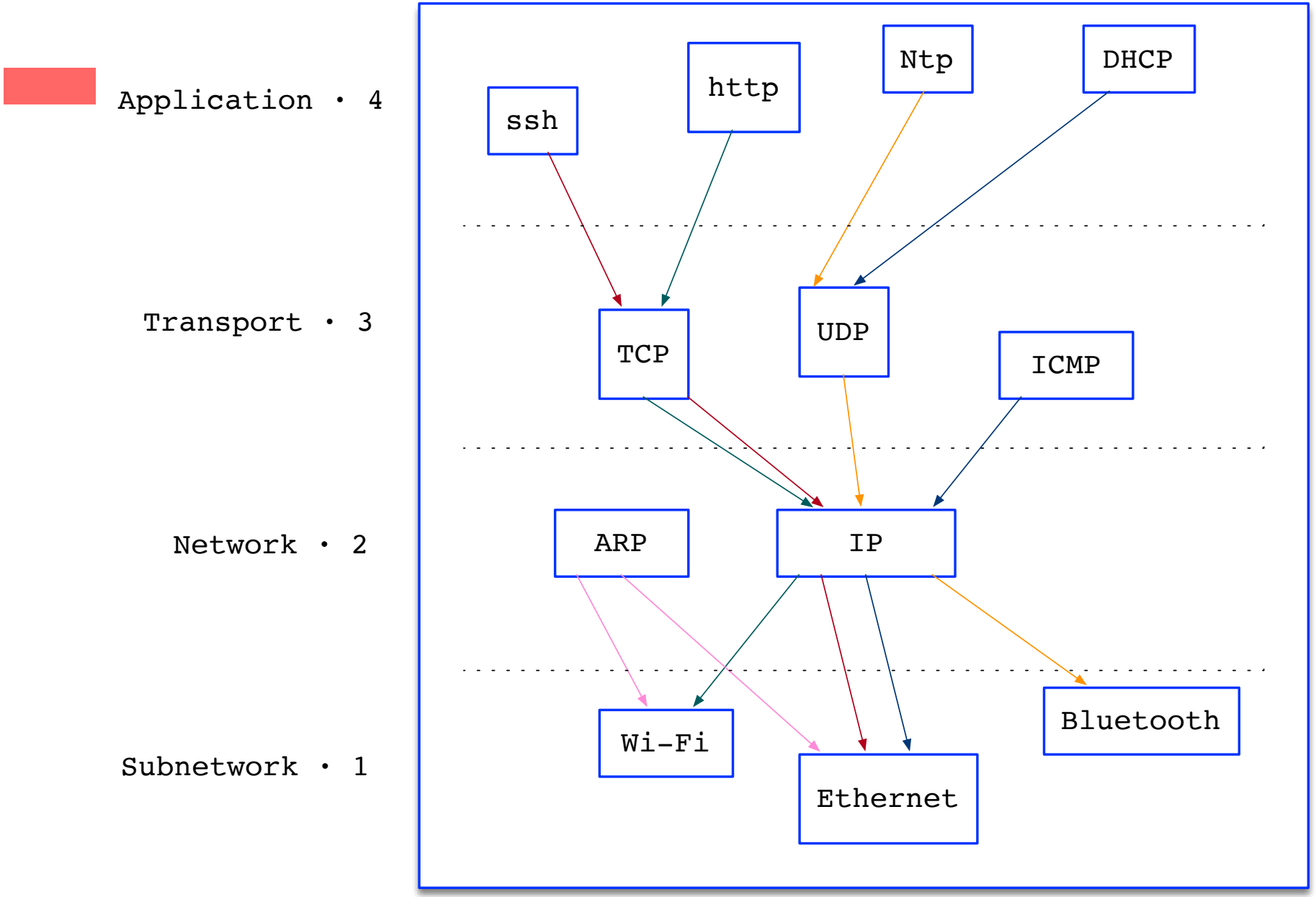
http://paloalto.unileon.es/cn-ch1-s2.pdf

