ROSP 903 2018

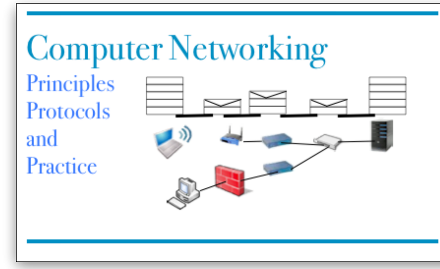
Introduction to TCP/IP Networking

Luigi Iannone

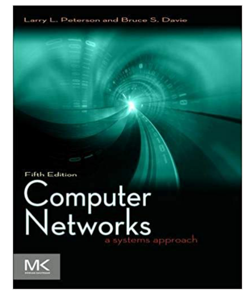
luigi.iannone@telecom-paristech.fr

References

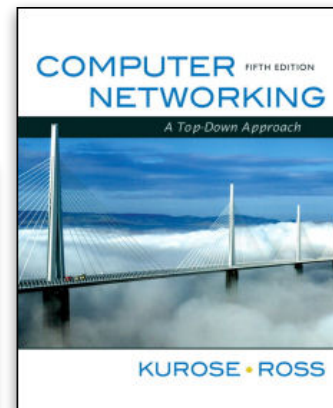
- Bibliography
  - Computer Networking: Principles, Protocols, and Practice
    - by O. Bonaventure
    - http://cnp3book.info.ucl.ac.be/1st/html/_downloads/cnp3.pdf



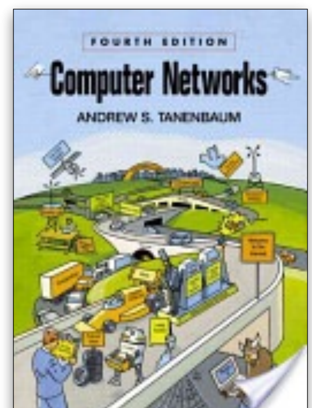
  - Computer Networks: a System Approach
    - By L. Peterson & B.S. Davie
    - https://github.com/SystemsApproach/book



  - Computer Networking - A Top Down Approach
    - by J. Kurose & K. Ross



  - Computer Networks
    - by A. S. Tanenbaum



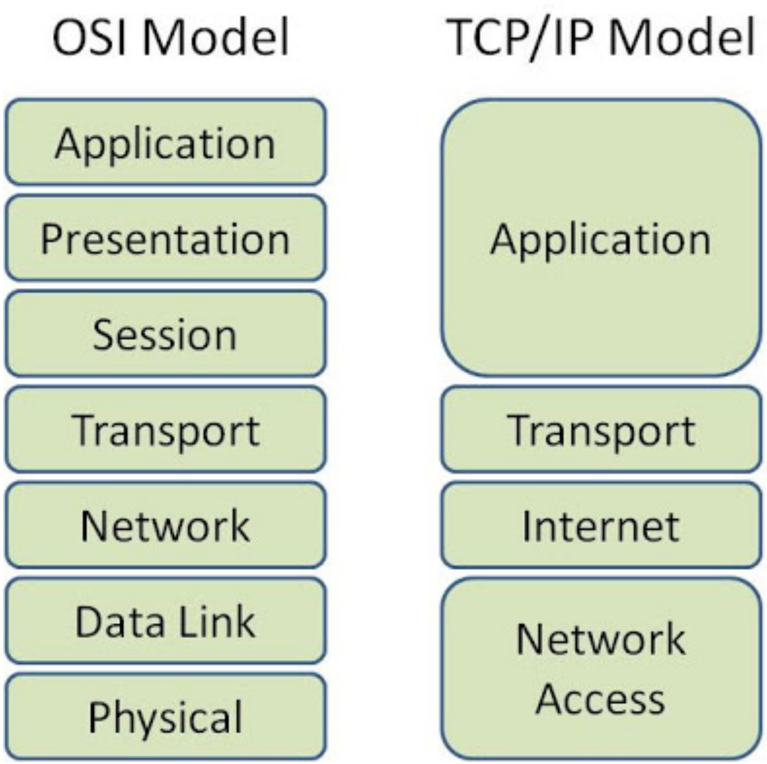
  - These slides:
    - **https://rosp.wp.imt.fr/curriculum/refresher-communication-networks/**

Road Map

- Internet & Layering
- Transport Layer
  - Layer 4
- Network Layer
  - Layer 3

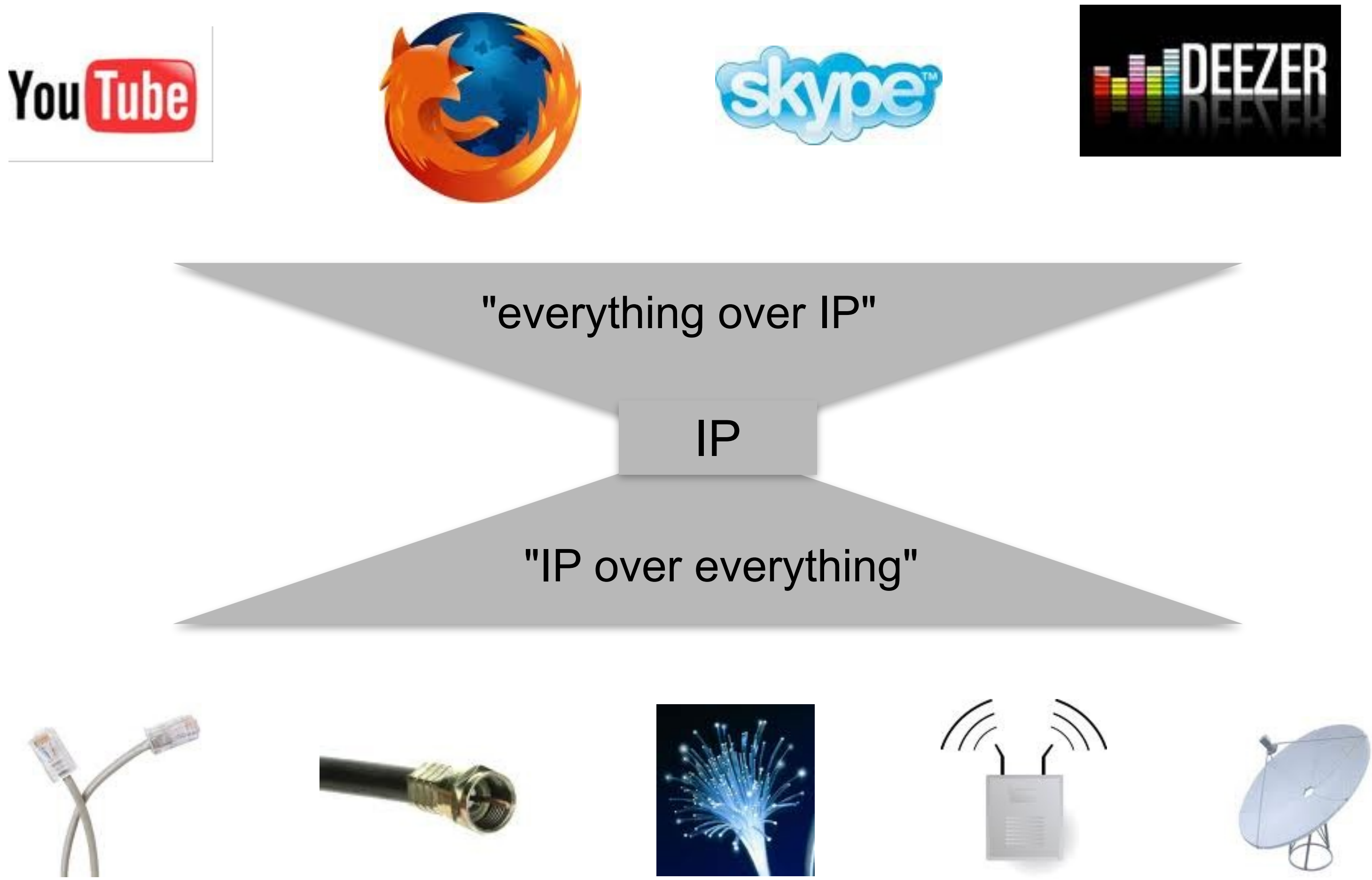
Internet & Layering

Why Layers?

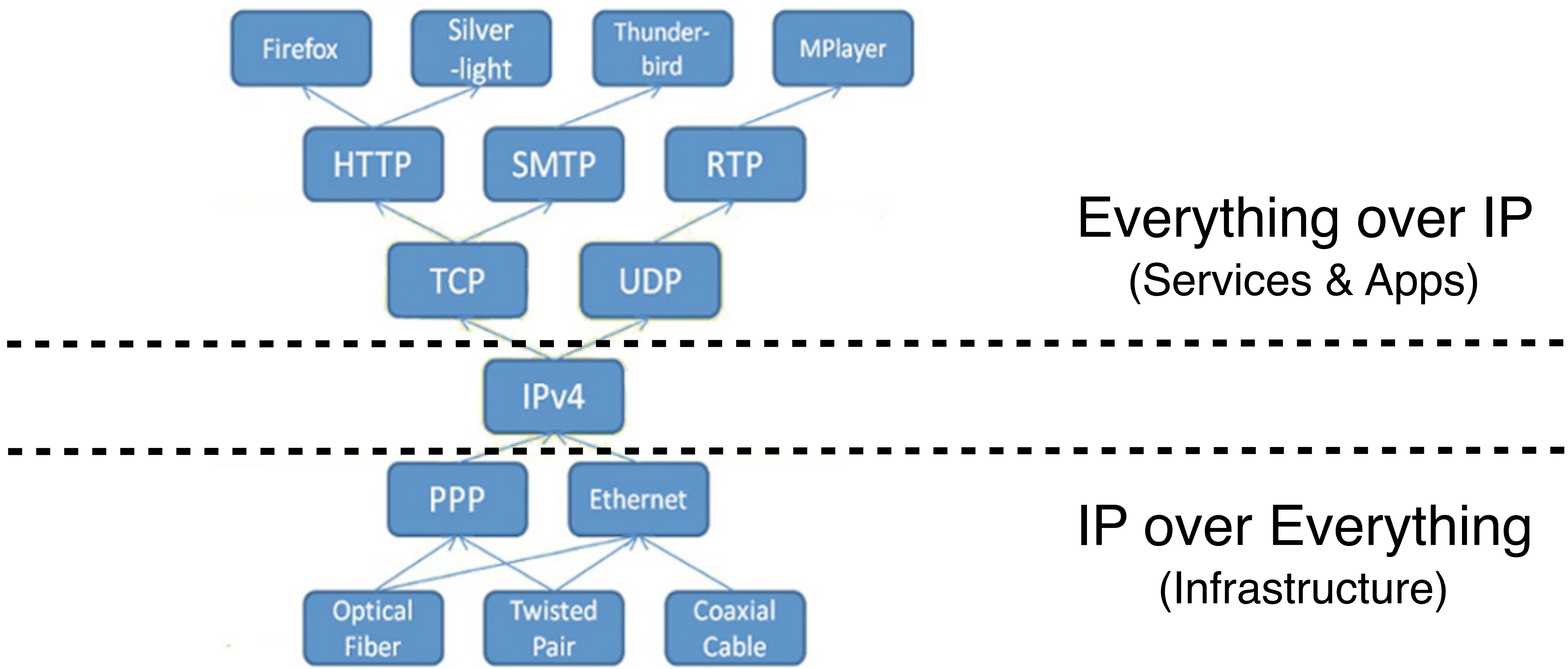


- Presents data to the user, encoding and session control
- Support of communication between diverse devices and networks
- Determines the path in the network
- Controls the hardware components of the network

Thin waist…



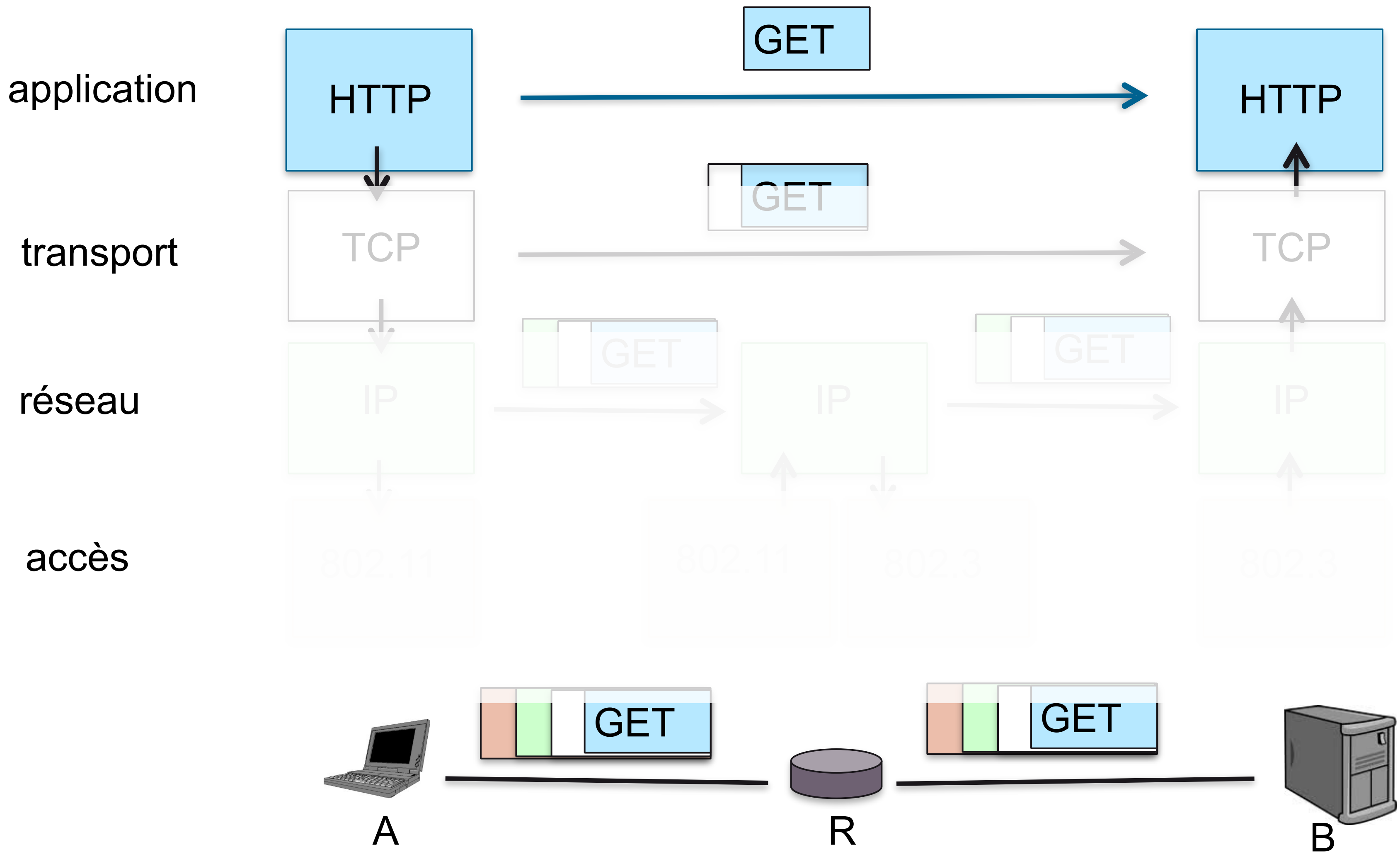
TCP/IP Protocol Stack Software



Everything over IP
(Services & Apps)

IP over Everything
(Infrastructure)

Layered Approach



Upper Layers

Road Map

- Internet & Layering
- Transport Layer
  - Sockets and Port Numbers
  - UDP
    - Header
  - TCP
    - Header
    - Connection setup & teardown
    - Flow Control
    - Congestion Control
  - DNS
- Network Layer
  - Layer 3

Client/Server Architecture - WEB

- HyperText Transfer Protocol - HTTP (RFC 2616)

Requests:

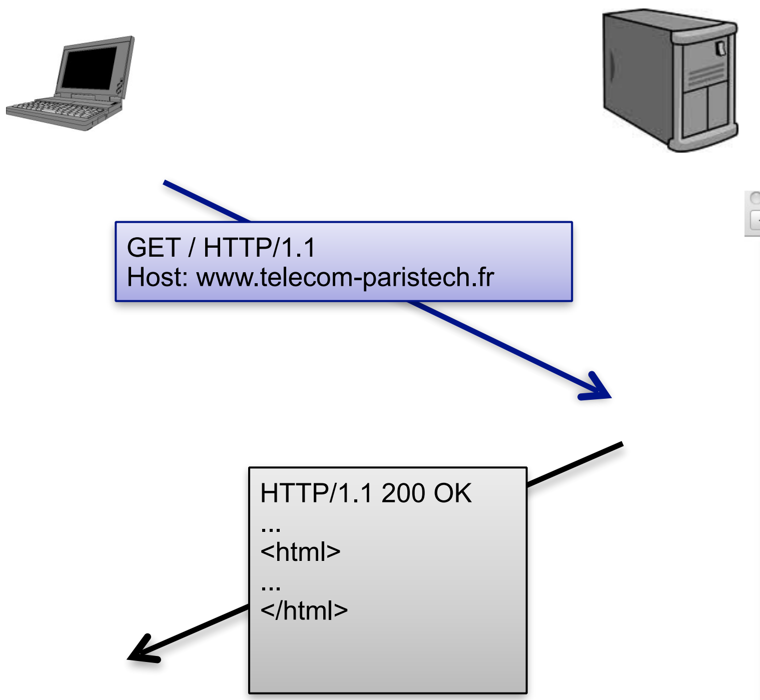
GET, POST, PUT, DELETE, ...

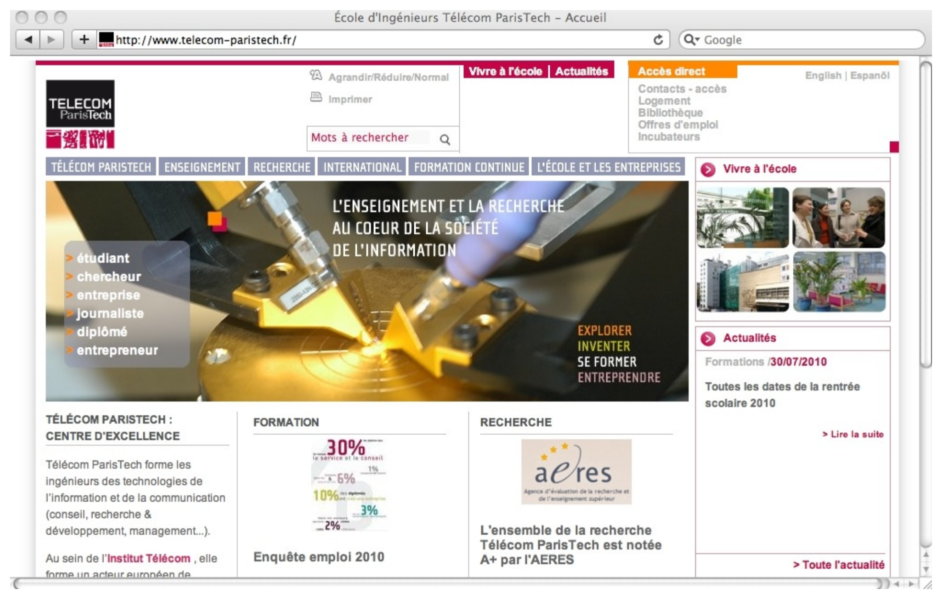
Replies:

200, 301, 302, 400, 404, 505, ...

Client HTTP

Serveur HTTP

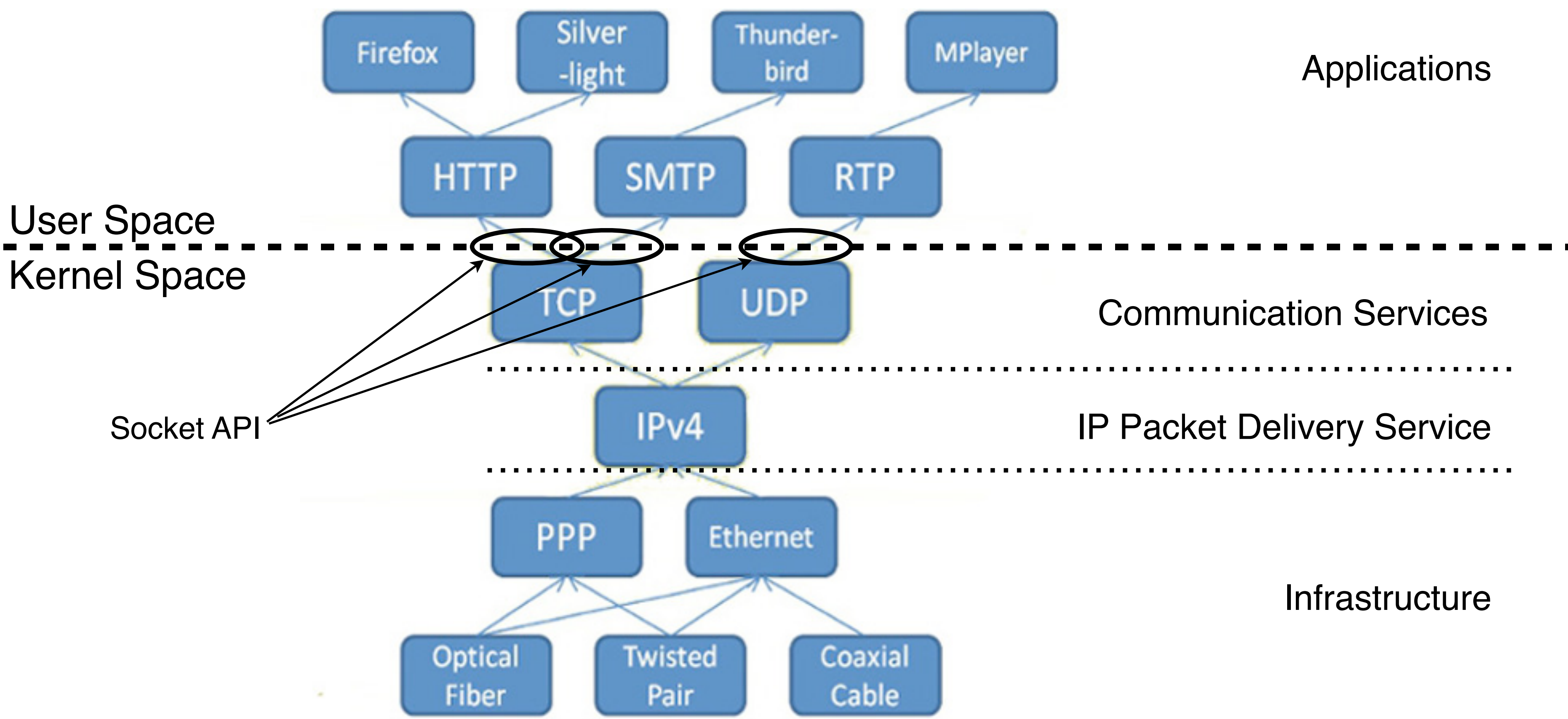




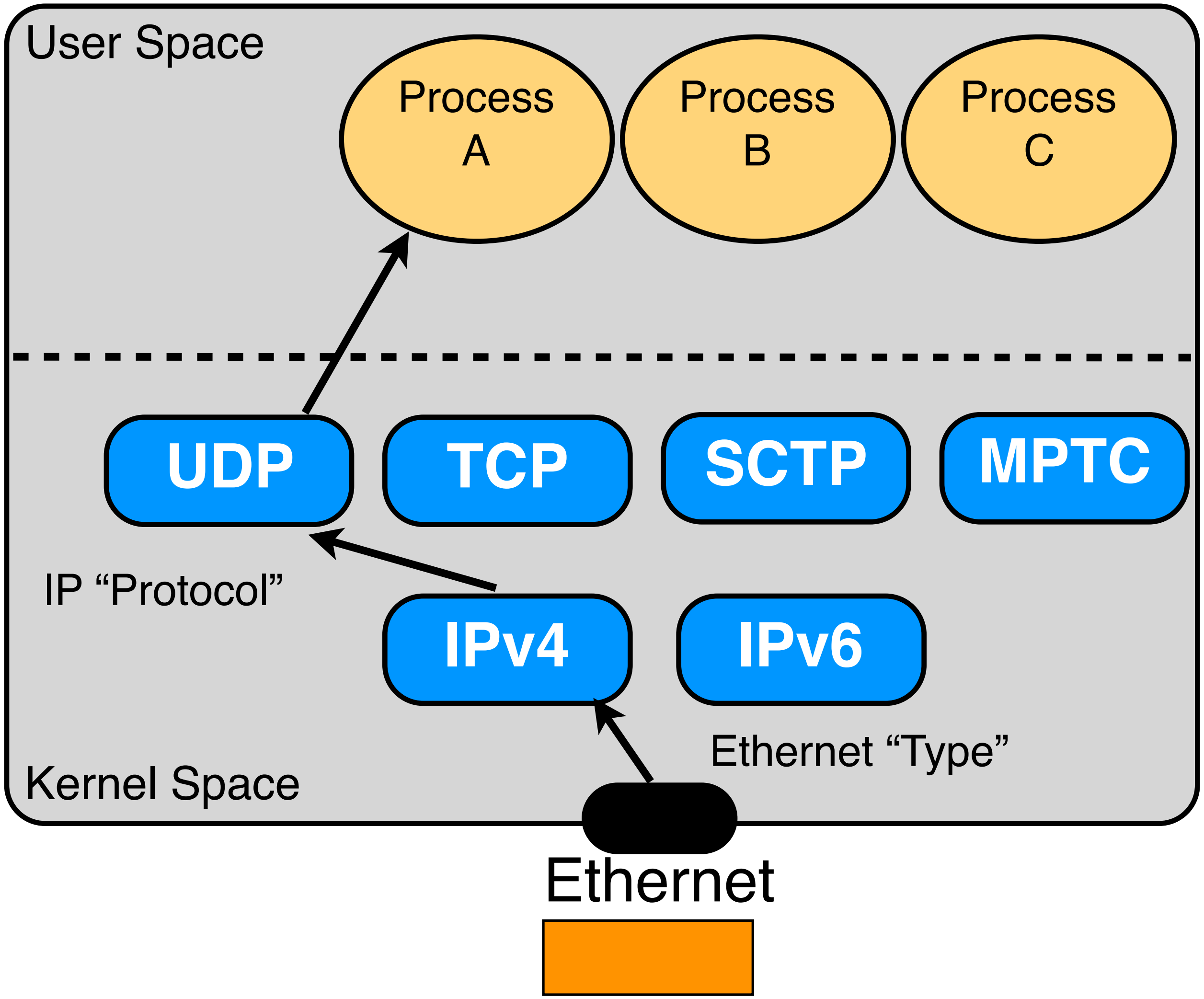
Hands-on: (g)netcat -vt www.google.fr 80 | GET

TCP/IP Protocol Stack Software (II)

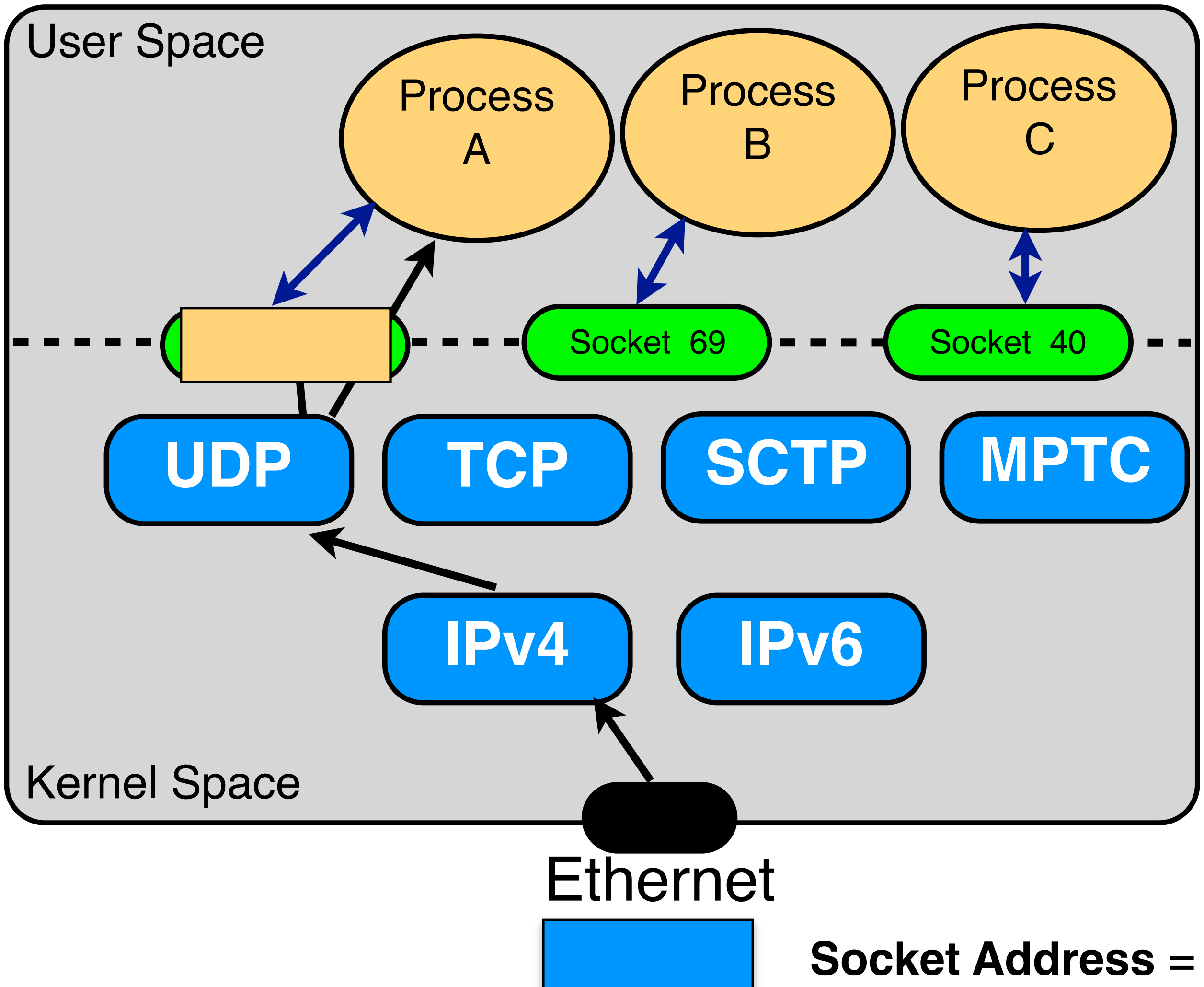
Port Numbers Allow multiplexing/demultiplexing several flow from/to a machine



Packet Demultiplexing - Kernel Space



Packet Demultiplexing - From Kernel to User Space



Transport layer sends/receives data from Apps through sockets

Sockets ID (port number) allow packet demultiplexing

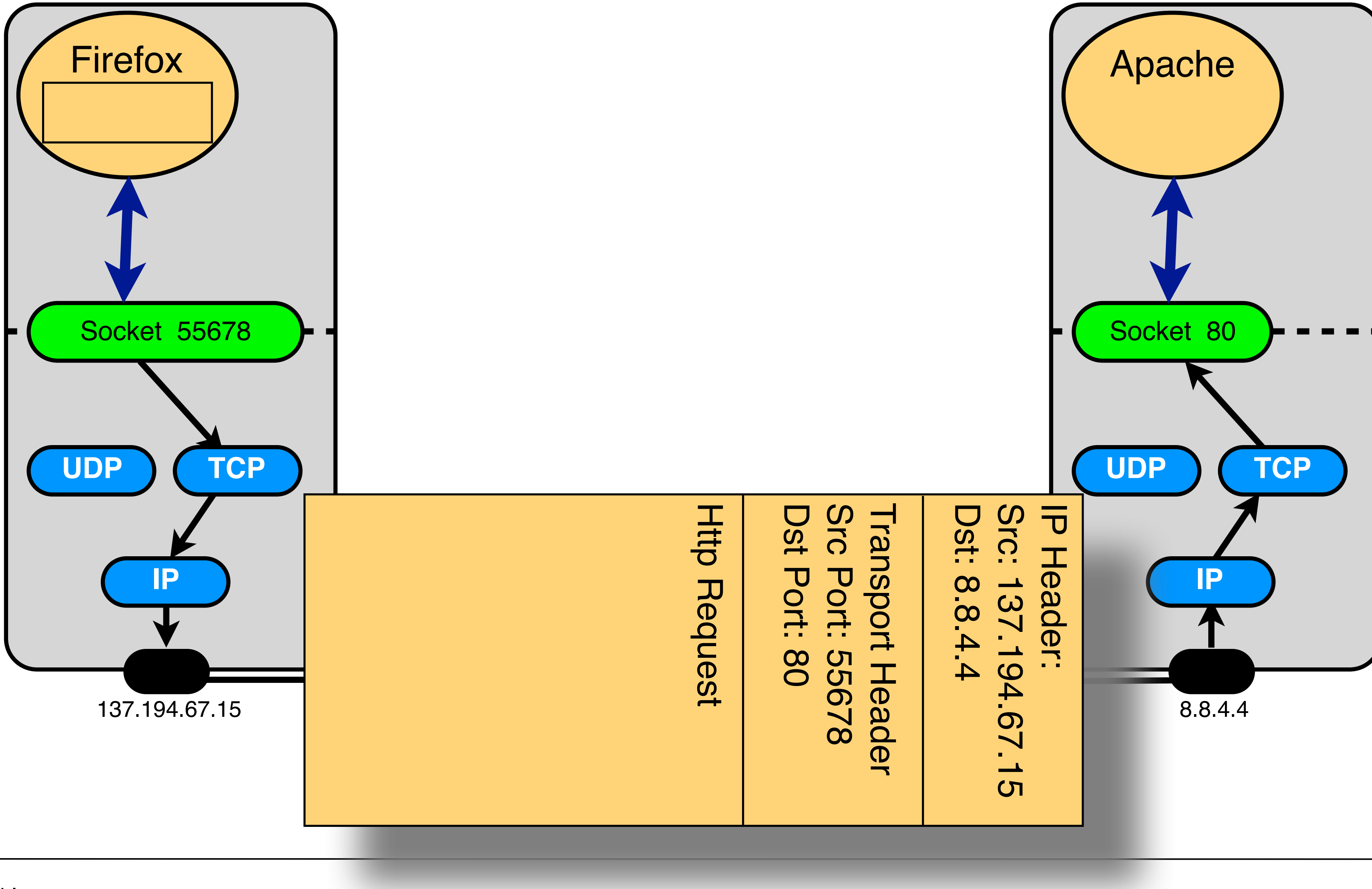
Socket ID is not globally unique (OS scope)

- 16 bits (0 -> 65535)

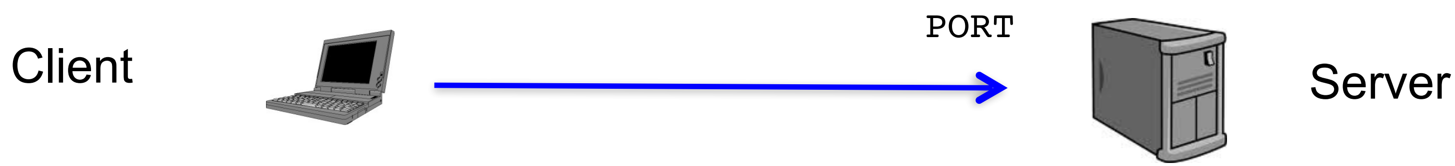
Sockets should have globally unique identifiers (socket address) because are the end-points of communications

Socket Address = IP Address & Port Number

Socket (Process) to Socket (Process) Communication



Socket Programming in Python



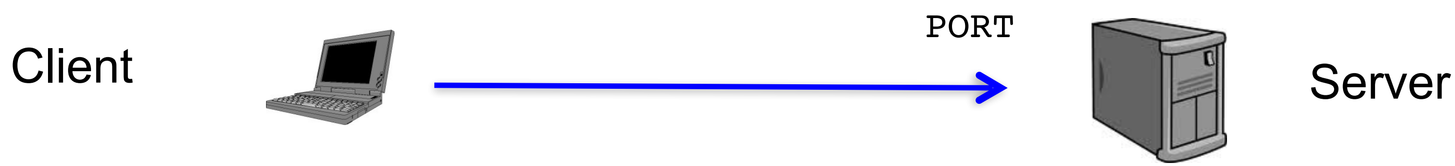

```
#!/usr/bin/python      # This is client.py file

import socket          # Import socket module

s = socket.socket()   # Create a socket object
host = socket.gethostname()#Get local machine name
port = PORT           #Reserve a port for your service

s.connect((host, port))
print s.recv(1024)
s.close               # Close the socket when done
```

Socket Programming in Python




```python
#!/usr/bin/python        # This is server.py file

import socket            # Import socket module

s = socket.socket()      # Create a socket object
host = socket.gethostname() # Get local machine name
port = PORT           #Reserve a port for your service
s.bind((host, port))     # Bind to the port

s.listen(5)           # Now wait for client connection
while True:
   newconnection, addr = s.accept()#Establish connect
                         # with client
               #Here you could eventually fork
   print 'Got connection from', addr
   newconnection.send('Thank you for connecting')
   newconnection.close()       # Close the connection
```

Type of Port Numbers

- IANA defines three types of port numbers:

- Well-Known
  - Range: 0 -> 1023
  - System processes providing common network services
  - Only super user can use them

- Registered
  - Range: 1024 -> 49151
  - Specific services requested by companies
  - Can be used for other purposes

- Dynamic (also known are private or ephemeral)
  - Range: 49152 -> 65535
  - Automatic allocation by the OS to each open socket

Transport Layer

TCP vs. UDP

- TCP - Transmission Control Protocol
  - Connection Oriented
    - explicit establishment and termination (stateful protocol)
  - Byte Stream
    - data is read as a stream of byte
  - Reliable data transmission
    - retransmission of lost packets
  - Ordered transmission
    - data delivered to the App in the same order as it has been transmitted by the source App
    - (Does not mean it is received in the same order)
  - Flow Control
    - to avoid overflowing the destination
  - Congestion Control
    - adapt transmission rate to the network congestion

- UDP - User Datagram Protocol
  - Connectionless
    - like IP
  - Datagram
    - data is encapsulated in individual packets
    - like IP
  - Unreliable data transmission
    - like IP
  - No Ordered Transmission
    - data can be delivered in a random order
    - like IP

UDP - User Datagram Protocol

UDP Basics

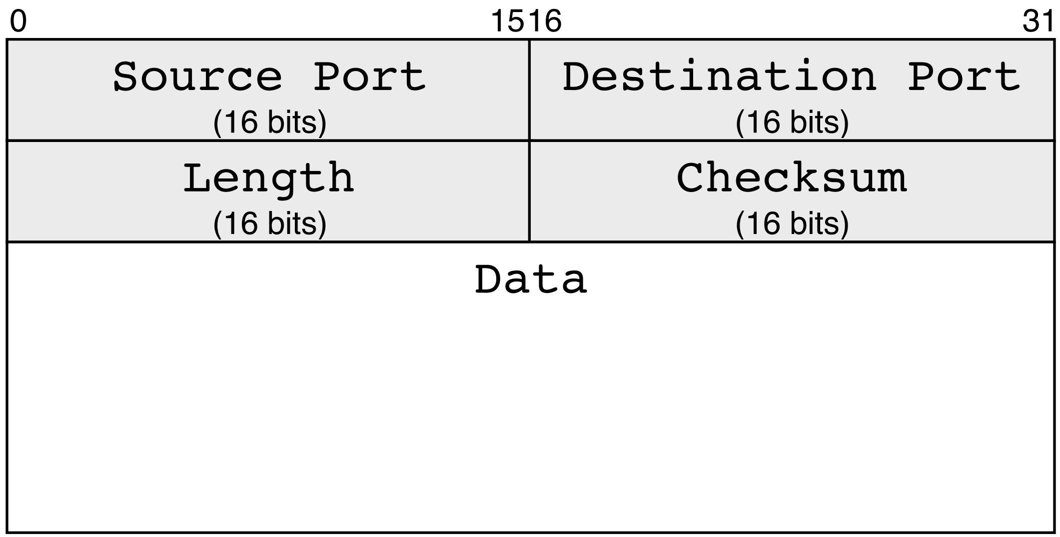
- Connectionless
  - No handshaking between UDP sender, receiver
  - Each UDP datagram handled independently of others

- Unreliable communication
  - No flow, congestion, or error* control
  - A UDP datagram can be lost, arrive out of order, duplicated, or corrupted
  - Checksum field checks error in the entire UDP datagram.
    - Optional IPv4 but mandatory in IPv6
  - No error recovering, datagram simply discarded

- Advantages
  - Simple, minimum overhead, no connection delay

- Protocol Number
  - 17

UDP Datagram



- UDP Header

  `Source Port:` source port number

  `Destination Port:` destination port number

  `Length:` total length of the UDP datagram (including header)

  UDP length = IP length - IP Header

  `Checksum:` error checking code (0 if not used)

- UDP data

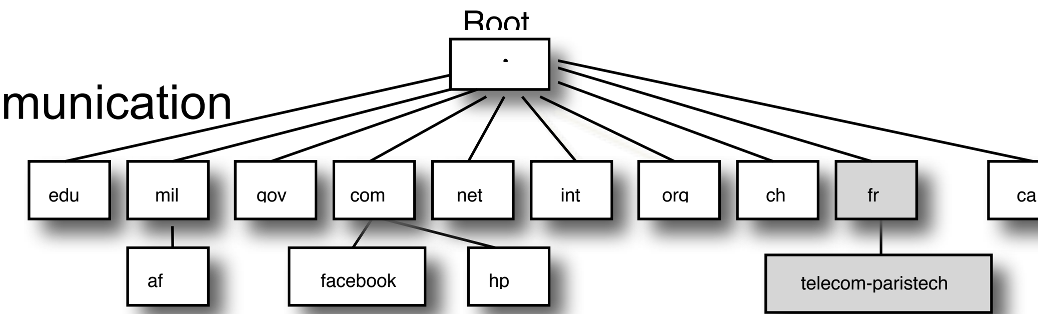
  Application specific data

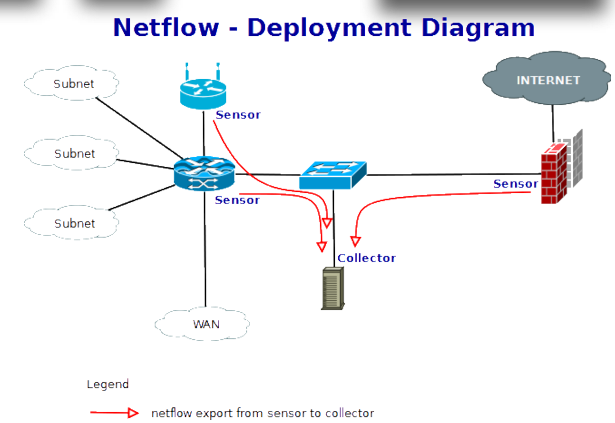
Examples of UDP well-known Ports

Port	Service	Description
22	SSH	Secure Shell
53	DOMAIN	Domain Name Server
67	BOOTPS	Boot protocol server (DHCP)
68	BOOTPC	Boot protocol client (DHCP)
69	TFTP	Trivial File transfert protocol
123	NTP	Network Time Protocol
161	SNMP	Simple Network Management Protocol
631	IPP	Internet Printing Protocol
666	Doom	First Person Shooter

UDP Usefulness

- UDP offers a connectionless service like IP
- So why is it needed?
  - Provide Process to Process Communication
  - Avoid excessive state
    - Example: DNS
  - Fire&Forget Protocols
    - Example: NetFlow
  - Proprietary Rate Control Algorithms
    - Example: Real Time Messaging Protocol (RTMP)





TCP - Transmission Control Protocol

TCP Basics

- Connection Oriented
  - Explicit establishment and termination of a virtual circuit
  - Connection establishment ensures the process at the other end of the virtual circuit exists and is will to communicate
  - Statefull
  - Full-Duplex byte stream communication
- Reliable Communication
  - TCP recovers from corrupted or lost packets
  - TCP preserves transmitted byte order and suppresses duplicated data
  - Flow control
  - Congestion control
- Protocol Number
  - 6

