NTNU – Trondheim
Norwegian University of
Science and Technology

Network Layer – Data Plane

TTM4200 – Computer Networks

individual Readiness Assurance Test

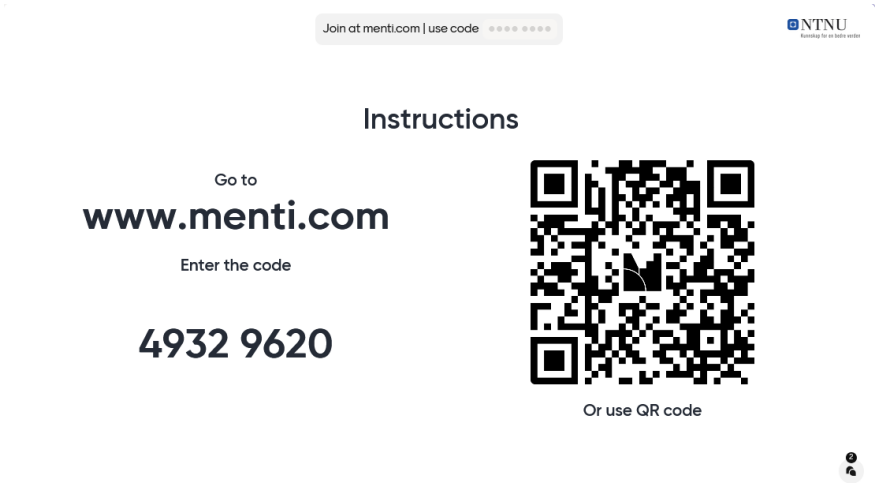
- Find the right answer **alone.**
- Choose the answer that fits **best.**
- **One** handwritten A4 page as helping material.

team Readiness Assurance Test

- Find the right answer **in teams**.
- Choose the answer that fits **best.**
- **One** handwritten A4 page as helping material.

Time for a break

we restart at 11:20



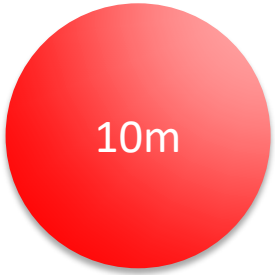
Problem 8 (*pg. 400*)

Consider a datagram network using 32-bit host addresses. Suppose a router has four links, numbered 0 through 3, and packets are to be forwarded to the link as follows:

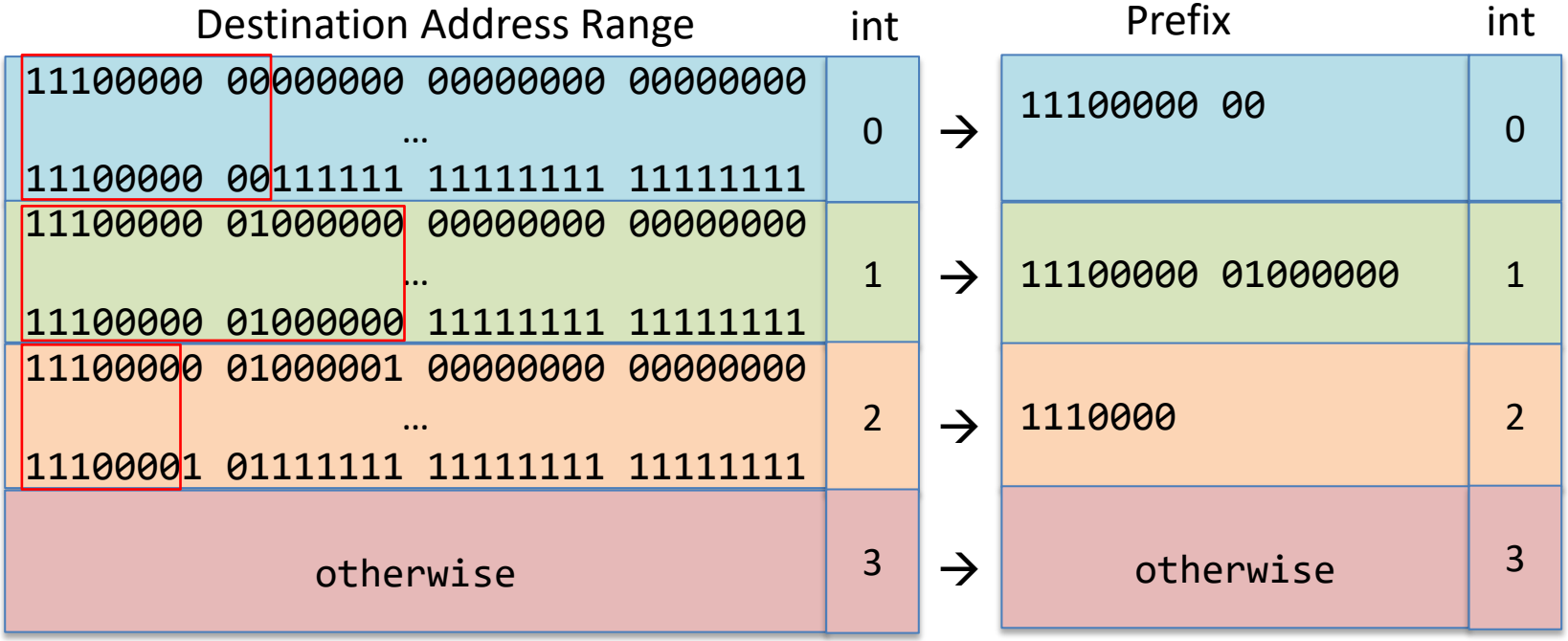
Destination Address Range	Link iface
11100000 00000000 00000000 00000000 through 11100000 00111111 11111111 11111111	0
11100000 01000000 00000000 00000000 through 11100000 01000000 11111111 11111111	1
11100000 01000001 00000000 00000000 through 11100001 01111111 11111111 11111111	2
otherwise	3

a) Provide a forwarding table that has **five entries**, uses longest prefix matching and forward packets to the correct interfaces.

b) …



Solving Problem 8 a) (*pg. 400)*



11100001 10000000 00000000 00000000 → int 2

misrouted

Solution Problem 8 a) (*pg. 400*)

Destination Address Range	int
11100000 00000000 00000000 00000000 … 11100000 00111111 11111111 11111111	0
11100000 01000000 00000000 00000000 … 11100000 01000000 11111111 11111111	1
11100000 01000001 00000000 00000000 … 11100001 01111111 11111111 11111111	2
otherwise	3

Prefix	int
11100000 00	0
11100000 01000000	1
1110000	2
11100001 1 otherwise	3

11100001 10000000 00000000 00000000 → int 3 ✓

Problem 8 b) *(pg. 400)*

Consider a datagram network using 32-bit host addresses. Suppose a router has four links, numbered 0 through 3, and packets are to be forwarded to the link as follows:

Prefix	int
11100000 00	0
11100000 01000000	1
1110000	2
11100001 1 otherwise	3

a) done

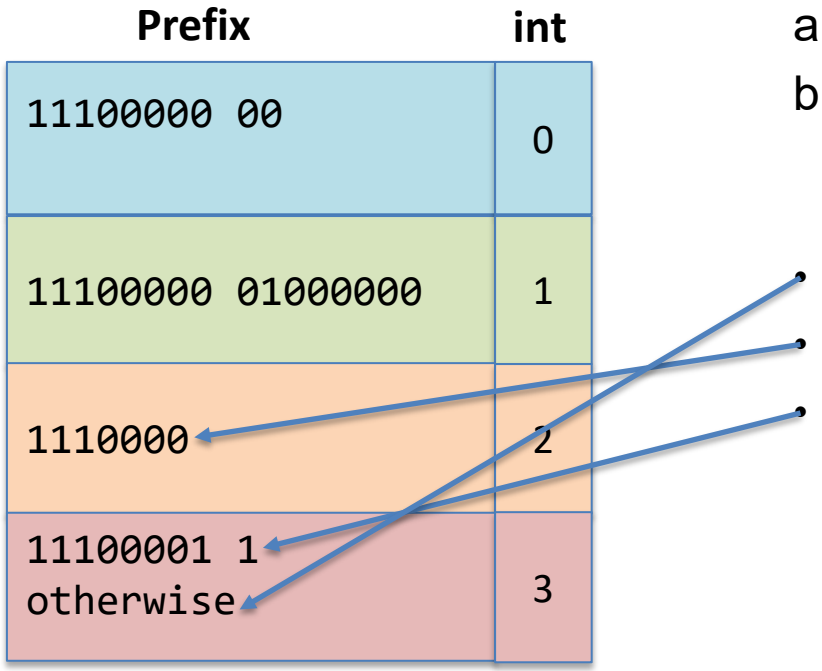
b) Determine how the forwarding table determines the appropriate link interface for these destination addresses:

- 11001000 10010001 01010001 01010101
- 11100001 01000000 11000011 00111100
- 11100001 10000000 00010001 01110111

5m

Solution Problem 8 b) (*pg. 400*)

