



UNIVERSIDAD  
COMPLUTENSE  
MADRID

# Smart Objects Interconnection with IP

## Review of basic concepts

Networks and Protocols

*Based on: Interconnecting Smart Objects with IP. The next Internet*

# Challenges in IoT

## General overview

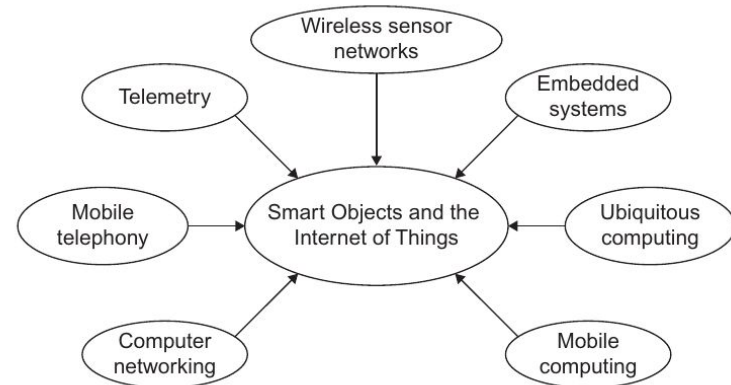
- **Smart Object**

- Object equipped with a a sensor and/or actuator: interaction with real world
- Microprocessor limited in compute capabilities: basic transformations
- Communication technologies: data extraction and submission to external networks

**Interaction + (Processing) + Communication**

- **Areas of interest (intersection between):**

- Embedded systems
- Ubiquitous computing
- Mobile telephony
- Telemetry
- M2M communication
- Sensor networks. Ubiquitous sensorization
- Mobile computing
- Computer networks



- **Specific challenges**
  - **Node level**
    - Energy consumption, size, cost
  - **Network level**
    - High dimensionality (number of objects)
  - **Standardization.** Manufacturer interests
  - **Interoperability.** Even using different underlying technologies
- **Discussion:**
  - **Implications on routing algorithms?**
    - **Think about well-known protocols (RIP, OSPF, ...)**
      - **Computing requirements, memory, generated traffic, availability, losses,**  
...
    - **Lossy networks**
    - **Are current routing protocols ready for IoT?**
    - **Administration: manual or automatic? (*bootstrapping*)**

- **Smart Object: Embedded system + Communication**
  - **Any real-time requirement?**
    - This is a common requirement in Embedded Systems. Not always in IoT
  - **Common properties:**
    - Hardware type
    - Energy consumption restrictions
    - Limitations in compute capabilities
      - E.g. floating point support

- **IoT gathers a plethora of technologies:**
  - Global System for Mobile communications (GSM)
  - General Packet Radio Service (GPRS), Enhanced Data
  - Rates for GSM Evolution (EDGE)
  - and Universal Mobile Telecommunications System (UMTS)
  - Bluetooth (IEEE 802.15.1)
  - NFC

- **Revisiting routing protocols:**
  - **Centralized vs. distributed routing**
    - Routing server (at origin) vs. individual decisions per node
  - **Convergence speed**
    - E.g. Mesh with LoRA. Unfeasible convergence time (always?)
  - **Stability**
    - Tolerance for node failure (even voluntary, not due to errors)
  - **Fault tolerance**
    - Routing information should be redundant/reliable
  - **Compute capabilities vs. generated traffic**
    - Remember: RIP vs. OSPF
    - Should be efficient from the traffic and computational perspectives
  - **Lossy**
    - Routing should consider loss probability

## GATEWAYS...

- **What about addressing?**
  - **IPv6 is the response to the lack of IP addresses but...**
    - Is that true? Is it truly necessary?
  - **Do all nodes need to be addressable world-wide?**
    - **Many times, no**
    - **Others, it is just not supported**
      - E.g. NB-IoT typically uses NAT
      - Limits communication to one direction (Sensor -> Internet)
- **Network management**
  - How do we manage a network if we cannot reconfigure nodes?
  - Bootstrapping
  - OTA updates

## GATEWAYS...









- **IoT / Smart Objects are relatively new technologies**

- Not only gather many types of objects...
- ...each one managed by a company...
- ...that wants its **piece of cake**.

- **Example:**

- LPWAN alternatives

	 LoRa	 sigfox	<b>NB-IoT</b>	 <b>lte</b>	 INGENU	 WEIGHTLESS	 LinkLabs
	LoRa / LoRaWAN	Sigfox	NB-IoT	LTE-M	RPMA	Weightless-P	Symphony Link
Origin	France	France	USA (Global)	USA (Global)	USA	UK	USA
Proprietary or open	LoRa – proprietary LoRaWAN - open	Net – proprietary Devices – open	Open	Open	Proprietary	Open	Proprietary
Cellular	No	No	Yes	Yes	No	No	No
Spectrum	Unlicensed	Unlicensed	Licensed	Licensed	Unlicensed	Unlicensed	Unlicensed
Range, km	urban: 2-5 rural: 15	urban: 3-10 rural: 30-50	urban: 1-5 rural: 10-15	urban: 2-5	urban: 1-3 rural: 25-50	urban: 2	urban: 2-5 rural: 15
Speed, uplink / downlink	50 kbps / 50 kbps	300 bps / –	250 kbps / 250 kbps	1 Mbps / 1 Mbps	634 kbps / 156 kbps	100 kbps / 100 kbps	100 kbps / 100 kbps
Power consumption	●●●	●	●	●●●	●●	●	●●
Security	●●	●●	●●●	●●●	●●●	●●●	●●●
Availability of devices	●●	●●●	●●	●	●●	●	●●
Price*	●●	●	●●	●●●	●●●	●	●●
Areas of application	Precision farming, manufacturing automation, pipeline monitoring	Predictive maintenance, capacity planning, demand forecasting	Electric metering, manufacturing automation, retail PoS	tracking objects, wearables, energy management, utility metering, city infrastructure	Digital oilfield, connected cities, usage-based insurance, agriculture	Smart grid, healthcare, automotive, smart cities, asset tracking	Industrial control systems, lighting control, alarm systems
Supporting companies	IBM, Semtech, Cisco, HP, Orange, Kerlink, Actility	STMicroelectronic, Texas Instruments, Atmel, Silicon Labs	Huawei, Ericsson, Qualcomm, Vodafone	Verizon, AT&T, Nokia	Ingenu	Accenture, Sony Europe, uniik, ARM, Telensa	Link Labs

**GATEWAYS...**

- **Cons for the lack of standardization**
  - Portability
  - Lack of ability to upgrade deployments
  - Maintainability problems
  - Bigger acquisition costs
- **Open standards vs. de-facto proprietary standards**
- **In IoT, this problems arises at all levels**

- **Derived/related with standardization**
  - Easy interaction between devices from different manufacturers
  - Usually problems at physical levels, but also at other levels:
    - Application (protocols)
    - Networking (addressing)
- **Three challenges:**
  - Architectural: e.g. TCP/IP
  - Reuse of existing standards
  - Development of testsuites. As of today, nonexistent.
    - E.g. IPSO Alliance in Smart Object definition

# The IP architecture. Motivation for IoT

- **Initial TCP/IP goals (1981): (sic)**
  - Communication needs to go on beyond **failures** in networks or gateways
  - Internet must support **multiple service types** for communication
  - The architecture of Internet must:
    - Accommodate variety of networks (link and physical layers)
    - Allow for distributed resource management
    - Be cost-efficient
    - Allow for additional host addition with little effort
- **Initially, proposed as a unique protocol:**
  - Discussion: Pros/cons?
  - Response: layered design
    - No assumptions about low-level layers

**Lighter, less efficiency. Dave Clark, 1988:**

*Since Internet does not insist that lost packets be recovered at the network level, it may be necessary to retransmit a lost packet from one end of the Internet to the other. This means that the retransmitted packet may cross several intervening nets a second time, whereas recovery at the network level would not generate this repeat traffic. This is an example of the trade-off resulting from the decision, discussed above, of providing services from the end-points. The network interface code is much simpler, but the overall efficiency is potentially less.*

- **An example: division between TCP <-> IP**
  - Type of service: reliable / unreliable
  - Reliability implemented in multiple layers:
    - Link layer
    - Application layer
  - Lighter routing
  
- **Could TCP/IP fulfill all goals?**
  - Simplicity: yes and no.
    - It is necessary to know the features of each protocol at each layer
  - Management: no.
    - Using external protocols
  - Resource management: no.
    - Use of RSVP for resource reservation a priori and to guarantee QoS

- **Layered design. Goals covered?**
  - Flexibility
  - Reliability/fault tolerance
  - Footprint. Supported devices
  - Multiple types of services
- **Cons:**
  - **Many times, it is necessary to break layer separation**
    - Security (package inspection)
  - **In Smart Objects/IoT, temptation of performing cross-layer optimizations**
    - E.g. joining link and network layers. Routing at Level 2
    - Reason: less complexity, less maintenance, higher efficiency, less overhead, less energy consumption...
    - Necessary? No. IPv6 implementations are usually light enough

- **Layered design. Is it also important for IoT?**

- Again, temptation to implement everything in a monolithic fashion
- Proprietary protocols
- A protocol stack for each service type required
- **Pros:**
  - **Optimal...only for a target**
- **Cons:**
  - **Lack of flexibility**
    - Dependence on link layer
      - 802.15.4 originally
      - Today, a plethora of proprietary link layers
    - Dependence on applications
      - Reimplementation necessary depending on lower layers!!



# Why IP for Smart Objects?

- **Easy evolution**
  - Adaptation to new challenges and unknown applications
- **Scaling**
  - Potentially unlimited number of nodes
- **Application diversity**
  - Present and future
- **Diversity in communication technologies**
  - The same object should be able to work with any underlying technology
- **Interoperability**
- **Standardization**
- **Support for lossy networks**
- **Node lifetime**
  - Energy consumption, SW/HW maintenance
- **Cost**
  - HW...and SW

**Discussion: examples in which IP adapts to these challenges**

## GATEWAYS!!

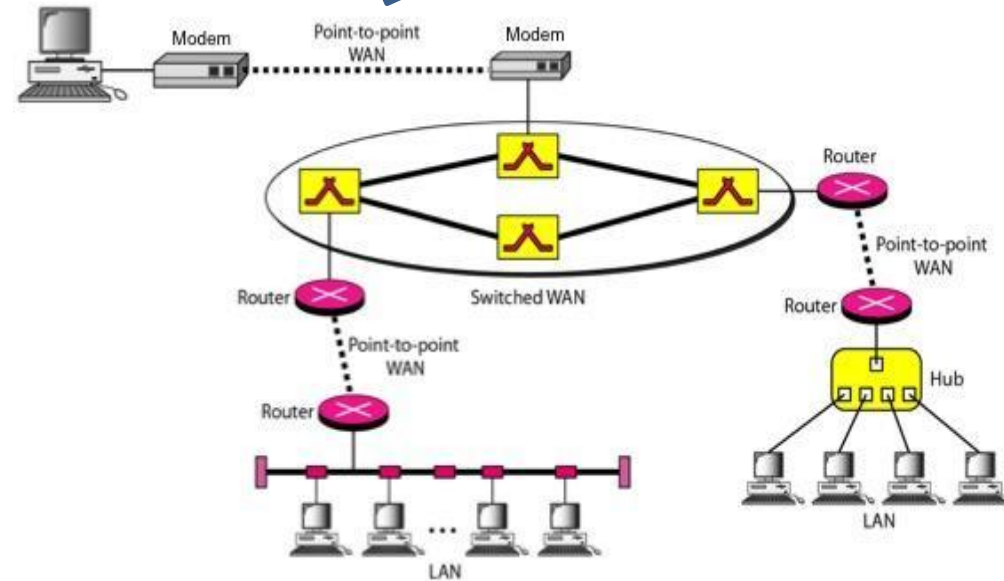
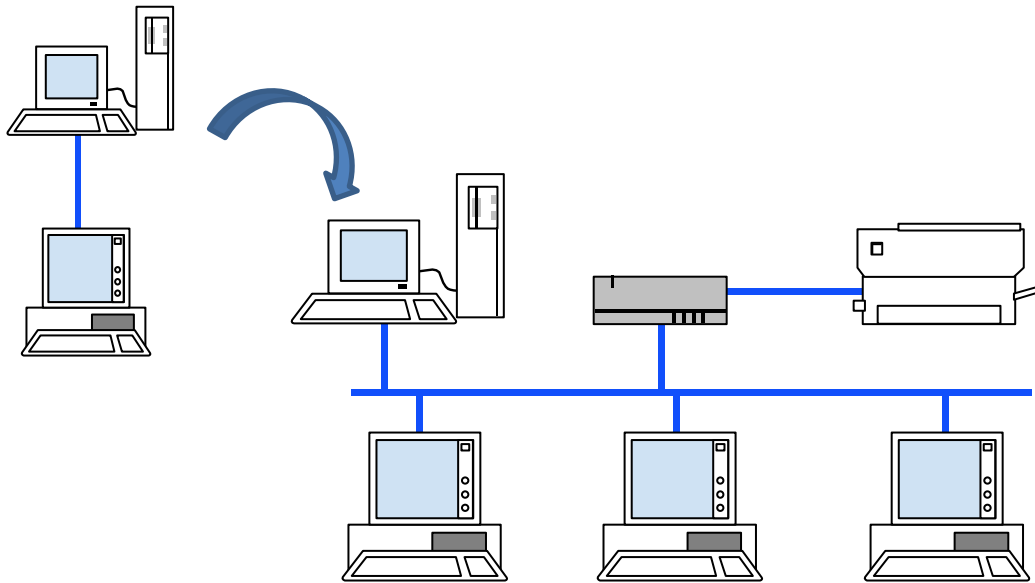
Are they good?

