NETWORKING – NETWORK SERVICES

TCP/IP model

INTRODUCTION

- In this lecture, you will learn on how to install and manage some network services that run on top of a network that is already functionning on TCP/IP model

- **What is a Network service**

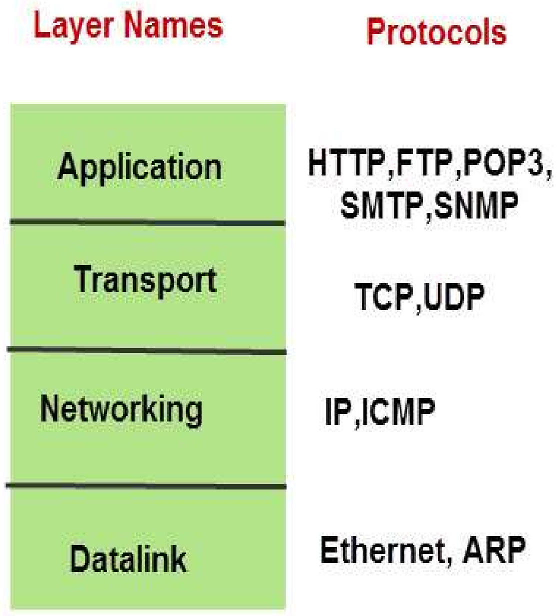
Wikipedia Definition:

« a **network service** is an application running at the network application layer and above, that provides data storage, manipulation, presentation, communication or other capability which is often implemented using a client–server or peer-to-peer architecture based on application layer network protocols »

TCP/IP MODEL

- It was designed to describe the functions of the communication system.
- the communication procedure is based on standard protocols

- It is responsible for node-to-node communication and controls user-interface specifications.
- It is responsible for end-to-end communication and error-free delivery of data.
- It defines the protocols which are responsible for logical transmission of data over the entire network.
- It looks out for hardware addressing and the protocols present in this layer allows for the physical transmission of data.

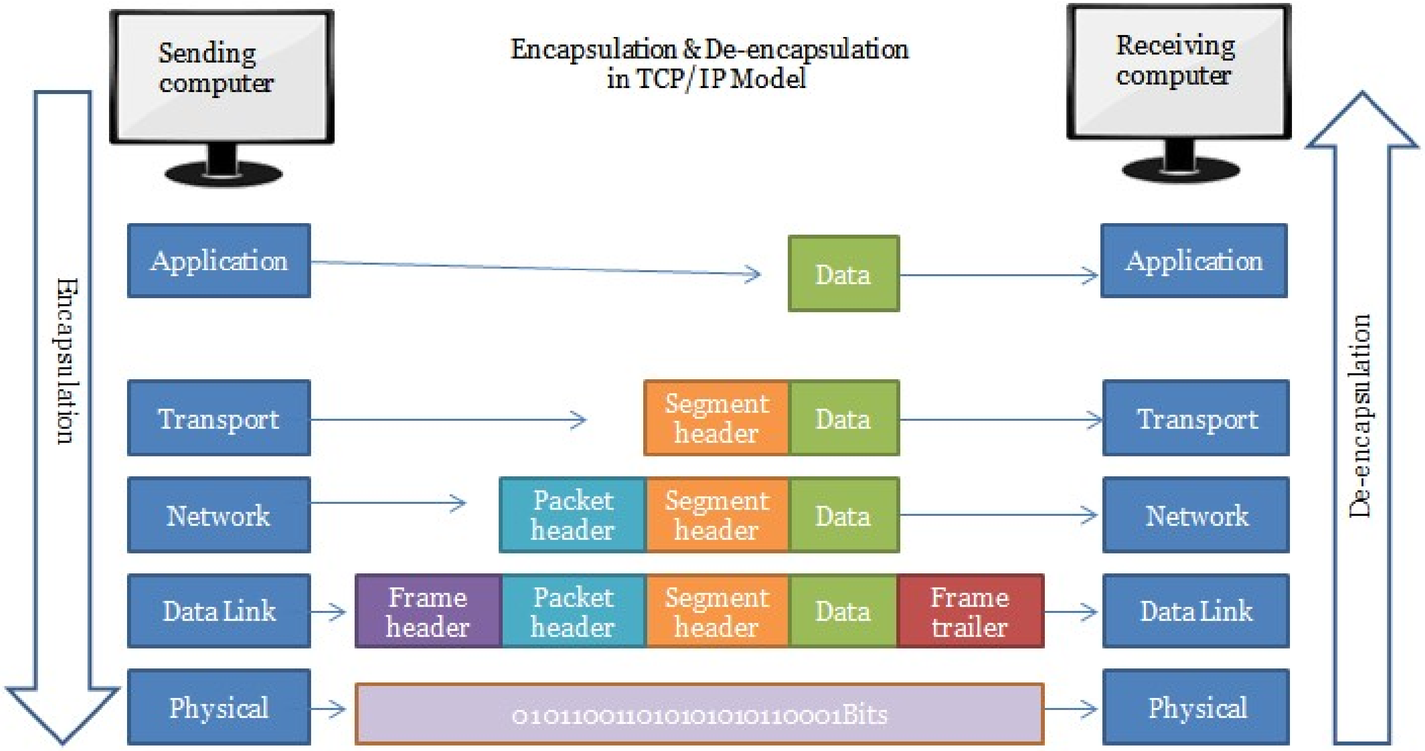


Layer Names	Protocols
Application	HTTP,FTP,POP3, SMTP,SNMP
Transport	TCP,UDP
Networking	IP,ICMP
Datalink	Ethernet, ARP

TCP/IP Networking Model

TCP/IP MODEL - ENCAPSULATION

- The TCP/IP model respects layered communication
- Only the corresponding layer that can process the content of the IP packet



TRANSPORT LAYER

- The two main protocols present in this layer are :
- **Transmission Control Protocol (TCP)** –

  It is known to provide reliable and error-free communication between end systems. It is connection-oriented protocol

  It performs sequencing and segmentation of data.

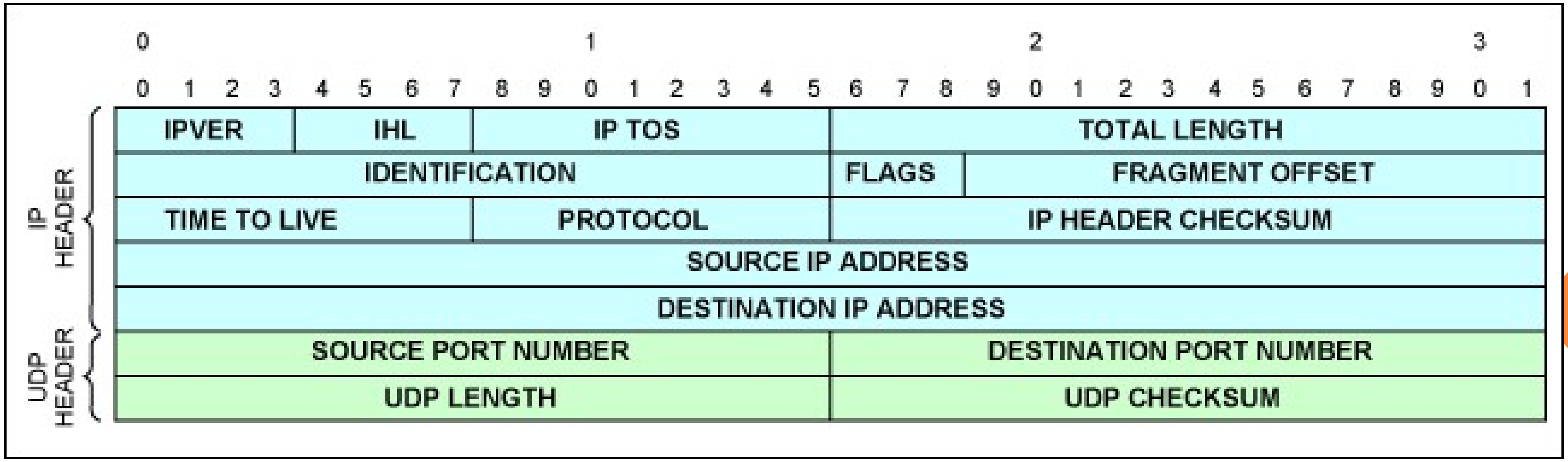
  It also has acknowledgment feature and controls the flow of the data through flow control mechanism.

- **User Datagram Protocol (UDP)** –

  On the other hand does not provide any such features. It does not require reliable transport. Unlike TCP, UDP is connectionless and not reliable

UDP – User Datagram Protocol

- UDP provides an unreliable datagram service
  - Packets may be lost or delivered out of order
  - Message split into datagrams, user sends datagrams as packets on network layer
  - The communication is fast : no data control
  - Full duplex Application must deal with lost packets
  - Used in streaming and multicast communications



<table>
<tr><td></td><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>1</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>2</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>3</td><td></td></tr>
<tr><td></td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>0</td><td>1</td></tr>
<tr><td rowspan="5">IP HEADER</td><td colspan="4">IPVER</td><td colspan="4">IHL</td><td colspan="8">IP TOS</td><td colspan="16">TOTAL LENGTH</td></tr>
<tr><td colspan="16">IDENTIFICATION</td><td colspan="3">FLAGS</td><td colspan="13">FRAGMENT OFFSET</td></tr>
<tr><td colspan="8">TIME TO LIVE</td><td colspan="8">PROTOCOL</td><td colspan="16">IP HEADER CHECKSUM</td></tr>
<tr><td colspan="32">SOURCE IP ADDRESS</td></tr>
<tr><td colspan="32">DESTINATION IP ADDRESS</td></tr>
<tr><td rowspan="2">UDP HEADER</td><td colspan="16">SOURCE PORT NUMBER</td><td colspan="16">DESTINATION PORT NUMBER</td></tr>
<tr><td colspan="16">UDP LENGTH</td><td colspan="16">UDP CHECKSUM</td></tr>
</table>

TCP – Transmission Control Protocol

- Reliable Transport in connected Mode
  - Connection establishment procedure : connection oriented between the server and the client
- Reliability: making sure that everything that happens is
  - exactly what was sent
  - Data must not be lost: lossless
  - Data must not be subject to errors: error-free
    - An error = a bit that changes its value during the transfer
  - The data must arrive in order
  - The data must not arrive in duplicate: without duplication

TCP Protocol

- a virtual circuit connection is established before data is exchanged: **call + negotiation + Connection**.
  - The process is called Three Ways Handshake
- A connection is happen between two endpoints
- A connection endpoint = pair (IP address, port)
    - **Example of a connection: ((12.3.12.1, 1034), (192.4.67.2, 80))**
- The connection implementation is done in two steps:
  - one application (end) performs a **passive opening** by indicating that it is accepting an incoming connection,
  - another application (end) performs an **active opening** to request the establishment of the connection.

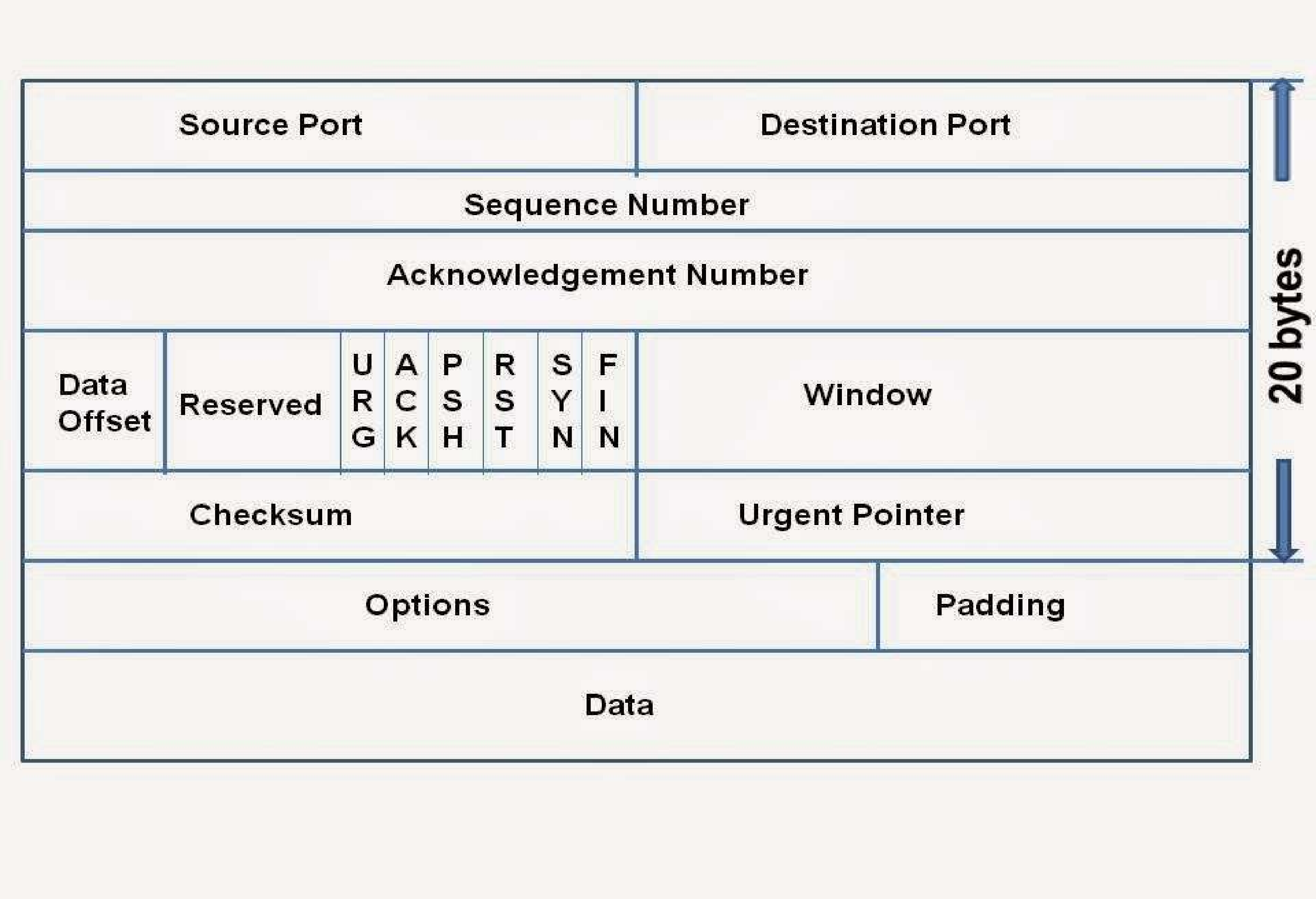
FLOW CONTROL

- **Segmentation, flow control:**
  - TCP divides this data stream into segments using a windowing mechanism.
  - A segment is transmitted in an IP datagram.
- **Acknowledging messages:**
  - Unlike UDP, TCP guarantees the arrival of messages, i.e. in case of loss, both ends are notified.
  - This concept is based on message acknowledgement techniques:
    - when a source S sends a message Mi to a destination D, S waits for an acknowledgement Ai from D before sending the next message Mi+1. If the acknowledgement Ai does not reach S, S considers after a timeout that the message is lost and retransmits Mi :

EXAMPLE

- 3-segments to transmit from A to B, if the bytes from 1 to1024 are **received** correctly, then B sends an **ACK with the value 1025**.
- If the next segment containing bytes from **1025 to 2048 is lost** and B receives the segment of bytes 2049 to 3072 correctly first, **B will not send a positive** acknowledgement for **this third segment**.
- Only when B receives the second segment it will be able to send an ACK with the value 3073. which A will interpret it as a positive acknowledgement of the of the last two segments it has sent.
- This is a cumulative

TCP HEADER



Source Port | Destination Port

Sequence Number

Acknowledgement Number

Data Offset | Reserved | URG | ACK | PSH | RST | SYN | FIN | Window

Checksum | Urgent Pointer

Options | Padding

Data

TCP HEADER

- **Source and destination ports:** identify the connection between the source and the destination. These ports, associated with IP addresses, represent a unique identification of each connection
- **Sequence number :** the sequence number gives the position of the segment in the DATA stream sent by the sender.
- **Acknowledgement number :** the next NS sequence number expected by the sender of this acknowledgment. Acknowledges implicitly the NS-1, NS-2, etc. bytes.

- **Window:** Is used for flow control using the sliding window method. This is the amount of data that the sender of this segment is able to receive; this is mentioned in each segment (data or acknowledgement).

FLAGS

- flags are used to indicate a particular state of connection or to provide some additional useful information like troubleshooting purposes or to handle a control of a particular connection.

- Most commonly used flags are **“SYN”, “ACK” and “FIN”**. Each flag corresponds to 1 bit information.

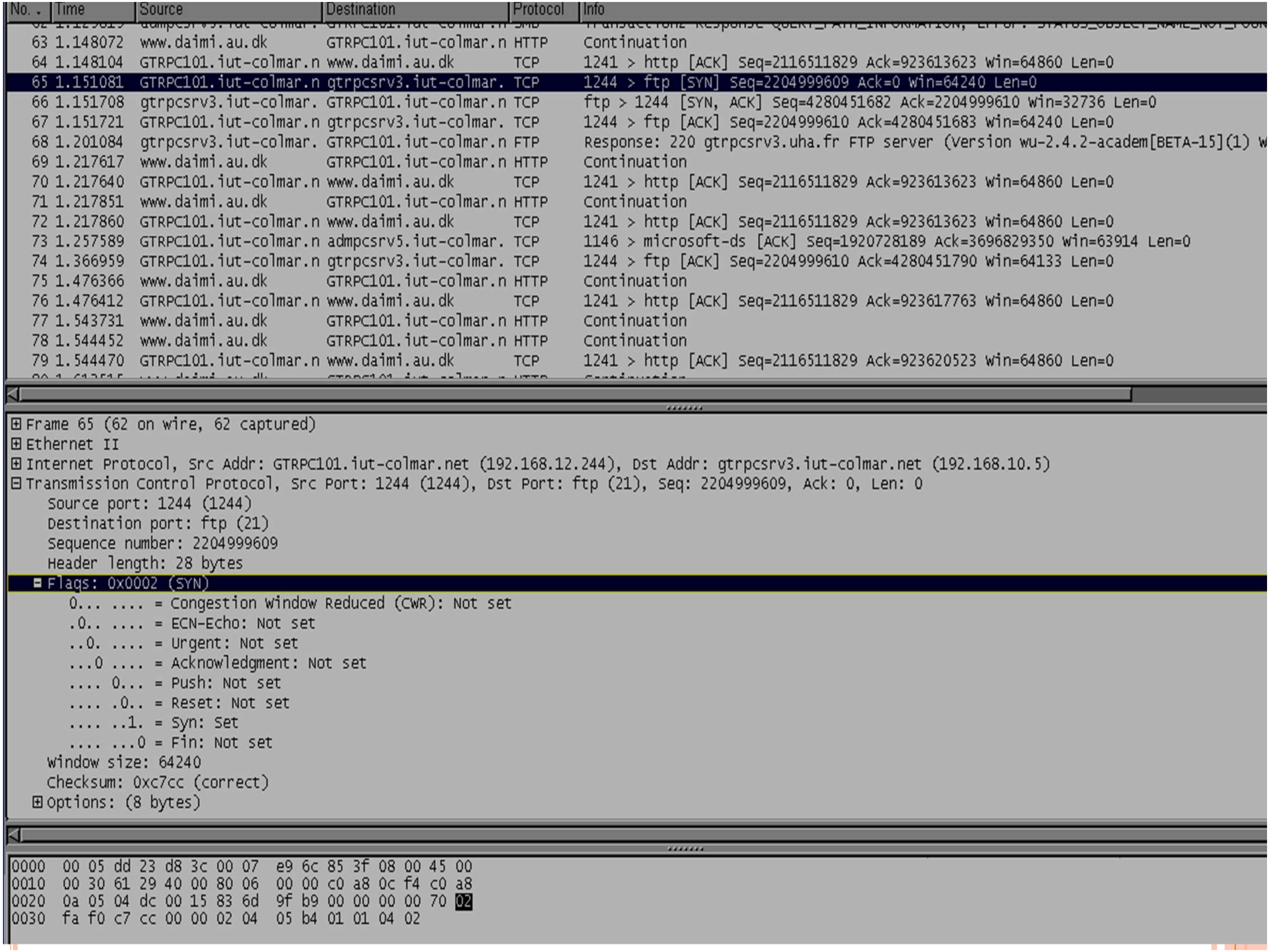
TYPES OF FLAGS

- **Synchronization (SYN)** – It is used in first step of connection establishment phase. This is used for synchronizing sequence number i.e. to tell the other end which sequence number they should except.
- **Acknowledgement (ACK)** – It is used to acknowledge packets which are successful received by the host.
- **Finish (FIN)** – It is used to request for connection termination. This is the last packet sent by sender.
- **URG :** the urgent data pointer (example : remote login), the data is sent without delay, the received data is given without delay.