Consider a datagram network using 32-bit host addresses. Suppose a router has four links, numbered 0 through 3, and packets are to be forwarded to the link as follows:



a) done

b) Determine how the forwarding table determines the appropriate link interface for these destination addresses:

- `11001000 10010001 01010001 01010101`
- `11100001 01000000 11000011 00111100`
- `11100001 10000000 00010001 01110111`

Lab 6

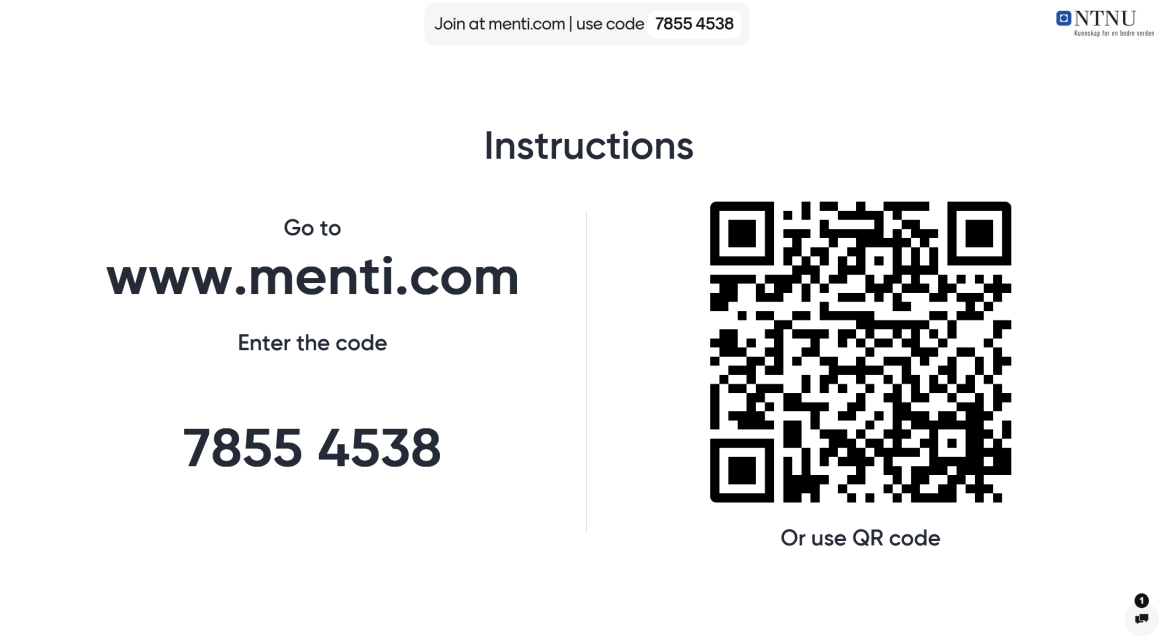
- Reliable vs. unreliable transport protocols
  - Simple UDP sender/receiver (unreliable)
  - Then implementing a reliable transport protocol on top of UDP
- Reliable transport protocol should handle packet loss, packet corruption, packet delays etc.
- We will implement the Go-Back-N protocol as an example of a reliable transport protocol

Enough for today

we continue Thursday at 10:15

Welcome

We start at 10:15



Problem 8 (*pg. 400)*

Consider a datagram network using 32-bit host addresses. Suppose a router has four links, numbered 0 through 3, and packets are to be forwarded to the link as follows:

Destination Address Range	Link iface
11100000 00000000 00000000 00000000 through 11100000 00111111 11111111 11111111	0
11100000 01000000 00000000 00000000 through 11100000 01000000 11111111 11111111	1
11100000 01000001 00000000 00000000 through 11100001 01111111 11111111 11111111	2
otherwise	3

a) Provide a forwarding table that has **five entries**, uses longest prefix matching and forward packets to the correct interfaces.

Solving Problem 8 a) (*pg. 400*)

- `11100000 00000000 00000000 00000000`
  `11100000 00111111 11111111 11111111`
  - `11100000 00` → interface 0

  224.0.0.0/10

- `11100000 01000000 00000000 00000000`
  `11100000 01000000 11111111 11111111`
  - `11100000 01000000` → interface 1

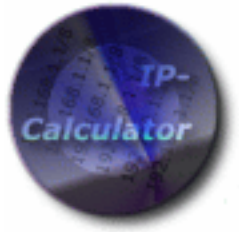
  224.64.0.0/16

- `11100000 01000001 00000000 00000000`
  `11100001 01111111 11111111 11111111`
  - `1110000` → interface 2

  224.0.0.0/7

- `11100001 10000000 00000000 00000000`
  `11100001 11111111 11111111 11111111`
  - `11100001 1` → interface 3

  225.128.0.0/9



Address (Host or Network) **Netmask** (i.e. 24) **Netmask** for sub/supernet (optional)

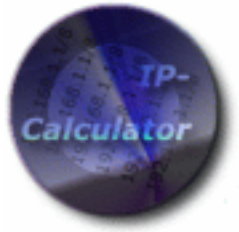
224.0.0.0 / 7 move to:

Calculate Help

```
Address:   224.0.0.0           1110000 0.00000000.00000000.00000000
Netmask:   254.0.0.0 = 7       1111111 0.00000000.00000000.00000000
Wildcard:  1.255.255.255       0000000 1.11111111.11111111.11111111
=>
Network:   224.0.0.0/7         1110000 0.00000000.00000000.00000000 (Class D)
Broadcast: 225.255.255.255     1110000 1.11111111.11111111.11111111
HostMin:   224.0.0.1           1110000 0.00000000.00000000.00000001
HostMax:   225.255.255.254     1110000 1.11111111.11111111.11111110
Hosts/Net: 33554430
```

IP-Calculator

Version 0.35.2 2005/07/07



Address (Host or Network) **Netmask** (i.e. 24) **Netmask** for sub/supernet (optional)

225.128.0.0 / 9 move to:

Calculate Help

```
Address:   225.128.0.0          11100001.1 0000000.00000000.00000000
Netmask:   255.128.0.0 = 9      11111111.1 0000000.00000000.00000000
Wildcard:  0.127.255.255        00000000.0 1111111.11111111.11111111
=>
Network:   225.128.0.0/9        11100001.1 0000000.00000000.00000000 (Class D)
Broadcast: 225.255.255.255      11100001.1 1111111.11111111.11111111
HostMin:   225.128.0.1          11100001.1 0000000.00000000.00000001
HostMax:   225.255.255.254      11100001.1 1111111.11111111.11111110
Hosts/Net: 8388606
```

IP-Calculator

Version 0.35.2 2005/07/07

RAT Revision

Question 5: Active Queue Management algorithms were proposed for:

\textbf{proactively} marking and dropping packet \textbf{before} buffers are full. An example is the Random Early Detection (RED) algorithm. (This is the best answer.)

68.1%

\textbf{reactively} dropping newly arriving packets \textbf{when} buffers are full. An example is the drop-tail policy.

12.8%

\textbf{proactively} marking and dropping packet \textbf{before} buffers are full. An example is the drop-tail policy.

17.0%

\textbf{reactively} dropping newly arriving packets \textbf{when} buffers are full. An example is the Random Early Detection (RED) algorithm.

2.1%

RAT Revision

Question 6: With Non-preemptive Priority Queuing:

Arriving high-priority packets \textbf{may} have to wait until a low-priority packet is transmitted. (This is the best answer.)

53.2%

Arriving high-priority packets will \textbf{always} be sent before low-priority packets.

36.2%

A queued low-priority packet \textbf{may} be transmitted even before the high-priority queue is emptied.

6.4%

Arriving packets are \textbf{always} classified into high-priority if the high-priority queue is empty.

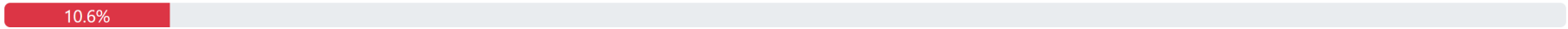
4.3%

RAT Revision

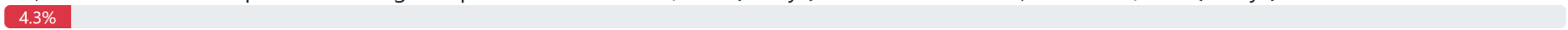
Question 7: Can Weighted Fair Queueing (WFQ) and Round Robin (RR) behave the same?

Yes, when all classes in WFQ have the same amount of service weight. (This is the best answer.)

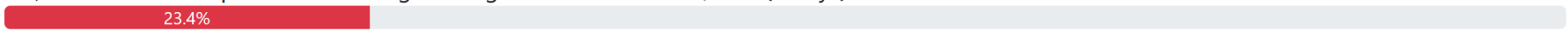
Yes, when all packets arriving in WFQ are placed within the same class.



No, because with WFQ packets arriving to a prioritised class will \textbf{always} leave before others, while RR \textbf{always} alternates between classes.



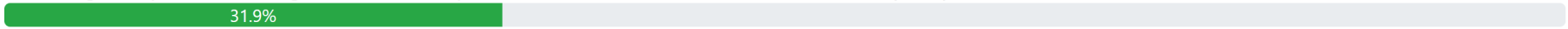
No, because WFQ depends on the weight of a given class while RR \textbf{always} alternates between classes.



