

Computer Networking

Lent Term M/W/F 11-midday
LT1 in Gates Building

Slide Set 3

Andrew W. Moore

andrew.moore@cl.cam.ac.uk

January 2014

Topic 4: Network Layer

Our goals:

- understand principles behind network layer services:
 - network layer service models
 - forwarding versus routing (versus switching)
 - how a router works
 - routing (path selection)
 - IPv6
- For the most part, the Internet is our example – again.

Name: a *something*

Address: Where a *something* is

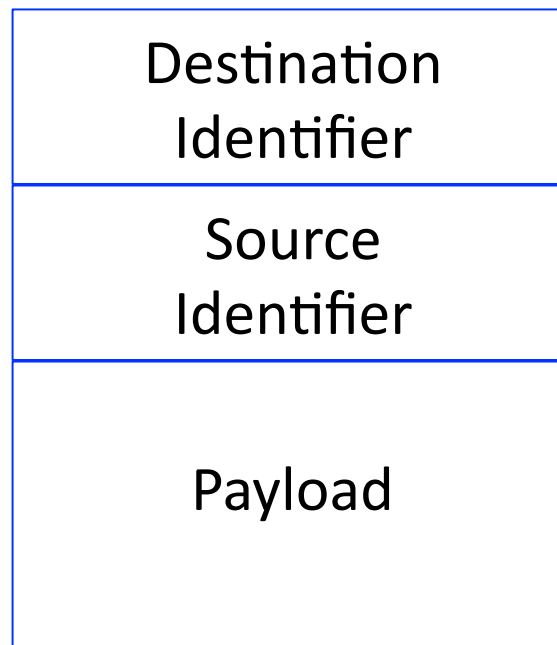
Routing: How do I get to the
something

Addressing (at a conceptual level)

- Assume all hosts have unique IDs
- No particular structure to those IDs
- Later in topic I will talk about real IP addressing
- Do I route on location or identifier?
- If a host moves, should its address change?
 - If not, how can you build scalable Internet?
 - If so, then what good is an address for identification?

Packets (at a conceptual level)

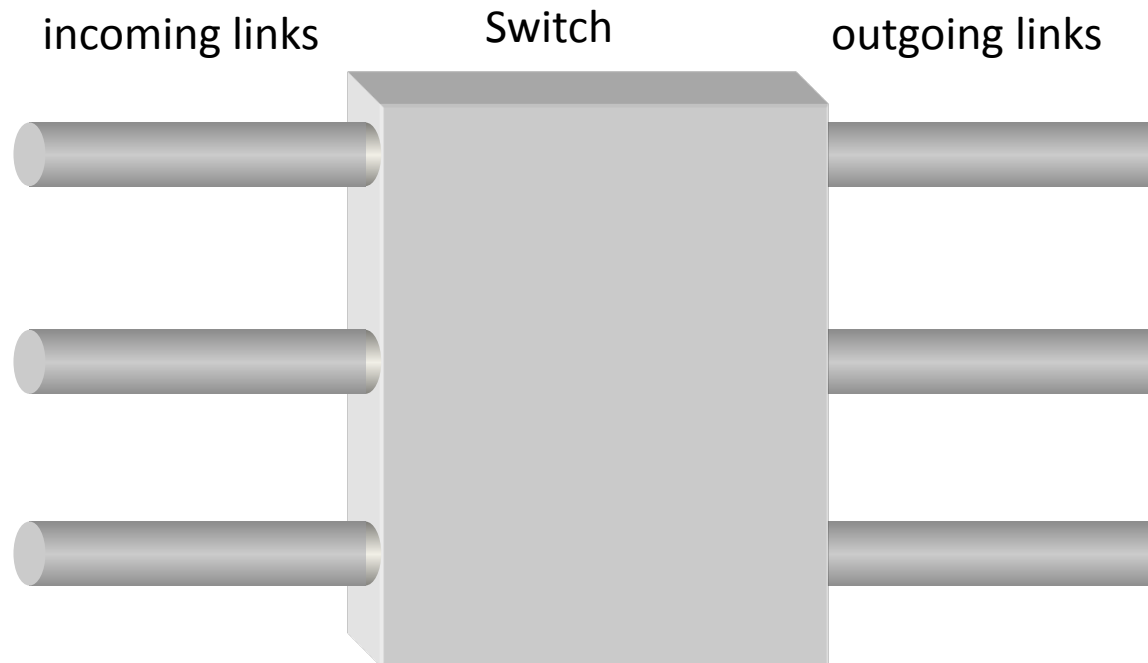
- Assume packet headers contain:
 - Source ID, Destination ID, and perhaps other information



Why include this?

Switches/Routers

- Multiple ports (attached to other switches or hosts)

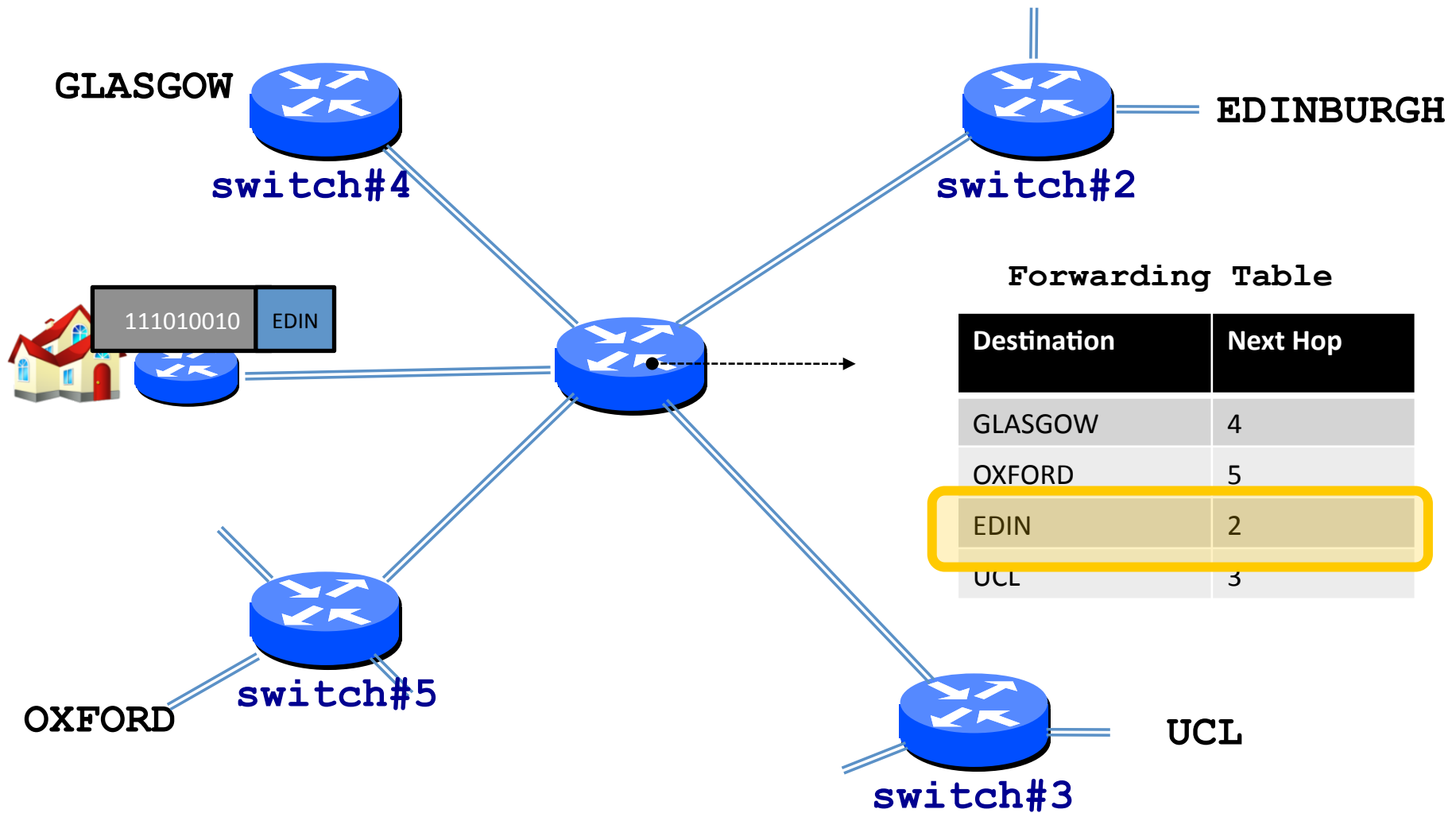


- Ports are typically duplex (incoming and outgoing)

A Variety of Networks

- ISPs: carriers
 - Backbone
 - Edge
 - Border (to other ISPs)
- Enterprises: companies, universities
 - Core
 - Edge
 - Border (to outside)
- Datacenters: massive collections of machines
 - Top-of-Rack
 - Aggregation and Core
 - Border (to outside)

Switches forward packets



Forwarding Decisions

- When packet arrives..
 - Must decide which outgoing port to use
 - In single transmission time
 - Forwarding decisions must be simple
- Routing state dictates where to forward packets
 - Assume decisions are **deterministic**
- *Global routing state* means collection of routing state in each of the routers
 - Will focus on where this routing state comes from
 - But first, a few preliminaries....

Forwarding vs Routing

- Forwarding: “data plane”
 - Directing a data packet to an outgoing link
 - Individual router using routing state
- Routing: “control plane”
 - Computing paths the packets will follow
 - Routers talking amongst themselves
 - Jointly creating the routing state
- Two very different timescales....

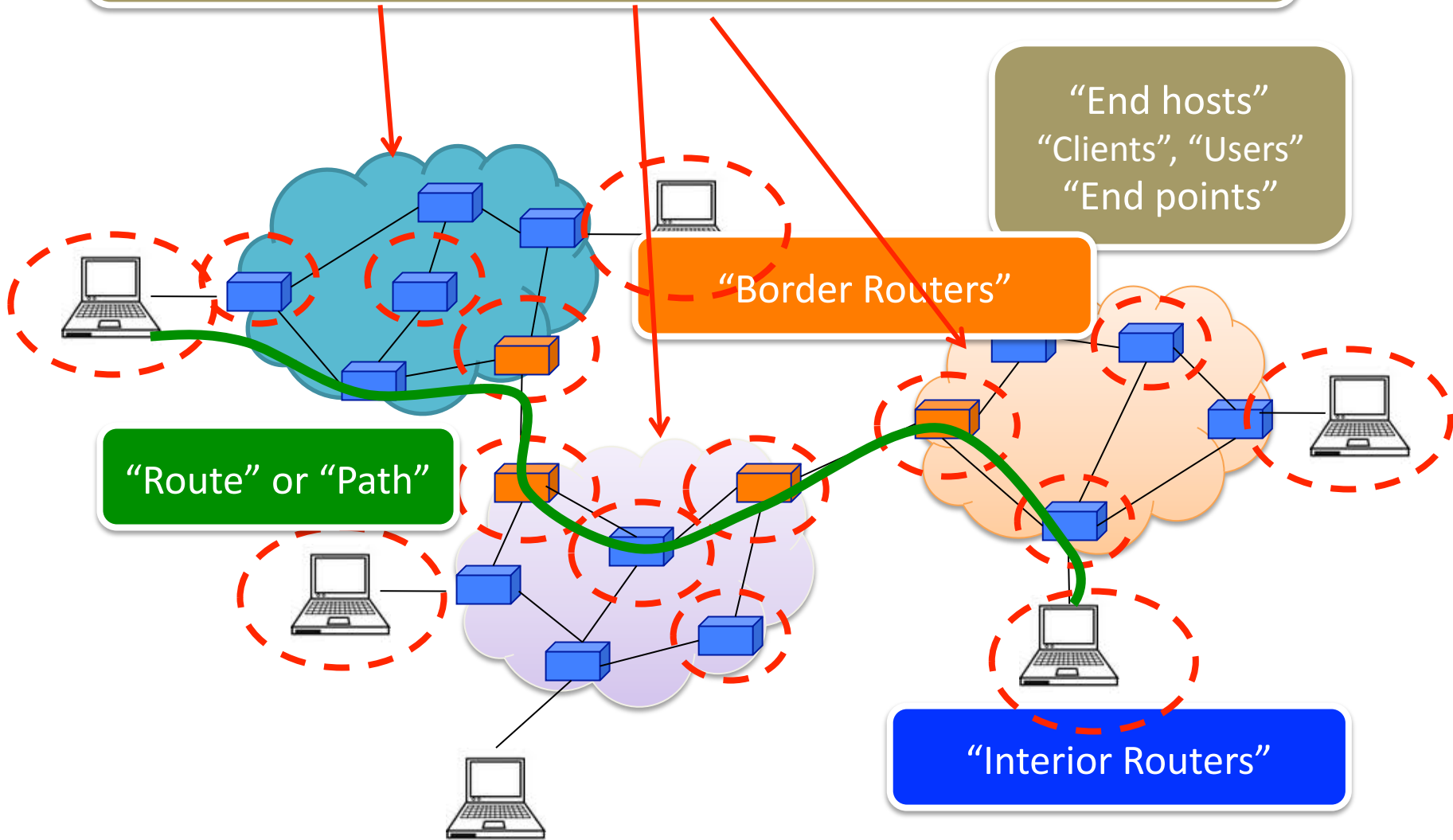
“Autonomous System (AS)” or “Domain”
Region of a network under a single administrative entity

“End hosts”
“Clients”, “Users”
“End points”

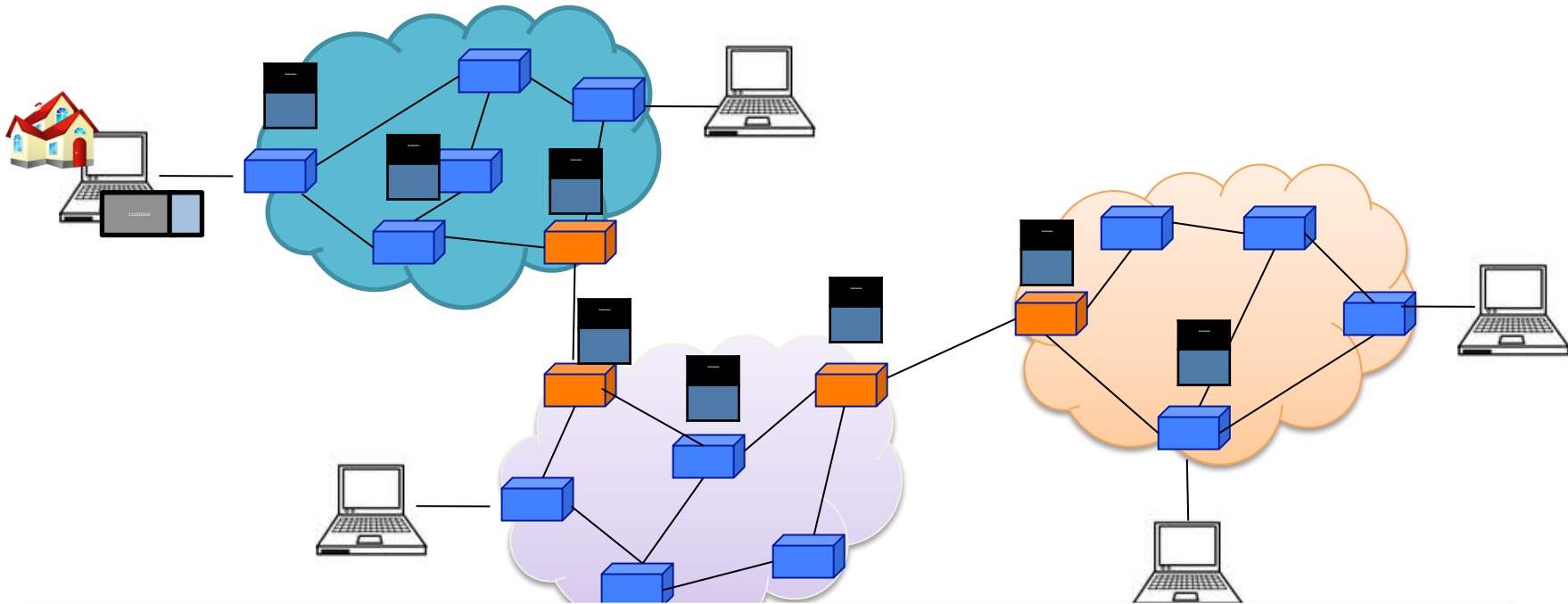
“Border Routers”

“Route” or “Path”

“Interior Routers”



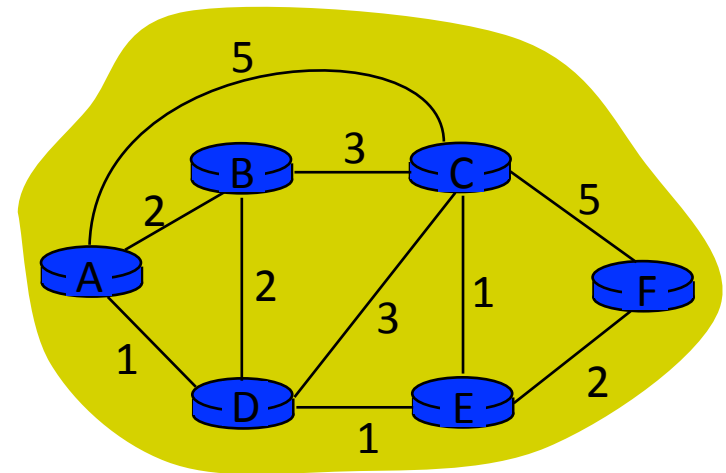
Context and Terminology



Internet routing protocols are responsible for constructing and updating the forwarding tables at routers

Routing Protocols

- Routing protocols implement the core function of a network
 - Establish paths between nodes
 - Part of the network’s “control plane”
- Network modeled as a graph
 - Routers are graph vertices
 - Links are edges
 - Edges have an associated “cost”
 - e.g., distance, loss
- Goal: compute a “good” path from source to destination
 - “good” usually means the shortest (least cost) path



Internet Routing

- Internet Routing works at two levels
- Each AS runs an **intra-domain** routing protocol that establishes routes within its domain
 - (AS -- region of network under a single administrative entity)
 - Link State, e.g., Open Shortest Path First (OSPF)
 - Distance Vector, e.g., Routing Information Protocol (RIP)
- ASes participate in an **inter-domain** routing protocol that establishes routes between domains
 - Path Vector, e.g., Border Gateway Protocol (BGP)

Addressing (for now)

- Assume each host has a unique ID (address)
- No particular structure to those IDs
- Later in course will talk about real IP addressing

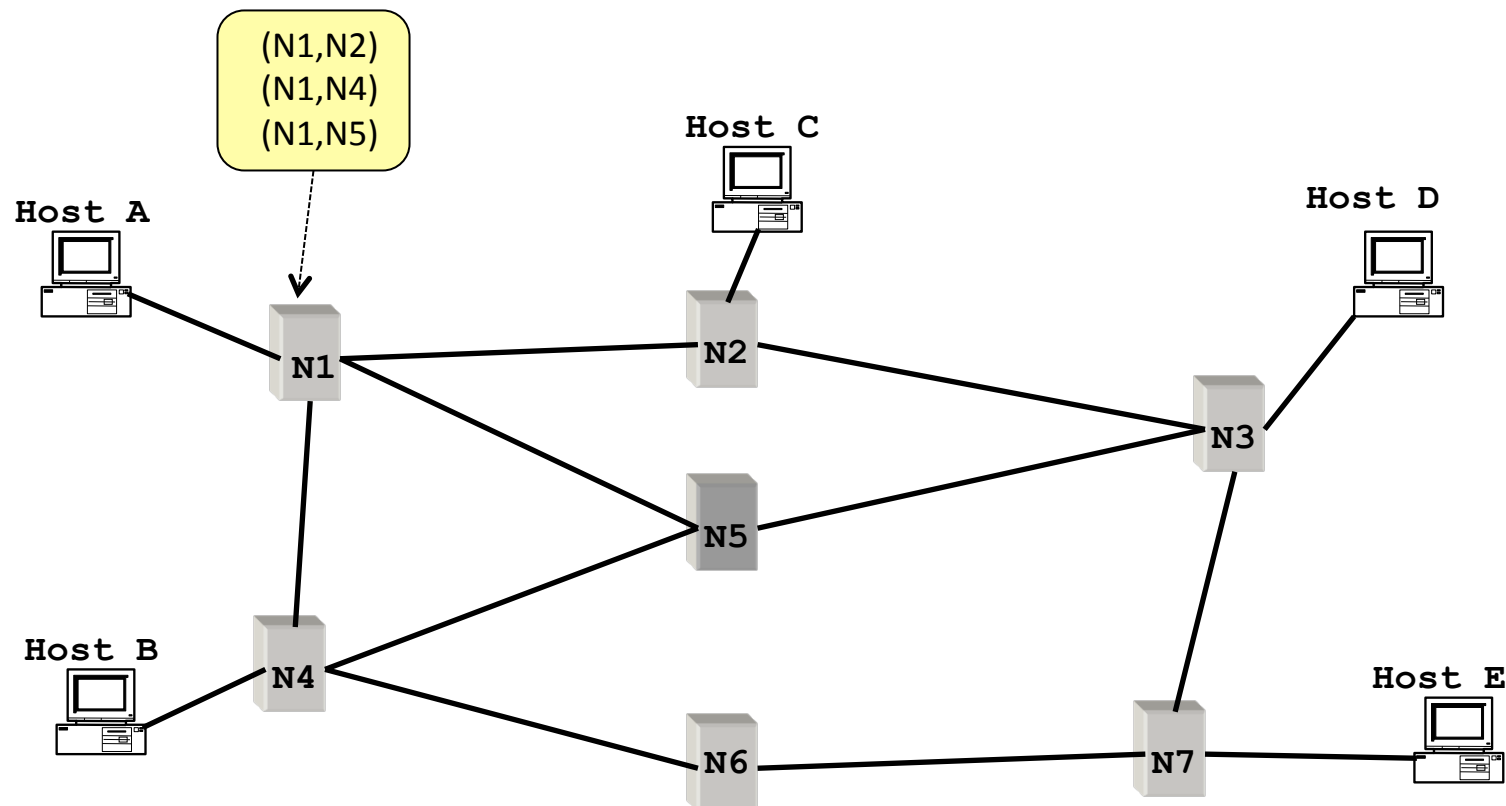
Outline

- Link State
- Distance Vector
- Routing: goals and metrics (if time)

Link-State

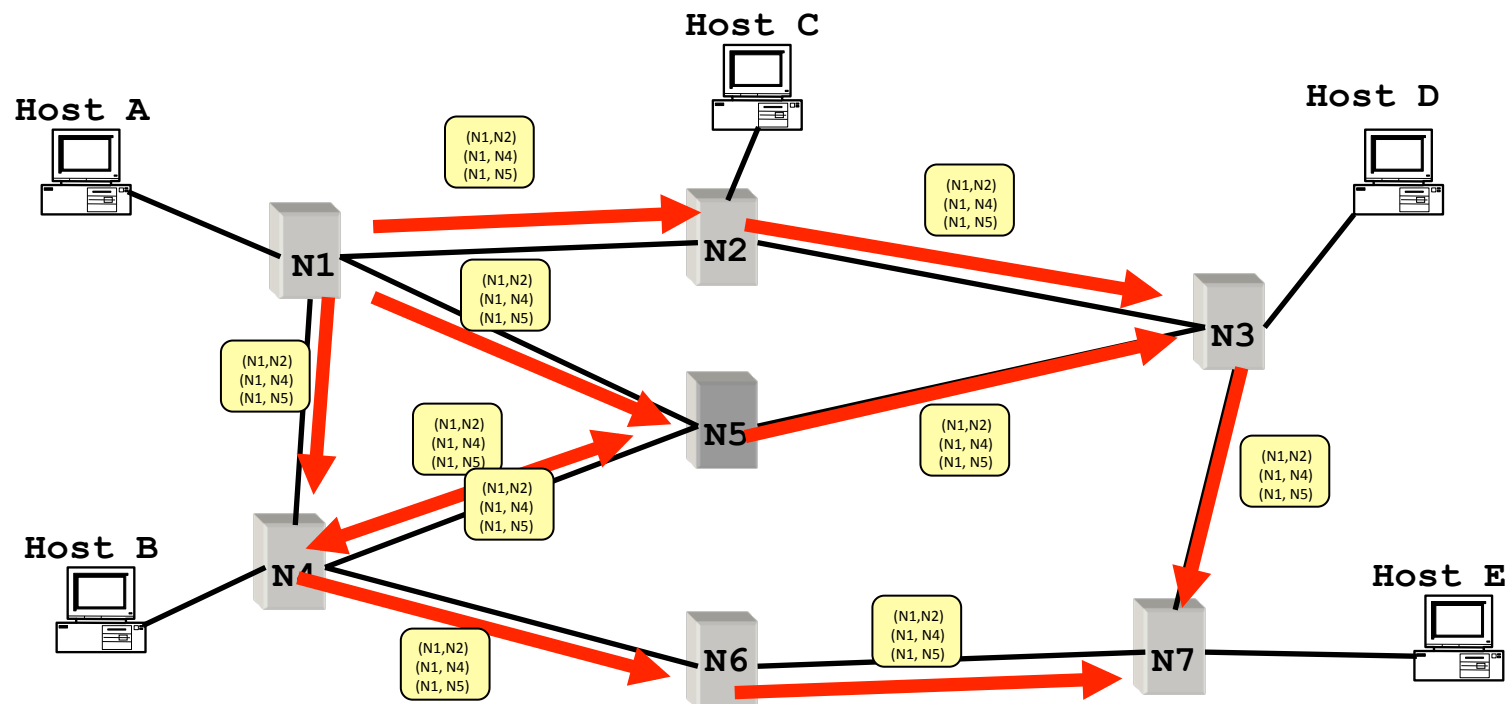
Link State Routing

- Each node maintains its **local** “link state” (LS)
 - i.e., a list of its directly attached links and their costs



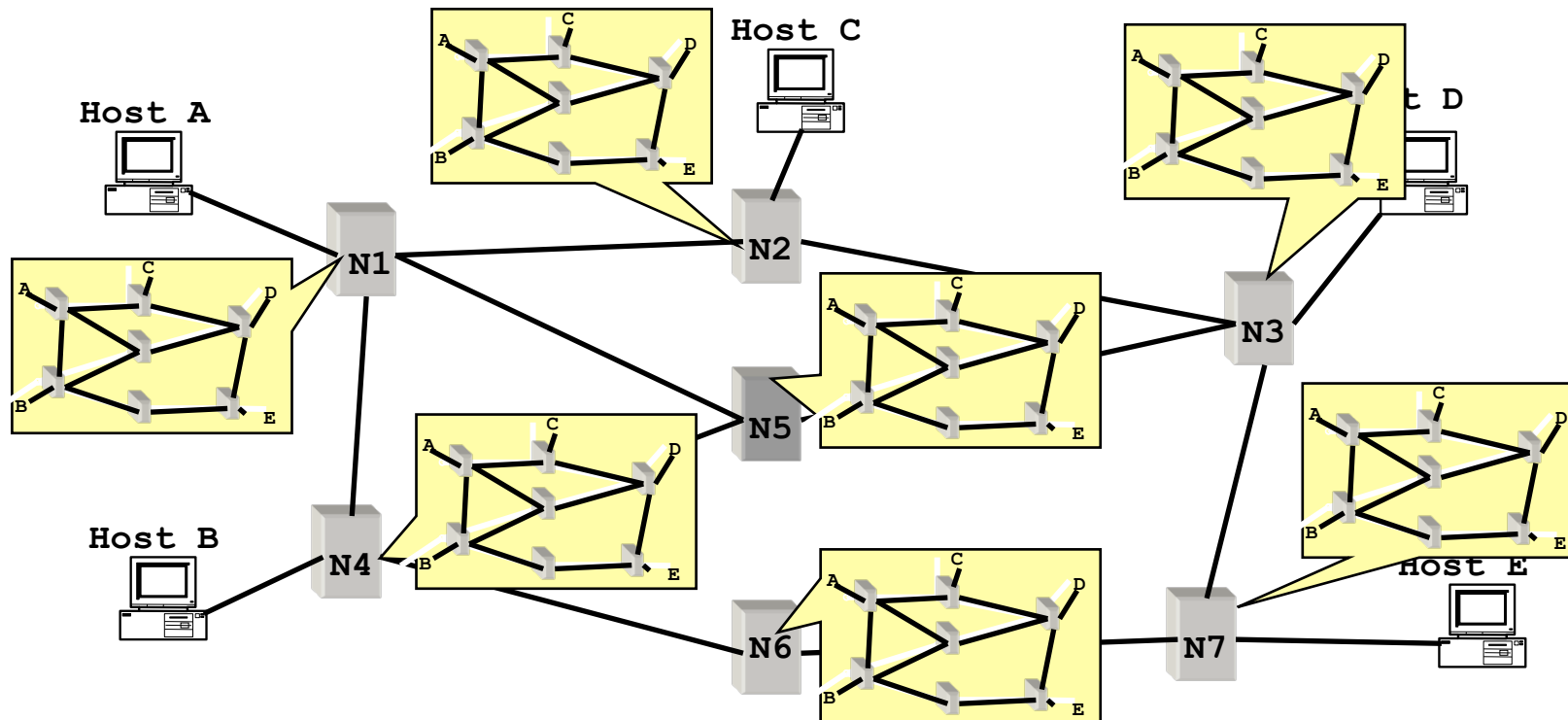
Link State Routing

- Each node maintains its local “link state” (LS)
- Each node floods its local link state
 - on receiving a **new** LS message, a router forwards the message to all its neighbors other than the one it received the message from



