Slideset version edited by **Marco Savi**

Chapter 4 Network Layer: Data Plane

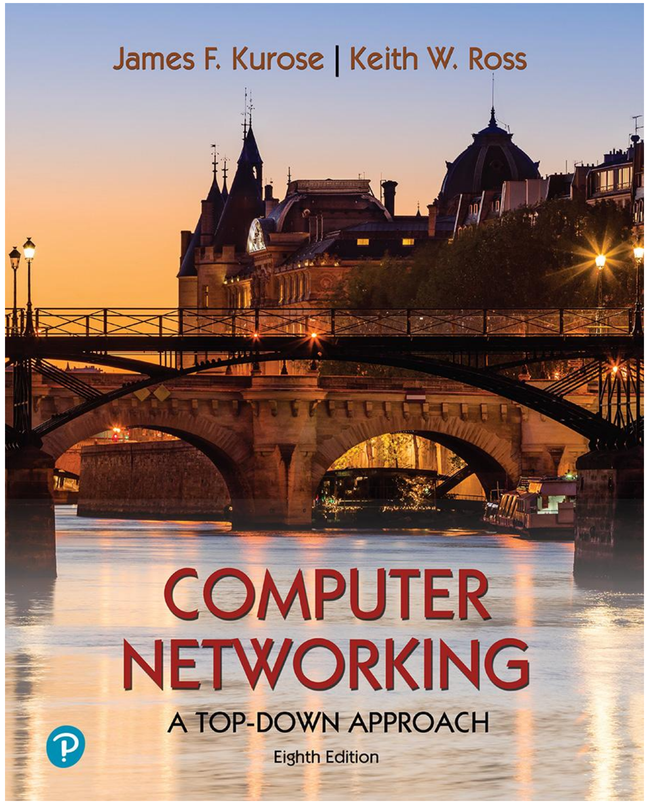
A note on the use of these PowerPoint slides:

We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you see the animations; and can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part. In return for use, we only ask the following:

- If you use these slides (e.g., in a class) that you mention their source (after all, we'd like people to use our book!)
- If you post any slides on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

For a revision history, see the slide note for this page.

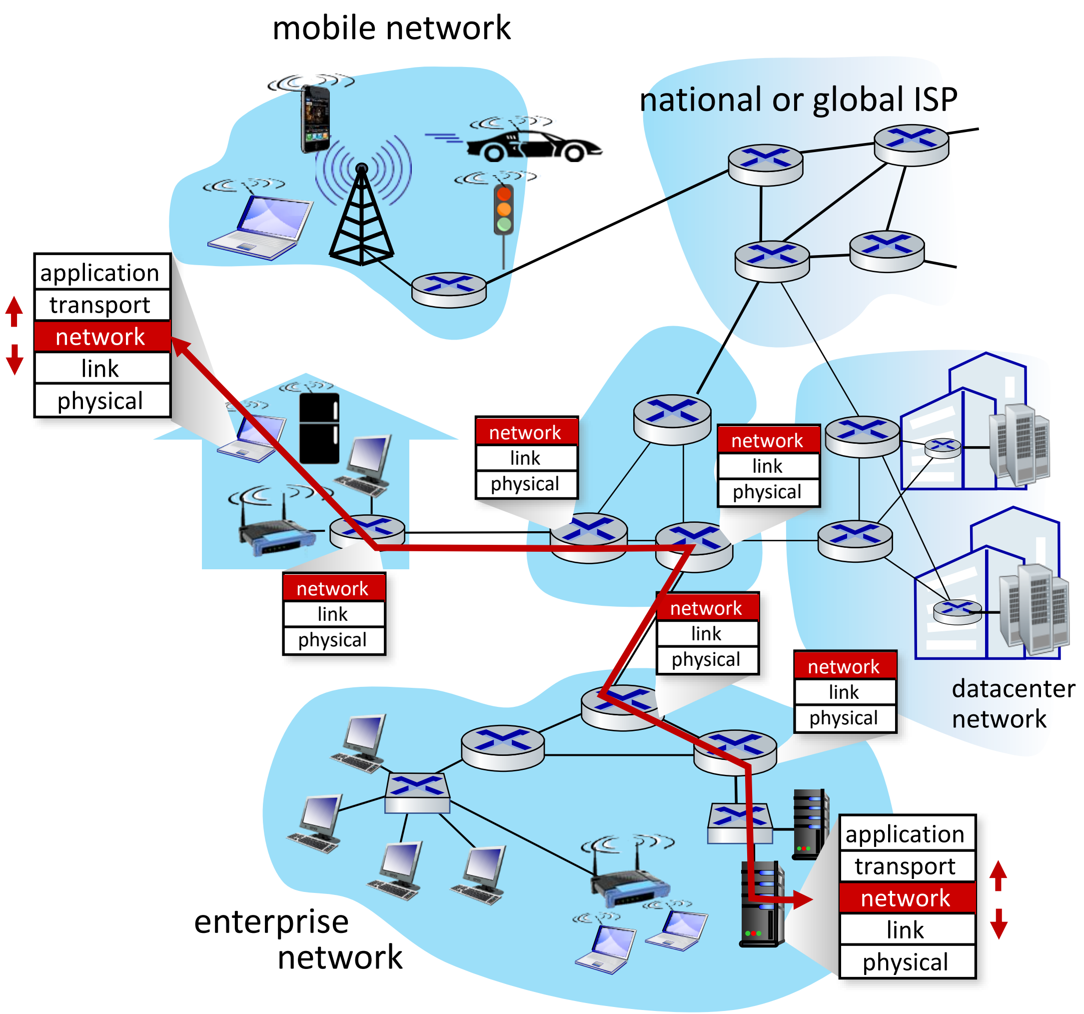
Thanks and enjoy! JFK/KWR



Computer Networking: A Top-Down Approach
8th edition
Jim Kurose, Keith Ross
Pearson, 2020

Network-layer services and protocols

- transport segment from sending to receiving host
  - sender: encapsulates segments into datagrams, passes to link layer
  - receiver: decapsulates segments from datagrams, delivers them to transport layer
- network layer protocols in *every Internet device*: hosts, routers
- routers:
  - examines header fields in all datagrams passing through it
  - moves datagrams from input ports to output ports to transfer datagrams along end-end path



Two key network-layer functions

network-layer functions:

- *forwarding:* move packets from a router's input link to appropriate router output link
- *routing:* determine route taken by packets from source to destination
  - *routing algorithms and protocols*

analogy: taking a trip

- *forwarding:* process of getting through single interchange
- *routing:* process of planning trip from source to destination

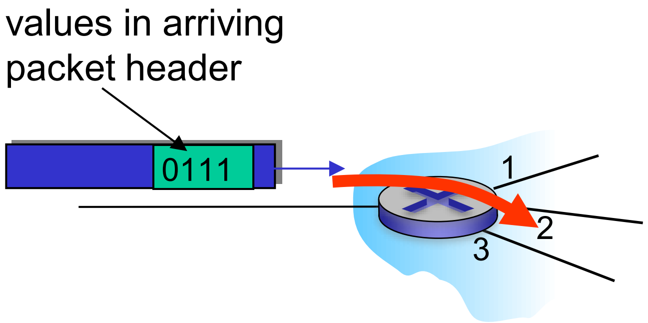
forwarding

routing

Network layer: data plane, control plane

Data plane:

- *local*, per-router function
- determines how datagram arriving on router input port is forwarded to router output port
  - implements *forwarding*



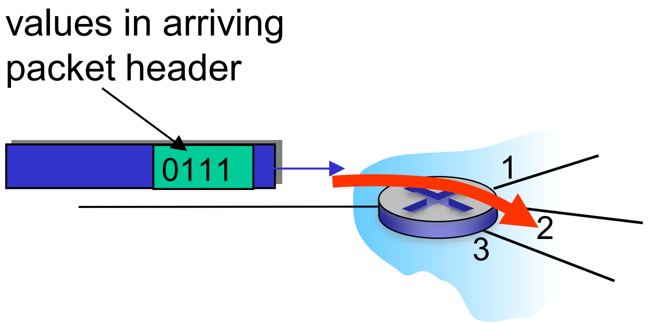
Control plane:

- *network-wide* logic
- determines how datagram is routed among routers along end-end path from source host to destination host
  - implements *routing*
- two control-plane approaches:
  - *traditional routing algorithms:* implemented in routers
  - *software-defined networking (SDN)*: implemented in (remote) servers

Network layer: data plane, control plane

Data plane:

- *local*, per-router function
- determines how datagram arriving on router input port is forwarded to router output port
  - implements *forwarding*



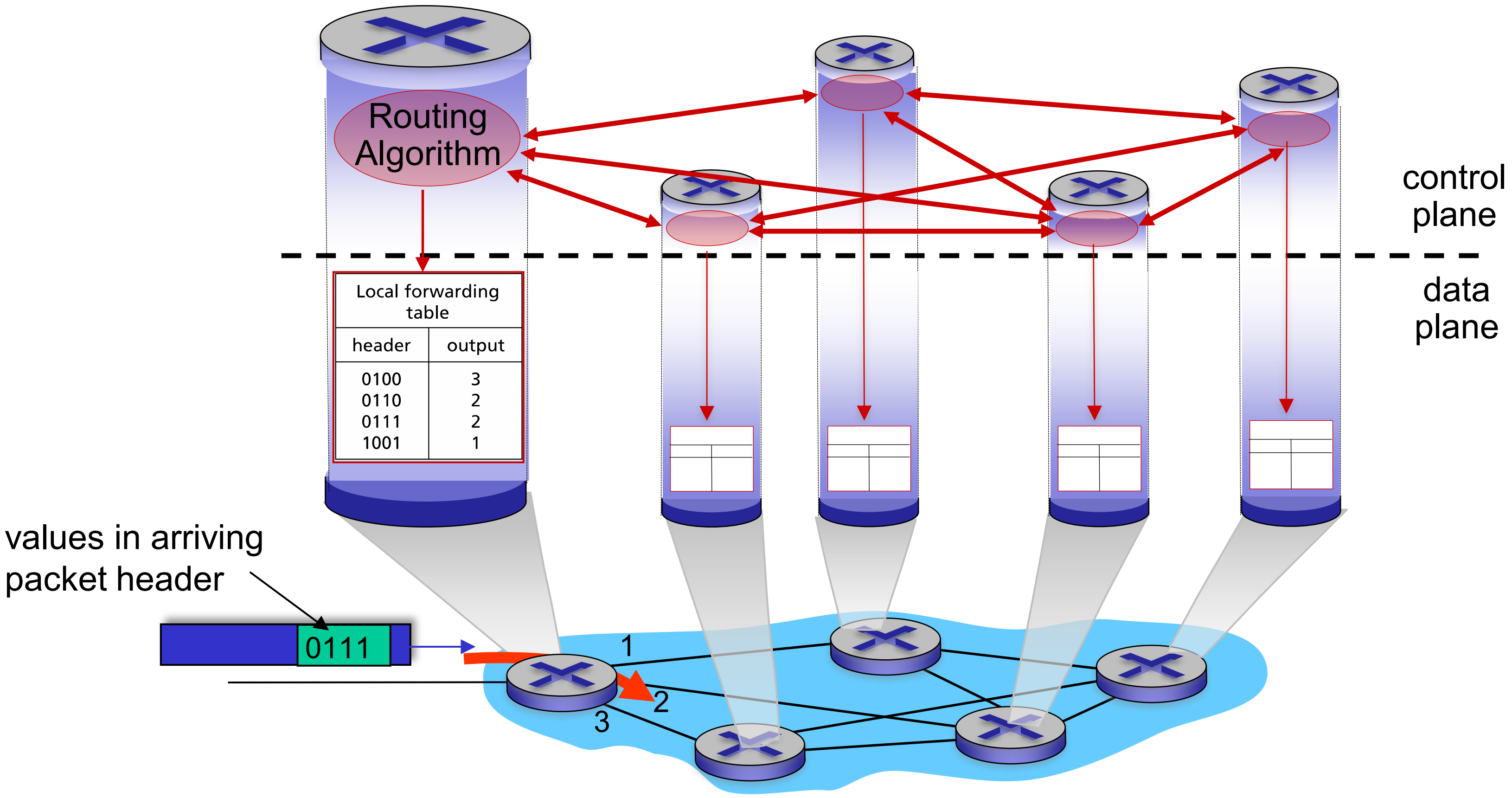
Control plane:

- *network-wide* logic
- determines how datagram is routed among routers along end-end path from source host to destination host
  - implements *routing*
- two control-plane approaches:
  - *traditional routing algorithms:* implemented in routers
  - ~~*software-defined networking (SDN)*: implemented in (remote) servers~~

out of scope

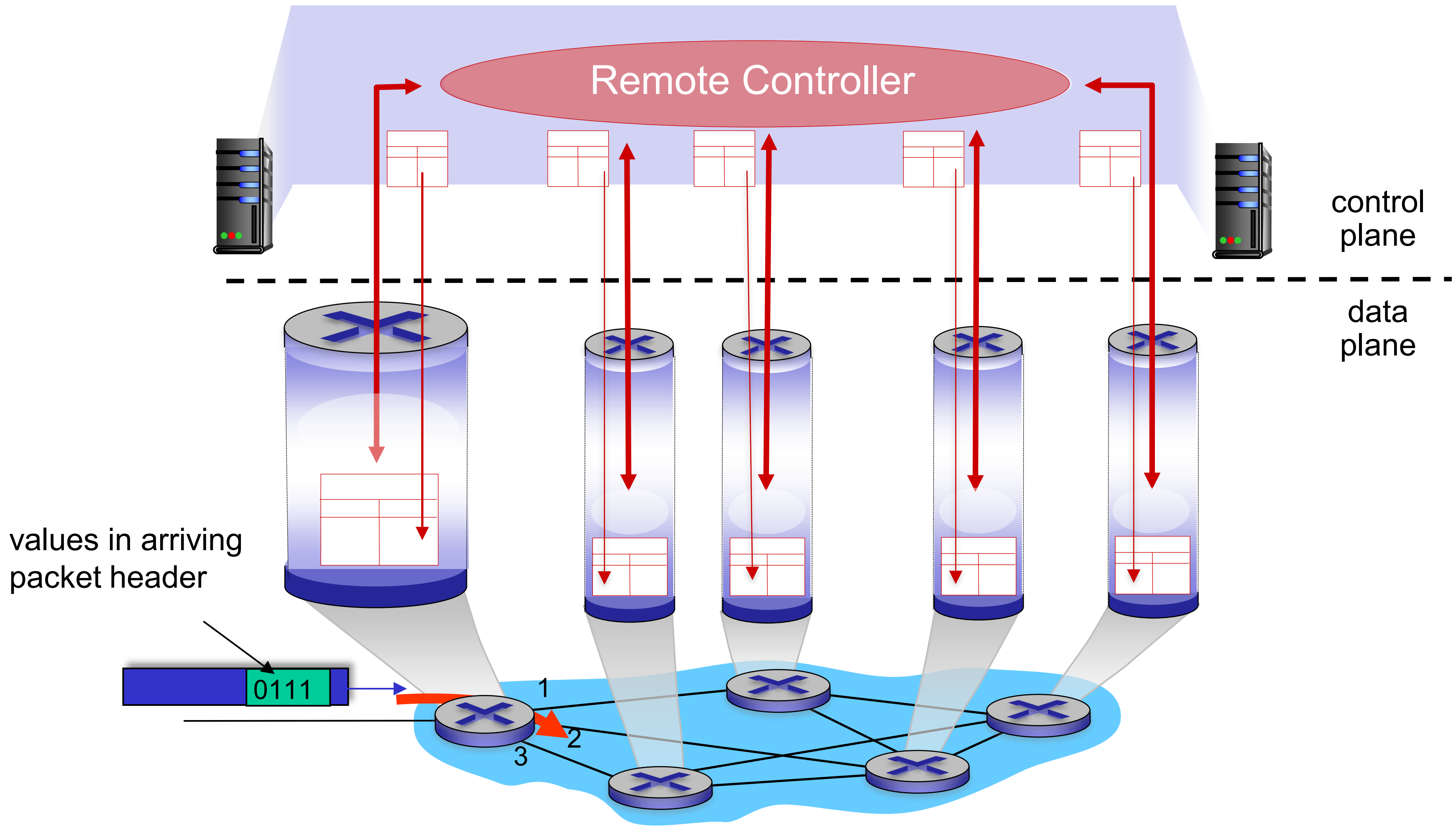
Traditional routing: per-router control plane

Individual routing algorithm components *in each and every router* interact in the control plane



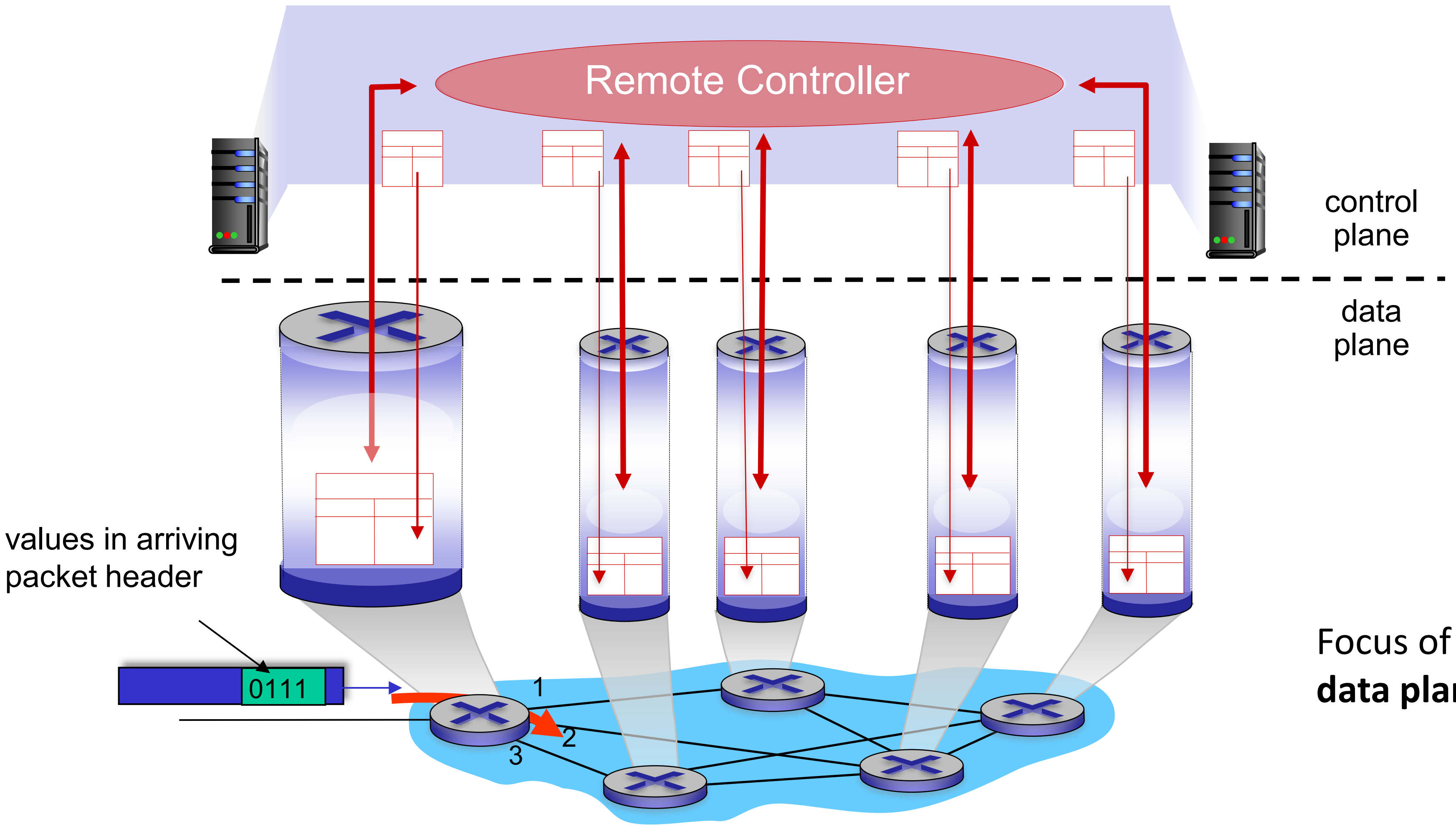
Software-Defined Networking (SDN) control plane

Remote controller computes, installs forwarding tables in routers



Software-Defined Networking (SDN) control plane

Remote controller computes, installs forwarding tables in routers



Focus of this chapter:
data plane

Network service model

Q: What *service model* for "channel" transporting datagrams from sender to receiver?

example services for *individual* datagrams:

- guaranteed delivery
- guaranteed delivery with less than 40 msec delay
- …

example services for a *flow* of datagrams:

- in-order datagram delivery
- guaranteed minimum bandwidth to flow
- …

Network-layer service model

Network Architecture	Service Model	Quality Guarantees? Loss	Order	Bandwidth
Internet	*best effort*	no	no	no

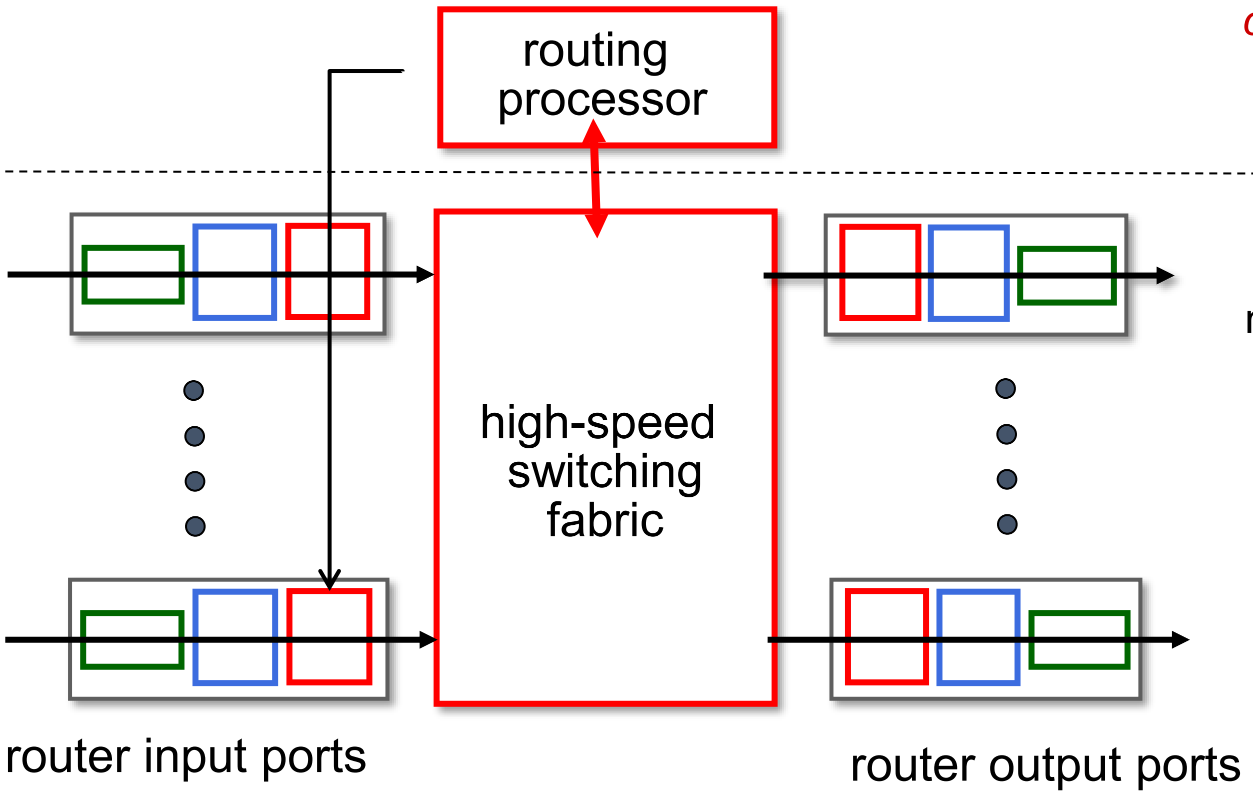
Internet "best effort" service model

No guarantees on*:*

i. successful datagram delivery to destination
ii. order of delivery
iii. bandwidth available to end-end flow

