Exam Booklet

TCP/IP Networking 2020-2021

Special Addresses

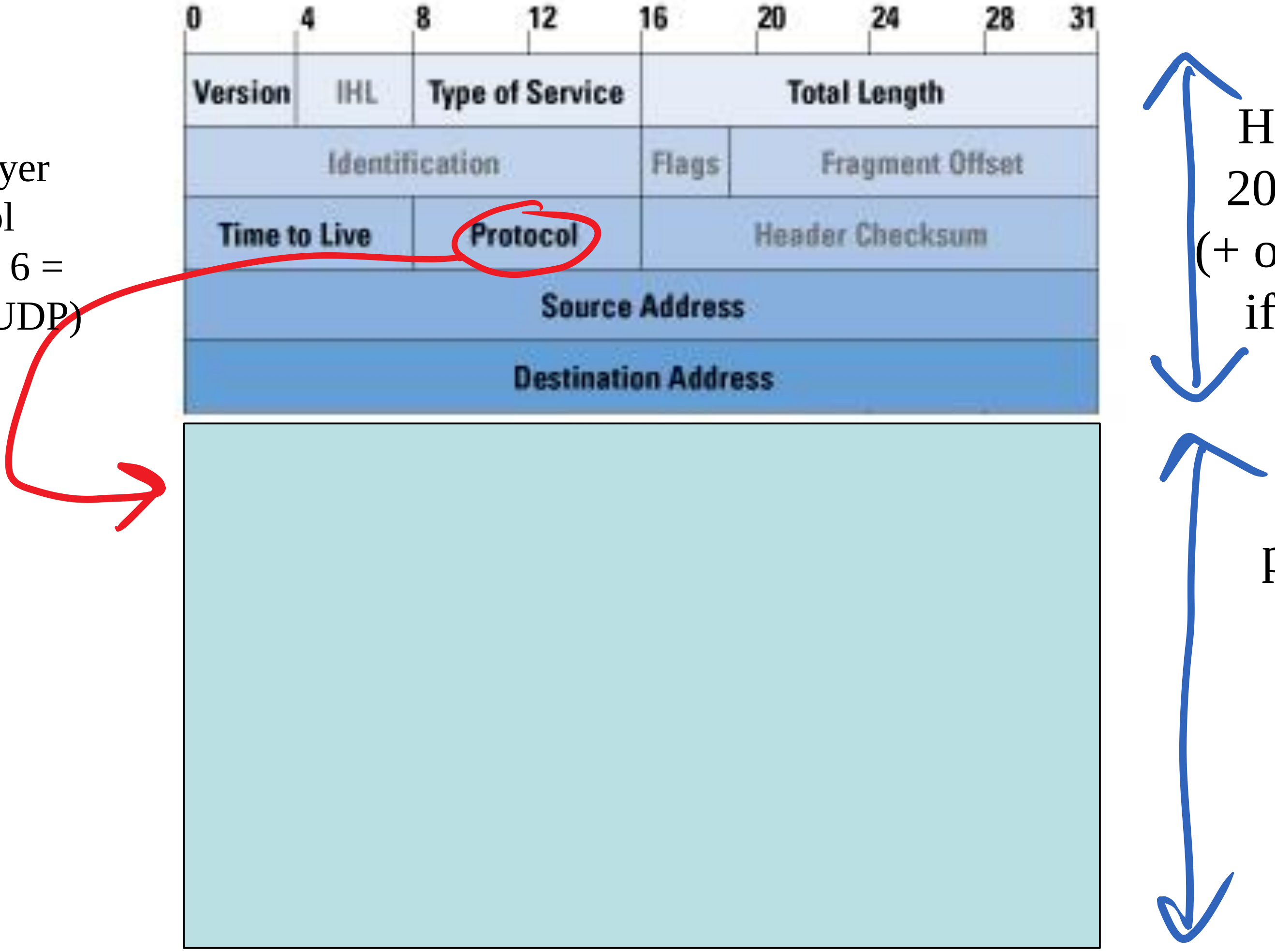
0.0.0.0	absence of address
127.0.0/24 for example 127.0.0.1	this host (loopback address)
10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16	private networks (e.g in IEW) cannot be used on the public Internet
169.254.0.0/16	link local address (can be used only between systems on same LAN)
224/4	multicast
240/5	reserved
255.255.255.255/32	link local broadcast