Are they bad?

Why do we agree they need to exist?

- **Inherent complexity**
  - **Multi-protocol gateways**
    - Costly protocol translation
    - QoS semantics are different for each network
    - Error recovery
    - Different transportation services across protocols
    - Security, management, administration...
- **Question:**
  - What if Edge Computing was just an excuse to justify the use of complex gateways?
- **Lack of flexibility and scalability**
  - Gateway is the bottleneck
  - Reliability in doubt
- **In the 90s, Multiprotocol Gateways made sense**
  - With IP...do they now?

# Review of basic concepts



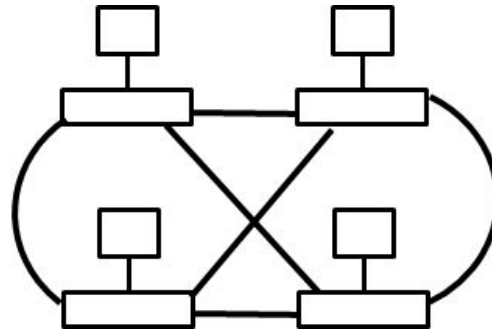
**How to manage it?**

- **Network scalability**
  - Let's migrate from one node to millions (or billions)
  - How to handle interconnection?
- **Information delivery**
  - How to physically reach one device from any other?
- **Element heterogeneity**
  - What if not always two-wire copper cable is used?
  - Each sub-network can feature different interconnection (physical) technology: copper, fiber, wireless...
  - How to develop something working in all cases?
- **Reliability**
  - How to guarantee no information is lost?
  - How to solve such a situation?
- **Privacy/security**

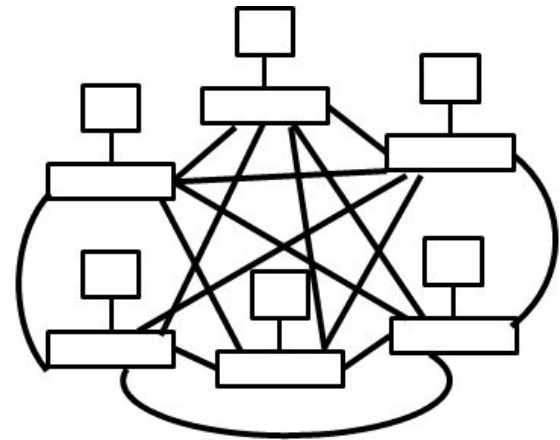
- Easiest way to connect two devices
- Increasing the number of devices increases exponentially the number of links
  - $n$  devices:  $n(n-1)/2$  links
    - **Problem:** Amount of wires and I/O ports per device
    - **Solution:** switched LANs



$n=2 \rightarrow 1$  enlace



$n=4 \rightarrow 6$  enlaces



$n=6 \rightarrow 15$  enlaces



- **LAN = Local Area Network**

- Private character
- Limited range
  - Interconnects devices in an office, building, home...
- Each device has a unique identifier in the network: network address
  - Messages are labelled by origin and destination

- **LAN types**

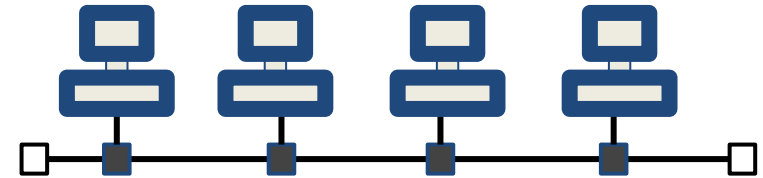
- **Broadcast LAN**
  - Devices interconnected by a shared transmission medium
  - When a device needs to send information, it broadcasts it to any other across the shared media
  - If two or more devices transmit simultaneously, a collision arises and information is invalid
- **Switched LAN**
  - Devices interconnected through a switch
  - Information is sent only to the destination
  - No collisions

- **Classic LAN topologies**

- Broadcast LAN

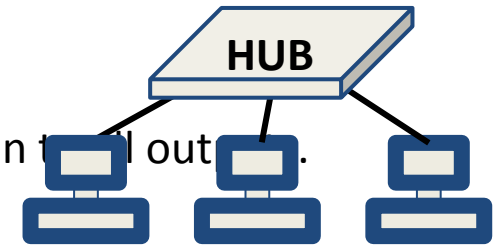
- Como wire (bus)

- Example: Ethernet 10Base2



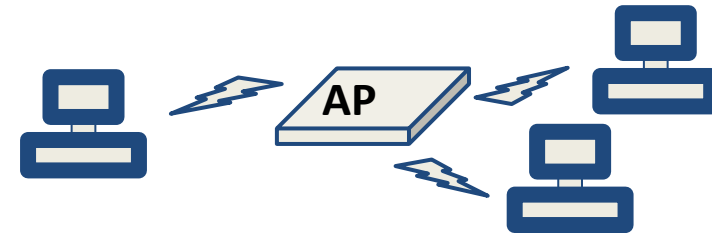
- Hub (star)

- The hub is a repeater that broadcasts information to all out.
- Example: Ethernet 10Base-T



- Wireless LAN (WLAN)

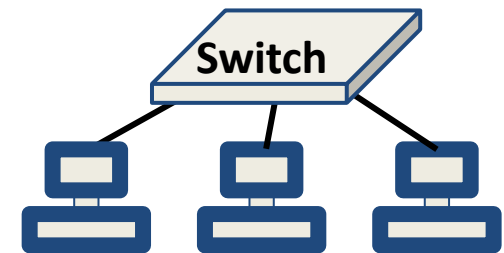
- The AP (Access Point) is a wireless hub
- Example: Wi-Fi network



- Switched LAN

- Switch (star)

- Ejemplo: Fast Ethernet 100BASE-TX



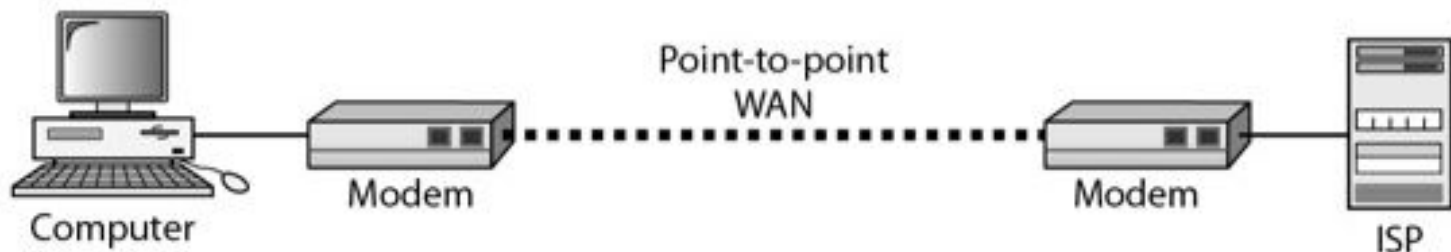
- **WAN = Wide Area Network**

- Larger geographical area (city, country, planet)
- Usually public, managed by communication companies

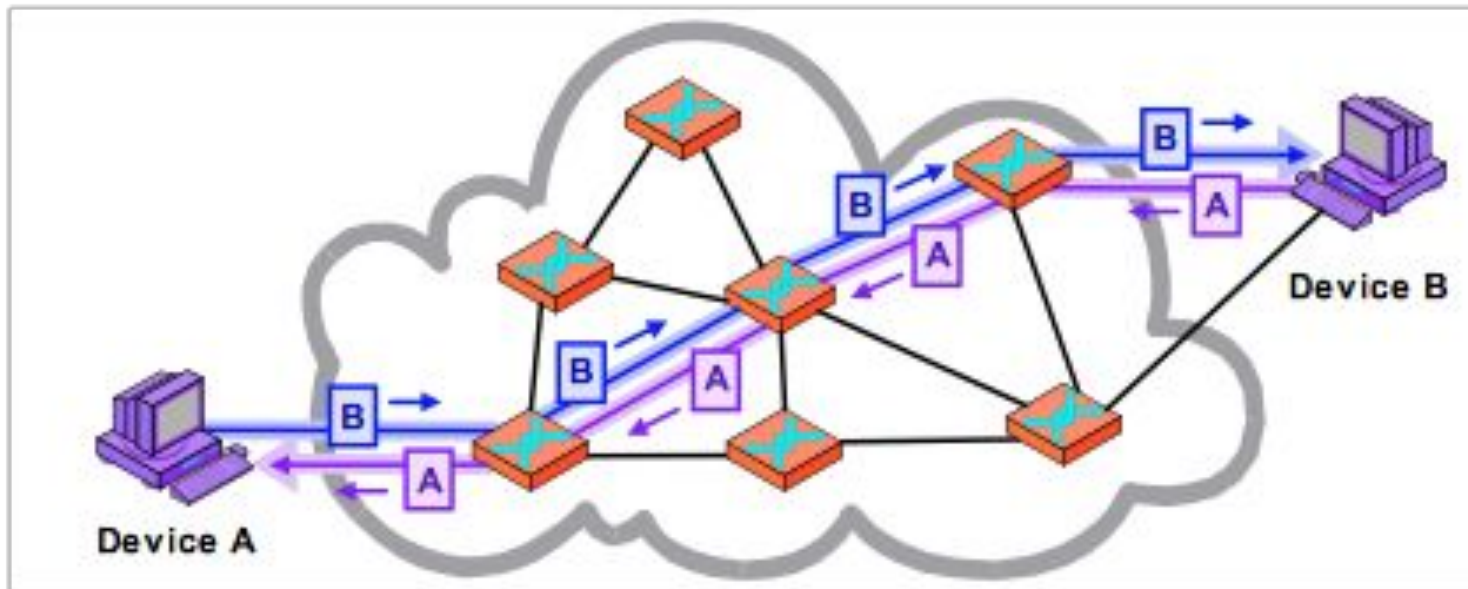
- **WAN types**

- **WAN point to point**

- Connect two devices via transmission medium (air, cable)
- Example: Conventional modem connection or ADSL between domestic computer and ISP