TCP Service vs. TCP Protocol

- Service Unit: byte (octet)
- Protocol Unit: Segment
  - Maximum Segment Size (MSS)
    - MSS = MTU - IP header - TCP header

- TCP on the sending side:
  - does not transmit single bytes
  - Bytes are buffered until MSS are ready to transmit
    - Unless explicit request from process (push)
    - Unless too much time is elapsed
  - Segment created and sent

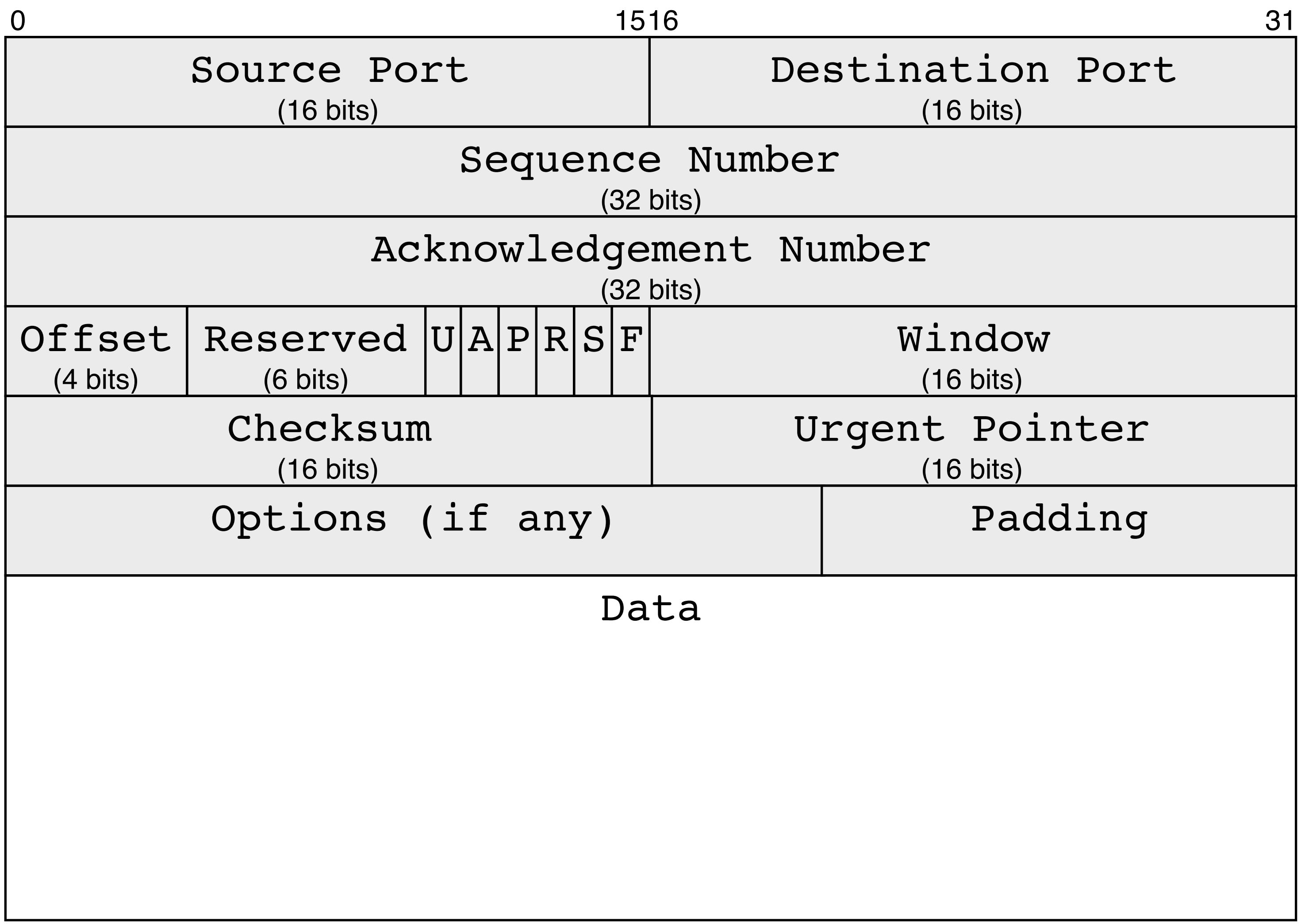
- TCP on the receiving side:
  - Received segments are buffered
  - The dst App reads as many bytes it wants from the buffer (freeing space)

Examples of TCP well-known Ports

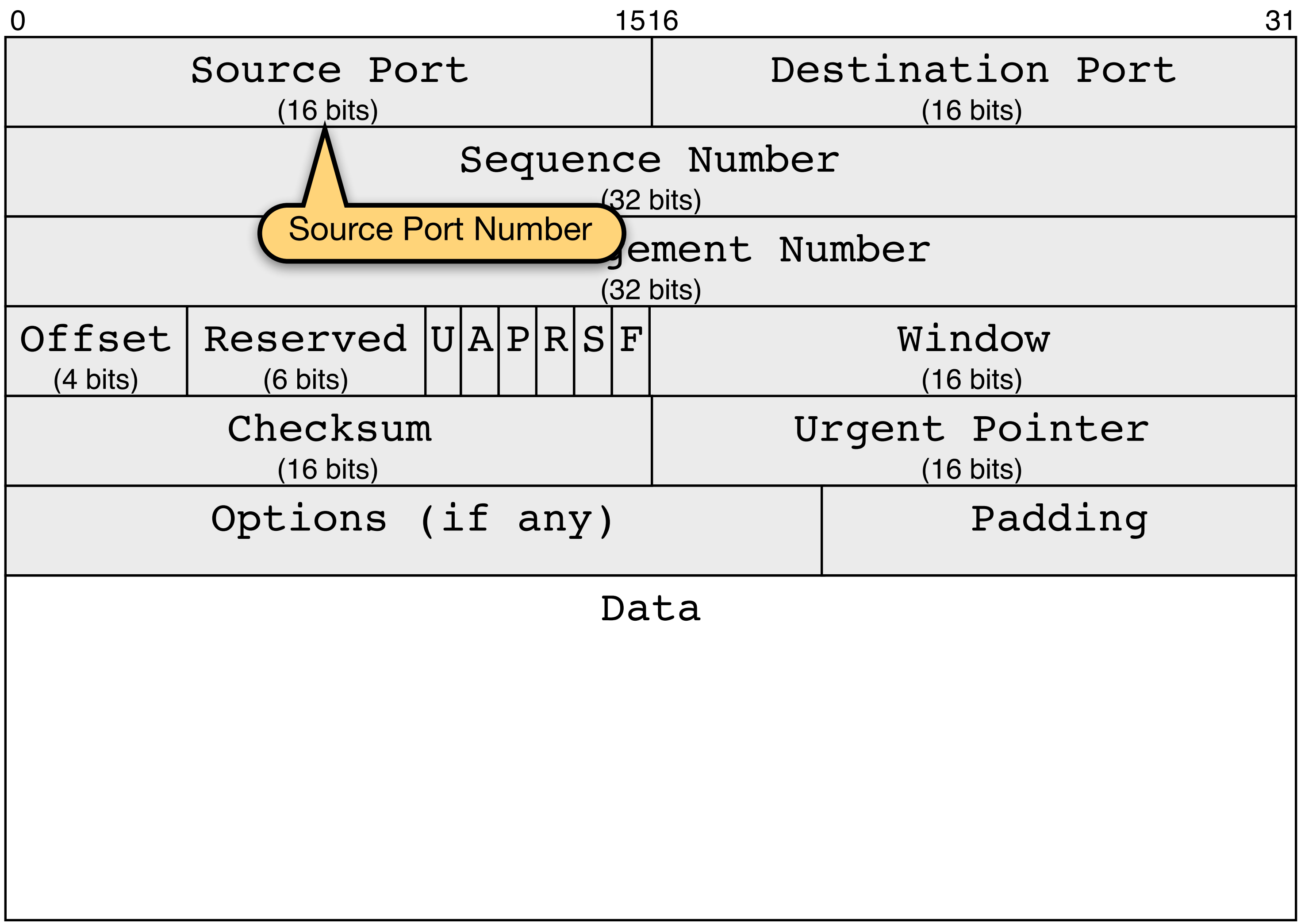
Port	Service	Description
20	FTP-DATA	File Transfer [Default Data]
21	FTP	File Transfer [Control]
22	SSH	Secure Shell
23	TELNET	Telnet
25	SMTP	Simple Mail Transfer Protocol
42	NAMESERVER	Host Name Server
43	NICNAME	Who Is
53	DOMAIN	Domain Name Server (DNS)
80	HTTP	WWW
110	POP3	Post Office Protocol - Version 3
143	IMAP	Internet Message Access Protocol

TCP Segment and Header

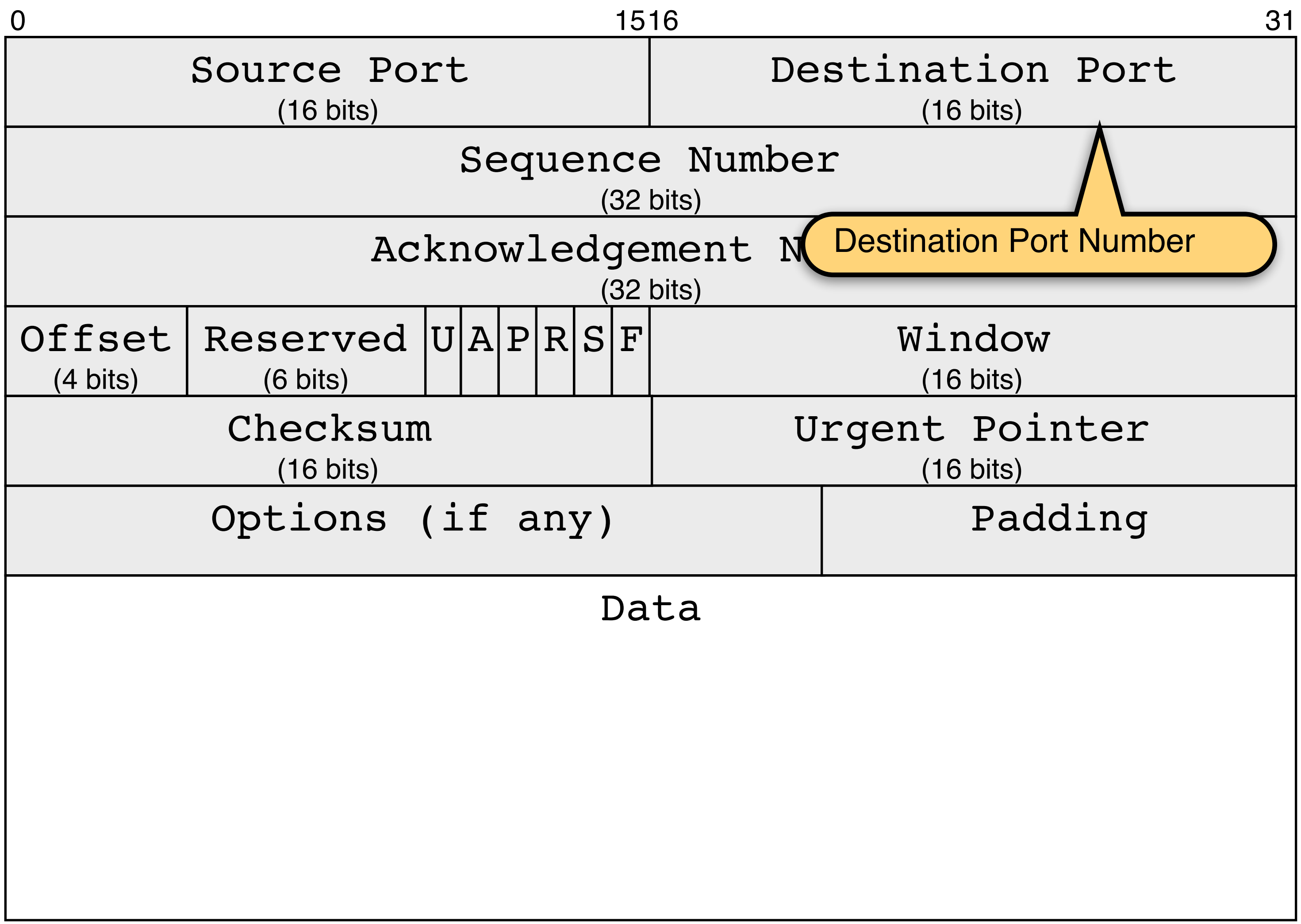
TCP Segment


0
15
16
31
Source Port
(16 bits)
Destination Port
(16 bits)
Sequence Number
(32 bits)
Acknowledgement Number
(32 bits)
Offset
(4 bits)
Reserved
(6 bits)
U
A
P
R
S
F
Window
(16 bits)
Checksum
(16 bits)
Urgent Pointer
(16 bits)
Options (if any)
Padding
Data

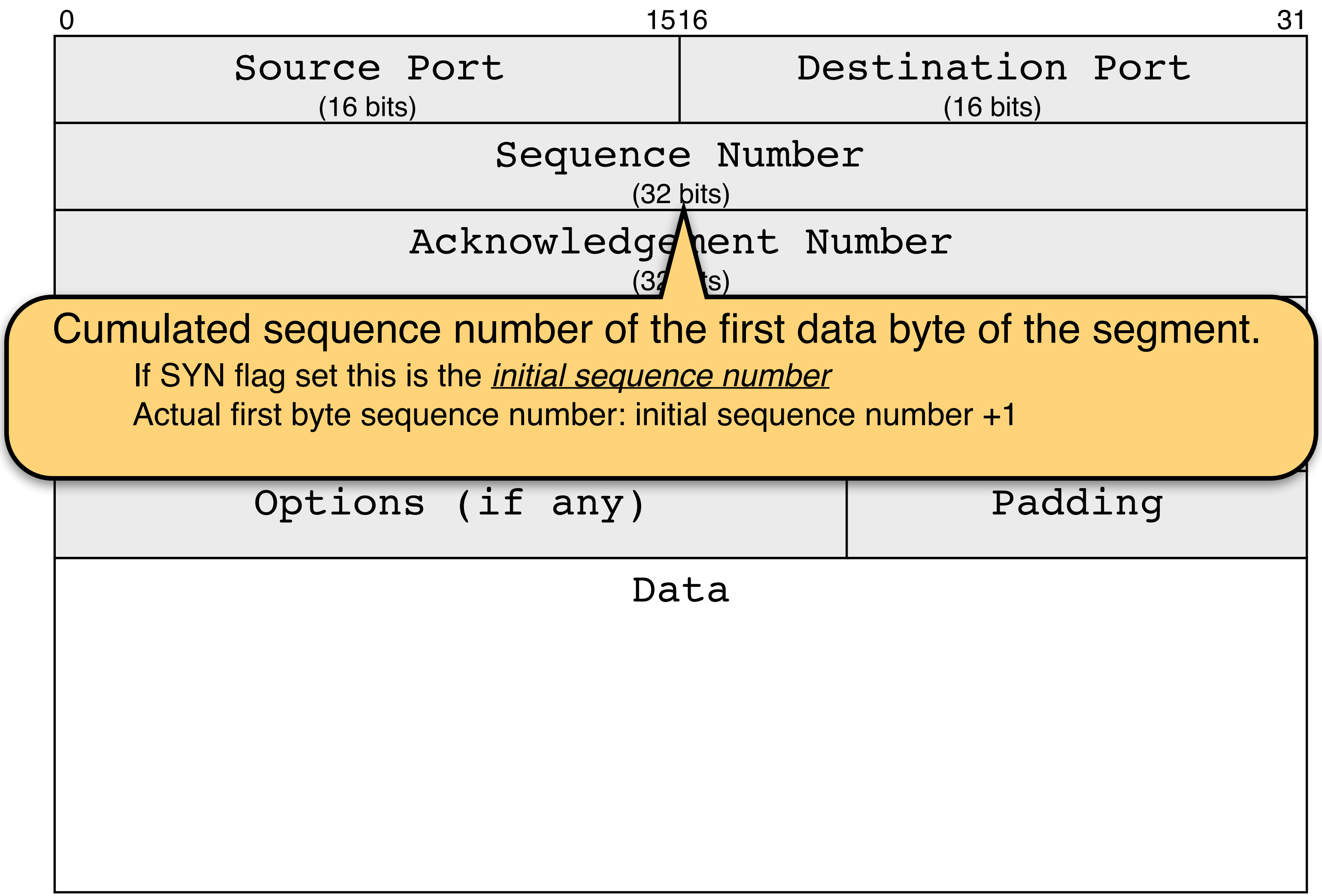
TCP Segment


0
1516
31
Source Port
(16 bits)
Destination Port
(16 bits)
Sequence Number
(32 bits)
Source Port Number
gement Number
(32 bits)
Offset
(4 bits)
Reserved
(6 bits)
U
A
P
R
S
F
Window
(16 bits)
Checksum
(16 bits)
Urgent Pointer
(16 bits)
Options (if any)
Padding
Data

TCP Segment


0
1516
31
Source Port
(16 bits)
Destination Port
(16 bits)
Sequence Number
(32 bits)
Acknowledgement N
(32 bits)
Destination Port Number
Offset
(4 bits)
Reserved
(6 bits)
U
A
P
R
S
F
Window
(16 bits)
Checksum
(16 bits)
Urgent Pointer
(16 bits)
Options (if any)
Padding
Data

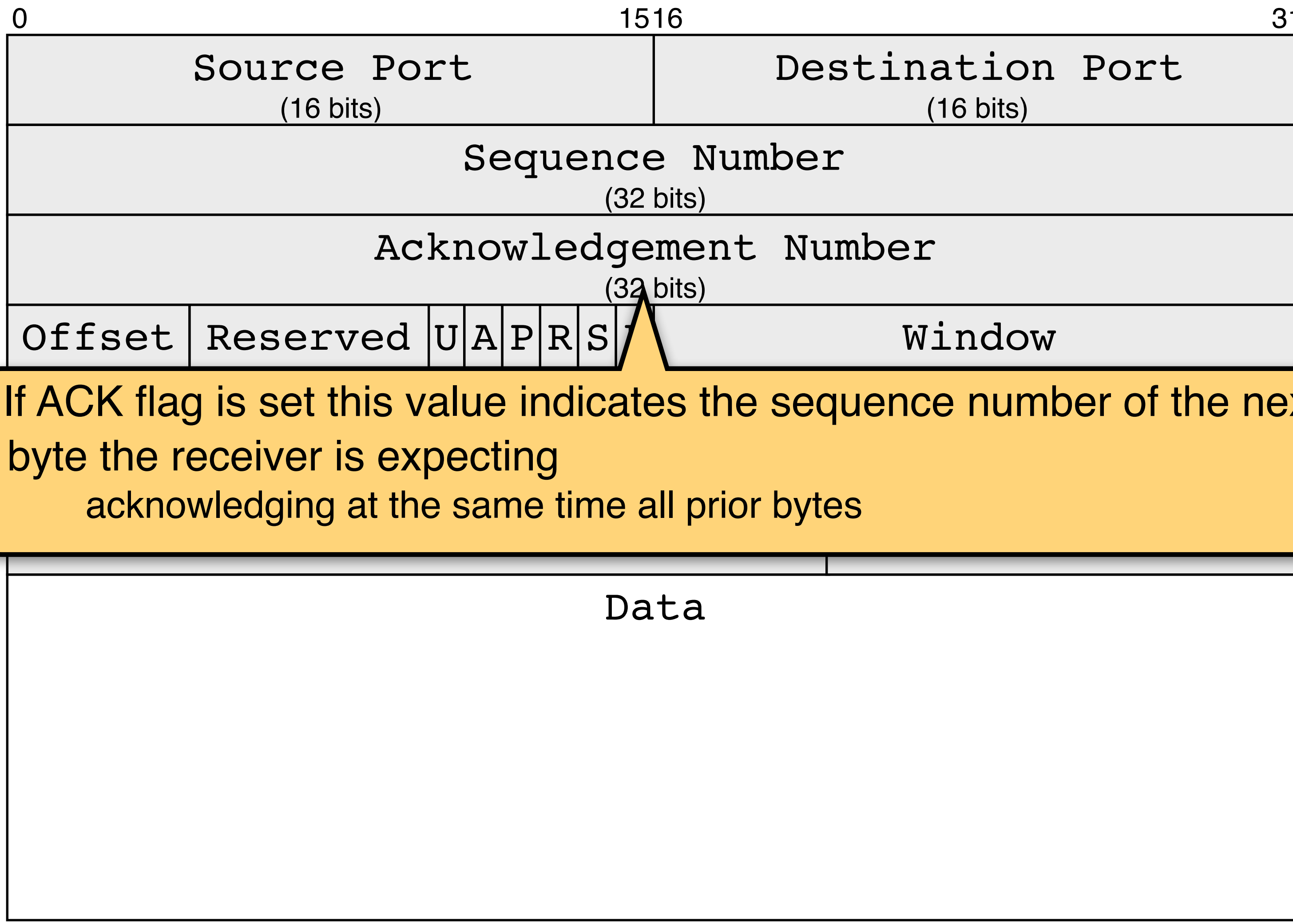
TCP Segment



Cumulated sequence number of the first data byte of the segment.

- If SYN flag set this is the ***initial sequence number***
- Actual first byte sequence number: initial sequence number +1

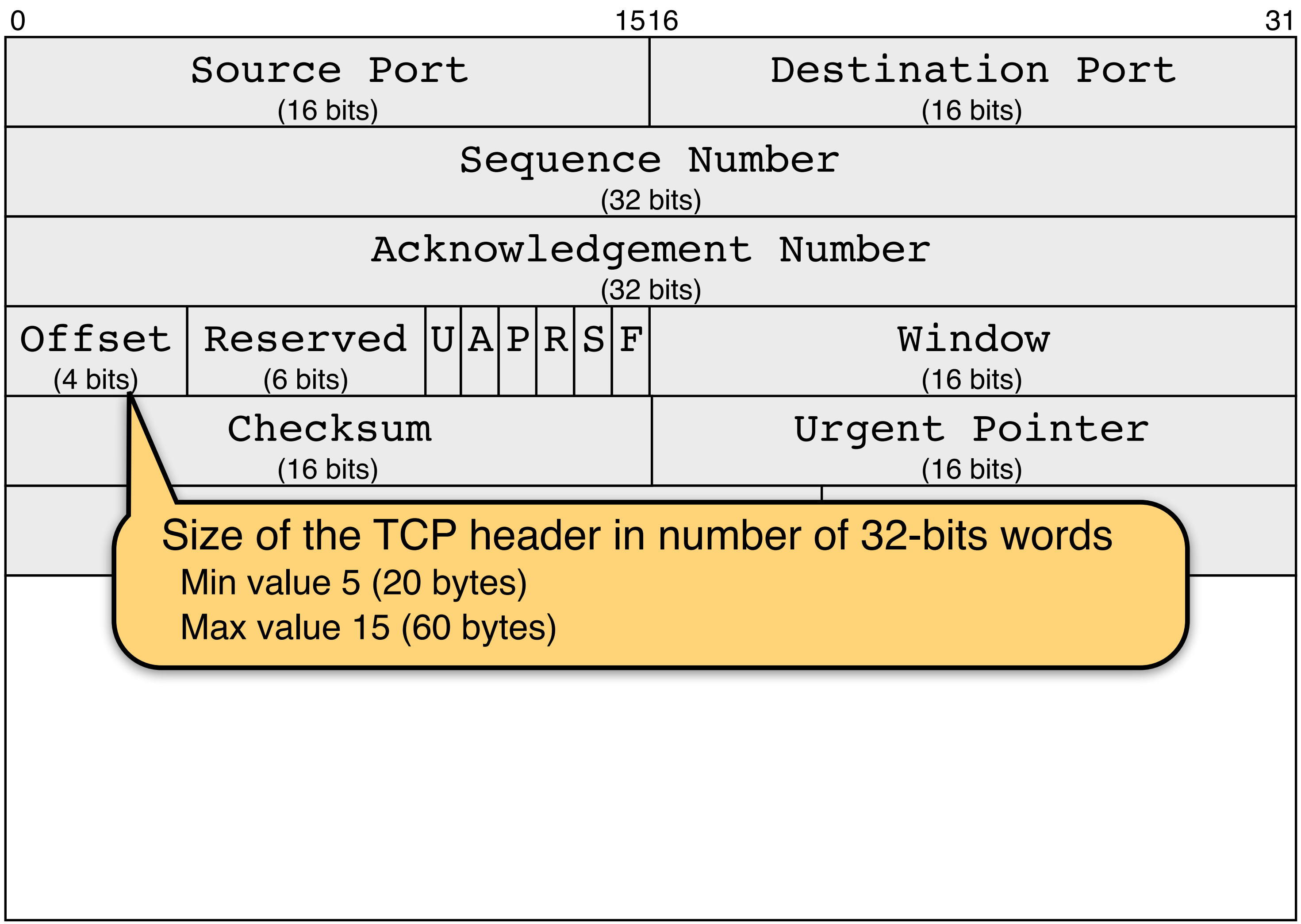
TCP Segment



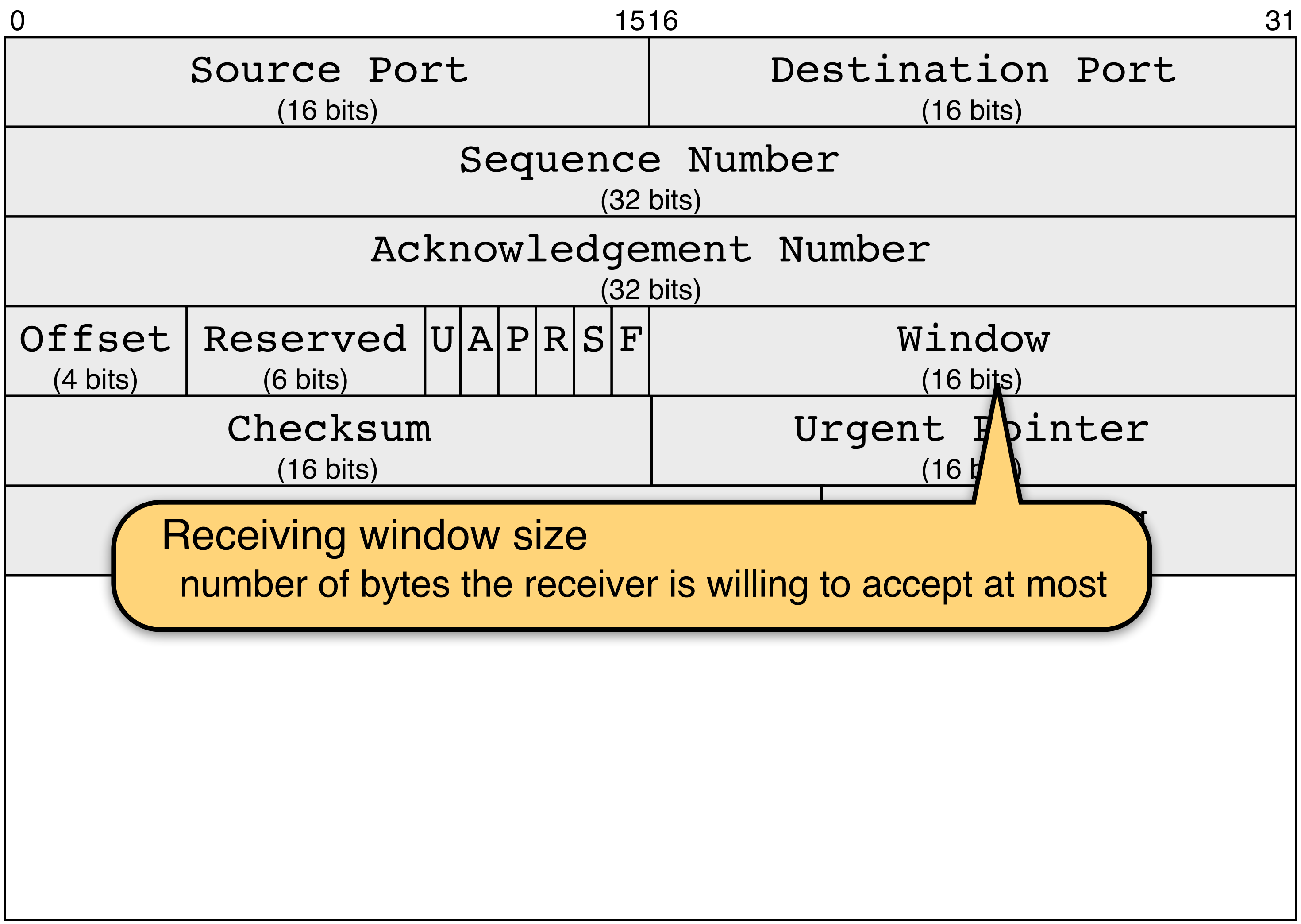
If ACK flag is set this value indicates the sequence number of the next byte the receiver is expecting

acknowledging at the same time all prior bytes

TCP Segment

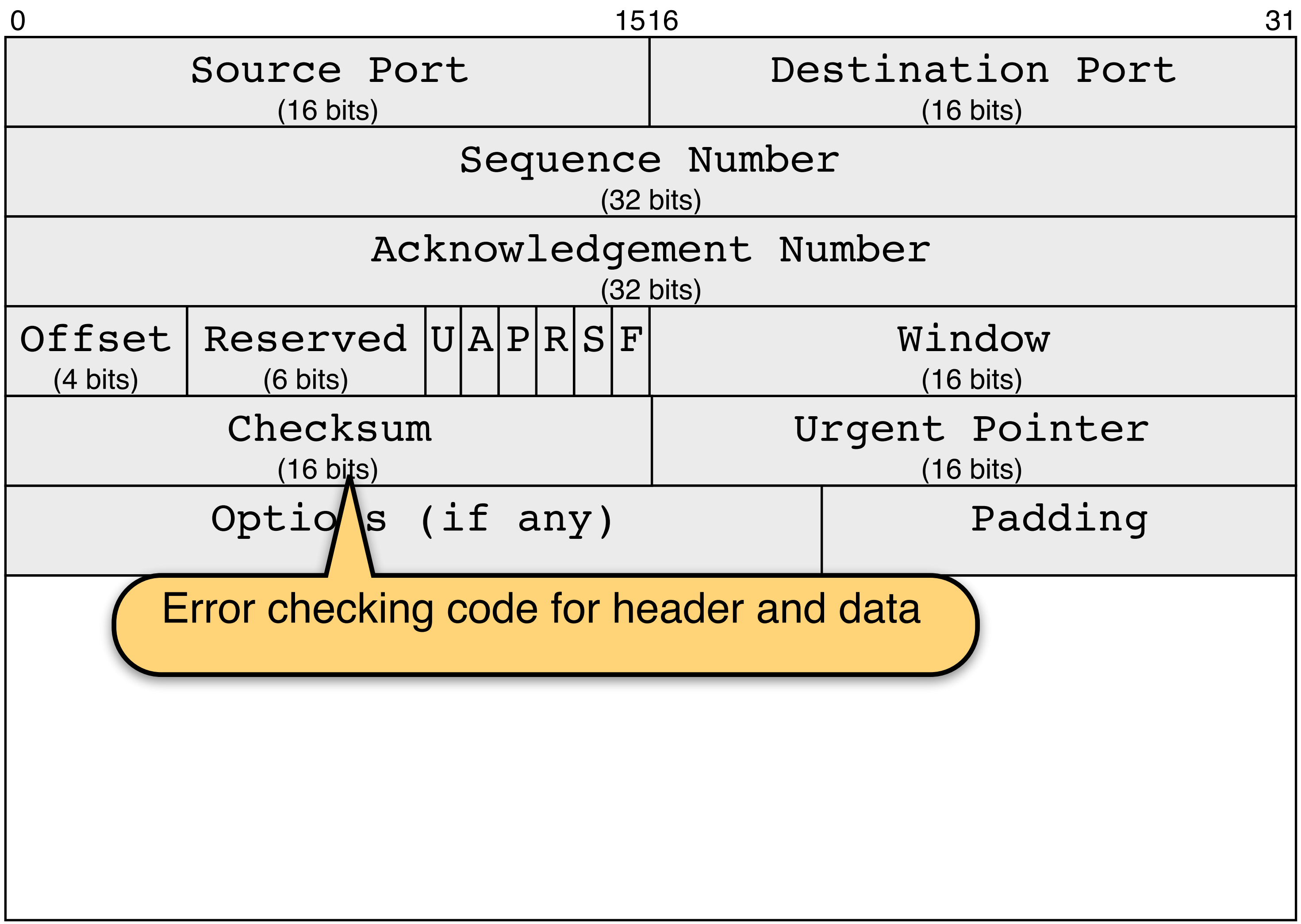

0
15
16
31
Source Port
(16 bits)
Destination Port
(16 bits)
Sequence Number
(32 bits)
Acknowledgement Number
(32 bits)
Offset
(4 bits)
Reserved
(6 bits)
U
A
P
R
S
F
Window
(16 bits)
Checksum
(16 bits)
Urgent Pointer
(16 bits)
Size of the TCP header in number of 32-bits words
Min value 5 (20 bytes)
Max value 15 (60 bytes)

TCP Segment

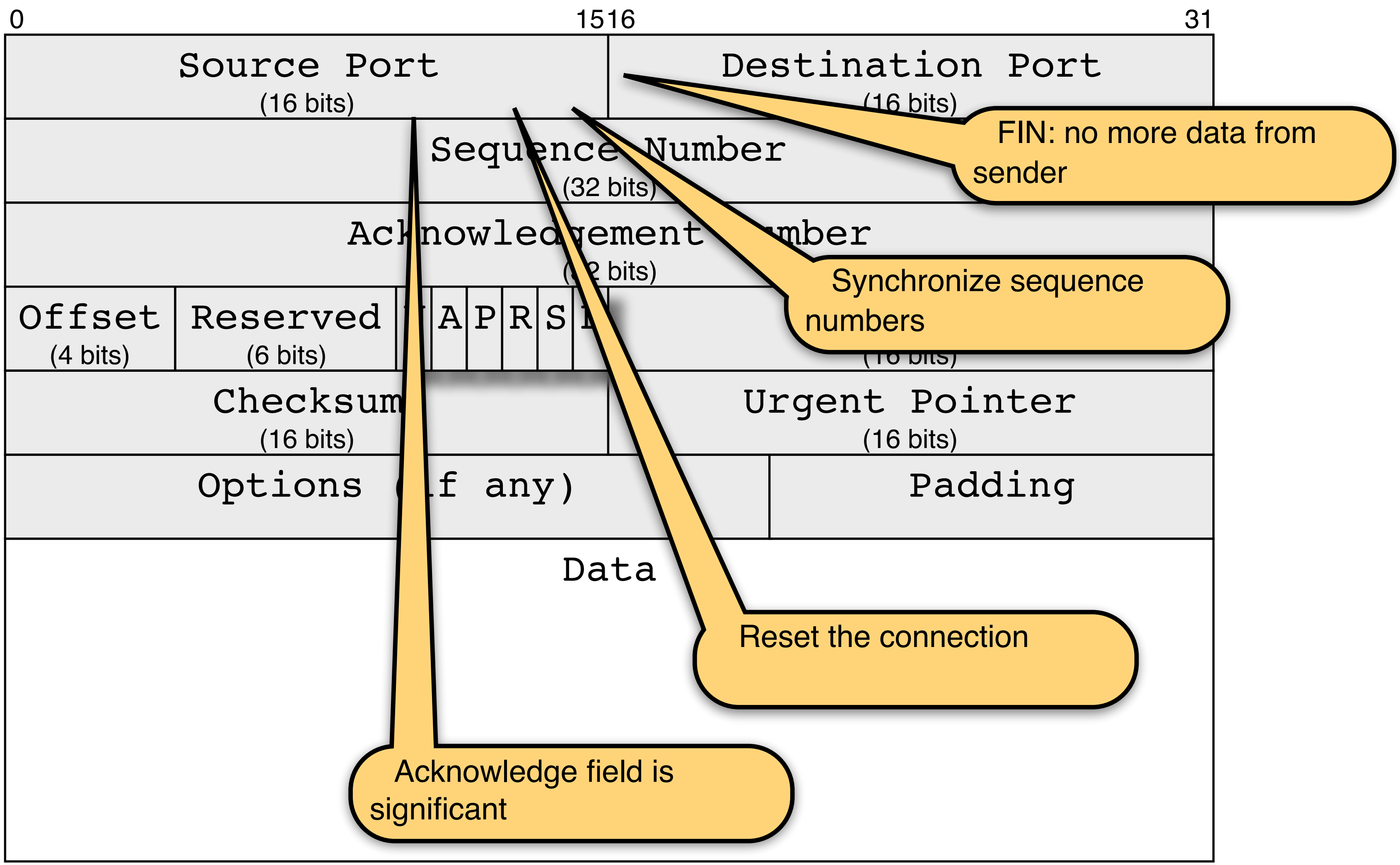


0
15
16
31
Source Port
(16 bits)
Destination Port
(16 bits)
Sequence Number
(32 bits)
Acknowledgement Number
(32 bits)
Offset
(4 bits)
Reserved
(6 bits)
U
A
P
R
S
F
Window
(16 bits)
Checksum
(16 bits)
Urgent Pointer
Receiving window size
number of bytes the receiver is willing to accept at most

TCP Segment


0
15
16
31
Source Port
(16 bits)
Destination Port
(16 bits)
Sequence Number
(32 bits)
Acknowledgement Number
(32 bits)
Offset
(4 bits)
Reserved
(6 bits)
U
A
P
R
S
F
Window
(16 bits)
Checksum
(16 bits)
Urgent Pointer
(16 bits)
Options (if any)
Padding
Error checking code for header and data

TCP Segment



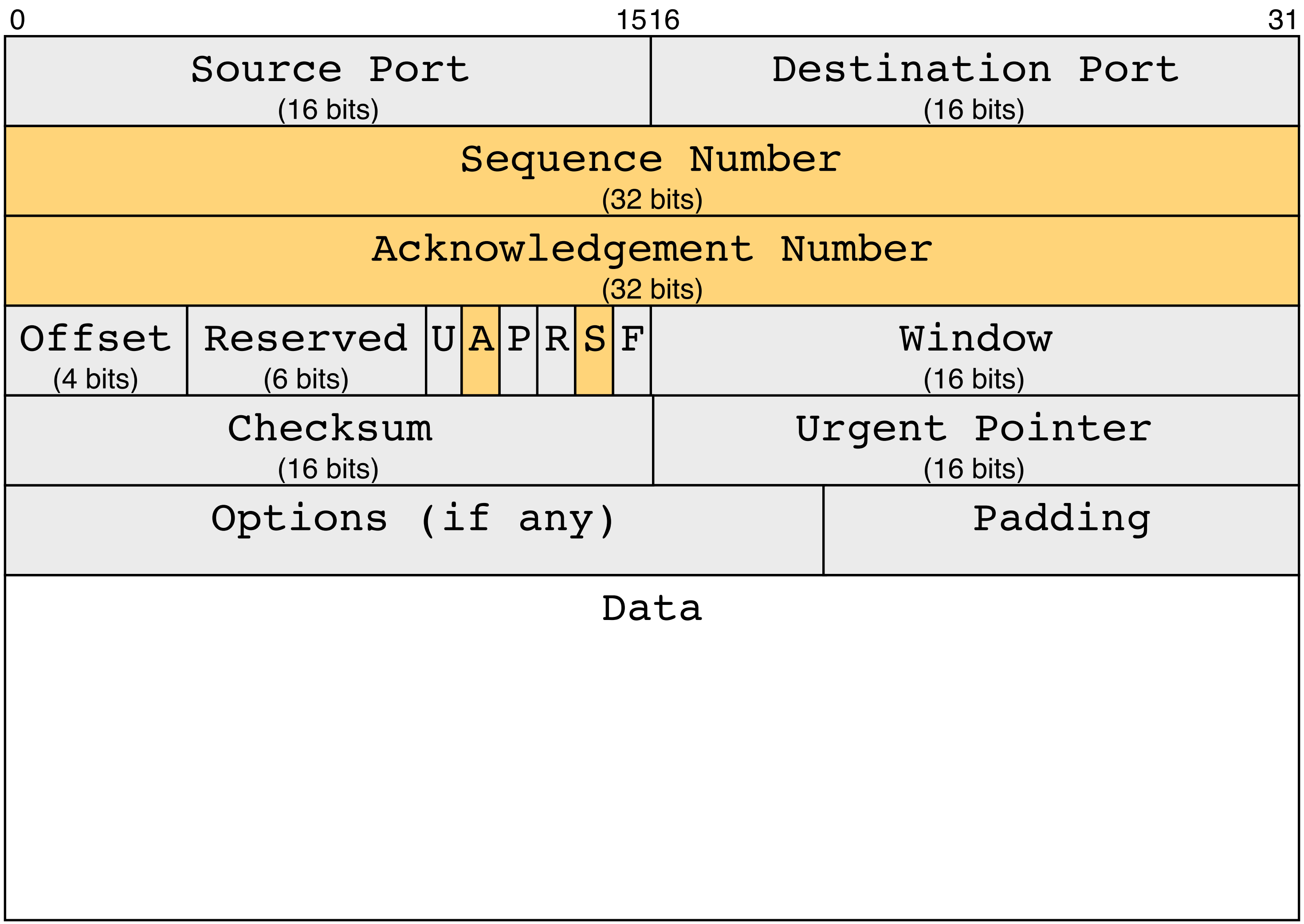
0
1516
31
Source Port
(16 bits)
Destination Port
(16 bits)
(32 bits)
Offset
(4 bits)
Reserved
(6 bits)
A
P
R
S
Checksum
(16 bits)
Urgent Pointer
(16 bits)
Padding
Data
FIN: no more data from sender
Synchronize sequence numbers
Reset the connection
Acknowledge field is significant

TCP Establishment & Termination

(How do you set up a TCP connection? How to cleanly terminate a connection?