Examples of Special Addresses

::/128	absence of address
::1/128	this host (loopback address)
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 IEW) cannot be used on the public Internet
fe80::/10	link local address (can be used only between systems on same LAN)
ff00::/8	multicast
ff02::1:ff00:0/104	Solicited node multicast
ff02::1/128 ff02::2/128	link local broadcast all link local routers

EPFL Private

IPv4 Packet Format



Higher layer protocol 1= ICMP, 6 = TCP, 17 = UDP)

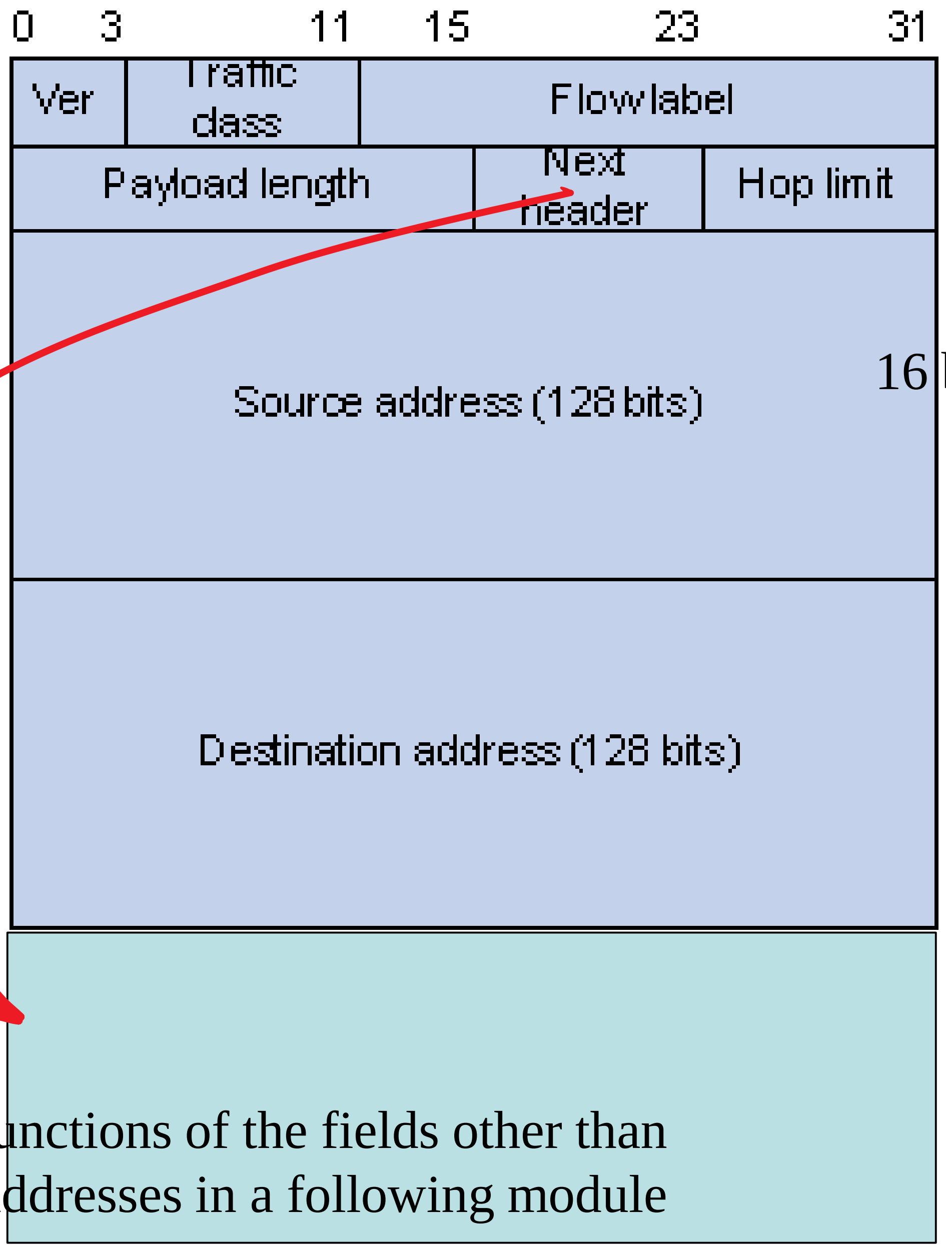
Header 20 bytes (+ options, if any)

