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 is a *something*

Routing: How do I get to the *something*

Forwarding: What path do I take next
to get to the *something*

Addressing (at a conceptual level)

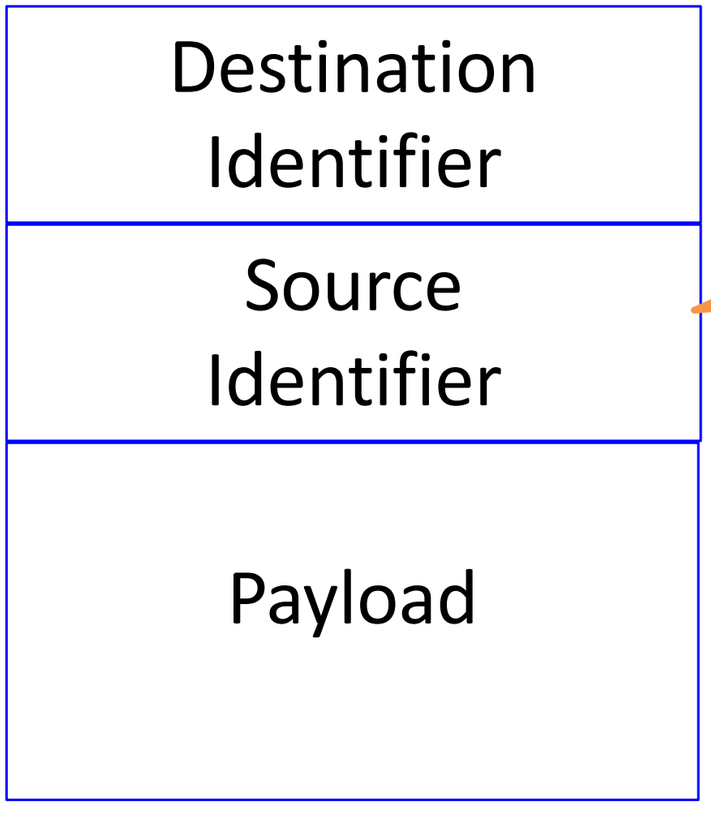
- Assume all hosts have unique IDs
- No particular structure to those IDs
- Later in topic I will talk about real IP addressing

- 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?

3

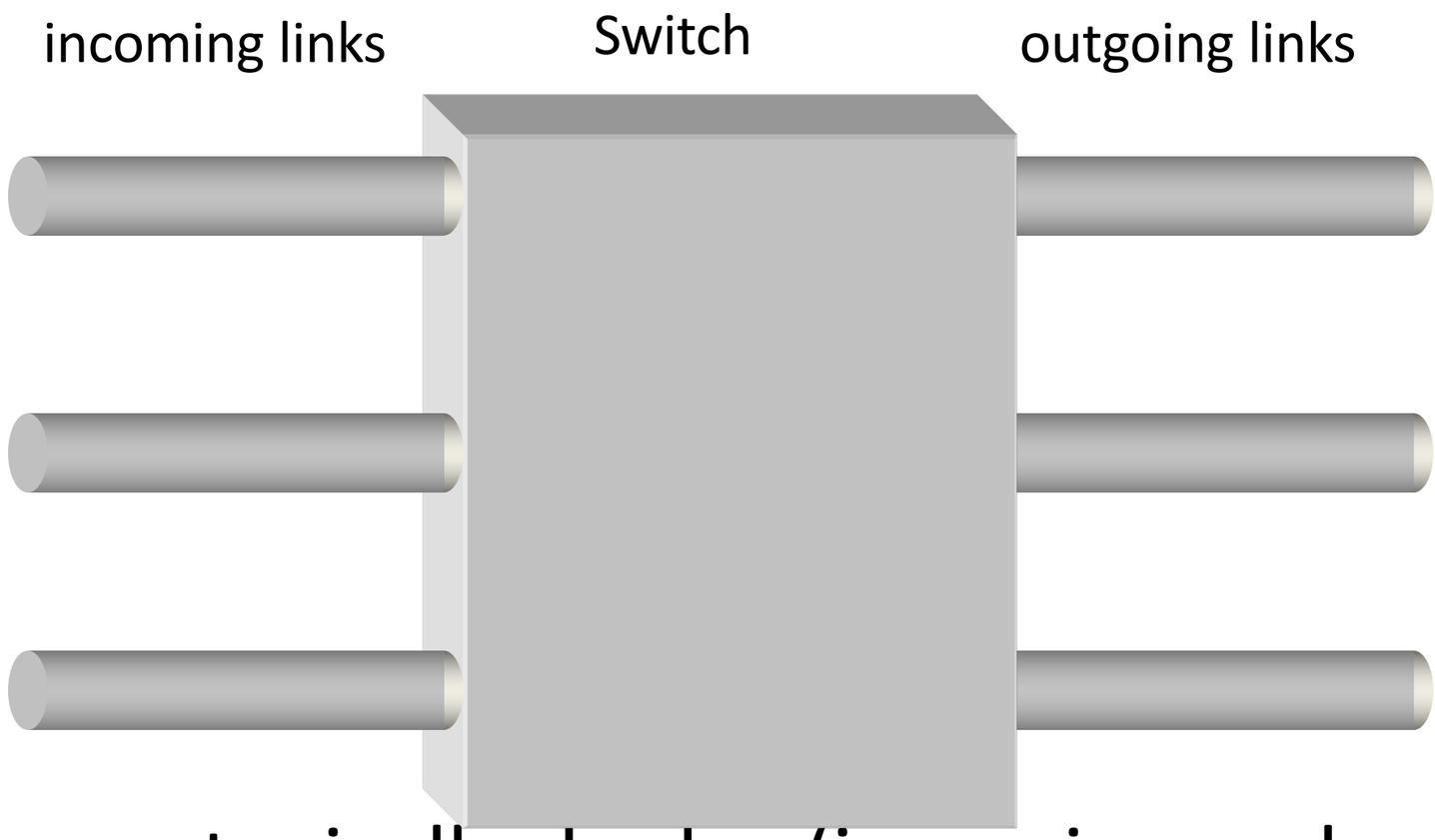
Packets (at a conceptual level)

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



Switches/Routers

- Multiple ports (attached to other switches or hosts)

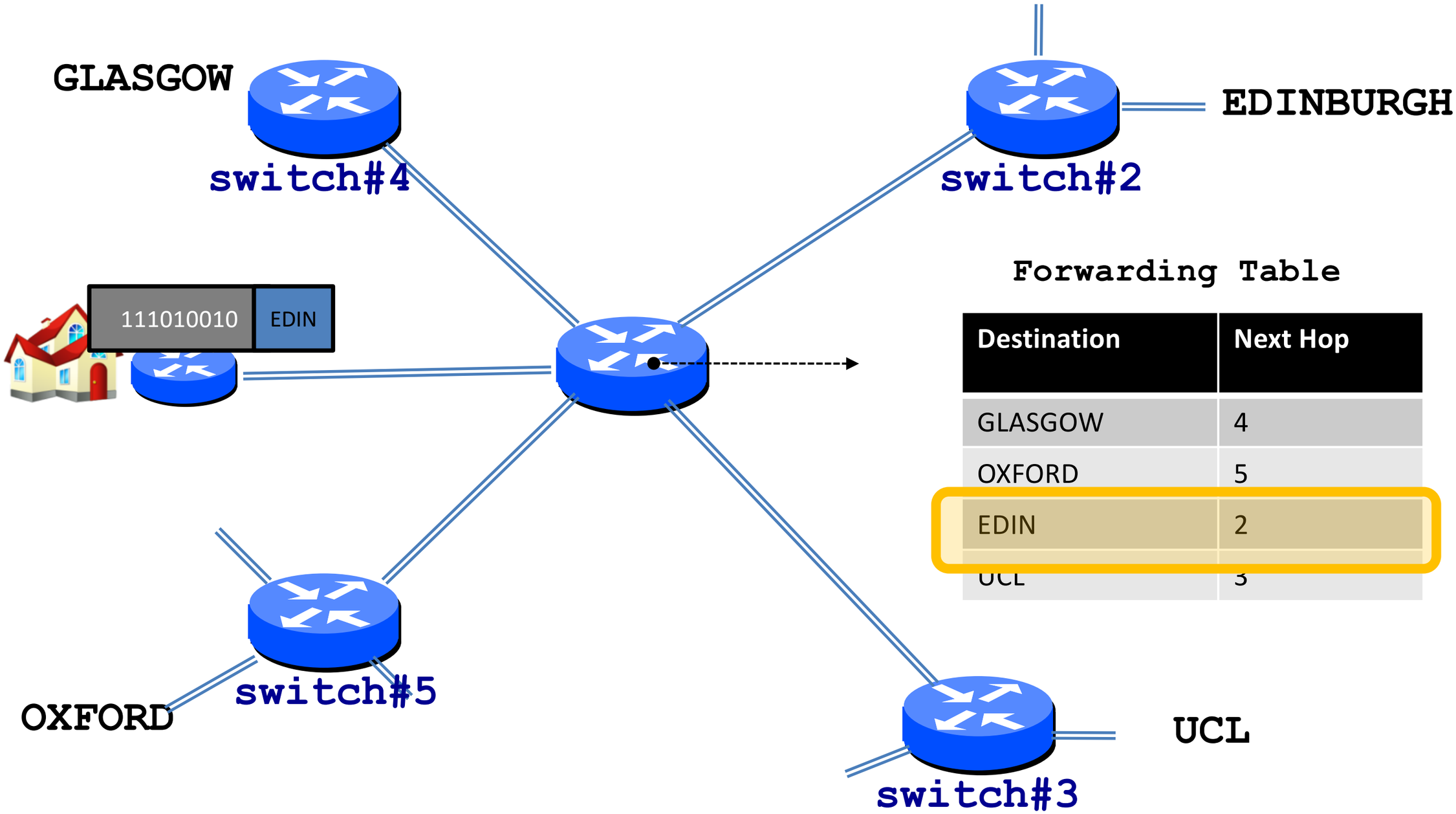


- Ports are typically duplex (incoming and outgoing)

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 Table

Destination	Next Hop
GLASGOW	4
OXFORD	5
EDIN	2
UCL	3

Forwarding Decisions

- When packet arrives..
  - Must decide which outgoing port to use
  - In single transmission time
  - Forwarding decisions must be ***simple***

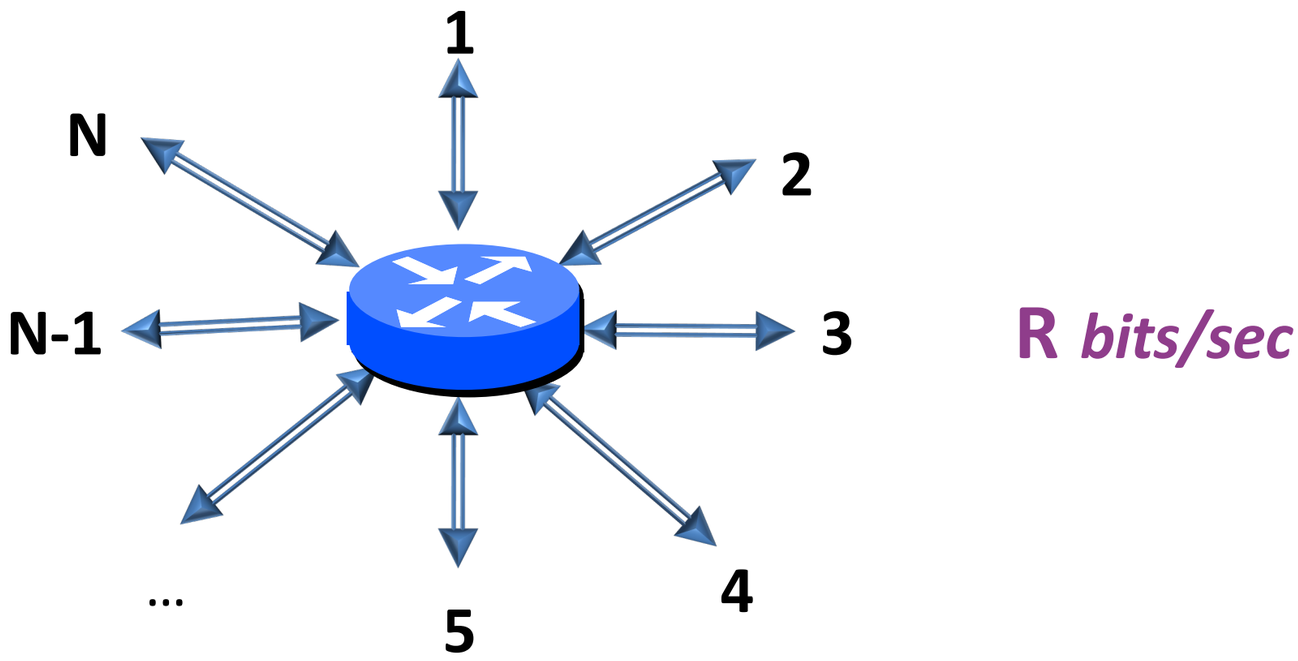
- Routing state dictates where to forward packets
  - Assume decisions are **deterministic**

- *Global routing state* is the 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

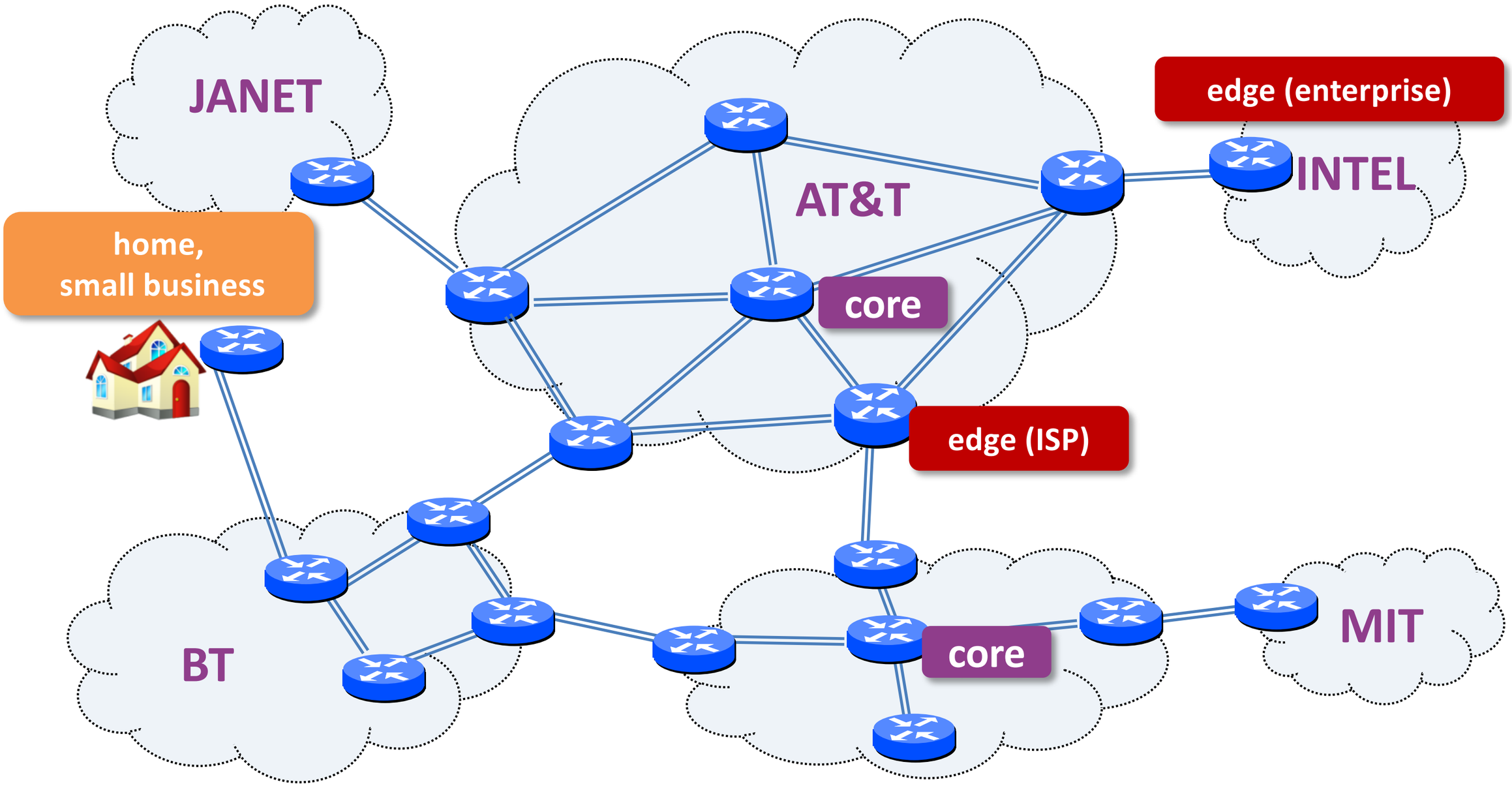
- Forwarding: “data plane”
  - Directing a data packet to an outgoing link
  - Individual router using routing state
- Routing: “control plane”
  - Computing paths the packets will follow
  - Routers talking amongst themselves
  - Jointly creating the routing state
- Two very different timescales….

Router definitions

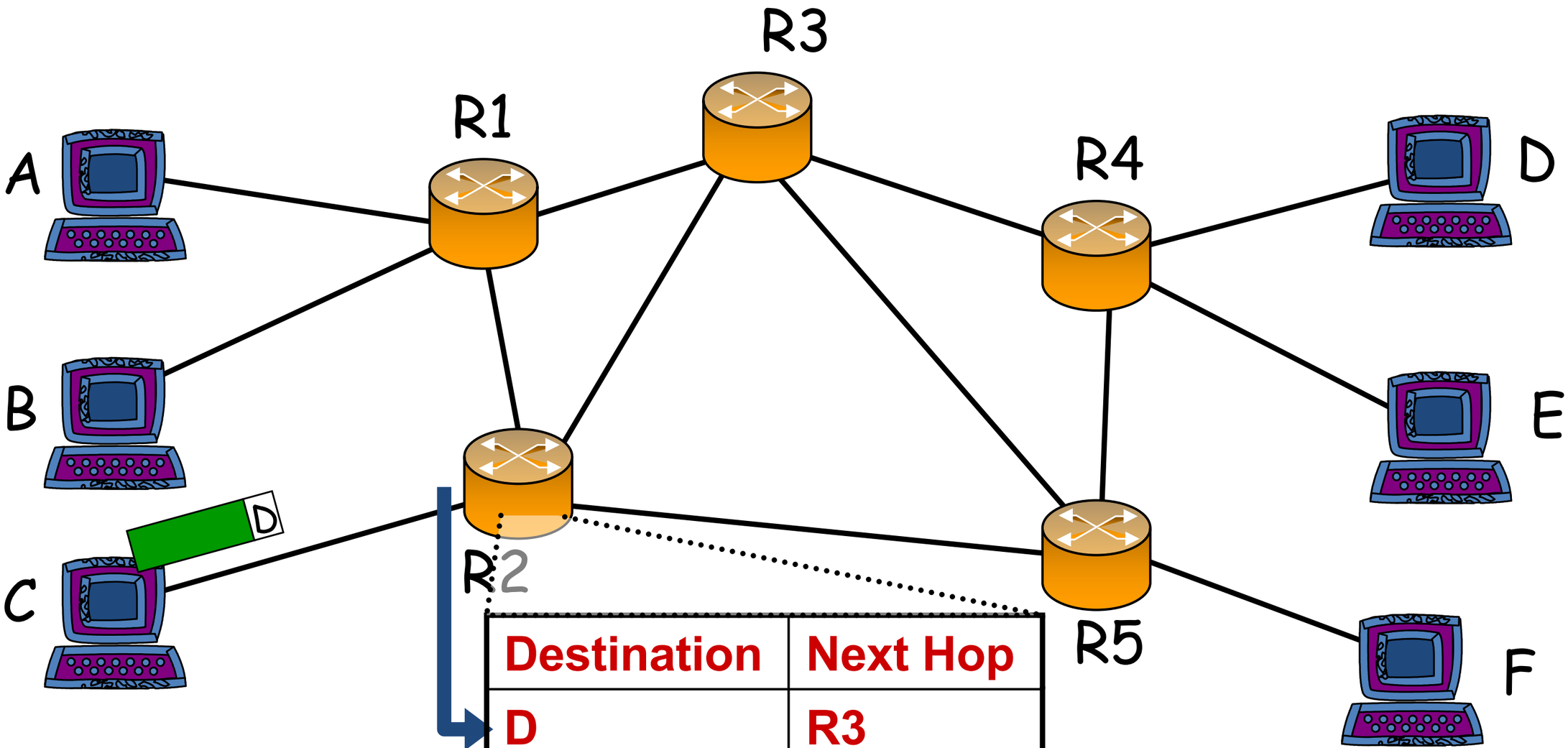


- N = number of external router “ports”
- R = speed (“line rate”) of a port
- Router capacity = N x R