Types of Flags

- **URG :** the urgent data pointer (example : remote login), the data is sent without delay, the received data is given without delay.

- **PSH :** informs the receiving host that the data should be pushed up to the receiving application immediately
- **RST :** used by the endpoint to indicate to the other end that it must reset the connection
  - ❑ The packet is an initial SYN packet trying to establish a connection to a server port on which no process is listening.
  - ❑ Malicious behaviour such as attempts to hijack a TCP connection



```
No. . Time     Source                Destination           Protocol Info
   63 1.148072 www.daimi.au.dk       GTRPC101.iut-colmar.n HTTP     Continuation
   64 1.148104 GTRPC101.iut-colmar.n www.daimi.au.dk       TCP      1241 > http [ACK] Seq=2116511829 Ack=923613623 Win=64860 Len=0
   65 1.151081 GTRPC101.iut-colmar.n gtrpcsrv3.iut-colmar. TCP      1244 > ftp [SYN] Seq=2204999609 Ack=0 Win=64240 Len=0
   66 1.151708 gtrpcsrv3.iut-colmar. GTRPC101.iut-colmar.n TCP      ftp > 1244 [SYN, ACK] Seq=4280451682 Ack=2204999610 Win=32736 Len=0
   67 1.151721 GTRPC101.iut-colmar.n gtrpcsrv3.iut-colmar. TCP      1244 > ftp [ACK] Seq=2204999610 Ack=4280451683 Win=64240 Len=0
   68 1.201084 gtrpcsrv3.iut-colmar. GTRPC101.iut-colmar.n FTP      Response: 220 gtrpcsrv3.uha.fr FTP server (Version wu-2.4.2-academ[BETA-15](1) W
   69 1.217617 www.daimi.au.dk       GTRPC101.iut-colmar.n HTTP     Continuation
   70 1.217640 GTRPC101.iut-colmar.n www.daimi.au.dk       TCP      1241 > http [ACK] Seq=2116511829 Ack=923613623 Win=64860 Len=0
   71 1.217851 www.daimi.au.dk       GTRPC101.iut-colmar.n HTTP     Continuation
   72 1.217860 GTRPC101.iut-colmar.n www.daimi.au.dk       TCP      1241 > http [ACK] Seq=2116511829 Ack=923613623 Win=64860 Len=0
   73 1.257589 GTRPC101.iut-colmar.n admpcsrv5.iut-colmar. TCP      1146 > microsoft-ds [ACK] Seq=1920728189 Ack=3696829350 Win=63914 Len=0
   74 1.366959 GTRPC101.iut-colmar.n gtrpcsrv3.iut-colmar. TCP      1244 > ftp [ACK] Seq=2204999610 Ack=4280451790 Win=64133 Len=0
   75 1.476366 www.daimi.au.dk       GTRPC101.iut-colmar.n HTTP     Continuation
   76 1.476412 GTRPC101.iut-colmar.n www.daimi.au.dk       TCP      1241 > http [ACK] Seq=2116511829 Ack=923617763 Win=64860 Len=0
   77 1.543731 www.daimi.au.dk       GTRPC101.iut-colmar.n HTTP     Continuation
   78 1.544452 www.daimi.au.dk       GTRPC101.iut-colmar.n HTTP     Continuation
   79 1.544470 GTRPC101.iut-colmar.n www.daimi.au.dk       TCP      1241 > http [ACK] Seq=2116511829 Ack=923620523 Win=64860 Len=0
```

```
⊞ Frame 65 (62 on wire, 62 captured)
⊞ Ethernet II
⊞ Internet Protocol, Src Addr: GTRPC101.iut-colmar.net (192.168.12.244), Dst Addr: gtrpcsrv3.iut-colmar.net (192.168.10.5)
⊟ Transmission Control Protocol, Src Port: 1244 (1244), Dst Port: ftp (21), Seq: 2204999609, Ack: 0, Len: 0
    Source port: 1244 (1244)
    Destination port: ftp (21)
    Sequence number: 2204999609
    Header length: 28 bytes
  ⊟ Flags: 0x0002 (SYN)
      0... .... = Congestion Window Reduced (CWR): Not set
      .0.. .... = ECN-Echo: Not set
      ..0. .... = Urgent: Not set
      ...0 .... = Acknowledgment: Not set
      .... 0... = Push: Not set
      .... .0.. = Reset: Not set
      .... ..1. = Syn: Set
      .... ...0 = Fin: Not set
    Window size: 64240
    Checksum: 0xc7cc (correct)
  ⊞ Options: (8 bytes)
```

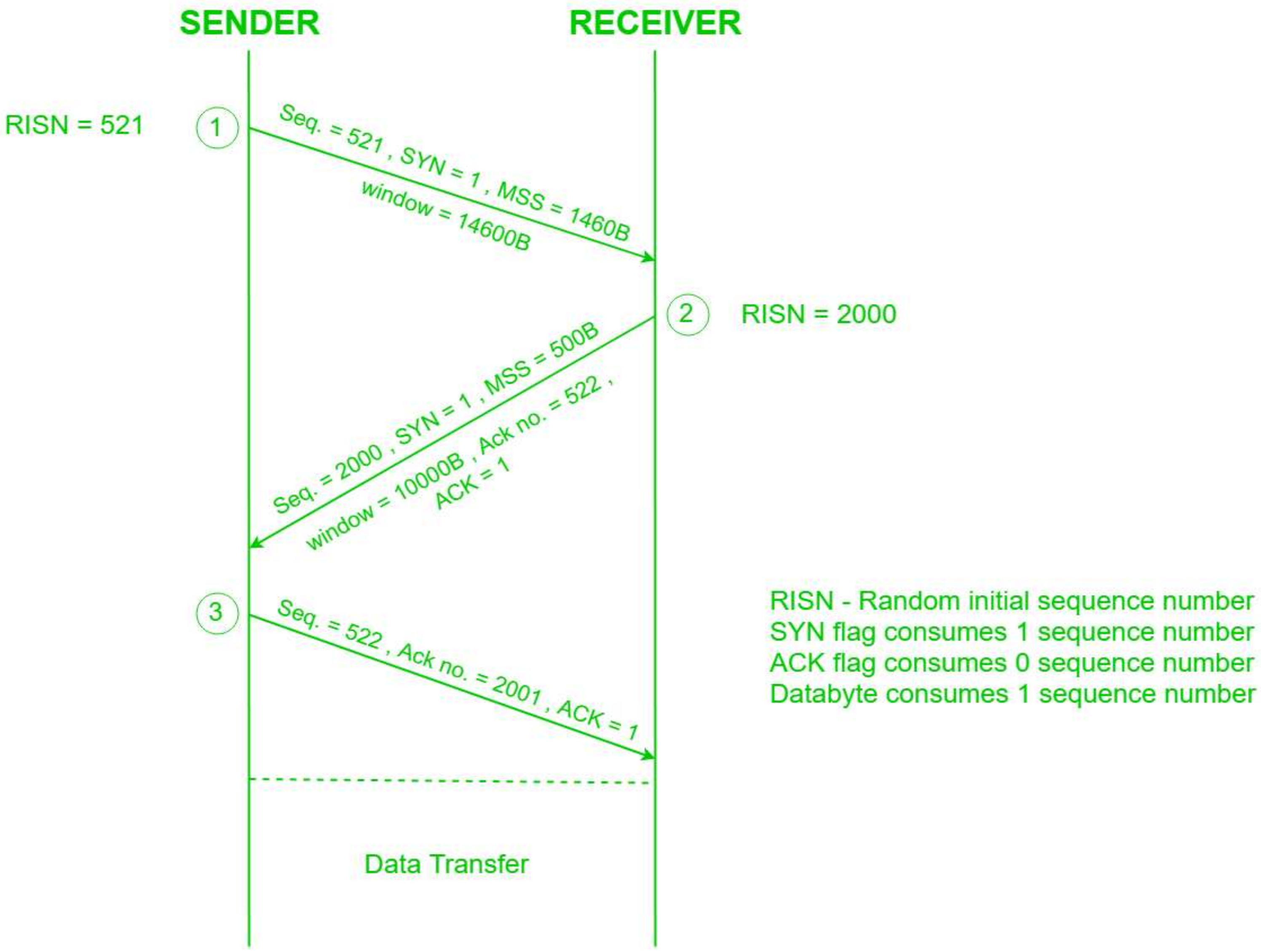
```
0000  00 05 dd 23 d8 3c 00 07  e9 6c 85 3f 08 00 45 00
0010  00 30 61 29 40 00 80 06  00 00 c0 a8 0c f4 c0 a8
0020  0a 05 04 dc 00 15 83 6d  9f b9 00 00 00 00 70 02
0030  fa f0 c7 cc 00 00 02 04  05 b4 01 01 04 02
```

TCP Protocol : Connection Establishment

- The connection establishment process is based on the knowledge of the initial sequence numbers of the two machines, i.e. the sender and the receiver must know the sequence numbers.

- The two machines must synchronize their connection via a mechanism called: Three Ways Handshake.

Connection Establishment



SENDER

RECEIVER

RISN = 521

1

Seq. = 521 , SYN = 1 , MSS = 1460B

window = 14600B

2

RISN = 2000

Seq. = 2000 , SYN = 1 , MSS = 500B

window = 10000B , Ack no. = 522 ,

ACK = 1

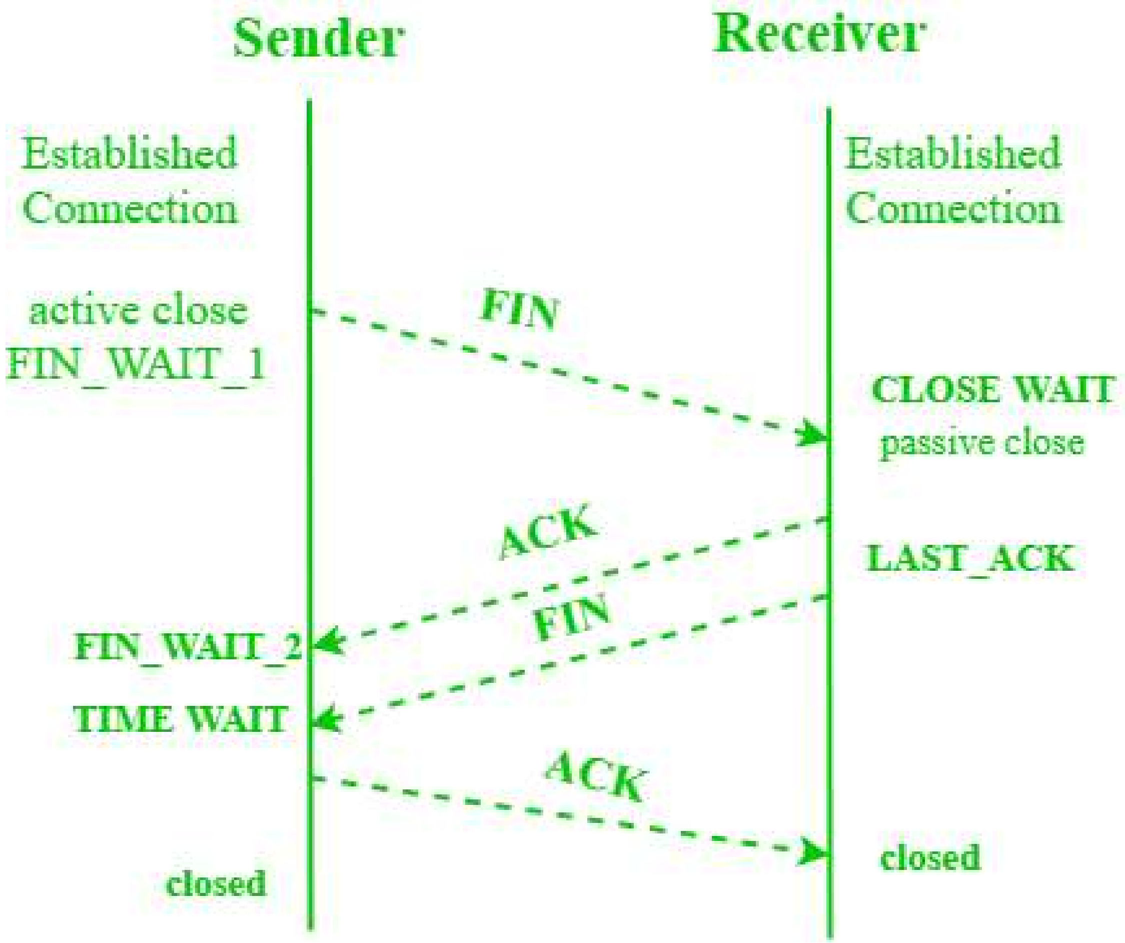
3

Seq. = 522 , Ack no. = 2001 , ACK = 1

Data Transfer

RISN - Random initial sequence number
SYN flag consumes 1 sequence number
ACK flag consumes 0 sequence number
Databyte consumes 1 sequence number

TCP Protocol – End of Connection



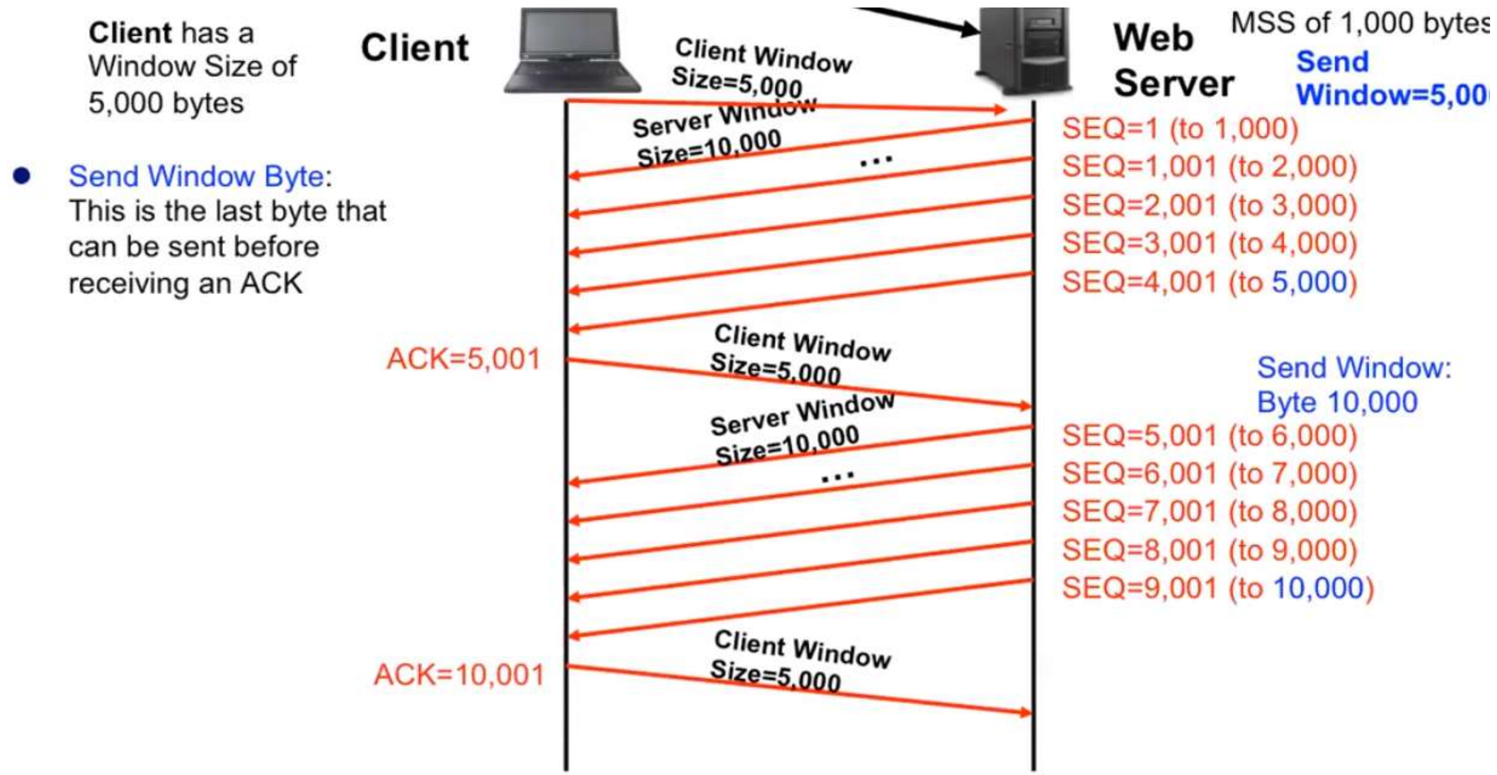
TCP flow Control & Congestion Control

TCP/IP model

Flow Control : host to host control

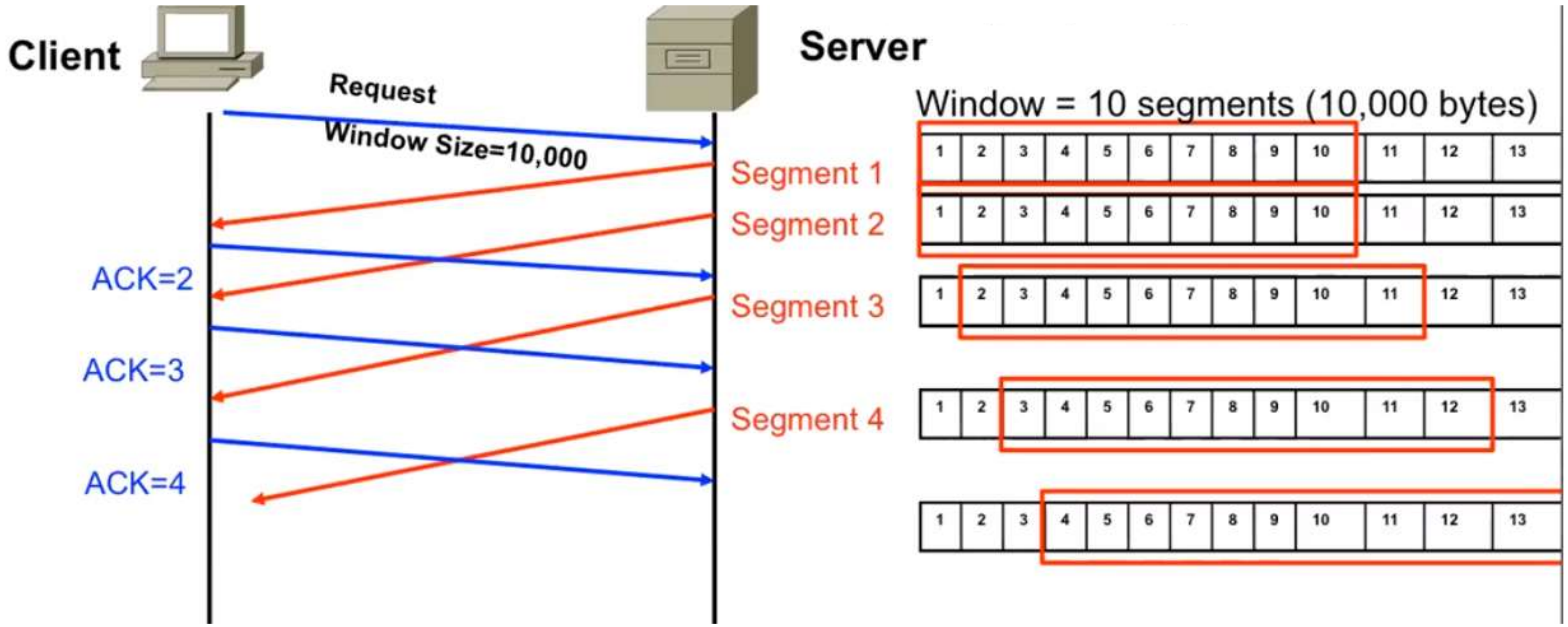
- Means that TCP will ensure that a sender is not overflowing the receiver's buffer by sending packets faster that it can consume
- TCP implements flow control by increasing / decreasing windows size as required
- To control the amount of data that TCP can send, the receiver will advertise its Receive Window (rwnd), that is, the spare room in the receive buffer.
- the receive window is used to give the sender an idea of how much free buffer space is available at the receiver

FLOW CONTROL – WINDOWS SIZE



Waiting for an Acknowledgement after each Windows Size would be vey ineficient : The server should wait for a long time de receive this acknowledgment

SLIDING WINDOWS



Sliding windows allows the sender to transmit multiple segments within the TCP window size without receiving an acknowledgment.

A device can send successive segments without waiting for an acknowledgement that a previous segment has been received.