payload

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

IPv6 Packet Format

e.g.
Higher layer
protocol
1= ICMP, 6 =
TCP, 17 = UDP)



16 bytes

Header
40 bytes
(+ options,
if any)

payload

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

Ethernet Frame format

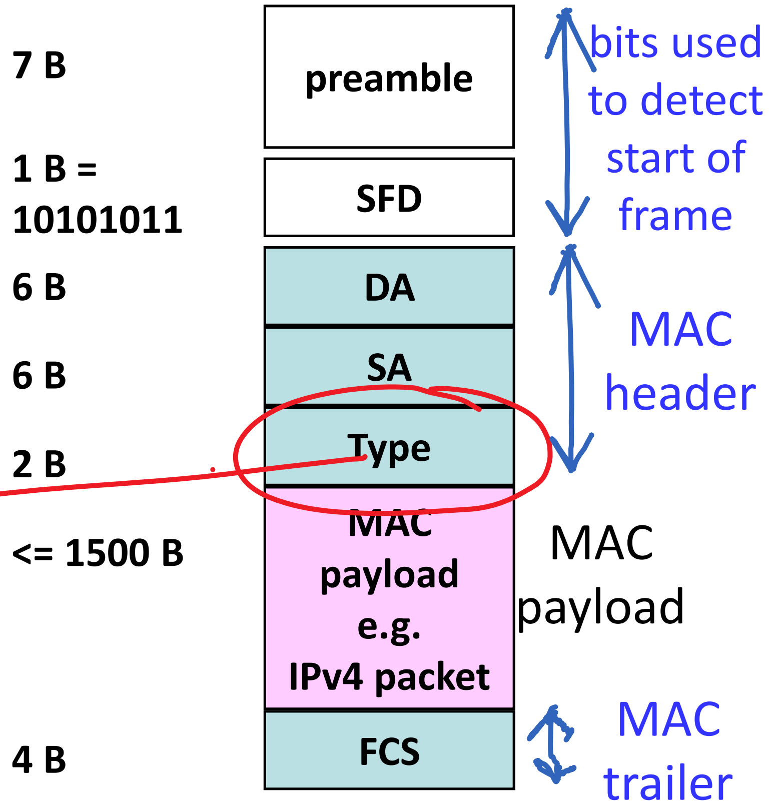
Ethernet frame = Ethernet PDU

An Ethernet frame typically transports an IP packet, sometimes also other

Type of protocol contained in the Ethernet packet (hexa):

0800: IPv4
0806: ARP (used by IPv4)
86DD: IPv6
8847: MPLS unicast
88F7: Precision Time Protocol

Ethernet V.2 frame



7 B	preamble
1 B = 10101011	SFD
6 B	DA
6 B	SA
2 B	Type
<= 1500 B	MAC payload e.g. IPv4 packet
4 B	FCS

bits used to detect start of frame

MAC header

MAC payload

MAC trailer

DA = destination address
SA = source address

Multicast MAC Addresses

IP multicast address is algorithmically mapped to a multicast MAC address.

