

CSCE 463/612

Networks and Distributed Processing

Spring 2017

Network Layer III

Dmitri Loguinov

Texas A&M University

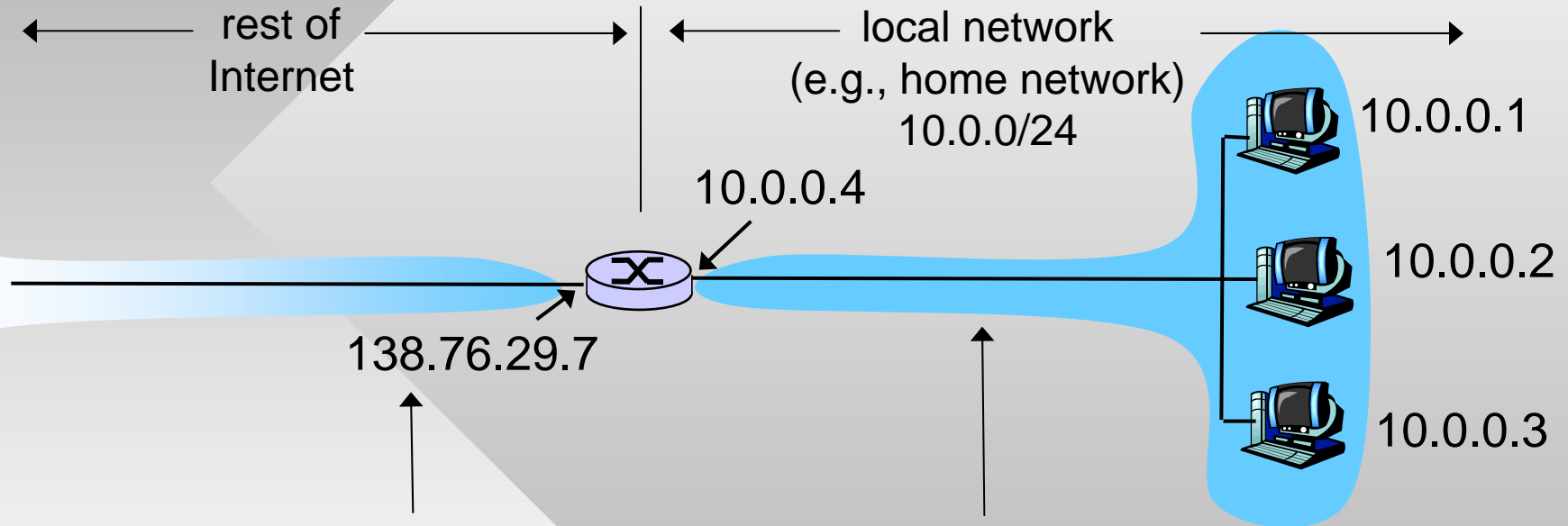
April 11, 2017

Original slides copyright © 1996-2004 J.F Kurose and K.W. Ross

Homework #4 Grading

- Default mode: final grading will use 3 homeworks
 - Homework contribution = $(hw1+hw2+hw3) / 3$
- **Extra-credit option A**: use hw4 in place of any previous homework
 - Swapping out hw1, we get $(hw4+hw2+hw3) / 3$
- **Extra-credit option B**: add 20% of hw4 to other homeworks
 - $(hw1 + hw2 + hw3 + 0.2*hw4) / 3$
- Example: hw1 = 21, hw2 = 80, hw3 = 70, hw4 = 60
 - Default = 57, option A = 70, option B = 61
- Example: hw1 = 90, hw2 = 102, hw3 = 90, hw4 = 90
 - Default = option A = 94, option B = 100

NAT: Network Address Translation



All datagrams *leaving* local network have the *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