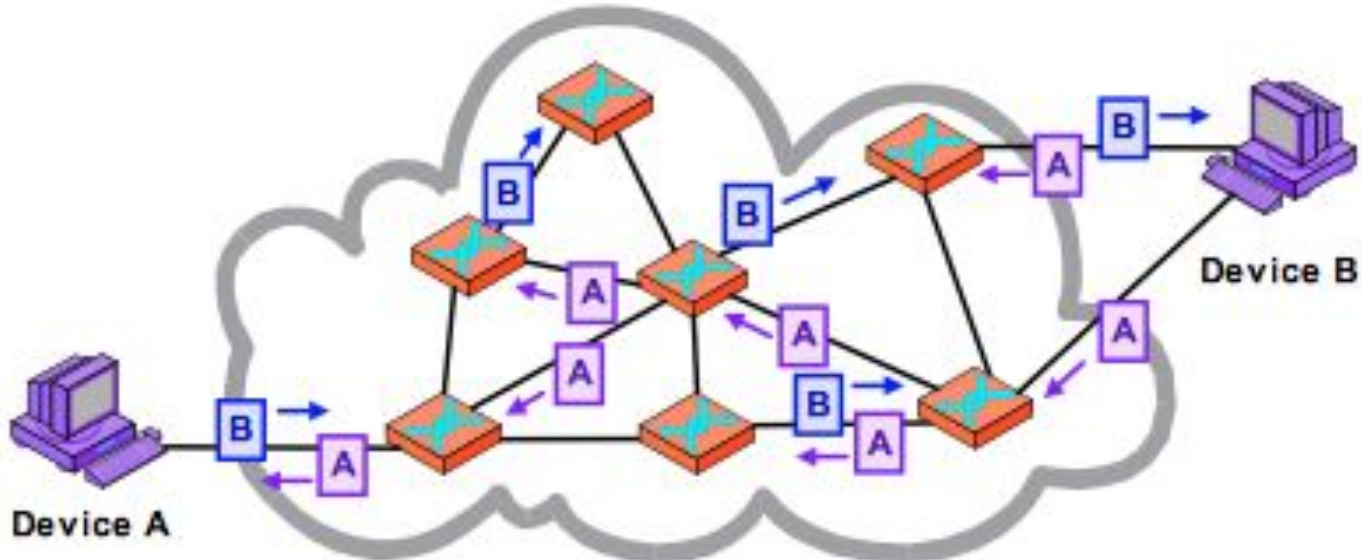
- **WAN (cont.)**
  - **Circuit Switched WAN**
    - Dedicated connection (circuit) between both ends
    - Switches do not process information
      - Only establish the necessary circuits for communication
    - Example: Conventional phone network (PSTN)
      - PSTN = Public Switched Telephone Network



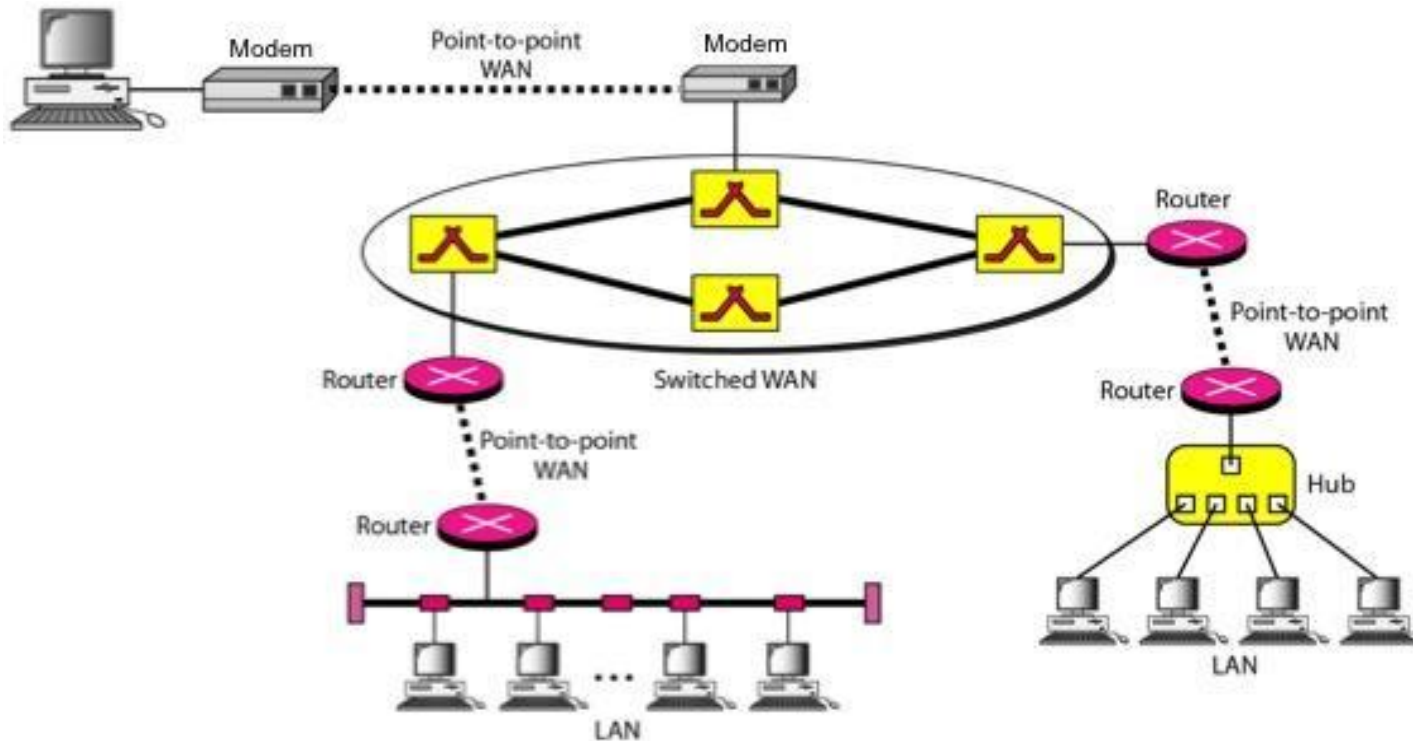
- WAN (cont.)

- Packet Switched WAN

- Information is divided into blocks (packets)
- Switches process packets, with two main functions:
  - **Packet routing:** decide the optimal route from source to destination
  - **Packet forwarding:** based on routing information, resend the packet to the next node, till the destination



- Network scalability
  - Creation of a network of networks (*internet*)
  - Interconnected nodes (forming LANs) joined together (by WANs) with other similar networks
  - Specific-purpose machines (*routers*) to organize traffic

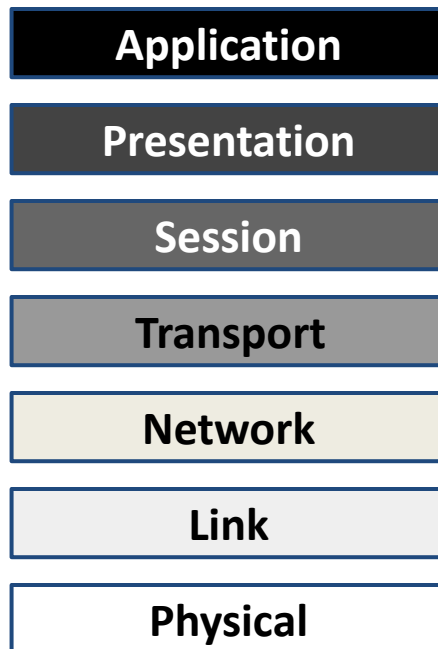


- Information delivery
  - Necessity of addressing each device
    - What does it mean?
  - Necessity of routing protocols
    - Both for circuit or packet switching
    - How to get from A to B? Which is the optimal path?
    - Maybe the algorithms need to adapt to a dynamic network (new nodes, nodes that disappear, ...)

- Heterogeneity
  - How to make applications work in such a heterogeneous environment?
  - Our applications: do they need to know all possible physical mechanisms between devices?
    - Can we modularize to reuse and focus on higher-level issues?



- **Modularization proposal: layered design**
  - Standard developed by ISO (International Organization for Standardization)
  - The OSI model (Open Systems Interconnection) handles network intercommunication issues -70s-
  - Goal: allow for an intercommunication of two independent systems with different underlying technologies

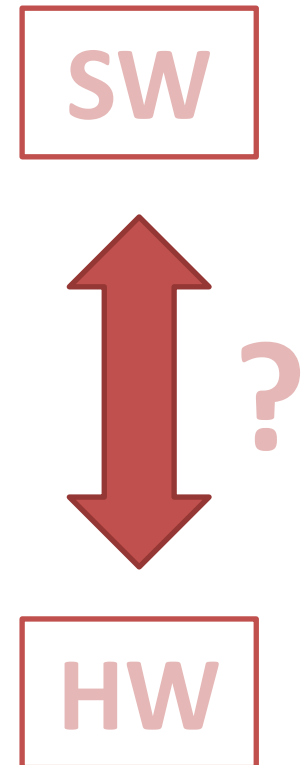


But...what is each one?

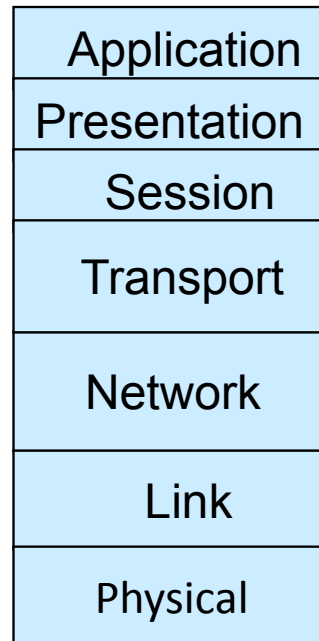
- **OSI is a model for developing protocols**

- Each layer is conceived to have protocols with a specific purpose
- Discussion:
  - What is a protocol?
  - Why is it important?
  - What is a standard protocol?
  - Why is it important?

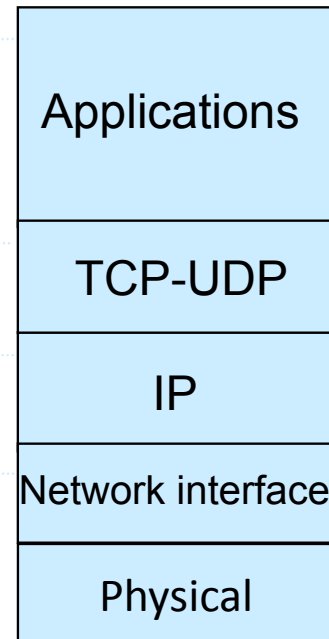
<b>Application</b>	Specific application
<b>Presentation</b>	Information representation and cypher
<b>Session</b>	Authentication, reconnection after failure
<b>Transport</b>	End-to-end connections and reliability
<b>Network</b>	Global addressing and routing
<b>Link</b>	Physical addressing
<b>Physical</b>	Physical access to medium, bit transmission