Networks and routers

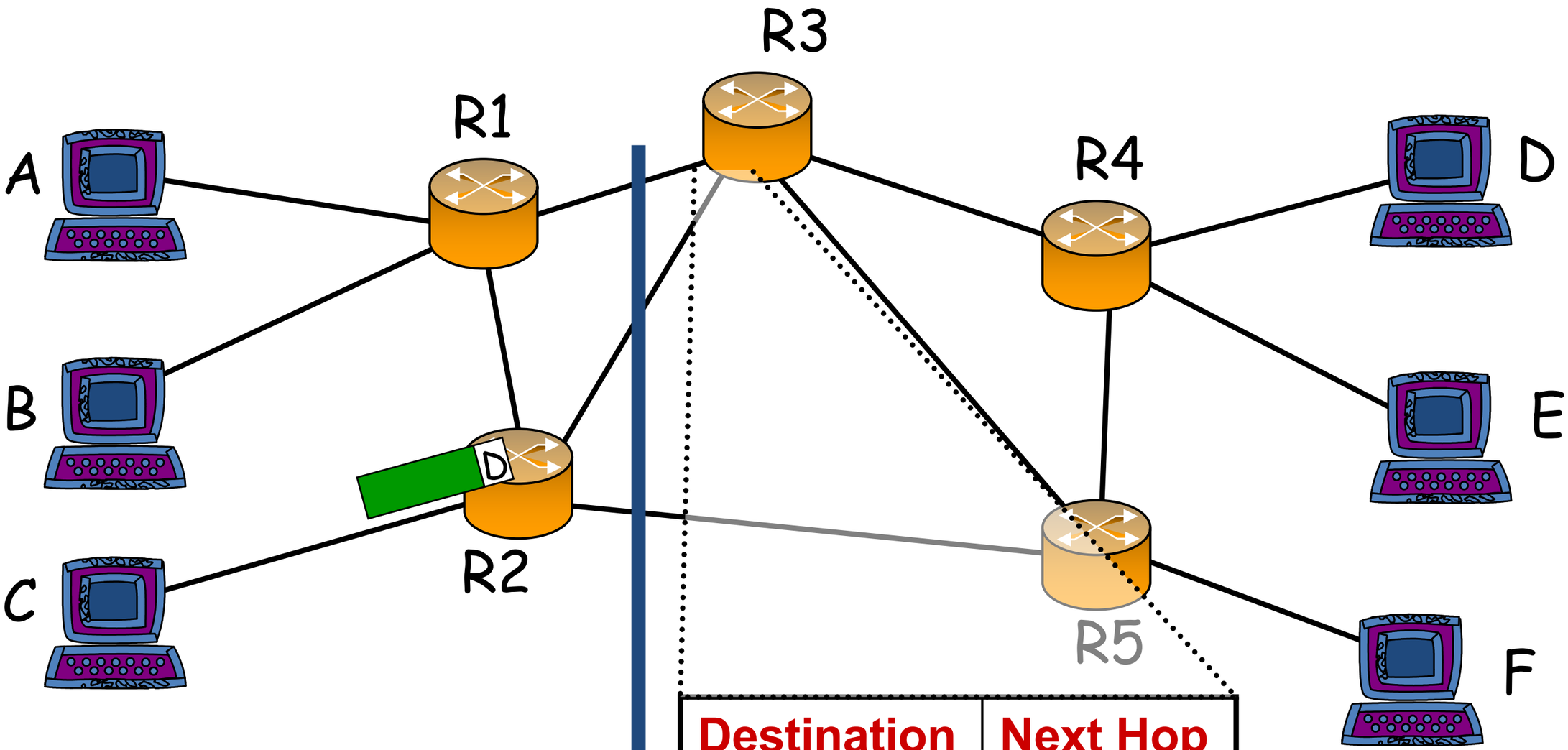


Basic Operation of Router



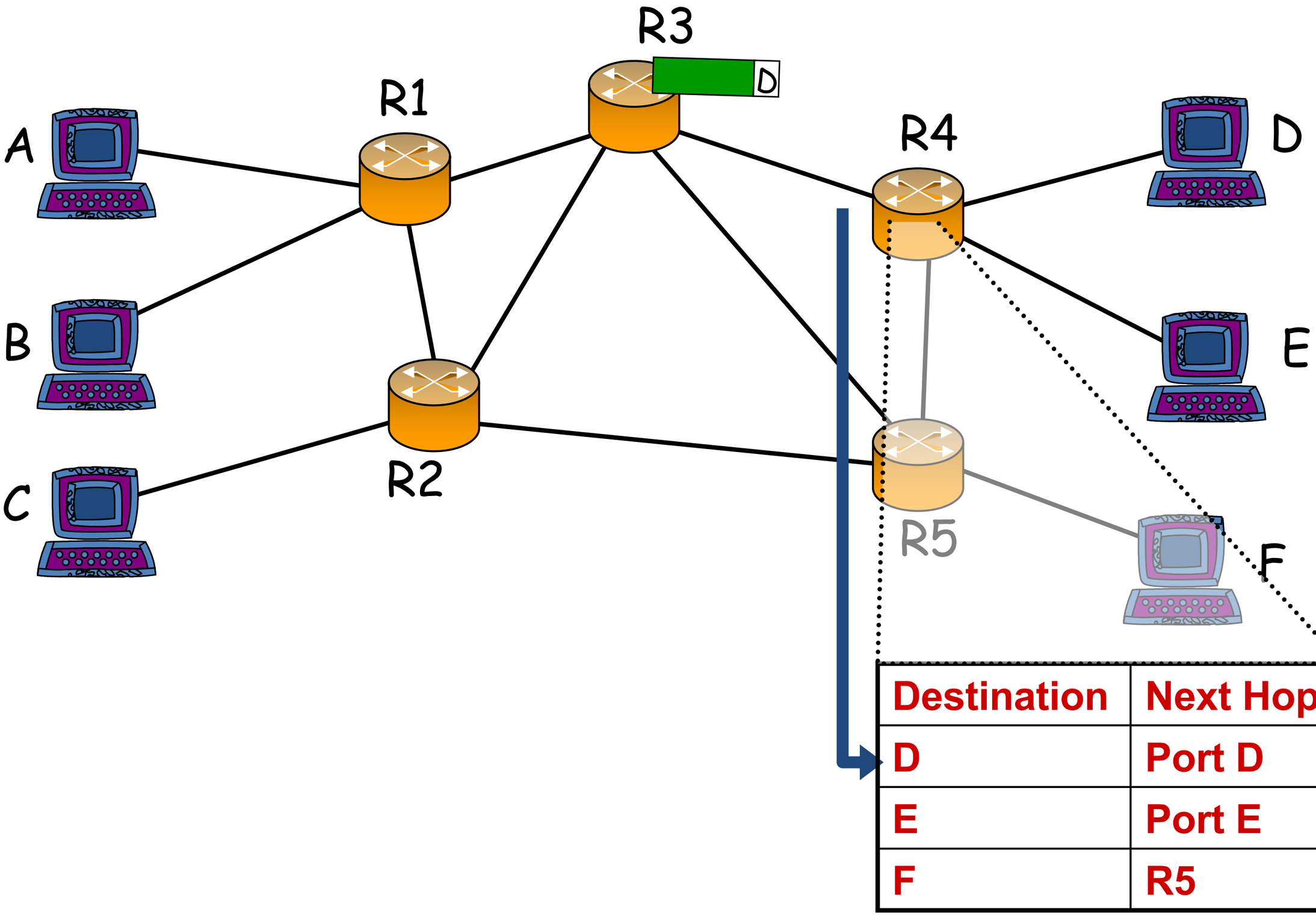
Destination	Next Hop
D	R3
E	R3
F	R5

Basic Operation of Router



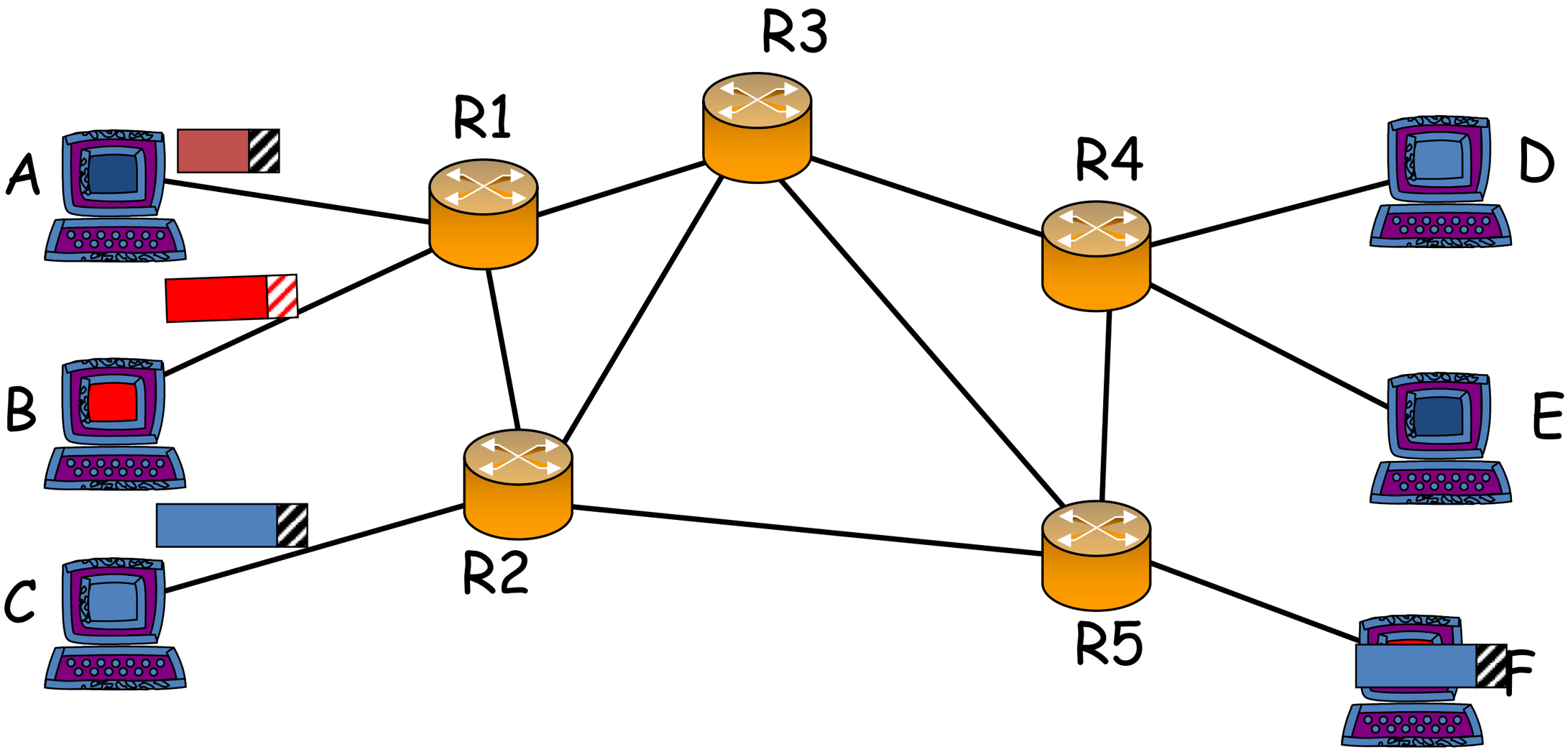
Destination	Next Hop
D	R4
E	R4
F	R5

Basic Operation of Router



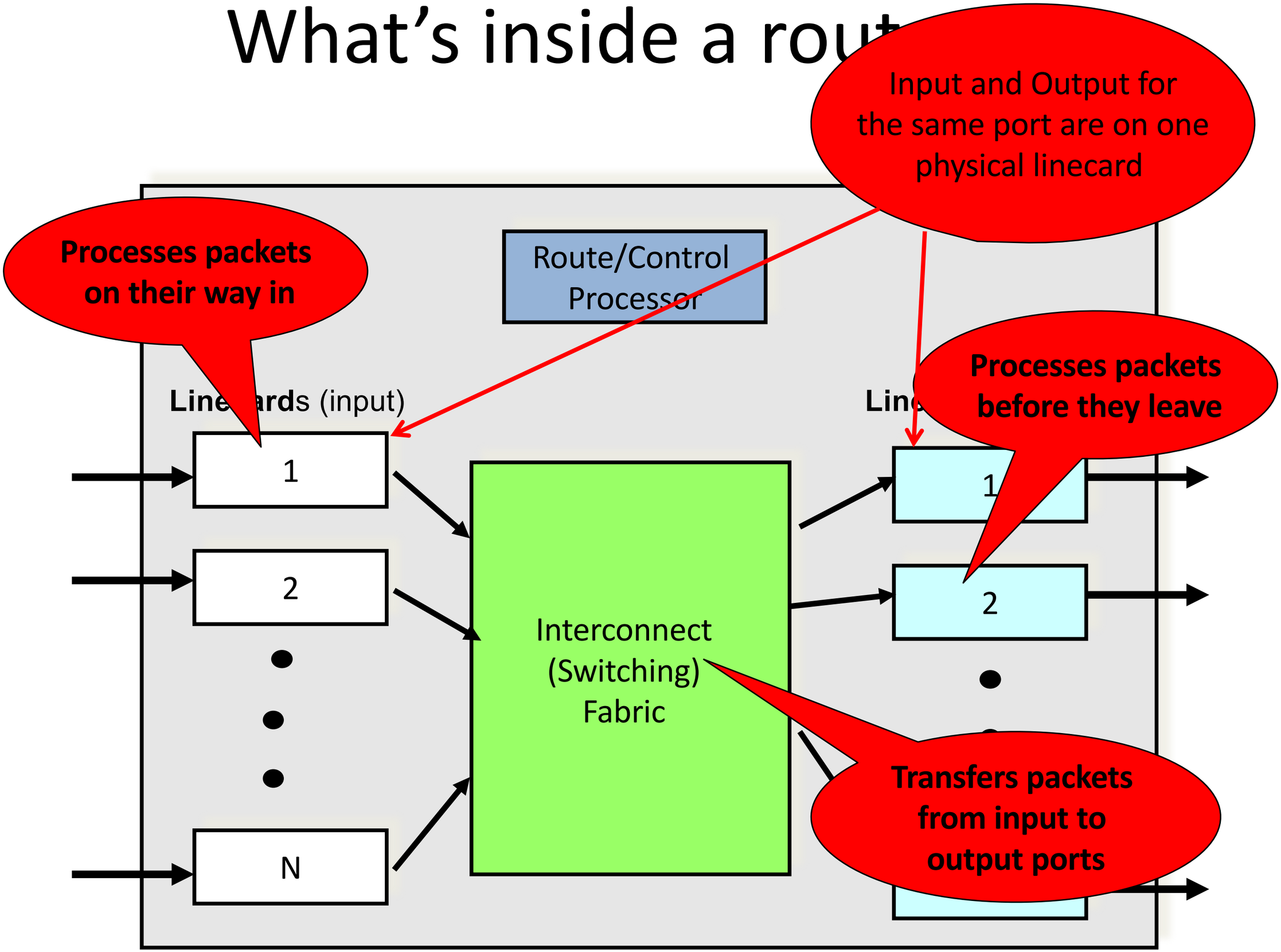
Destination	Next Hop
D	Port D
E	Port E
F	R5

What does a router do?

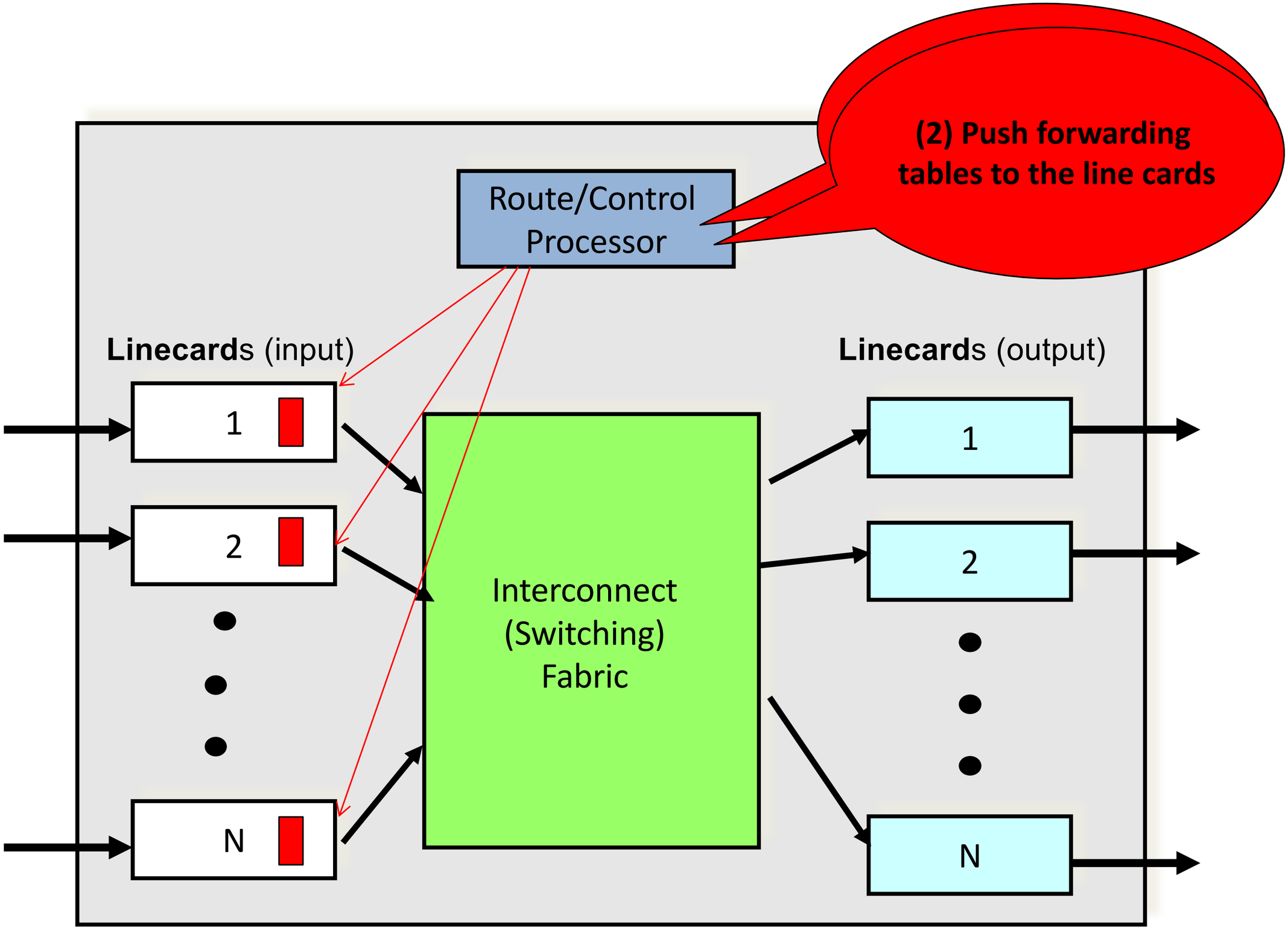


1. Every router performs a per-packet lookup for every packet
2. Each router performs a lookup in it’s local lookup table
3. Each router performs lookups (ENTIRELY) independently of every other router

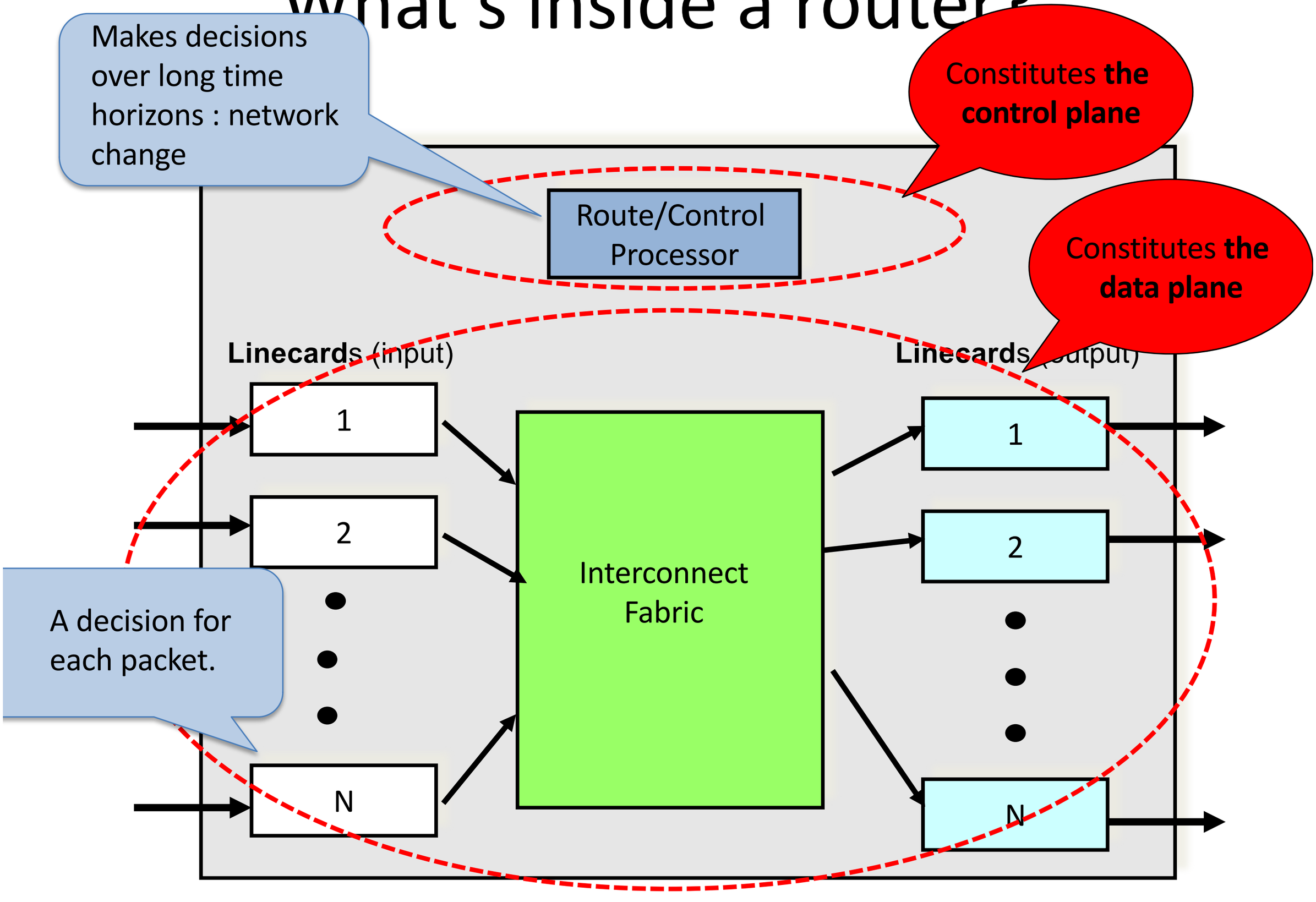
What's inside a rout



What's inside a router?



What's inside a router?



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

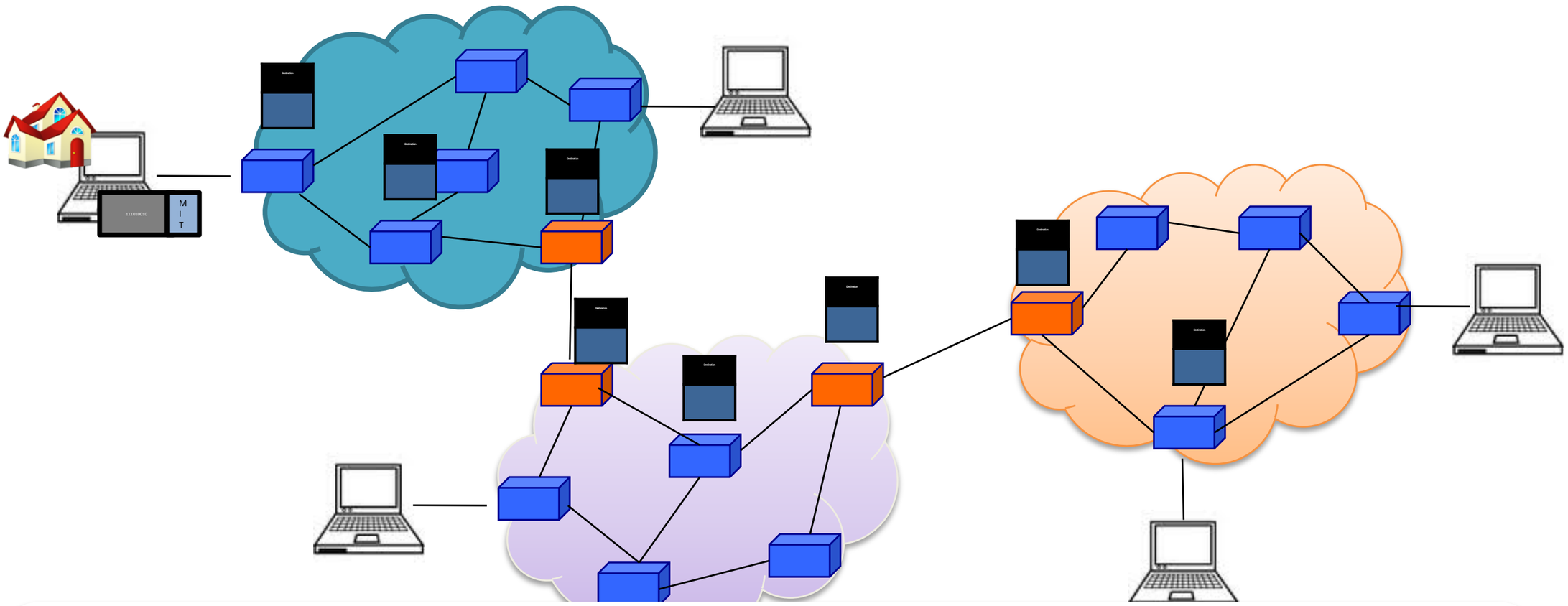
“End hosts”
“Clients”, “Users”
“End points”

“Border Routers”

“Route” or “Path”

“Interior Routers”

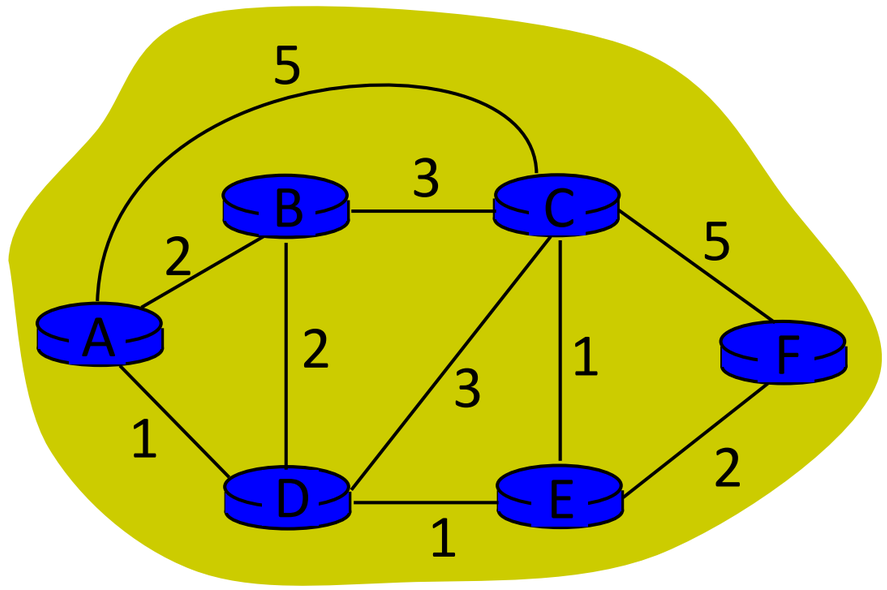
Context and Terminology



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 (to date)

- a reminder -

- Recall each host has a unique ID (address)

- No particular structure to those IDs (e.g. *Ethernet*)

- IP addressing – in contrast – has implicit structure

Outline

- Popular Routing Algorithms:
  - Link State Routing
  - Distance Vector Algorithm
- Routing: goals and metrics

Link-State Routing

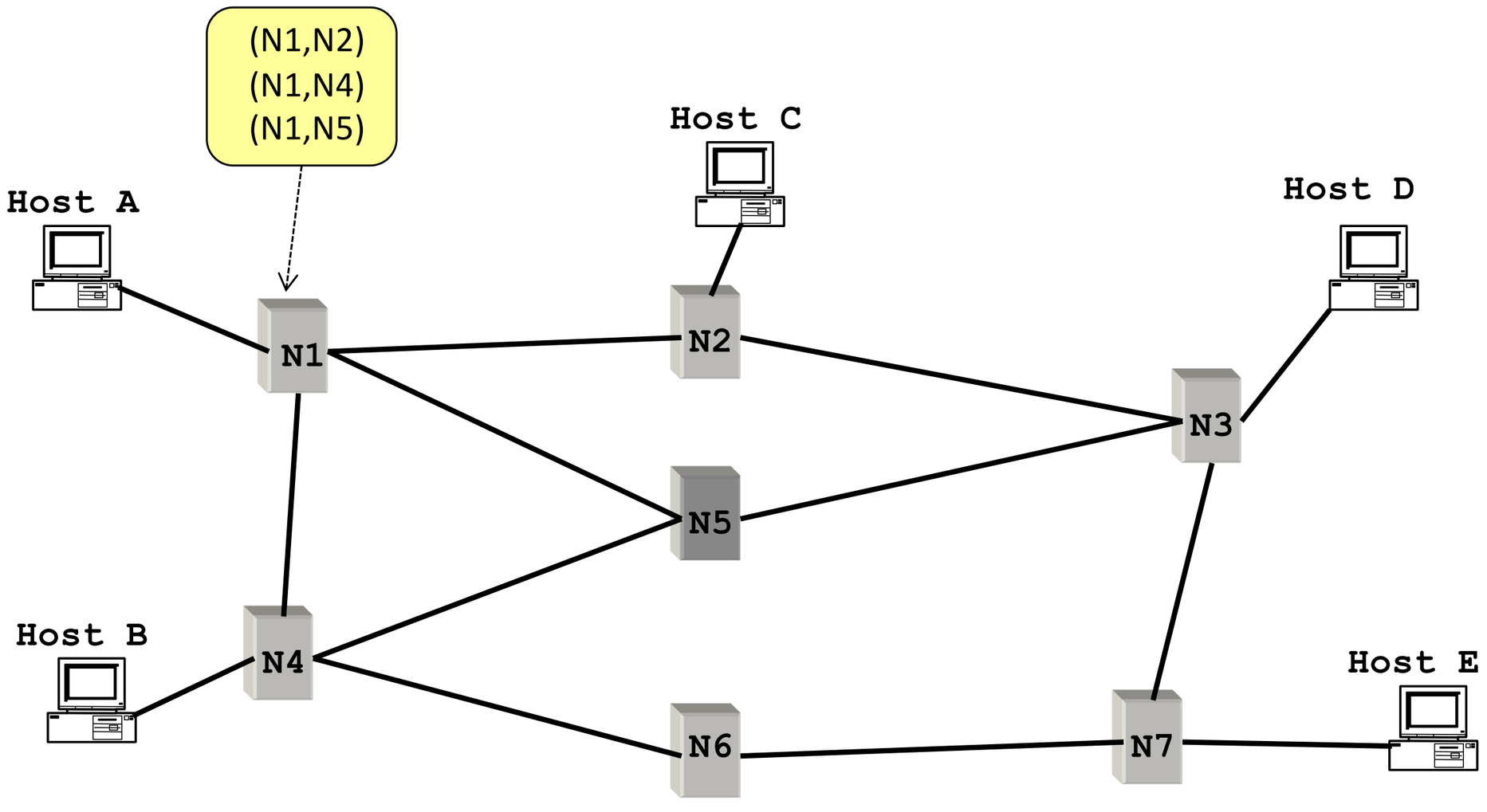
Examples:

Open Shortest Path First (**OSPF**) or
Intermediate System to Intermediate System
(written as **IS-IS**/**ISIS** and pronounced eye-esss-eye-esss)

The two common Intradomain routing or
interior gateway protocols (IGP)

