CS 43: Computer Networks

18: Network Layer, IP

November 14, 2024

SWARTHMORE COLLEGE

The Network Layer!

Application: the application (e.g., the Web, Email)

Transport: end-to-end connections, reliability

Network: routing

Link (data-link): framing, error detection

Physical: 1’s and 0’s/bits across a medium (copper, the air, fiber)

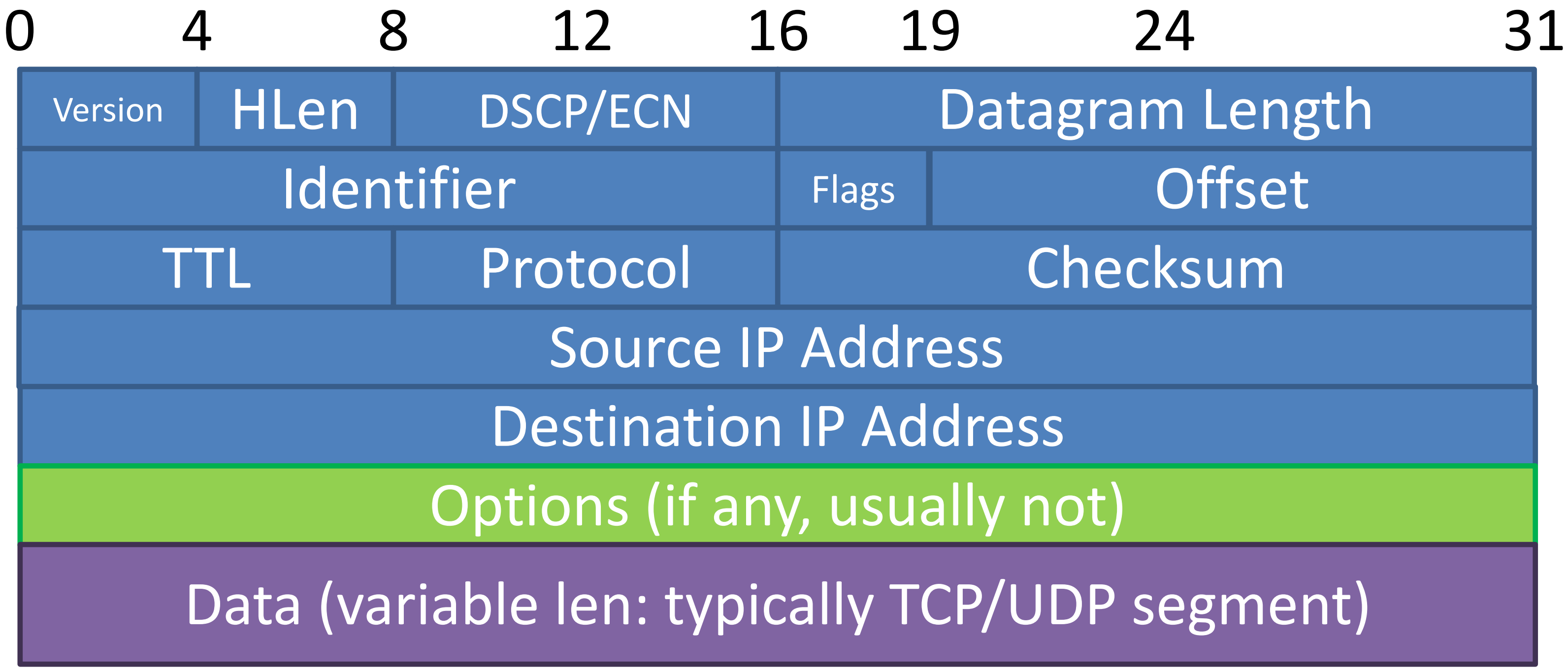
Network Layer Functions

- Forwarding: move packets from router’s input to appropriate router output
  - Look up in a table
- Routing: determine route taken by packets from source to destination.
  - Populating the table

IP Datagrams

how much overhead?

- 20 bytes of TCP
- 20 bytes of IP
- = 40 bytes + app layer overhead



0 4 8 12 16 19 24 31

Version | HLen | DSCP/ECN | Datagram Length

Identifier | Flags | Offset

TTL | Protocol | Checksum

Source IP Address

Destination IP Address

Options (if any, usually not)

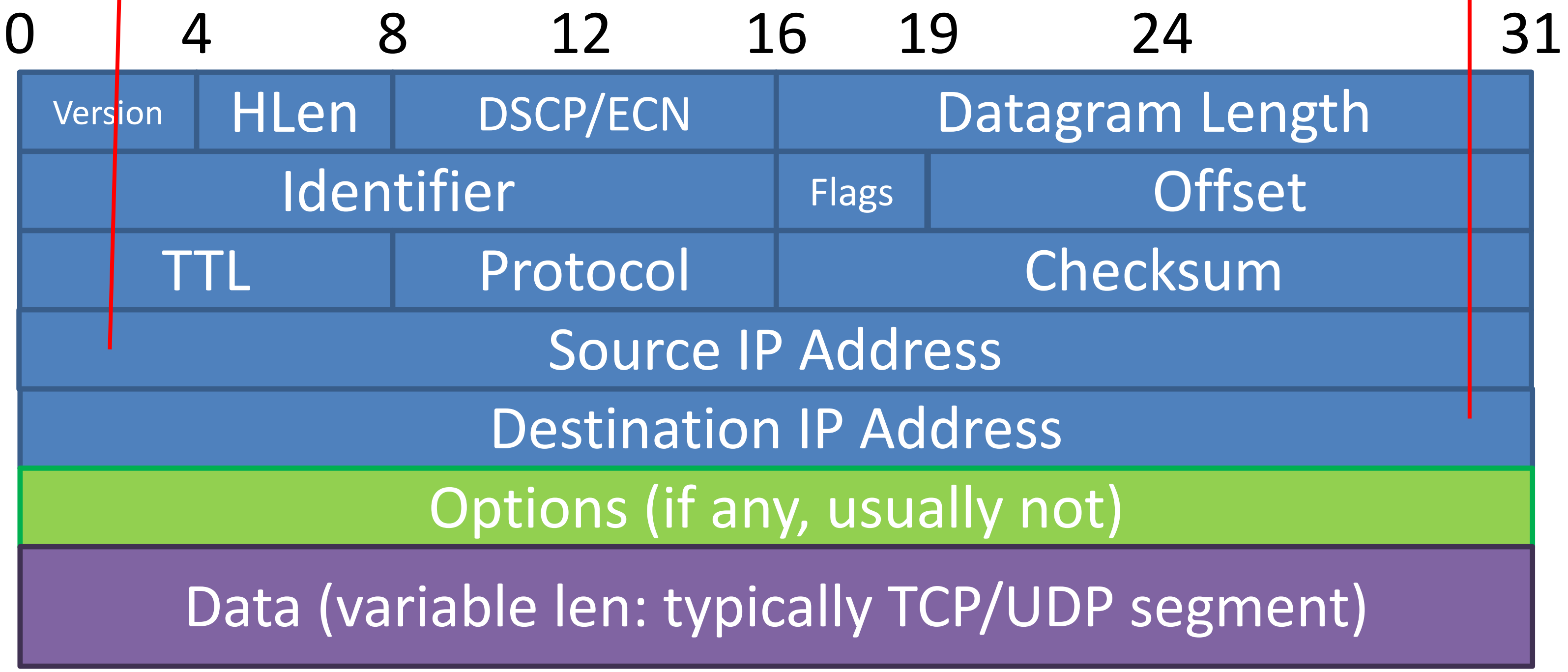
Data (variable len: typically TCP/UDP segment)

IP Datagrams

Source endpoint.

Addresses must be unique on the network!

Final destination endpoint.



0 4 8 12 16 19 24 31

Version | HLen | DSCP/ECN | Datagram Length

Identifier | Flags | Offset

TTL | Protocol | Checksum

Source IP Address

Destination IP Address

Options (if any, usually not)

Data (variable len: typically TCP/UDP segment)

What's in a name?

- Host name: web.cs.swarthmore.edu
  - Domain: registrar for each top-level domain (e.g., .edu)
  - Host name: local administrator assigns to each host

- IP addresses: 130.58.68.164
  - Prefixes: ICANN, regional Internet registries, and ISPs
  - Hosts: static configuration, or dynamic using DHCP

- MAC addresses: D8:D3:85:94:5F:1E
  - OIDs: assigned to vendors by the IEEE
  - Adapters: assigned by the vendor from its block

IP Addressing

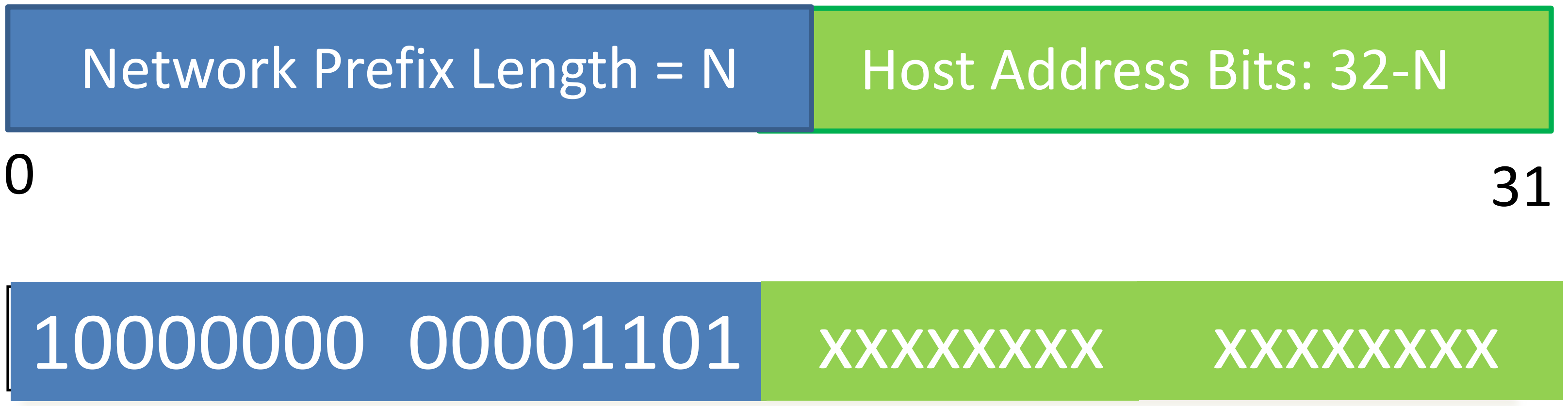
- IP: 32-bit addresses
  - Usually written in dotted notation, e.g. 192.168.21.76
  - Each number is a byte
  - Stored in Big Endian order (network byte order)

0 8 16 24 31

Decimal	192	168	21	76
Hex	C0	A8	15	4C
Binary	11000000	10101000	00010101	01001100

IP Prefixes

- Written in IP address/length notation
- Address is the lowest address in the allocated block. Length is prefix in bits.
- E.g. 128.13.0.0/16 is 128.13.0.0 to 128.13.255.255
  Read as: ”128.13.0.0 slash 16” prefix.



Network Prefix Length = N

Host Address Bits: 32-N

0

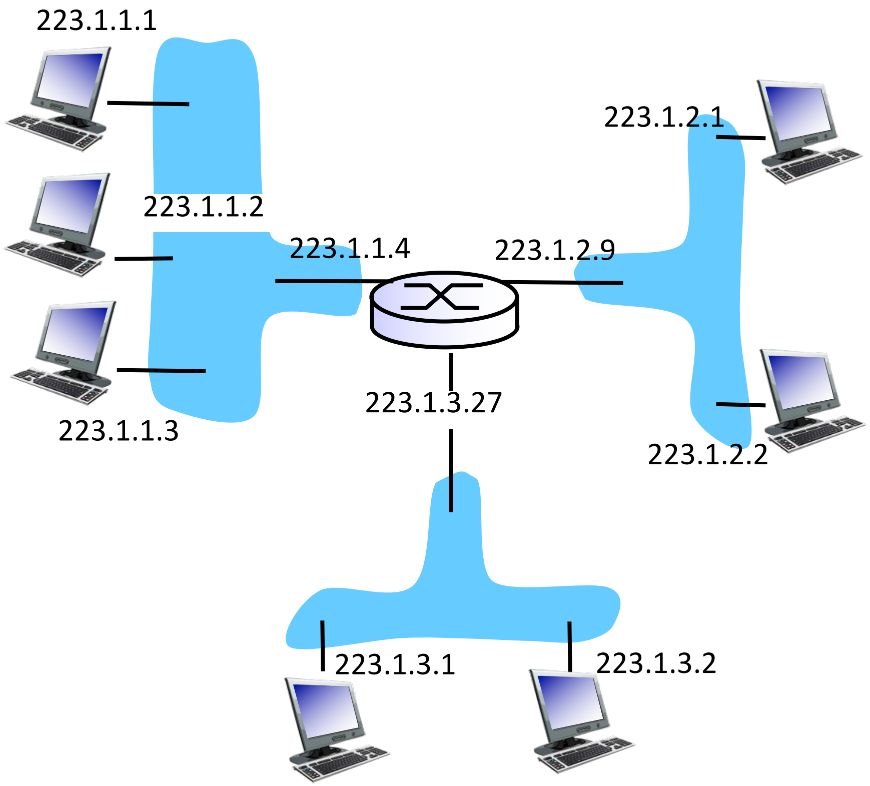
31

10000000 00001101 xxxxxxxx xxxxxxxx

Subnets

- IP address:
  - subnet part - high order bits
  - host part - low order bits

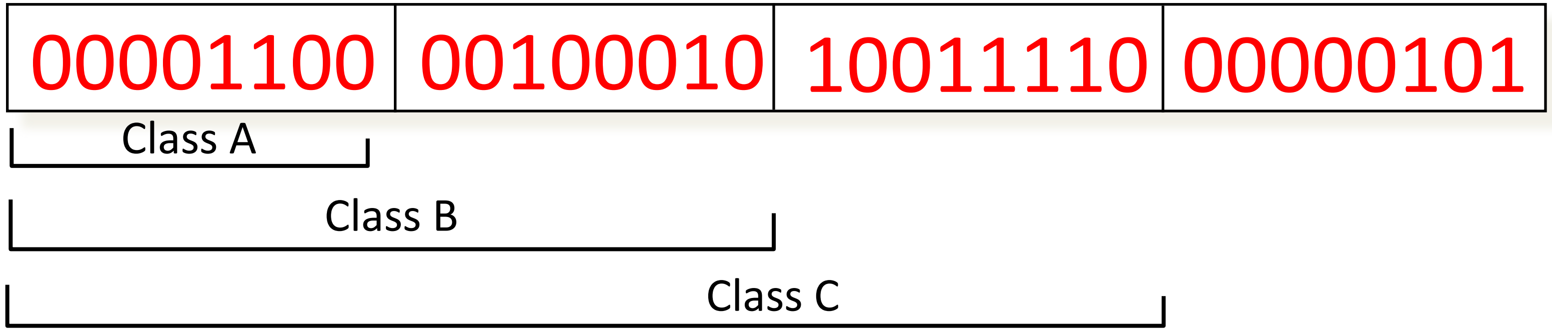
- what's a subnet?
  - device interfaces with same subnet part of IP address
  - can physically reach each other without intervening router
  - On the same link layer



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

Who gets an address? How many?

- Classful Addressing
  - Class A: 8-bit prefix, 24 bits for hosts (16,777,216)
  - Class B: 16-bit prefix, 16 bits for hosts (65,536)
  - Class C: 24-bit prefix, 8 bits for hosts (256)



00001100	00100010	10011110	00000101

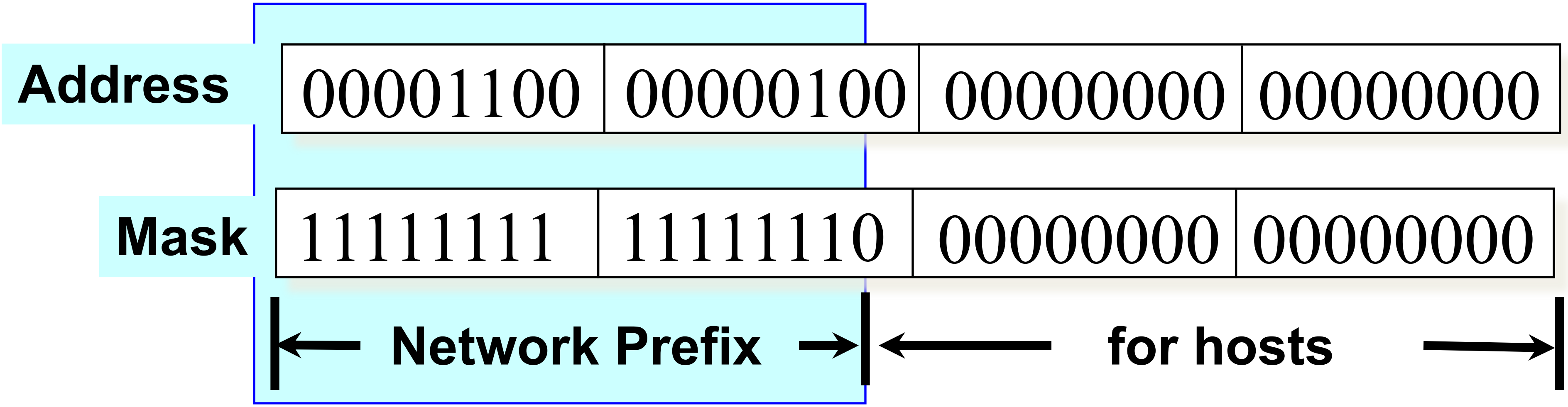
Class A

Class B

Class C

Classless Inter-Domain Routing (CIDR)

