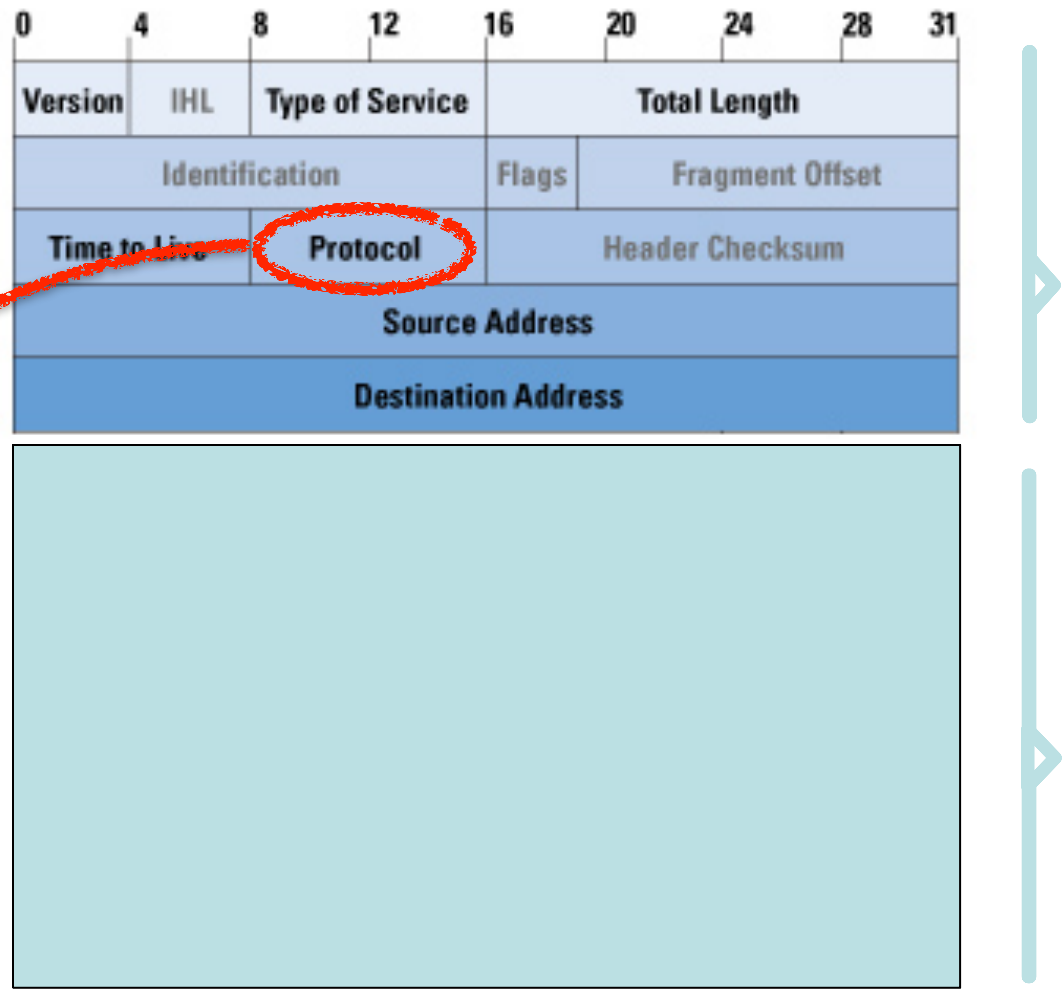
Exam booklet

2024 - 2025

Reserved address blocks (IPv4)

0.0.0.0	**absence** of address
127/8	**loopback** addresses (this host, e.g. 127.0.0.1)
10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16	**private** addresses (e.g. at home): used by *anyone*, but *not* in the public Internet (internet routers drop packets destined to them)
100.64/10	private addresses used only by Internet Service Providers (ISPs)—**Carrier Grade NAT addresses**
192.88.99/24	**IPv6-to-IPv4** relay routers
169.254.0.0/16	**link local** addresses (can be used only by systems in same LAN)
224/4	**multicast**
240/4	reserved “for experimental/future use” until recently
255.255.255.255/32	link local (LAN) **broadcast**

IPv4 Packet Format



Higher layer protocol [1 = ICMP*, 6 = TCP, 17 = UDP]

Header 20 bytes (+ options, if any)

protocol overhead

payload

useful bits (higher-layer data)