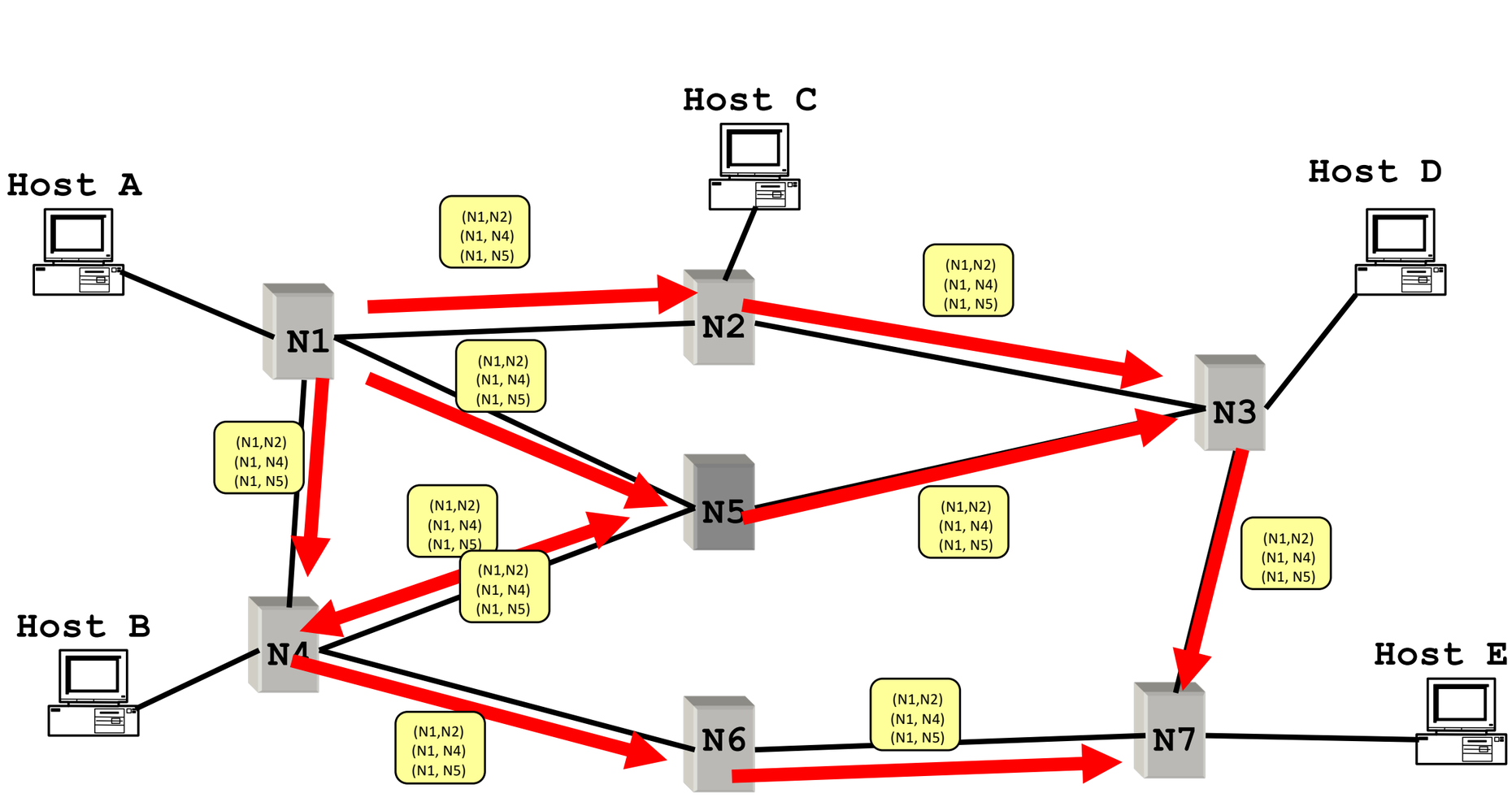
Link State Routing

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



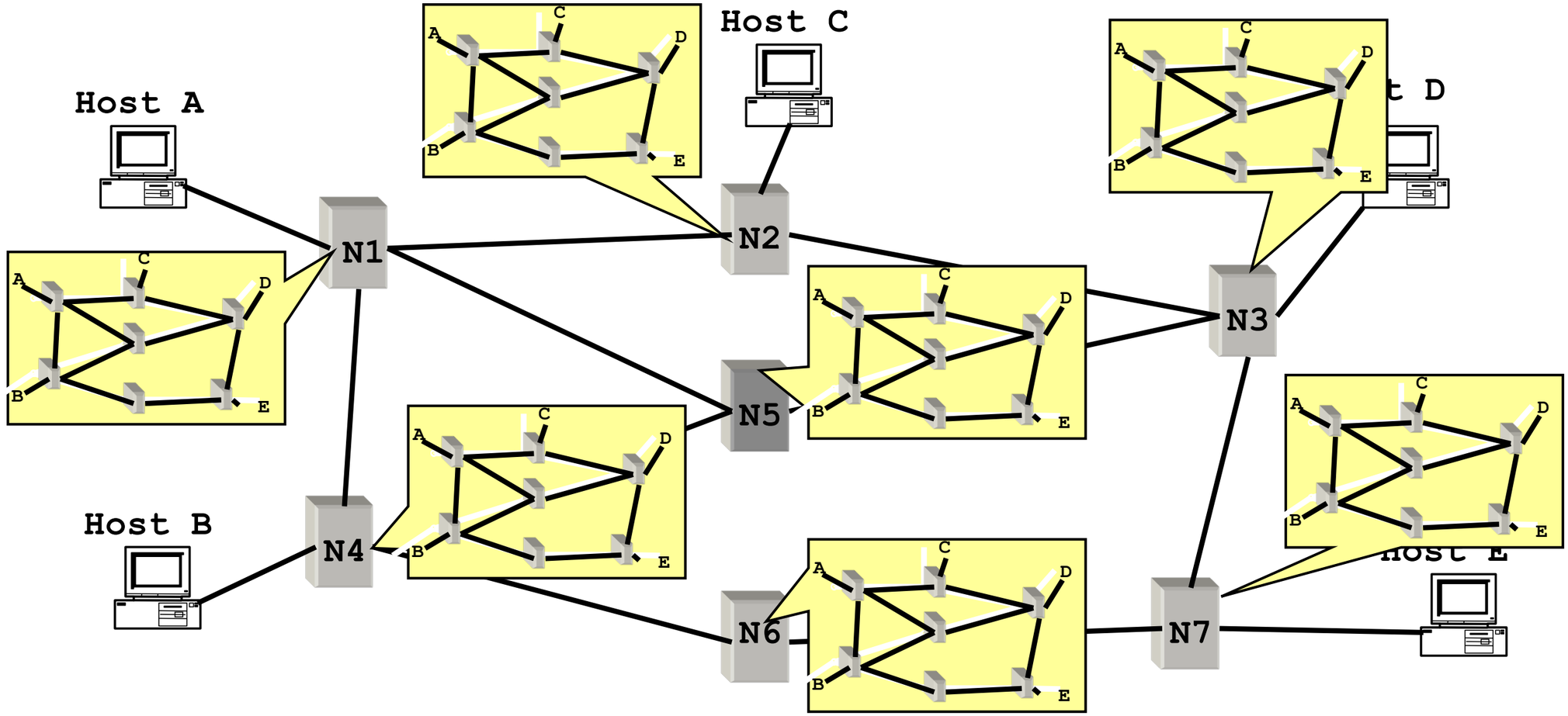
Link State Routing

- 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

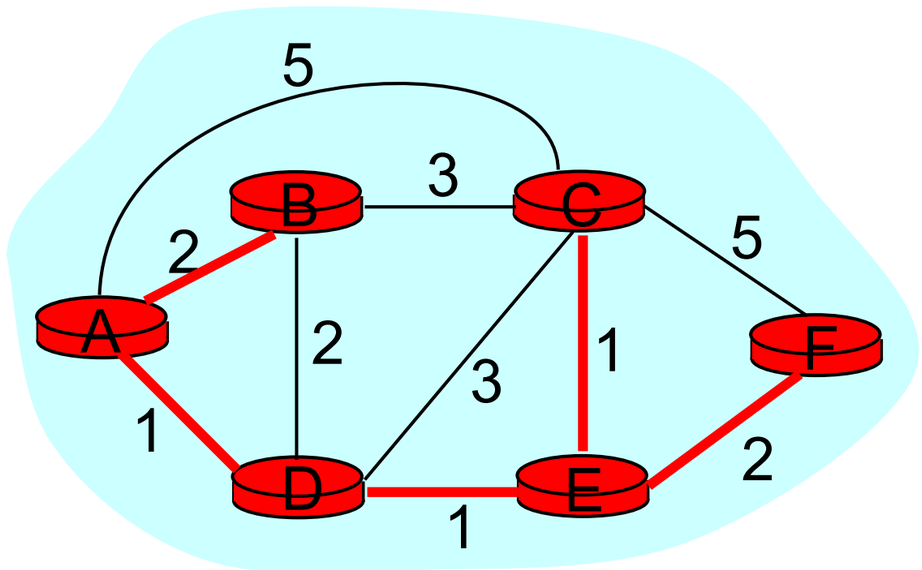
- OUTPUT:
  - Least cost paths from one node to all other nodes

- Iterative: after *k* iterations, a node knows the least cost path to its *k* closest neighbors

- This is covered in Algorithms

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 x E), where N is #nodes; E is #edges in graph

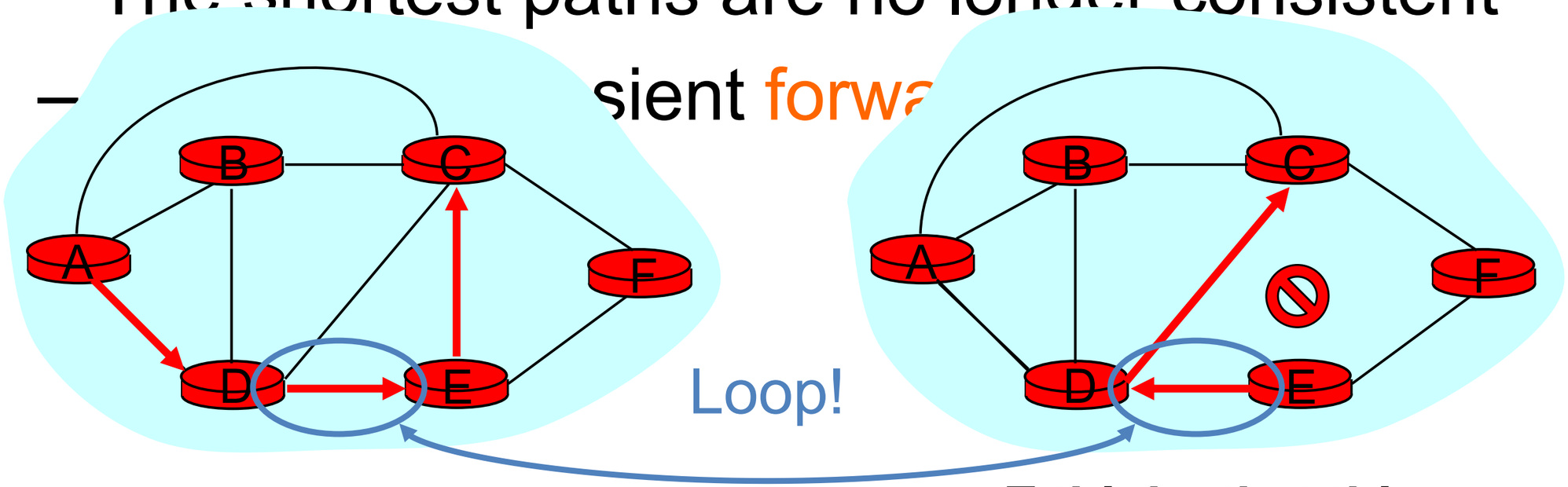
- Processing complexity for Dijkstra's algorithm?
  - $O(N^2)$, because we check all nodes w not in S at each iteration and we have O(N) iterations
  - more efficient implementations: O(N log(N))

- 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
  - Transient forwarding loops



A and D think that this is the path to C

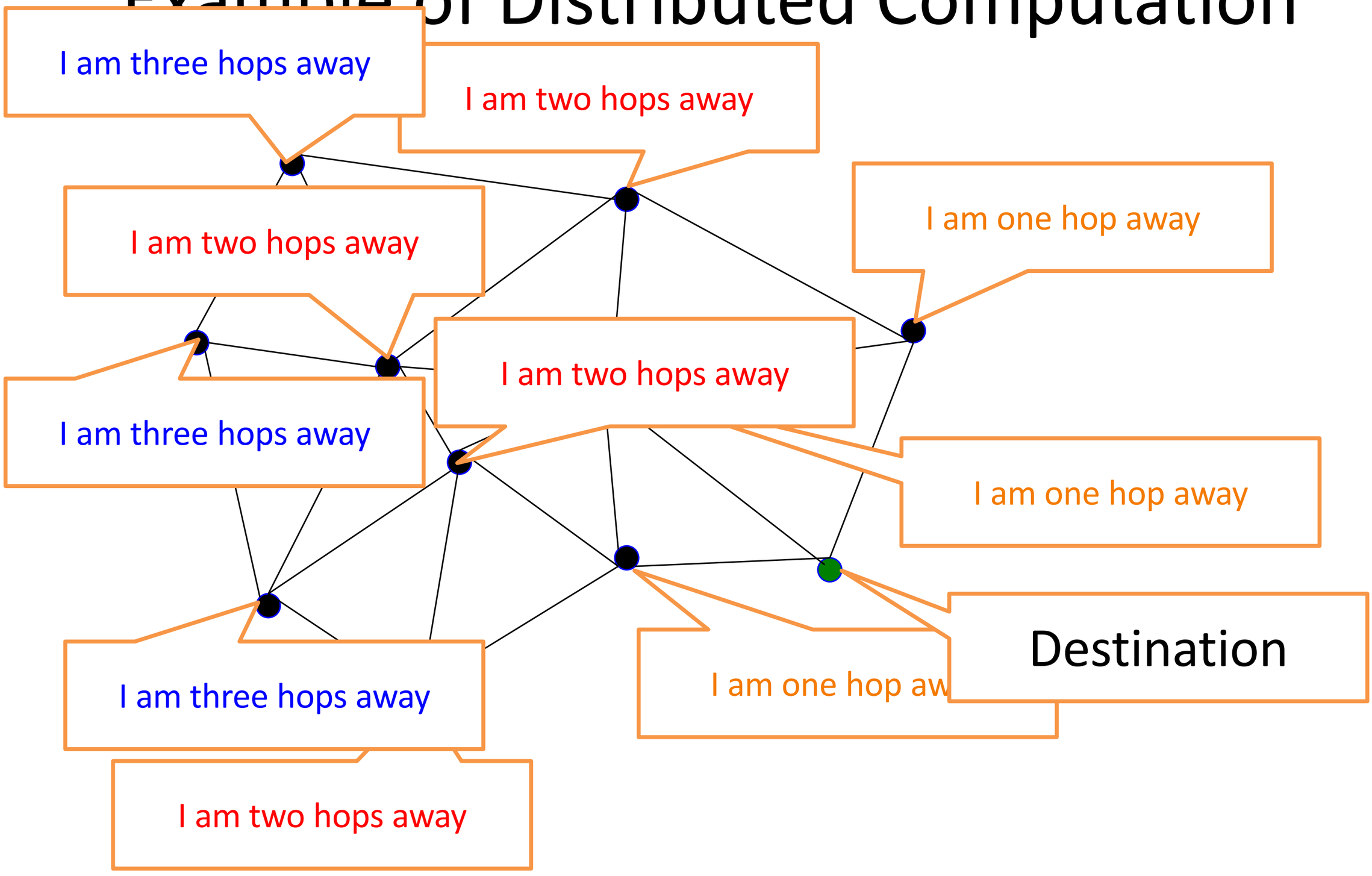
E thinks that this is the path to C

Distance-Vector Routing

Example:

Routing Information Protocol (RIP)

Example of Distributed Computation



Distance Vector Routing

Each router sends its knowledge about the "whole" network to its neighbors. Information sharing at regular intervals.

- 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 truths

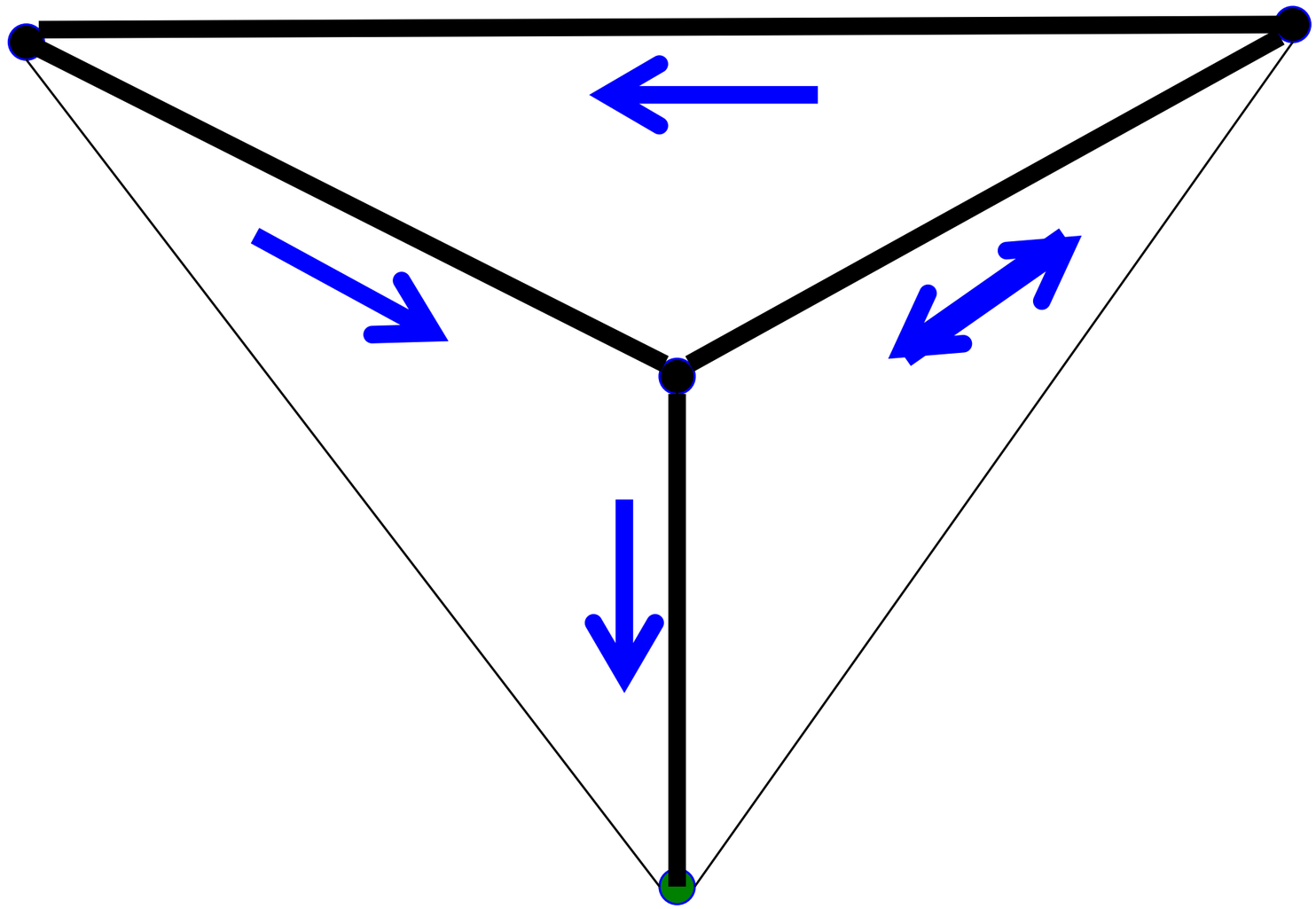
- What if we use a non-additive metric?
  - E.g., maximal capacity
- What if routers don't use the same metric?
  - I want low delay, you want low loss rate?
- What happens if nodes lie?

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



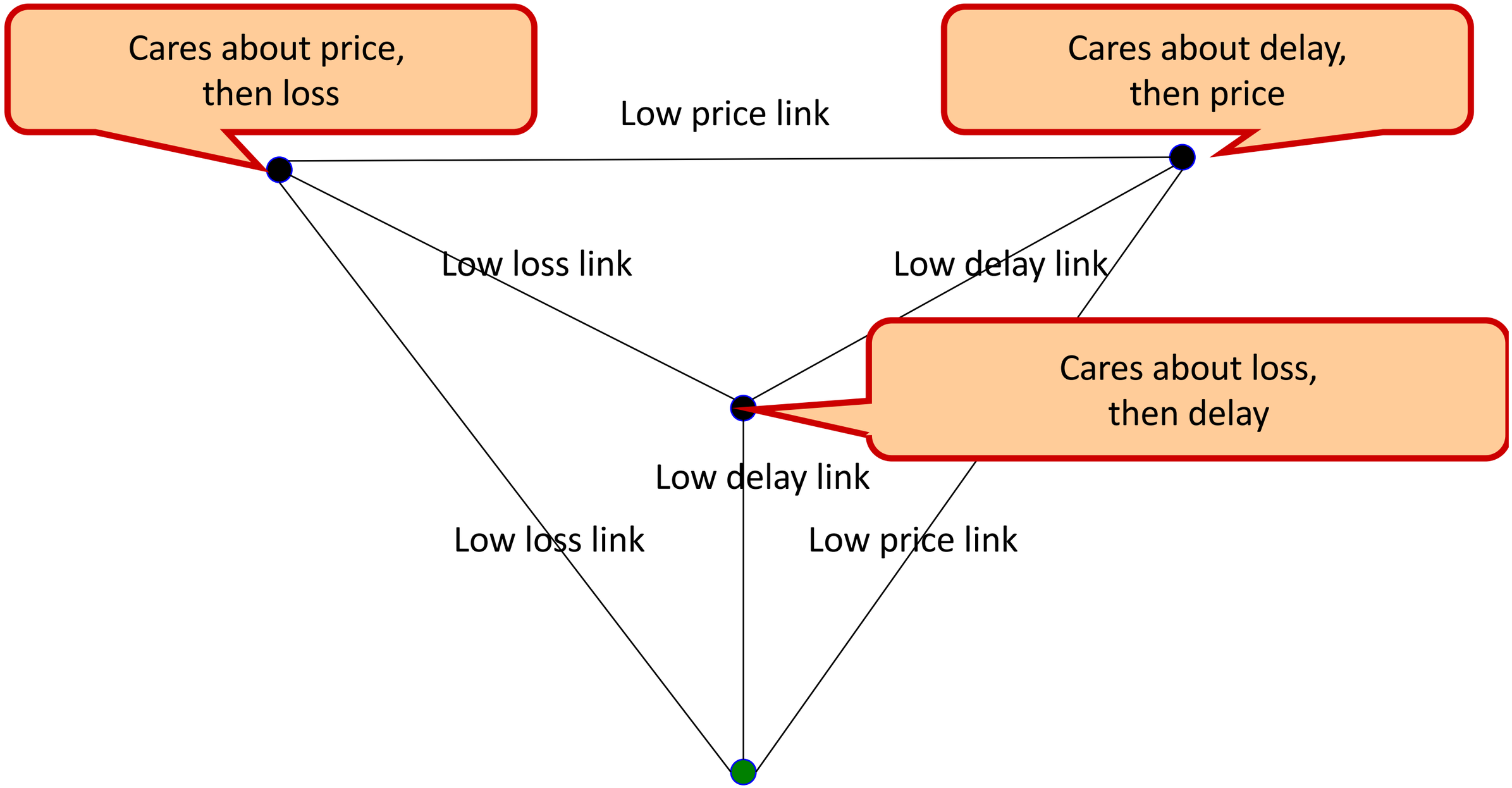
Additive measures avoid this problem!

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(NxE) messages;
  - N is #nodes; E is #edges
- DV: O(#Iterations x E)
  - where #Iterations is ideally O(network diameter) but varies due to routing loops or the count-to-infinity problem

Processing complexity

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

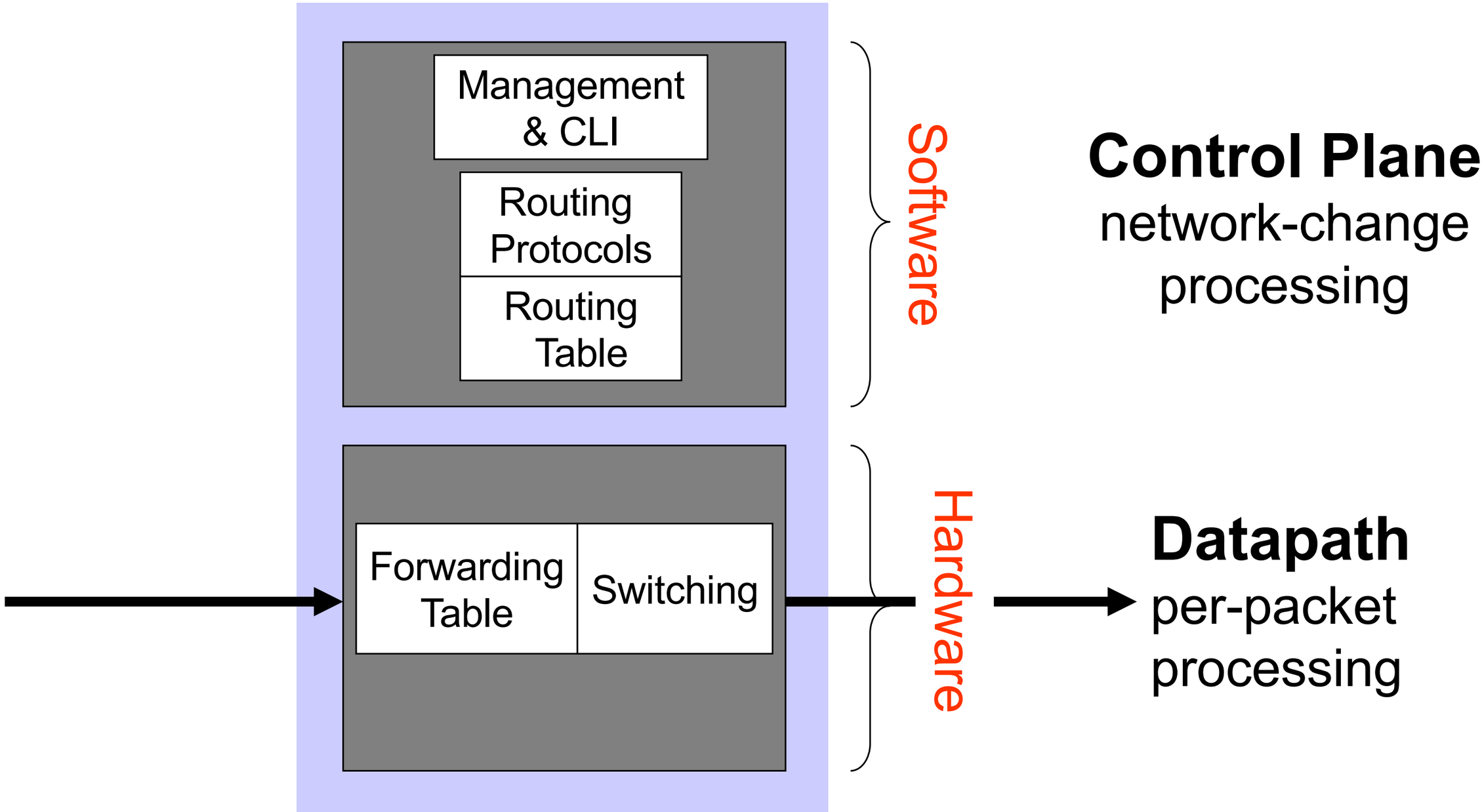
- Propagation delay
- Congestion
- Load balance
- Bandwidth (available, capacity, maximal, bbw)
- Price
- Reliability
- Loss rate
- Combinations of the above

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

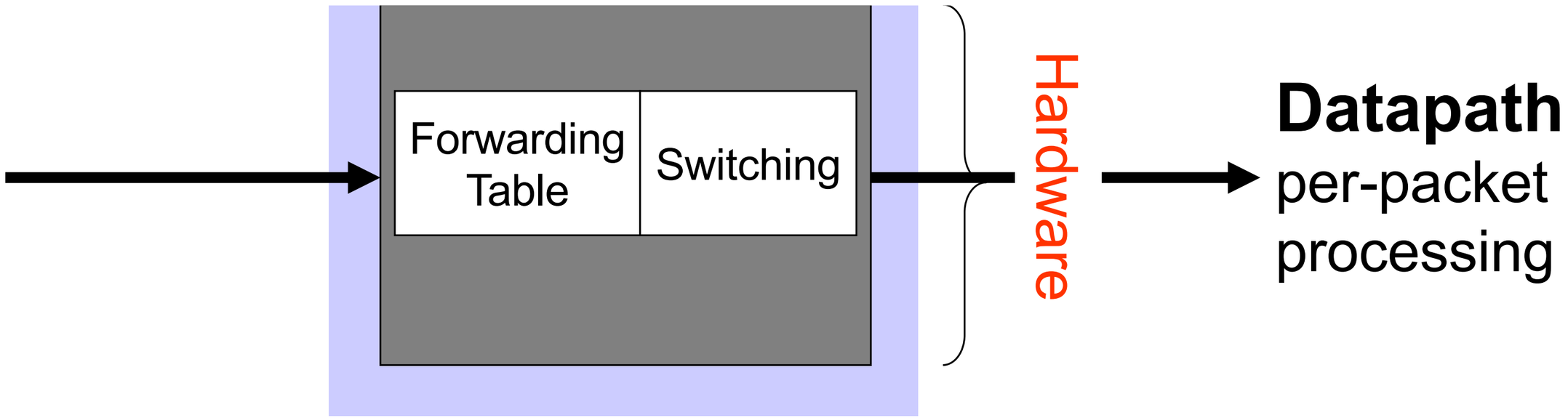


Independent operation!

If the control-plane **fails**…..