IP Address : 12.4.0.0   IP Mask: 255.254.0.0

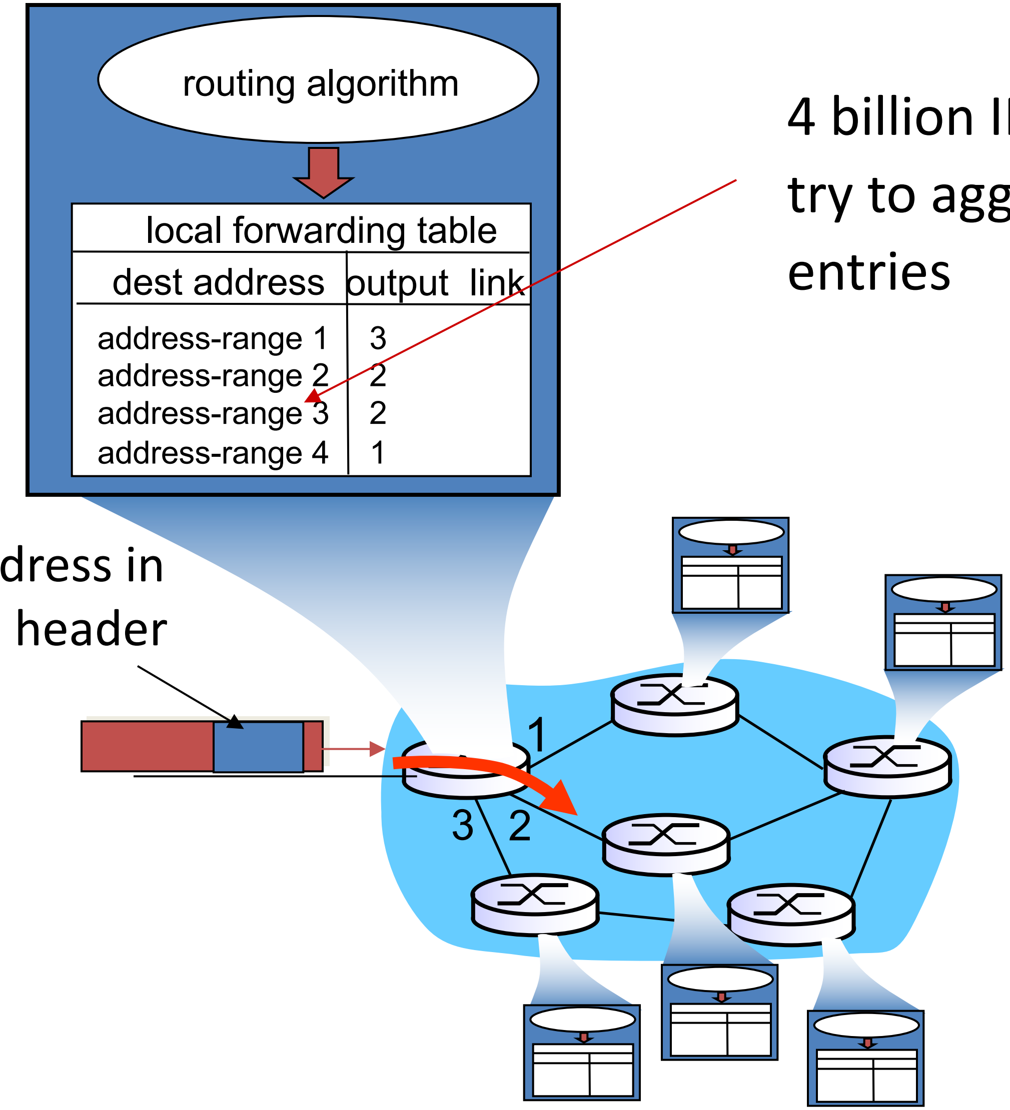


Address	00001100	00000100	00000000	00000000
Mask	11111111	11111110	00000000	00000000

Written as 12.4.0.0/15

Use two 32-bit numbers to represent a network.
Network number = IP address + Mask

Datagram forwarding table



local forwarding table

dest address	output link
address-range 1	3
address-range 2	2
address-range 3	2
address-range 4	1

4 billion IP addresses, try to aggregate table entries

IP destination address in arriving packet’s header

Routers exchange state (we'll save the what and when for later). They decide, for each destination, how to get there, and build a lookup structure for their forwarding table. What should they build?

A. A list – scan for the destination.

B. A hash table – look up the destination.

C. A tree – Follow branches that lead to the destination.

D. Some other software structure.

E. We can't do this in software, we need special hardware.

Look-up Algorithm

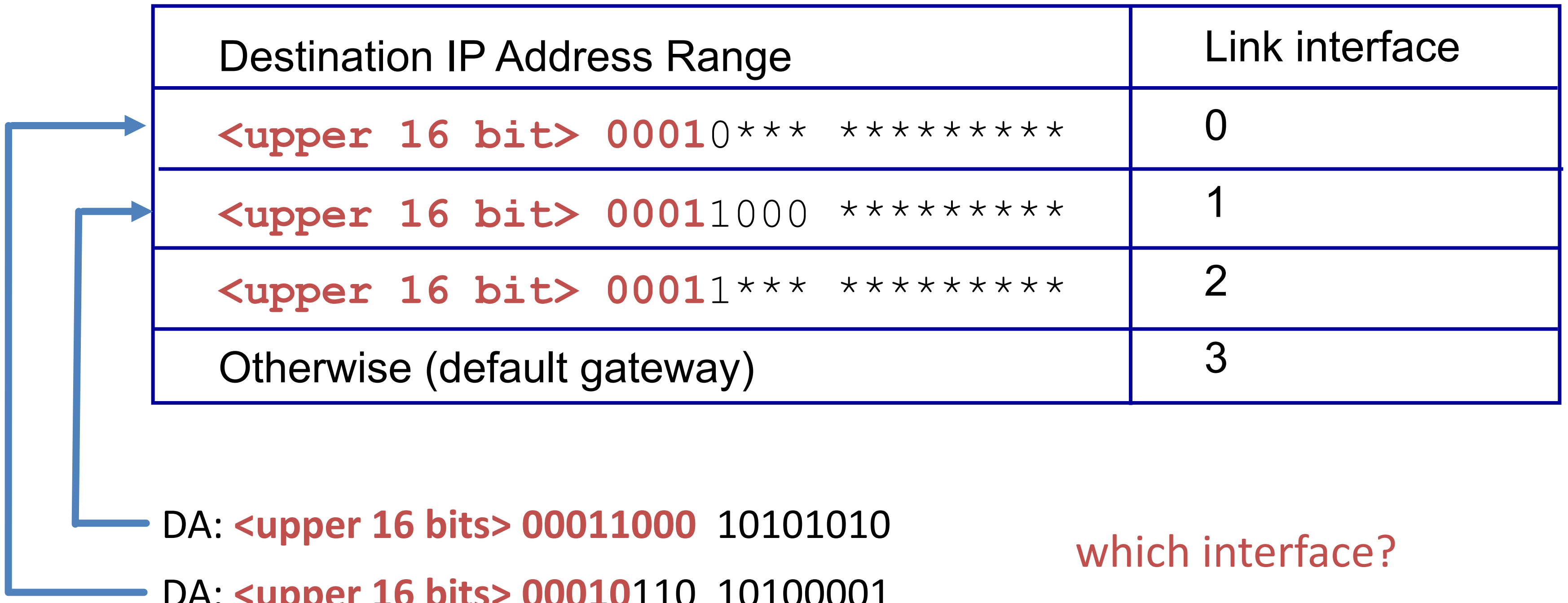
- Protocol: ATM (Virtual Circuits), Ethernet (Flat addresses)
  - Mechanism: Exact Match
  - Techniques: Direct lookup, Hash Tables, Binary Trees
- Protocol: IPv4, IPv6
  - Mechanism: Longest Prefix Match
  - Techniques: Prefix Trees, TCAM (Ternary Content Addressable Memories)

Datagram forwarding table

Destination Address Range	Link Interface
11001000 00010111 00010000 00000000 through 11001000 00010111 00010111 11111111	0
11001000 00010111 00011000 00000000 through 11001000 00010111 00011000 11111111	1
11001000 00010111 00011001 00000000 through 11001000 00010111 00011111 11111111	2
Otherwise (default gateway)	3

Longest prefix matching

In a forwarding table entry, use the longest address prefix that matches destination address.



Destination IP Address Range	Link interface
<upper 16 bit> 00010*** *********	0
<upper 16 bit> 00011000 *********	1
<upper 16 bit> 00011*** *********	2
Otherwise (default gateway)	3

DA: **<upper 16 bits> 00011000** 10101010

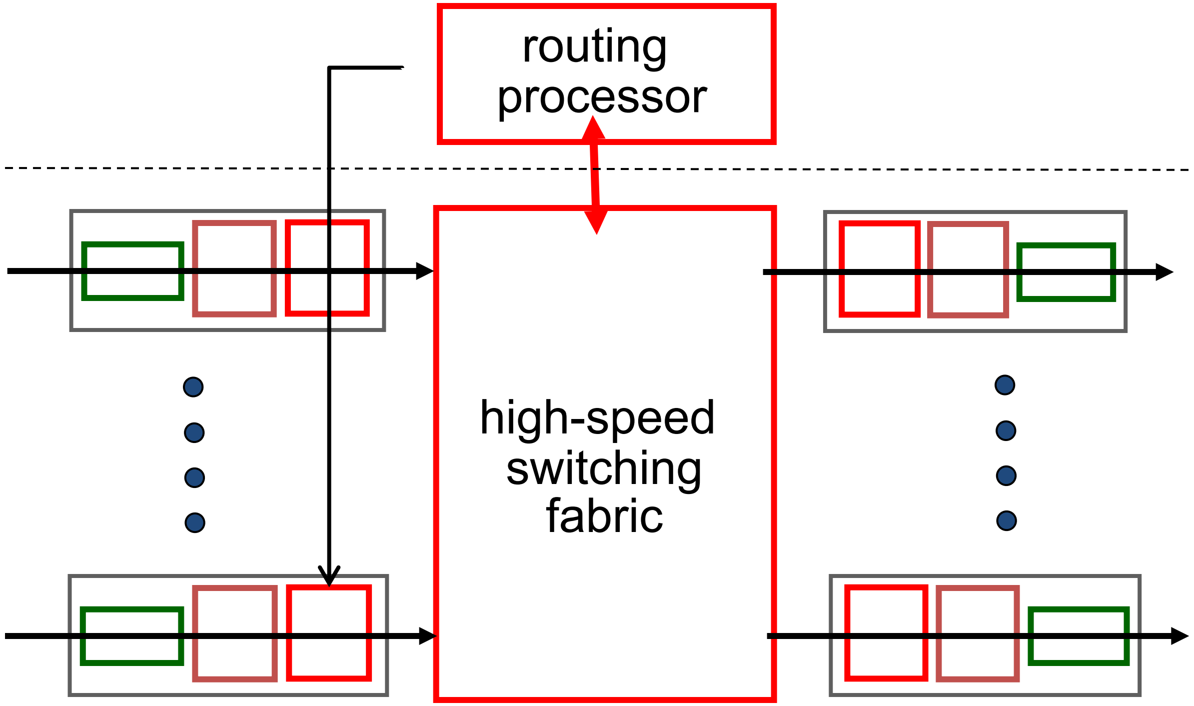
DA: **<upper 16 bits> 00010**110 10100001

which interface?

Router architecture overview

- high-level view of generic router architecture:

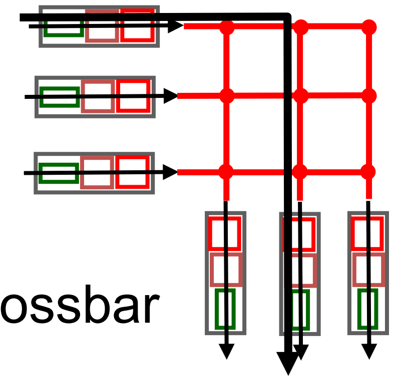
routing, management
control plane (software)
operates in millisecond
time frame



routing
processor

high-speed
switching
fabric

forwarding data plane
(hardware) operates
in nanosecond
timeframe

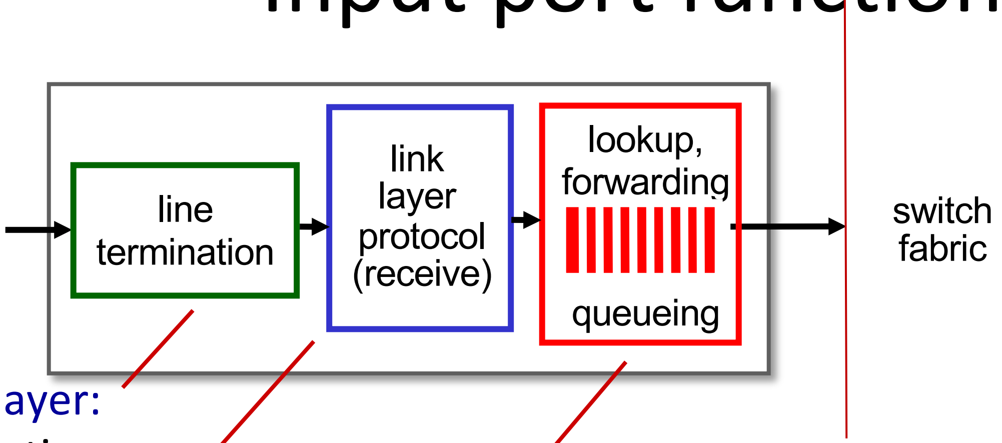


crossbar

router input ports
(different from TCP ports!!)
these are physical inputs/outputs to the router

router output ports

Input port functions



physical layer:
bit-level reception

link layer:
e.g., Ethernet
(chapter 6)

decentralized switching:

- using header field values, lookup output port using forwarding table in input port memory *("match plus action")*

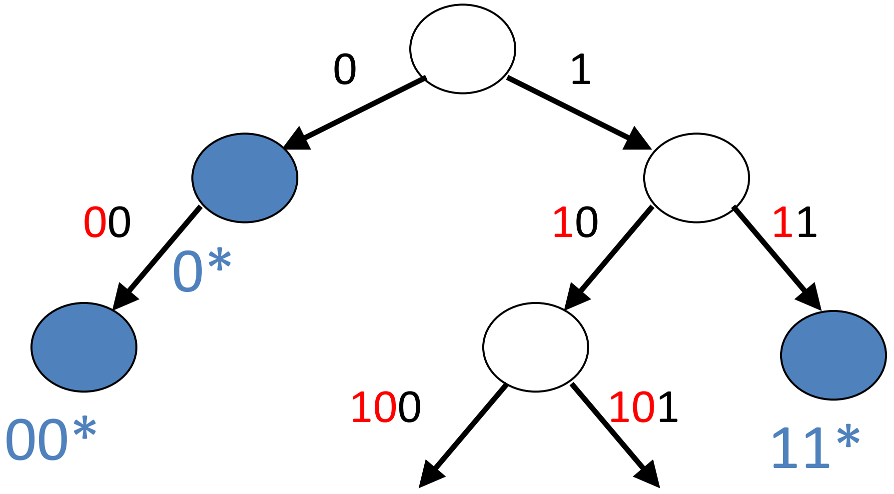
Longest prefix matching

In a forwarding table entry, use the longest address prefix that matches destination address.

Destination IP Address Range	Link interface
`<upper 16 bit> 00010*** *********`	0
`<upper 16 bit> 00011000 *********`	1
`<upper 16 bit> 00011*** *********`	2
Otherwise (default gateway)	3

Binary Prefix Tree

- Store the prefixes as a tree
  - Prefixes “spelled out” following a path from the root
  - One bit for each level of the tree
  - Some nodes correspond to valid prefixes
  - ... which have next-hop interfaces in a table
- When a packet arrives
  - Traverse the tree based on the destination address
  - Stop upon reaching the longest matching prefix

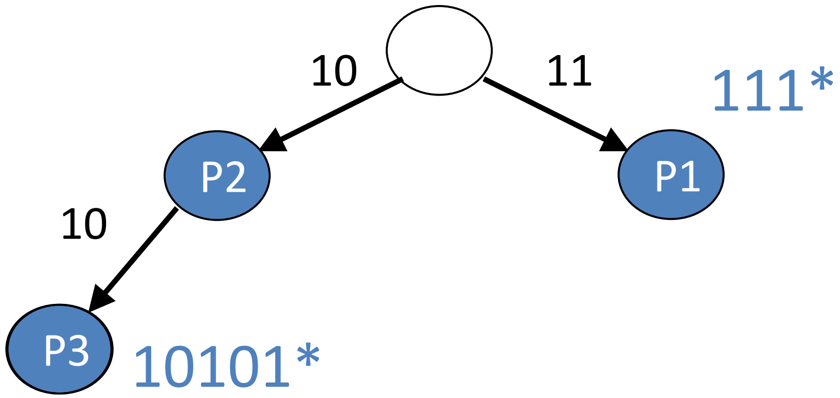


Prefix Range-1	0*	1
Prefix Range-2	00*	2
Prefix Range-3	11*	3

Depth = W
Degree = 2
Stride = 1 bit

Multi-bit Prefix Tree

- Store the prefixes as a tree: 4-ary tree
  - k bits for each level of the tree



Prefix Range-1	111*	1
Prefix Range-2	10*	2
Prefix Range-3	1010*	3
Prefix Range-4	10101*	4

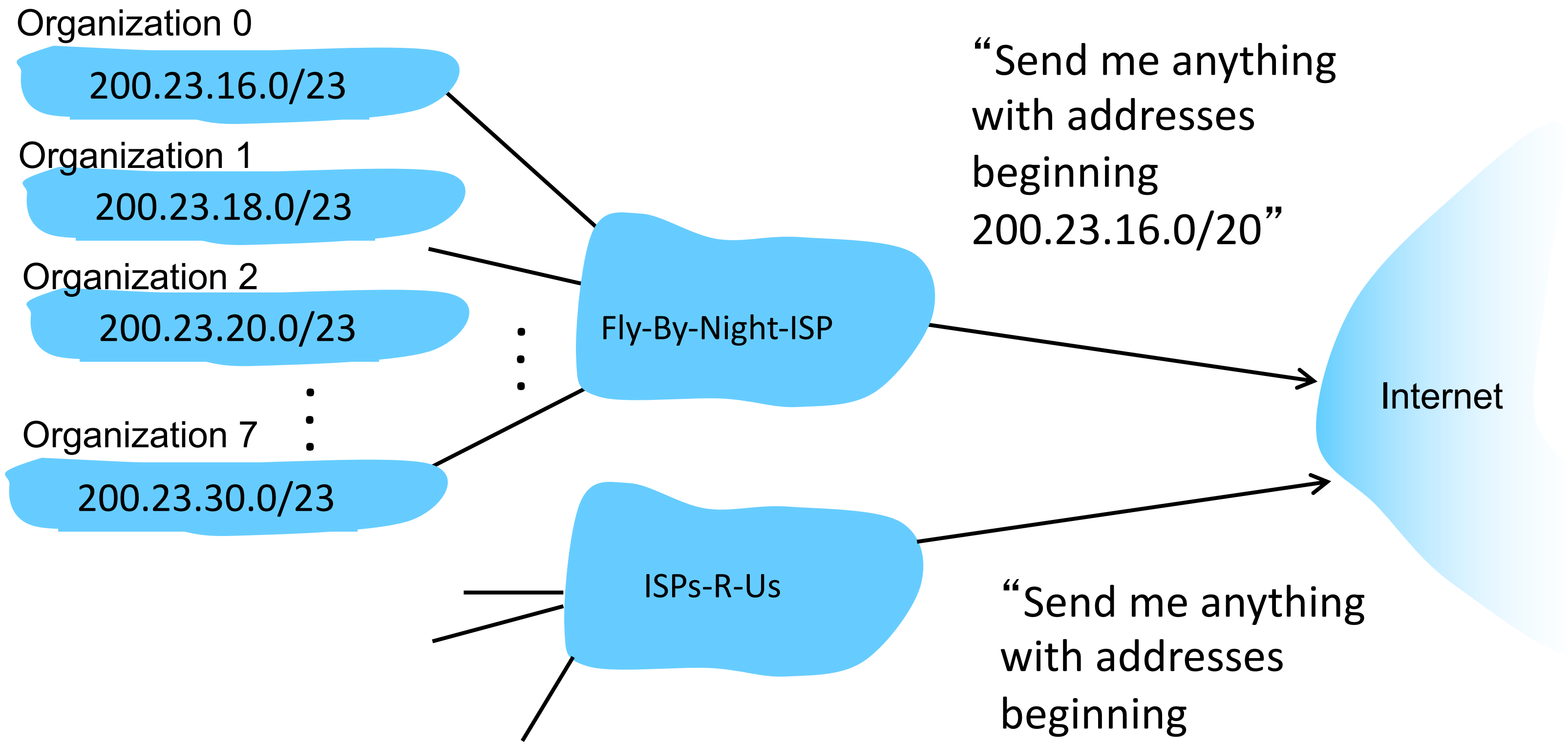
Depth = W/k
Degree = 2^k
Stride = k bits