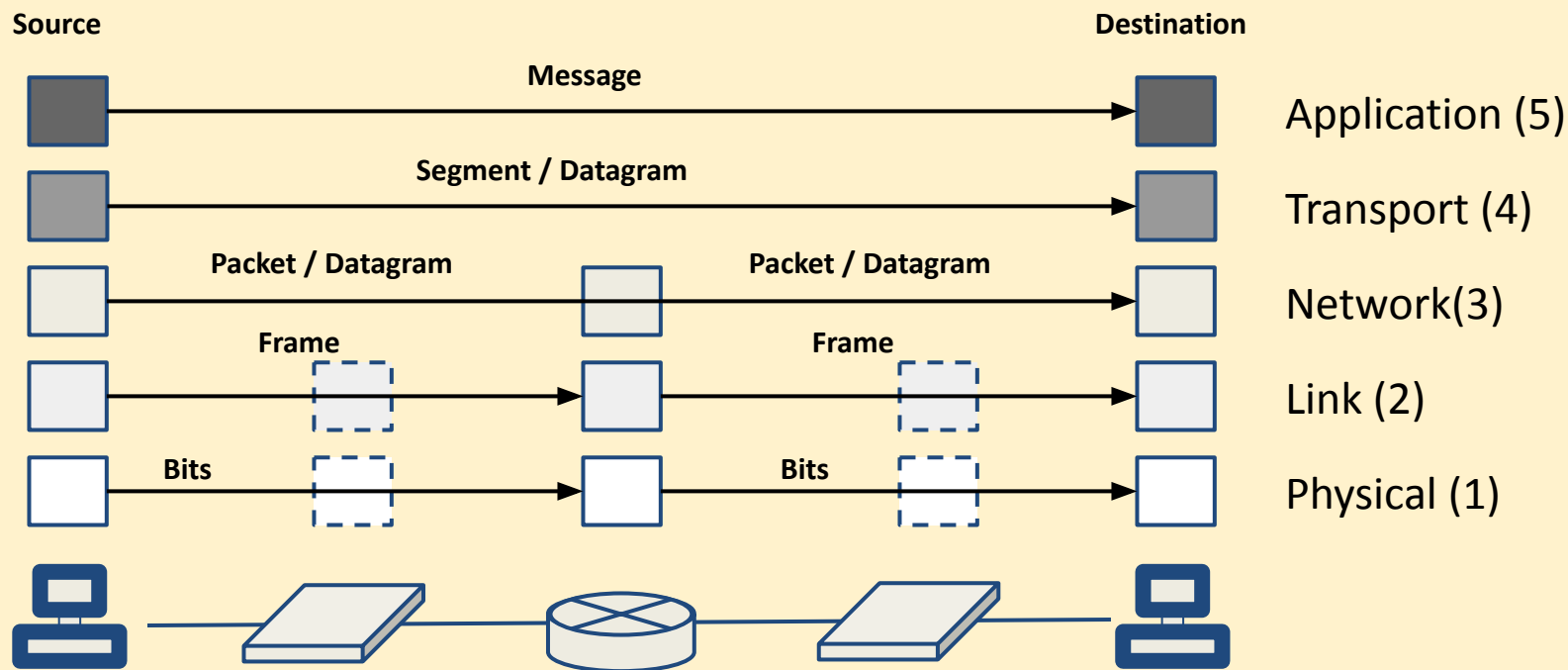
## Architecture OSI

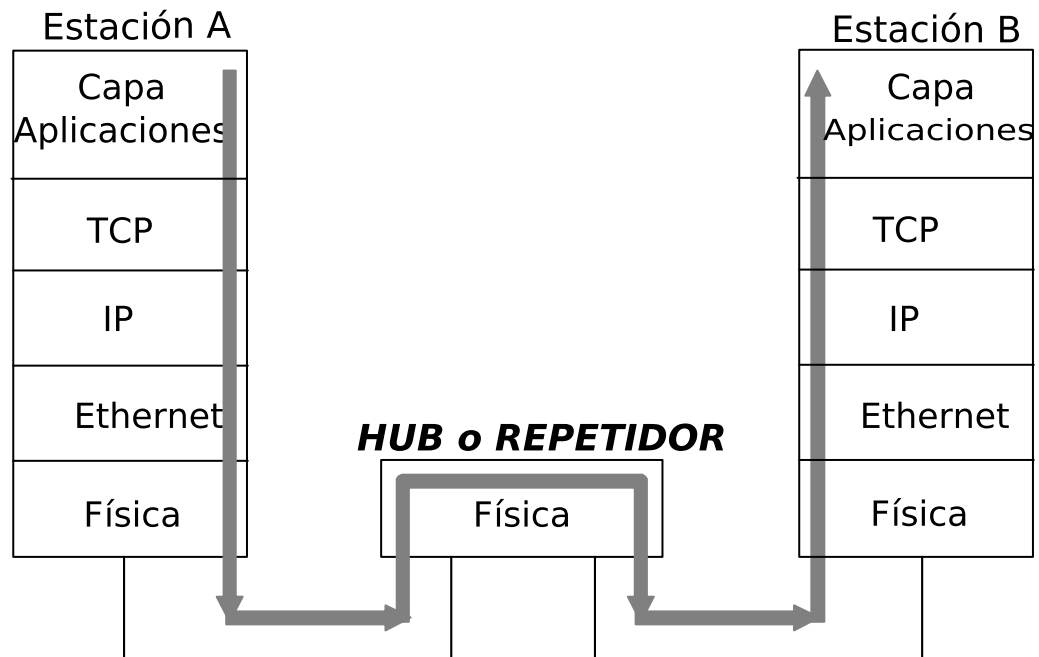


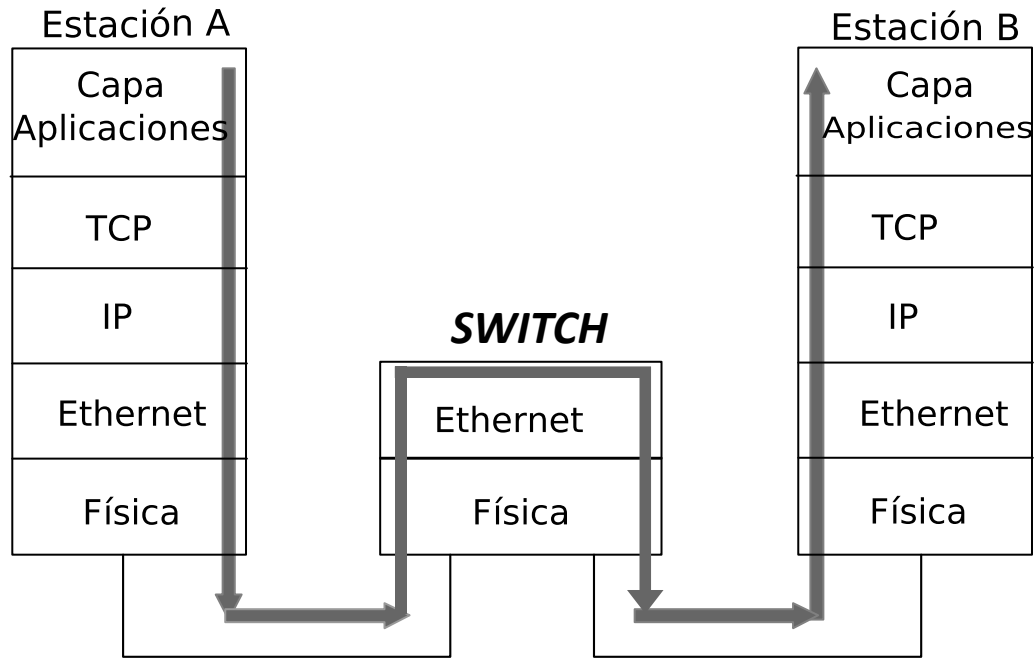
## Model TCP/IP 5 layers (generic)

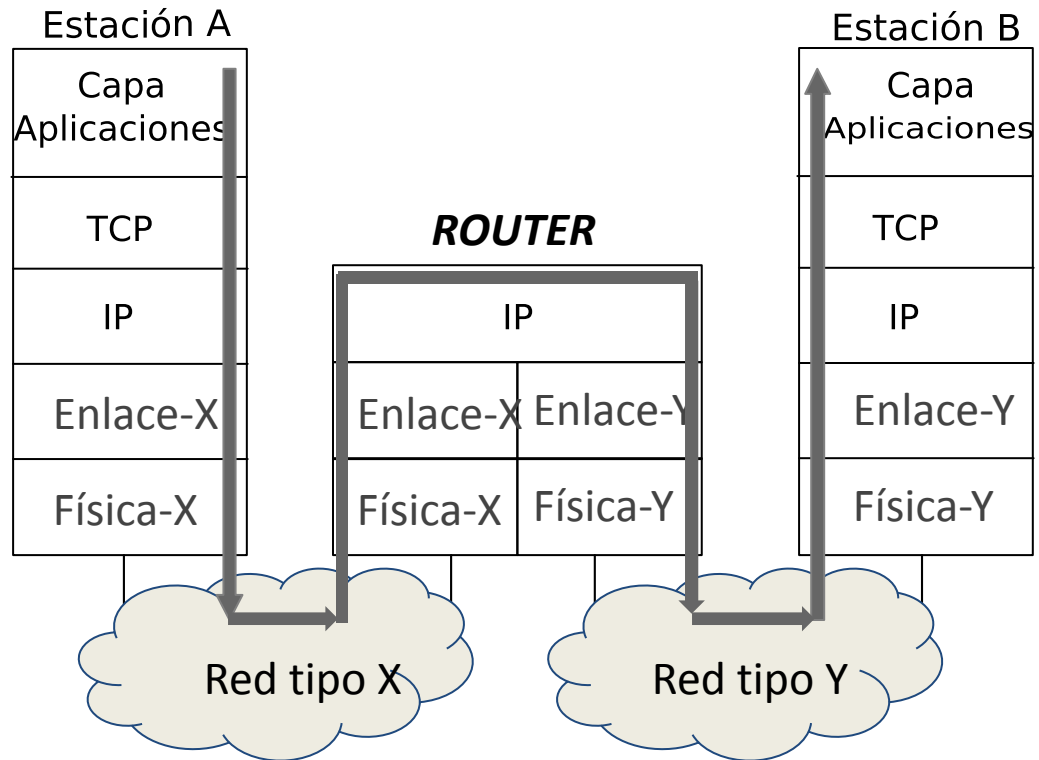


- Layers 5, 4 and 3 are end-to-end (internet)
- Layers 2 and 1 are step-by-step between hosts and routers (link)







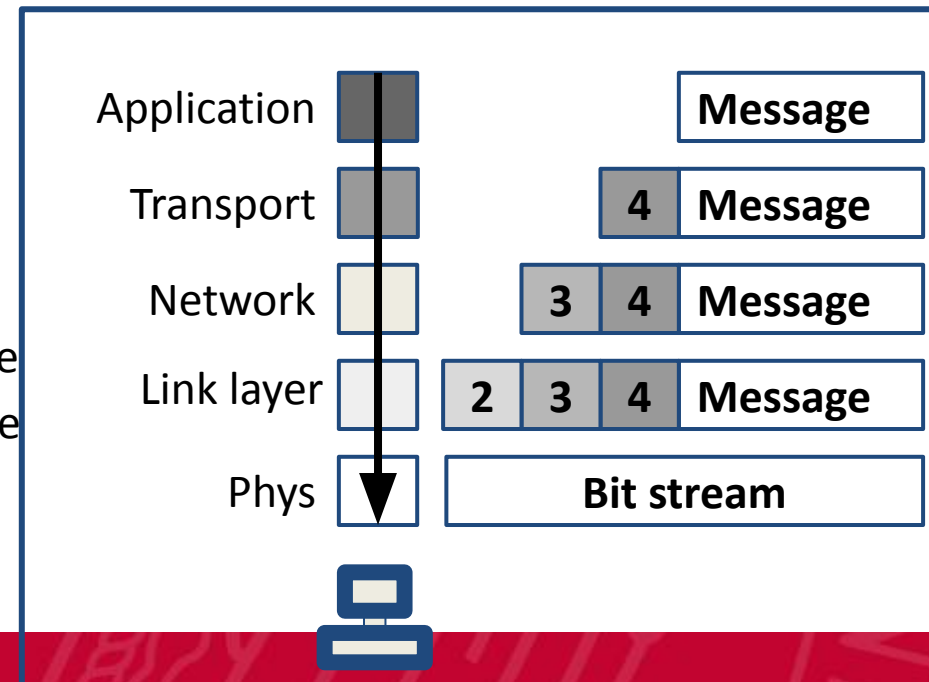


## Encapsulating (sending)

- At each level, the payload is extended with a header information from each protocol
- The transport layer includes information about source and destination processes that are communicating, error control (e.g. checksums) or flow control
- The network layer adds to the former payload information about source and destination hosts, error control and fragmentation
- Link layer includes a header with physical end addresses

## Decapsulating (receiving)

- Upon message rcv., it is decapsulated and sent to upper layers
- Each step carries out an error check
- Routers can re-encapsulate depending on the link in use. Datagrams (layer 3) in general, are not modified