The data-path is **not affected**…
like a loyal pet it will keep going using the current (last) table update

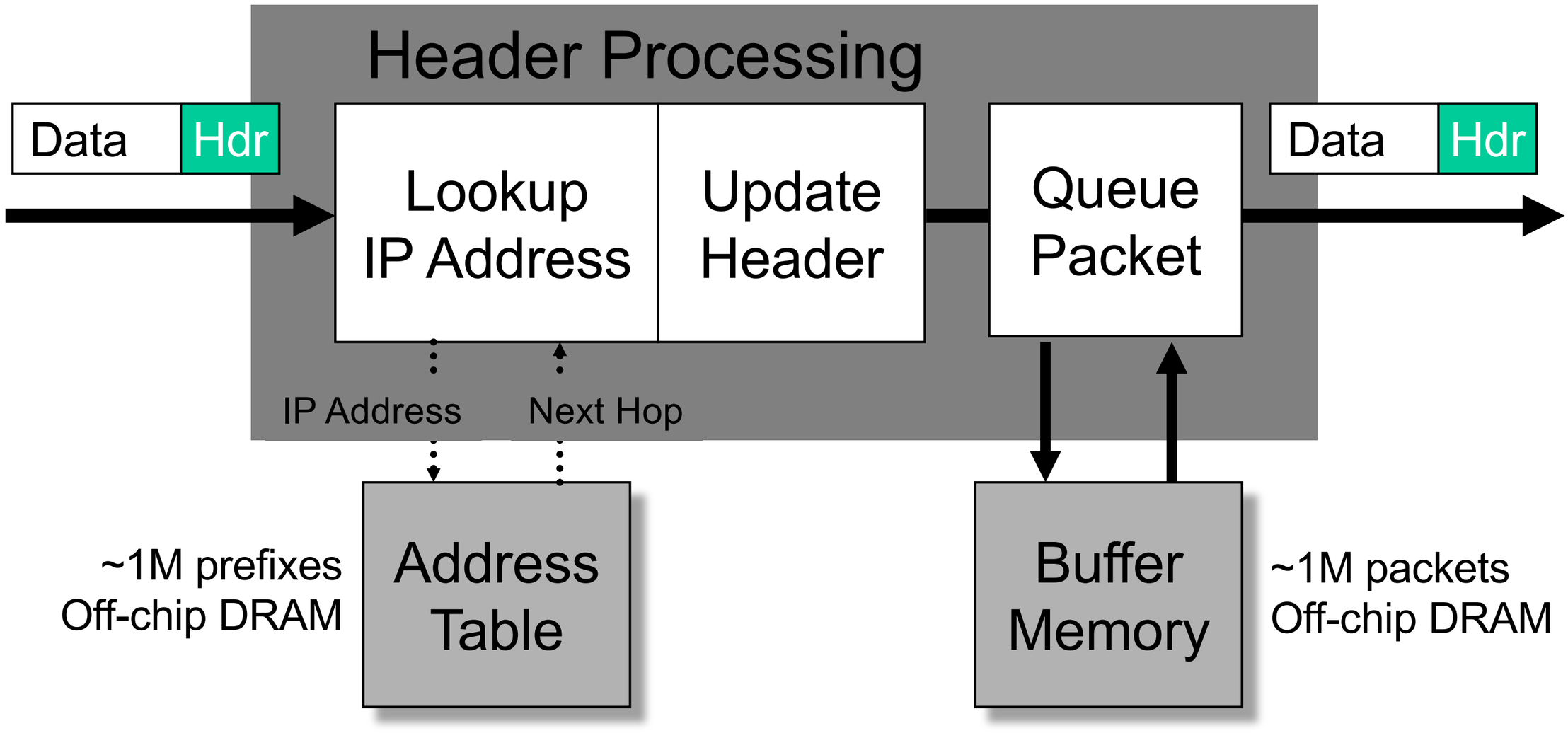
This is a feature **not** a bug



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



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

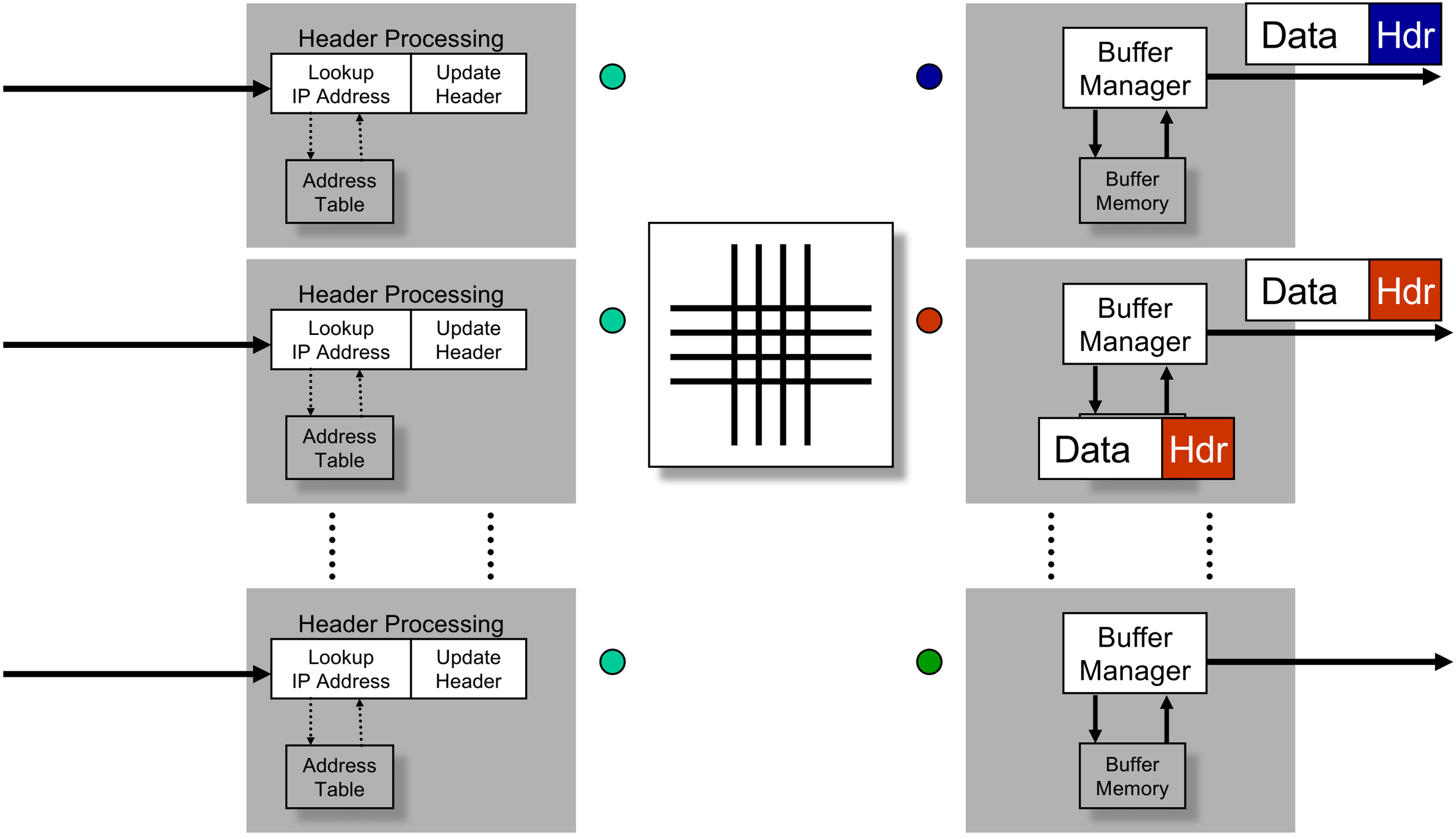
} **~ 4 billion entries**

Improved approach:

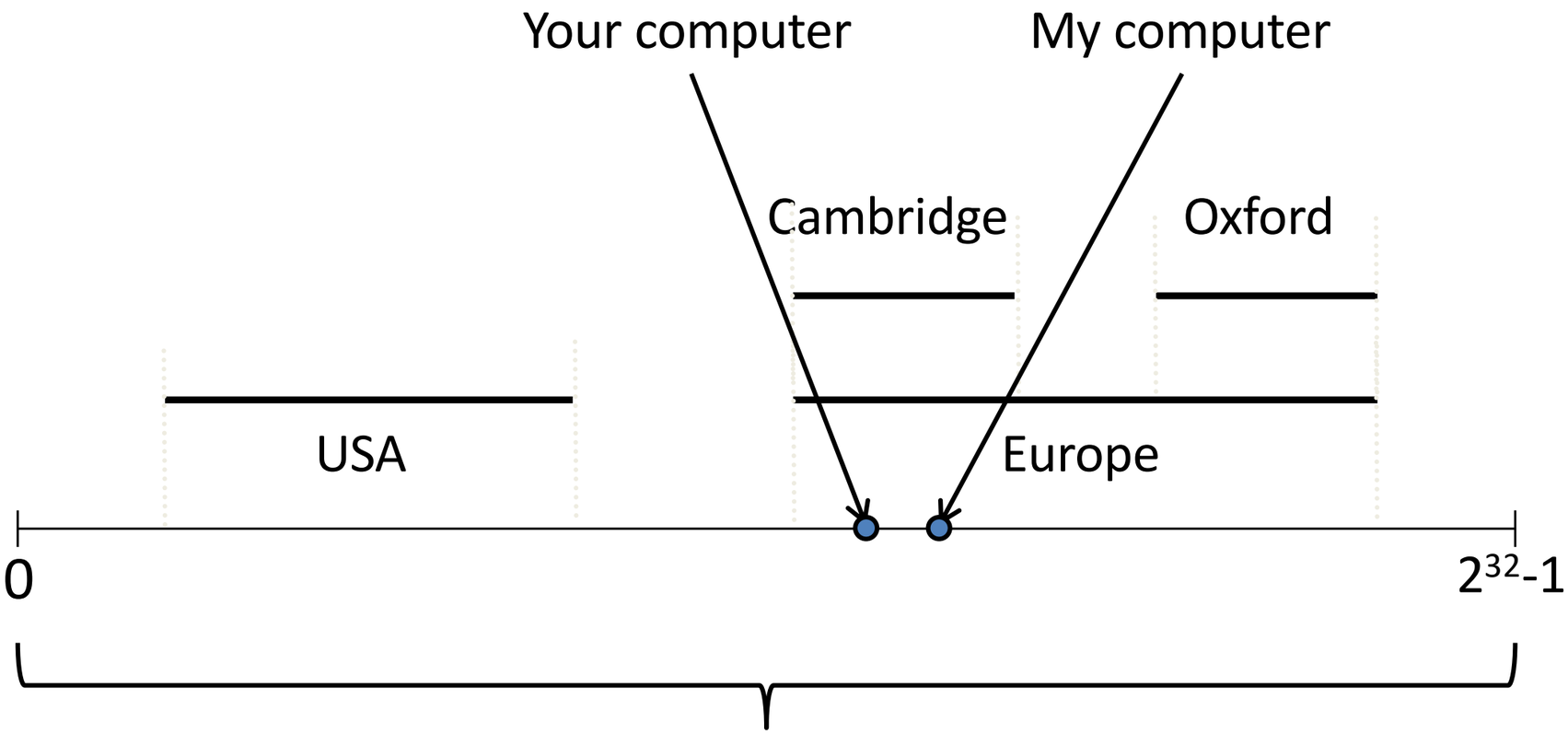
Group entries to reduce table size

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

Generic Router Architecture



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
2	Oxford	2
3	Europe	3
4	USA	4
5	Everywhere (default)	5

Universities

Continents

Planet

Matching entries:

- Cambridge — Most specific
- Europe
- Everywhere

To: Cambridge	Data

Longest Prefix Match (LPM)

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

Universities

Continents

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

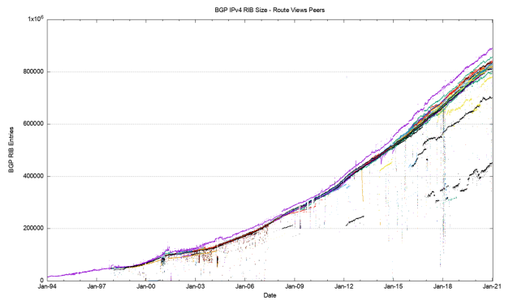
↓

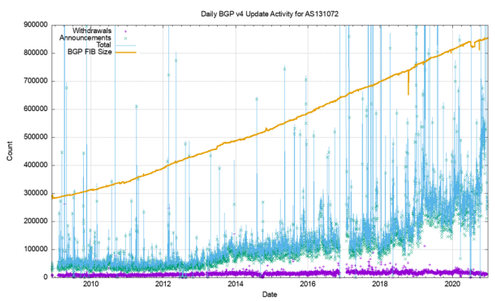
Least specific

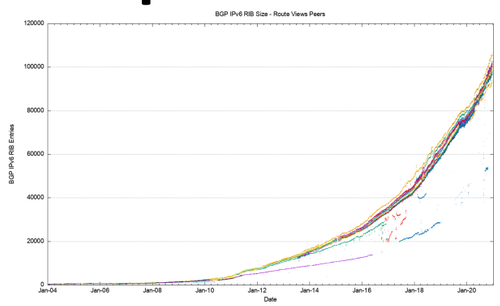
Forwarding table realities

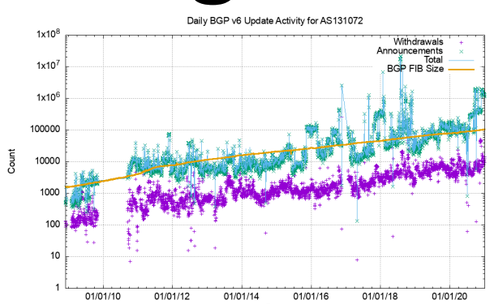
- High Speed: Must be “packet-rate” lookup
  - about 200M lookups / second for 100Gbps
- Large (messy) tables – (BGP Jan 2021 stats)
  - 866,000+ routing prefix entries for IPv4
  - 104,000+ routing prefix entries for IPv6
- Changing and Growing

the harsh side of “up and to the right”







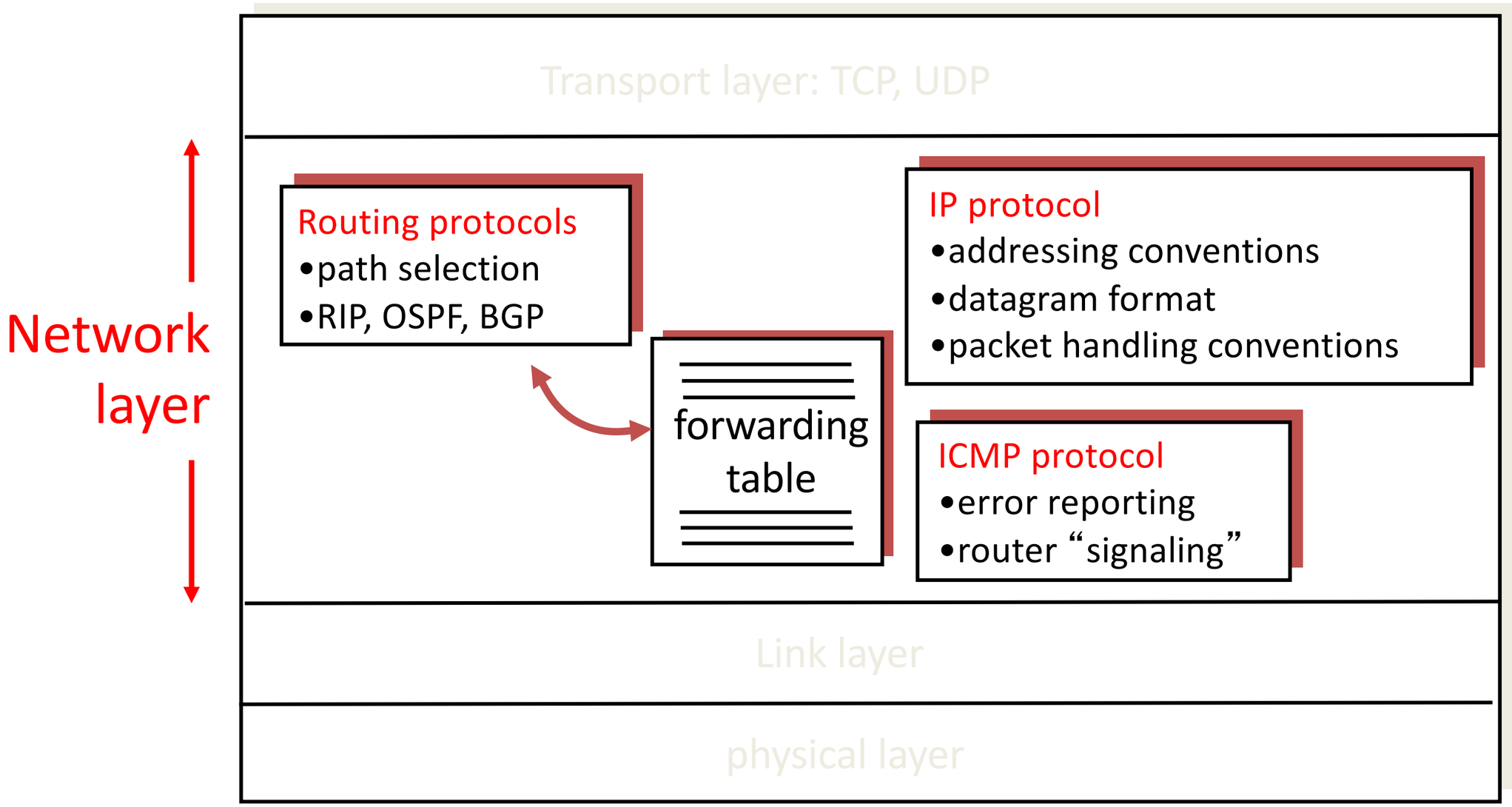


Open problems : continual growth is continual demand for innovation opportunities in control, algorithms, & network hardware

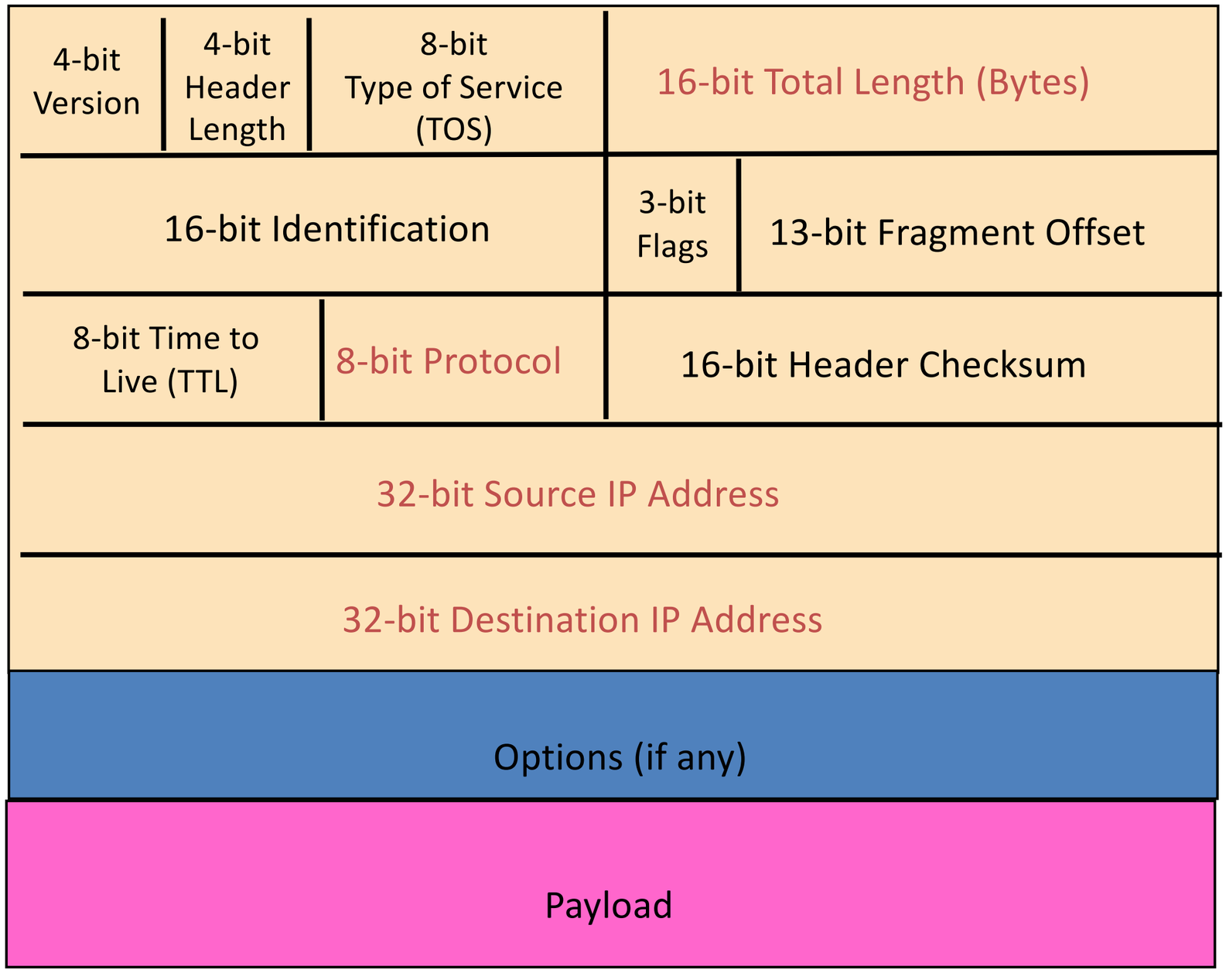
Hudson 2020 report https://blog.apnic.net/2021/01/05/bgp-in-2020-the-bgp-table/

The Internet version of a Network layer

Host, router network layer functions:



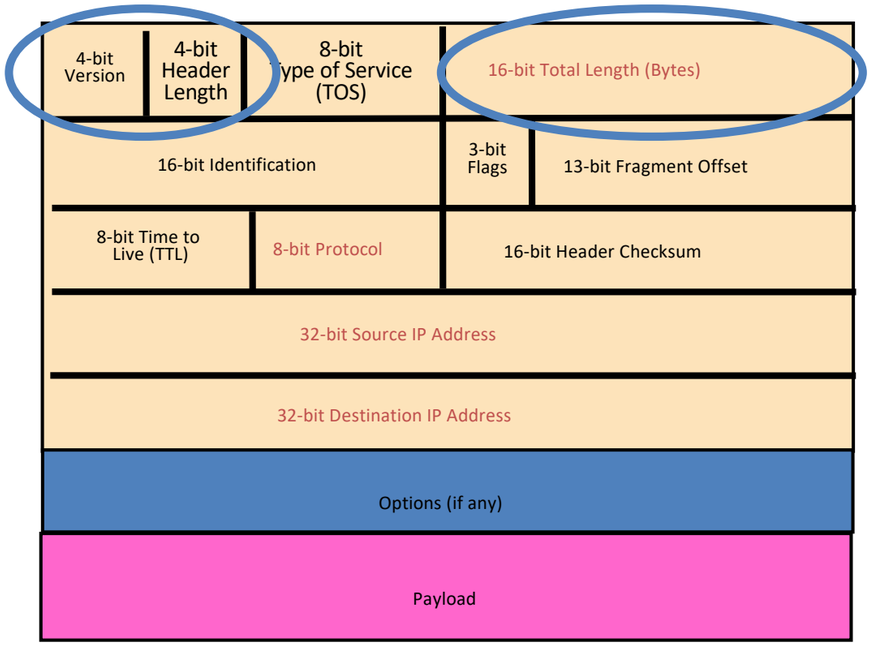
IPv4 Packet Structure
20 Bytes of Standard Header, then Options



(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

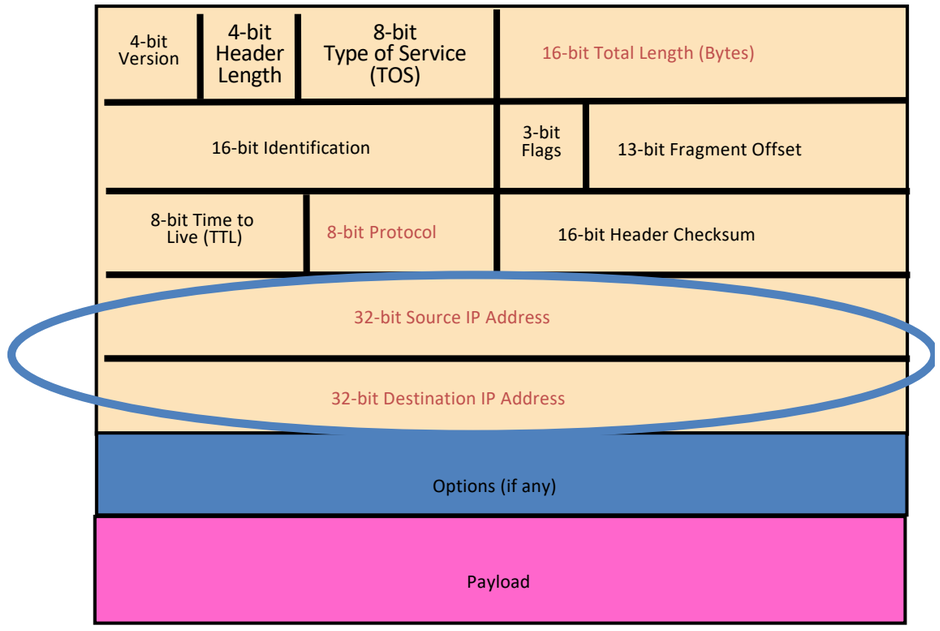
Reading Packet Correctly



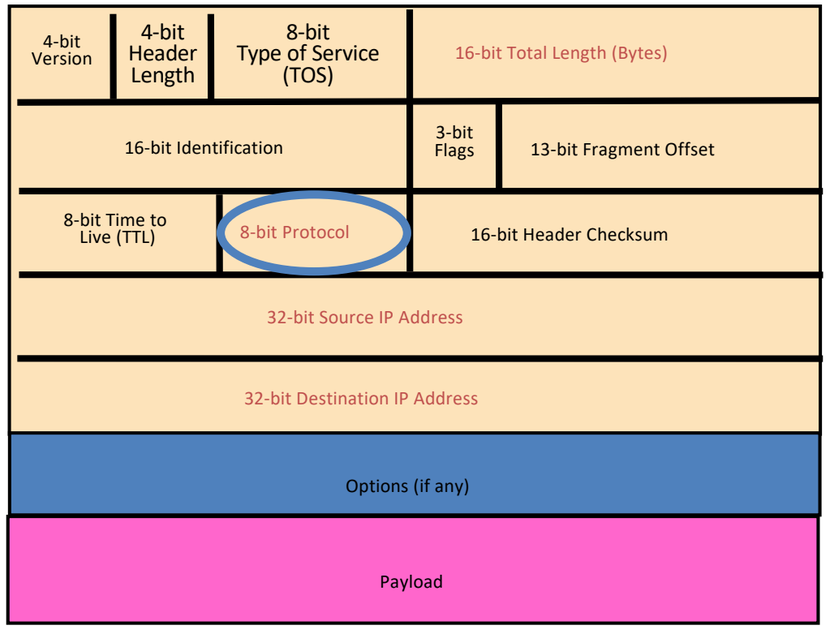
- Version number (4 bits)
  - Indicates the version of the IP protocol
  - Necessary to know what other fields to expect
  - Typically “4” (for IPv4), and sometimes “6” (for IPv6)
- Header length (4 bits)
  - Number of 32-bit words in the header
  - Typically “5” (for a 20-byte IPv4 header)
  - Can be 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

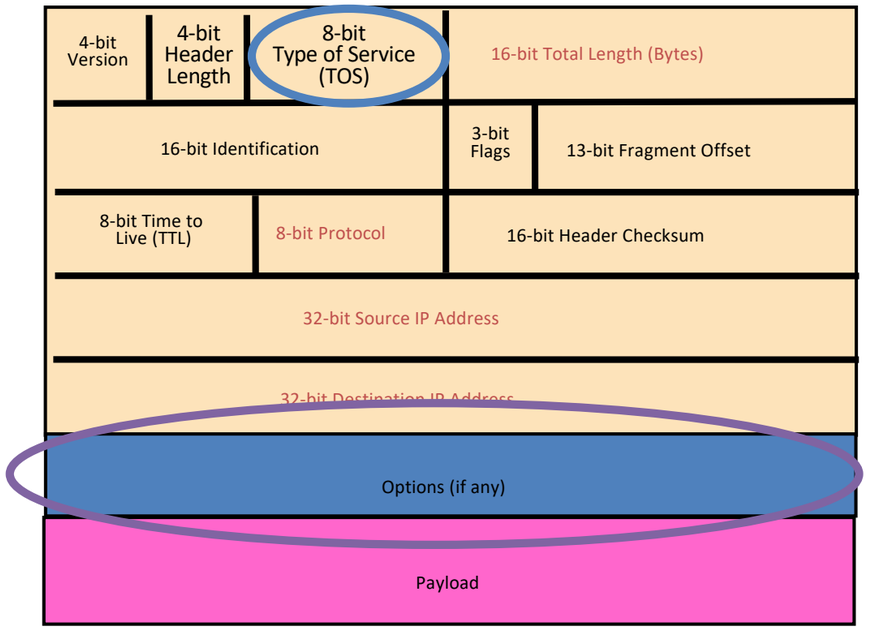


Telling Host How to Handle Packet



- 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)

Special Handling

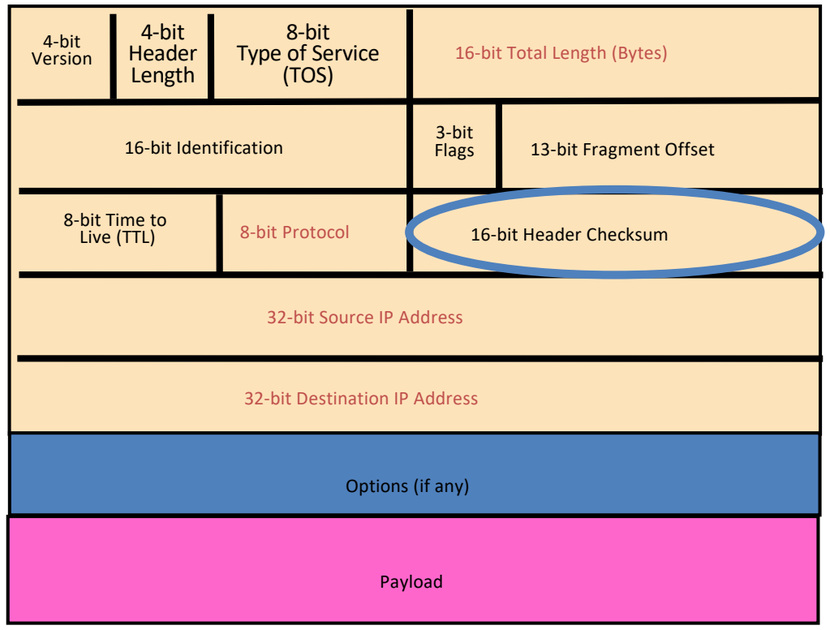


- Type-of-Service (8 bits)
  - Allow packets to be treated differently based on needs
  - E.g., low delay for audio, high bandwidth for bulk transfer
  - Has been redefined several times
- Options