Even Faster Lookups

- Can use special hardware
  - Content Addressable Memories (CAMs)
  - Allows look-ups on a key rather than flat address
- Huge innovations in the mid-to-late 1990s
  - After CIDR was introduced (in 1994)
  - … and longest-prefix match was a major bottleneck

Hierarchical Addressing: Route Aggregation

Hierarchical addressing allows efficient advertisement of routing information:



"Send me anything with addresses beginning 200.23.16.0/20"

"Send me anything with addresses beginning 199.31.0.0/16"

Hierarchical Addressing: Route Aggregation

“Send me anything with addresses beginning 200.23.16.0/20” translates to the following:

200 23 16 0

200.23.16.0/20 = 11001000 00010111 00010000 00000000 /20 Prefix

/20 prefix

200.23.16.0 = 11001000 00010111 00010000 00000000

|

200.23.31.255 = 11001000 00010111 00011111 11111111

Range represented by the /20 prefix

/20 prefix contains the range of IP addresses that match the the first 20 bits, and can have any value for the remaining 12 bits in the range of :

[first 20 bits] 0000 00000000

[first 20 bits] 1111 11111111

A total of 2^12 = 4,096 IP addresses

Route aggregation in Fly-By-Night ISP

Fly-By-Night-ISP

200.23.16.0/20 = 11001000 00010111 00010000 00000000

Individual Organizations: All of these organizations IP addresses lie withinFly-by-Night's /20 prefix (first 20 bits are the same)

- they more specifically match on the three more bits to form a /23 prefix (first 23 bits of all IP addresses within their organization are the same).
- The last 9 (32-23) bits provide 2^9 = 512 unique IP addresses within each organization.

/23 prefixes

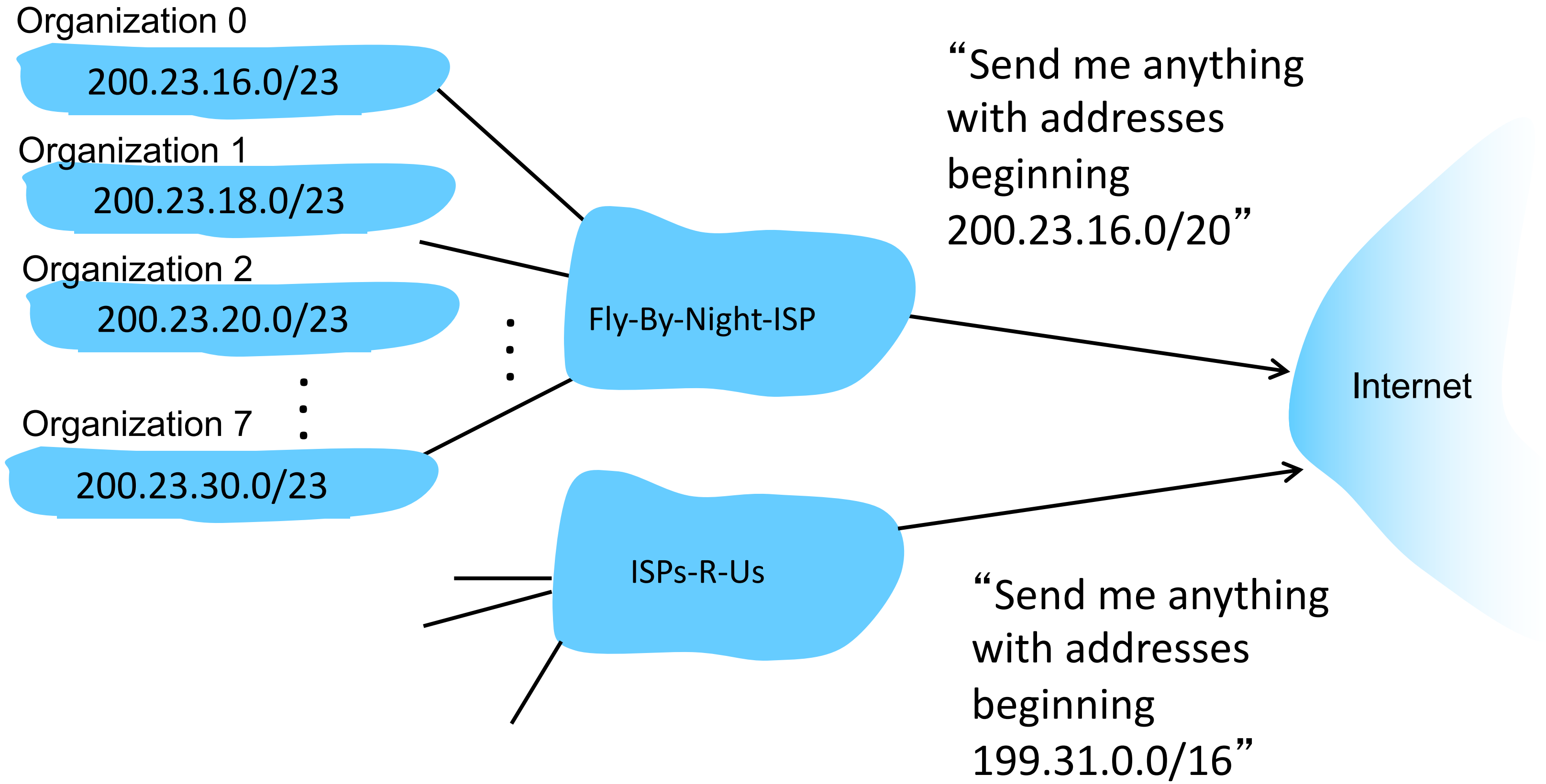
200.23.16.0/23 = 11001000 00010111 00010000 00000000

200.23.18.0/23 = 11001000 00010111 00010010 00000000

200.23.20.0/23 = 11001000 00010111 00010100 00000000

200.23.30.0/23 = 11001000 00010111 00011110 00000000

What should we do if organization 1 decides to switch to ISPs-R-Us?



Organization 0

200.23.16.0/23

Organization 1

200.23.18.0/23

Organization 2

200.23.20.0/23

Organization 7

200.23.30.0/23

Fly-By-Night-ISP

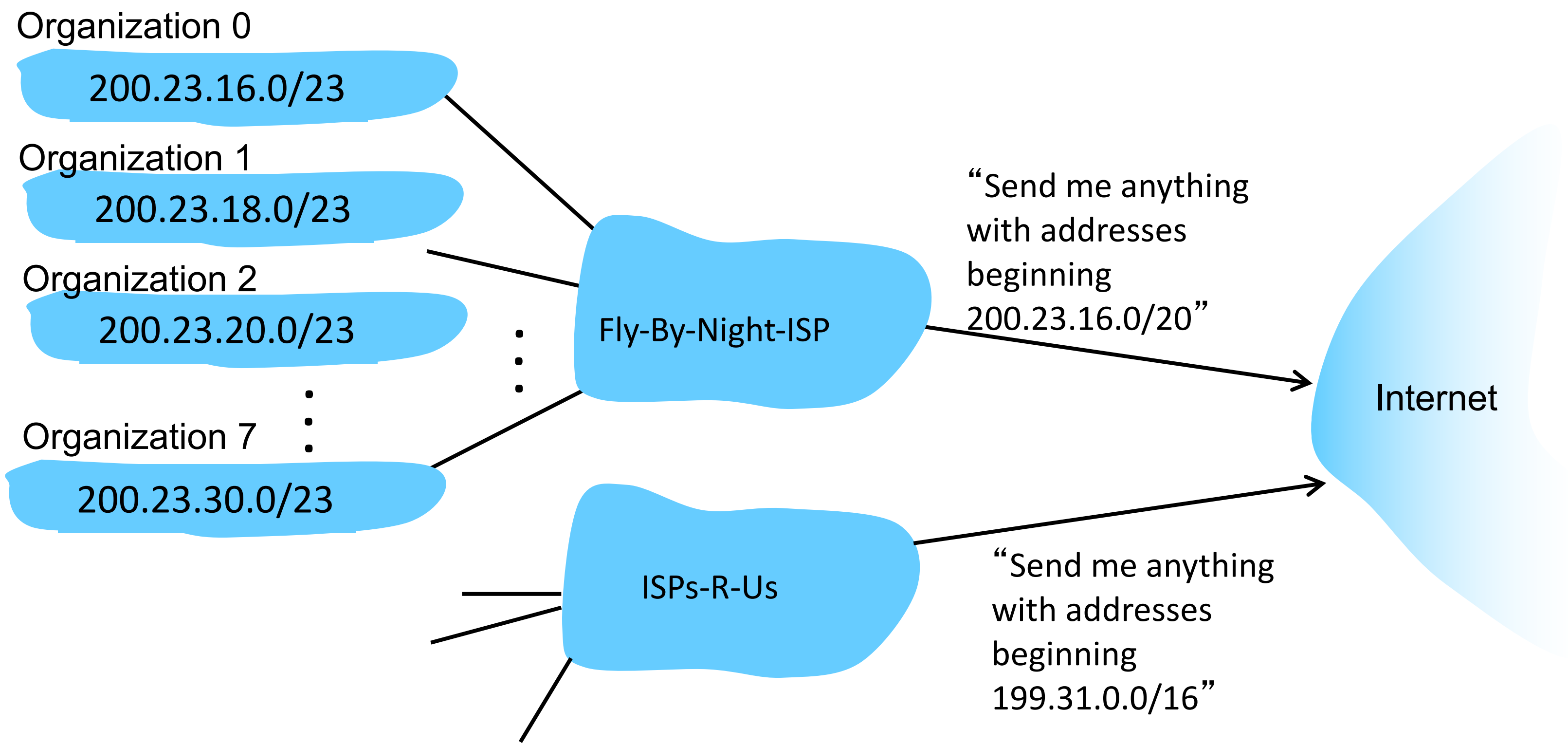
ISPs-R-Us

Internet

"Send me anything with addresses beginning 200.23.16.0/20"

"Send me anything with addresses beginning 199.31.0.0/16"

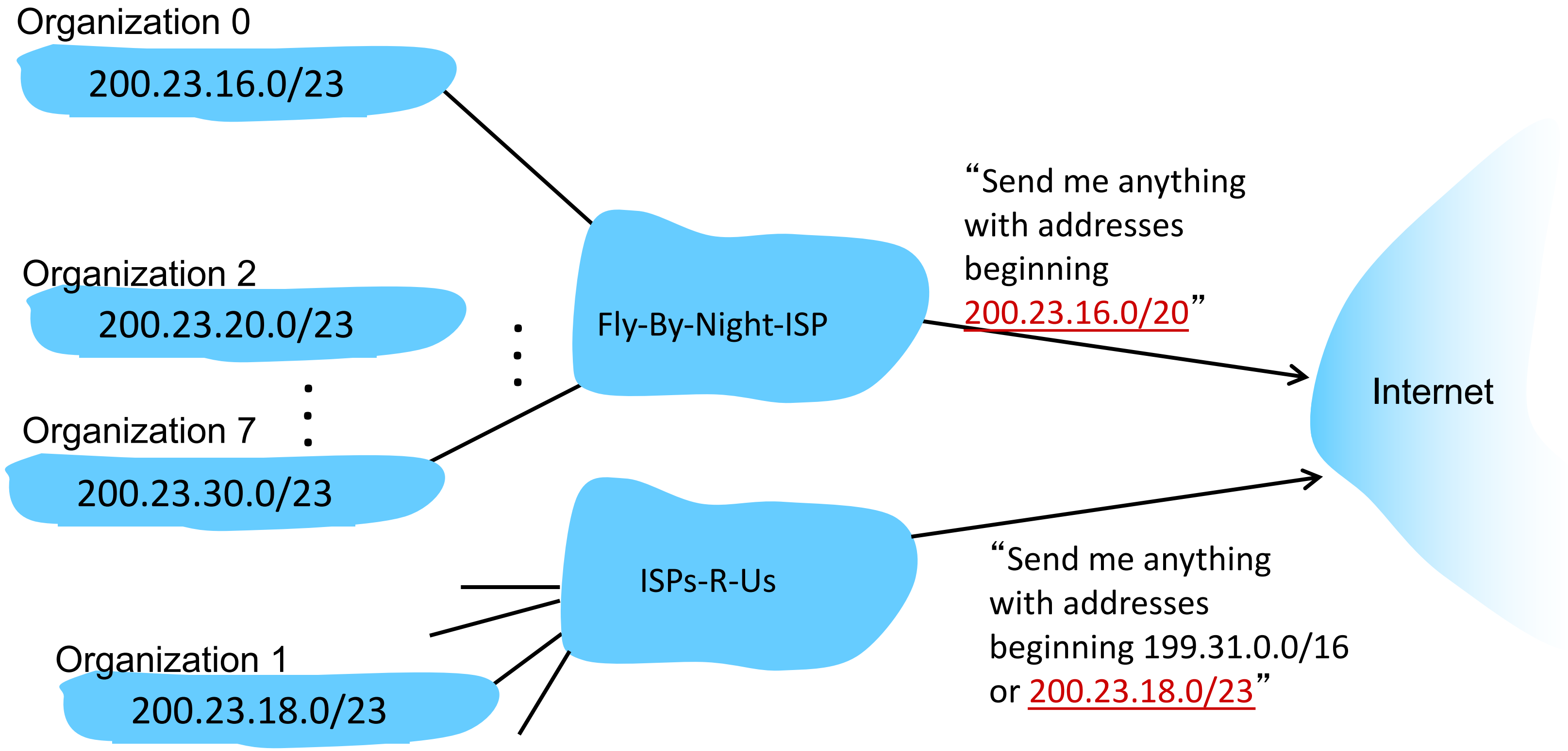
What should we do if organization 1 decides to switch to ISPs-R-Us?



A. Move 200.23.18.0/23 to ISPs-R-Us (and break up Fly-By-Night’s /20 block).
B. Give new addresses to Organization 1 (and force them to change all their addresses).
C. Some other solution.

Hierarchical addressing: More Specific Routes

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

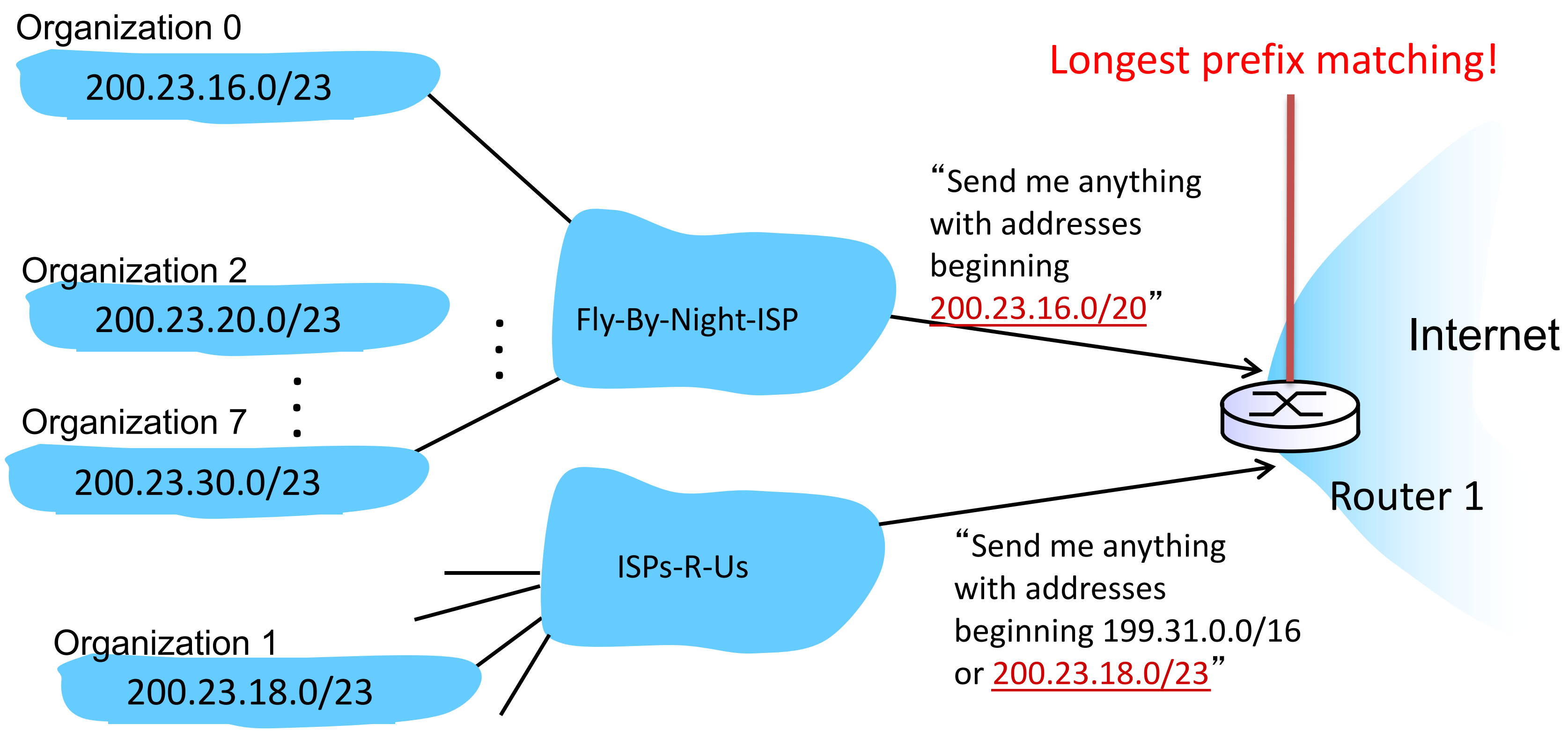


"Send me anything with addresses beginning 200.23.16.0/20"