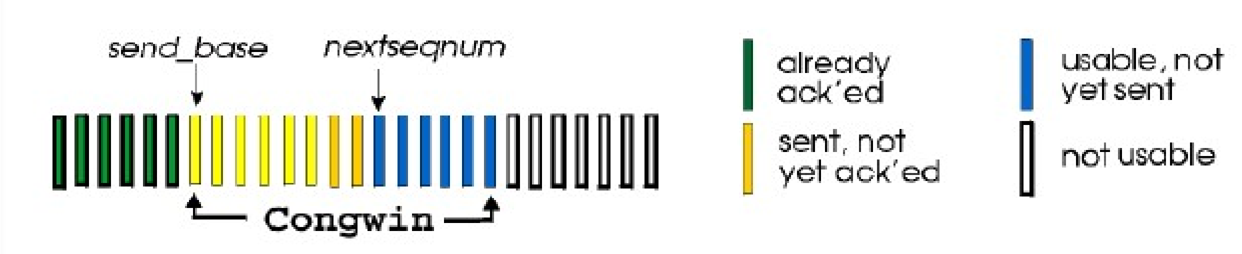
TCP Congestion Control

- Congestion occurs when windows size exceeds networks or receiver capacity

- Congestion is the **saturation** of network nodes resulting in a delay in the routing of data, until it is lost
- TCP manages end-to-end flow control but also congestion problems related to the network

TCP Congestion Control

- The idea of TCP congestion control is for each source determines how much capacity is available to a given flow in the network.



- CongestionWindow (cwnd) is a variable held by the TCP source for each connection.

MaxWindow :: min (**CongestionWindow** , AdvertisedWindow)

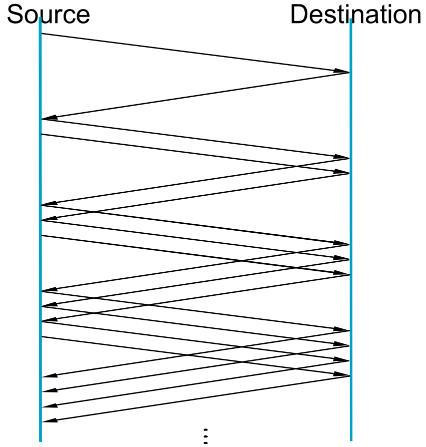
EffectiveWindow = MaxWindow – (LastByteSent -LastByteAcked)

TCP Congestion Control

- For each positive acknowledgement, increase cwnd by 1 packet

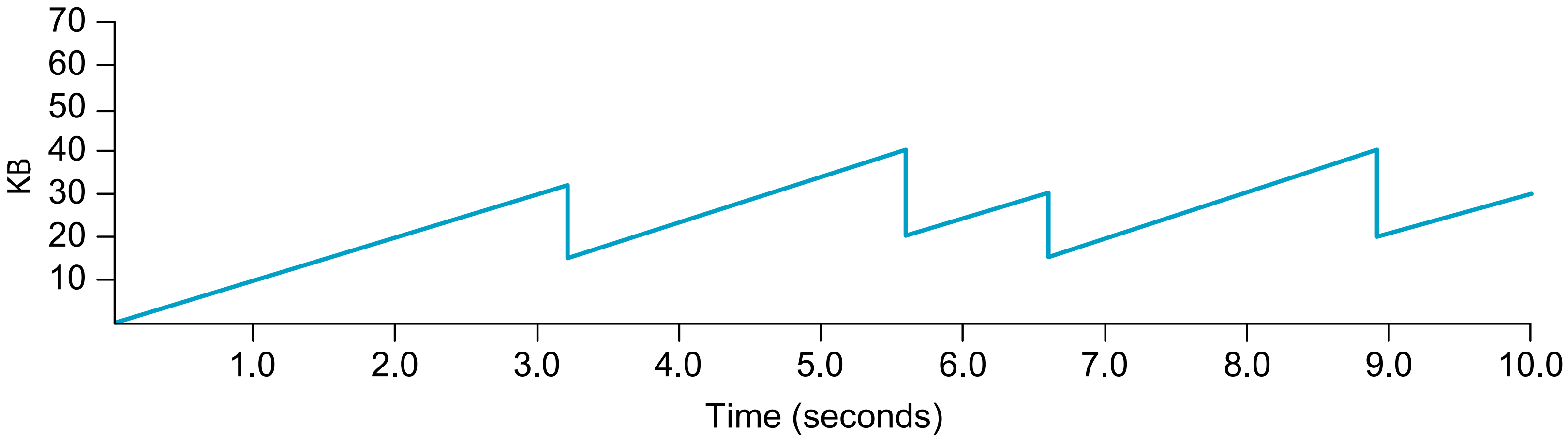
increment = MSS x (MSS /cwnd)
cwnd = cwnd + increment

➡ Linear Additive Increase Idea : Add one packet each **RTT**



TCP Congestion Control

- When a packet drop occurs or a timeout of a packet is observed due to a congestion : a **Multiplicative Decrease** fucntion is applied
- TCP reduces the rate at which it is transmetting
- Each time a timeout occurs, the source sets « *CWND* » to half of its previous value.



Slow Start Algorithm

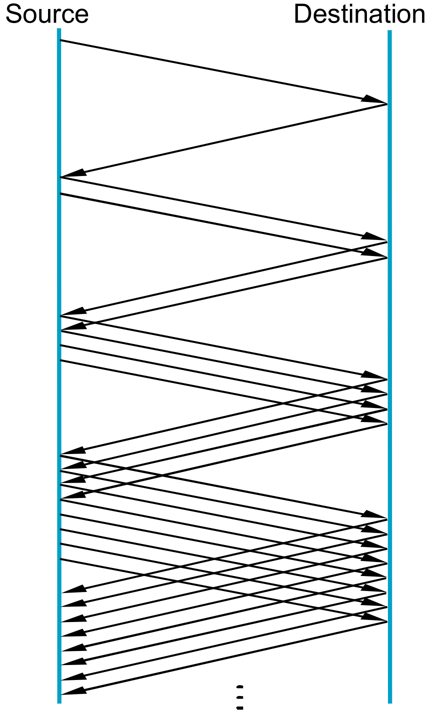
- Linear additive increase takes **too long** to ramp up a new TCP connection from cold start.

- The **slow start mechanism** was added to provide an initial exponential increase in the size of cwnd.

Slow Start Algorithm

- The source starts with cwnd = 1.
- Every time an ACK arrives, cwnd is incremented.
- **Slow start increases the CWND exponentially**

➔ cwnd is effectively doubled per RTT “epoch”.

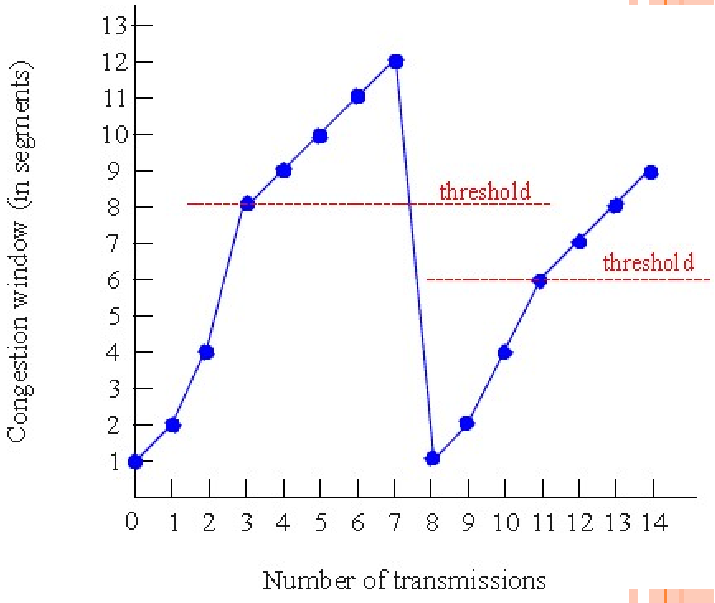


Slow Start
Add one packet per ACK

Congestion Avoidance – TCP Tahoe

- The aim idea is :
  - Use slow start until a threshold called « ssthresh »
  - After : Increase « cwnd » with 1 each RTT

- If lost by timeout (=> successive losses):
  - **ssthresh = cwnd / 2**
  - **enters in slow start with cwnd=1**
- If loss by 3 dupack (=> isolated loss):
  - **ssthresh = cwnd / 2**
  - **enters fast retransmission**



Fast Retransmission

- Very short, arrives here after a loss by 3 dupack
- Fast retransmission = no more waiting for the timeout, but :
  - **we retransmit the packet immediately**
  - **we enter in fast recovery (except Tahoe : slow start with cwnd=1)**

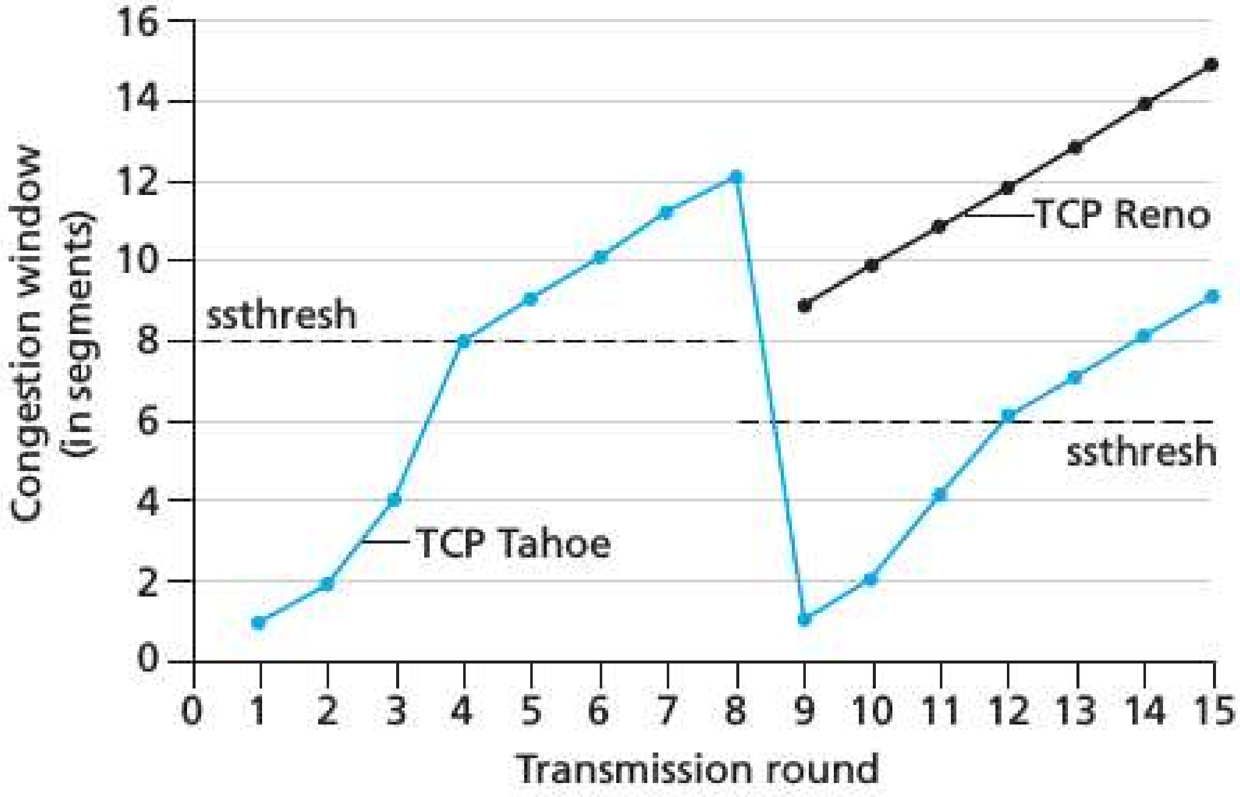
Fast Recovery – TCP Reno

- Temporary step, arrives here from fast retransmission
- The aim Idea : wait for cwnd/2 ACKs, then resend a new packet foreach ACK received
- ssthresh = cwnd / 2

Fast Recovery

After Fast Retransmit, half cwnd and commence recovery from this point using linear additive increase 'primed' by left over ACKs in pipe.

Congestion Control – Summary



NETWORK SERVICE - TCP ADDRESSING

- There are many network applications running on a host. When a packet arrive at network layer, how to know which application to send to?

  - Port: there are 216 = 65536 ports (0-65535) on one machine
  - One port is linked to only one application
  - One application may use many ports for different purposes
    - e.g. FTP: 20, 21

Client/Server architecture

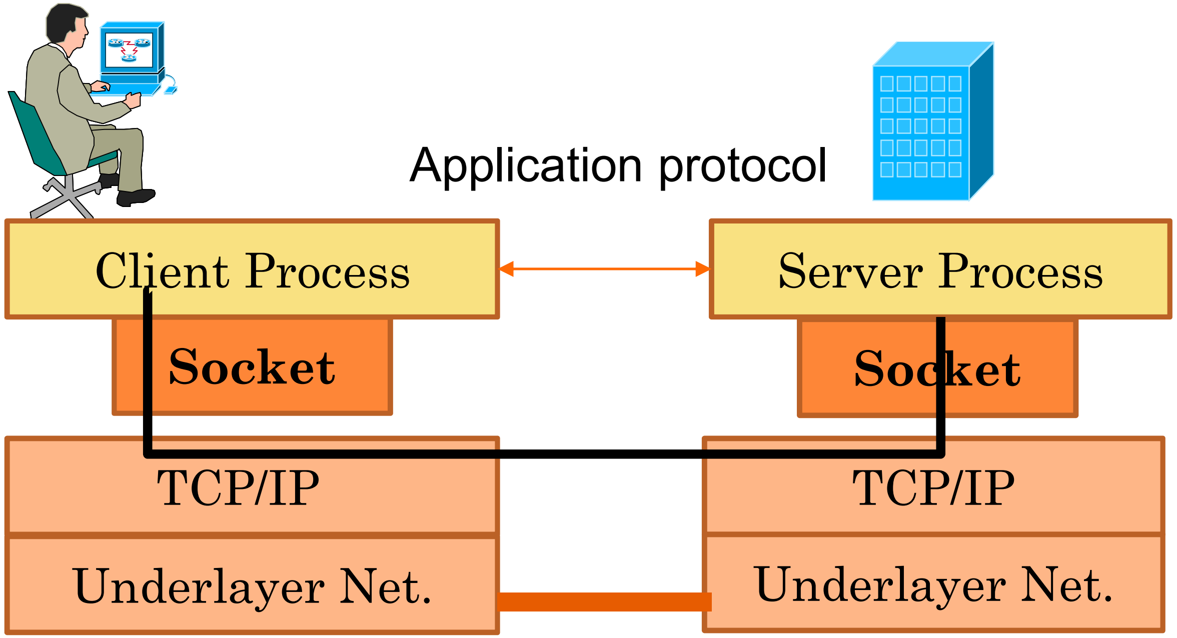
- The basic Idea :
- the application is distributed on different sites tooptimize processing, storage...
- **The client**
  - makes a service request to the server (request)
  - initiates the communication to open the session
- **The server**
  - is the part of the application that offers a service
  - listens for client requests
  - responds to the service requested by the client (response)

CLIENT/SERVER MODEL

- A Client/Server application is
- **a client part** that executes requests to a server
- **a server part** that processes and responds to client requests
- **an application protocol** that defines the exchanges between a client and a server
- **an access via an API** (programming interface, access via an API (programming interface, usually sockets) to the transport layer of messages

SOCKET – ACCESS INTERFACE TO NETWORKS

- How a Client/Server communication is made



A socket: local interface to the host, created by the application, controlled by the OS

It's a Communication gateway between the client and the server process

NETWORK SOCKET

- Wikipedia Definition

«

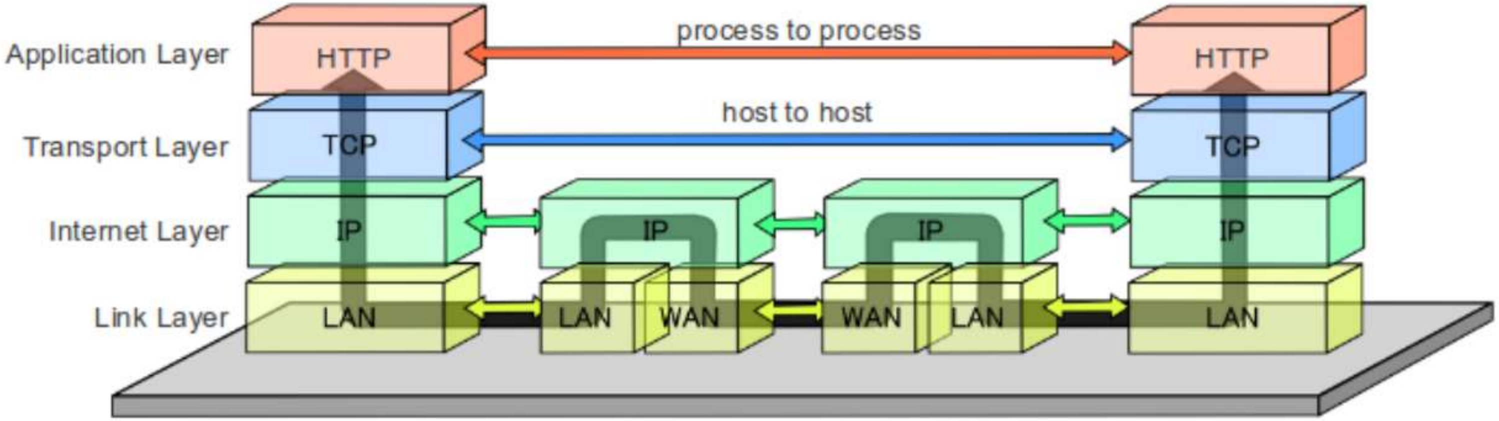
A **network socket** is a software structure within a network node of a computer network that serves as an endpoint for sending and receiving data across the network. The structure and properties of a socket are defined by an application programming interface (API) for the networking architecture. Sockets are created only during the lifetime of a process of an application running in the node.