Router architecture overview

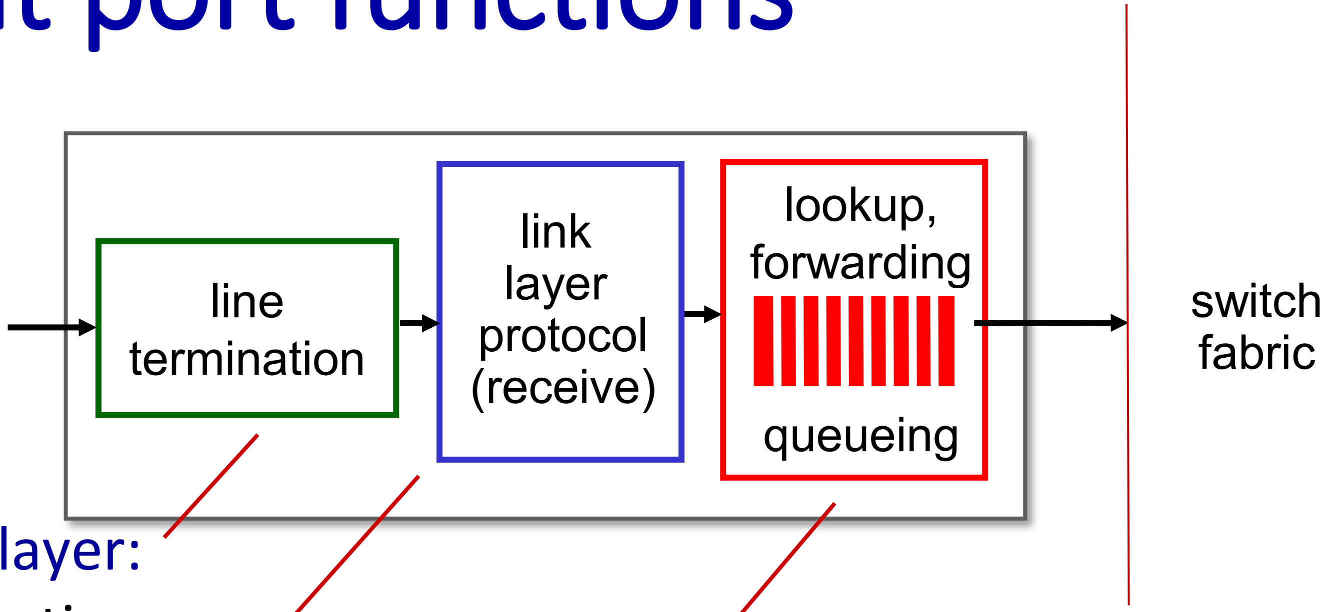
high-level view of generic router architecture:



control plane (software) operates in millisecond time frame

data plane (hardware) operates in nanosecond time frame

Input port functions



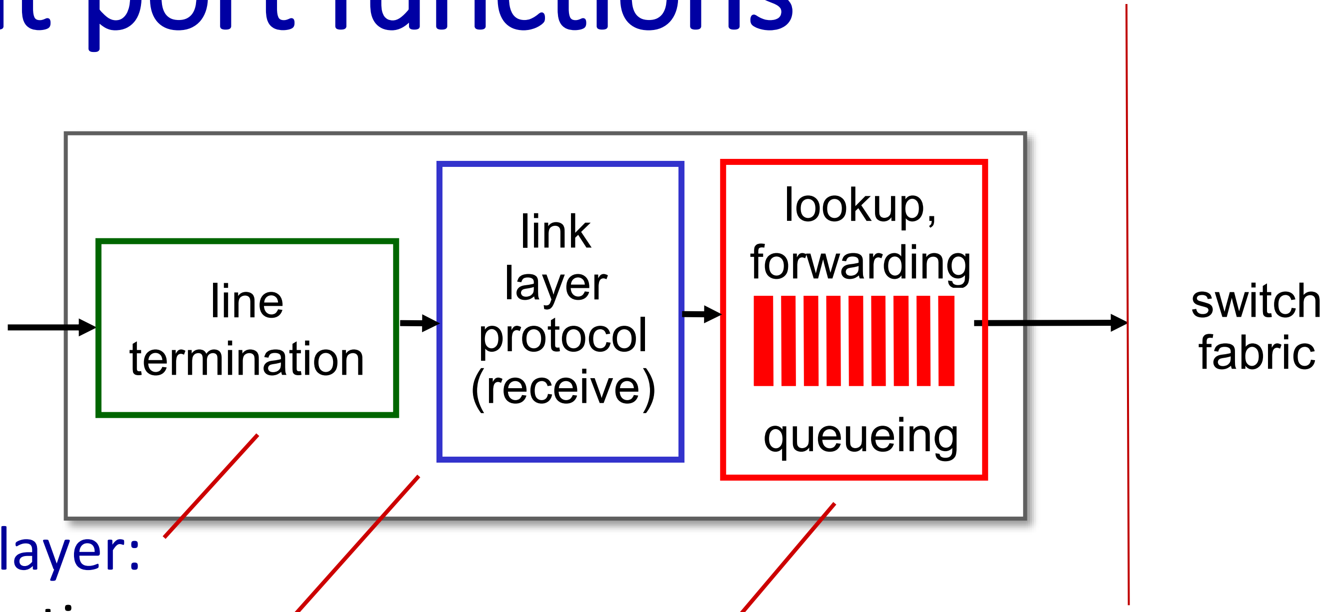
physical layer:
bit-level reception

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

decentralized switching/forwarding:

- using header field values, lookup output port using forwarding table in input port memory *("match plus action")*
  - destination-based forwarding: forward based only on destination network-layer (i.e., IP) address (traditional)
    - We will see how it works in a few slides…

Input port functions



physical layer:
bit-level reception

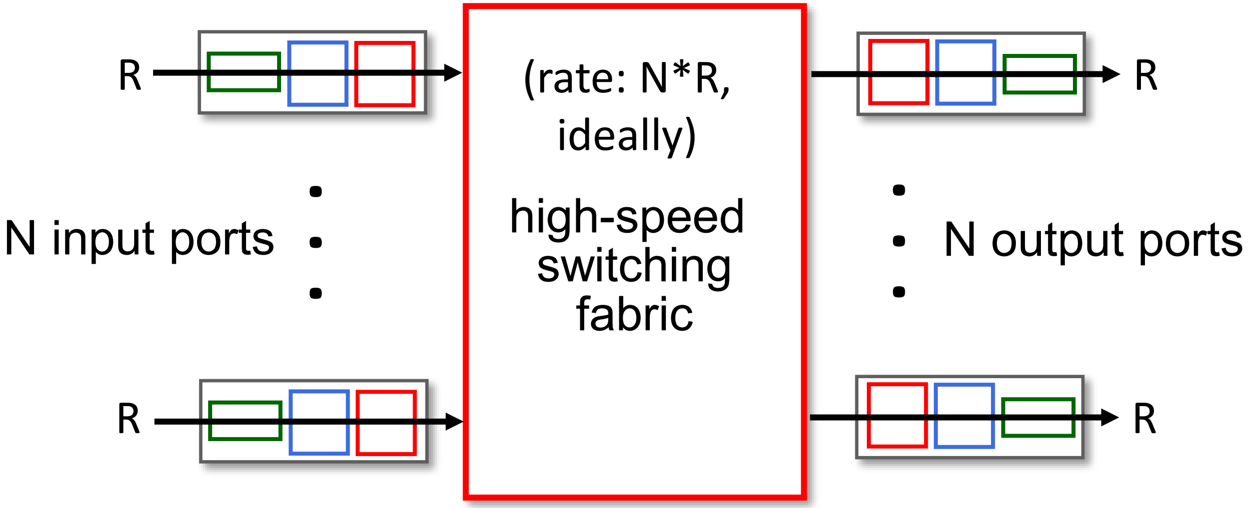
link layer:
e.g., Ethernet
(chapter 6)

decentralized switching/forwarding*:*

- using header field values, lookup output port using forwarding table in input port memory *("match plus action")*
- goal: complete input port processing in a very short time
- input port queuing: if datagrams arrive faster than *switching rate* of switching fabric

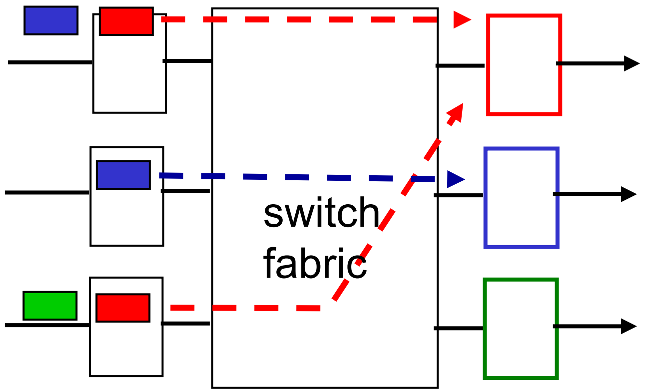
Switching fabric

- transfer packet from input port to appropriate output port
- switching rate: rate at which packets can be transferred from input ports to output ports
  - often measured as multiple of input/output port line rate (i.e., rate supported by the port)
  - N inputs: switching rate at least N times line rate R desirable → negligible input port queuing is ensured

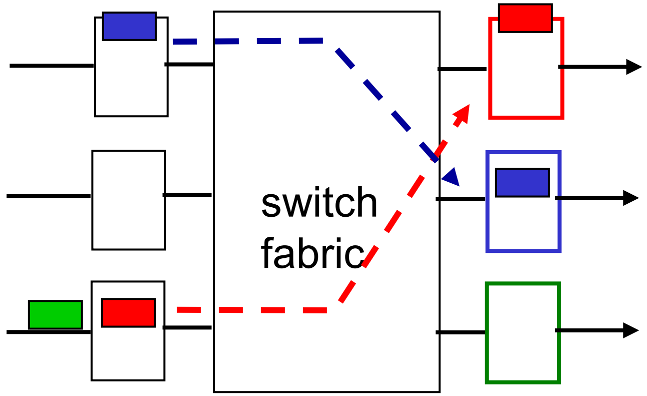


Input port queuing

- if switching rate slower than input ports combined → significant queueing may occur at input queues
  - queueing delay and loss due to input buffer overflow!
- Head-of-the-Line (HOL) blocking: queued datagram at front of queue prevents others in queue from moving forward

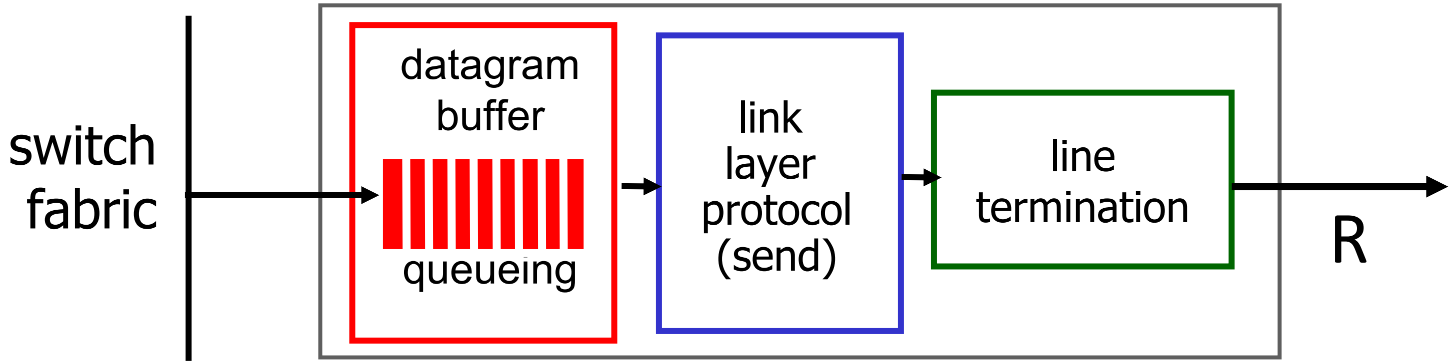


output port contention: only one red datagram can be transferred; lower red packet is *blocked*



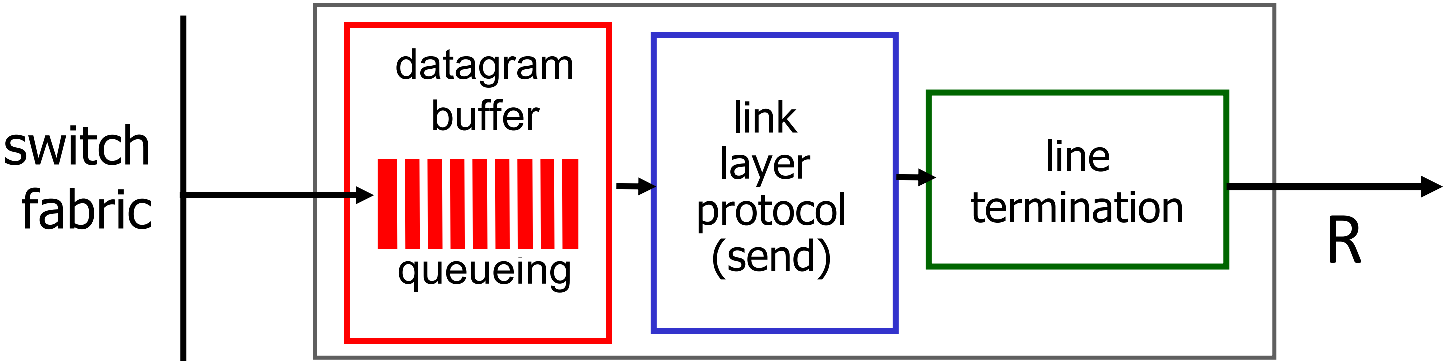
one packet time later: green packet has experienced *HOL blocking*

Output port queuing



- Buffering occurs when arrival rate from the switching fabric exceeds output line speed
  - *Queueing delay and loss due to output port buffer overflow!*

Output port queuing



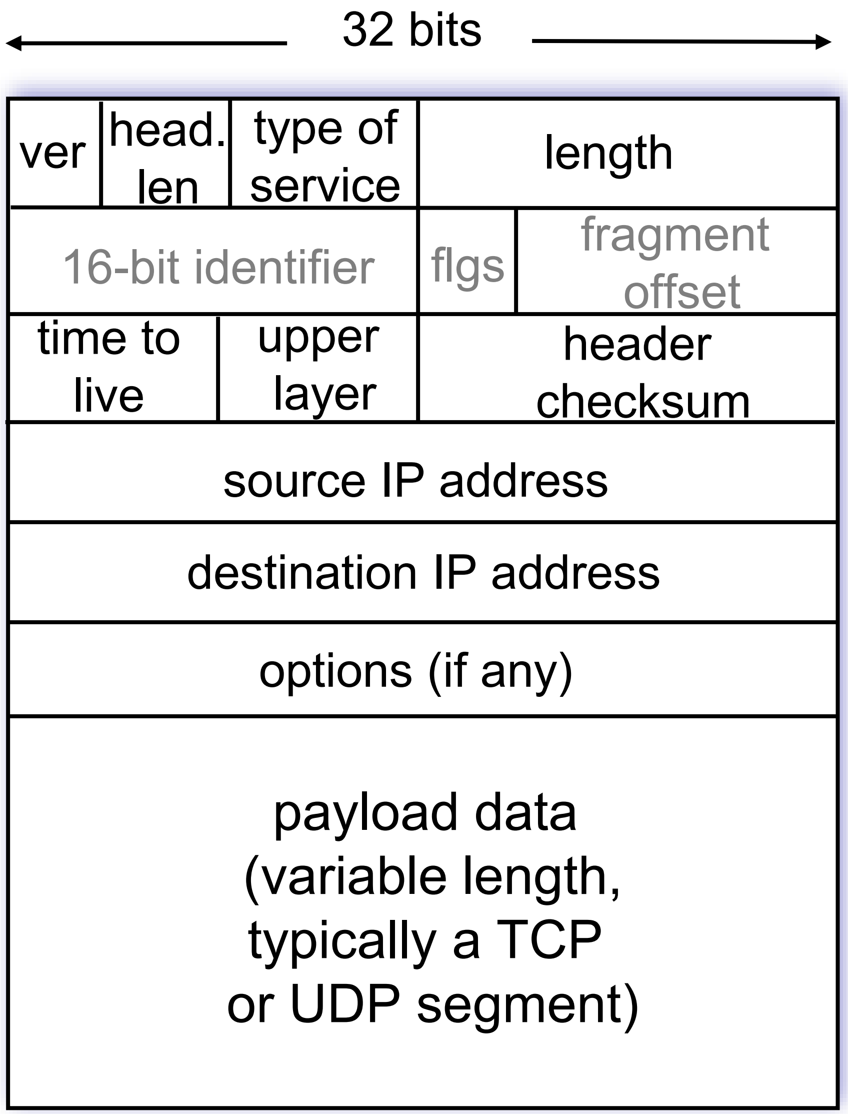
- Buffering occurs when arrival rate from the switching fabric exceeds output line speed
  - *Queueing delay and loss due to output port buffer overflow!*
- *Drop policy:* which datagrams to drop if buffers are almost full?

→ Datagrams are discarded

- *Scheduling discipline* chooses among queued datagrams for transmission

→ Priority scheduling – who gets best performance

IP Datagram format

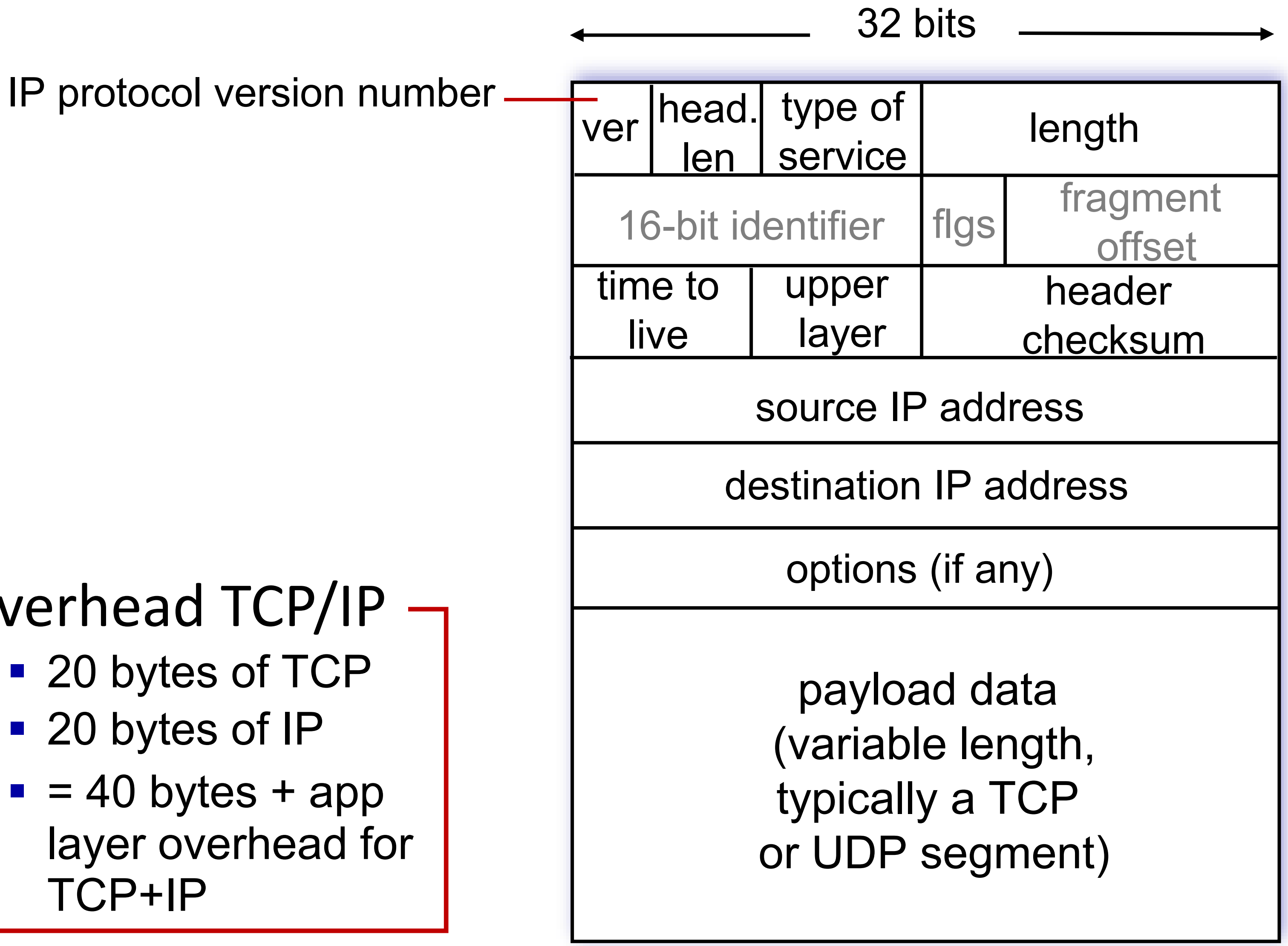


Overhead TCP/IP

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

Maximum length: **65K bytes**
Typically: **1500 bytes** (or less)

IP Datagram format

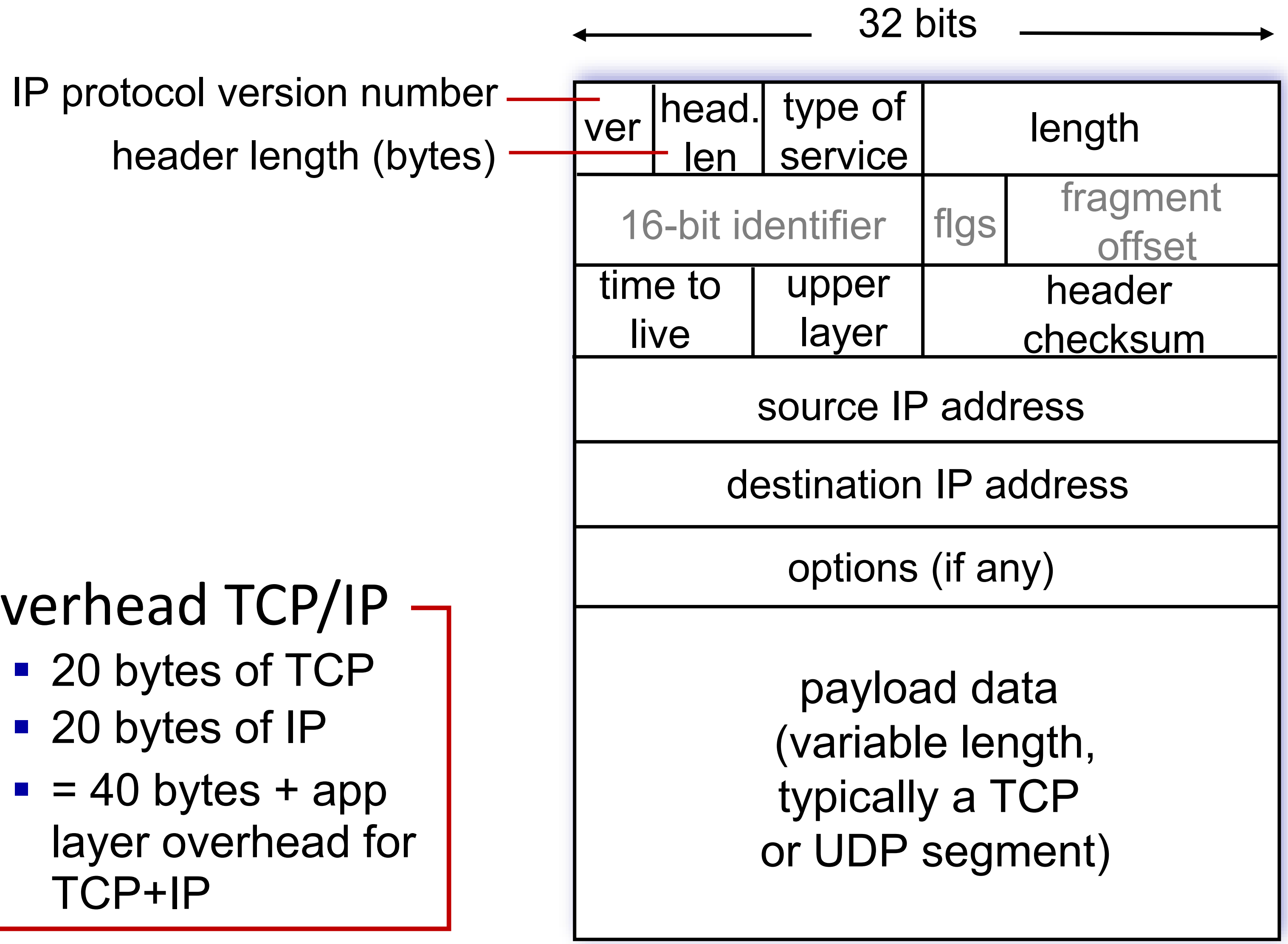


Overhead TCP/IP

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

Maximum length: **65K bytes**
Typically: **1500 bytes** (or less)