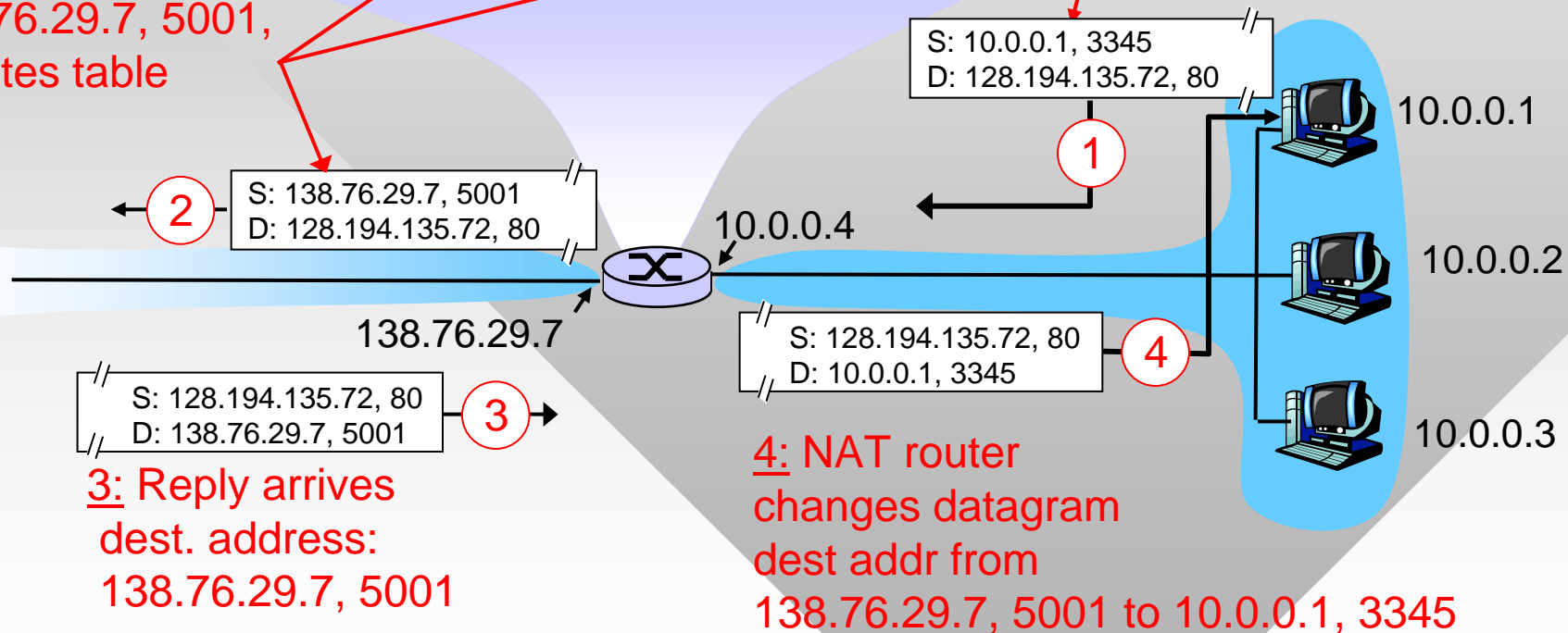
- Local network uses just one IP address as far as the outside world is concerned
 - No need to be allocated a range of addresses from ISP – just one IP address is used for all devices
 - Can change addresses of devices in local network without notifying outside world
 - Can change ISP without changing addresses of devices in local network
 - Devices inside local net not explicitly addressable or visible to outside world (a security plus)
- To see your NAT IP and current NAT port, visit <http://ipchicken.com/>

NAT: Network Address Translation

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

NAT translation table	
WAN side addr	LAN side addr
138.76.29.7, 5001	10.0.0.1, 3345
.....