Chapter 1: Conceptual Basis

Section 2

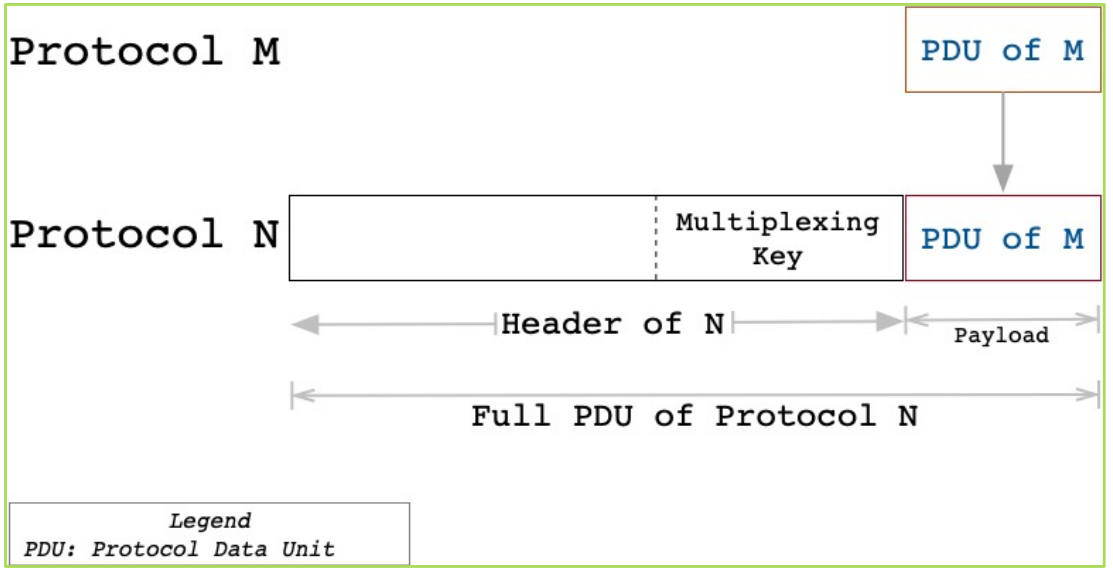
v 2.2 9/3/22

Typical Internet Protocol Stacks



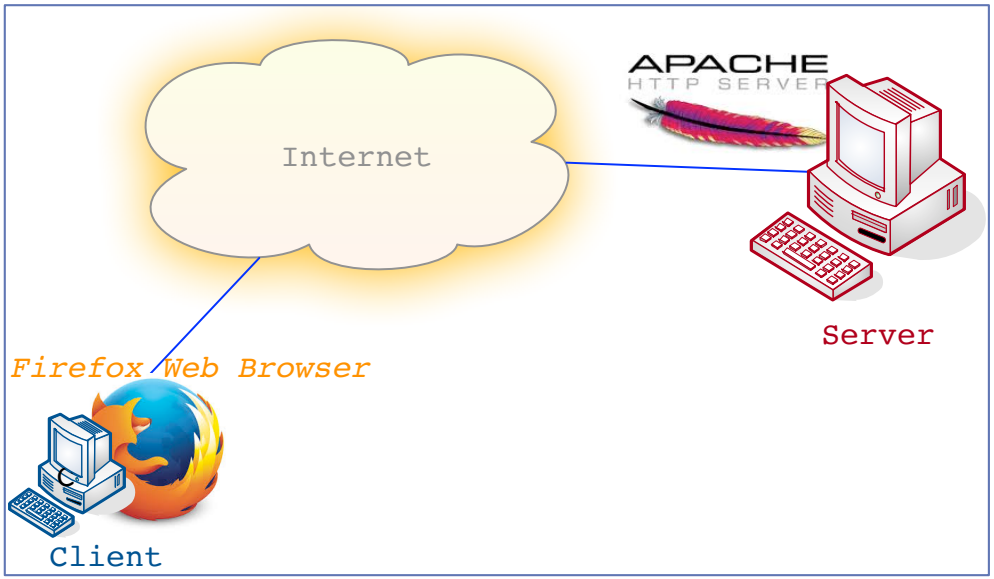
Recalling concept: Encapsulation

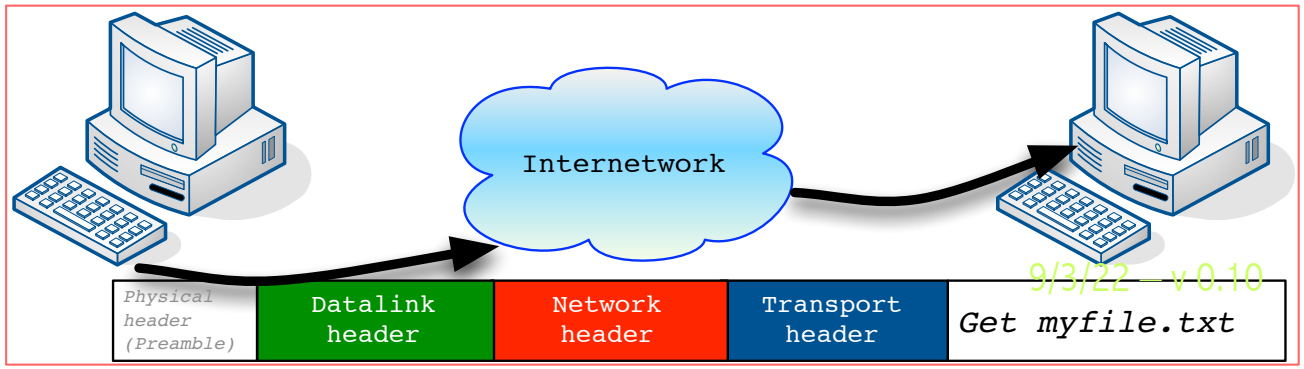
- Protocol M wants to use the service provided by protocol N
  - 1. M builds a PDU of its own and hands it on to N
    - The data handed by M is the **payload** received by N
  - 2. N builds the header of its own PDU
    - Header of N includes a **Multiplexing Key**
      - Mux Key = A standard identification of protocol M
  - 3. N appends its own header to the payload
    - A PDU of N is completed
  - 4. Protocol N typically will continue by using the service from another protocol P
- PDU of M has been encapsulated into a PDU of N



Encapsulation and multiplexing in the Internet Architecture

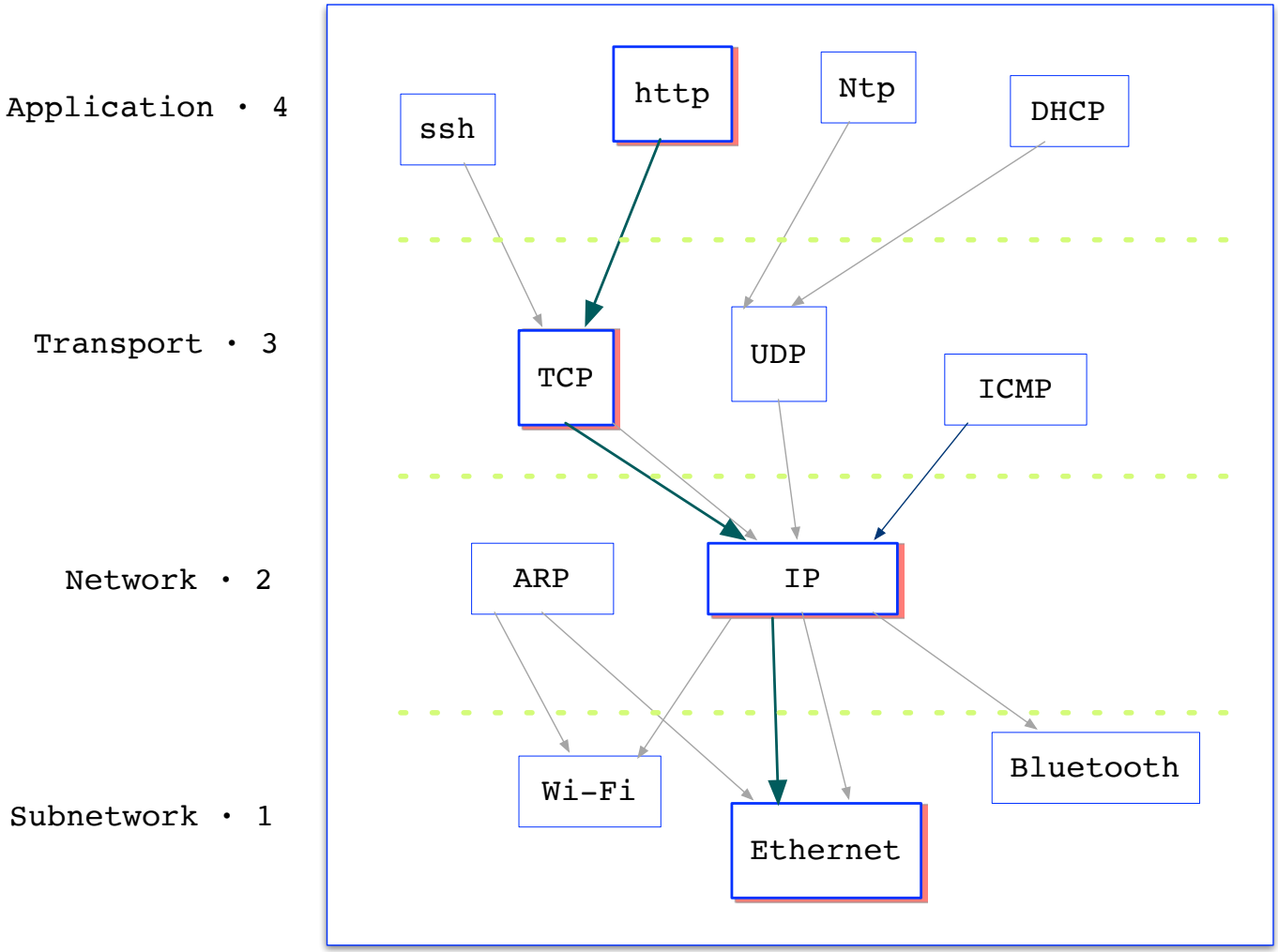
- **Example**: A web browser requesting a web page