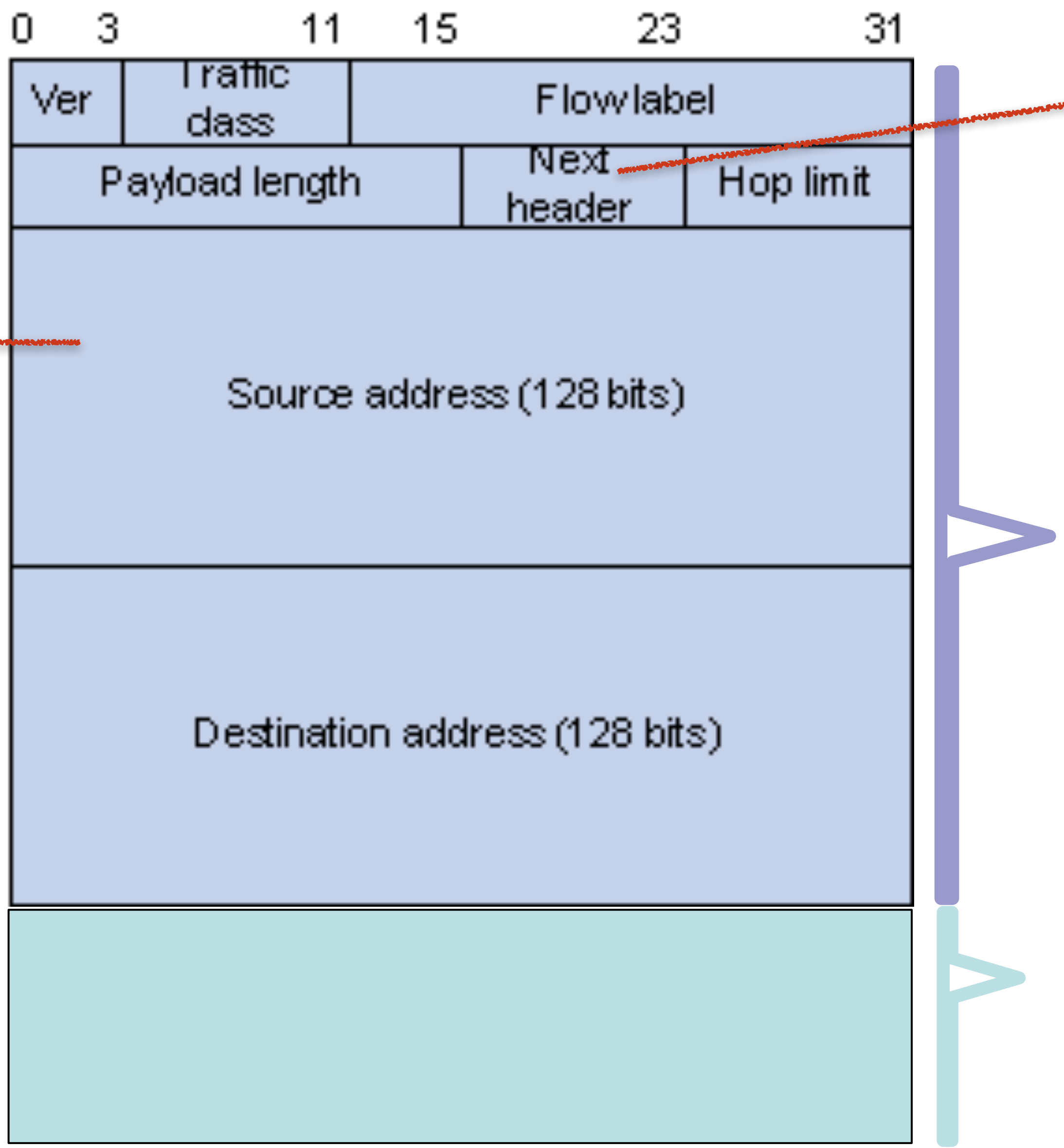
✳ (ICMP is used to carry error messages at the network layer)

Reserved address blocks (IPv6)

::/128	**absence** of address
::1/128	**loopback** address (this host)
fc00::/7 (i.e. fcxx: and fdxx:) for example: fd24:ec43:12ca:1a6:a00:20ff:fe78:30f9	unique local addresses = **private** networks (e.g. in EPFL): *not* to be used in the public Internet
fe80::/10	**link local** addresses (used only by systems in same LAN)
ff00::/8	**multicast**
ff02::1:ff00:0/104	**solicited node multicast** (see NDP later)
ff02::1/128	**link local broadcast**
ff02::2/128	**multicast to all link-local routers** (in same LAN)