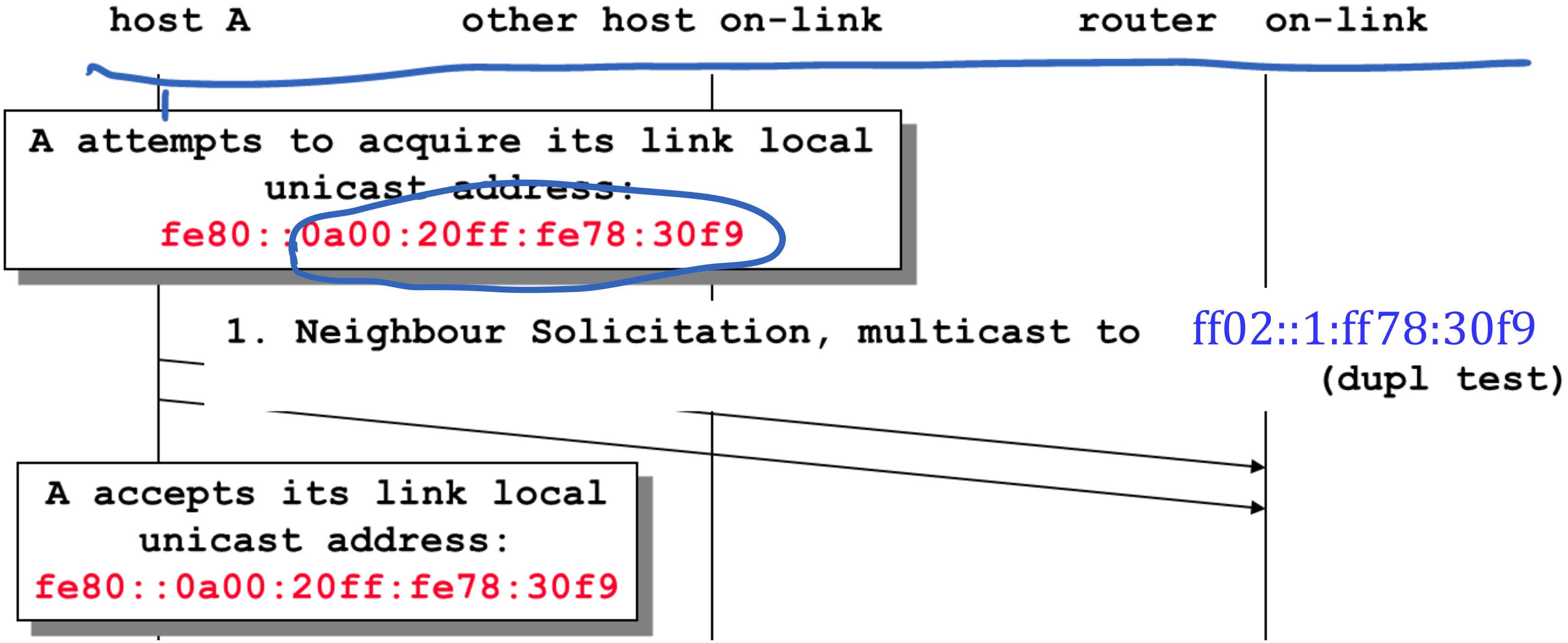
MAC multicast addr.	*Used for*
01-00-5e-XX-XX-XX	IPv4 multicast
33-33-XX-XX-XX-XX	IPv6 multicast