Potential Problems

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

Header Corruption



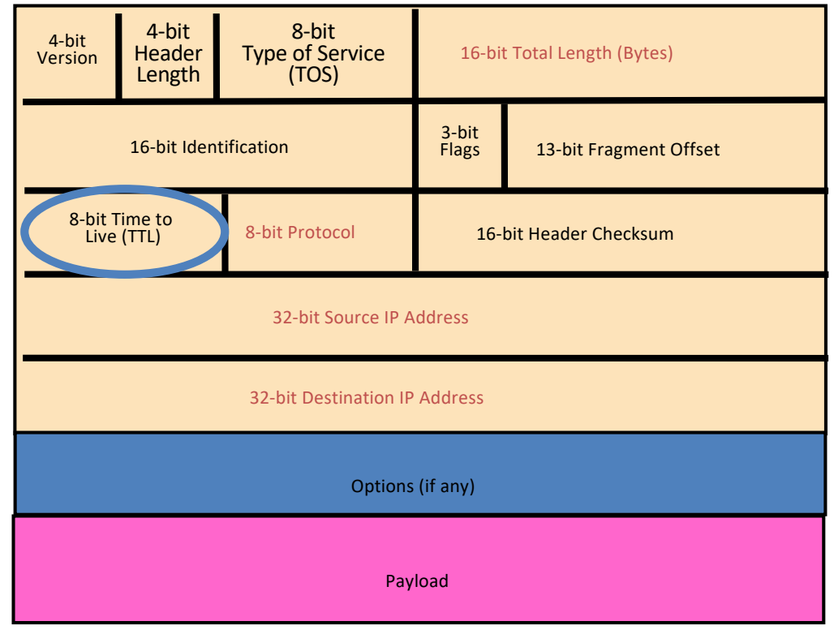
- Checksum (16 bits)
  - Particular form of checksum over packet header

- If not correct, router discards packets
  - So it doesn't act on bogus information

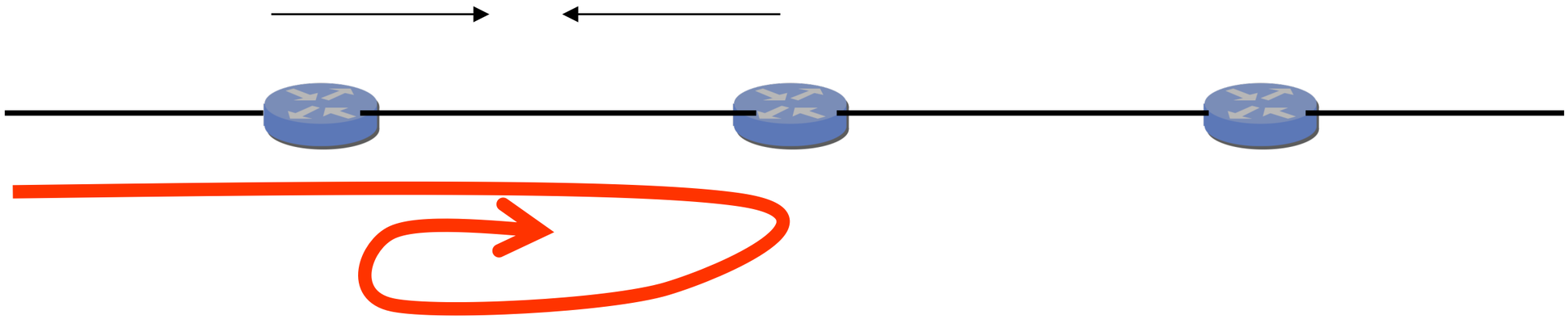
- Checksum recalculated at every router
  - **Why?**
  - **Why include TTL?**
  - **Why only header?**

Preventing Loops

(aka Internet Zombie plan)



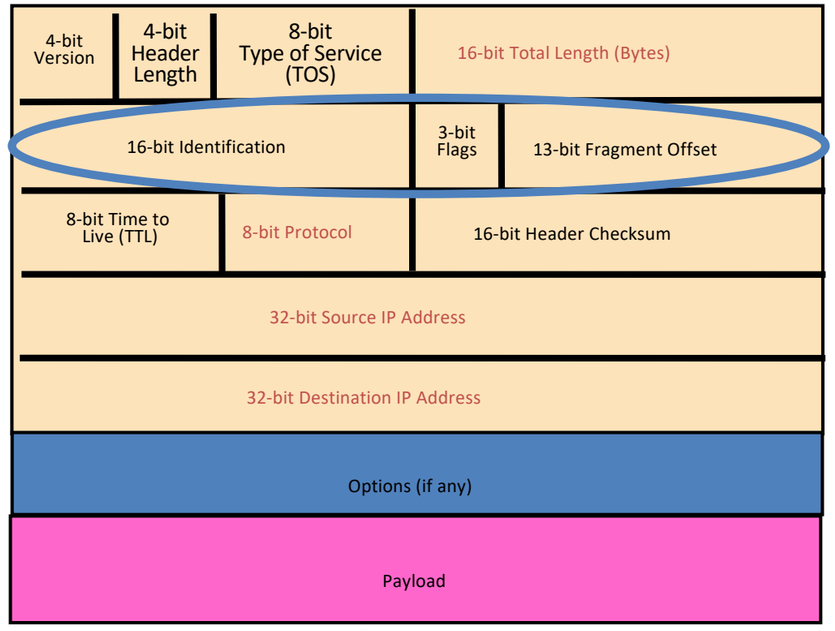
- Forwarding loops cause packets to cycle forever
  - As these accumulate, eventually consume **all** capacity



- Time-to-Live (TTL) Field (8 bits)
  - Decremented 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)

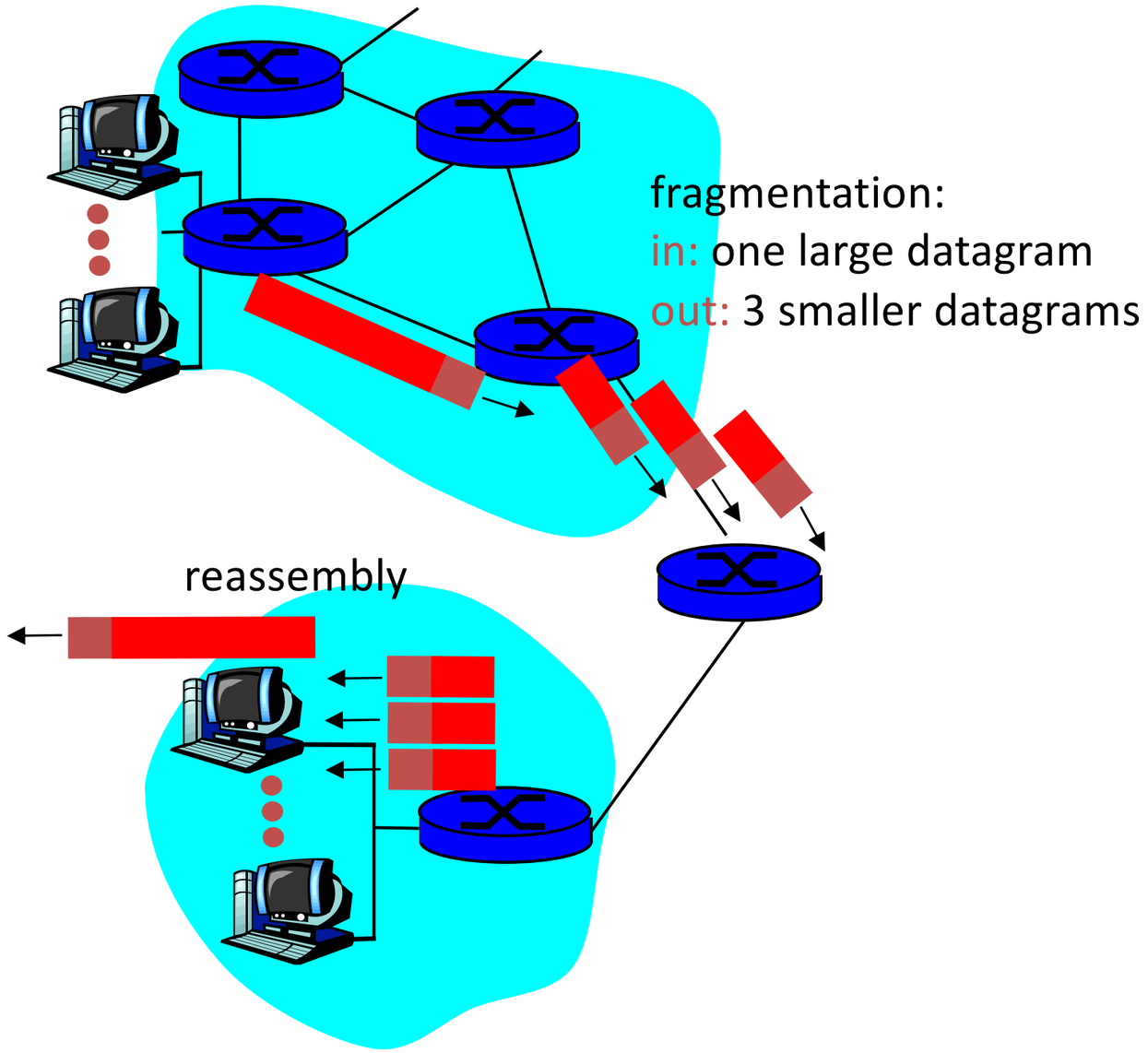


- Fragmentation: when forwarding a packet, an Internet router can split it into multiple pieces ("fragments") if too big for next hop link

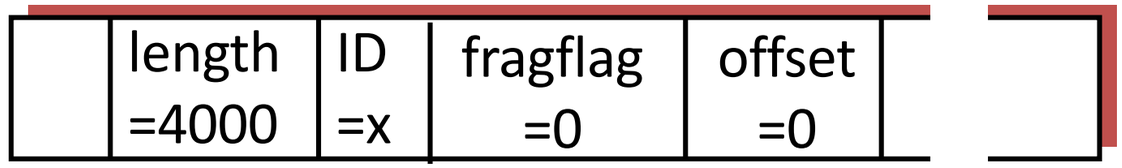
- Must reassemble to recover original packet
  - Need fragmentation information (32 bits)
  - Packet identifier, flags, and fragment offset

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



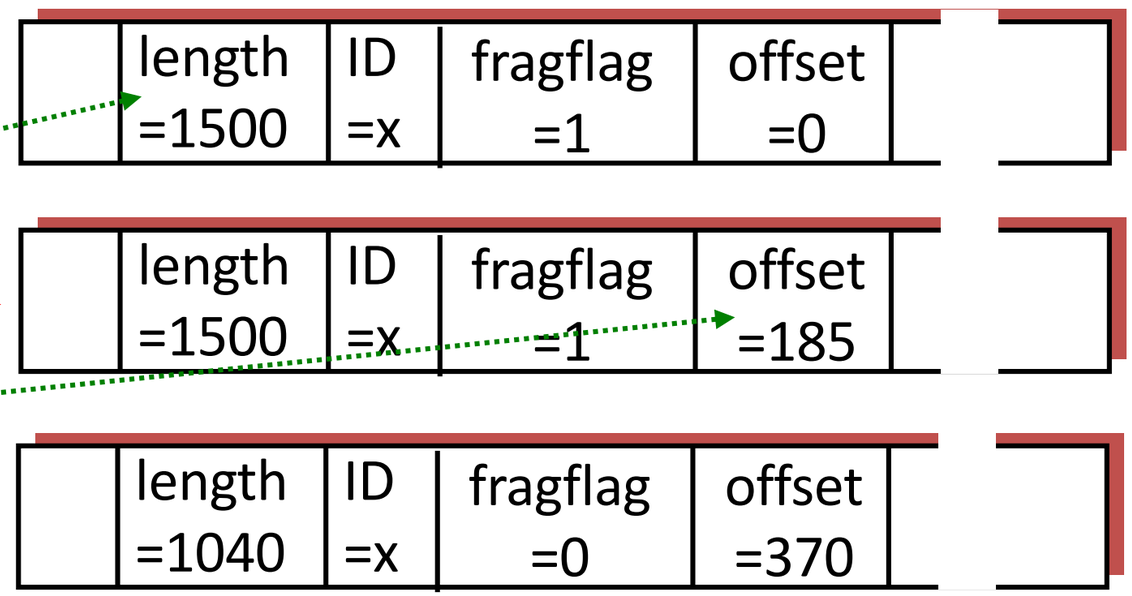
Example

- 4000 byte datagram
- MTU = 1500 bytes

One large datagram becomes several smaller datagrams

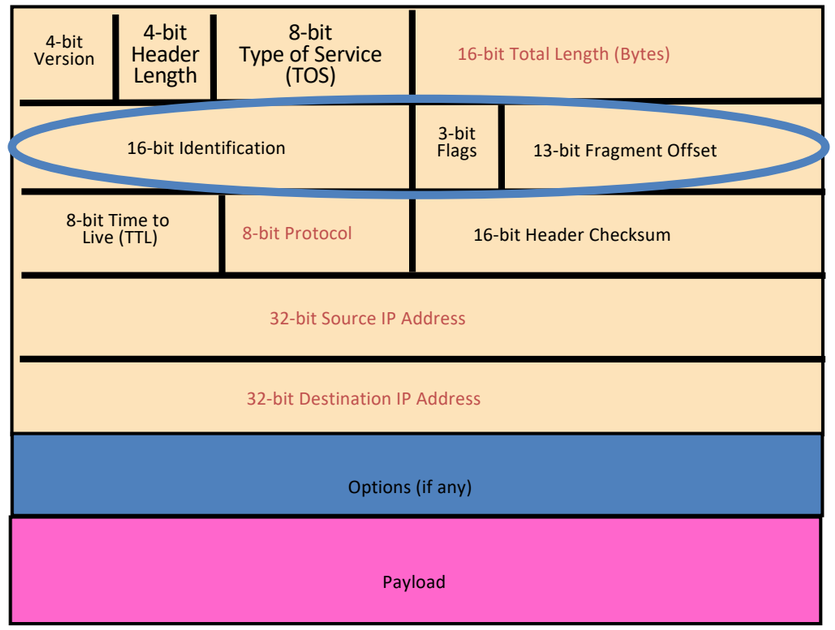
1480 bytes in data field

offset = 1480/8



Question: What happens when a fragment is lost?

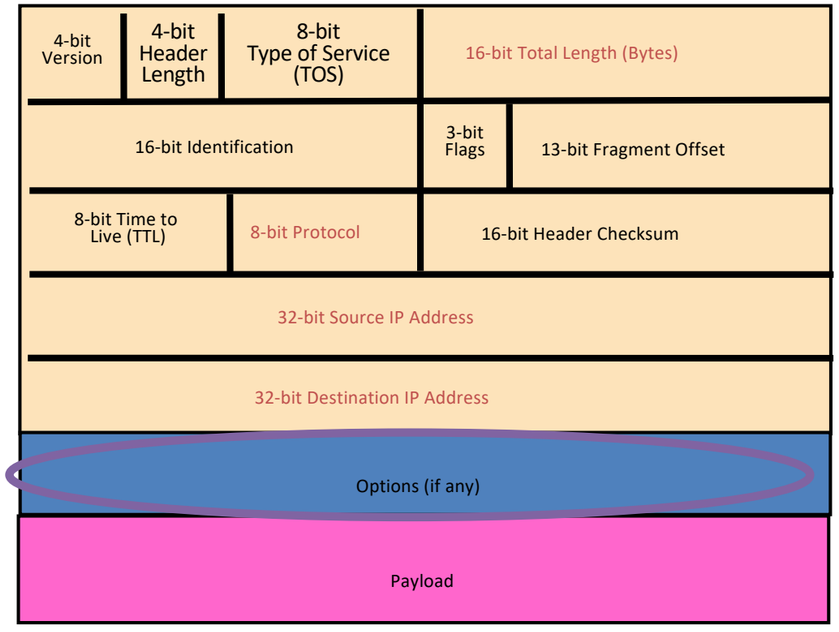
Fragmentation Details



- 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?

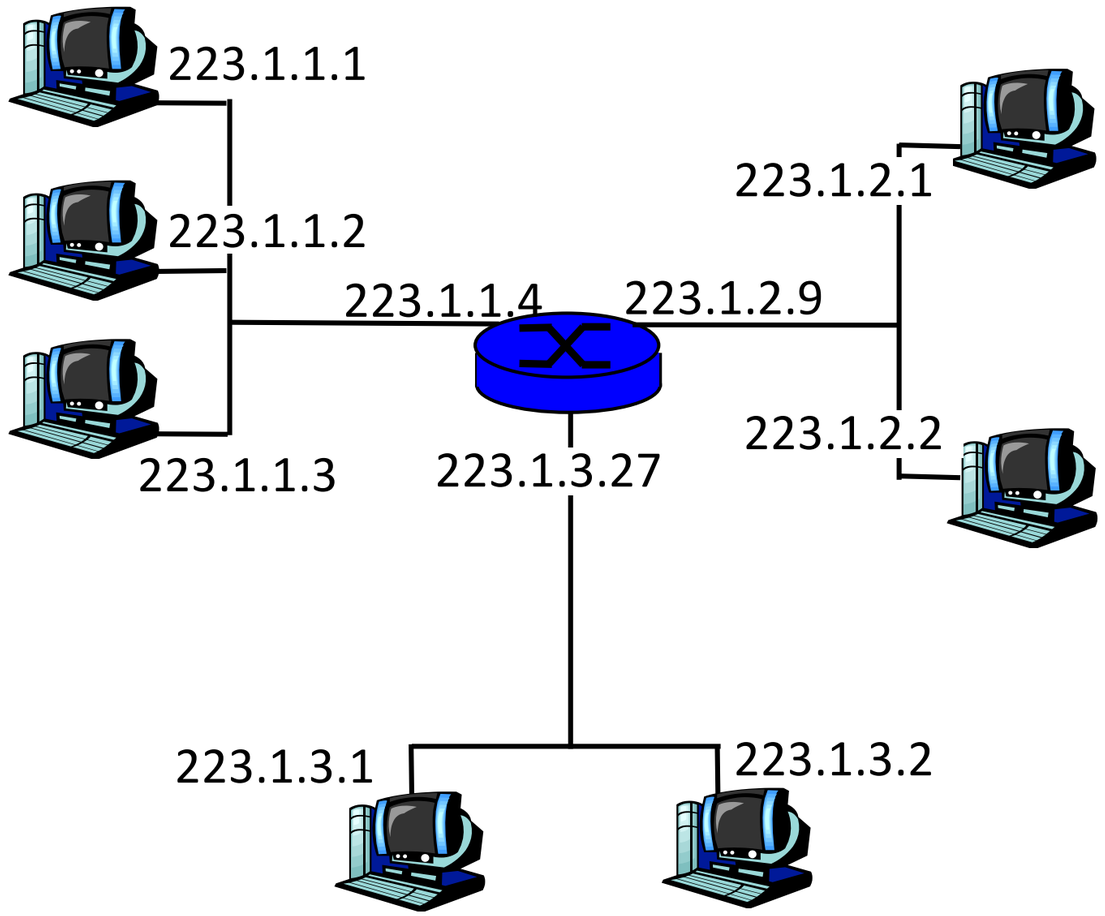
Options



- 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
  - routers typically have multiple interfaces
  - host typically has one interface
  - IP addresses associated with each interface



223.1.1.1 = 11011111 00000001 00000001 00000001

223 1 1 1

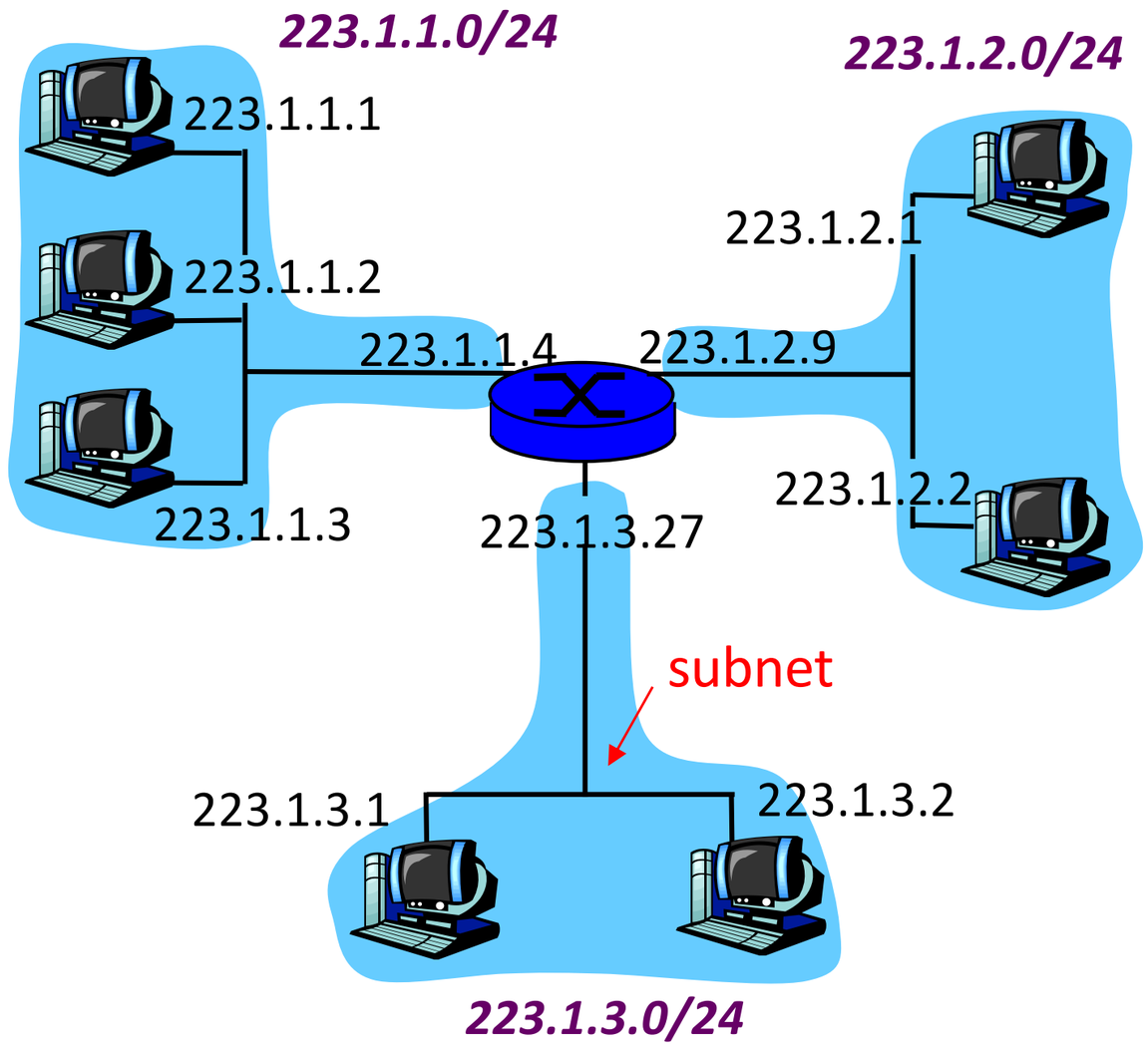
Subnets

- 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

223.1.3.0/24

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?

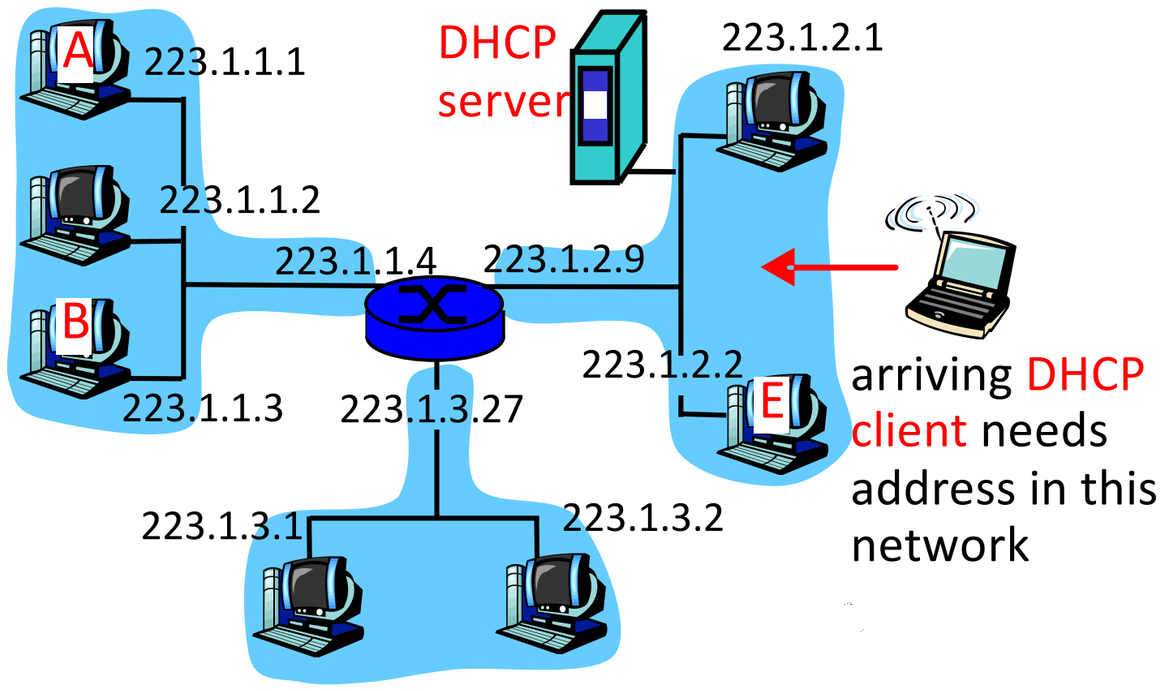
Q: How does a *host* get IP address?

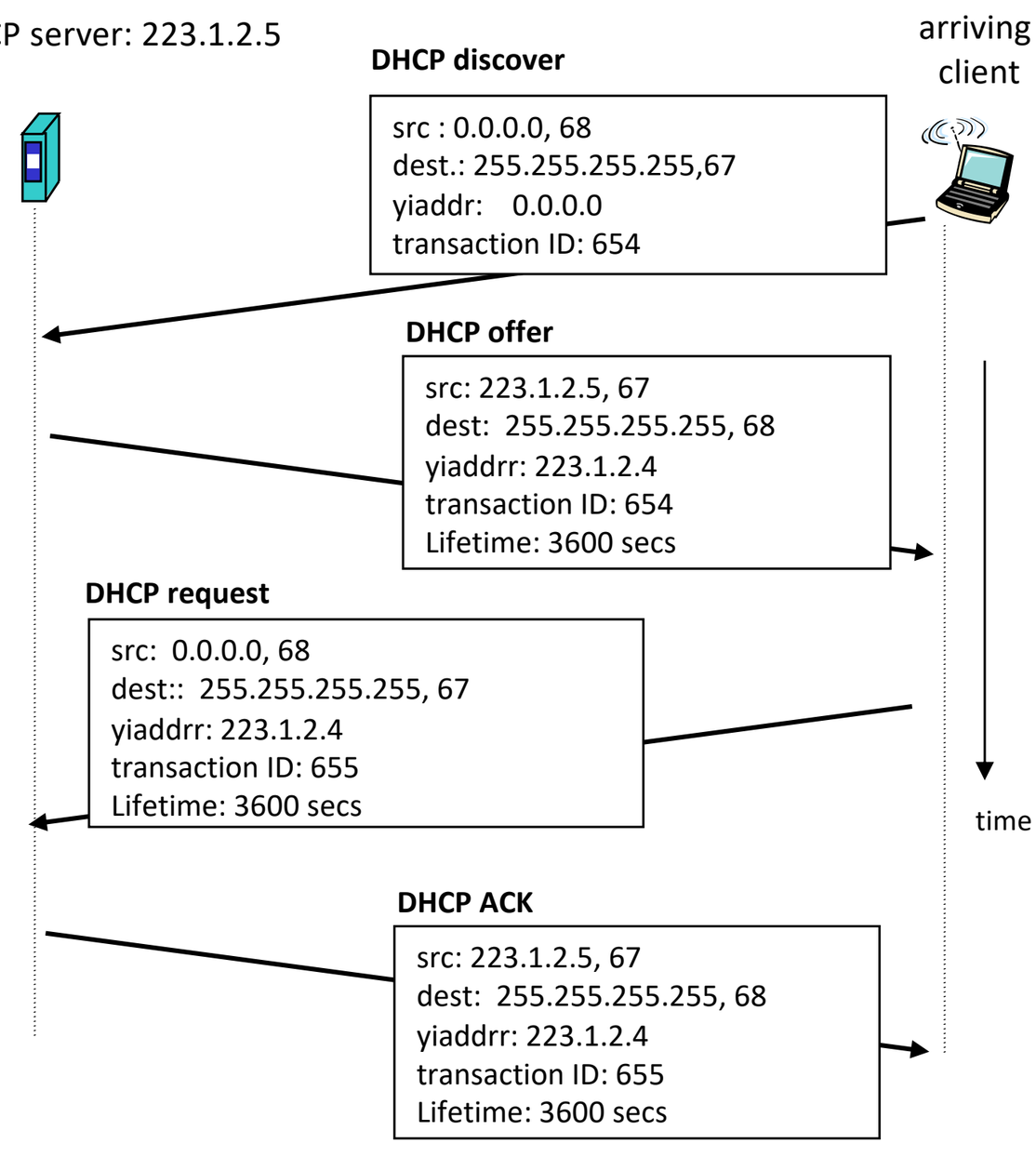
- hard-coded by system admin in a file
  - Windows: control-panel->network->configuration->tcp/ip->properties
  - UNIX: /etc/rc.config (circa 1980's your mileage will vary)
- DHCP: Dynamic Host Configuration Protocol: dynamically get address from as server
  - "plug-and-play"

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?

