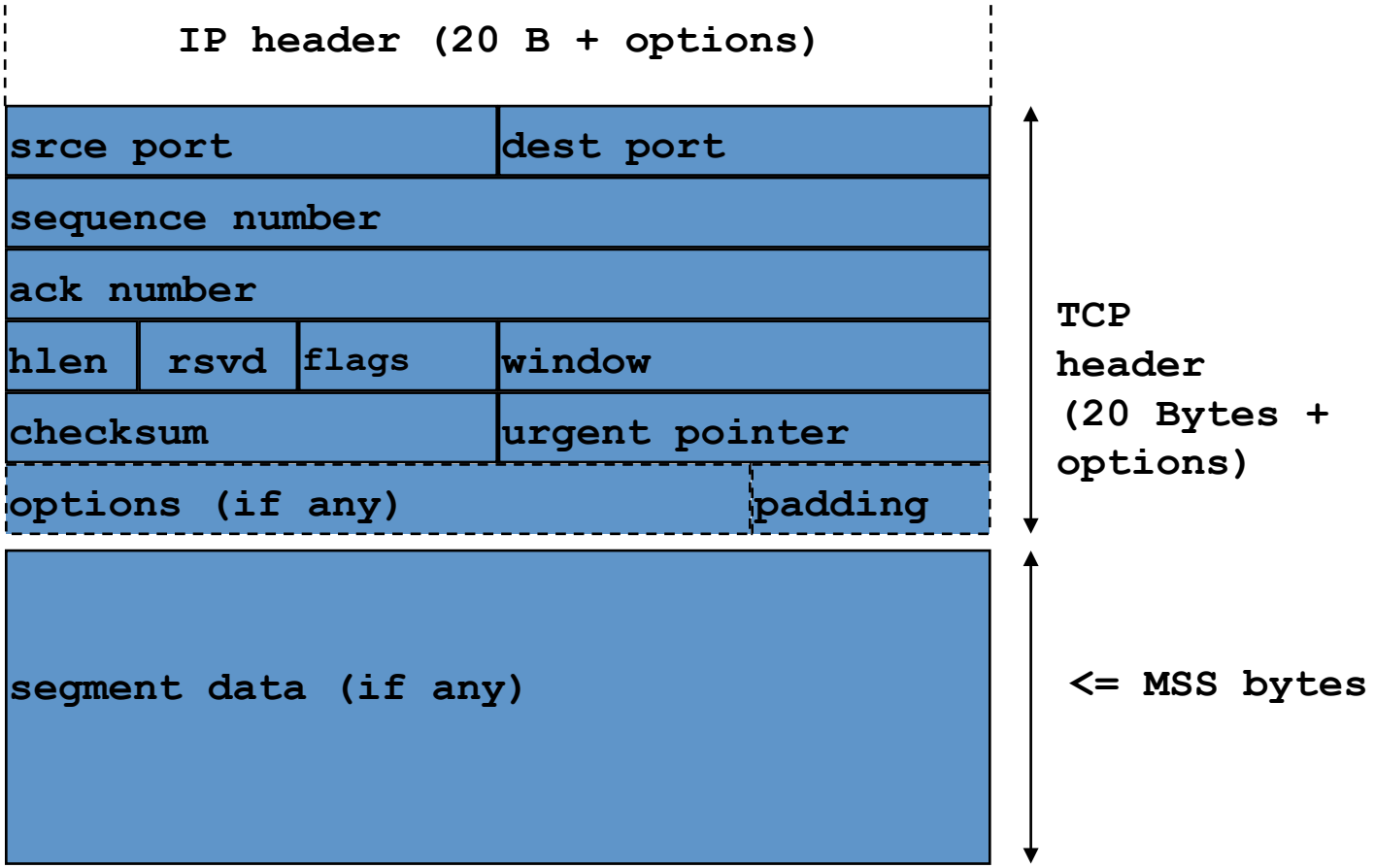
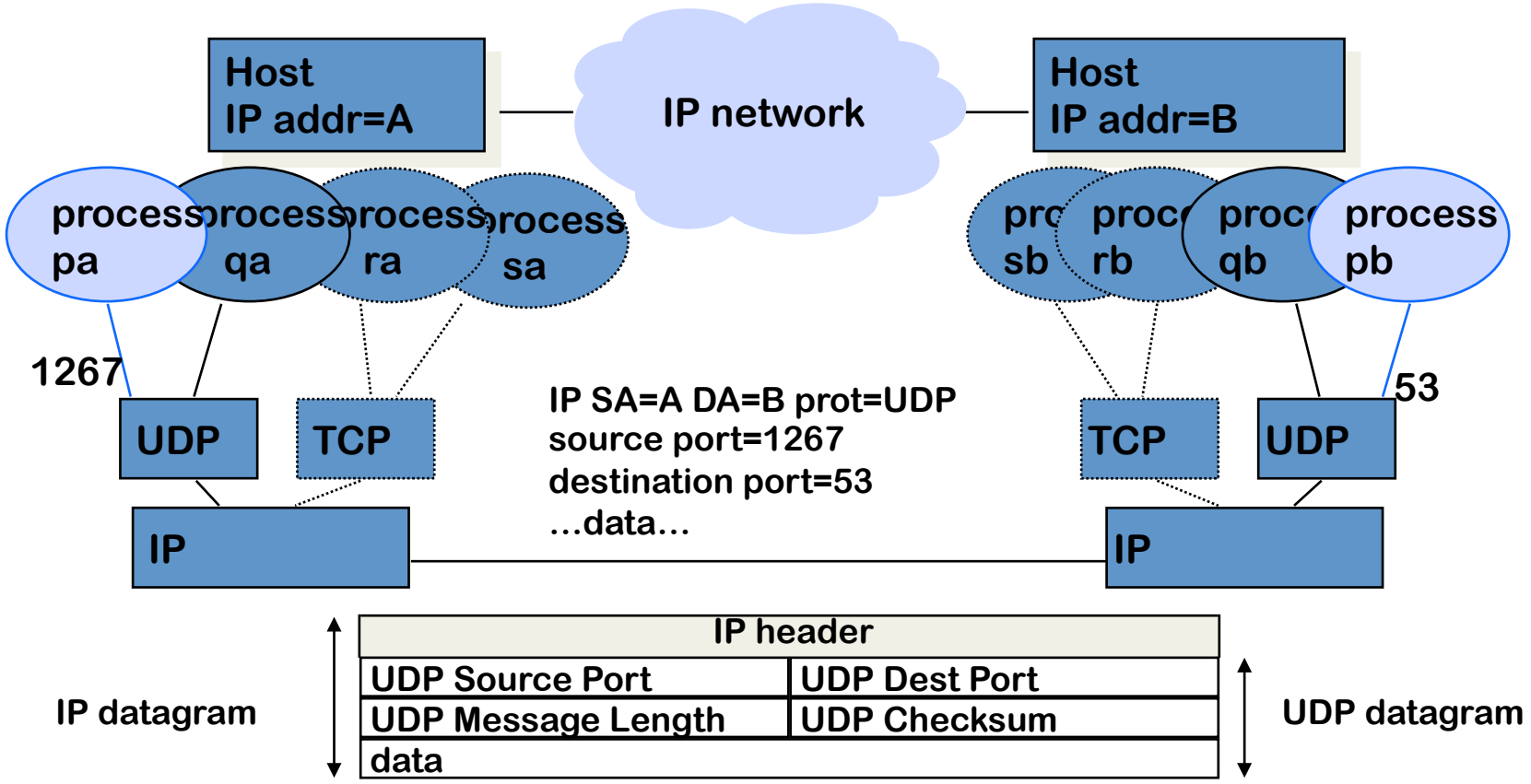
TCP Header