1: host 10.0.0.1 sends datagram to 128.194.135.72, 80



3: Reply arrives
dest. address:
138.76.29.7, 5001

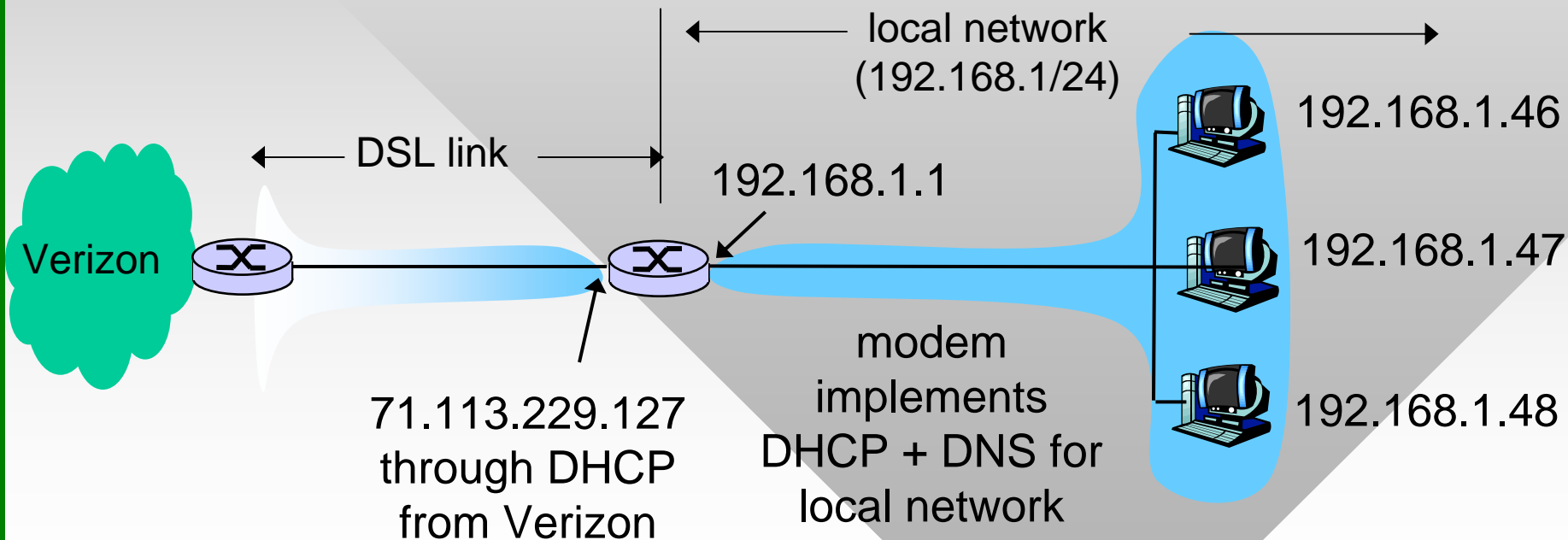
4: NAT router changes datagram dest addr from 138.76.29.7, 5001 to 10.0.0.1, 3345

NAT: Network Address Translation

- 16-bit port-number field
 - Up to 64K simultaneous connections with a single LAN-side address
- NAT is controversial:
 - Routers should only process up to layer 3
 - Violates the end-to-end argument
- Makes inbound connections difficult
 - Inbound connections needed in P2P and other applications
 - May be overcome by UPnP or manually configuring NAT to route incoming connections to a particular host
- Some believe that address shortage should instead be solved by IPv6

NAT: Network Address Translation

- Private networks: 10.0.0.0/8, 192.168.0.0/16, 172.16.0.0/12
- Example: Verizon DSL with a Westell DSL modem



Chapter 4: Roadmap

4.1 Introduction

4.2 Virtual circuit and datagram networks

4.3 What's inside a router

4.4 IP: Internet Protocol

- Datagram format
- IPv4 addressing
- **ICMP**
- IPv6

4.5 Routing algorithms

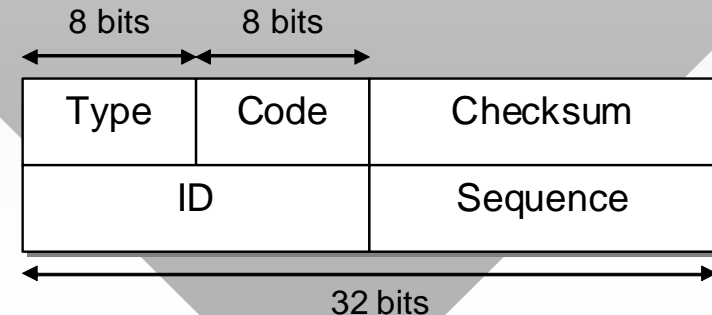
4.6 Routing in the Internet

4.7 Broadcast and multicast routing

ICMP: Internet Control Message Protocol

- Communicates network-level debug information
 - Error reporting: unreachable host, network, port, protocol
 - Echo request/reply (ping)
- Network-layer above IP
 - ICMP msgs carried in IP datagrams (“layer 3.5”)
- **ICMP reply message**
 - Payload contains first 28 bytes of IP pkt causing error