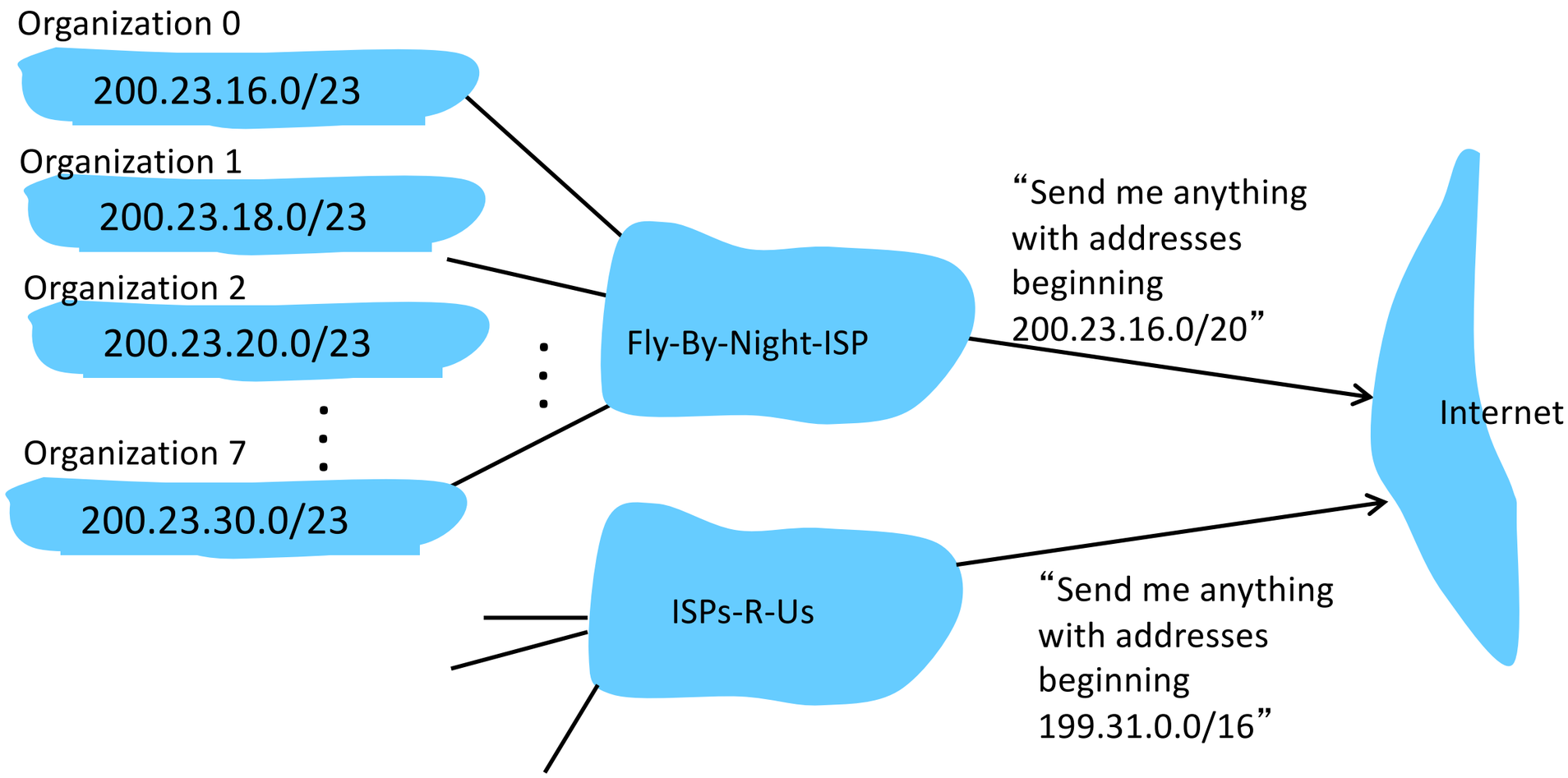
Q: How does *network* get subnet part of IP addr?

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

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

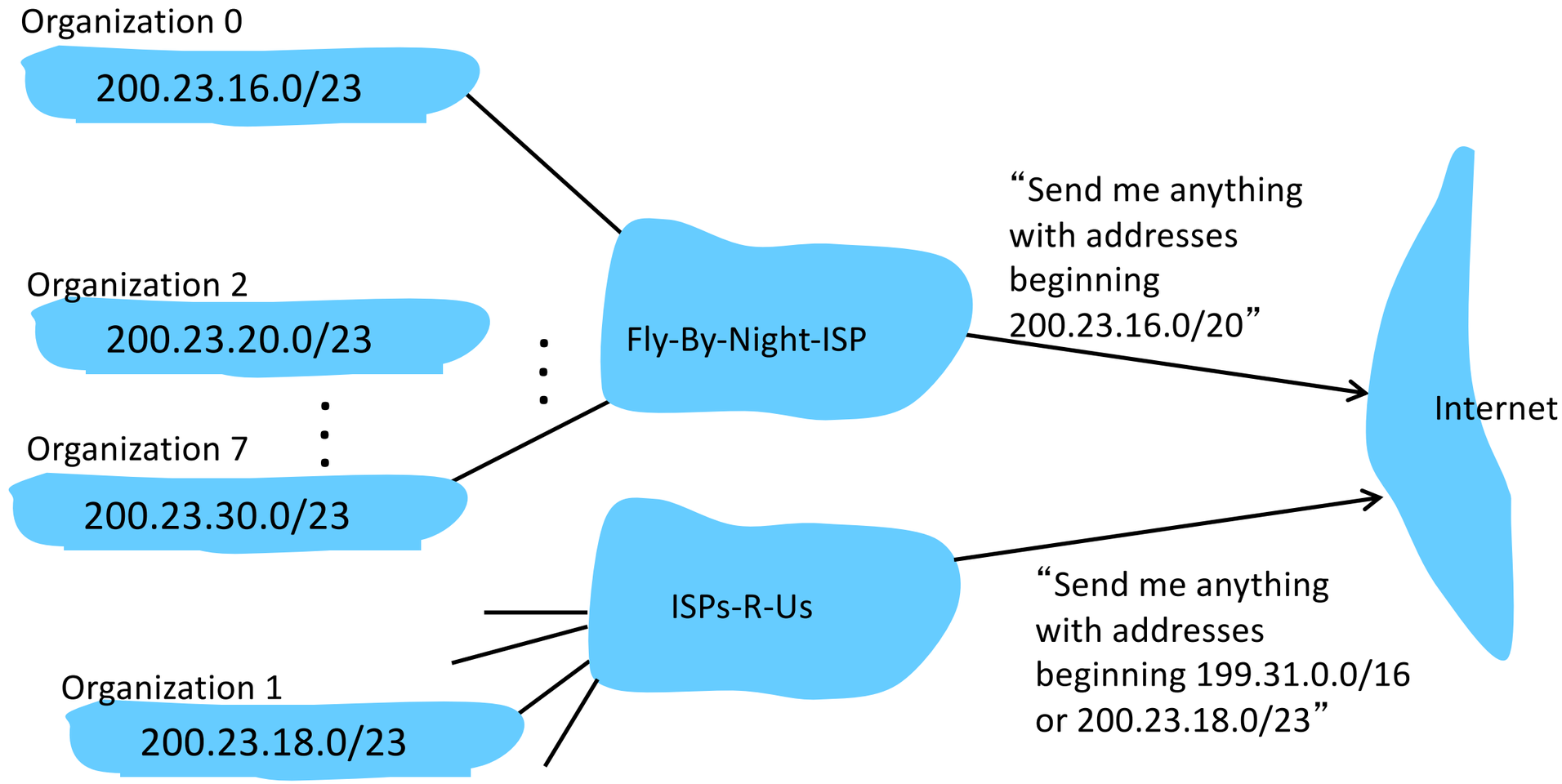
Hierarchical addressing: route aggregation

Hierarchical addressing allows efficient advertisement of routing information:



Hierarchical addressing: more specific routes

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



IP addressing: the last word...

Q: How does an ISP get a block of addresses?

A: ICANN: Internet Corporation for Assigned Names and Numbers

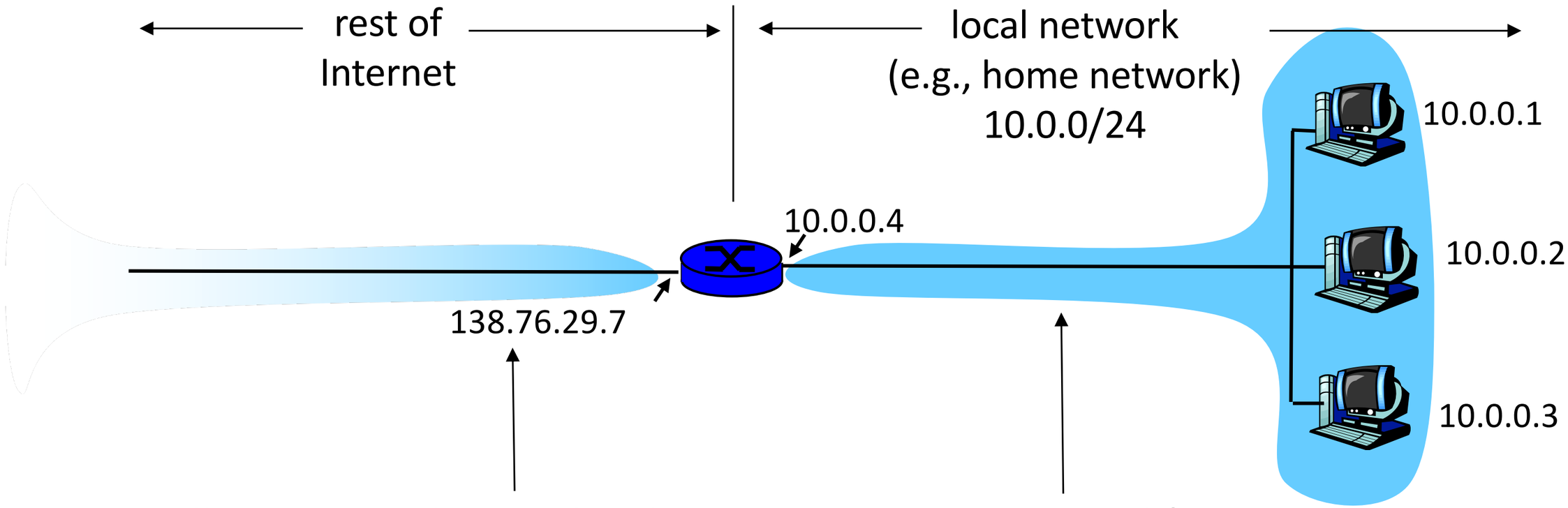
– allocates addresses

– 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

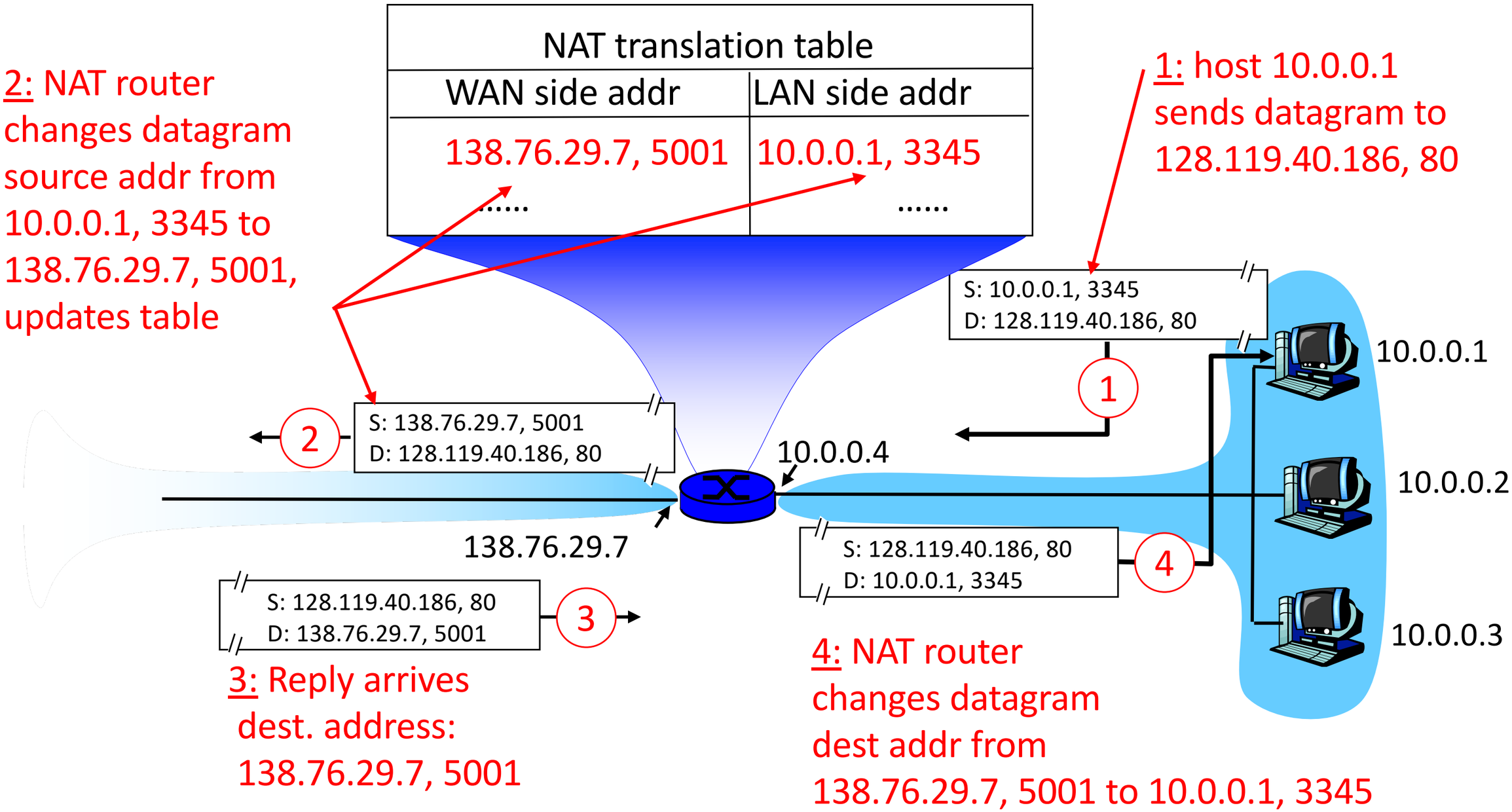
- Motivation: local network uses just one IP address as far as outside world is concerned:
  - range of addresses not needed from ISP: just one IP address for all devices
  - can change addresses of devices in local network without notifying outside world
  - can change ISP without changing addresses 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

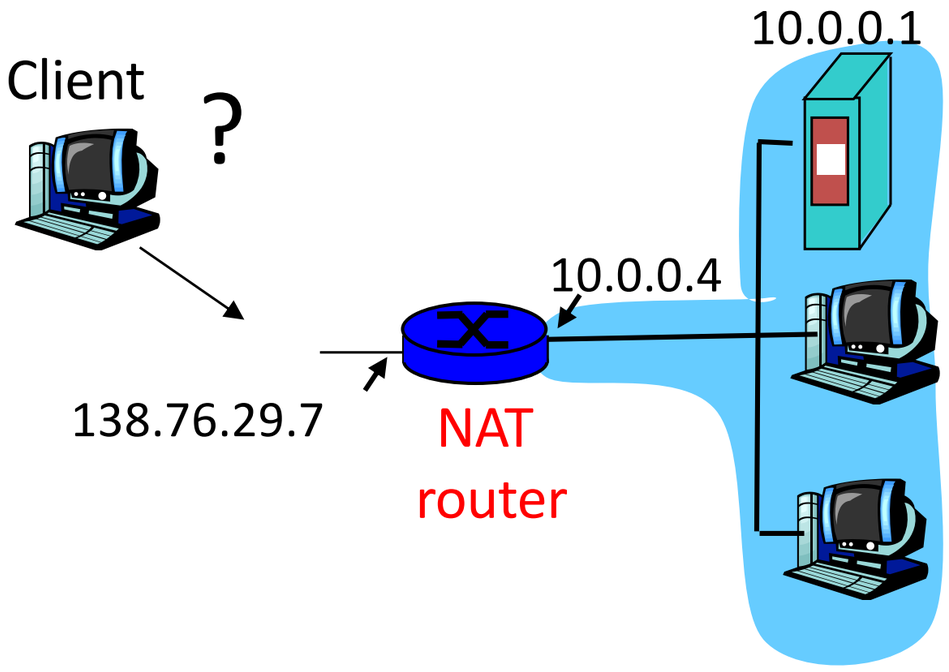


NAT: Network Address Translation

- 16-bit port-number field:
  - 60,000+ simultaneous connections with a single WAN-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

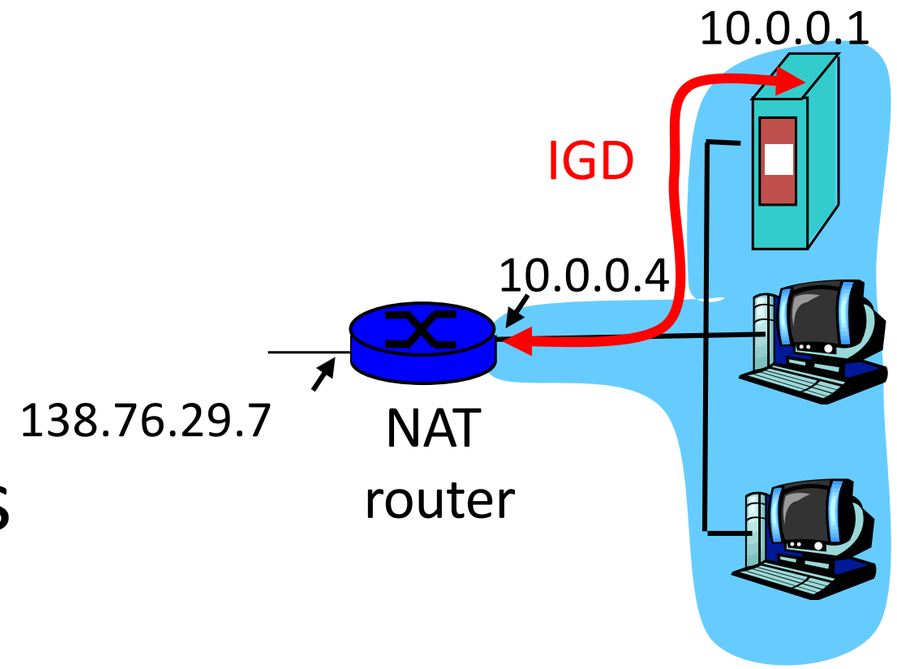
- client wants to connect to server with address 10.0.0.1
  - server address 10.0.0.1 local to LAN (client can't use it as destination addr)
  - only one externally visible NATted address: 138.76.29.7
- solution 1: statically configure NAT to forward incoming connection requests at given port to server
  - e.g., (138.76.29.7, port 2500) always forwarded to 10.0.0.1 port 25000



NAT traversal problem

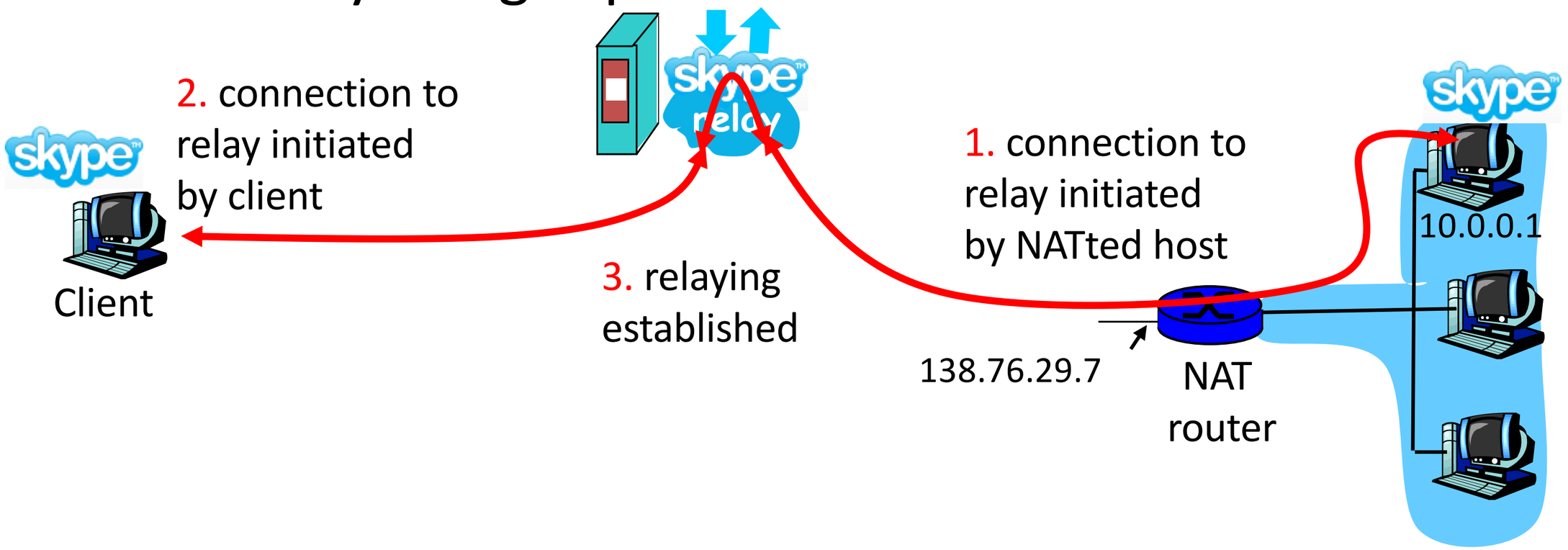
- solution 2: Universal Plug and Play (UPnP) Internet Gateway Device (IGD) Protocol. Allows NATted host to:
  - ❖learn public IP address (138.76.29.7)
  - ❖add/remove port mappings (with lease times)

i.e., automate static NAT port map configuration



NAT traversal problem

- solution 3: relaying (was used in (really old) 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 windows is similar)

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

```
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
```

trans-continent link

* 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

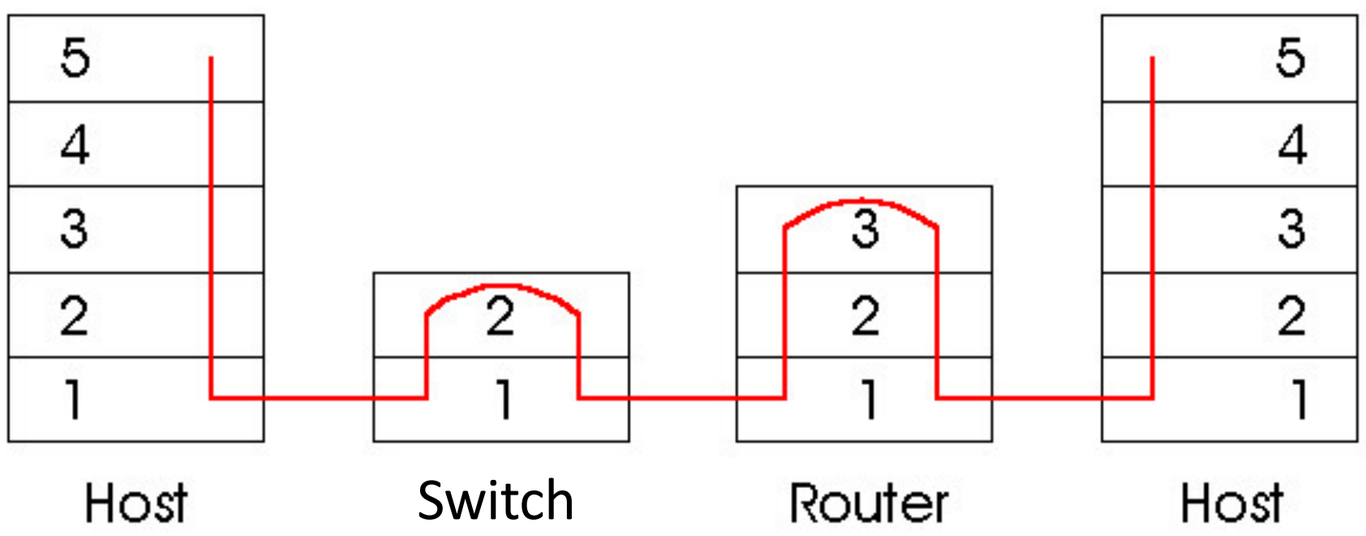
Type	Code	description
0	0	echo reply (ping)
3	0	dest. network unreachable
3	1	dest host unreachable
3	2	dest protocol unreachable
3	3	dest port unreachable
3	6	dest network unknown
3	7	dest host unknown
4	0	source quench (congestion control - not used)
8	0	echo request (ping)
9	0	route advertisement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

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 eg IP)
  - switches are link layer devices (examine Data-Link-Layer headers eg Ethernet)
- Routers: implement routing algorithms, maintain routing tables of the network – create network forwarding tables from routing tables
- Switches: implement learning algorithms, learn switch/DLL forwarding tables



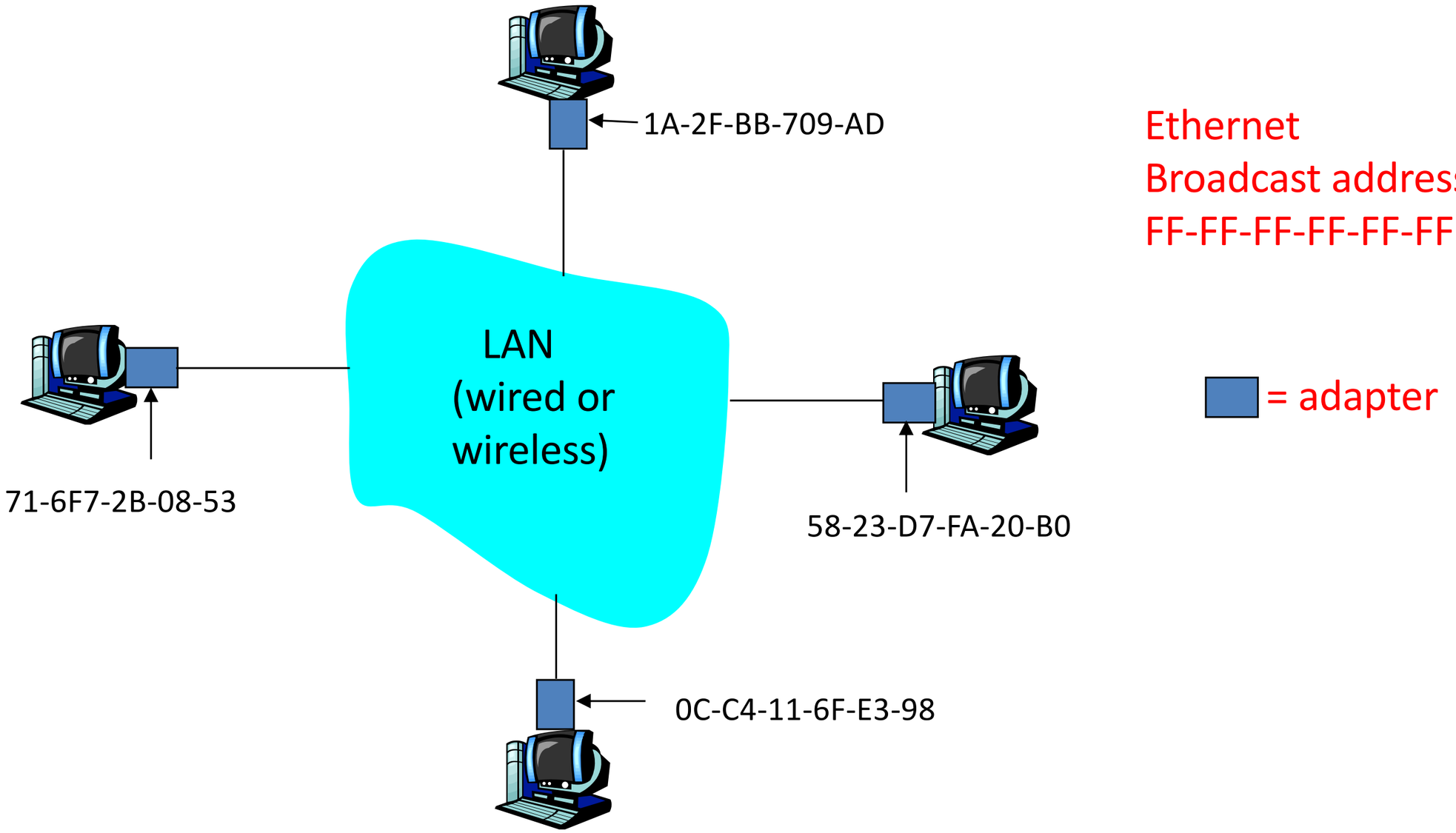
MAC Addresses (and IPv4 ARP)

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, firmware, etc.