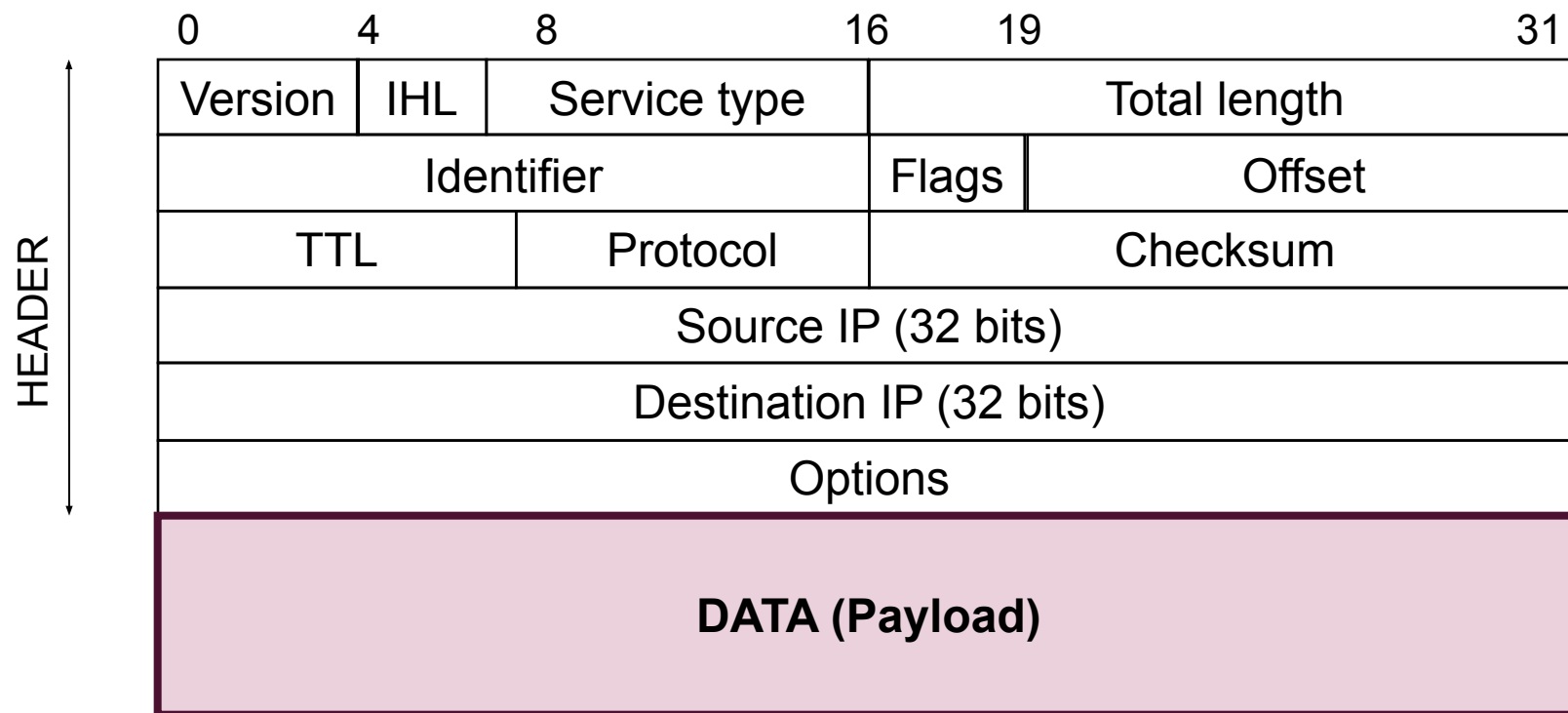


## The Network Protocol in Internet

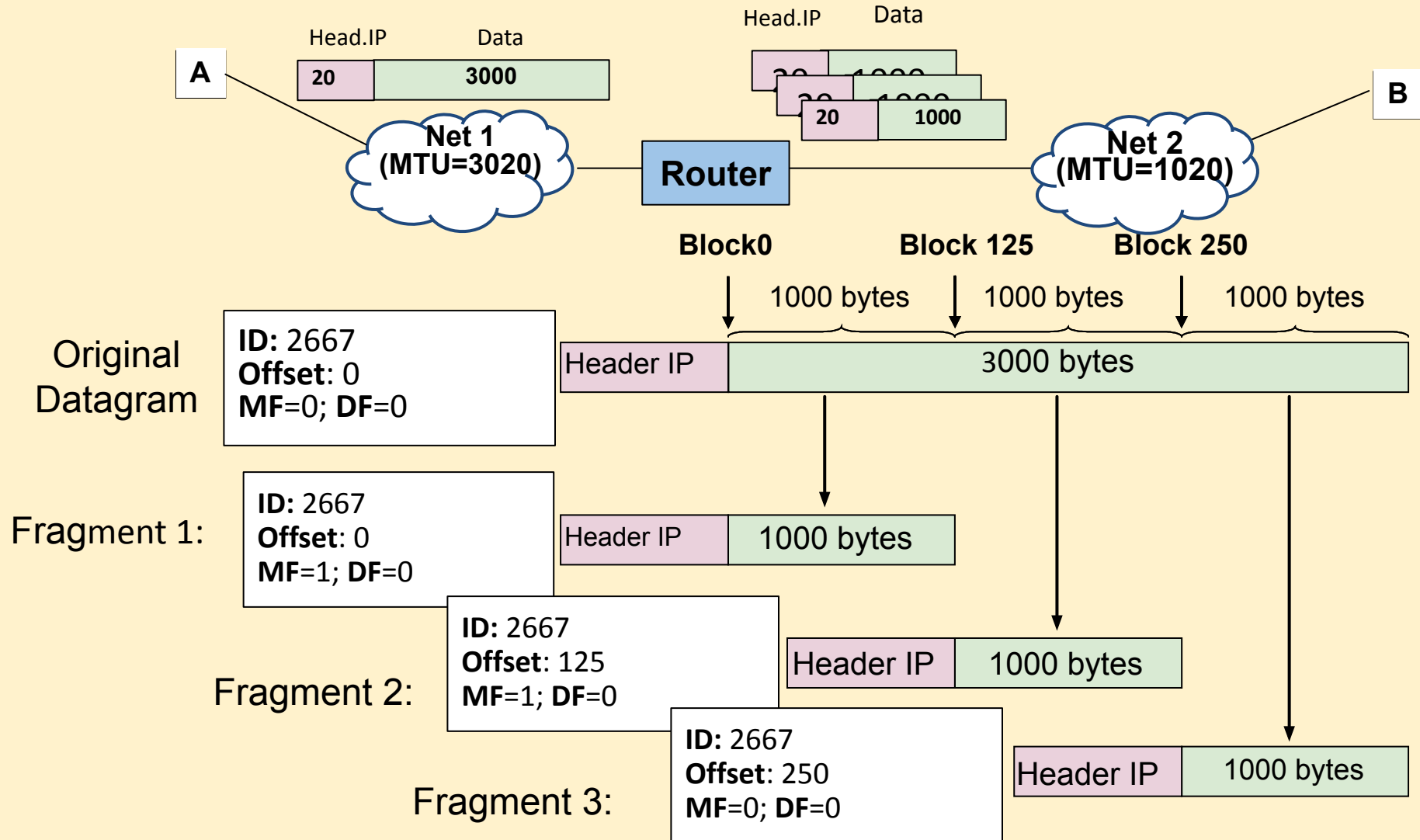
- Offers a basic packet delivery service
  - TCP/IP networks are built on top of it
- Protocol **not connection-oriented** (unreliable)
  - No detection or recovery of lost or erroneous packets
  - No in-order delivery
  - No duplicate package detection

## Basic functions of IP

- **Addressing**
  - Global addressing scheme
- **Fragmentation and reassembly of packets**
  - Divides packages in fragments of acceptable size for the network
  - **Problems for IoT?**
- **Datagram routing**
  - Packet routing considering the information of a routing table
  - Building a route table can be:
    - A manual process (static routing)
    - An automatic/dynamic process using a dynamic routing protocol (RIP, OSPF, BGP, etc.)



## Ejemplo



## IPv4 addresses

- IP addresses are 4 byte-long (32 bits)
- We use “dot notation” to represent them:
  - E.g.: 128.2.7.9 = 10000000 . 00000010 . 00000111 . 00001001

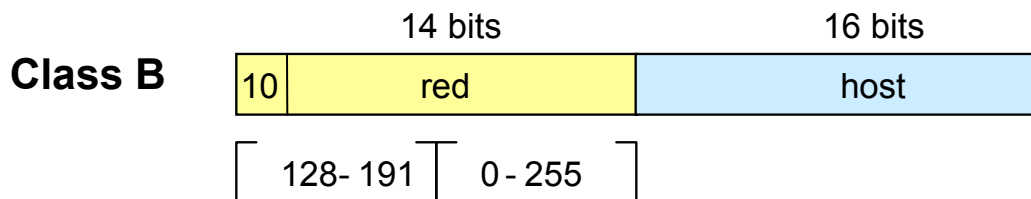
## Address types

- Unicast
  - One host
- Multicast
  - Group of hosts
- Broadcast
  - All hosts within the network

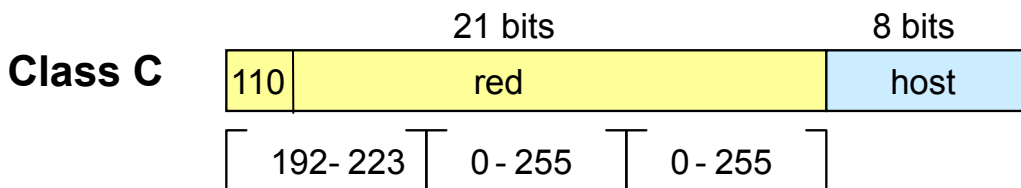
## Classful addresses



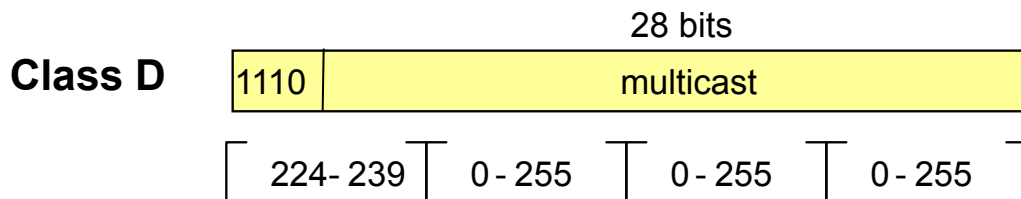
$2^7 = 128$  networks  
 $2^{24} = 16.777.216$  addresses  
 Ex: **26.56.120.9**



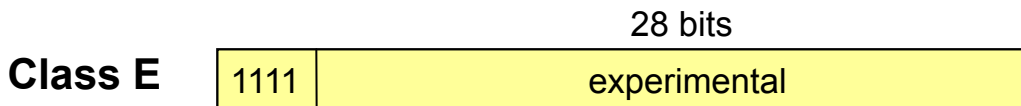
$2^{14} = 16.384$  networks  
 $2^{16} = 65.536$  addresses  
 Ex: **147.96.50.110**



$2^{21} = 2.097.152$  networks  
 $2^8 = 256$  addresses  
 Ex: **217.6.95.44**



Ex.: **224.0.0.1**



- **Addresses reserved for private networks**
  - Not valid for Internet use
    - Can be assigned to isolated networks in Internet
    - Can be connected via a router that performs Network Translation (NAT)
  - Private IP ranges:
    - 10.0.0.0 – 10.255.255.255 ~ 1 private network (class A)
    - 172.16.0.0 – 172.31.255.255 ~ 16 private networks (class B)
    - 192.168.0.0 – 192.168.255.255 ~ 256 private networks (class C)
- **Loopback addresses (127.x.y.z)**
  - Local communication
  - Almost always **127.0.0.1**
- **Broadcast addresses (ending in 11...111)**
  - Used to send a packet to all hosts in a local network
  - Format:
    - All host identifier bits to 1
    - Last range value for network addresses

Network	Host
Net. Id.	1111 ... 111

## Network Addresses

- Red de clase A:
 