IPv6 Packet Format

Address field
16 bytes



Higher layer protocol
[58 = IPv6-ICMP,
6 = TCP,
17 = UDP]

Header
40 bytes
(+ options,
if any)

Payload

***We will see the functions of the fields other than the addresses in a following lecture*

Multicast MAC addresses

- Is there Multicast ARP?
  No, multicast MAC address is *algorithmically* derived from multicast IP address:
  - Last 23 bits of IPv4 multicast address are used in MAC address
  - Last 32 bits of IPv6 multicast address are used in MAC address

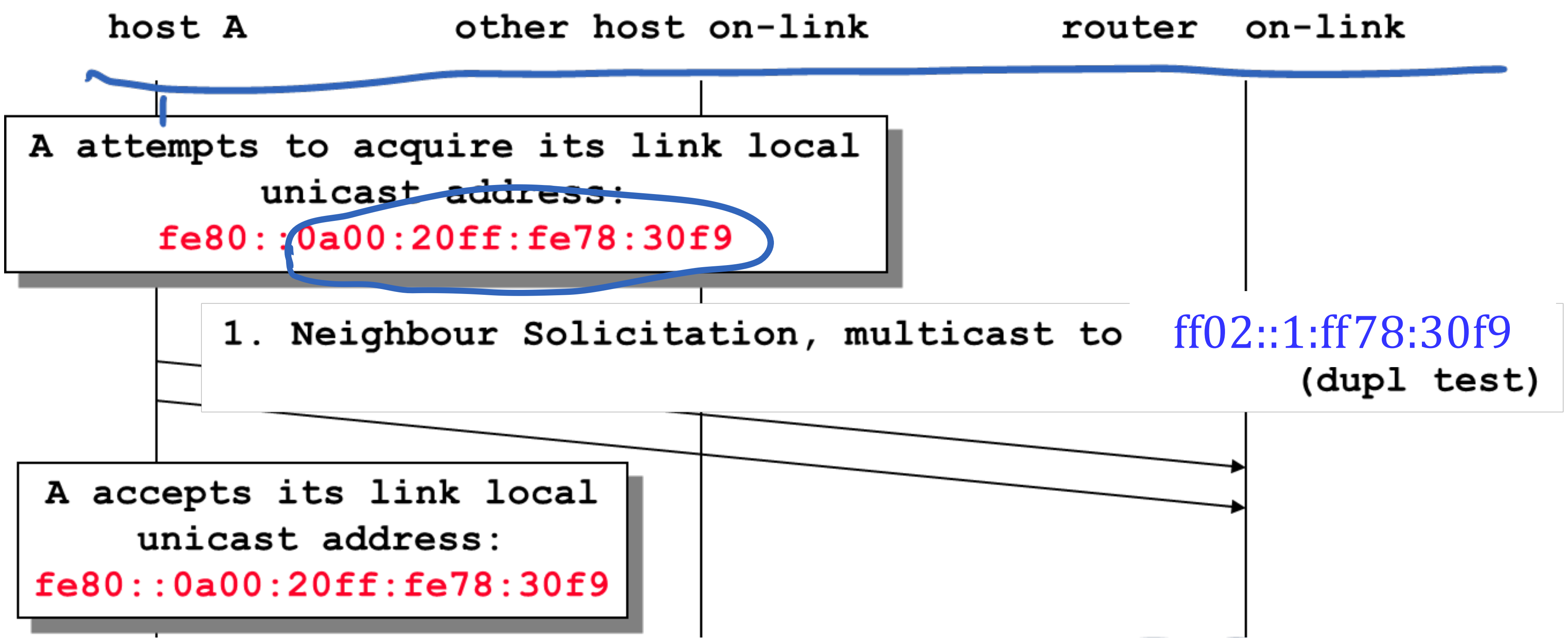
1st bit of hextet is 0

MAC multicast addr.	*Used for*
01-00-5e-YX-XX-XX	IPv4 multicast
33-33-XX-XX-XX-XX	IPv6 multicast

- Note:
  - Multicast MAC depends only on multicast IP address `m`, not on source address `s`, even if `m` is an SSM address
  - Several multicast IP addresses may yield the same MAC
    - packets received unnecessarily at the MAC layer are removed by the OS; hopefully this happens rarely