Last 23 bits of IPv4 multicast address are used in MAC address

Last 32 bits of IPv6 multicast address are used in MAC address

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-03-36-cf

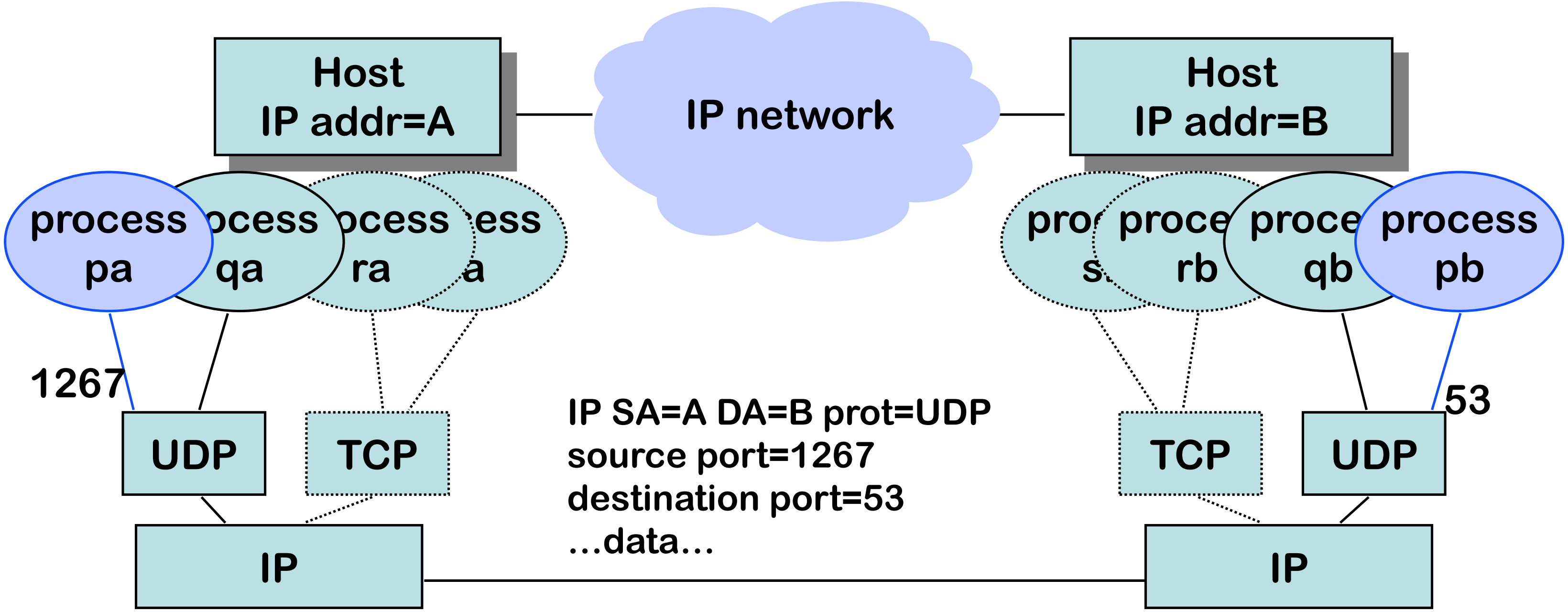
SLAAC Step 2: Duplicate Test


host A
other host on-link
router on-link
A attempts to acquire its link local unicast address:
fe80::0a00:20ff:fe78:30f9
1. Neighbour Solicitation, multicast to
ff02::1:ff78:30f9
(dupl test)
A accepts its link local unicast address:
fe80::0a00:20ff:fe78:30f9

