from the server
  - Configuration parameters (proposed IP address, mask, gateway router, DNS server, ...)
  - Lease time (the time the information remains valid)
- Multiple servers may respond
  - Multiple servers on the same broadcast media
  - Each may respond with an offer
  - The client can decide which offer to accept
- Accepting one of the offers
  - Client sends a DHCP request echoing the parameters
  - The DHCP server responds with an ACK to confirm
  - ... and the other servers see they were not chosen

Source: Freedman



# DHCP client-server scenario

DHCP server: 223.1.2.5



Why are DHCP request and ACK broadcast?

Source:  
Kurose &  
Ross

# Deciding What IP Address to Offer

- Server as centralized configuration database
  - All parameters are statically configured in the server
  - E.g., a dedicated IP address for each MAC address
  - Avoids complexity of configuring hosts directly
  - ... while still having a permanent IP address per host
- Or, dynamic assignment of IP addresses
  - Server maintains a pool of available addresses
  - ... and assigns them to hosts on demand
  - Leads to less configuration complexity
  - ... and more efficient use of the pool of addresses
  - Though, it is harder to track the same host over time

Source: Freedman



# Soft State: Refresh or Forget

- Why is a lease time necessary?
  - Client can release the IP address (DHCP RELEASE)
    - E.g., “ipconfig /release” at the DOS prompt
    - E.g., clean shutdown of the computer
  - But, the host might not release the address
    - E.g., the host crashes (blue screen of death!)
    - E.g., buggy client software
  - And you don’t want the address to be allocated forever
- Performance trade-offs
  - Short lease time: returns inactive addresses quickly
  - Long lease time: avoids overhead of frequent renewals

Source: Freedman



# Middleboxes

- Middleboxes are intermediaries
  - Interposed in-between the endpoints
  - Often without knowledge of endpoints
- Myriad uses
  - Network address translators
  - Firewalls
  - Tunnel endpoints
  - Traffic shapers
  - Intrusion detection systems
  - Transparent Web proxy caches
  - Application accelerators

“An abomination!”

- Violation of layering
- Hard to reason about
- Responsible for subtle bugs

“A practical necessity!”

- Solve real/pressing problems
- Needs not likely to go away



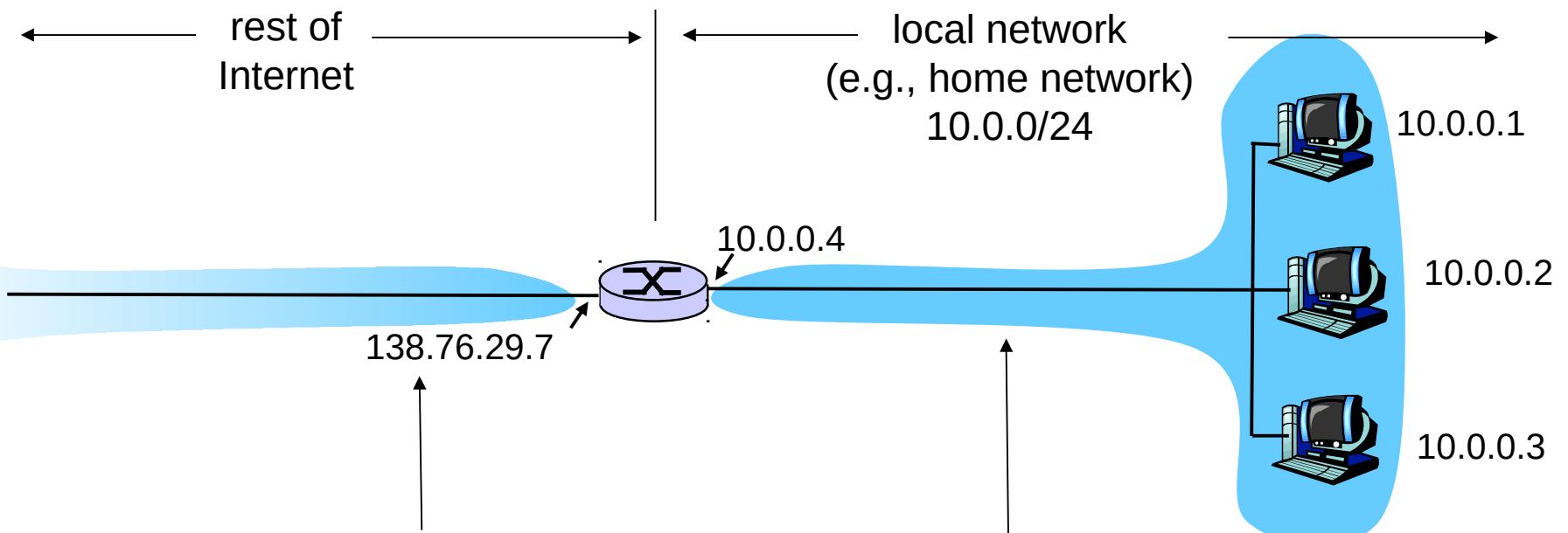
# History of NATs

- IP address space depletion
  - Clear in early 90s that  $2^{32}$  addresses not enough
  - Work began on a successor to IPv4
- In the meantime...
  - Share addresses among numerous devices
  - ... without requiring changes to existing hosts
- Meant to provide short-term remedy
  - Now: NAT is widely deployed, much more than IPv6

Source: Freedman



# NAT: Network Address Translation I



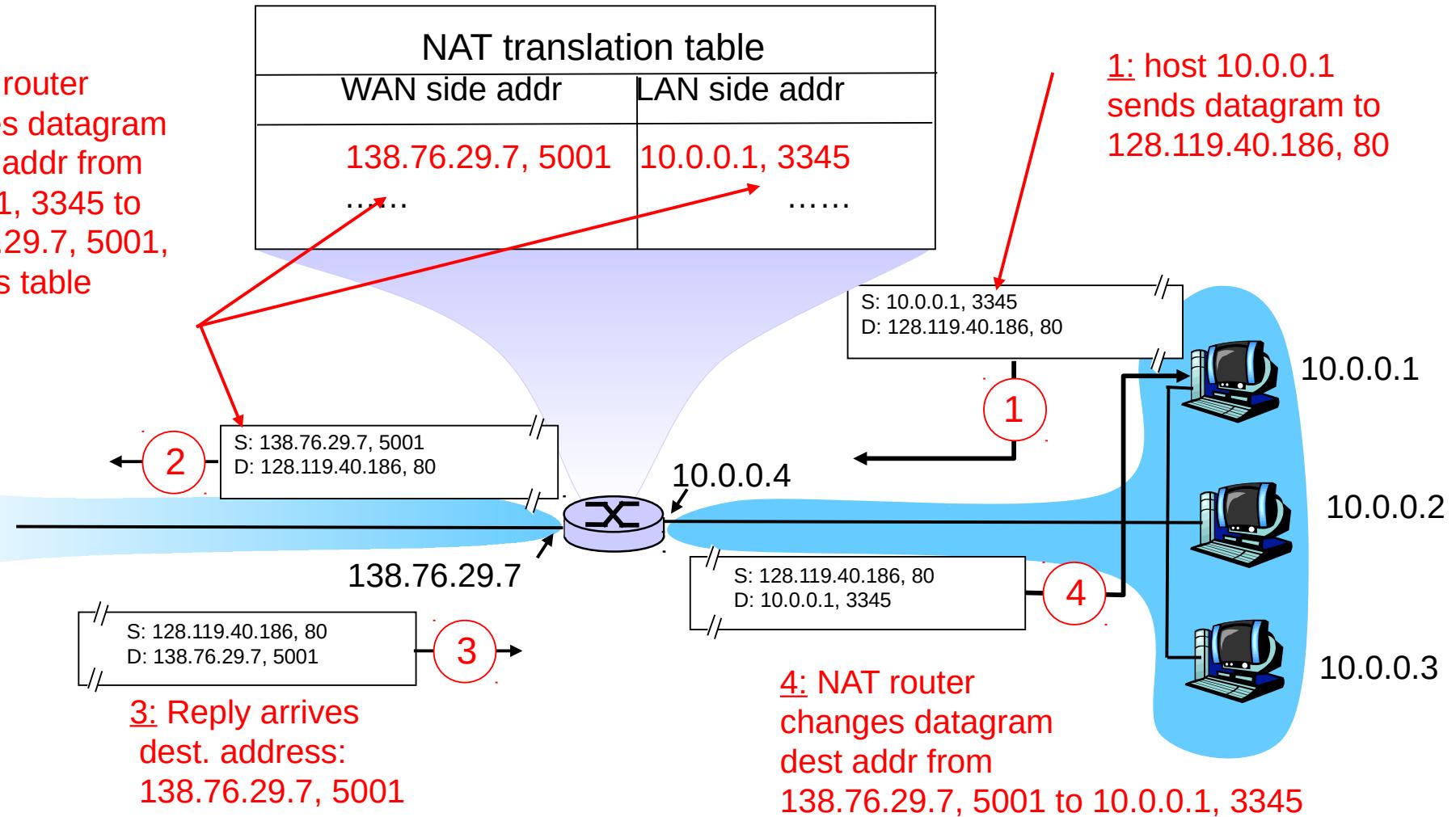
All datagrams *leaving* local network have *same* single source NAT IP address: 138.76.29.7, different source port numbers