IP dest address	229.130.54.207
IP dest address (hexa)	e5-82-36-cf
IP dest address (bin)	…-10000010-…
Keep last 23 bits (bin)	…-00000010-…
Keep last 23 bits (hexa)	02-36-cf
MAC address	01-00-5e-02-36-cf

Duplicate address test

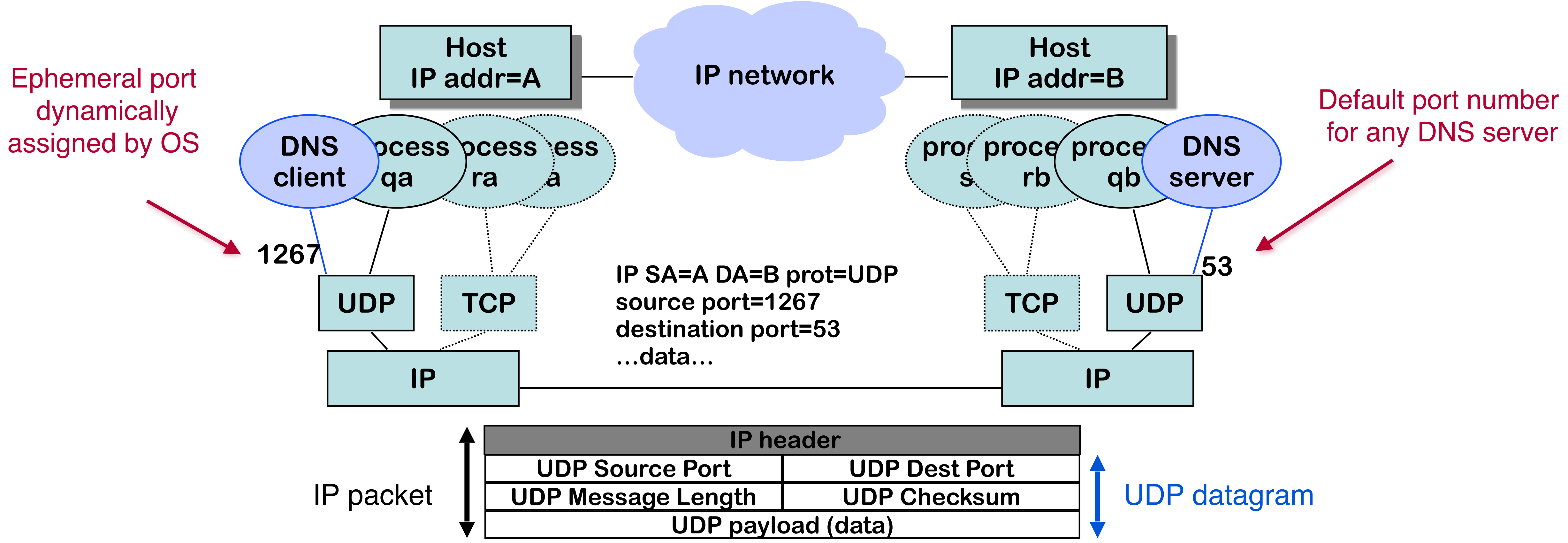


A sends a Neighbour Solication (NS) message to check for address duplication, sent to the Solicited Node Multicast Address.

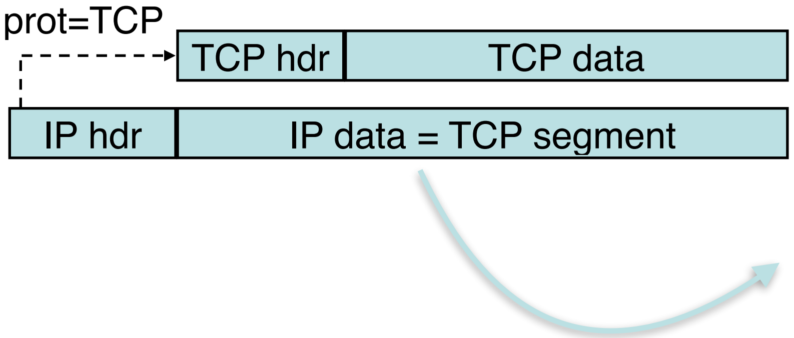
Any host that would have to same link local address listens to this multicast address