IP Datagram format

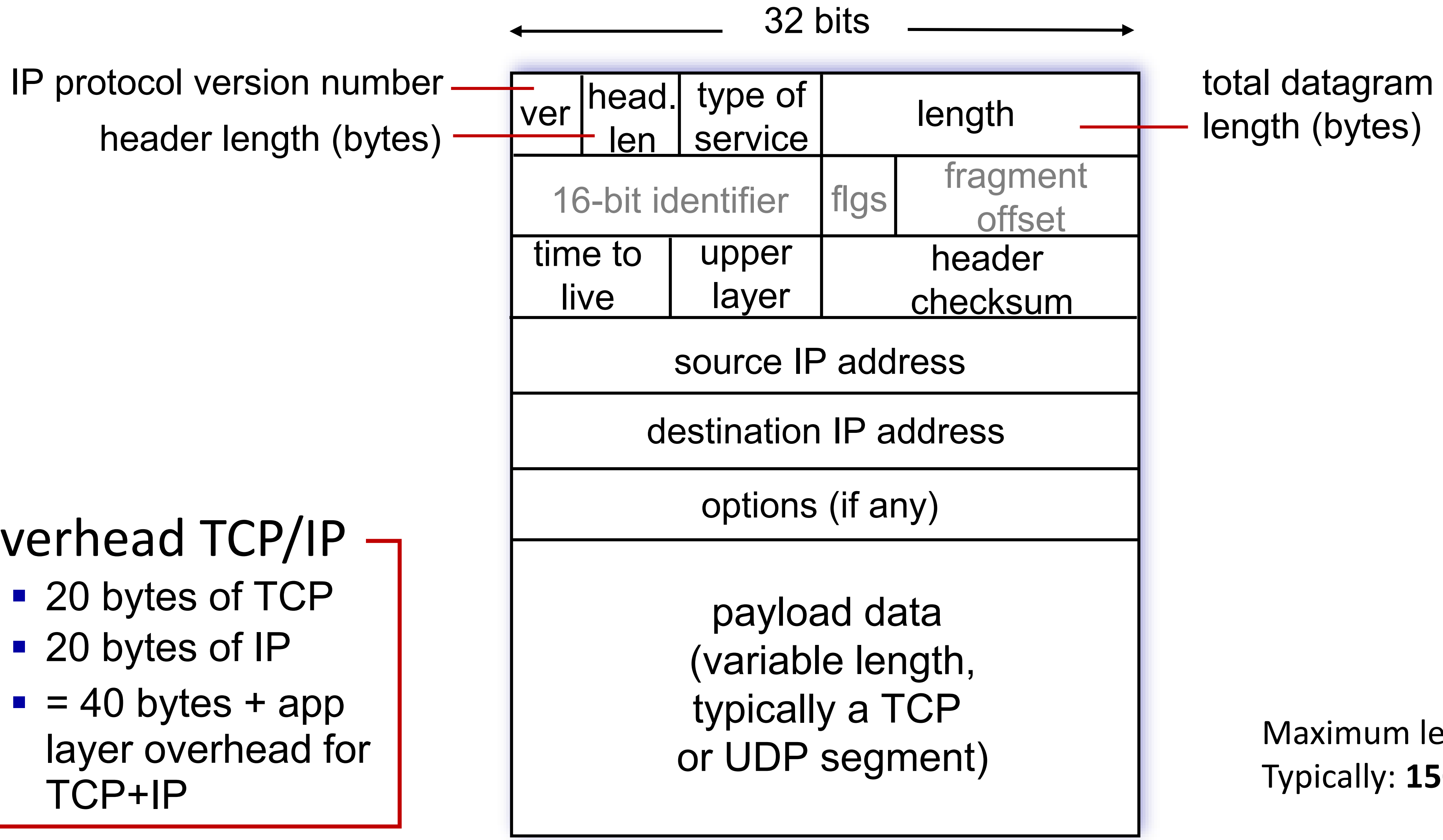


Overhead TCP/IP

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

Maximum length: **65K bytes**
Typically: **1500 bytes** (or less)

IP Datagram format

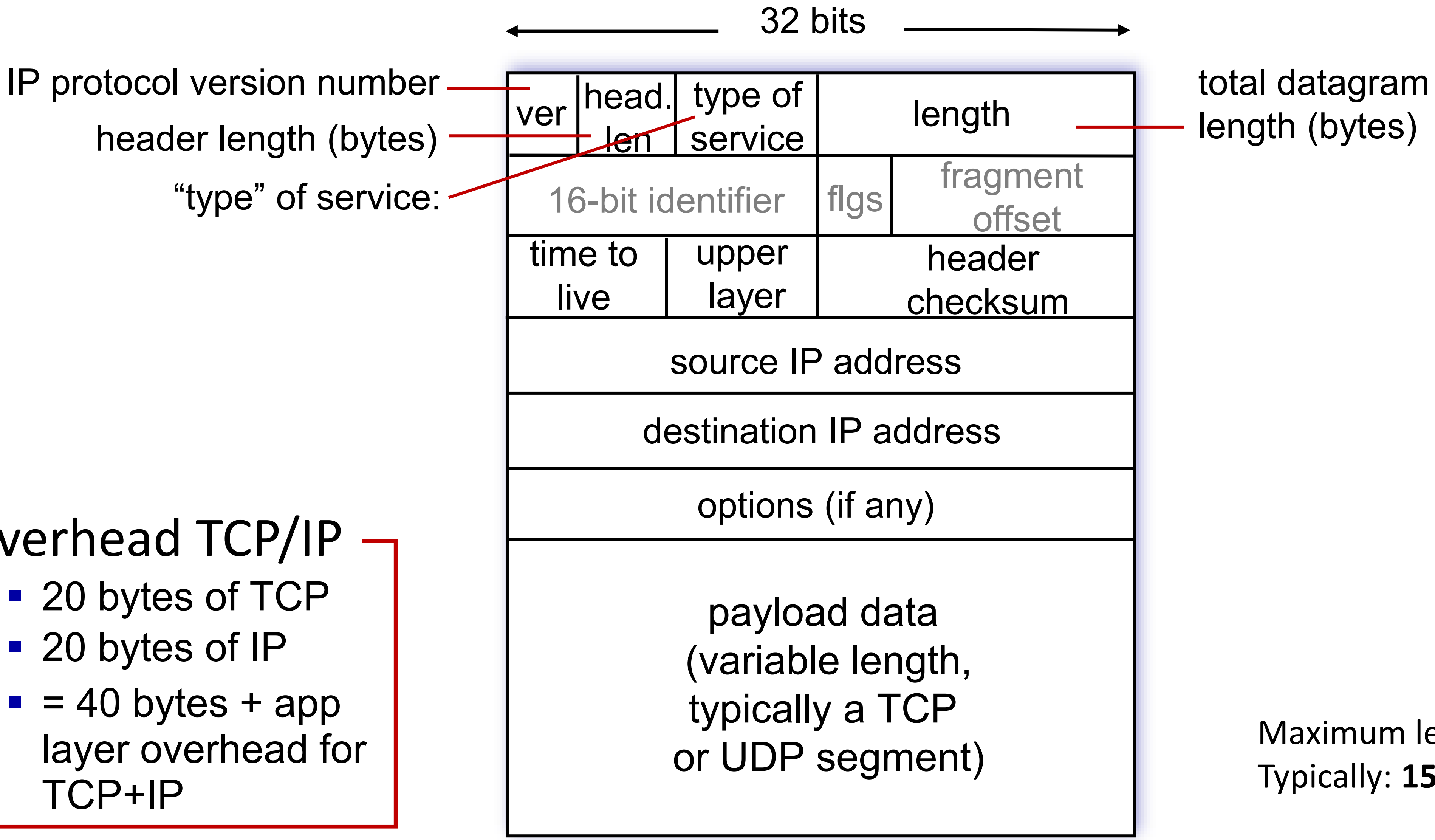


Overhead TCP/IP

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

Maximum length: **65K bytes**
Typically: **1500 bytes** (or less)

IP Datagram format



Overhead TCP/IP

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

Maximum length: **65K bytes**
Typically: **1500 bytes** (or less)

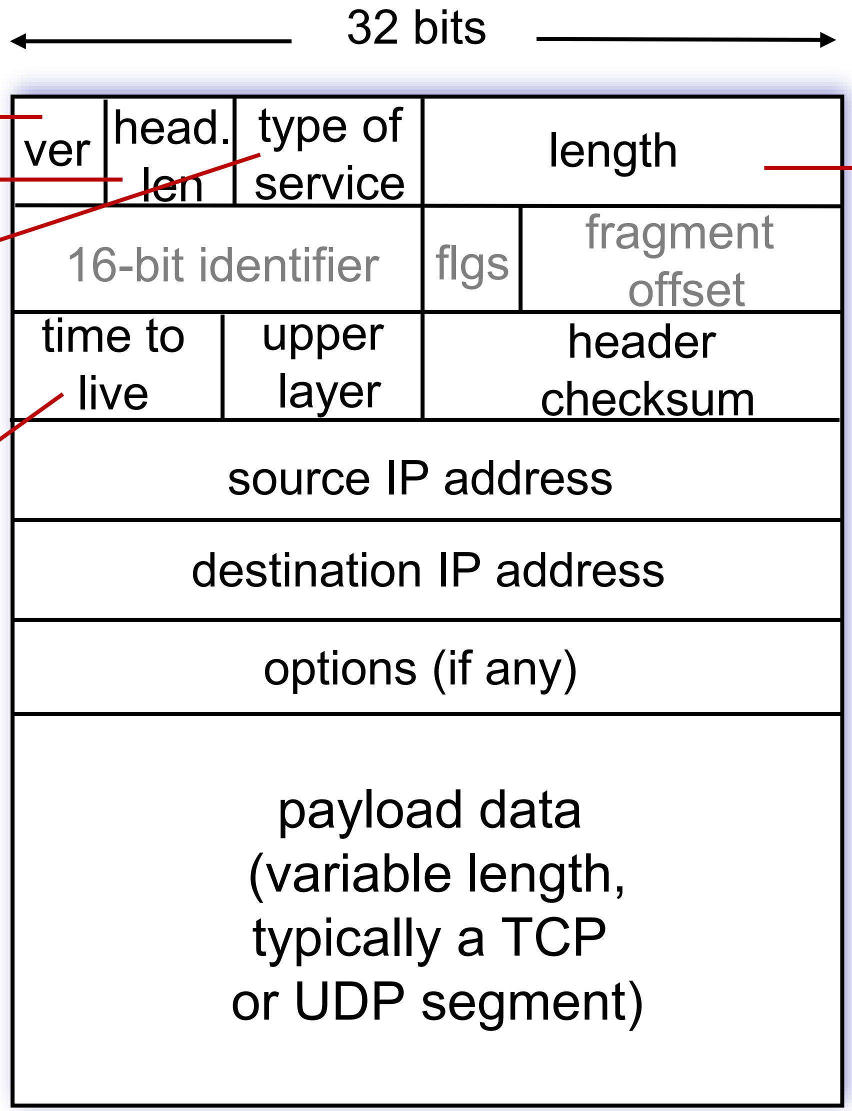
IP Datagram format

IP protocol version number

header length (bytes)

"type" of service:

TTL: remaining max hops
(decremented at each router)



total datagram length (bytes)

Overhead TCP/IP

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

Maximum length: **65K bytes**
Typically: **1500 bytes** (or less)

IP Datagram format

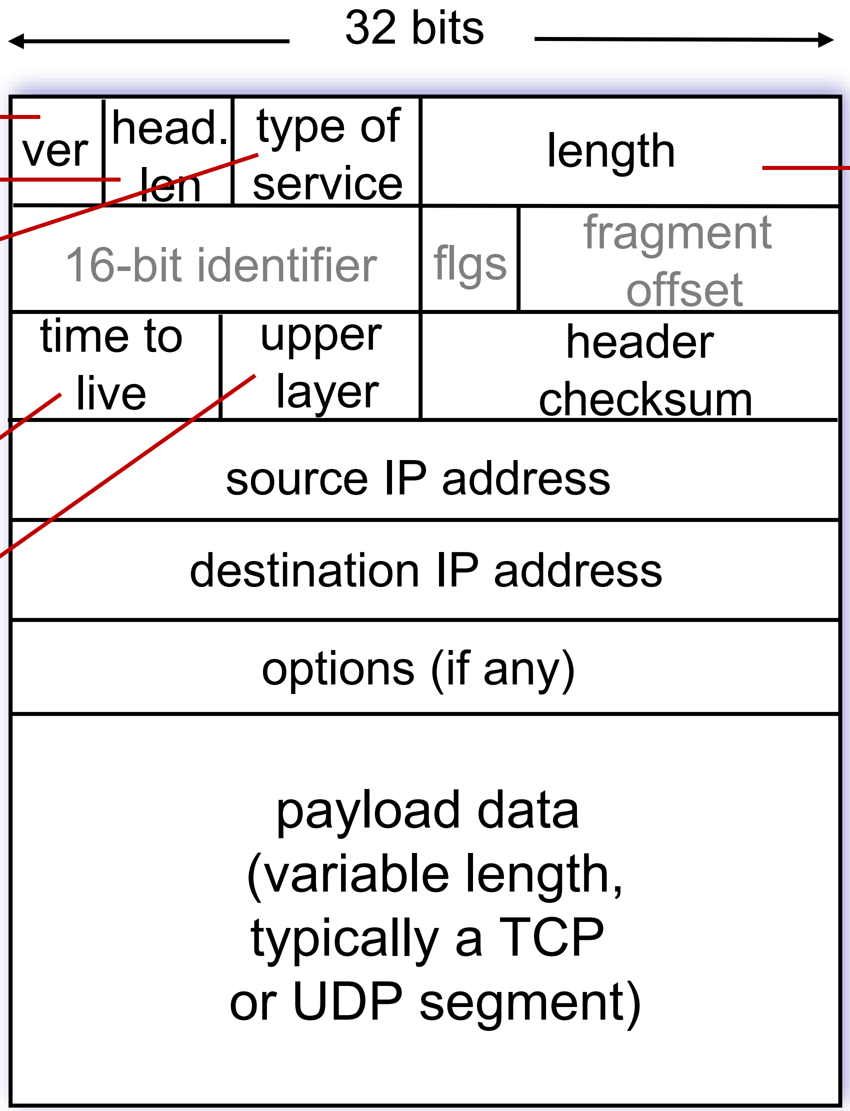
IP protocol version number

header length (bytes)

"type" of service:

TTL: remaining max hops
(decremented at each router)

upper layer protocol (e.g., TCP or UDP)



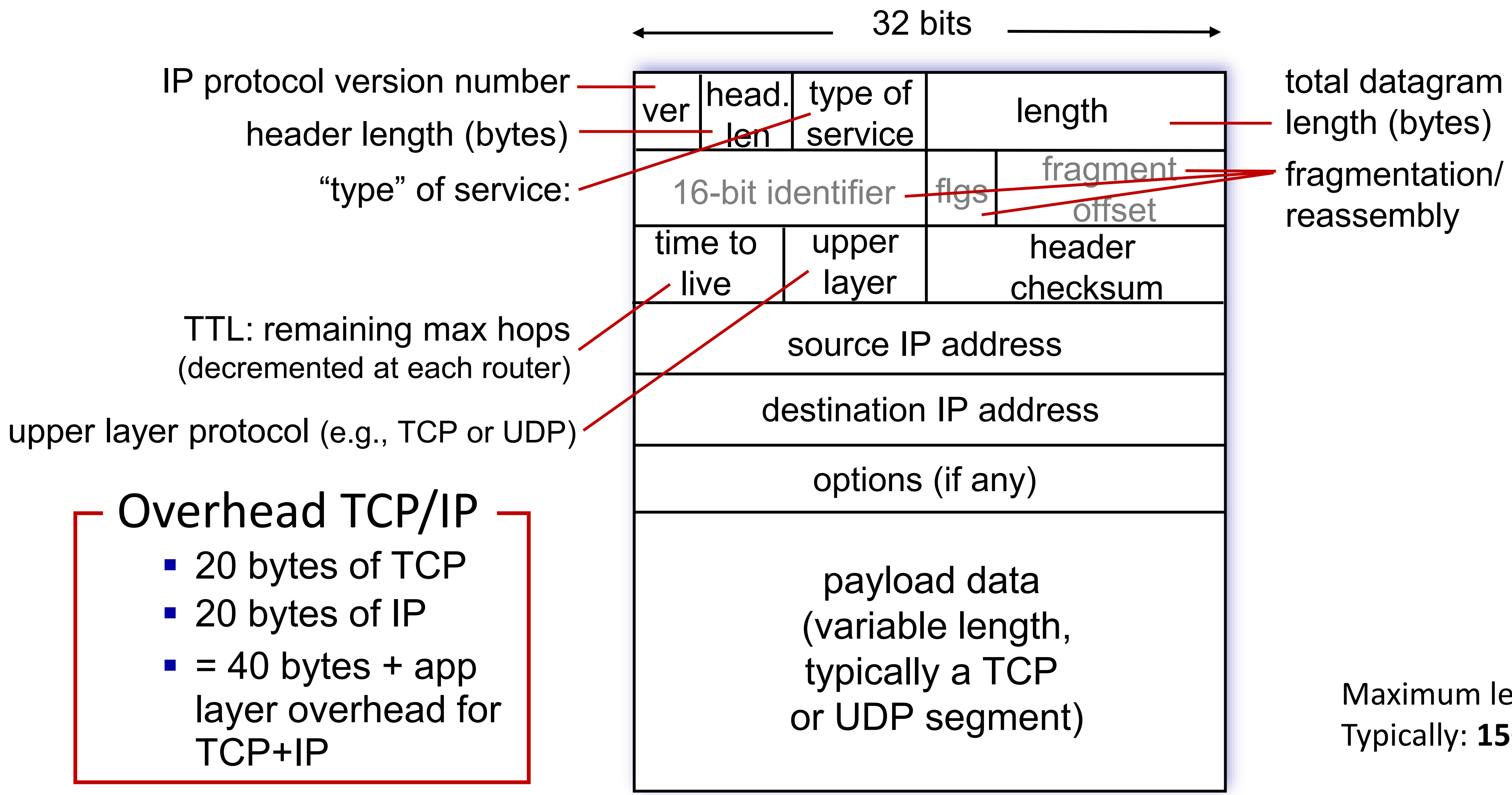
total datagram length (bytes)

Overhead TCP/IP

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

Maximum length: **65K bytes**
Typically: **1500 bytes** (or less)

IP Datagram format

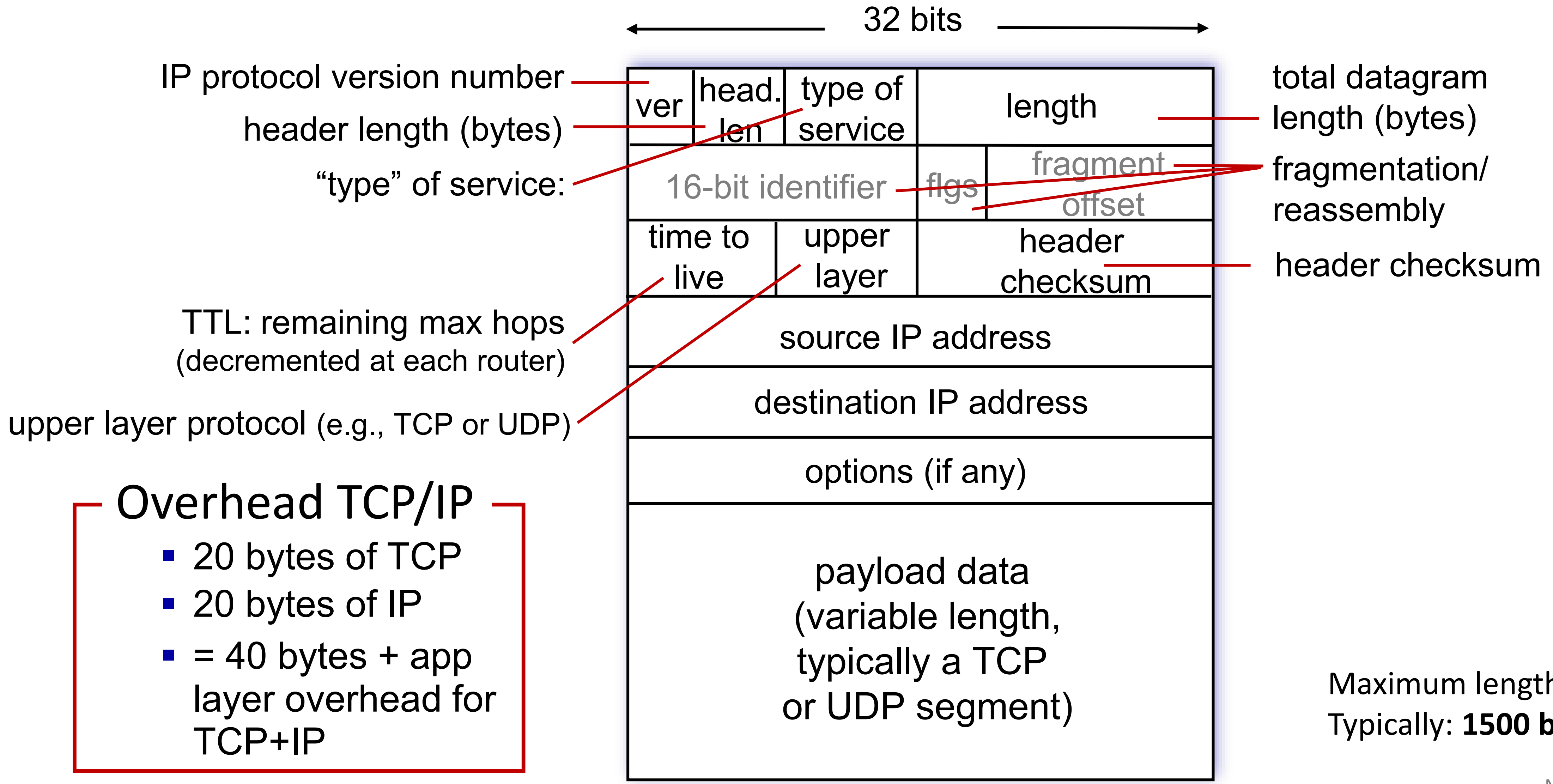


Overhead TCP/IP

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

Maximum length: **65K bytes**
Typically: **1500 bytes** (or less)