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

UDP Uses Port Numbers

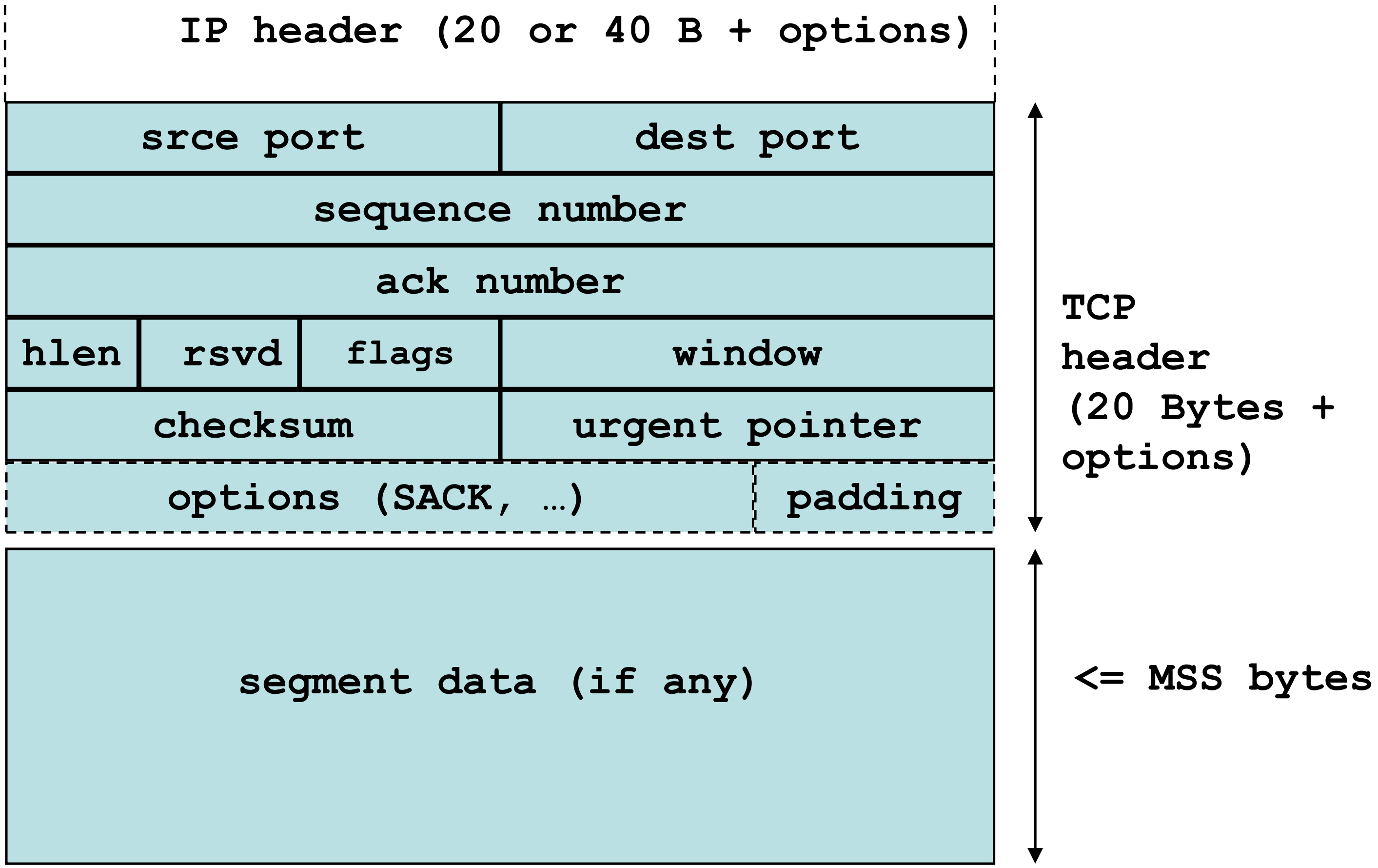


IP SA=A DA=B prot=UDP
source port=1267
destination port=53
…data…

IP datagram

IP header	
UDP Source Port	UDP Dest Port
UDP Message Length	UDP Checksum
data	

UDP datagram



IP header (20 or 40 B + options)

srce port	dest port
sequence number	
ack number	
hlen, rsvd, flags	window
checksum	urgent pointer
options (SACK, …)	padding

segment data (if any)

TCP header (20 Bytes + options)

<= MSS bytes

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
rst	reset the connection
syn	connection setup
fin	sender has reached end of byte stream

Dijkstra's Shortest Path Algorithm

```
$m(0) = 0;\ m(i) = \infty\ \forall\, i \neq 0; V = \emptyset\ ; pred(i) = \emptyset\ \forall i;$
for $k = 0: N$ do
    find $i \notin V$ that minimizes $m(i)$
    if $m(i)$ is finite
        add $i$ to $V$
        for all neighbours $j \notin V$ of $i$
            if $m(i) + c(i,j) < m(j)$
                $m(j) = m(i) + c(i,j)$
                $pred(j) = \{i\}$
            else if $m(i) + c(i,j) = m(j)$
                $m(j) = m(i) + c(i,j)$
                $pred(j) = pred(j) \cup \{i\}$
```

The nodes are 0…N ; the algorithm computes shortest paths from node 0.