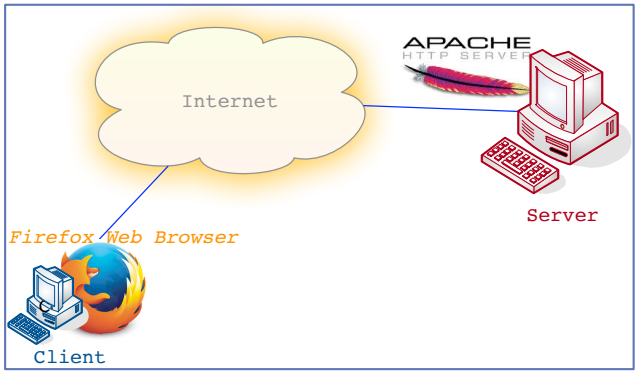




Encapsulation and multiplexing in the Internet Architecture

- **Example**: A web browser requesting a web page



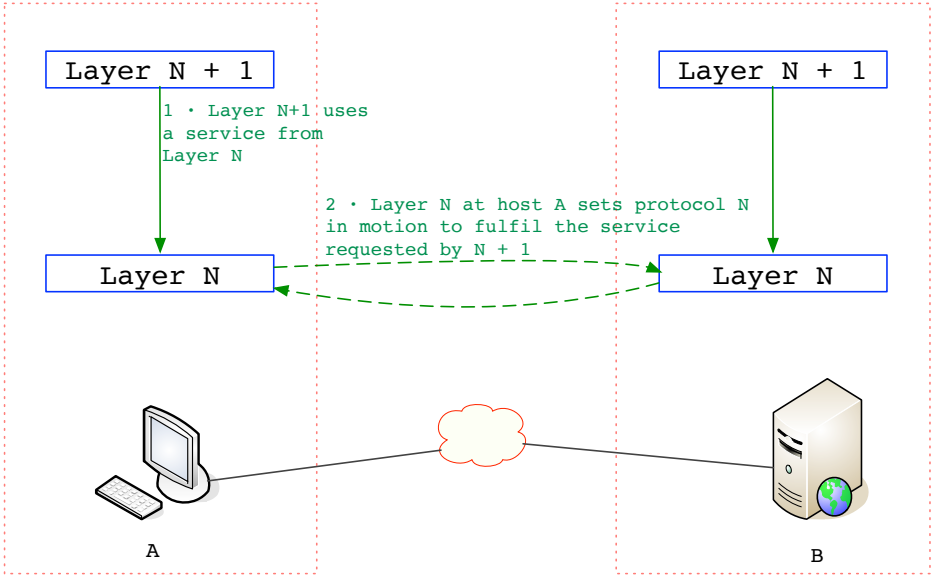


Encap & Mux at interface

What gets sent at each SIF?

- What does the Application protocol send to TCP?
- What does TCP send to IP?
- Finally, what does IP send to Ethernet?

Concept: Protocol N+1 sending onto Protocol N



Encap & Mux at interface

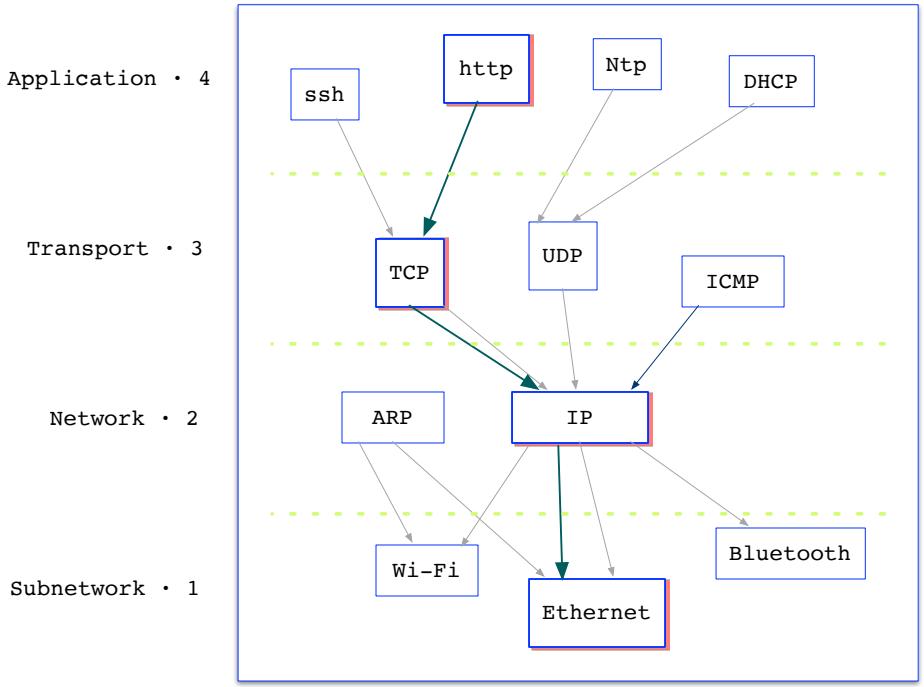
What gets sent at each SIF?

- What does the Application protocol send to TCP?
- What does TCP send to IP?
- Finally, what does IP send to Ethernet?

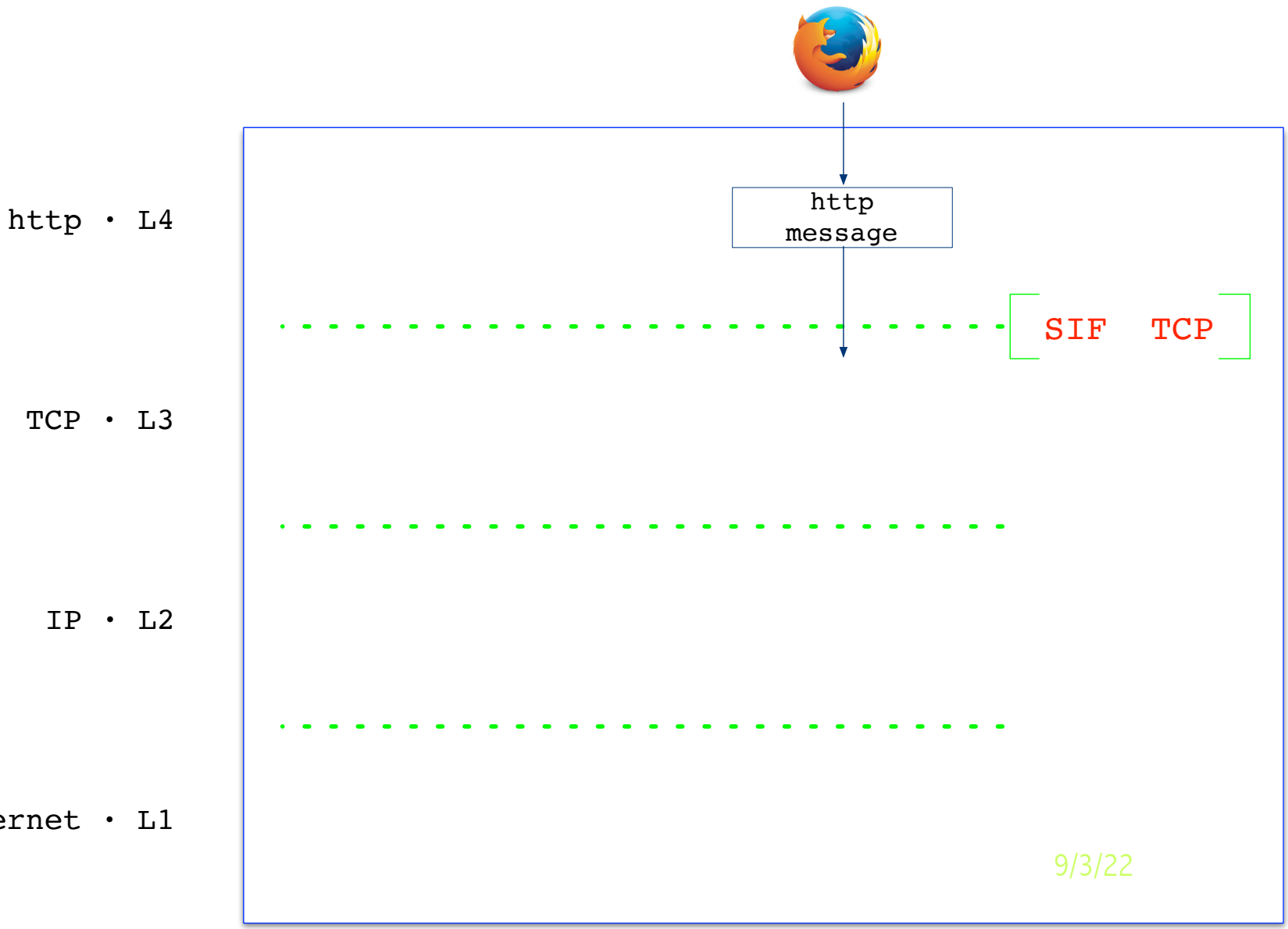
Http-> TCP?

TCP -> IP?