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

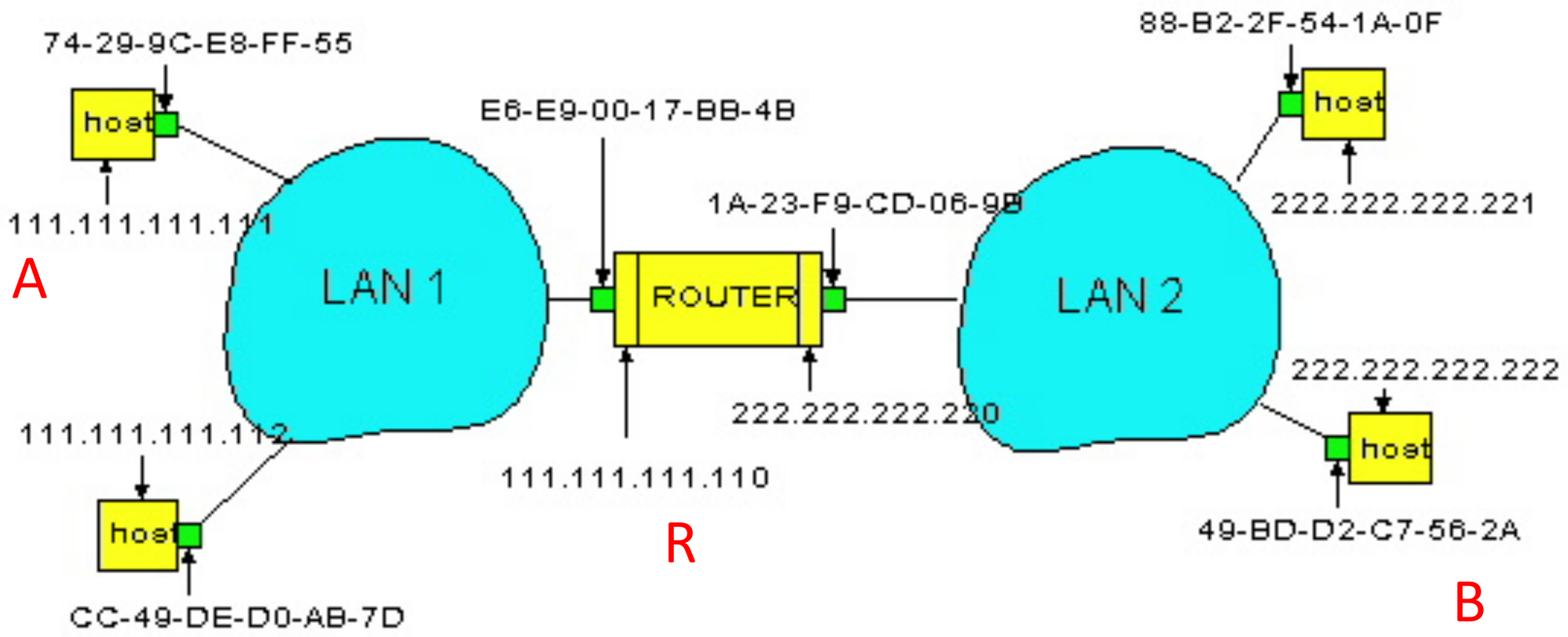
- Every node maintains an ARP table
  - <IP address, MAC address> pair

- 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

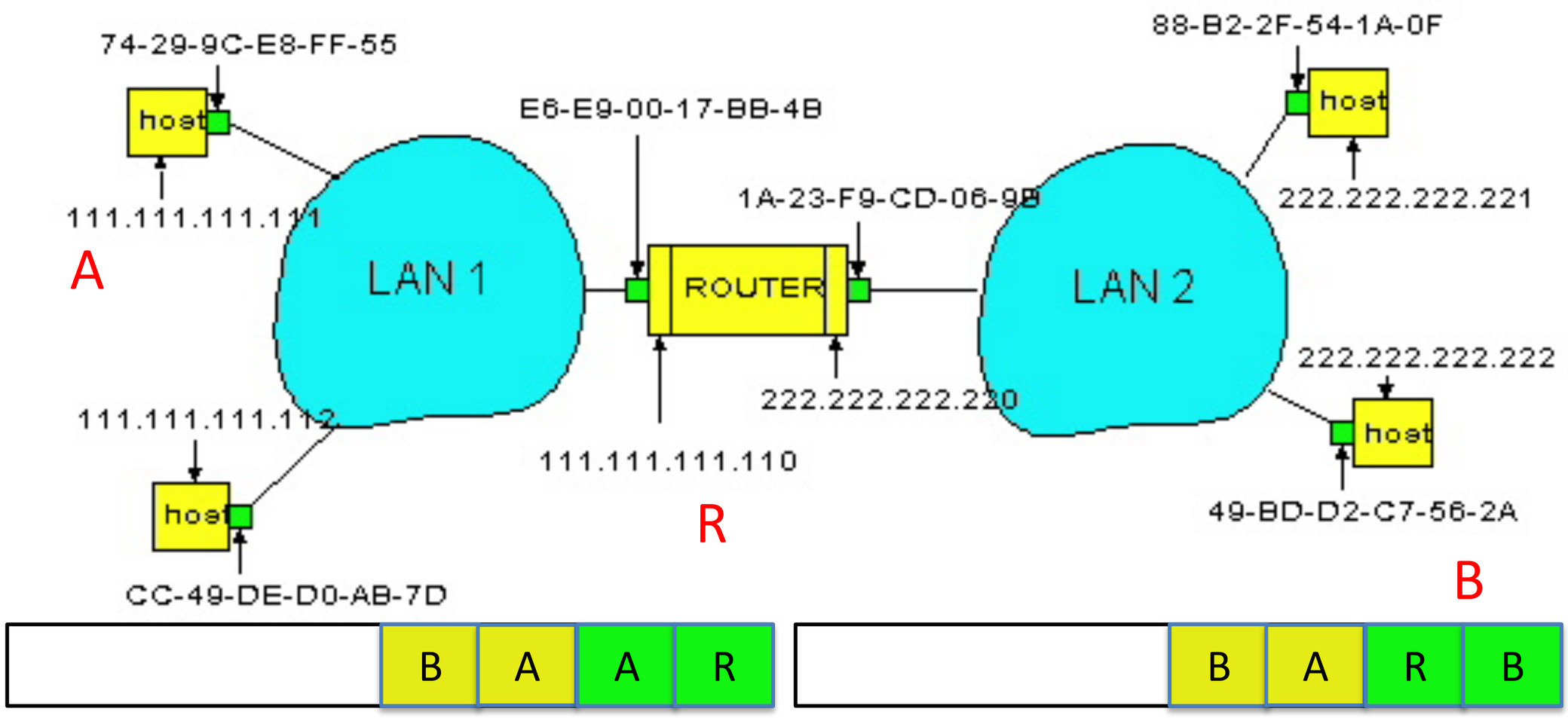
Example: A Sending a Packet to B

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



Example: A Sending a Packet to B

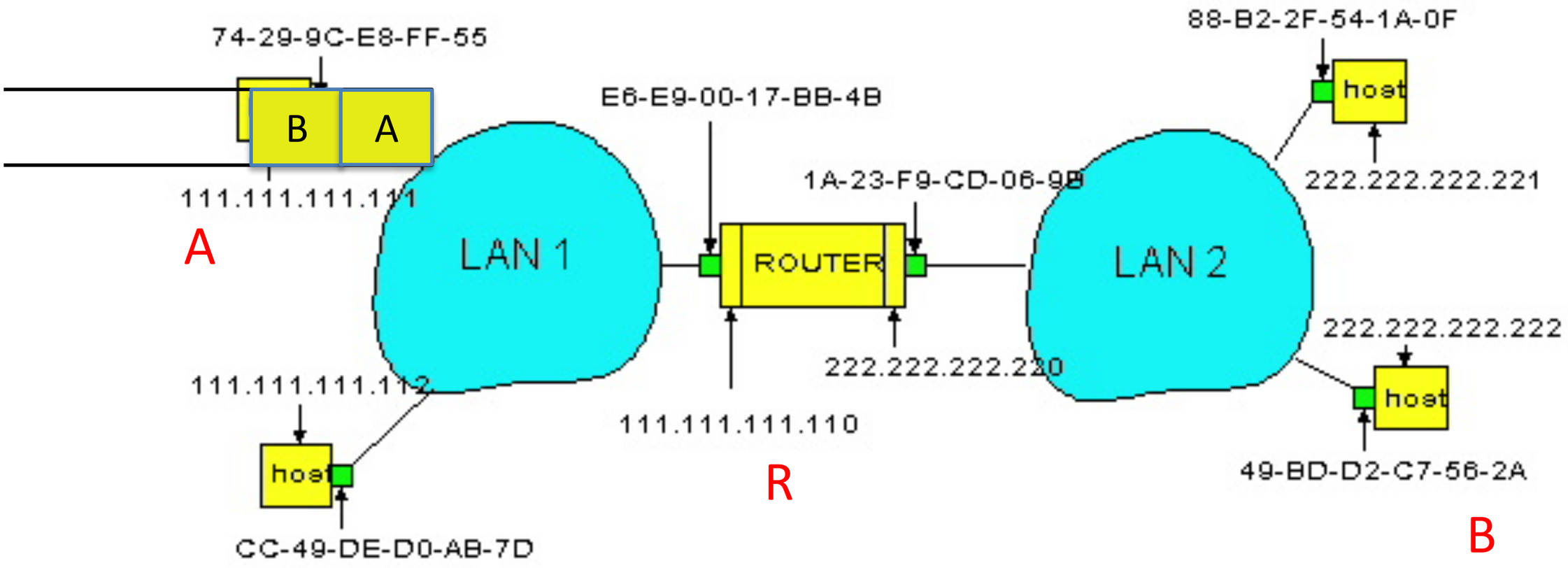
How does host A send an IP packet to host B?



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

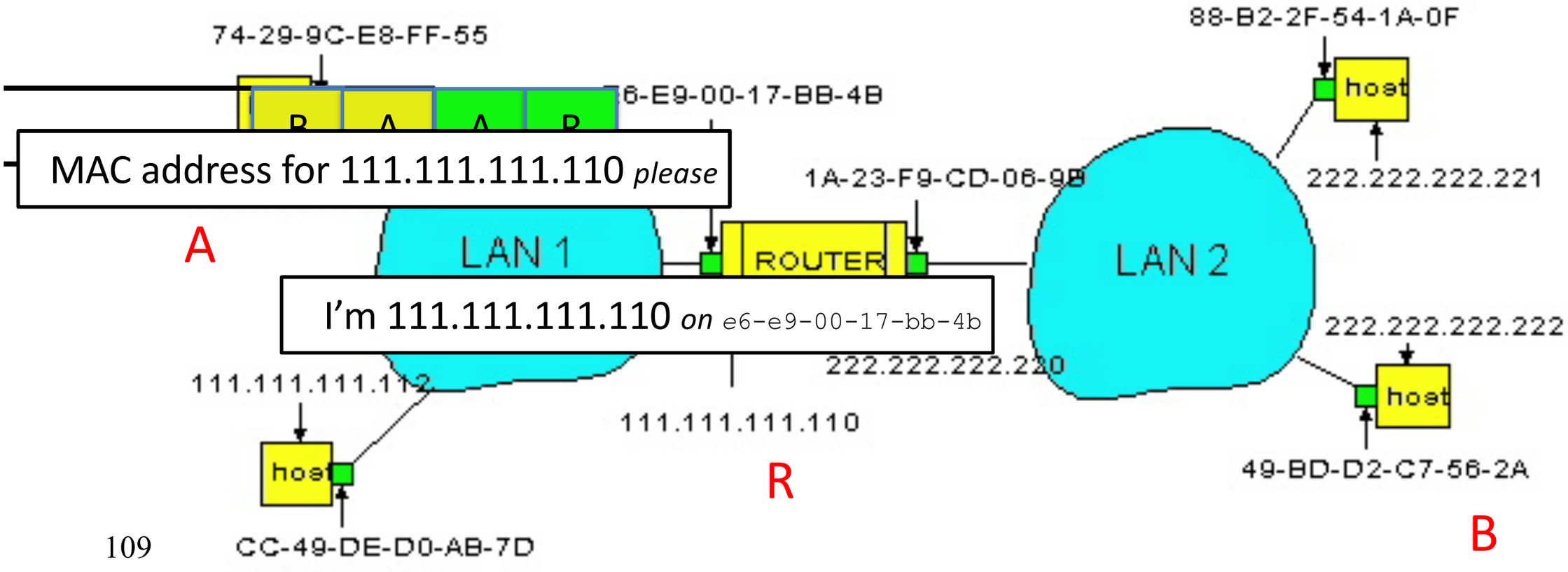
Host A Decides to Send Through R

- 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



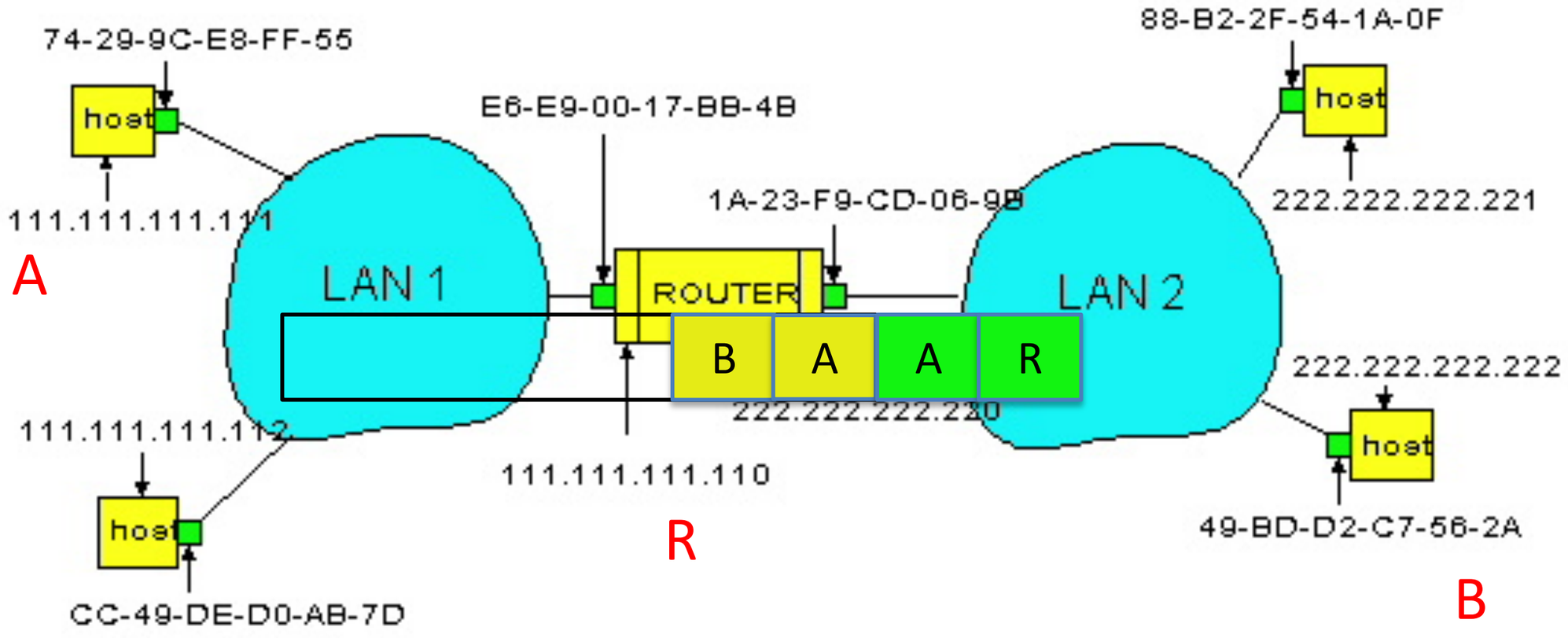
Host A Sends Packet Through R

- 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



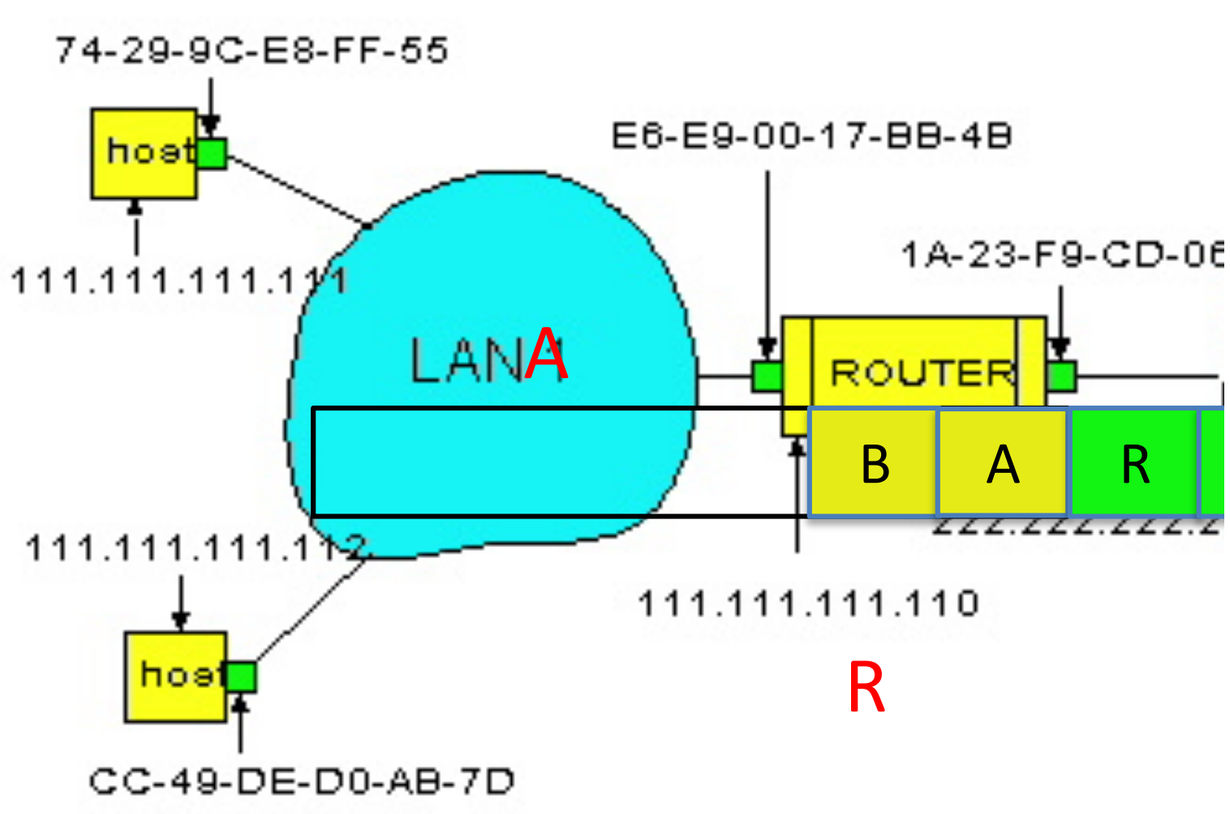
R Decides how to Forward Packet

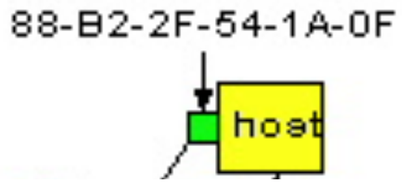
- Router R's adaptor receives the packet
  - R extracts the IP packet from the Ethernet frame
  - R sees the IP packet is destined to 222.222.222.222
- Router R consults its forwarding table
  - Packet matches 222.222.222.0/24 via other adaptor



R Sends Packet to B

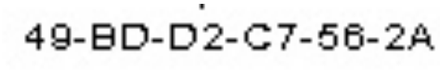
- Router R's learns the MAC address of host B
  - ARP request: broadcast request for 222.222.222.222
  - ARP response: B responds with 49-BD-D2-C7-52A
- Router R encapsulates the packet and sends to B





R broadcasts: Who is 222.222.22.222.222?

B replies to R
Me! I'm 222.22.222.222 on 49-BD-D2-C7-56-2A



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 the (naughty) packets are not its MAC address

Key Ideas in Both ARP and DHCP

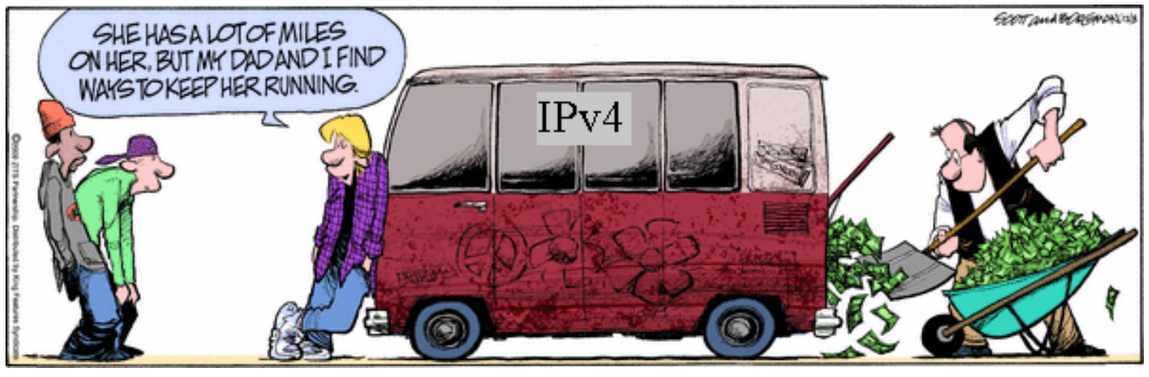
- Broadcasting: Can use broadcast to make contact
  - Scalable because of limited size
- Caching: remember the past for a while
  - Store the information you learn to reduce overhead
  - Remember your own address & other host's addresses
- Soft state: eventually forget the past
  - Associate a time-to-live 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

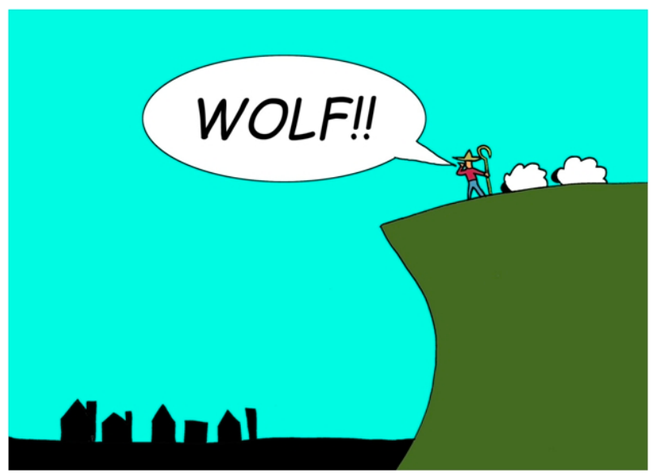
IPv6



- Motivated prematurely by address exhaustion
  - addresses are larger
  - packet headers are laid out differently
  - address management and configuration are completely different
  - some DNS behavior changes
  - some sockets code changes
  - *everybody now has a hard time parsing IP addresses*

- Steve Deering focused on simplifying IP
  - Got rid of all fields that were not absolutely necessary
  - “Spring Cleaning” for IP

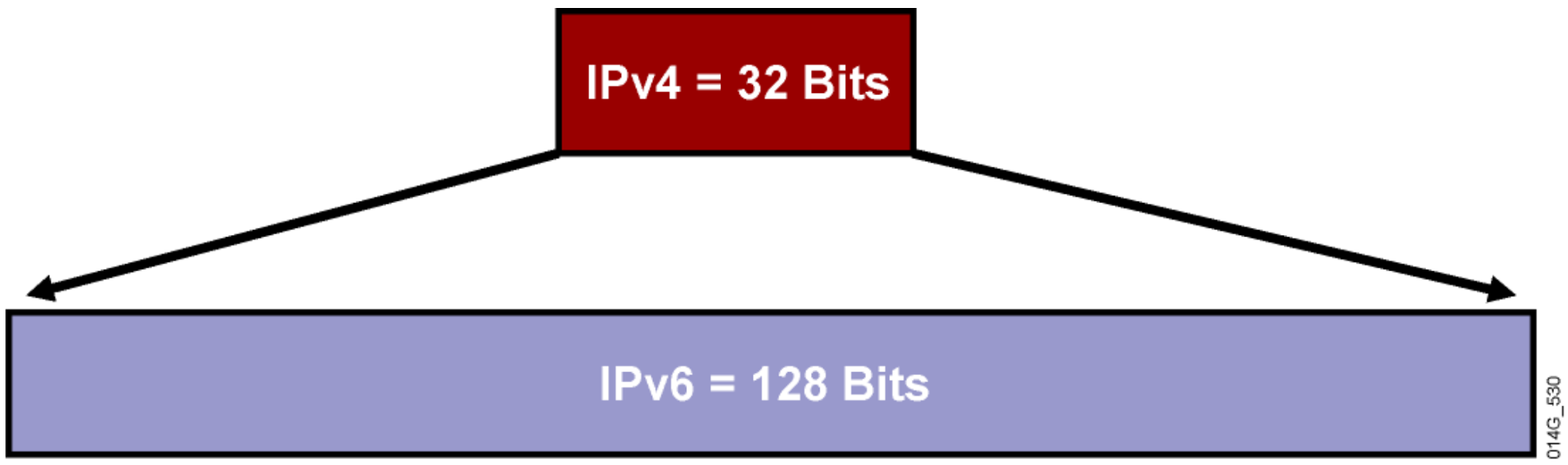
- Result is an elegant, if unambitious, protocol



IPv4	IPv6
Addresses are 32 bits (4 bytes) in length.	Addresses are 128 bits (16 bytes) in length
Address (A) resource records in DNS to map host names to IPv4 addresses.	Address (AAAA) resource records in DNS to map host names to IPv6 addresses.
Pointer (PTR) resource records in the IN-ADDR.ARPA DNS domain to map IPv4 addresses to host names.	Pointer (PTR) resource records in the IP6.ARPA DNS domain to map IPv6 addresses to host names.
IPSec is optional and should be supported externally	IPSec support is not optional
Header does not identify packet flow for QoS handling by routers	Header contains Flow Label field, which Identifies packet flow for QoS handling by router.
Both routers and the sending host fragment packets.	Routers do not support packet fragmentation. Sending host fragments packets
Header includes a checksum.	Header does not include a checksum.
Header includes options.	Optional data is supported as extension headers.
ARP uses broadcast ARP request to resolve IP to MAC/Hardware address.	Multicast Neighbor Solicitation messages resolve IP addresses to MAC addresses.
Internet Group Management Protocol (IGMP) manages membership in local subnet groups.	Multicast Listener Discovery (MLD) messages manage membership in local subnet groups.
Broadcast addresses are used to send traffic to all nodes on a subnet.	IPv6 uses a link-local scope all-nodes multicast address.
Configured either manually or through DHCP.	Does not require manual configuration or DHCP.
Must support a 576-byte packet size (possibly fragmented).	Must support a 1280-byte packet size (without fragmentation).