»

SOCKET AND THE PORT NUMBERS

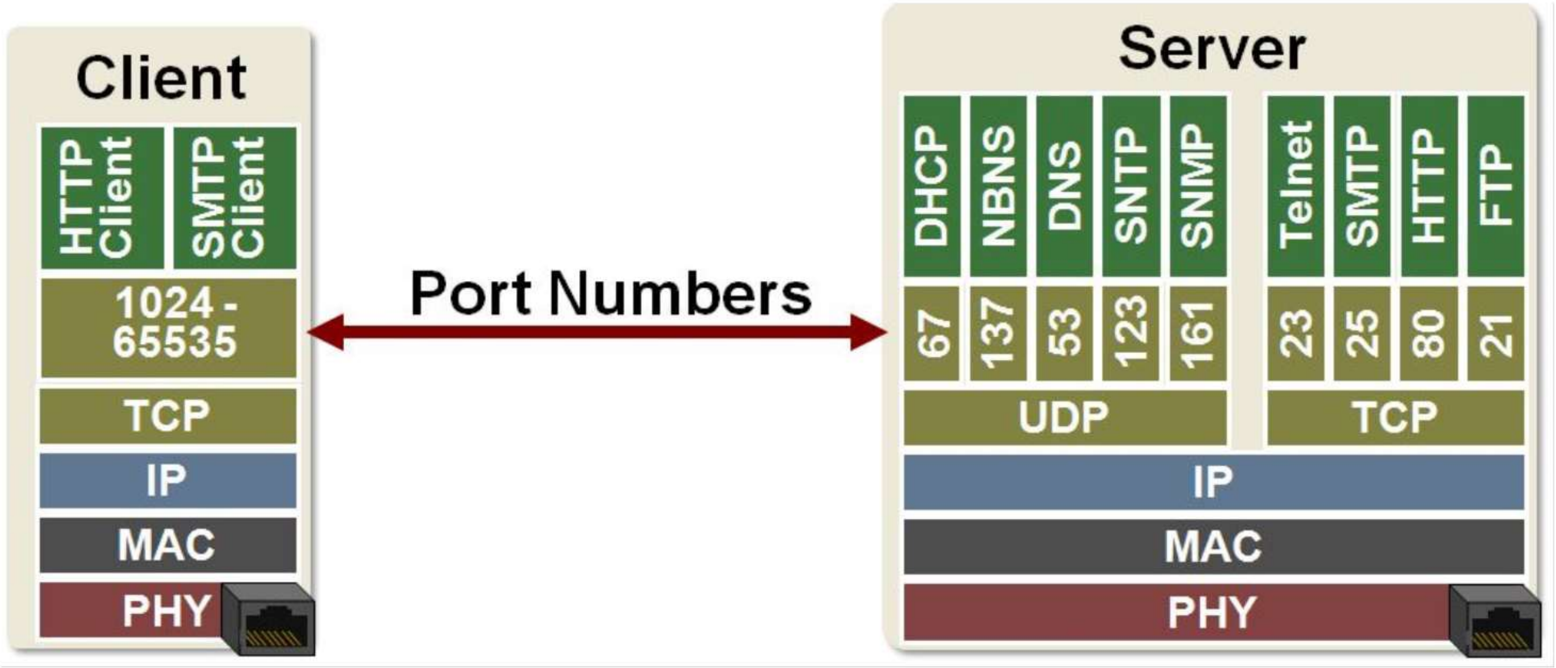
- Each network communication is identified by $(@IP_{src}, Port_{src})$ and $(@IP_{dst}, Port_{dst})$

Data Flow of the Internet Protocol Suite



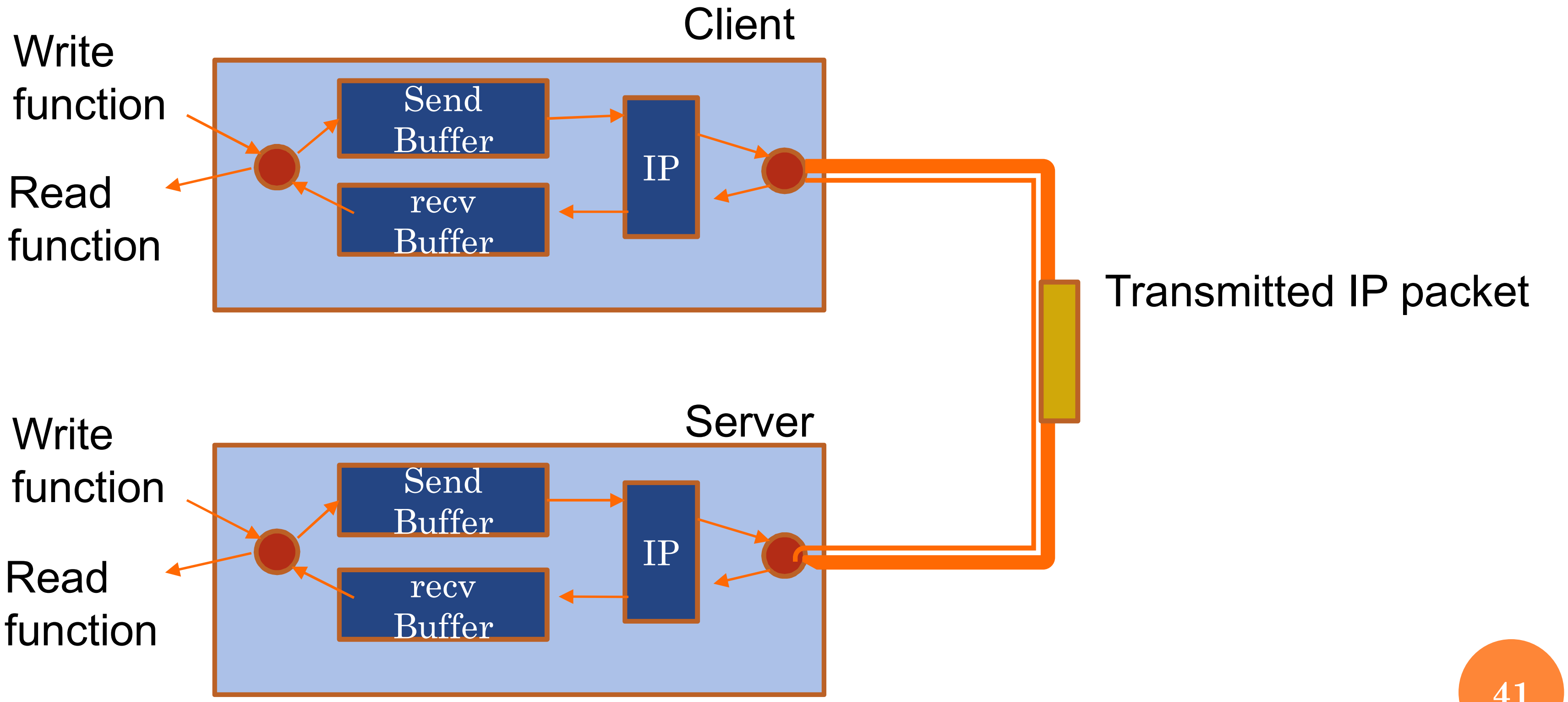
Socket and the port numbers

- Most Network services use a Well-known port numbers less than 1023.

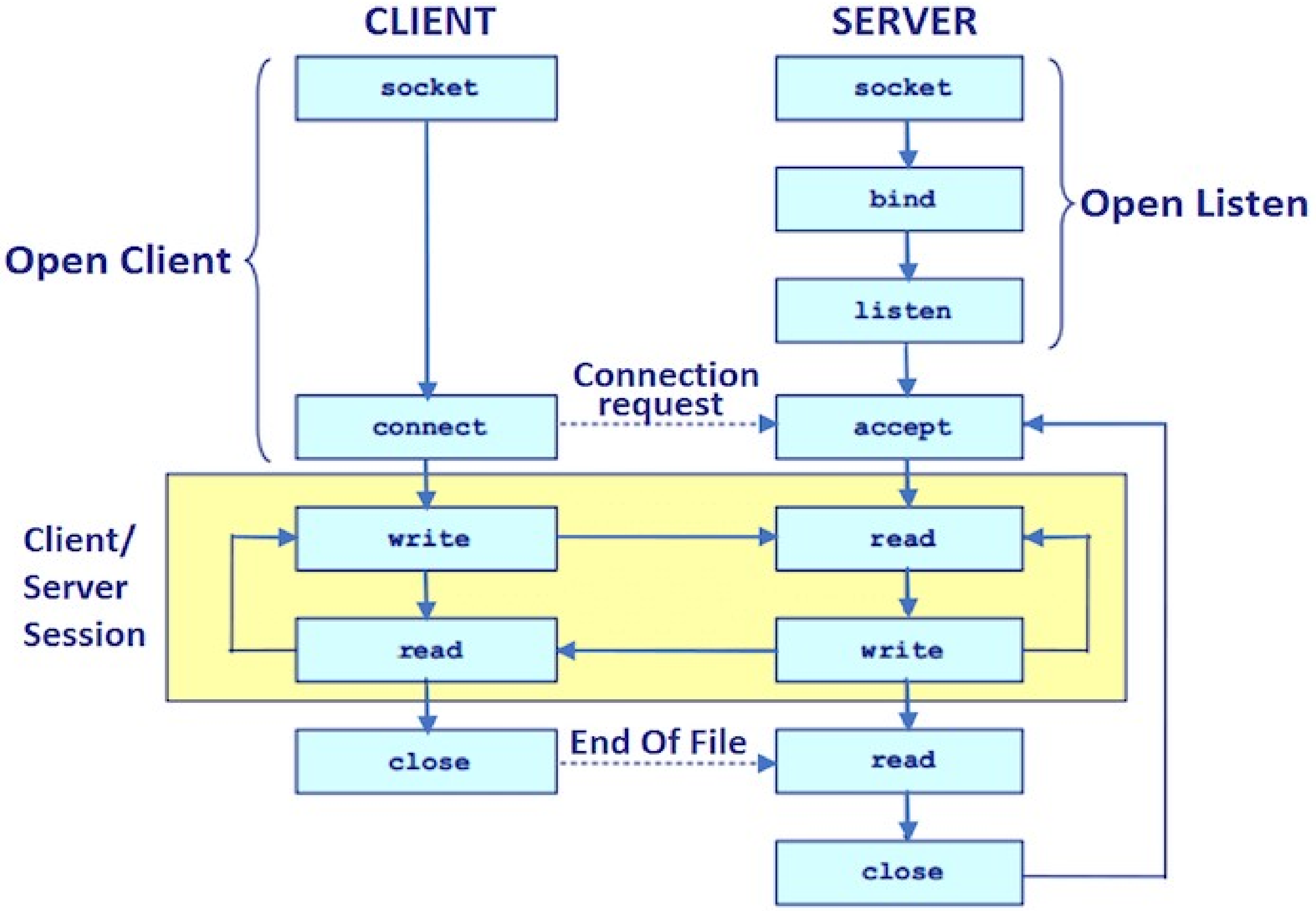


Socket and Port Numbers

- Client/Server Communication model



Socket - Implementation



SOCKET API

DNS – Domain Name System

TCP/IP model

INTRODUCTION

- The basic technology (TCP/IP) allows access to machines by their IP address.
- These addresses are part of the fundamental infrastructure of the Internet:
  - allow machines to be identified by IP addressing
  - communicate with remote machines (IP routing).
- It has become unthinkable for humans to know all (IP) addresses of the machines they want to access.
- The DNS system makes it possible to identify a machine by a name (s) representative of the machine and the network (s) on which it is located; example:

www.ufaz.az identifies the machine www on the network **ufaz.az**

DNS

- The DNS system is a fundamental part of the Internet. Because Internet applications are "dependent" on DNS:
  - how to reach web sites (URL)
  - how to exchange electronic mail
  - how to memorize the IP addresses of all the machines to be reached
  - how to manage IP address changes (Name === IP)

- The DNS system is implemented by a hierarchical database and is distributed in the Internet

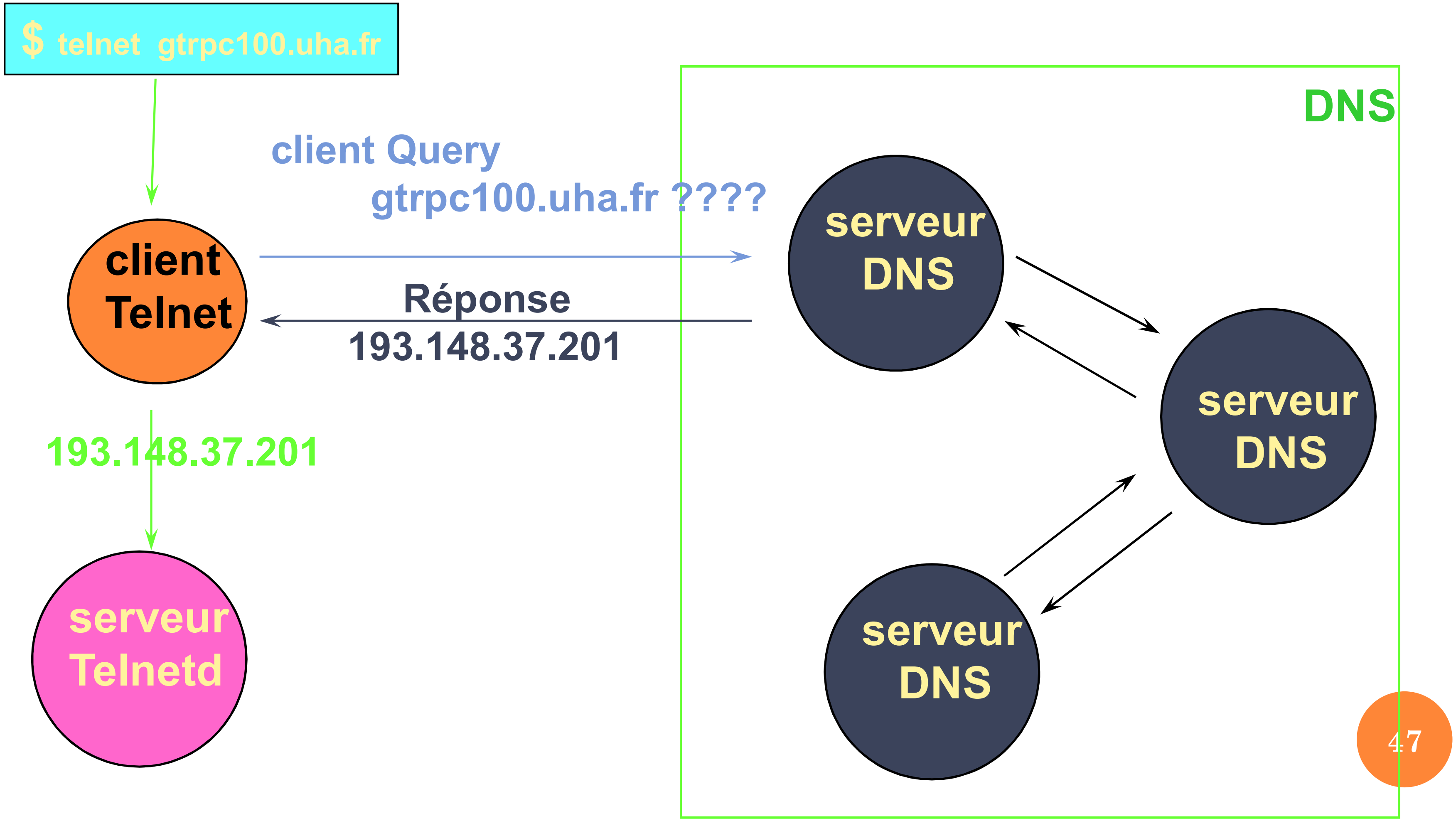
Principle of Operation

- DNS is based on the client/server model
- the client queries a name server; typically :
  - the user associates a domain name with an application; example :telnet gtrpc100.uha.fr
  - the client application requests the translation of the domain name from a name server (DNS):

    this operation is called **name resolution**
  - the server queries other name servers until the domain name/IP address association is found
  - the **name server returns the IP address** to the client : 193.148.37.201
  - the client contacts the server (telnetd) as if the user had specified an IP address:

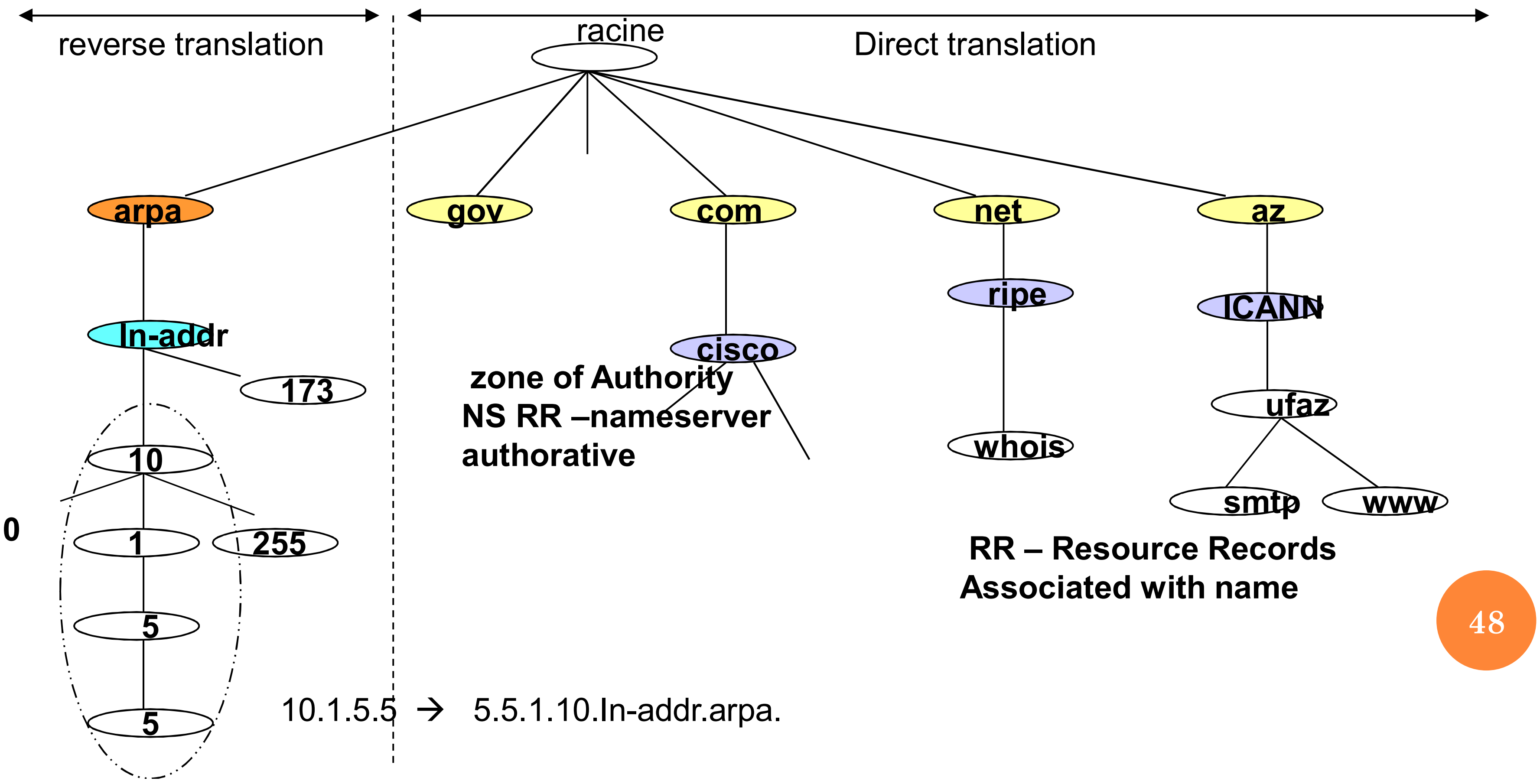
    telnet 193.148.37.201

PRINCIPLE OF OPERATION



- The DNS namespace is hierarchical and presented in a manner identical to the UNIX file system

- The domain name of any node in the tree is the list starting with the node itself and going up to the root



- DNS is a hierarchical client-server protocol. Each domain (e.g.microsoft.com, ufaz.az, etc) is served by one or more DNS servers
- A large domain can delegate authority over sub-domains of their allocated name space to other name servers
- Replies that come directly from the responsible DNS server are termed ***authoratative*** while replies that come from other DNS servers are ***non-authoritative***
- The most common types of records stored in the DNS database are for Start of Authority (SOA), IP addresses (A and AAAA), SMTP mail exchangers (MX), name servers (NS), pointers for reverse DNS lookups (PTR), and domain name aliases (CNAME).

DNS QUERY

- To ensure interoperability on the Internet of name resolutions, the IETF standardized the DNS protocol in 1987.
- Some definitions :
  - **DNS client**: to request a name translation; the client must formulate a request to the DNS server. This request is done by identifying the IP address of DNS, which is found (in the case of UNIX) in **/etc/resolv.conf**
  - Syntax: search ufaz.fr

    nameserver 10.252.254.10

  - **DNS serve**r: contains matches between machine names and IP addresses for a domain name.
- DNS client queries its server via UDP/TCP port 53. If the DNS query/answers exceeds 572 bytes (like : Zone transfer), then DNS works with TCP.

DNS – Zone Definition

- To better manage the updates of information in the name translation tree, the DNS will rely on the notion of zone delegation. This delegation consists in to give the responsibility of a part of the naming tree
- primary server of the zone, this last one can again delegate parts of the tree to other (secondary) servers.
- Zone file is a file that contains the header and records of a domain
- Header of the zone file contains :
  - Domain name and lifetime for the records of this domain
  - One SOA (Start Of Authority) record per domain:
    - Identifies a DNS server as the primary authority
    - Serial number
    - Interval for refreshing the file by secondary servers
    - Interval for zone file expiration (TTL)
  - SOA: Identifies the authority and defines the parameters for a domain

DNS – Zone Definition

- NS: Name Server (DNS Server)
- MX: Mail eXchanger: identifies the mail server(s) (SMTP) responsible for receiving responsible for receiving mail for this domain.

- A: Name-to-Address Mapping
- CNAME : Canonical Name; alias
- PTR : Address-to-Name Mapping

Domain name server : Zone File

Domain named zone declaration file

```
options
{
directory "namedb";
};

zone "0.0.127.IN-ADDR.ARPA"
{
type master;
file "127.0.0";
};

zone ‘’Nom_domaine ‘’
{
notify yes;
type master;
file ‘’/etc/bind/direct’.db”;
};

zone ‘’XX.XX.XX.IN-ADDR.ARPA”
{
notify yes;
type master;
file ‘’/etc/bind/reverse.db”;
};
```

-> reverse zone IP addresses ⇨ name corresponding to the zone
Your_domain.az.

```
zone "0.198.192.IN-ADDR.ARPA" {
        notify no;
        type master;
        file « /etc/bind/reverse.db";
};
```

the **master type** designates the name server that has authority of this zone.

Notify identifies if the server **sends** a notification to the slave servers when an update is happened :

yes - notifie les serveurs esclaves,

no - ne notifie les serveurs esclaves,

Explicit - only the slave servers existing in a list **also-notify** can be notified in a given zone.