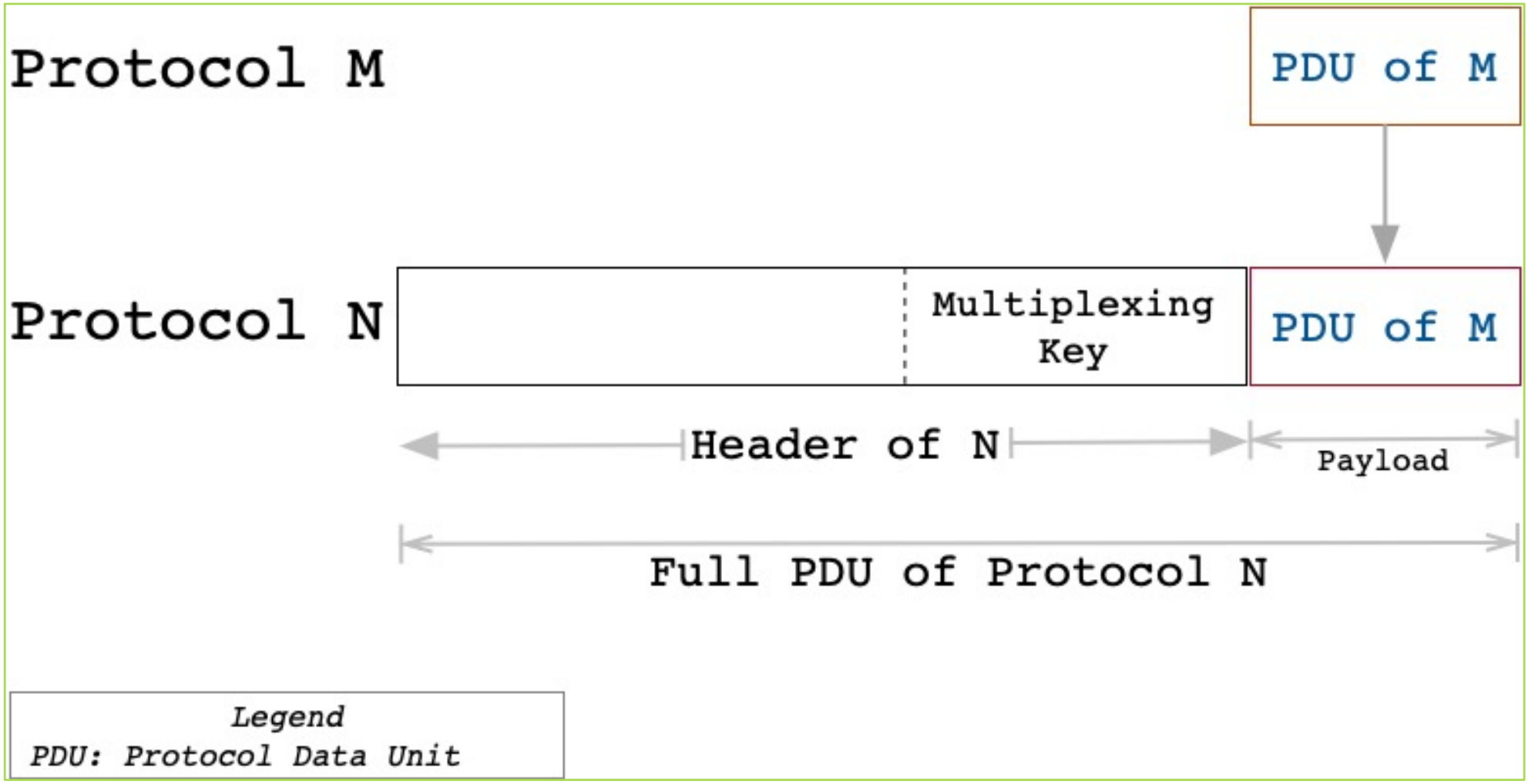
IP-> Ethernet?



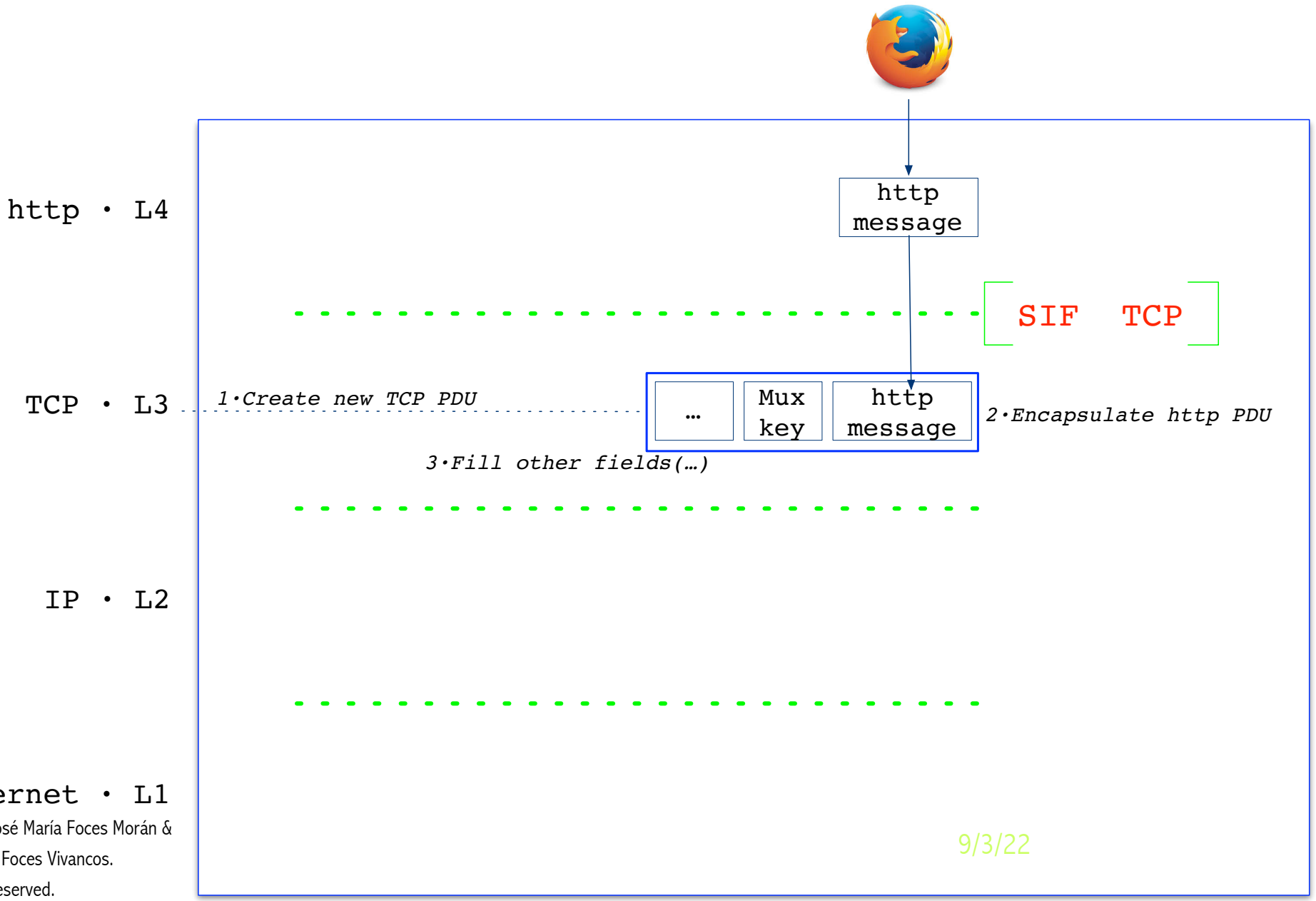
Browser calls TCP SIF, sends http message