IP Datagram format

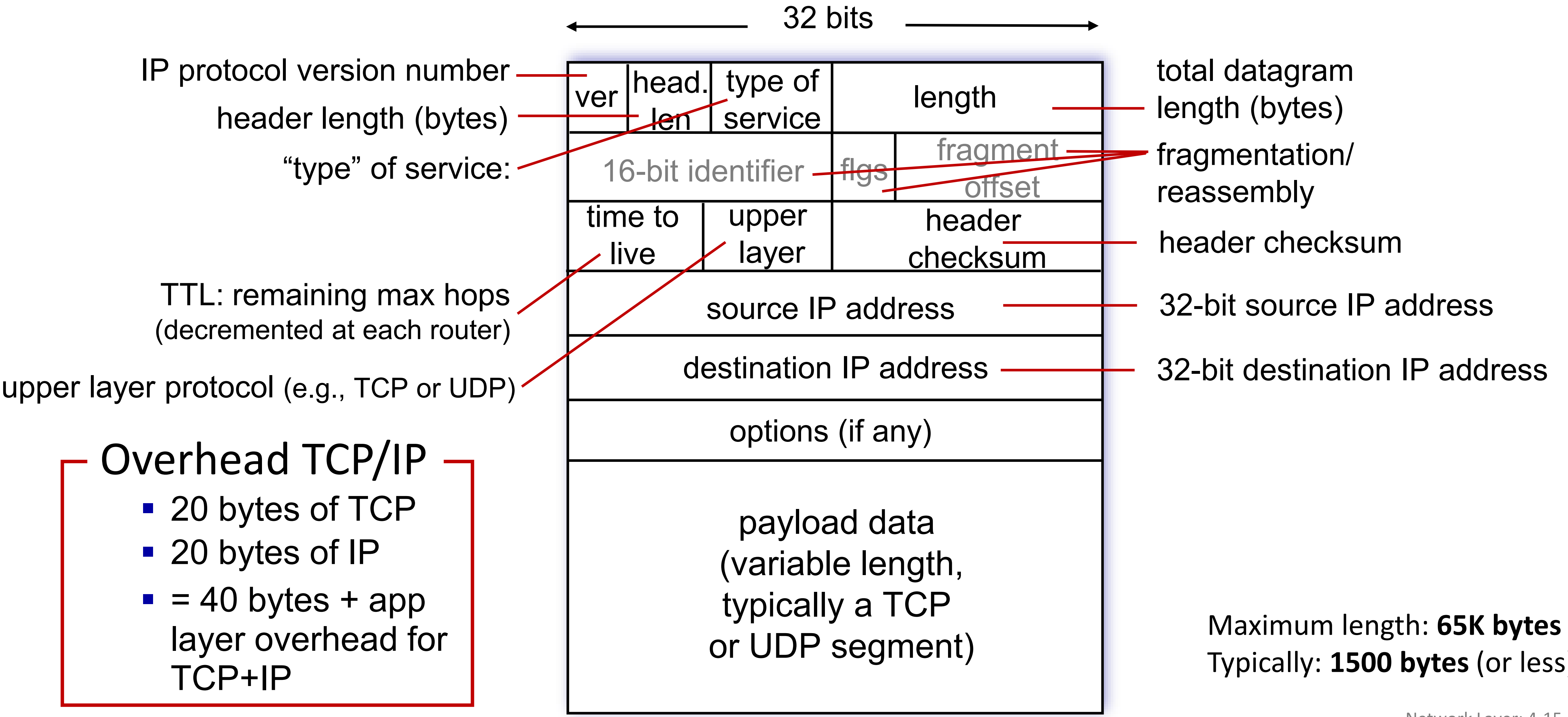


Overhead TCP/IP

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

Maximum length: **65K bytes**
Typically: **1500 bytes** (or less)

IP Datagram format



Overhead TCP/IP

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

Maximum length: **65K bytes**
Typically: **1500 bytes** (or less)

IP Datagram format

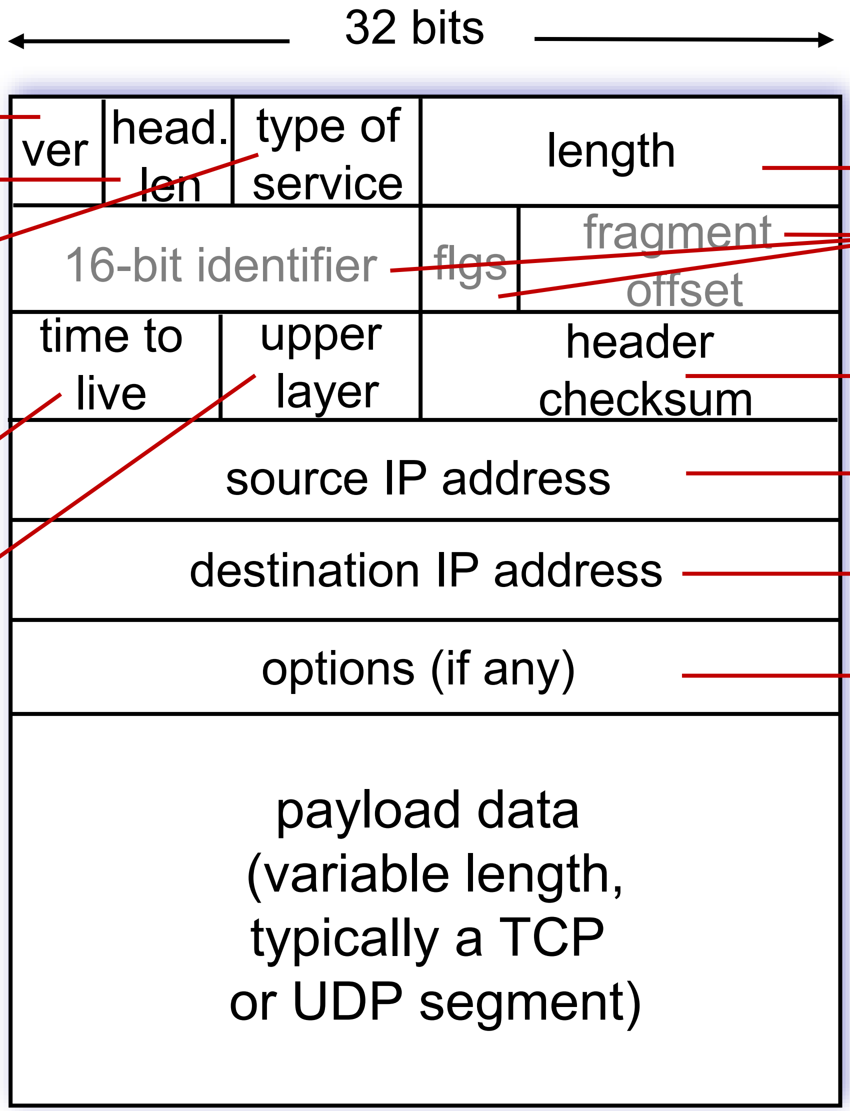
IP protocol version number

header length (bytes)

“type” of service:

TTL: remaining max hops
(decremented at each router)

upper layer protocol (e.g., TCP or UDP)



total datagram length (bytes)

fragmentation/ reassembly

header checksum

32-bit source IP address

32-bit destination IP address

e.g., record route taken

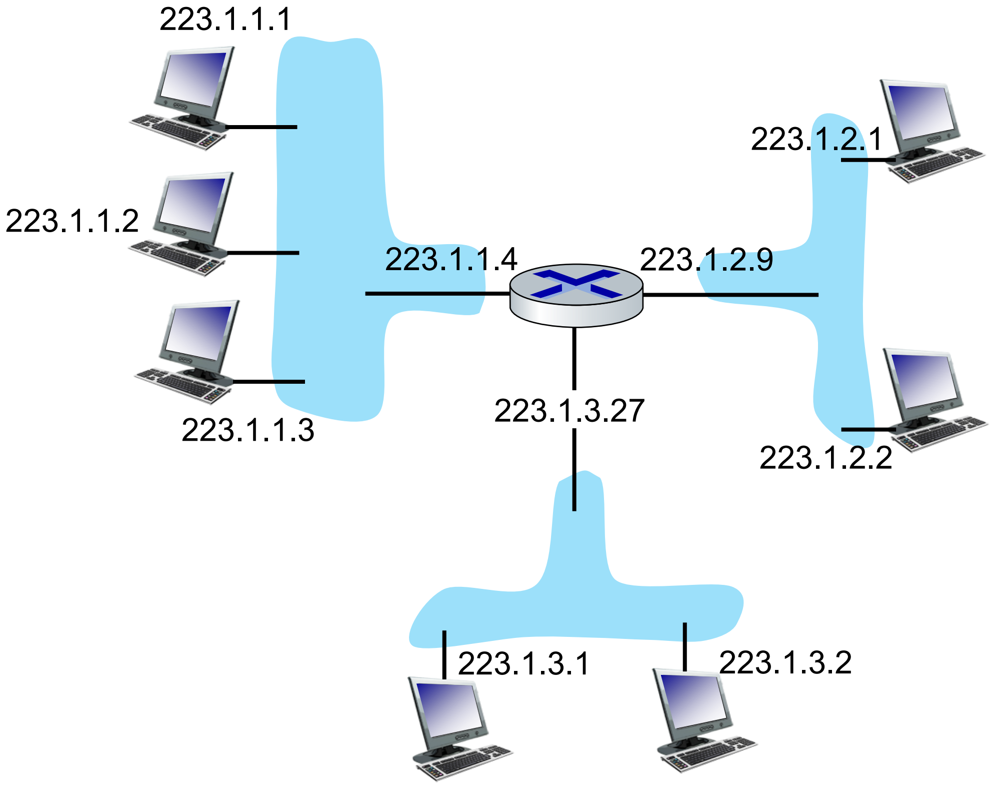
Overhead TCP/IP

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

Maximum length: **65K bytes**
Typically: **1500 bytes** (or less)

IP addressing: introduction

- IP address: 32-bit identifier associated with each host or router *interface*
- interface: connection between host/router and physical link
  - router's typically have multiple interfaces
    - *port* and *interface* are synonyms in our context
  - host typically has one or two interfaces (e.g., wired Ethernet, wireless 802.11)



dotted-decimal IP address notation:

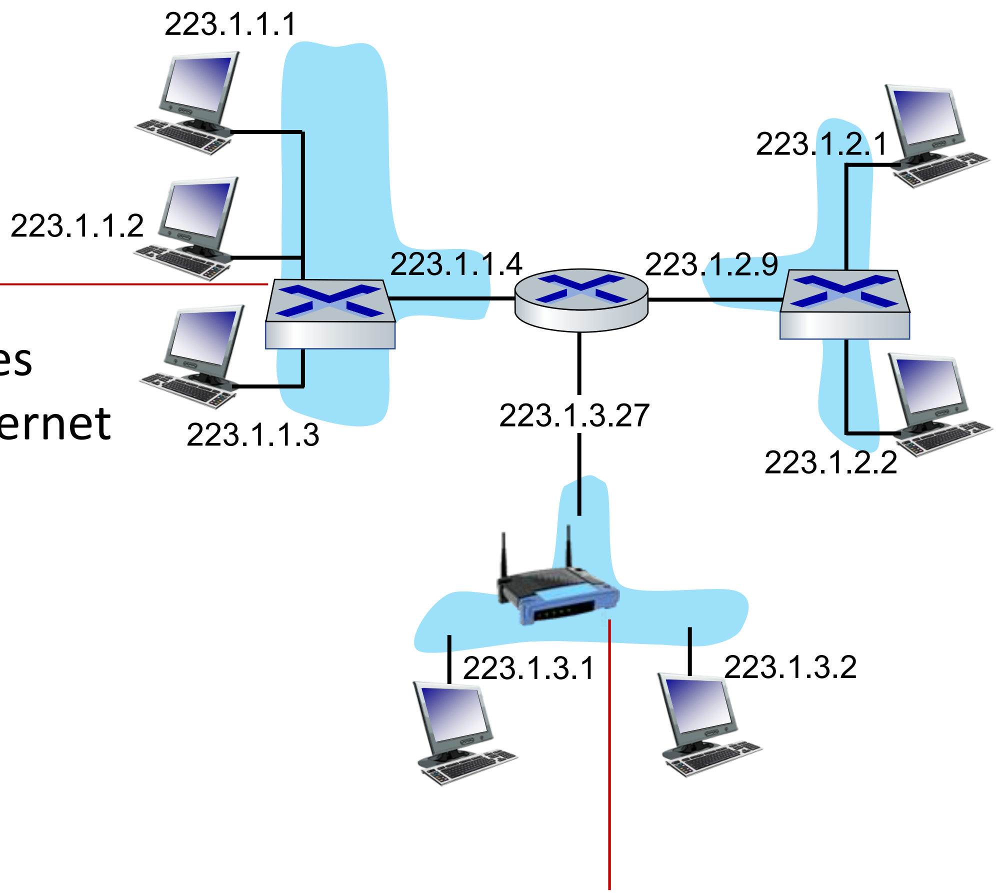
$$223.1.1.1 = \underbrace{11011111}_{223}\ \underbrace{00000001}_{1}\ \underbrace{00000001}_{1}\ \underbrace{00000001}_{1}$$

IP addressing: introduction

Q: how are interfaces actually connected?

A: we'll partially learn about that in chapter 6

A: wired Ethernet interfaces connected by Ethernet switches

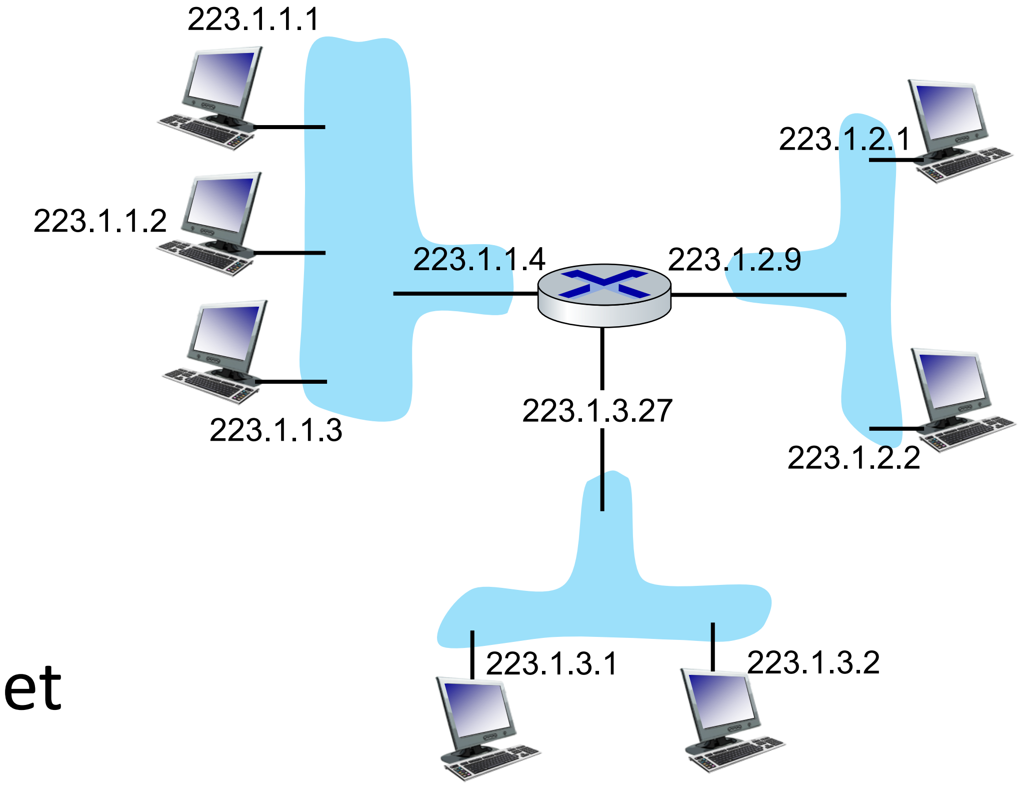


A: wireless WiFi interfaces connected by WiFi base station (not investigated)

For now: don't need to worry about how one interface is connected to another (with no router in between)

Subnets

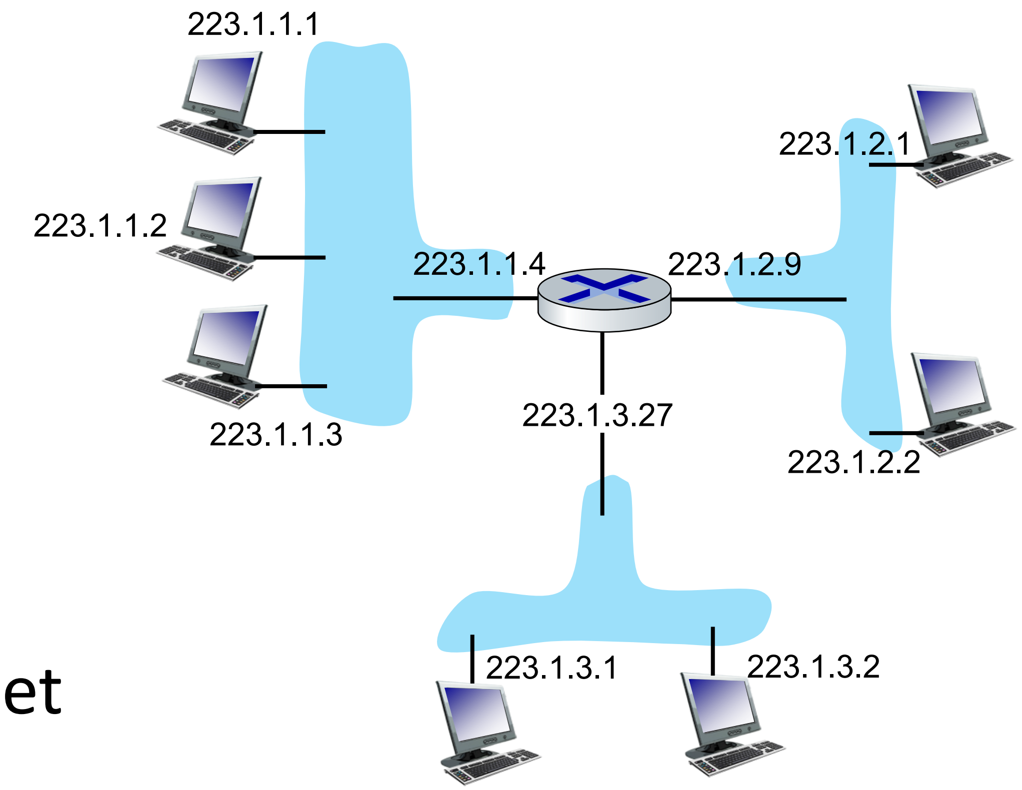
- *What's a subnet?*
  - device interfaces that can reach each other without passing through a router
- IP addresses have a structure:
  - subnet part: devices in same subnet have common high order bits
  - host part: remaining low order bits



How many subnets in this case?

Subnets

- *What's a subnet?*
  - device interfaces that can reach each other without passing through a router
- IP addresses have a structure:
  - subnet part: devices in same subnet have common high order bits
  - host part: remaining low order bits



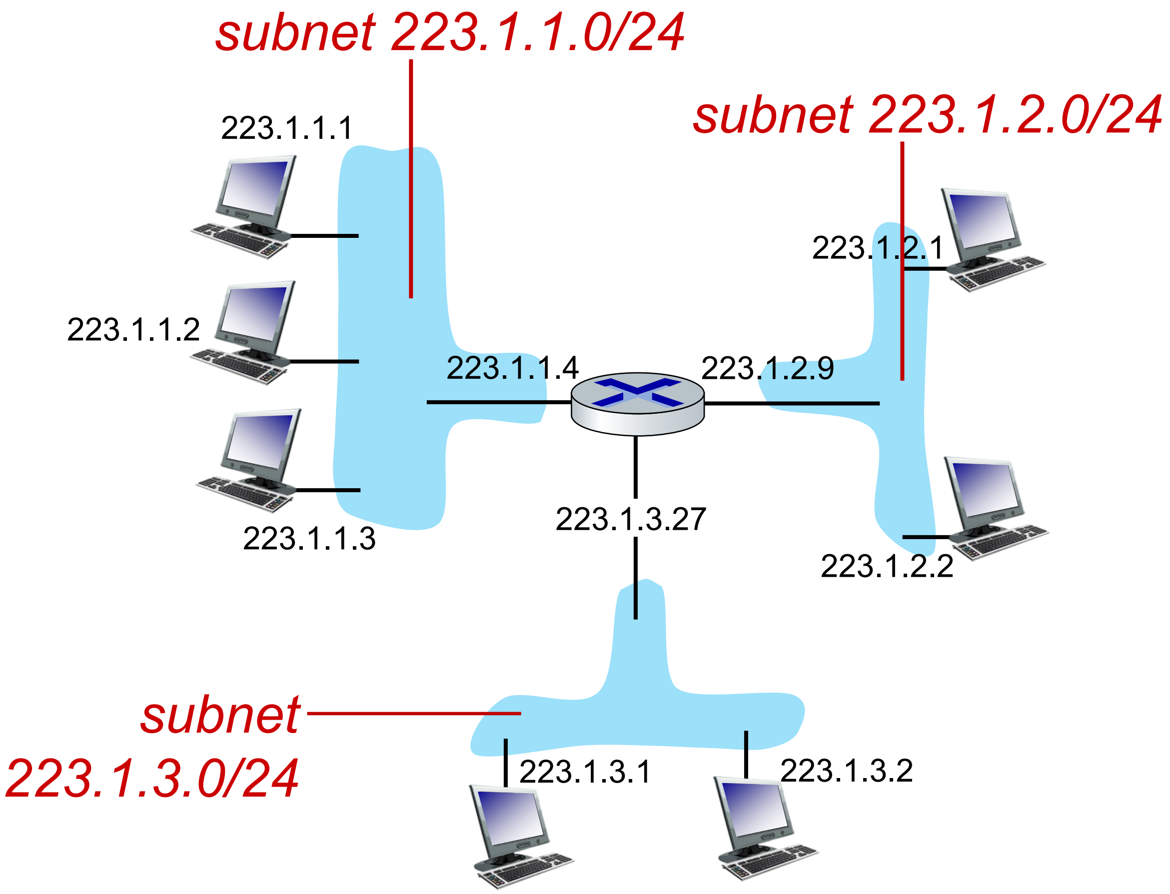
How many subnets in this case?

Network consisting of 3 subnets

Subnets

Recipe for defining subnets:

- detach each interface from its host or router, creating "islands" of isolated networks
- each isolated network is called a *subnet*
- what do the **/24** subnet addresses mean?
  - They are the *subnet addresses* (all 0s for the host part)



IP addressing: Classless (CIDR)

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 part (prefix) | host part

11001000 00010111 00010000 00000000

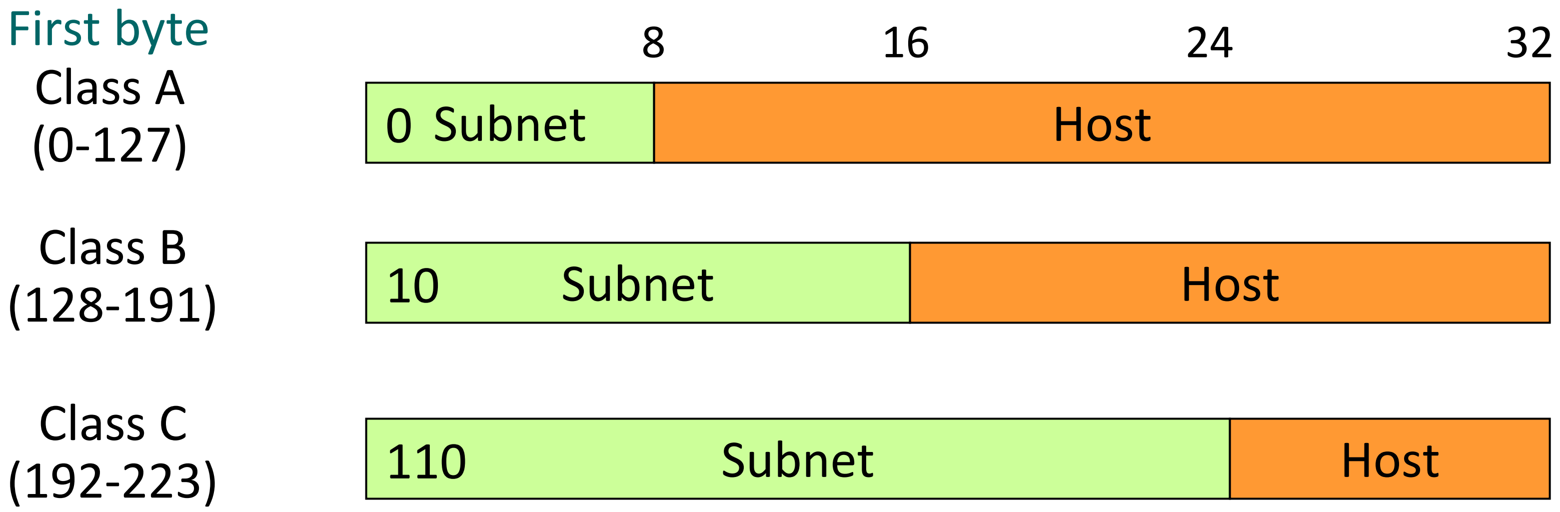
200.23.16.0/23

Alternative representation of /23: **Subnet Mask**

11111111 11111111 11111110 00000000

255.255.254.0

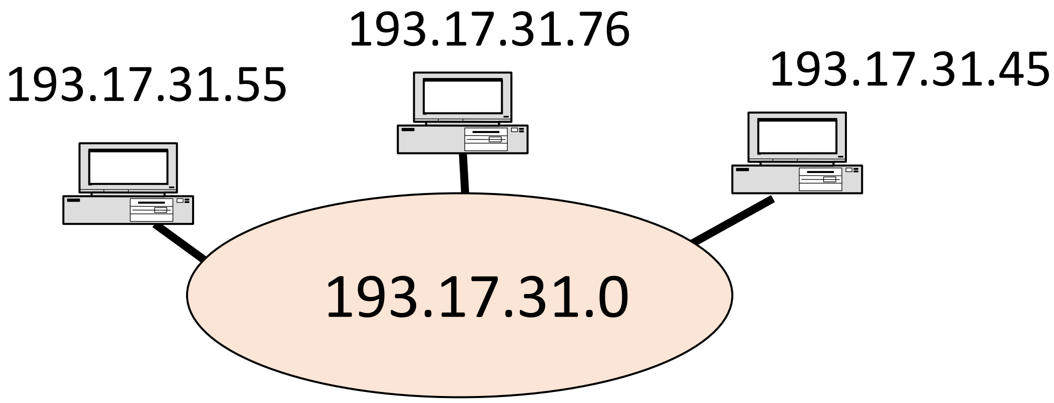
IP addressing: Classful



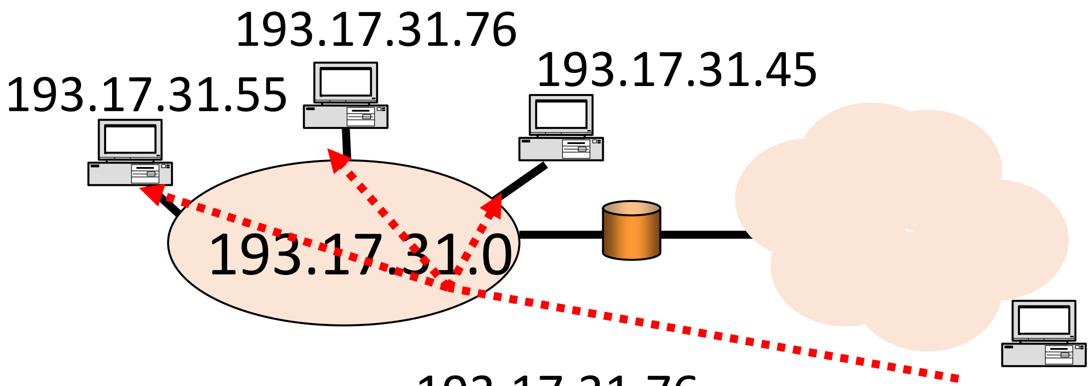
- No need for subnet mask
  - Subnet and host portions can be identified by looking at the first bits of subnet portion
- **Not used anymore**
  - Addressing is too rigid!