TCP Connection Setup

- TCP Header used fields



TCP 3-Way Handshake

Active opener sends first packet

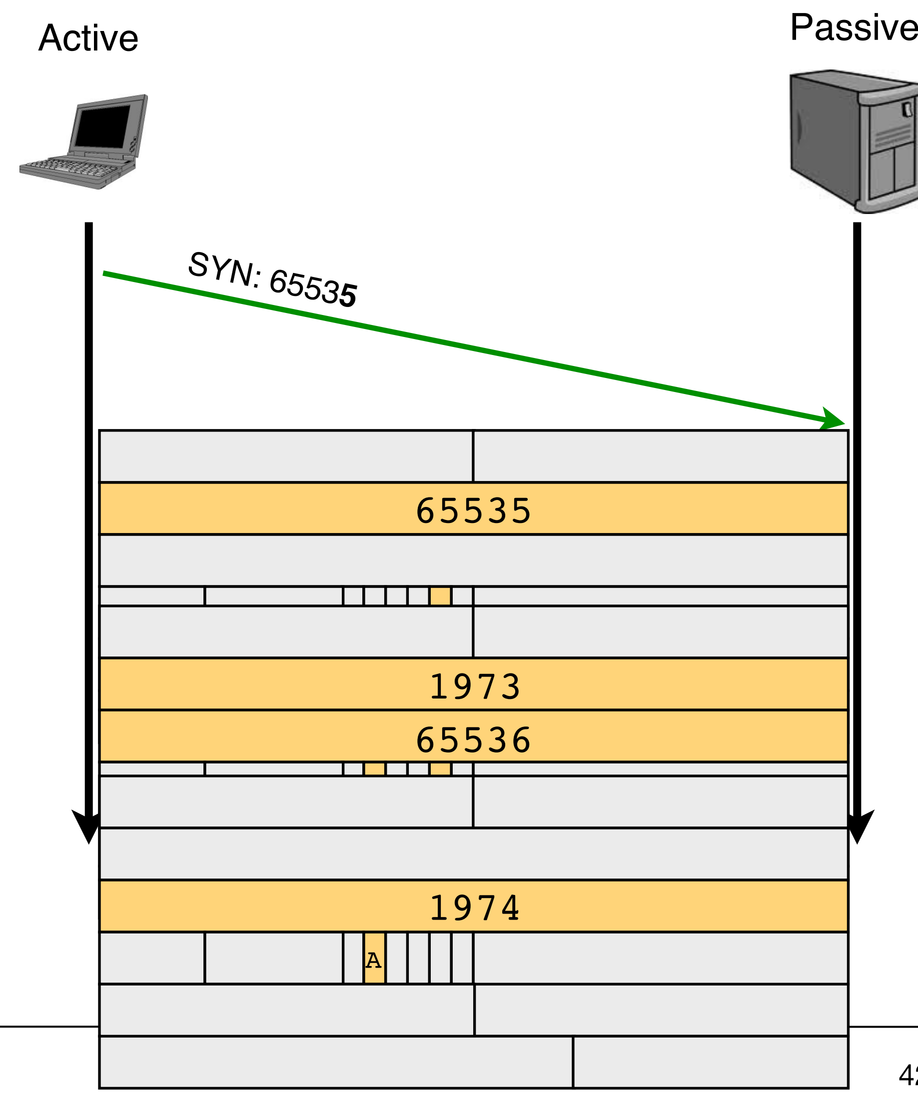
- SYN with initial sequence number

Passive opener replies

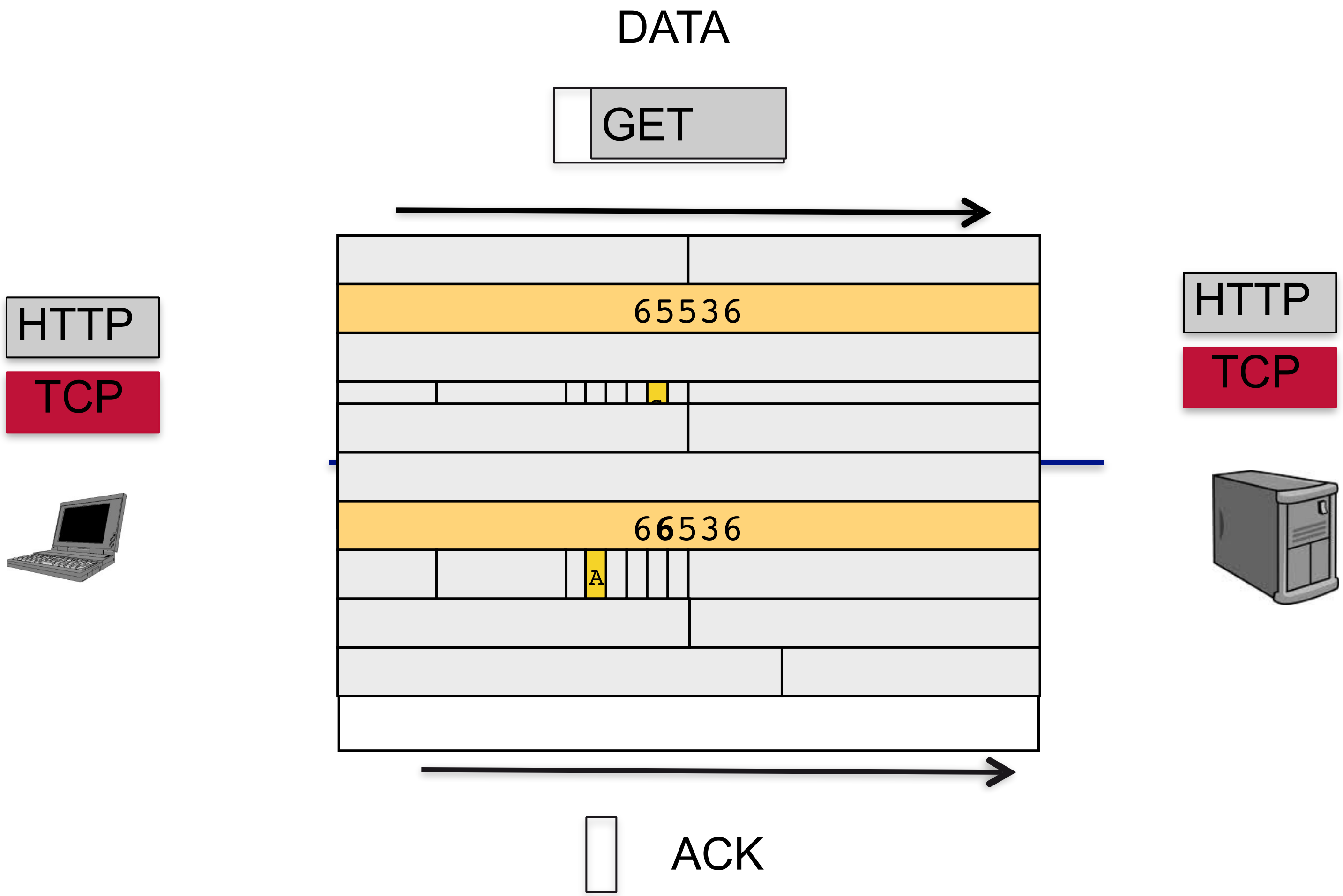
- SYN with initial sequence number
- Ack to active opener initial sequence number

Active opener replies

- Ack to passive opener initial sequence number

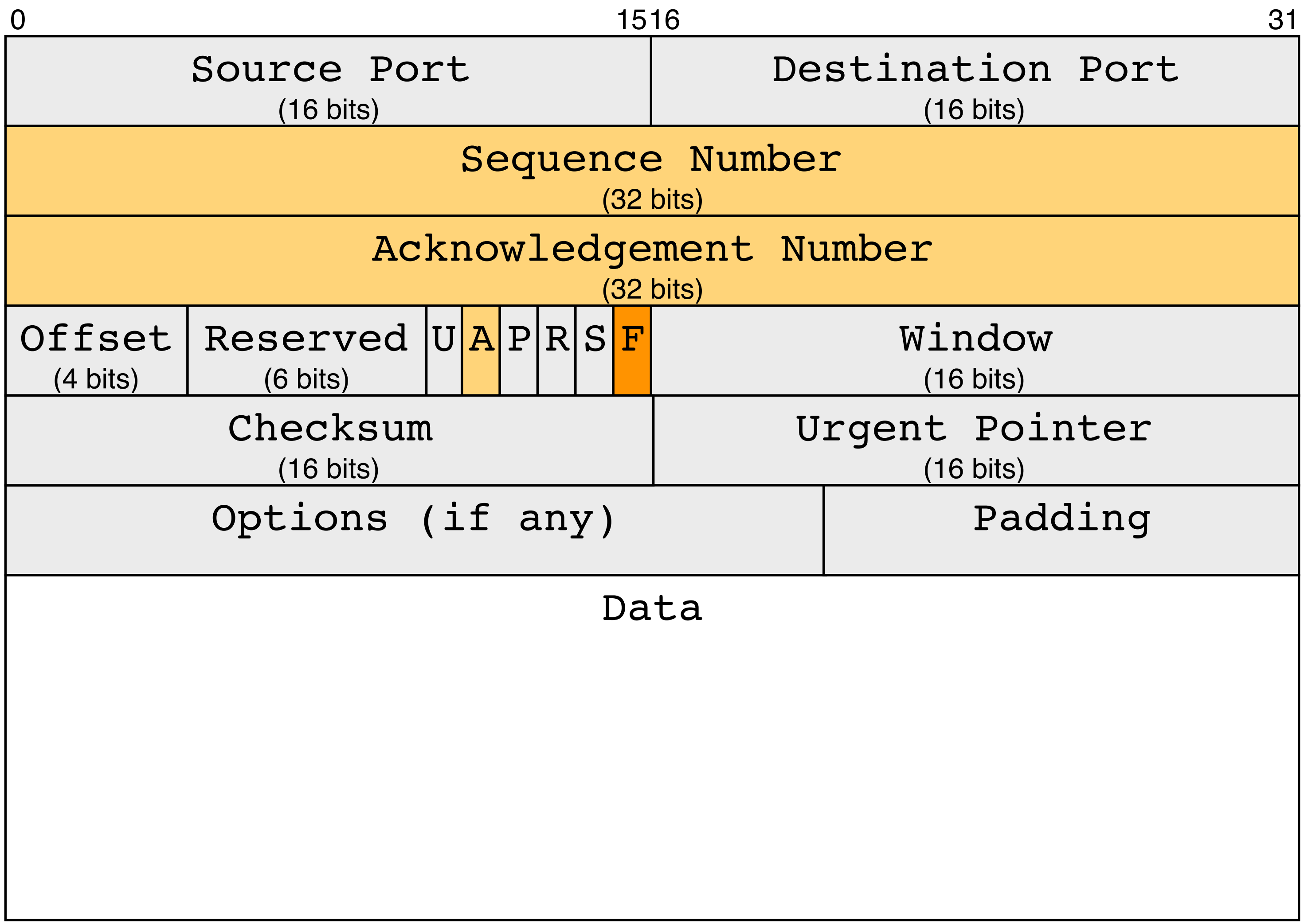


Data & Acks

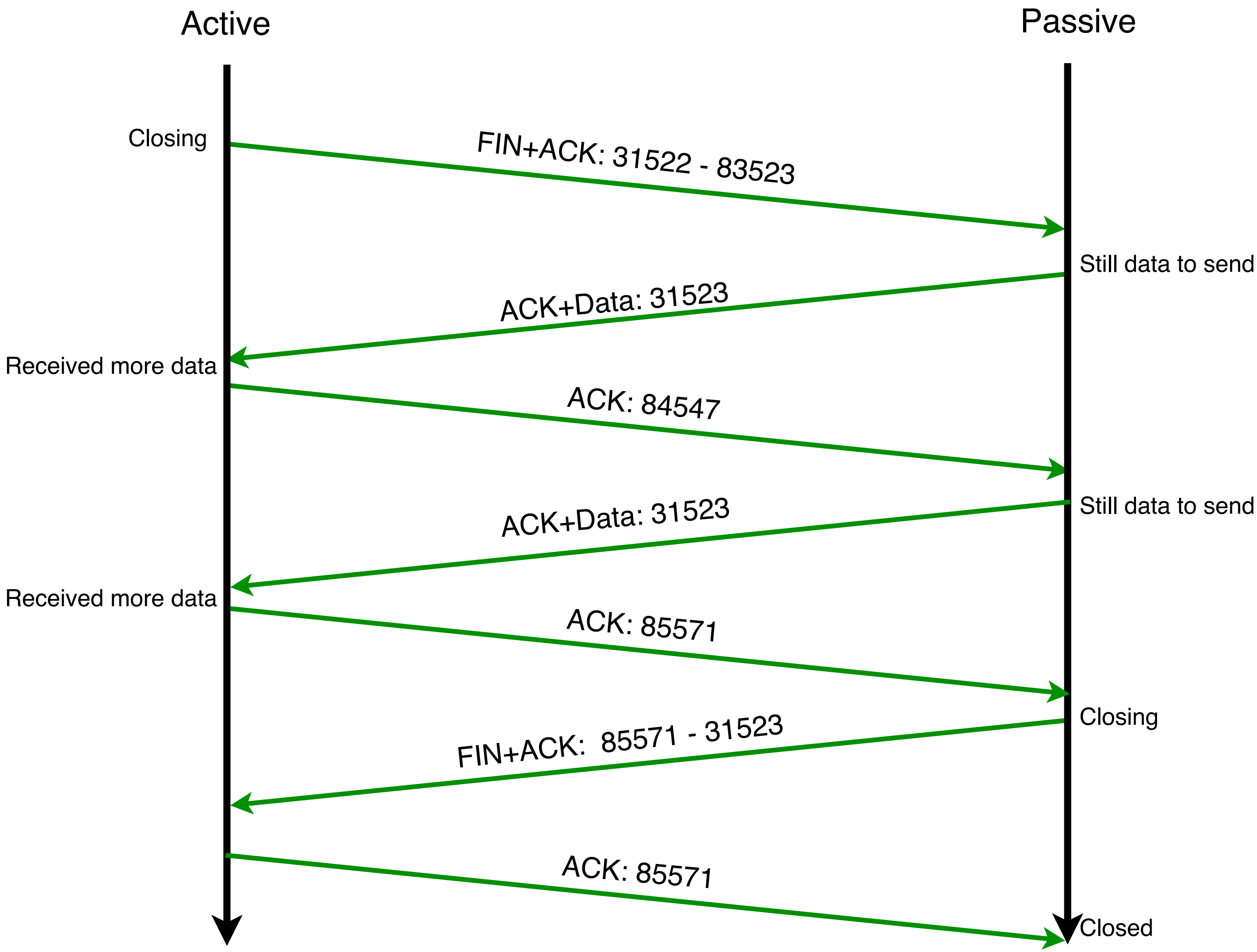


TCP Connection Termination

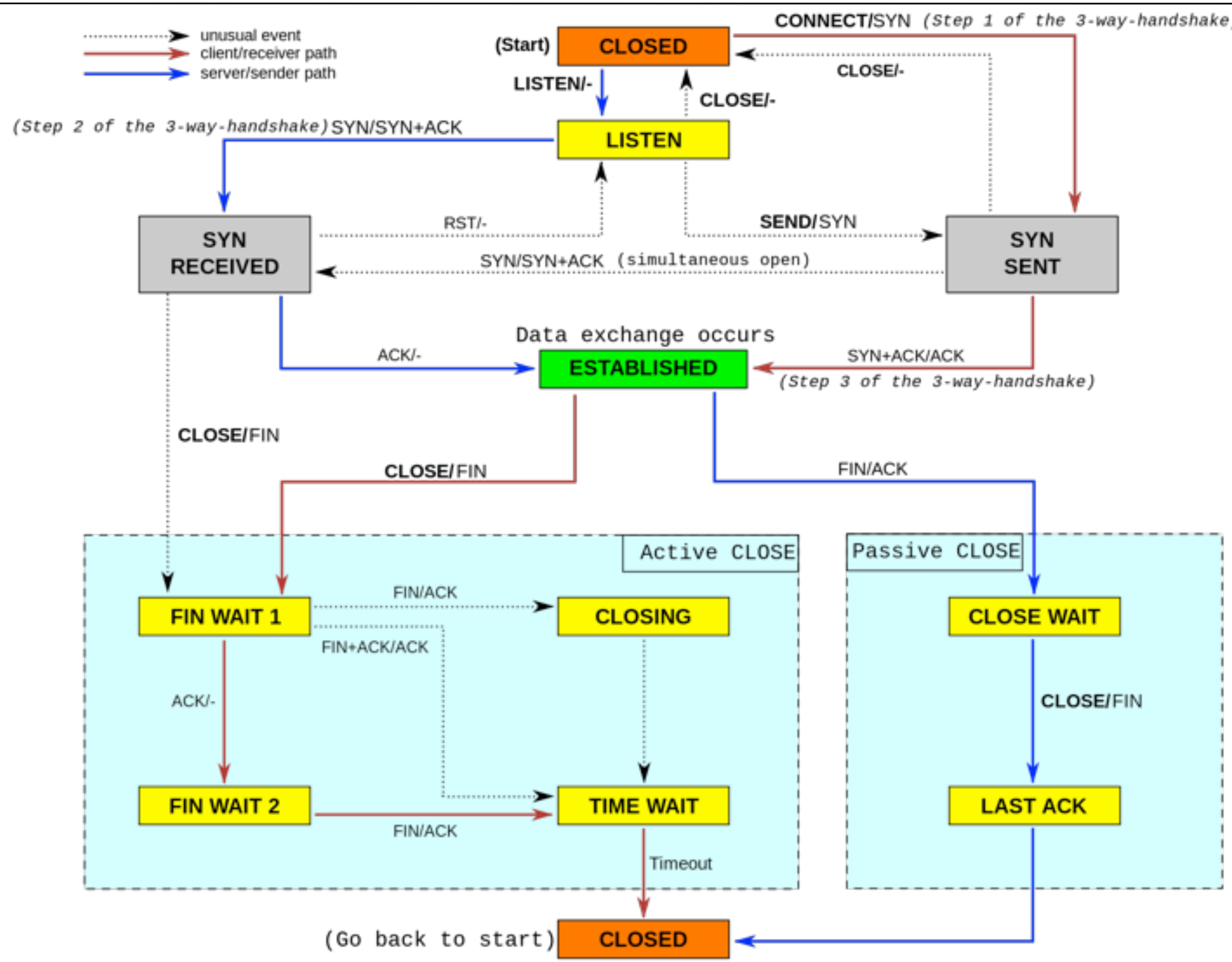
TCP Header used fields



TCP Connection Termination



TCP State Diagram



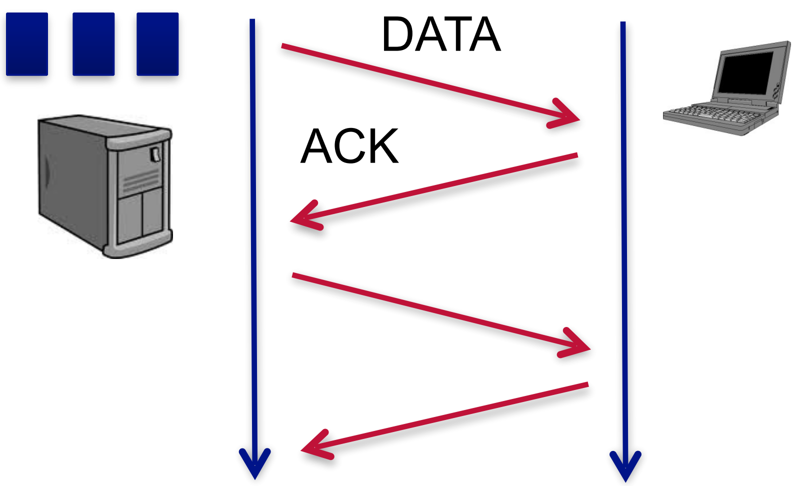
Source: http://en.wikipedia.org/wiki/File:Tcp_state_diagram_fixed.svg

TCP Traffic Control

(How much bytes to transmit)

Stop-and-Wait Algorithm

- Principle
  - Send a packet only if the previous one is acknowledged
  - Retransmission occurs only on timeout
    - (Automatic Repeat reQuest – ARQ)

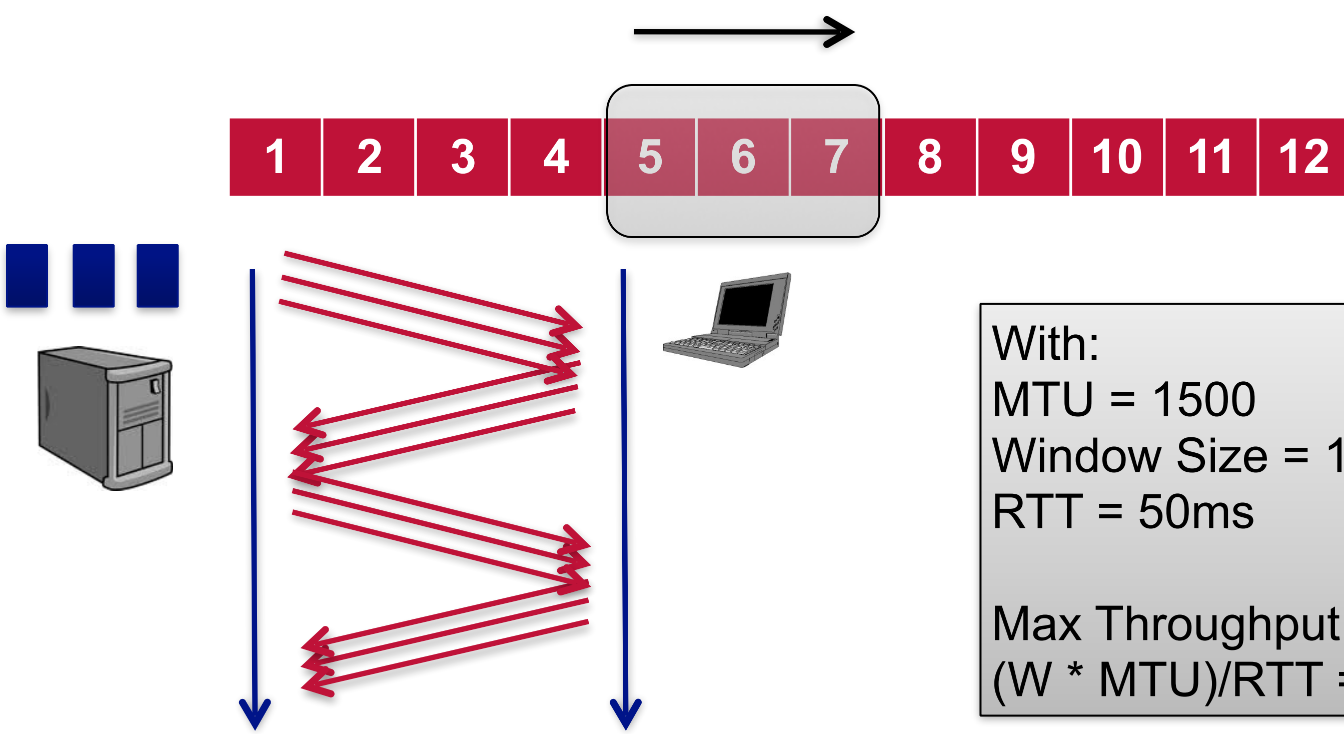


With:
MTU = 1500 Octets
RTT = 50ms

Max Throughput:
MTU / RTT = **240 kbit/s**

Sliding Window

- Principle
  - Apply stop-and-wait for a set of packets
  - Window size (# Packets) decided by the destination

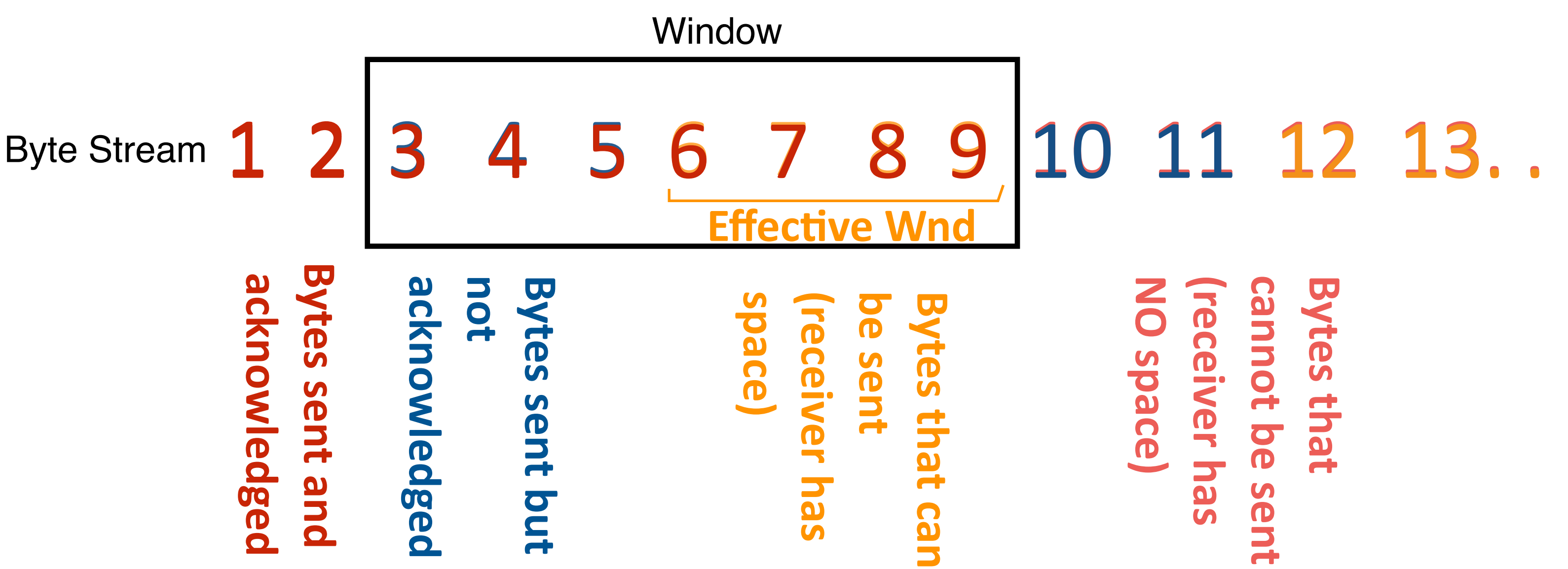


With:
MTU = 1500
Window Size = 10 Pkts
RTT = 50ms

Max Throughput:
(W * MTU)/RTT = **2.4 Mbit/s**

Sliding Window

- Window of size N allows to send N bytes split in more than one segment without acknowledgement



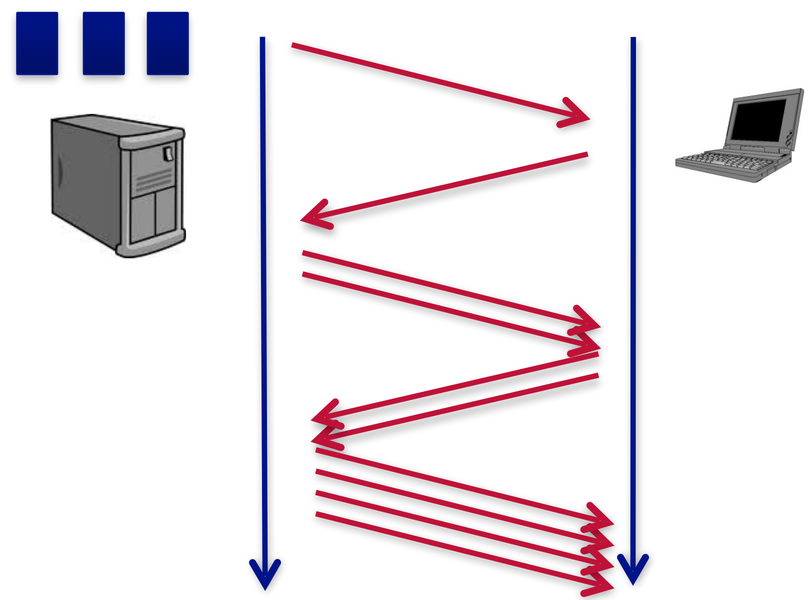
- Transmission stops when effective window = 0   (closed window)

Variable Window Size

- Principle
  - Adaptive window size
  - Most used Algorithm:
    - AIMD - Additive Increase Multiplicative Decrease

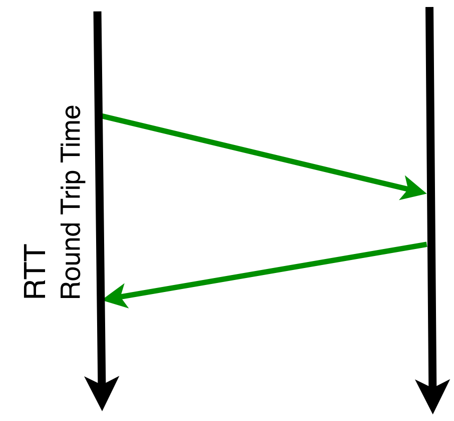
```
if (cwnd < ssth)
    each ACK cwnd+=1
else
    each ACK cwnd+=1/cwnd
```

```
each LOSS
    ssth=cwnd/2
```



RTO - Retransmission TimeOut

- **At connection setup:**
  - 3 seconds



- **Parameters**
  - RTT: new RTT measured on successful transmissions
  - SRTT: Smoothed RTT
    - Exponential Weighted Moving Average
  - RTTVAR: RTT variance
  - K =4
  - alpha = 1/8
  - beta = 1/4

- **Algorithm**
  - SRTT = (1 - alpha) x SRTT + alpha x RTT
  - RTTVAR = (1 - beta) x RTTVAR + beta x (RTT - SRTT)
  -

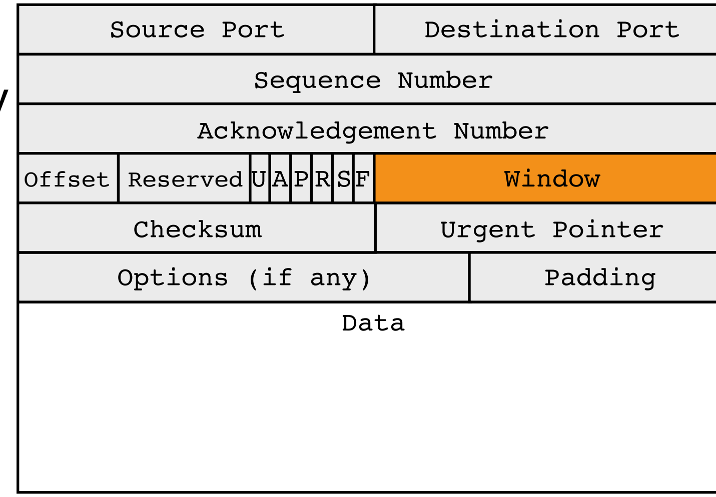
- **RTO**
  - RTO = min(1sec, [SRTT + K x RTTVAR])

TCP Flow Control

(Resource polling in the Internet)

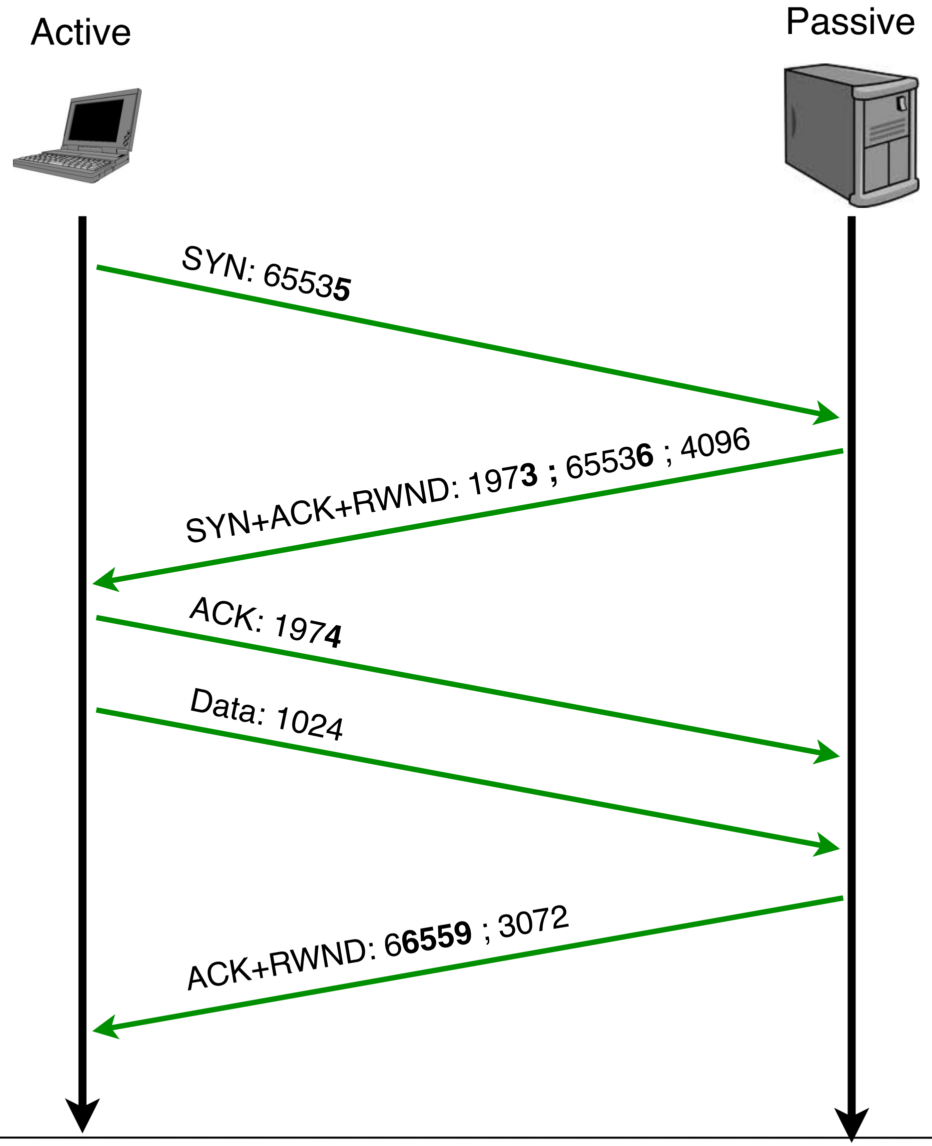
Flow Control

- Goals
  - Sender does not overflow receiver's memory
  - Maximizing sending rate



- Principle
  - sliding window controlling how much data can be sent
  - counting unit is byte (not segments)
  - each acknowledgement move the window forward for as many byte that are acknowledged
  - the receiver communicate the size in the "window size" field of the TCP header
    - receiver window size => RWND
  - **RWND** changes dynamically

Using RWND

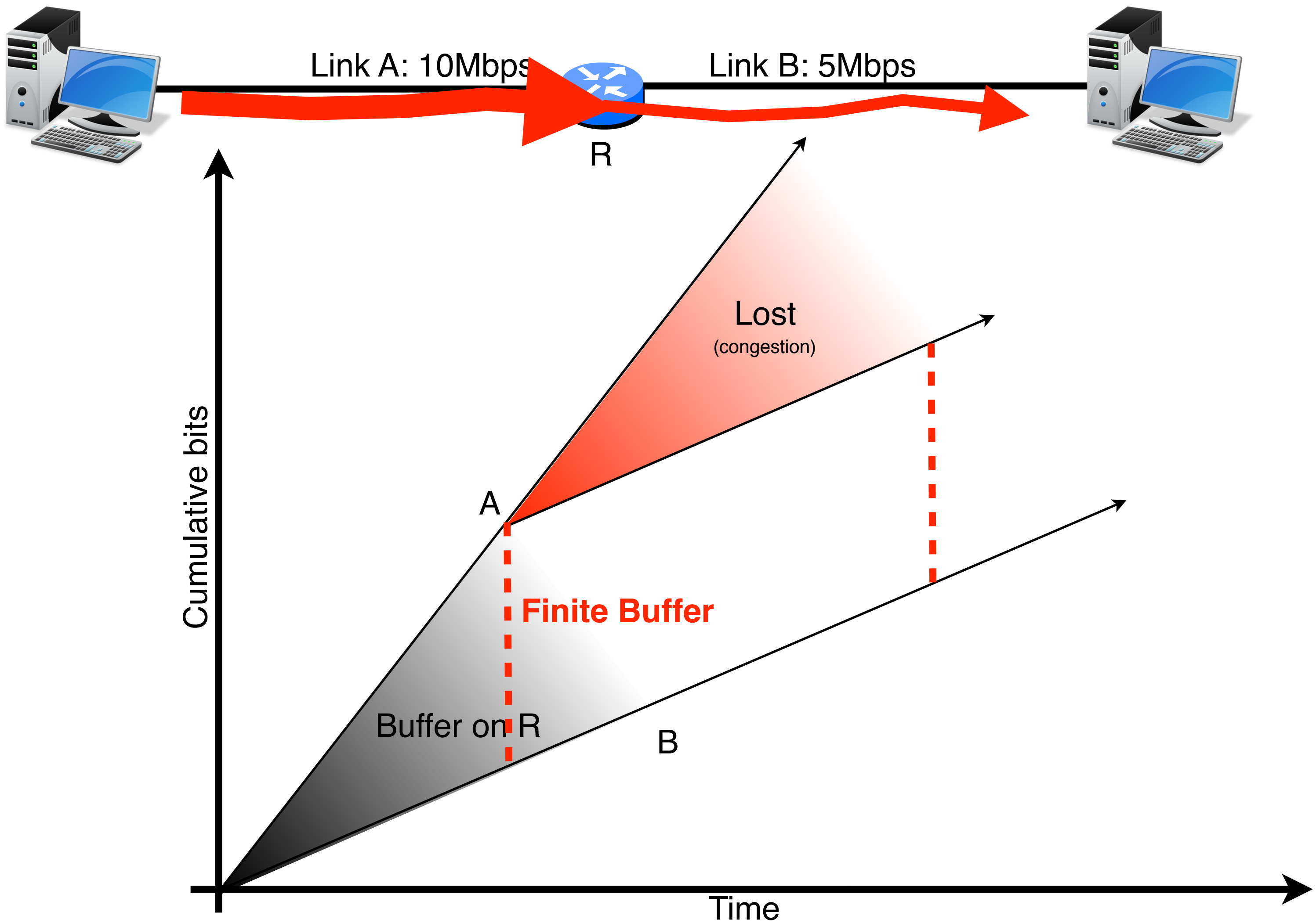


Active
Passive
SYN: 65535
SYN+ACK+RWND: 1973 ; 65536 ; 4096
ACK: 1974
Data: 1024
ACK+RWND: 66559 ; 3072

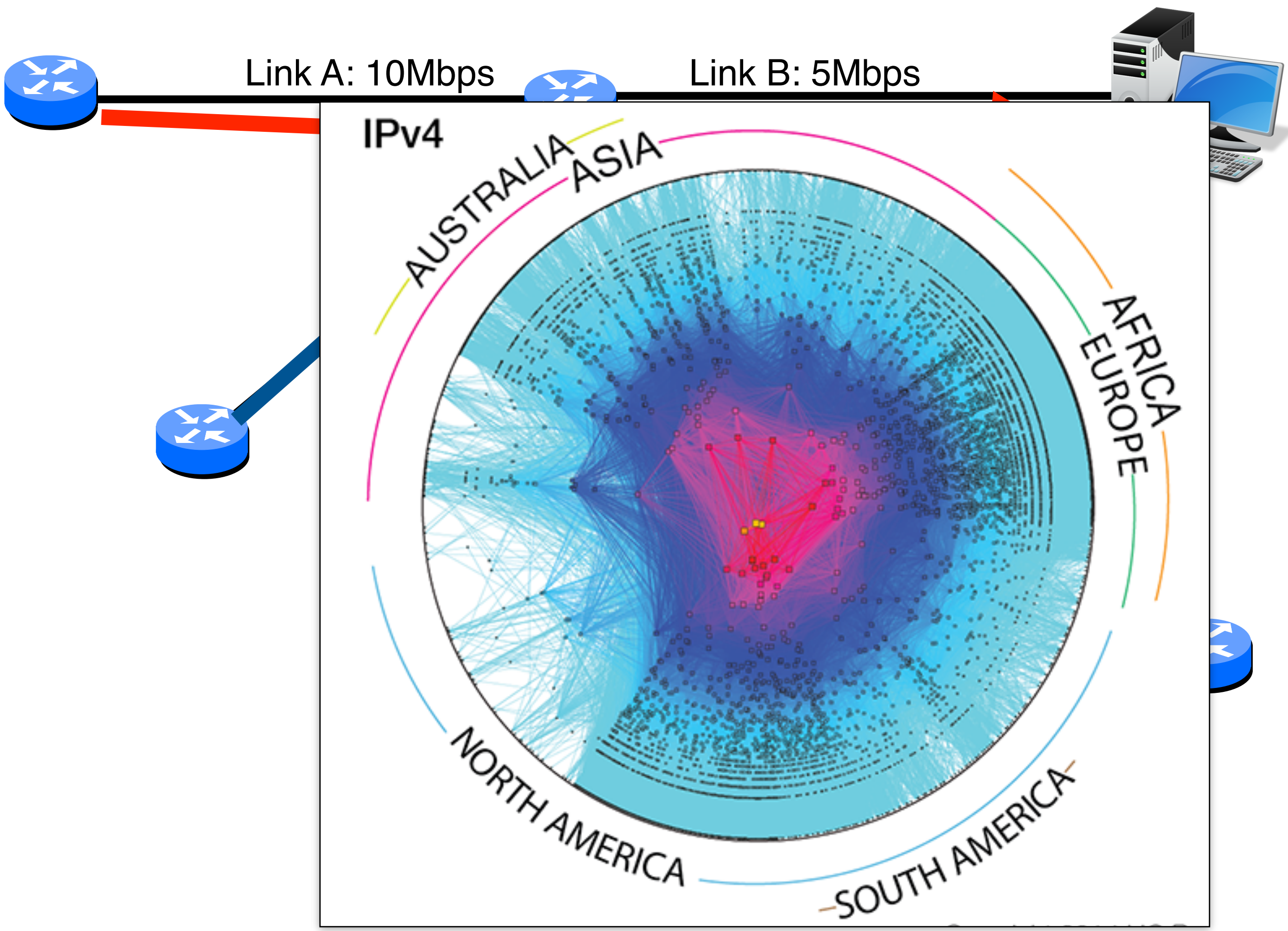
TCP Congestion Control

(Fair share of Internet resources)

What is congestion? The Simple Scenario



What is congestion? The Complex Scenario



https://www.caida.org/research/topology/as_core_network/pics/2014/ascore-2014-jan-ipv4v6-standalone-1200x710.png

TCP Congestion Control principle

- Control the congestion by varying the number of outstanding bytes in the network by adapting the window size

- Window Size = min{Advertised Window, Congestion Window}

Receiver (RWND) Transmitter (CWND)

- What is the size of CWND and how is it dynamically updated?