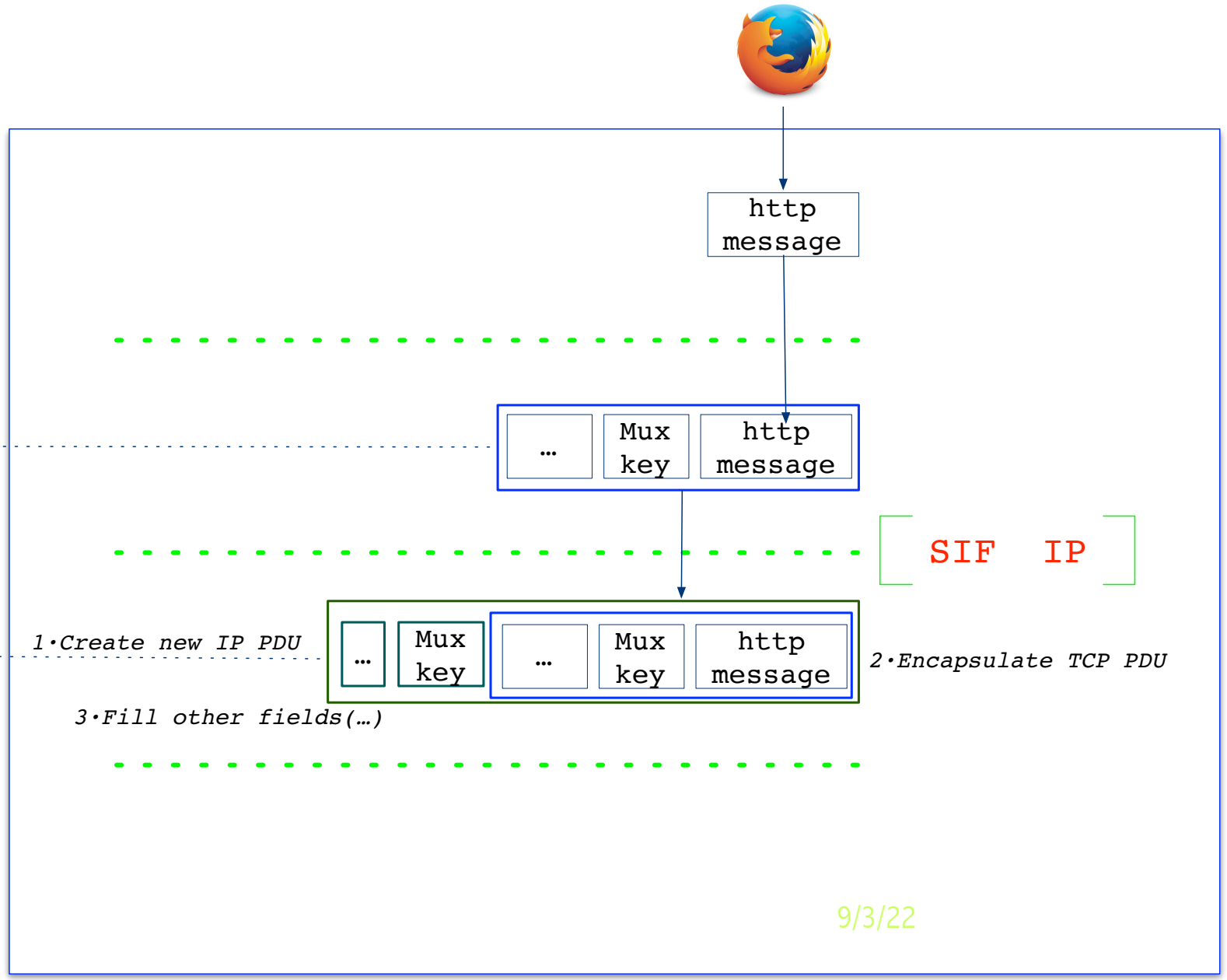
Recalling concept: Encapsulation



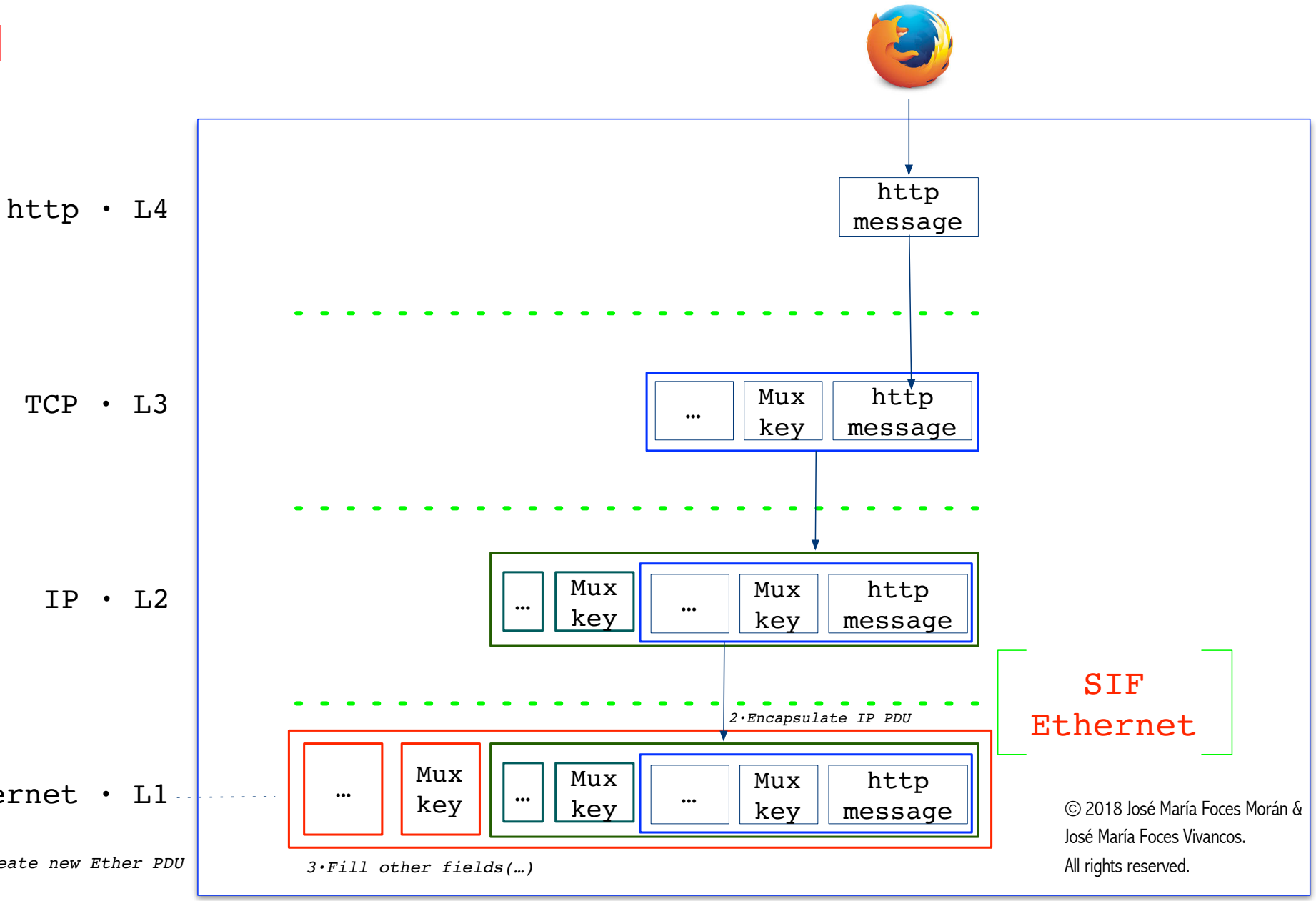
TCP creates a TCP PDU



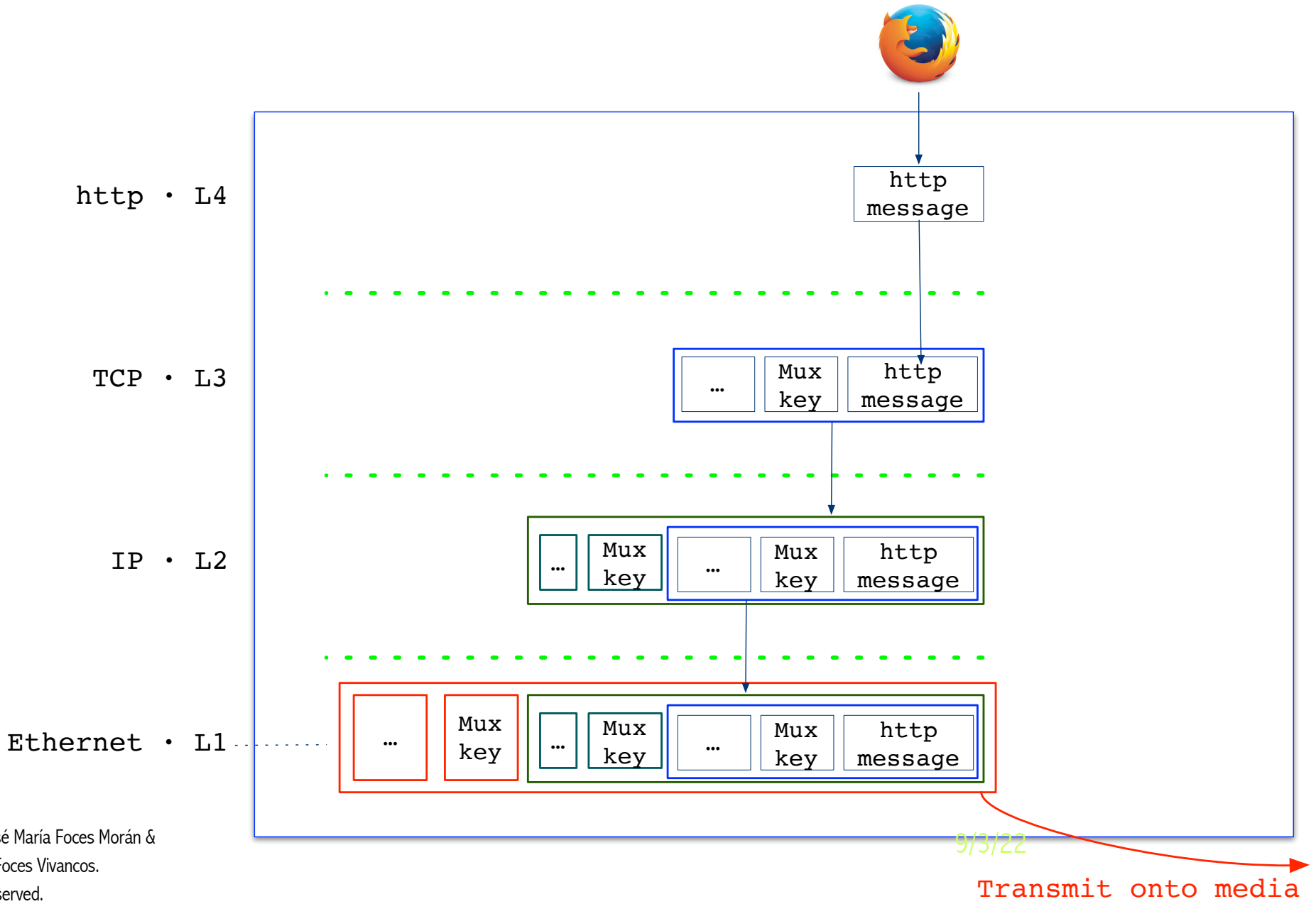