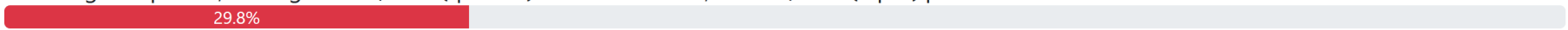
RAT Revision

Question 8: A packet scheduler is responsible for:

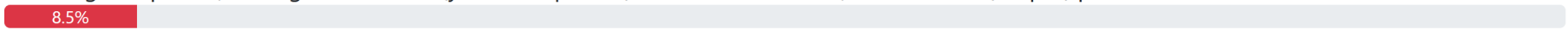
choosing one packet, among those \textbf{queued} for transmission, at the \textbf{output} port. (This is the best answer.)



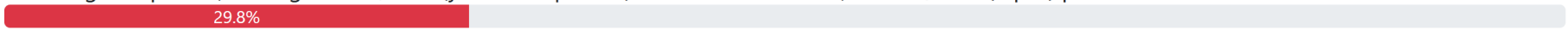
choosing one packet, among those \textbf{queued} for transmission, at the \textbf{input} port.



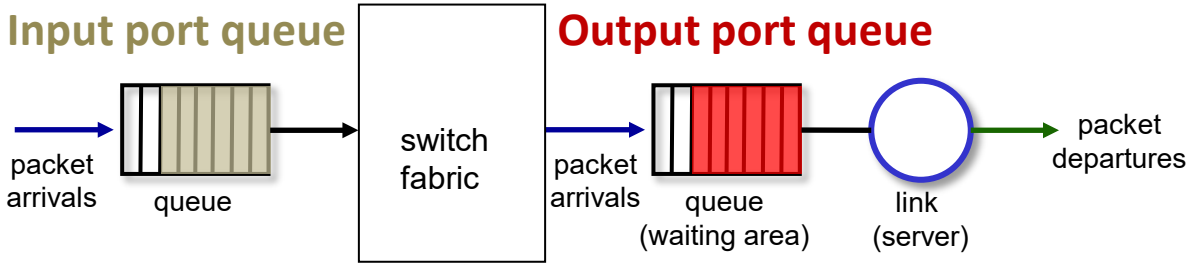
choosing one packet, among those \textbf{yet to be queued} before transmission, at the \textbf{output} port.



choosing one packet, among those \textbf{yet to be queued} before transmission, at the \textbf{input} port.



Queue (Buffer) Management vs Packet Scheduling



At input port

Buffering is required when

- Arrival rate is faster than switching fabric transfer rate (Faster switching fabrics can resolve this)
- Multiple input ports compete for the same output port
- Output port is busy, so datagrams cannot leave (Why? Where is the bottleneck?)

At output port

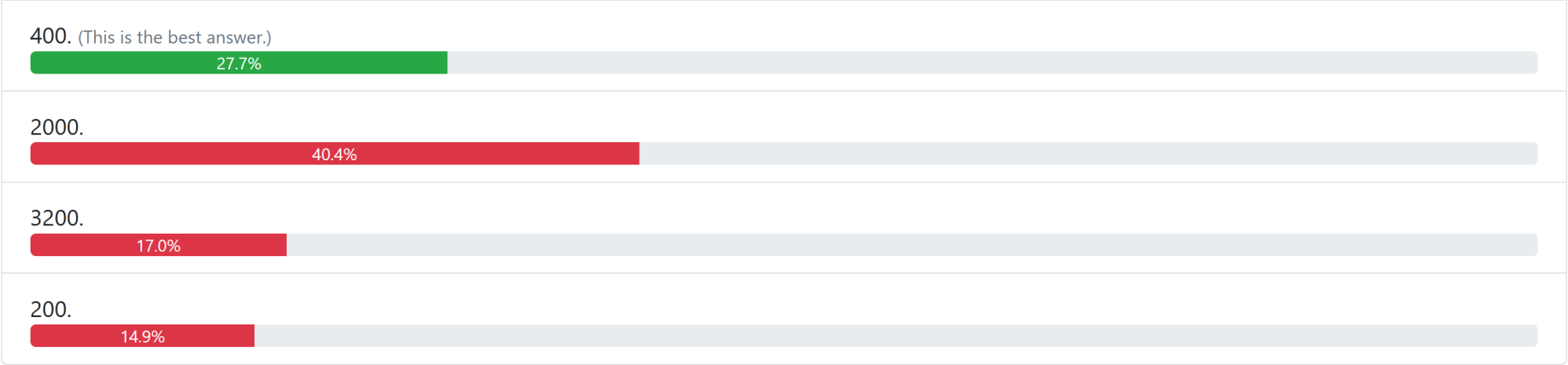
Buffering is required when datagrams arrive from fabric (arrival rate via switch is) faster than **link transmission rate.**

Queue Management - *Dropping (and marking) policy:* which queued datagrams to drop if no free buffers?

Packet ***Scheduling*** – in which order queued datagrams are to be served (send on transmission link)? FCFS, Priority Queuing, RR, WFQ

RAT Revision

Question 10: leave this one for last.\\\newline Consider a TCP Reno connection that exclusively uses a $20\,\textrm{Mbps}$ link. Suppose that this link is \textbf{the only congested link} between the two hosts participating in the connection. Consider a \textbf{large file transfer} under the following assumptions: \begin{itemize} \item the receive buffer of the receiving host is \textbf{much larger} than the congestion window; \item the TCP segment size equals 1250 bytes; \item the RTT equals $200\,\textrm{ms}$; \item the TCP connection is \textbf{always in the congestion avoidance phase}, ignoring slow start; \item packet headers are not considered. \end{itemize} What is the maximum window size (\textbf{in segments}) that this TCP connection can achieve?\newline\newline \textbf{Tip:} Remember that the congestion window cannot increase before an ACK for the corresponding segment is received, which takes RTT seconds.



- We’re asked for the maximum congestion window (in segments) for a TCP Reno flow over a single 20 Mbps bottleneck link, with RTT = 200 ms and MSS = 1250 B

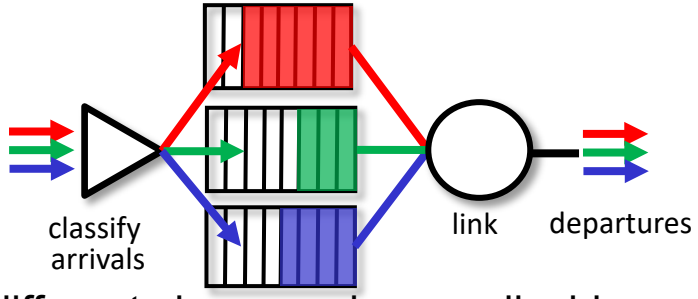
- Assumption:
  - Large file transfer
  - Receiver window ≫ congestion window
  - Always in congestion avoidance (ignore slow start)
  - No header overhead considered
  - This is the only congested link

- Under these conditions, the sender can at most fill the bandwidth-delay product (BDP) of the path. That’s because the congestion window cannot grow until ACKs return (after one RTT), so the “in-flight” data must be enough to keep the pipe full.
- Thus, the maximum useful cwnd (before losses force a reduction) is:

$$\text{cwnd}_{\max} \approx \frac{\text{Capacity} \times \text{RTT}}{\text{MSS}}$$

- BDP (bits) = 20,000,000 bps×0.2 s=4,000,000
- BDP (bytes) = 4,000,000/8=500,000 bytes
- Segments = 500,000/1250=400

Queueing Problem



Consider the following packet arrivals considering their different classes, when applicable. Assume that transmitting a packet lasts for 1 time instant.

Class (prio)	t1	t2	t3	t4	t5	t6	t7
1 (high)	p1	p3	p4		p9	p11	
2 (normal)			p5, p6	p8			
3 (low)	p2		p7		p10		p12

What is the leaving/departure order of the packets using FIFO (FCFS), Priority, Round Robin?

Extra: Weighted Fair Queuing (w1=2, w2=1, w3=1)?