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
4	0	source quench (congestion control - not used)
8	0	echo request (ping)
9	0	router advertisement
10	0	router discovery
11	0	TTL expired
12	0	bad IP header



Traceroute and ICMP

- Source sends series of **UDP** segments to dest
 - First with TTL = 1
 - Second with TTL = 2
 - **Unlikely** port number
 - When the n -th datagram arrives to the n -th router:
 - Router discards datagram
 - Sends to source a TTL Expired (type 11, code 0)
 - Message includes IP hdr from router & first 28 bytes of original packet
 - When ICMP message arrives, source calculates RTT
 - Traceroute does this 3 times per hop
- Stopping criterion
- UDP segment eventually arrives at destination host
 - Destination returns ICMP “port unreachable” packet (type 3, code 3)
 - When source gets this ICMP, it stops

Chapter 4: Roadmap

4.1 Introduction

4.2 Virtual circuit and datagram networks

4.3 What's inside a router

4.4 IP: Internet Protocol

- Datagram format
- IPv4 addressing
- ICMP
- **IPv6**

4.5 Routing algorithms

4.6 Routing in the Internet

4.7 Broadcast and multicast routing

IPv6

16-byte IP, e.g.,

FEBC:A574:382B:23C1:AA49:4592:4EFE:9982

- Initial motivation: 32-bit address space has been completely allocated
- Additional motivation:
 - Simpler header format helps speed up forwarding
 - Header changes to facilitate QoS and extensions

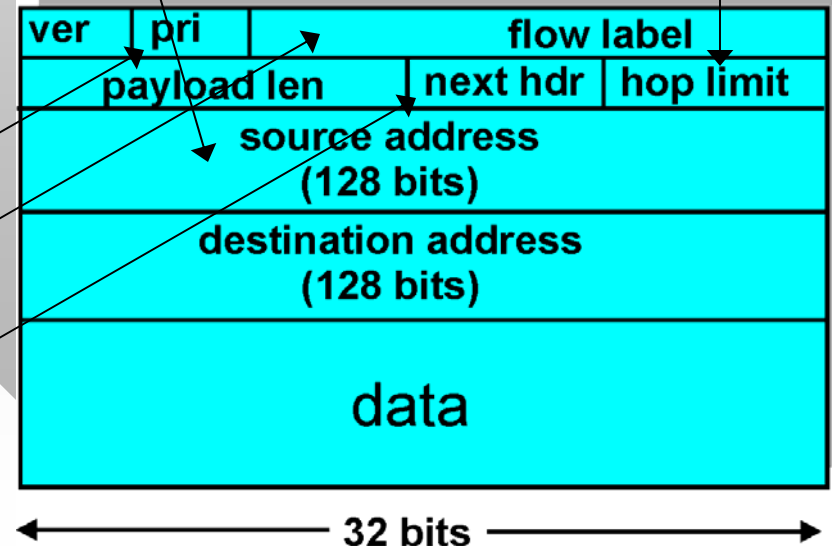
IPv6 datagram format:

- Fixed-length 40 byte header
- No fragmentation allowed

priority of packet (QoS)

flow ID (not well defined)