TCP sends segment onto IP SIF



IP sends packet onto Ethernet SIF



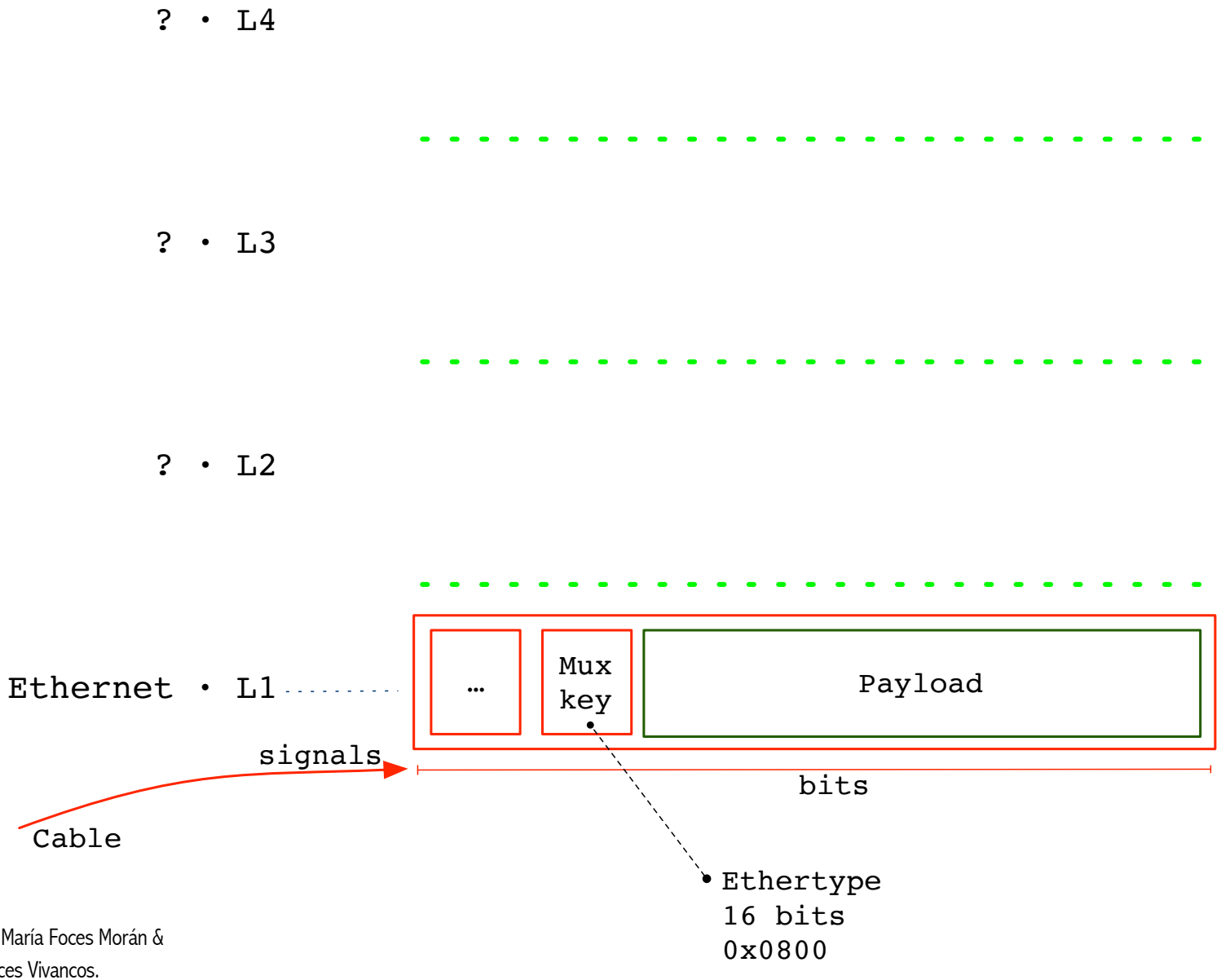
Ethernet transmits whole frame onto the wire