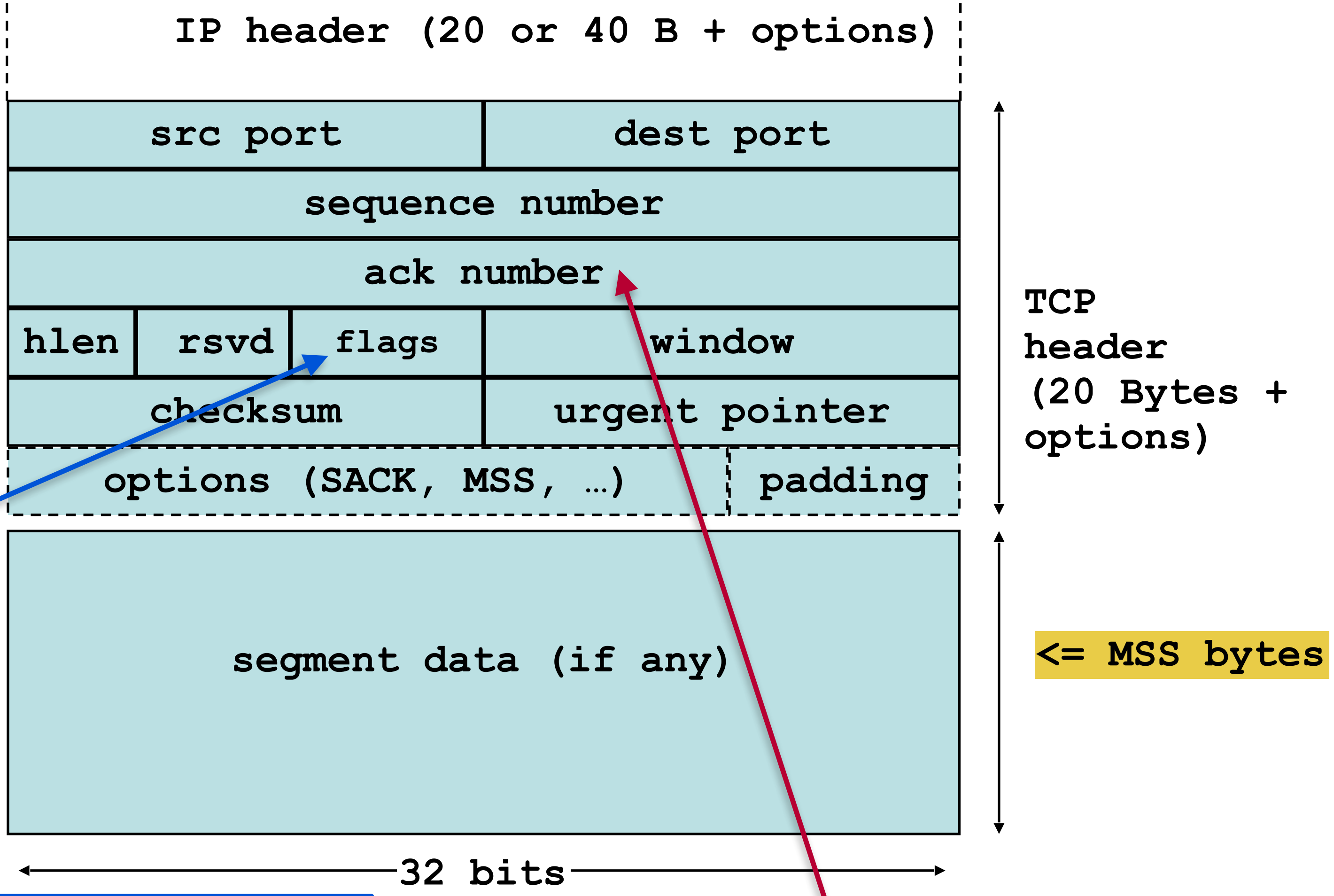
Port Numbers

- assigned by OS to identify processes within a host
- servers' port numbers must be *well-known* to clients (e.g. 53 for DNS, 80 for HTTP, 443 for HTTPs)
- src and dest port numbers are *inside transport-layer header*



TCP Segment Format





flags	meaning
NS	used for explicit congestion notification
CWR	used for explicit congestion notification
ECN	used for explicit congestion notification
urg	urgent ptr is valid
ack	ack field is valid
psh	this seg requests a push (creating a segment immediately)
rst	reset the connection
syn	connection setup
fin	sender has reached end of byte stream