Red (8)	Host (24)	
00011011	.00000000.00000000.00000000	= 27.0.0.0
  
- Red de clase B:
 

Red (16)	Host (16)	
10001110.01011000	.00000000.00000000	= 142.88.0.0
  
- Red de clase C:
 

Red (24)	Host (8)	
11000111.01000011.11101111	.00000000	= 199.67.239.0
  
- Red sin clase (n=14):
 

Red (18)	Host (14)	
01011010.00100000.10	00000000.00000000	= 90.32.128.0
  
- Red sin clase (n=5):
 

Red (27)	Host (5)	
10001111.00011010.00000111.011	00000	= 143.26.7.96

## A network mask is used to indicate:

- Which part of the IP identifies a network
  - Mask bits fixed to 1
- Which part of the IP address identify the host within the network
  - Mask bits fixed to 0

## Example

- Class C address: 221.98.22.2
- Mask: 255.255.255.0

	Red	Host
IP:	11011101 . 01100010 . 00010110	. 00000010 = 221.98.22.2
Máscara:	11111111 . 11111111 . 11111111	. 00000000 = 255.255.255.0

- **Alternative notation: 221.98.22.2/24**
  - The value **/24** is used to indicate the length of the network part (num. of 1s)
  - Referred as **CIDR** (*Classless Interdomain Routing*) notation



## Ejemplo: Máscaras de red

				Notación CIDR (Dir. IP extendida)	
				↓	
Dir. de clase A =	Red (8)	Host (24)			
	00011011.	00000111.10000010.00000011	= 27.7.130.3	}	27.7.130.3/8
Máscara =	11111111.	00000000.00000000.00000000	= 255.0.0.0		
Dir. de clase B =	Red (16)	Host (16)			
	10001110.01011000.	00001100.00000100	= 142.88.12.4	}	142.88.12.4/16
Máscara =	11111111.11111111.	00000000.00000000	= 255.255.0.0		
Dir. de clase C =	Red (24)	Host (8)			
	11000111.01000011.11101111.	00000110	= 199.67.239.6	}	199.67.239.6/24
Máscara =	11111111.11111111.11111111.	00000000	= 255.255.255.0		
Dir. sin clase =	Red (18)	Host (14)			
	01011010.00100000.10	000011.00000101	= 90.32.131.5	}	90.32.131.5/18
Máscara =	11111111.11111111.11	000000.00000000	= 255.255.192.0		
Dir. sin clase =	Red (27)	Host (5)			
	10001111.00011010.00000111.011	00011	= 143.26.7.99	}	143.26.7.99/27
Máscara =	11111111.11111111.11111111.111	00000	= 255.255.255.224		

- **Routing** consists on finding a path, from source to destination, through intermediate nodes or routers
- If more than one path exists, it is necessary to decide which is the best: ***shortest***
- What is “the shortest path”?:
  - **Number of hops:** takes into account the number of routers and/or intermediate networks to traverse till the destination
  - **Geographical distance:** distance (kms) to destination
  - **Average delay:** line delay. Similar to geographical distance
  - **Bandwidth:** transmission speed of intermediate networks
  - **Traffic level:** considers the amount of usage of the lines, avoiding those with more congestion
  - **Combination of metrics**
  - **Other?**

## Local routing

- Network topology is not considered
- Common techniques:
  - Random routing.
  - Isolated routing.
  - Flooding.

## Static routing

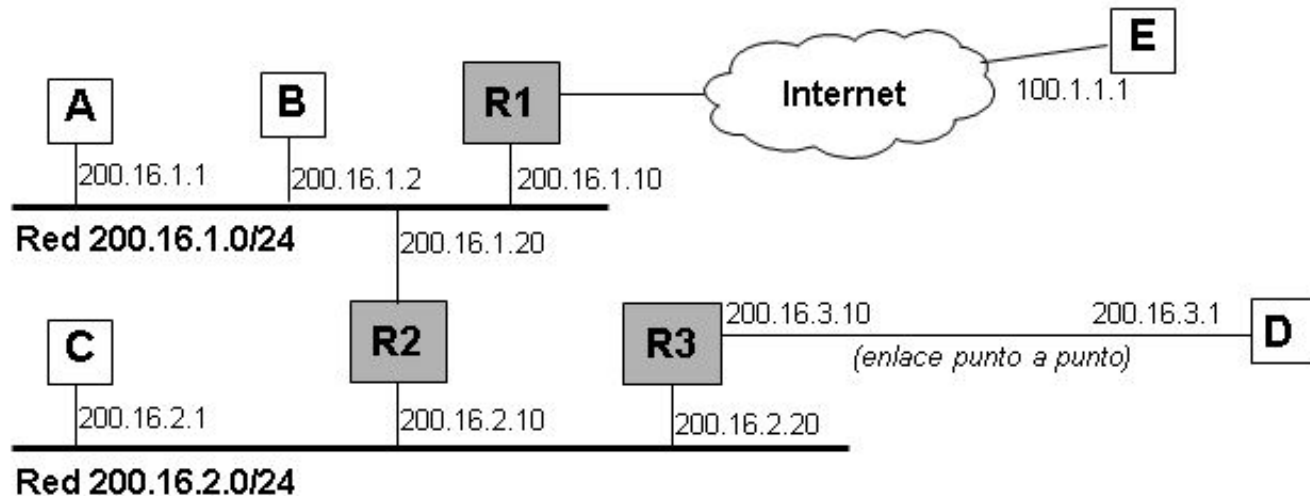
- Routing decisions are based on network topology:
  - The administrator builds (manually) the routing tables
  - Not adapted automatically to changes in the structure or topology of the network

## Dynamic routing

- Routing tables are built automatically, through interchange (periodic) of information between routers.
  - No admin intervention
  - Adapted automatically to network topology
- Common techniques
  - Distance-vector routing (RIP)
  - Link-state routing (OSPF)

## Routing tables in IP networks

- Network example



- For A, the routing table could be:

```
# route -n
Kernel IP routing table
Destination      Gateway          Genmask         Flags         Iface
1---> 200.16.1.0      0.0.0.0         255.255.255.0   U             eth0
2---> 200.16.2.0      200.16.1.20    255.255.255.0   UG            eth0
3---> 200.16.3.1      200.16.1.20    255.255.255.255 UGH           eth0
4---> 0.0.0.0         200.16.1.10    0.0.0.0         UG            eth0
```

- IPv4 uses 32 bit addresses
  - Up to 4.294.967.296 different addresses
  - Classful addressing avoids using all of them (~250 million)
- Problems with classful addressing
  - Class B addresses are necessary...no left
  - Consecutive class C (super classes) granted
  - Routing tables grow exponentially
- CIDR (Classless Interdomain Routing) alleviates the situation
  - Routing tables use destination address plus mask
  - This allows to reduce the length of the table (address summary)
  - NAT (Network Address Translation)
    - A router hides a local network, that appears as just one address for the rest of the Internet

- Address length was fixed for IPv6 to 128 bits
  - $340 \times 10^{36}$  addresses
- Allows for coming back to the original plan: one address per host
  - **NAT avoided the use of many applications**

- **TCP (Transmission Control Protocol)**

- “Connection oriented” protocol
- Guarantees a reliable end-to-end connection
  - Detects segments from lost or erroneous data and retransmits them
  - Detects duplicated segments and discards them
  - Orders segments at destination and delivers them in order to the application
- Used in applications in which reliability is more important than speed
  - Examples: FTP, HTTP, Telnet, SMTP, etc.

- **UDP (User Datagram Protocol)**

- “Non-connection oriented” protocol
- **Does not guarantee** a reliable end-to-end service
  - Does not control package loss, errors or duplicity
- Used in application sin which speed is more important than reliability
  - DNS, SNMP, RIP, RTP, etc.

Feature	TCP	UDP
Connection type	Connection oriented: <ul style="list-style-type: none"><li>• In order delivery</li><li>• Retransmission of lost or erroneous segments</li><li>• Duplicate detection</li></ul>	No connection (best effort): <ul style="list-style-type: none"><li>• Out-of-order delivery</li><li>• No packet retransmission if lost or erroneous</li><li>• No detecta duplicados</li></ul>
Transfer unit	Segment (20 bytes minimum header)	Datagram (8 bytes minimum header)
Communication phases	<ol style="list-style-type: none"><li>1. Connection establishment</li><li>2. Data transfer</li><li>3. Connection close</li></ol>	<ol style="list-style-type: none"><li>1. Data transfer (data block)</li></ol>
Error/flow control	Sliding window <ul style="list-style-type: none"><li>• Segment numbering</li><li>• Confirmation</li><li>• Retransmission</li></ul>	No flow or error detection/correction
Examples	Telnet, FTP, HTTP, SMTP, POP3...	DNS, RIP, SNMP, DHCP...



- **Ports**

- Each application/network process (client or server) in a computer is identified by a **port number**, unique within the system:
  - 16-bit identifier
  - Used in TCP and UDP applications

- **Port in a client process**

- When the user starts a client process, the OS assigns automatically a free port number
- This number is always  $> 1023$  (ephemeral ports)  
*(Ports  $\leq 1023$  are used for servers with superuser privileges)*

- **Port in a server process**

- When a client process asks for a communication to a server, the client must know a priori the port number in the server
- Usually, each server type uses a fixed port number, known as well-known port

- **Ports in a server process**
  - Well-known portas

**Server****Port**

FTP

20 and 21

SSH

22

TELNET

23

SMTP

25

**Server****Port**

DNS

53

HTTP

80

POP3

110

NTP

123

- In Linux, **/etc/services** contains the well-known port associated to each server type
- Usually  $\leq 1023$

- **Sockets**

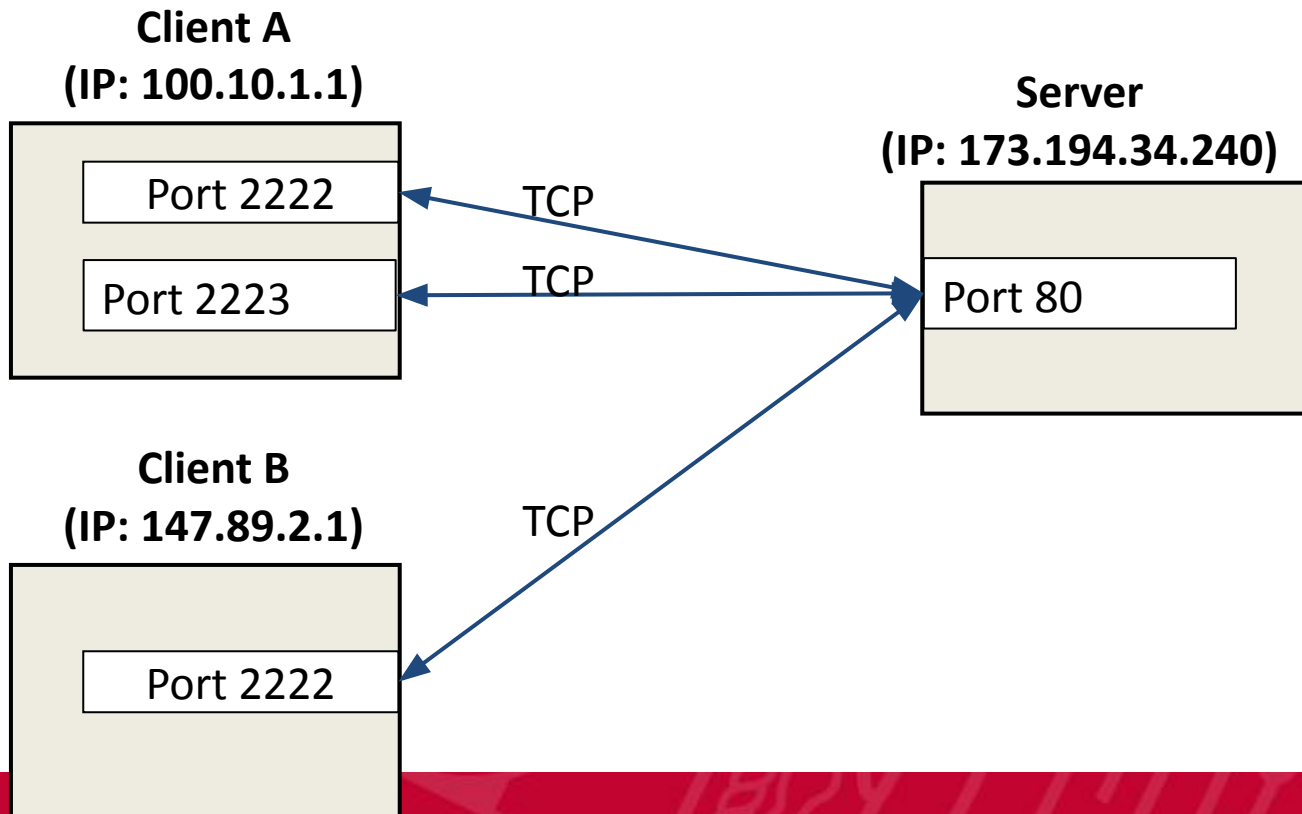
- When a communication channel is established between two ends (TCP or UDP), the ends of the channel are known as “**sockets**”
- A socket is identified by three parameters:
  - IP address, port number and protocol (TCP or UDP)
- Once created, sockets allow for a bidirectional data interchange between client and server