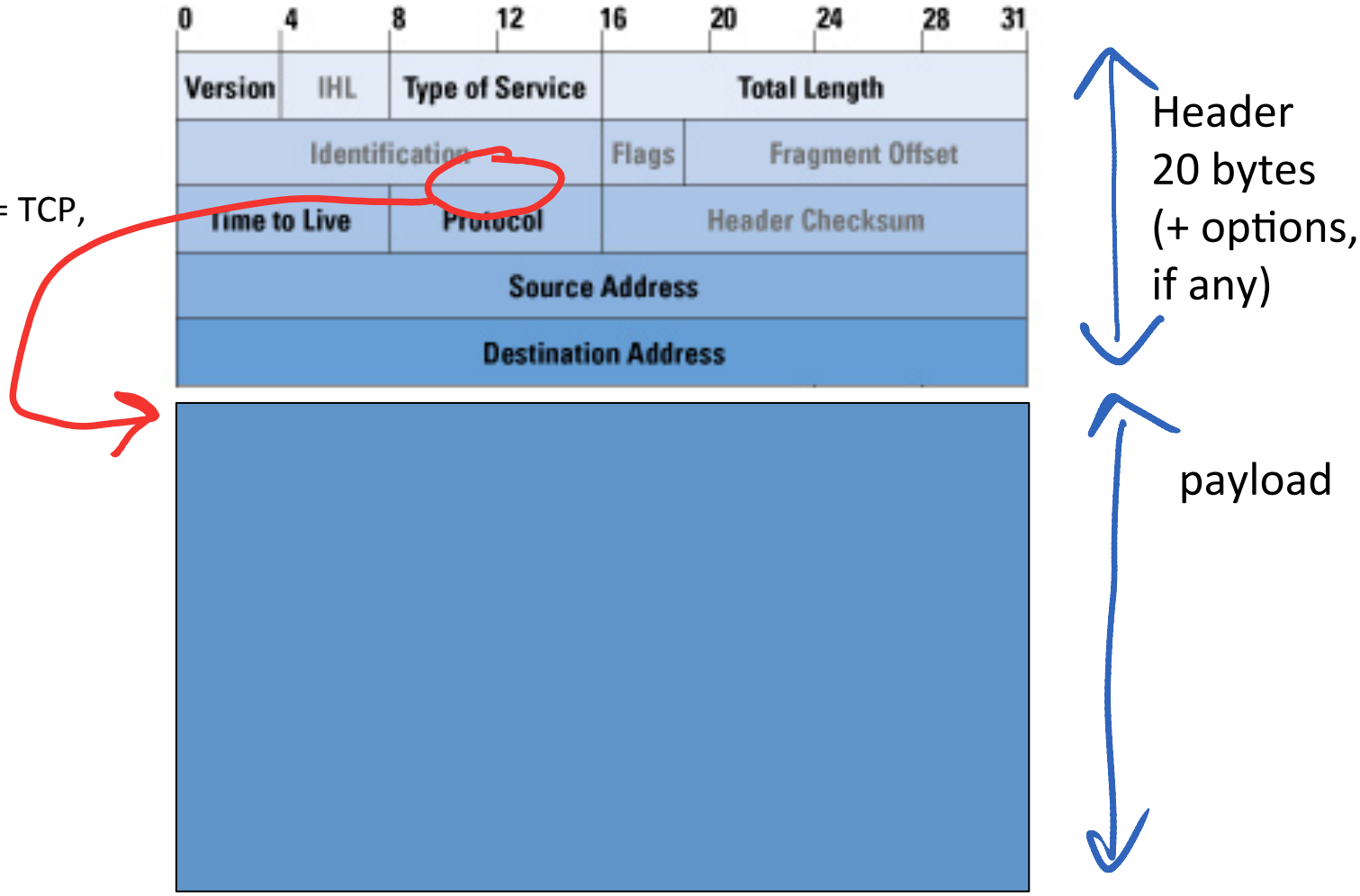
flags	meaning
NS	used for explicit congestion notification
CWR	used for explicit congestion notification
ECN	used for explicit congestion notification
urg	urgent ptr is valid
ack	ack field is valid
psh	this seg requests a push
rst	reset the connection
syn	connection setup
fin	sender has reached end of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