DOMAIN NAME SERVER : ZONE FILE

- Zone file for a direct translation (direct.db)

```
;
; BIND reverse data file for broadcast zone
;
$TTL            604800
@               IN              SOA             ns.ufaz.az. root.ufaz.az. (
                                                     1                          ; Serial
                                                 6H                             ; Refresh
                                                  86400                         ; Retry
                                                2419200                         ; Expire
                                                 604800 )                       ; Negative Cache TTL
;
                                NS               ns.ufaz.az.
                                MX              10              mail.ufaz.Az.

ns                              A               10.0.0.2
mail                            A               10.0.0.55
```

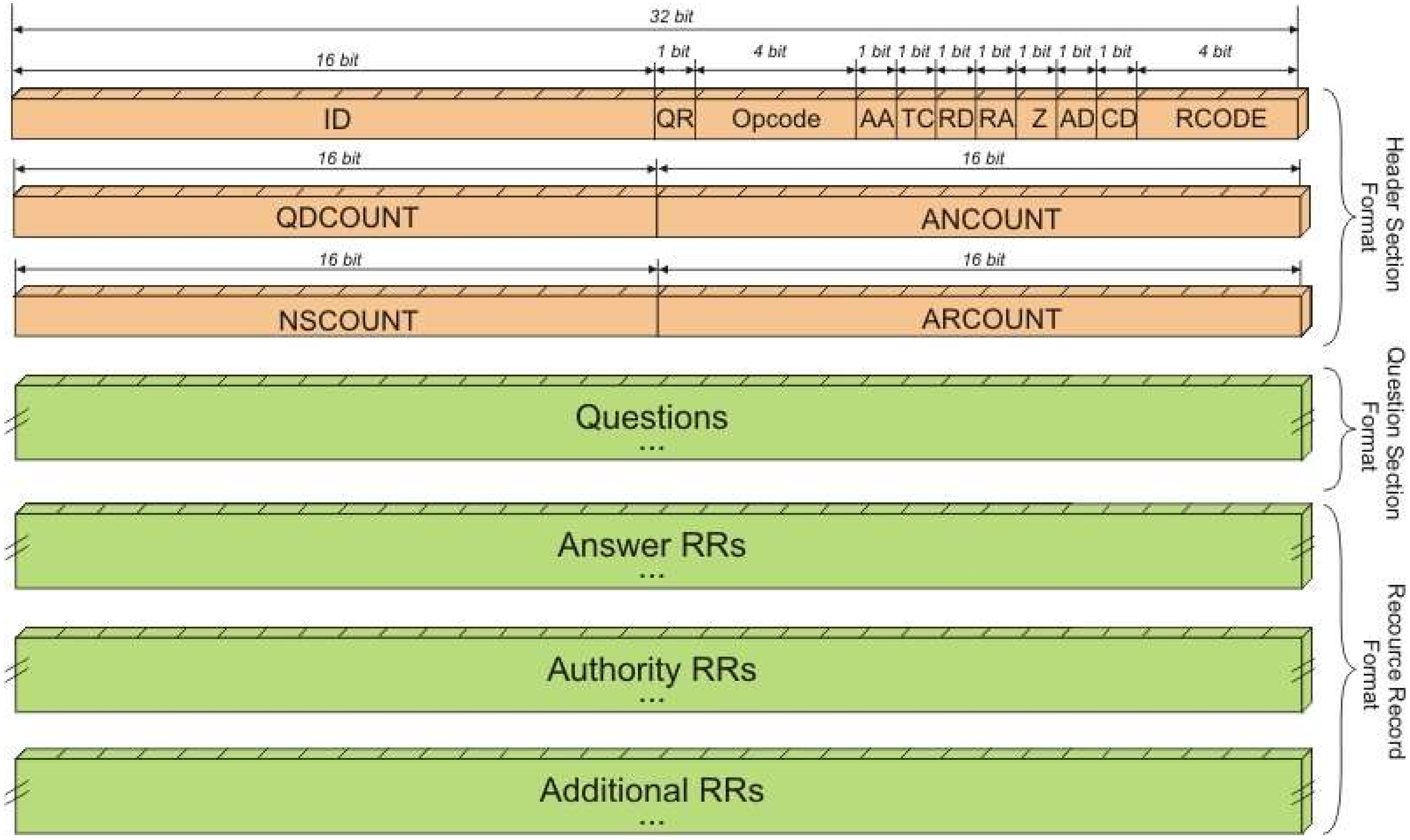
FILE ZONE

- Reverse translation file : reverse.db

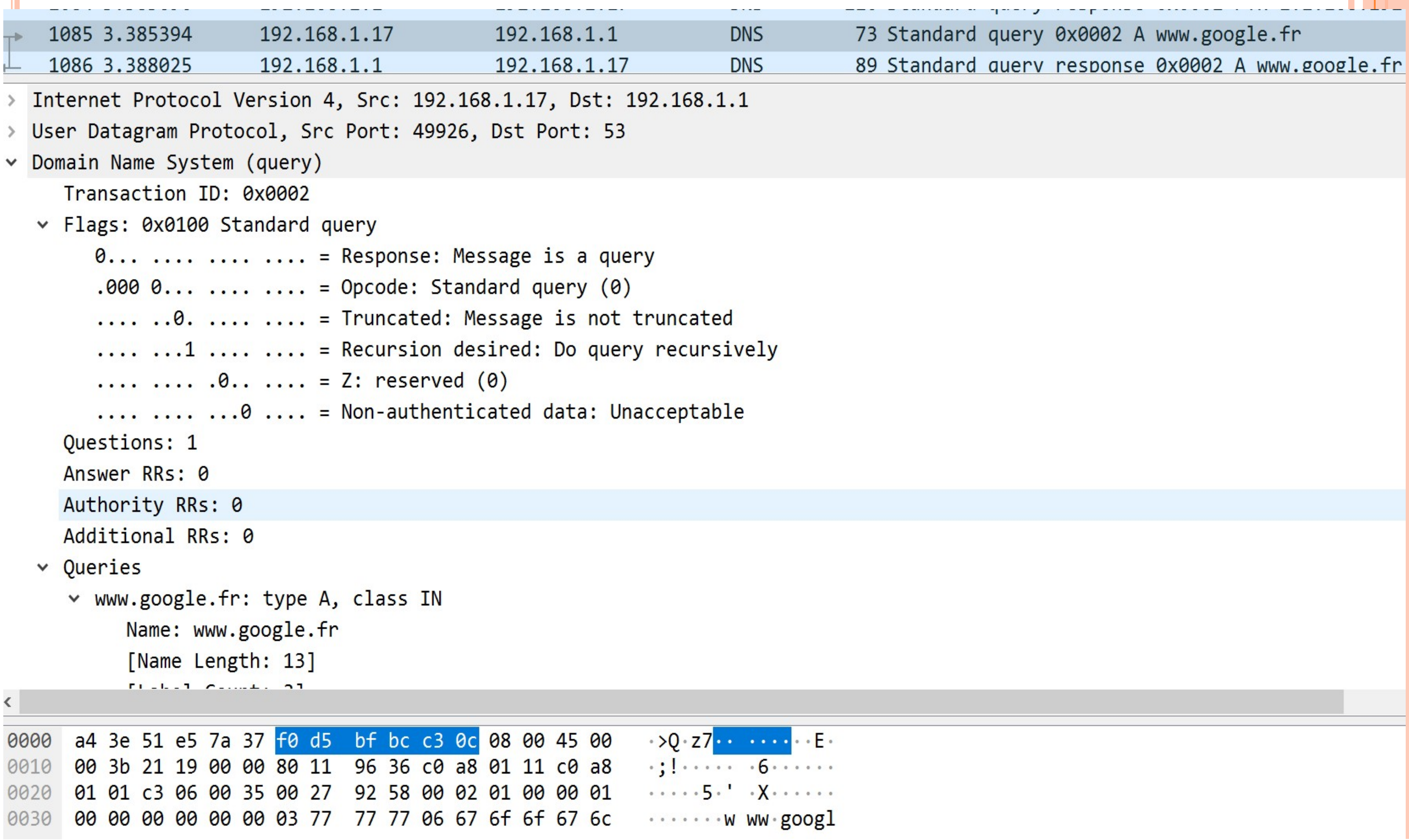
```
;
; BIND reverse data file for broadcast zone
;
$TTL        604800
@           IN          SOA         ns.ufaz.az. root.ufaz.az. (
                                         1                  ; Serial
                                     6H                     ; Refresh
                                      86400                 ; Retry
                                    2419200                 ; Expire
                                     604800 )               ; Negative Cache TTL
;
                        NS           ns.ufaz.az.
                        MX          10          mail.ufaz.Az.

2                       PTR         ns.ufaz.az.
55                      PTR         ns.ufaz.az.
```

DNS Message


32 bit
16 bit
1 bit
4 bit
1 bit
1 bit
1 bit
1 bit
1 bit
1 bit
1 bit
4 bit
ID
QR
Opcode
AA
TC
RD
RA
Z
AD
CD
RCODE
16 bit
16 bit
QDCOUNT
ANCOUNT
16 bit
16 bit
NSCOUNT
ARCOUNT
Questions
...
Answer RRs
...
Authority RRs
...
Additional RRs
...
Header Section Format
Question Section Format
Resource Record Format

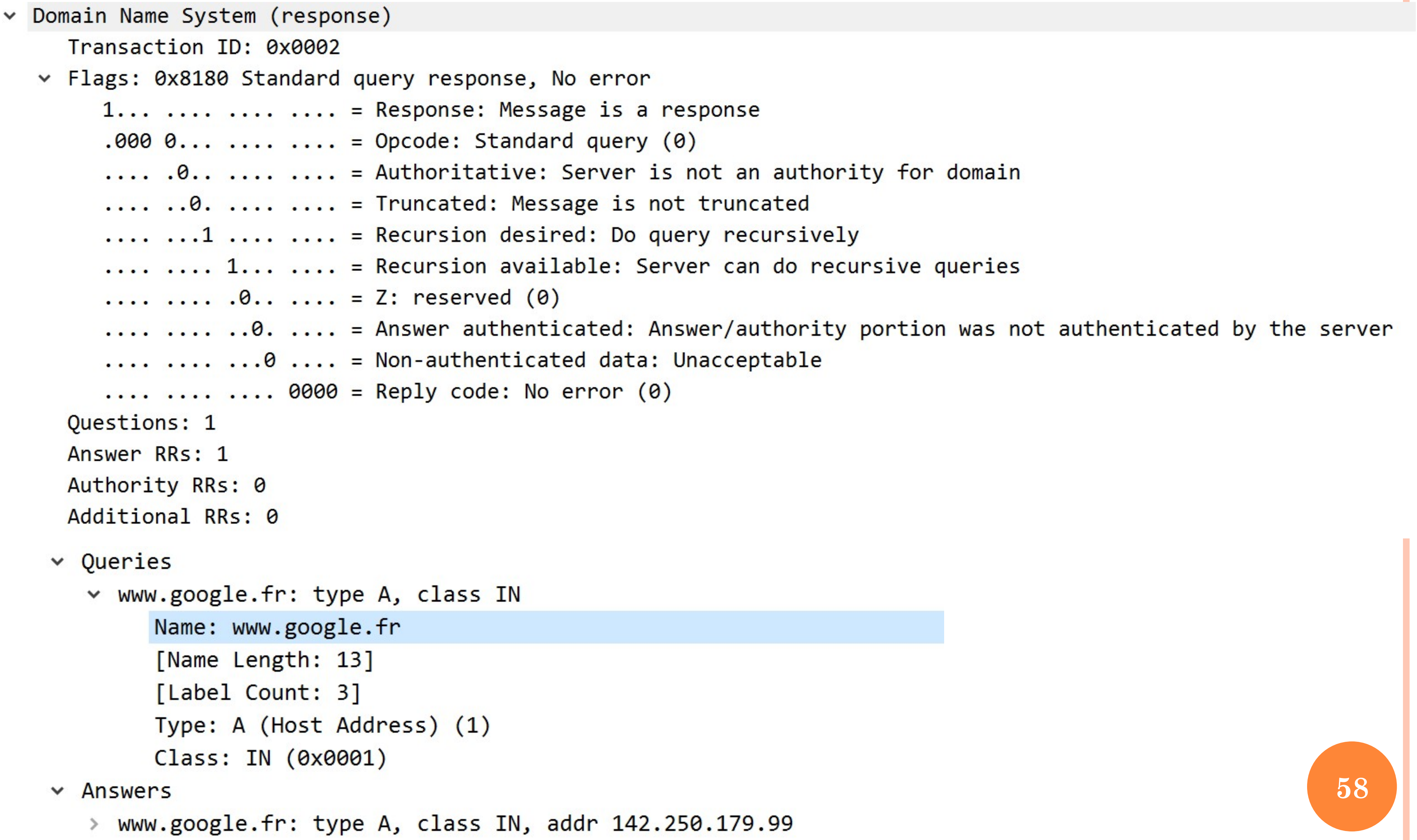
DNS- QUERY



```
1085 3.385394    192.168.1.17    192.168.1.1     DNS    73 Standard query 0x0002 A www.google.fr
1086 3.388025    192.168.1.1     192.168.1.17    DNS    89 Standard query response 0x0002 A www.google.fr
> Internet Protocol Version 4, Src: 192.168.1.17, Dst: 192.168.1.1
> User Datagram Protocol, Src Port: 49926, Dst Port: 53
v Domain Name System (query)
    Transaction ID: 0x0002
  v Flags: 0x0100 Standard query
      0... .... .... .... = Response: Message is a query
      .000 0... .... .... = Opcode: Standard query (0)
      .... ..0. .... .... = Truncated: Message is not truncated
      .... ...1 .... .... = Recursion desired: Do query recursively
      .... .... .0.. .... = Z: reserved (0)
      .... .... ...0 .... = Non-authenticated data: Unacceptable
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  v Queries
    v www.google.fr: type A, class IN
        Name: www.google.fr
        [Name Length: 13]
        [Label Count: 3]

0000  a4 3e 51 e5 7a 37 f0 d5  bf bc c3 0c 08 00 45 00   ·>Q·z7·· ······E·
0010  00 3b 21 19 00 00 80 11  96 36 c0 a8 01 11 c0 a8   ·;!····· ·6······
0020  01 01 c3 06 00 35 00 27  92 58 00 02 01 00 00 01   ·····5·' ·X······
0030  00 00 00 00 00 00 03 77  77 77 06 67 6f 6f 67 6c   ·······w ww·googl
```

DNS - RESPONSE



```
⌄ Domain Name System (response)
     Transaction ID: 0x0002
   ⌄ Flags: 0x8180 Standard query response, No error
        1... .... .... .... = Response: Message is a response
        .000 0... .... .... = Opcode: Standard query (0)
        .... .0.. .... .... = Authoritative: Server is not an authority for domain
        .... ..0. .... .... = Truncated: Message is not truncated
        .... ...1 .... .... = Recursion desired: Do query recursively
        .... .... 1... .... = Recursion available: Server can do recursive queries
        .... .... .0.. .... = Z: reserved (0)
        .... .... ..0. .... = Answer authenticated: Answer/authority portion was not authenticated by the server
        .... .... ...0 .... = Non-authenticated data: Unacceptable
        .... .... .... 0000 = Reply code: No error (0)
     Questions: 1
     Answer RRs: 1
     Authority RRs: 0
     Additional RRs: 0

   ⌄ Queries
      ⌄ www.google.fr: type A, class IN
           Name: www.google.fr
           [Name Length: 13]
           [Label Count: 3]
           Type: A (Host Address) (1)
           Class: IN (0x0001)
   ⌄ Answers
      › www.google.fr: type A, class IN, addr 142.250.179.99
```

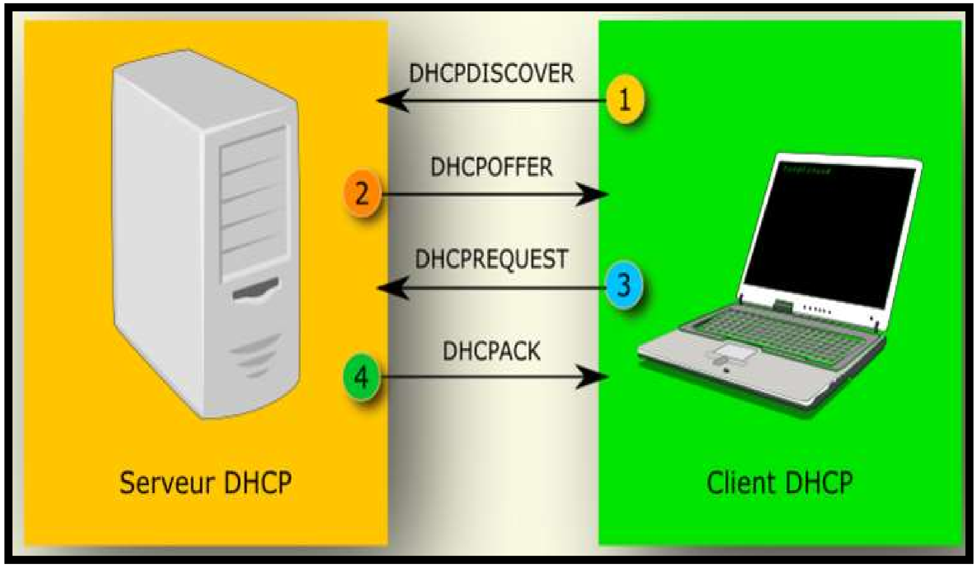
DHCP SERVER

Dynamic Host Configuration Protocol

Dynamic Assignment of IP addresses

- Dynamic assignment of IP addresses is desirable for several reasons:
  - IP addresses are assigned on-demand
  - Avoid manual IP configuration
  - Support mobility of laptops
- To connect a device to an IP network, we must have:
  - An unique IP address belonging in the same logical network
  - A subnet mask, the same for all hosts on the network
  - A DNS address, to be able to resolve host names
  - A gateway address to access the internet or other networks

DHCP OPERATION



- Broadcast « DHCPDISCOVER » : no idea of DHCP servers.
- Server responses with « DHCPOFFER » : IP address and a lease proposal
- Broadcast « DHCPREQUEST » to all servers to reserve the selected IP address
- DHCPACK message to confirm the reservation

DHCP SERVER

- A DHCP server has a range of addresses to distribute to its clients. It maintains a database of addresses already in use and those used a short time ago (this is why the same address is often recovered).

- The allocation of addresses is done through a lease. This lease normally has a limited duration.

- After expiration of the lease or termination of the lease by the client, the information about the lease remains stored in the server database for a certain period of time.