Special IP addresses

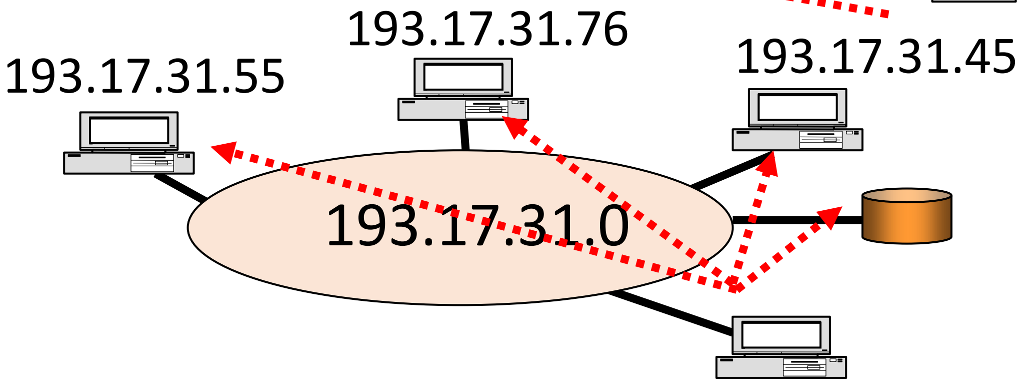
- **Subnet address:** host part with all zeros
  - E.g. 193.17.31.0/24



- **Direct broadcast address:** host part with all ones
  - E.g. 193.17.31.255



- **Limited broadcast address:** all ones (255.255.255.255)
  - Broadcast in the same subnet
  - The message cannot overcome routers



Destination-based forwarding

Destination-based forwarding

Example of forwarding table

	Destination IP Address Range	Link interface
Subnet mask: \21	11001000 00010111 00010*** ********	0
Subnet mask: \24	11001000 00010111 00011000 ********	1
Subnet mask: \21	11001000 00010111 00011*** ********	2
	otherwise	3

Destination-based forwarding

Example of forwarding table

	Destination IP Address Range	Link interface
Subnet mask: \21	11001000 00010111 00010*** ********	0
Subnet mask: \24	11001000 00010111 00011000 ********	1
Subnet mask: \21	11001000 00010111 00011*** ********	2
	otherwise	3

examples:

11001000 00010111 00010110 10100001 which interface?

11001000 00010111 00011000 10101010 which interface?

Destination-based forwarding

Destination IP Address Range	Link interface
11001000 00010111 00010*** ********	0
11001000 00010111 00011000 ********	1
11001000 00010111 00011*** ********	2
otherwise	3

match!

examples:

11001000 00010111 00010110 10100001 which interface?

11001000 00010111 00011000 10101010 which interface?

Destination-based forwarding

Destination IP Address Range	Link interface
11001000 00010111 00010*** ********	0
11001000 00010111 00011000 ********	1
11001000 00010111 00011*** ********	2
otherwise	3

match!

examples:

11001000 00010111 00010110 10100001 which interface?

11001000 00010111 00011000 10101010 which interface?

Destination-based forwarding

longest prefix match

when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

Destination IP Address Range	Link interface
11001000 00010111 00010*** ********	0
11001000 00010111 00011000 ********	1
11001000 00010111 00011*** ********	2
otherwise	3

chosen!

match!

examples:

11001000 00010111 00010110 10100001 which interface?

11001000 00010111 00011000 10101010 which interface?

IP addresses: how to get one?

That's actually two questions:

1. **Q:** How does a *host* get IP address within its network (host part of address)?
2. **Q:** How does a *network* get IP address for itself (network part of address)

How does a *host* get IP address?

- Statically specified in config file of the OS (e.g., /etc/rc.config in UNIX)
- DHCP: Dynamic Host Configuration Protocol → dynamically get address from a server (called *DHCP server*)
  - "plug-and-play"

DHCP: Dynamic Host Configuration Protocol

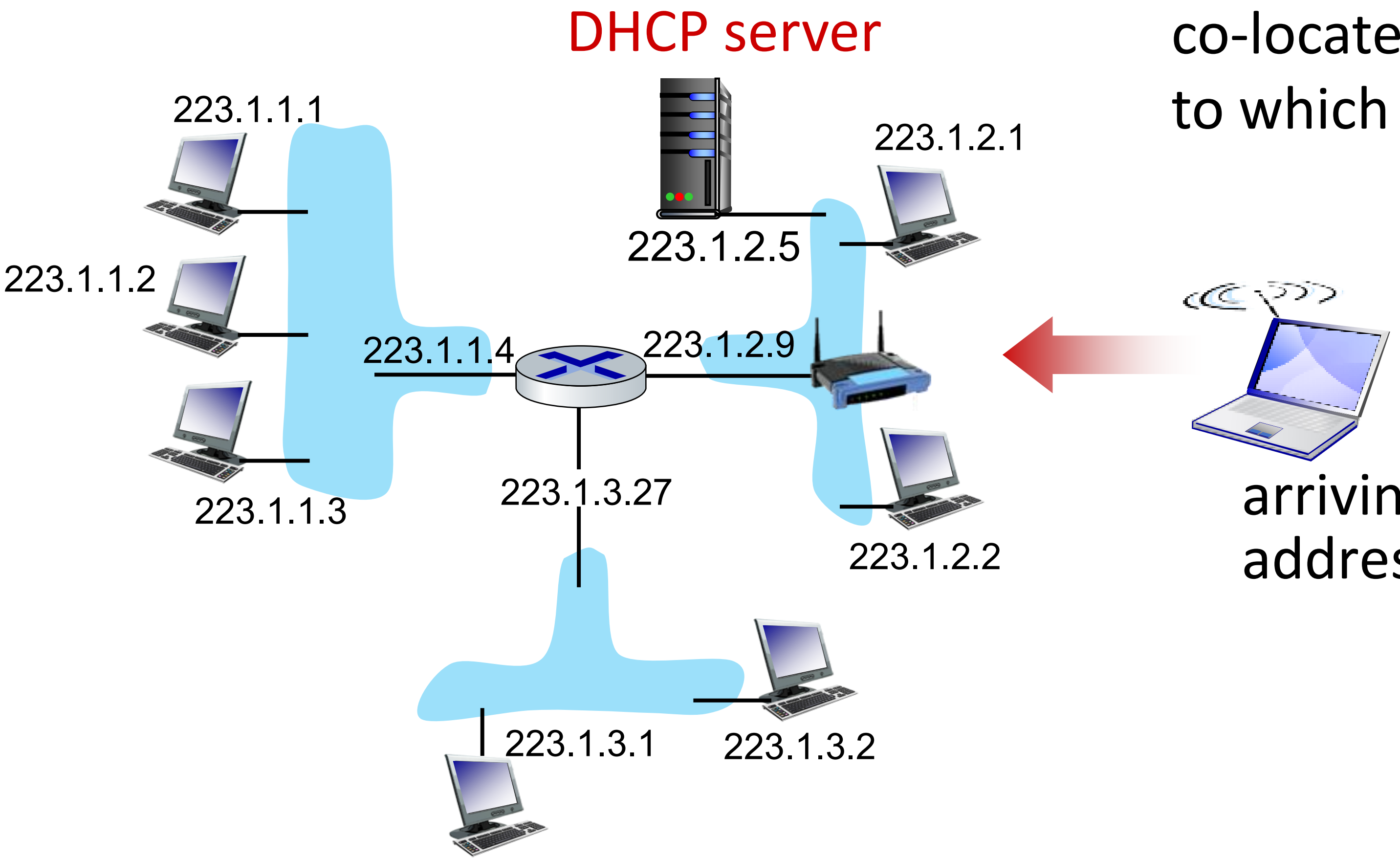
goal: host *dynamically* obtains IP address from network server when it "joins" the network

- can renew its lease on address in use
- allows reuse of addresses (only hold address while connected/on)

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

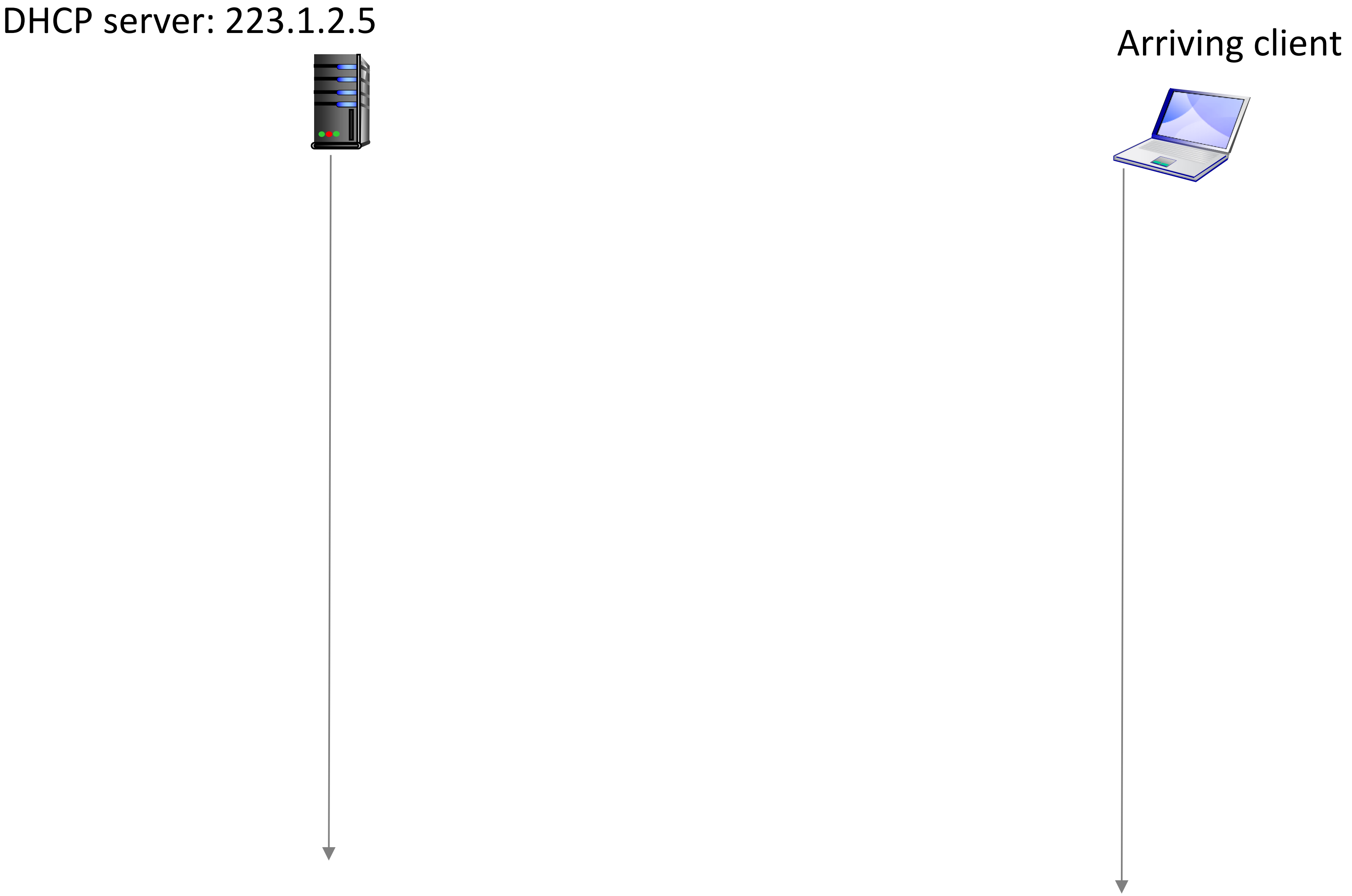
DHCP client-server scenario



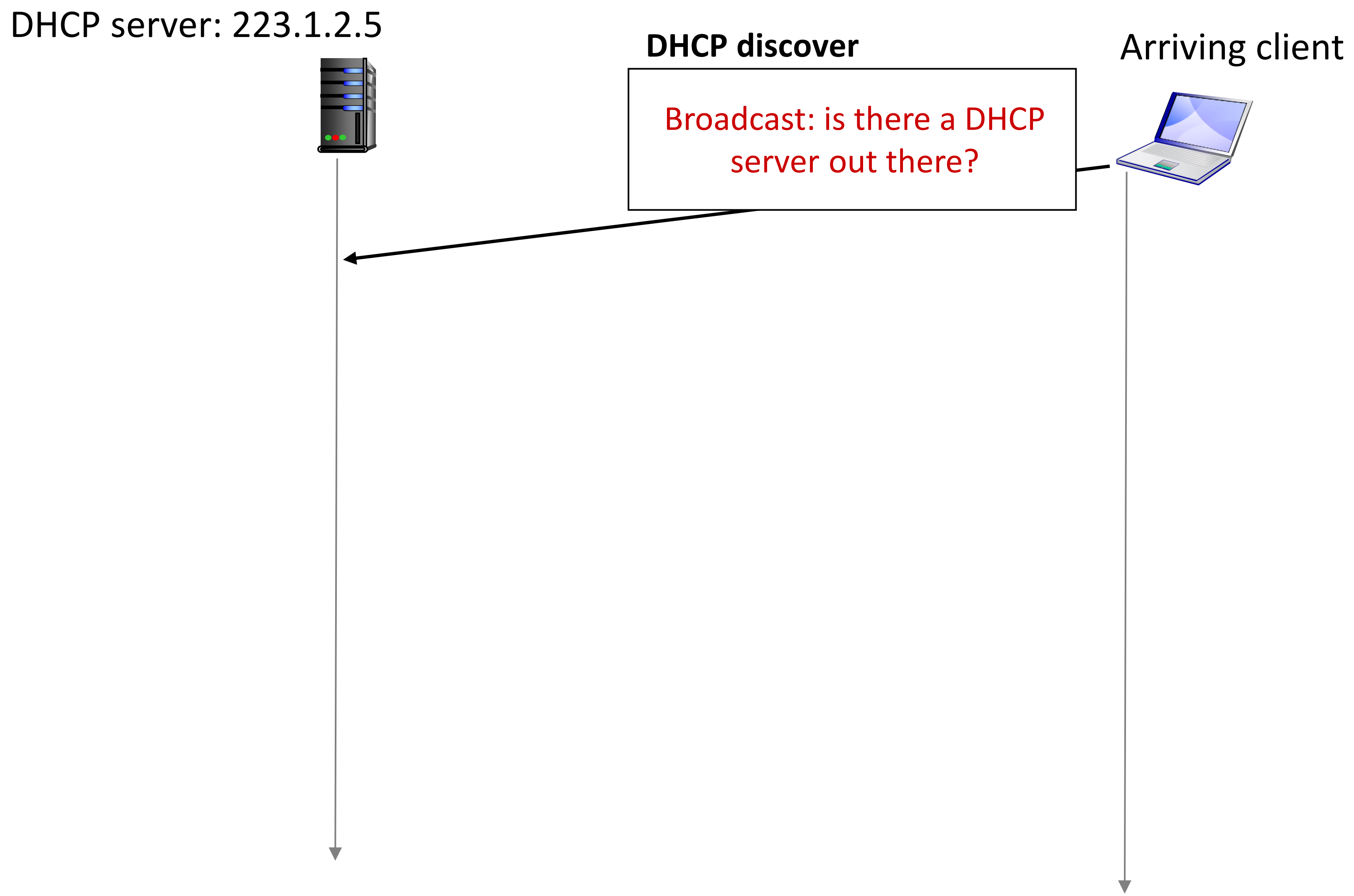
Typically, DHCP server will be co-located in router, serving subnets to which router is attached