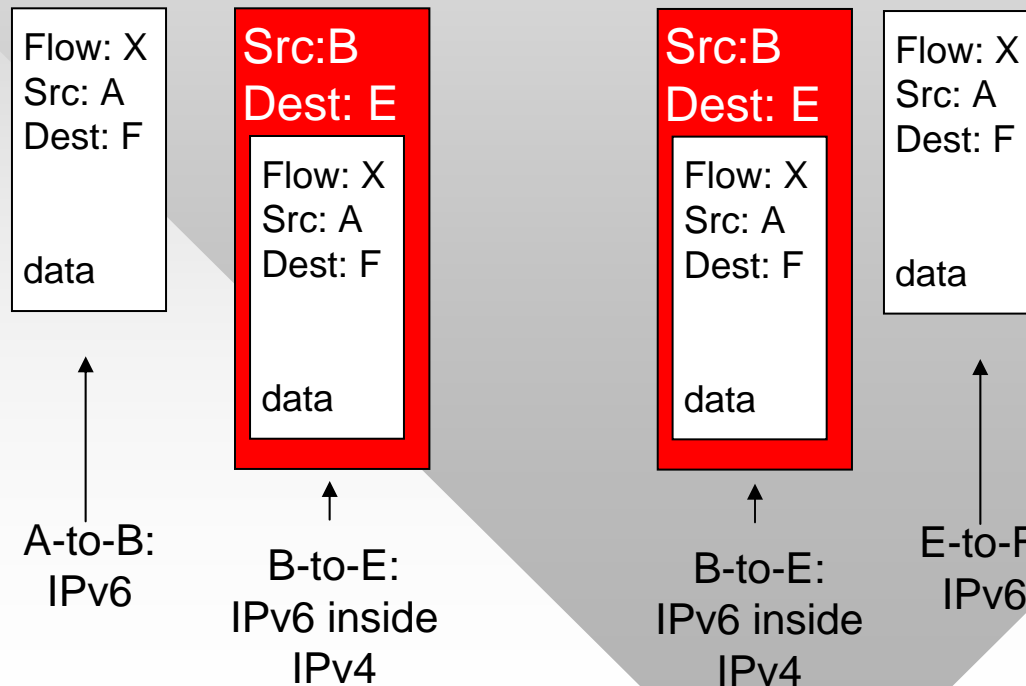
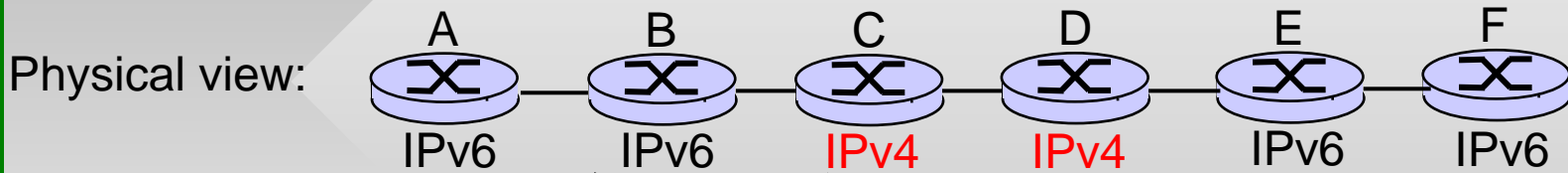
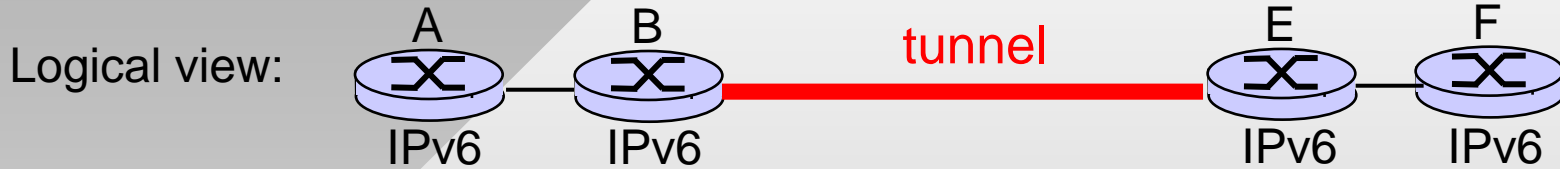
upper-layer protocol
(e.g., TCP, ICMP) or
IPv6 extension header



IPv6 Notes

- *Checksum*: removed entirely to reduce processing time at each hop
 - Recall that IPv4 checksums the header only (TCP/UDP checksum the entire packet)
- *Options*: allowed, but outside of header, indicated by “Next Header” field
- All routers cannot be upgraded simultaneously
 - How will the network operate with mixed IPv4 / IPv6 routers?
- *Tunneling*: IPv6 carried as payload in IPv4 datagram among IPv4 routers

Tunneling



Q: how does E know the packet has encapsulated IPv6 data?