DHCP SERVER - CONFIGURATION

- Global parameters:

```
option domain-name « ufaz.az";
option domain-name-servers ns1.ufaz.az, ns2.ufaz.az;
default-lease-time 3600;
max-lease-time 7200;
authoritative;
```

- Zone configuration:

```
subnet 192.168.10.0 netmask 255.255.255.0 {
      option routers 192.168.10.1;
      option subnet-mask 255.255.255.0;
      option domain-search « ufaz.az";
      option domain-name-servers 192.168.10.1;
      range 192.168.10.10 192.168.10.100;
 }
```

```
host lap {
      hardware ethernet 00:f0:m4:6y:89:0g;
      fixed-address 192.168.10.105;
}
```

Mail Server

SMTP, POP, IMAP Protocols

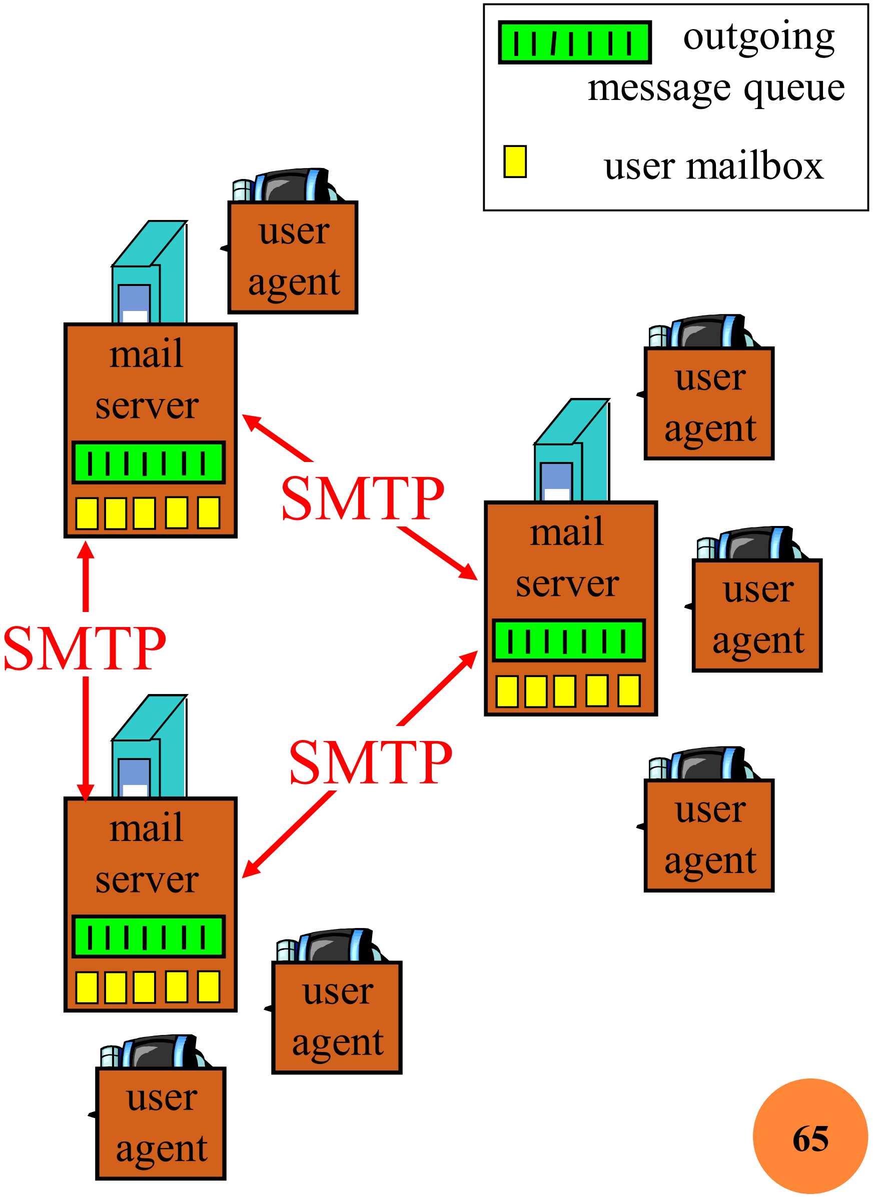
Electronic Mail

Three major components:

- user agents
- mail servers
- simple mail transfer protocol: SMTP

<u>User Agent</u>

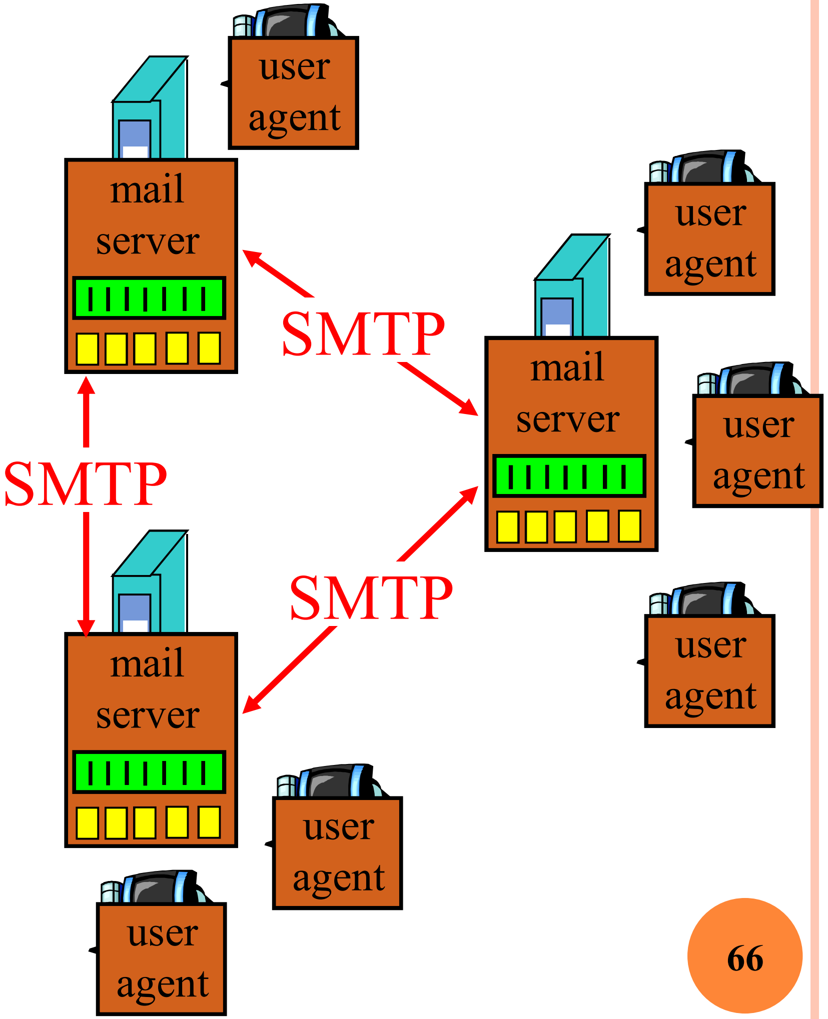
- a.k.a. "mail reader"
- composing, editing, reading mail messages
- e.g., Outlook, rainloop, roundcube, Mozilla Thunderbird
- outgoing, incoming messages stored on server



ELECTRONIC MAIL: MAIL SERVERS

Mail Servers

- mailbox contains incoming messages for user
- message queue of outgoing (to be sent) mail messages
- SMTP protocol between mail servers to send email messages
  - client: sending mail server
  - "server": receiving mail server



Electronic Mail: SMTP [RFC 2821]

- uses TCP to reliably transfer email message from client to server, port 25
- direct transfer: sending server to receiving server
- three phases of transfer
  - handshaking (greeting)
  - transfer of messages
  - closure
- command/response interaction
  - commands: ASCII text
  - response: status code and phrase
- messages must be in 7-bit ASCII

SCENARIO: ALICE SENDS MESSAGE TO BOB

1) Alice uses UA to compose message and "to" `bob@someschool.edu`

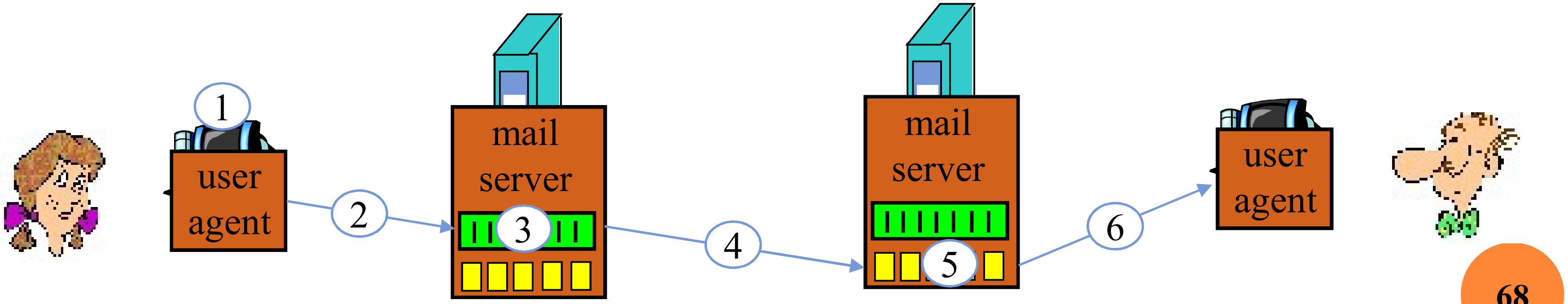
2) Alice's UA sends message to her mail server; message placed in message queue

3) Client side of SMTP opens TCP connection with Bob's mail server

4) SMTP client sends Alice's message over the TCP connection

5) Bob's mail server places the message in Bob's mailbox

6) Bob invokes his user agent to read message



Sample SMTP interaction

```
S: 220 hamburger.edu
C: HELO crepes.fr
S: 250  Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alice@crepes.fr>
S: 250 alice@crepes.fr... Sender ok
C: RCPT TO: <bob@hamburger.edu>
S: 250 bob@hamburger.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Do you like ketchup?
C: How about pickles?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection
```

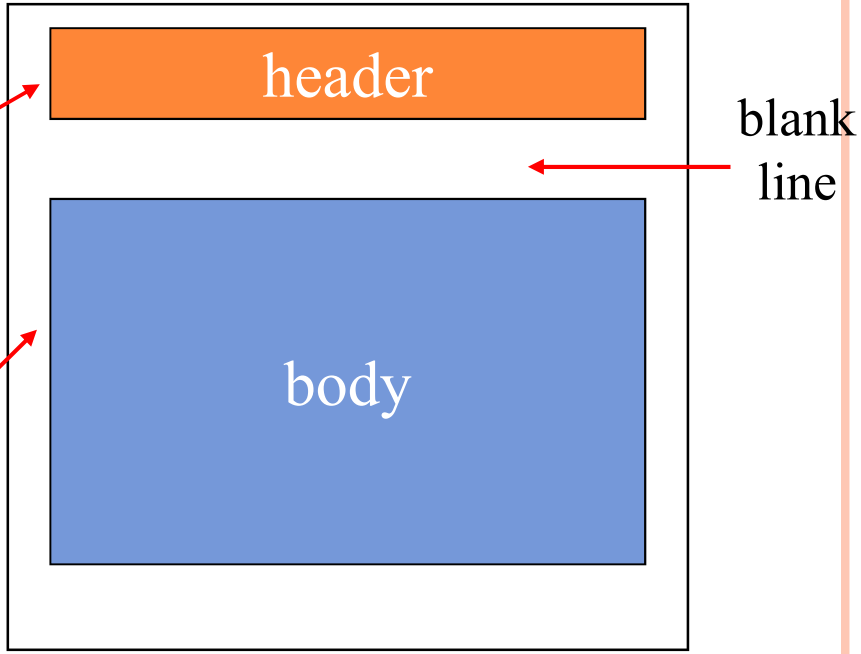
MAIL MESSAGE FORMAT

SMTP: protocol for exchanging email msgs

RFC 822: standard for text message format:

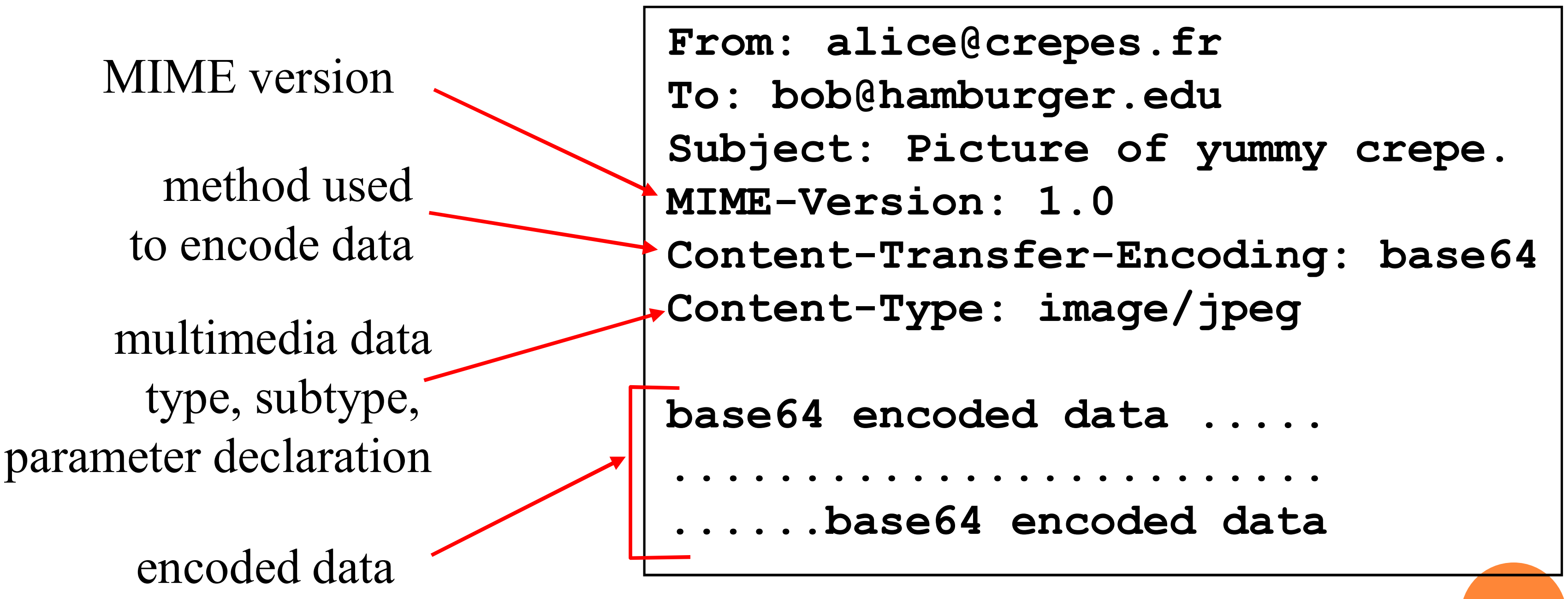
- header lines, e.g.,
  - To:
  - From:
  - Subject:

  different from SMTP commands!
- body
  - the "message", ASCII characters only



MESSAGE FORMAT: MULTIMEDIA EXTENSIONS

- MIME: multimedia mail extension, RFC 2045, 2056
- additional lines in msg header declare MIME content type



MIME version

method used to encode data

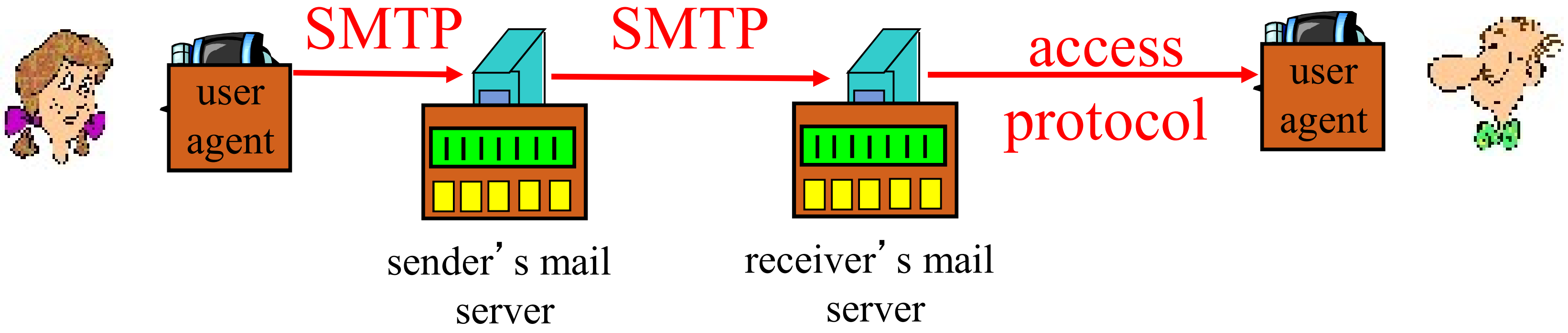
multimedia data type, subtype, parameter declaration

encoded data

```
From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Picture of yummy crepe.
MIME-Version: 1.0
Content-Transfer-Encoding: base64
Content-Type: image/jpeg

base64 encoded data .....
.........................
......base64 encoded data
```

MAIL ACCESS PROTOCOLS



- SMTP: delivery/storage to receiver's server
- Mail access protocol: retrieval from server
  - POP: Post Office Protocol [RFC 1939]
    - authorization (agent <-->server) and download
  - IMAP: Internet Mail Access Protocol [RFC 1730]
    - more features (more complex)
    - manipulation of stored msgs on server
  - HTTP: gmail, Hotmail, Yahoo! Mail, etc.

POP3 PROTOCOL

authorization phase

- client commands:
  - **user**: declare username
  - **pass**: password
- server responses
  - **+OK**
  - **-ERR**

transaction phase, client:

- **list**: list message numbers
- **retr**: retrieve message by number
- **dele**: delete
- **quit**

```
S:  +OK POP3 server ready
C:  user bob
S:  +OK
C:  pass hungry
S:  +OK user successfully logged on
```

```
C:  list
S:  1 498
S:  2 912
S:  .
C:  retr 1
S:  <message 1 contents>
S:  .
C:  dele 1
C:  retr 2
S:  <message 1 contents>
S:  .
C:  dele 2
C:  quit
S:  +OK POP3 server signing off
```

POP3 (MORE) AND IMAP

More about POP3

- Previous example uses "download and delete" mode.
- Bob cannot re-read e-mail if he changes client
- "Download-and-keep": copies of messages on different clients
- POP3 is stateless across sessions

IMAP

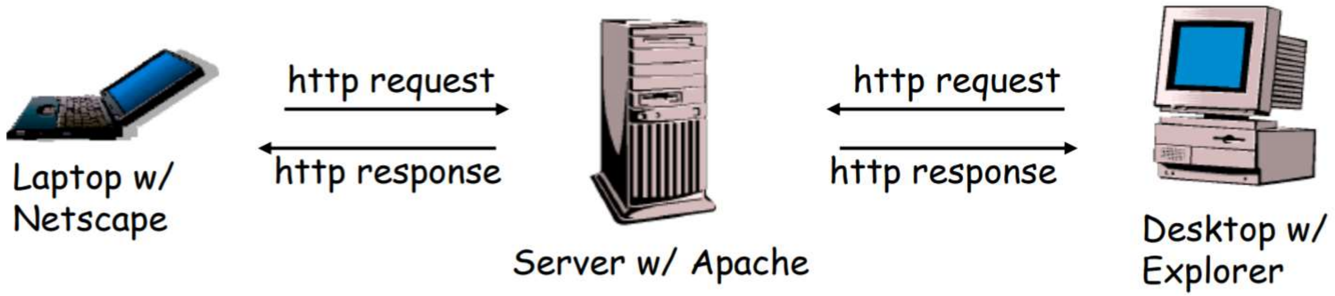
- Keep all messages in one place: the server
- Allows user to organize messages in folders
- IMAP keeps user state across sessions:
  - names of folders and mappings between message IDs and folder name

HTTP – HyperText Transfer Protocol

- http 1.1
- http 2.0

HTTP

- HTTP: HyperText Transfer Protocol
  - Communication protocol between clients and servers
  - Application layer protocol for WWW
- Client/Server model:
  - Client: browser that requests, receives, displays object
  - Server: receives requests and responds to them
- Protocol consists of various operations
  - HTTP 1.0 (RFC 1945, 1996)
  - HTTP 1.1 (RFC 2616, 1999)
  - HTTP 2.0 (RFC 7540, 8740)



HTTP

- Client downloads HTML document
  - Sometimes called “container page”
  - Typically in text format (ASCII)
  - Contains instructions for rendering
    (e.g., background color, frames)
  - Links to other pages

- Many have embedded objects:
  - Images: GIF, JPG (logos, banner ads)

```
<html>
<head>
<meta
name=“Author”
content=“Erich Nahum”>
<title> Linux Web
Server Performance
</title>
</head>
<body text=“#00000”>
<img width=31
height=11
src=“ibmlogo.gif”>
<img
src=“images/new.gif>
<h1>Hi There!</h1>
Here’s lots of cool
linux stuff!
<a href=“more.html”>
Click here</a>
for more!
</body>
</html>
```

HTTP Request Format

- Messages are in ASCII (human-readable)
- Carriage-return and line-feed indicate end of headers
- Headers may communicate private information
  - (browser, OS, cookie information, etc.)

```
GET /images/penguin.gif HTTP/1.0
User-Agent: Mozilla/0.9.4 (Linux 2.2.19)
Host: www.kernel.org
Accept: text/html, image/gif, image/jpeg
Accept-Encoding: gzip
Accept-Language: en
Accept-Charset: iso-8859-1,*,utf-8
Cookie: B=xh203jfsf; Y=3sdkfjej
<cr><lf>
```

HTTP Request Types

Called Methods:

- GET: retrieve a file (95% of requests)
- HEAD: just get meta-data (e.g., mod time)
- POST: submitting a form to a server
- PUT: store enclosed document as URI
- DELETE: removed named resource
- ….