UDP Uses Port Numbers

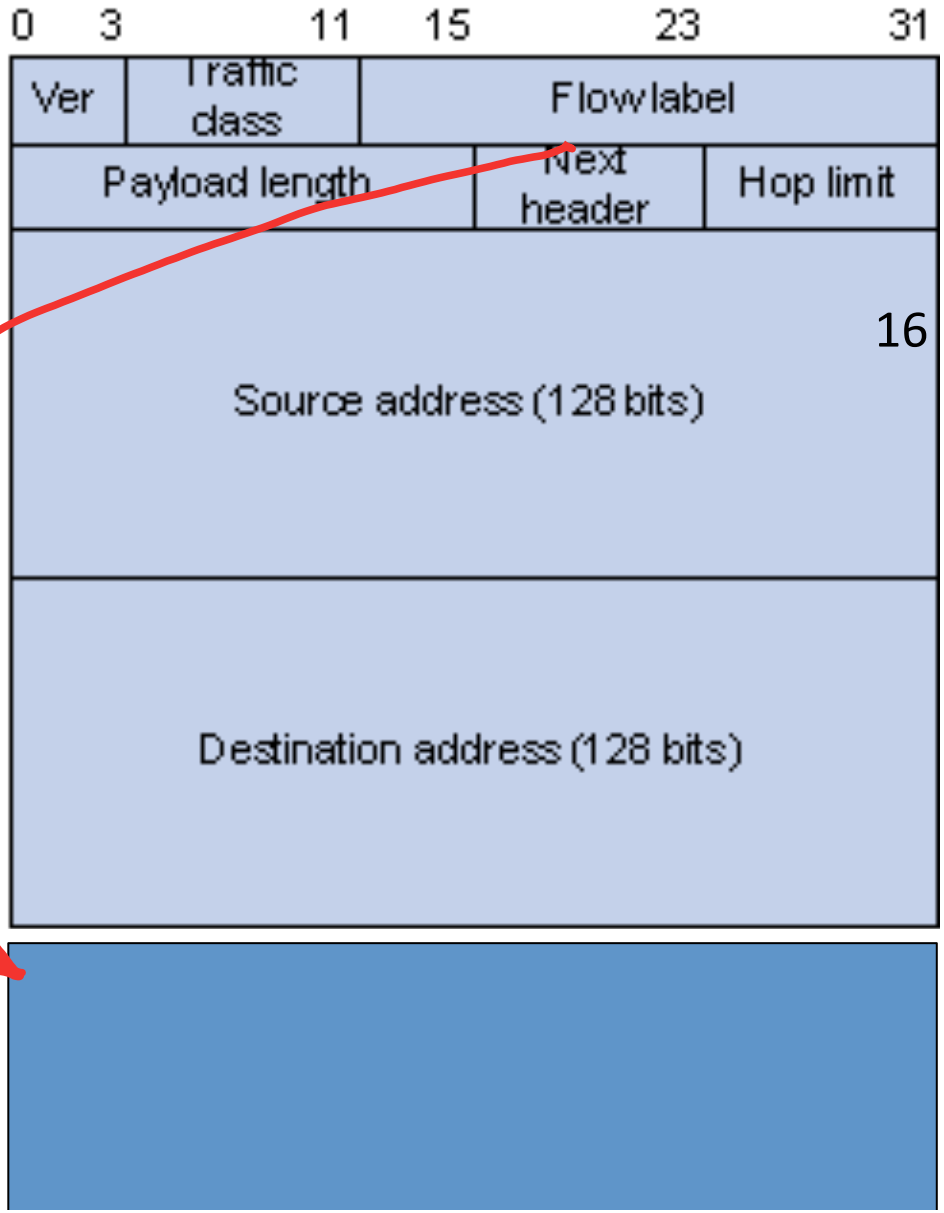


IPv4 Packet Format

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



IPv6 Packet Format



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

16 bytes

Header
40 bytes
(+ options, if any)

payload

The Solicited Node Multicast Address

- Add last 24 bits of target IP address to ff02::1:ff00:0/104