Solving Queueing Problem

Class (prio)	t1	t2	t3	t4	t5	t6	t7
1 (high)	p1	p3	p4		p9	p11	
2 (normal)			p5, p6	p8			
3 (low)	p2		p7		p10		p12

Q	t1	t2	t3	t4	t5	t6	t7	t8	t9	t10	t11	t12
1												
2												
3												

	t1	t2	t3	t4	t5	t6	t7	t8	t9	t10	t11	t12
Output												

Solving Queueing Problem (FIFO)

Class (prio)	t1	t2	t3	t4	t5	t6	t7
1 (high)	p1	p3	p4		p9	p11	
2 (normal)			p5, p6	p8			
3 (low)	p2		p7		p10		p12

Q	t1	t2	t3	t4	t5	t6	t7	t8	t9	t10	t11	t12
1	p1 p2	p2 p3	p3 p4 p5 p6 p7	p4 p5 p6 p7 p8	p5 p6 p7 p8 p9 p10	p6 p7 p8 p9 P10 p11	p7 p8 p9 p10 p11 p12	p8 p9 p10 p11 p12	p9 p10 p11 p12	p10 p11 p12	p11 p12	p12

	t1	t2	t3	t4	t5	t6	t7	t8	t9	t10	t11	t12
Output	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12

Solving Queueing Problem (Priority)

Class (prio)	t1	t2	t3	t4	t5	t6	t7
1 (high)	p1	p3	p4		p9	p11	
2 (normal)			p5, p6	p8			
3 (low)	p2		p7		p10		p12

Q	t1	t2	t3	t4	t5	t6	t7	t8	t9	t10	t11	t12
1	**p1**	**p3**	**p4**		**p9**	**p11**						
2			p5, p6	**p5**, p6, p8	p6, p8	p6, p8	**p6**, p8	**p8**				
3	p2	p2	p2, p7		p2, p7, p10	p2, p7, p10	p2, p7, p10, p12	p2, p7, p10, p12	**p2**, p7, p10, p12	**p7**, p10, p12	**p10**, p12	**p12**

	t1	t2	t3	t4	t5	t6	t7	t8	t9	t10	t11	t12
Output	p1	p3	p4	p5	p9	p11	p6	p8	p2	p7	p10	p12

Solving Queueing Problem (RR)

Class (prio)	t1	t2	t3	t4	t5	t6	t7
1 (high)	p1	p3	p4		p9	p11	
2 (normal)			p5, p6	p8			
3 (low)	p2		p7		p10		p12

Q	t1	t2	t3	t4	t5	t6	t7	t8	t9	t10	t11	t12
1												
2												
3												

	t1	t2	t3	t4	t5	t6	t7	t8	t9	t10	t11	t12
Output												

Solving Queueing Problem (RR)

Class (prio)	t1	t2	t3	t4	t5	t6	t7
1 (high)	p1	p3	p4		p9	p11	
2 (normal)			p5, p6	p8			
3 (low)	p2		p7		p10		p12

Q	t1	t2	t3	t4	t5	t6	t7	t8	t9	t10	t11	t12
1	**p1**	p3	**p3**, p4		p4, p9	**p4**, p9 p11			**p9**, p11			**p11**
2			p5, p6	**p5**, p6, p8	p6, p8	p6, p8	**p6**, p8		p8	**p8**		
3	p2	**p2**	p7	p7	**p7**, p10	p10	p10, p12	**p10**, p12	p12	p12	**p12**	

	t1	t2	t3	t4	t5	t6	t7	t8	t9	t10	t11	t12
Output	p1	p2	p3	p5	p7	p4	p6	p10	p9	p8	p12	p11

Problem 10 (*pg. 401)*

Consider a datagram network using 8-bit host addresses. Suppose a router uses longest prefix matching and has the following forwarding table:

Prefix Match	Link iface
1	0
10	1
111	2
otherwise	3

For each of the four interfaces, give the associated range of destinations host addresses and the number of addresses in the range.

Solving Problem 10

(pg. 401)

Prefix Match	Link iface
1	0
10	1
111	2
otherwise	3

- **interface 0** – 1<u>0000000</u> through 1<u>1111111</u> (2^7 addresses)
- **interface 1** – 10<u>000000</u> through 10<u>111111</u> (2^6 addresses)
- **interface 2** – 111<u>00000</u> through 111<u>11111</u> (2^5 addresses)
- **interface 3** – 00000000 through 01111111 (2^7 addresses)

Time for a break

we restart at 11:20

Join at menti.com | use code 7855 4538

NTNU

Instructions

Go to

www.menti.com

Enter the code

7855 4538

Or use QR code

Readings for next week

Book pages

- 360-381 (4.3.1-4)

Optional:

- 390-393 (Sec. 4.5)

Key concepts

IPv4/IPv6 Addressing

From IPv4 to IPv6

Subnet & subnet mask

DHCP

NAT

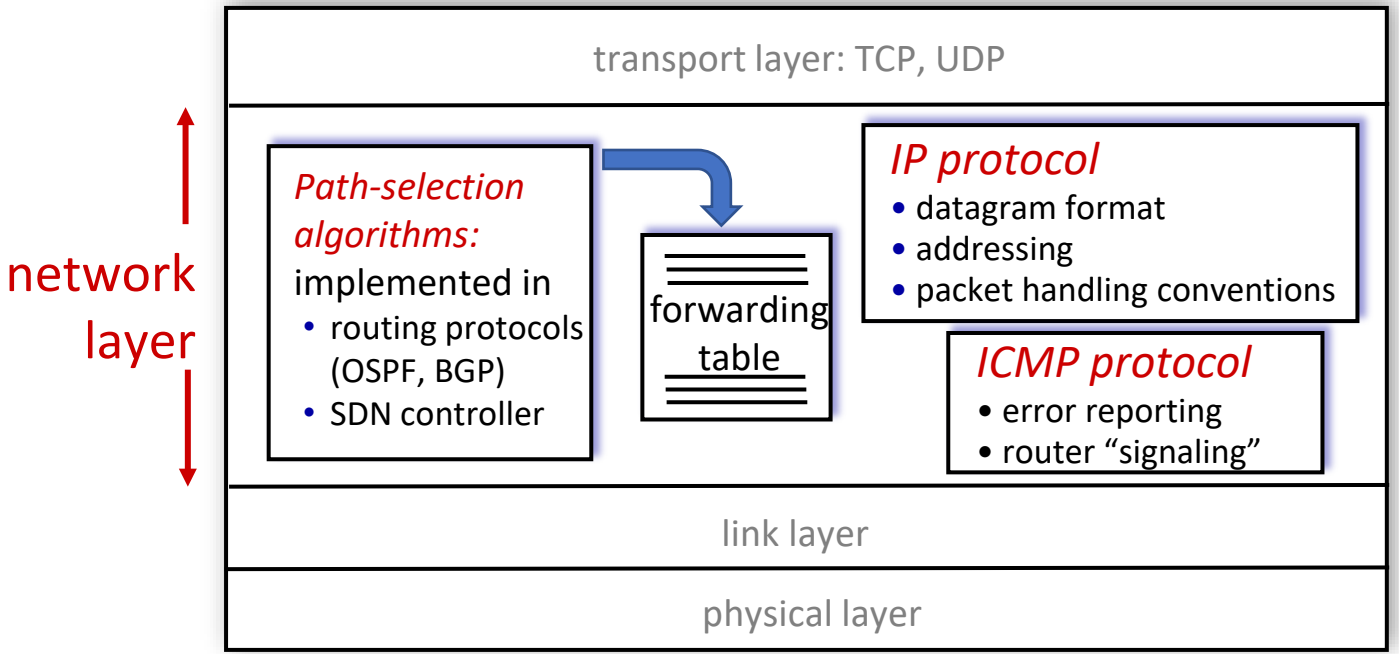
Network layer: "data plane" roadmap

- Network layer: overview
  - data plane
  - control plane
- What's inside a router
  - input ports, switching, output ports
  - buffer management, scheduling
- IP: the Internet Protocol
  - datagram format
  - addressing
  - network address translation
  - IPv6

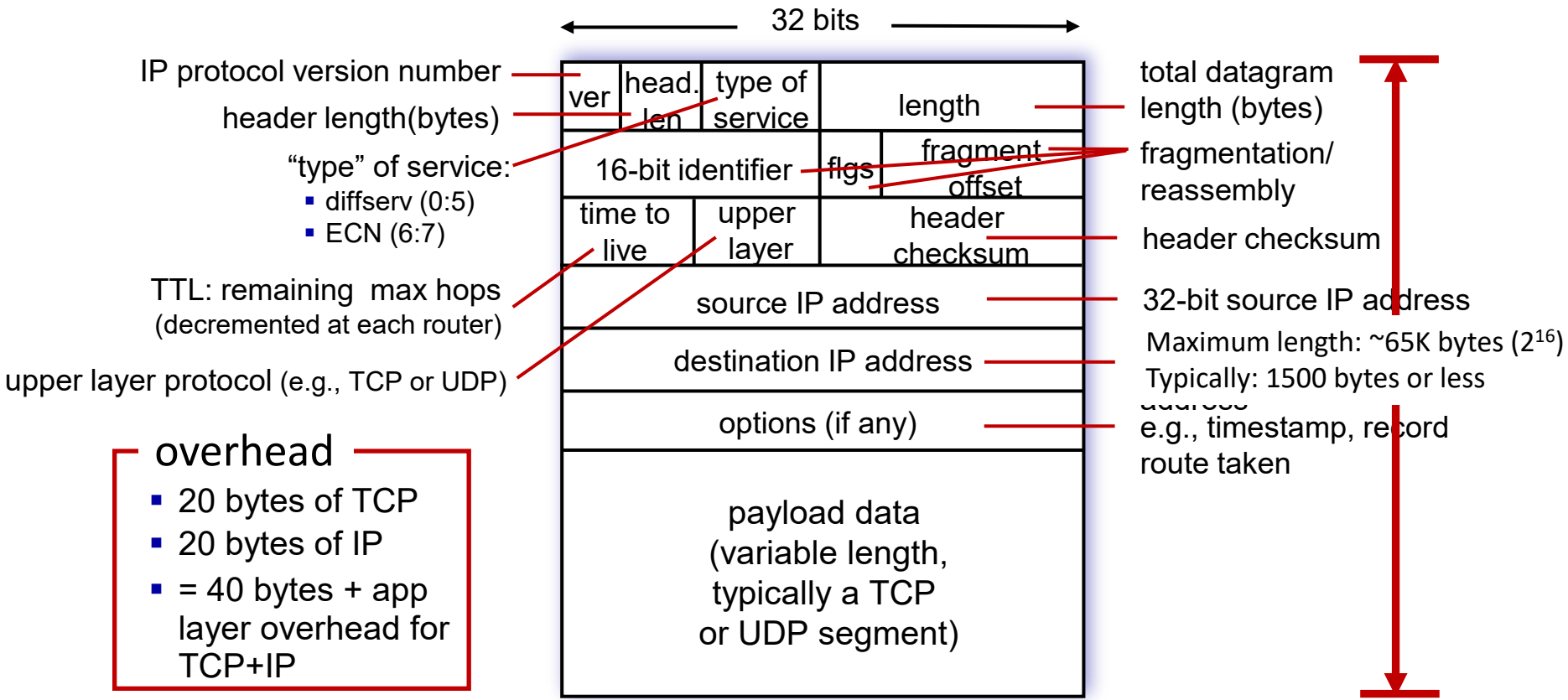
- Generalized Forwarding, SDN
  - match+action
  - OpenFlow: match+action in action
- Middleboxes