Larger Address Space

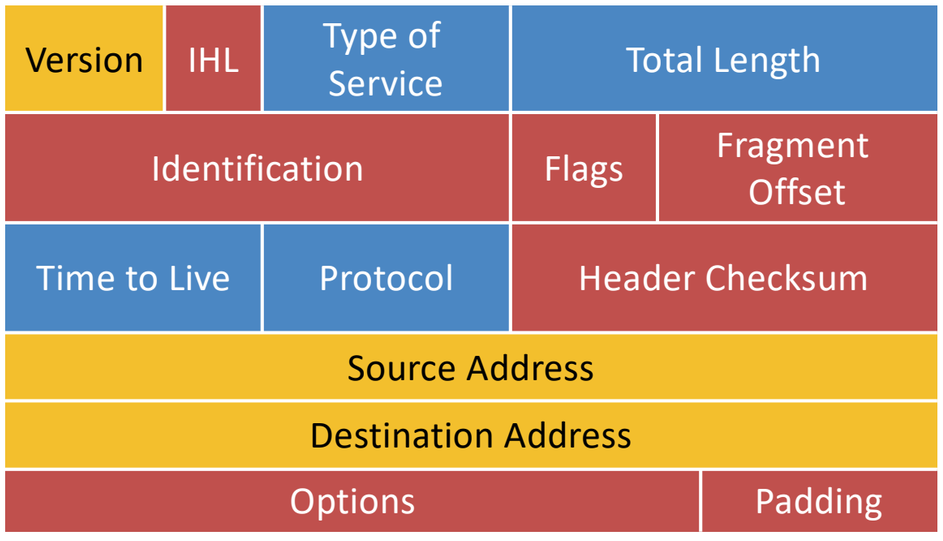
- IPv4 = 4,294,967,295 addresses
- IPv6 = 340,282,366,920,938,463,374,607,432,768,211,456 addresses
- 4x in number of bits translates to **huge** increase in address space!



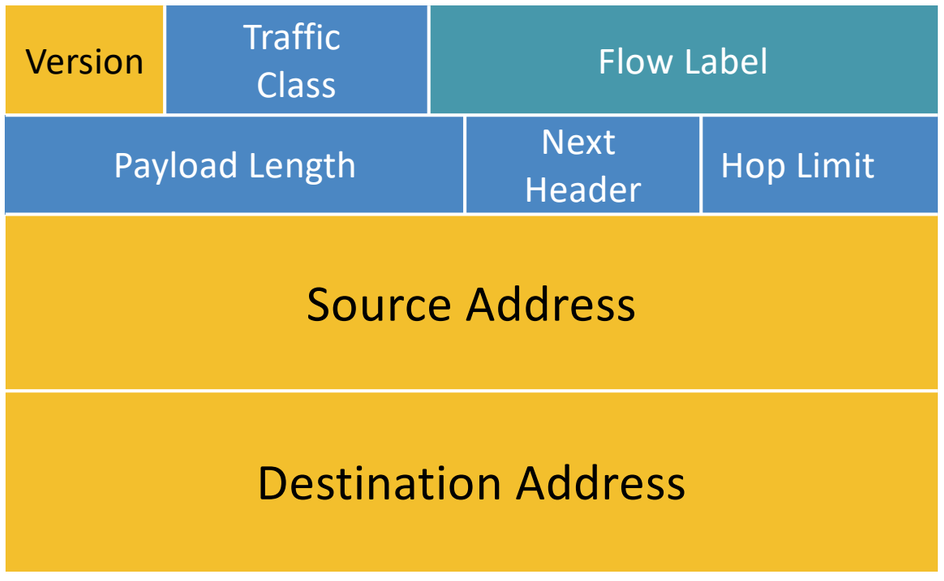
Other Significant Protocol Changes - 1

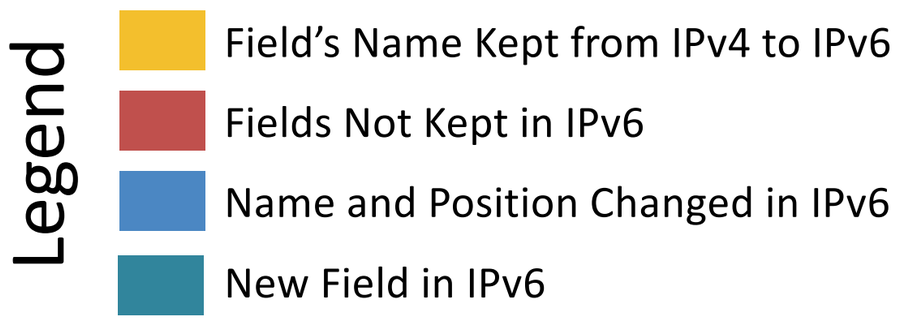
- Increased minimum MTU from 576 to 1280
- No enroute fragmentation… fragmentation only at source
- Header changes (20bytes to 40bytes)
- Replace broadcast with multicast

IPv4



IPv6





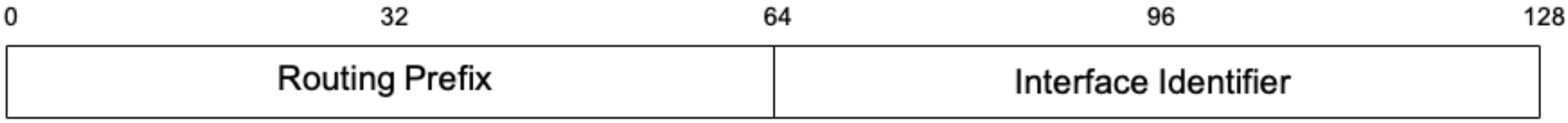
Other Significant Protocol Changes - 2

operation is intended to be simpler within the network:

- no *in-network* fragmentation
- no checksums in IPv6 header
- UDP checksum required (wasn't in IPv4) rfc6936: **No more zero**
- optional state carried in *extension headers*
  - Extension headers notionally replace IP options
  - Each extension header indicates the type of the *following* header, so they can be chained
  - The final 'next header' either indicates there is no 'next', or escapes into an transport-layer header (e.g., TCP)

IPv6 Basic Address Structure

IPv6 addresses are split into two primary parts:



- 64 bits is dedicated to an addressable interface (equivalent to the host, if it only has one interface)
- The network prefix allocated to a network by a registry can be up to 64-bits long
- An allocation of a /64 (i.e. a 64-bit network prefix) allows *one* subnet (it cannot be subdivided)
- A /63 allows two subnets; a /62 offers four, etc. /48s are common for older allocations (RFC 3177, obsoleted by RFC 6177).
- Longest-prefix matching operates as in IPv4.

IPv6 Address Representation (quick)

IPv6 addresses represented as eight 16-bit blocks (4 hex chars) separated by colons:

- `2001:4998:000c:0a06:0000:0000:0002:4011`

But we can condense the representation by removing leading zeros in each block:

- `2001:4998:c:a06:0:0:2:4011`

And by reducing the consecutive block of zeros to a "::"
(this double colon rule can only be applied once)

- `2001:4998:c:a06::2:4011`

IPv6 Address Families

The address space is carved, like v4, into certain categories [1]:

host-local : localhost; **`::1`** is equivalent to **`127.0.0.1`**

link-local : not routed: **`fe80::/10`** is equivalent to **`169.254.0.0/16`**

site-local : not routed *globally*: **`fc00::/7`** is equivalent to **`192.168.0.0/16`** or **`10.0.0.0/8`**

global unicast : **`2000::/3`** is basically any v4 address not reserved in some other way

multicast : **`ff00::/8`** is equivalent to **`224.0.0.0/4`**

[1] http://www.ripe.net/lir-services/new-lir/ipv6_reference_card.pdf

Problem with /64 Subnets

- Scanning a subnet becomes a DoS attack!
  - Creates IPv6 version of 2^{64} ARP entries in routers
  - Exhaust address-translation table space

- So now we have:

`ping6 ff02::1` All nodes in broadcast domain

`ping6 ff02::2` All routers in broadcast domain

- Solutions
  - RFC 6164 recommends use of /127 to protect router-router links
  - RFC 3756 suggest “clever cache management” to address more generally

Neighbour Discovery

- The Neighbour Discovery Protocol[2] specifies a set of ICMPv6 message types that allow hosts to discover other hosts or routing hardware on the network
  - neighbour solicitation
  - neighbour advertisement
  - router solicitation
  - router advertisement
  - redirect
- In short, a host can *solicit* neighbour (host) state to determine the layer-2 address of a host *or* to check whether an address is in use
- or it can solicit router state to learn more about the network configuration
- In both cases, the solicit message is sent to a well-known multicast address

[2] http://tools.ietf.org/html/rfc4861

IPv6 Dynamic Address Assignment

We have the two halves of the IPv6 address: the network component and the host component. Those are derived in different ways.

Network (top 64 bits):

- Router Advertisements (RAs)

  Interface

Identifier (bottom 64 bits):

- Stateless, automatic: SLAAC
- Stateful, automatic: DHCPv6

SLAAC: overview

SLAAC is:

- ... intended to make network configuration easy without manual configuration *or even a DHCP server*
- ... an algorithm for hosts to automatically configure their network interfaces (set up addresses, learn routes) without intervention

SLAAC: overview

- When a host goes live or an interface comes up, the system wants to know more about its environment

- It *can* configure link-local addresses for its interfaces: it uses the interface identifier, the EUI-64

- It uses this to ask (solicit) router advertisements sooner than the next periodic announcements; ask the network for information

SLAAC: overview

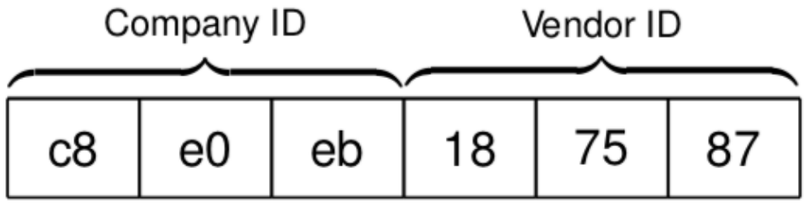
The algorithm (assuming one interface):

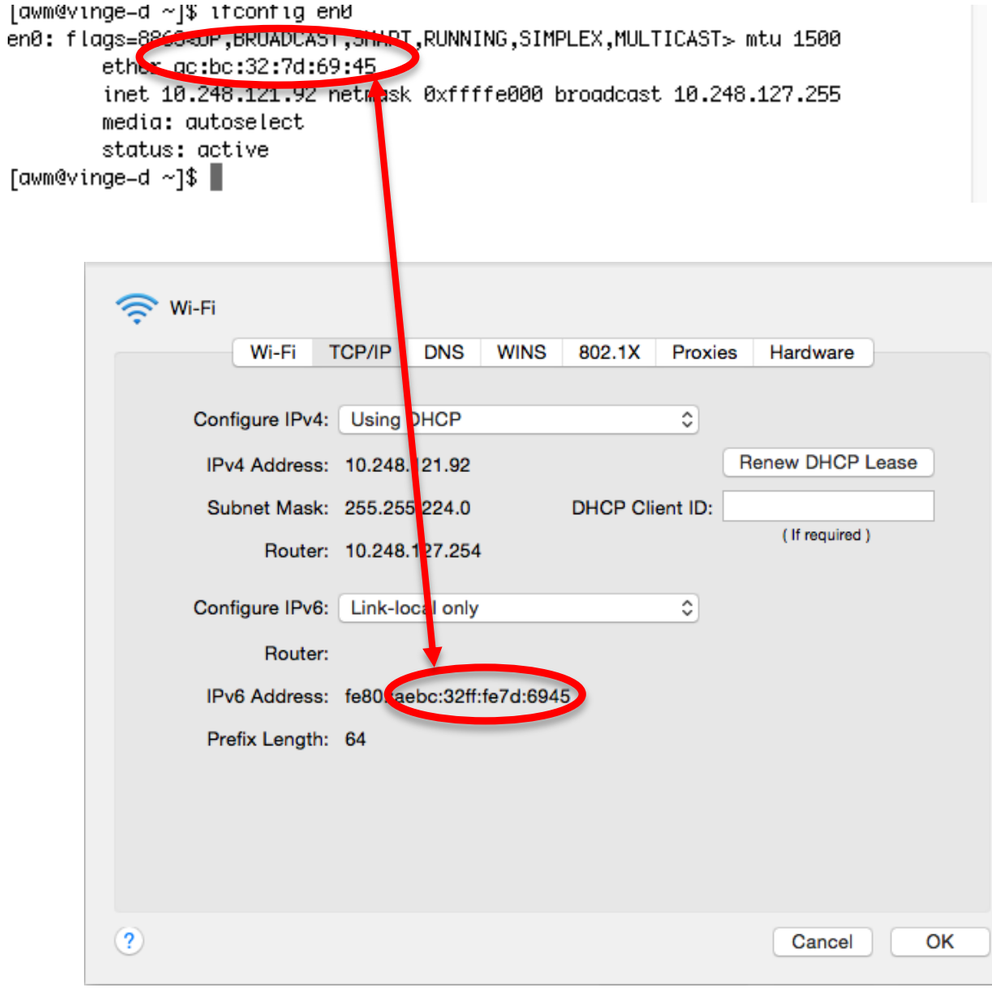
1. Generate potential link-local address
2. Ask the network (multicast[4]) if that address is in use: *neighbour solicitation*
3. Assuming no responses, assign to interface

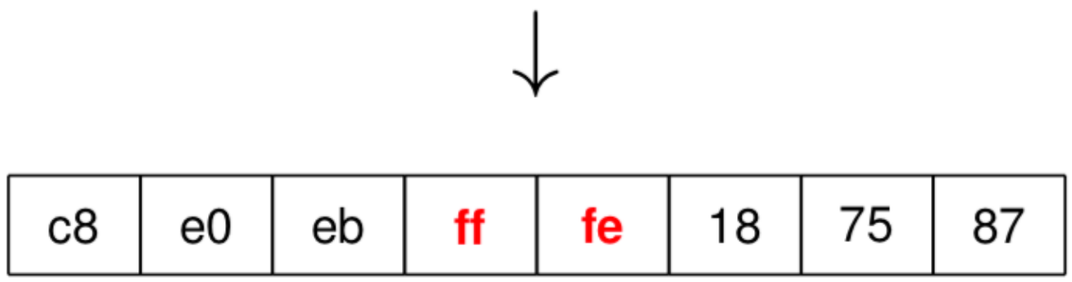
[4] https://tools.ietf.org/html/rfc2373

The EUI-64 Interface Identifier

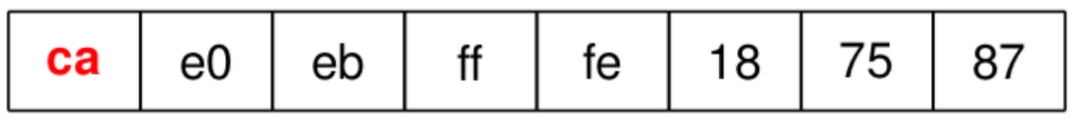
- IEEE 64-bit Extended Unique Identifier (EUI-64)[3]
- There are various techniques to derive a 64-bit value, but often times we derive from the 48-bit MAC address







the seventh bit from the left, or the universal/local (U/L) bit, needs to be inverted, 0 = local admin 1 = universal admin



[3] http://tools.ietf.org/html/rfc2373

SLAAC: overview; Router Solicitation

Then,

- Once the host has a unique *link-local* address, it can send packets to anything else sharing that link substrate
  ... but the host doesn't yet know any routers, or public routes
  ... bootstrap: routers listen to a well-known multicast address

4. host asks the network (multicast) for router information: *router solicitation*

5. responses from the routers are sent directly (unicast) to the host that sent the router solicitation

6. the responses *may* indicate that the host should do more (e.g., use DHCP to get DNS information)

Router Advertisement

Without solicitation, router advertisements are generated intermittently by routing hardware.

Router Advertisements:

- nodes that forward traffic periodically advertise themselves to the network
- periodicity and expiry of the advertisement are configurable

Router Advertisement (RA), among other things, tells a host where to derive its network state with two flags: M(anaged) and O(ther info):

- M: “Managed Address Configuration”, which means: use DHCPv6 to find your host address (and ignore option O)
- O: Other information is available via DHCPv6, such as DNS configuration

Uh-oh

What problem(s) arises from totally decentralised address configuration?

Concerns that arise from using an EUI-64:

- Privacy: SLAAC interface identifiers don't change over time, so a host can be identified across networks

- Security: embedding a MAC address into an IPv6 address will carry that vendor's ID(s)[5], a possible threat vector

[5]http://standards.ieee.org/develop/regauth/oui/public.html

Address Configuration: SLAAC Privacy Addresses

Privacy extensions for SLAAC[6]

- temporary addresses for initiating outgoing sessions
- generate one temporary address per prefix
- when they expire, they are not used for new sessions, but can continue to be used for existing sessions
- the addresses should appear random, such that they are difficult to predict
- lifetime is configurable; this OSX machine sets an 86,400s timer (1 day)

[6] https://tools.ietf.org/html/rfc4941

Address Configuration: SLAAC Privacy Addresses

The algorithm:

- Assume: a stored 64-bit input value from previous iterations, or a pseudo-randomly generated value

1. take that input value and append it to the EUI-64
2. compute the MD5 message digest of that value
3. set bit 6 to zero
4. compare the leftmost 64-bits against a list of reserved interface identifiers and those already assigned to an address on the local device. If the value is unacceptable, re-run using the rightmost 64 bits of the result instead of the historic input value in step 1
5. use the leftmost 64-bits as the randomised interface identifier
6. store the rightmost 64-bits as the history value to be used in the next iteration of the algorithm

IPv6: why has the transition taken so long?

IPv4 and IPv6 are not compatible:

- different packet formats
- different addressing schemes

as the Internet has grown bigger and accumulated many IPv4-only services, transition has proven ... Tricky

Incentive issues

Virgin Media policy in 2010

….When IPV6 is rolled out across the whole of the Internet then a lot of the ISP's will roll out IPV6, ….

IPv6: why has the transition taken so long?

- IPv4 has/had the momentum
  - ... which led to CIDR
  - ... and encouraged RFC1918 space and NAT

- IPv4 NAT was covered earlier in this topic (reminder)
  - your ISP hands you only one IPv4 address
  - you share that across multiple devices in your household
  - The NAT handles all the translation between internal ("private") and external ("public") space

Transition tech: outline

- Tunnelling
- dual-stacked services, and happy eyeballs
- DNS64 and NAT64[8]
- 464XLAT
- DNS behaviour

[8] https://tools.ietf.org/html/rfc6146

Transition tech: outline

- Tunnelling

HURRICANE ELECTRIC
INTERNET SERVICES

Hurricane Electric Free IPv6 Tunnel Broker

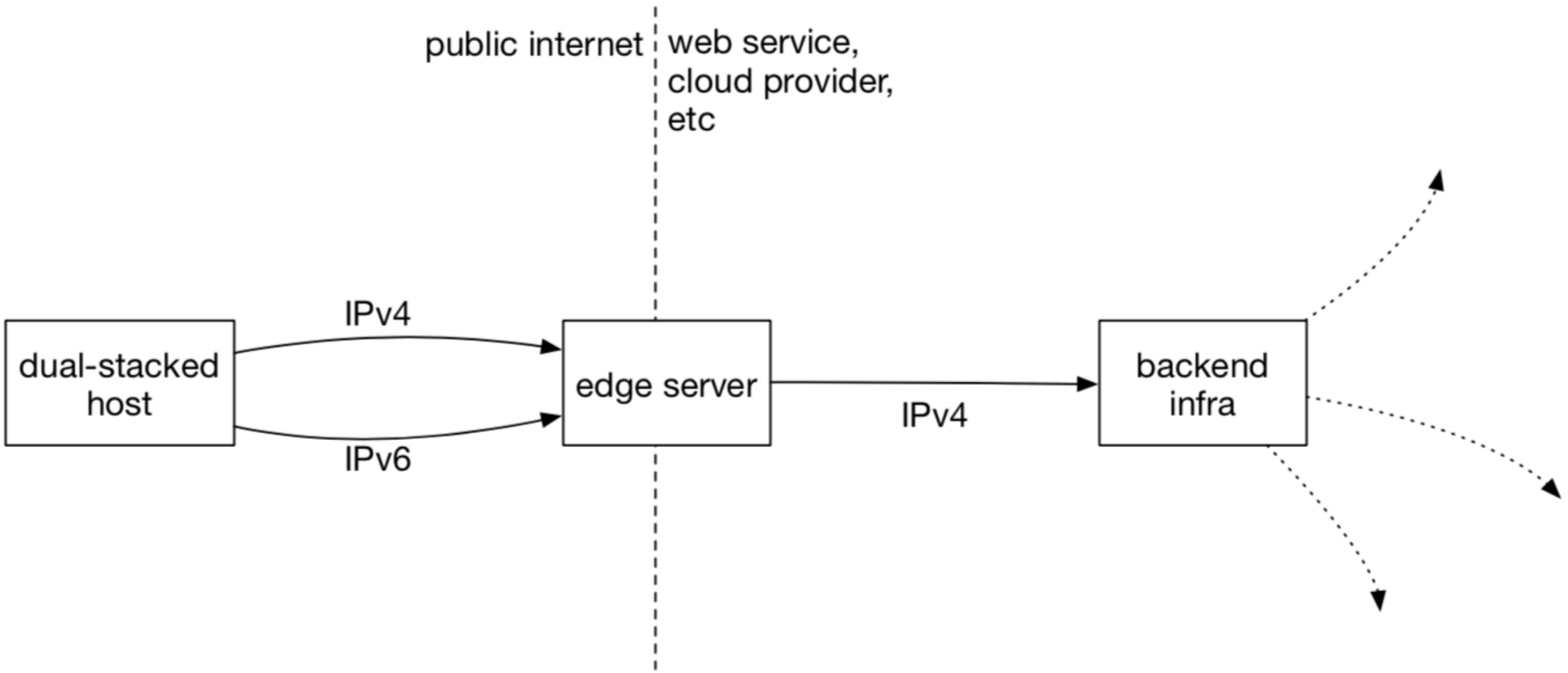
IPv6 Tunnel Broker

Think of it as an IPv6 VPN service; which is essentially what it is

[8] https://tools.ietf.org/html/rfc6146

Dual-Stack Services: Common Deployment

It’s common for web services to play conservatively: dual-stack your edge services (e.g., load balancers), leaving some legacy infrastructure for later:



Dual-Stack Services: Common Deployment

Aim is to reduce the pain:

- You can dual-stack the edge hosts, and carry state in, say, HTTP headers indicating the user's IP address (common over v4 anyway)
- You can dual-stack the backend opportunistically, over a longer period of time
- You use DNS to enable/disable the v6 side last (if there is no AAAA record in DNS, no real users will connect to the IPv6 infrastructure

Happy Eyeballs and DNS

- The introduction of IPv6 carried with it an obligation that applications attempt to use IPv6 before falling back to IPv4.
- What happens though if you try to connect to a host which doesn't exist?[9]
- But the presence of IPv6 modifies the behaviour of DNS responses and response preference[10]

[9] https://tools.ietf.org/html/rfc5461
[10] https://tools.ietf.org/html/rfc3484

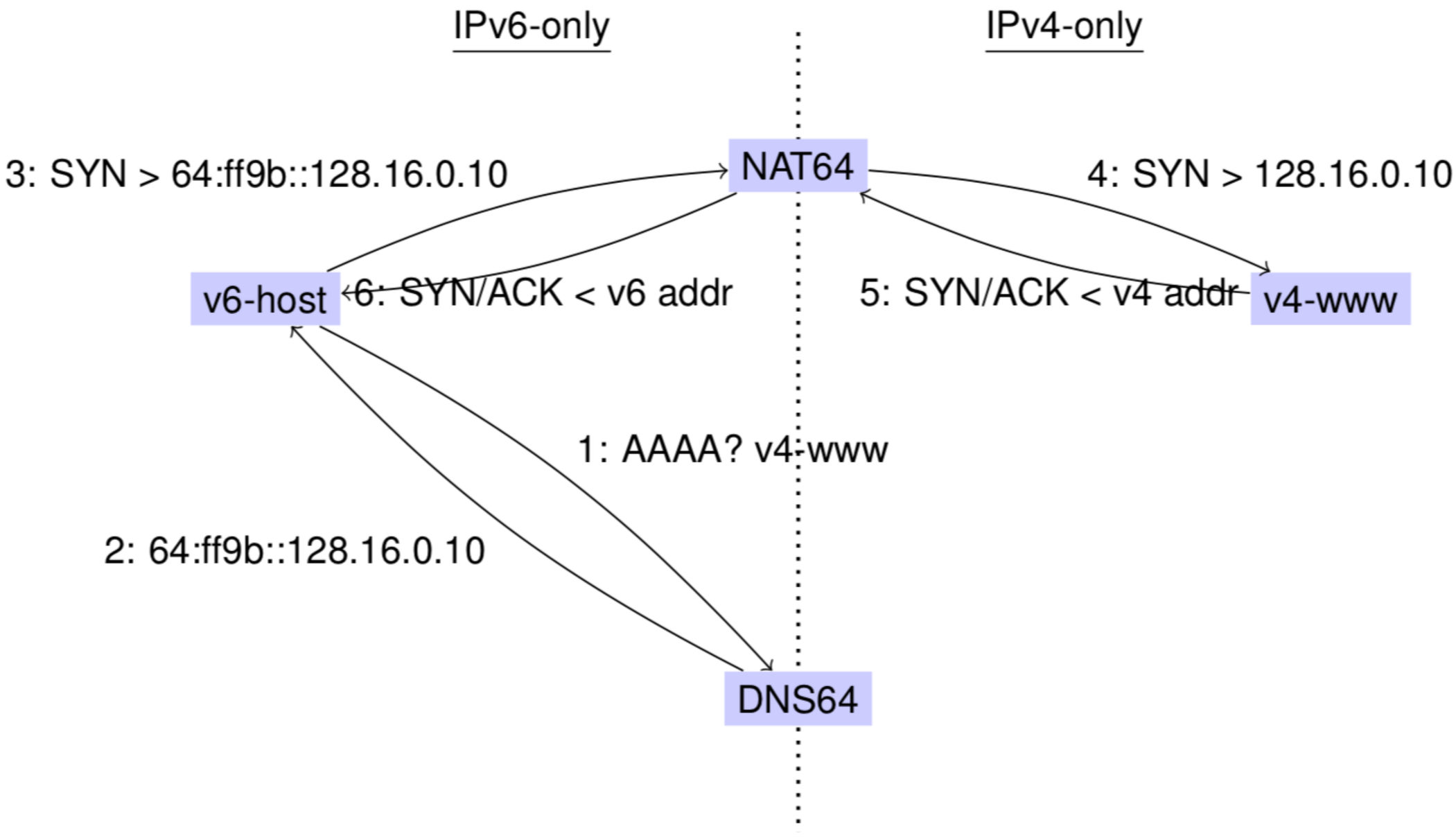
Happy Eyeballs

- Happy Eyeballs[11] was the proposed solution
  - the eyeballs in question are yours, or mine, or whoever is sitting in front of their browser getting mad that things are unresponsive

- Modifies application behaviour

[11] https://tools.ietf.org/html/rfc8305

DNS64 & NAT64


IPv6-only
IPv4-only
3: SYN > 64:ff9b::128.16.0.10
NAT64
4: SYN > 128.16.0.10
v6-host
6: SYN/ACK < v6 addr
5: SYN/ACK < v4 addr
v4-www
1: AAAA? v4-www
2: 64:ff9b::128.16.0.10
DNS64

464XLAT

- Problem: IPv6-only to the host, but an IPv4-only app trying to access an IPv4-only service
  - Some *applications* do not understand IPv6, so having an IPv6 address doesn't help
  - 464XLAT[12] solves this problem
  - In essence, DNS64 + NAT64 + a shim layer on the host itself to offer IPv4 addresses to apps

[12]https://tools.ietf.org/html/rfc6877

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?

Summary Network Layer

- understand principles behind network layer services:
  - network layer service models
  - forwarding versus routing (versus switching)
  - how a switch & router works
  - routing (path selection)
  - IPv6
- Algorithms
  - Two routing approaches (LS vs DV)
  - One of these in detail (LS)
  - ARP
- Other Core ideas
  - Caching, soft-state, broadcast
  - Fate-sharing in practice….