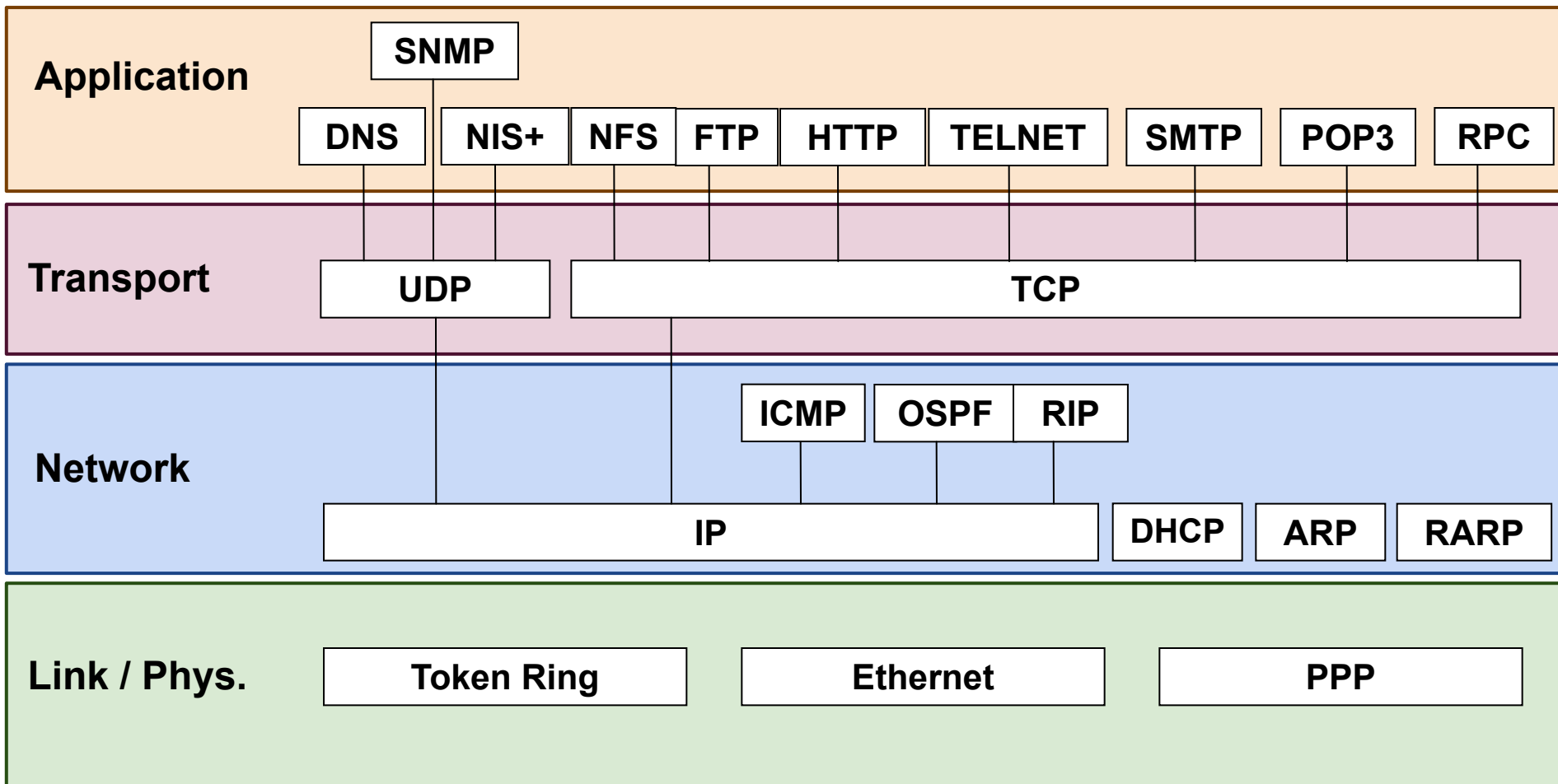
- **Parameters in a client/server connection:**

- Protocol (TCP or UDP, same for both ends)
- Client IP address
- Client port
- Server IP address
- Server port

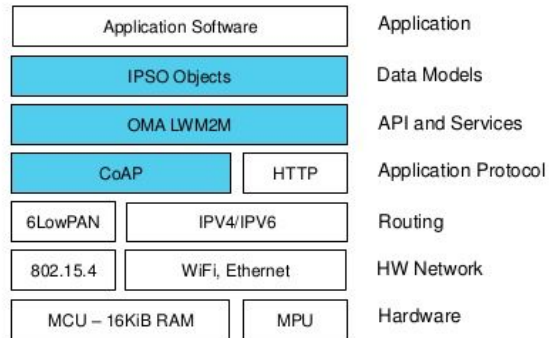
- **Concurrent servers**
  - Concurrent attention to multiple users is necessary



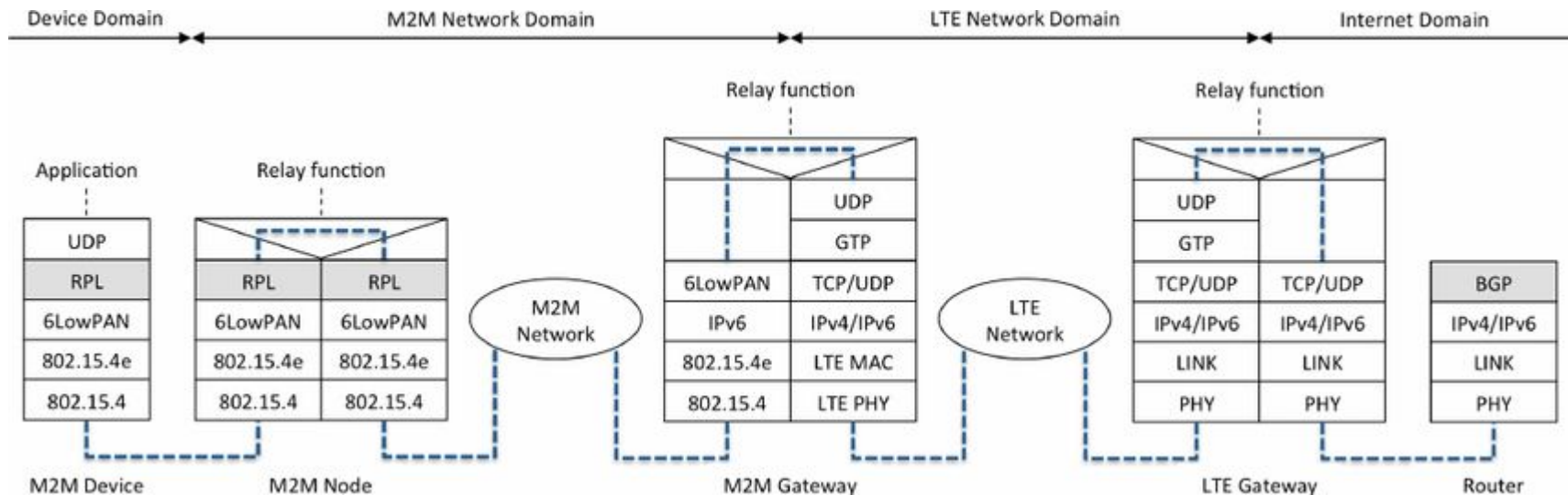
- So...IP encapsulates TCP, or the other way round?
- What about HTTP? Which layer?
- TCP/IP... *is it hardware or software?*