Link State Routing

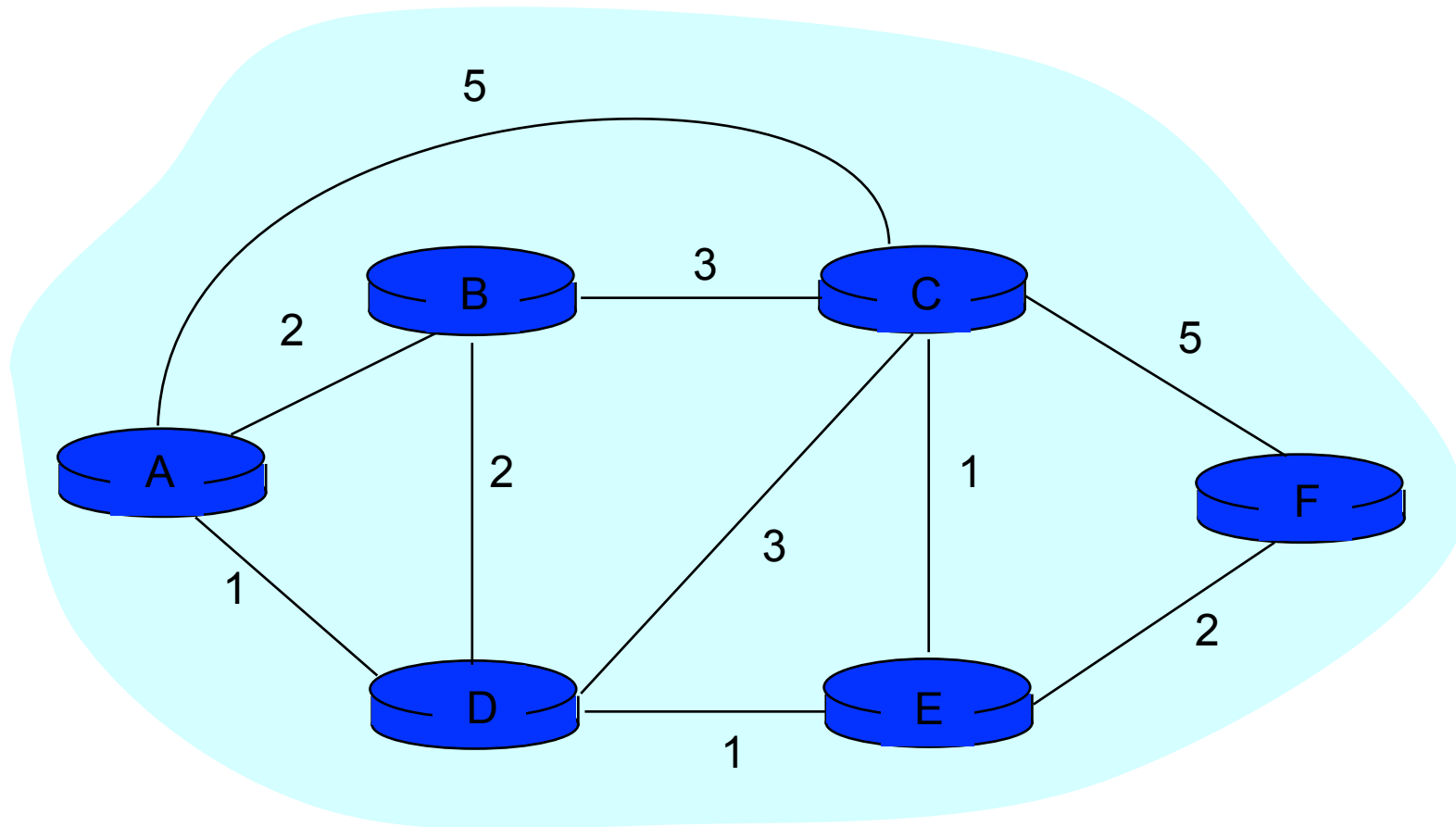
- Each node maintains its local “link state” (LS)
- Each node floods its local link state
- Hence, each node learns the entire network topology
 - Can use Dijkstra’s to compute the shortest paths between nodes



Dijkstra's Shortest Path Algorithm

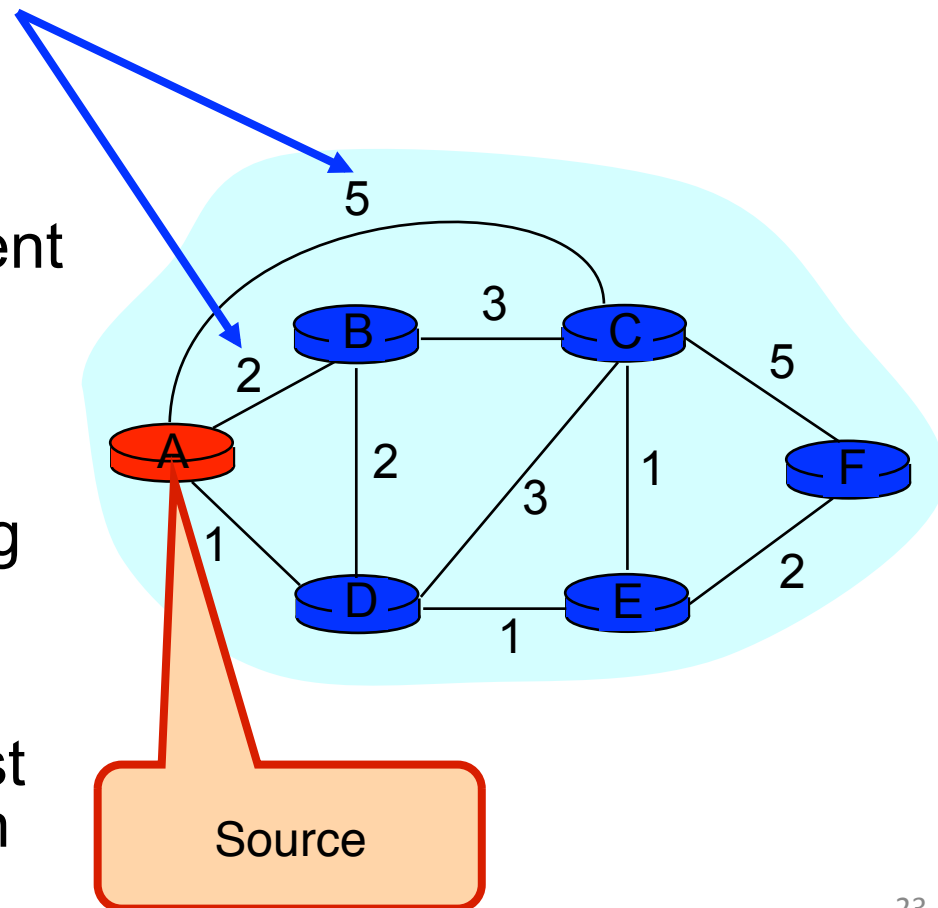
- INPUT:
 - Network topology (graph), with link costs
- OUTPUT:
 - Least cost paths from one node to all other nodes
- Iterative: after k iterations, a node knows the least cost path to its k closest neighbors

Example



Notation

- $c(i,j)$: link cost from node i to j ; cost is infinite if not direct neighbors; ≥ 0
- $D(v)$: total cost of the current least cost path from source to destination v
- $p(v)$: v 's predecessor along path from source to v
- S : set of nodes whose least cost path definitively known



Dijkstra's Algorithm

1 **Initialization:**

2 $\mathbf{S} = \{\mathbf{A}\};$

3 for all nodes \mathbf{v}

4 if \mathbf{v} adjacent to \mathbf{A}

5 then $D(\mathbf{v}) = c(\mathbf{A}, \mathbf{v});$

6 else $D(\mathbf{v}) = \infty;$

7

8 **Loop**

9 find \mathbf{w} not in \mathbf{S} such that $D(\mathbf{w})$ is a minimum;

10 add \mathbf{w} to $\mathbf{S};$

11 update $D(\mathbf{v})$ for all \mathbf{v} adjacent to \mathbf{w} and not in $\mathbf{S};$

12 if $D(\mathbf{w}) + c(\mathbf{w}, \mathbf{v}) < D(\mathbf{v})$ then

// \mathbf{w} gives us a shorter path to \mathbf{v} than we've found so far

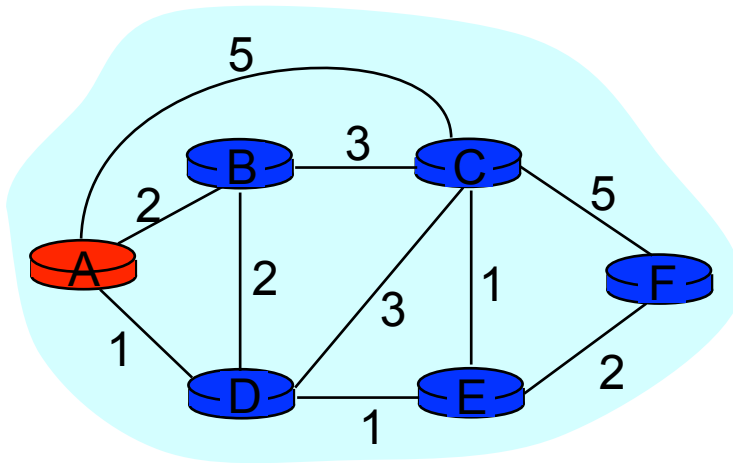
13 $D(\mathbf{v}) = D(\mathbf{w}) + c(\mathbf{w}, \mathbf{v}); p(\mathbf{v}) = \mathbf{w};$

14 **until all nodes in $\mathbf{S};$**

- $c(i,j)$: link cost from node i to j
- $D(\mathbf{v})$: current cost source $\rightarrow \mathbf{v}$
- $p(\mathbf{v})$: \mathbf{v} 's predecessor along path from source to \mathbf{v}
- \mathbf{S} : set of nodes whose least cost path definitively known

Example: Dijkstra's Algorithm

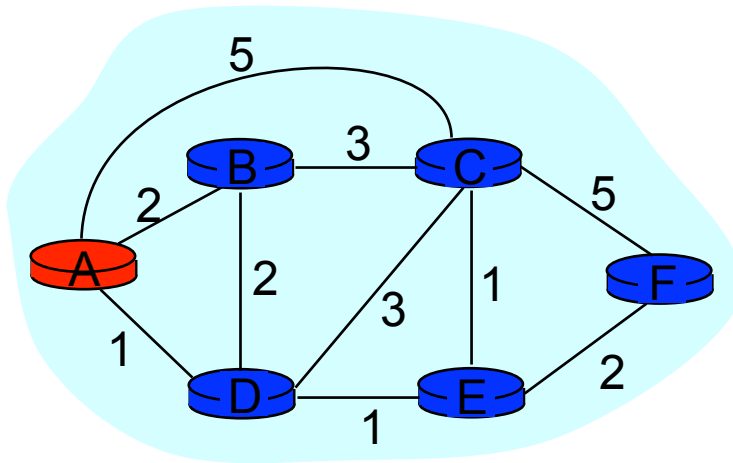
Step	set S	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	∞	∞
1						
2						
3						
4						
5						



- 1 **Initialization:**
- 2 **S** = {A};
- 3 for all nodes **v**
- 4 if **v** adjacent to **A**
- 5 then $D(v) = c(A,v)$;
- 6 else $D(v) = \infty$;
- ...

Example: Dijkstra's Algorithm

Step	set S	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	∞	∞
1						
2						
3						
4						
5						

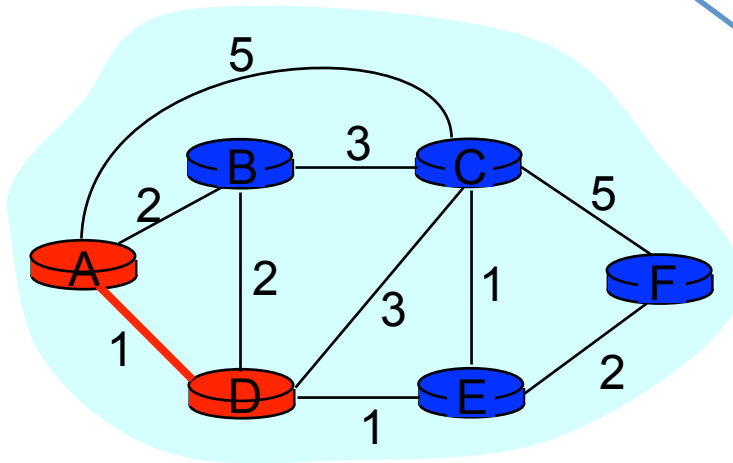


```

...
8 Loop
9   find w not in S s.t. D(w) is a minimum;
10  add w to S;
11  update D(v) for all v adjacent
    to w and not in S:
12  If D(w) + c(w,v) < D(v) then
13    D(v) = D(w) + c(w,v); p(v) = w;
14  until all nodes in S;
    
```

Example: Dijkstra's Algorithm

Step	set S	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	∞	∞
1	AD					
2						
3						
4						
5						

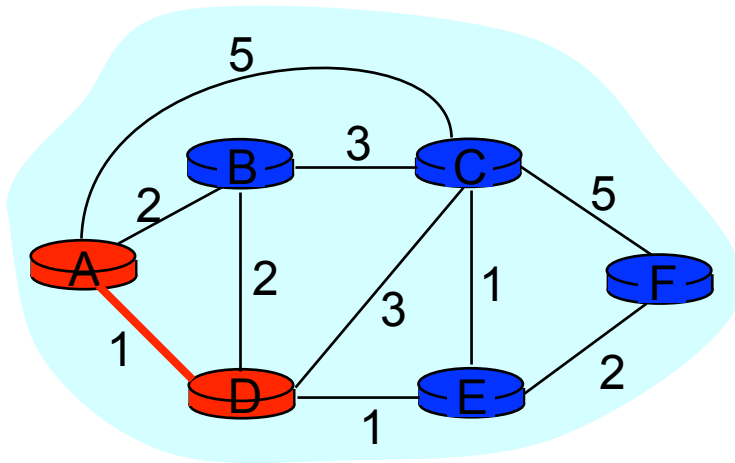


```

...
8 Loop
9   find w not in S s.t. D(w) is a minimum;
10  add w to S;
11  update D(v) for all v adjacent
    to w and not in S:
12  If  $D(w) + c(w,v) < D(v)$  then
13     $D(v) = D(w) + c(w,v)$ ;  $p(v) = w$ ;
14  until all nodes in S;
    
```

Example: Dijkstra's Algorithm

Step	set S	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	∞	∞
1	AD		4,D		2,D	
2						
3						
4						
5						

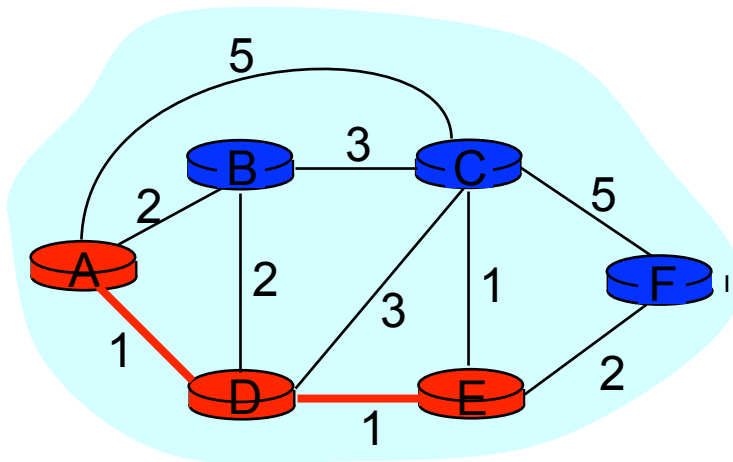


```

...
8 Loop
9   find w not in S s.t. D(w) is a minimum;
10  add w to S;
11  update D(v) for all v adjacent
    to w and not in S:
12  If  $D(w) + c(w,v) < D(v)$  then
13     $D(v) = D(w) + c(w,v)$ ;  $p(v) = w$ ;
14  until all nodes in S;
    
```

Example: Dijkstra's Algorithm

Step	set S	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	∞	∞
1	AD		4,D		2,D	
2	ADE		3,E			4,E
3						
4						
5						

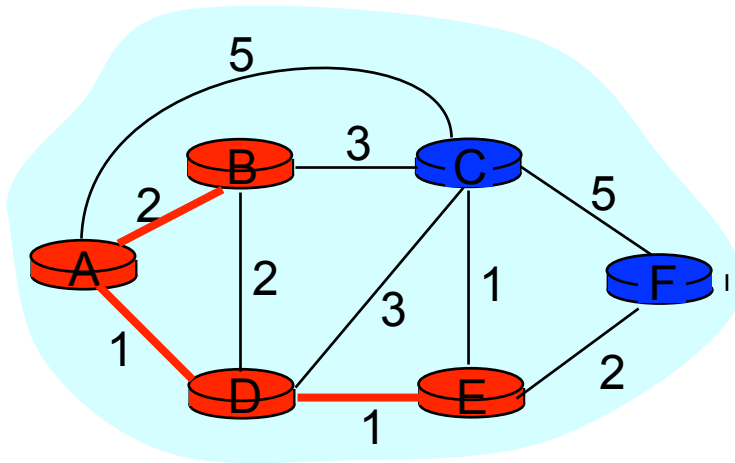


```

...
8 Loop
9   find w not in S s.t. D(w) is a minimum;
10  add w to S;
11  update D(v) for all v adjacent
    to w and not in S:
12  If  $D(w) + c(w,v) < D(v)$  then
13     $D(v) = D(w) + c(w,v)$ ;  $p(v) = w$ ;
14  until all nodes in S;
    
```

Example: Dijkstra's Algorithm

Step	set S	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	∞	∞
1	AD		4,D		2,D	
2	ADE		3,E			4,E
3	ADEB					
4						
5						

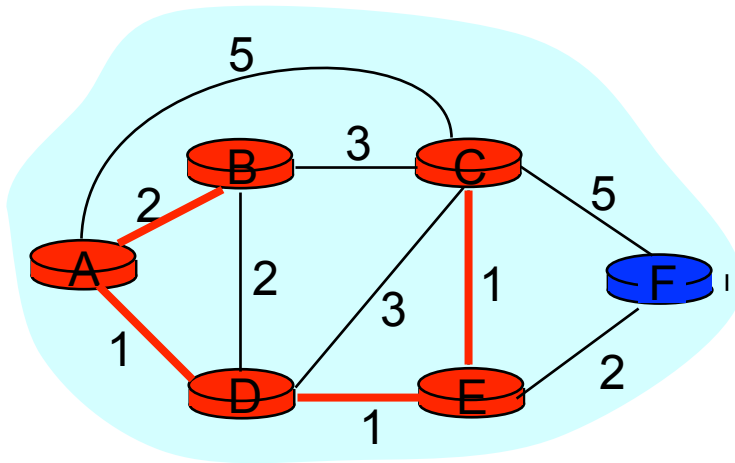


```

...
8 Loop
9   find w not in S s.t.  $D(w)$  is a minimum;
10  add w to S;
11  update  $D(v)$  for all v adjacent
    to w and not in S:
12  If  $D(w) + c(w,v) < D(v)$  then
13     $D(v) = D(w) + c(w,v)$ ;  $p(v) = w$ ;
14  until all nodes in S;
    
```

Example: Dijkstra's Algorithm

Step	set S	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	∞	∞
1	AD		4,D		2,D	
2	ADE		3,E			4,E
3	ADEB					
4	ADEBC					
5						

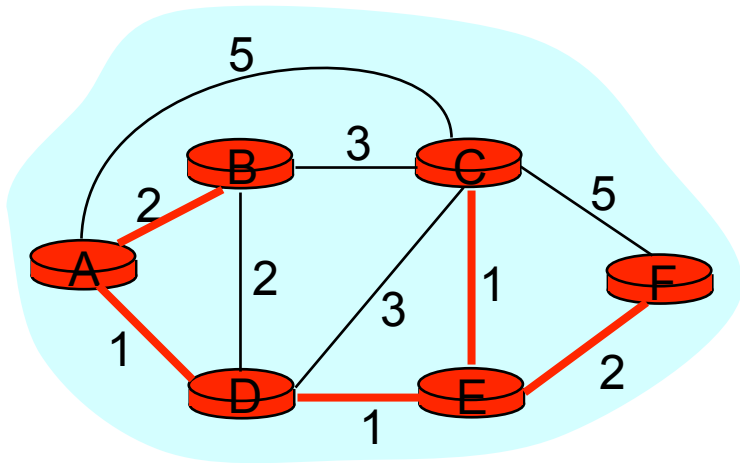


```

...
8 Loop
9   find w not in S s.t.  $D(w)$  is a minimum;
10  add w to S;
11  update  $D(v)$  for all v adjacent
    to w and not in S:
12  If  $D(w) + c(w,v) < D(v)$  then
13     $D(v) = D(w) + c(w,v)$ ;  $p(v) = w$ ;
14  until all nodes in S;
    
```

Example: Dijkstra's Algorithm

Step	set S	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	∞	∞
1	AD		4,D		2,D	
2	ADE		3,E			4,E
3	ADEB					
4	ADEBC					
5	ADEBCF					

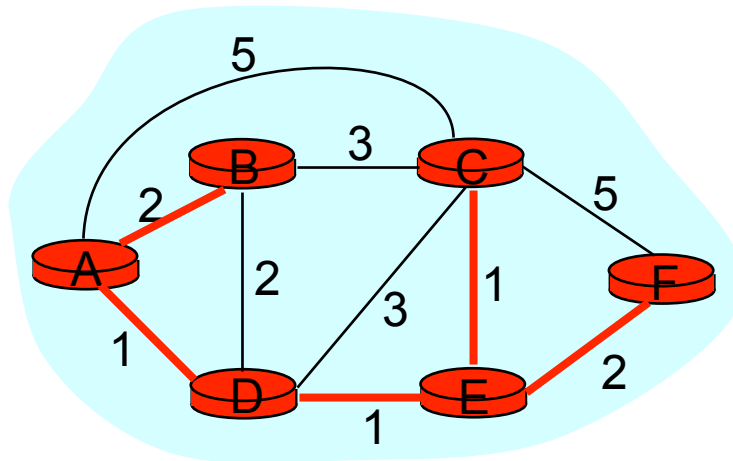


```

...
8 Loop
9   find w not in S s.t.  $D(w)$  is a minimum;
10  add w to S;
11  update  $D(v)$  for all v adjacent
    to w and not in S:
12  If  $D(w) + c(w,v) < D(v)$  then
13     $D(v) = D(w) + c(w,v)$ ;  $p(v) = w$ ;
14  until all nodes in S;
    
```


Example: Dijkstra's Algorithm

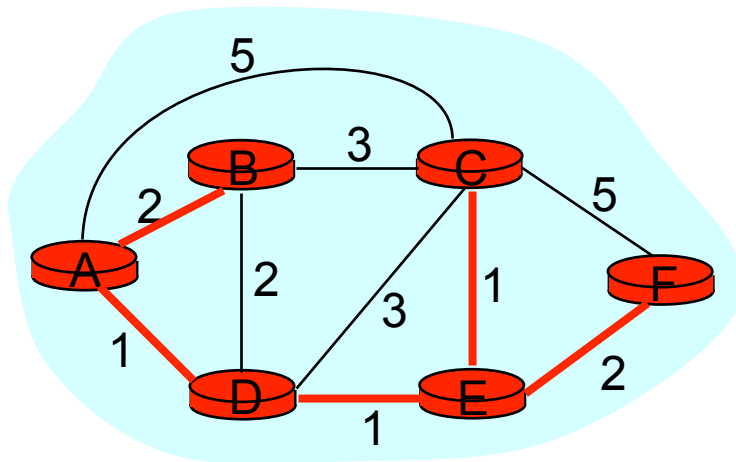
Step	set S	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	∞	∞
1	AD		4,D		2,D	
2	ADE		3,E			4,E
3	ADEB					
4	ADEBC					
5	ADEBCF					



To determine path $A \rightarrow C$ (say), work backward from C via $p(v)$

The Forwarding Table

- Running Dijkstra at node A gives the shortest path from A to all destinations
- We then construct the *forwarding table*



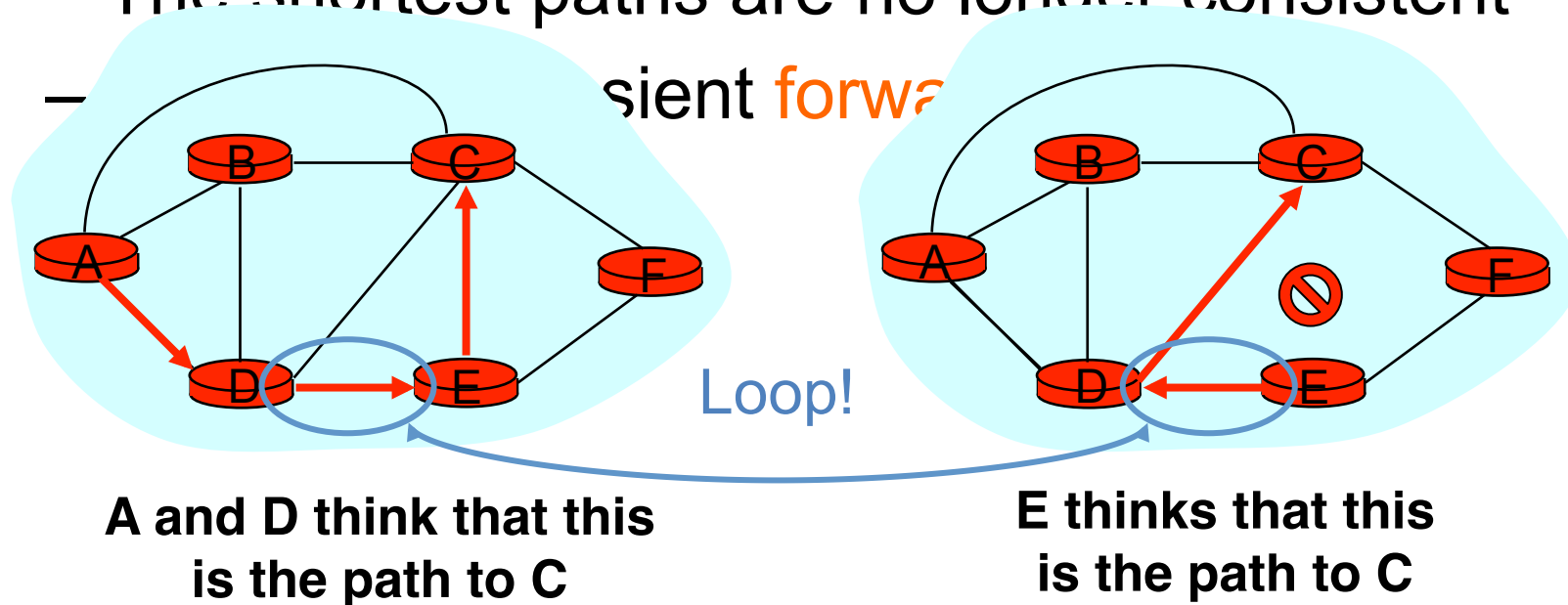
Destination	Link
B	(A,B)
C	(A,D)
D	(A,D)
E	(A,D)
F	(A,D)