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



# NAT: Network Address Translation II

2: NAT router changes datagram source addr from 10.0.0.1, 3345 to 138.76.29.7, 5001, updates table



# Maintaining the Mapping Table

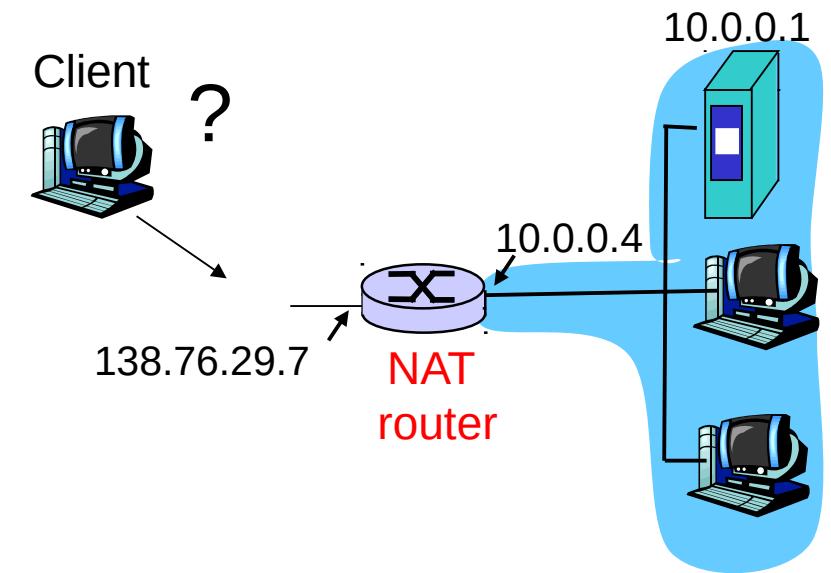
- Create an entry upon seeing an outgoing packet
  - Packet with new (source addr, source port) pair
- Eventually, need to delete entries to free up #'s
  - When? If no packets arrive before a timeout
  - (At risk of disrupting a temporarily idle connection)
- Yet another example of “soft state”
  - I.e., removing state if not refreshed for a while

Source: Freedman



# NAT Traversal Problem

- client wants to connect to server with address 10.0.0.1
  - server address 10.0.0.1 local to LAN (client can't use it as destination addr)
  - only one externally visible NATed address: 138.76.29.7



- How can we deal with incoming connections?
- How do we map to multiple services inside the NAT'ed subnet?

Source: Freedman



# Where is NAT Implemented?

- Home router (e.g., Linksys box)
  - Integrates router, DHCP server, NAT, etc.
  - Use single IP address from the service provider
- Campus or corporate network
  - NAT at the connection to the Internet
  - Share a collection of public IP addresses
  - Avoid complexity of renumbering hosts/routers when changing ISP (w/ provider-allocated IP prefix)

Source: Freedman



# NAT limitations and criticism

- 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, FTP
  - address shortage should instead be solved by IPv6

Source: Kurose & Ross (partial)



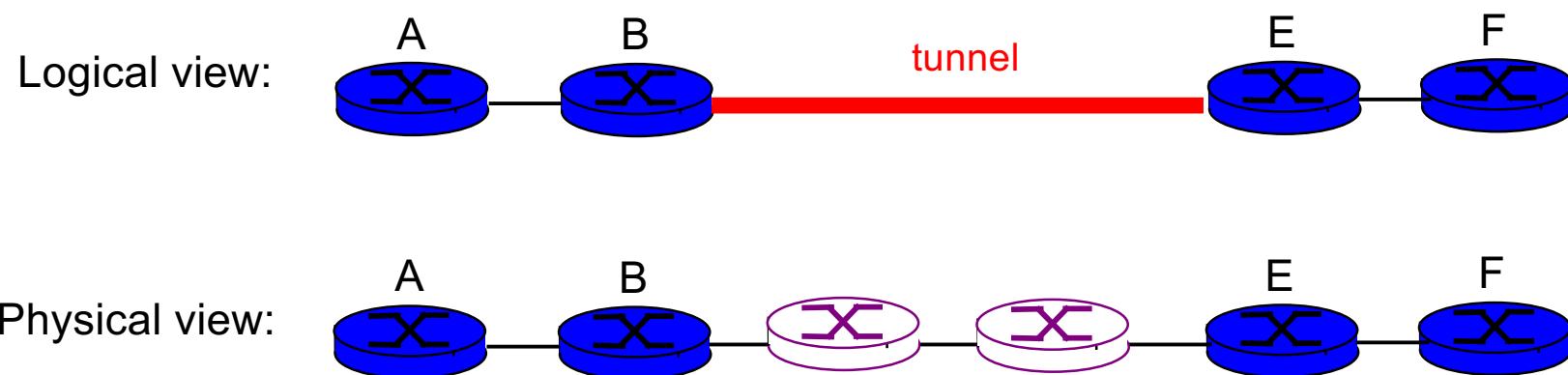
# Transition From IPv4 To IPv6

- Not all routers can be upgraded simultaneous
  - no “flag days”
  - How will the network operate with mixed IPv4 and IPv6 routers?
- *Tunneling*: IPv6 carried as payload in IPv4 datagram among IPv4 routers



# IP Tunneling to Build Overlay Links

- IP tunnel is a virtual point-to-point link
  - Illusion of a direct link between two separated nodes



- Encapsulation of the packet inside an IP datagram
  - Node B sends a packet to node E
  - ... containing another packet as the payload