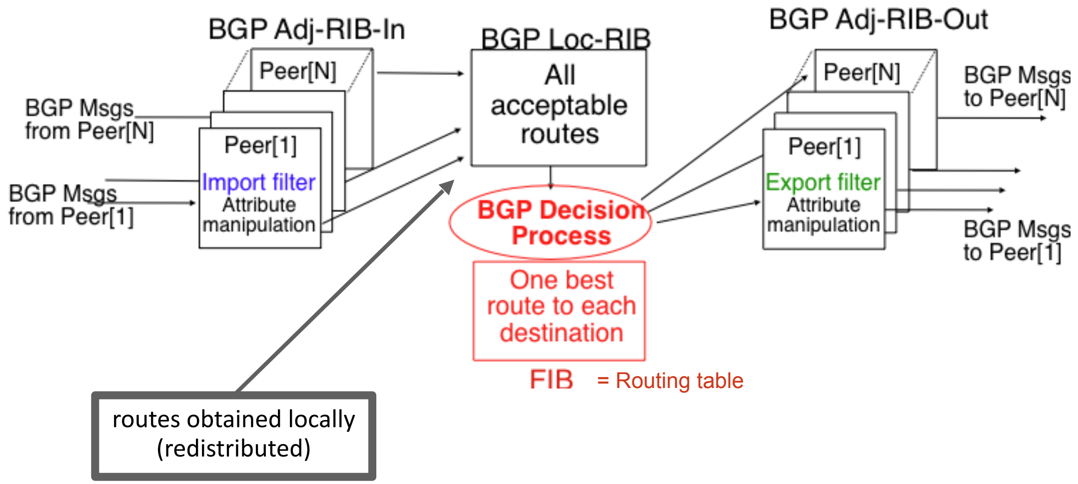
Indicates the next expected seq num from the other host

The Decision Process

The decision process chooses *at most one route* to each different destination *prefix* as best

e.g.: only one route to 2.2/16 can be chosen,

but there can be different routes to 2.2.2/24 and 2.2/16



How?

- A route can be selected only if its next-hop is reachable
- For each dest prefix, all acceptable routes are compared w.r.t. their attributes using a sequence of criteria (until only one route remains); a common sequence is:

0. Highest weight (Cisco proprietary)
1. Highest LOCAL-PREF
2. Shortest AS-PATH
3. Lowest MED, if taken seriously by this network
4. e-BGP > i-BGP (= if route is learnt from e-BGP, it has priority)
5. Shortest path to NEXT-HOP, according to IGP
6. Lowest BGP identifier (router-id of the BGP peer from whom route is received)

(The Cisco and FRR implementation of BGP, used in lab 6, have additional cases, not shown here)

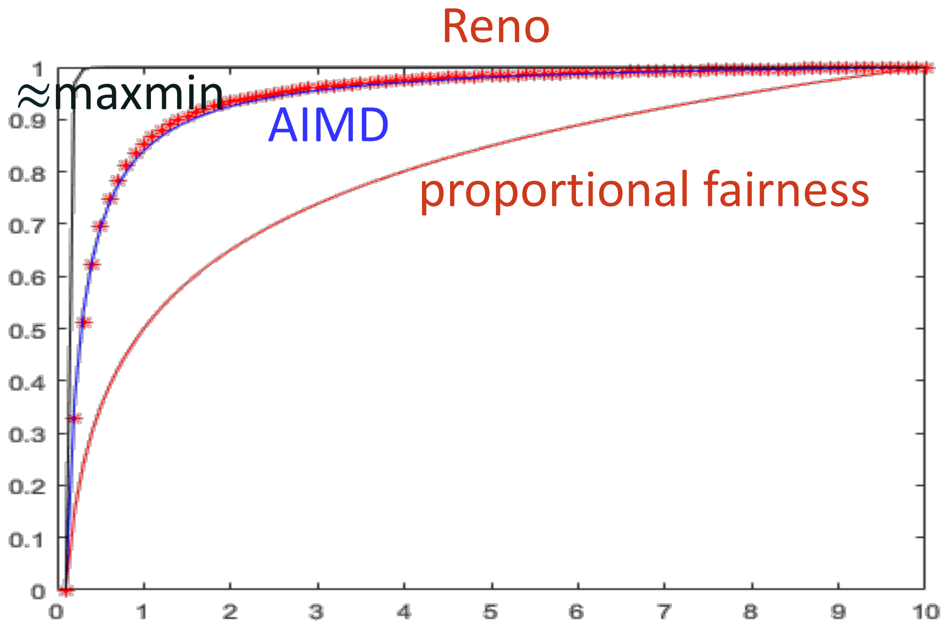
The result of the decision process is stored in forwarding table and in Adj-RIB-out (*one route per destination for each BGP peer*).
The router sends updates when Adj-RIB-out *changes* (addition or deletion) after applying *export rules*.