Issue #1: Scalability

- How many messages needed to flood link state messages?
 - $O(N \times E)$, where N is #nodes; E is #edges in graph
- Processing complexity for Dijkstra's algorithm?
 - $O(N^2)$, because we check all nodes w not in S at each iteration and we have $O(N)$ iterations
 - more efficient implementations: $O(N \log(N))$
- How many entries in the LS topology database? $O(E)$
- How many entries in the forwarding table? $O(N)$

Issue#2: Transient Disruptions

- Inconsistent link-state database
 - Some routers know about failure before others
 - The shortest paths are no longer consistent
 -



Distance Vector

Learn-By-Doing

Let's try to collectively develop
distance-vector routing from first principles

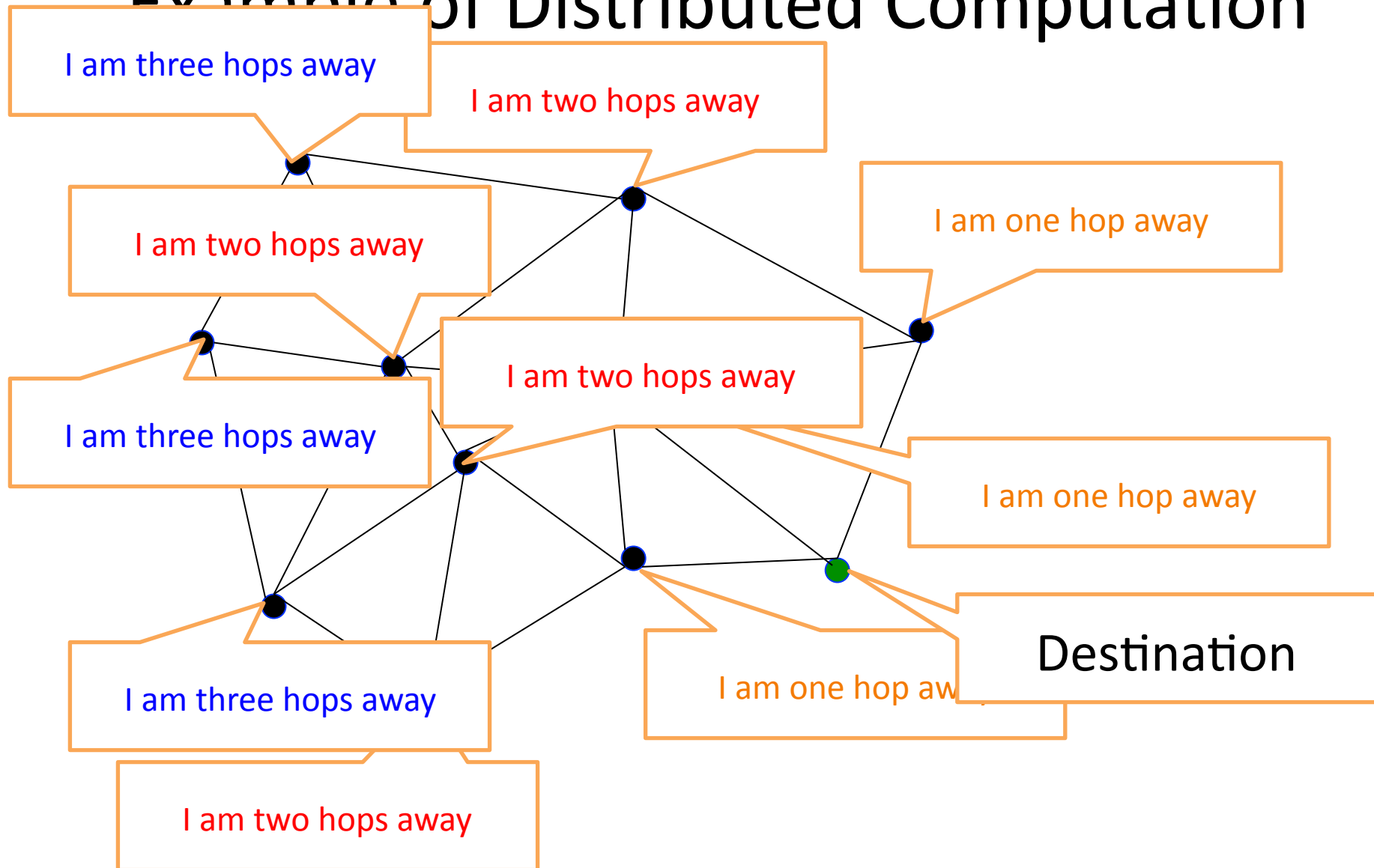
Experiment

- Your job: find the (route to) the youngest person in the room
- Ground Rules
 - **You may not** leave your seat, nor shout loudly across the class
 - **You may** talk with your immediate neighbors
(N-S-E-W only)
(hint: “exchange updates” with them)
- At the end of **5 minutes**, I will pick a victim and ask:
 - who is the youngest person in the room? (date&name)
 - which one of your neighbors first told you this info.?

Go!

Distance-Vector

Example of Distributed Computation



Distance Vector Routing

- Each router knows the links to its neighbors
 - Does *not* flood this information to the whole network
- Each router has provisional “shortest path” to **every** other router
 - E.g.: Router A: “I can get to router B with cost 11”
- Routers exchange this **distance vector** information with their neighboring routers
 - Vector because one entry per destination
- Routers look over the set of options offered by their neighbors and select the best one
- Iterative process converges to set of shortest paths

A few other inconvenient aspects

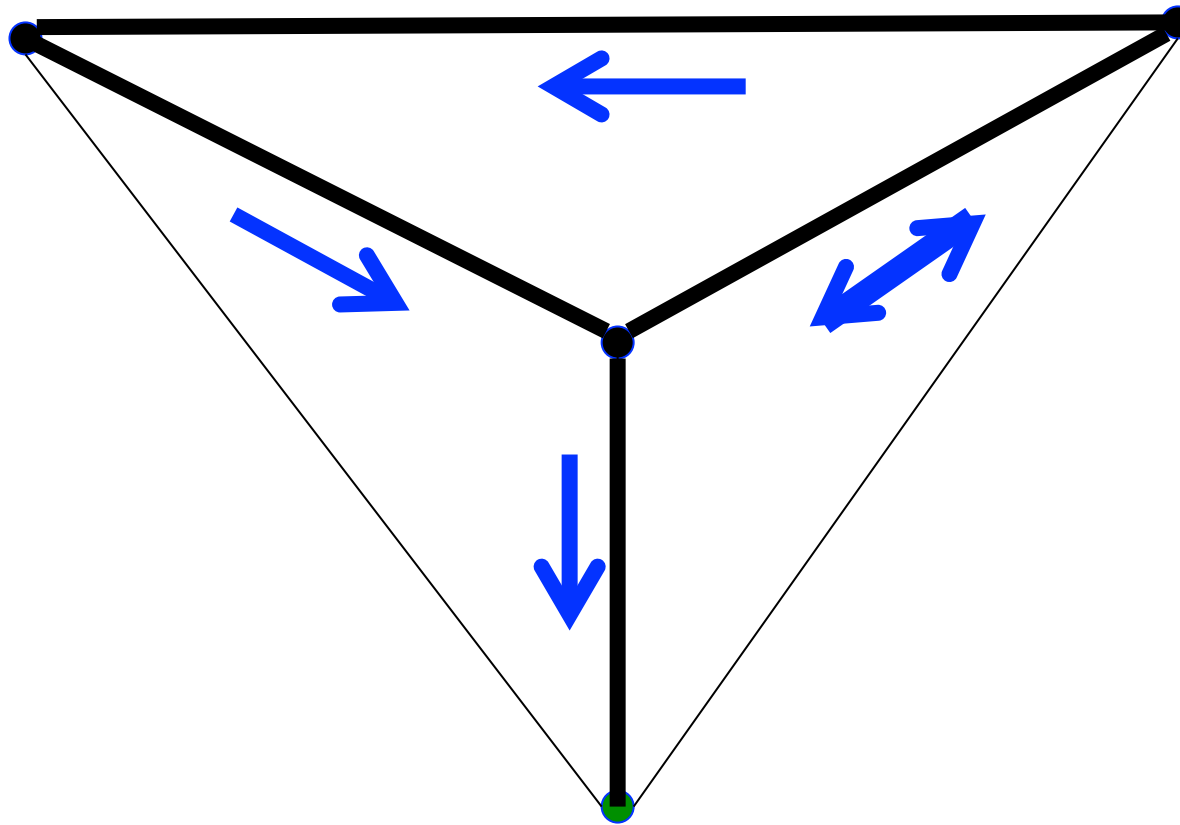
- What if we use a non-additive metric?
 - E.g., maximal capacity
- What if routers don't use the same metric?
 - I want low delay, you want low loss rate?
- What happens if nodes lie?

Can You Use Any Metric?

- I said that we can pick any metric. Really?
- What about maximizing capacity?

What Happens Here?

Problem: "cost" does not change around loop

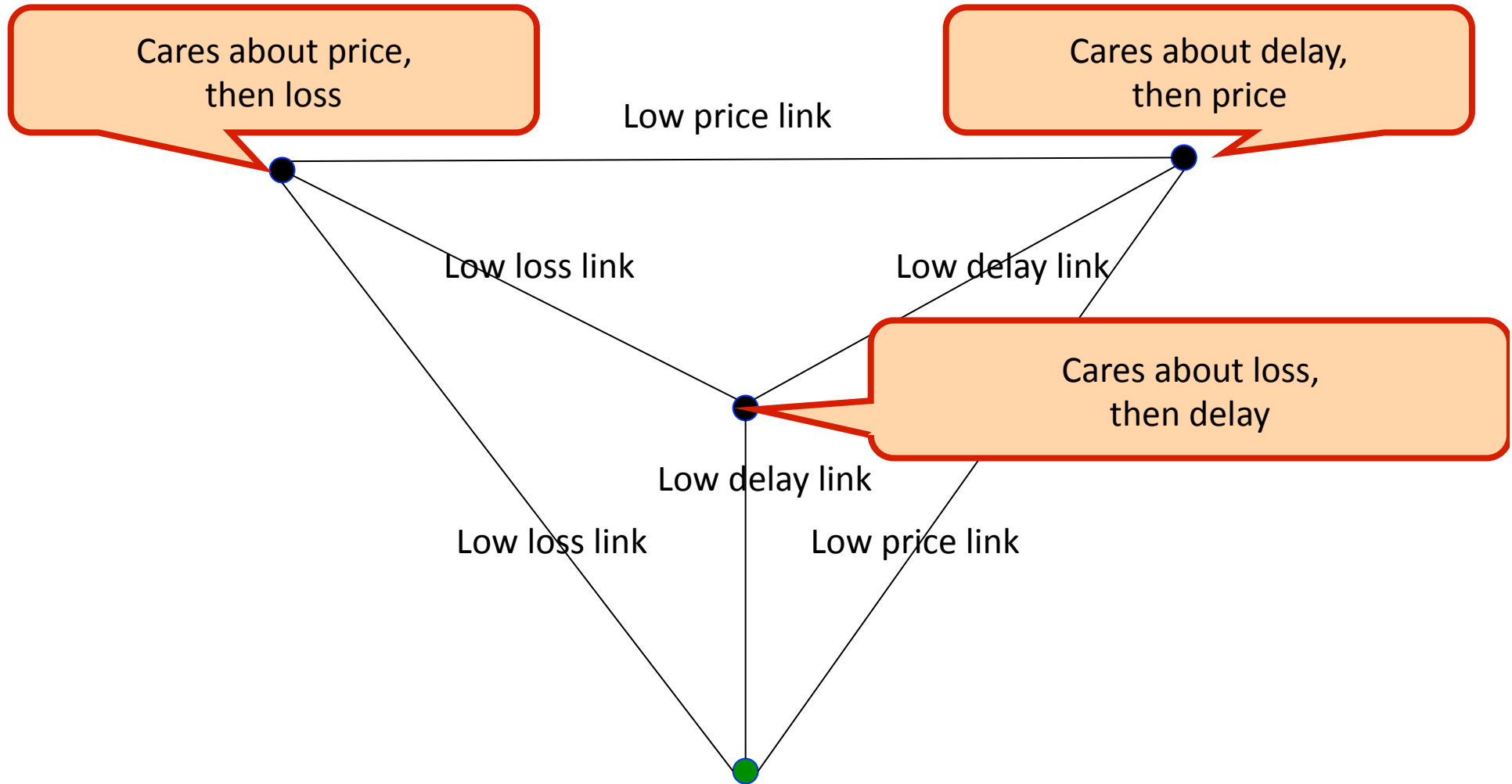


Additive measures avoid this problem!

No agreement on metrics?

- If the nodes choose their paths according to different criteria, then bad things might happen
- Example
 - Node A is minimizing latency
 - Node B is minimizing loss rate
 - Node C is minimizing price
- Any of those goals are fine, if globally adopted
 - Only a problem when nodes use different criteria
- Consider a routing algorithm where paths are described by delay, cost, loss

What Happens Here?



Must agree on loop-avoiding metric

- When all nodes minimize same metric
- And that metric increases around loops
- Then process is guaranteed to converge

What happens when routers lie?

- What if a router claims a 1-hop path to everywhere?
- All traffic from nearby routers gets sent there
- How can you tell if they are lying?
- Can this happen in real life?
 - It has, several times....

Link State vs. Distance Vector

- Core idea
 - LS: tell all nodes about your immediate neighbors
 - DV: tell your immediate neighbors about (your least cost distance to) all nodes

Link State vs. Distance Vector

- LS: each node learns the complete network map; each node computes shortest paths independently and in parallel
 - DV: no node has the complete picture; nodes cooperate to compute shortest paths in a distributed manner
-
- LS has higher messaging overhead
 - LS has higher processing complexity
 - LS is less vulnerable to looping

Link State vs. Distance Vector

Message complexity

- LS: $O(N \times E)$ messages;
 - N is #nodes; E is #edges
- DV: $O(\#Iterations \times E)$
 - where #Iterations is ideally $O(\text{network diameter})$ but varies due to routing loops or the count-to-infinity problem

Processing complexity

- LS: $O(N^2)$
- DV: $O(\#Iterations \times N)$

Robustness: what happens if router malfunctions?

- LS:
 - node can advertise incorrect *link* cost
 - each node computes only its *own* table
- DV:
 - node can advertise incorrect *path* cost
 - each node's table used by others; error propagates through network

Routing: Just the Beginning

- Link state and distance-vector are the deployed routing paradigms for intra-domain routing
- Inter-domain routing (BGP)
 - more Part II (Principles of Communications)
 - A version of DV

What are desirable goals for a routing solution?

- “Good” paths (least cost)
- Fast convergence after change/failures
 - no/rare loops
- Scalable
 - #messages
 - table size
 - processing complexity
- Secure
- Policy
- Rich metrics (more later)

Delivery models

- What if a node wants to send to more than one destination?
 - broadcast: send to all
 - multicast: send to all members of a group
 - anycast: send to any member of a group
- What if a node wants to send along more than one path?

Metrics

- Propagation delay
- Congestion
- Load balance
- Bandwidth (available, capacity, maximal, bbw)
- Price
- Reliability
- Loss rate
- Combinations of the above

In practice, operators set abstract “weights” (much like our costs); how exactly is a bit of a black art

From Routing back to Forwarding

- Routing: “control plane”
 - Computing paths the packets will follow
 - Routers talking amongst themselves
 - Jointly creating the routing state
- Forwarding: “data plane”
 - Directing a data packet to an outgoing link
 - Individual router using routing state
- Two very different timescales....

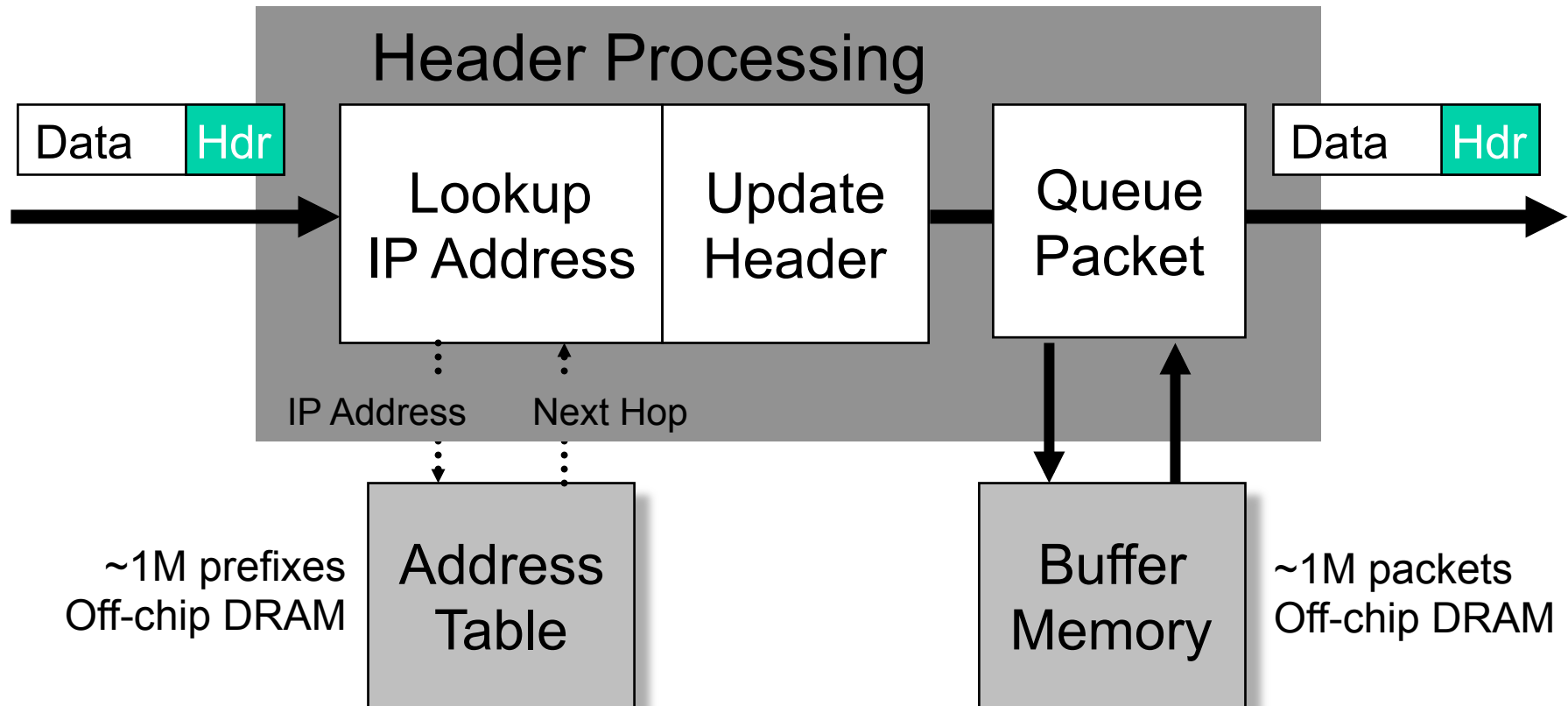
Basic Architectural Components of an IP Router



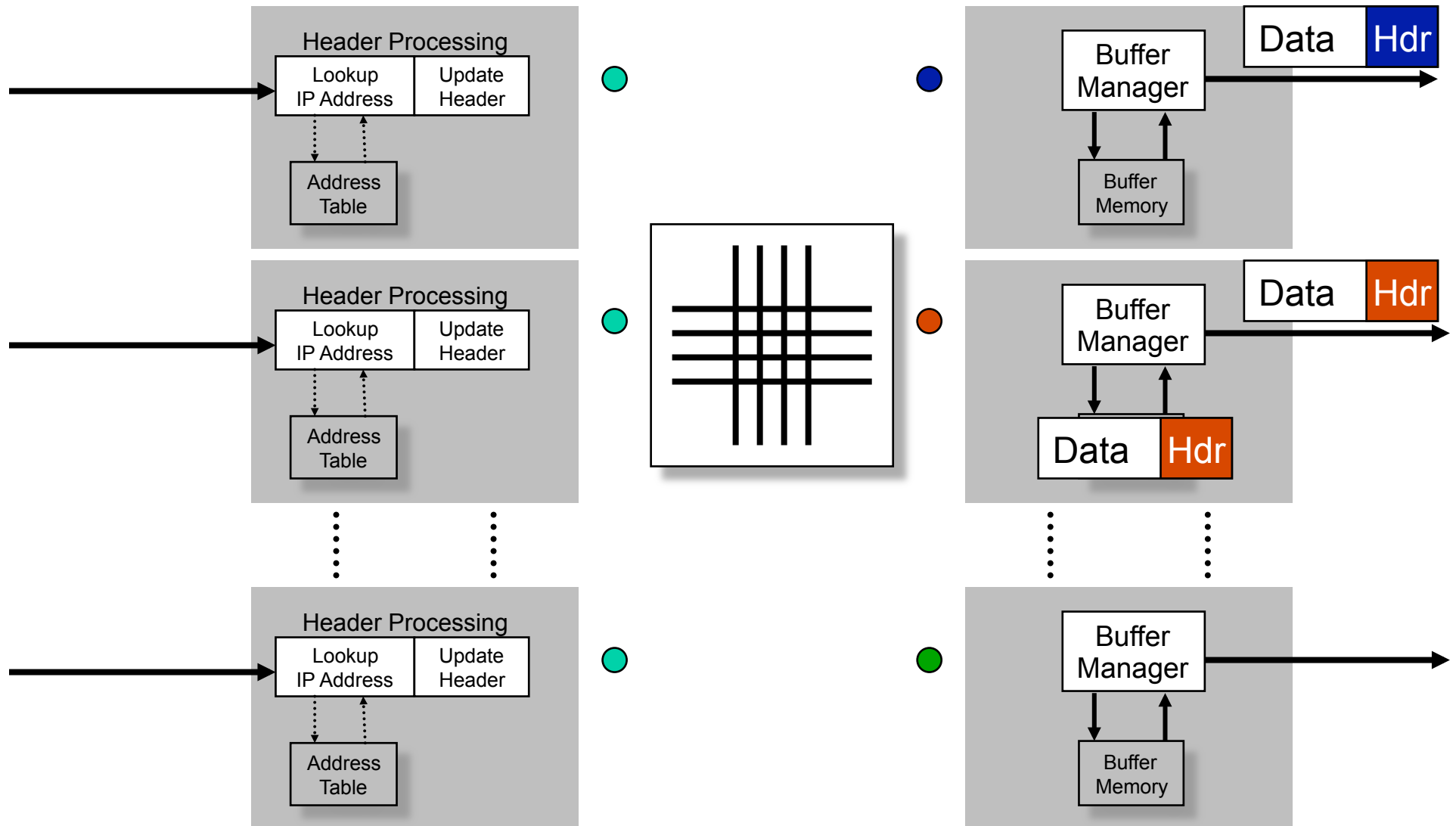
Per-packet processing in an IP Router

1. Accept packet arriving on an incoming link.
2. Lookup packet destination address in the forwarding table, to identify outgoing port(s).
3. Manipulate packet header: e.g., decrement TTL, update header checksum.
4. Send packet to the outgoing port(s).
5. Buffer packet in the queue.
6. Transmit packet onto outgoing link.

Generic Router Architecture



Generic Router Architecture



Forwarding tables

IP address

 } 32 bits wide → ~ 4 billion unique address

Naïve approach:

One entry per address

Entry	Destination	Port
1	0.0.0.0	1
2	0.0.0.1	2
⋮	⋮	⋮
2^{32}	255.255.255.255	12

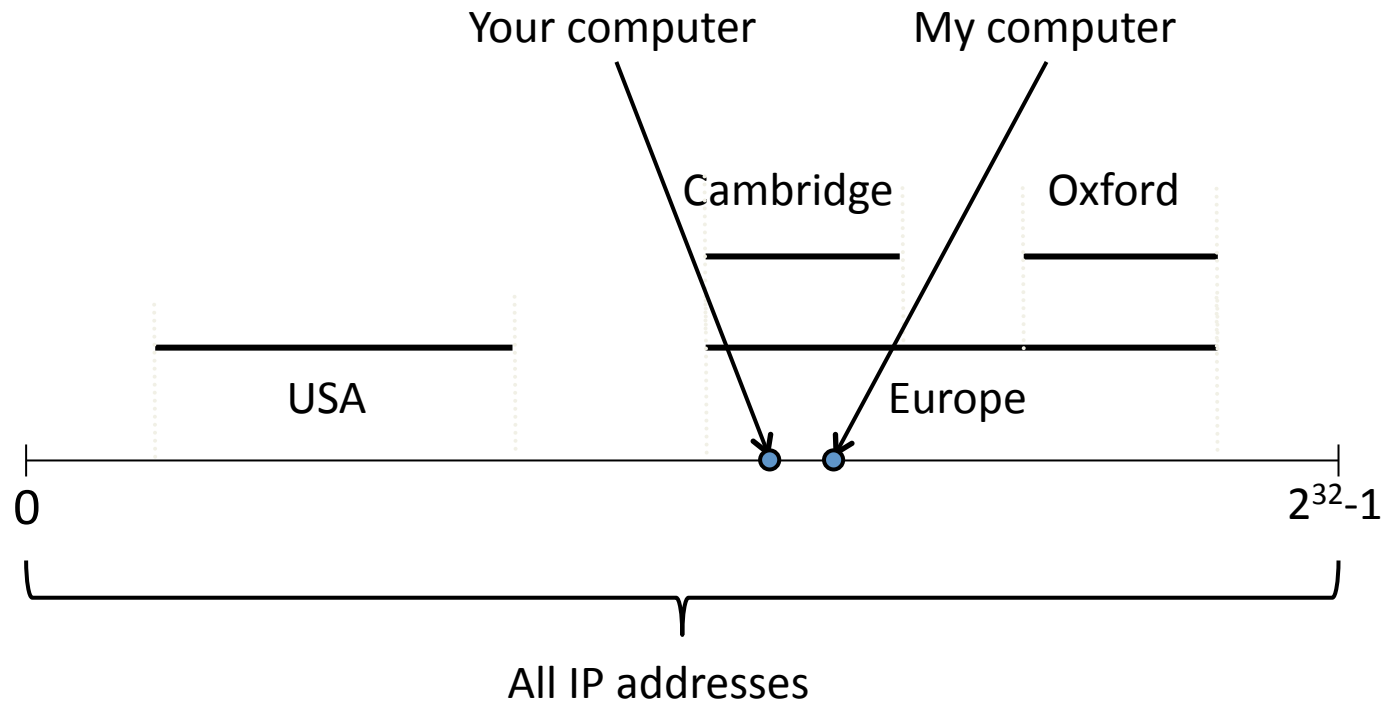
} ~ 4 billion entries

Improved approach:

Group entries to reduce table size

Entry	Destination	Port
1	0.0.0.0 – 127.255.255.255	1
2	128.0.0.1 – 128.255.255.255	2
⋮	⋮	⋮
50	248.0.0.0 – 255.255.255.255	12

IP addresses as a line



Entry	Destination	Port
1	Cambridge	1
2	Oxford	2
3	Europe	3
4	USA	4
5	Everywhere (default)	5

Longest Prefix Match (LPM)

Entry	Destination	Port	
1	Cambridge	1	Universities
2	Oxford	2	
3	Europe	3	Continents
4	USA	4	
5	Everywhere (default)	5	Planet

Matching entries:

- Cambridge Most specific
- Europe
- Everywhere

To: Cambridge	Data
------------------	------

Longest Prefix Match (LPM)

Entry	Destination	Port	
1	Cambridge	1	Universities
2	Oxford	2	
3	Europe	3	Continents
4	USA	4	
5	Everywhere (default)	5	Planet

Matching entries:

- Europe Most specific
- Everywhere

To: France

Data

Implementing Longest Prefix Match

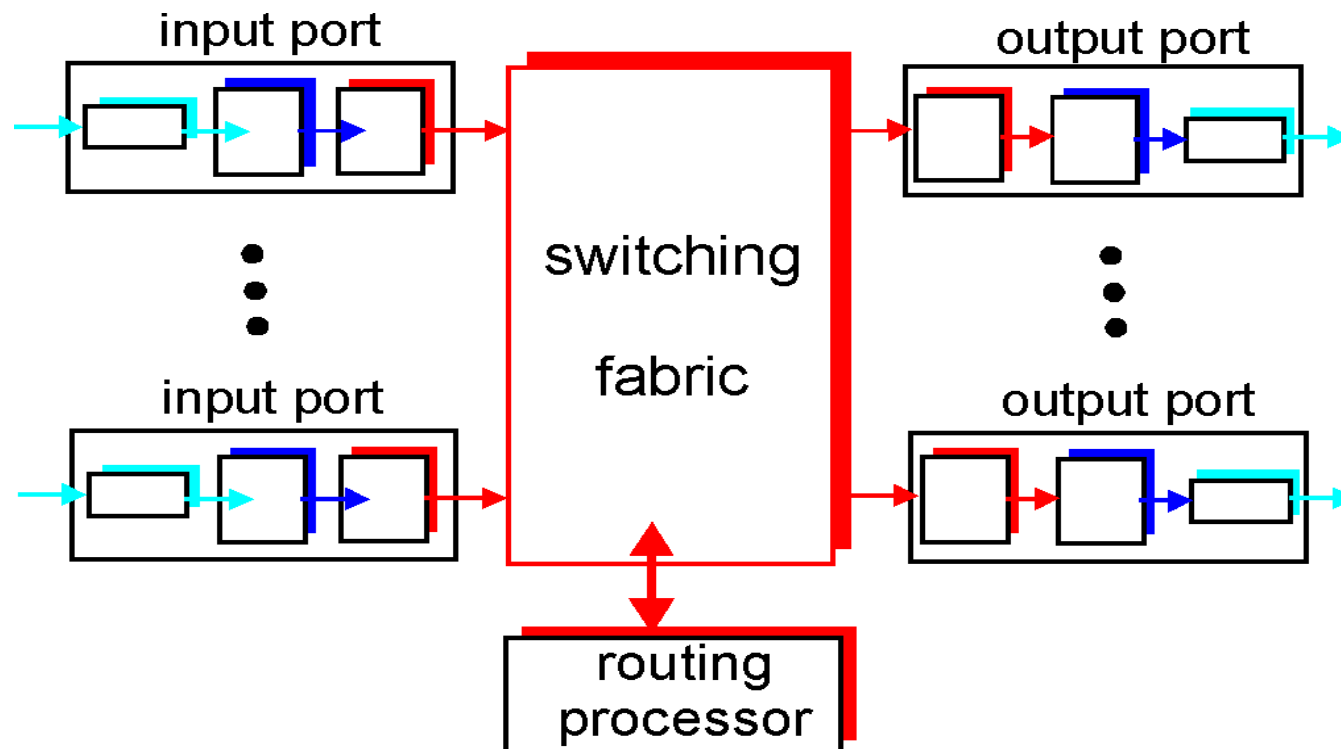
Entry	Destination	Port	
1	Cambridge	1	Searching
2	Oxford	2	
3	Europe	3	
4	USA	4	FOUND
5	Everywhere (default)	5	

Most specific
↓
Least specific

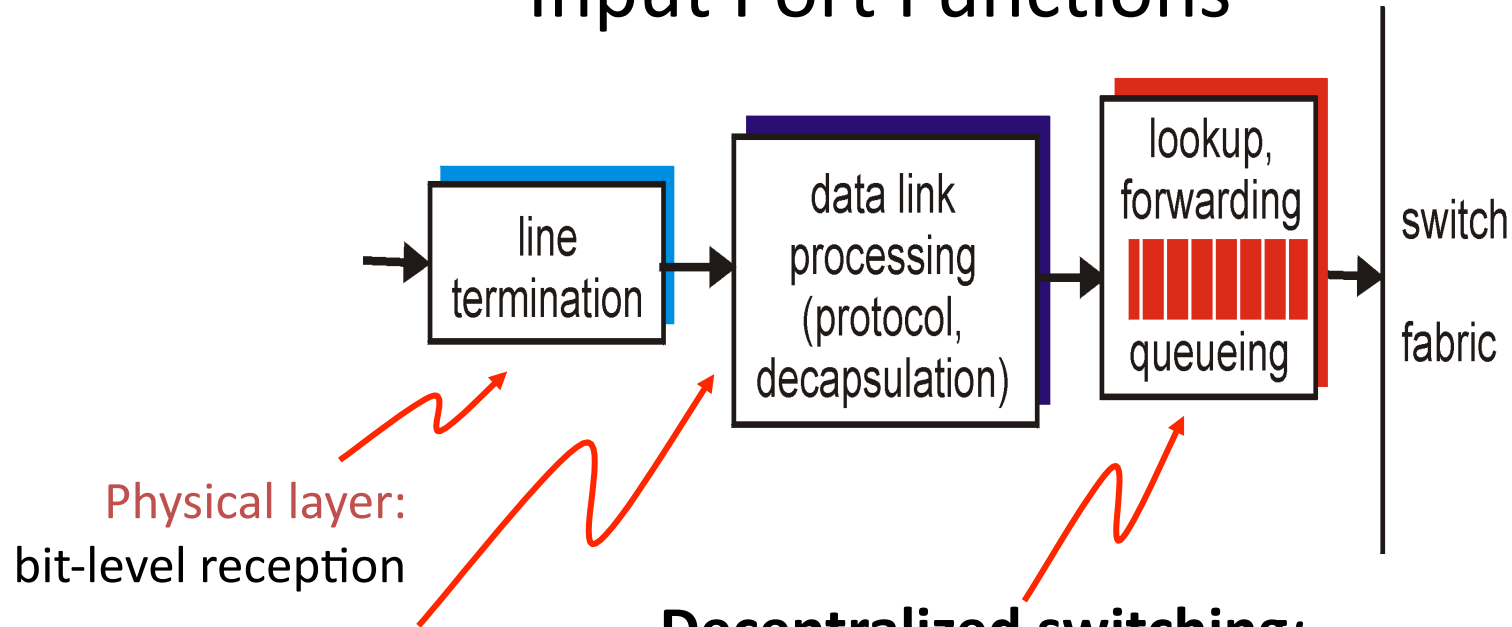
Router Architecture Overview

Two key router functions:

- run routing algorithms/protocol (RIP, OSPF, BGP)
- *forwarding* datagrams from incoming to outgoing link



Input Port Functions



Physical layer:
bit-level reception

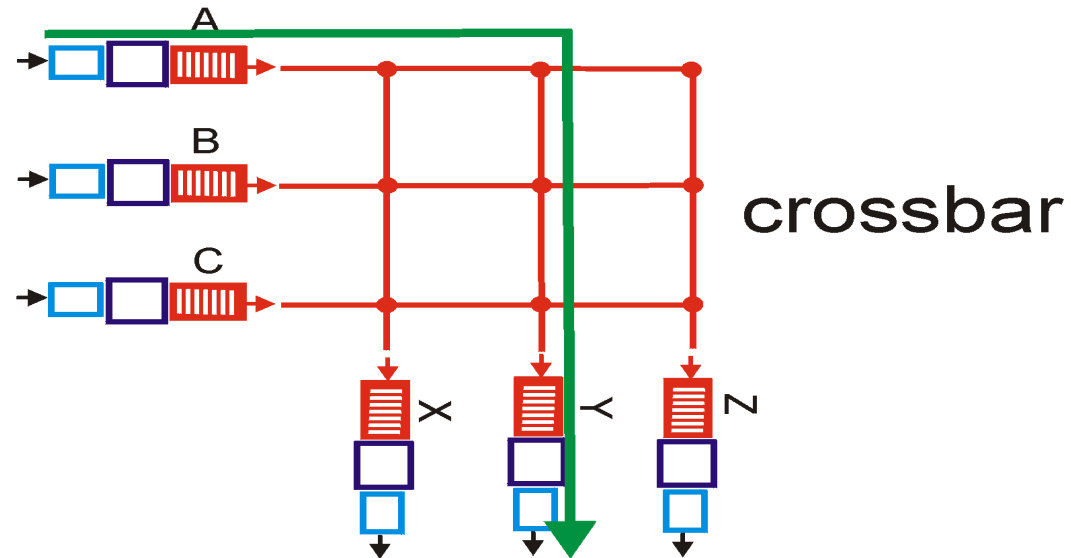
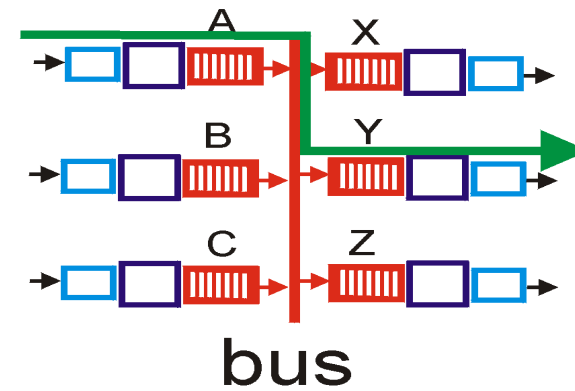
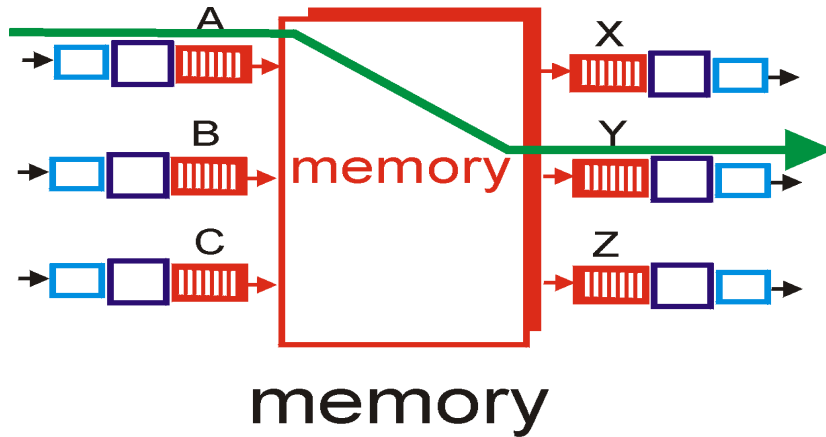
Data link layer:
e.g., Ethernet
see chapter 5

Decentralized switching:

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

Three types of switching fabrics

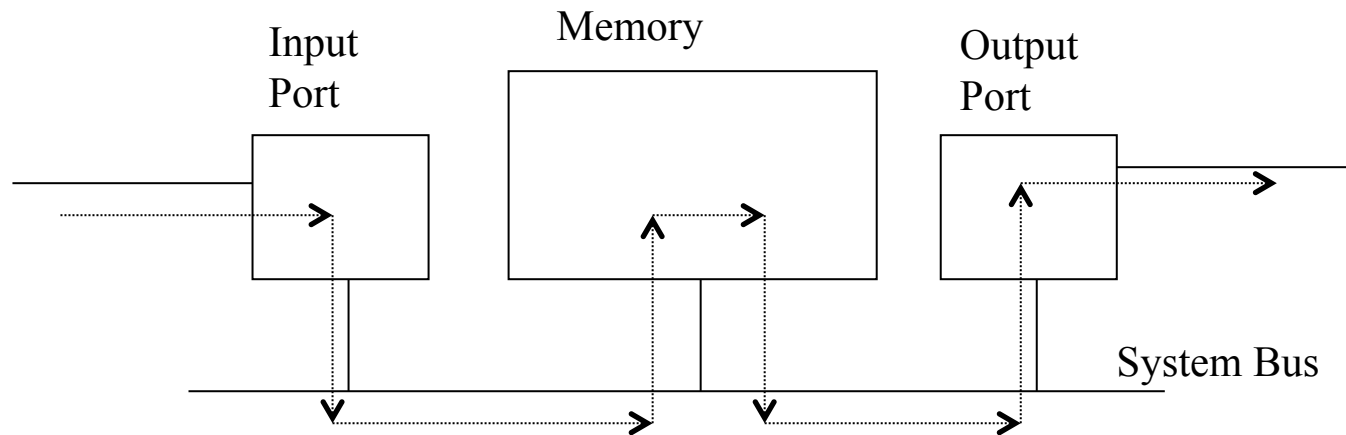
(comparison criteria: speed, contention, complexity)



Switching Via Memory

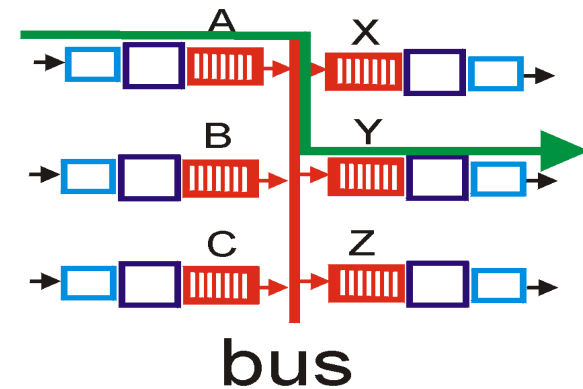
First generation routers:

- traditional computers with switching under direct control of CPU
- packet copied to system's memory
- speed limited by memory bandwidth (2 bus crossings per datagram)



Switching Via a Bus

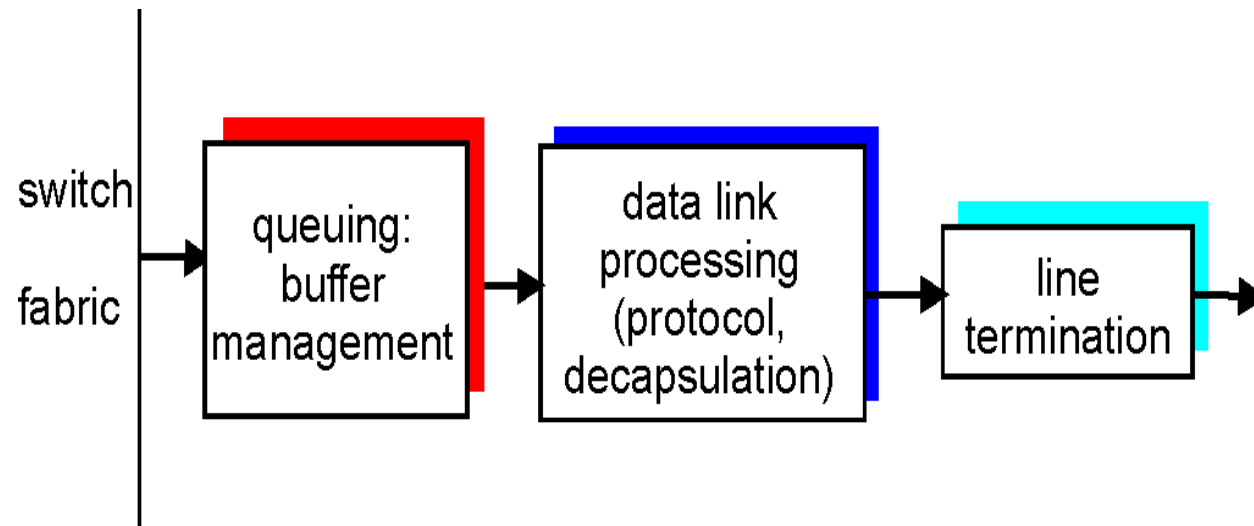
- datagram from input port memory to output port memory via a shared bus
- **bus contention:** switching speed limited by bus bandwidth
- Lots of ports?? speed up the bus
no contention bus speed =
 $2 \times \text{port speed} \times \text{port count}$
- 32 Gbps bus, Cisco 5600: sufficient speed for access and enterprise routers



Switching Via An Interconnection Network

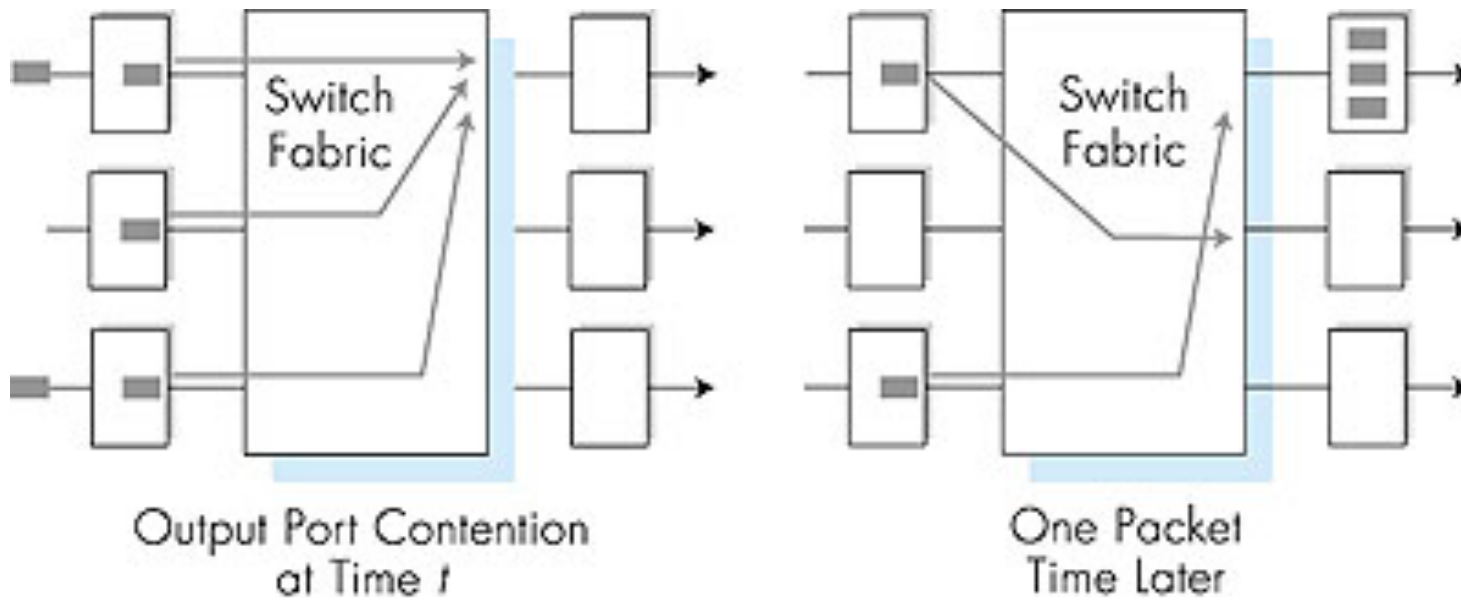
- overcome bus bandwidth limitations
- Banyan networks, other interconnection nets initially developed to connect processors in multiprocessor
- advanced design: fragmenting datagram into fixed length cells, switch cells through the fabric.
- Cisco 12000: switches 60 Gbps through the interconnection network

Output Ports



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

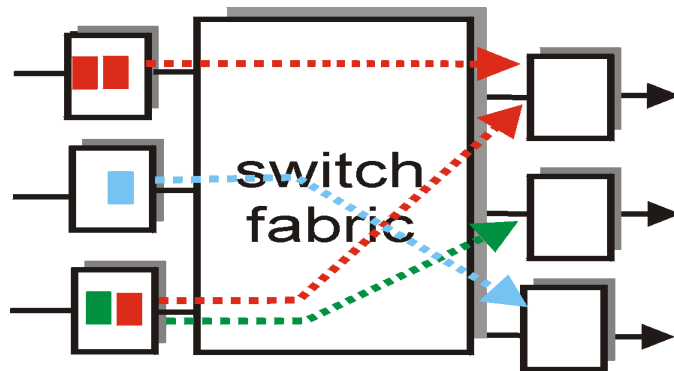
Output port queueing



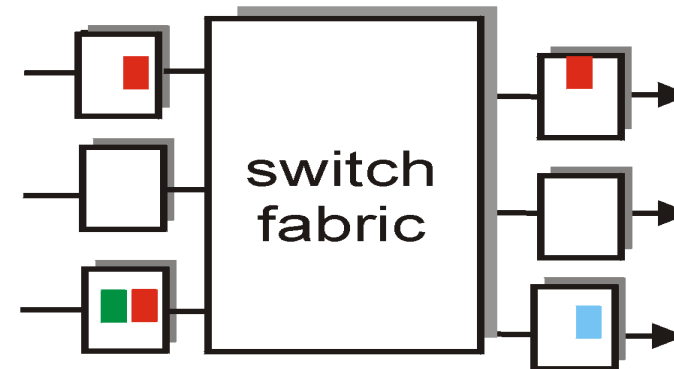
- buffering when arrival rate via switch exceeds output line speed
- *queueing (delay) and loss due to output port buffer overflow!*

Input Port Queuing

- Fabric slower than 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
- *queueing delay and loss due to input buffer overflow!*



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



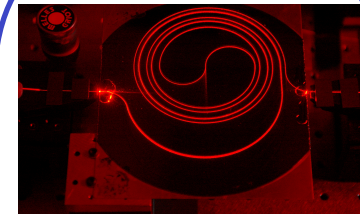
green packet
experiences HOL blocking

Buffers in Routers

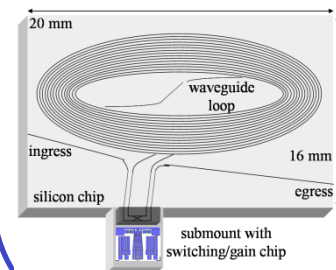
- So how large should the buffers be?

Buffer size matters

- End-to-end delay
 - Transmission, propagation, and queueing delay
 - The only variable part is queueing delay
- Router architecture
 - Board space, power consumption, and cost
 - On chip buffers: higher density, higher capacity
 - Optical buffers: all-optical routers



1.4m long spiral waveguide with input from HeNe laser



You are now touching the edge of the *research zone*.....

Buffer Sizing Story



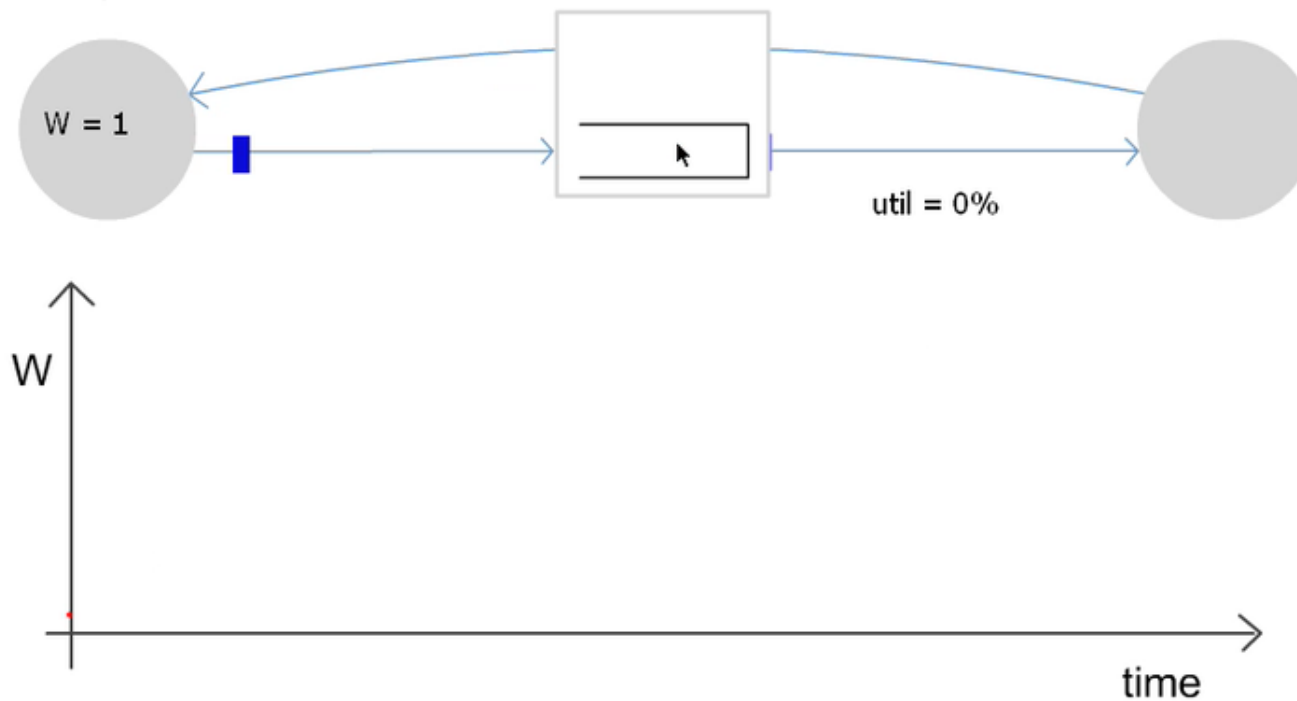
		$2T \times C$	$\frac{2T \times C}{\sqrt{n}}$	$O(\log W)$
# of packets	Rule-of-thumb	1,000,000	10,000	20 - 50
Intuition		TCP Sawtooth	Sawtooth Smoothing	Non-bursty Arrivals
Assume		Single TCP Flow, 100% Utilization	Many Flows, 100% Utilization	Paced TCP, 85-90% Utilization
Evidence		Simulation, Emulation	Simulations, Test-bed and Real Network Experiments	Simulations, Test-bed Experiments

Continuous ARQ (TCP) adapting to congestion

Only W packets
may be outstanding

Rule for adjusting W

- If an ACK is received: $W \leftarrow W + 1/W$
- If a packet is lost: $W \leftarrow W/2$

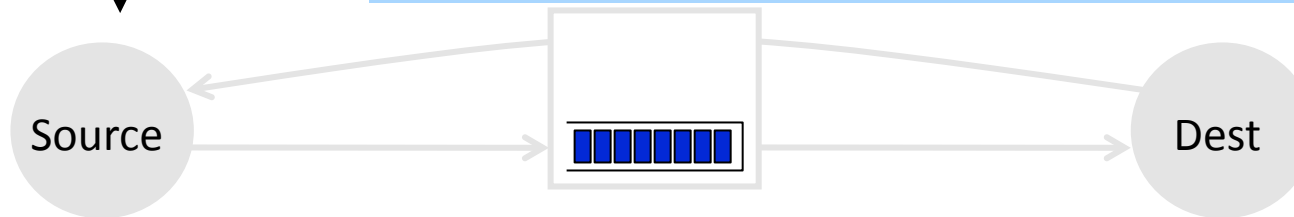


Rule-of-thumb – Intuition

Only W packets
may be outstanding

Rule for adjusting W

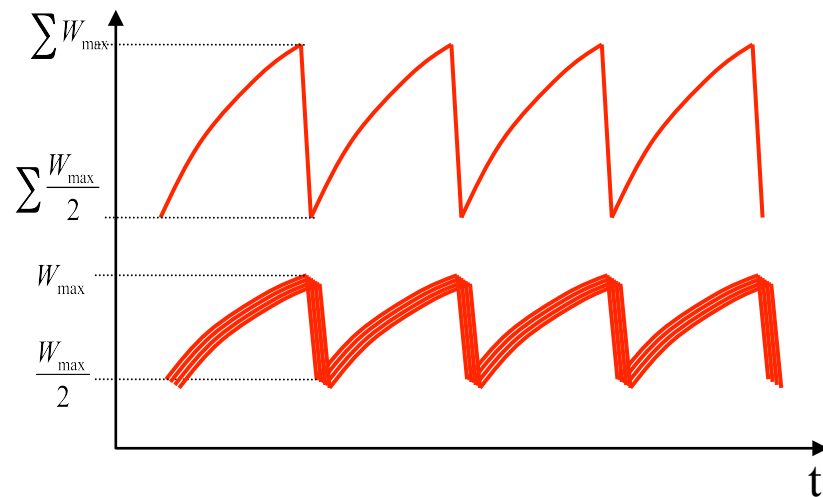
- If an ACK is received: $W \leftarrow W + 1/W$
- If a packet is lost: $W \leftarrow W/2$



Small Buffers – Intuition

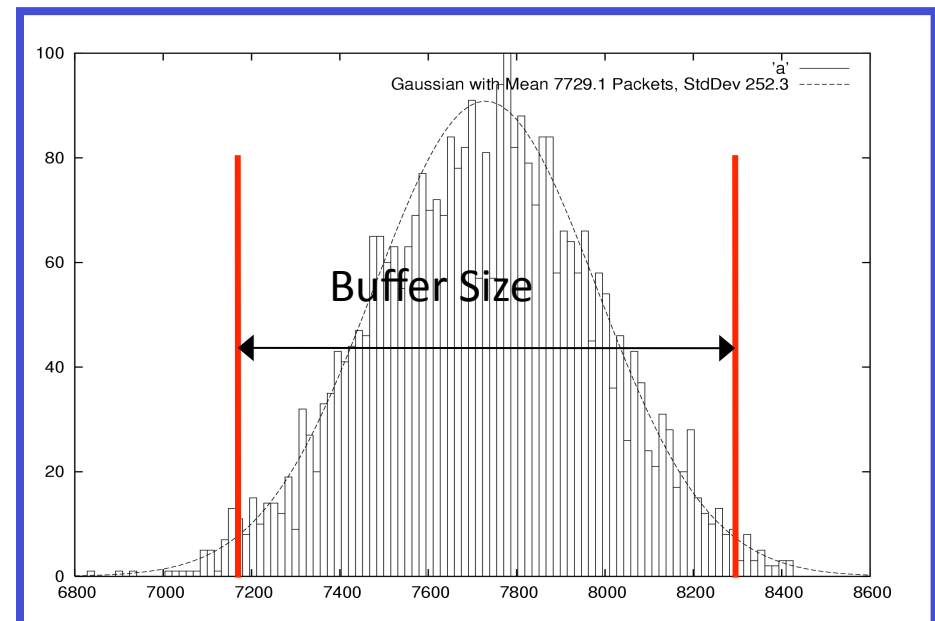
Synchronized Flows

- Aggregate window has same dynamics
- Therefore buffer occupancy has same dynamics
- Rule-of-thumb still holds.