Source: Freedman

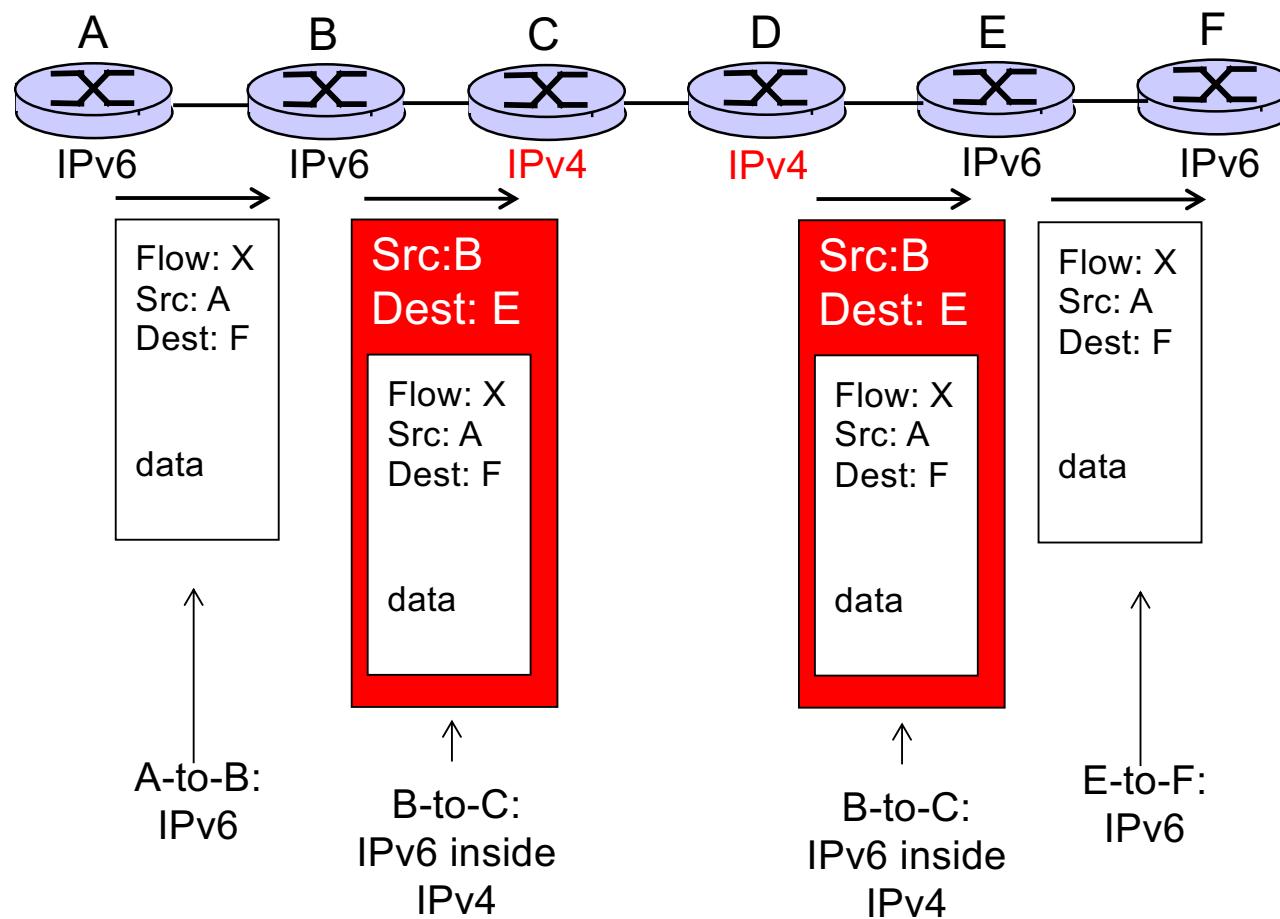


# IP Tunneling

Logical view:



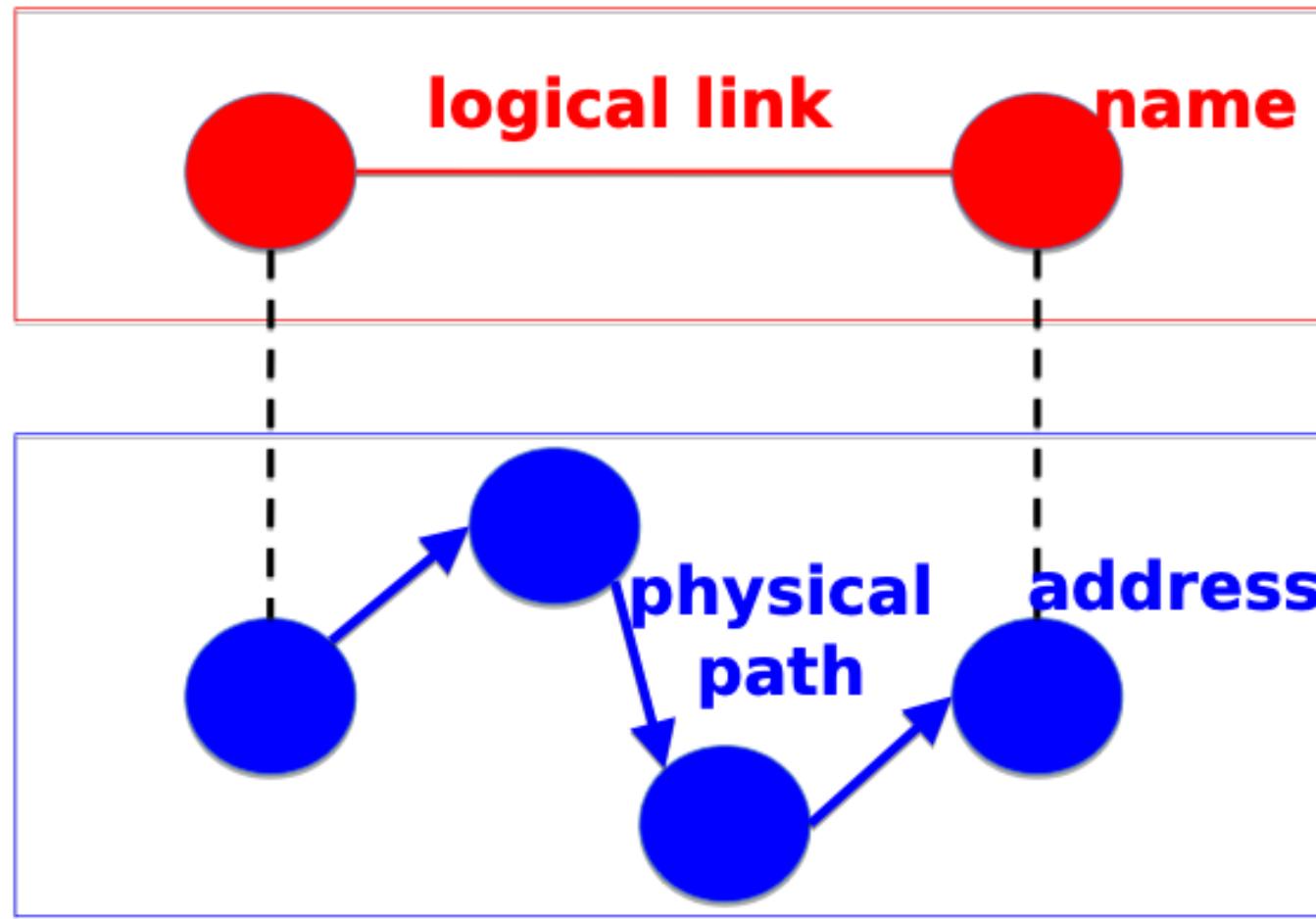
Physical view:



Source:  
Kurose &  
Ross



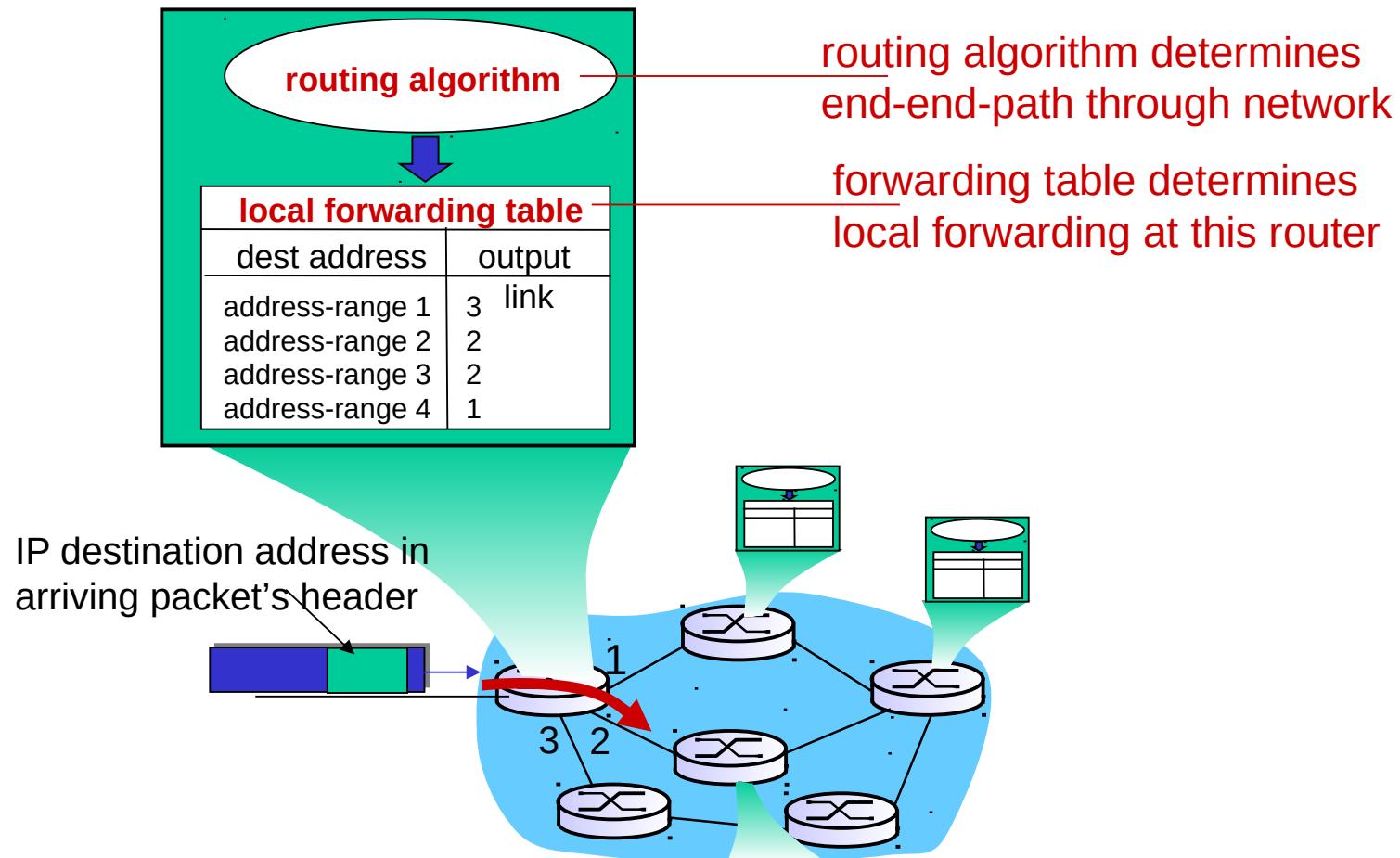
# Routing: Mapping Link to Path



Source:  
Freedman



# Interplay of routing and forwarding



# Three Issues to Address

- What does the protocol compute?
  - E.g., shortest paths
- What algorithm does the protocol run?
  - E.g., link-state routing
- How do routers learn end-host locations?
  - E.g., injecting into the routing protocol

Source: Freedman



# Routing Algorithm Classification

*Q: global or decentralized information?*

*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

*Q: static or dynamic?*

*static:*

- routes change slowly over time

*dynamic:*

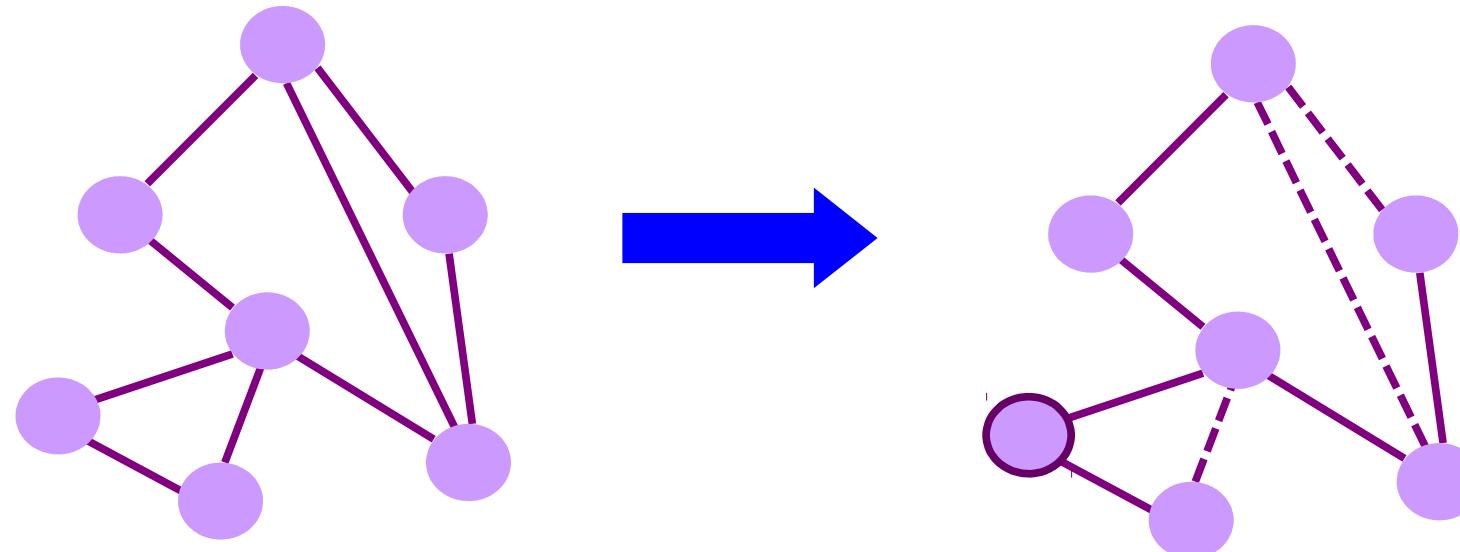
- routes change more quickly
  - periodic update
  - in response to link cost changes

Source: Kurose & Ross



# What to Compute ?

- Shortest path(s) between pairs of nodes
  - A shortest-path tree rooted at each node
  - Min hop count or min sum of edge weights