Congestion Window

- Congestion Window Algorithm:

  – AIMD: Additive Increase Multiplicative Decrease

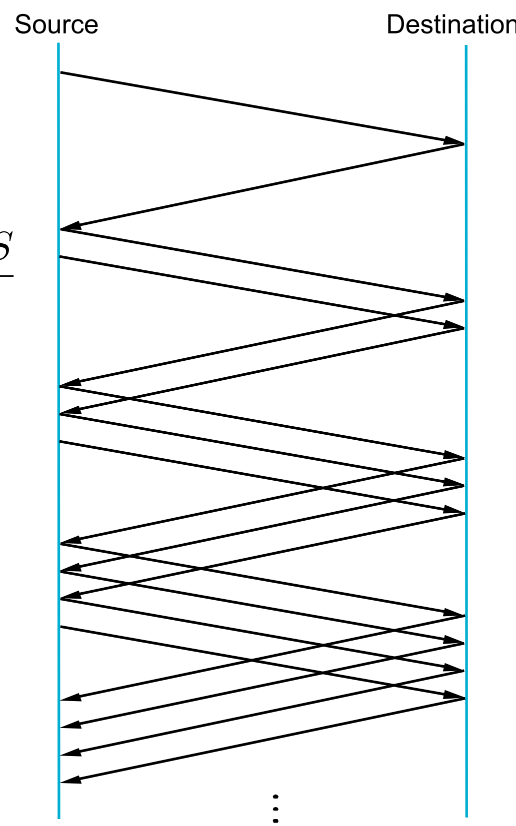
    - If segment acknowledged: $CWND = CWND + \frac{MSS \cdot MSS}{CWND}$

    - If segment is dropped: $CWND = \frac{CWND}{2}$

AIMD In Practice

- For each CWND octets acknowledged increase the CWND by 1 MSS
- For each lost segment half the CWND

- In practice CWND augmented by fractions of MSS
  - If segment acknowledged: $CWND = CWND + \frac{MSS \cdot MSS}{CWND}$
  - If segment is dropped: $CWND = \frac{CWND}{2}$
  - Obtained sending rate R = CWND/RTT



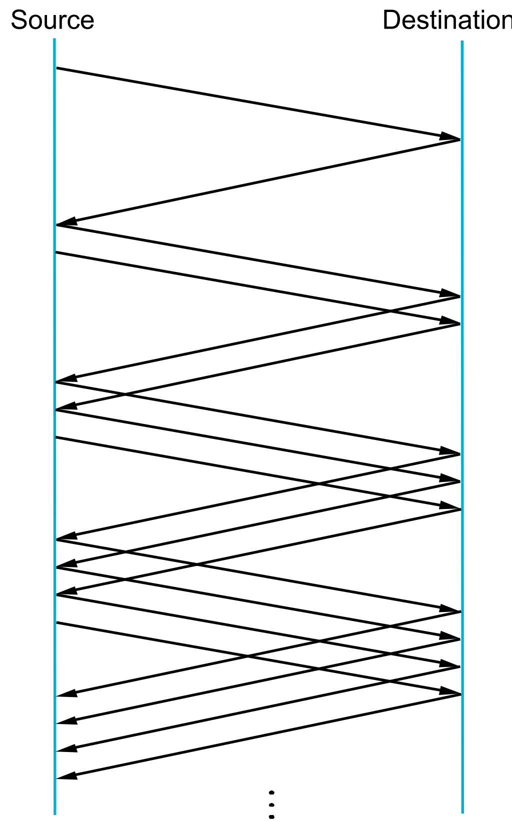
Congestion Window Increase

CWND = 1 MSS = 1024

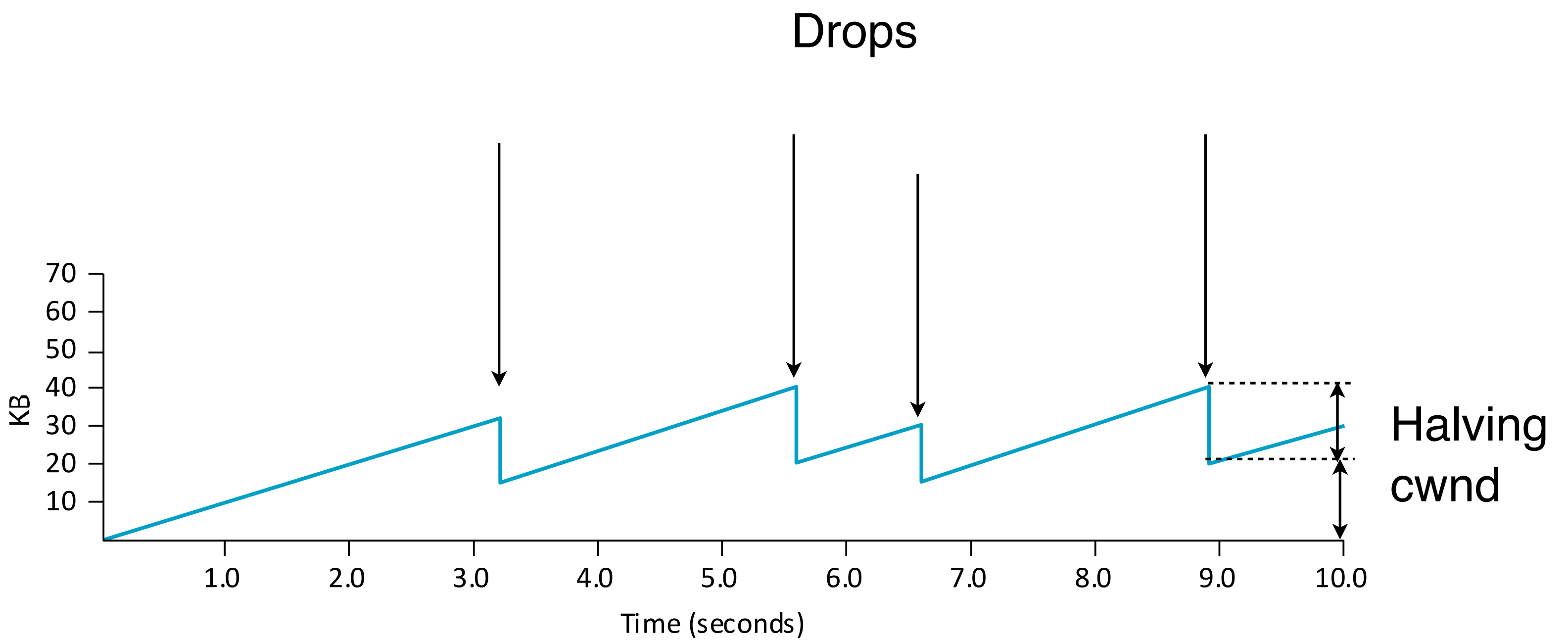
CWND = 1024 + (1024*1024)/1024 = 2048

CWND = 2048 + (1024*1024)/2048 + (1024*1024)/2048 = 3072

CWND = 3072+ (1024*1024)/3072 + (1024*1024)/3072 + (1024*1024)/3072 = 4096



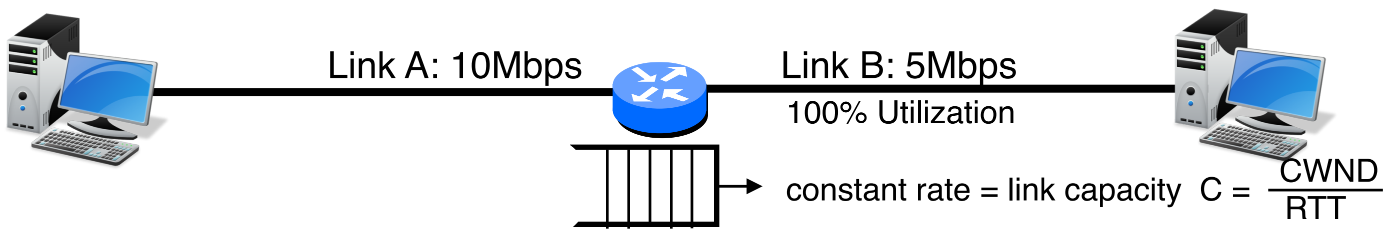
Congestion Window Evolution

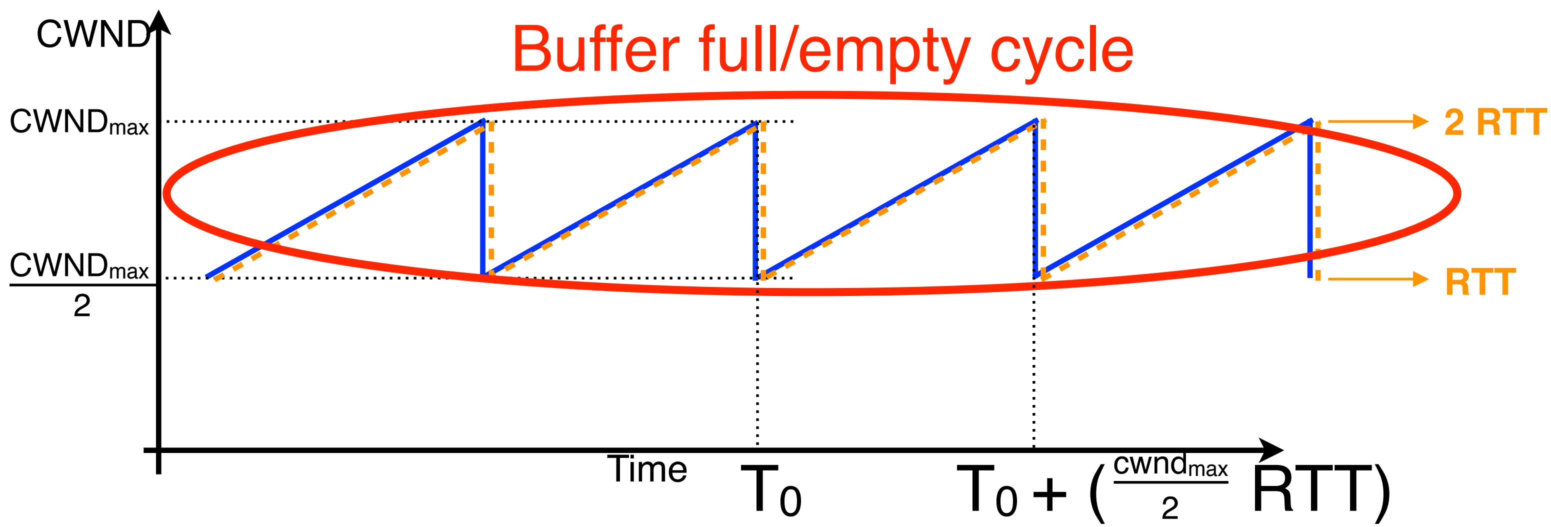


TCP Sawtooth pattern

Animations http://guido.appenzeller.net/anims/

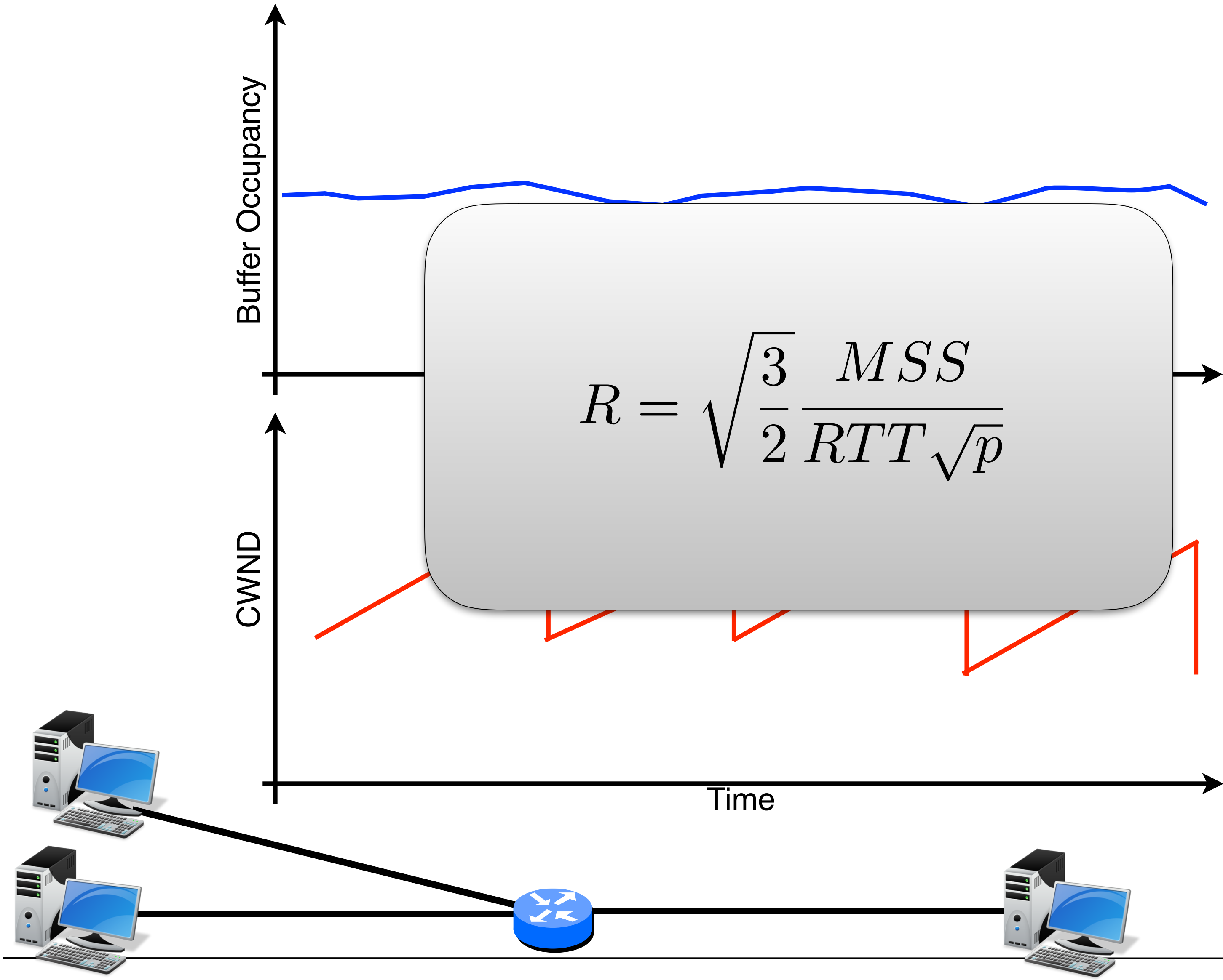
Single Flow Throughput





Needed Buffer Size = C x RTT

Multiple Flows



$$R = \sqrt{\frac{3}{2}}\frac{MSS}{RTT\sqrt{p}}$$

$$R = \frac{CWND}{RTT} \propto CWND$$

TCP Variants

(How TCP adapts to a changing Internet)

TCP Evolution

- 1974: 3-way handshake
- 1978: TCP and IP split into TCP/IP
- 1983: January 1, ARPAnet switches to TCP/IP
- 1986: Internet begins to suffer congestion collapse
- 1987-8: Van Jacobson fixes TCP, publishes seminal TCP paper (Tahoe)
- 1990: Fast recovery and fast retransmit added (Reno)
- 1993: Early congestion detection (Vegas)
- 1996: Modified Fast recovery for multiple losses (NewReno)
- 1996: Selective Ack (TCP SACK)
- 2008: CUBIC TCP
- 2012: Multipath TCP

- Other Variants
  - Compound TCP
  - TCP Hybla
  - FAST TCP
  - H-TCP
  - Data Center TCP
  - High Speed TCP
  - HSTCP-LP
  - TCP-Illinois
  - TCP-Ghergani
  - TCP-LP
  - Scalable TCP
  - TCP Veno
  - Westwood
  - Westwood+
  - XCP
  - YeAH-TCP
  - TCP-FI

Source: https://en.wikipedia.org/wiki/TCP_congestion-avoidance_algorithm

1986 - The Congestion Collapse

- **Problem:**
  - Early TCP implementations had very bad retransmission behavior.
  - Upon a loss TCP sender re-transmitted the lost segment without adapting the rate of new segments.
- **Result:**
  - Increased (instead that avoided) congestion
  - Entire network fall into a steady state where most packets are lost and throughput is negligible.

- **Identified as a problem in 1984** (RFC 896)

- **Observed in October 1986**
  - NSFnet phase-I backbone dropped its capacity of 32 **k**bit/s to 40 bit/s

TCP before Tahoe

- Flow control only on receiver side
  - “window size”
- “Fast” start
  - At TCP connection establishment send a full “window size” of byte
-
- Retransmission
  - Start a timeout on every single packet
- Note: window size can be larger than what the network can support.

TCP Tahoe

(How to not overflow the network)

TCP Tahoe main features

- **Congestion Avoidance**
  - Congestion window based on AIMD Algorithm
    - (Already seen)
- **Slow Start**
  - “Slower” approach to start TCP connections
- **Fast Retransmit**
  - Retransmission not based (solely) on timeouts

Slow Start

Rationale:

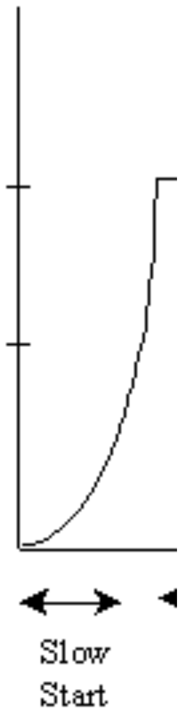
- Linear additive increase takes too long to ramp up a new TCP connection from cold start.
- Windows size start may cause early congestion

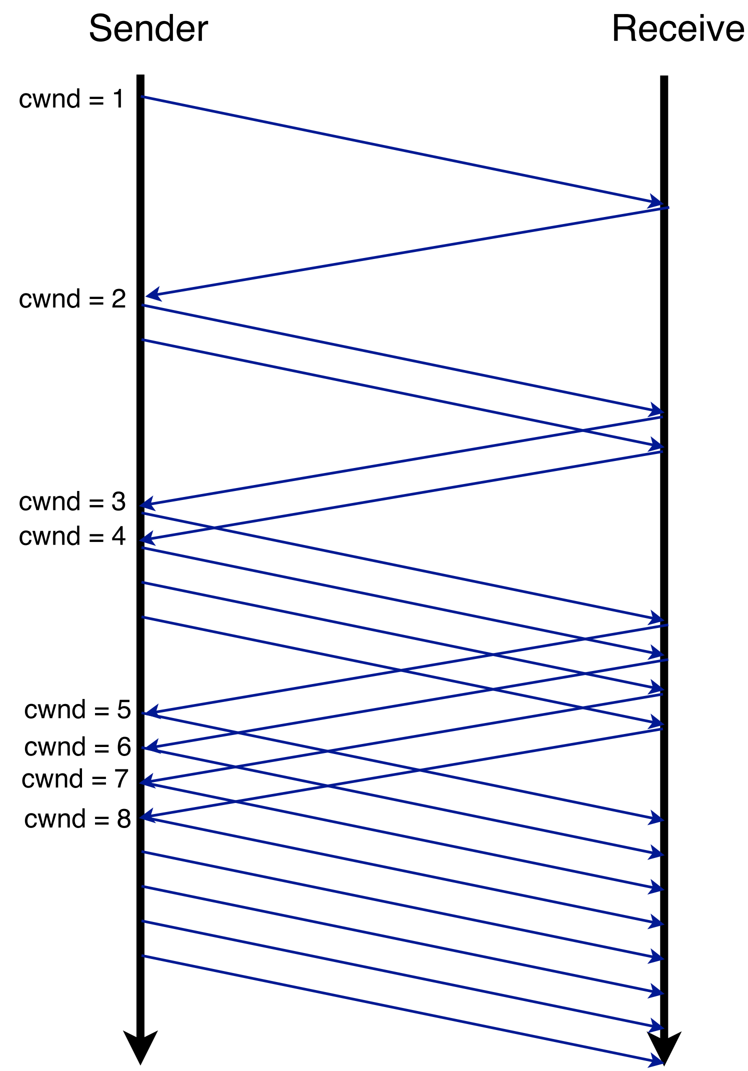
Principle:

- Initial exponential increase in the size of cwnd.
- “Slower” than a full windows size start (hence the name)
- Faster than linear increase

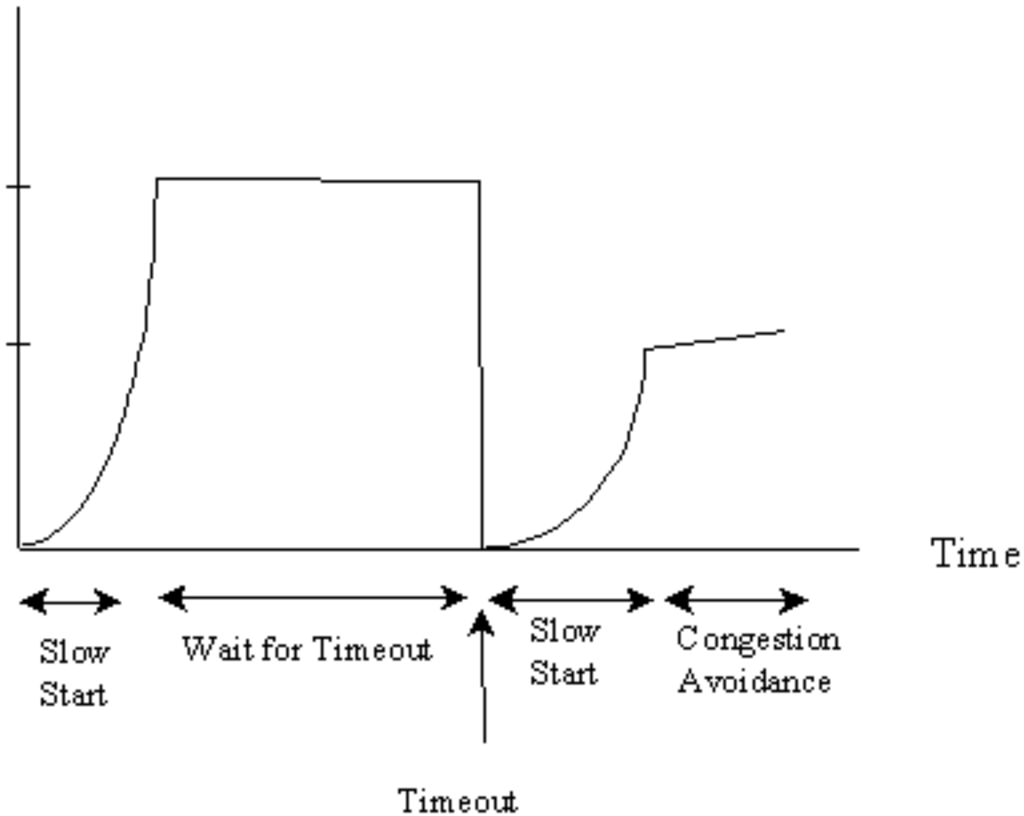
- **Mechanism:**
  - Start with CWND = 1.
  - For every ACK, CWND is incremented.
    - Results in doubled CWND every RTT
  - Applied:
    - Cold start (TCP connection establishment)
    - When advertised window is zero

Slow Start Example





From Slow Start to Collision Avoidance



When to stop slow start:

At packet loss fix ssthresh (slow start threshold) = CWND/2

restart slow start until CWND = sstresh than collision avoidance

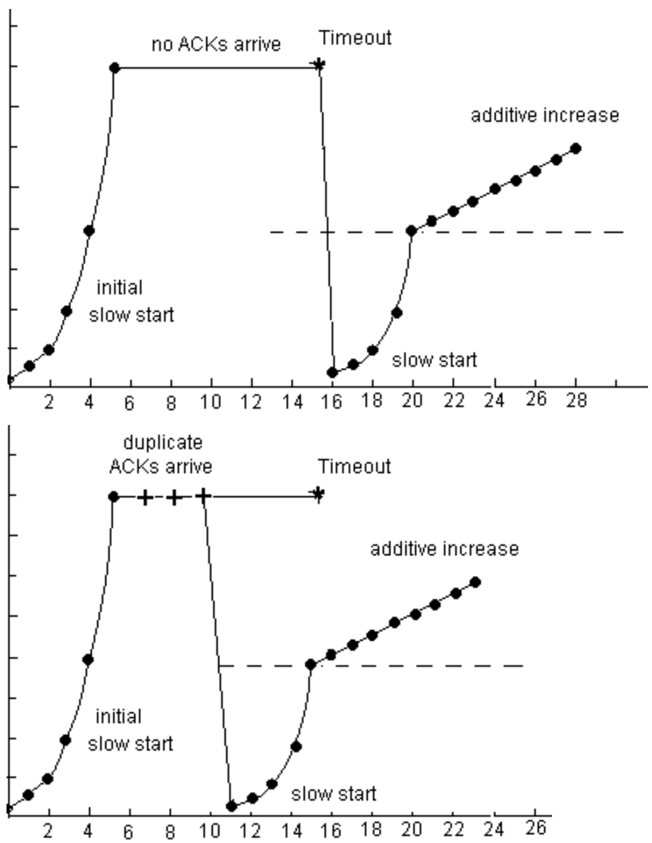
Fast Retransmit

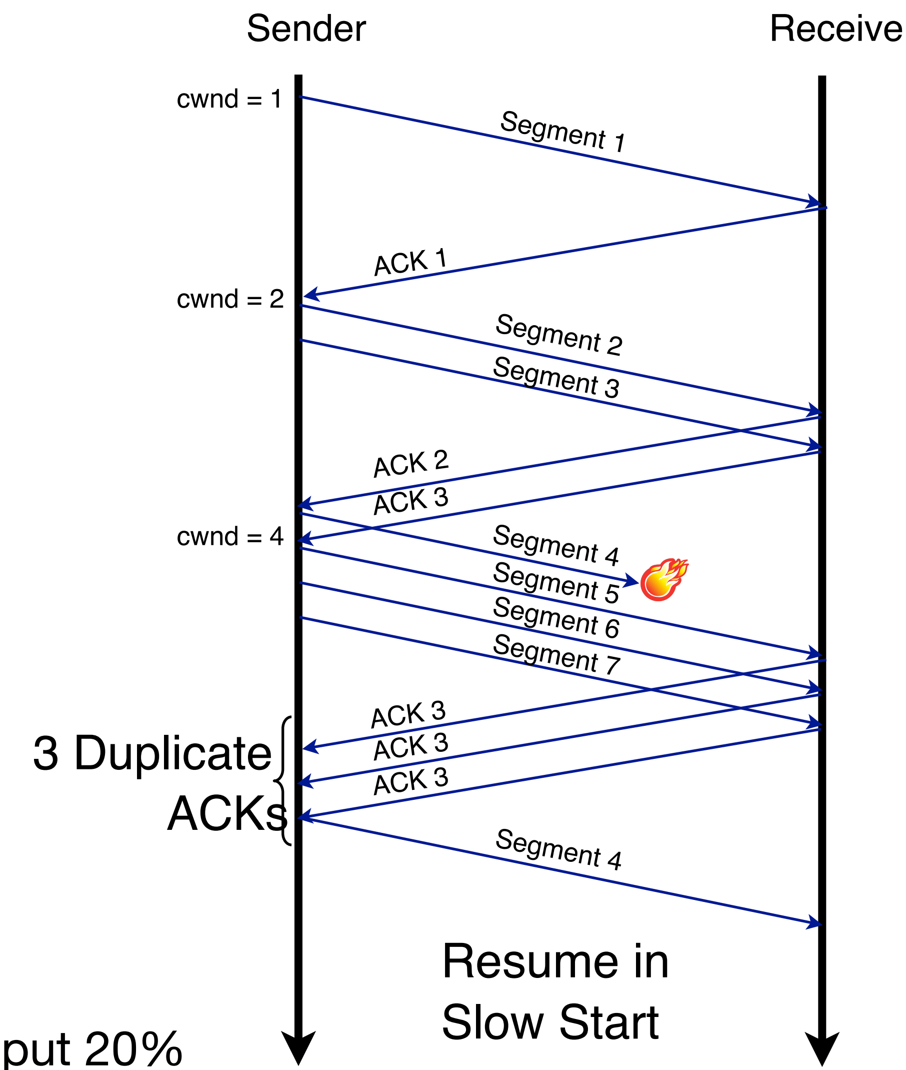
- **Rationale:**
  - Waiting timeouts may be waste of time
  - Receiver responds to every packet with sender seeing duplicate ACKs

- **Principle:**
  - Use duplicate ACKs to signal lost segments

- **Mechanism:**
  - Upon three duplicates ACKs retransmit lost segment
    - Note 3 DUP ACKs = 4 ACKs with same sequence number

Fast Retransmit Example





- Fast retransmit can increase throughput 20%

TCP Reno

(Can segment loss be recovered efficiently???)

TCP Reno main features

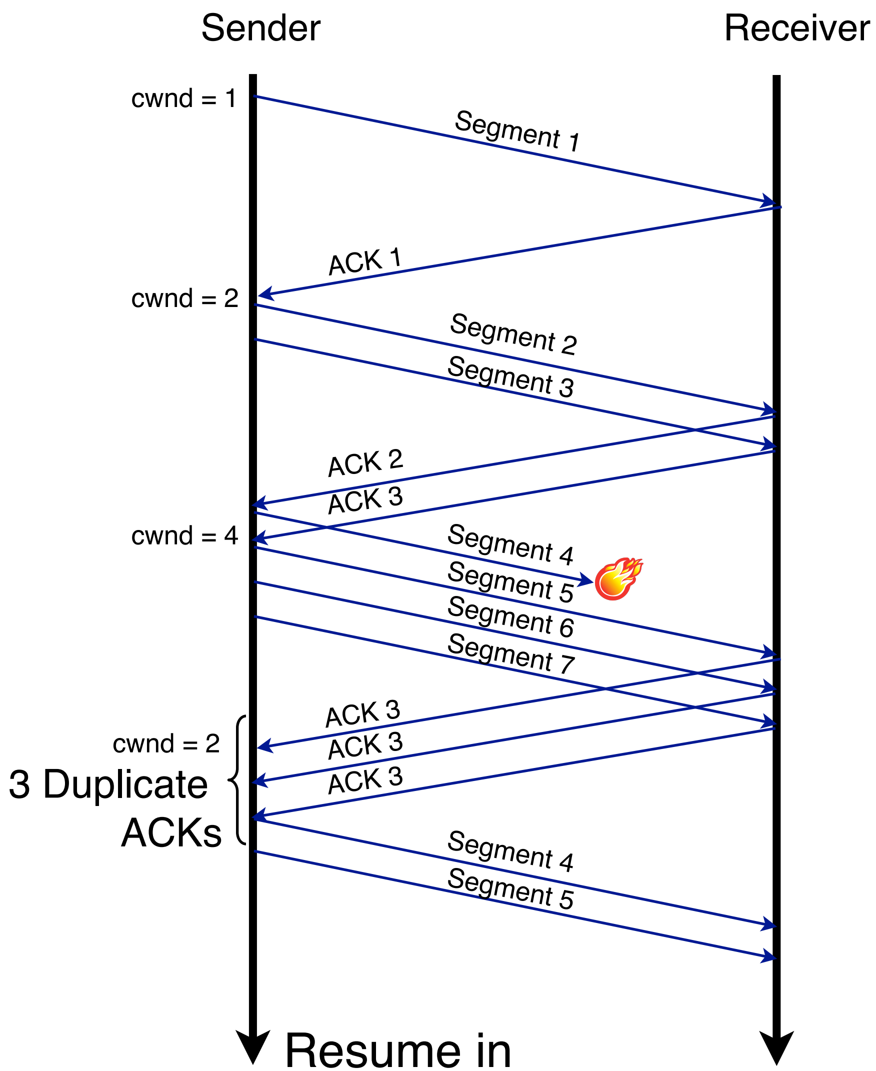
- **TCP Tahoe**
  - Congestion Avoidance
    - Congestion window based on AIMD Algorithm
  - Slow Start
    - “Slower” approach to start TCP connections
  - Fast Retransmit
    - Retransmission not based (solely) on timeouts

- **+ Fast Recovery**
  - Faster recovery maintaining collision avoidance state

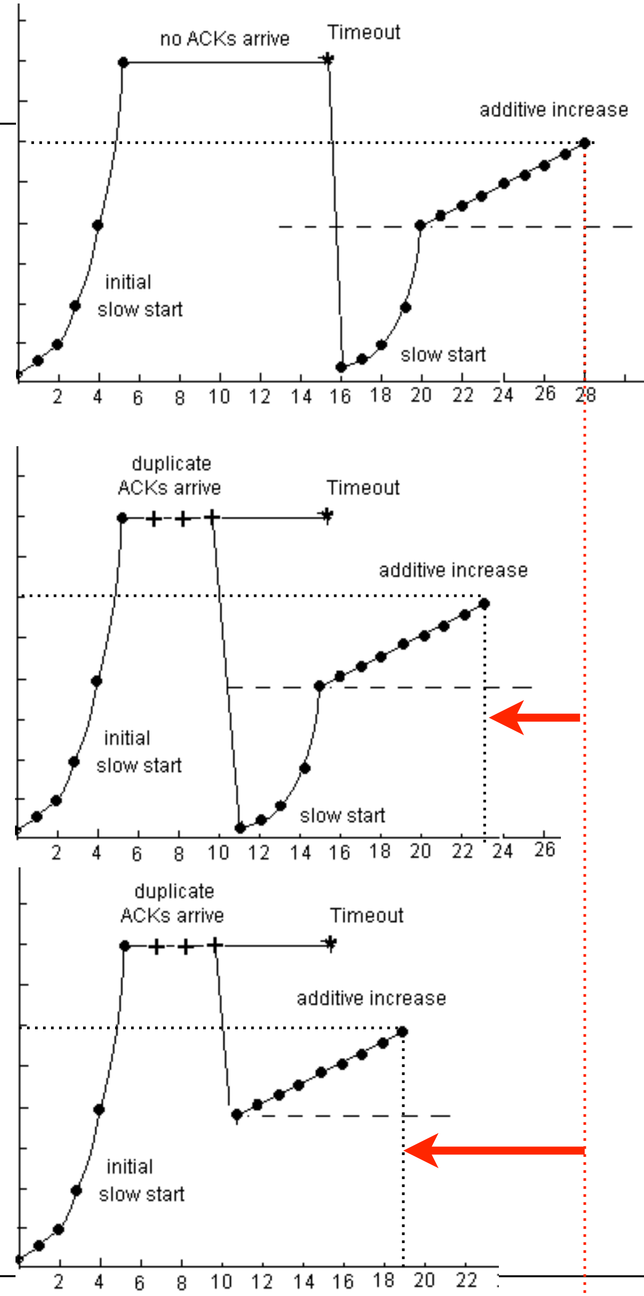
Fast Recovery

- **Rationale:**
  - Receiving duplicate ACKs indicates
    - that the receiver still receives segments from the sender
    - that the sender can receive ACKs
    - that congestion may be a temporary situation in the network
- **Principle:**
  - After a fast retransmit avoid falling back to slow start
- **Mechanism:**
  - After receiving three duplicate ACK start recovery from congestion avoidance state
  - Use ACKs to pace the sending packets

Fast Recovery Example



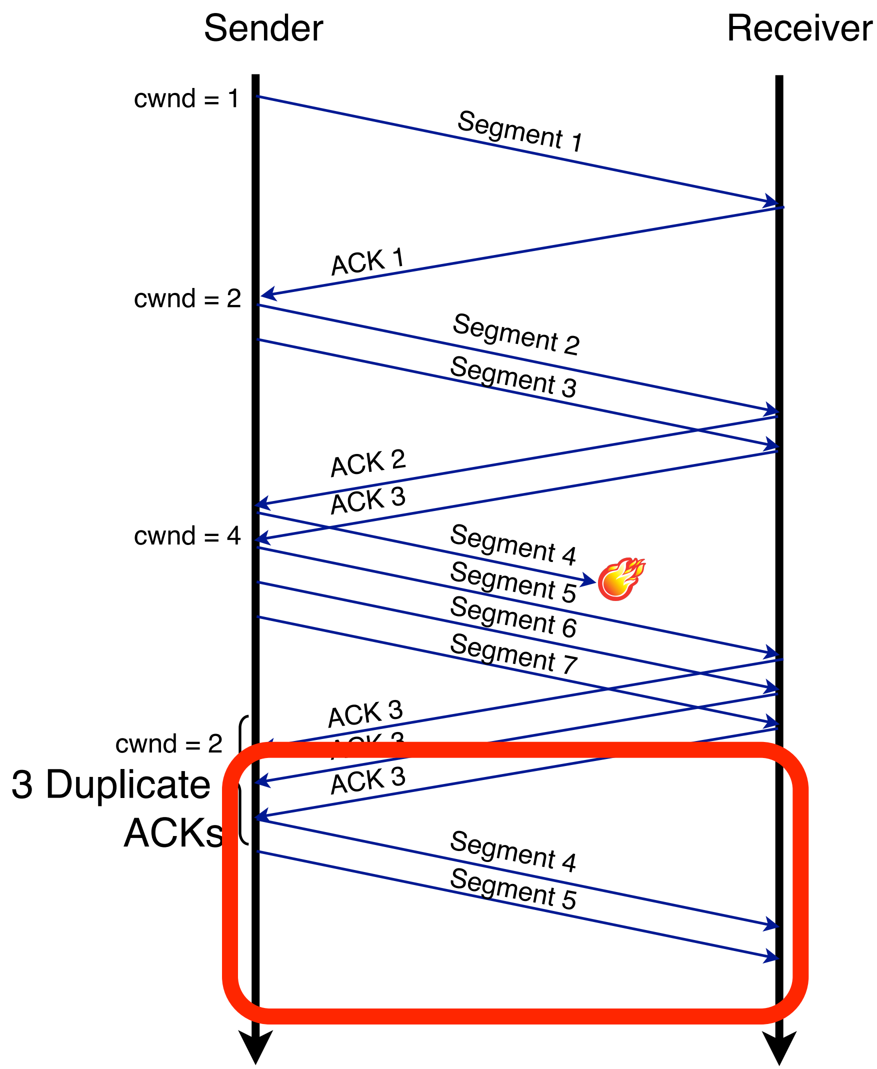
Resume in Congestion Avoidance



TCP SACK Option

(Can TCP Acknowledgment mechanism be improved?)

TCP Acknowledgment Inefficiency



TCP Selective Acknowledgement

Rationale:

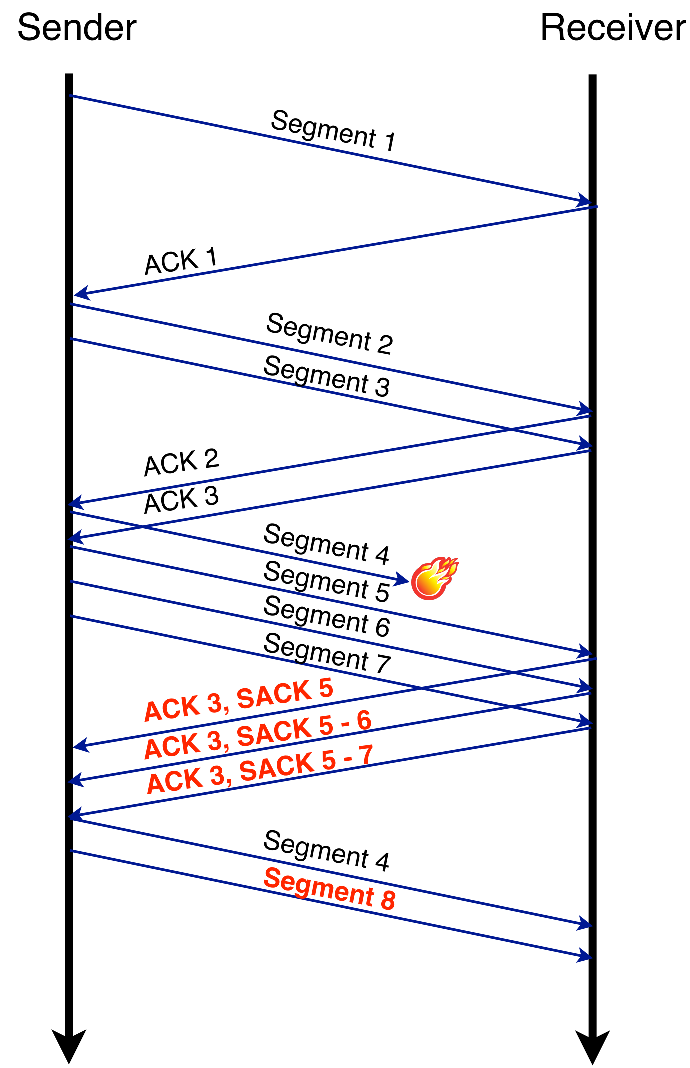
- Increasing the throughput when multiple packets are lost from the same window.

Principle:

- Receiver tells the sender not only the next in-sequence expected byte, but also a range of bytes received out-of-order.

•Mechanism:

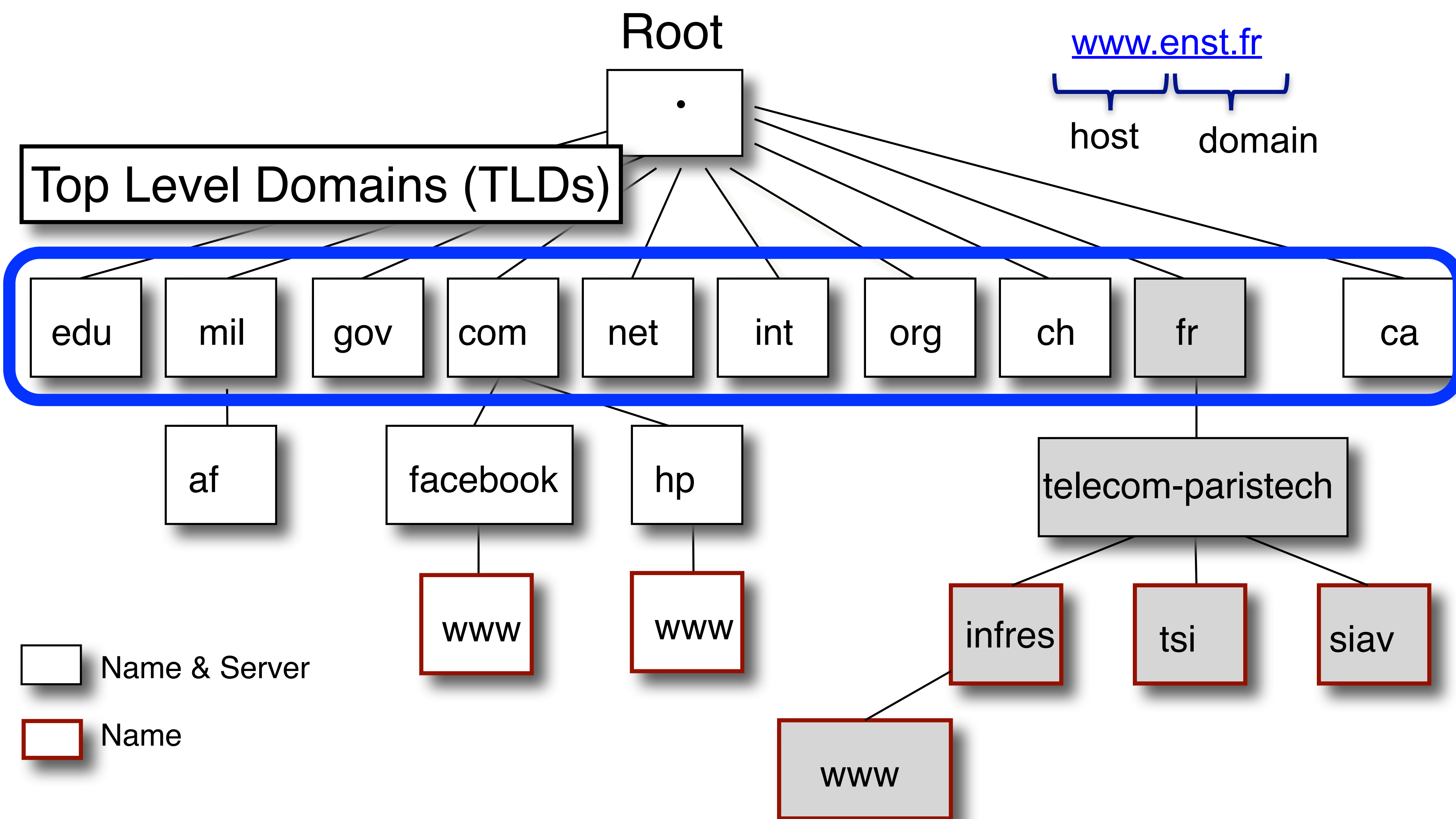
- The receiver replies with a mask of packets received rather than the last segment that was received in a continuity.
- SACK uses the Option field in the TCP header.
- Invoked only if both sides support it



The Domain Name System (DNS)

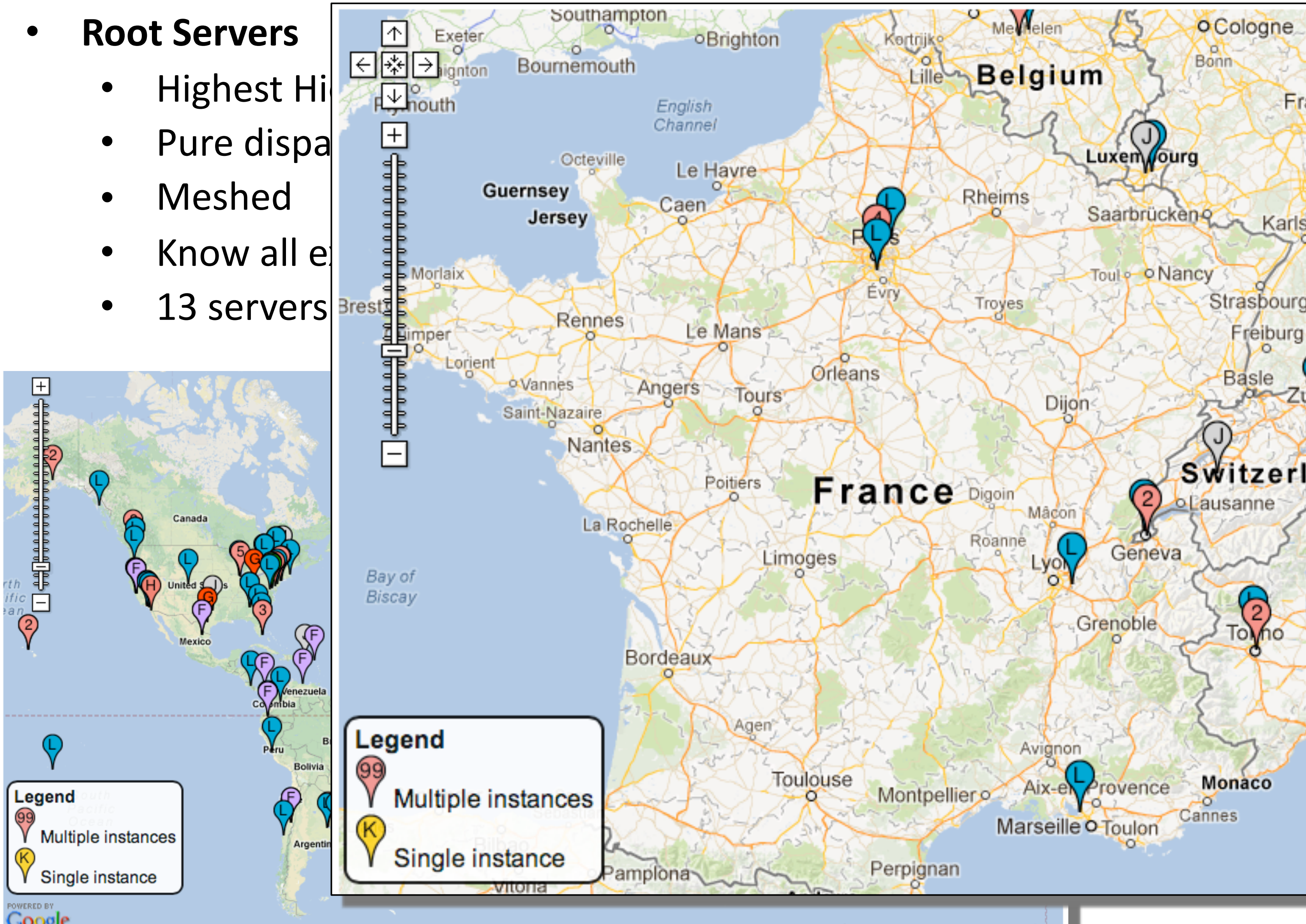
(I want to reach facebook but I do not know its IP ... how do I do?)