A: protocol field (often 41)

Chapter 4: Roadmap

4.1 Introduction

4.2 Virtual circuit and datagram networks

4.3 What's inside a router

4.4 IP: Internet Protocol

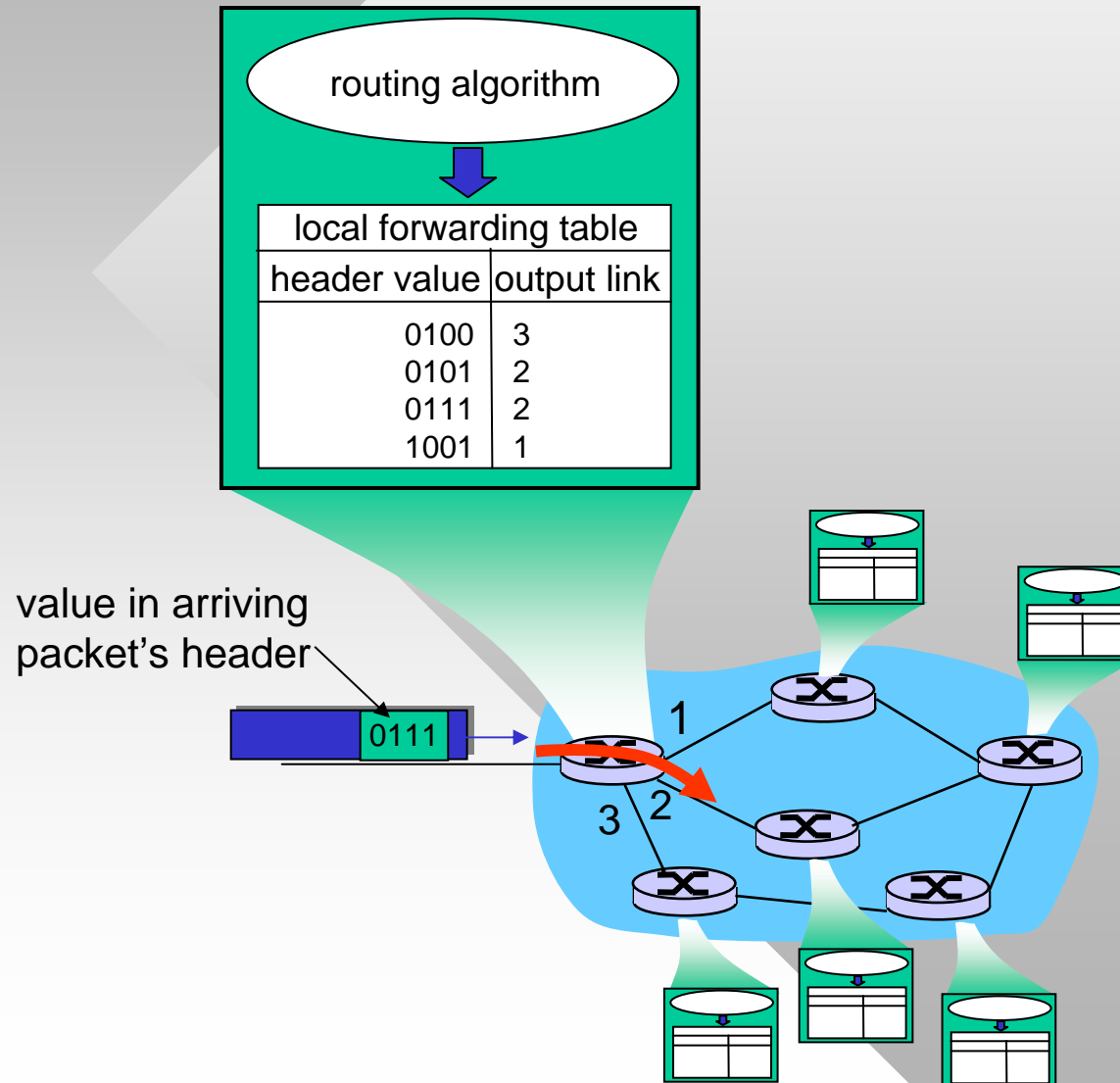
4.5 Routing algorithms

- Link state
- Distance Vector
- Hierarchical routing

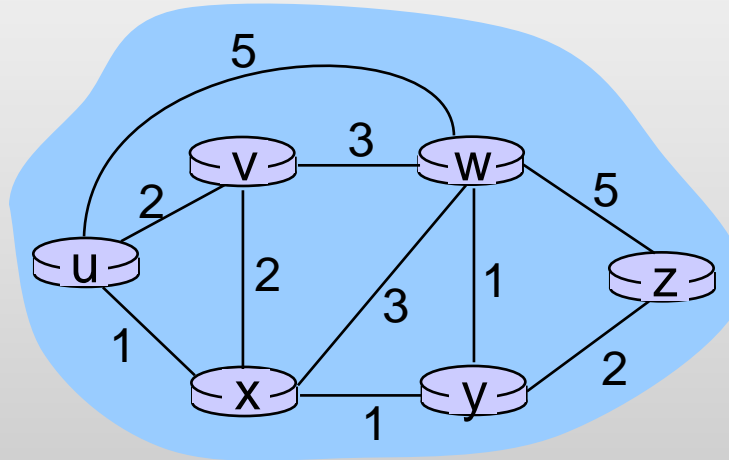
4.6 Routing in the Internet

4.7 Broadcast and multicast routing

Interplay Between Routing and Forwarding



Graph Abstraction



Graph: $G = (V, E)$

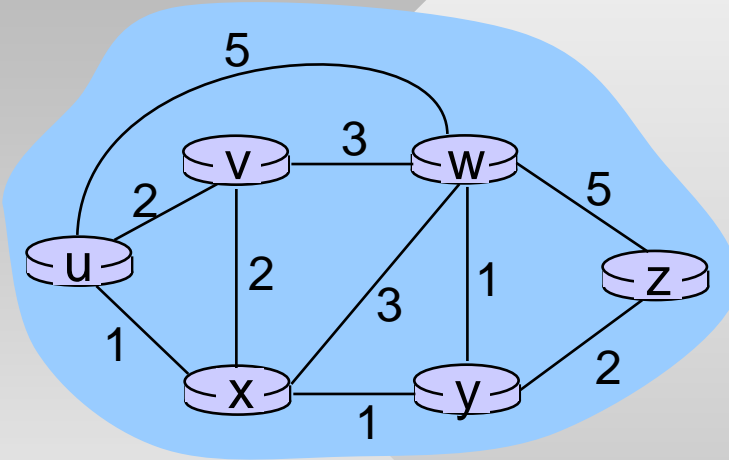
$V =$ set of routers $= \{u, v, w, x, y, z\}$

$E =$ set of links $= \{ (u,v), (u,x), (u,w), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) \}$

Remark: Graph abstraction is useful in other network contexts

Example: P2P, where V is set of peers and E is set of TCP connections

Graph Abstraction: Costs



- $c(x,y)$ = cost of link (x,y)
 - E.g., $c(w,z) = 5$
- Cost options:
 - Could always be 1
 - Could be inversely related to bandwidth or be proportional to congestion
 - Physical distance

Cost of path $(x_1, x_2, x_3, \dots, x_p) = c(x_1, x_2) + c(x_2, x_3) + \dots + c(x_{p-1}, x_p)$

Question: What's the least-cost path between u and z ?

Routing algorithm: algorithm that finds least-cost path

Routing Algorithm Classification

Global or local information?

- Global:
 - Routers have complete topology, link cost info
 - “Link state” algorithms
- Local (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:
 - Useful when routes change slowly over time
 - Manual or DHCP-based route creation
- Dynamic:
 - Routes change more quickly
 - Periodic update in response to link cost changes

Chapter 4: Roadmap

4.1 Introduction

4.2 Virtual circuit and datagram networks

4.3 What's inside a router

4.4 IP: Internet Protocol

4.5 Routing algorithms

- Link state
- Distance Vector
- Hierarchical routing

4.6 Routing in the Internet

4.7 Broadcast and multicast routing

Simple Link-State Routing Algorithm

Dijkstra's algorithm

- Entire network topology and link costs known
 - Accomplished via “link state broadcast”
 - Eventually, all nodes have same info
- Computes least cost paths from one node (“source”) to all other nodes
 - Gives **forwarding table** for that node
- **Iterative**: after k iterations, know least-cost path to k closest destinations