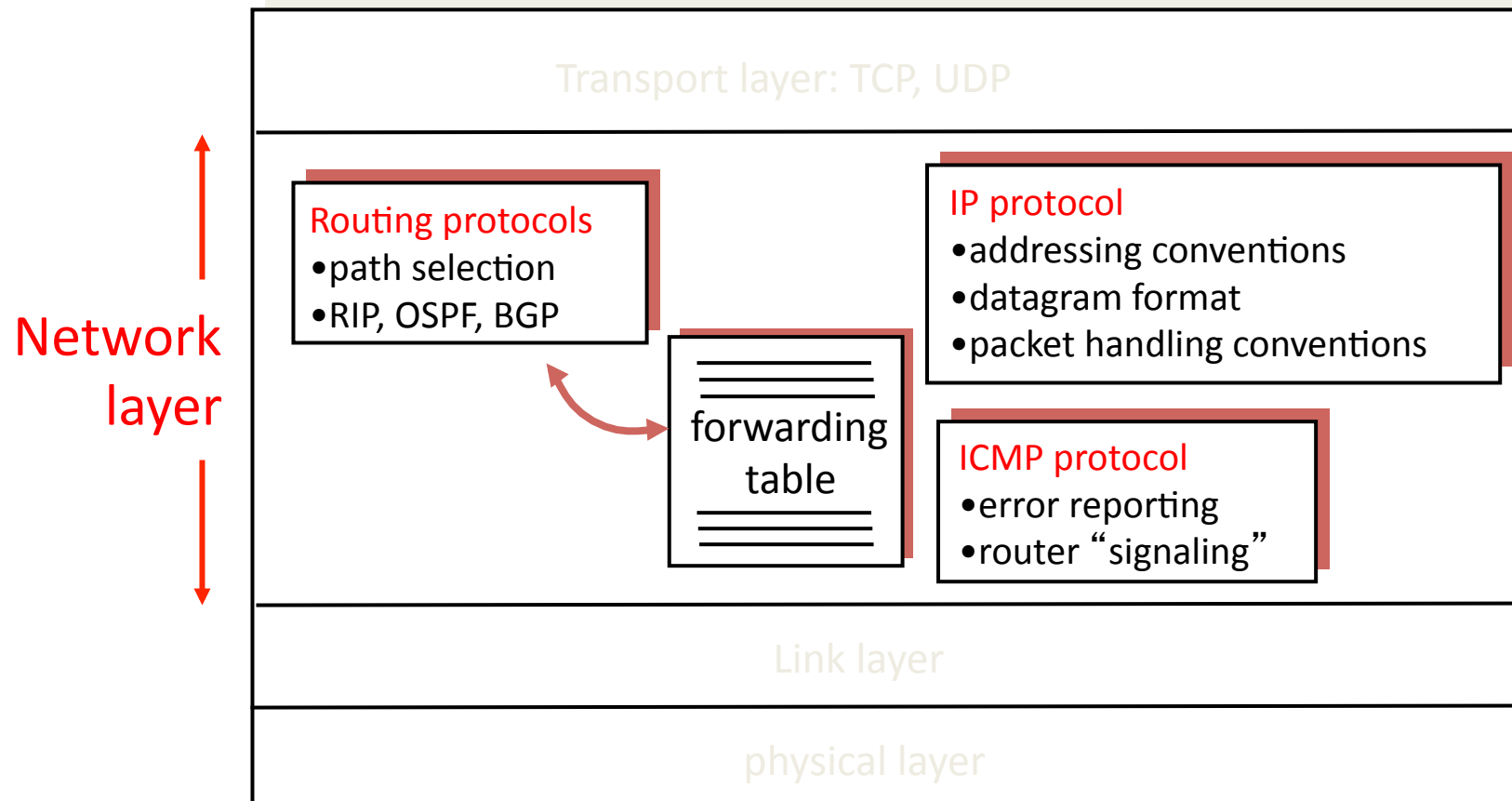
Many TCP Flows

- Independent, desynchronized
- Central limit theorem says the aggregate becomes Gaussian
- Variance (buffer size) decreases as N increases



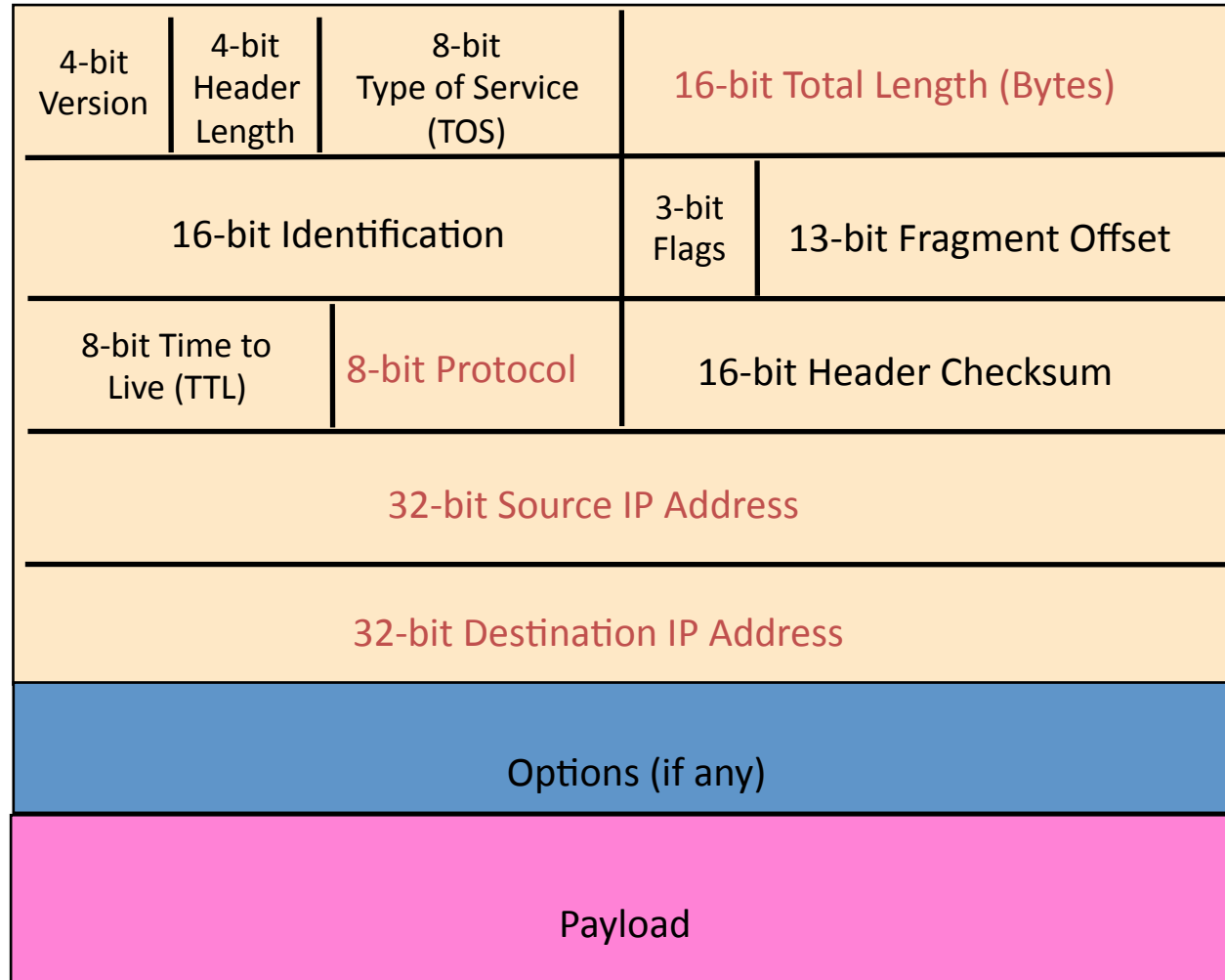
The Internet version of a Network layer

Host, router network layer functions:



IPv4 Packet Structure

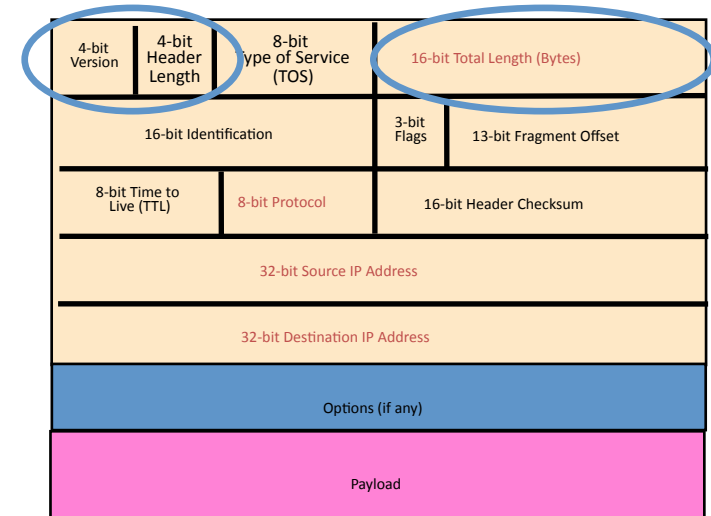
20 Bytes of Standard Header, then Options



(Packet) Network Tasks One-by-One

- Read packet correctly
- Get packet to the destination
- Get responses to the packet back to source
- Carry data
- Tell host what to do with packet once arrived
- Specify any special network handling of the packet
- Deal with problems that arise along the path

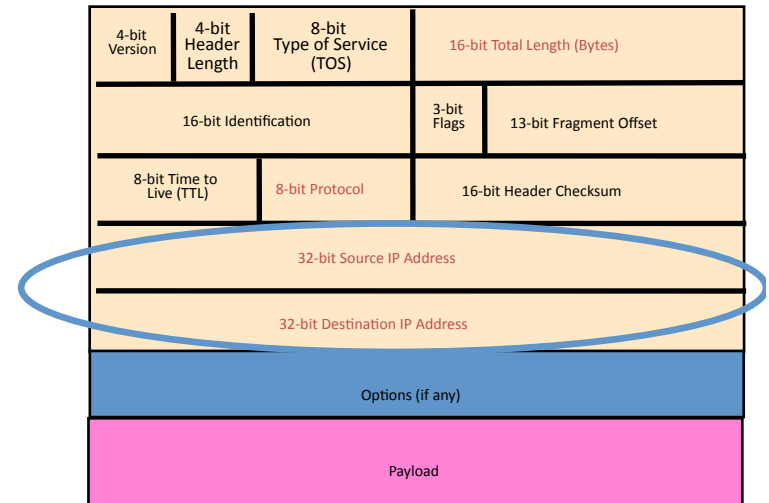
Reading Packet Correctly



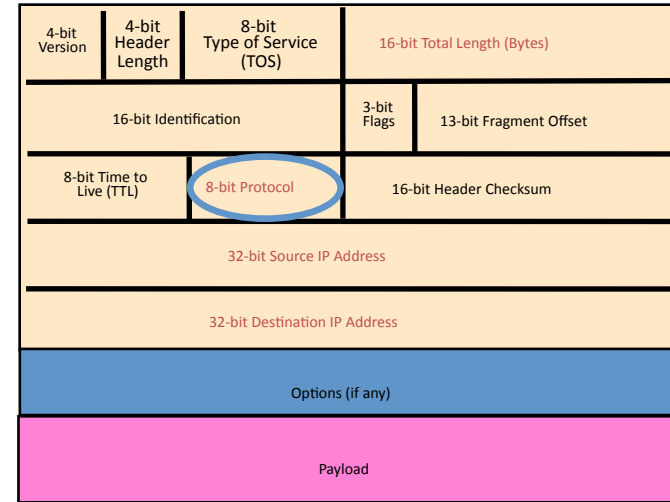
- Version number (4 bits)
 - Indicates the version of the IP protocol
 - Necessary to know what other fields to expect
 - Typically “4” (for IPv4), and sometimes “6” (for IPv6)
- Header length (4 bits)
 - Number of 32-bit words in the header
 - Typically “5” (for a 20-byte IPv4 header)
 - Can be more when IP **options** are used
- Total length (16 bits)
 - Number of bytes in the packet
 - Maximum size is 65,535 bytes ($2^{16} - 1$)
 - ... though underlying links may impose smaller limits

Getting Packet to Destination and Back

- Two IP addresses
 - Source IP address (32 bits)
 - Destination IP address (32 bits)
- Destination address
 - Unique identifier/locator for the receiving host
 - Allows each node to make forwarding decisions
- Source address
 - Unique identifier/locator for the sending host
 - Recipient can decide whether to accept packet
 - Enables recipient to send a reply back to source

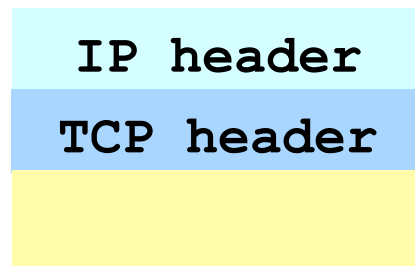


Telling Host How to Handle Packet

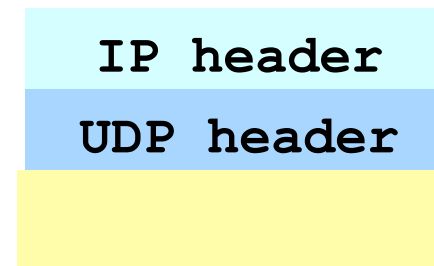


- Protocol (8 bits)
 - Identifies the higher-level protocol
 - Important for demultiplexing at receiving host
- Most common examples
 - E.g., “6” for the Transmission Control Protocol (TCP)
 - E.g., “17” for the User Datagram Protocol (UDP)

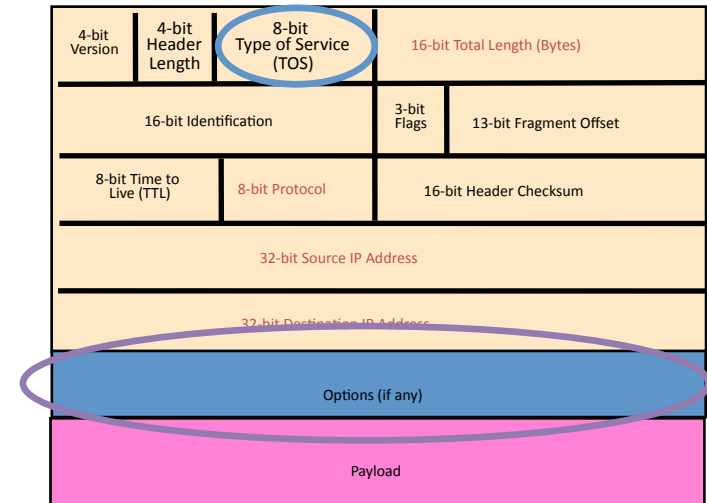
`protocol=6`



`protocol=17`



Special Handling

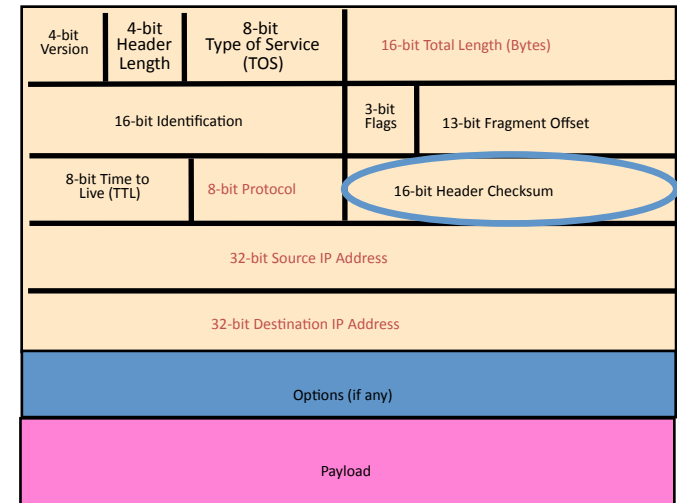


- Type-of-Service (8 bits)
 - Allow packets to be treated differently based on needs
 - E.g., low delay for audio, high bandwidth for bulk transfer
 - Has been redefined several times
- Options

Potential Problems

- Header Corrupted: **Checksum**
- Loop: **TTL**
- Packet too large: **Fragmentation**

Header Corruption



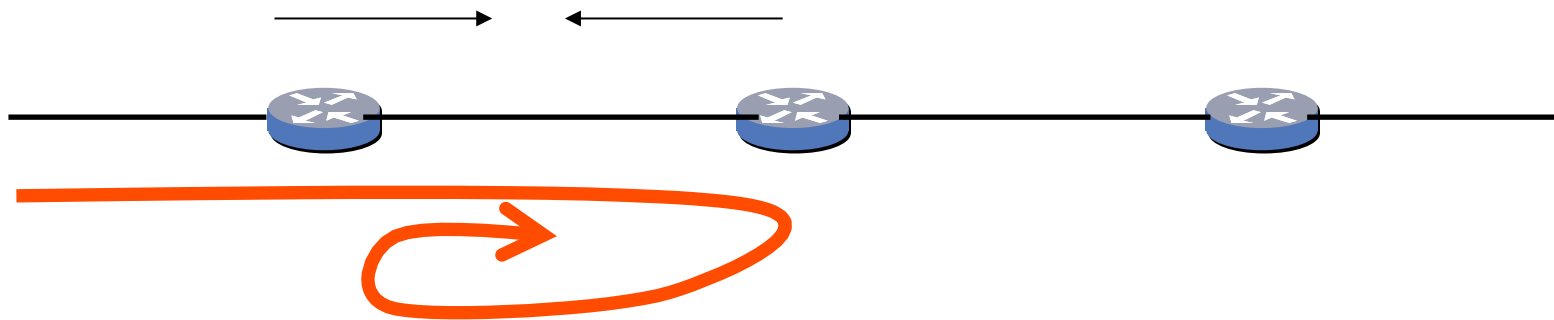
- Checksum (16 bits)
 - Particular form of checksum over packet header
- If not correct, router discards packets
 - So it doesn't act on bogus information
- Checksum recalculated at every router
 - **Why?**
 - **Why include TTL?**
 - **Why only header?**

Preventing Loops

(aka Internet Zombie plan)

4-bit Version	4-bit Header Length	8-bit Type of Service (TOS)	16-bit Total Length (Bytes)	
16-bit Identification			3-bit Flags	13-bit Fragment Offset
8-bit Time to Live (TTL)		8-bit Protocol	16-bit Header Checksum	
32-bit Source IP Address				
32-bit Destination IP Address				
Options (if any)				
Payload				

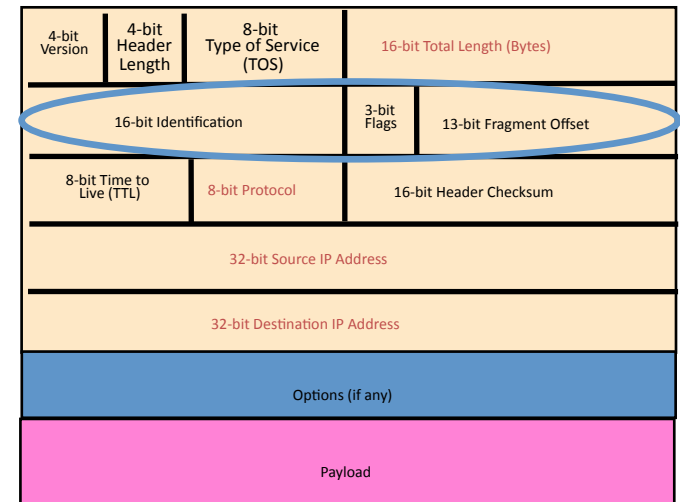
- Forwarding loops cause packets to cycle forever
 - As these accumulate, eventually consume **all** capacity



- Time-to-Live (TTL) Field (8 bits)
 - Decrement at each hop, packet discarded if reaches 0
 - ...and “time exceeded” message is sent to the source
 - Using “ICMP” control message; basis for **traceroute**

Fragmentation

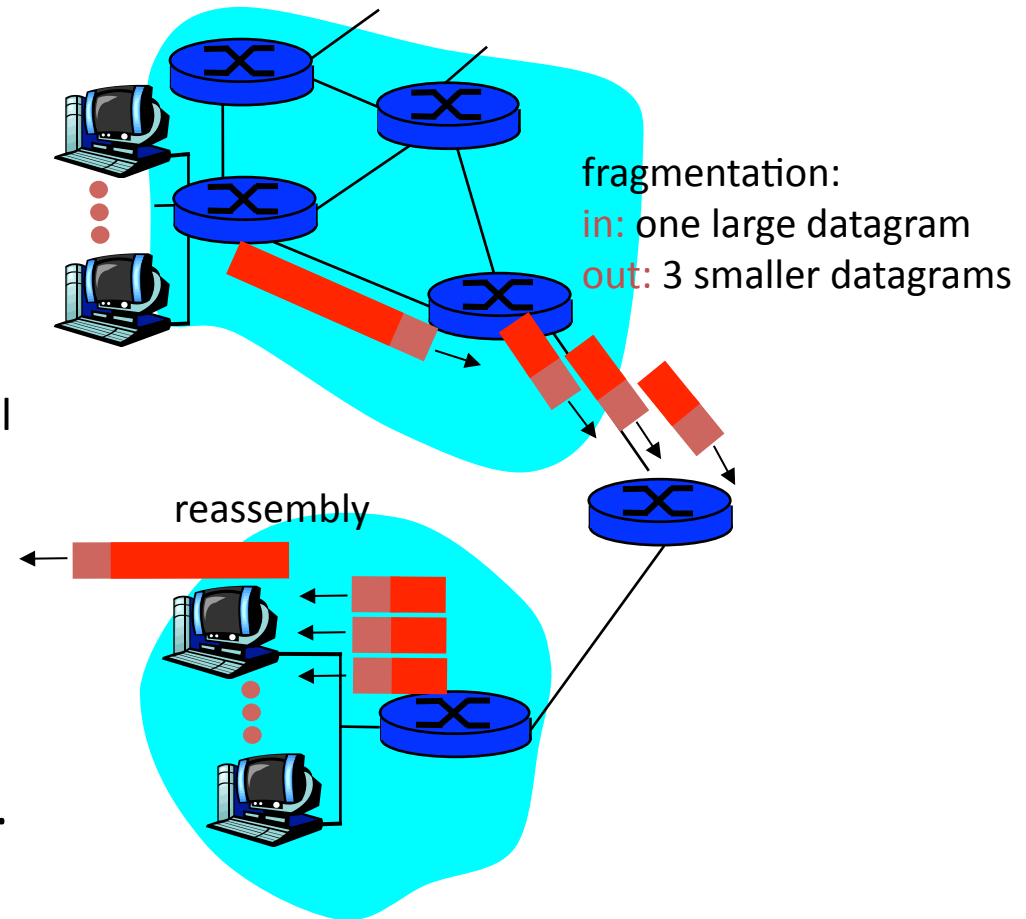
(some assembly required)



- Fragmentation: when forwarding a packet, an Internet router can **split** it into multiple pieces (“fragments”) if too big for next hop link
- Must **reassemble** to recover original packet
 - Need fragmentation information (32 bits)
 - Packet **identifier**, **flags**, and fragment **offset**

IP Fragmentation & Reassembly

- network links have MTU (max.transfer size) - largest possible link-level frame.
 - different link types, different MTUs
- 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
- IPv6 does things differently...



IP Fragmentation and Reassembly

Example

- ❑ 4000 byte datagram
- ❑ MTU = 1500 bytes

	length =4000	ID =x	fragflag =0	offset =0	
--	-----------------	----------	----------------	--------------	--

One large datagram becomes several smaller datagrams

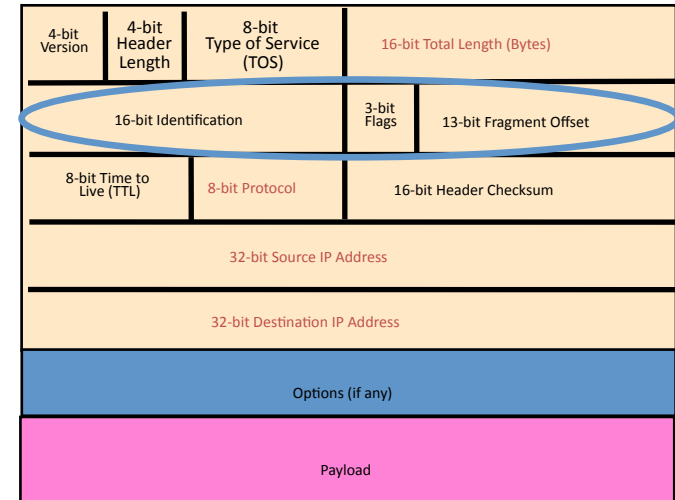
1480 bytes in data field

offset =
1480/8

	length =1500	ID =x	fragflag =1	offset =0	
	length =1500	ID =x	fragflag =1	offset =185	
	length =1040	ID =x	fragflag =0	offset =370	

Pop quiz question: What happens when a fragment is lost?

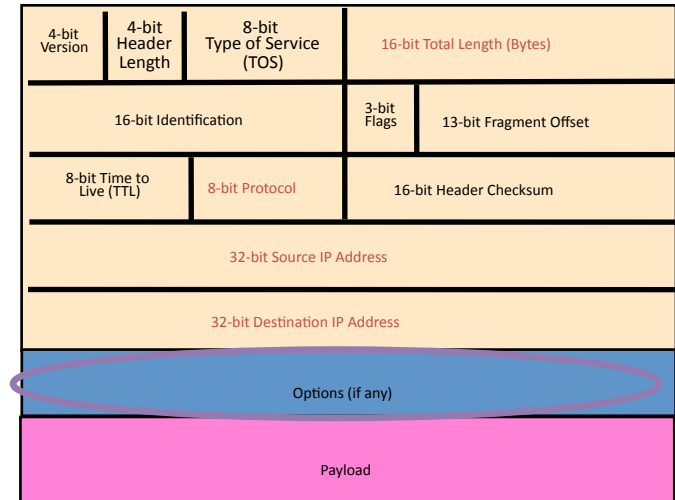
Fragmentation Details



- Identifier (16 bits): used to tell which fragments belong together
- Flags (3 bits):
 - Reserved (**RF**): unused bit
 - Don't Fragment (**DF**): instruct routers to **not** fragment the packet even if it won't fit
 - Instead, they **drop** the packet and send back a “Too Large” ICMP control message
 - Forms the basis for “Path MTU Discovery”
 - More (**MF**): this fragment is not the last one
- Offset (13 bits): what part of datagram this fragment covers **in 8-byte units**

Pop quiz question: Why do frags use offset and not a frag number?