- A packet with such a destination address is forwarded by layer 2 to all nodes that listen to this multicast address using MAC multicast address 33:33:<last 32 bits of IP address>

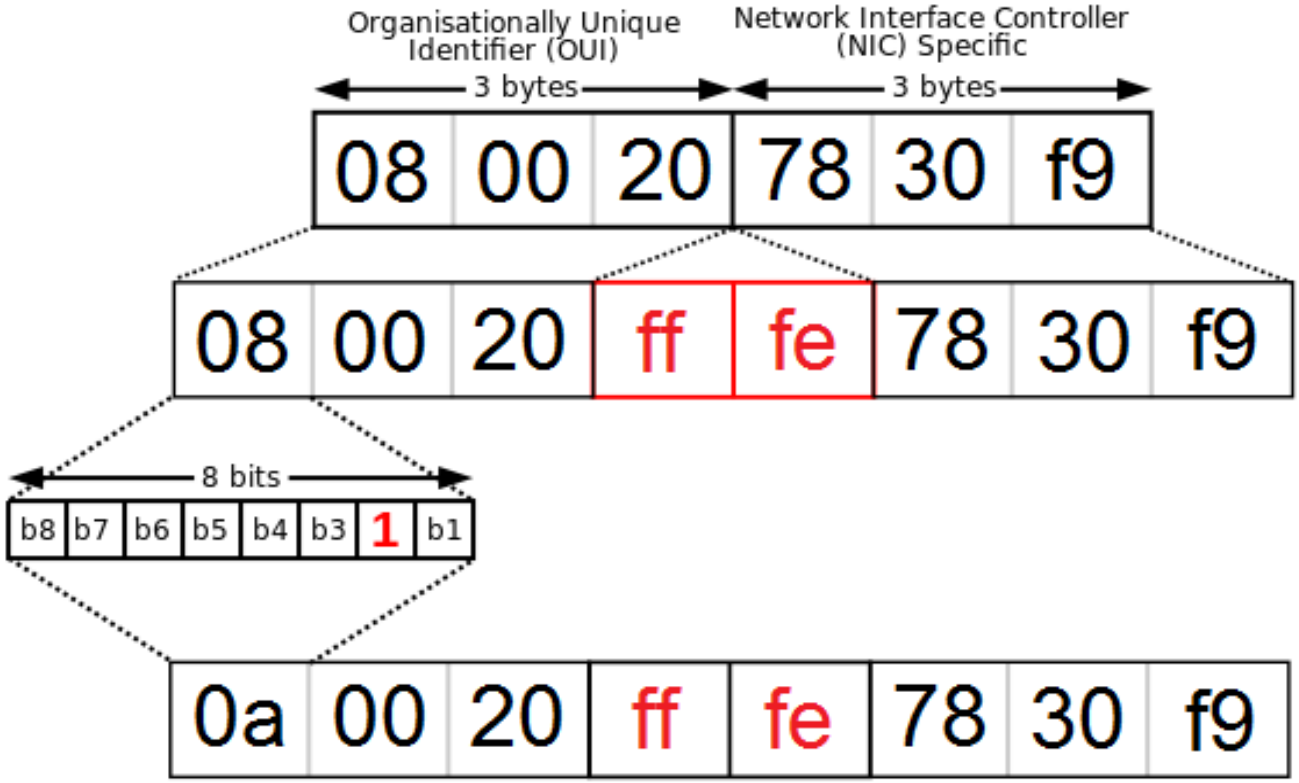
Target address	Compressed	2001:620:618:1a6:001:80b2:f66:1
	Uncompressed	2001:0620:0618:01a6:0001:80b2:0f66:0001
Solicited Node multicast address	Uncompressed	ff02:0000:0000:0000:0000:0001:ff66:0001
	Compressed	ff02::1:ff66:1