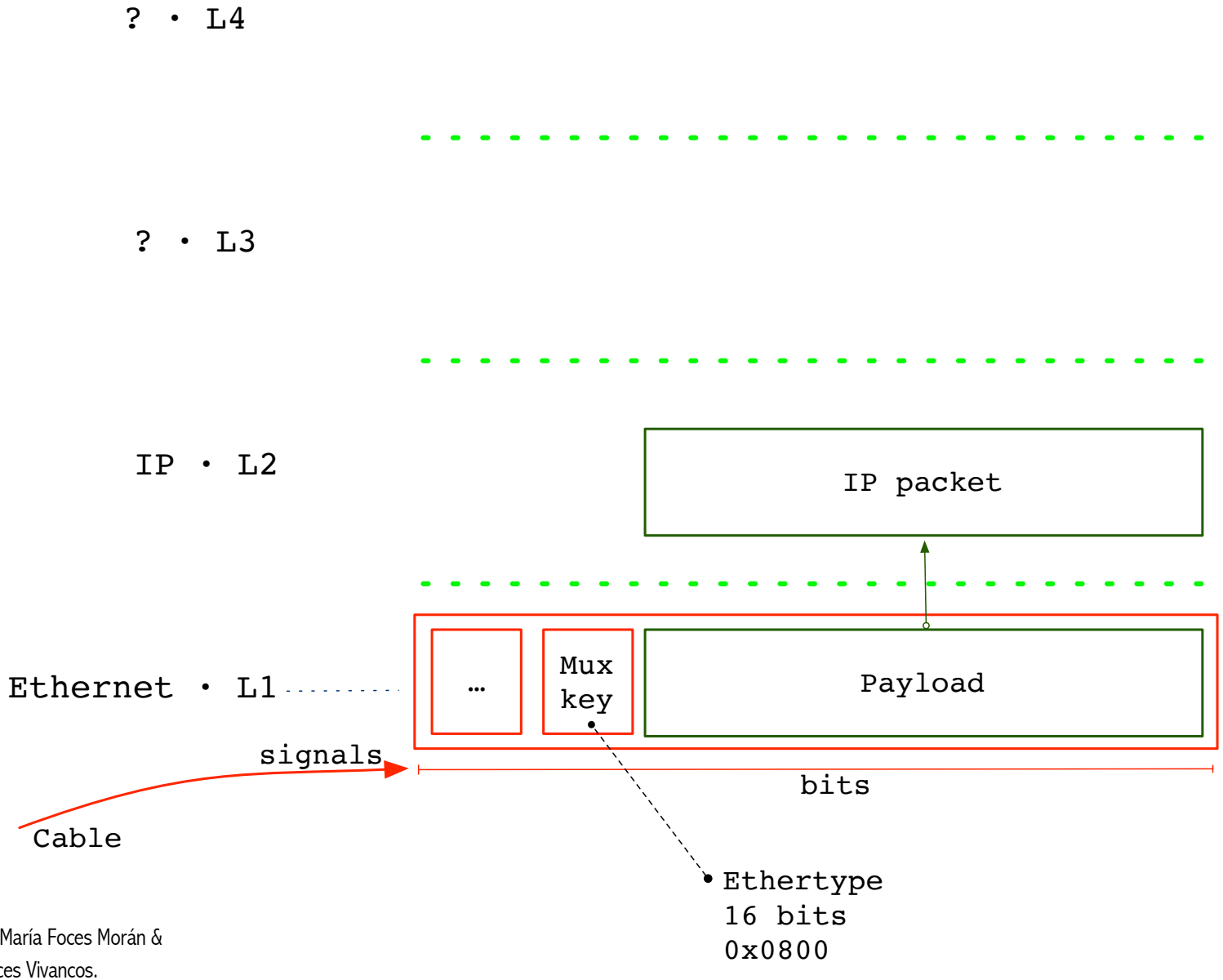
Frame arrives at destination host

- The encapsulation + multiplexing is reversed
- Upper layer protocol receives PDU
  - Deencapsulates Payload
  - Hands Payload to protocol indicated in the Mux Key
- Repeat this process until original http message arrives at the destination application protocol
  - **Exercise**: Explain with detail by using real protocol numbers and port numbers