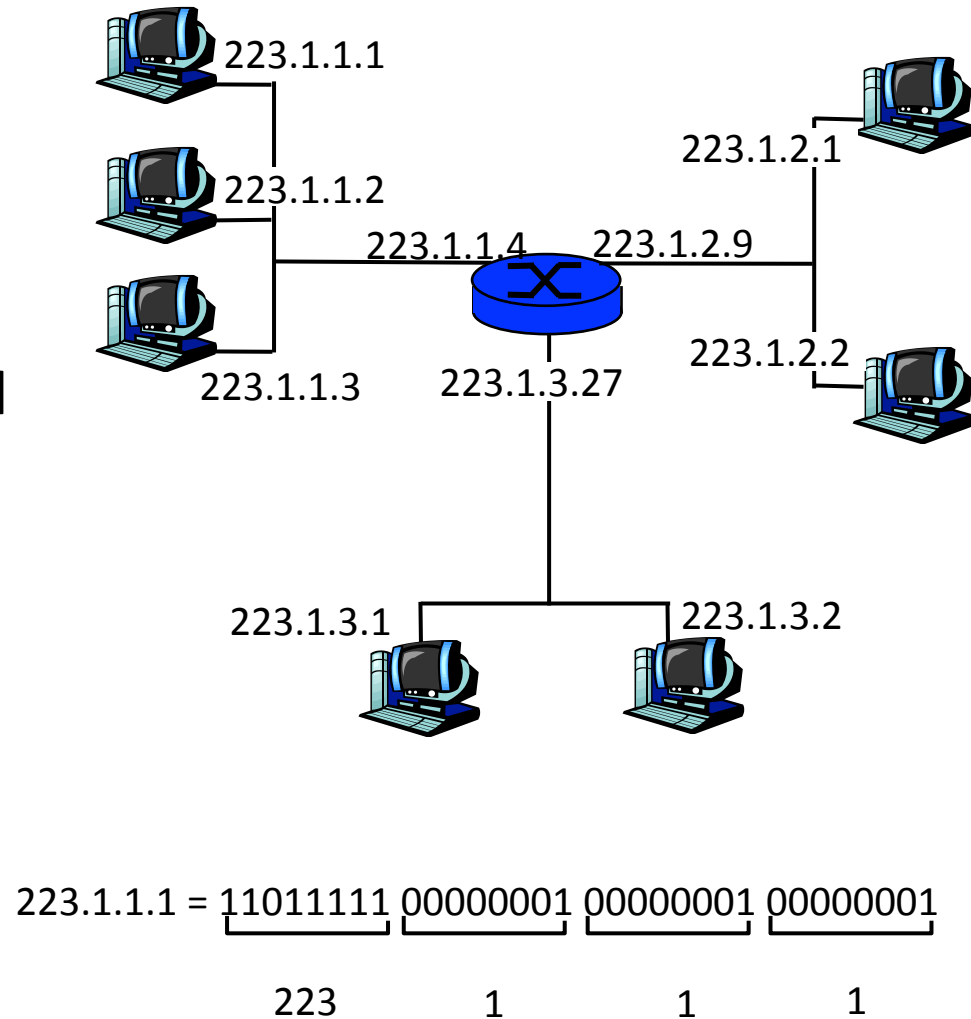
Options



- End of Options List
- No Operation (padding between options)
- Record Route
- Strict Source Route
- Loose Source Route
- Timestamp
- Traceroute
- Router Alert
-

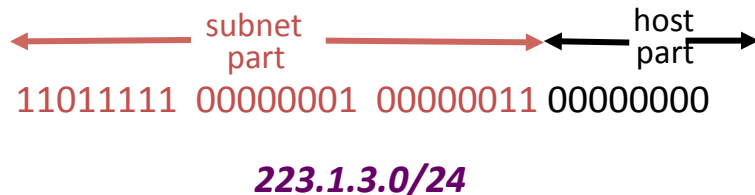
IP Addressing: introduction

- **IP address:** 32-bit identifier for host, router *interface*
- **interface:** connection between host/router and physical link
 - router's typically have multiple interfaces
 - host typically has one interface
 - IP addresses associated with each interface



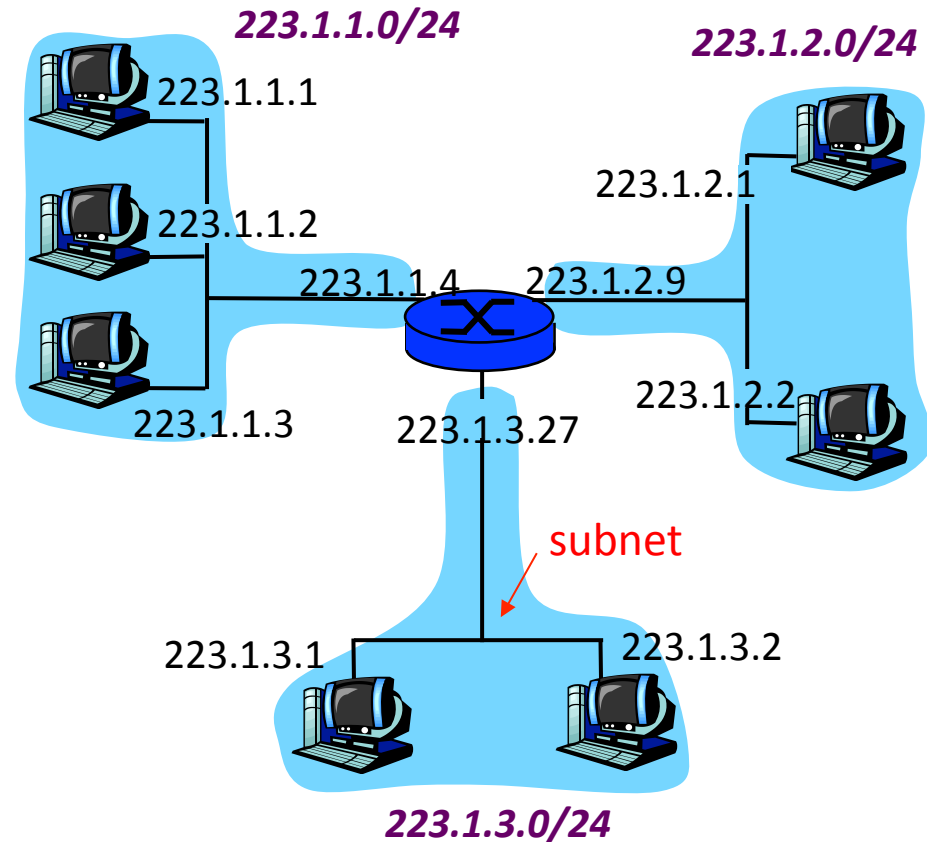
Subnets

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



CIDR: Classless InterDomain Routing

- subnet portion of address of arbitrary length
- address format: **a.b.c.d/x**, where x is # bits in subnet portion of address



Subnet mask: /24

network consisting of 3 subnets

IP addresses: how to get one?

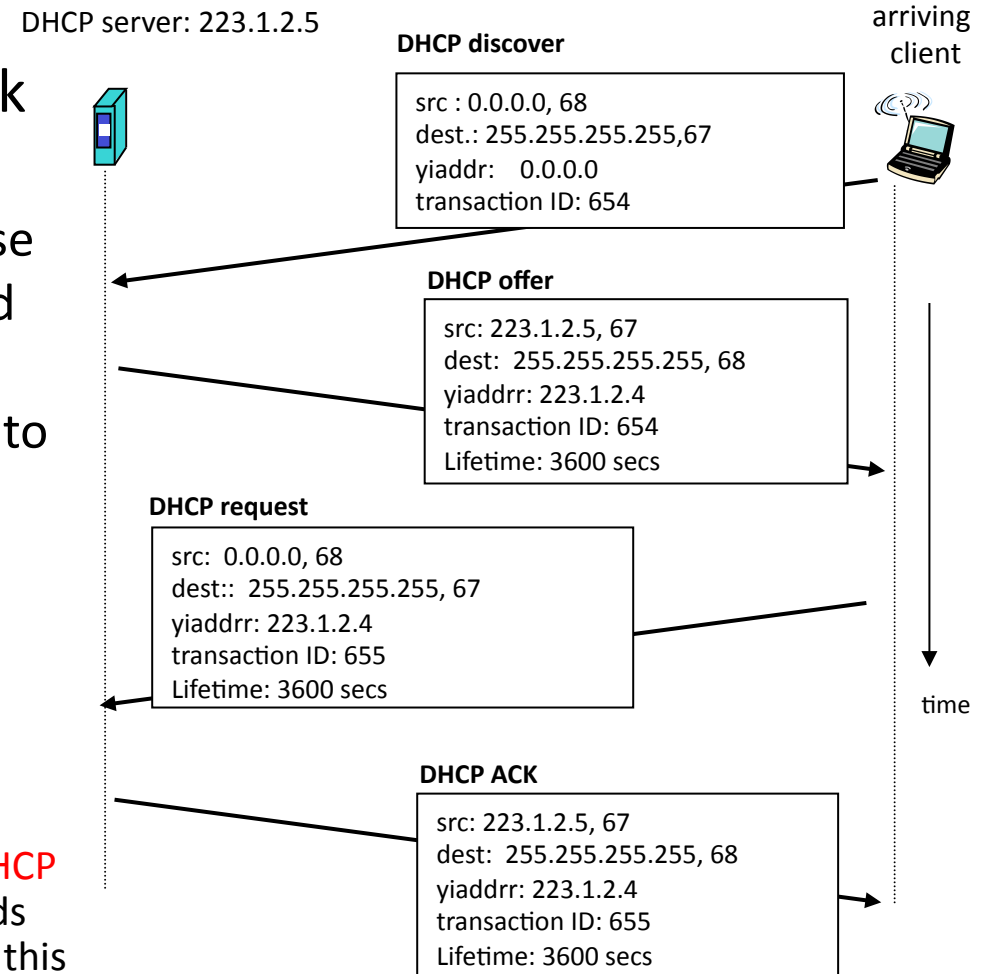
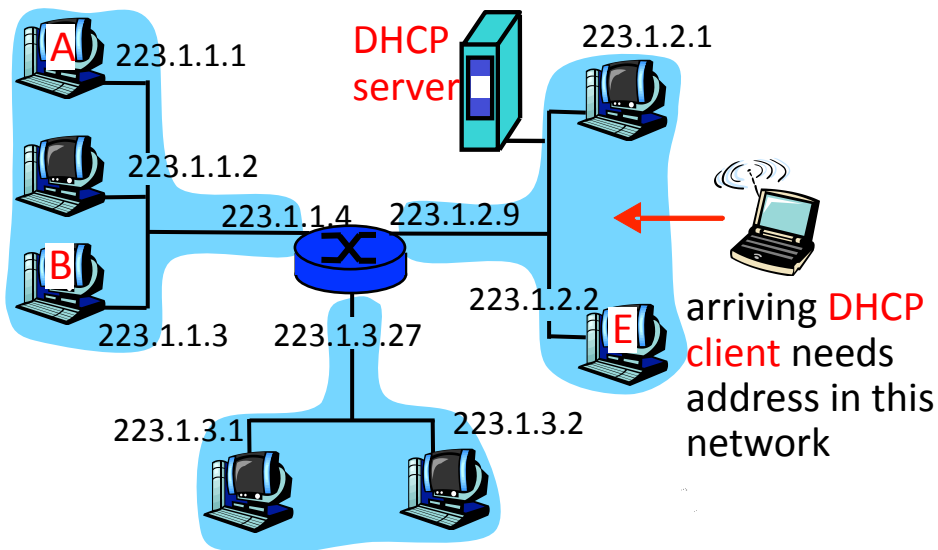
Q: How does a *host* get IP address?

- hard-coded by system admin in a file
 - Windows: control-panel->network->configuration->tcp/ip->properties
 - UNIX: /etc/rc.config
- **DHCP: Dynamic Host Configuration Protocol**: dynamically get address from as server
 - “plug-and-play”

DHCP client-server scenario

Goal: 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 an “on”)
- Support for mobile users who want to join network (more shortly)



IP addresses: how to get one?

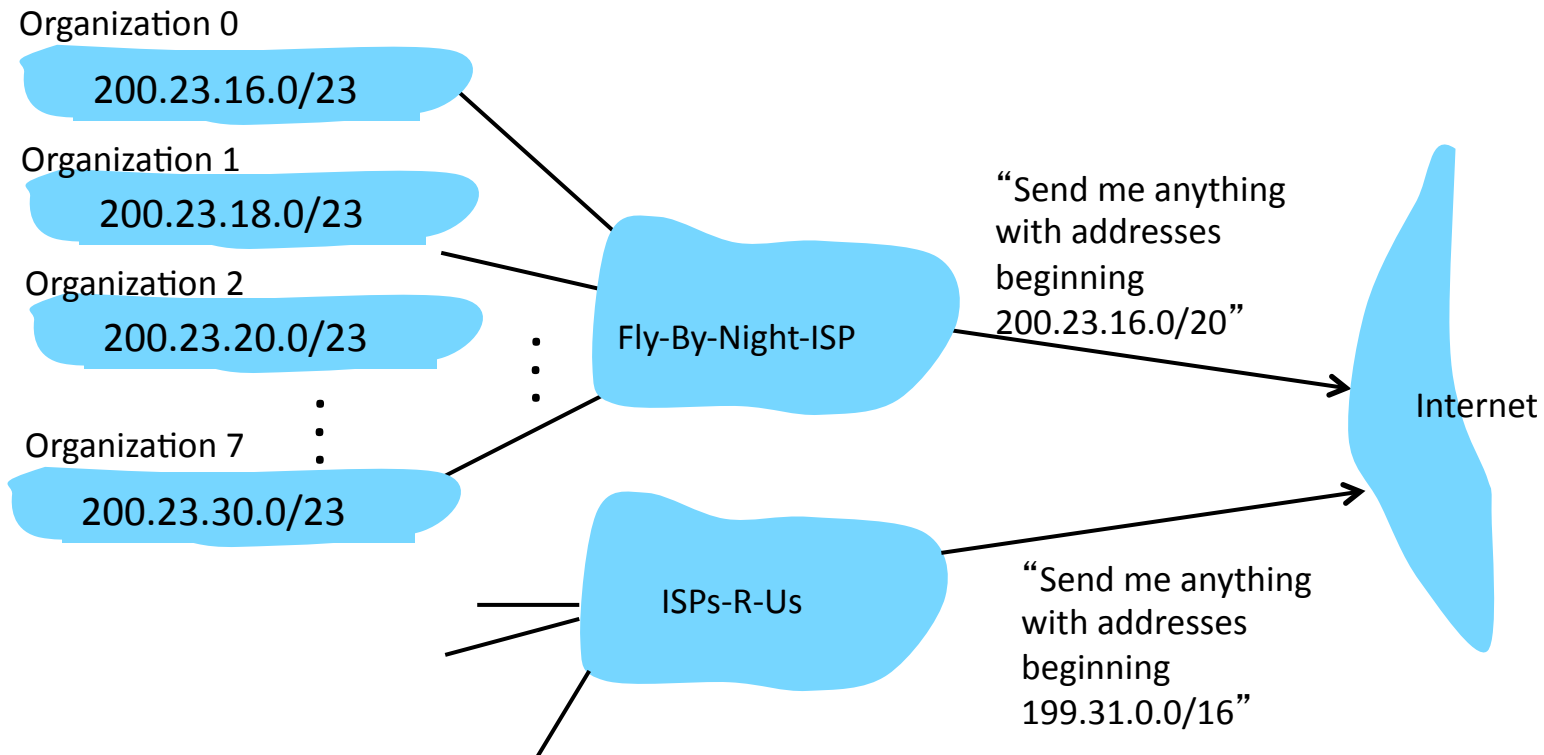
Q: How does *network* get subnet part of IP addr?

A: gets allocated portion of its provider ISP's address space

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

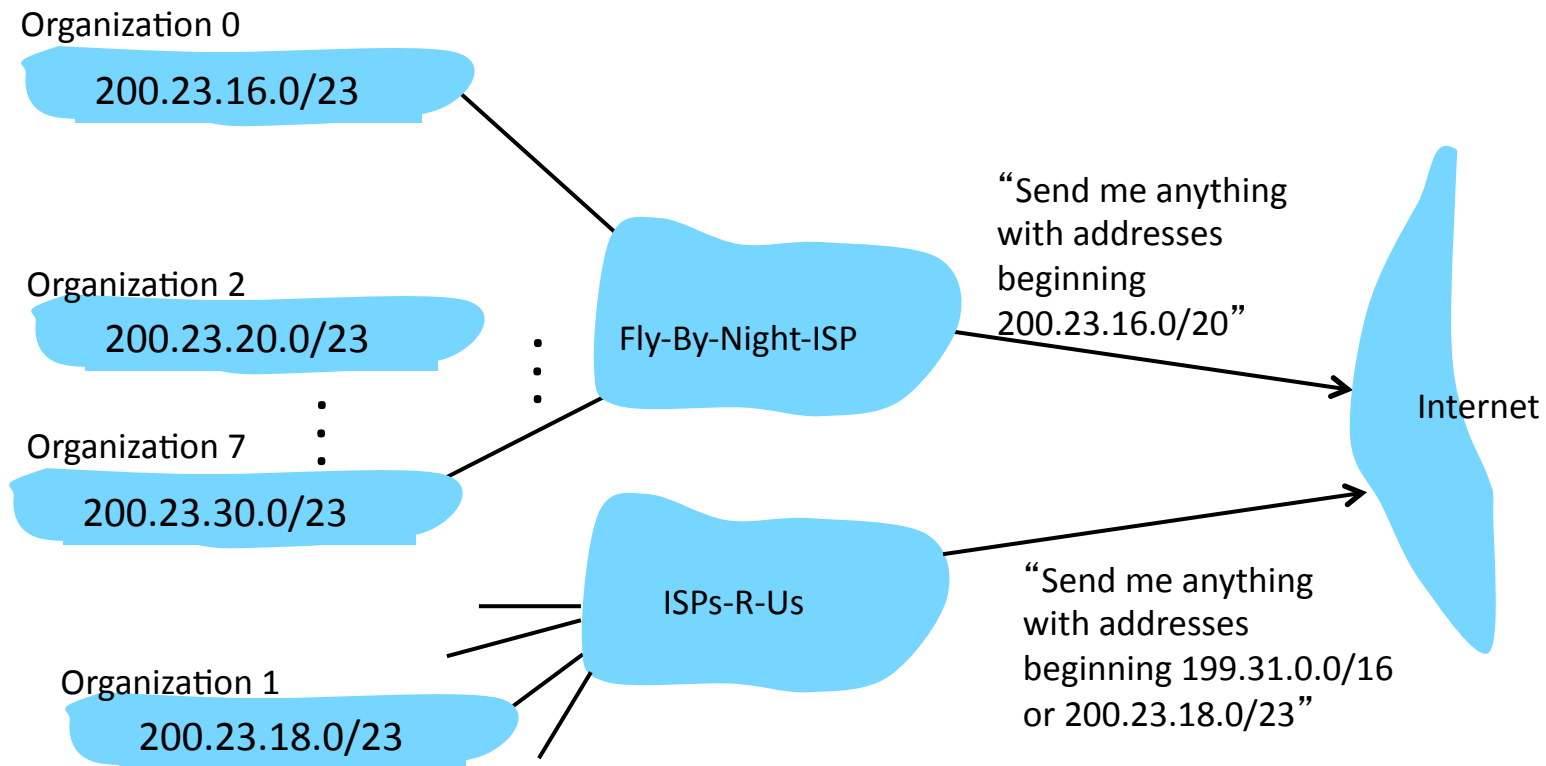
Hierarchical addressing: route aggregation

Hierarchical addressing allows efficient advertisement of routing information:



Hierarchical addressing: more specific routes

ISPs-R-Us has a more specific route to Organization 1



IP addressing: the last word...

Q: How does an ISP get block of addresses?

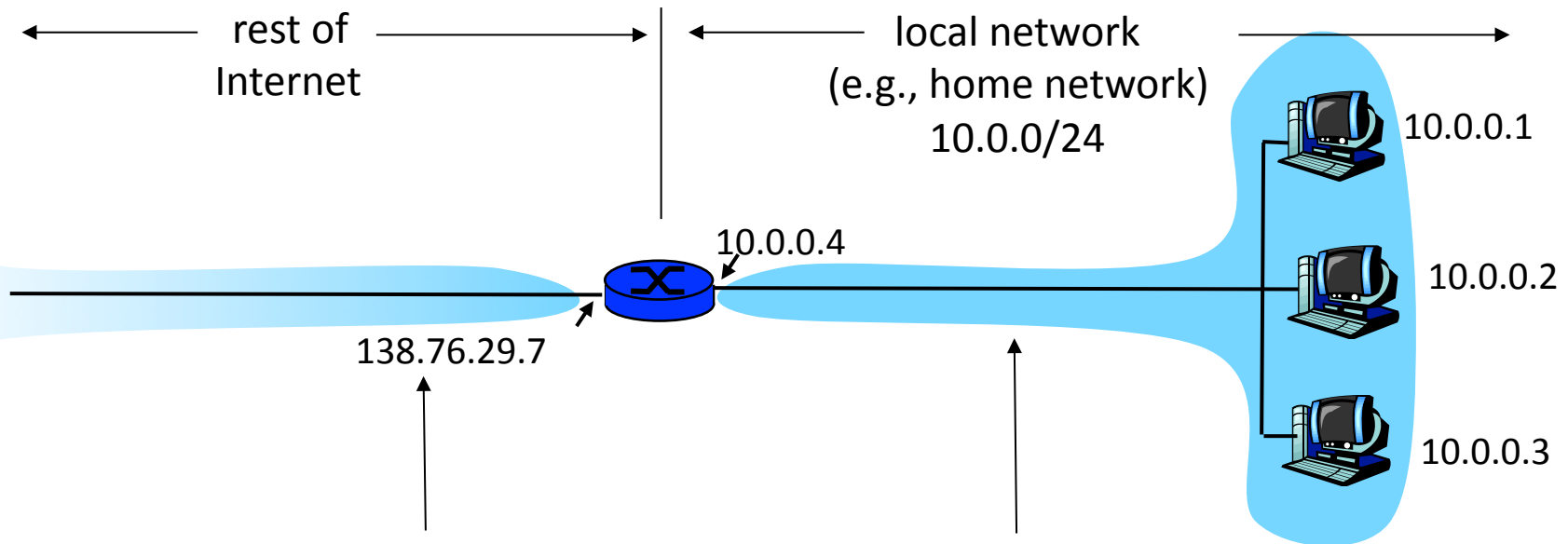
A: **ICANN**: Internet **C**orporation for **A**ssigned

Names and **N**umbers

- allocates addresses
- manages DNS
- assigns domain names, resolves disputes

Cant get more IP addresses? well there is always.....

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

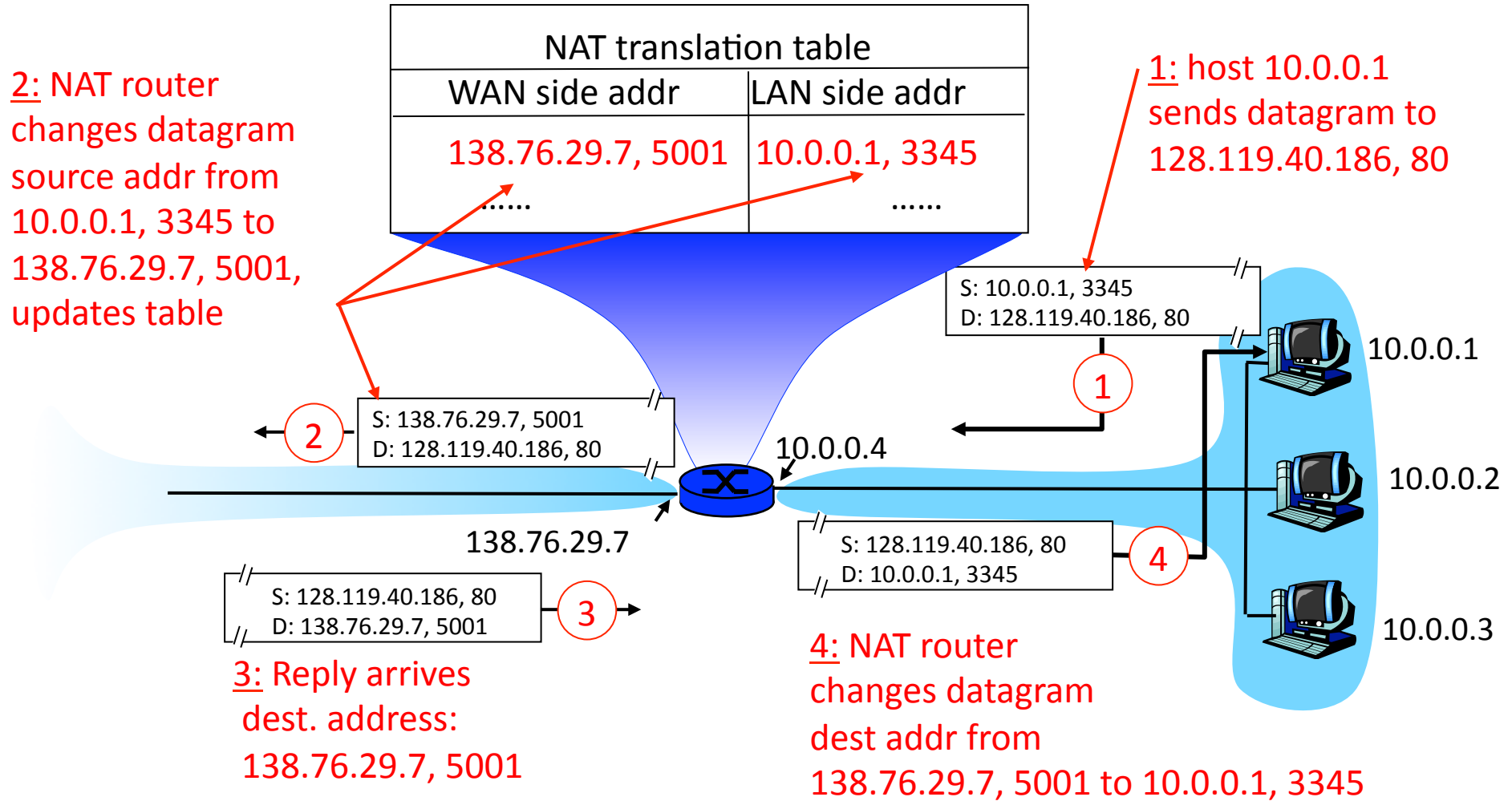
- **Motivation:** local network uses just one IP address as far as outside world is concerned:
 - range of addresses not needed from ISP: just one IP address for all devices
 - can change addresses of devices in local network without notifying outside world
 - can change ISP without changing addresses of devices in local network
 - devices inside local net not explicitly addressable, visible by outside world (a security plus).