Compression Rules for IPv6 Addresses

- 1 *piece* = 16 bits = [0-4]hexa digits; leading zeroes in one piece are omitted ;
- prefer lower case
- pieces separated by “:”
- one IPv6 address uncompressed = 8 pieces
- **::** replaces any number of 0s in more than one piece; appears only once in address

uncompressed	*compressed*
2002:0000:0000:0000:0000:ffff:80b2:0c26	2002::ffff:80b2:c26
2001:0620:0618:01a6:0000:20ff:fe78:30f9	2001:620:618:1a6:0:20ff:fe78:30f9

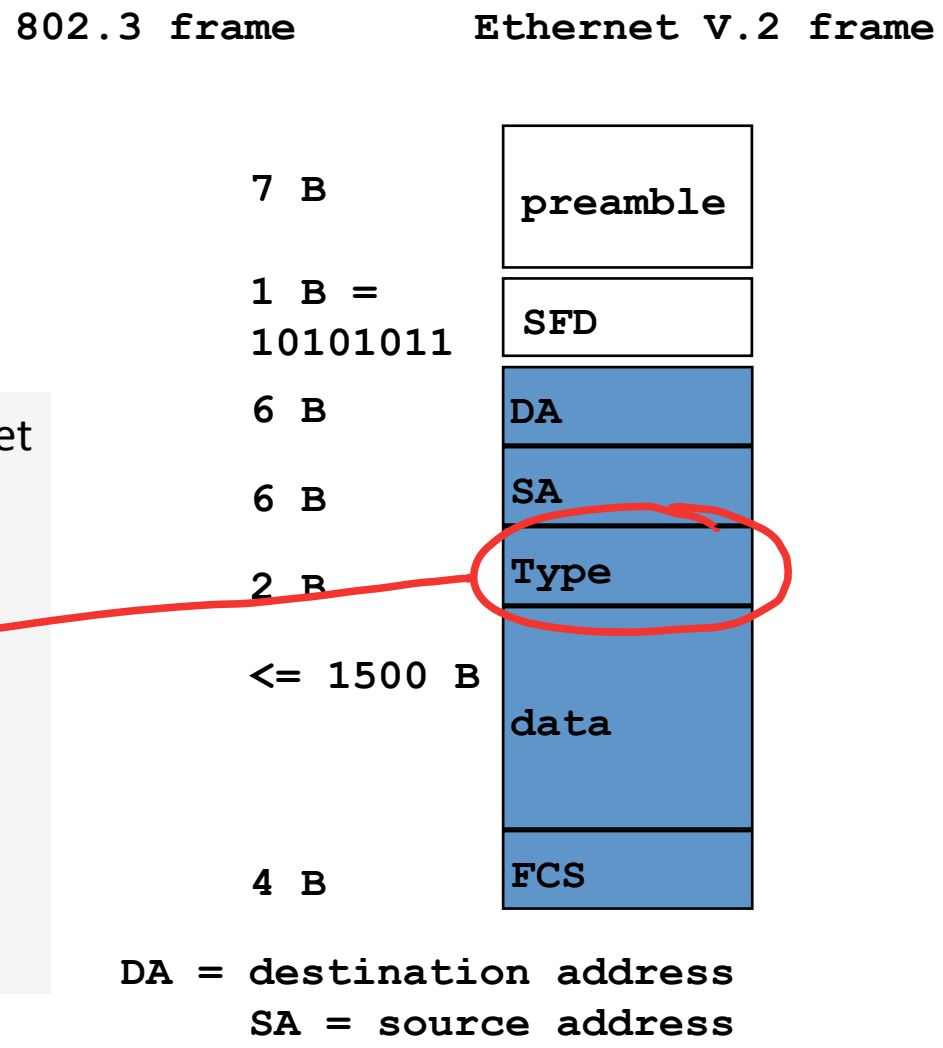
From MAC address to Modified EUI



- Bit 7 of EUI is 1 for EUI derived from globally assigned MAC addresses
- Bit 7 of EUI is 0 for locally assigned address
  Ex: 2001:620:618:100::1