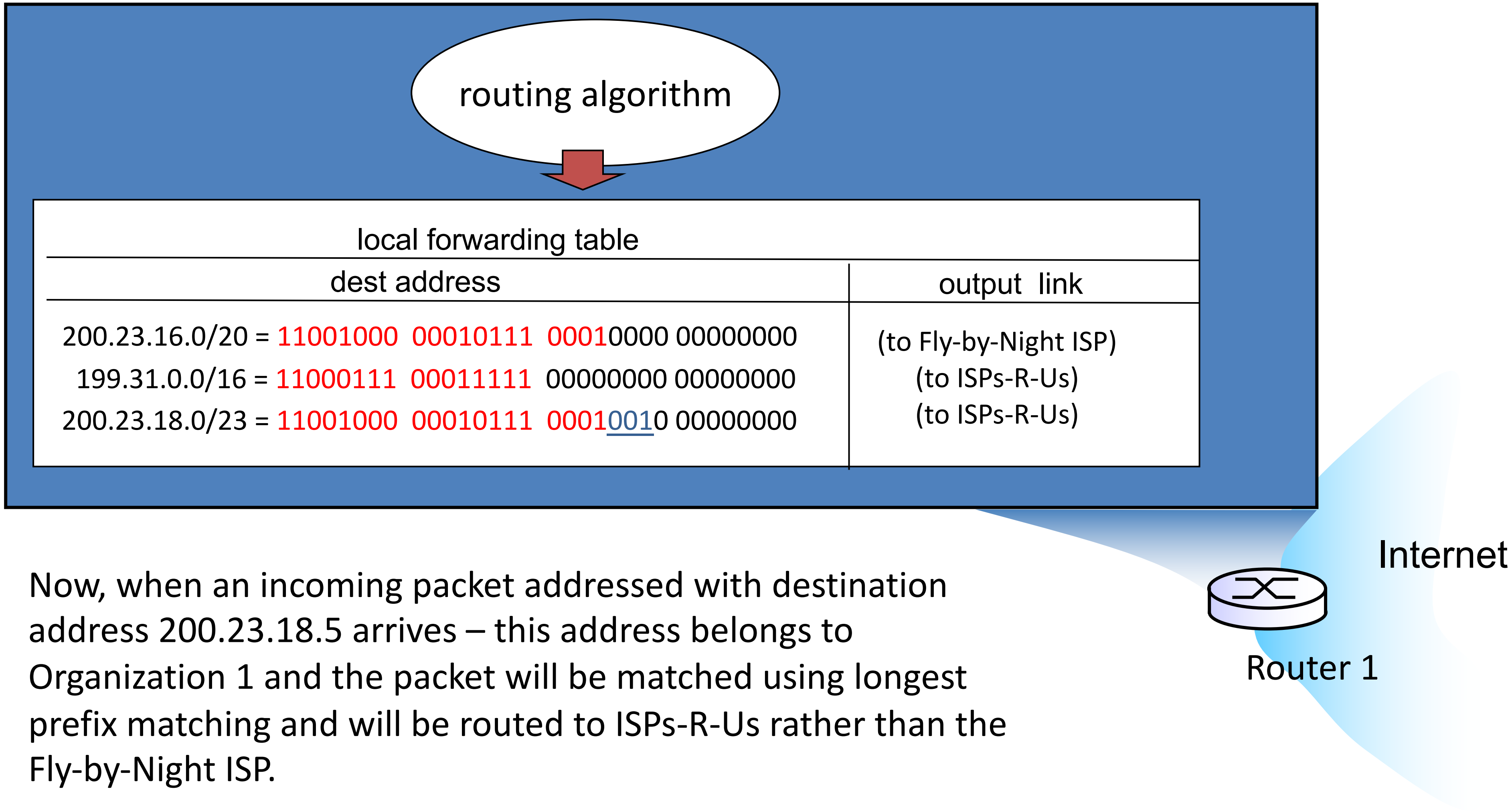
"Send me anything with addresses beginning 199.31.0.0/16 or 200.23.18.0/23"

Hierarchical addressing: More Specific Routes

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



Longest Prefix Matching at Router 1



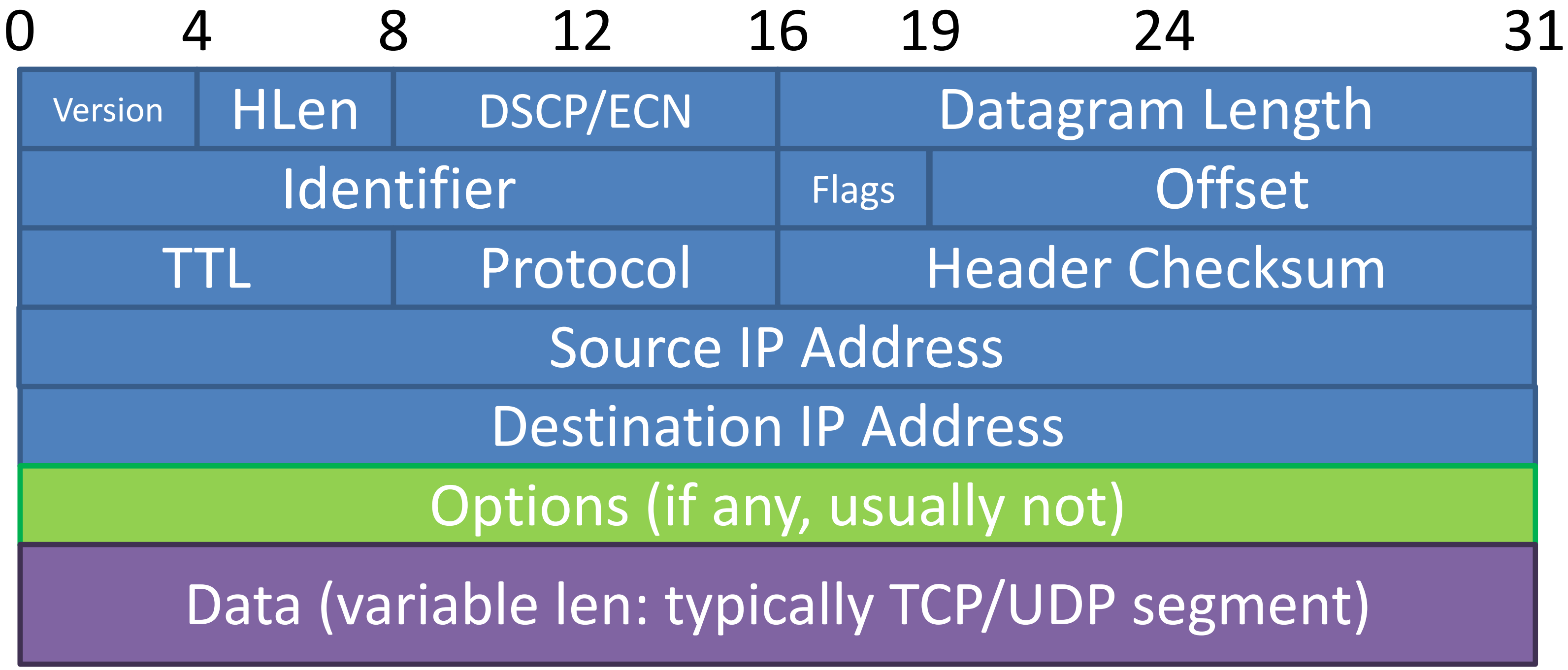
local forwarding table

dest address	output link
200.23.16.0/20 = 11001000 00010111 00010000 00000000	(to Fly-by-Night ISP)
199.31.0.0/16 = 11000111 00011111 00000000 00000000	(to ISPs-R-Us)
200.23.18.0/23 = 11001000 00010111 00010010 00000000	(to ISPs-R-Us)

Now, when an incoming packet addressed with destination address 200.23.18.5 arrives – this address belongs to Organization 1 and the packet will be matched using longest prefix matching and will be routed to ISPs-R-Us rather than the Fly-by-Night ISP.

IP Datagrams

- IP Datagrams are like a letter
  - Totally self-contained
  - Include all necessary addressing information
  - No advanced setup of connections or circuits



How does an end host get an IP address?

- Static IP: hard-coded
  - 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: Dynamic Host Configuration Protocol

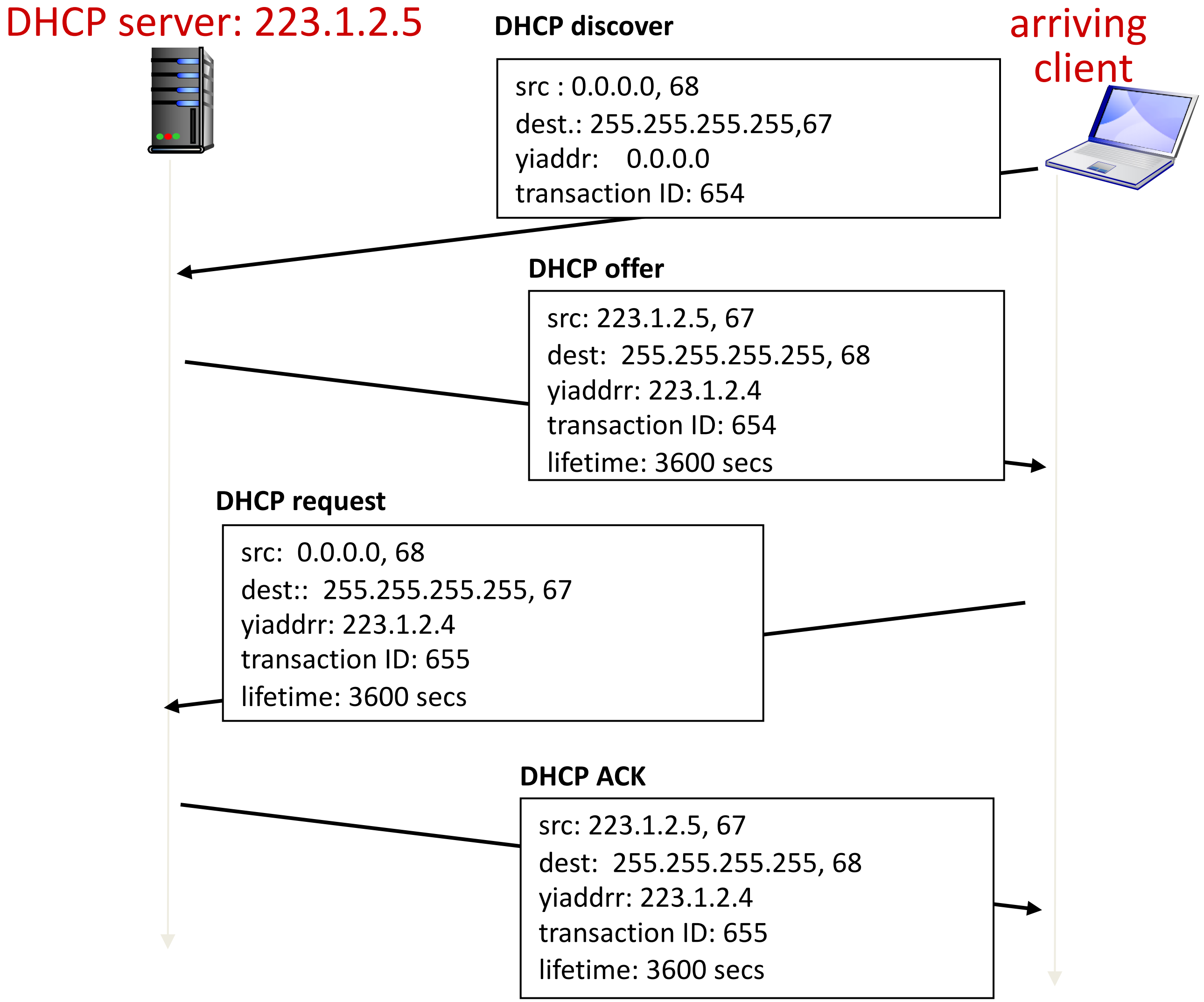
Goal: allow host to dynamically obtain its IP address from network server when it joins network

- can renew its lease on address in use
- allows reuse of addresses
- support for mobile users who want to join network

DHCP overview:

- host broadcasts "DHCP discover" msg [optional]
- DHCP server responds with "DHCP offer" msg [optional]
- host requests IP address: "DHCP request" msg
- DHCP server sends address: "DHCP ack" msg

DHCP client-server scenario



DHCP server: 223.1.2.5

arriving client

DHCP discover

src : 0.0.0.0, 68
dest.: 255.255.255.255,67
yiaddr: 0.0.0.0
transaction ID: 654

DHCP offer

src: 223.1.2.5, 67
dest: 255.255.255.255, 68
yiaddrr: 223.1.2.4
transaction ID: 654
lifetime: 3600 secs

DHCP request

src: 0.0.0.0, 68
dest:: 255.255.255.255, 67
yiaddrr: 223.1.2.4
transaction ID: 655
lifetime: 3600 secs

DHCP ACK

src: 223.1.2.5, 67
dest: 255.255.255.255, 68
yiaddrr: 223.1.2.4
transaction ID: 655
lifetime: 3600 secs

DHCP: More than IP Addresses

DHCP can return more than just allocated IP address on subnet:

- address of first-hop router for client (default GW)
- name and IP address of DNS server(s)
- subnet mask

IP Fragmentation, Reassembly

- Higher layer’s data unit is too large for the lower layer
- Fragmentation: taking a large data unit and breaking it into smaller chunks
- Assembly: combining chunks into the original data unit.

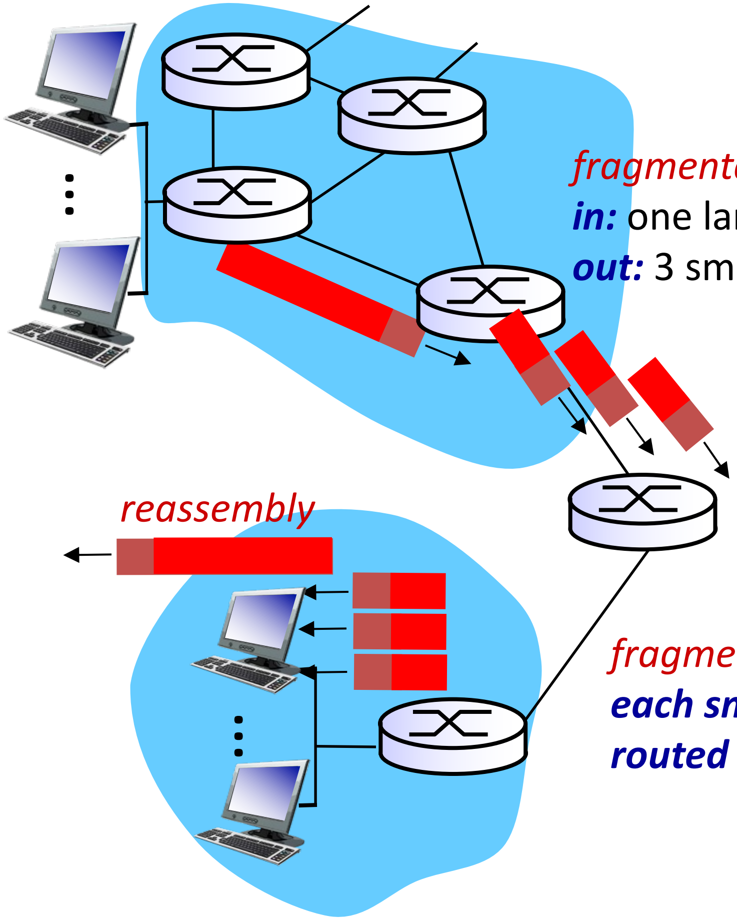
Examples:

- Transport: TCP takes stream of bytes and breaks into TCP segments
- Network: IP takes packets too big for a link and breaks them up into IP fragments
- Link: 6lowpan takes IPv6 packets and breaks them into link fragments if needed.