arriving DHCP client needs address in this network

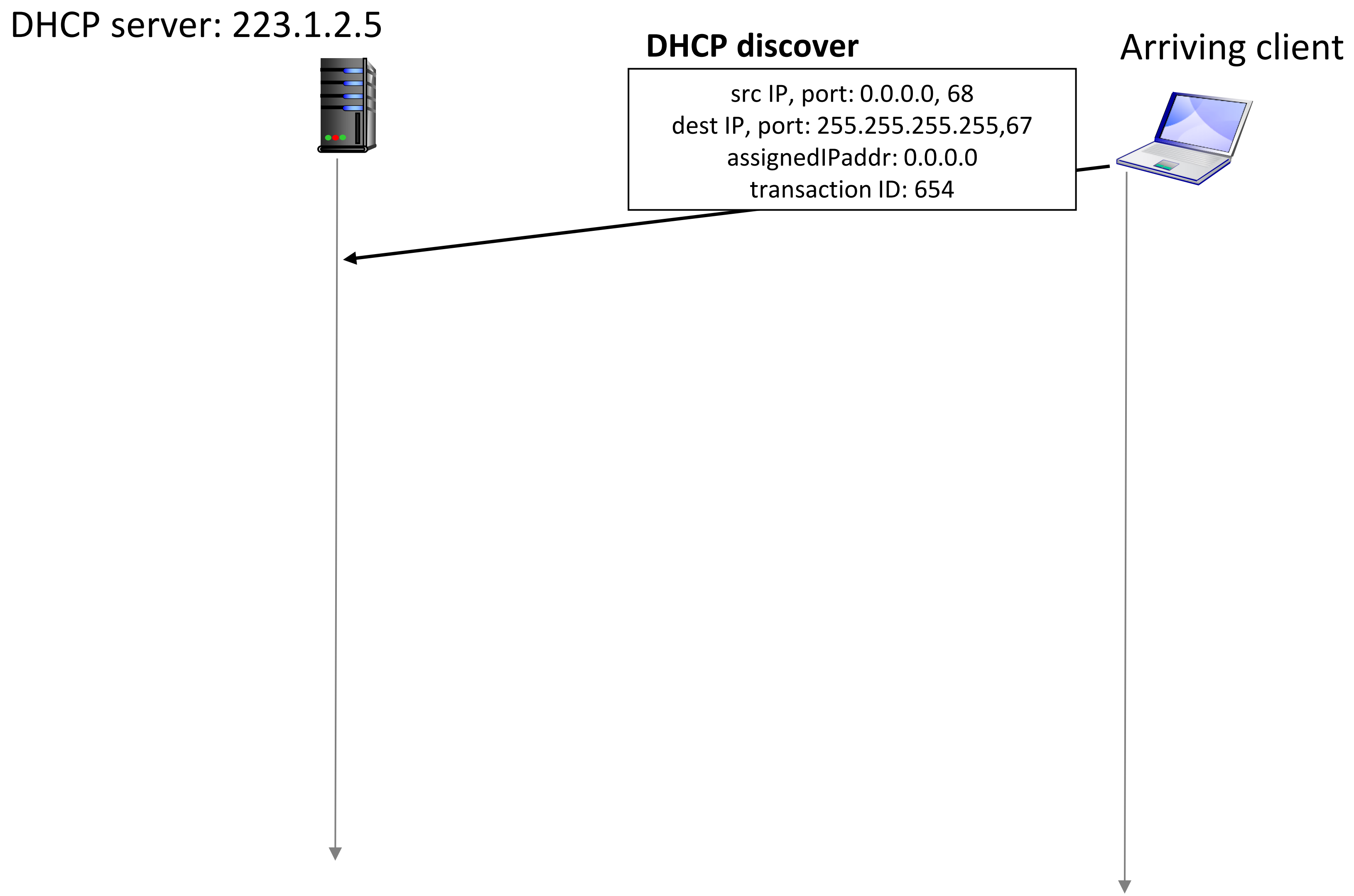
DHCP client-server scenario



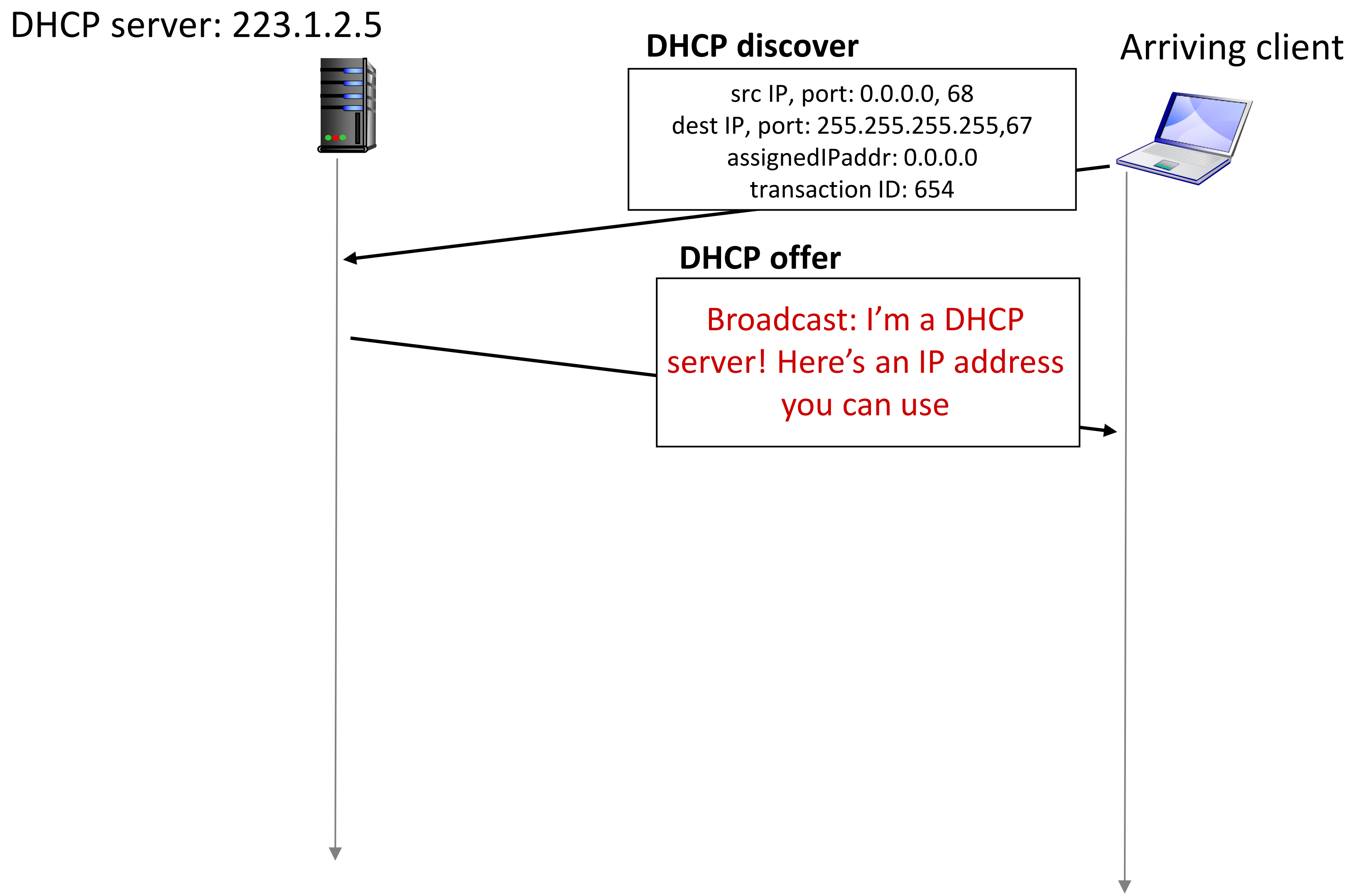
DHCP client-server scenario



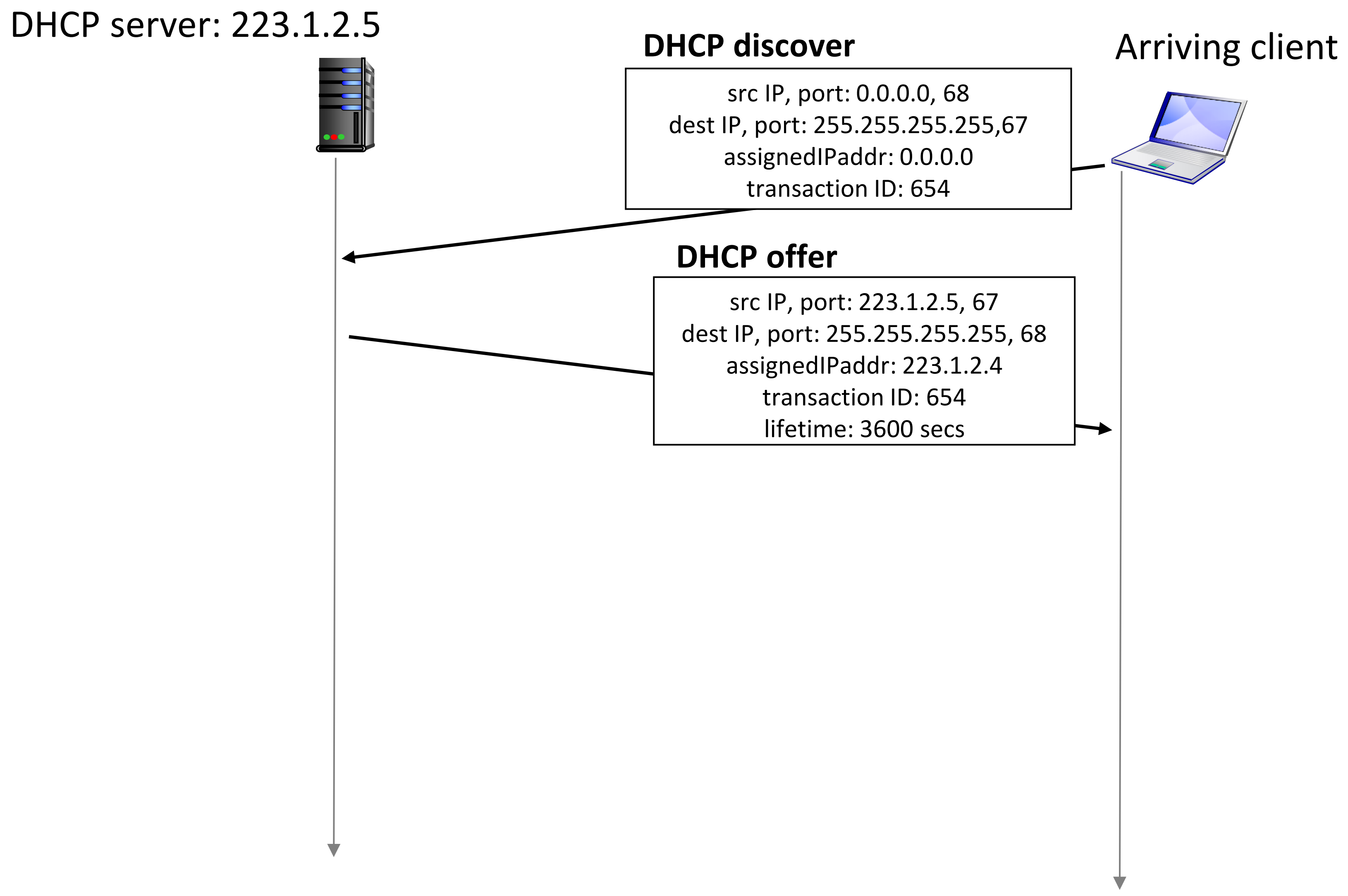
DHCP client-server scenario



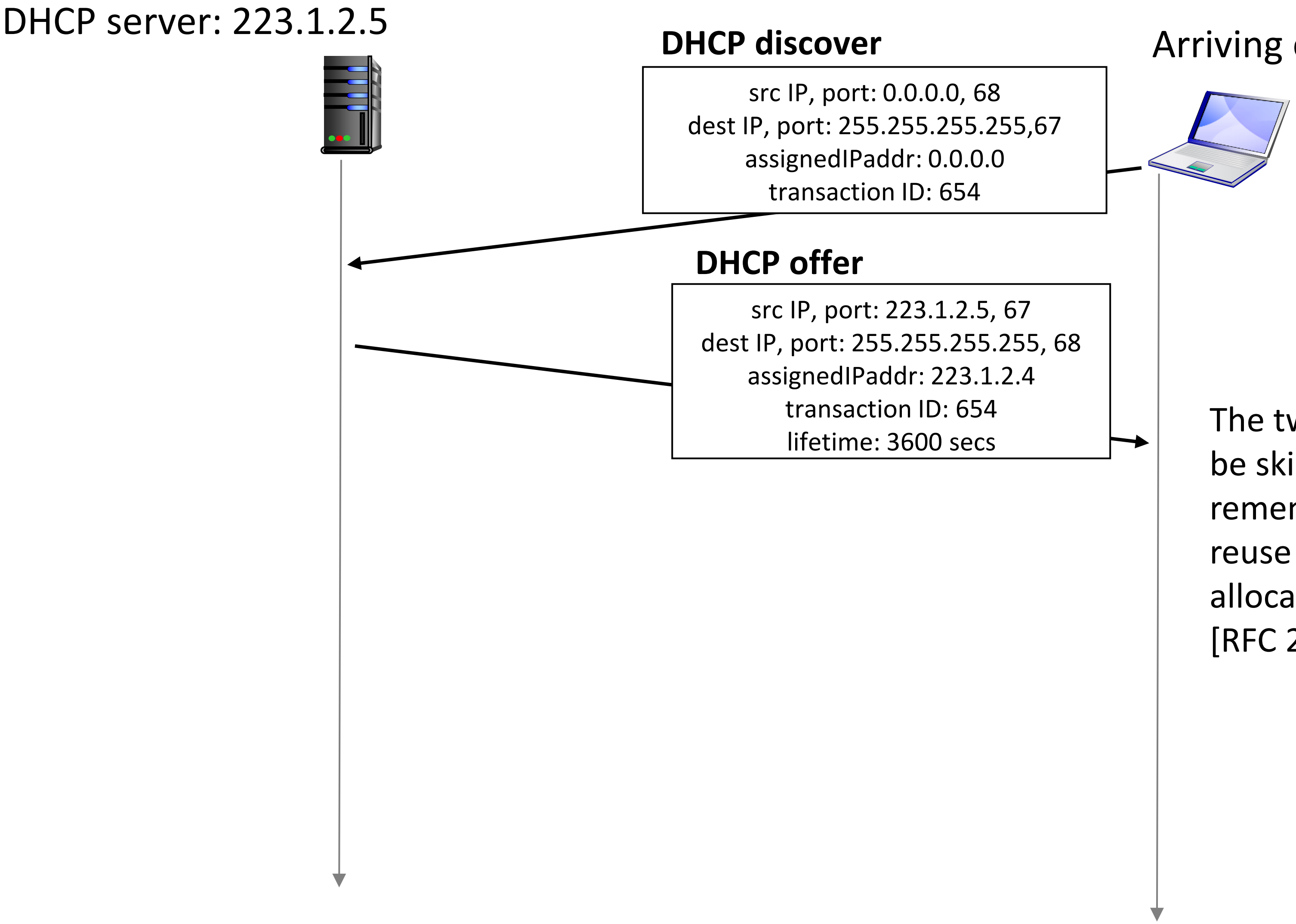
DHCP client-server scenario



DHCP client-server scenario

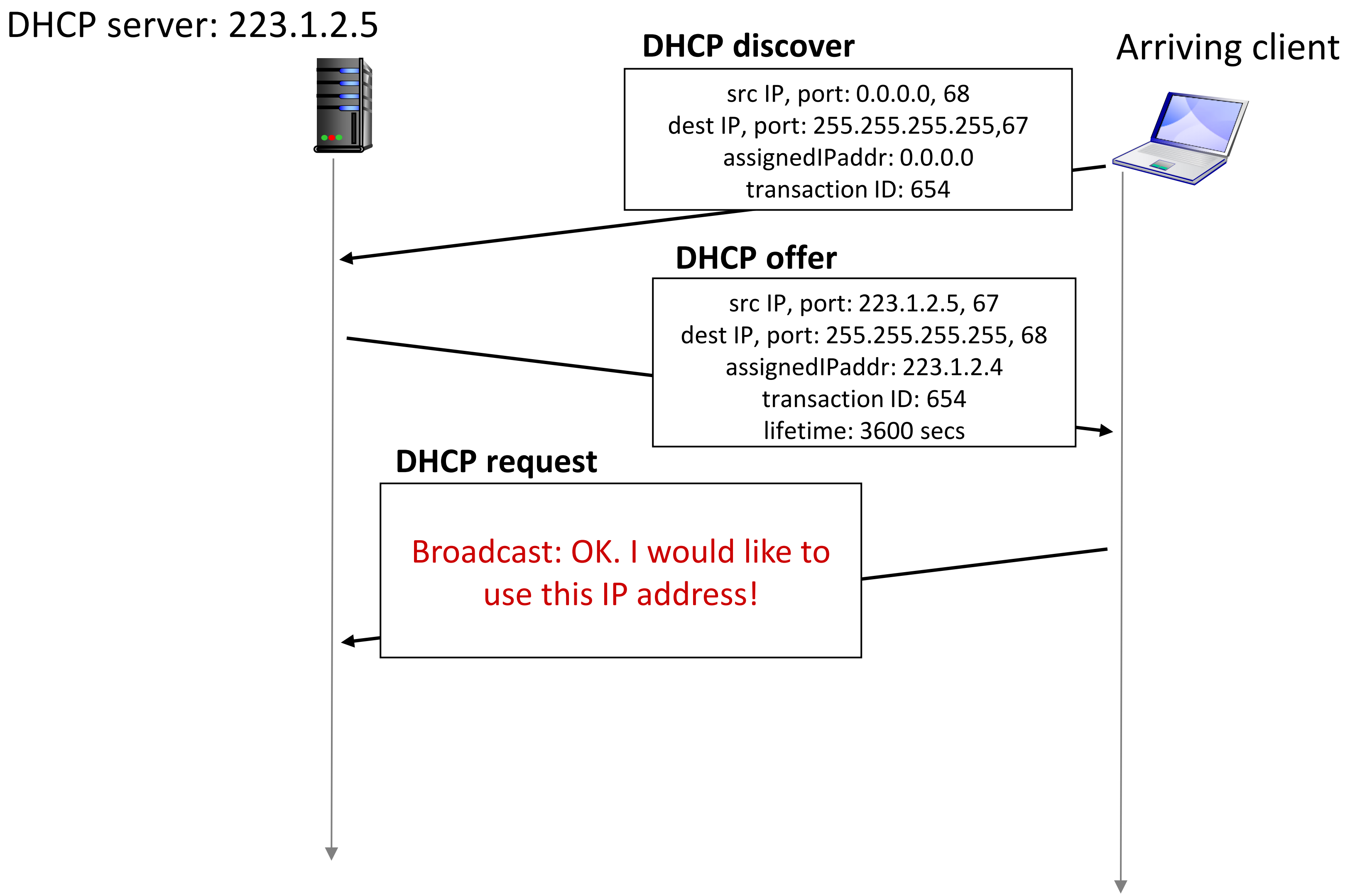


DHCP client-server scenario

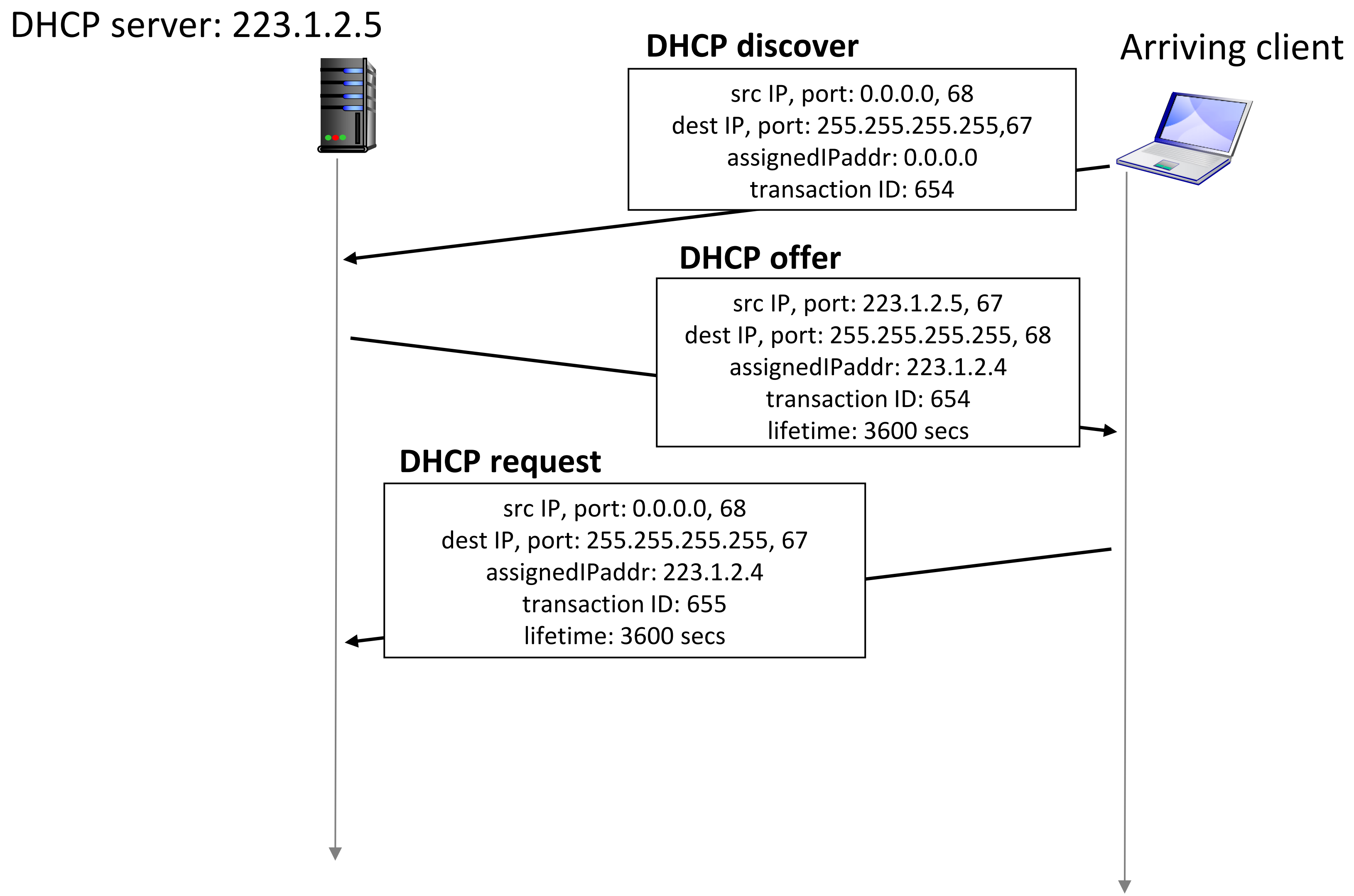


The two steps above can be skipped “if a client remembers and wishes to reuse a previously allocated network address” [RFC 2131]

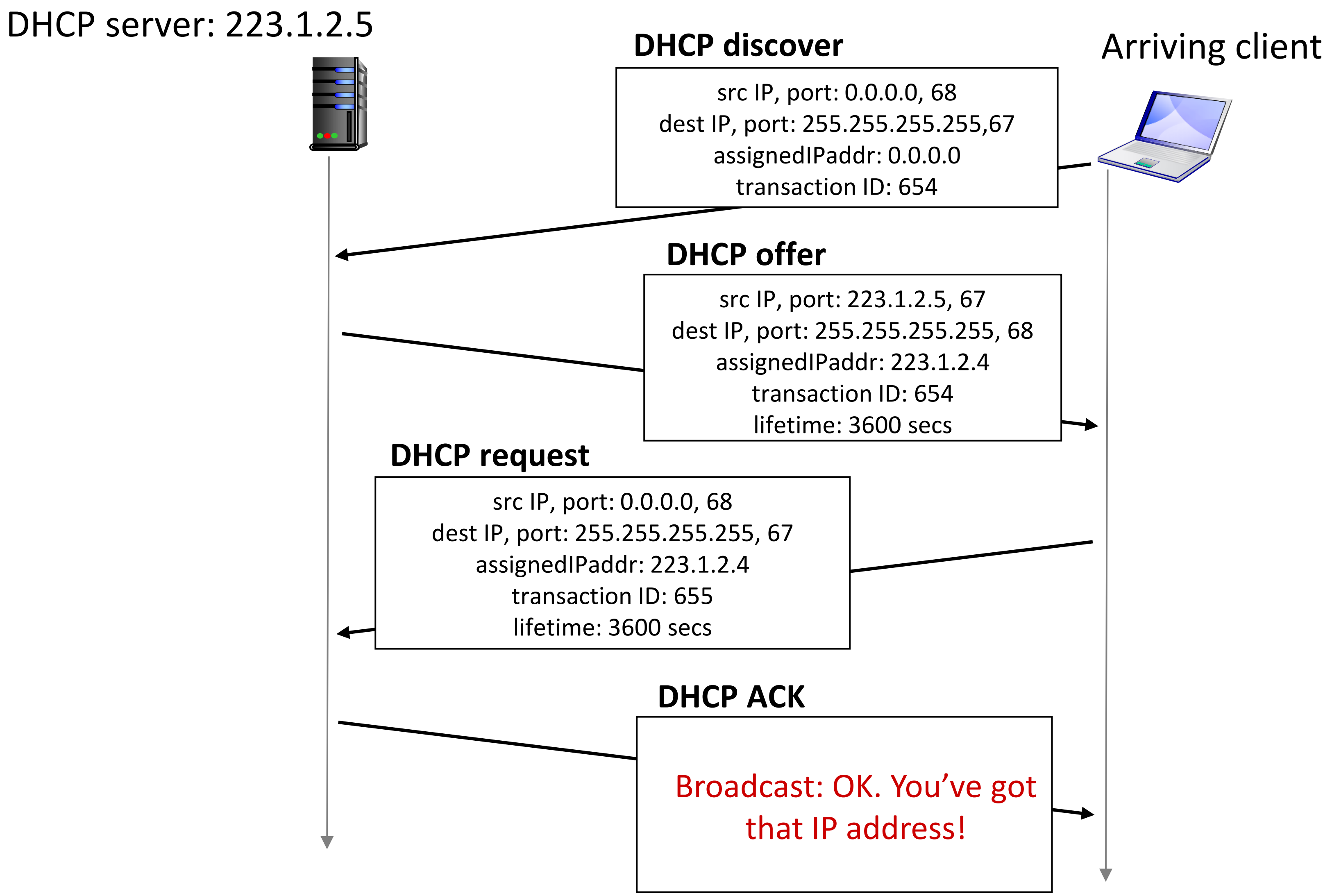
DHCP client-server scenario



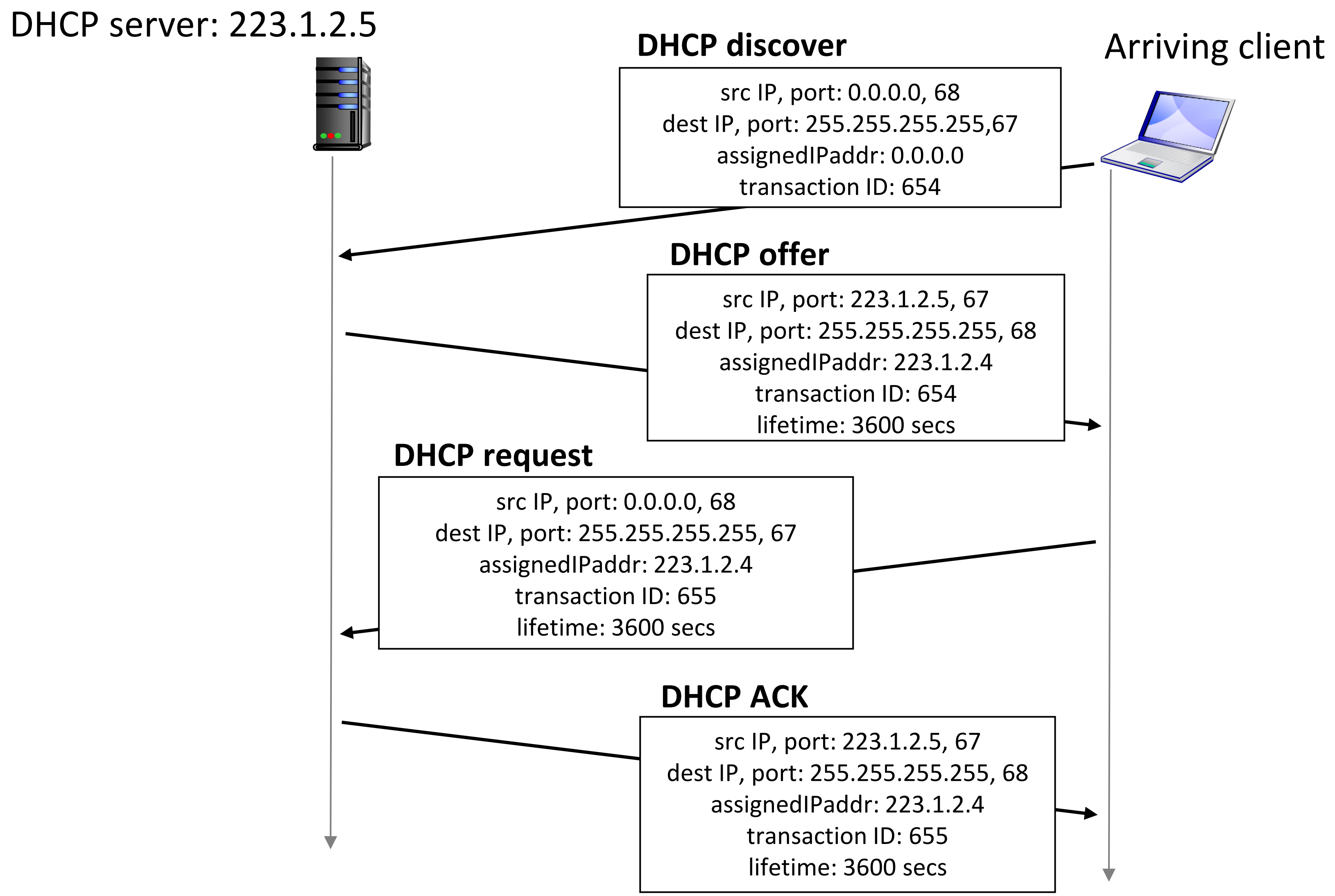
DHCP client-server scenario



DHCP client-server scenario



DHCP client-server scenario



DHCP: more than IP addresses

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

- name and IP address of the **local DNS server**
- **subnet mask** (indicating network versus host portion of address)
- **default gateway** (IP address of first-hop router)

IP addresses: how to get one?