Naming Space & Server Hierarchy



DNS Root Servers

- **Root Servers**
  - Highest Hi
  - Pure dispa
  - Meshed
  - Know all e
  - 13 servers



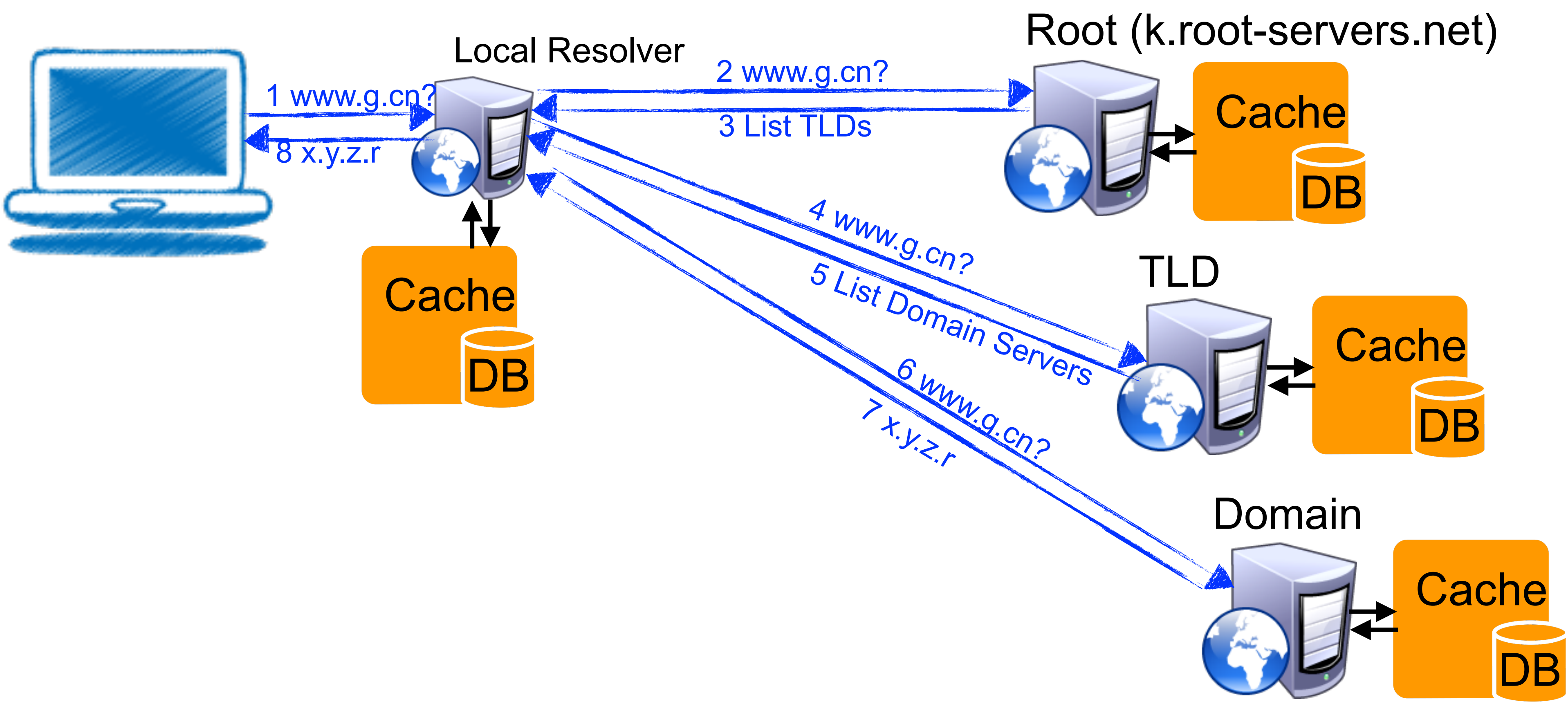
y

Site
-servers.org/

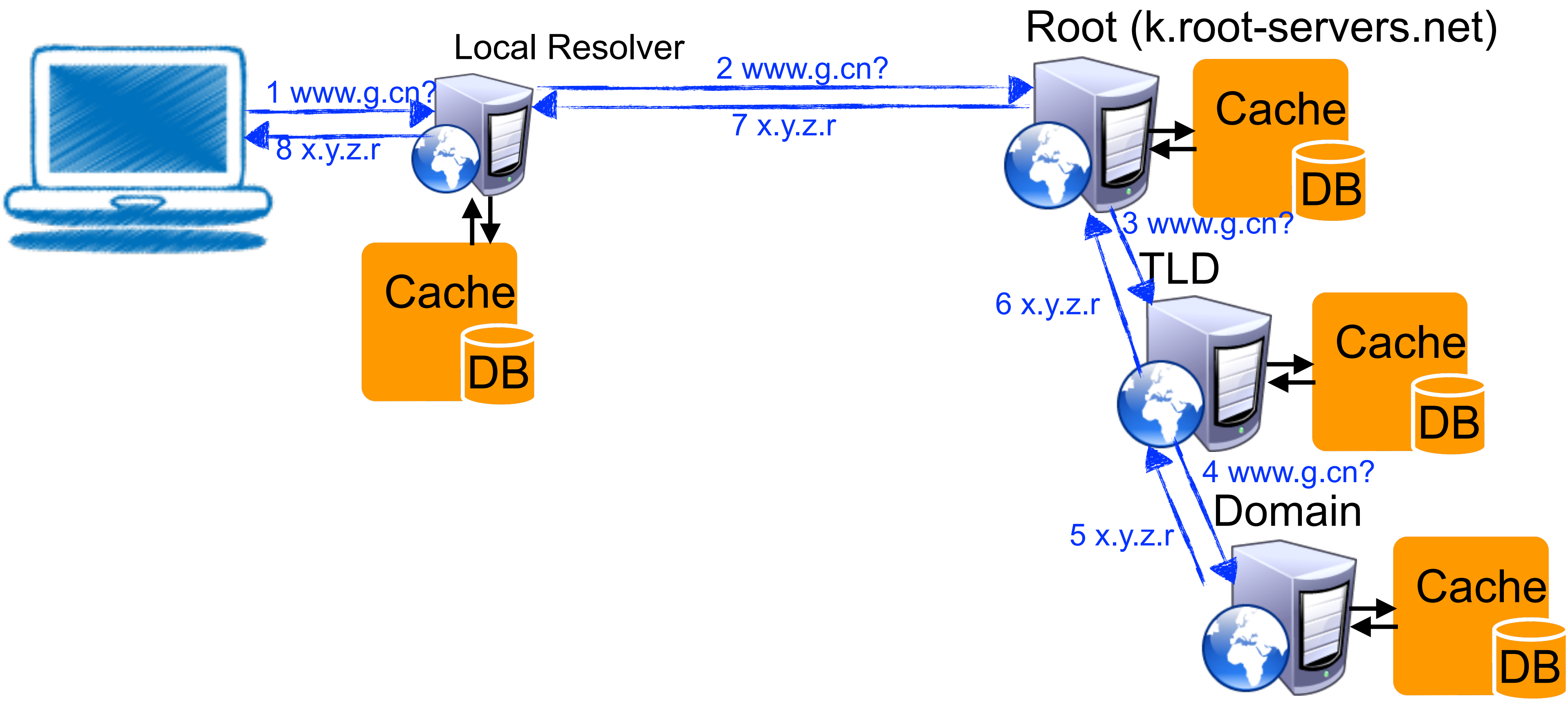
DNS Architecture/Protocol

- Lookup Modes
  - Iterative
    - Required (all name servers implement the iterative method)
    - Returns the response, error, or closest server response
    - Complexity on the client
  - recursive mode
    - Optional (One can choose which clients can use the recursive mode)
    - The easiest for the customer: the name server acts as a resolver and returns an error or response, but never referents
    - Difficult to troubleshoot
    - Benefits from Cache
  - In practice: Hybrid Mode
    - Recursive DNS server to the local
    - Iterative since the local DNS server

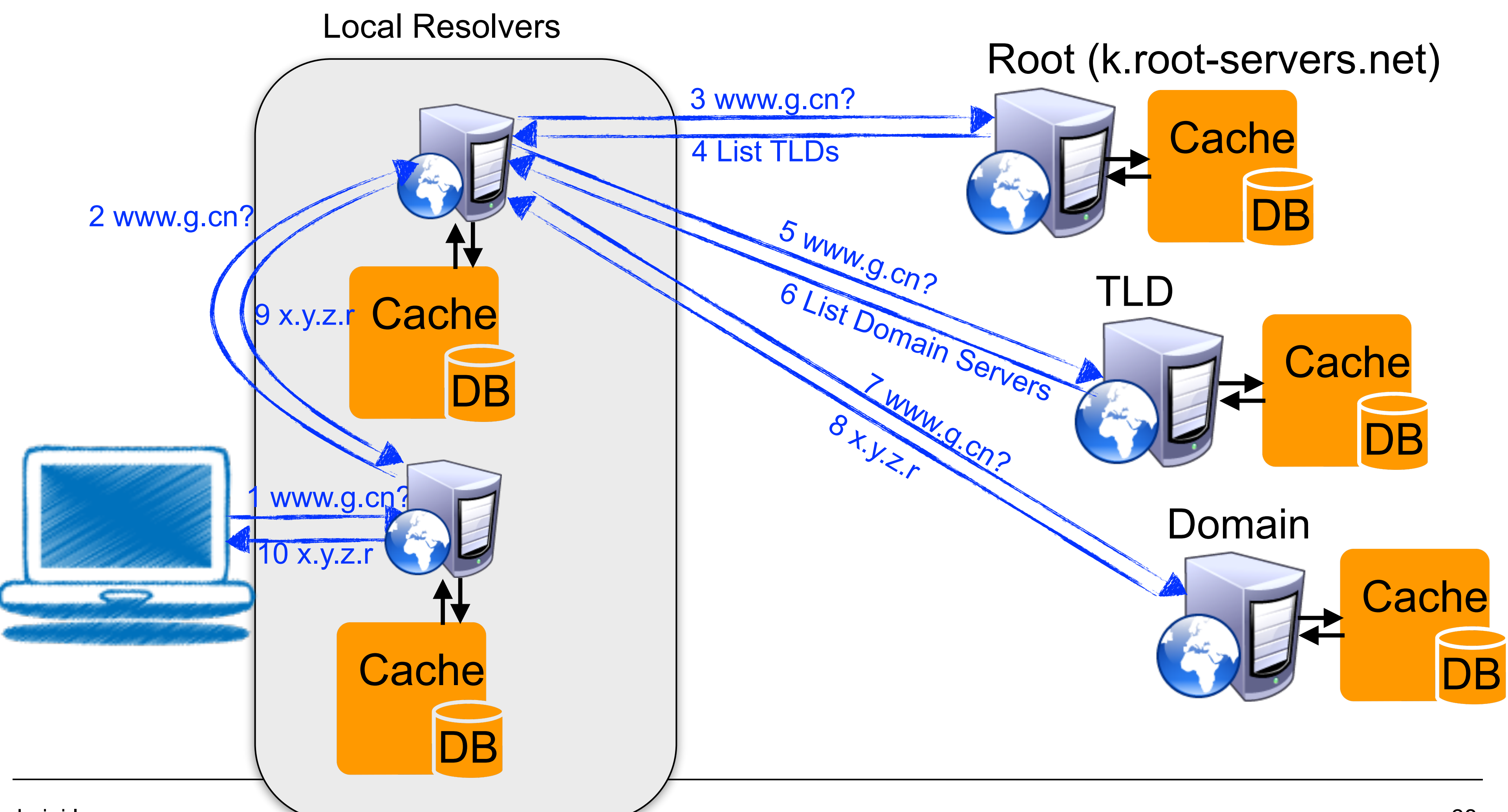
Iterative Mode



Recursive Mode



Hybrid Mode


Local Resolvers
Root (k.root-servers.net)
3 www.g.cn?
4 List TLDs
2 www.g.cn?
9 x.y.z.r
5 www.g.cn?
6 List Domain Servers
7 www.g.cn?
8 x.y.z.r
TLD
Domain
1 www.g.cn?
10 x.y.z.r
Cache
DB

DNS Lookup Tweaks

- Cache:
  - Temporary storage
  - Allows to speed up lookup process
  - Records have a validity time
    - after that they expire and are purged from the cache

- DNS Can use both UDP and TCP, but UDP has priority
  - DNS messages have a MTU of octets
  - If reply is longer, it is sent truncated and the client re-issue the query using TCP.

DNS Resources

- Information Unit: RR (Ressources Records)
  - Name, Type, Class, TTL, Value
- Types of RR:
  - NS (Name Server) :
    - Identifies Authoritative Domain Name Servers
      - telecom-paristech.fr -> ns6.enst.fr
      - telecom-paristech.fr -> ns8-ext.enst.fr
  - CNAME (Canonical Name) :
    - alias
      - www.telecom-paristech.fr -> vili.enst.fr
  - MX (Mail eXchanger) :
    - Identifies Mail Server for the Domain
  - A (Address IPv4) :
    - Correspondance to IPv4 Addresses
      - vili.enst.fr -> 137.194.52.8
  - AAAA (Address IPv6) :
    - Correspondance to IPv6 Addresses

DNS Example

```
Terminal — tcsh — 80×40
[dhcp164-04] ~ # dig vili.enst.fr ANY

; <<>> DiG 9.8.3-P1 <<>> vili.enst.fr ANY
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 20375
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 5, ADDITIONAL: 10

;; QUESTION SECTION:
;vili.enst.fr.                  IN      ANY

;; ANSWER SECTION:
vili.enst.fr.           600     IN      A       137.194.52.8
vili.enst.fr.           600     IN      AAAA    2001:660:330f:34::8

;; AUTHORITY SECTION:
enst.fr.                172800  IN      NS      ns8-ext.enst.fr.
enst.fr.                172800  IN      NS      ns3.enst.fr.
enst.fr.                172800  IN      NS      ns7-ext.enst.fr.
enst.fr.                172800  IN      NS      ns6.enst.fr.
enst.fr.                172800  IN      NS      ns2.enst.fr.

;; ADDITIONAL SECTION:
ns2.enst.fr.            172800  IN      A       137.194.2.34
ns2.enst.fr.            172800  IN      AAAA    2001:660:330f:2::22
ns3.enst.fr.            172800  IN      A       137.194.32.84
ns3.enst.fr.            172800  IN      AAAA    2001:660:330f:20::54
ns6.enst.fr.            172800  IN      A       137.194.2.16
ns6.enst.fr.            172800  IN      AAAA    2001:660:330f:2::10
ns7-ext.enst.fr.        172800  IN      A       178.32.50.142
ns7-ext.enst.fr.        172800  IN      AAAA    2001:41d0:2:2ee4::1
ns8-ext.enst.fr.        172800  IN      A       79.143.243.129
ns8-ext.enst.fr.        172800  IN      AAAA    2a01:678:2:100::53

;; Query time: 40 msec
;; SERVER: 137.194.2.34#53(137.194.2.34)
;; WHEN: Wed Dec 12 17:20:50 2012
;; MSG SIZE  rcvd: 392

[dhcp164-04] ~ #
```

IPv4 and IPv6 Addresses

Authoritative Servers....

.. and their addresses

End Part 1

ROSP 903 2019

Introduction to TCP/IP Networking

Luigi Iannone

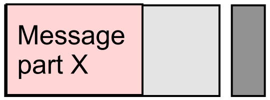
luigi.iannone@telecom-paristech.fr

Road Map

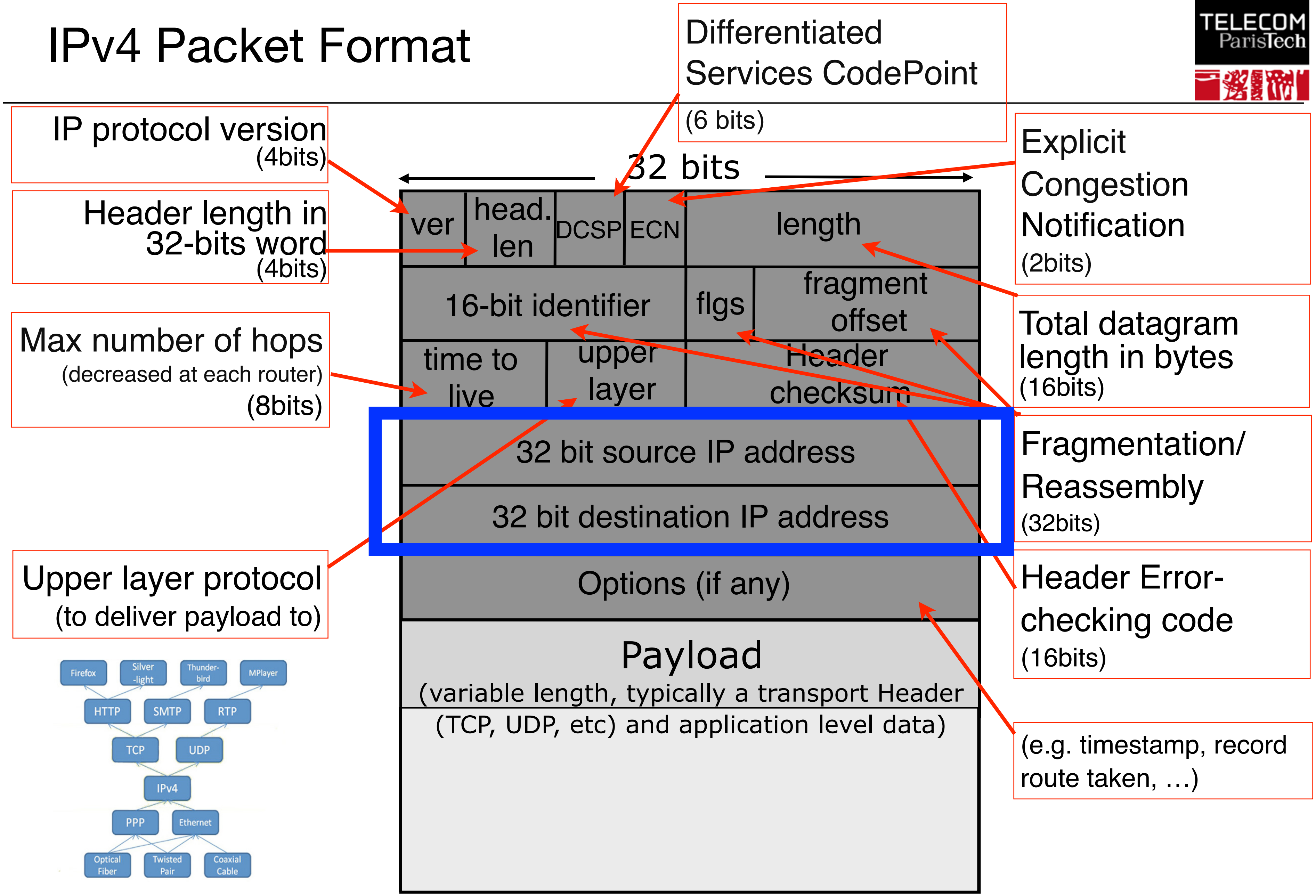
- Internet & Layering
- Transport Layer
- Network Layer
  - IPv4 Header
  - IPv4 Addressing
    - Internet Assigned Numbers Authority (IANA)
  - Routing and Forwarding
    - Border Gateway Protocol (BGP)
    - Interior Gateway Protocol (IGP)
  - IPv4 over Ethernet
  - Dynamic Host Configuration Protocol
  - Internet Control Message Protocol

IPv4 Packet Header

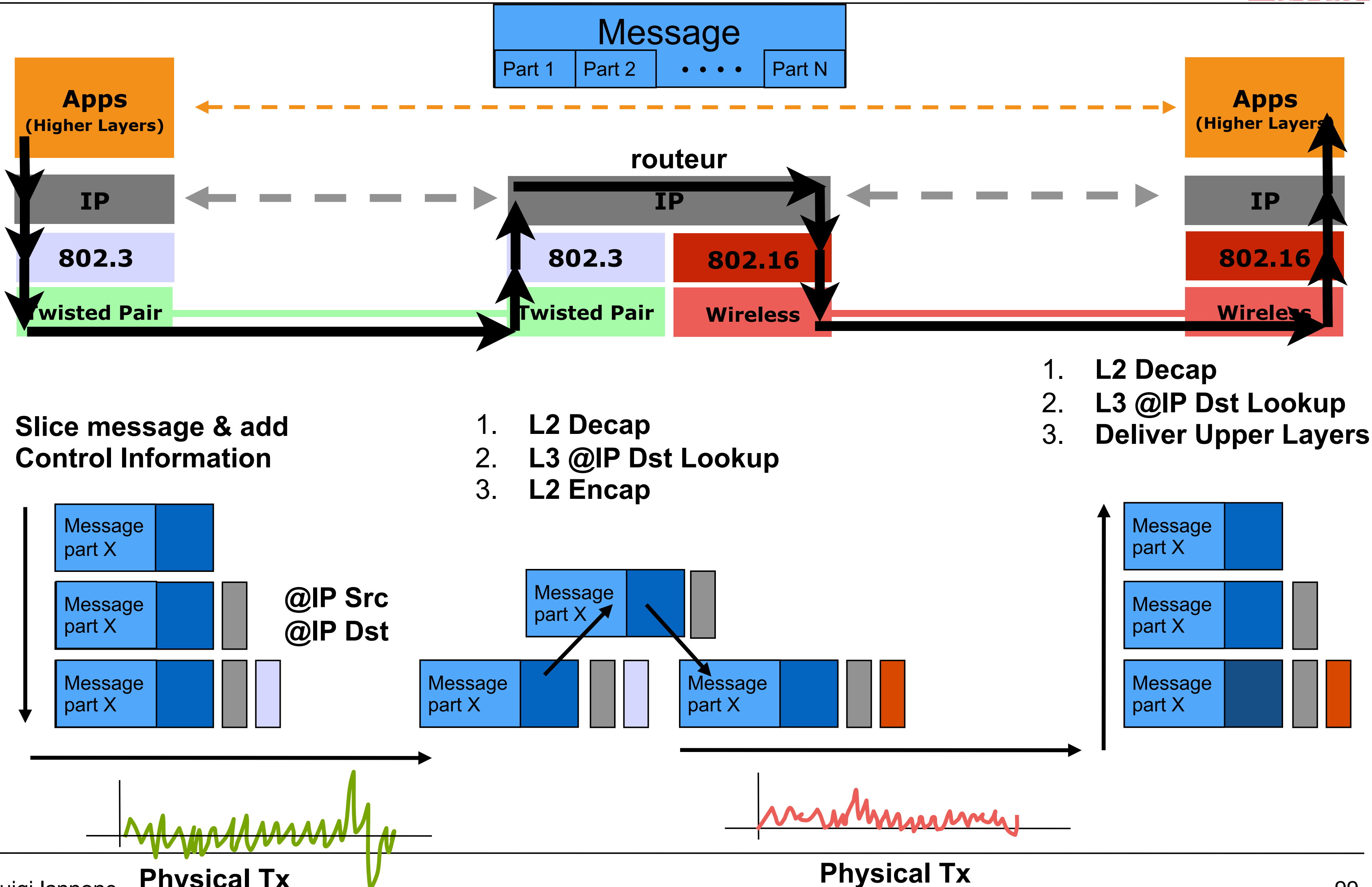
(What's inside an IPv4 packet?)



IPv4 Packet Format

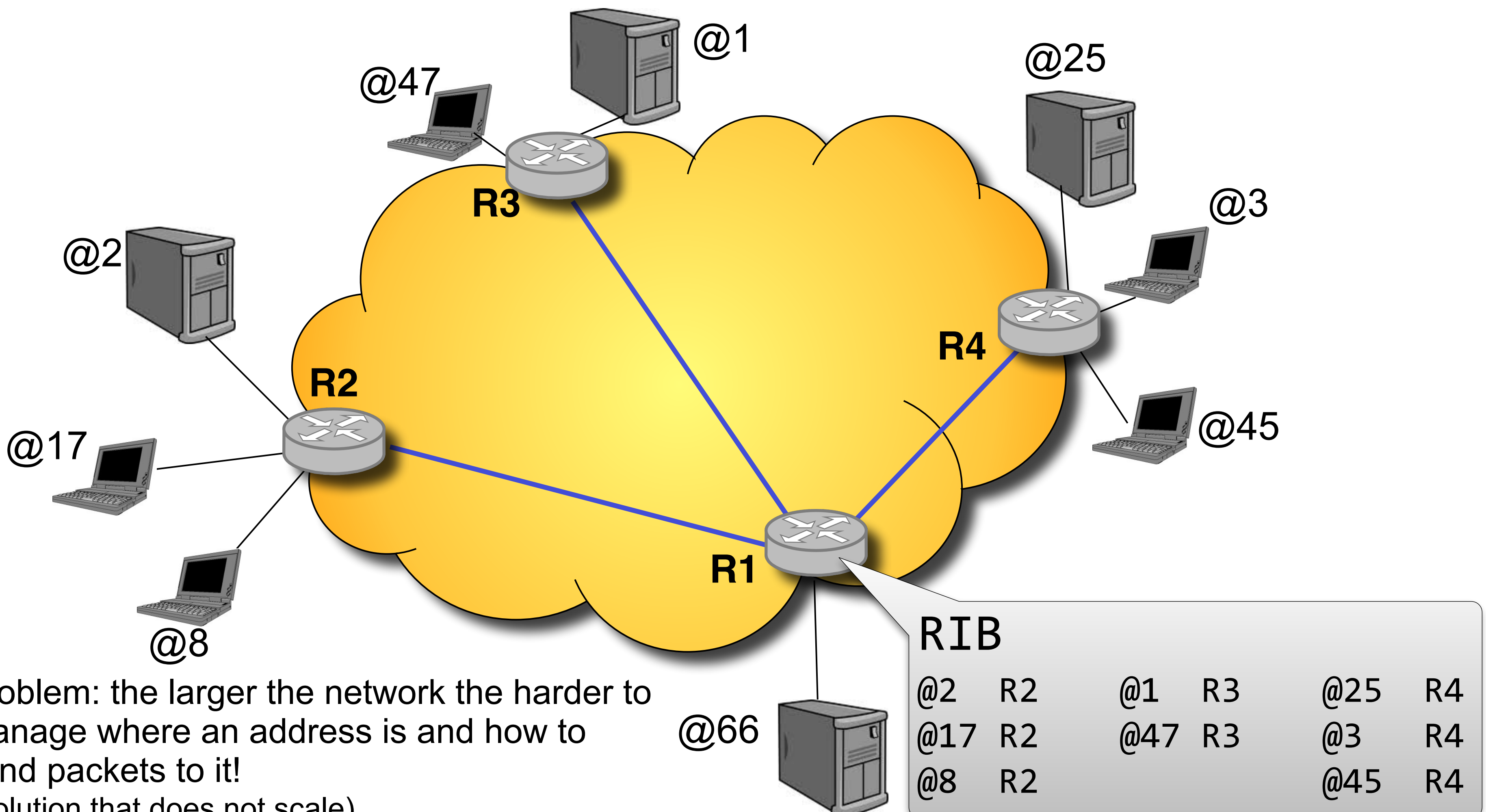


End-to-End transmission



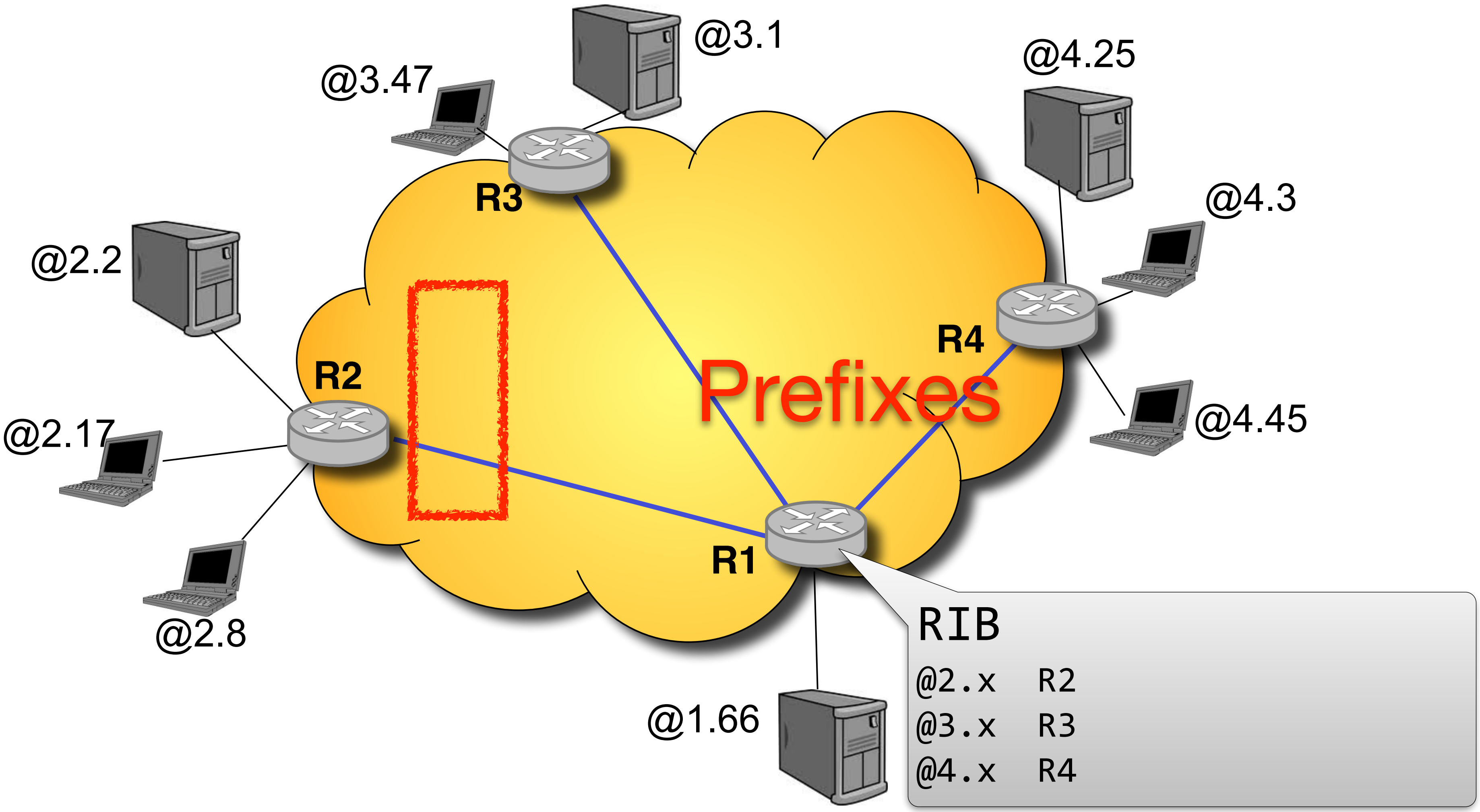
IPv4 Addressing

Flat vs Hierarchical Addressing

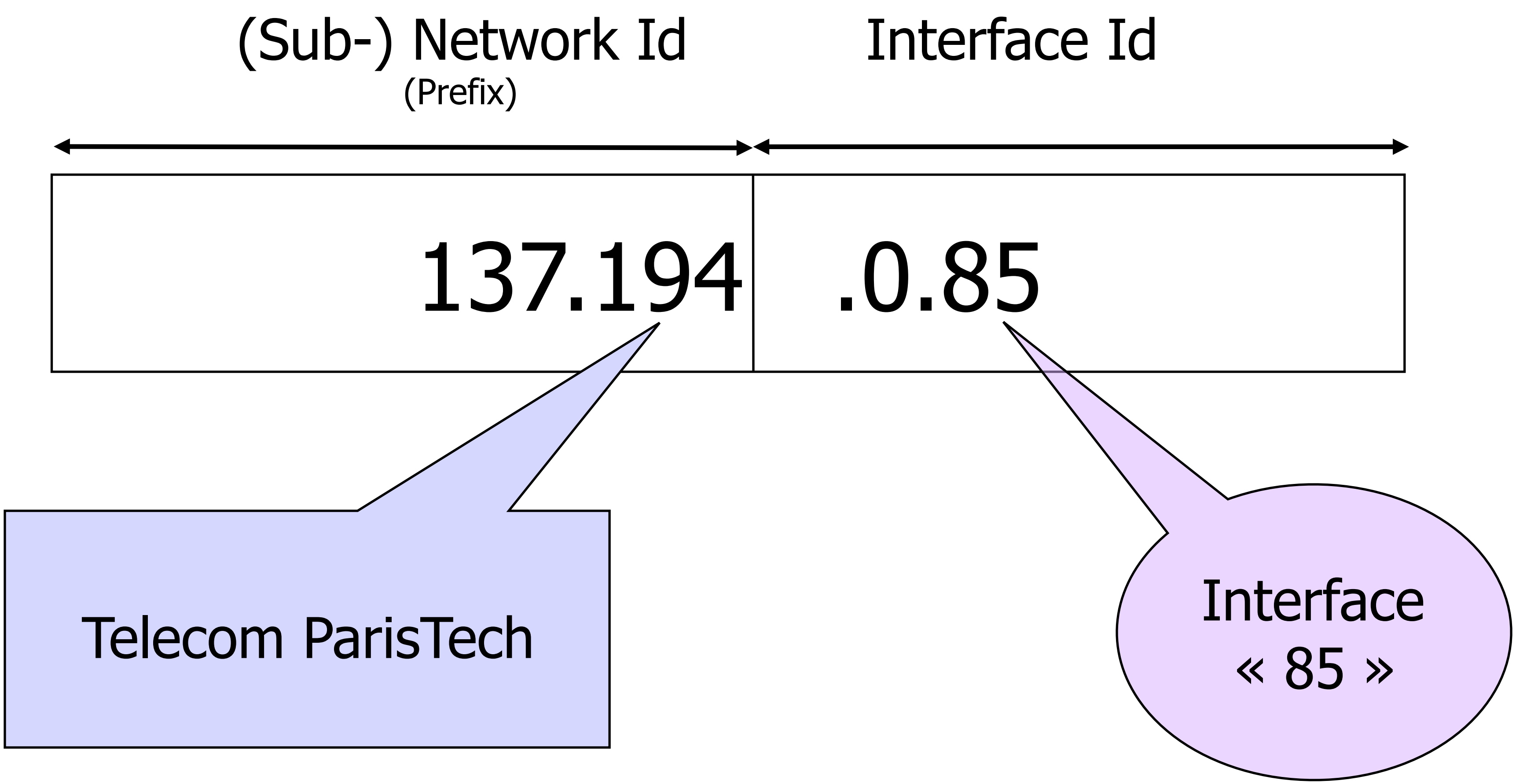


Problem: the larger the network the harder to manage where an address is and how to send packets to it!
(Solution that does not scale)

Flat vs Hierarchical Addressing



IP Address components



Classless Inter-Domain Routing

- An IP address is defined by two values:
  - The address
  - The prefix length

- Notation:
  - A.B.C.D/X

- Examples
  - 137.194.134.70/27
  - 137.194/16
  - 137.194.134.70 (/32 can be omitted)

Address Range

Telecom-Paritech.fr

137.194.0.0/16

137.194.0.0
137.194.0.1
137.194.0.2
...
137.194.255.255

2^{16} = 65536 addresses

Prefix	Number of Addresses
/8	~ 16 millions
/16	~ 65 000
/24	256
/25	128
/26	64

Special Addresses

Telecom-Paritech.fr

137.194.0.0/16

137.194.0.0
137.194.0.1
137.194.0.2
...
137.194.255.255

Addresses to be assigned to interfaces

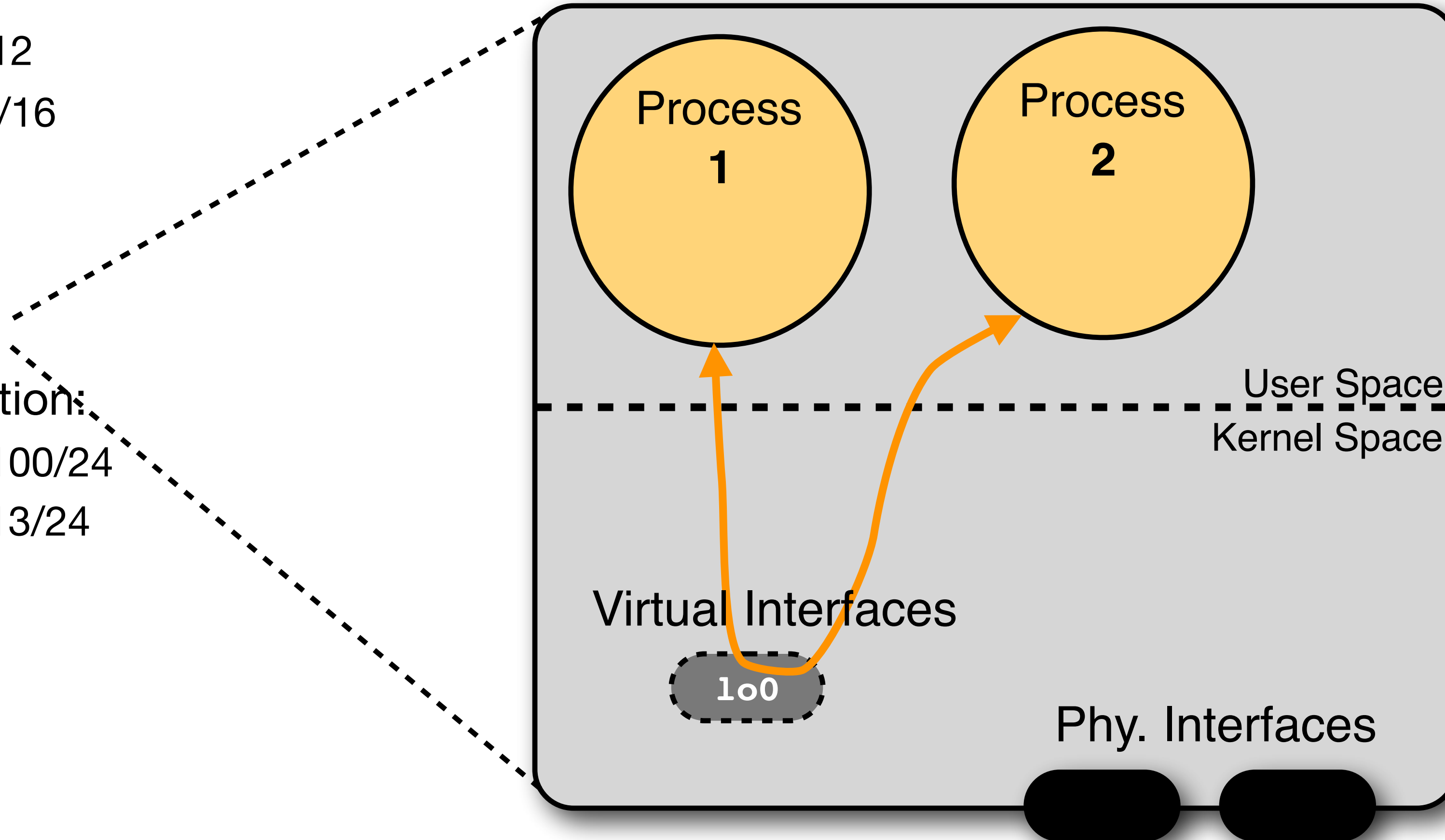
Network Address
- All bits of Interface ID part are 0
- *First* Address of Prefix Range
- Technically good but not used in practice because of buggy implementation

Broadcast Address
- All bits of Interface ID part are 1
- *Last* Address of Prefix Range
- Used to contact all device in an broadcast domain
- Cannot be used to identify an interface

$2^{16} = 65536$ addresses

Important Address Prefixes in the CIDR world

- Private:
  - 10/8
  - 172.16/12
  - 192.168/16
- Loopback:
  - 127/8
- Documentation:
  - 198.51.100/24
  - 203.0.113/24
- Reserved:
  - 240/4
- Multicast:
  - 224/4
-

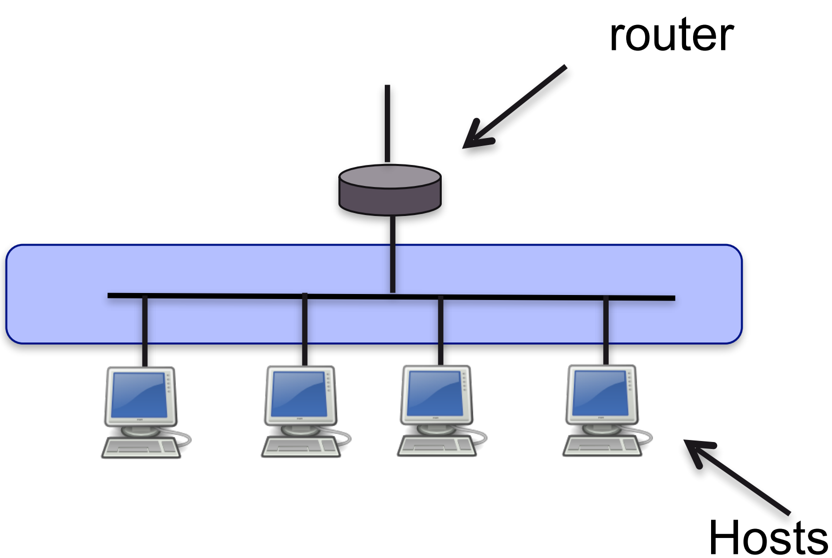


Forwarding

(How to decide where to send a packet?)

Sub-Networking

- Definition
  - Set of devices (or interfaces) sharing the same IP prefix and directly reachable (at IP level)
-



30.83.3.0/25

Is **30.83.3.169** a valid address for such subnet?