IP Fragmentation, Reassembly

Different link layers have different MTUs (max transfer size) - largest possible link-level frame

large IP datagram divided ("fragmented") into several datagrams

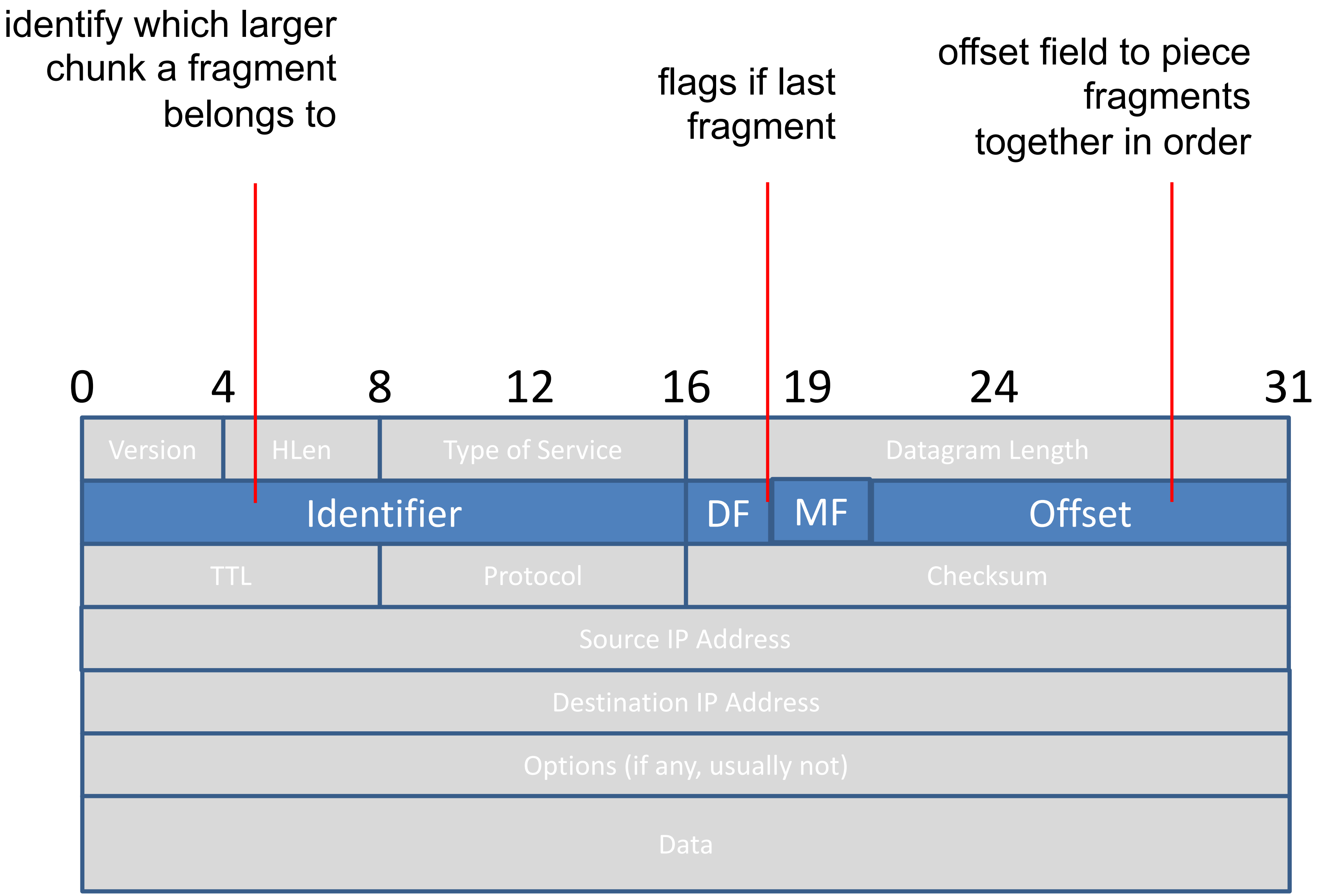


fragmentation:
in: one large datagram
out: 3 smaller datagrams

reassembly

fragmentation:
each smaller fragment is routed independently

IP Datagram Format



identify which larger chunk a fragment belongs to

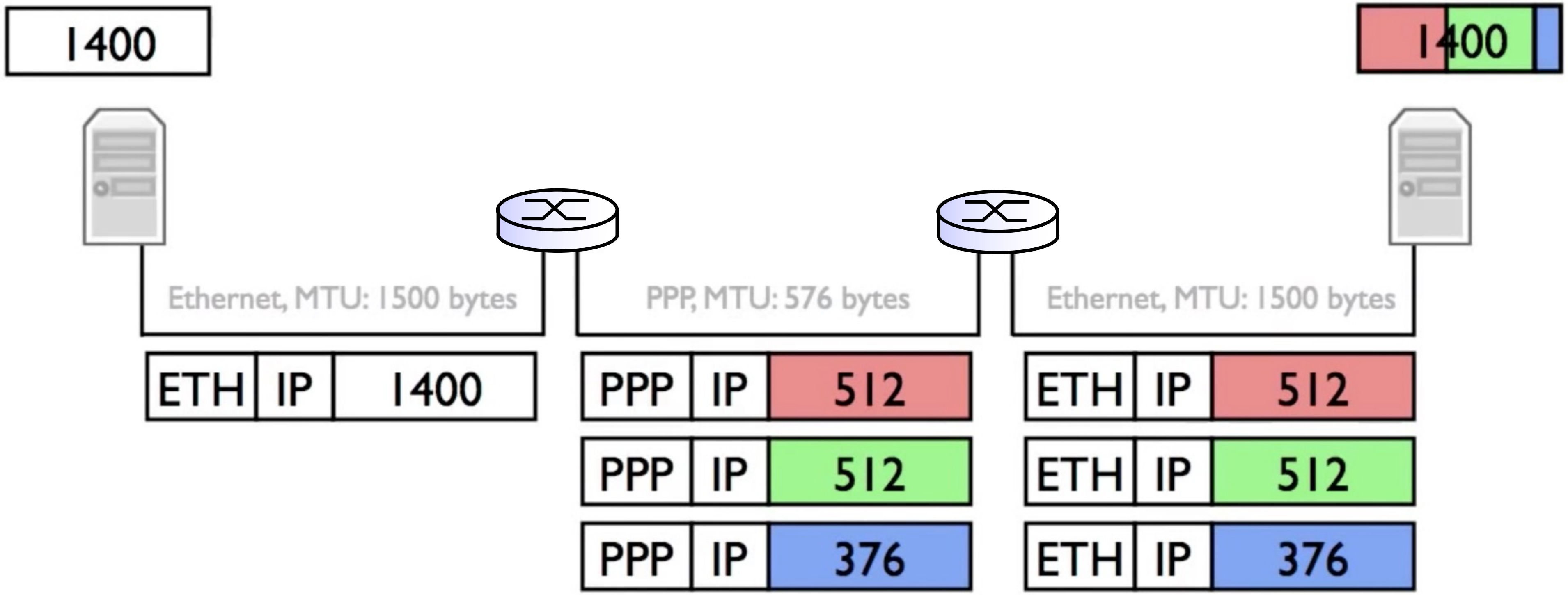
flags if last fragment

offset field to piece fragments together in order

<table>
<tr><td>Version</td><td>HLen</td><td>Type of Service</td><td colspan="3">Datagram Length</td></tr>
<tr><td colspan="3">Identifier</td><td>DF</td><td>MF</td><td>Offset</td></tr>
<tr><td colspan="2">TTL</td><td>Protocol</td><td colspan="3">Checksum</td></tr>
<tr><td colspan="6">Source IP Address</td></tr>
<tr><td colspan="6">Destination IP Address</td></tr>
<tr><td colspan="6">Options (if any, usually not)</td></tr>
<tr><td colspan="6">Data</td></tr>
</table>

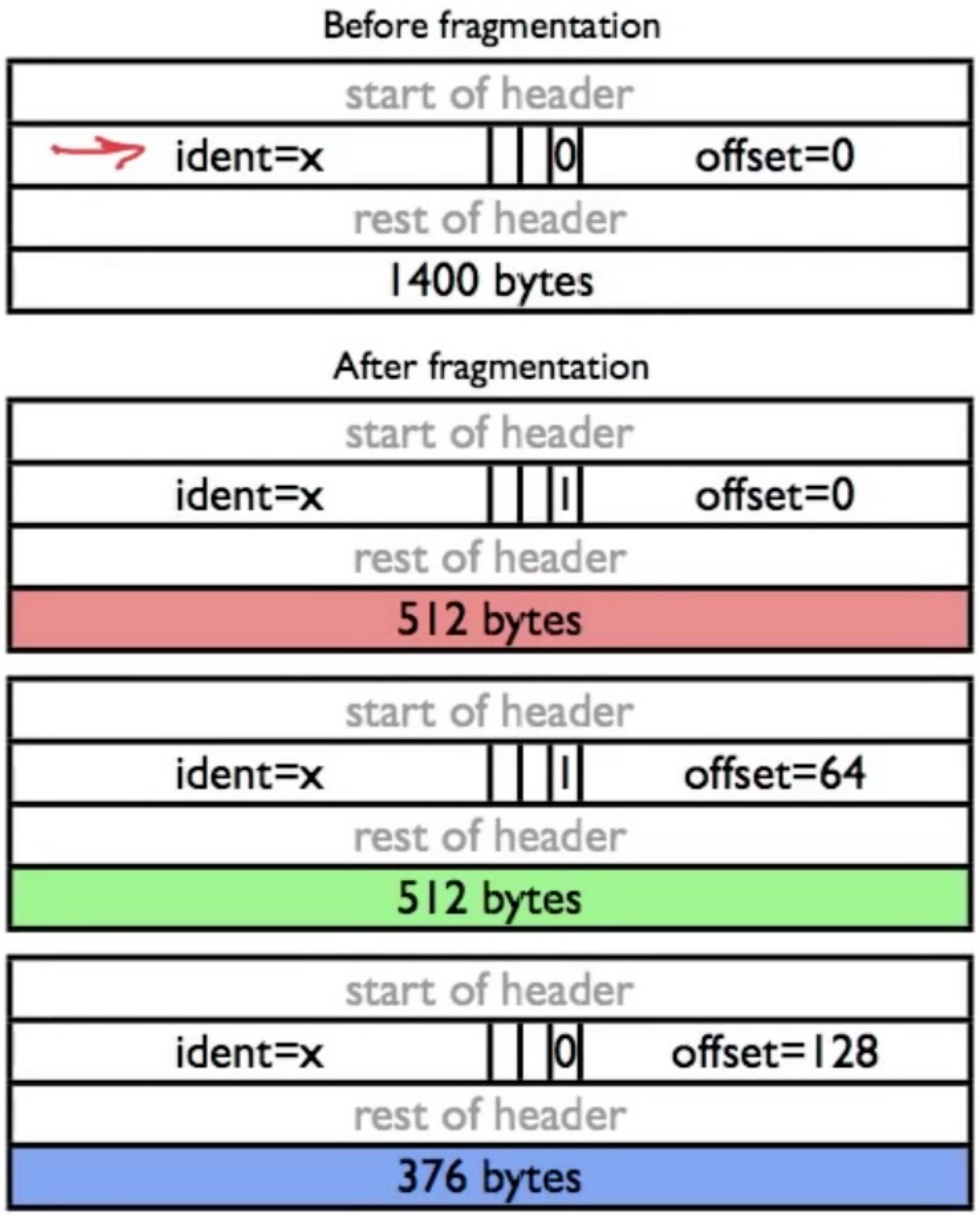
IP Fragmentation, Reassembly

- Different link layers have different MTUs (max transfer size) - largest possible link-level frame
- large IP datagram divided (“fragmented”) into several datagrams
  - Reassembled only at final destination
  - IP header bits used to identify, order related fragments



IP Fragmentation, Reassembly

- IP addresses plus ident field identify fragments of a packet
- MF bit is 1 in all but last fragment
- Offset field says location of fragment (in 8 byte chunks)
  - All fragments except last one must be multiple of 8 bytes long



Before fragmentation

start of header
ident=x | | | 0 | offset=0
rest of header
1400 bytes

After fragmentation

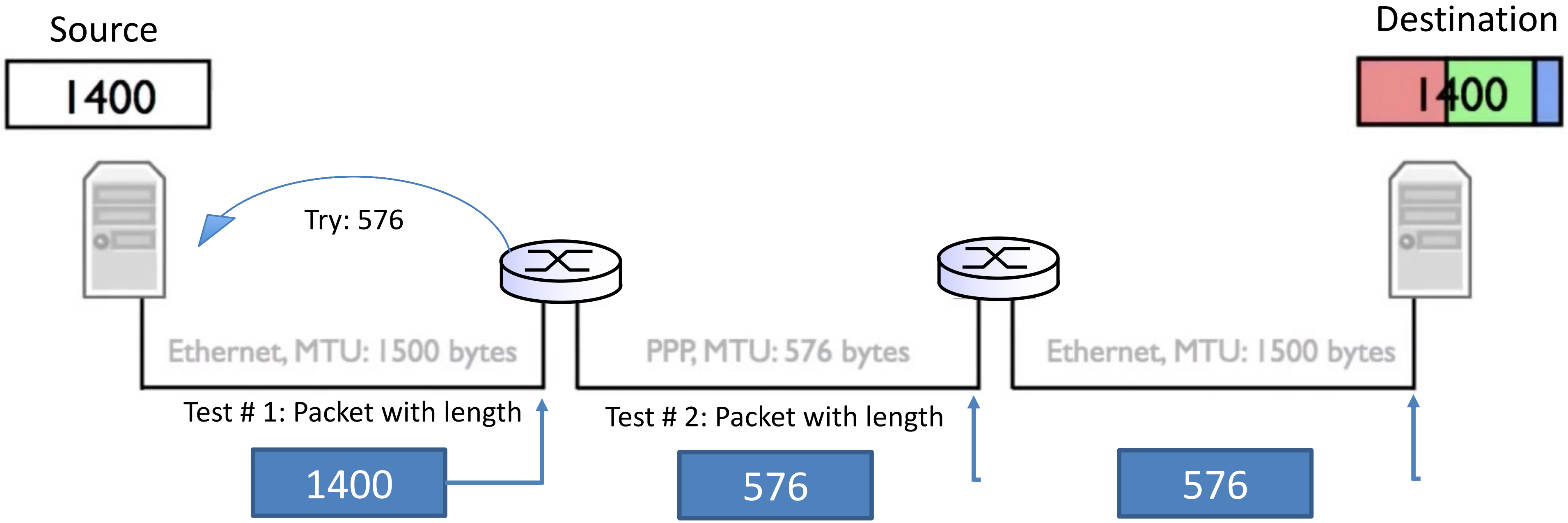
start of header
ident=x | | | 1 | offset=0
rest of header
512 bytes

start of header
ident=x | | | 1 | offset=64
rest of header
512 bytes

start of header
ident=x | | | 0 | offset=128
rest of header
376 bytes

IP Path MTU Discovery

- Avoid fragmentation: Host tests link with a large packet
- Implemented with ICMP: set DF – do not fragment. Triggers error response from a router



How can we use this for evil?

A. Send fragments that overlap.

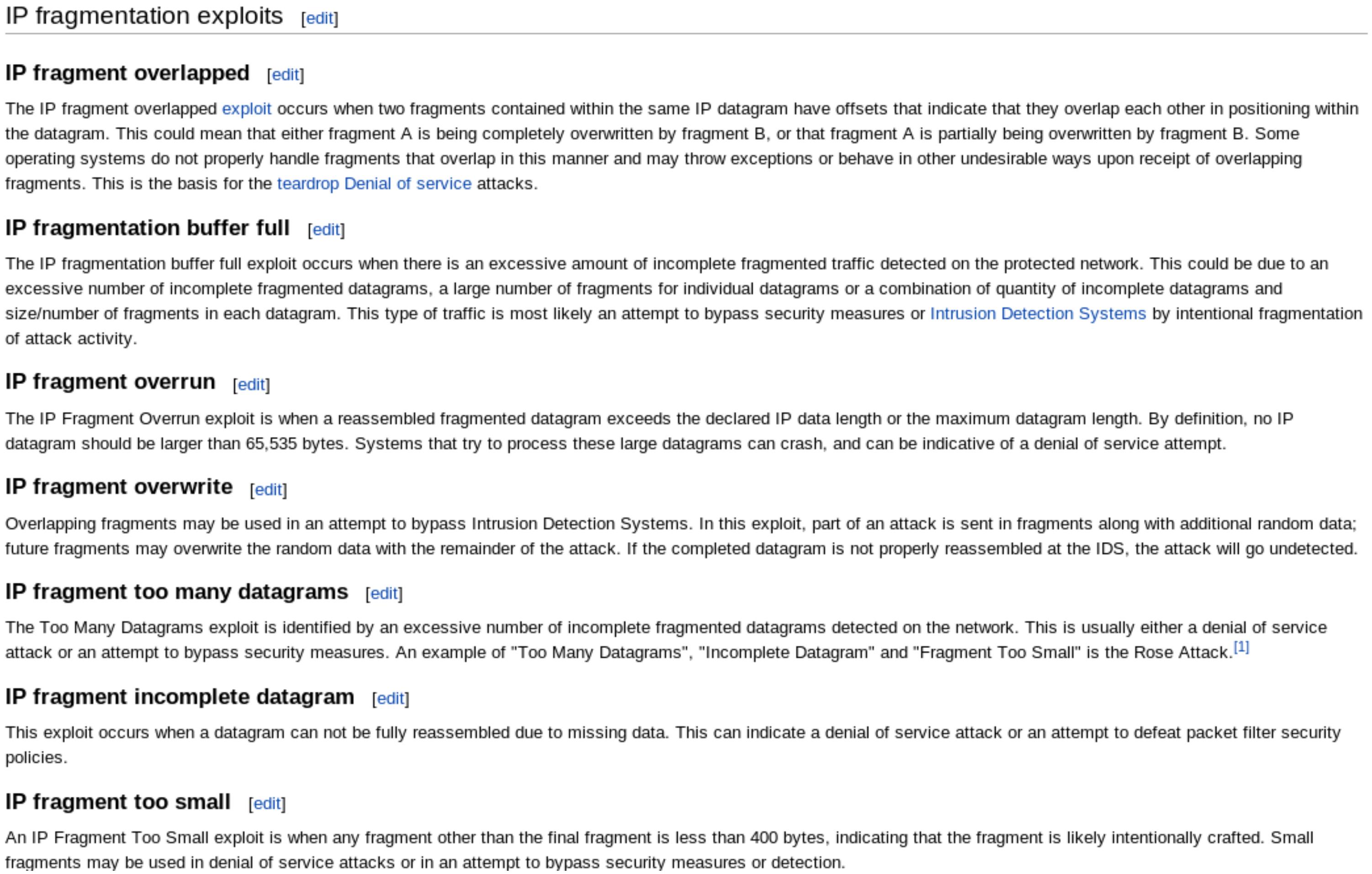
B. Send many tiny fragments, none of which have offset 0.

C. Send fragments that, when assembled, are bigger than the maximum IP datagram.

D. More than one of the above.

E. Nah, networks (and operating systems) are too robust for this to cause problems.

IP Fragmentation Attacks…



IP fragmentation exploits [edit]

IP fragment overlapped [edit]

The IP fragment overlapped exploit occurs when two fragments contained within the same IP datagram have offsets that indicate that they overlap each other in positioning within the datagram. This could mean that either fragment A is being completely overwritten by fragment B, or that fragment A is partially being overwritten by fragment B. Some operating systems do not properly handle fragments that overlap in this manner and may throw exceptions or behave in other undesirable ways upon receipt of overlapping fragments. This is the basis for the teardrop Denial of service attacks.

IP fragmentation buffer full [edit]

The IP fragmentation buffer full exploit occurs when there is an excessive amount of incomplete fragmented traffic detected on the protected network. This could be due to an excessive number of incomplete fragmented datagrams, a large number of fragments for individual datagrams or a combination of quantity of incomplete datagrams and size/number of fragments in each datagram. This type of traffic is most likely an attempt to bypass security measures or Intrusion Detection Systems by intentional fragmentation of attack activity.