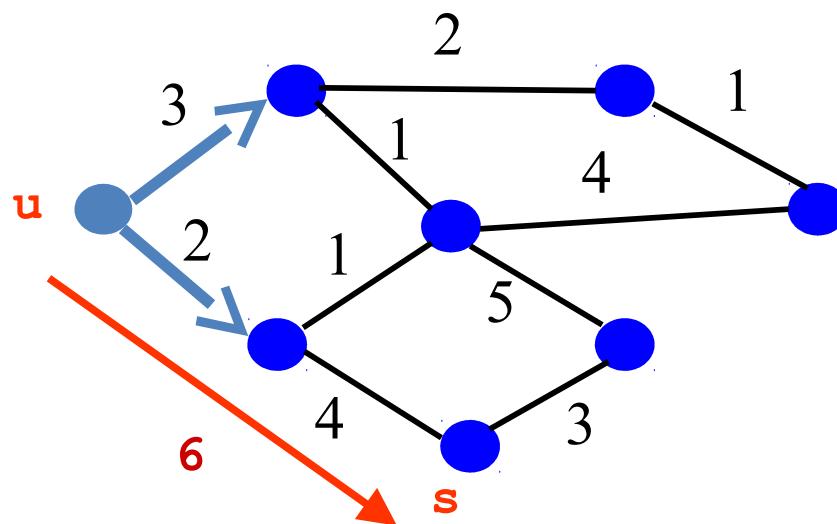


Source: Freedman



# Shortest Path Problem

- Compute: path costs to all nodes
  - From a given source  $u$  to all other nodes
  - Cost of the path through each outgoing link
  - Next hop along the least-cost path to  $s$



Source: Freedman



# Link State : Dijkstra's Algorithm

- Flood the topology information to all nodes
- Each node computes shortest paths to other nodes

## Initialization

```
S = {u}  
for all nodes v  
    if (v is adjacent to  
        u)  
        D(v) = c(u,v)  
    else D(v) = ∞
```

## Loop

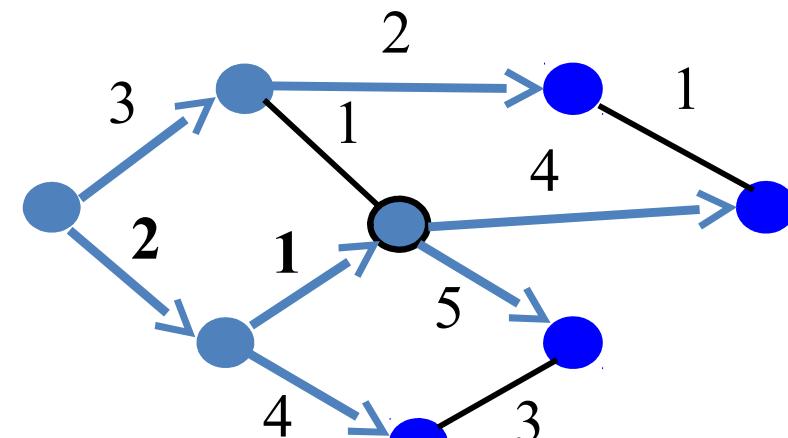
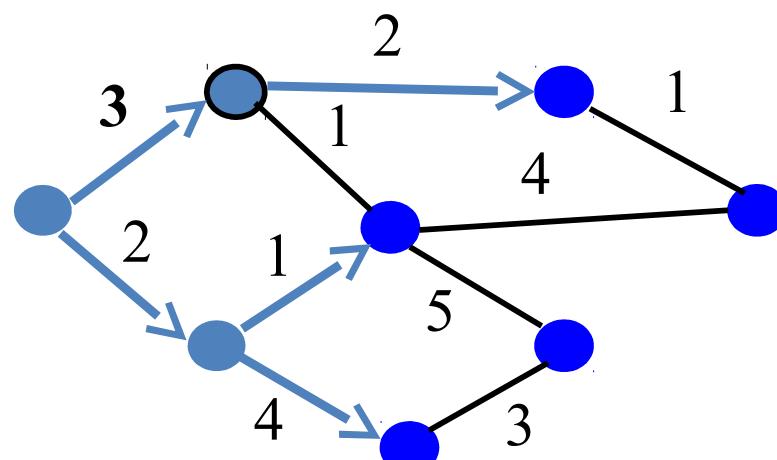
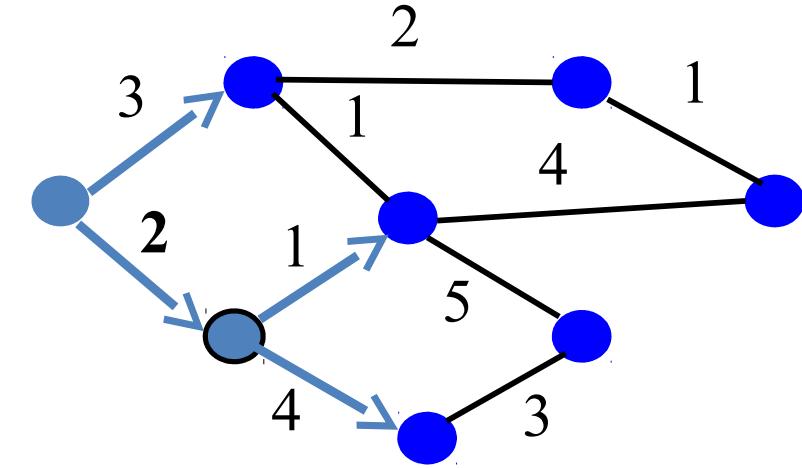
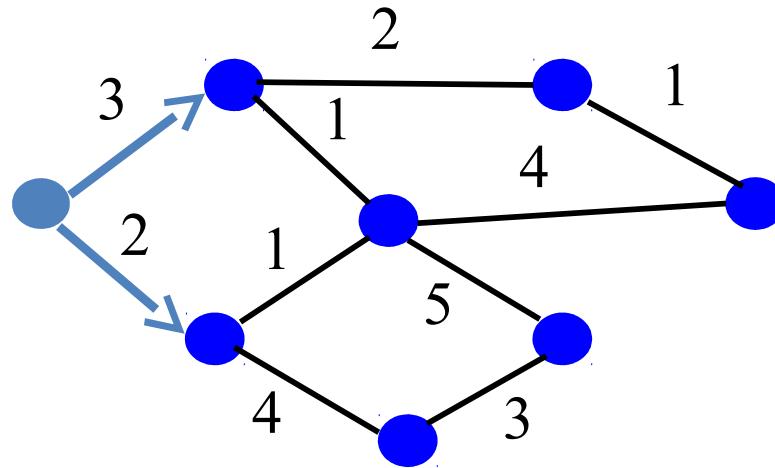
```
add w with smallest D(w) to S  
update D(v) for all adjacent v:  
    D(v) = min{D(v), D(w) +  
                c(w,v)}  
until all nodes are in S
```