HTTP Response Format

- Similar format to requests

```
HTTP/1.0 200 OK
Server: Tux 2.0
Content-Type: image/gif
Content-Length: 43
Last-Modified: Fri, 15 Apr 2020 02:36:21 GMT
Expires: Wed, 20 Feb 2021 18:54:46 GMT
Date: Mon, 12 Dec 2020 14:29:48 GMT
Cache-Control: no-cache
Pragma: no-cache
Connection: close
Set-Cookie: PA=wefj2we0-jfjf
<cr><lf>
<data follows…>
```

HTTP Response Types

- **1XX:** Informational

  `100 Continue, 101 Switching Protocols`

- 2XX: Success

  `200 OK, 206 Partial Content`

- 3XX: Redirection

  `301 Moved Permanently, 304 Not Modified`

- **4XX:** Client error

  `400 Bad Request, 403 Forbidden, 404 Not Found`

- **5XX:** Server error

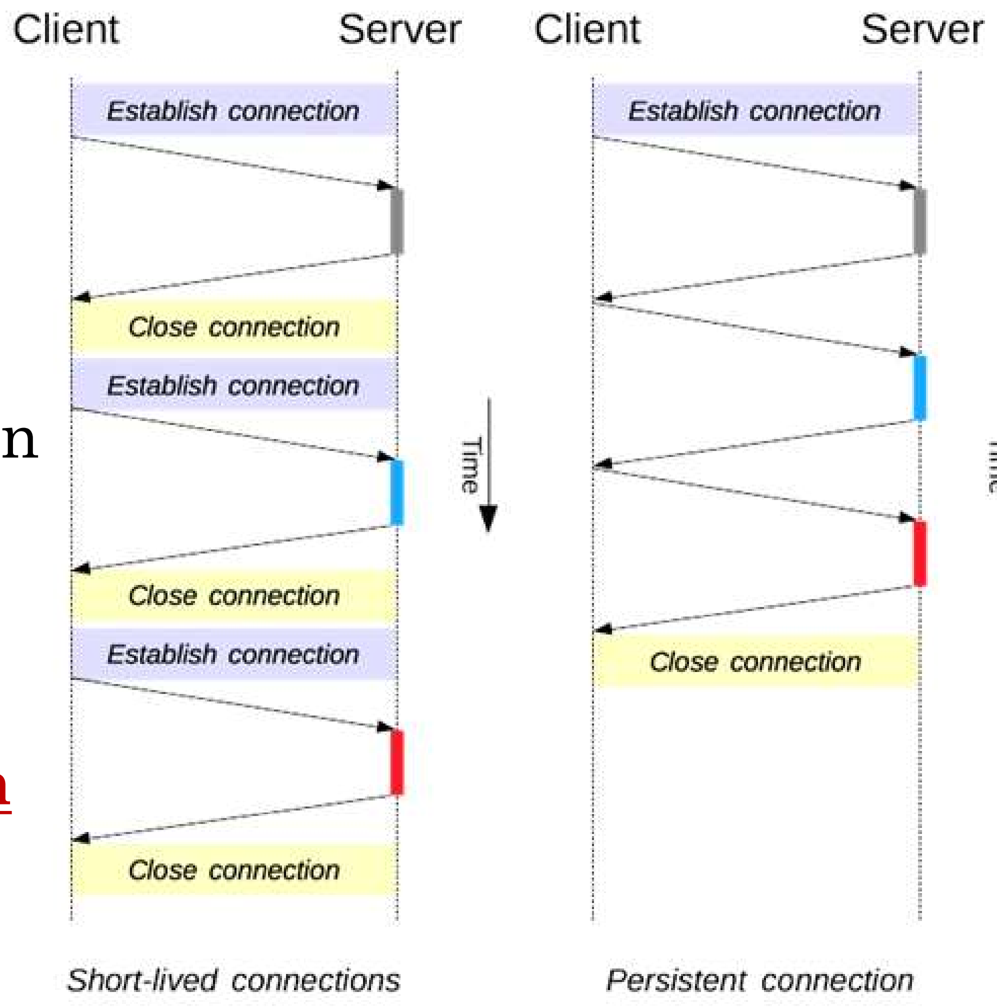
  `500 Internal Server Error, 503 Service Unavailable, 505 HTTP Version Not Supported`

HTTP 1.1

- Better than HTTP1.0
- Impose host identification
  - Virtual servers
- To speed up connections
  - Keep.Alive

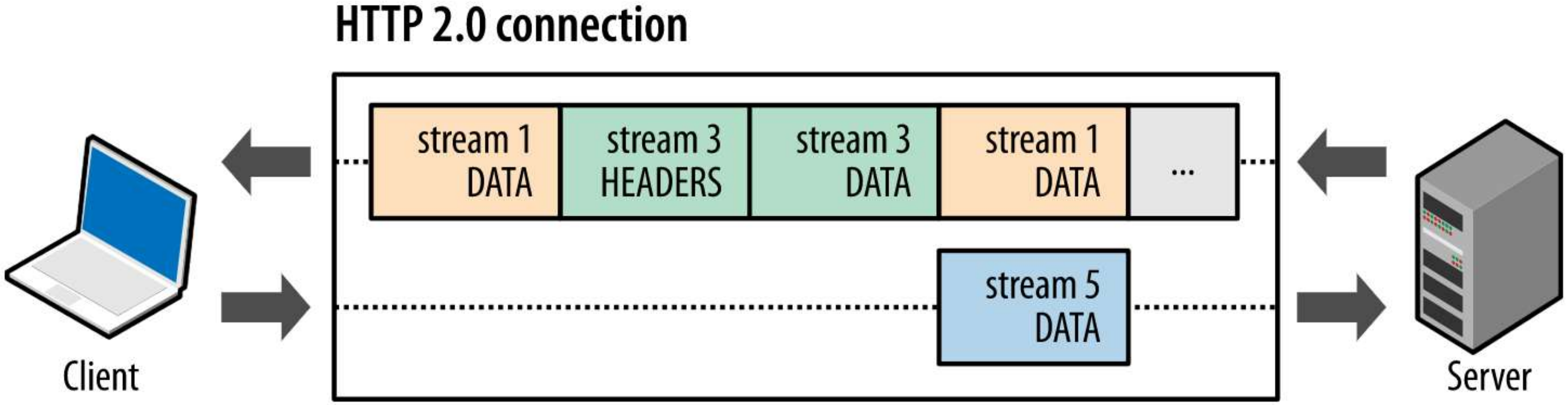
Persistent connection

- Low latency
- Stream with chunk
  - Pipielines



HTTP 2.0

- HTTP 2 is an HTTP 1.1 connection with some additional features
- One single and secured TCP connection is setup in which HTTP requests are transferred in from of streams

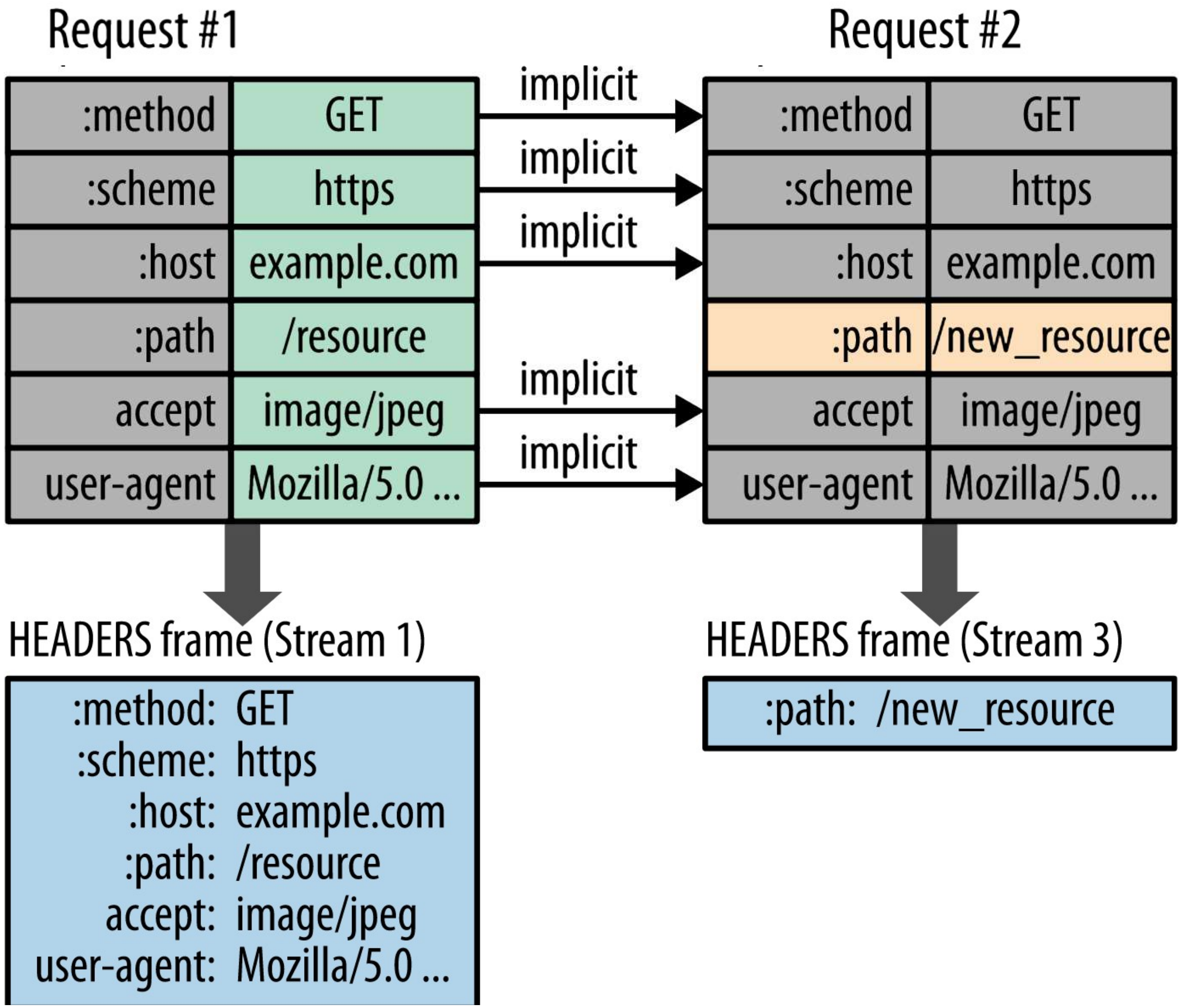


- Stream Multiplexing is used as one pipeline
- Secured TCP connection is used

HTTP 2.0 - FEATURES

- HPACK: Header Data is separate from Request Data and can be zipped
  - It allows the transmitted header fields to be encoded via a static Huffman code, which reduces their individual transfer size.
  - It requires that both the client and server maintain and update an indexed list of previously seen header fields, which is then used as a reference to efficiently encode previously transmitted values.
  - HPACK also enables reuse of Header data which is repeated in every request
- HPACK reduces HTTP request size
- PUSH: enables server to send mandatory resources in advance along with an HTTP response

HPACK



Request #1

:method	GET
:scheme	https
:host	example.com
:path	/resource
accept	image/jpeg
user-agent	Mozilla/5.0 ...

implicit

implicit

implicit

implicit

implicit

Request #2

:method	GET
:scheme	https
:host	example.com
:path	/new_resource
accept	image/jpeg
user-agent	Mozilla/5.0 ...

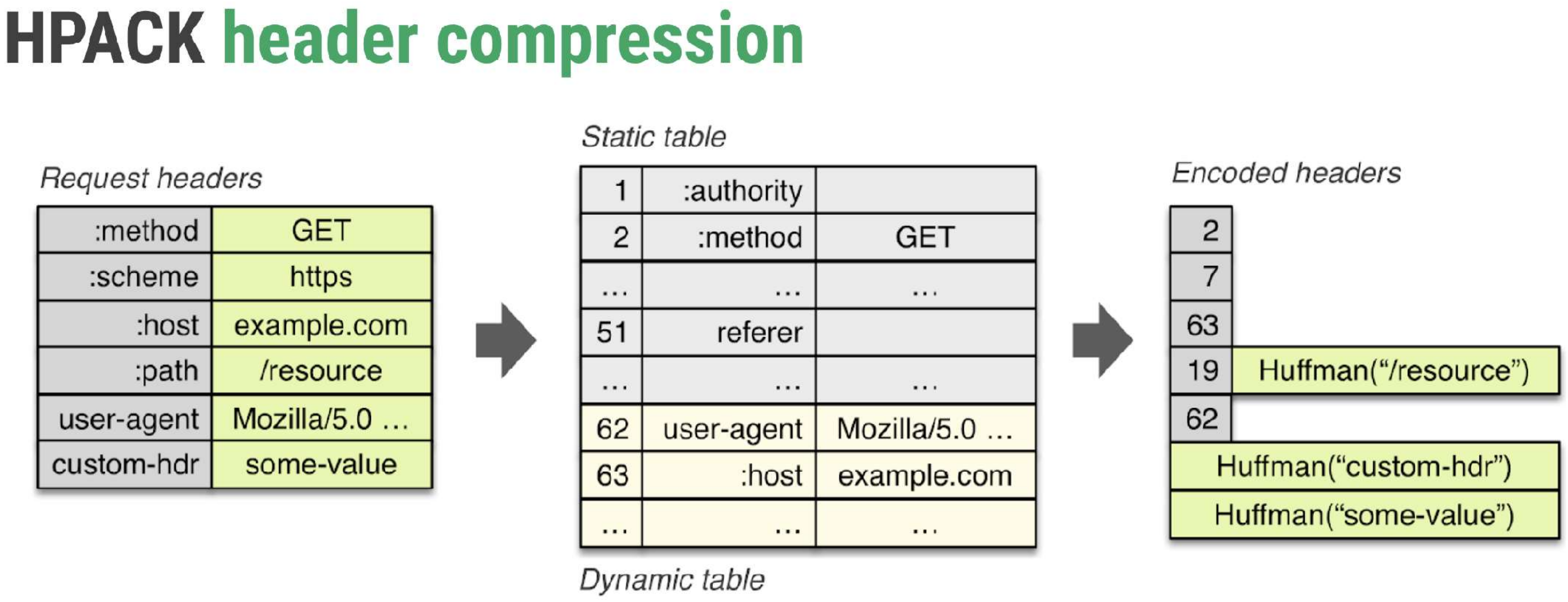
HEADERS frame (Stream 1)

:method:	GET
:scheme:	https
:host:	example.com
:path:	/resource
accept:	image/jpeg
user-agent:	Mozilla/5.0 ...

HEADERS frame (Stream 3)

:path:	/new_resource

HPACK



HPACK header compression

Request headers

:method	GET
:scheme	https
:host	example.com
:path	/resource
user-agent	Mozilla/5.0 …
custom-hdr	some-value

Static table

1	:authority	
2	:method	GET
…	…	…
51	referer	
…	…	…
62	user-agent	Mozilla/5.0 …
63	:host	example.com
…	…	…

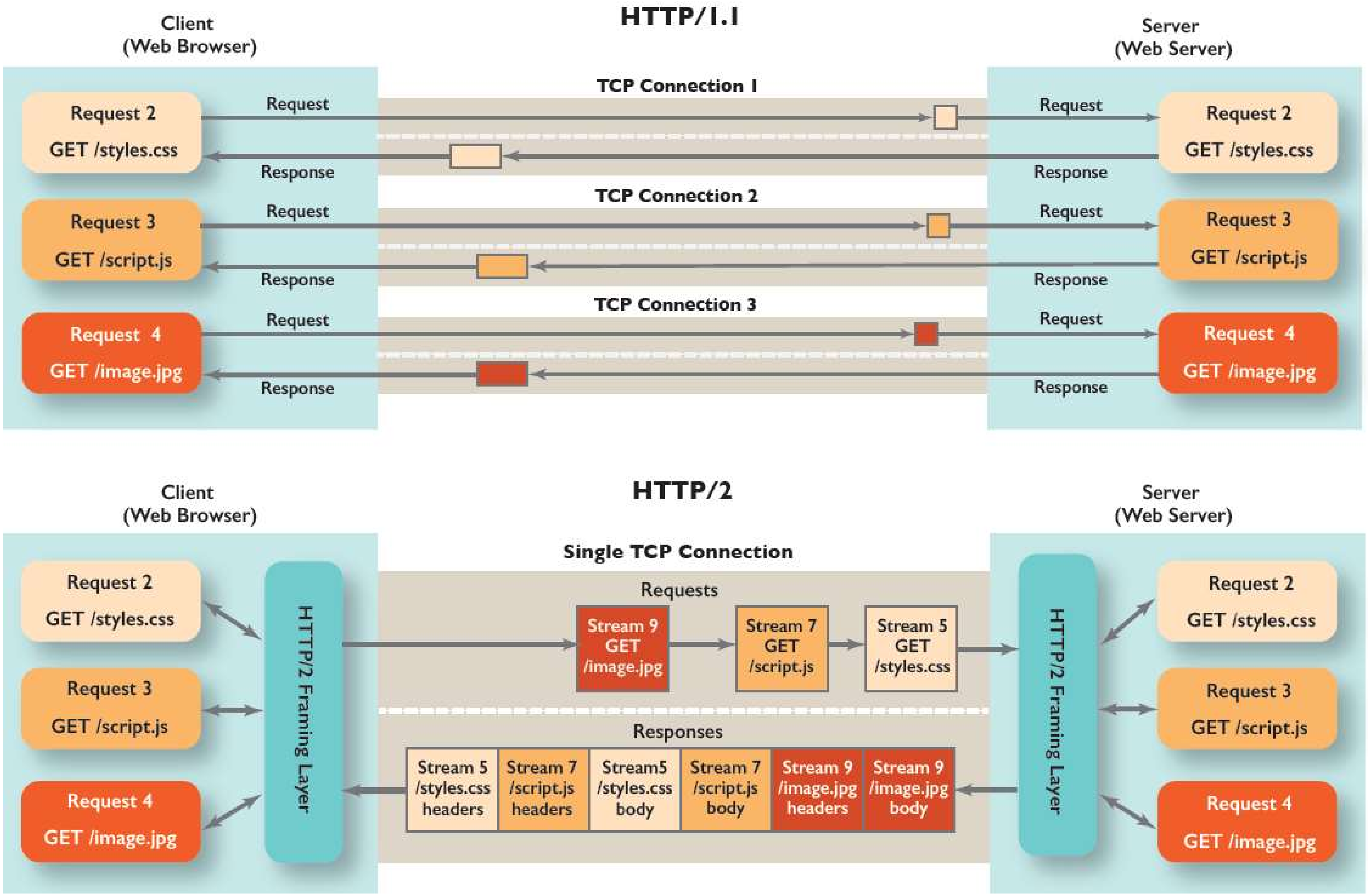
Dynamic table

Encoded headers

2	
7	
63	
19	Huffman("/resource")
62	
Huffman("custom-hdr")	
Huffman("some-value")	

- Previously sent values are (optionally) indexed
  - « 2 » in this example expands to « method: GET »

SUMMARY



HTTP/1.1

Client (Web Browser)

Server (Web Server)

TCP Connection 1

Request 2 GET /styles.css — Request → / ← Response — Request 2 GET /styles.css

TCP Connection 2

Request 3 GET /script.js — Request → / ← Response — Request 3 GET /script.js

TCP Connection 3

Request 4 GET /image.jpg — Request → / ← Response — Request 4 GET /image.jpg

HTTP/2

Client (Web Browser)

Server (Web Server)

Request 2 GET /styles.css

Request 3 GET /script.js

Request 4 GET /image.jpg

HTTP/2 Framing Layer

Single TCP Connection

Requests

Stream 9 GET /image.jpg → Stream 7 GET /script.js → Stream 5 GET /styles.css

Responses

Stream 5 /styles.css headers | Stream 7 /script.js headers | Stream5 /styles.css body | Stream 7 /script.js body | Stream 9 /image.jpg headers | Stream 9 /image.jpg body

HTTP/2 Framing Layer

Request 2 GET /styles.css

Request 3 GET /script.js

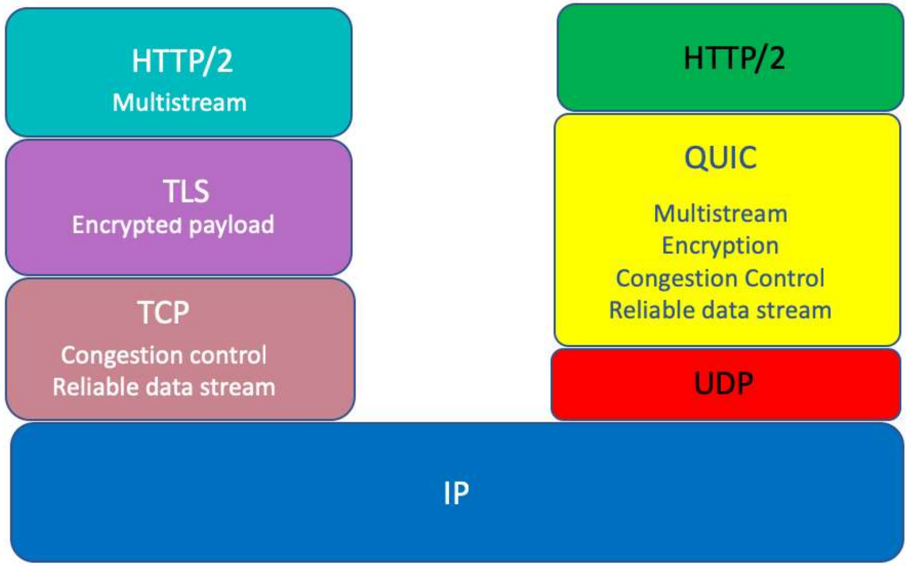
Request 4 GET /image.jpg

HTTP3/QUIC PROTOCOL

Quick UDP Internet Connections

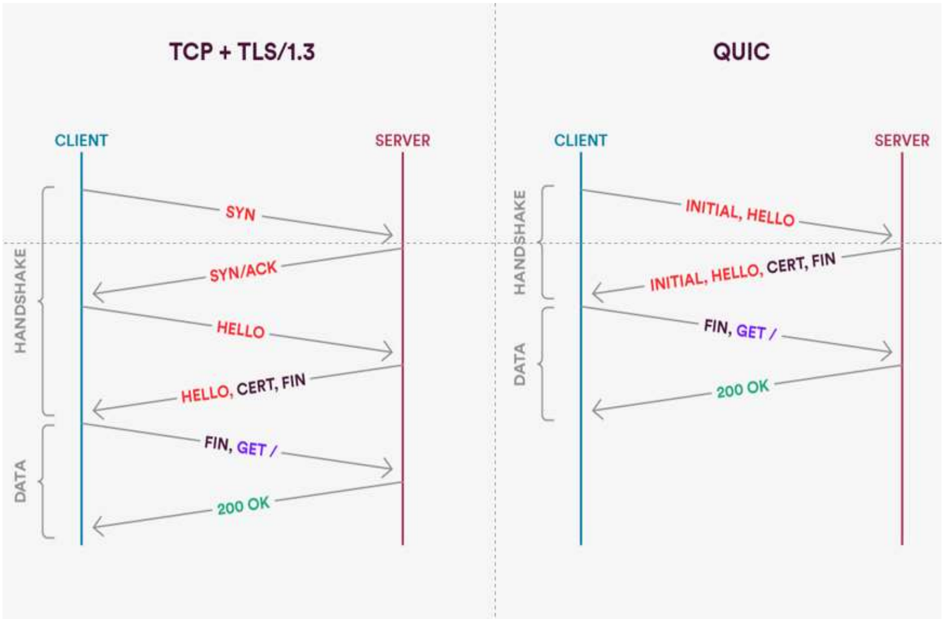
HTTP3/QUIC

- QUIC (Quick UDP Internet Connection) is a new encrypted transport layer network protocol. QUIC was designed to make HTTP traffic more secure, efficient, and faster.