IP fragment overrun [edit]

The IP Fragment Overrun exploit is when a reassembled fragmented datagram exceeds the declared IP data length or the maximum datagram length. By definition, no IP datagram should be larger than 65,535 bytes. Systems that try to process these large datagrams can crash, and can be indicative of a denial of service attempt.

IP fragment overwrite [edit]

Overlapping fragments may be used in an attempt to bypass Intrusion Detection Systems. In this exploit, part of an attack is sent in fragments along with additional random data; future fragments may overwrite the random data with the remainder of the attack. If the completed datagram is not properly reassembled at the IDS, the attack will go undetected.

IP fragment too many datagrams [edit]

The Too Many Datagrams exploit is identified by an excessive number of incomplete fragmented datagrams detected on the network. This is usually either a denial of service attack or an attempt to bypass security measures. An example of "Too Many Datagrams", "Incomplete Datagram" and "Fragment Too Small" is the Rose Attack.[1]

IP fragment incomplete datagram [edit]

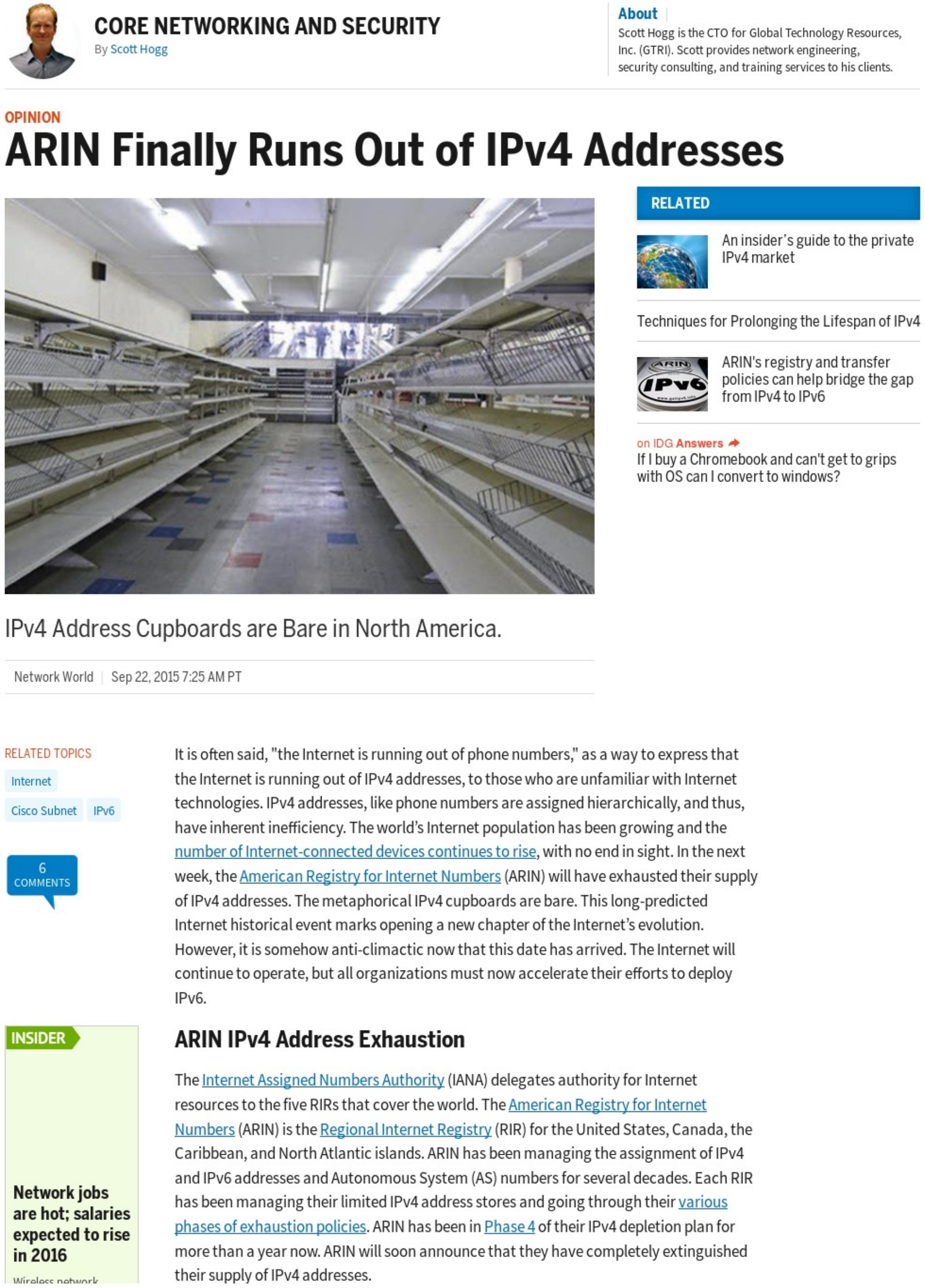
This exploit occurs when a datagram can not be fully reassembled due to missing data. This can indicate a denial of service attack or an attempt to defeat packet filter security policies.

IP fragment too small [edit]

An IP Fragment Too Small exploit is when any fragment other than the final fragment is less than 400 bytes, indicating that the fragment is likely intentionally crafted. Small fragments may be used in denial of service attacks or in an attempt to bypass security measures or detection.

Recall: IPv4 Addresses

- 32-bit number, must be globally unique
- 2^{32} => 4,294,967,296 possible addresses
- How many do you have?



CORE NETWORKING AND SECURITY

By Scott Hogg

About

Scott Hogg is the CTO for Global Technology Resources, Inc. (GTRI). Scott provides network engineering, security consulting, and training services to his clients.

OPINION

ARIN Finally Runs Out of IPv4 Addresses

RELATED

- An insider's guide to the private IPv4 market
- Techniques for Prolonging the Lifespan of IPv4
- ARIN's registry and transfer policies can help bridge the gap from IPv4 to IPv6

on IDG **Answers**

If I buy a Chromebook and can't get to grips with OS can I convert to windows?

IPv4 Address Cupboards are Bare in North America.

Network World | Sep 22, 2015 7:25 AM PT

RELATED TOPICS

Internet

Cisco Subnet | IPv6

6 COMMENTS

It is often said, "the Internet is running out of phone numbers," as a way to express that the Internet is running out of IPv4 addresses, to those who are unfamiliar with Internet technologies. IPv4 addresses, like phone numbers are assigned hierarchically, and thus, have inherent inefficiency. The world's Internet population has been growing and the number of Internet-connected devices continues to rise, with no end in sight. In the next week, the American Registry for Internet Numbers (ARIN) will have exhausted their supply of IPv4 addresses. The metaphorical IPv4 cupboards are bare. This long-predicted Internet historical event marks opening a new chapter of the Internet's evolution. However, it is somehow anti-climactic now that this date has arrived. The Internet will continue to operate, but all organizations must now accelerate their efforts to deploy IPv6.

INSIDER

Network jobs are hot; salaries expected to rise in 2016

Wireless network

ARIN IPv4 Address Exhaustion

The Internet Assigned Numbers Authority (IANA) delegates authority for Internet resources to the five RIRs that cover the world. The American Registry for Internet Numbers (ARIN) is the Regional Internet Registry (RIR) for the United States, Canada, the Caribbean, and North Atlantic islands. ARIN has been managing the assignment of IPv4 and IPv6 addresses and Autonomous System (AS) numbers for several decades. Each RIR has been managing their limited IPv4 address stores and going through their various phases of exhaustion policies. ARIN has been in Phase 4 of their IPv4 depletion plan for more than a year now. ARIN will soon announce that they have completely extinguished their supply of IPv4 addresses.

Seriously, we're done now. We're done

Exhausted with never-ending internet exhaustion

By Kieren McCarthy in San Francisco 15 Feb 2017 at 23:07 214 SHARE ▼

You may have heard this before, but we are really, really running out of public IPv4 addresses.

This week, the regional internet registry responsible for Latin America and the Caribbean, LACNIC, announced it has moved to "phase 3" of its plan to dispense with the remaining network addresses, meaning that only companies that have not received any IPv4 space are eligible. There is no phase 4.

That means LACNIC is down to its last 4,698,112 public IPv4 addresses (although that may increase as it recovers a little bit of space over time).

Data Centre ▸ Networks

OK, this time it's for real: The last available IPv4 address block has gone

Now for the last time, will you all please shift to IPv6?!

By Kieren McCarthy in San Francisco 18 Apr 2018 at 22:10 211 SHARE ▼

You may have heard this one before, but we have now really run out of public IPv4 address blocks.

The Internet Assigned Numbers Authority – the global overseers of network addresses – said it had run out of new addresses to dish out to regional internet registries (RIRs) in 2011. One of those RIRs, the Asia-Pacific Network Information Centre, said it was out of available IPv4 addresses later that year.

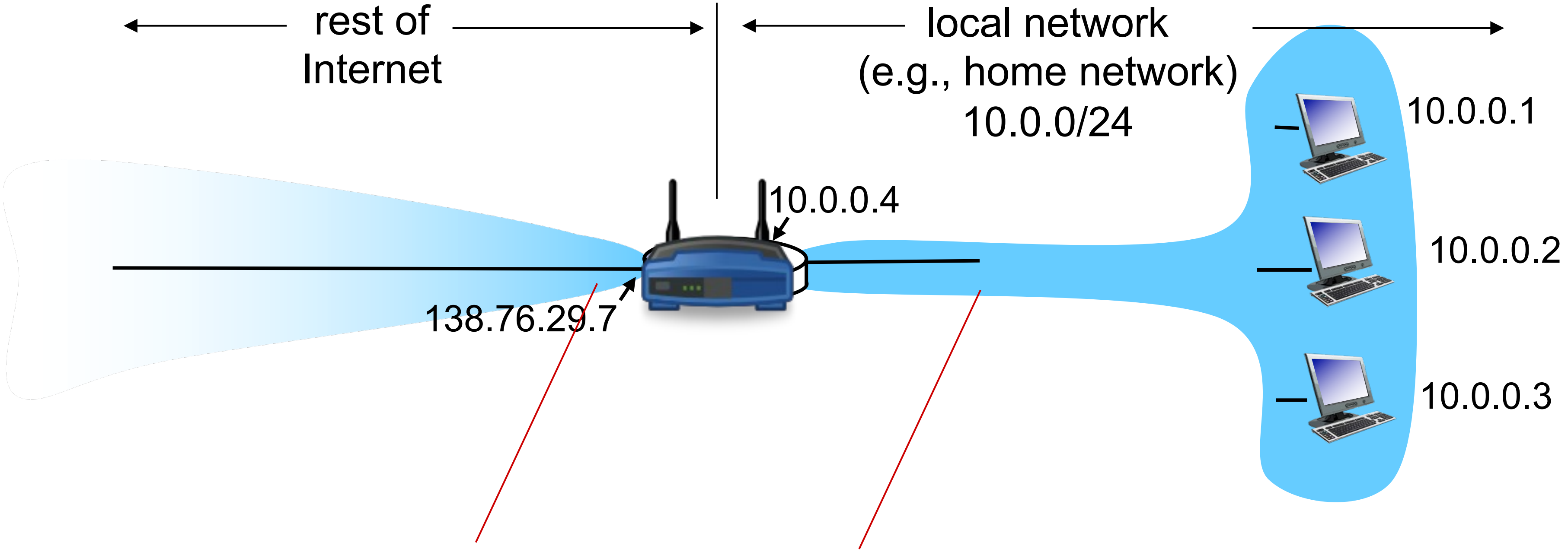
Then Europe's RIR, Réseaux IP Européens aka RIPE, ran dry in September 2012, followed by the Latin America and Caribbean Network Information Centre (LACNIC) in June 2014. Next, the American Registry for Internet Numbers hit an IPv4 drought in September 2015.

Private Addresses

- Defined in RFC 1918:
  - 10.0.0.0/8 (16,777,216 hosts)
  - 172.16.0.0/12 (1,048,576 hosts)
  - 192.168.0.0/16 (65536 hosts)

- These addresses shouldn’t be routed.
  - Anyone can use them.
  - Often adopted for use with NAT.

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

Implementing NAT

- Two hosts communicate with same destination
  - Destination needs to differentiate the two
- Map outgoing packets
  - Change source address and source port
- Maintain a translation table
  - Map of (src addr, port #) to (NAT addr, new port #)
- Map incoming packets
  - Map the destination address/port to the local host

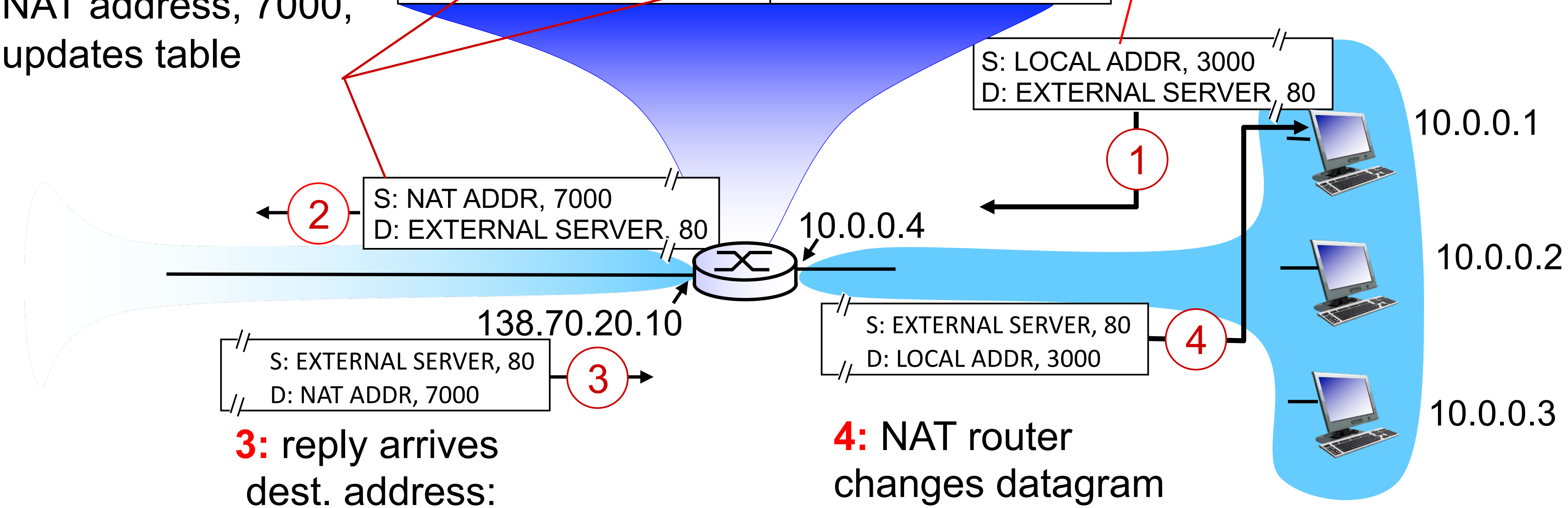
NAT: network address translation

NAT translation table

Wide Area Network side addr	Local Area Network side addr
NAT Address, 7000	Local address, 3000
……	……

1: host 10.0.0.1 sends datagram to external server, 80

2: NAT router changes datagram source addr from local address, 3000 to NAT address, 7000, updates table



3: reply arrives dest. address: NAT address, 7000

4: NAT router changes datagram dest addr from NAT Address, 7000 to Local Address, 3000

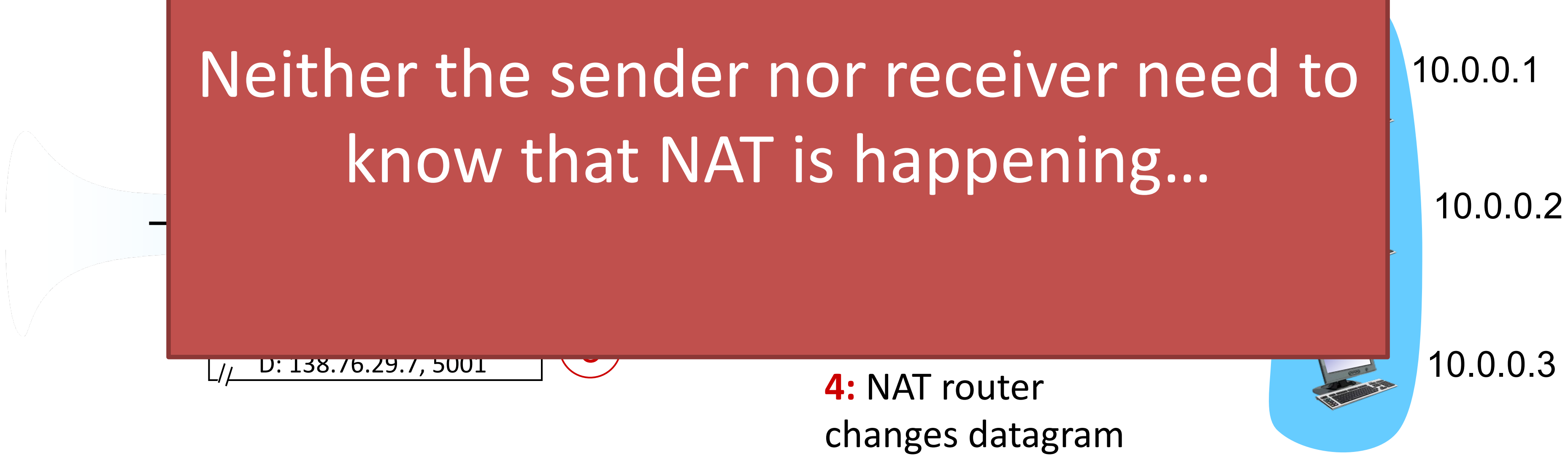
EXTERNAL SERVER: 120.130.140.150
LOCAL ADDR: 10.0.0.1
NAT ADDR: 138.70.20.10

NAT: network address translation

NAT translation table

WAN side addr	LAN side addr
NAT Address, 7000	Local address, 3000
……	……

Neither the sender nor receiver need to know that NAT is happening…



4: NAT router changes datagram dest addr from NAT Address, 7000 to Local Address, 3000

NAT Advantages

- Organizations need fewer IP addresses from their ISP.
  - With a 16-bit port field, we can put 65535 connections behind one external IP address!
- Organizations can change internal network IPs without having to change outside world IPs.

Principled Objections Against NAT

- Routers are not supposed to look at port #s
  - Network layer should care only about *IP* header
  - … and not be looking at the *port numbers* at all
- NAT violates the end-to-end argument
  - Network nodes should not modify the packets
- IPv6 is a cleaner solution
  - Better to migrate than to limp along with a hack

That's what happens when network puts power in hands of end users!