Network Layer: Internet

host, router network layer functions:



IPv4 Datagram format

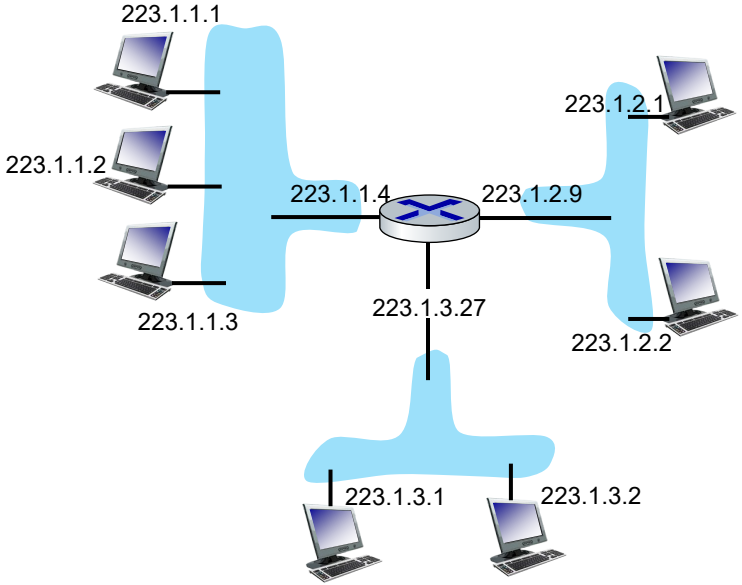


overhead

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

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
  - host typically has one or two interfaces (e.g., wired Ethernet, wireless 802.11)



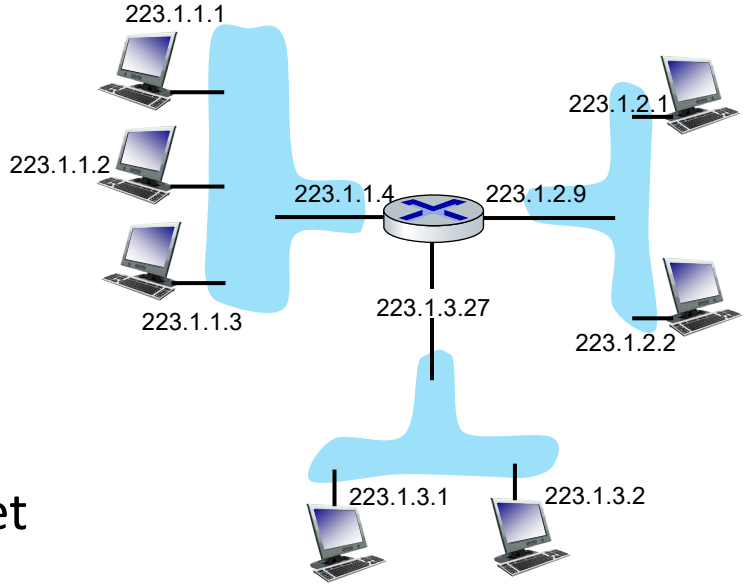
dotted-decimal IP address notation:

223.1.1.1 = 11011111 00000001 00000001 00000001

223 1 1 1

Subnets

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

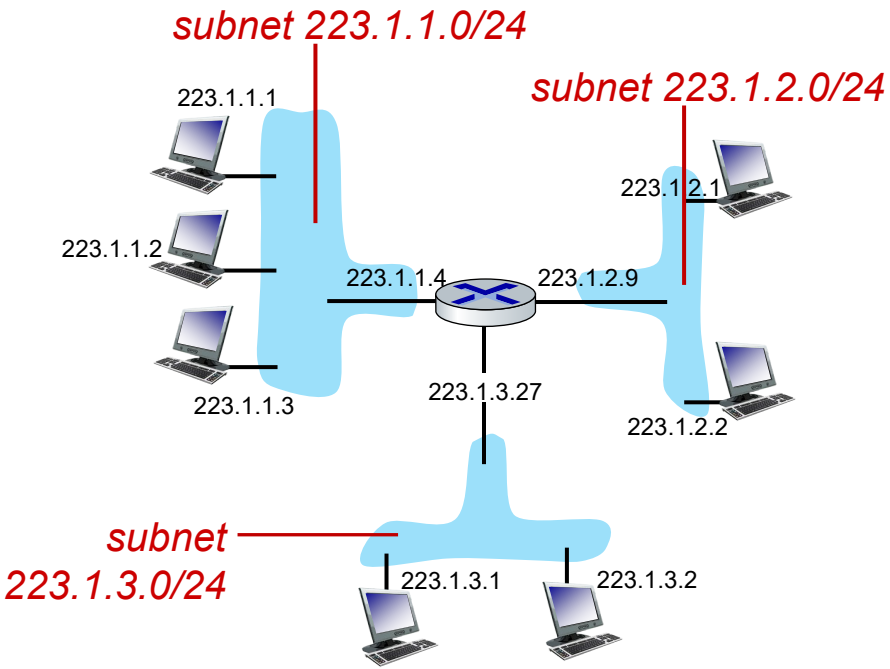


network consisting of 3 subnets

Subnets

Recipe for defining subnets:

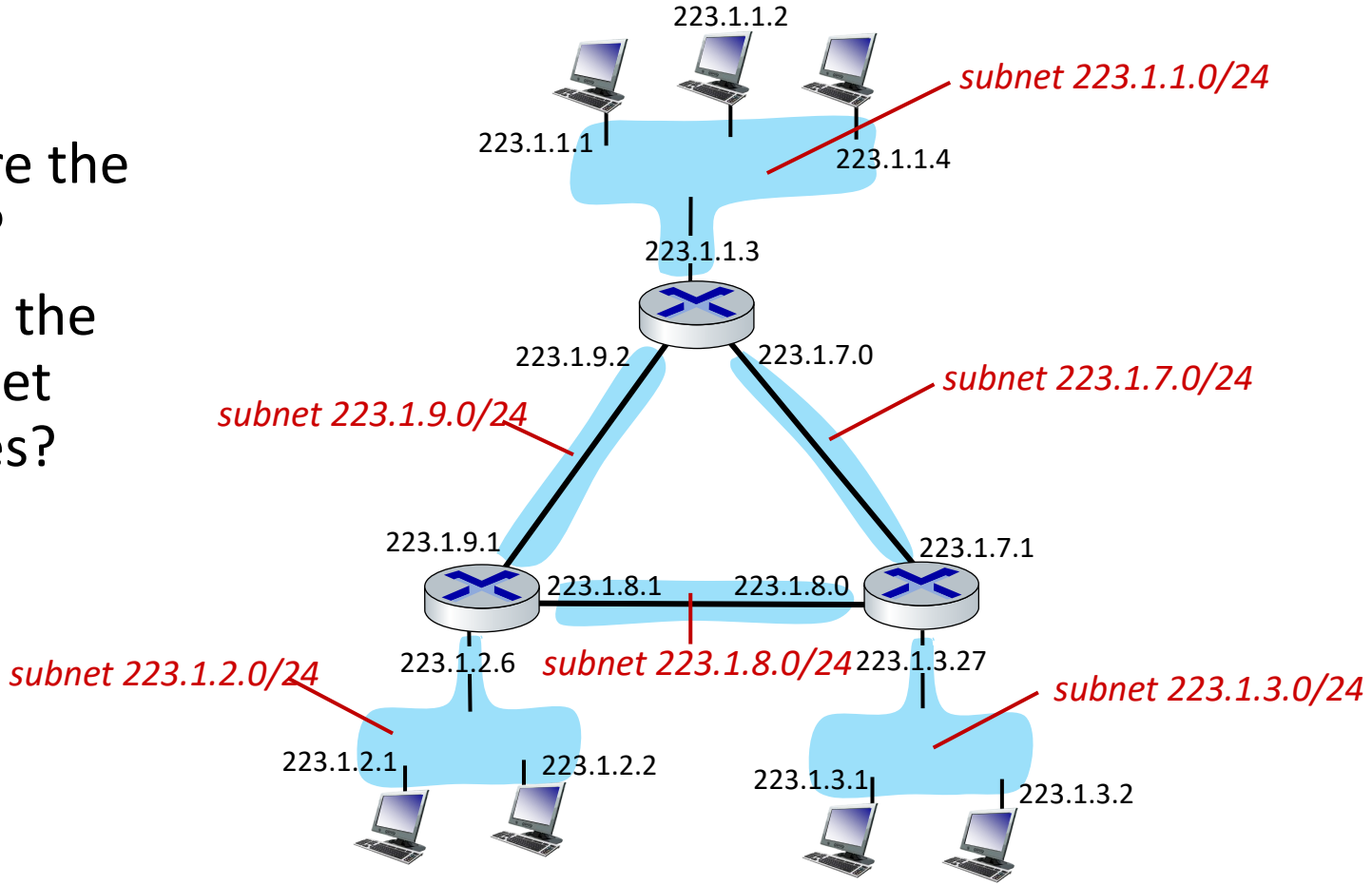
- detach each interface from its host or router, creating "islands" of isolated networks
- each isolated network is called a *subnet*