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

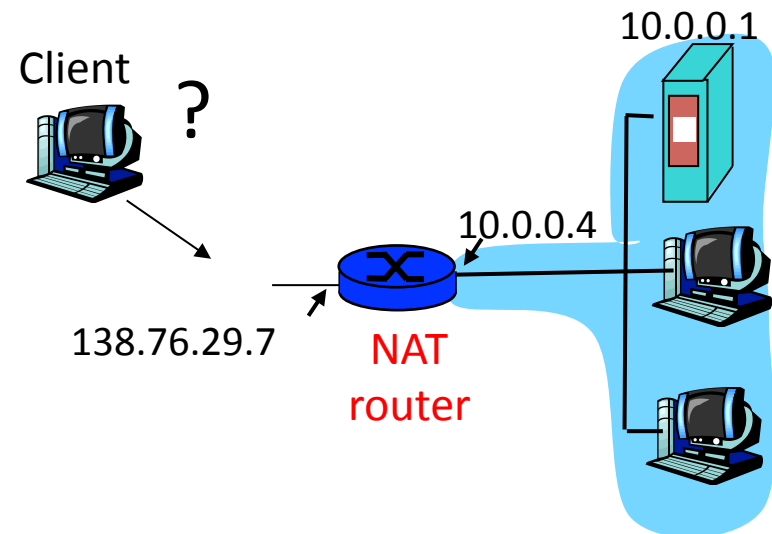


NAT: Network Address Translation

- 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, eg, P2P applications
 - address shortage should instead be solved by IPv6

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 NATted address: 138.76.29.7
- solution 1: statically configure NAT to forward incoming connection requests at given port to server
 - e.g., (138.76.29.7, port 2500) always forwarded to 10.0.0.1 port 25000

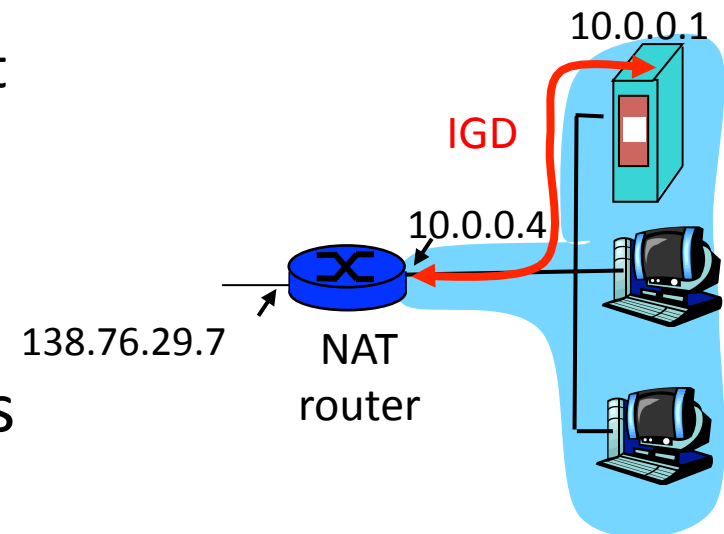


NAT traversal problem

- solution 2: Universal Plug and Play (UPnP) Internet Gateway Device (IGD) Protocol. Allows NATted host to:

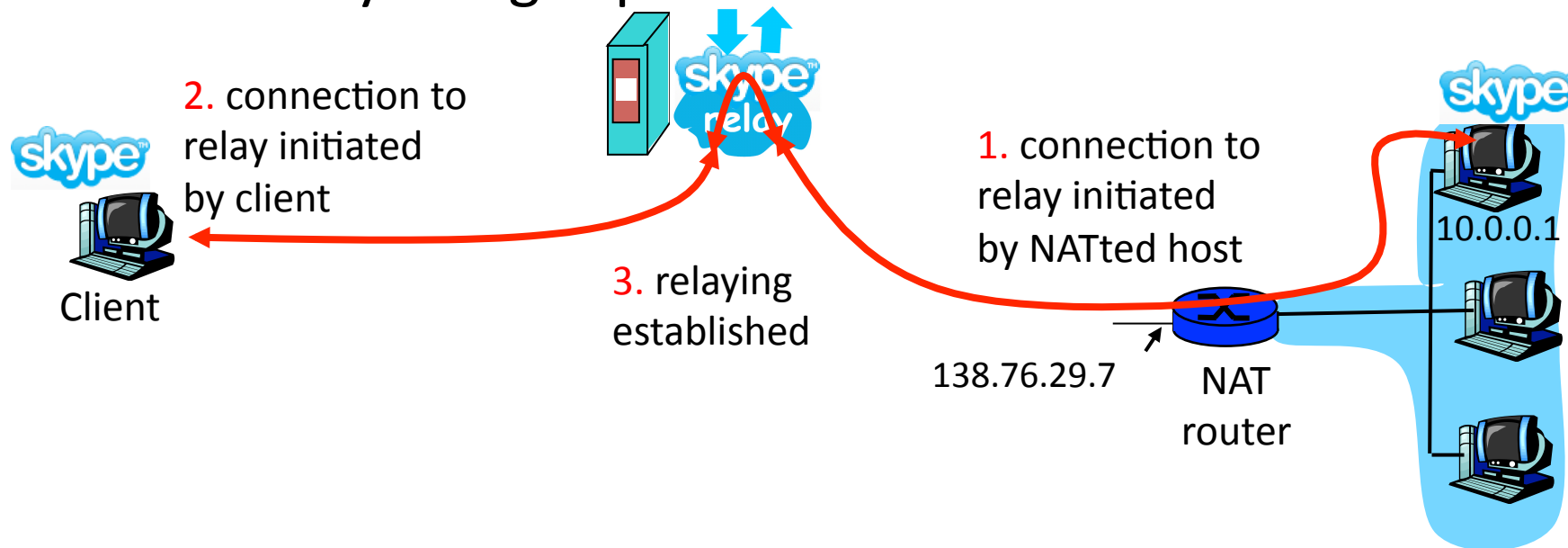
- ❖ learn public IP address (138.76.29.7)
- ❖ add/remove port mappings (with lease times)

i.e., automate static NAT port map configuration



NAT traversal problem

- solution 3: relaying (used in Skype)
 - NATed client establishes connection to relay
 - External client connects to relay
 - relay bridges packets between to connections



Remember this? Traceroute at work...

traceroute: rio.cl.cam.ac.uk to munnari.oz.au

(tracepath on pwf is similar)

Three delay measurements from
rio.cl.cam.ac.uk to gatwick.net.cl.cam.ac.uk

trans-continent
link

traceroute munnari.oz.au

traceroute to munnari.oz.au (202.29.151.3), 30 hops max, 60 byte packets

```
1  gatwick.net.cl.cam.ac.uk (128.232.32.2) 0.416 ms 0.384 ms 0.427 ms
2  cl-sby.route-nwest.net.cam.ac.uk (193.60.89.9) 0.393 ms 0.440 ms 0.494 ms
3  route-nwest.route-mill.net.cam.ac.uk (192.84.5.137) 0.407 ms 0.448 ms 0.501 ms
4  route-mill.route-enet.net.cam.ac.uk (192.84.5.94) 1.006 ms 1.091 ms 1.163 ms
5  xe-11-3-0.camb-rbr1.eastern.ja.net (146.97.130.1) 0.300 ms 0.313 ms 0.350 ms
6  ae24.lowdss-sbr1.ja.net (146.97.37.185) 2.679 ms 2.664 ms 2.712 ms
7  ae28.londhx-sbr1.ja.net (146.97.33.17) 5.955 ms 5.953 ms 5.901 ms
8  janet.mx1.lon.uk.geant.net (62.40.124.197) 6.059 ms 6.066 ms 6.052 ms
9  ae0.mx1.par.fr.geant.net (62.40.98.77) 11.742 ms 11.779 ms 11.724 ms
10 ae1.mx1.mad.es.geant.net (62.40.98.64) 27.751 ms 27.734 ms 27.704 ms
11 mb-so-02-v4.bb.tein3.net (202.179.249.117) 138.296 ms 138.314 ms 138.282 ms
12 sg-so-04-v4.bb.tein3.net (202.179.249.53) 196.303 ms 196.293 ms 196.264 ms
13 th-pr-v4.bb.tein3.net (202.179.249.66) 225.153 ms 225.178 ms 225.196 ms
14 pyt-thairen-to-02-bdr-pyt.uni.net.th (202.29.12.10) 225.163 ms 223.343 ms 223.363 ms
15 202.28.227.126 (202.28.227.126) 241.038 ms 240.941 ms 240.834 ms
16 202.28.221.46 (202.28.221.46) 287.252 ms 287.306 ms 287.282 ms
17 * * *
18 * * *
19 * * *
20 coe-gw.psu.ac.th (202.29.149.70) 241.681 ms 241.715 ms 241.680 ms
21 munnari.OZ.AU (202.29.151.3) 241.610 ms 241.636 ms 241.537 ms
```

* means no response (probe lost, router not replying)

Traceroute and ICMP

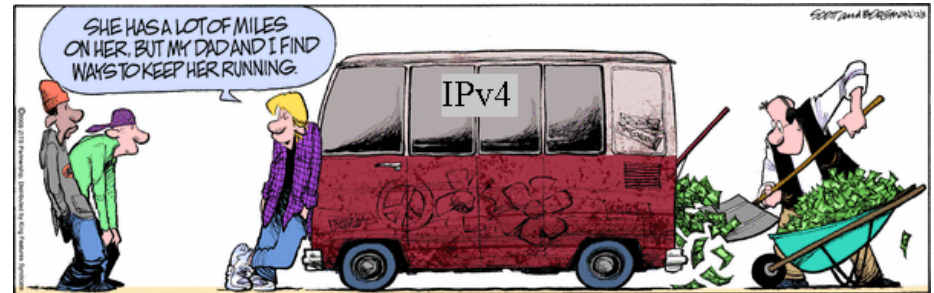
- Source sends series of UDP segments to dest
 - First has TTL =1
 - Second has TTL=2, etc.
 - Unlikely port number
 - When nth datagram arrives to nth router:
 - Router discards datagram
 - And sends to source an ICMP message (type 11, code 0)
 - Message includes name of router& IP address
 - When ICMP message arrives, source calculates RTT
 - Traceroute does this 3 times
- Stopping criterion
- UDP segment eventually arrives at destination host
 - Destination returns ICMP “host unreachable” packet (type 3, code 3)
 - When source gets this ICMP, stops.

ICMP: Internet Control Message Protocol

- used by hosts & routers to communicate 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

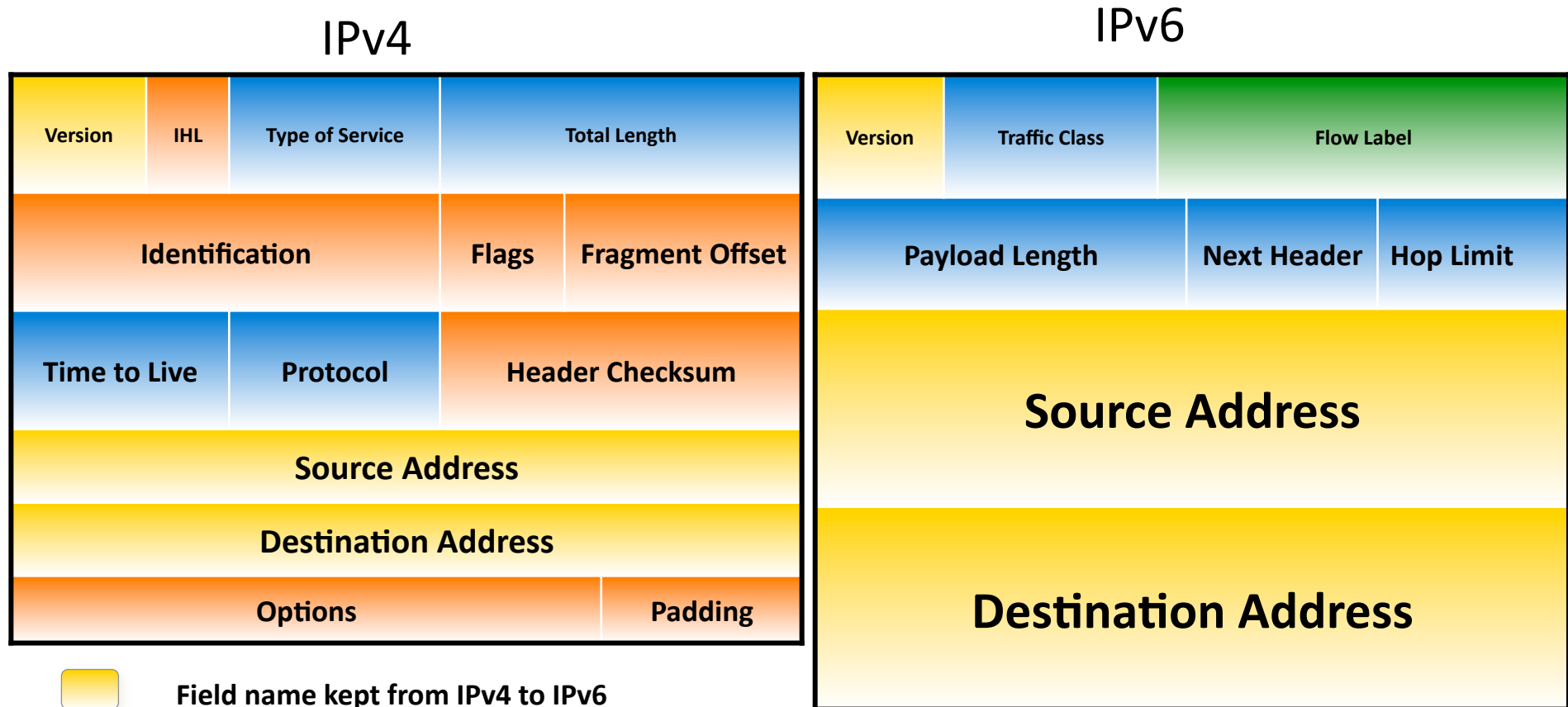
<u>Type</u>	<u>Code</u>	<u>description</u>
0	0	echo reply (ping)
3	0	dest. network unreachable
3	1	dest host unreachable
3	2	dest protocol unreachable
3	3	dest port unreachable
3	6	dest network unknown
3	7	dest host unknown
4	0	source quench (congestion control - not used)
8	0	echo request (ping)
9	0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





IPv6



- Motivated (prematurely) by address exhaustion
 - Address field *four* times as long
- Steve Deering focused on simplifying IP
 - Got rid of all fields that were not absolutely necessary
 - “Spring Cleaning” for IP
- Result is an elegant, if unambitious, protocol

IPv4 and IPv6 Header Comparison



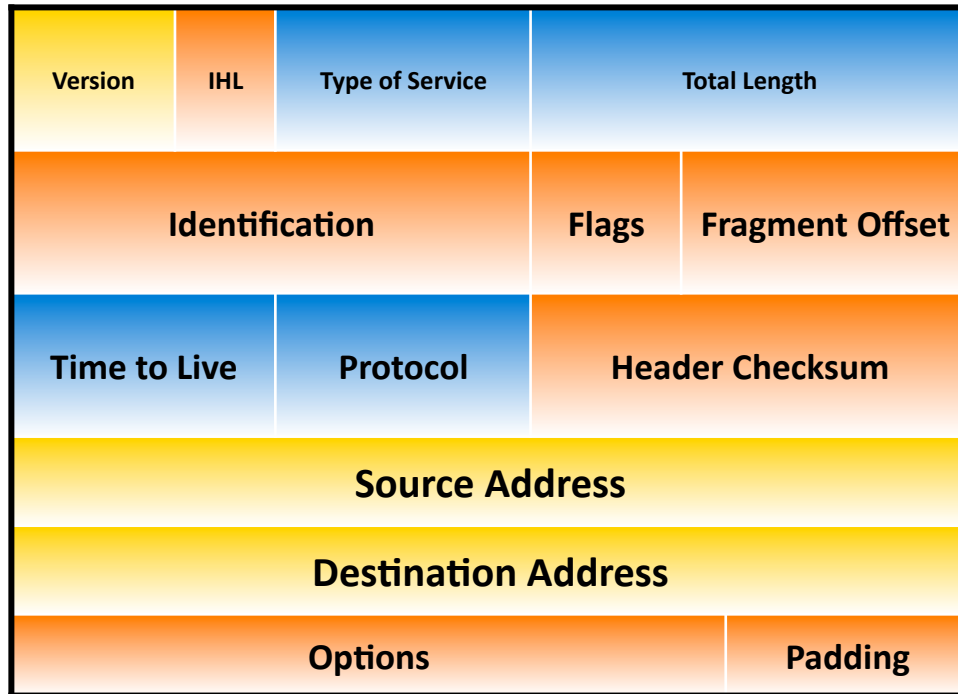
-  Field name kept from IPv4 to IPv6
-  Fields not kept in IPv6
-  Name & position changed in IPv6
-  New field in IPv6

Summary of Changes

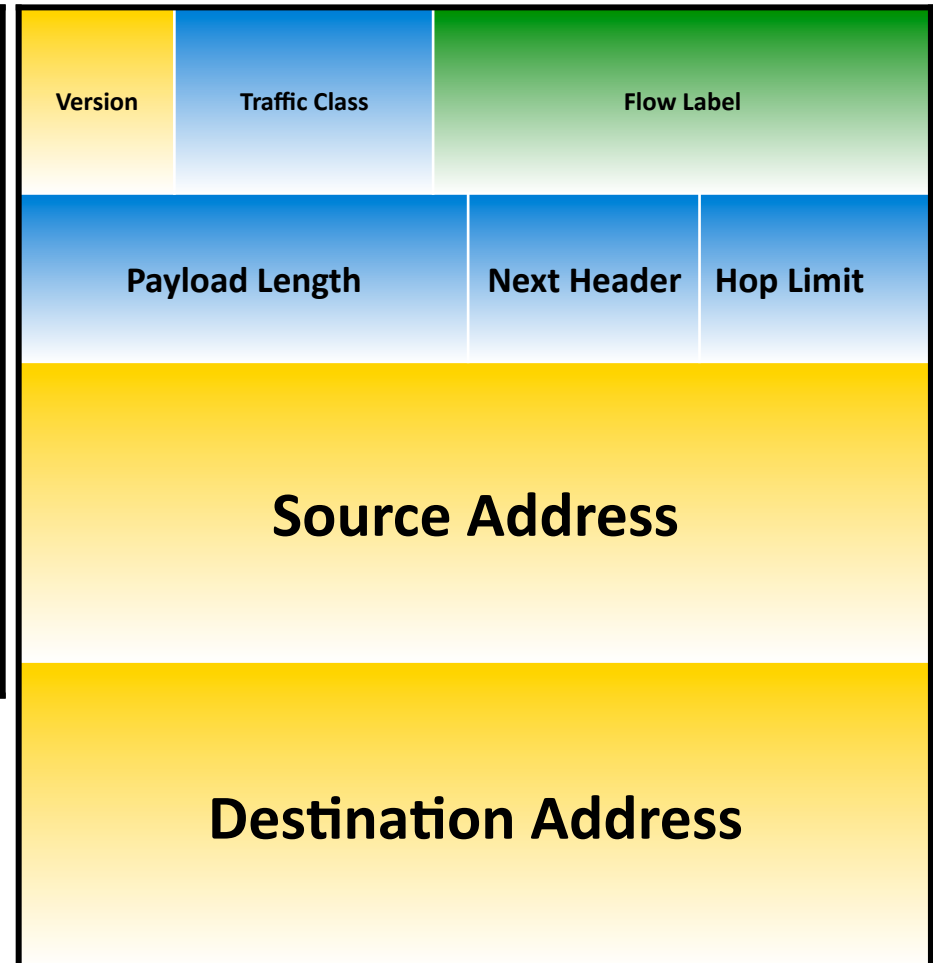
- Eliminated fragmentation *(why?)*
- Eliminated header length *(why?)*
- Eliminated checksum *(why?)*
- New options mechanism (next header) *(why?)*
- Expanded addresses *(why?)*
- Added Flow Label *(why?)*





IPv4 and IPv6 Header Comparison

IPv4



IPv6



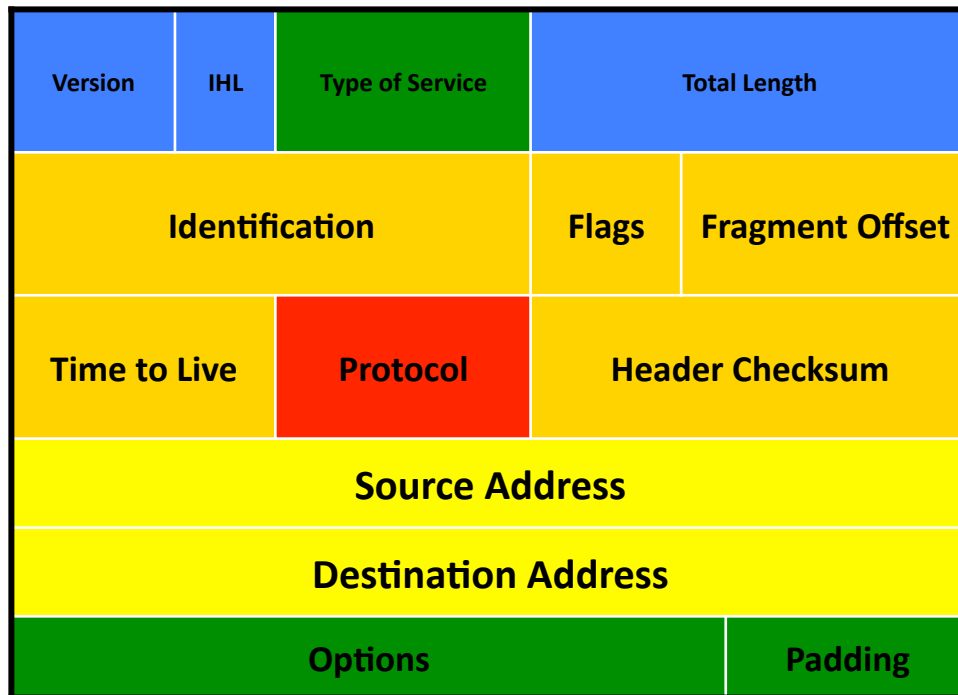
-  Field name kept from IPv4 to IPv6
-  Fields not kept in IPv6
-  Name & position changed in IPv6
-  New field in IPv6

Philosophy of Changes

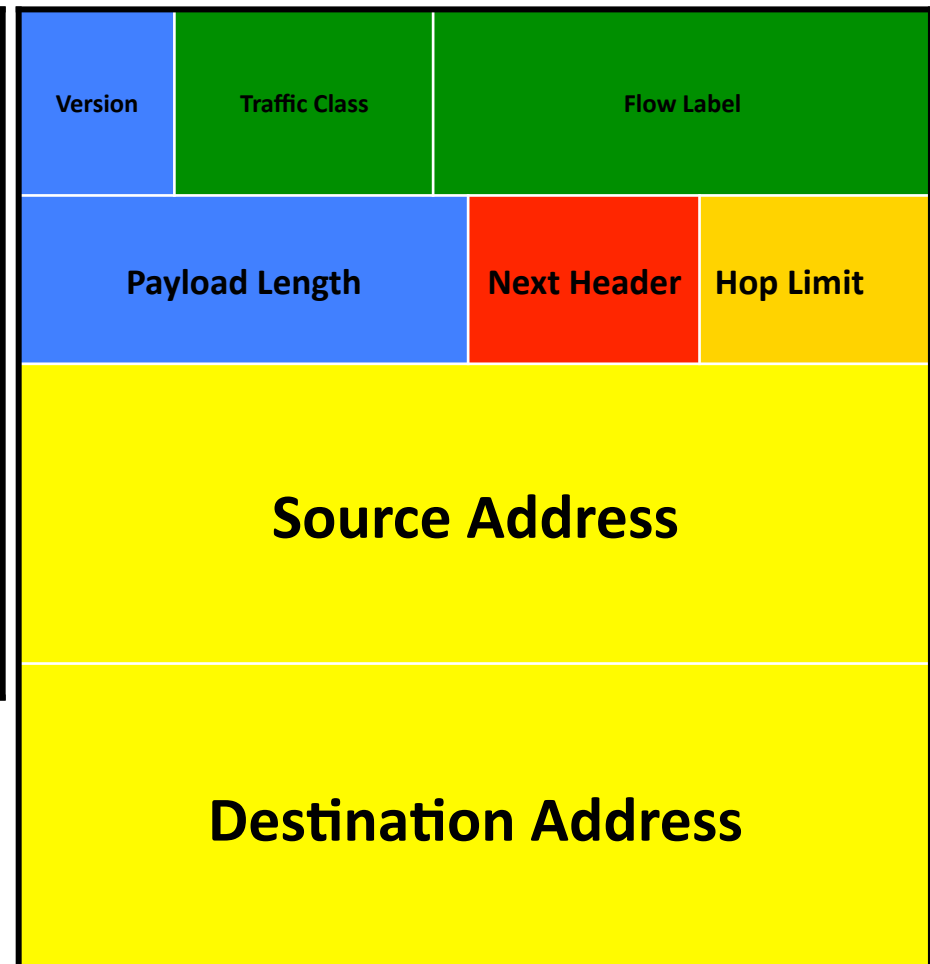
- Don't deal with problems: leave to ends
 - Eliminated fragmentation
 - Eliminated checksum
 - *Why retain TTL?*
- Simplify handling:
 - New options mechanism (uses next header approach)
 - Eliminated header length
 - *Why couldn't IPv4 do this?*
- Provide general flow label for packet
 - Not tied to semantics
 - Provides great flexibility





Comparison of Design Philosophy

IPv4



IPv6



-  To Destination and Back (expanded)
-  Deal with Problems (greatly reduced)
-  Read Correctly (reduced)
-  Special Handling (similar)

Transition From IPv4 To IPv6

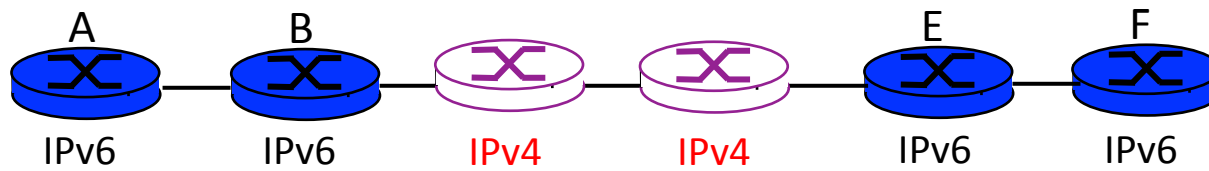
- Not all routers can be upgraded simultaneous
 - no “flag days”
 - How will the network operate with mixed IPv4 and IPv6 routers?
- *Tunneling*: IPv6 carried as payload in IPv4 datagram among IPv4 routers

Tunneling

Logical view:



Physical view:

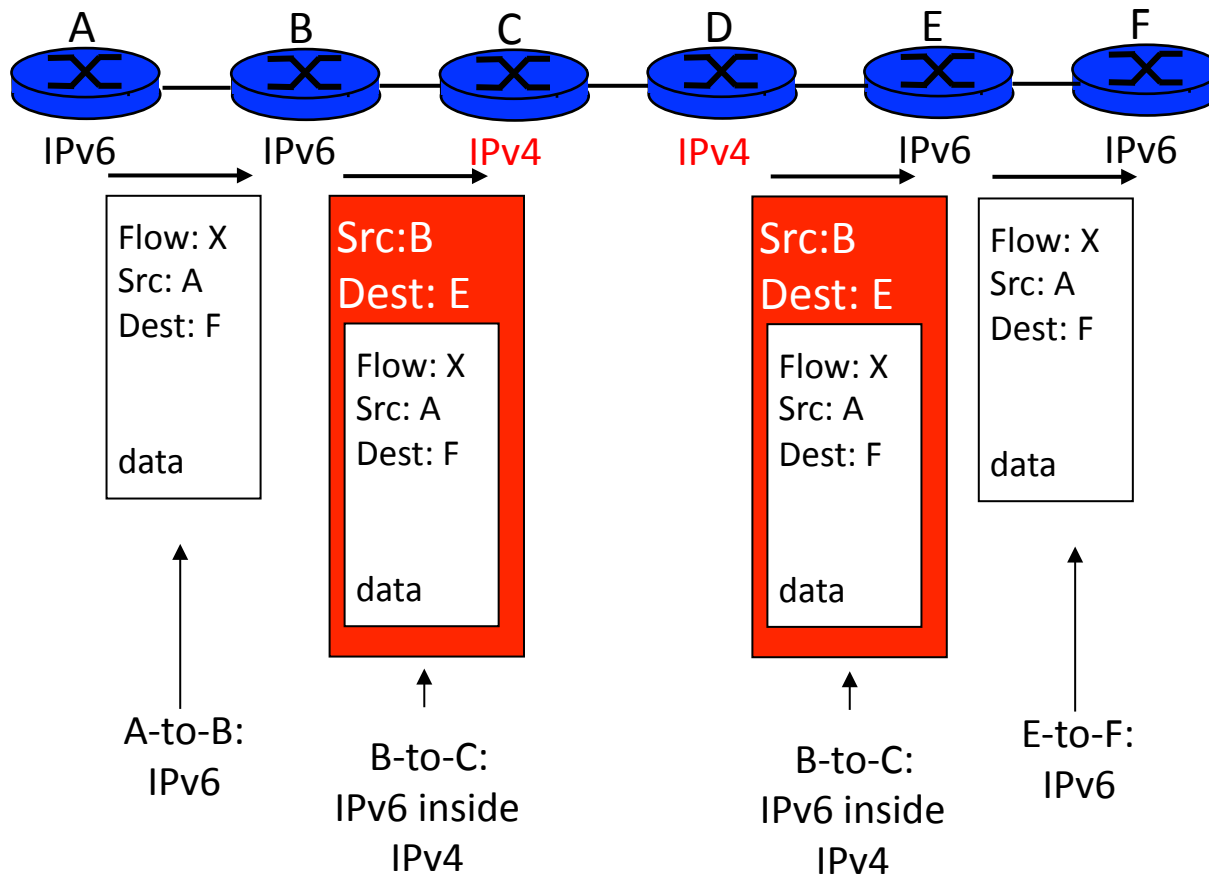


Tunneling

Logical view:



Physical view:



Improving on IPv4 and IPv6?