Ethernet Frame format

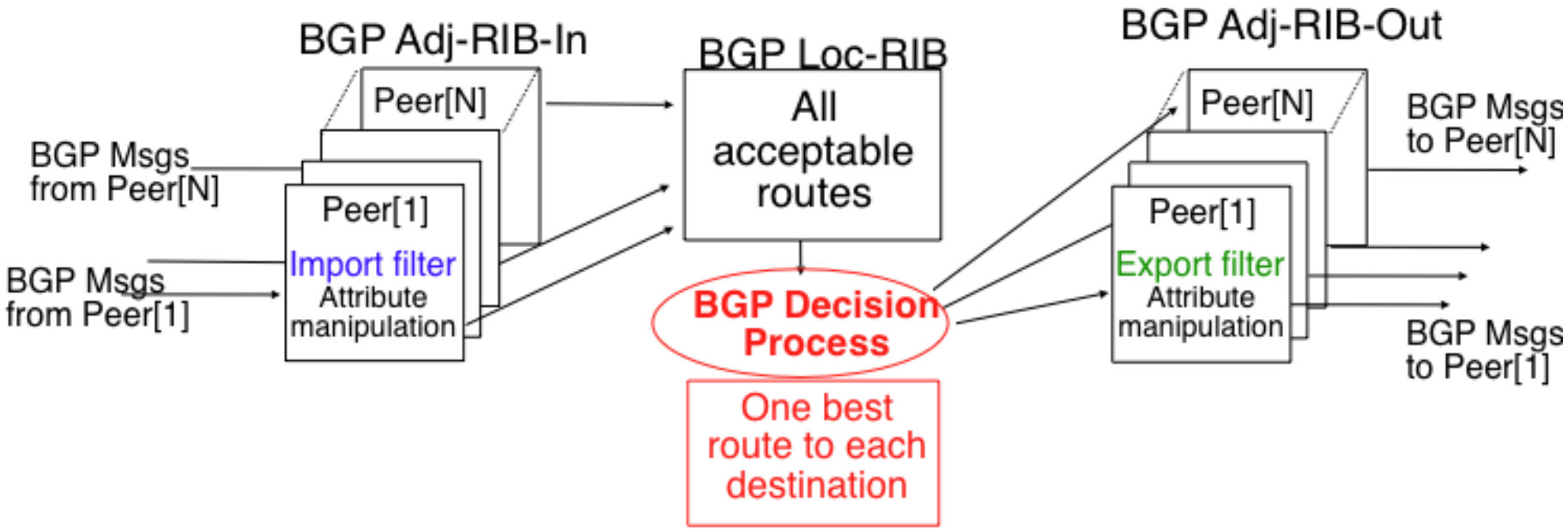
- Ethernet frame = Ethernet PDU
- An Ethernet frame typically transports an IP packet, sometimes also other



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

Model of a BGP Router



FIB = Routing table

Routes, RIBs, Routing Table

- The records sent in BGP messages are called "**Routes**". Routes + their attributes are stored in the Adj-RIB-in, Loc-RIB, Adj-RIB-out.
  A route is made of:
  - destination (subnetwork prefix)
  - path to the destination (AS-PATH)
  - Attributes
    - Well-known Mandatory
      - ORIGIN (route learnt from IGP, BGP or static)
      - AS-PATH
      - NEXT-HOP
    - Well-known Discretionary
      - LOCAL-PREF (see later)
      - ATOMIC-AGGREGATE (= route cannot be dis-aggregated)
    - Optional Transitive
      - MULTI-EXIT-DISC (MED)(see later)
      - AGGREGATOR (who aggregated this route)
    - Optional Nontransitive
      - WEIGHT (see later)
- In addition, like any IP host or router, a BGP router also has a **Routing Table** = IP forwarding table
  - Used for packet forwarding, in real time

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. The default 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

Dijkstra's Shortest Path Algorithm

- The nodes are `0...N` and the algorithm computes best paths from node `0`
- `c(i,j)` is the cost of `(i,j)`,
- `pred(i)` is the predecessor of node `i` on the tree `M` being built
- `m(j)` is the distance from node `0` to node `j`.

```
m(0) = 0; M = {0};
for k=1 to N {
   find (i0, j0) that minimizes m(i) + c(i,j),
                        with i in M, j not in M
   m(j0) = m(i0) + c(i0, j0)
   pred(j0) = i0
   M = M ∪ {j0}
}
```

- as Bellman-Ford, works for any min-plus algebra

The *Centralized* Bellman-Ford Algorithm

- **What:** Given a directed graph with links costs A(i,j), computes the best path from i to j for any couple (i,j).
  - We assume A(i, j) > 0 and A(i,j) = $\propto$ when i and j are not connected.
- **How**: Take for example j=1 and let p(i) be the cost of the best path from i to 1.
  - Define $p^k(i)$ as the cost of the best path from i to 1 in at most k hops. Let $p^0(1) = 0$, $p^0(i) = \propto$ for $i \neq 1$.