Q: how does *network* get subnet part of IP address?

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

ISP's block 11001000 00010111 00010000 00000000 200.23.16.0/20

Let's assume that there are 8 organizations that need to get a subnet address → ISP can then split its address space in 8 blocks:

IP addresses: how to get one?

Q: how does *network* get subnet part of IP address?

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

ISP's block	11001000 00010111 00010000 00000000	200.23.16.0/20

Let's assume that there are 8 organizations that need to get a subnet address → ISP can then split its address space in 8 blocks:

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

IP addresses: how to get one?

Q: how does *network* get subnet part of IP address?

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

ISP's block	11001000 00010111 00010000 00000000	200.23.16.0/20

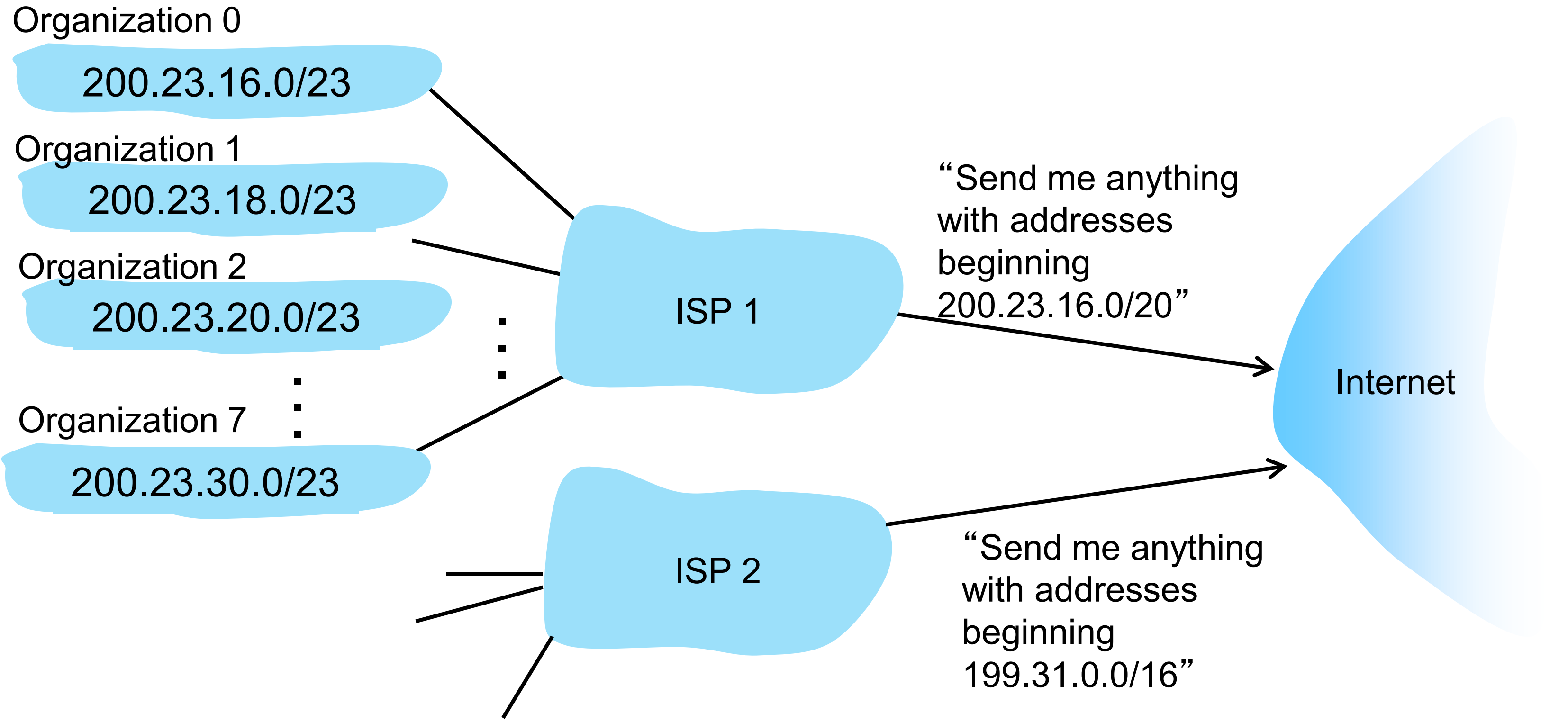
Let's assume that there are 8 organizations that need to get a subnet address → ISP can then split its address space in 8 blocks:

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

This operation is called *subnetting* or *hierarchical addressing*

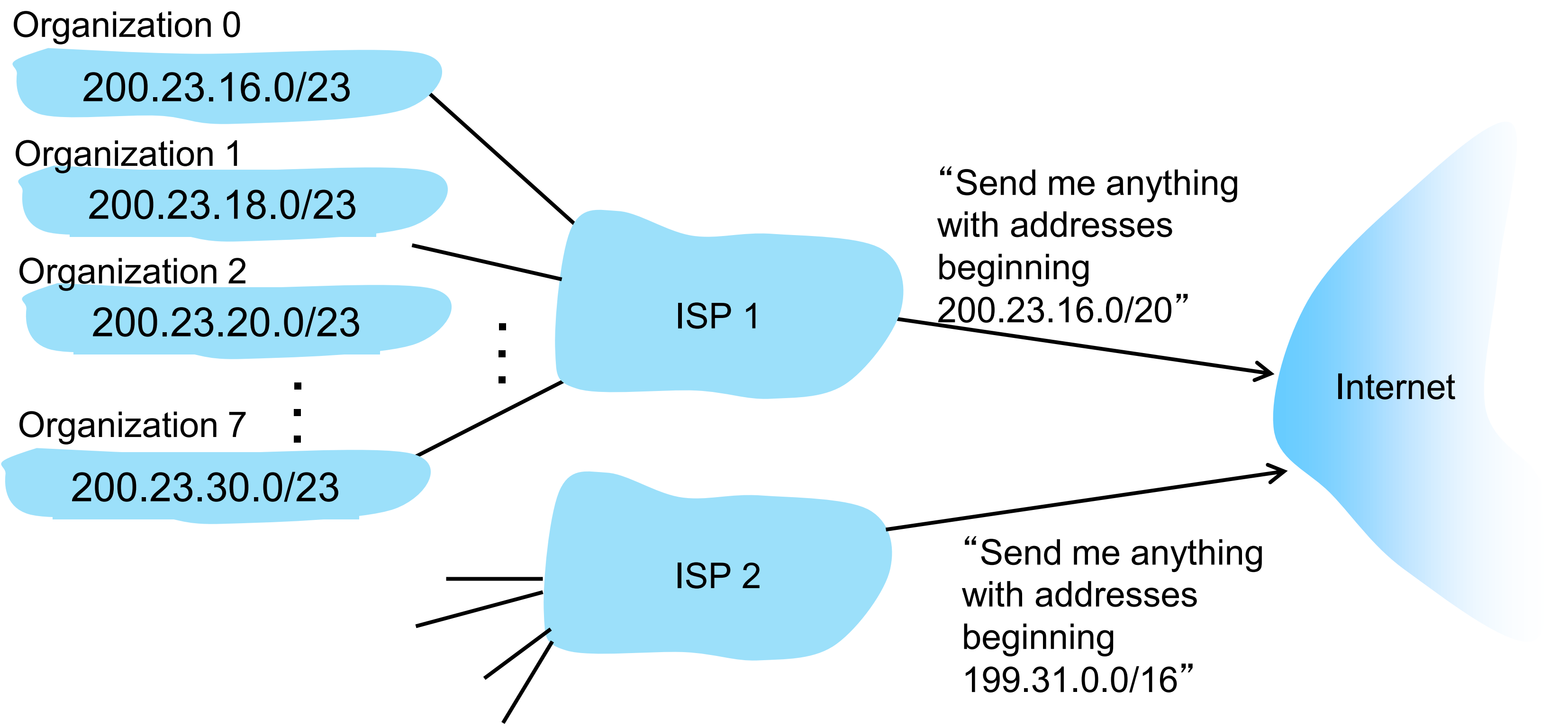
Hierarchical addressing: route aggregation

hierarchical addressing allows efficient advertisement of routing information (*route aggregation*):



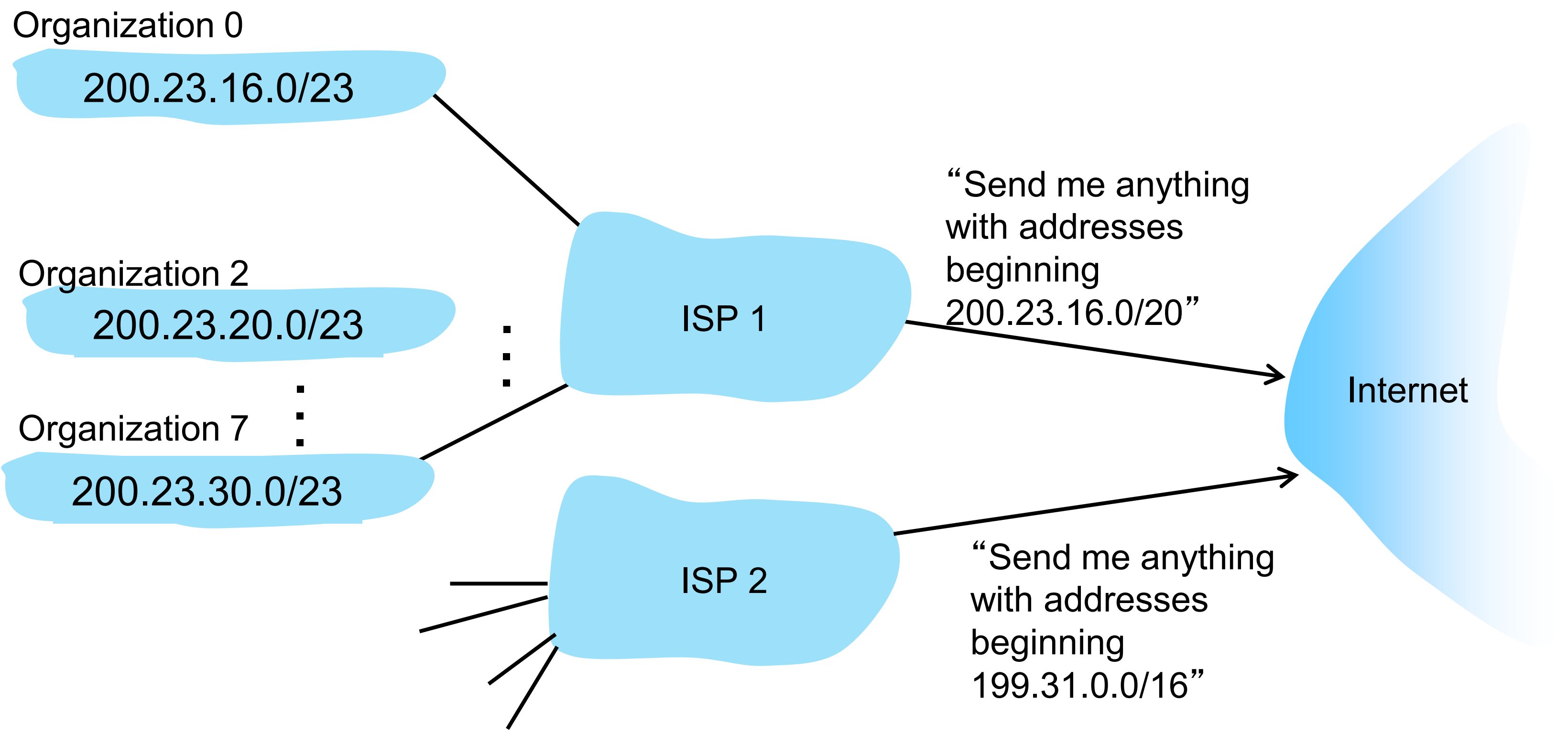
Hierarchical addressing: more specific routes

- Organization 1 moves from ISP 1 to ISP 2
- ISP 2 now advertises a more specific route to Organization 1



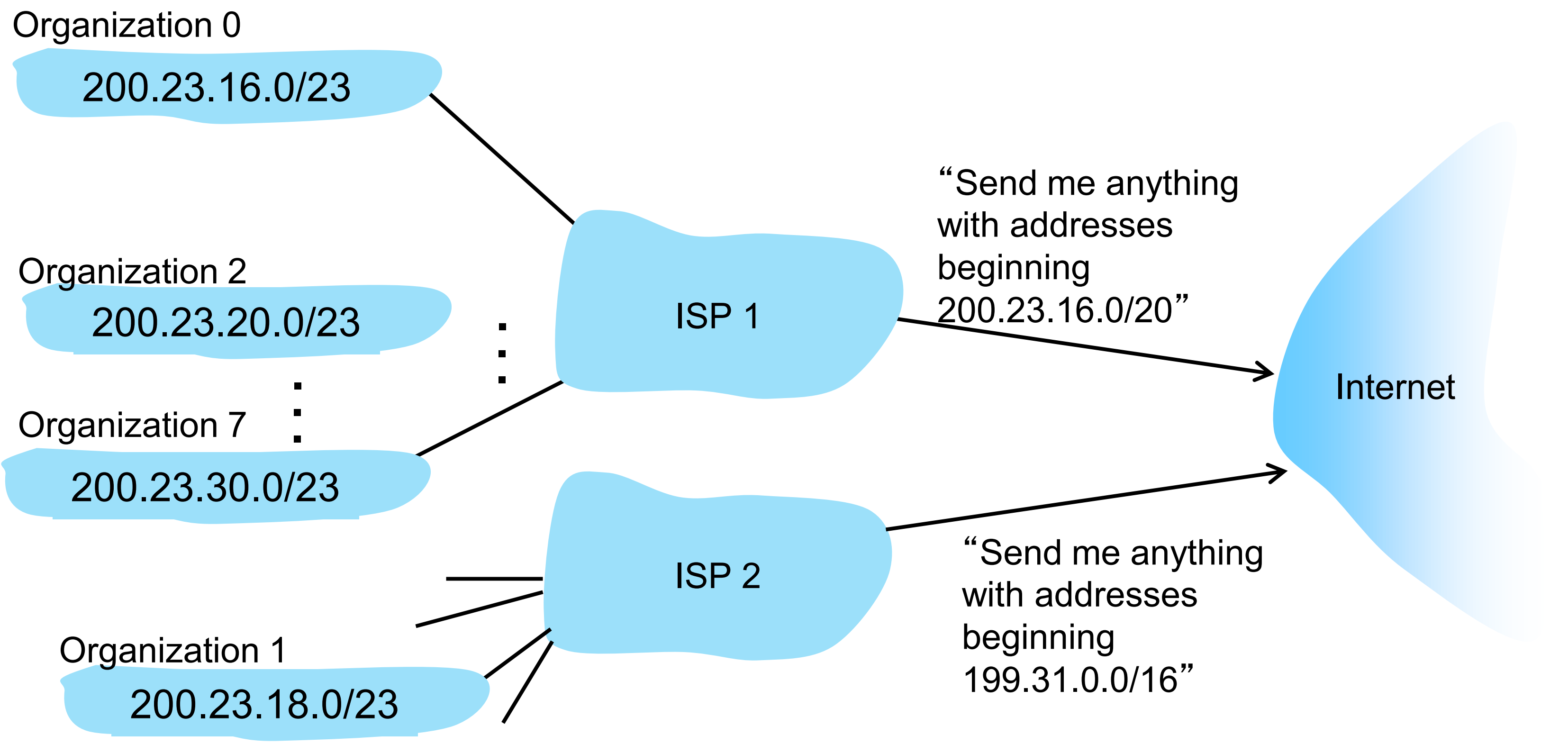
Hierarchical addressing: more specific routes

- Organization 1 moves from ISP 1 to ISP 2
- ISP 2 now advertises a more specific route to Organization 1



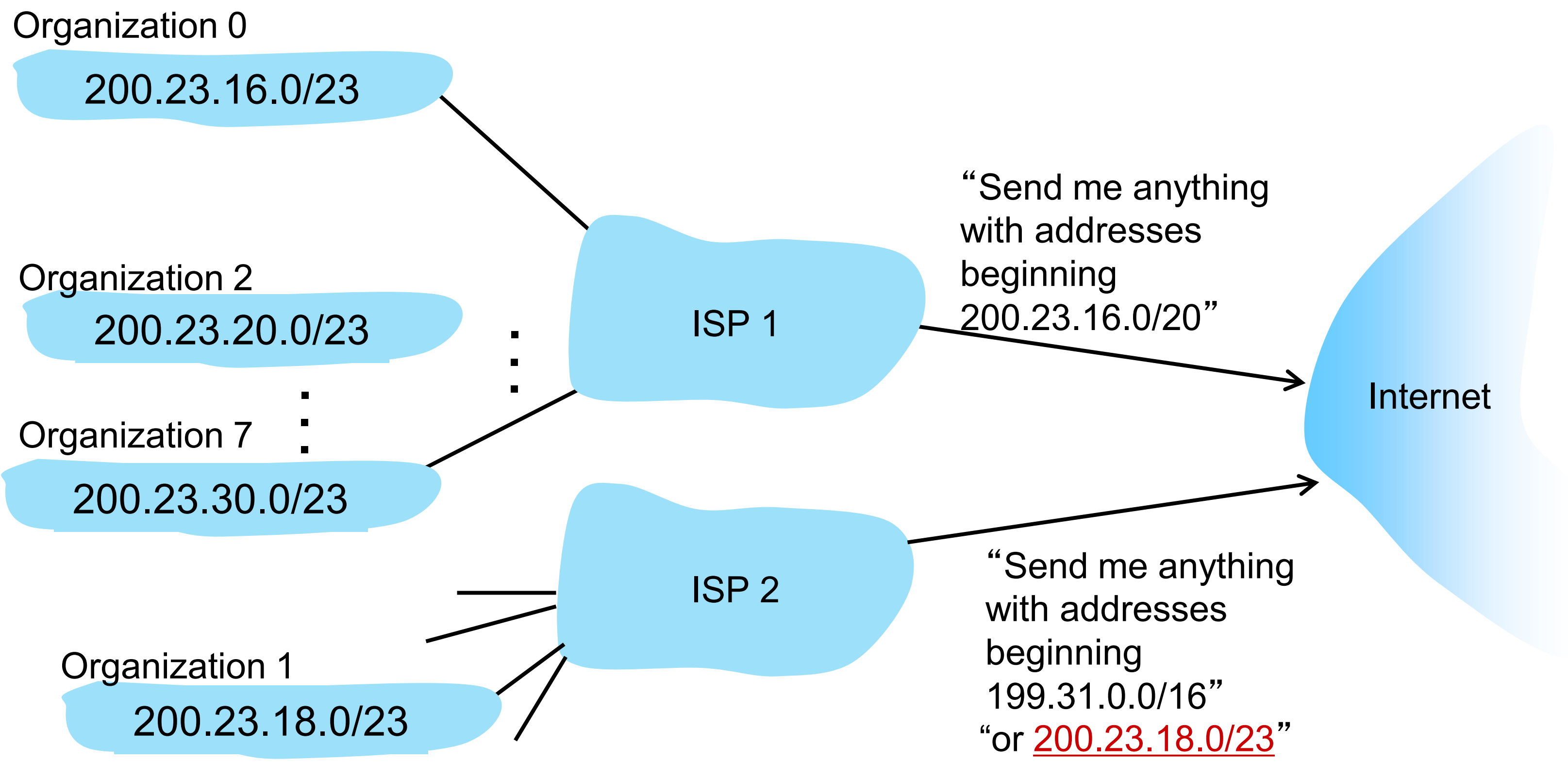
Hierarchical addressing: more specific routes

- Organization 1 moves from ISP 1 to ISP 2
- ISP 2 now advertises a more specific route to Organization 1



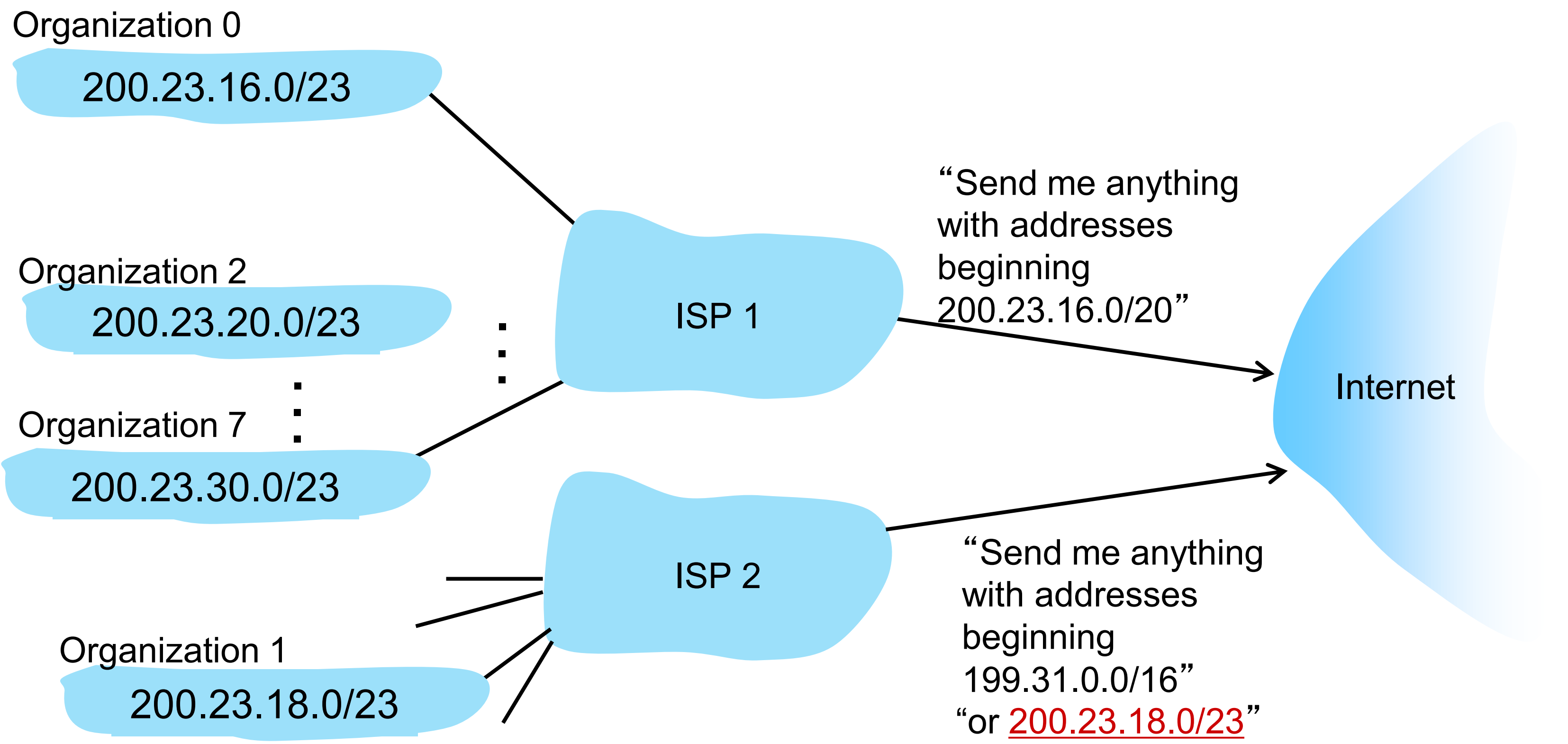
Hierarchical addressing: more specific routes

- Organization 1 moves from ISP 1 to ISP 2
- ISP 2 now advertises a more specific route to Organization 1



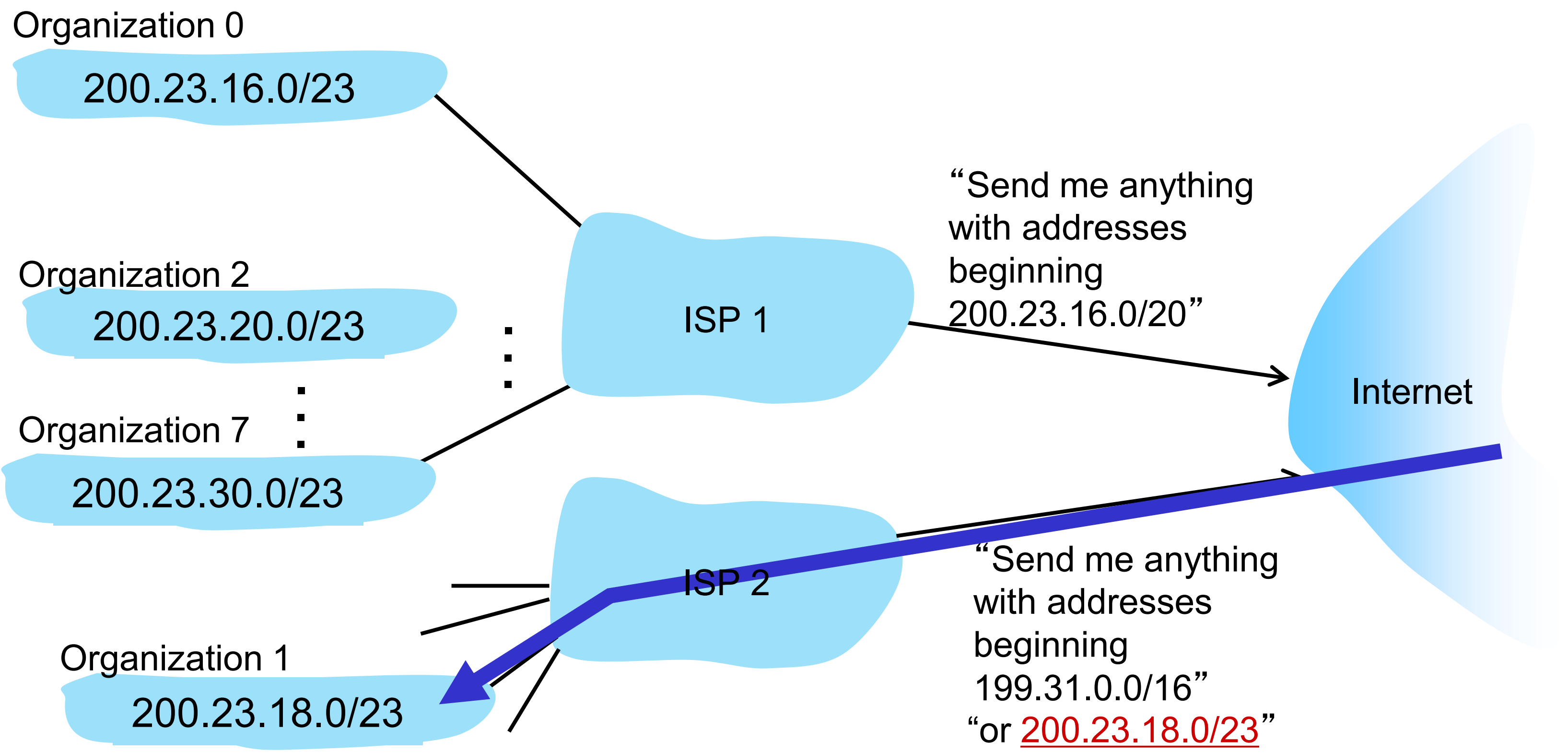
Hierarchical addressing: more specific routes

- Organization 1 moves from ISP 1 to ISP 2
- ISP 2 now advertises a more specific route to Organization 1



Hierarchical addressing: more specific routes

- Organization 1 moves from ISP 1 to ISP 2
- ISP 2 now advertises a more specific route to Organization 1



IP addressing: last words...