When we use NATs, devices inside the local network are not explicitly addressable or visible to the outside world.

A. This is an advantage.

B. This is a disadvantage.

How do we feel about NAT?

A. NAT is great! It conserves IP addresses and makes it harder to reach non-public machines.

B. NAT is mostly good, but has a few negative features. No big deal.

C. NAT is mostly bad, but in some cases, it’s a necessary evil.

D. NAT is an abomination that violates the end to end principle, and we should not use it!

IPv6

- Initial motivation: 32-bit address space soon to be completely allocated, any day now™.

- Additional motivation:
  - header format helps speed processing/forwarding
  - header changes to facilitate QoS

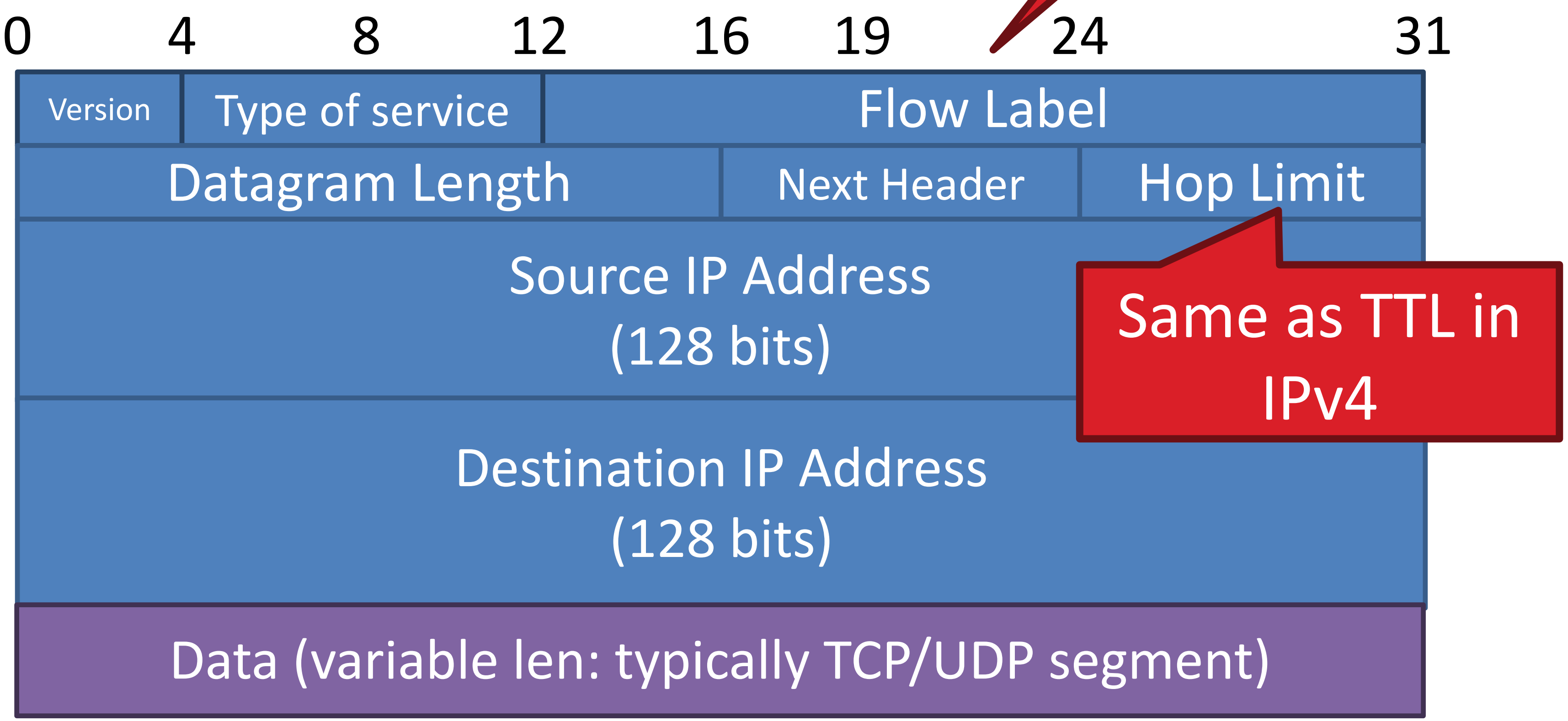
IPv6 datagram format:

- fixed-length 40 byte header
- no fragmentation allowed

IPv6 Header

- Double the size of IPv4 (320 bits vs. 160 bits)

Groups packets into flows, used for QoS



0 4 8 12 16 19 24 31

Version | Type of service | Flow Label

Datagram Length | Next Header | Hop Limit

Source IP Address (128 bits)

Destination IP Address (128 bits)

Data (variable len: typically TCP/UDP segment)

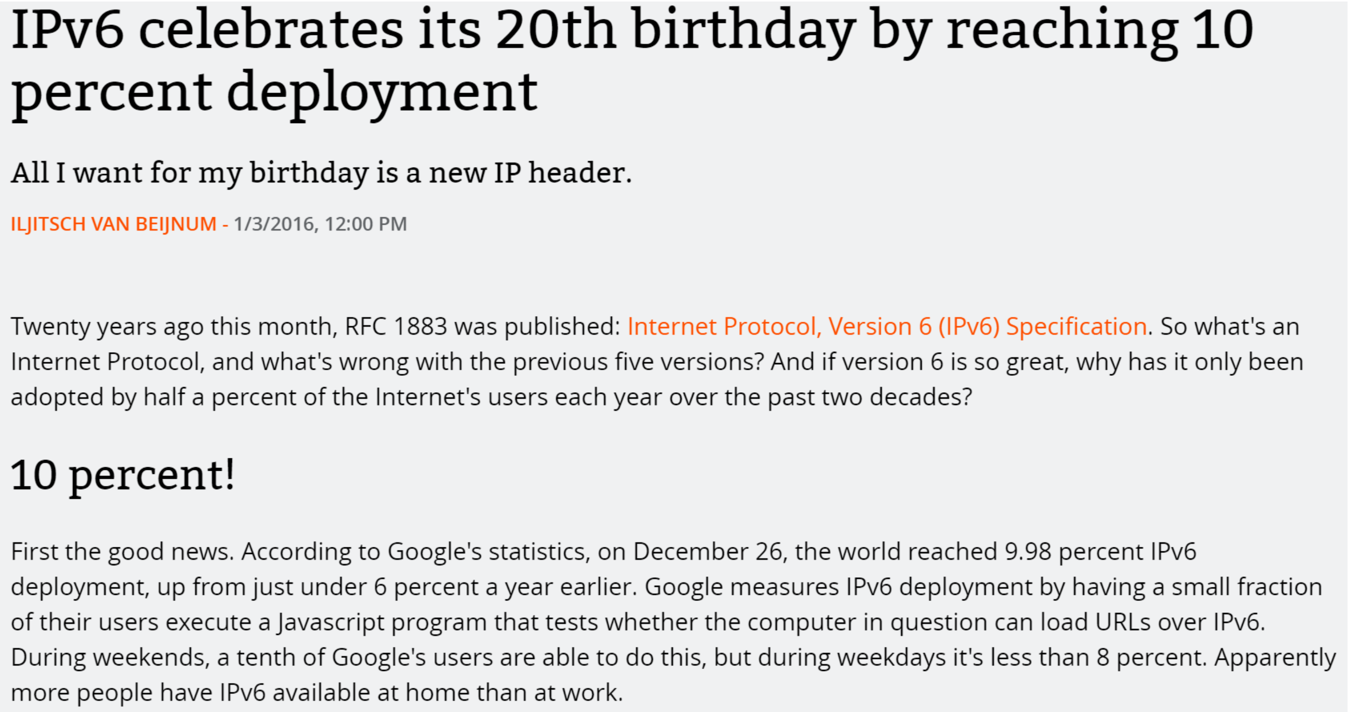
Same as TTL in IPv4

Other changes from IPv4

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

IPv6 (vs. IPv4)

- Simpler, faster, better
- How much traffic on the Internet is IPv6?
- Why?!



IPv6 celebrates its 20th birthday by reaching 10 percent deployment

All I want for my birthday is a new IP header.

ILJITSCH VAN BEIJNUM - 1/3/2016, 12:00 PM

Twenty years ago this month, RFC 1883 was published: Internet Protocol, Version 6 (IPv6) Specification. So what's an Internet Protocol, and what's wrong with the previous five versions? And if version 6 is so great, why has it only been adopted by half a percent of the Internet's users each year over the past two decades?

10 percent!

First the good news. According to Google's statistics, on December 26, the world reached 9.98 percent IPv6 deployment, up from just under 6 percent a year earlier. Google measures IPv6 deployment by having a small fraction of their users execute a Javascript program that tests whether the computer in question can load URLs over IPv6. During weekends, a tenth of Google's users are able to do this, but during weekdays it's less than 8 percent. Apparently more people have IPv6 available at home than at work.

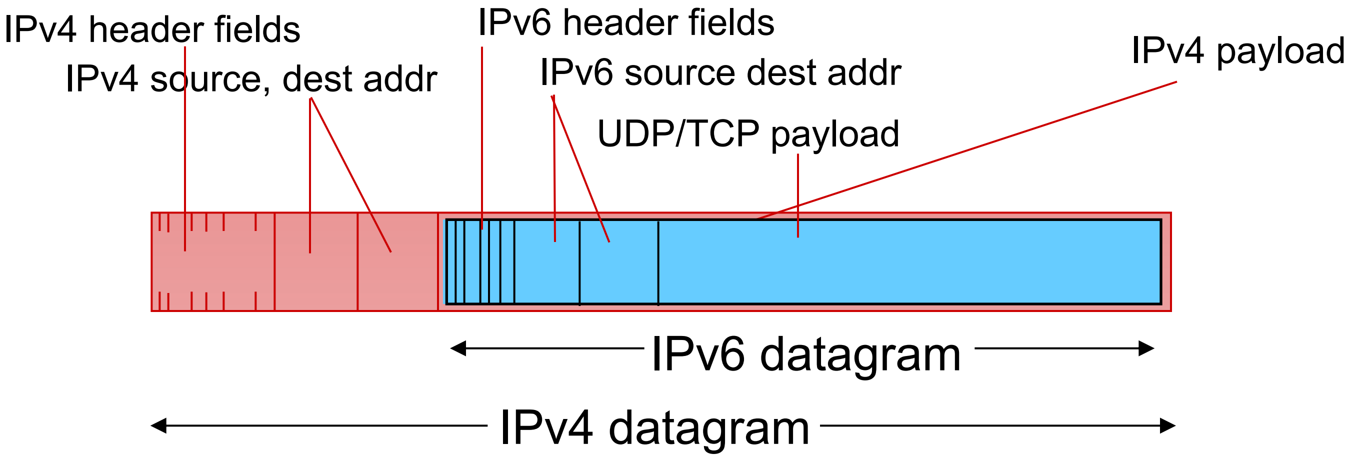
Transitioning to IPv6

- Option 1: “Flag day”
  - How do we get *everyone* on the Internet to agree?
  - Whose authority to decide when?
  - Can you imagine how much would break?

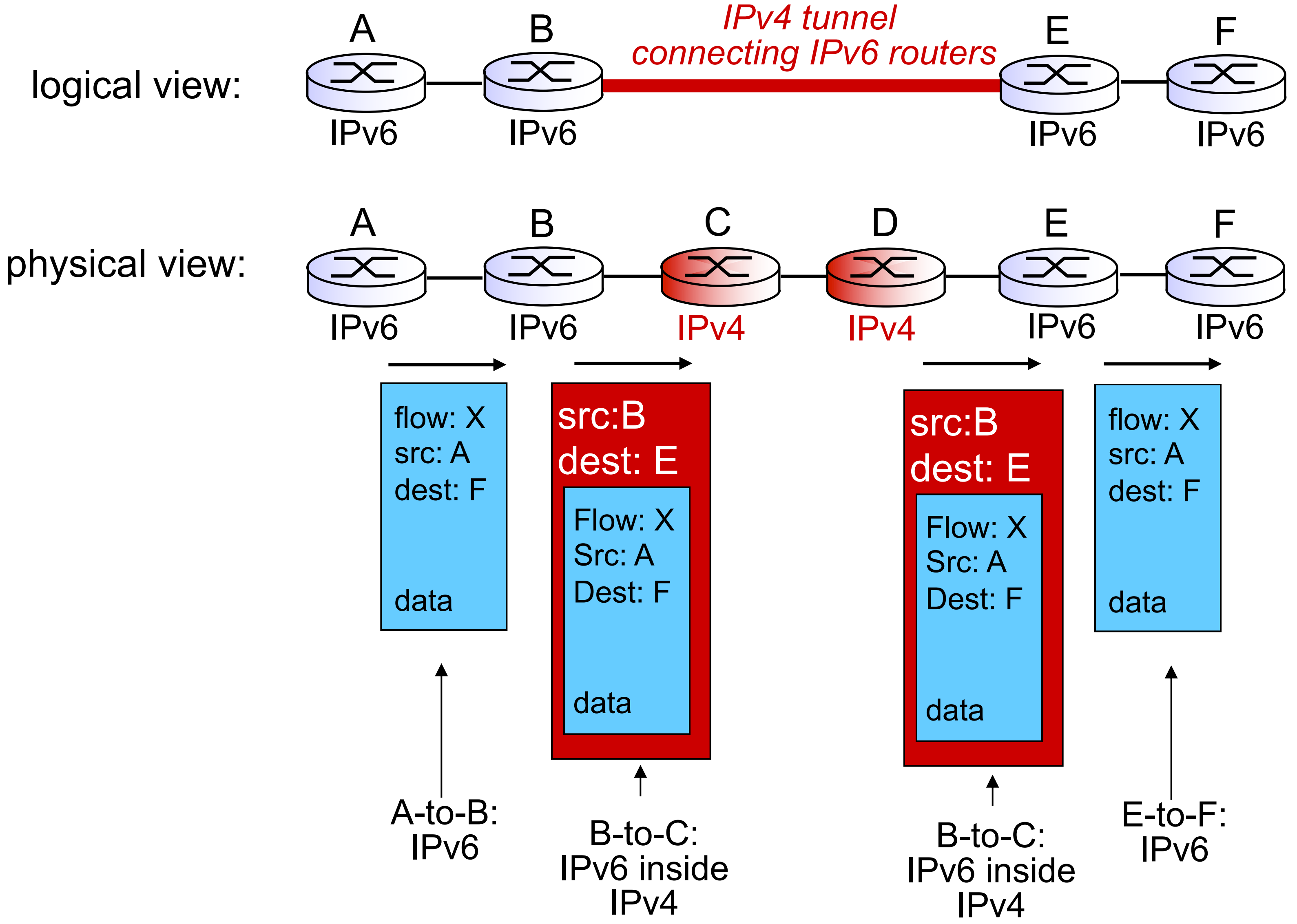
- Option 2: Slow transition
  - Some hosts/routers speak both versions
  - Must have some way to deal with those who don’t
  - Lack of incentive to switch

Tunneling

- IPv6 datagram carried as *payload* in IPv4 datagram among IPv4 routers



Tunneling


logical view:
A
B
IPv4 tunnel
connecting IPv6 routers
E
F
IPv6
IPv6
IPv6
IPv6
physical view:
A
B
C
D
E
F
IPv6
IPv6
IPv4
IPv4
IPv6
IPv6
flow: X
src: A
dest: F
data
src:B
dest: E
Flow: X
Src: A
Dest: F
data
src:B
dest: E
Flow: X
Src: A
Dest: F
data
flow: X
src: A
dest: F
data
A-to-B:
IPv6
B-to-C:
IPv6 inside
IPv4
B-to-C:
IPv6 inside
IPv4
E-to-F:
IPv6

ICMP: Internet Control Message Protocol

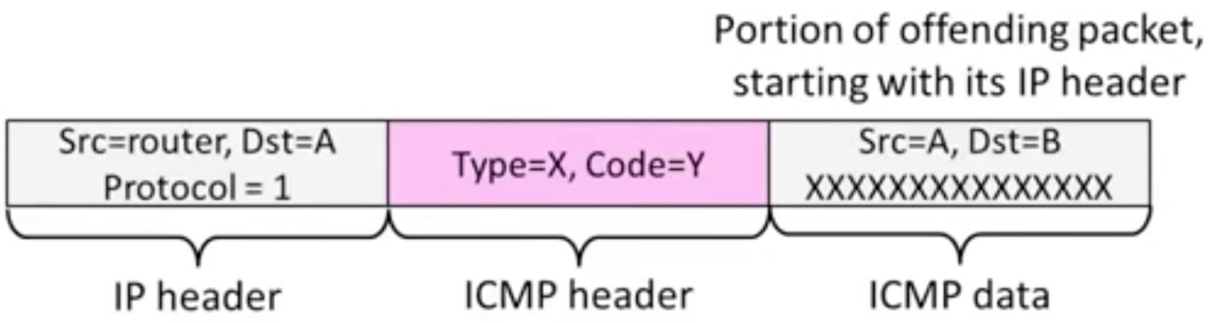
- Service Model
  - Reporting message: self-contained message reporting error
  - Unreliable: Simple datagram service – no retries.