Fairness of TCP Reno

For *long lived flows*, the rates obtained with TCP Reno are as if they were distributed according to utility fairness, with utility of flow i given by $U_i(x_i) = \frac{\sqrt{2}}{\tau_i} \arctan \frac{x_i \tau_i}{\sqrt{2}}$

with x_i = rate (in MSSs) = W/τ_i, τ_i = RTT (see "Rate adaptation, Congestion Control and Fairness: A Tutorial")

For *flows that have same RTT*, the fairness of TCP is *between max-min and proportional fairness*, closer to proportional fairness:



rescaled utility functions;

RTT = 100 ms

maxmin approx. is $U(x) = 1 - x^{-5}$

TCP Reno

Loss - Throughput Formula

Consider a *large* TCP flow size (many bytes to transmit).

Assume we observe that, in average, a fraction ***q*** of packets is lost (or marked with ECN).

The throughput should be close to $\theta = \frac{MSS\ 1.22}{RTT\sqrt{q}}$.

Formula **assumes**:

transmission time is negligible compared to RTT,

losses are rare and occur periodically,

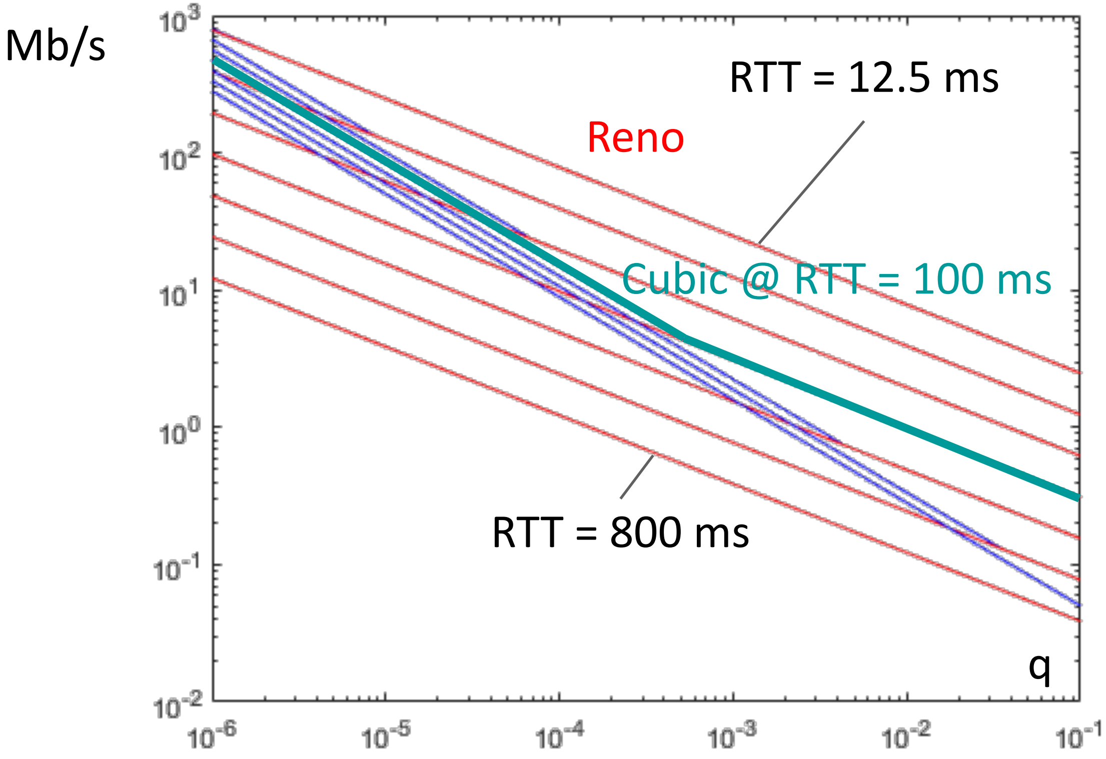
time spent in Slow Start and Fast Recovery is negligible.

[see “Rate adaptation, Congestion Control and Fairness: A Tutorial”]

Cubic's Loss throughput formula

Given the ***same assumptions*** as for TCP Reno:

$$\theta \approx \max\left(\frac{1.054}{RTT^{0.25}q^{0.75}}, \frac{1.22}{RTT\sqrt{q}}\right)$$ in MSS per second.