Subnet Mask

IP Address `30.83.3.95` — 00011110010100110000001101011111

Mask `255.255.255.128` (/25) — 11111111111111111111111110000000

AND

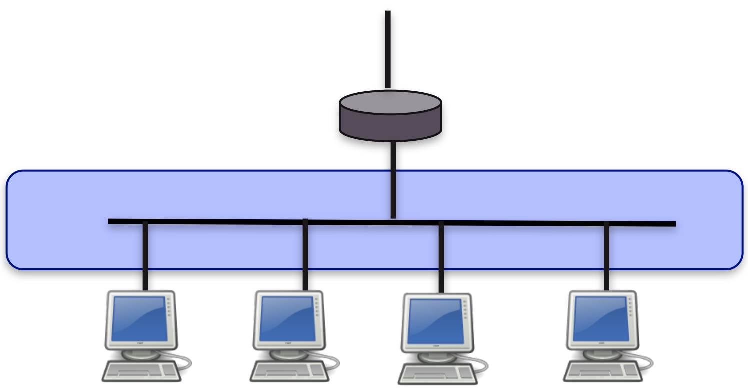
Subnet `30.83.3.0` — 00011110010100110000001100000000

00011110010100110000001110000000

AND

Mask `255.255.255.128` — 11111111111111111111111110000000

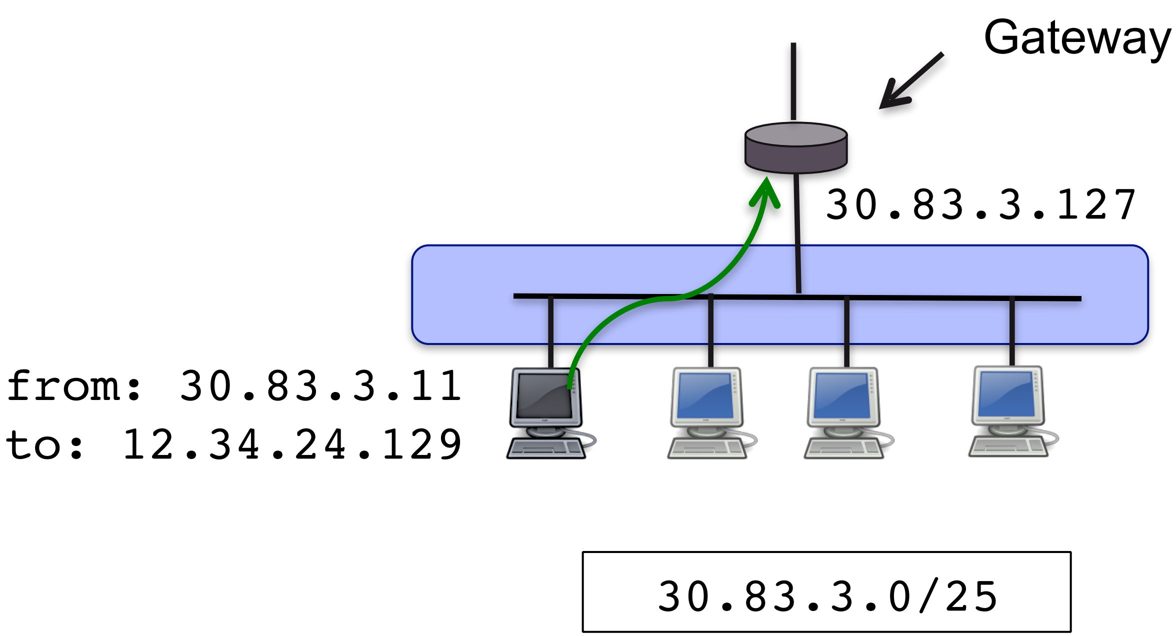
`30.83.3.169` — 00011110010100110000001110101001



`30.83.3.0/25`

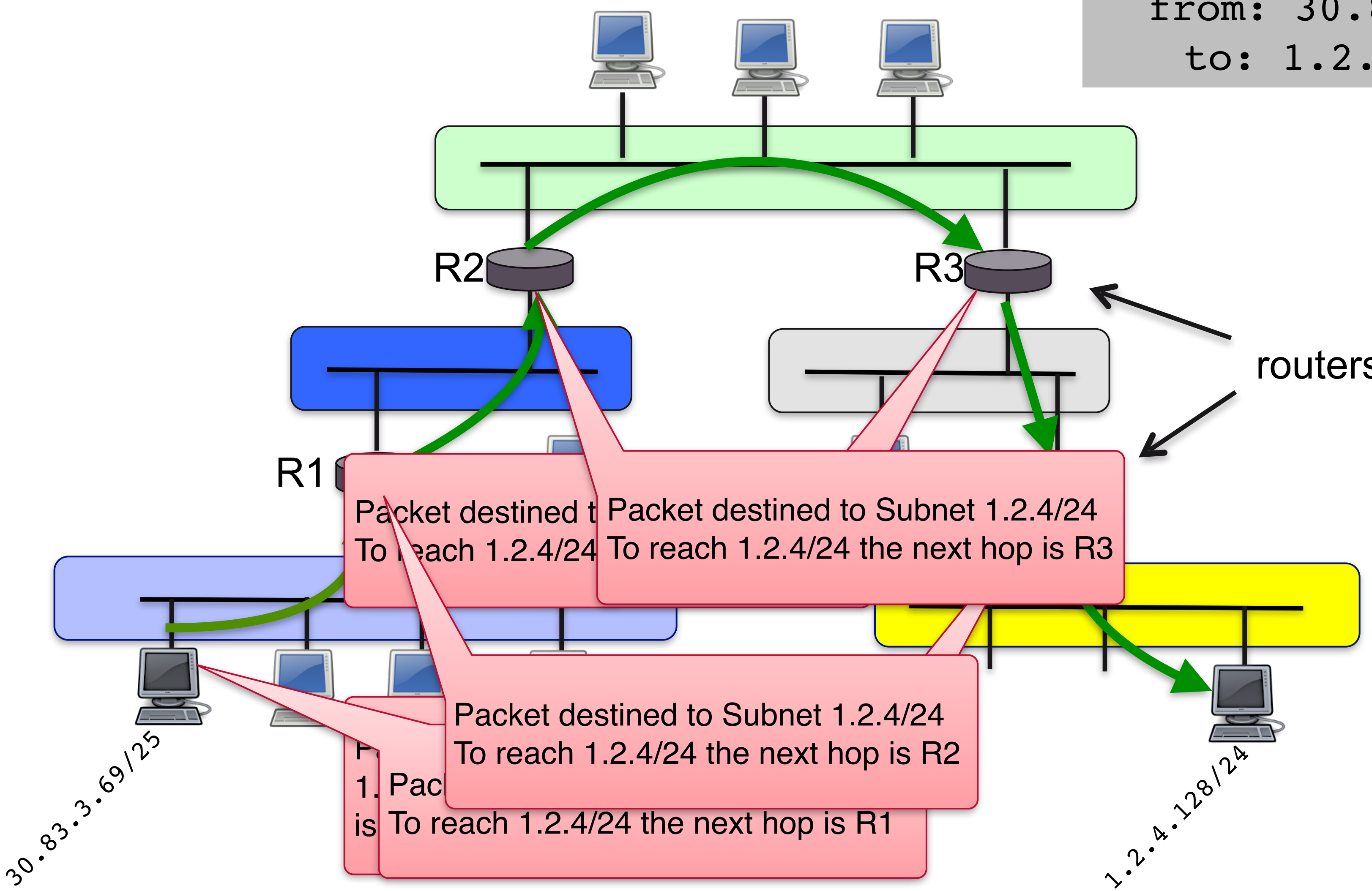
Gateway

- Definition
  - routeur connecting to other subnets
-



Following a path…

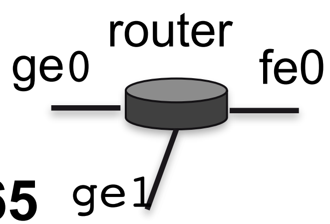
from: 30.83.3.69
to: 1.2.4.128



Routing Table

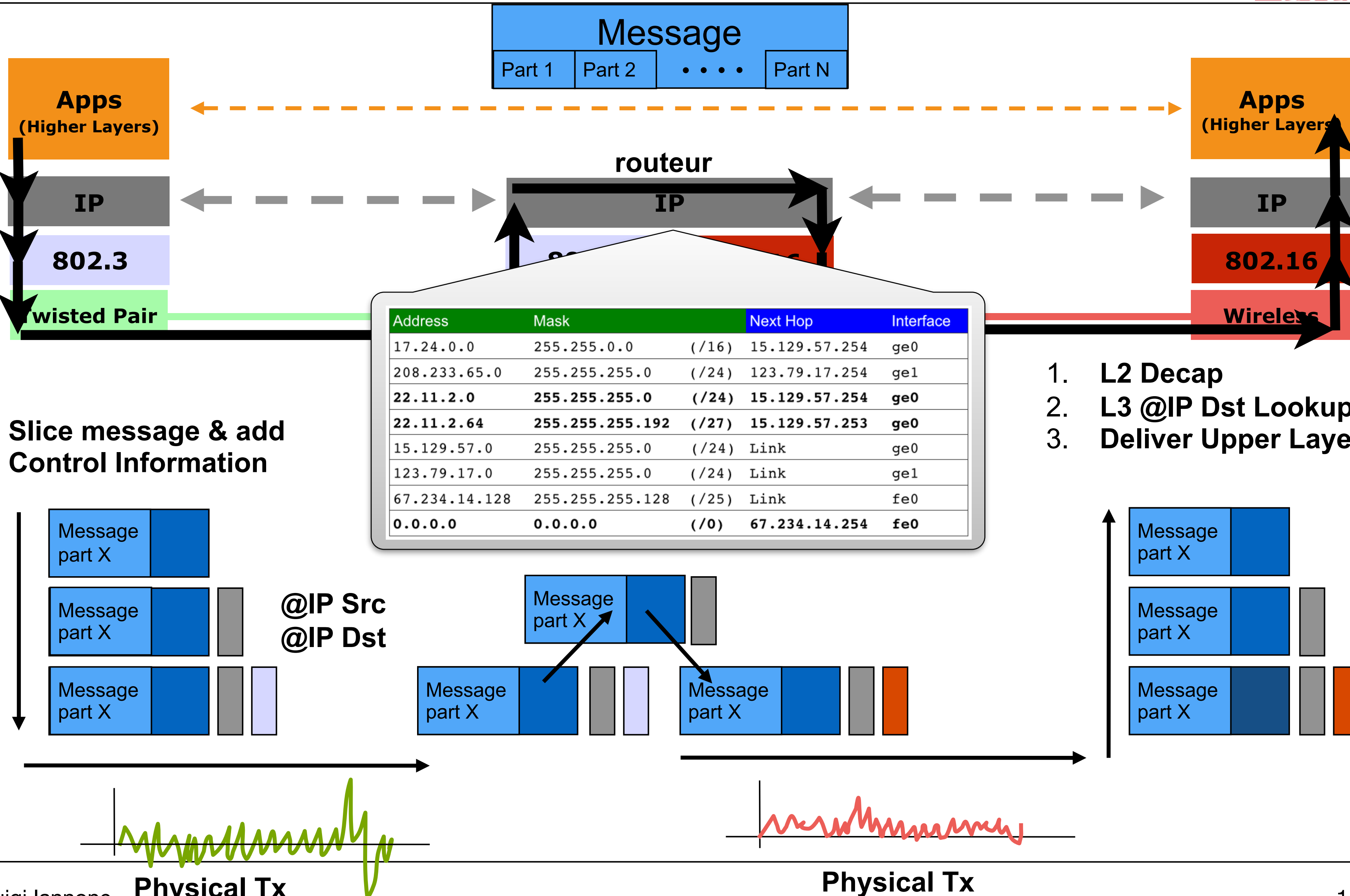
Longest-Prefix Match Rule

Where to send a packet with destination : 22.11.2.65



Address	Mask		Next Hop	Interface
17.24.0.0	255.255.0.0	(/16)	15.129.57.254	ge0
208.233.65.0	255.255.255.0	(/24)	123.79.17.254	ge1
22.11.2.0	**255.255.255.0**	**(/24)**	**15.129.57.254**	**ge0**
22.11.2.64	**255.255.255.192**	**(/27)**	**15.129.57.253**	**ge0**
15.129.57.0	255.255.255.0	(/24)	Link	ge0
123.79.17.0	255.255.255.0	(/24)	Link	ge1
67.234.14.128	255.255.255.128	(/25)	Link	fe0
0.0.0.0	**0.0.0.0**	**(/0)**	**67.234.14.254**	**fe0**

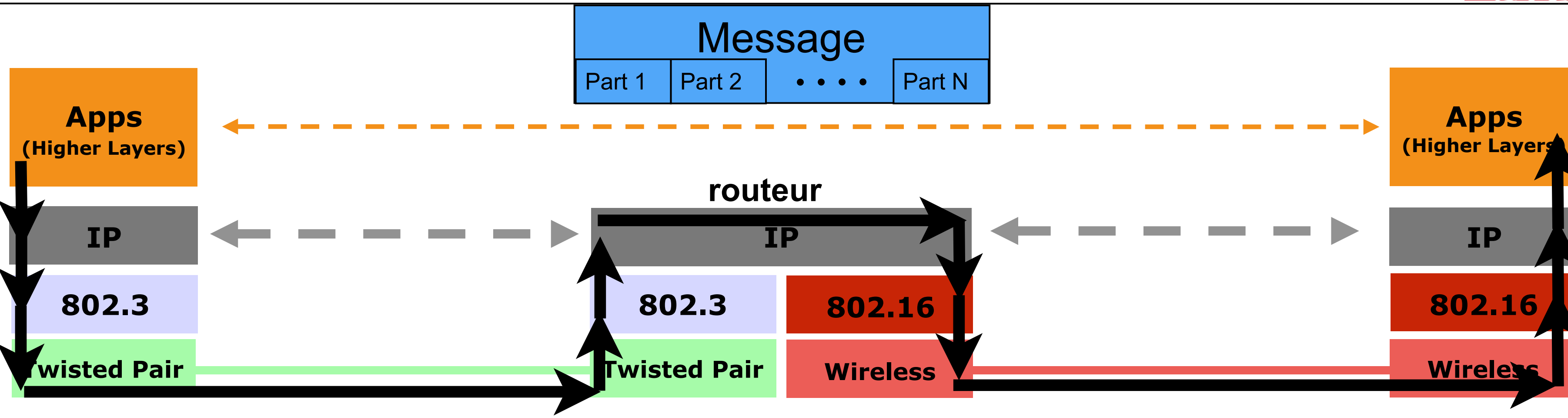
End-to-End transmission



Address	Mask		Next Hop	Interface
17.24.0.0	255.255.0.0	(/16)	15.129.57.254	ge0
208.233.65.0	255.255.255.0	(/24)	123.79.17.254	ge1
22.11.2.0	**255.255.255.0**	**(/24)**	**15.129.57.254**	**ge0**
22.11.2.64	**255.255.255.192**	**(/27)**	**15.129.57.253**	**ge0**
15.129.57.0	255.255.255.0	(/24)	Link	ge0
123.79.17.0	255.255.255.0	(/24)	Link	ge1
67.234.14.128	255.255.255.128	(/25)	Link	fe0
0.0.0.0	**0.0.0.0**	**(/0)**	**67.234.14.254**	**fe0**

1. **L2 Decap**
2. **L3 @IP Dst Lookup**
3. **Deliver Upper Layers**

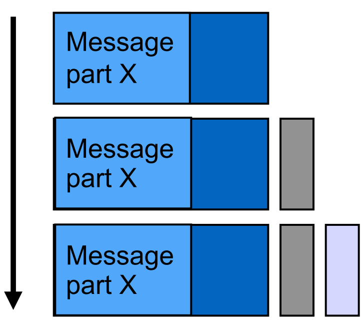
End-to-End transmission



Slice message & add Control Information

1. **L2 Decap**
2. **L3 @IP Dst Lookup**
3. **L2 Encap**

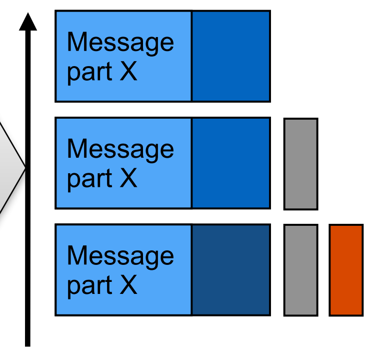
1. **L2 Decap**
2. **L3 @IP Dst Lookup**
3. **Deliver Upper Layers**

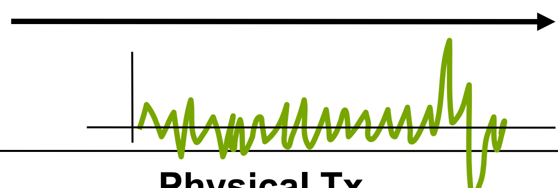


@IP Src
@IP Dst

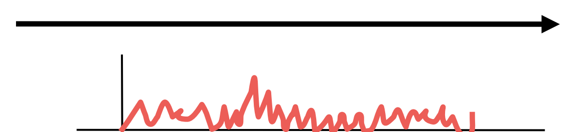
Address	Mask		Next Hop	Interface
17.24.0.0	255.255.0.0	(/16)	15.129.57.254	ge0
208.233.65.0	255.255.255.0	(/24)	123.79.17.254	ge1
22.11.2.0	**255.255.255.0**	**(/24)**	**15.129.57.254**	**ge0**
22.11.2.64	**255.255.255.192**	**(/27)**	**15.129.57.253**	**ge0**
15.129.57.0	255.255.255.0	(/24)	Link	ge0
123.79.17.0	255.255.255.0	(/24)	Link	ge1
67.234.14.128	255.255.255.128	(/25)	Link	fe0
0.0.0.0	**0.0.0.0**	**(/0)**	**67.234.14.254**	**fe0**

	Next Hop	Interface
(/16)	15.129.57.254	ge0
(/24)	123.79.17.254	ge1
(/24)	**15.129.57.254**	**ge0**
(/27)	**15.129.57.253**	**ge0**
(/24)	Link	ge0
(/24)	Link	ge1
(/25)	Link	fe0
(/0)	**67.234.14.254**	**fe0**





Physical Tx



Physical Tx

Routing

(How to know where to send a packet?)

Why Dynamic Routing?

- Automatically detect and adapt to topology changes
- Provide optimal routing
- Scalability
- Robustness
- Simplicity
- Rapid convergence
- Some control of routing choices
  - E.g. which links we prefer to use

What dynamic routing does do and what doesn't do?

What does do

- Spread knowledge about IP prefixes
  - So to fill routing tables

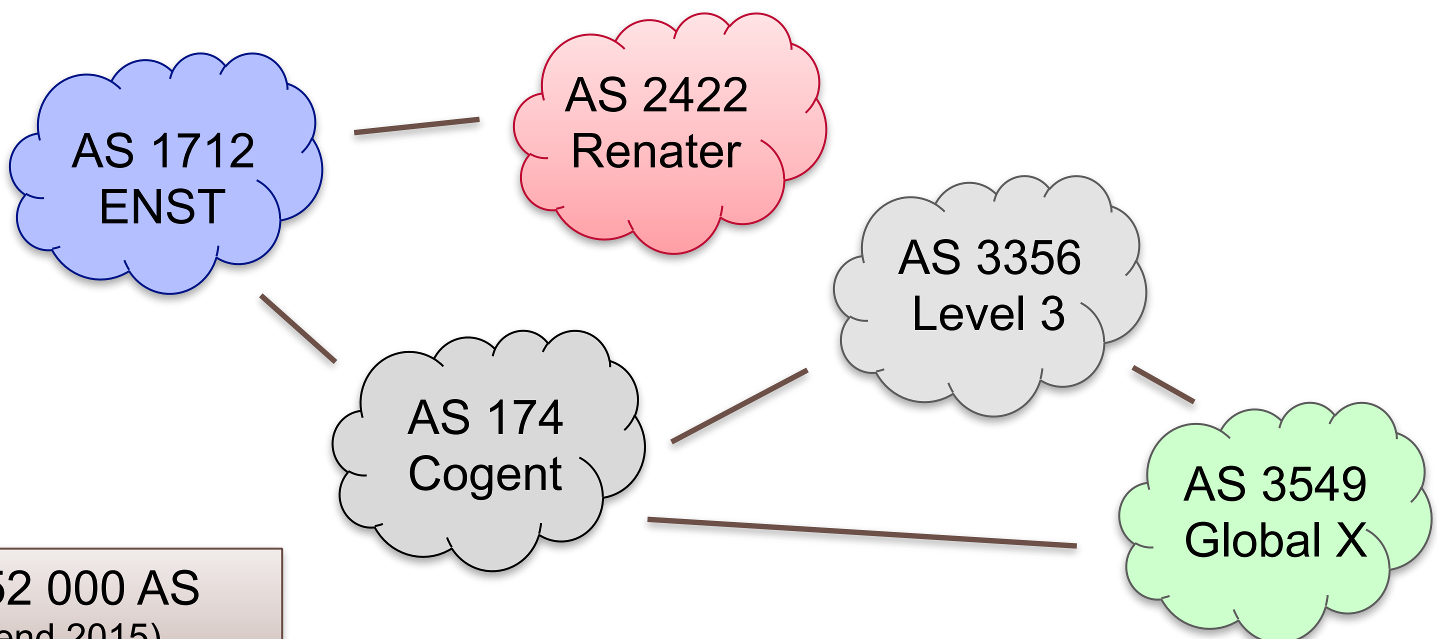
What doesn't do

- Configure IP addresses on your interfaces
  - DHCP for end-host
  - Static Routes

    Static ≠ Manually configured
- Provide Default routes
  - Unless explicitly configured
- Decide Policies
  - What to advertise toward whom
  - Filtering

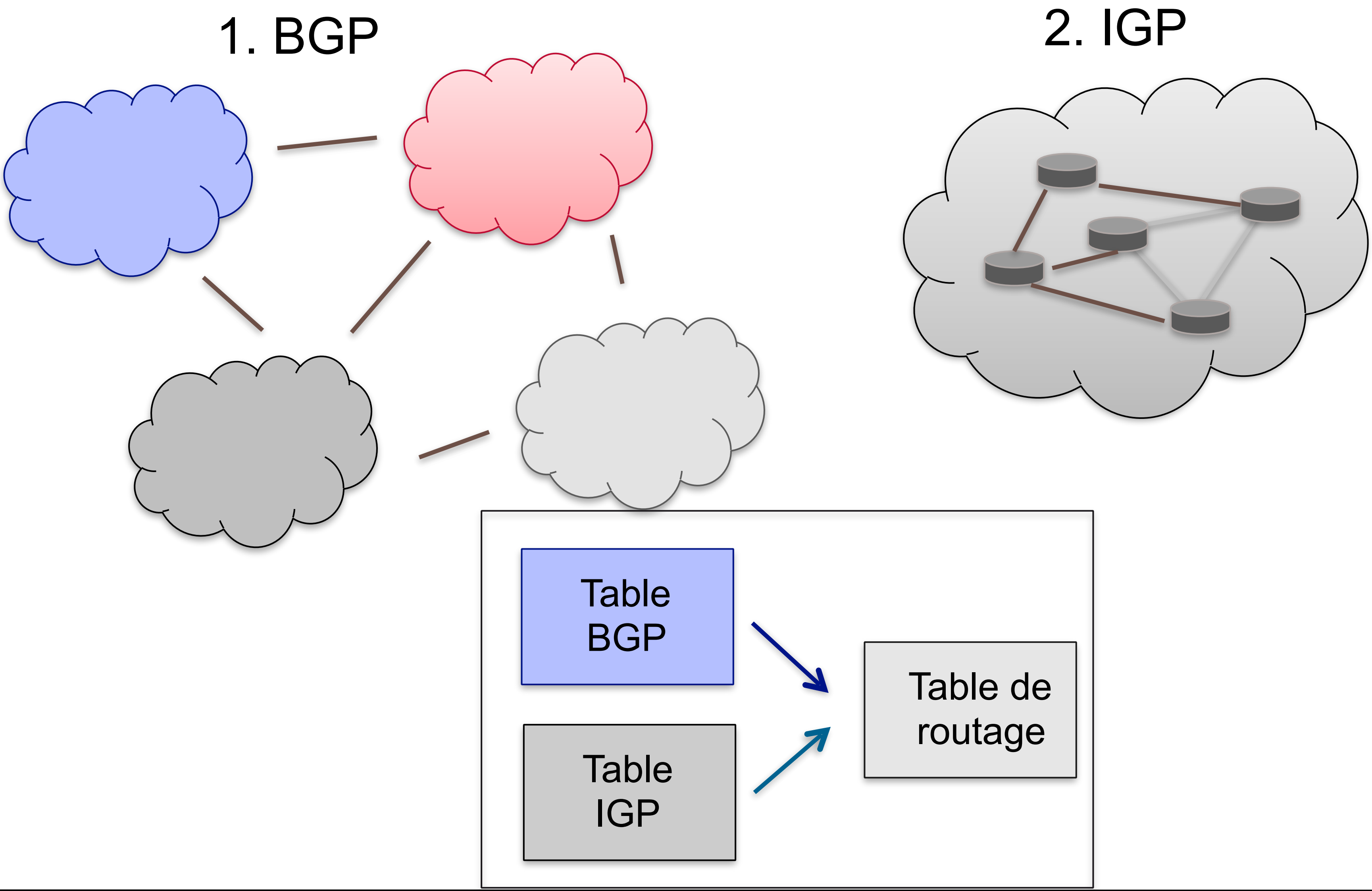
Internet Organisation

- AS - Autonomous Systems
  - Network under the same administrative entity



~ 52 000 AS
(end 2015)

Two-tier Internet Routing

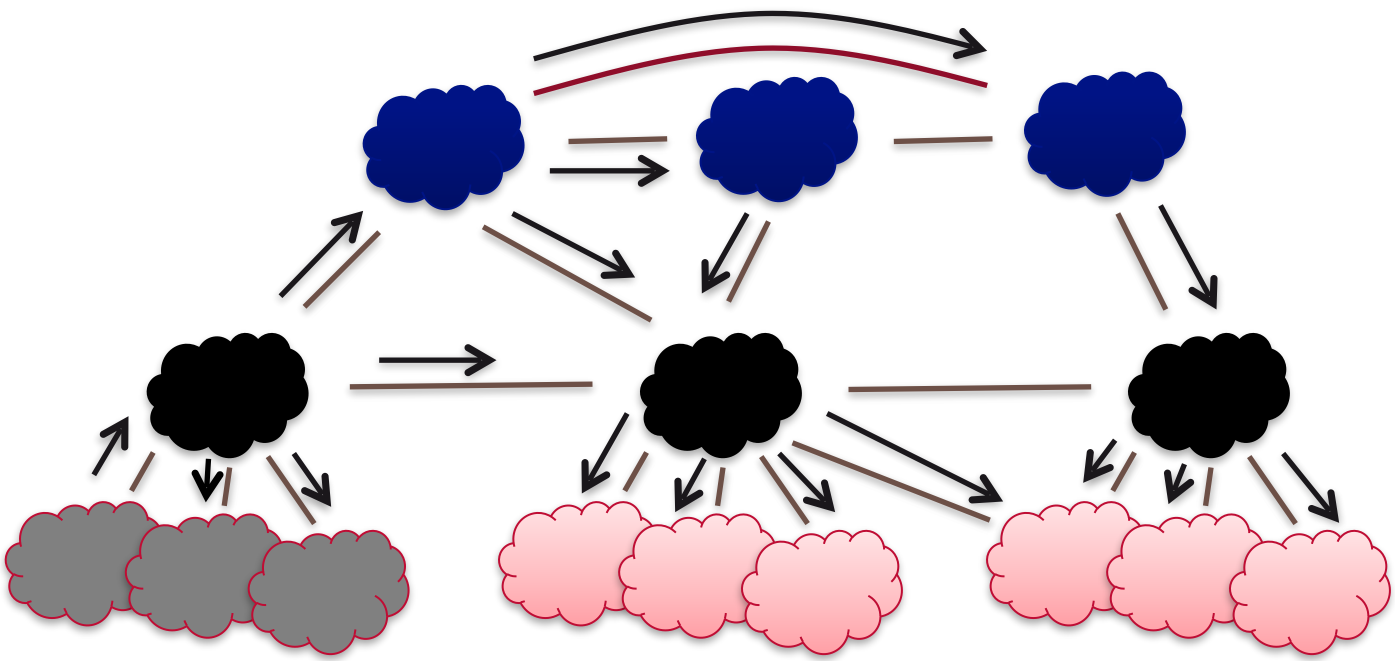


BGP

(Inter-Domain Routing)

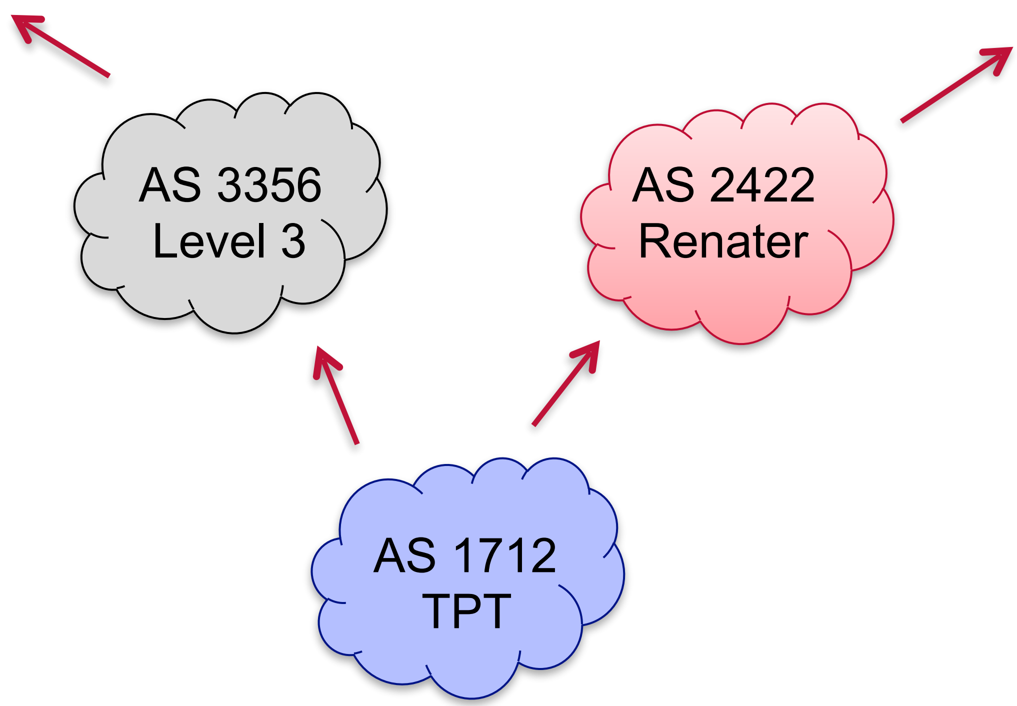
Border Gateway Protocol – BGP

- (Selectively) Advertises the (best) route toward every prefix it knows
  - Selectively: depending on AS policy
  - Best: may depend on local decision



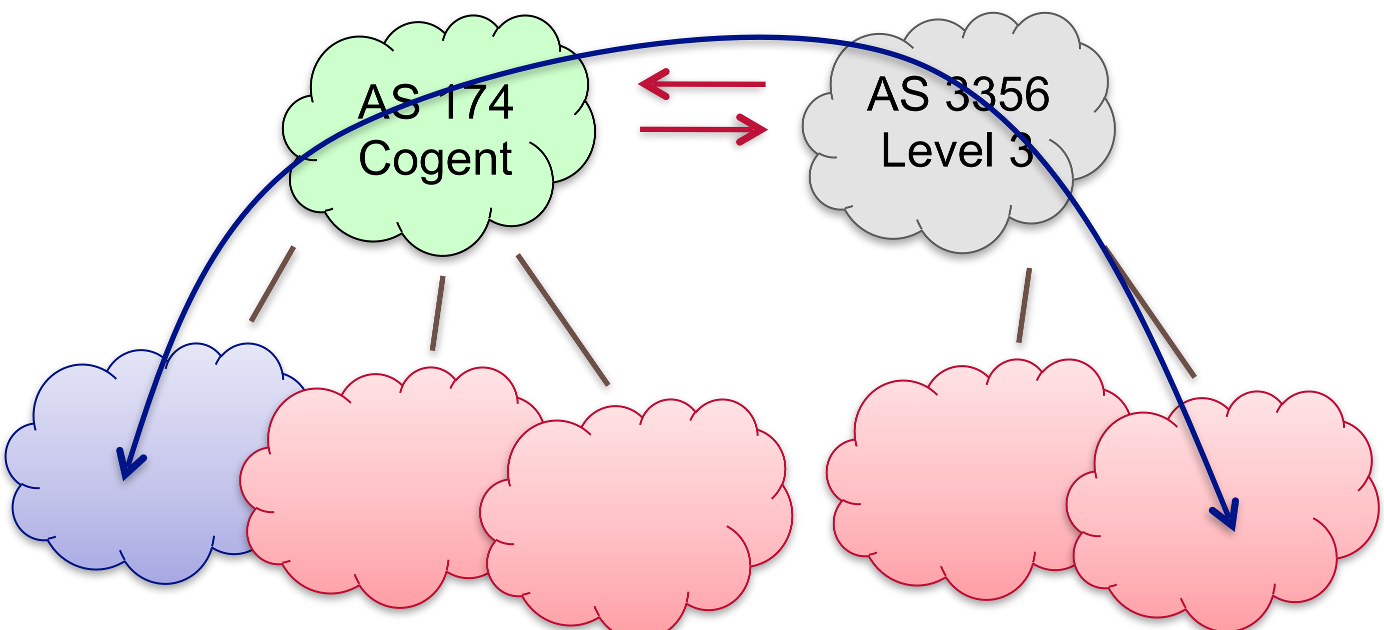
BGP: Customer - Provider

- Principle
  - Customer AS pays Provider AS for connectivity



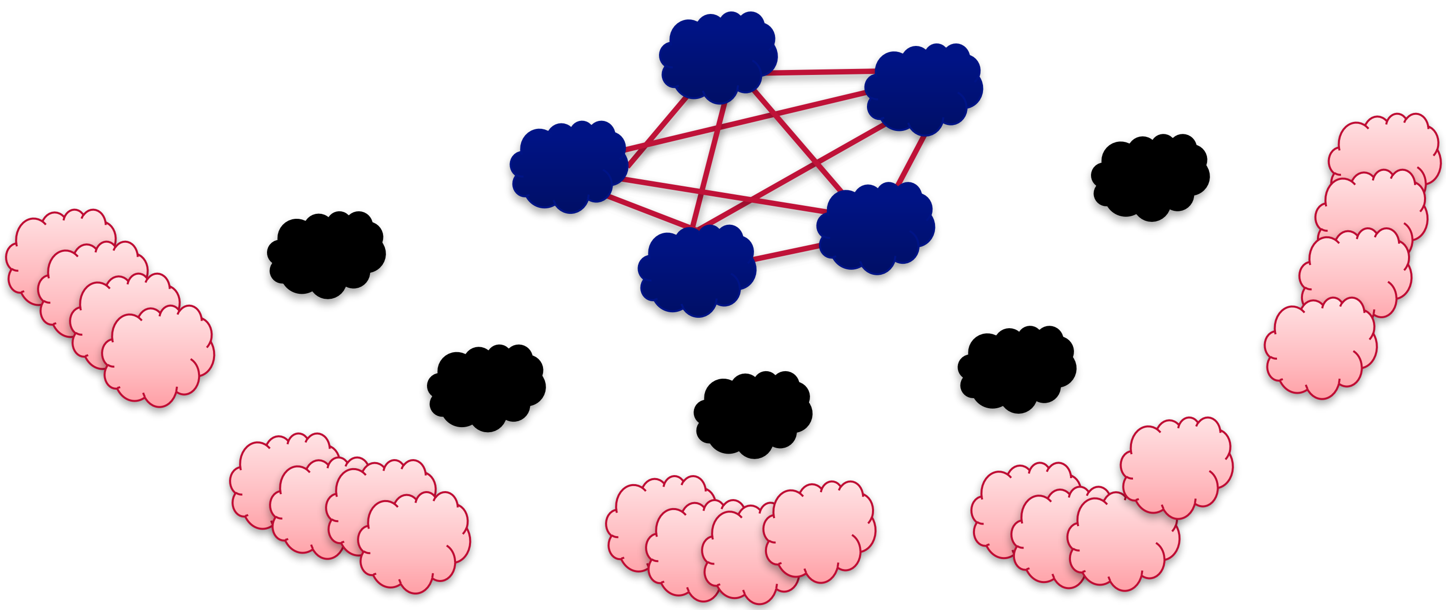
BGP: Peering

- Principle
  - Money-free settlement for AS exchanging same amount of traffic
  - Only for traffic to/from their own customers



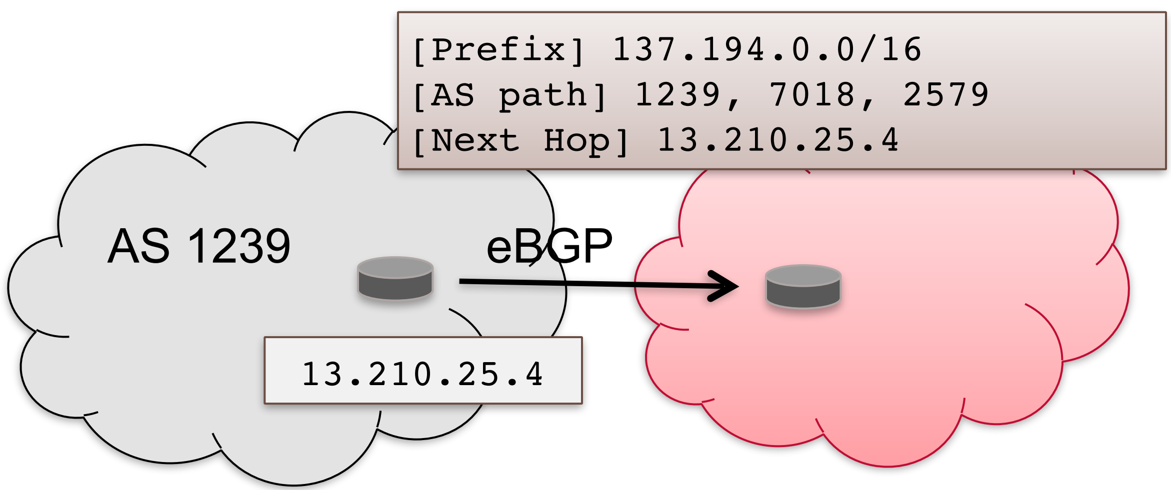
Internet AS Hierarchy

- 3 Classes
  - tier-1: Provider-only peering to each other in a full-mesh
  - tier-2: Provider & Customers (transit)
  - tier-3: Customer-only (stub)



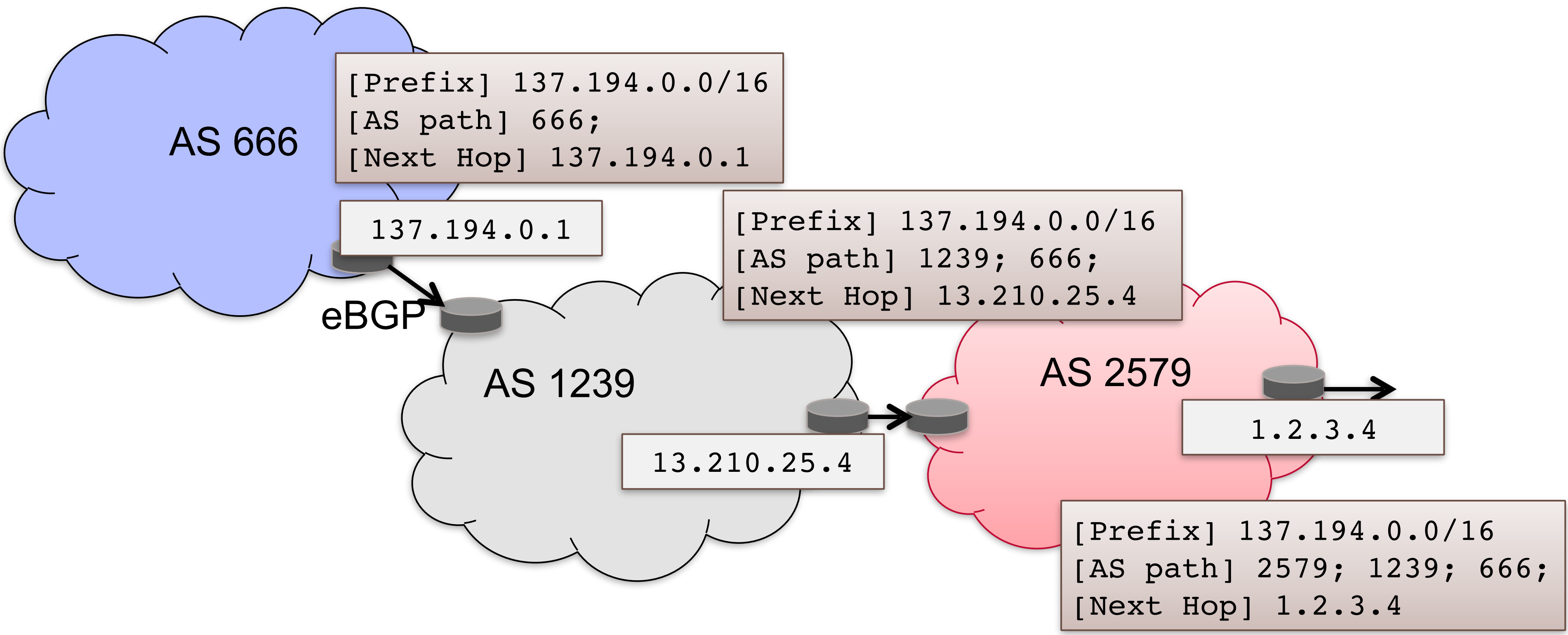
BGP Path Atributes

- Attributes of every path to a prefix
  - AS path: vector of AS to the prefix
  - Next Hop: IP address of the next router

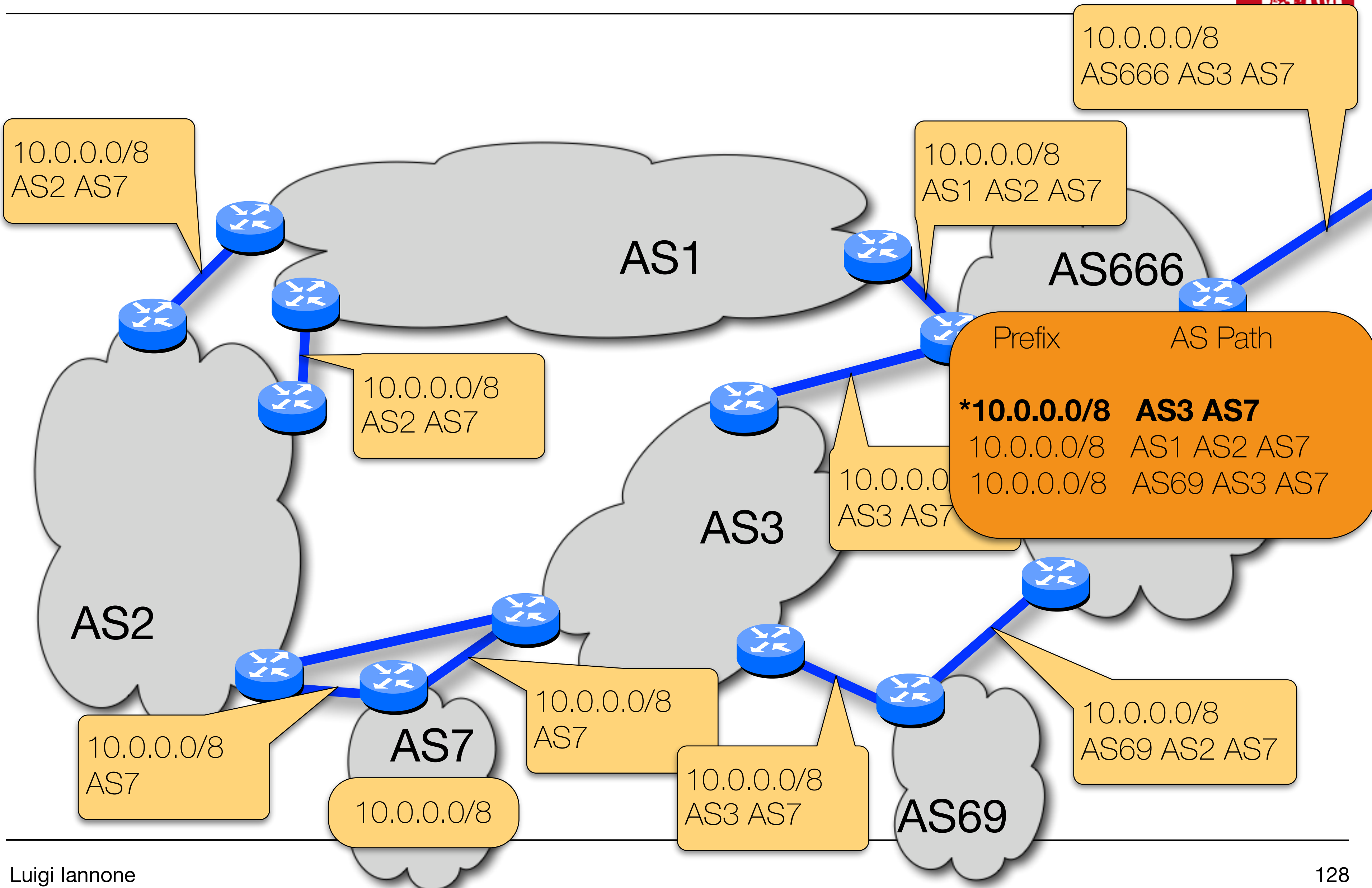


BGP Path Attributes

- Attributes of every path to a prefix
  - AS path: vector of AS to the prefix
  - Next Hop: IP address of the next router