subnet mask: /24
(high-order 24 bits: subnet part of IP address)

Subnets

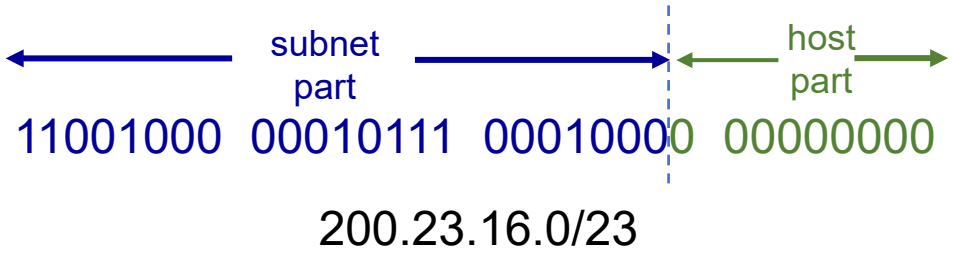
- where are the subnets?
- what are the /24 subnet addresses?



IP addressing: Internet's address assignment strategy

CIDR: Classless InterDomain Routing (pronounced "cider")

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



Q: How many hosts can this subnet support?

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 *host* get IP address?

- hard-coded by sysadmin in config file (e.g., /etc/rc.config in UNIX)
- DHCP: Dynamic Host Configuration Protocol: dynamically get address from as server
  - "plug-and-play"

DHCP: Dynamic Host Configuration Protocol

goal: host *dynamically* obtains 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/on)
- support for mobile users who join/leave 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: more than IP addresses

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

- address of first-hop router for client
- name and IP address of DNS sever
- network mask (indicating network versus host portion of address)

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

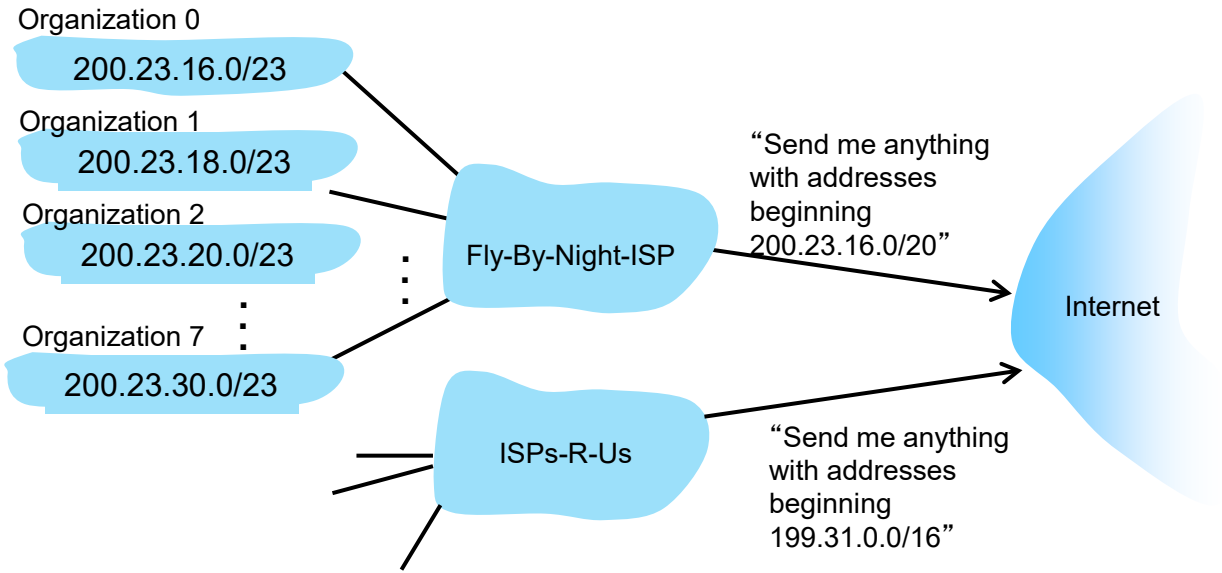
ISP's block	11001000 00010111 00010000 00000000	200.23.16.0/20

ISP can then allocate out its address space in, e.g., 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

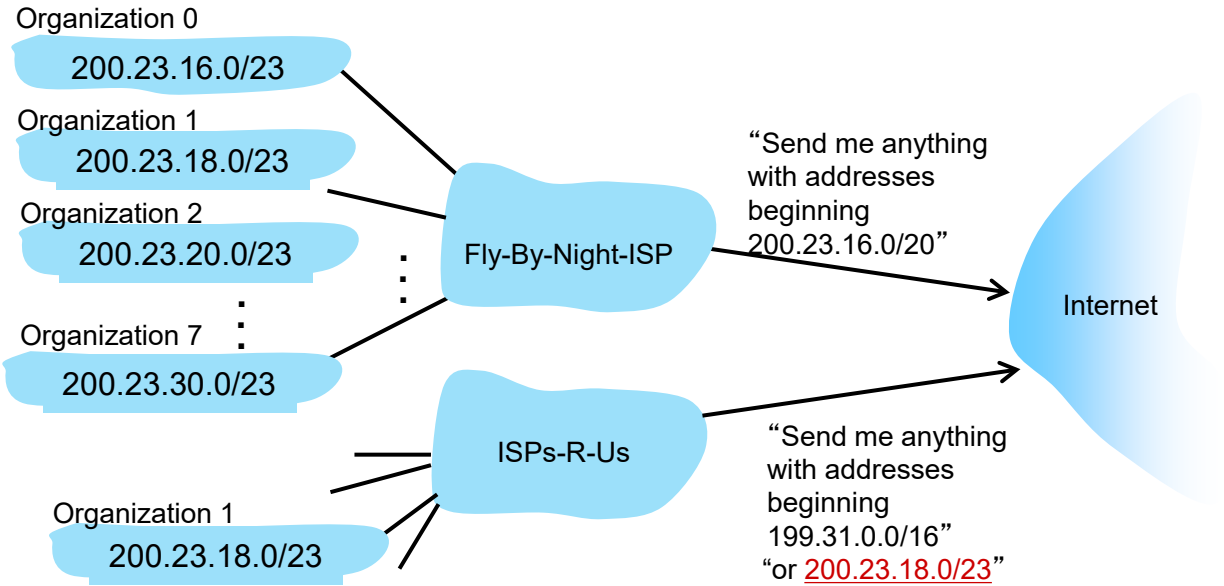
Hierarchical addressing: route aggregation

hierarchical addressing allows efficient advertisement of routing information:



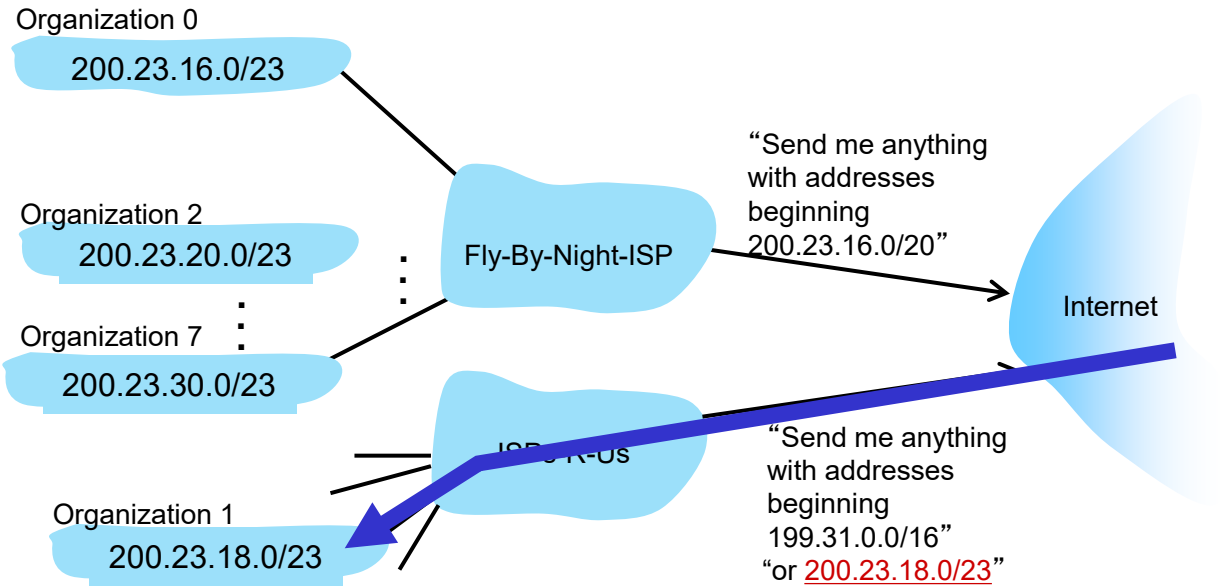
Hierarchical addressing: more specific routes

- Organization 1 moves from Fly-By-Night-ISP to ISPs-R-Us
- ISPs-R-Us now advertises a more specific route to Organization 1



Hierarchical addressing: more specific routes

- Organization 1 moves from Fly-By-Night-ISP to ISPs-R-Us
- ISPs-R-Us 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/

- allocates IP addresses, through 5 regional registries (RRs) (who may then allocate to local registries)
- manages DNS root zone, including delegation of individual TLD (.com, .edu , …) management

Q: are there enough 32-bit IP addresses?

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

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

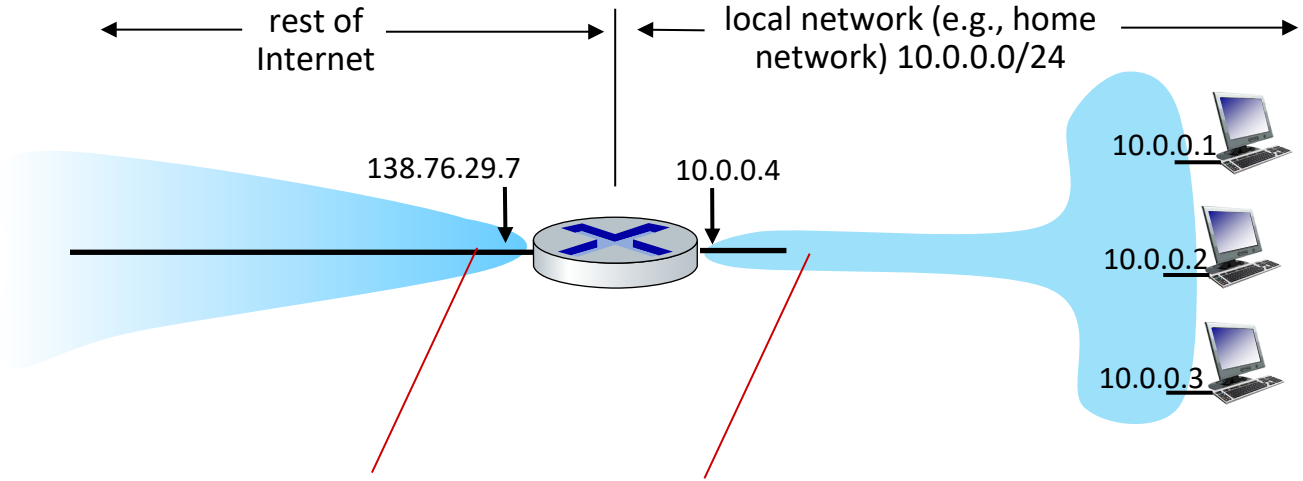
Network layer: "data plane" roadmap

- Network layer: overview
  - data plane
  - control plane
- What's inside a router
  - input ports, switching, output ports
  - buffer management, scheduling
- IP: the Internet Protocol
  - datagram format
  - addressing
  - network address translation
  - IPv6

- Generalized Forwarding, SDN
  - match+action
  - OpenFlow: match+action in action
- Middleboxes

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 10.0.0.0/24 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 (10/8, 172.16/12, 192.168/16 prefixes) that can only be used in local network
- advantages:
  - just one 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