## Used in OSPF and IS-IS

Source: Freedman



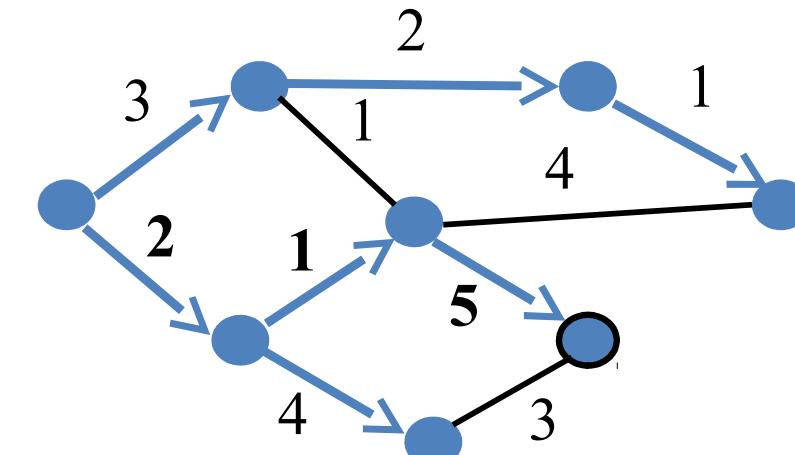
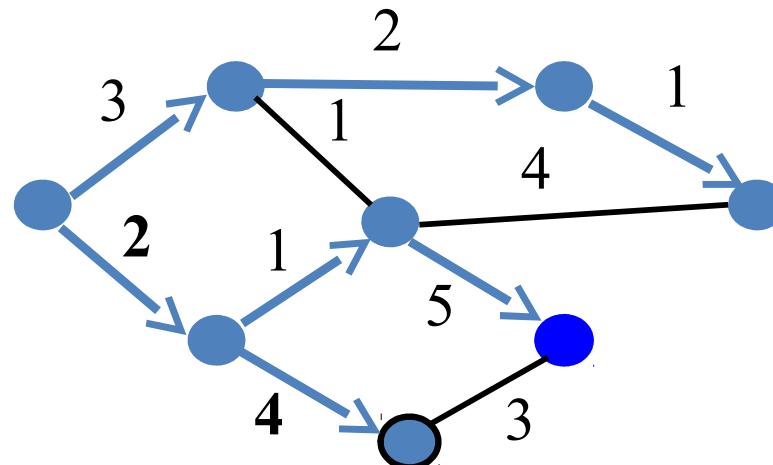
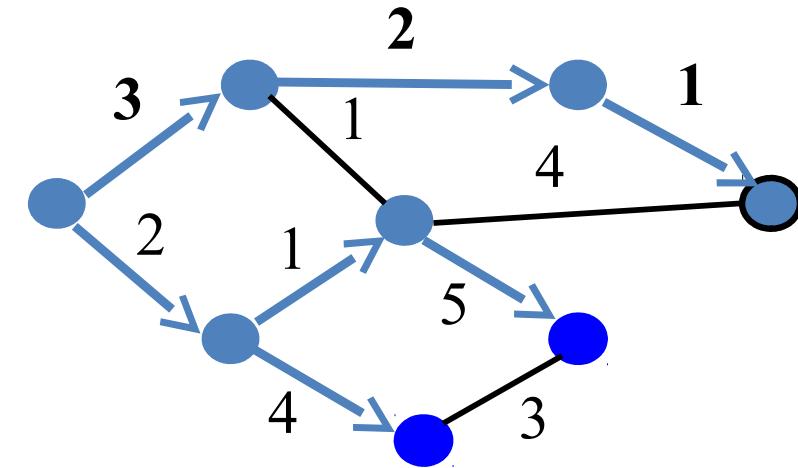
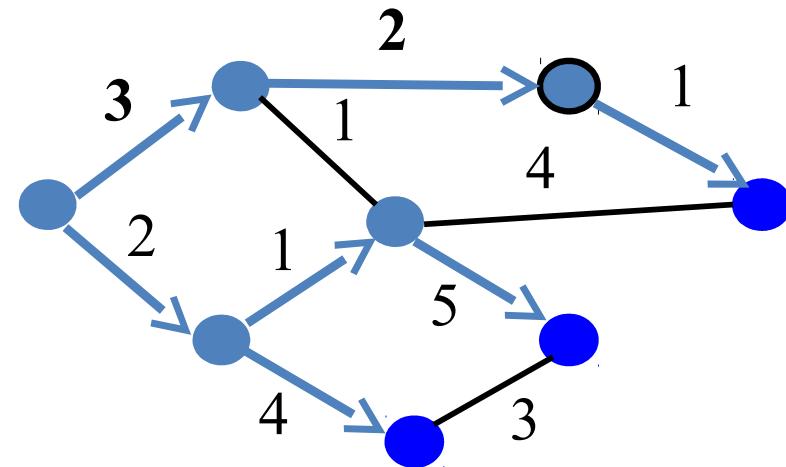
# Link State : Routing Example



Source: Freedman



# Link State : Routing Example (contd.)

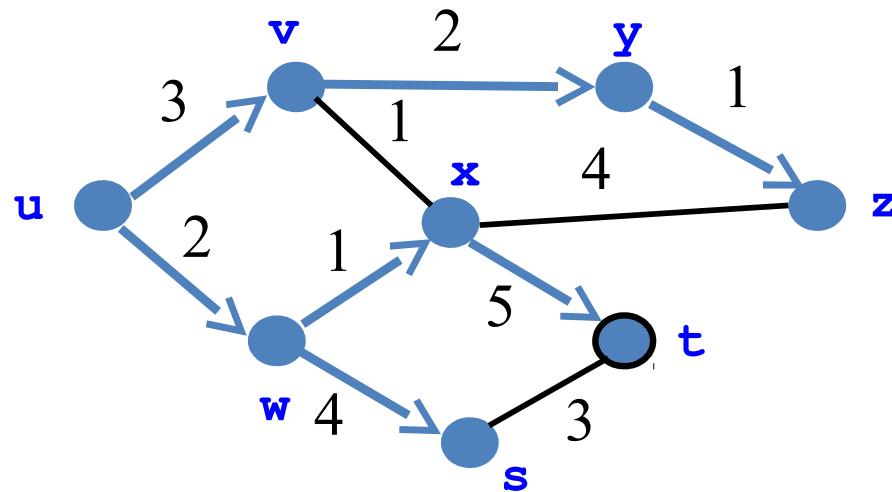


Source: Freedman



# Link State : Routing Example (contd.)

- Shortest-path tree from u
- Forwarding table at u



dest	link
v	(u,v)
w	(u,w)
x	(u,w)
y	(u,v)
z	(u,v)
s	(u,w)
t	(u,w)

Source: Freedman



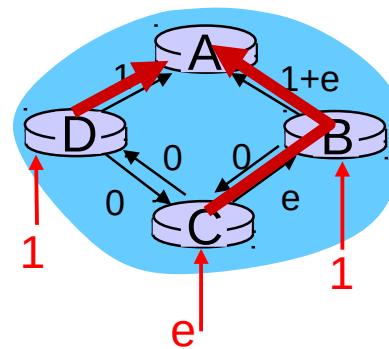
# Link State Algorithm Discussion

algorithm complexity:  $n$  nodes

- each iteration: need to check all nodes,  $w$ , not in  $N$
- $n(n+1)/2$  comparisons:  $O(n^2)$
- more efficient implementations possible:  $O(n \log n)$

oscillations possible:

- e.g., support link cost equals amount of carried traffic:

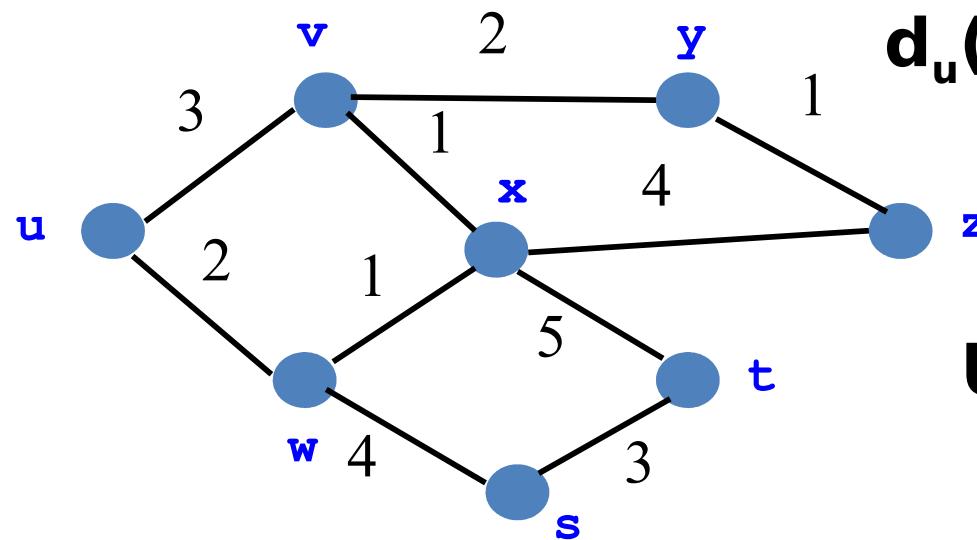


Source: Freedman



## Distance Vector : Bellman Ford Algorithm

- Define distances at each node  $x$ 
  - $d_x(y) = \text{cost of least-cost path from } x \text{ to } y$
- Update distances based on neighbors
  - $d_x(y) = \min \{ c(x,v) + d_v(y) \}$  over all neighbors  $v$



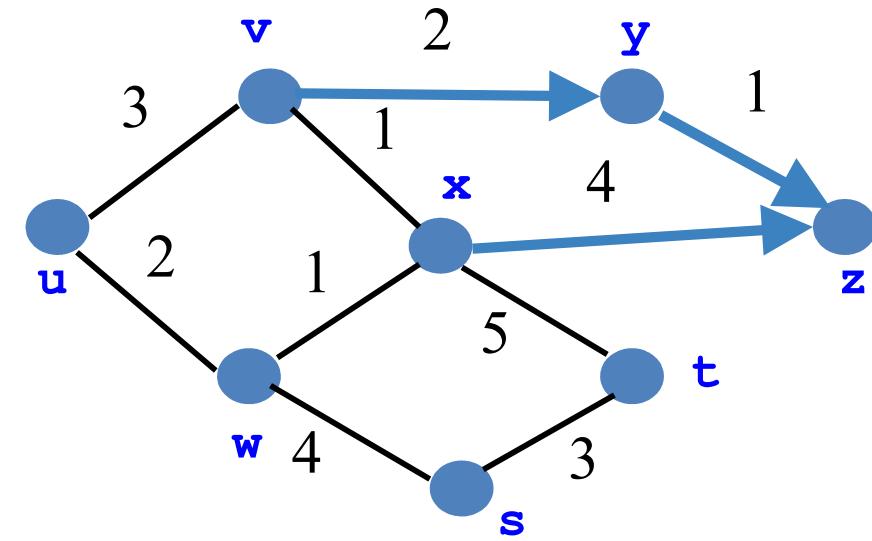
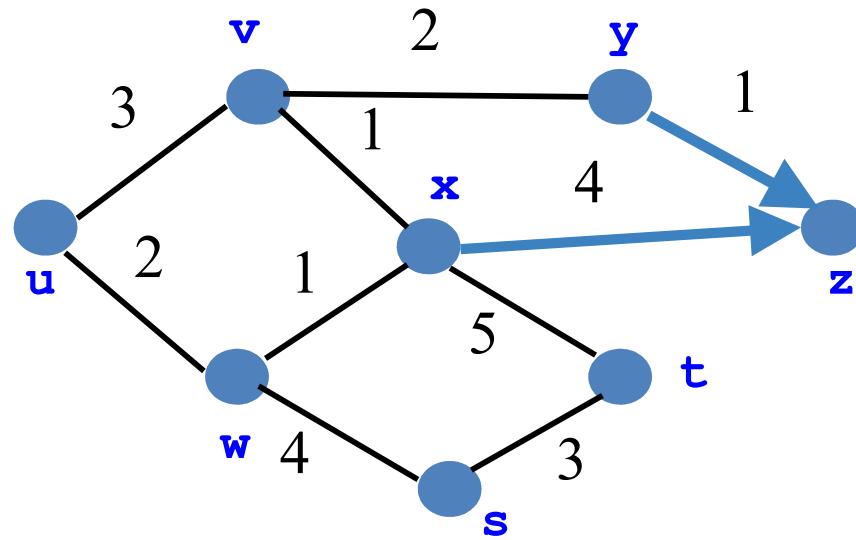
$$d_u(z) = \min \{ c(u,v) + d_v(z), c(u,w) + d_w(z) \}$$

**Used in RIP and EIGRP**

Source: Freedman



# Distance Vector : Routing Example



$$d_y(z) = 1$$

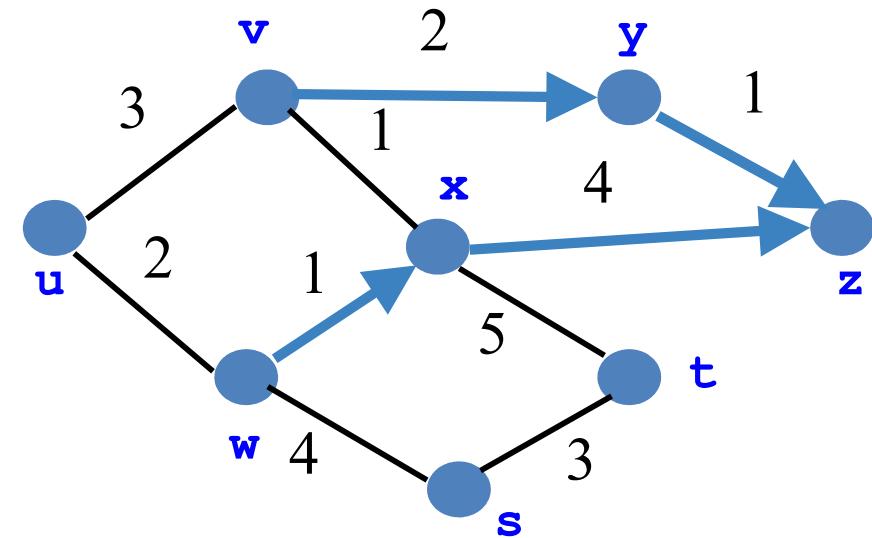
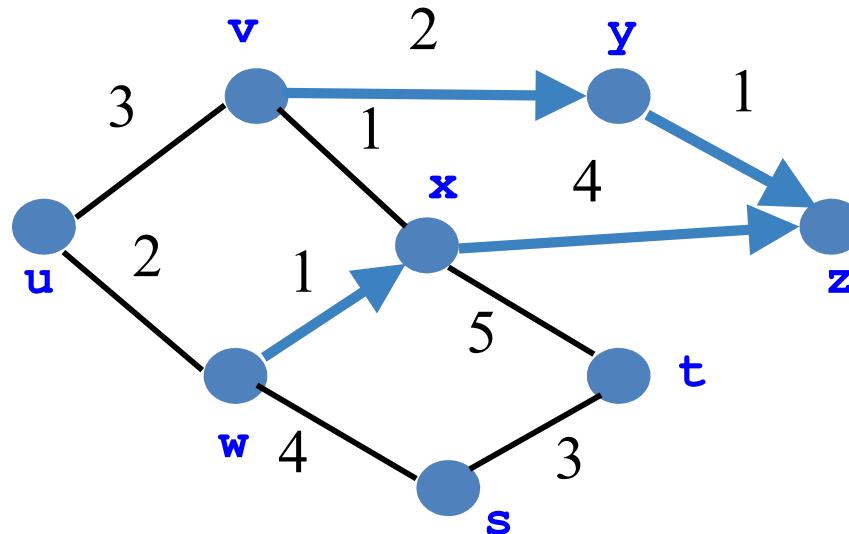
$$d_x(z) = 4$$