ICMP: Internet Control Message Protocol

- Used to communicate network information
  - "Control messages", i.e., not data themselves
  - Error reporting
    - Unreachable host
    - Unreachable network
    - Unreachable port
    - TTL expired
  - Test connectivity
    - Echo request/response (ping)

ICMP: Internet Control Message Protocol

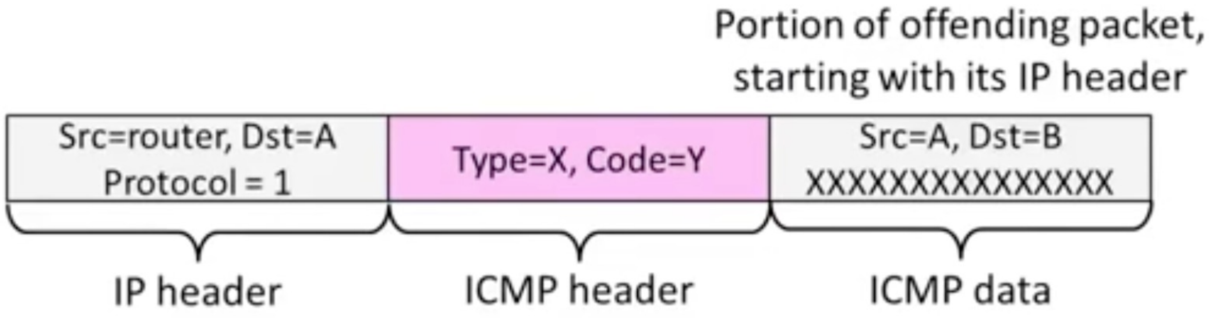
- Header:
  - 1-byte type
  - 1-byte code
  - 2-byte checksum
  - 4 bytes vary by type
- Sits above IP
  - Type 1 in IP header
  - Usually considered part of IP



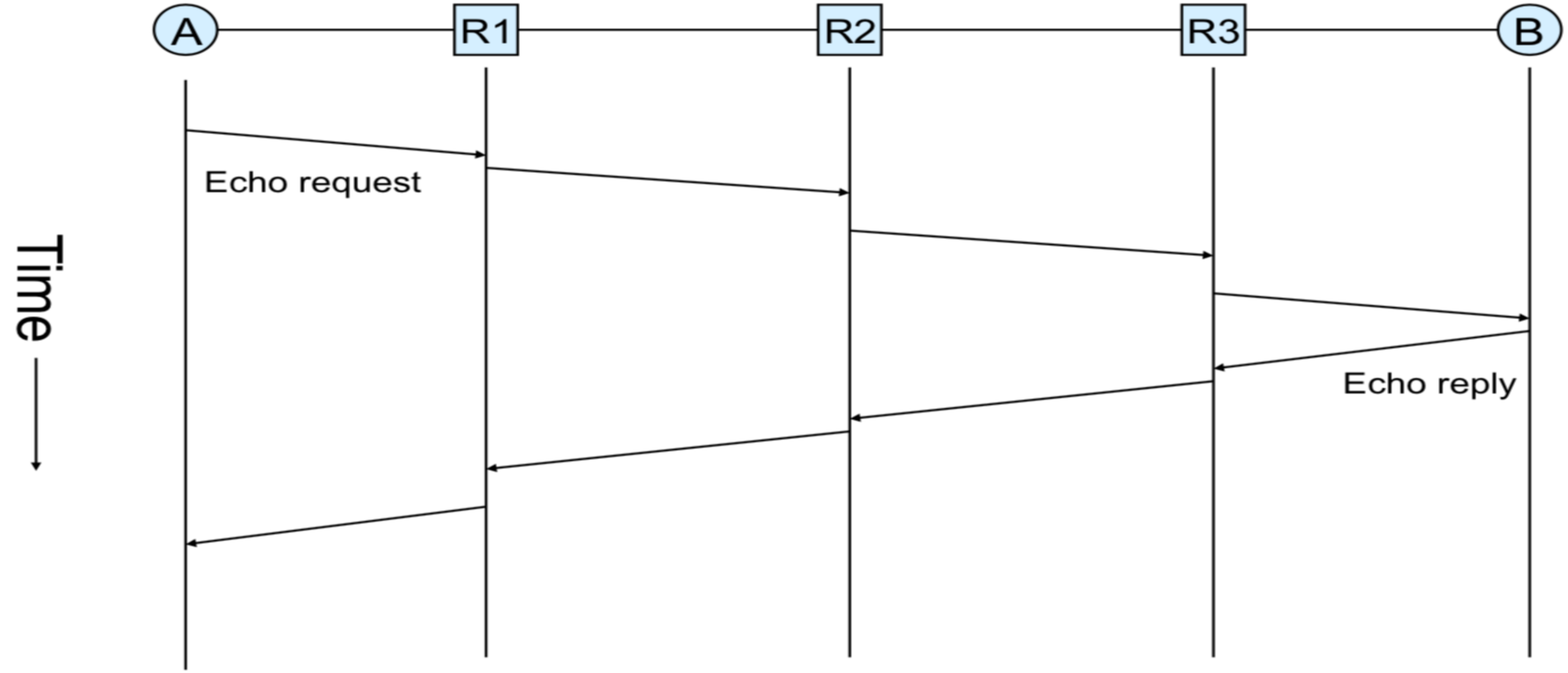
ICMP: Internet Control Message Protocol

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

Type	Code	Description
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

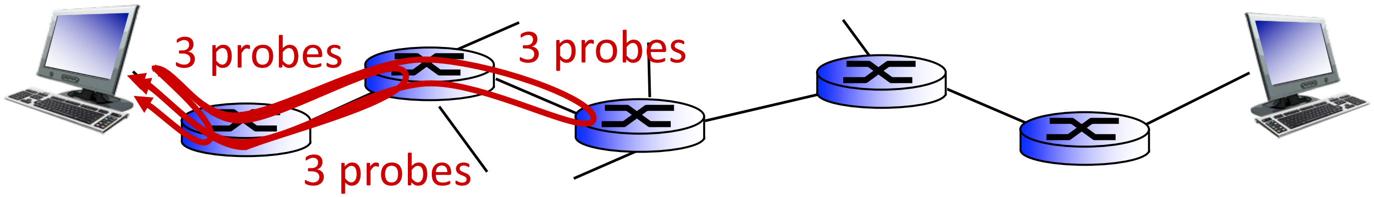


Ping



Traceroute and ICMP

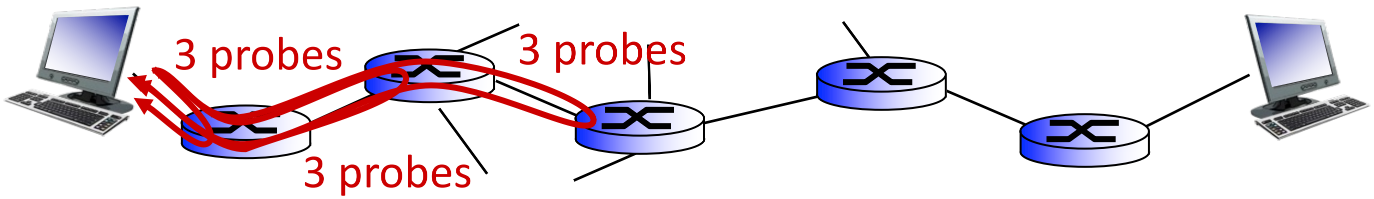
- Source sends sets of UDP segments (usually 3) to dest
  - first set has TTL =1
  - second set has TTL=2, etc.
  - unlikely port number
- When *n*th set of datagrams arrives to nth router:
  - router discards datagrams
  - and sends source ICMP messages (type 11, code 0)
  - ICMP messages includes name of router & IP address
- When ICMP messages arrives, source records RTTs



Traceroute and ICMP

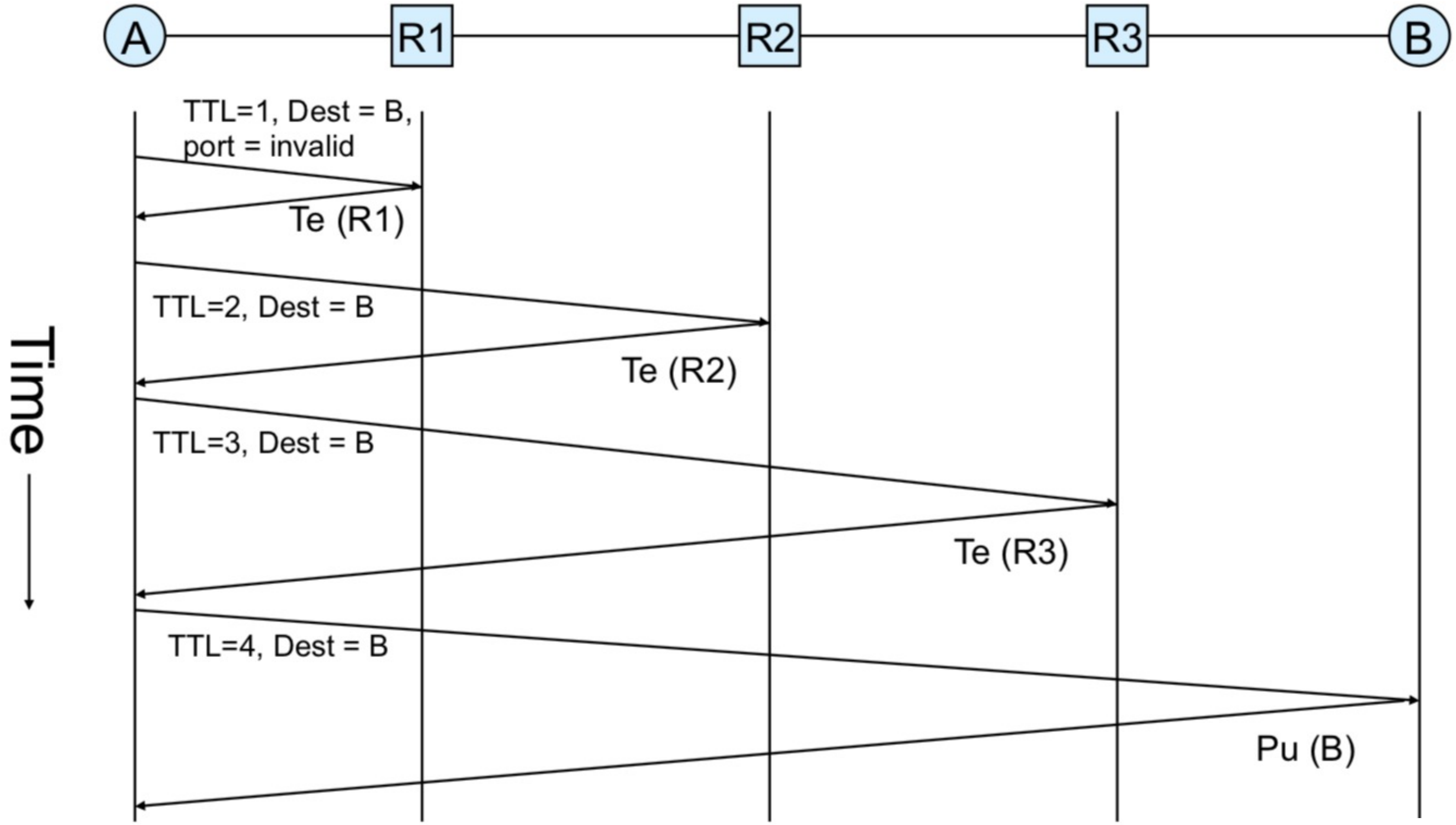
stopping criteria:

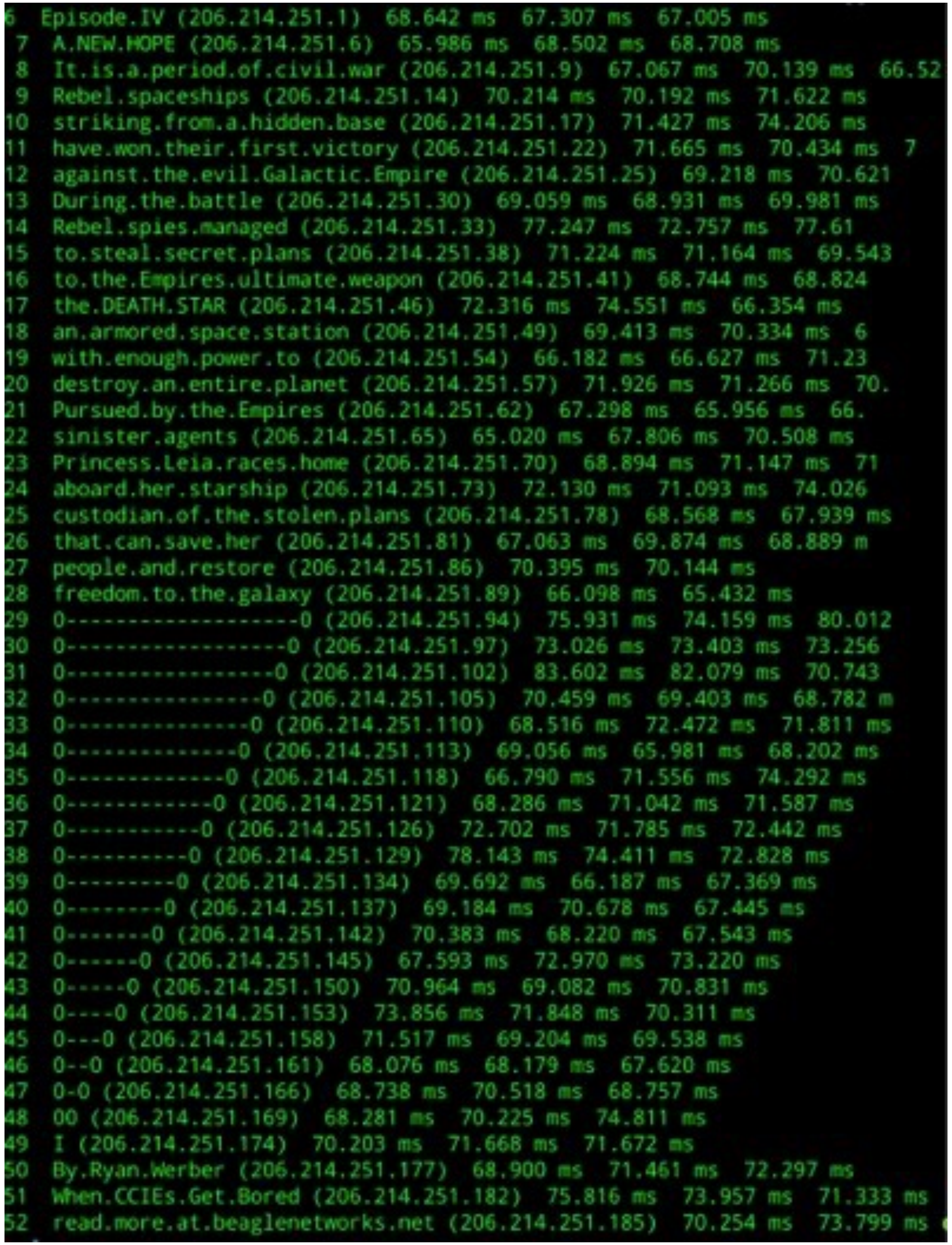
- UDP segment eventually arrives at destination host
- destination returns ICMP "port unreachable" message (type 3, code 3)
- source stops



Traceroute Demo

Te = Time exceeded
Pu = Port unreachable





```
 6  Episode.IV (206.214.251.1)  68.642 ms  67.307 ms  67.005 ms
 7  A.NEW.HOPE (206.214.251.6)  65.986 ms  68.502 ms  68.708 ms
 8  It.is.a.period.of.civil.war (206.214.251.9)  67.067 ms  70.139 ms  66.52
 9  Rebel.spaceships (206.214.251.14)  70.214 ms  70.192 ms  71.622 ms
10  striking.from.a.hidden.base (206.214.251.17)  71.427 ms  74.206 ms
11  have.won.their.first.victory (206.214.251.22)  71.665 ms  70.434 ms  7
12  against.the.evil.Galactic.Empire (206.214.251.25)  69.218 ms  70.621
13  During.the.battle (206.214.251.30)  69.059 ms  68.931 ms  69.981 ms
14  Rebel.spies.managed (206.214.251.33)  77.247 ms  72.757 ms  77.61
15  to.steal.secret.plans (206.214.251.38)  71.224 ms  71.164 ms  69.543
16  to.the.Empires.ultimate.weapon (206.214.251.41)  68.744 ms  68.824
17  the.DEATH.STAR (206.214.251.46)  72.316 ms  74.551 ms  66.354 ms
18  an.armored.space.station (206.214.251.49)  69.413 ms  70.334 ms  6
19  with.enough.power.to (206.214.251.54)  66.182 ms  66.627 ms  71.23
20  destroy.an.entire.planet (206.214.251.57)  71.926 ms  71.266 ms  70.
21  Pursued.by.the.Empires (206.214.251.62)  67.298 ms  65.956 ms  66.
22  sinister.agents (206.214.251.65)  65.020 ms  67.806 ms  70.508 ms
23  Princess.Leia.races.home (206.214.251.70)  68.894 ms  71.147 ms  71
24  aboard.her.starship (206.214.251.73)  72.130 ms  71.093 ms  74.026
25  custodian.of.the.stolen.plans (206.214.251.78)  68.568 ms  67.939 ms
26  that.can.save.her (206.214.251.81)  67.063 ms  69.874 ms  68.889 m
27  people.and.restore (206.214.251.86)  70.395 ms  70.144 ms
28  freedom.to.the.galaxy (206.214.251.89)  66.098 ms  65.432 ms
29  0-------------------0 (206.214.251.94)  75.931 ms  74.159 ms  80.012
30  0------------------0 (206.214.251.97)  73.026 ms  73.403 ms  73.256
31  0-----------------0 (206.214.251.102)  83.602 ms  82.079 ms  70.743
32  0----------------0 (206.214.251.105)  70.459 ms  69.403 ms  68.782 m
33  0---------------0 (206.214.251.110)  68.516 ms  72.472 ms  71.811 ms
34  0--------------0 (206.214.251.113)  69.056 ms  65.981 ms  68.202 ms
35  0-------------0 (206.214.251.118)  66.790 ms  71.556 ms  74.292 ms
36  0------------0 (206.214.251.121)  68.286 ms  71.042 ms  71.587 ms
37  0-----------0 (206.214.251.126)  72.702 ms  71.785 ms  72.442 ms
38  0----------0 (206.214.251.129)  78.143 ms  74.411 ms  72.828 ms
39  0---------0 (206.214.251.134)  69.692 ms  66.187 ms  67.369 ms
40  0--------0 (206.214.251.137)  69.184 ms  70.678 ms  67.445 ms
41  0-------0 (206.214.251.142)  70.383 ms  68.220 ms  67.543 ms
42  0------0 (206.214.251.145)  67.593 ms  72.970 ms  73.220 ms
43  0-----0 (206.214.251.150)  70.964 ms  69.082 ms  70.831 ms
44  0----0 (206.214.251.153)  73.856 ms  71.848 ms  70.311 ms
45  0---0 (206.214.251.158)  71.517 ms  69.204 ms  69.538 ms
46  0--0 (206.214.251.161)  68.076 ms  68.179 ms  67.620 ms
47  0-0 (206.214.251.166)  68.738 ms  70.518 ms  68.757 ms
48  00 (206.214.251.169)  68.281 ms  70.225 ms  74.811 ms
49  I (206.214.251.174)  70.203 ms  71.668 ms  71.672 ms
50  By.Ryan.Werber (206.214.251.177)  68.900 ms  71.461 ms  72.297 ms
51  When.CCIEs.Get.Bored (206.214.251.182)  75.816 ms  73.957 ms  71.333 ms
52  read.more.at.beaglenetworks.net (206.214.251.185)  70.254 ms  73.799 ms
```