BGP Route Propagation



BGP More Path Attributes

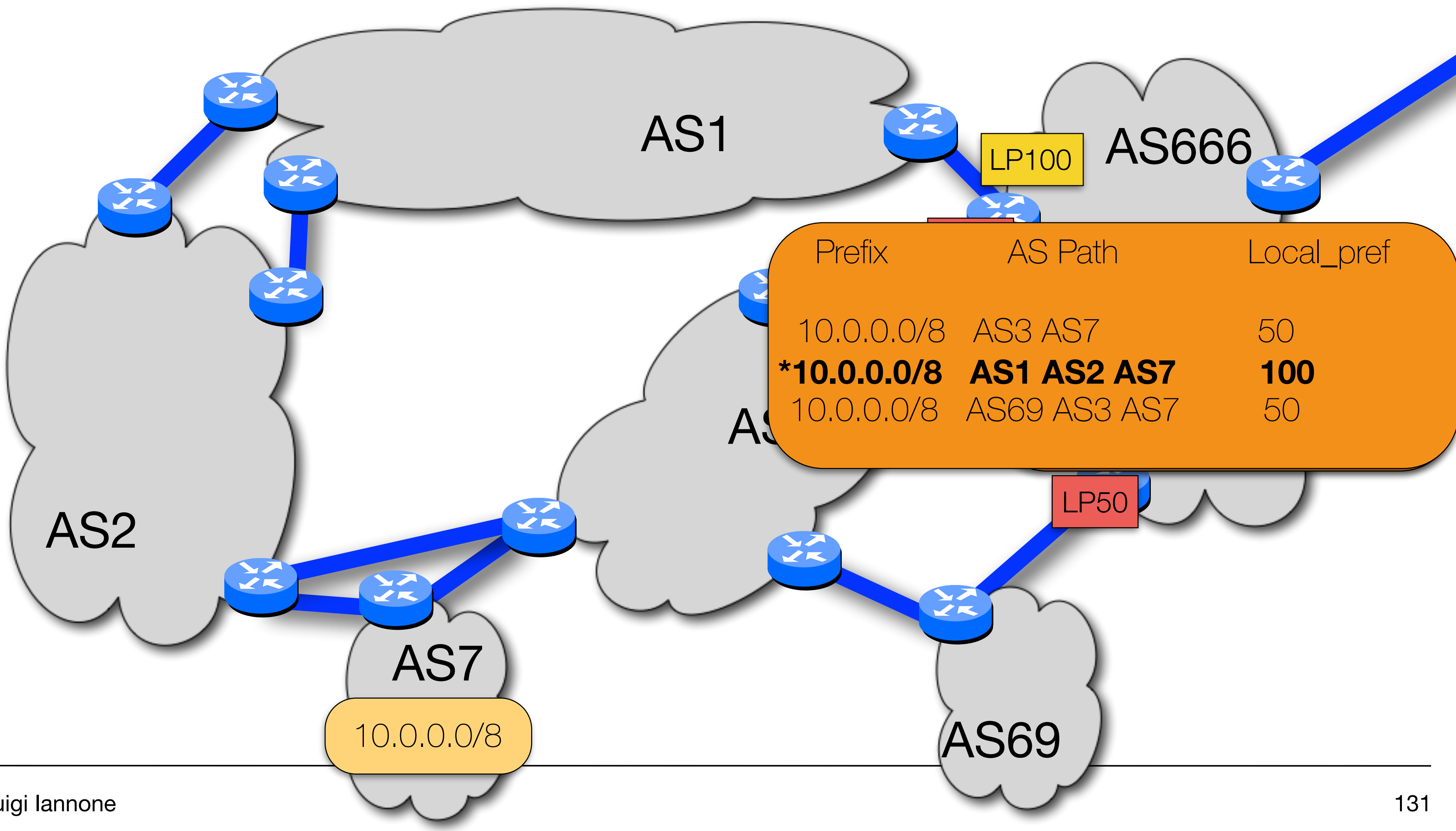
- Origin
  - How this route was injected into BGP in the first place
- Next_hop
  - Exit border router
- Multi-Exit-Discriminator
  - Preference between 2 or more sessions among the same AS pair
- Local-Pref
  - Local preference setting
- Atomic Aggregate
  - The path is the result of aggregation
- Aggregator
  - ID of proxy aggregator
- Community
  - Locally defined information field
- Destination-Pref
  - Preference setting for remote AS

BGP Detailed (Best) Route Selection

For a set of received advertisements of the same prefix the local "best" selection is based on:

0. Apply Filtering Policies
1. Highest value Local_Pref
2. Shortest AS Path length
3. Lowest MED
4. Minimum IGP cost to Next_Hop Address
5. eBGP-learned routes preferred to iBGP-learned routes
6. Prefer paths learned from router with smaller ID (selected in the same way as for OSPF)

Local_Pref Attribute Example

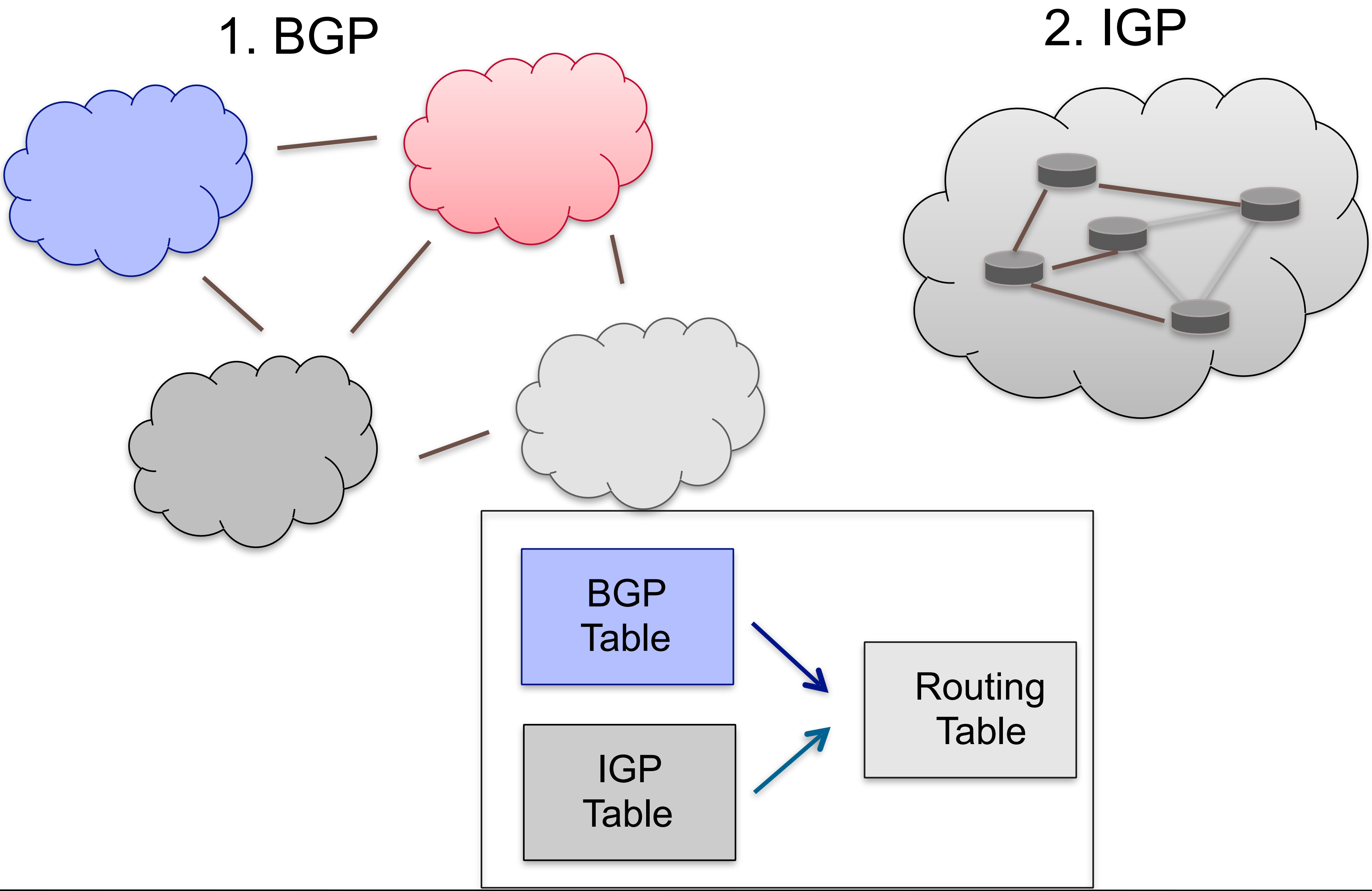


Prefix	AS Path	Local_pref
10.0.0.0/8	AS3 AS7	50
***10.0.0.0/8**	**AS1 AS2 AS7**	**100**
10.0.0.0/8	AS69 AS3 AS7	50

IGP

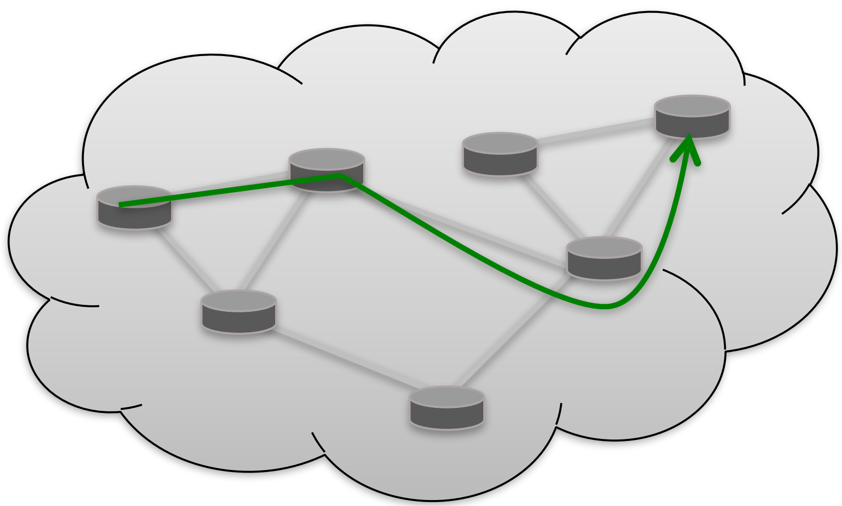
(Interior Gateway Protocol)
(Aka Intra-Domain Routing)

Two-tier Internet Routing



Interior Gateway Protocol – IGP

- Principle
  - Select the “shortest-path” inside an AS
  - Must be based on an additive metric
    - $f\left(LINK_a + LINK_b\right) = f\left(LINK_a\right) + f\left(LINK_a\right)$



Routing Algorithms

Distance Vector

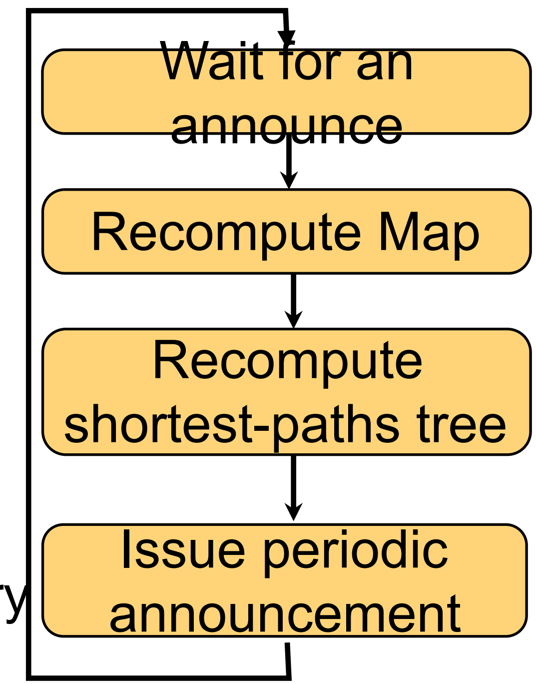
- I tell you all my “best” routes for all destinations that I know and you tell me yours.
- Build simplified topology from local perspective
- E.g. RIP (Routing Information Protocol)

Link State

- I announce to everyone about my links and the addresses I originate on each link and listen to everyone’s announcement.
- Build full topology
- E.g. OSPF (Open Shortest-Path First)

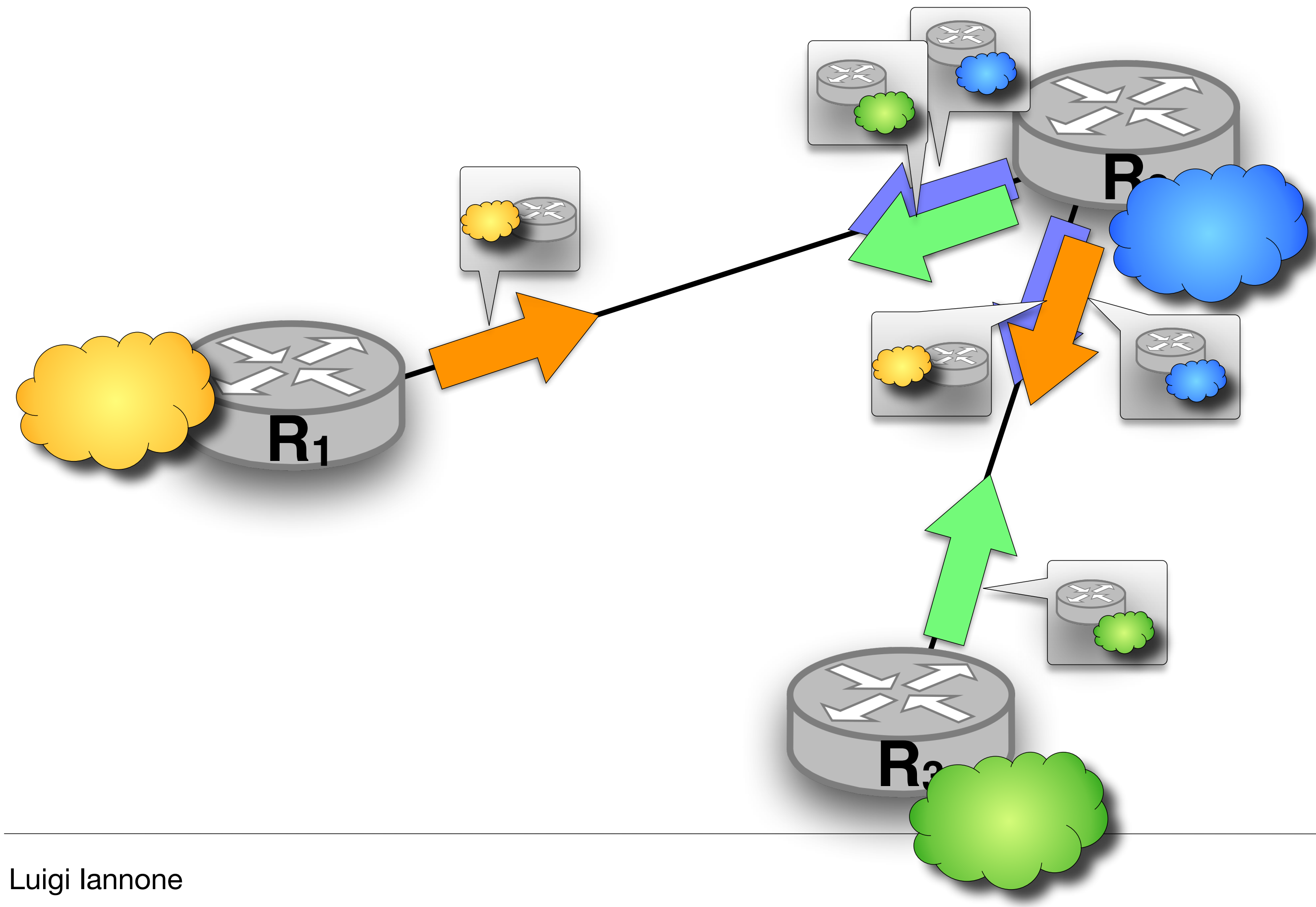
OSPF: Open Shortest-Path First Informally Defined

- I tell everyone about all my connections(links), with link up/down announcements
- I tell everyone about the addresses I originate on each link
- I listen to everyone else's link announcements
- I build a topology of every link (map)
- Then I compute the shortest path to every address prefix
- I assume (trust) that everyone else has



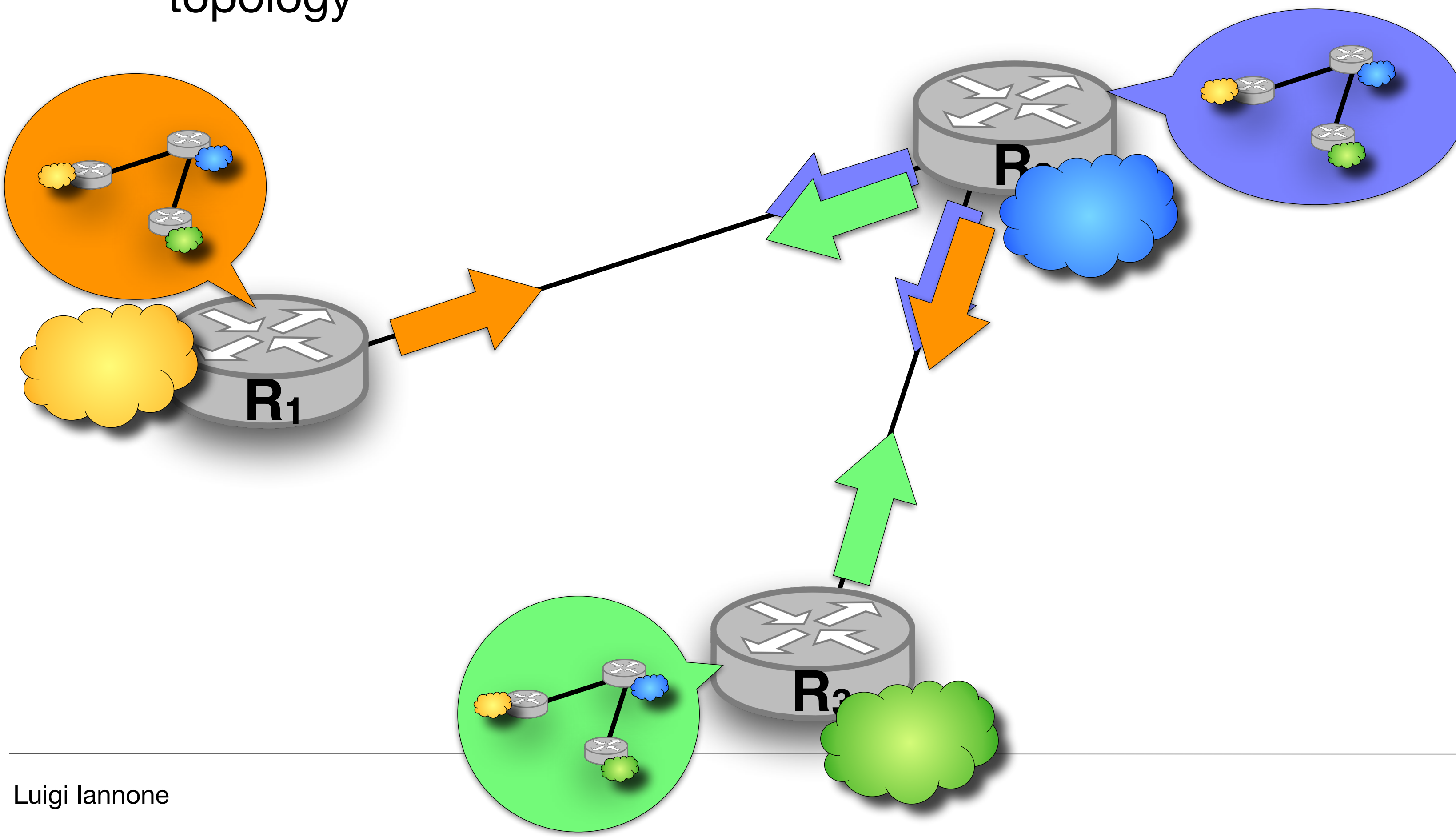
OSPF: Link-State Advertisements

- Routing information (reachability, link state) is broadcasted



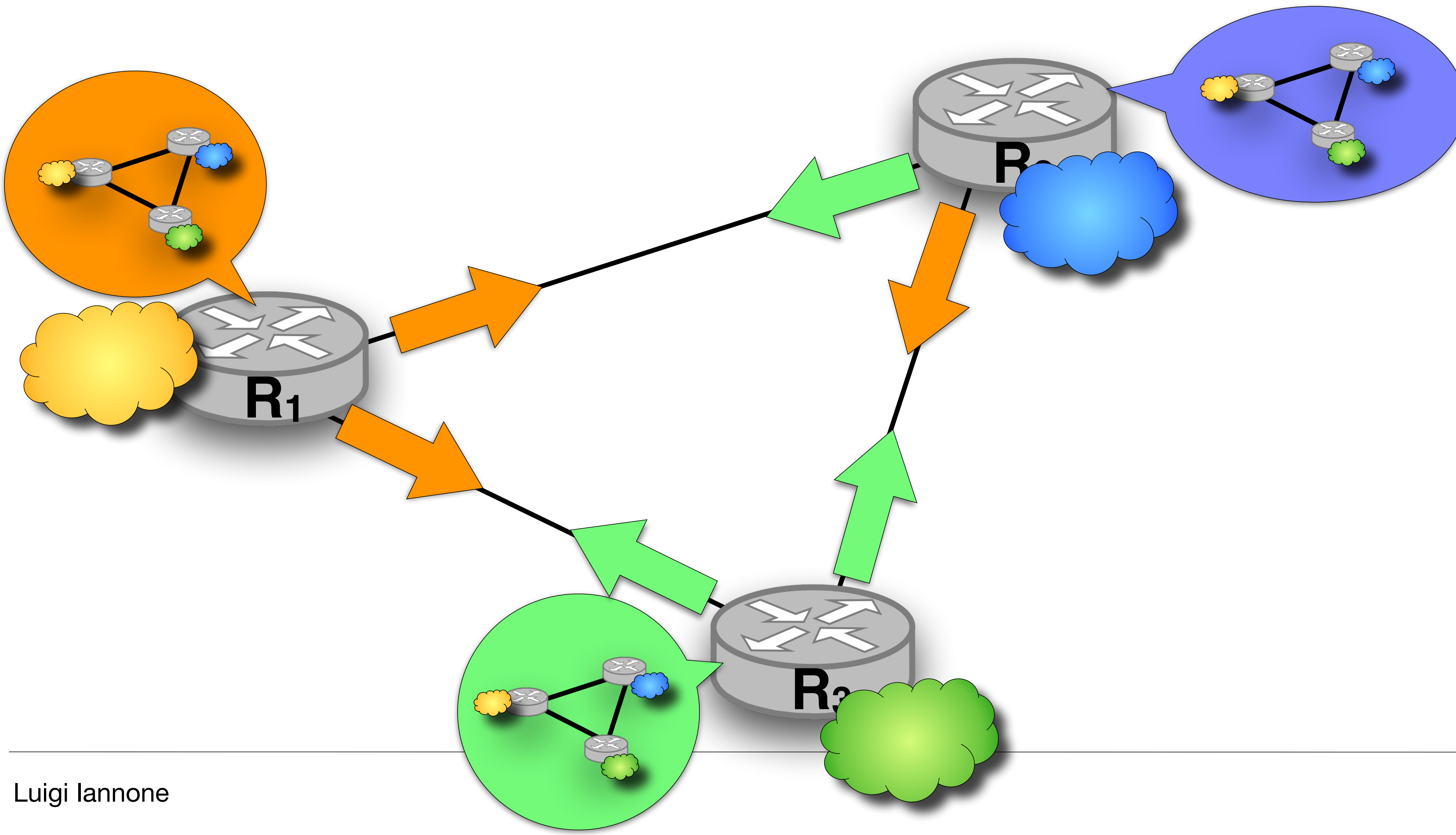
OSPF: Link-State Advertisements

- Routers build global view of the topology
- Routing table obtained by computing the shortest path on the topology



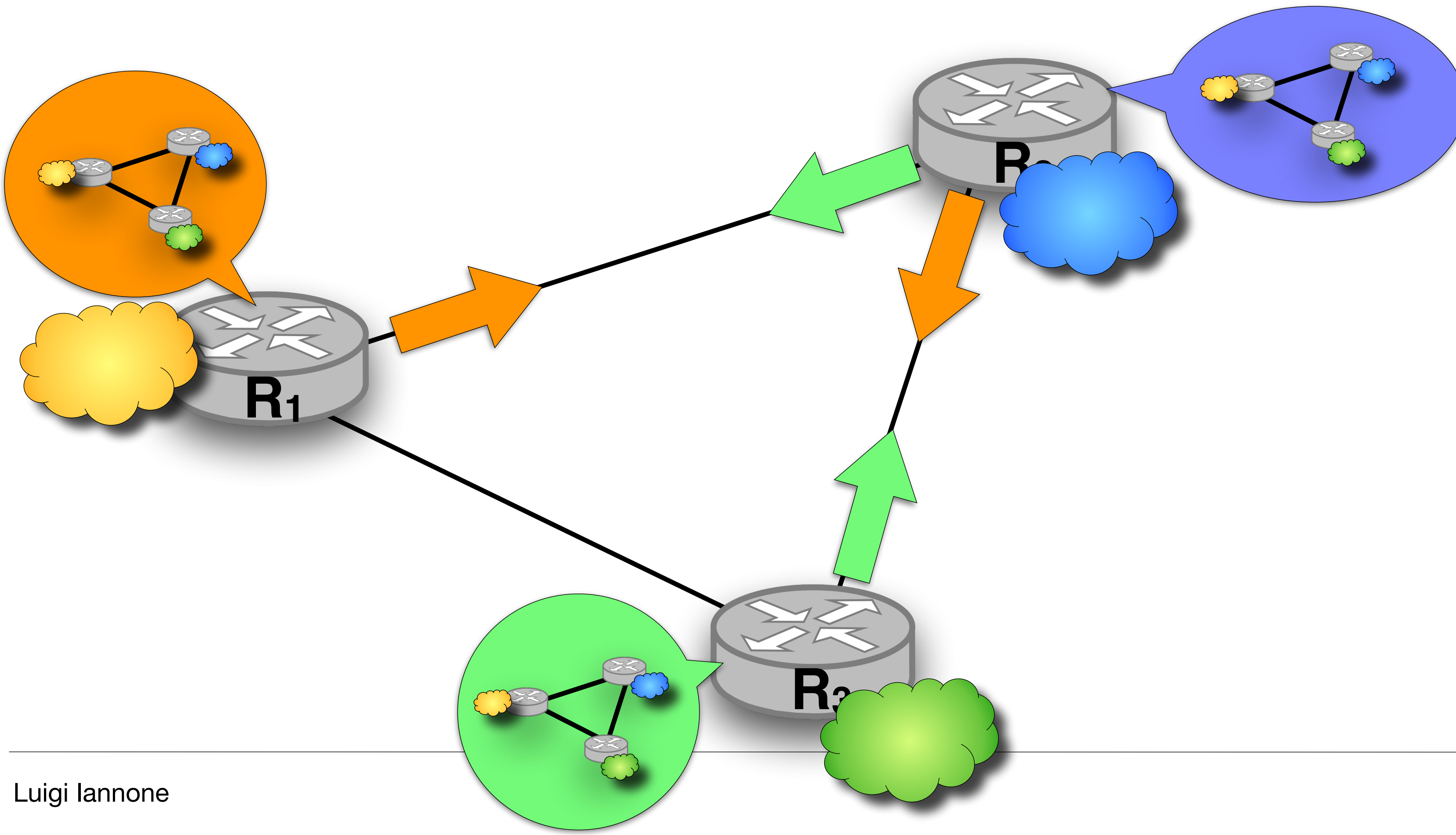
OSPF: Link-State Advertisements

- Convergence is rapid



OSPF: Link-State Advertisements

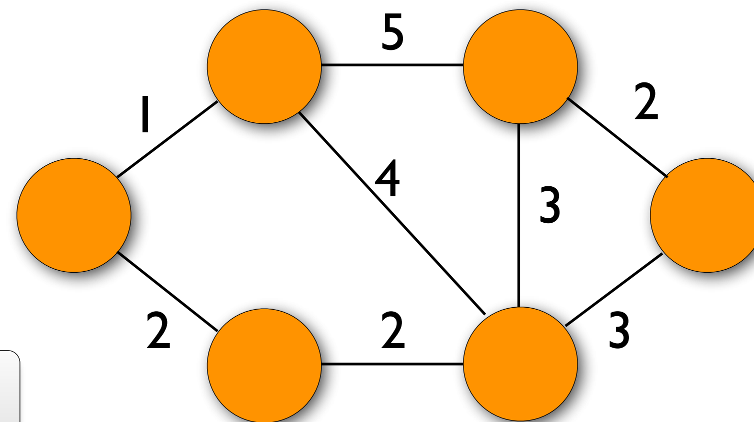
- Convergence is rapid even in case of failures



Shortest-Path Tree

Destinatio	Cost	Next-
R1	0	*
R2	1	link
R3	2	link
R5	4	
R4	6	
R6	7	R3

Path Cost (total cost to reach the target prefix)



After adding a destination update the Candidate Destination List and select the fist one

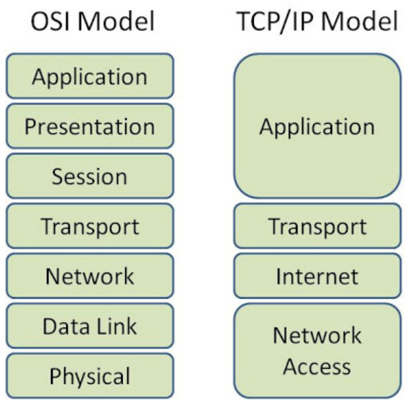
1. Candidate List = <R1,0>
2. Added Destination = <R1,0>; Candidate Destination List = <R2,1> <R3,2>
3. Added Destination = <R2,1>; Candidate Destination List = <R3,2> **<R5,5> <R4,6>**
4. Added Destination ... date Destination List = <R5,4> <R4,6>
5. Added Destination ... date Destination List = <R4,6> **<R6,7>**
6. Added Destination ... date Destination List = <R6,7>
7. Added Destination = <R6,7>; Candidate Destination List =
8. Done!

Target Prefix (here using only the router for simplicity& laziness)

<Target, Cost>

IPv4 over Ethernet*

(How to send IP packets (layer 3) through layer 2 frames?)



* Same principles apply to any MAC/PHY Layers

MAC vs. IP Adresses

- MAC (Medium Access Control) Address
  - 6 bytes (48 bits)
  - Static
  - Needs to be locally unique
  - flat
  - IEEE - Institute of Electrical and Electronics Engineers

`d4:9a:20:06:fe:6e`

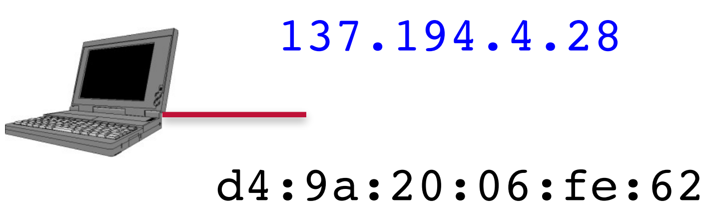
Manufacturer ID

Broadcast

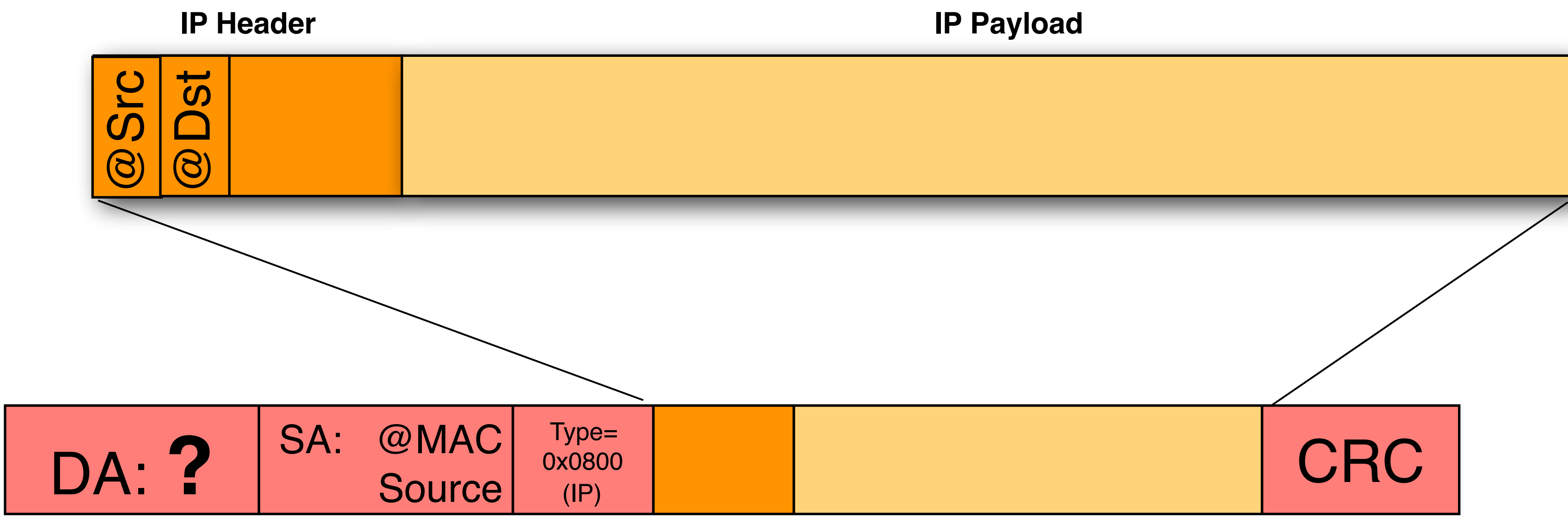
`ff:ff:ff:ff:ff:ff`

http://www.iana.org/assignments/ethernet-numbers

- IP Address
  - 4bytes (32bits) [16 bytes (128 bits) for IPv6]
  - dynamic
  - globally unique
  - Hierarchical
  - IETF/IANA/RIR/LIR

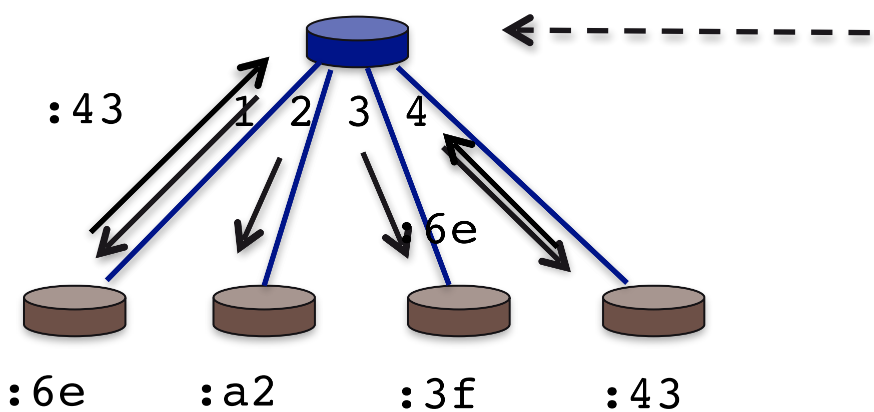


The need for an Address Resolution Protocol



Ethernet Switching

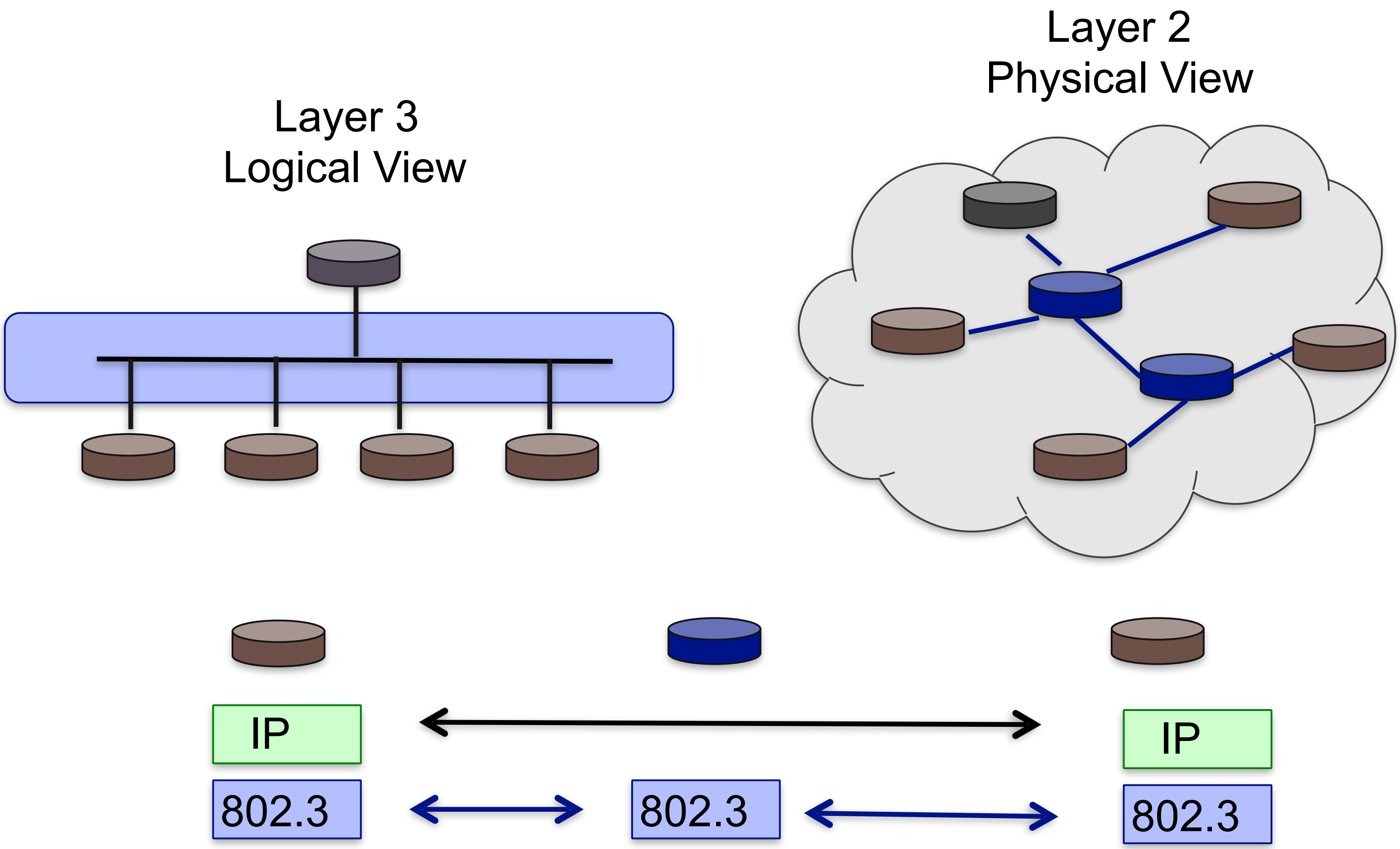
- Principle
  - Send to port where the destination is connected - if known
  - Send to all ports (except input port) otherwise



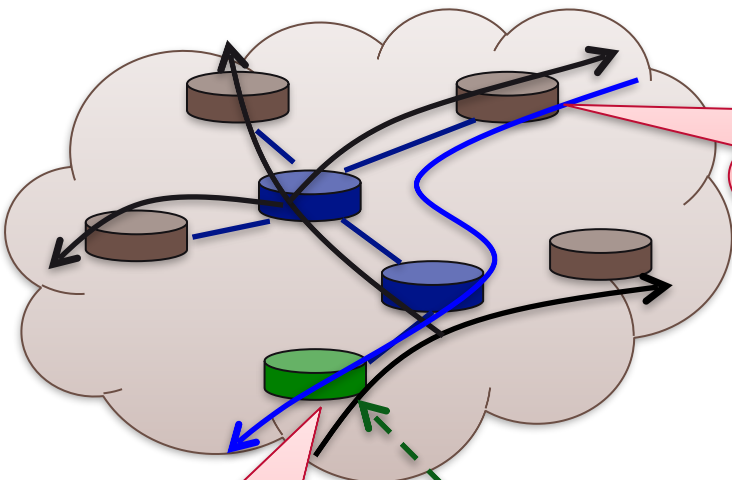
Switching Table

MAC	Port	TTL
:3f	*3*	*2:22*
:6e	*1*	*3:00*
:43	*4*	*3:00*

Ethernet Networks



Associating IP to MAC Addresses
ARP: Address Resolution Protocol



That would be me:
d4:9a:20:92:a2:5c

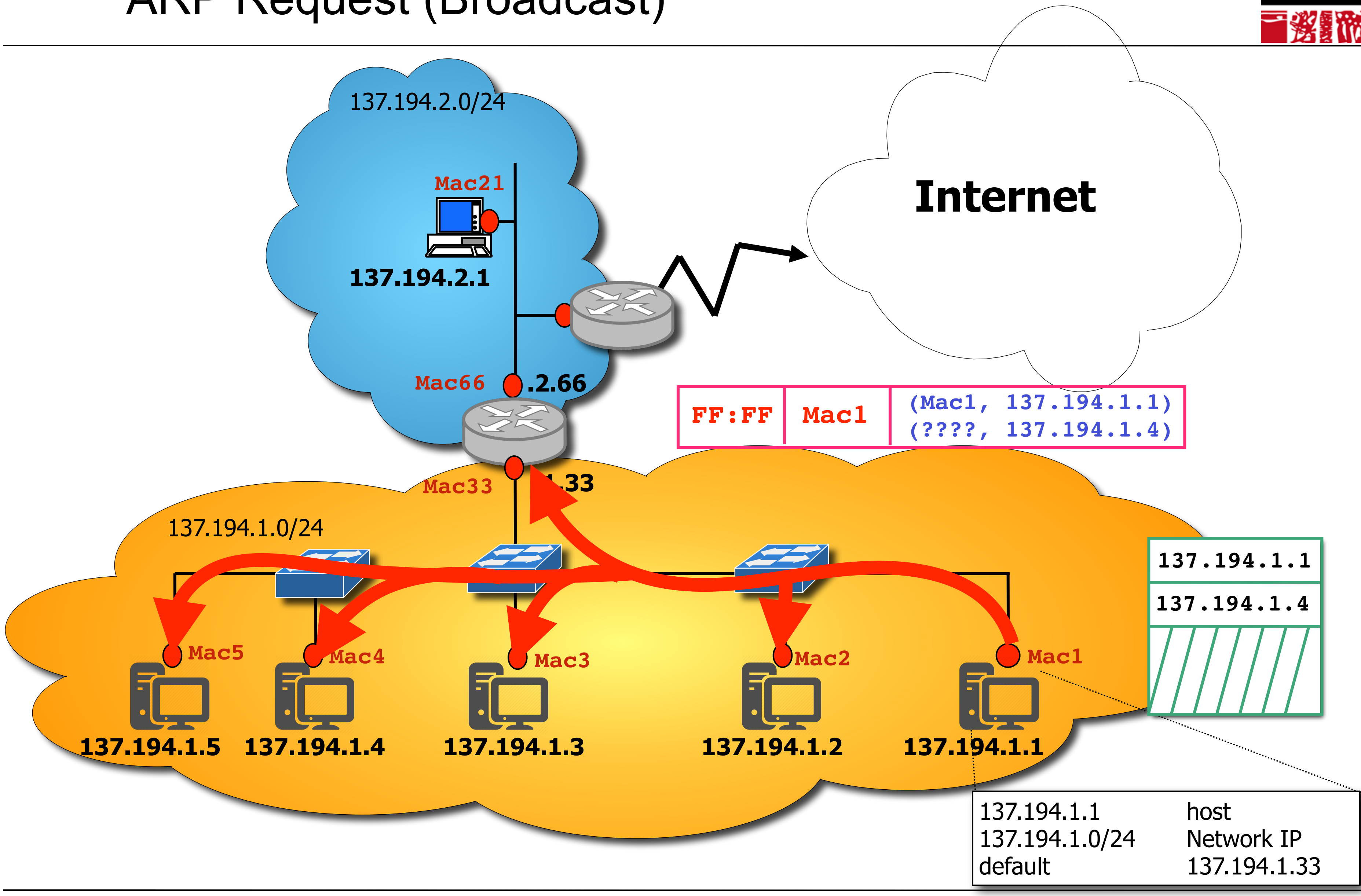
Who has the IP Address
137.194.2.56 ?
(FF:FF:FF:FF:FF:FF)