Algorithm BF-C

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

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

- **Theorem**
  1. If the network is fully connected, the algorithm stops at the latest for k=n and then $p^k(i)=p(i)$ for all i
  2. The shortest path from $i \neq 1$ to 1 is defined by pred(i) = $\text{Argmin}_{j \neq i}$ [A(i,j) + p(j)].

Distributed Bellman Ford

- BF-C can be used to compute p(i) i.e. find the shortest path. However, this is not its main interest, because there is a better algorithm (Dijkstra) that can be used in a centralized way.
- But: it can be distributed, as follows.

Distributed Bellman-Ford Algorithm , BF-prelim

node i maintains an estimate $q(i)$ of the distance $p(i)$ to node 1;
node i also keeps a record of latest values $q(j)$ for all neighbors j
initial conditions are arbitrary but $q(1)=0$ at all steps;

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

when i receives an updated value $q(j)$ from neighbor j, node i recomputes $q(i)$:
eq (1) $q(i) \leftarrow \min_{\neg j \text{ neighbor}} (A(i,j)+q(j))$
pred(i) is set to a value of j that achieves the min in eq(1)

- **Theorem**: if the time to reliably send a message is bounded by T, the algo converges to the same result as the centralized version in at most nT time units (if the network is fully connected).

Distributed Bellman-Ford, the true version

- Requires only to remember distance from self to destination + the best neighbour (pred(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 pred(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 $j==\text{pred}(i)$
then $q(i) \leftarrow A(i,j)+q(j)$
else $q(i) \leftarrow \min(A(i,j)+q(j),\ q(i))$

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

Fairness of TCP

- A: TCP tends to distribute rate so as to maximize utility of source given by

$$\frac{\sqrt{2}}{\tau_i} \arctan \frac{x_i \tau_i}{\sqrt{2}}$$

with x_i = rate, τ_i = RTT for source i

TCP Loss - Throughput Formula

$$\theta = \frac{LC}{T\sqrt{q}}$$

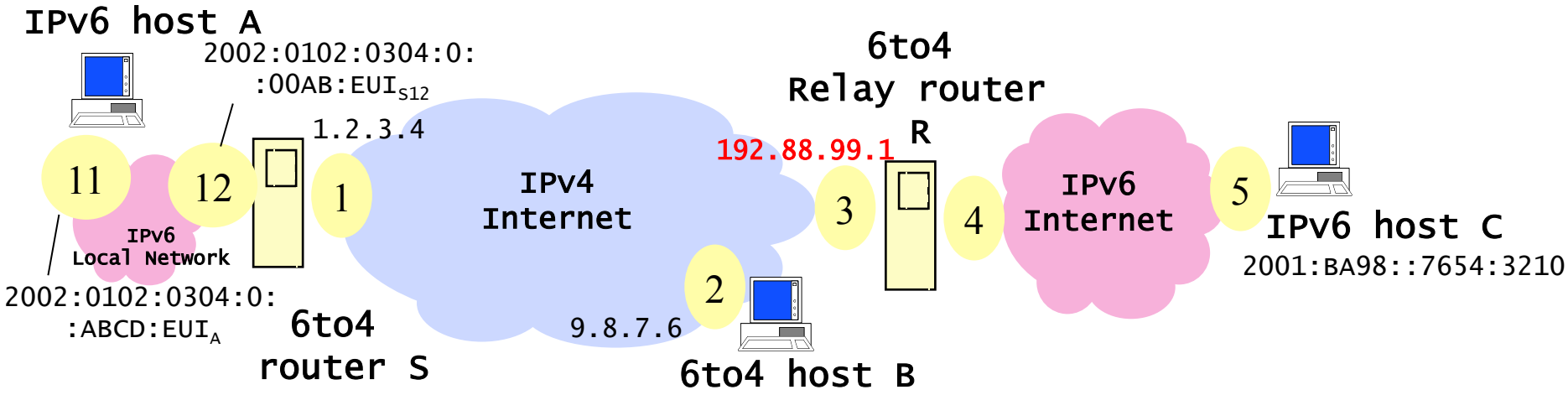
- TCP connection with
  - RTT *T*
  - segment size *L*
  - average packet loss ratio *q*
  - constant *C* = 1.22
- Transmission time negligible compared to RTT, losses are rare, time spent in Slow Start and Fast Recovery negligible