So:

- Cubic's formula is same as Reno for small RTTs and small BW-delay products
- but a TCP Cubic connection gets ***more throughput*** than TCP Reno when bit-rate and RTT are large

Other details: computation of W_{max} uses a more complex mechanism called "fast convergence" - see Latest IETF Cubic RFC / Internet Draft or http://elixir.free-electrons.com/linux/latest/source/net/ipv4/tcp_cubic.c

TCP Reno — recap

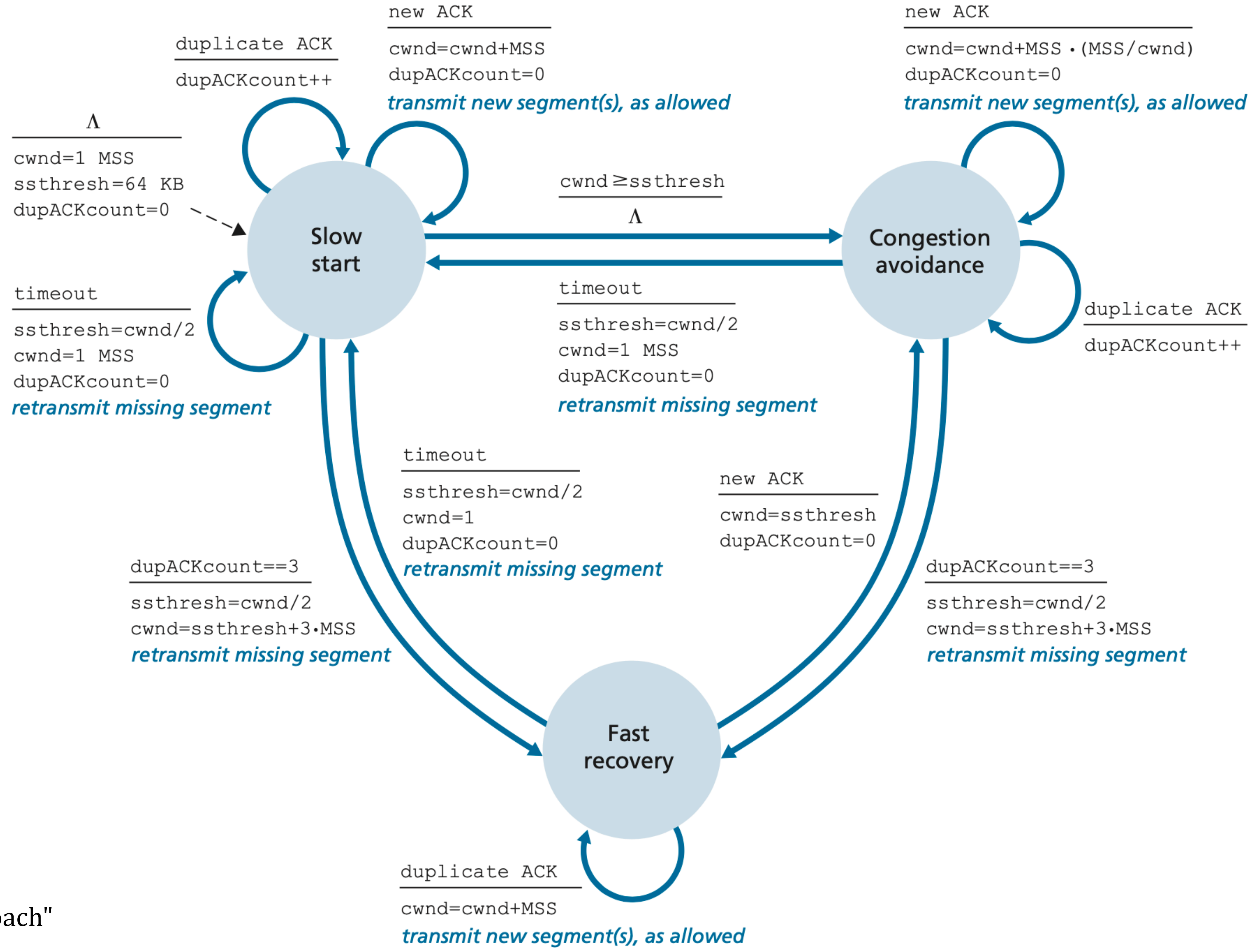


Figure from our textbook:
"Computer Networking: A top-down approach"
by J. Kurose and K. Ross