Table/Cache ARP

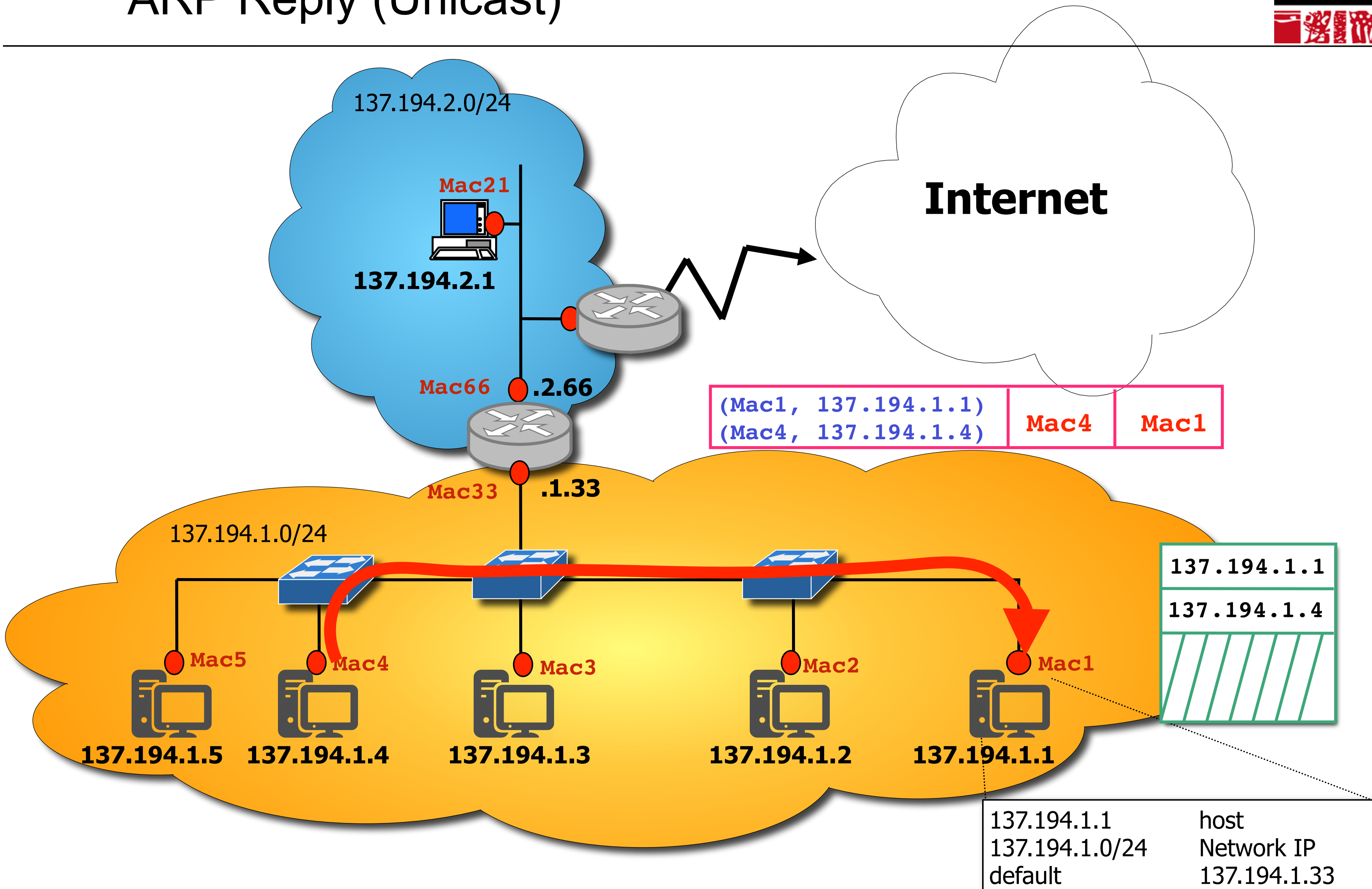
IP	MAC	TTL
137.194.2.26	d4:9a:20:06:fe:6e	7:32
137.194.2.3	e8:32:02:24:1f:63	7:38
137.194.2.56	d4:9a:20:92:a2:5c	7:45

Hand-On: arp -nl -i enX -a

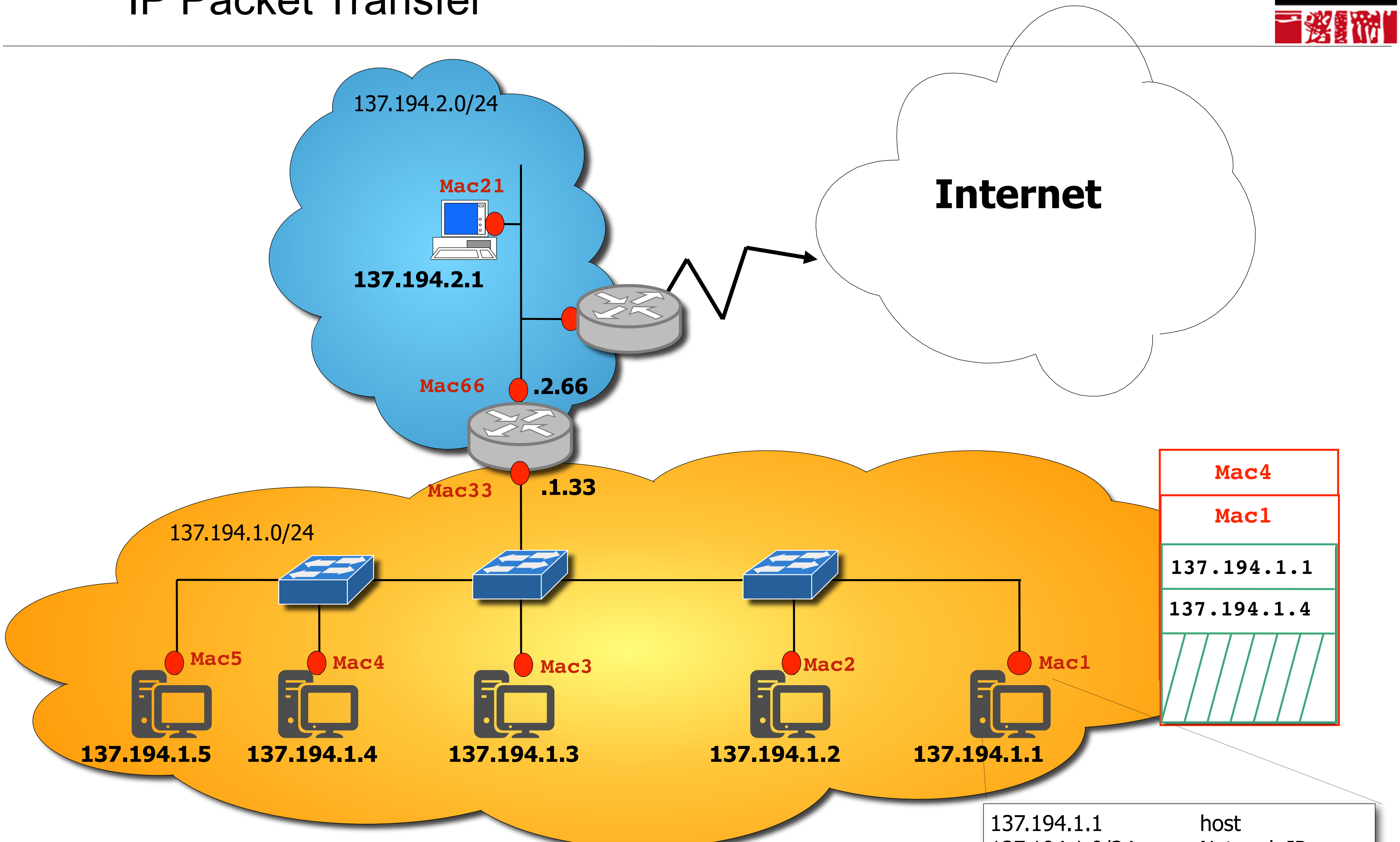
ARP Request (Broadcast)



ARP Reply (Unicast)

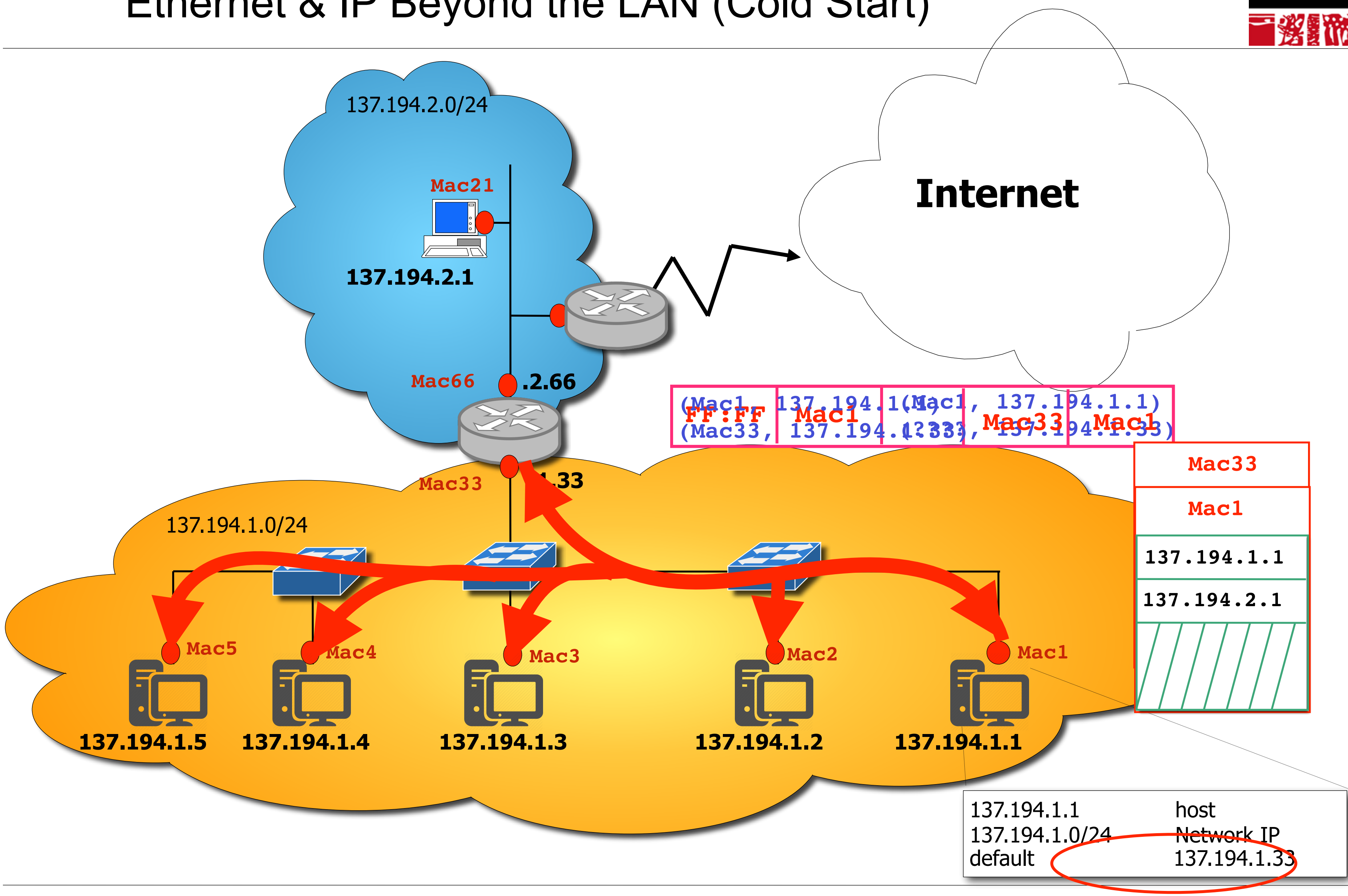


IP Packet Transfer

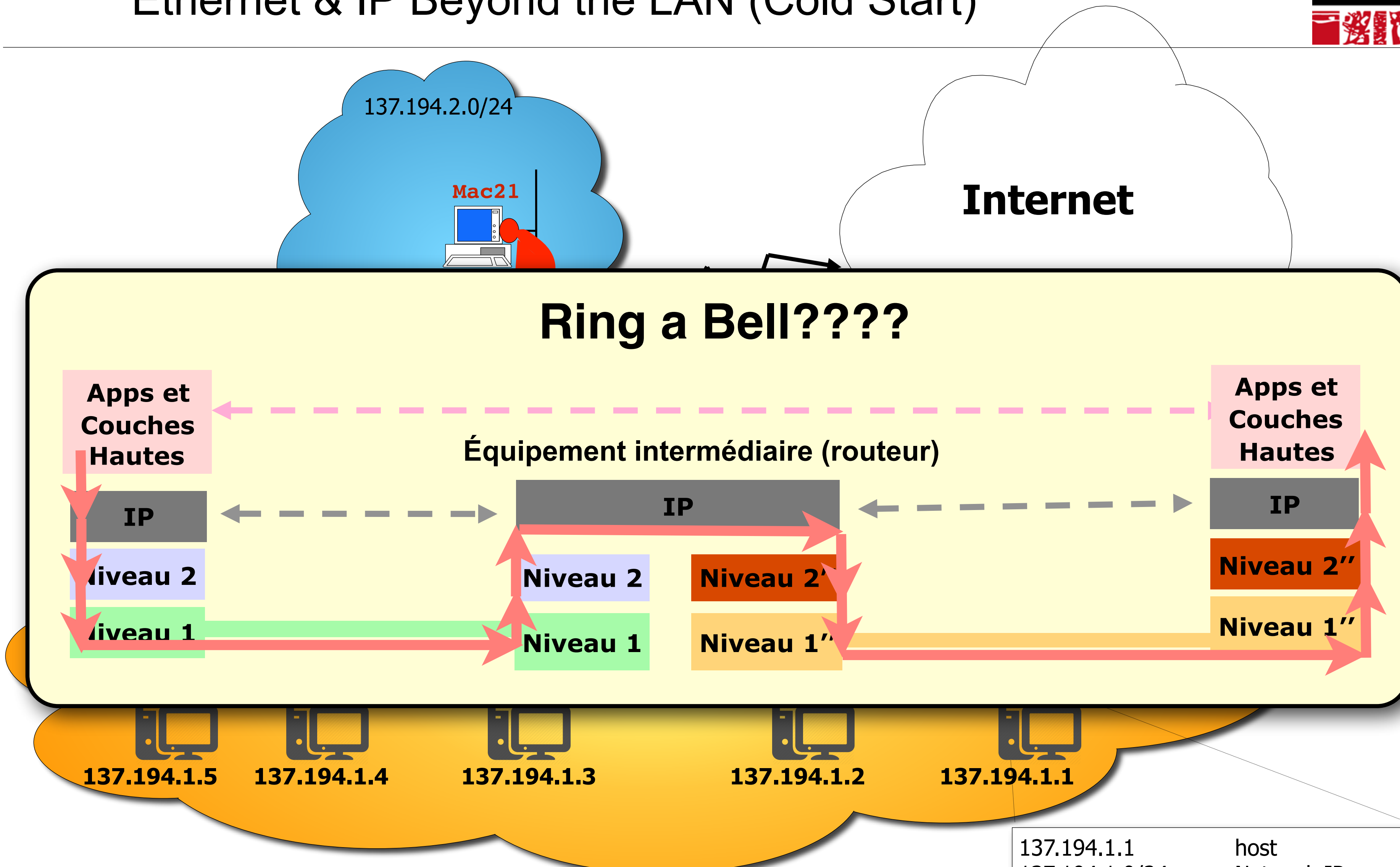


137.194.1.1	host
137.194.1.0/24	Network IP
default	137.194.1.33

Ethernet & IP Beyond the LAN (Cold Start)

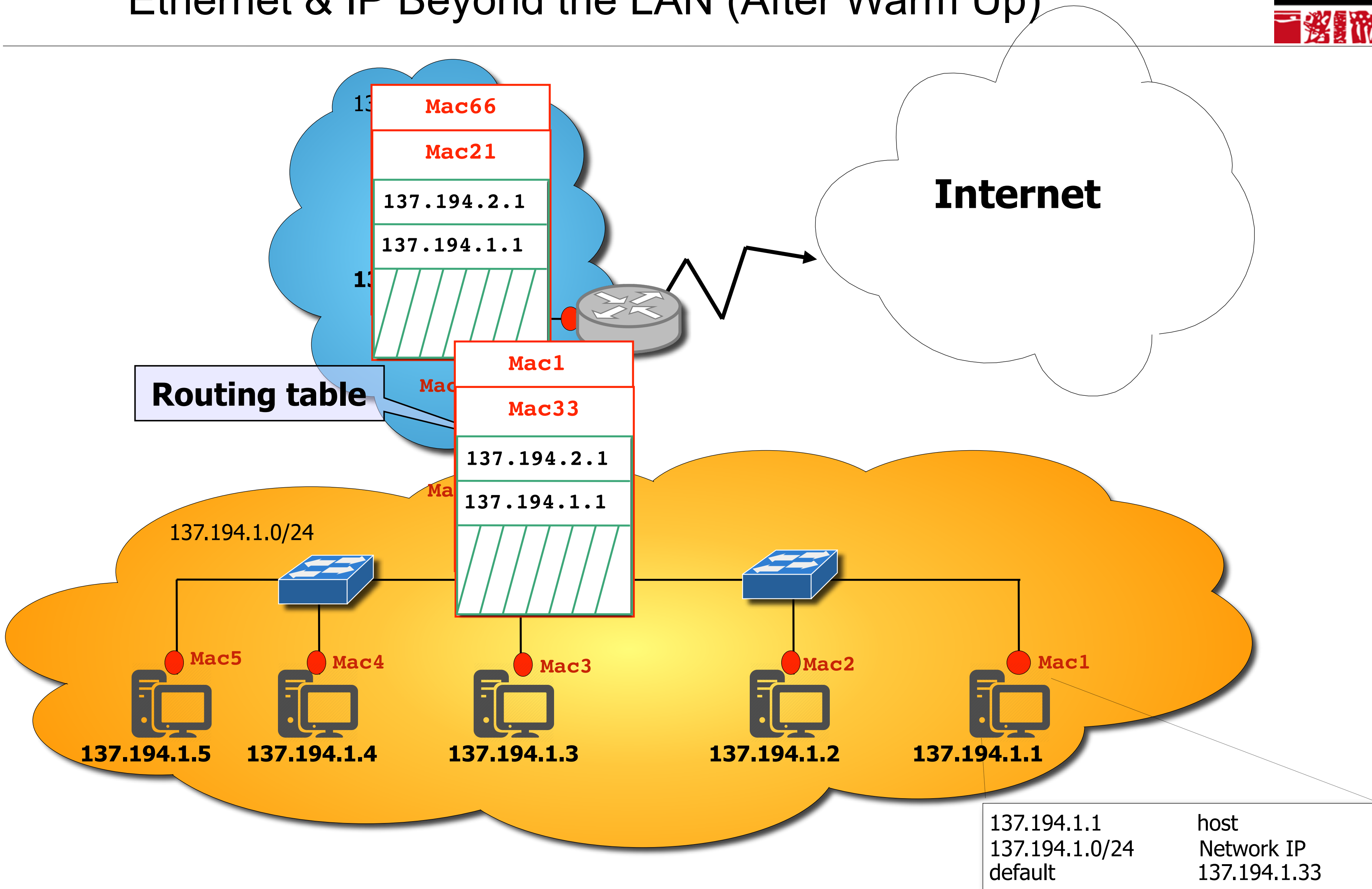


Ethernet & IP Beyond the LAN (Cold Start)



137.194.1.1	host
137.194.1.0/24	Network IP
default	137.194.1.33

Ethernet & IP Beyond the LAN (After Warm Up)



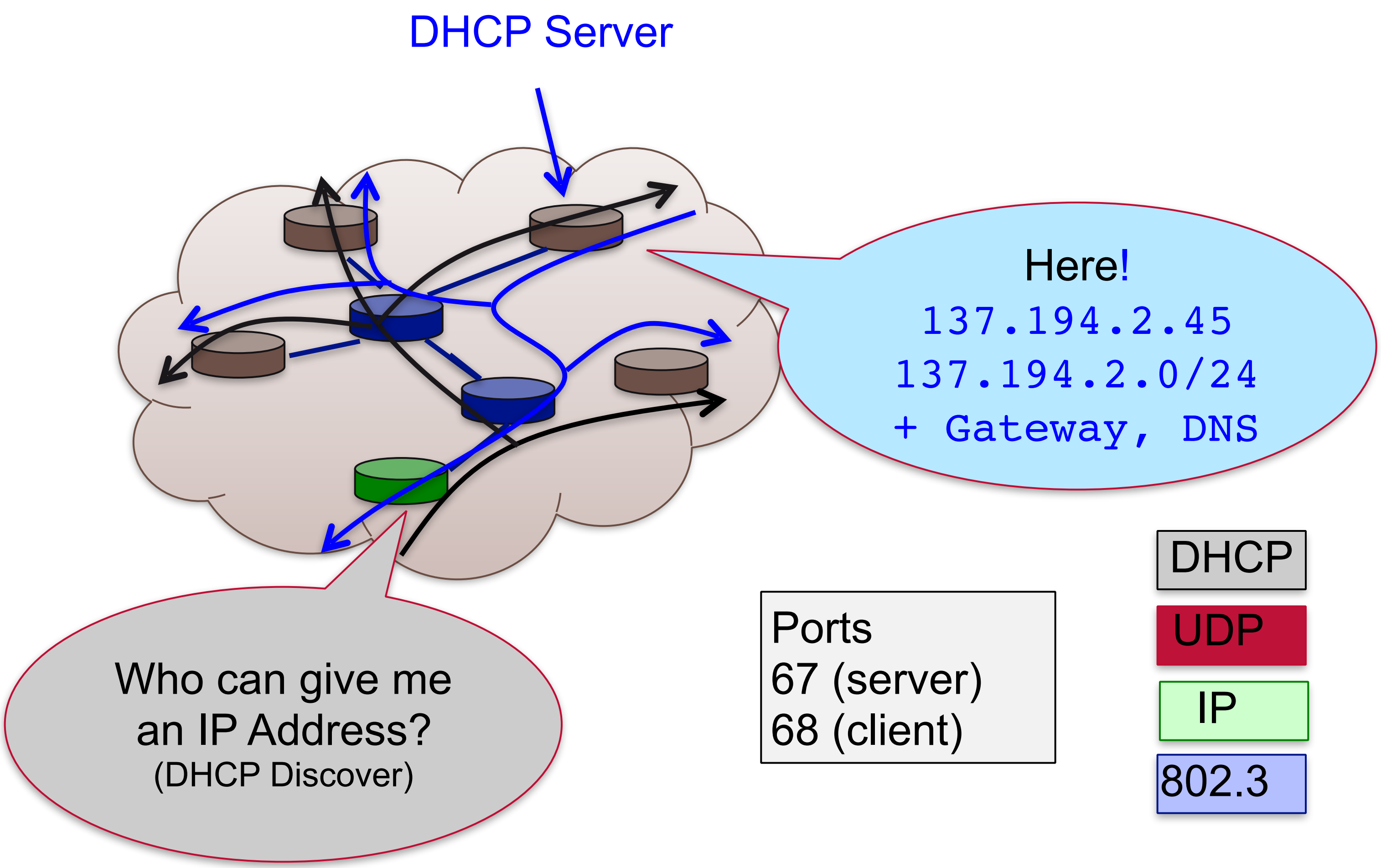
Dynamic Host Configuration Protocol (DHCP)

(I turn on my laptop, how do I get an IP address so that I can access my facebook account?)

DHCP: Dynamic Host Configuration Protocol

- From 1993
- Extensions:
  - Supports temporary allocation ("leases") of IP addresses
  - DHCP client can acquire all IP configuration parameters needed to operate
  - DHCP is the preferred mechanism for dynamic assignment of IP addresses

Obtention d'une adresse IP



DHCP Information

```
[dhcp164-03] ~ # ipconfig getpacket en3
op = BOOTREPLY
htype = 1
flags = 0
hlen = 6
hops = 0
xid = 2857836072
secs = 4
ciaddr = 0.0.0.0
yiaddr = 137.194.165.3
siaddr = 137.194.164.1
giaddr = 0.0.0.0
chaddr = 40:6c:8f:4:39:7c
sname =
file =
options:
Options count is 8
dhcp_message_type (uint8): ACK 0x5
server_identifier (ip): 137.194.164.1
lease_time (uint32): 0x15180
subnet_mask (ip): 255.255.254.0
router (ip_mult): {137.194.164.254}
domain_name_server (ip_mult): {137.194.2.34, 137.194.164.1, 137.194.164.5}
domain_name (string): enst.fr
end (none):
```

Internet Control Message Protocol (ICMP)

(How to control IP if something goes wrong or something special happens?)

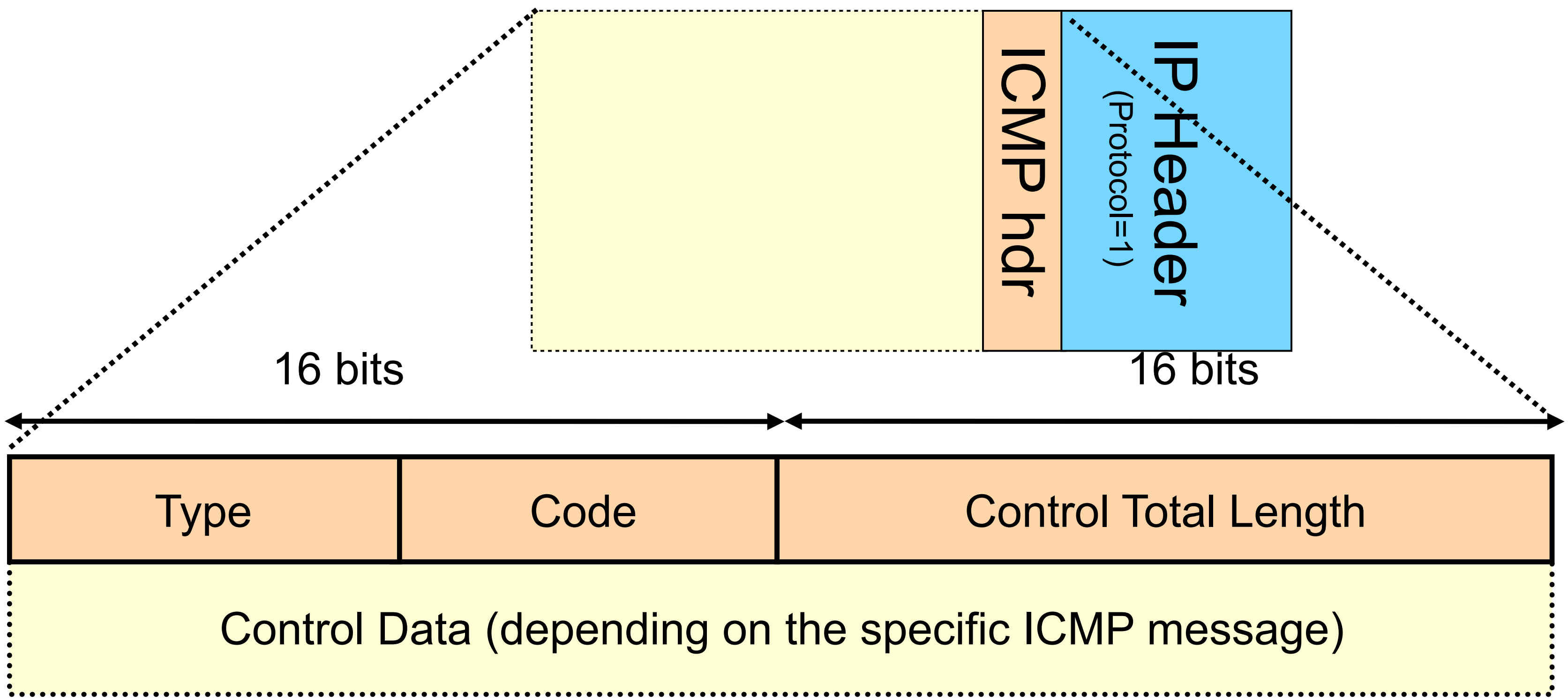
ICMP: Internet Control Message Protocol

- Defines Control messages for:
  - Error notification
  - Information request
  - Etc… (e.g., redirect)

	Type	Code	description
	0	0	echo reply (ping)
Destination Unreachable	3	0	dest. network unreachable
Destination Unreachable	3	1	dest host unreachable
Destination Unreachable	3	2	dest protocol unreachable
Destination Unreachable	3	3	dest port unreachable
Destination Unreachable	3	6	dest network unknown
Destination Unreachable	3	7	dest host unknown
	4	0	source quench (congestion control - not used)
	8	0	echo request (ping)
	9	0	route advertisement
	10	0	router discovery
	11	0	TTL expired
	12	0	bad IP header

- ICMP Messages are IP encapsulated
  - Conceptually at the same layer of IP
  - In practice this solution allows to re-use IP routing & forwarding
  - ICMP Protocol Number = 1
  - Content: type, code, and 8 first octets of the IP datagram causing the error

ICMP: Internet Control Message Protocol



ICMP: the `ping` utility

- Goal: verify reachability

- Mechanism: ICMP Echo Request/Echo Reply messages
  - Echo Request: ICMP Type 8 Code 0
  - Echo Reply: ICMP Type 0 Code 0

- [casellas@cassoulet casellas]$ **ping morgane.enst.fr**
- PING **morgane.enst.fr (137.194.160.31)** from 137.194.162.64 : 56(84) bytes of data.
- 64 bytes from morgane.enst.fr (137.194.160.31): icmp_seq=0 ttl=254 time=326 usec
- 64 bytes from morgane.enst.fr (137.194.160.31): icmp_seq=1 ttl=254 time=317 usec
- 64 bytes from morgane.enst.fr (137.194.160.31): icmp_seq=2 ttl=254 time=324 usec
- 64 bytes from morgane.enst.fr (137.194.160.31): icmp_seq=3 ttl=254 time=309 usec

- --- morgane.enst.fr ping statistics ---
- 4 packets transmitted, 4 packets received, 0% packet loss
- round-trip min/avg/max/mdev = 0.309/0.319/0.326/0.006 ms

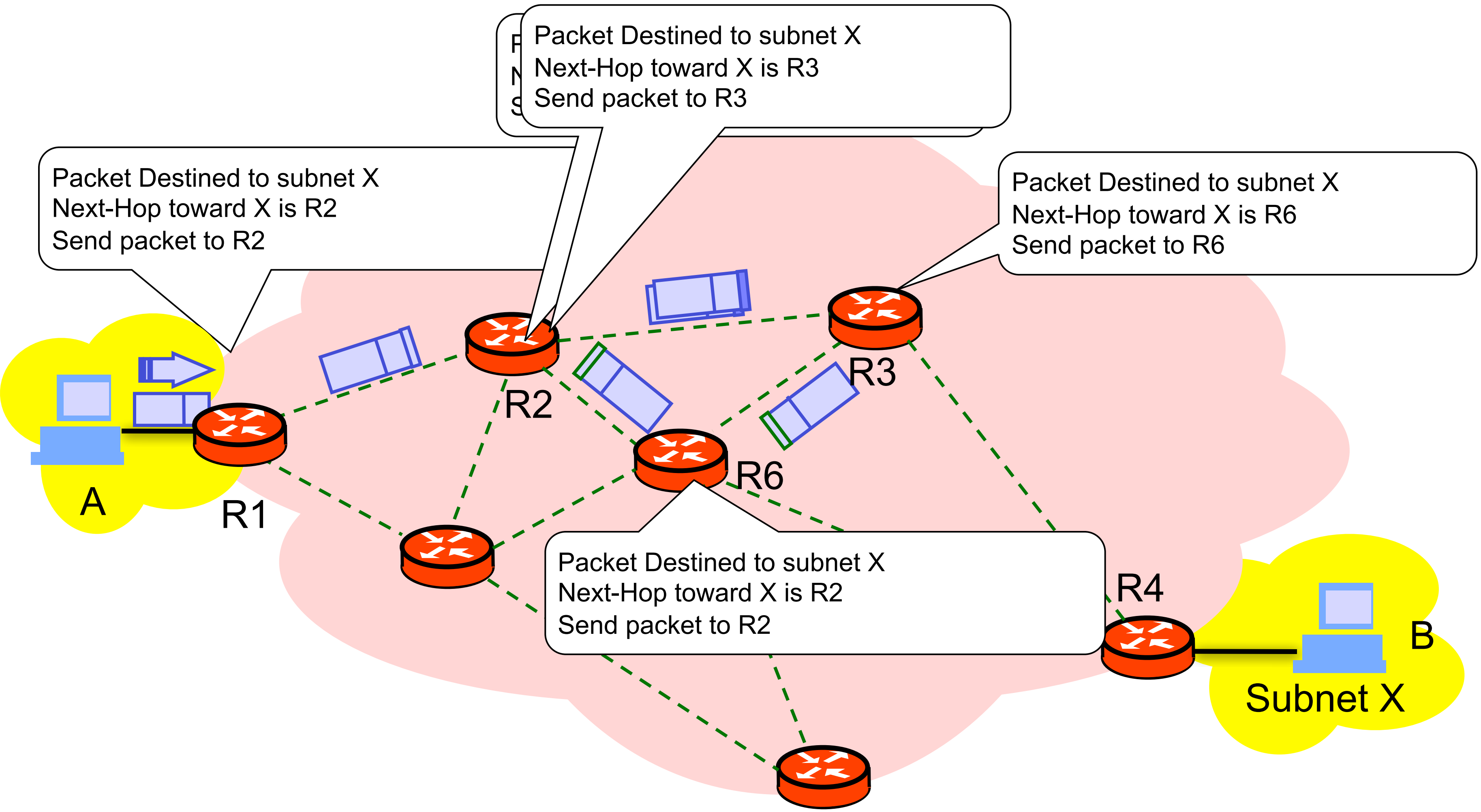
ICMP: the traceroute utility

- Goal: provides the list of IP addresses forming a path toward a destination

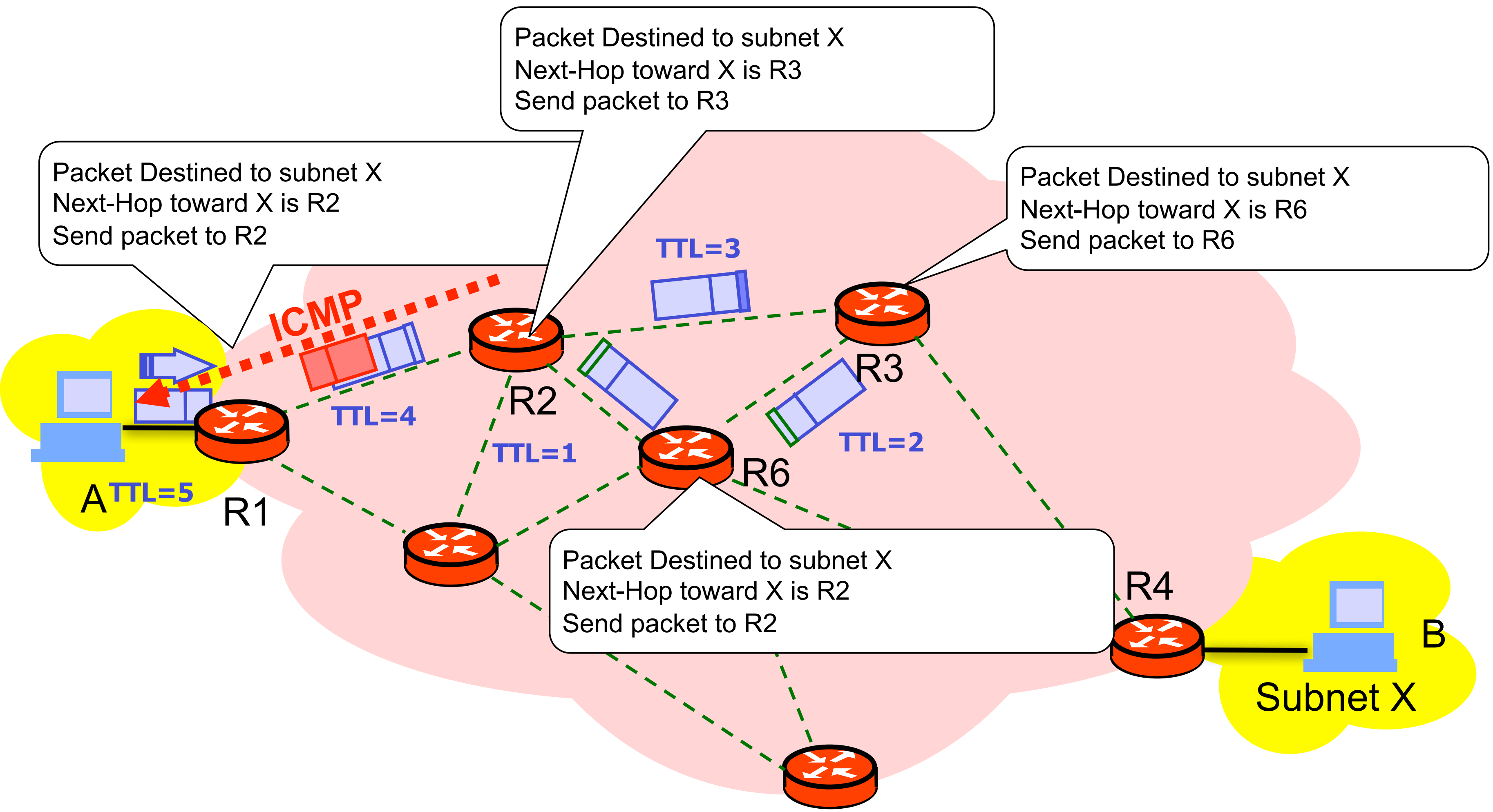
- Uses the IP TTL field and attempts to elicit an ICMP TIME_EXCEEDED response from each gateway along the path

  - First packet sent with TTL = 1
    - First gateway sends back an ICMP message
  - Second packet sent with TTL = 2
    - Second gateway sends back an ICMP message
  - .....

Loops happen……

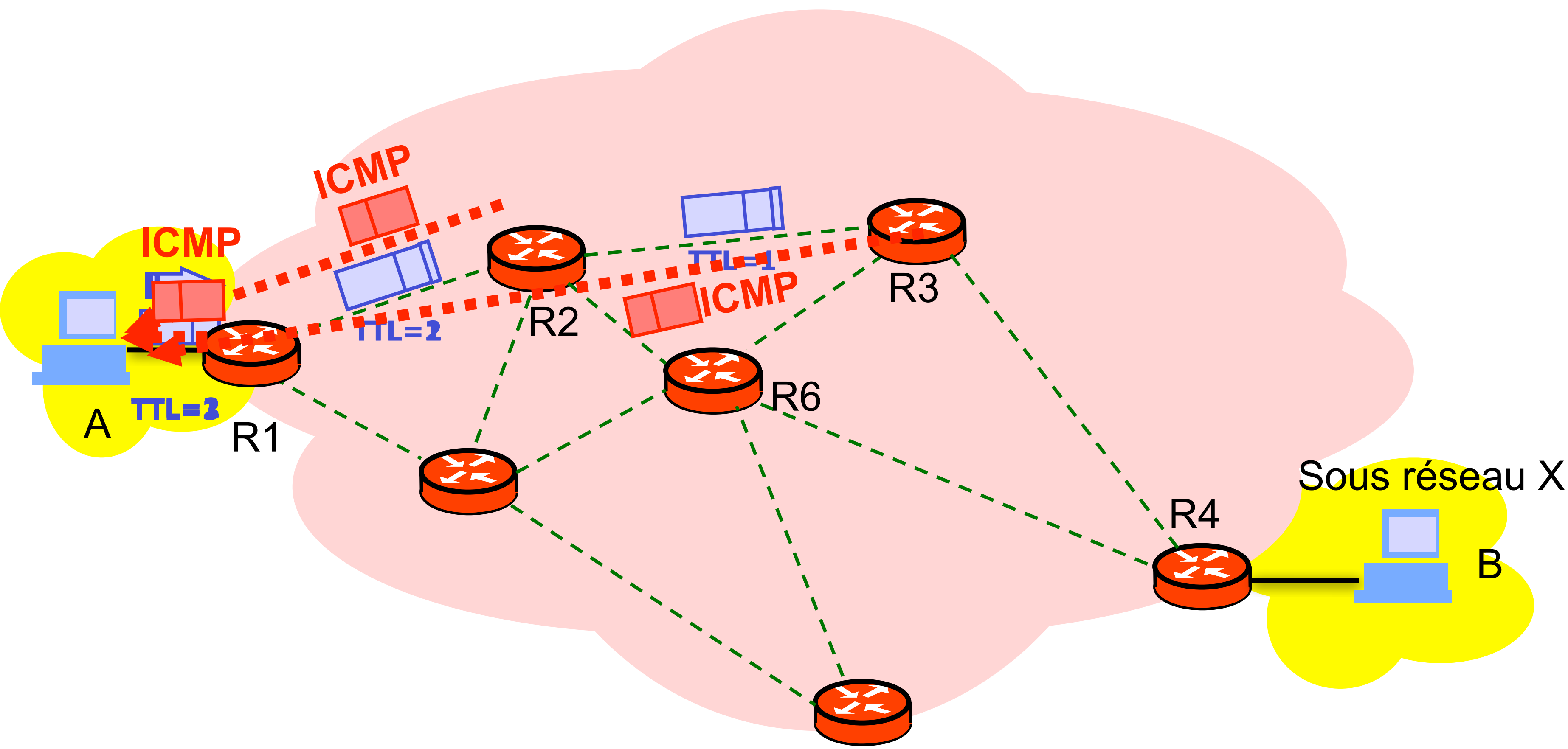


TTL Exceeded Original usage



traceroute in practice

Path: R1 R2 R3



traceroute in reality

```
• [casellas@cassoulet casellas]$ traceroute www.gnu.org
• traceroute to www.gnu.org (199.232.41.10), 30 hops max, 38 byte packets
•  1  benelos (137.194.162.3)  0.354 ms  0.167 ms  0.157 ms
•  2  enst-free (137.194.2.253)  0.385 ms  0.360 ms  0.285 ms
•  3  telehouse-3.routers.proxad.net (213.228.3.3)  2.126 ms  2.152 ms  1.422 ms
•  4  blackd-cbv-3-a6.routers.proxad.net (213.228.3.25)  2.330 ms  3.109 ms  5.154 ms
•  5  prs-b1-geth6-2.telia.net (213.248.70.253)  12.499 ms  1.732 ms  2.419 ms
•  6  prs-bb2-pos0-3-0.telia.net (213.248.70.9)  3.444 ms  3.969 ms  3.076 ms
•  7  ldn-bb2-pos0-2-0.telia.net (213.248.64.165)  8.104 ms  8.455 ms  8.256 ms
•  8  nyk-bb2-pos2-3-0.telia.net (213.248.65.38)  80.696 ms  80.099 ms  80.444 ms
•  9  nyk-i2-pos1-0.telia.net (213.248.82.18)  82.625 ms  81.931 ms  83.116 ms
• 10  205.198.19.117 (205.198.19.117)  217.573 ms  211.313 ms  215.262 ms
• 11  205.198.0.82 (205.198.0.82)  218.611 ms  207.738 ms  207.255 ms
• 12  res1-avici1-cl-qcy1.gnaps.net (199.232.42.110)  233.728 ms  237.814 ms  234.657 ms
• 13  qcy1-avici1-g1-5-4.gnaps.net (199.232.42.113)  213.301 ms  213.678 ms  233.165 ms
• 14  qcy1-ar1-g1-0.gnaps.net (199.232.42.114)  224.329 ms  212.461 ms  217.087 ms
```

Congrats
You are now an expert of IPv4 (and its surroundings)

Anything you want to discuss???