Q: how does an ISP get block of addresses?

A: ICANN: Internet Corporation for Assigned Names and Numbers http://www.icann.org/

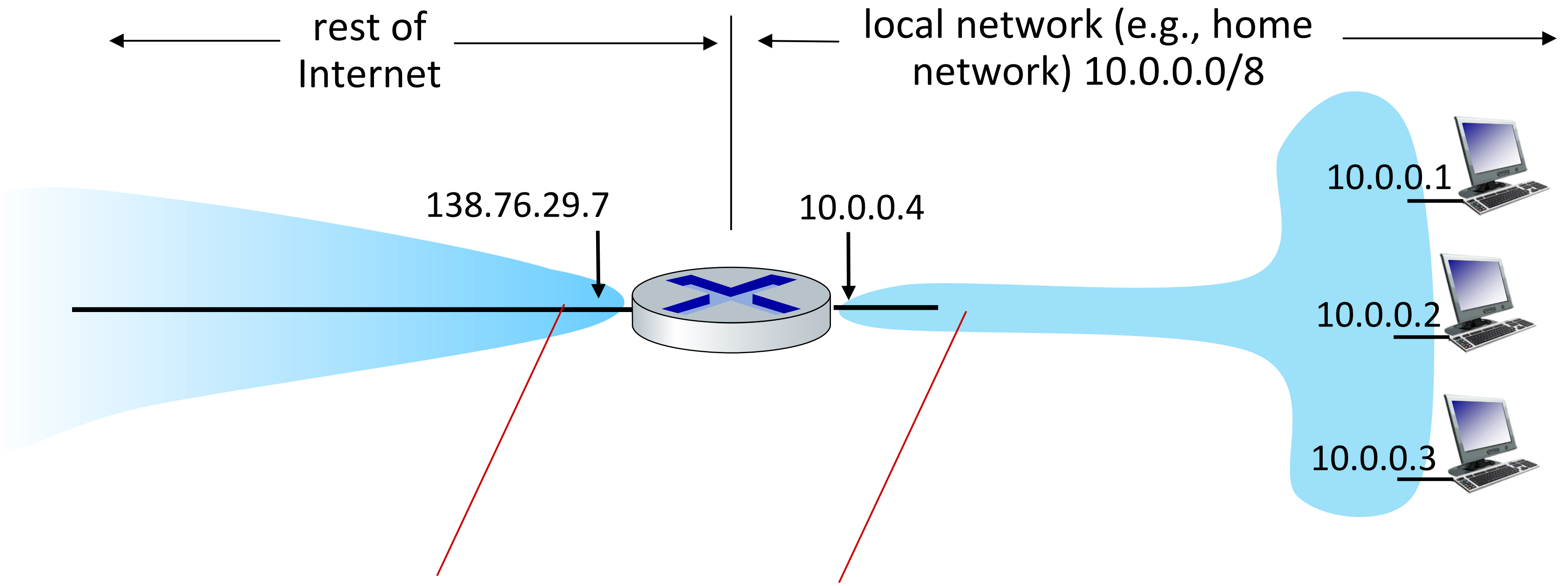
Q: are there enough 32-bit IP addresses?

- ICANN allocated last chunk of IPv4 addresses in 2011
- IPv6 has 128-bit address space
- NAT (next) helps IPv4 address space exhaustion

"Who the hell knew how much address space we needed?" - *Vint Cerf* (reflecting on decision to make IPv4 address 32 bits long)

NAT: network address translation

NAT: all devices in local network share just one IPv4 address as far as outside world is concerned



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

datagrams with source or destination in this network have 10.0.0.0/8 address for source, destination (as usual)

NAT: network address translation

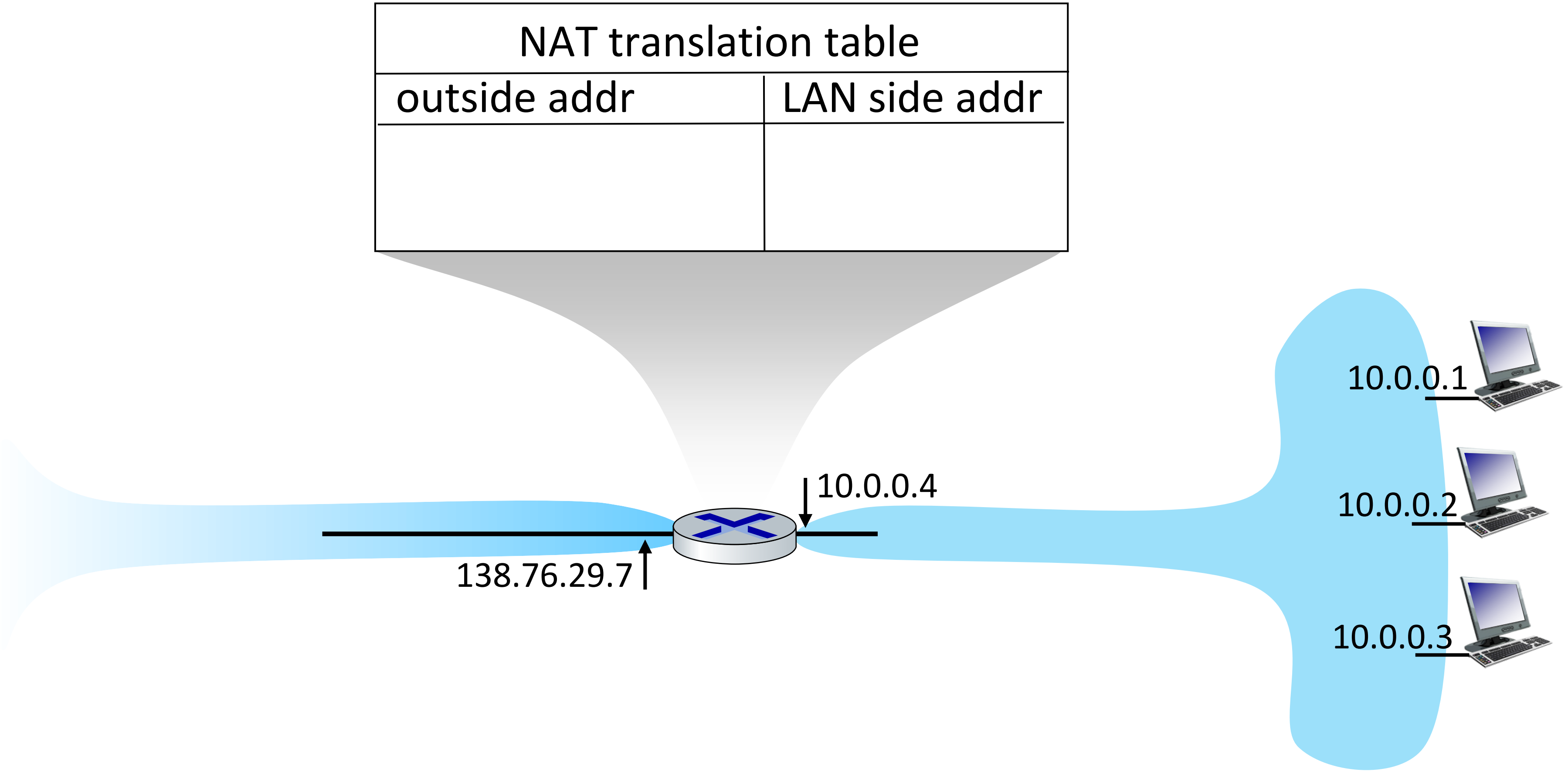
- all devices in local network have 32-bit addresses in a “private” IP address space (10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16 prefixes) that can only be used in local network
- advantages:
  - just one “public” IP address needed from provider ISP for *all* devices
  - can change addresses of host in local network without notifying outside world
  - can change ISP without changing addresses of devices in local network
  - security: devices inside local net not directly addressable, visible by outside world

NAT: network address translation

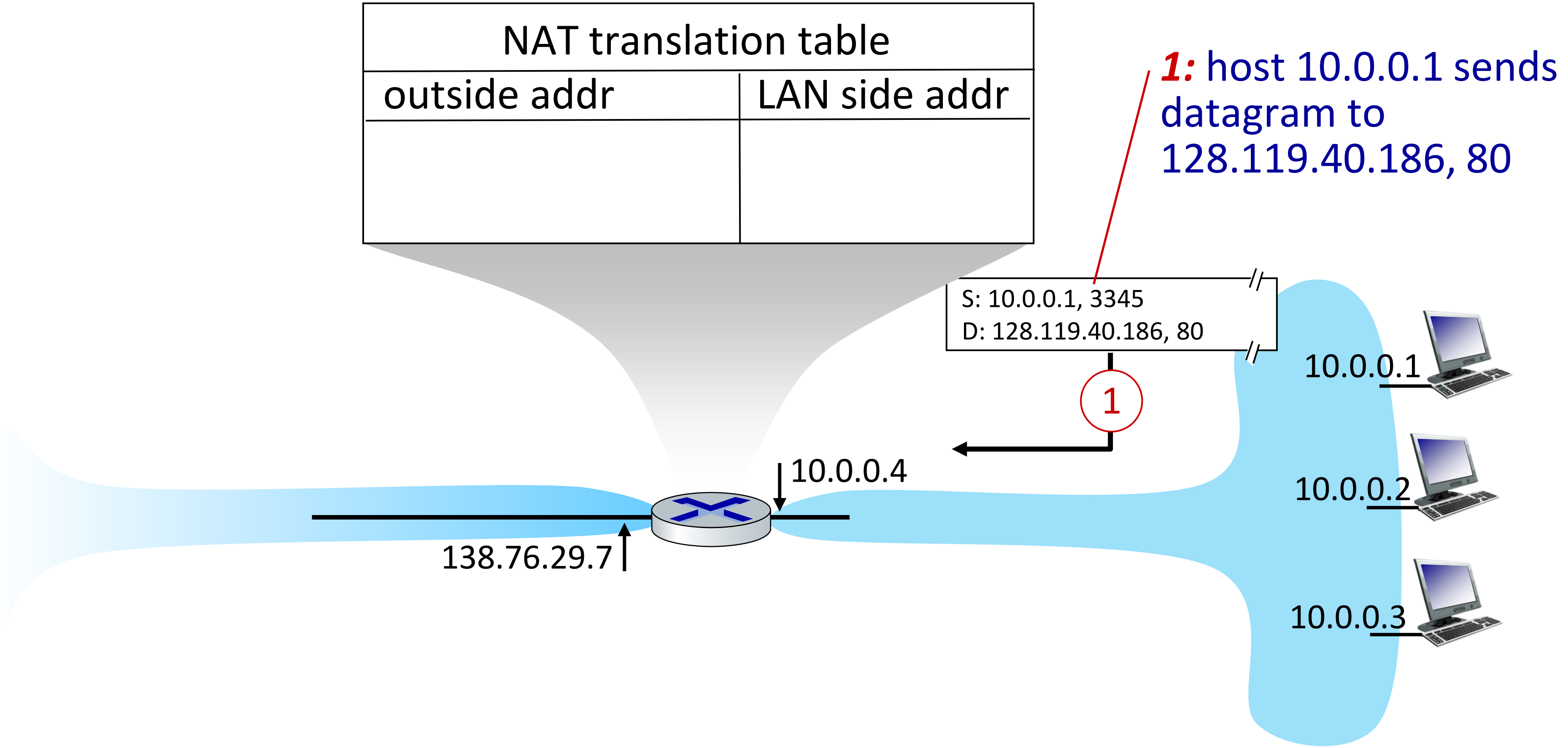
implementation: NAT router must (transparently):

- 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 address
- 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 destination fields of every incoming datagram with corresponding (source IP address, port #) stored in NAT table

NAT: network address translation

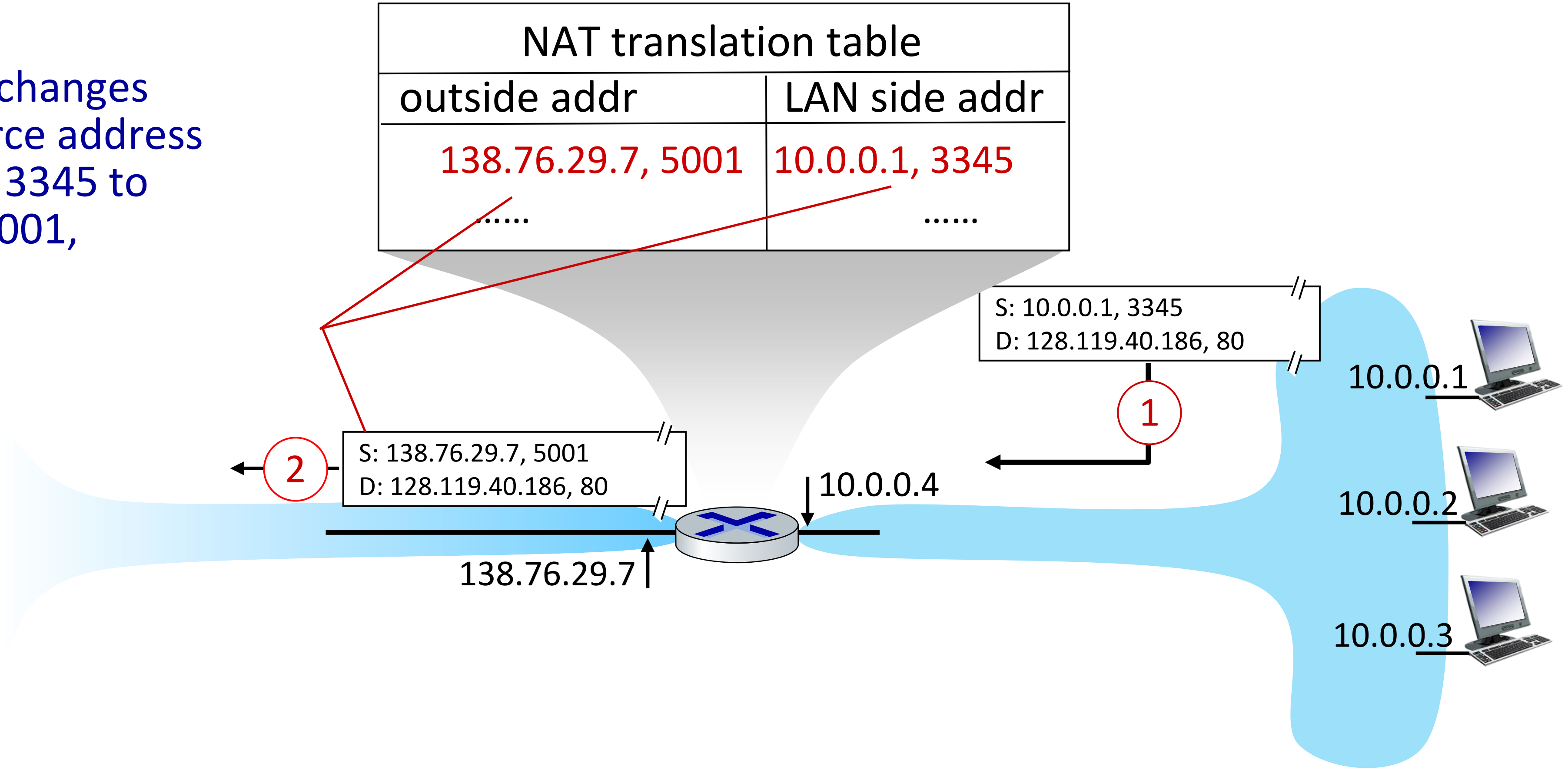


NAT: network address translation



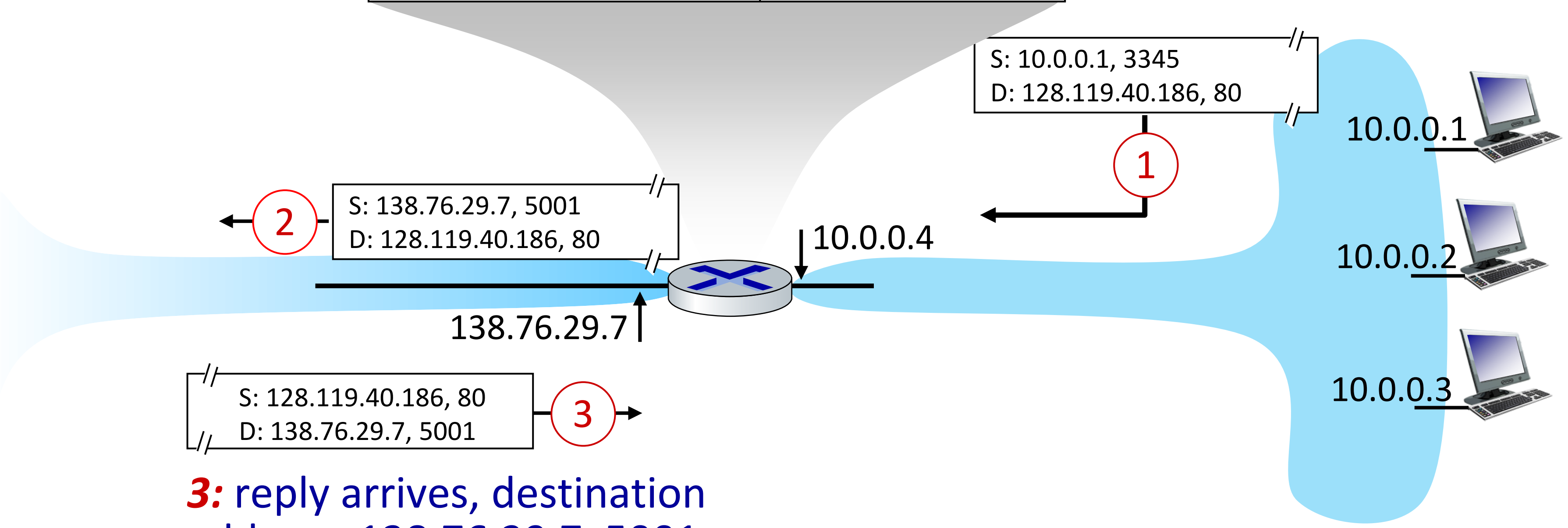
NAT: network address translation

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



NAT: network address translation

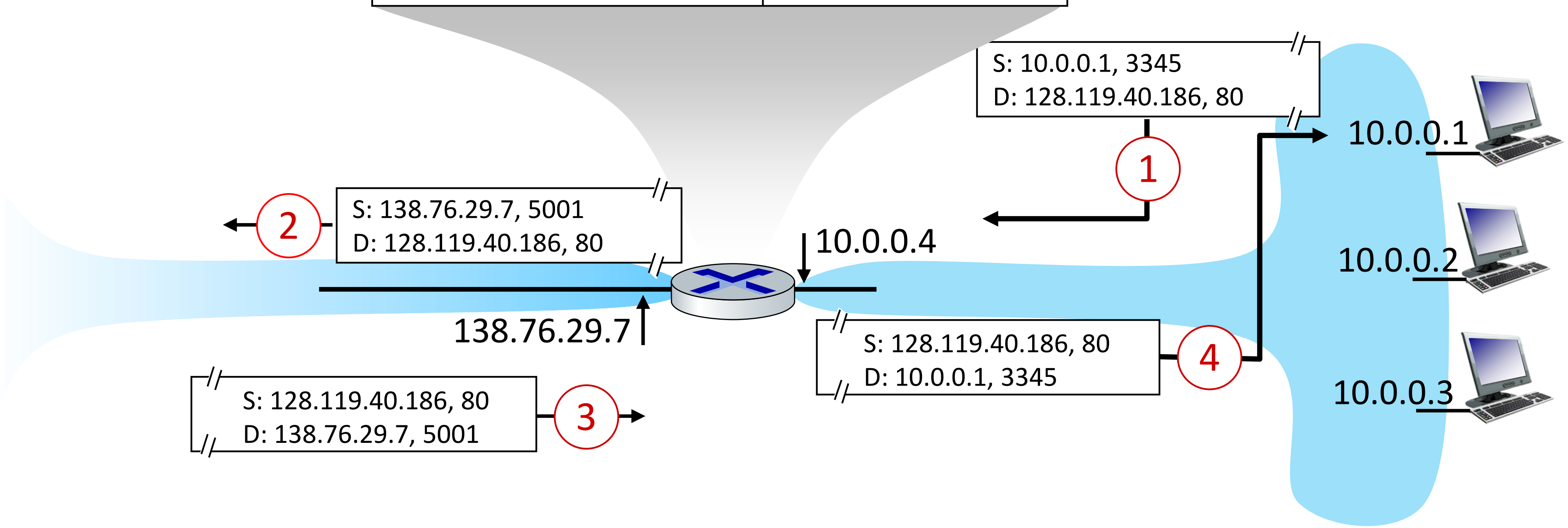
NAT translation table	
outside addr	LAN side addr
138.76.29.7, 5001	10.0.0.1, 3345
……	……



3: reply arrives, destination address: 138.76.29.7, 5001

NAT: network address translation

NAT translation table	
outside addr	LAN side addr
138.76.29.7, 5001	10.0.0.1, 3345
……	……



NAT: network address translation

- NAT has been controversial:
  - routers "should" only process up to layer 3
    - violates end-to-end argument (port # manipulation by network-layer device)
  - address "shortage" should be solved by IPv6
- but NAT is here to stay:
  - extensively used in home and institutional nets, 4G/5G cellular nets