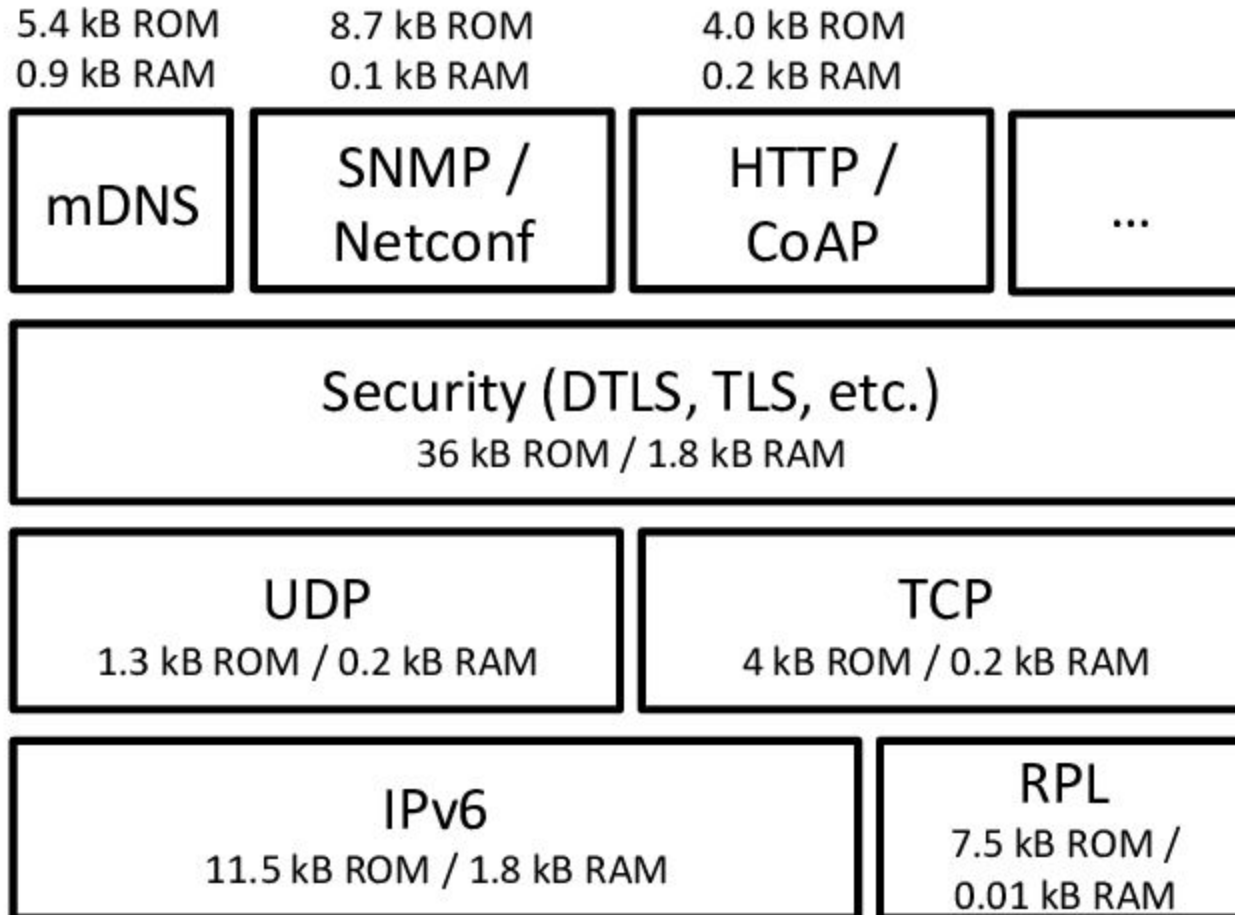


## IoT Standards "Stack"

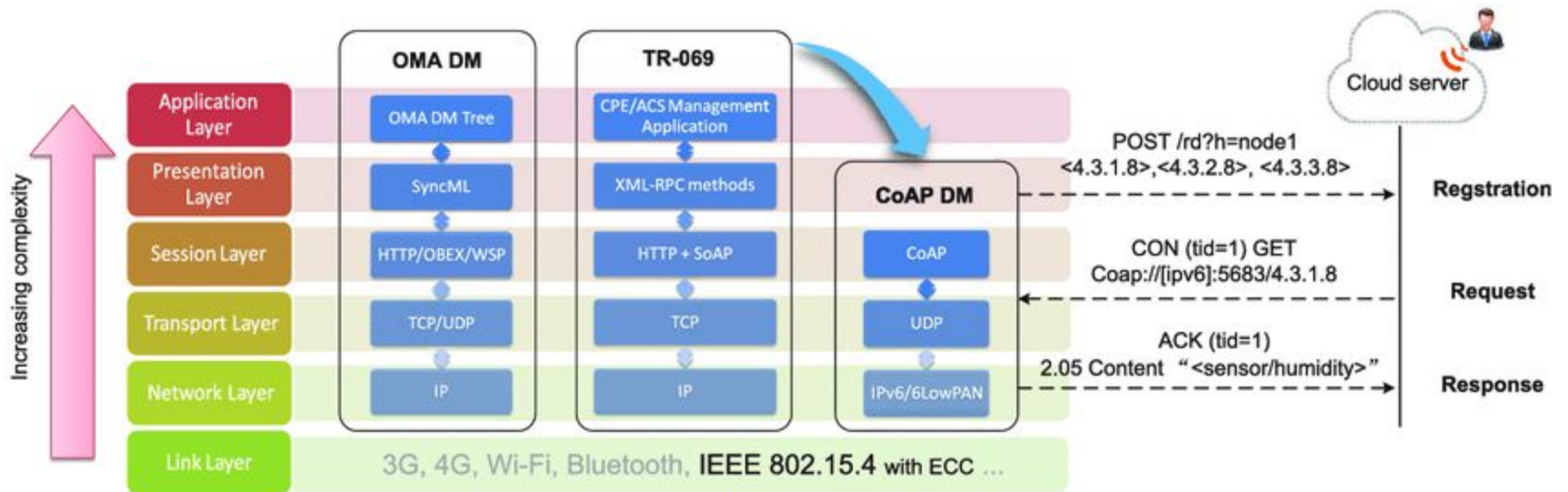


ARM

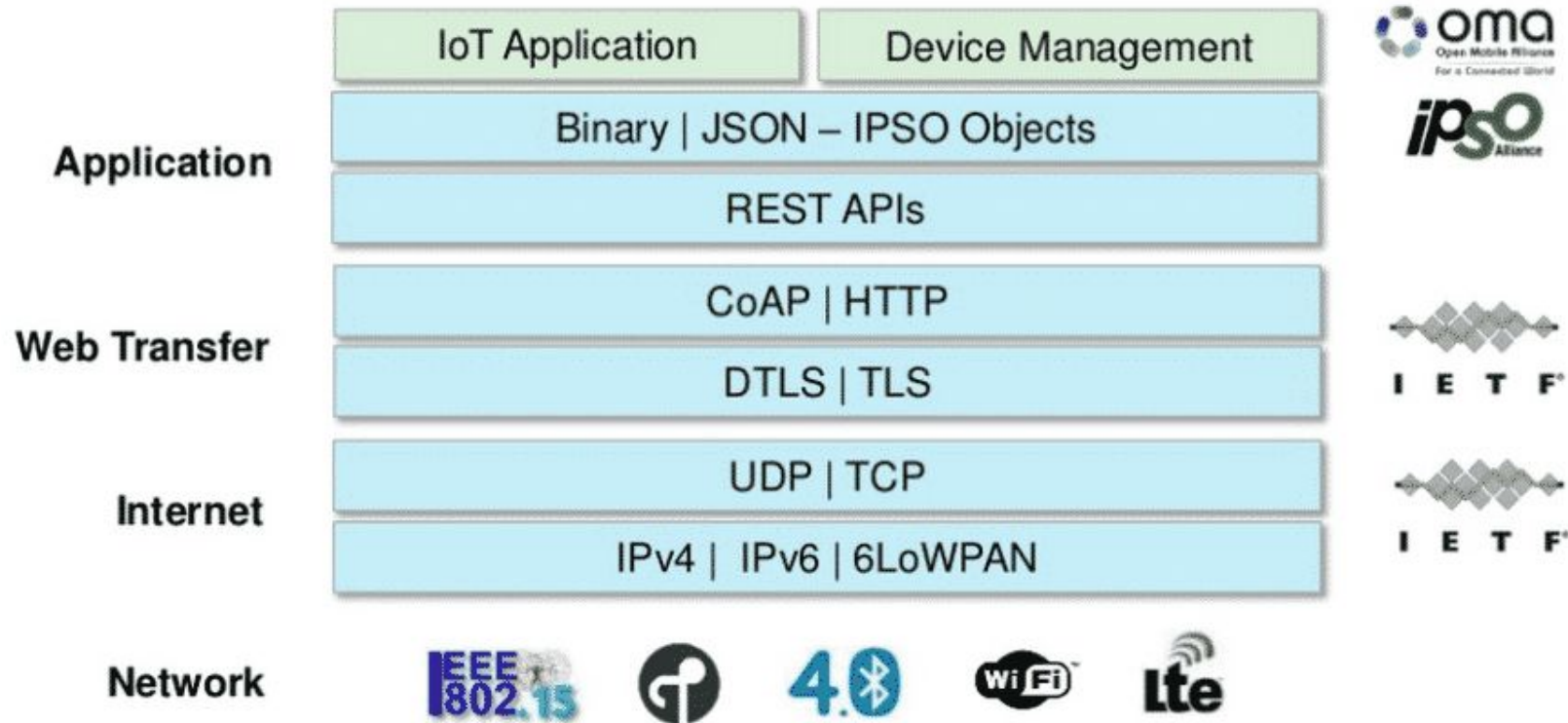




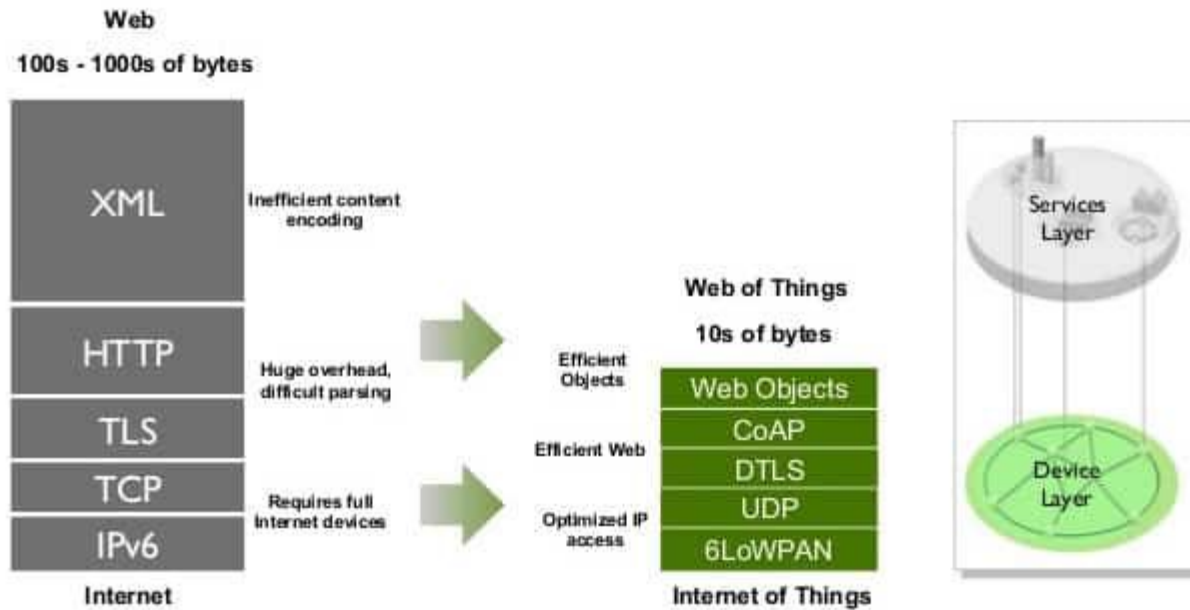


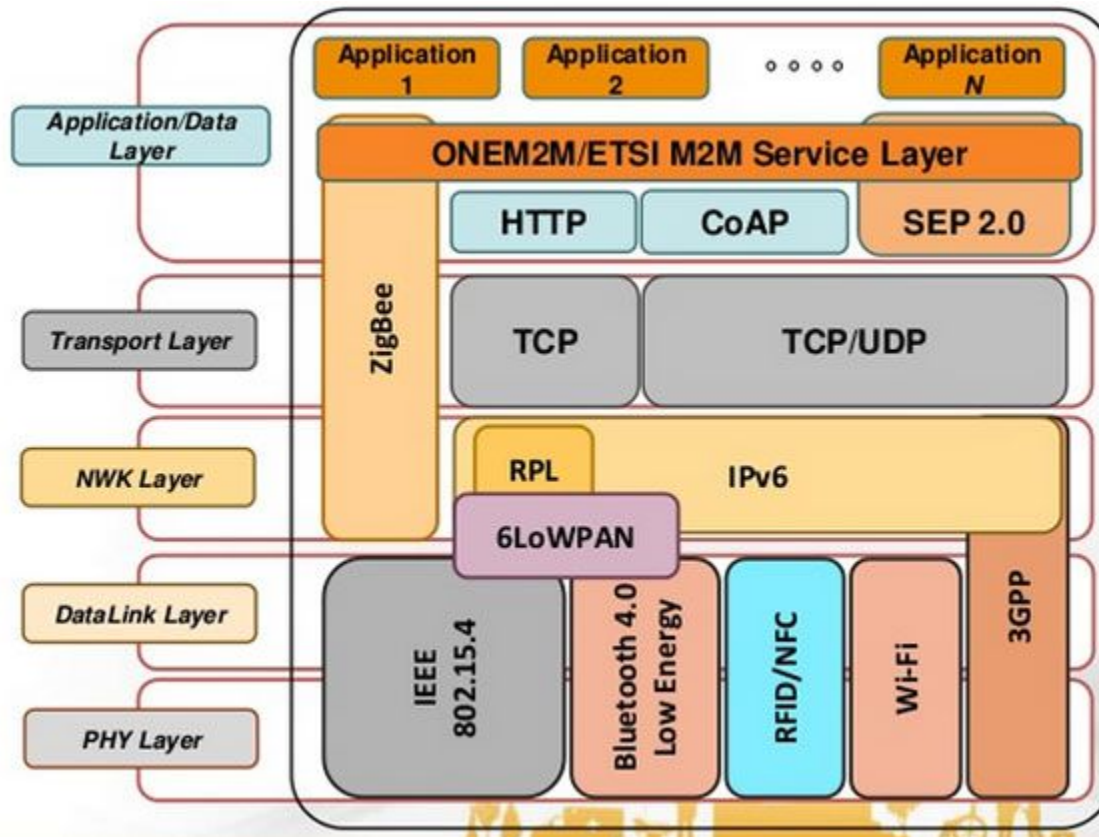


# Remember the I in IoT!



# Is the Internet Protocol enough?





<https://www.postscapes.com/internet-of-things-protocols/>

*Interconnecting Smart Objects with IP. The next Internet*

JP Vasseur, Adam Dunkels