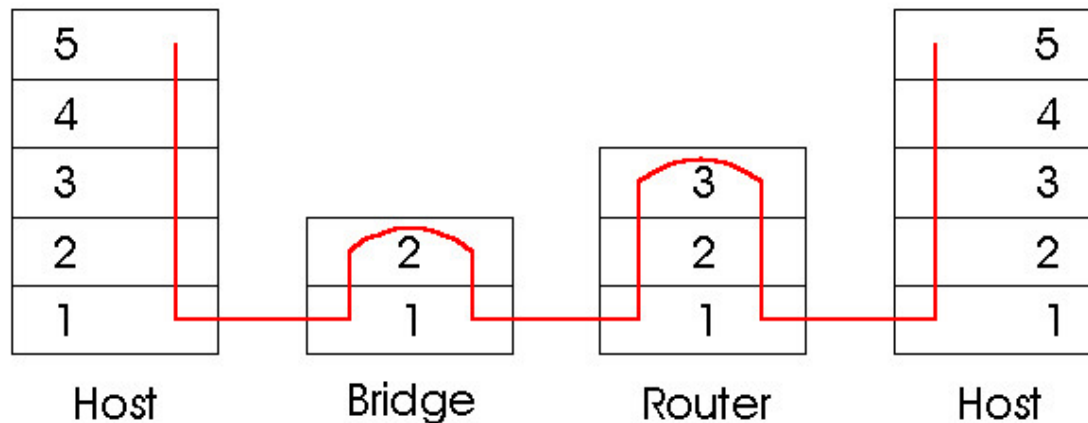
- Why include unverifiable source address?
 - Would like accountability *and* anonymity (now neither)
 - Return address can be communicated at higher layer
- Why packet header used at edge same as core?
 - Edge: host tells network what service it wants
 - Core: packet tells switch how to handle it
 - One is local to host, one is global to network
- Some kind of payment/responsibility field?
 - Who is responsible for paying for packet delivery?
 - Source, destination, other?
- Other ideas?

Gluing it together:

**How does my Network (address) interact
with my Data-Link (address) ?**

Switches vs. Routers Summary

- both store-and-forward devices
 - routers: network layer devices (examine network layer headers)
 - switches are link layer devices
- routers maintain routing tables, implement routing algorithms
- switches maintain switch tables, implement filtering, learning algorithms



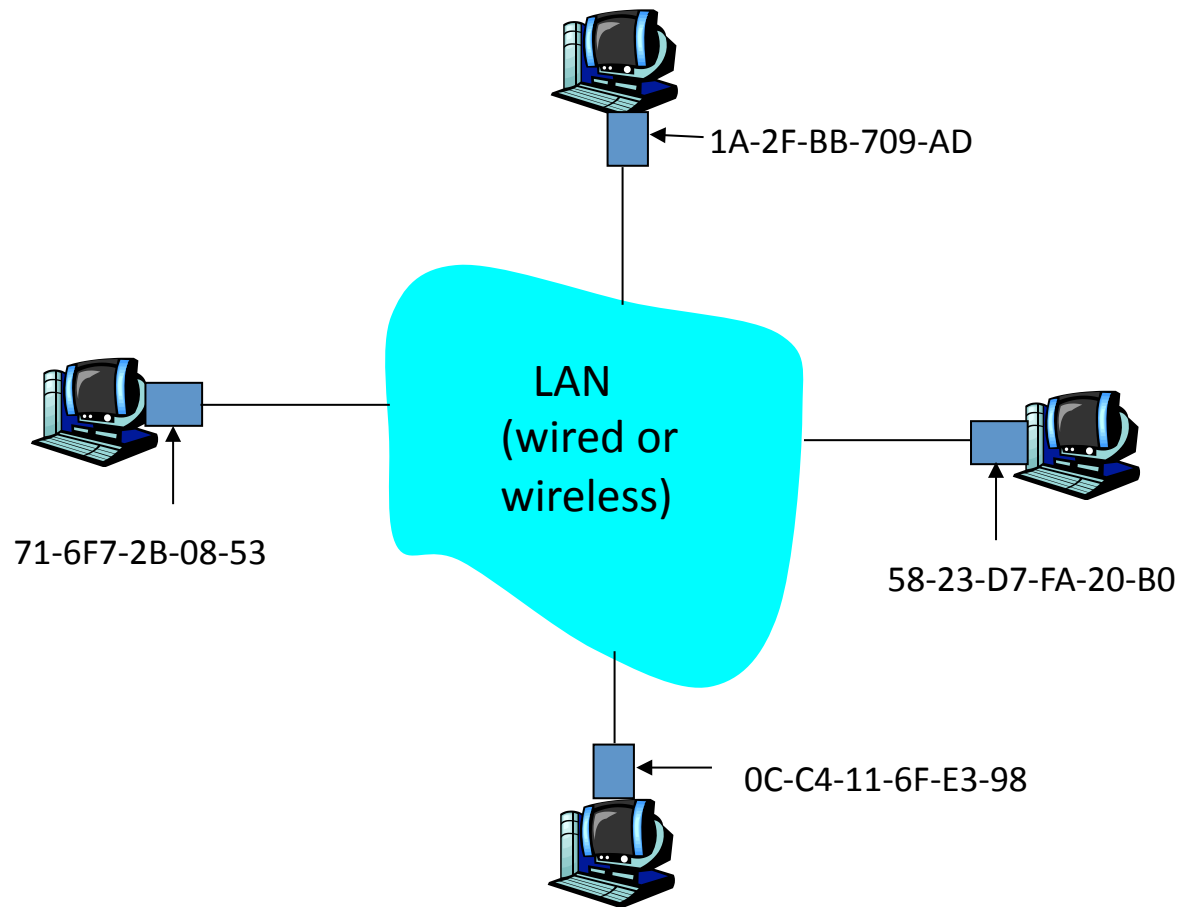
MAC Addresses (and ARP)

or How do I glue my network to my data-link?

- 32-bit IP address:
 - *network-layer* address
 - used to get datagram to destination IP subnet
- MAC (or LAN or physical or Ethernet) address:
 - function: *get frame from one interface to another physically-connected interface (same network)*
 - 48 bit MAC address (for most LANs)
 - burned in NIC ROM, also (commonly) software settable

LAN Addresses and ARP

Each adapter on LAN has unique LAN address



Ethernet
Broadcast address =
FF-FF-FF-FF-FF-FF

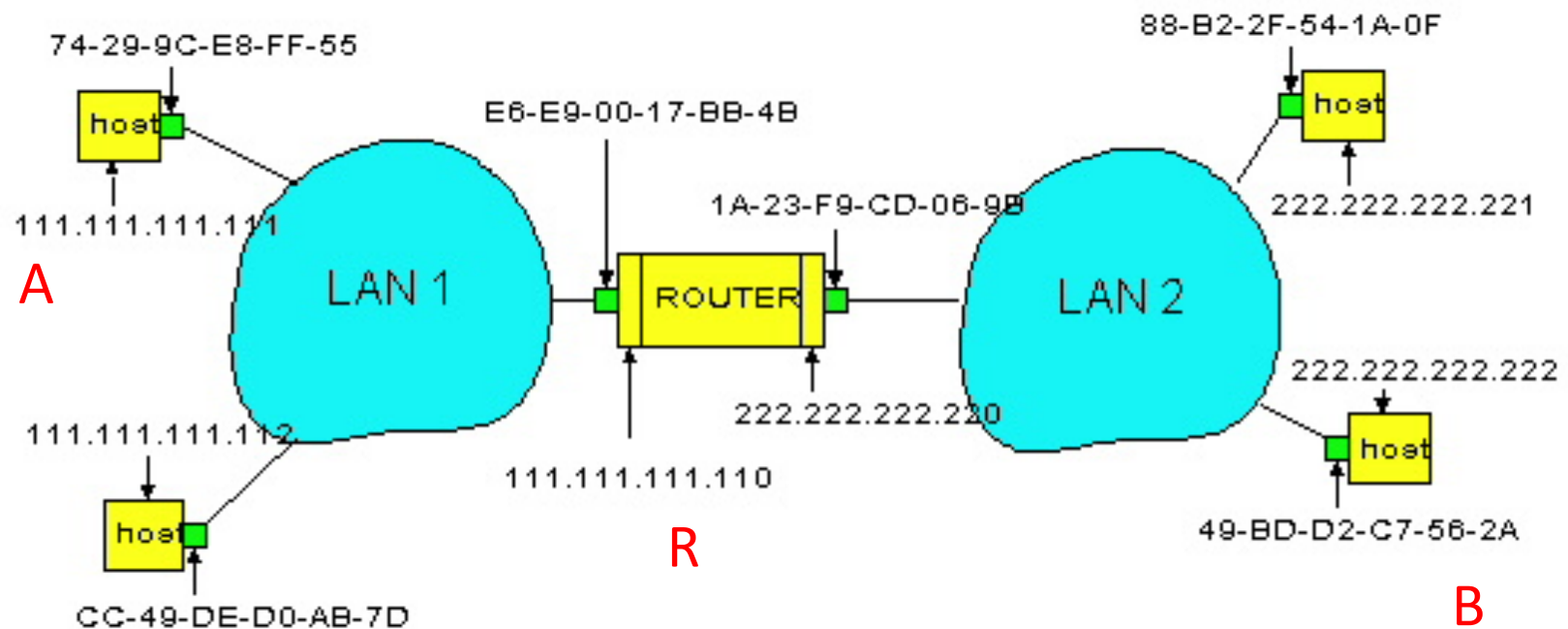
■ = adapter

Address Resolution Protocol

- Every node maintains an **ARP** table
 - <IP address, MAC address> pair
- Consult the table when sending a packet
 - Map destination IP address to destination MAC address
 - Encapsulate and transmit the data packet
- But: what if IP address **not** in the table?
 - Sender **broadcasts**: “**Who has IP address 1.2.3.156?**”
 - Receiver responds: “**MAC address 58-23-D7-FA-20-B0**”
 - Sender **caches** result in its ARP table

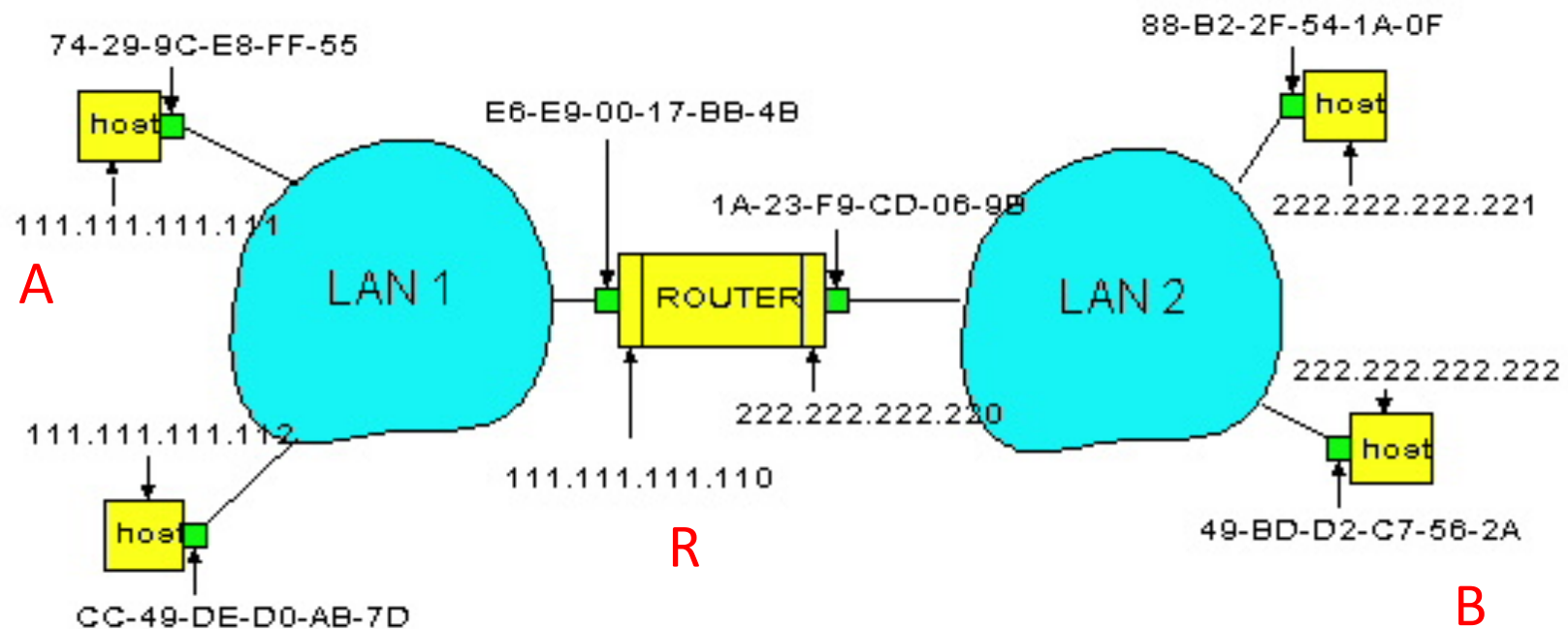
Example: A Sending a Packet to B

How does host **A** send an IP packet to host **B**?



Example: A Sending a Packet to B

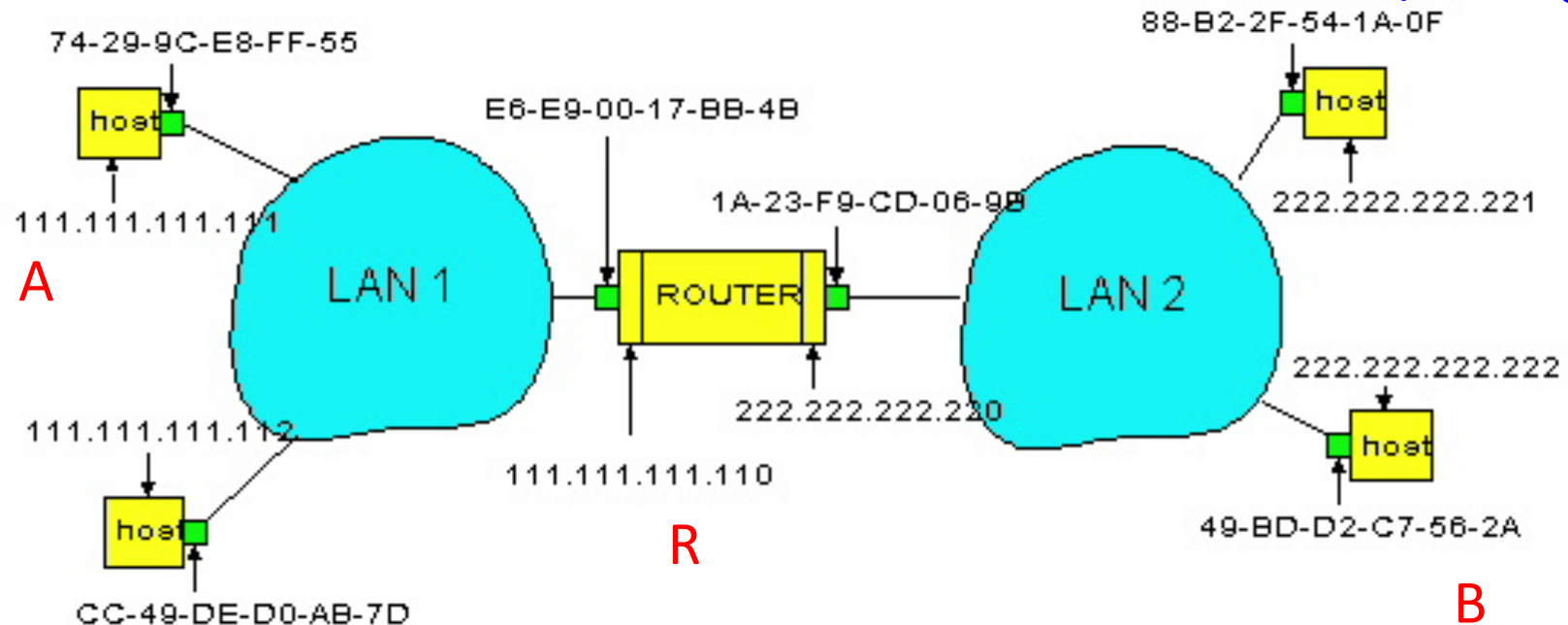
How does host **A** send an IP packet to host **B**?



1. **A** sends packet to **R**.
2. **R** sends packet to **B**.

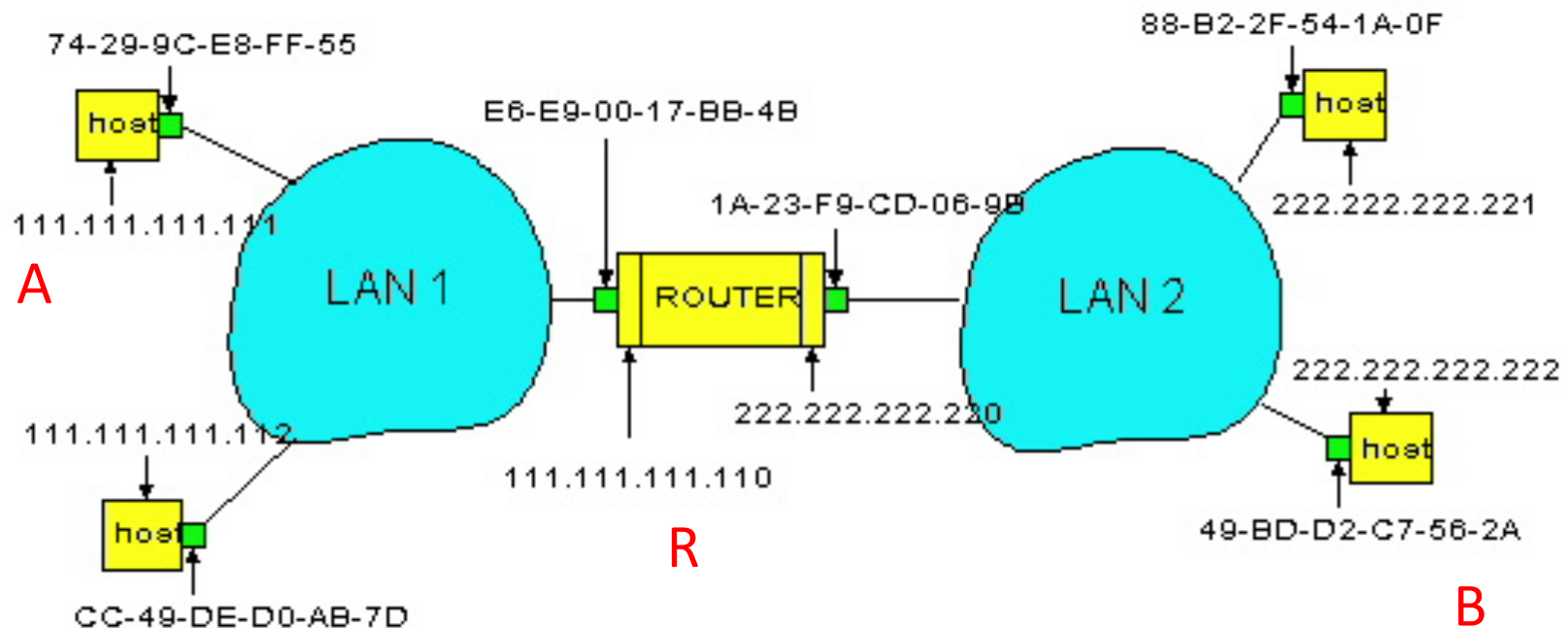
Host A Decides to Send Through R

- Host **A** constructs an IP packet to send to **B**
 - Source 111.111.111.111, destination 222.222.222.222
- Host **A** has a gateway router **R**
 - Used to reach destinations outside of 111.111.111.0/24
 - Address 111.111.111.110 for R learned via [DHCP/config](#)



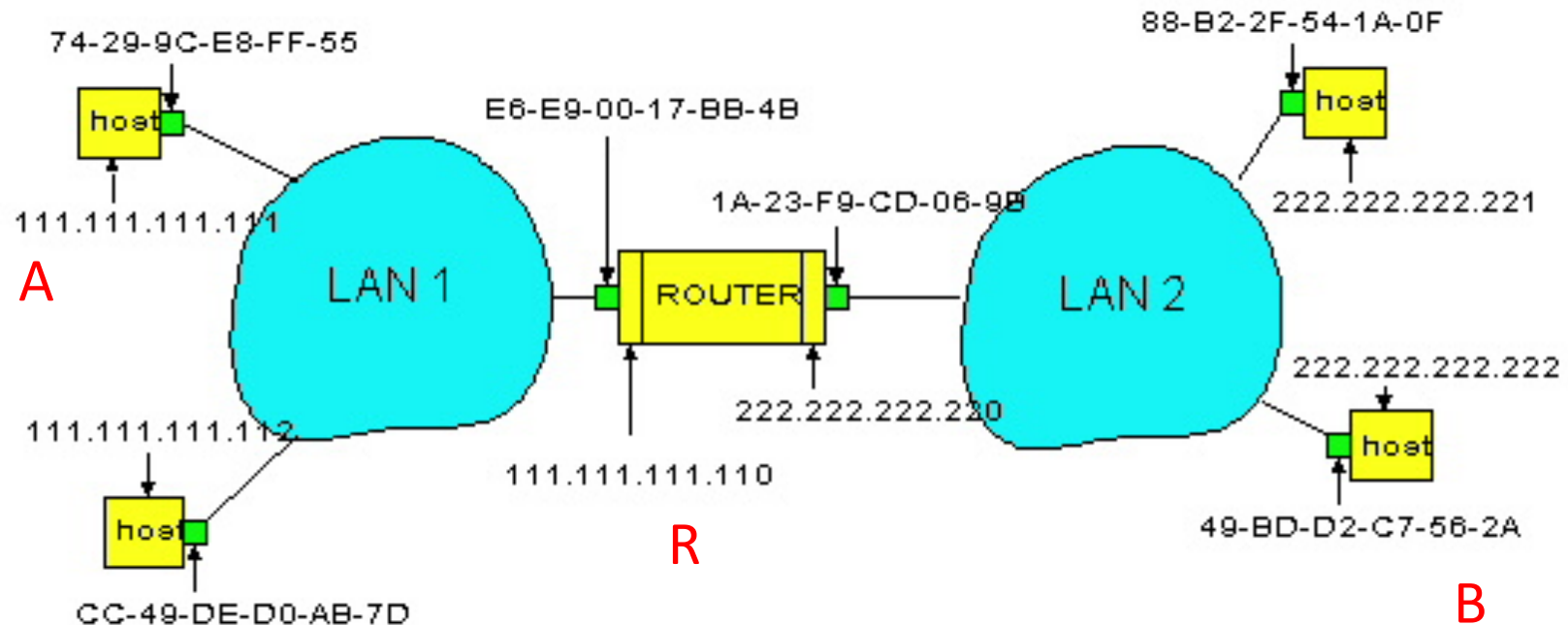
Host A Sends Packet Through R

- Host **A** learns the MAC address of **R**'s interface
 - **ARP** request: broadcast request for 111.111.111.110
 - **ARP** response: **R** responds with E6-E9-00-17-BB-4B
- Host **A** encapsulates the packet and sends to **R**



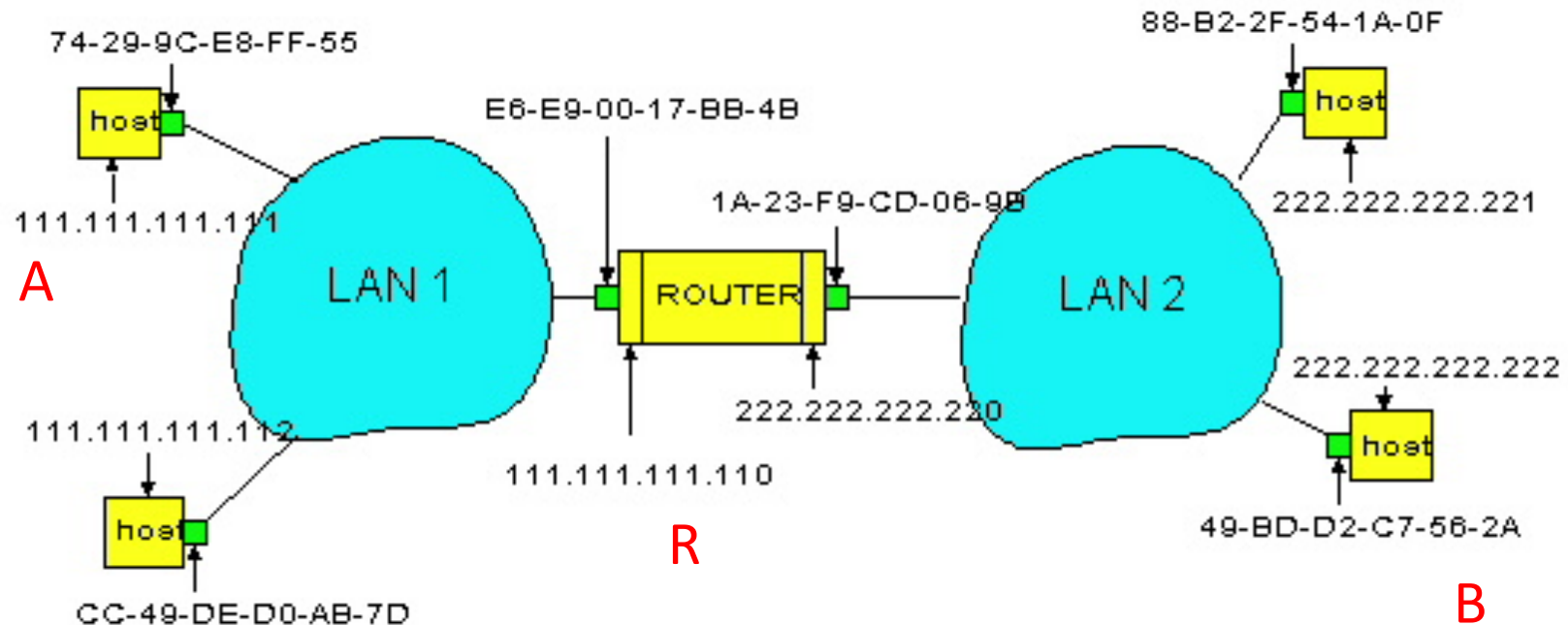
R Decides how to Forward Packet

- Router **R**'s adaptor receives the packet
 - **R** extracts the IP packet from the Ethernet frame
 - **R** sees the IP packet is destined to 222.222.222.222
- Router **R** consults its forwarding table
 - Packet matches 222.222.222.0/24 via other adaptor



R Sends Packet to B

- Router **R**'s learns the MAC address of host **B**
 - **ARP** request: broadcast request for 222.222.222.222
 - **ARP** response: **B** responds with 49-BD-D2-C7-52A
- Router **R** encapsulates the packet and sends to **B**



Security Analysis of ARP



- Impersonation
 - Any node that hears request can answer ...
 - ... and can say whatever they want
- Actual legit receiver never sees a problem
 - Because even though later packets carry its IP address, its NIC doesn't capture them since not its MAC address

Key Ideas in Both ARP and DHCP

- **Broadcasting**: Can use broadcast to make contact
 - Scalable because of limited size
- **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**: 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

Why Not Use DNS-Like Tables?

- When host arrives:
 - Assign it an IP address that will last as long it is present
 - Add an entry into a table in DNS-server that maps MAC to IP addresses
- Answer:
 - Names: explicit creation, and are plentiful
 - Hosts: come and go without informing network
 - Must do mapping on demand
 - Addresses: not plentiful, need to reuse and remap
 - Soft-state enables dynamic reuse

Summary Network Layer

- understand principles behind network layer services:
 - network layer service models
 - forwarding versus routing (versus switching)
 - how a router works
 - routing (path selection)
 - IPv6
- Algorithms
 - Two routing approaches (LS vs DV)
 - One of these in detail (LS)
 - ARP