$c(i,j)$: cost of link (i,j).

V: set of nodes visited so far.

$pred(i)$: estimated set of predecessors of node i along a shortest path (multiple shortest paths are possible).

$m(j)$: estimated distance from node 0 to node j.

At completion, $m(i)$ is the true distance from 0 to i.

Practical Aspects

OSPF packet are sent directly over IP (OSPF=protocol 89 (0x59)).

Reliable transmission is managed by OSPF with OSPF Acks and timers.

OSPFv2 supports IPv4 only

OSPFv3 supports IPv6 and dual-stack networks

OSPF routers are identified by a 32 bit number

OSPF areas are identified by a 32 bit number

The *Centralized* Bellman-Ford Algorithm

```
Algorithm BF-C
input: a directed graph with links costs $A(i,j)$; assume $A(i, j)$>0
and $A(i,j) = \infty$ when nodes i and j are not connected.
output: vector $p$ s.t. $p(i)$= cost of best path from node $i$ to node 1

$p^0(1) = 0, \;\; p^0(i) = \infty$ for $i \neq 1$

for $k = 1,2,\ldots$ do
        $p^k(i) = \min_{j \neq i}[A(i,j) + p^{k-1}(j)]$ for $i \neq 1$
        $p^k(1) = 0$
until $p^k = p^{k-1}$
return($p^k$)
```

Distributed Bellman-Ford

Requires only to remember distance from self to destination + the best neighbor (nextHop(i))

and works for all initial conditions

Distributed Bellman-Ford Algorithm, BF-D

node i maintains an estimate $q(i)$ of the distance $p(i)$ to node 1;
node i remembers the best neighbor nextHop(i)

initial conditions are arbitrary but $q(1) = 0$ at all steps;

from time to time, i sends its value $q(i)$ to all neighbors

when i receives an updated value $q(j)$ from j, node i recomputes $q(i)$:

```
eq (2)    if $j == \text{nextHop}(i)$
          then $q(i) \leftarrow A(i,j) + q(j)$
          else $q(i) \leftarrow \min\big(A(i,j) + q(j),\ q(i)\big)$
```

if eq(2) causes $q(i)$ to be modified, $\text{nextHop}(i) \leftarrow j$

The Decision Process

The decision process decides which route is selected;

At most one best route to exactly the same prefix is chosen

- 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

A route can be selected only if its next-hop is reachable

Routes are compared against each other 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
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, have a few additional cases, not shown here)