Notation:

- $c(x,y)$: link cost from x to y
 - Cost is ∞ if not direct neighbors
- $D(v)$: current **estimate** of the cost from source to destination v
- $p(v)$: predecessor of v along the least-cost path back to source
- F : set of closest nodes whose least-cost path has been finalized (i.e., known for a fact)

Dijkstra's Algorithm

Initialization:

$$F = \{u\}, D(u) = 0$$

for all nodes $v \neq u$

if v is adjacent to u

$$D(v) = c(u,v)$$

else

$$D(v) = \infty$$

do {

find node i not in F such that $D(i)$ is minimum

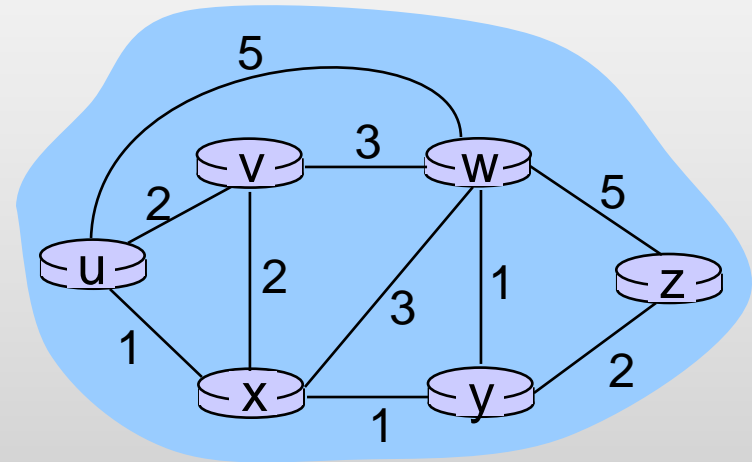
add i to F

for all j adjacent to i and not in F :

$$D(j) = \min(D(j), D(i) + c(i,j))$$

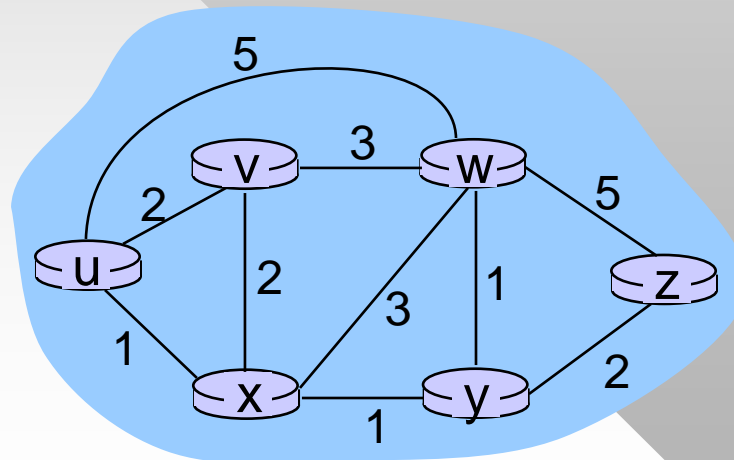
/ new cost to j is either old cost to j or known shortest path cost to i plus cost from i to j */*

} while (not all nodes in F)



Dijkstra's Algorithm: Example

Step	F	$D(v),p(v)$	$D(w),p(w)$	$D(x),p(x)$	$D(y),p(y)$	$D(z),p(z)$
0	u	$2,u$	$5,u$	$1,u$	∞	∞
1	ux	$2,u$	$4,x$		$2,x$	∞
2	uxy	$2,u$	$3,y$			$4,y$
3	$uxyv$		$3,y$			$4,y$
4	$uxyvw$					$4,y$
5	$uxyvwz$					



Dijkstra's Algorithm Discussion

Algorithm complexity: n nodes

- Iteration k : need to find min of $(n-k)$ costs
- Total: $n(n-1)/2$ comparisons, $O(n^2)$ complexity
- Heap-based implementation: $O(n \log n)$

Oscillations possible, but only for traffic-dependent cost:

- e.g., Link cost = amount of carried traffic