QUIC - FEATURES

- **Reduced connection times**: To establish TLS encryption, the client and the server need to perform a TLS handshake and exchange encryption keys. QUIC replaces all of this with a single handshake.



Better performance when data packets are lost. A head-of-line blocking is happen. Quic Resolves it by allowing streams of data to reach their destination independently.

Stable connections when networks are changed. If you are connected to a web server via TCP and your network suddenly changes (from Wi-Fi to 4G, for example), each connection times out and needs to be reestablished. QUIC allows for a smoother transition by giving each connection to a web server a unique identifier.

LDAP Protocol

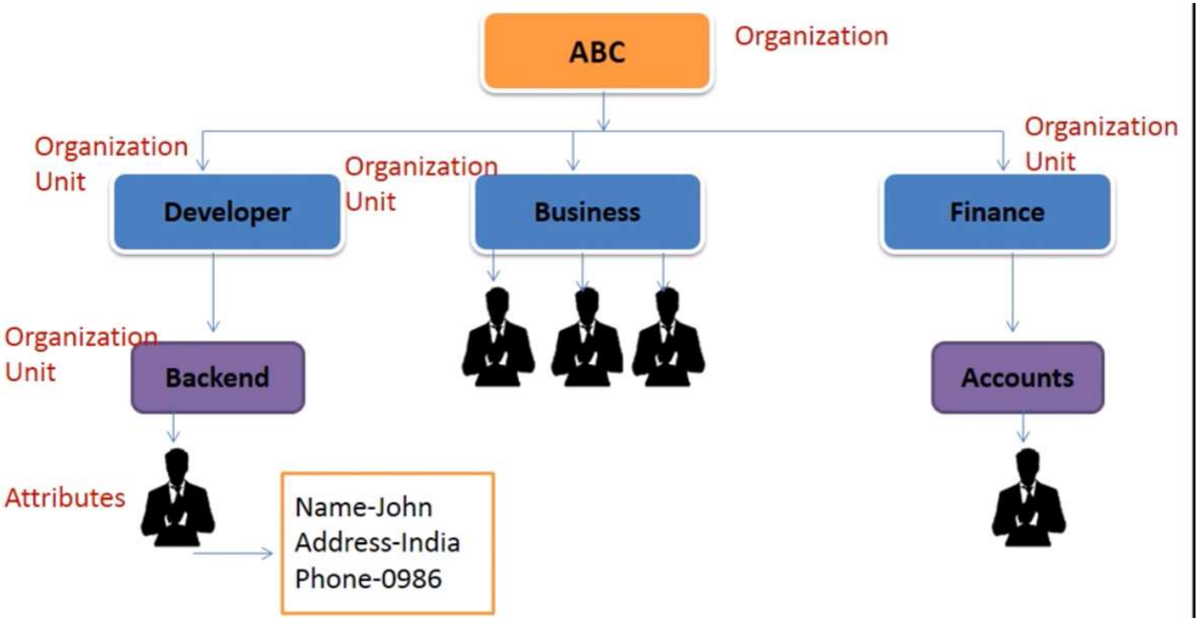
Lightweight Directory Access Protocol

LDAP

- LDAP is an application protocol for querying and modifying items is Directory service.
- LDAP defines the content of messages exchanged between an LDAP client and an LDAP server. The messages specify :
  - the operations requested by the client (that is, add, search, modify, and delete),
  - the responses from the server,
  - and the format of data carried in the messages
- LDAP is based on a client/server model
- Directory service Agent is used to connect to the server
- The service is identified by TCP and port 389.

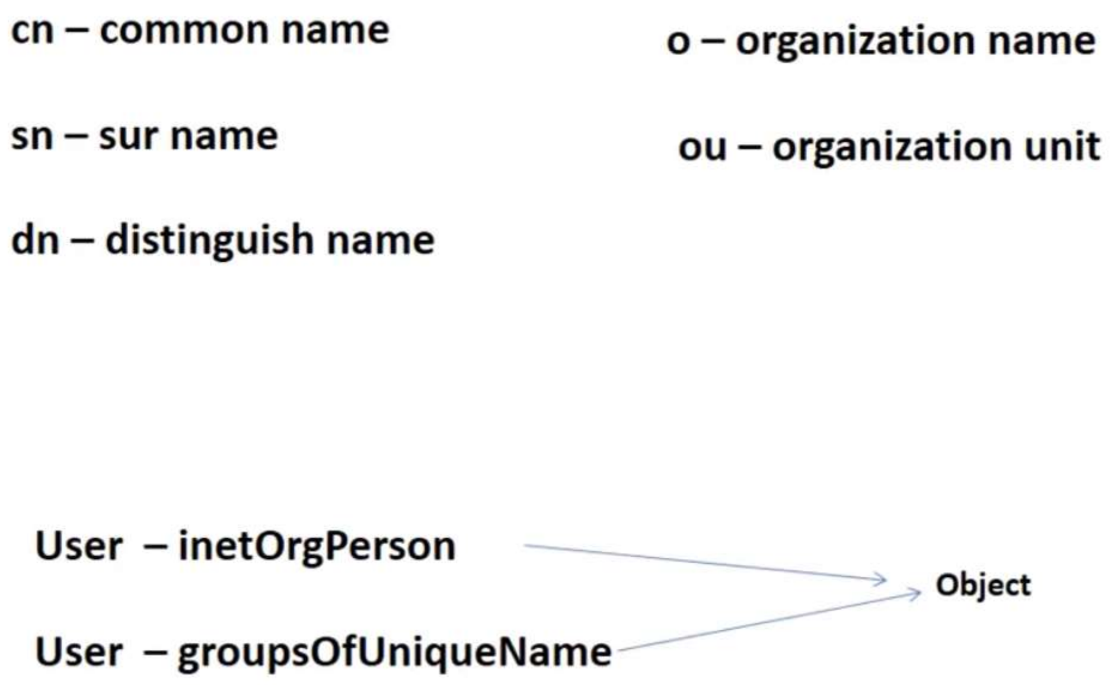
LDAP

- The directory stores and organizes data structures known as entries.
- A directory entry usually describes an object in a hierarchical structure, such as a person, device, a location, and so on.
- Each entry has a name called a distinguished name (DN) that uniquely identifies it.



TERMS

- Entries are composed of a collection of attributes that contain information about the object. Every attribute has a type and one or more values
- LDAP uses somes terms



cn – common name

sn – sur name

dn – distinguish name

o – organization name

ou – organization unit

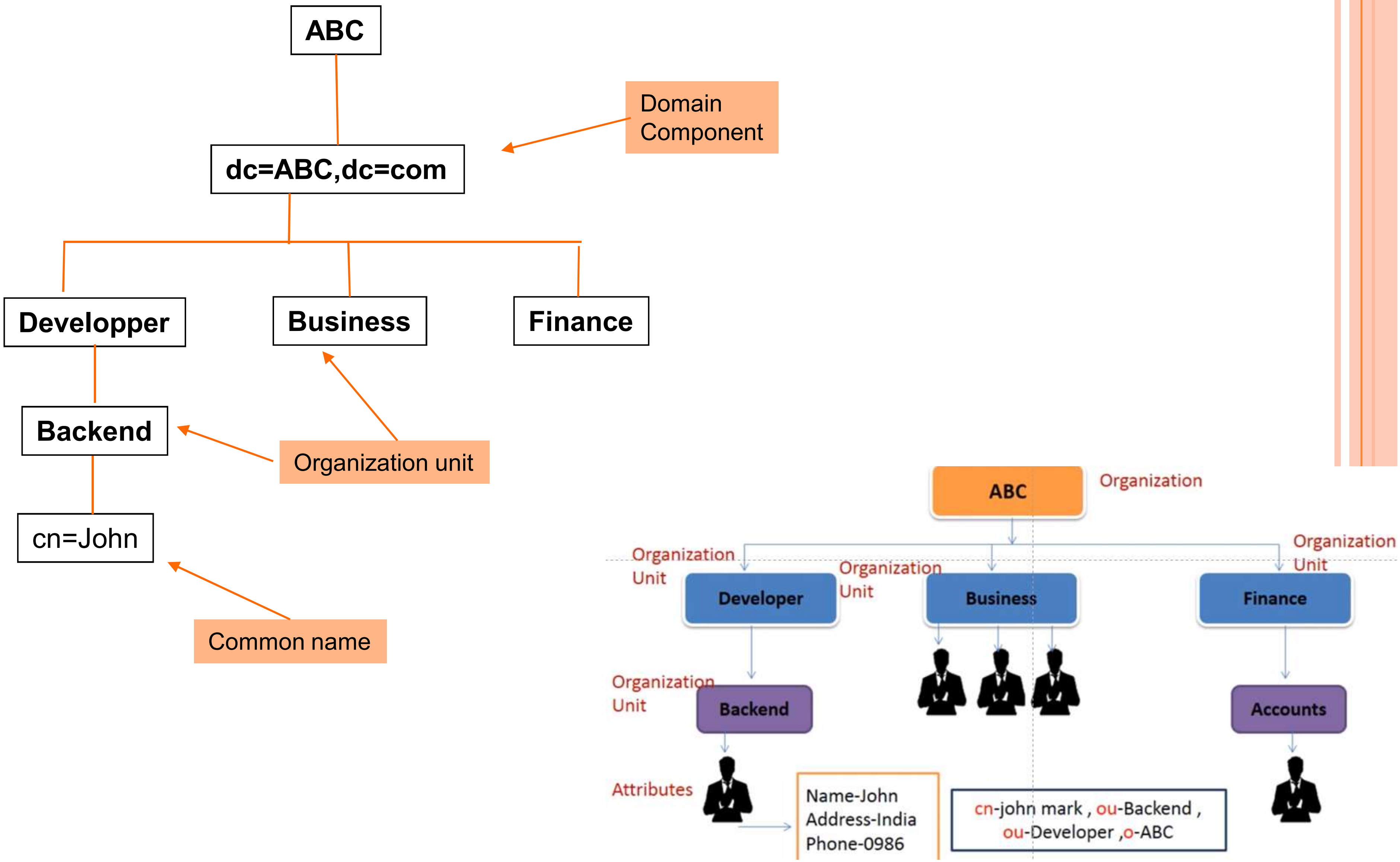
User – inetOrgPerson

User – groupsOfUniqueName

Object

EXAPME

- How to identify an attribute



ABC

dc=ABC,dc=com

Domain Component

Developper

Business

Finance

Backend

Organization unit

cn=John

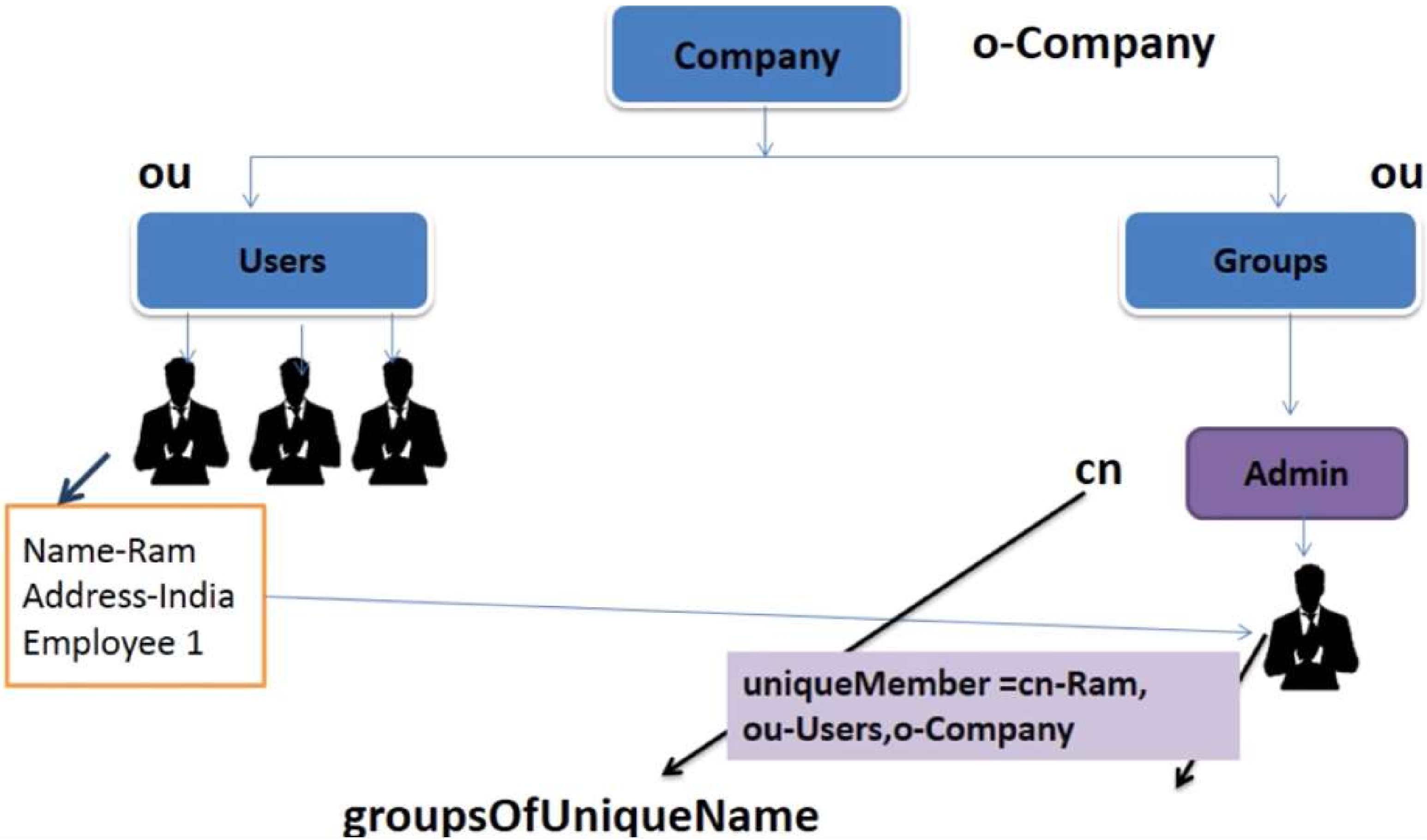
Common name

Name-John
Address-India
Phone-0986

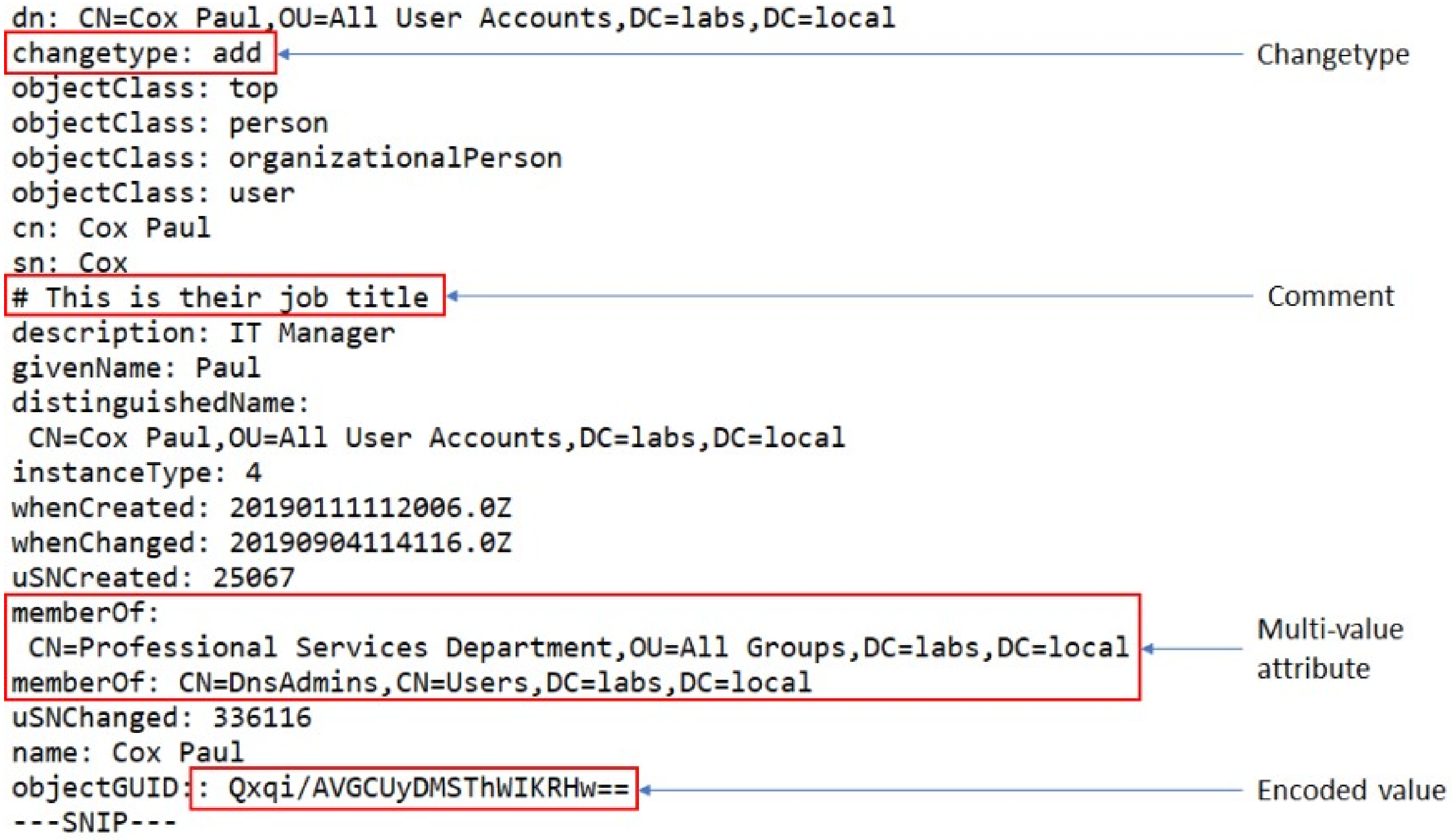
cn-john mark , ou-Backend , ou-Developer ,o-ABC

EXAMPLE

- Attribute beloging to differents OU



LDIF FORMAT



```
dn: CN=Cox Paul,OU=All User Accounts,DC=labs,DC=local
changetype: add
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: user
cn: Cox Paul
sn: Cox
# This is their job title
description: IT Manager
givenName: Paul
distinguishedName:
 CN=Cox Paul,OU=All User Accounts,DC=labs,DC=local
instanceType: 4
whenCreated: 20190111112006.0Z
whenChanged: 20190904114116.0Z
uSNCreated: 25067
memberOf:
 CN=Professional Services Department,OU=All Groups,DC=labs,DC=local
memberOf: CN=DnsAdmins,CN=Users,DC=labs,DC=local
uSNChanged: 336116
name: Cox Paul
objectGUID:: Qxqi/AVGCUyDMSThWIKRHw==
---SNIP---
```