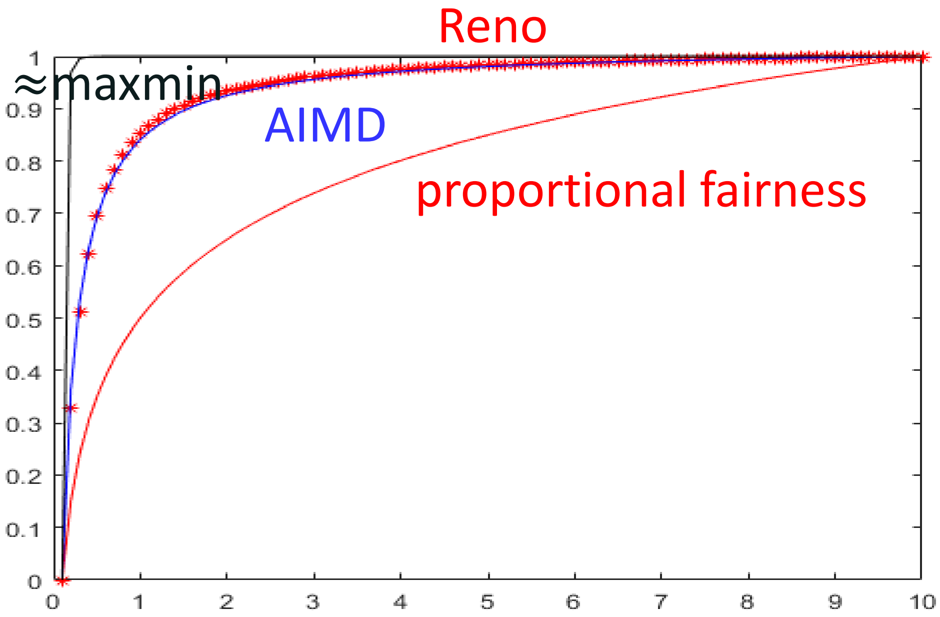
Fairness of TCP Reno

For long lived flows, the rates obtained with TCP are as if they were distributed according to utility fairness, with utility of flow i given by

$$U(x_i) = \frac{\sqrt{2}}{\tau_i} \arctan \frac{x_i \tau_i}{\sqrt{2}}$$

with x_i = rate = W/τ_i, τ_i = RTT

For sources that have same RTT, the fairness of TCP is between maxmin fairness 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 connection (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 negligible compared to RTT, losses are rare, time spent in Slow Start and Fast Recovery negligible, losses occur periodically

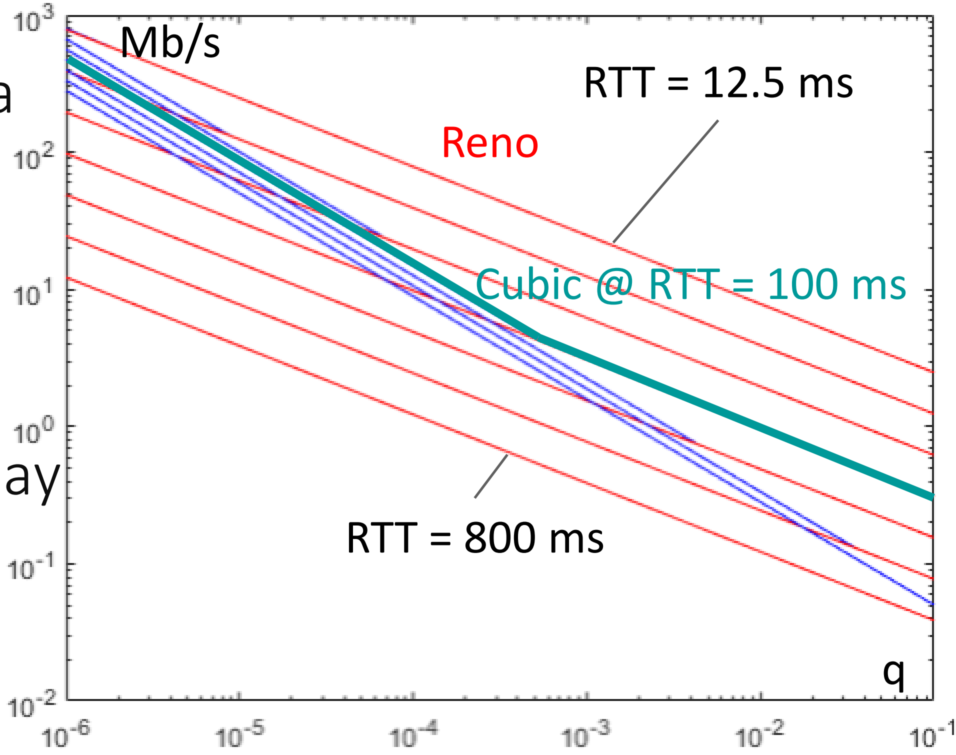
Cubic's Other Bells and Whistles

Cubic's Loss throughput formula

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

in MSS per second.

Cubic's formula is same as Reno for small RTTs and small BW-delay products.



Other Cubic details

W_{max} computation 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