6to4 Addresses

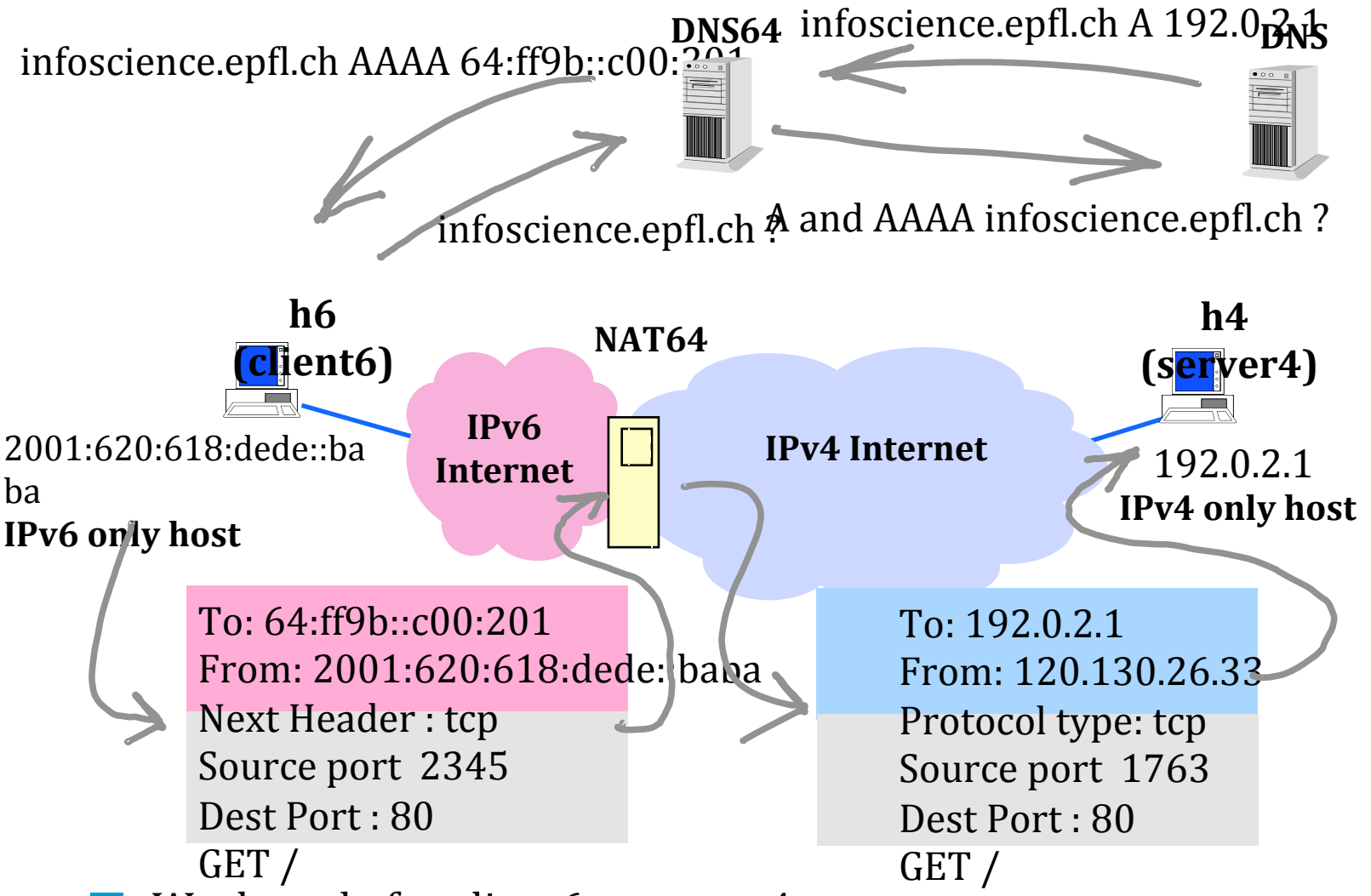
- To any valid IPv4 address *n* we associate the IPv6 prefix 2002:n / 48
  example: the 6to4 address prefix that corresponds to
  128.178.156.38 is
  2002: 80b2:9c26
- An IPv6 address that starts with 2002:… is called a 6to4 address
- The bits 17 to 48 of a 6to4 address are the corresponding IPv4 address
- 2002::/16 is the prefix reserved for 6to4 addresses
- A 6to4 host or router is one that is dual stack and uses 6to4 as IPv6 address
  - As in the IEW
- In addition, the IPv4 address 192.88.99.1 is reserved for use in the context of 6to4 addresses

6to4 Relay Routers

- 6to4 Router = a dual stack router that has a 6to4 address, can terminate routers and connects an IPv6 island to the IPv4 internet
- 6to4 *Relay* Router = a dual stack router that has a 6to4 address, can terminate routers and connects the IPv4 and IPv6 internets
  All v4 interfaces of relay router have an IPv4 address plus the address 192.88.99.1



Stateful NAT64, putting things together



- Works only for client6 to server4