$$d_v(z) = \min\{ 2+d_y(z), 1+d_x(z) \}$$

$$= 3$$



## Distance Vector : Routing Example (contd.)

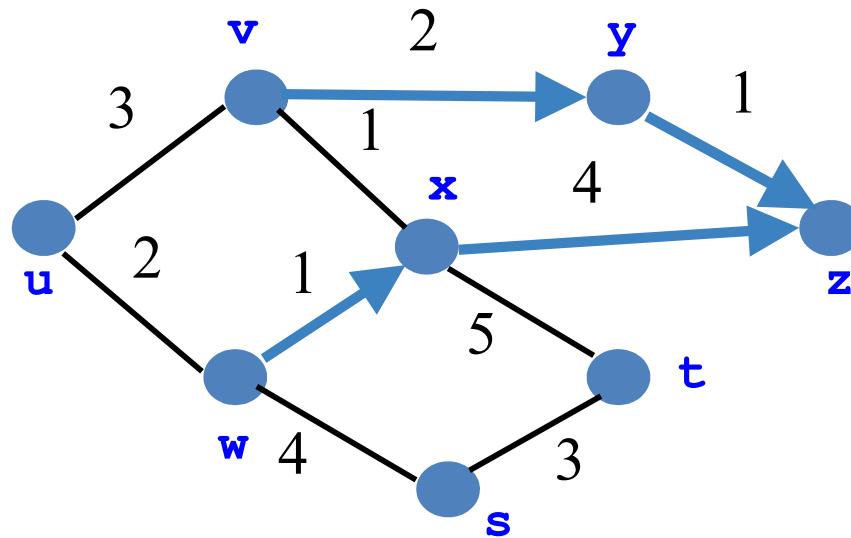


$$\begin{aligned}
 d_w(z) &= \min\{1+d_x(z), \\
 &\quad 4+d_s(z), \\
 &\quad 2+d_u(z) \} \\
 &= 5
 \end{aligned}$$

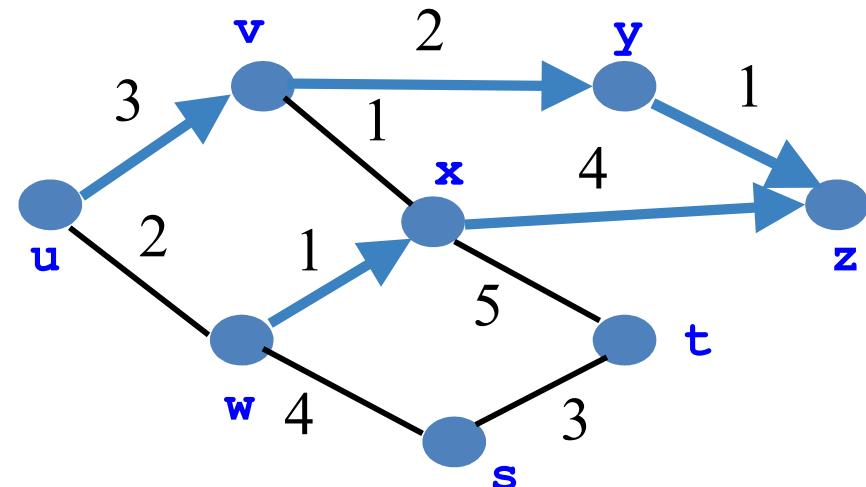
$$\begin{aligned}
 d_u(z) &= ?? \\
 &\text{(A) 5 } \text{(B) 6 } \text{(C) 7}
 \end{aligned}$$



## Distance Vector : Routing Example (contd.)



$$\begin{aligned}
 d_w(z) &= \min\{1+d_x(z), \\
 &\quad 4+d_s(z), \\
 &\quad 2+d_u(z)\} \\
 &= 5
 \end{aligned}$$



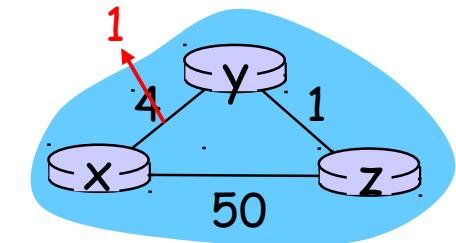
$$\begin{aligned}
 d_u(z) &= \min\{3+d_v(z), \\
 &\quad 2+d_w(z)\} \\
 &= 6
 \end{aligned}$$



# Distance Vector : Link Cost Changes

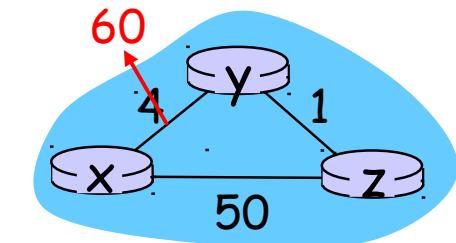
## *link cost changes:*

- Node detects local link cost change
- Updates routing info, recalculates distance vector
- If DV changes, notify neighbors
- **Good News travels fast**



## *link cost changes:*

- Node detects local link cost change
- ***Bad news travels slow*** - “count to infinity” problem!
- 44 iterations before algorithm stabilizes: see text
- **Poisoned Reverse for faster convergence.** Will this completely solve count to infinity problem?



Source: Kurose & Ross



# Distance Vector : Link Cost Changes

## *message complexity*

- **LS:** with  $n$  nodes,  $E$  links,  $O(nE)$  msgs sent
- **DV:** exchange between neighbors only
  - convergence time varies

## *speed of convergence*

- **LS:**  $O(n^2)$  algorithm requires  $O(nE)$  msgs
  - may have oscillations
- **DV:** convergence time varies
  - may be routing loops
  - count-to-infinity problem

*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
  - error propagate through network

Source: Kurose & Ross



# Routing Issues

## Our routing study thus far - idealization

- All routers identical
- Network “flat”  
... *not true in practice*

**Scale:** with 600 million destinations:

- Can't store all dest's in routing tables!
- Routing table exchange would swamp links!

## *Administrative autonomy*

- Internet = network of networks
- Each network admin may want to control routing in its own network

Source: Kurose & Ross



# Hierarchical Routing – Standard CS trick

- Aggregate routers into regions, “autonomous systems” (AS)
- Routers in same AS run same routing protocol
  - “Intra-AS” routing protocol
  - Routers in different AS can run different intra-AS routing protocol

## *Gateway router:*

- At “edge” of its own AS
- Has link to router in another AS

Source: Kurose & Ross



# Summary

- DHCP: bootstrapping IP addresses
  - Broadcasting, caching, soft state
- NAT
  - A hack! ☺ Reading and reflection: Why?
- Many other hacks too! ☺
  - Tunneling, firewalls, mobile gateways, VPNs
- Routing Algorithms are graph based
  - Centralized → Link State
  - Distributed → Distance Vector
- Routing algorithms have different characteristics