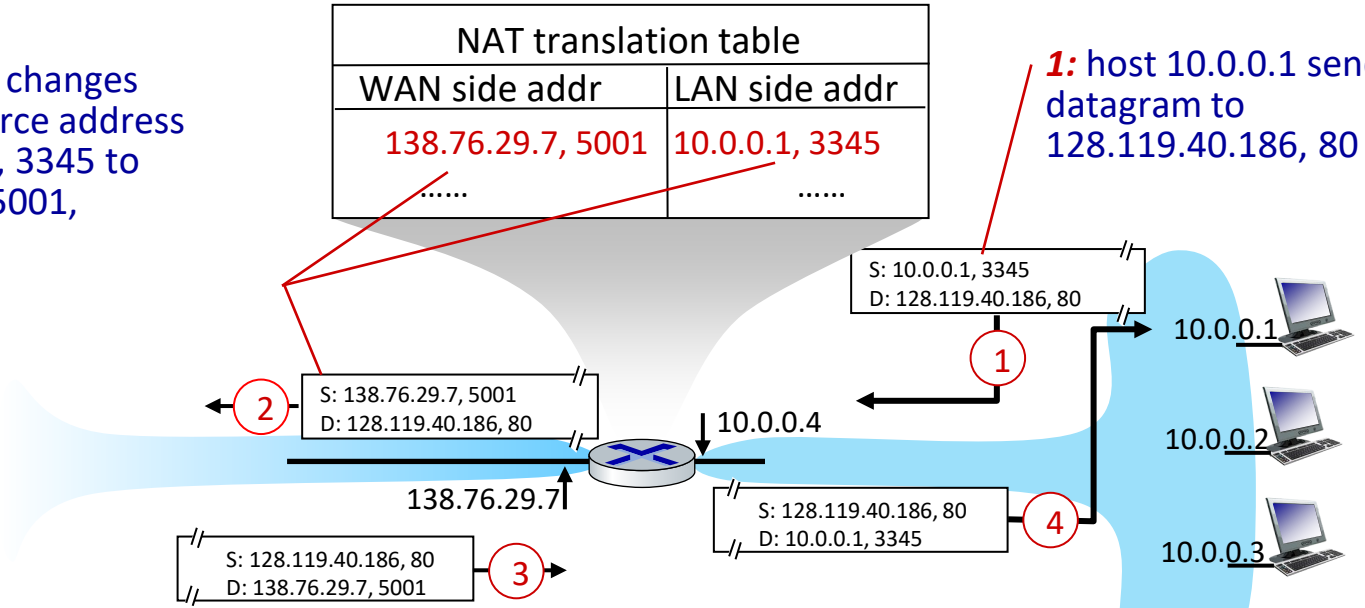
NAT: network address translation

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

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



1: host 10.0.0.1 sends datagram to 128.119.40.186, 80

3: reply arrives, destination address: 138.76.29.7, 5001

NAT: network address translation

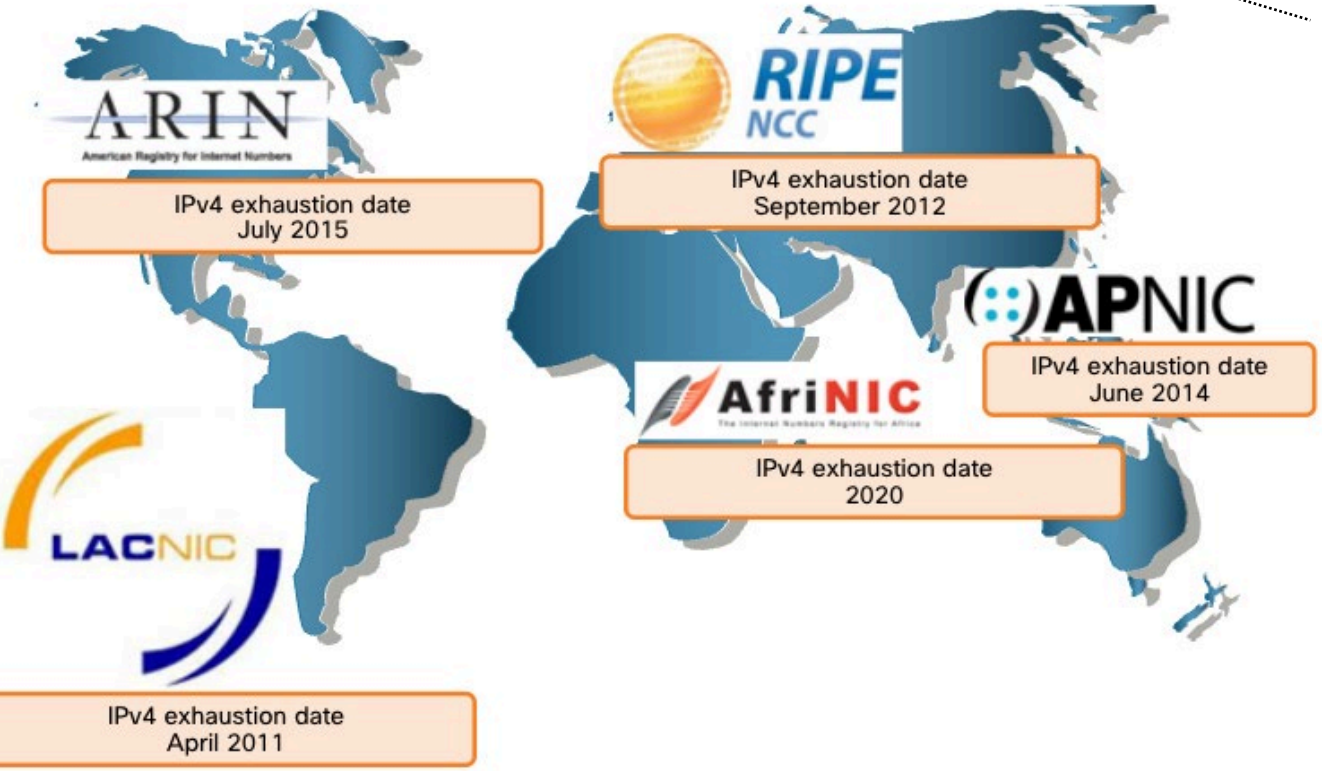
- NAT has been controversial:
  - routers "should" only process up to layer 3
  - address "shortage" should be solved by IPv6
  - violates end-to-end argument (port # manipulation by network-layer device)
  - NAT traversal: what if client wants to connect to server behind NAT?
- but NAT is here to stay: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

IPv6: motivation

- initial motivation: 32-bit IPv4 address space would be completely allocated
- additional motivation:
  - speed processing/forwarding: 40-byte fixed length header
  - enable different network-layer treatment of "flows"

RIR IPv4 Exhaustion dates

Internet Assigned Numbers Authority
Top-Level

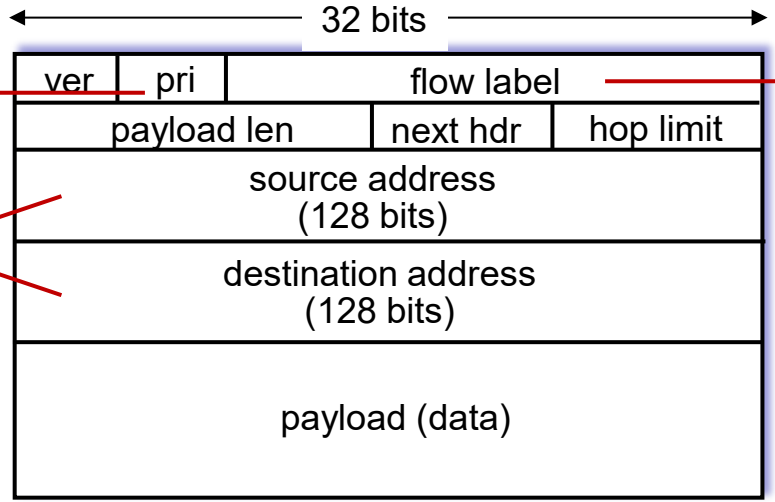


RIR	IPv4 per person	IPv4 per 1,000 people	Inputs used (IPv4; population proxy)
ARIN	**~4.33**	**~4,329**	1,651,270,144; Northern America 381,464,223
RIPE NCC	**~0.75**	**~748**	850,704,328; Europe+W. Asia+Central Asia 1,136,660,507
APNIC	**~0.20**	**~199**	886,332,608; (ESCAP – W. Asia – C. Asia + Oceania) 4,447,546,033
LACNIC	**~0.28**	**~276**	182,957,056; Latin America & Caribbean 662,186,388
AFRINIC	**~0.08**	**~76.6**	115,998,208; Africa 1,515,140,849

IPv6 datagram format

priority: identify priority among datagrams within a flow

128-bit IPv6 addresses



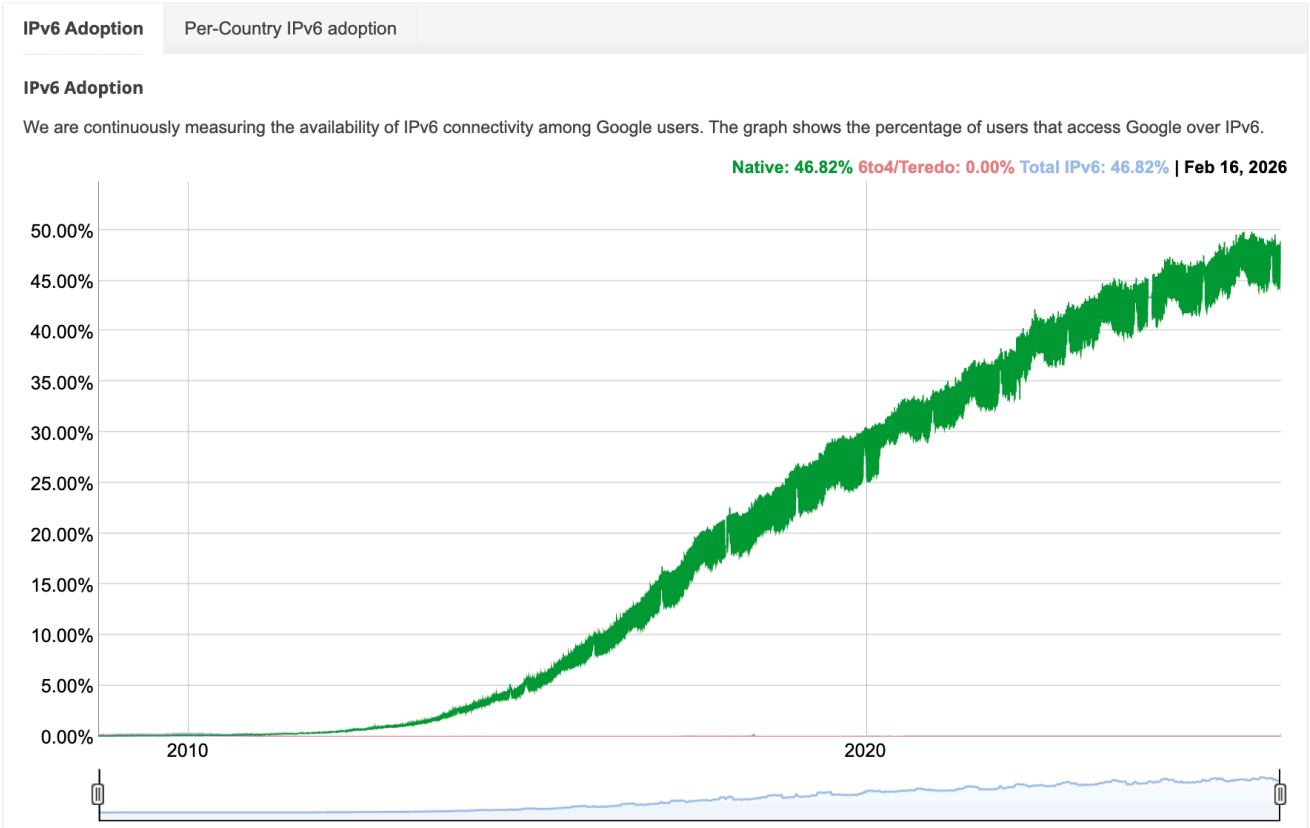
flow label: identify datagrams in same "flow." (concept of "flow" not well defined).

What's missing (compared with IPv4):

- no checksum (to speed processing at routers)
- no fragmentation/reassembly
- no options (available as upper-layer, next-header protocol at router)

IPv6: adoption

- Google[1]: ~ 45% of clients access services via IPv6
- NIST: 1/3 of all US government domains are IPv6 capable



1 https://www.google.com/intl/en/ipv6/statistics.html

IPv6: adoption

- Google[1]: ~ 45% of clients access services via IPv6
- NIST: 1/3 of all US government domains are IPv6 capable
- Long (long!) time for deployment, use
  - > 25 years and counting!
  - think of application-level changes in last 25 years: WWW, social media, streaming media, gaming, telepresence, …
  - *Why?*

[1] https://www.google.com/intl/en/ipv6/statistics.html

Readings for next week

Book pages

- 360-381 (4.3.1-4)

Optional:

- 390-393 (Sec. 4.5)

Key concepts

IPv4/IPv6 Addressing

From IPv4 to IPv6

Subnet & subnet mask

DHCP

NAT

Enough for today

See you Monday at 10:15

God helg!

(Room R22)