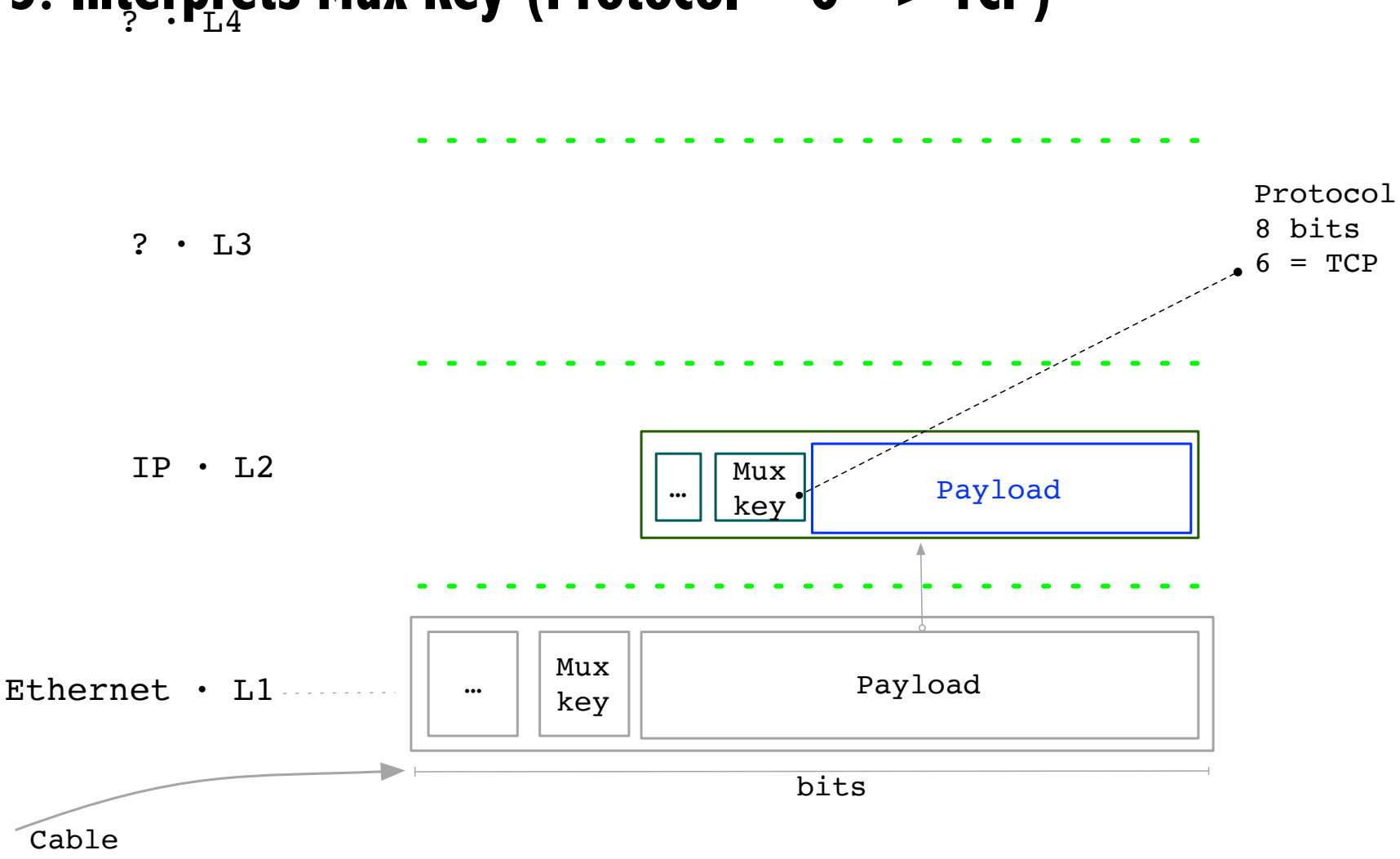
Ethernet receives new frame from wire



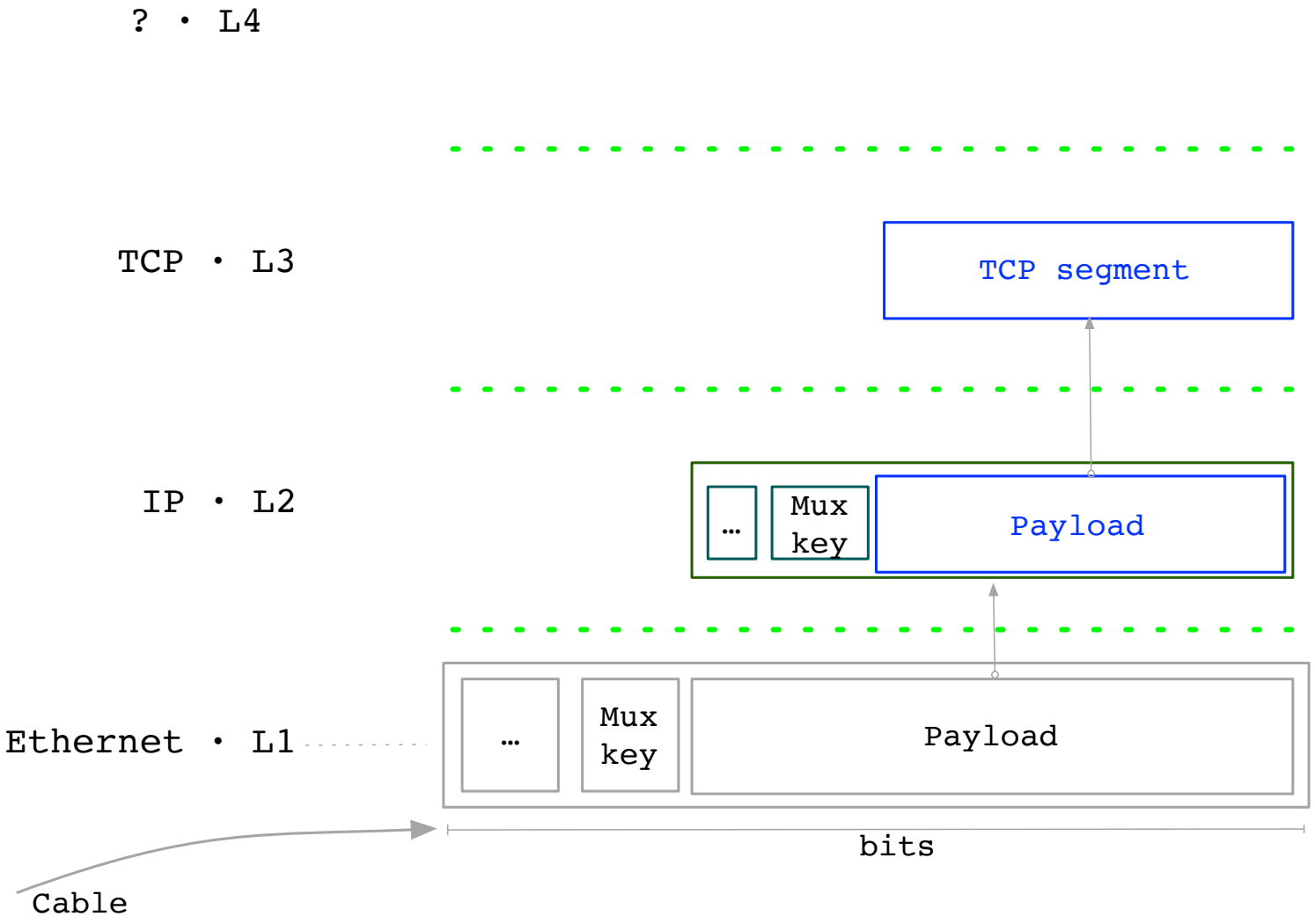
1. Ethernet deencapsulates Payload
2. hands it to IP since mux key is 0x0800



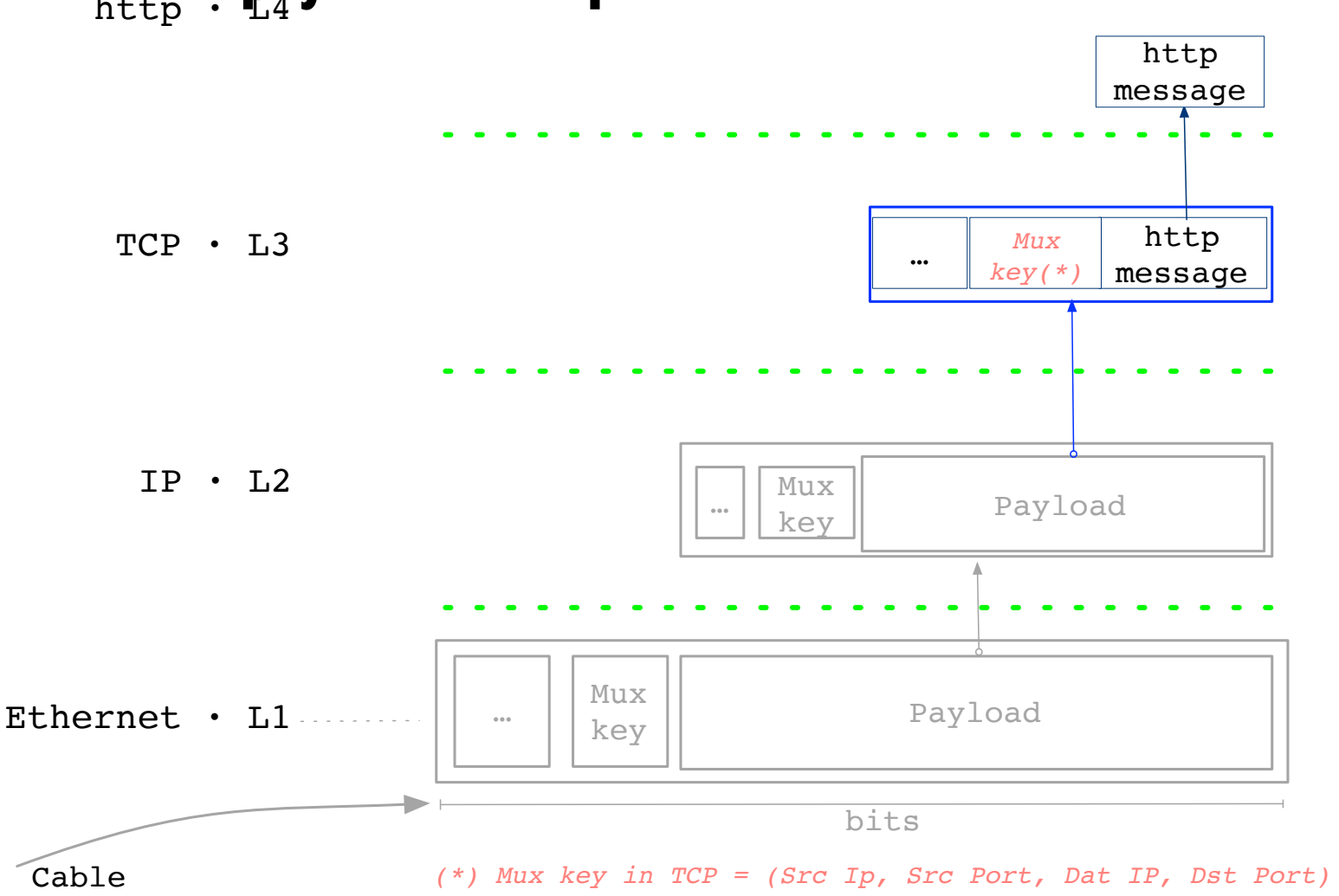
3. IP receives IP packet
4. Deencapsulates its Payload
5. Interprets Mux Key (Protocol = 6 => TCP)



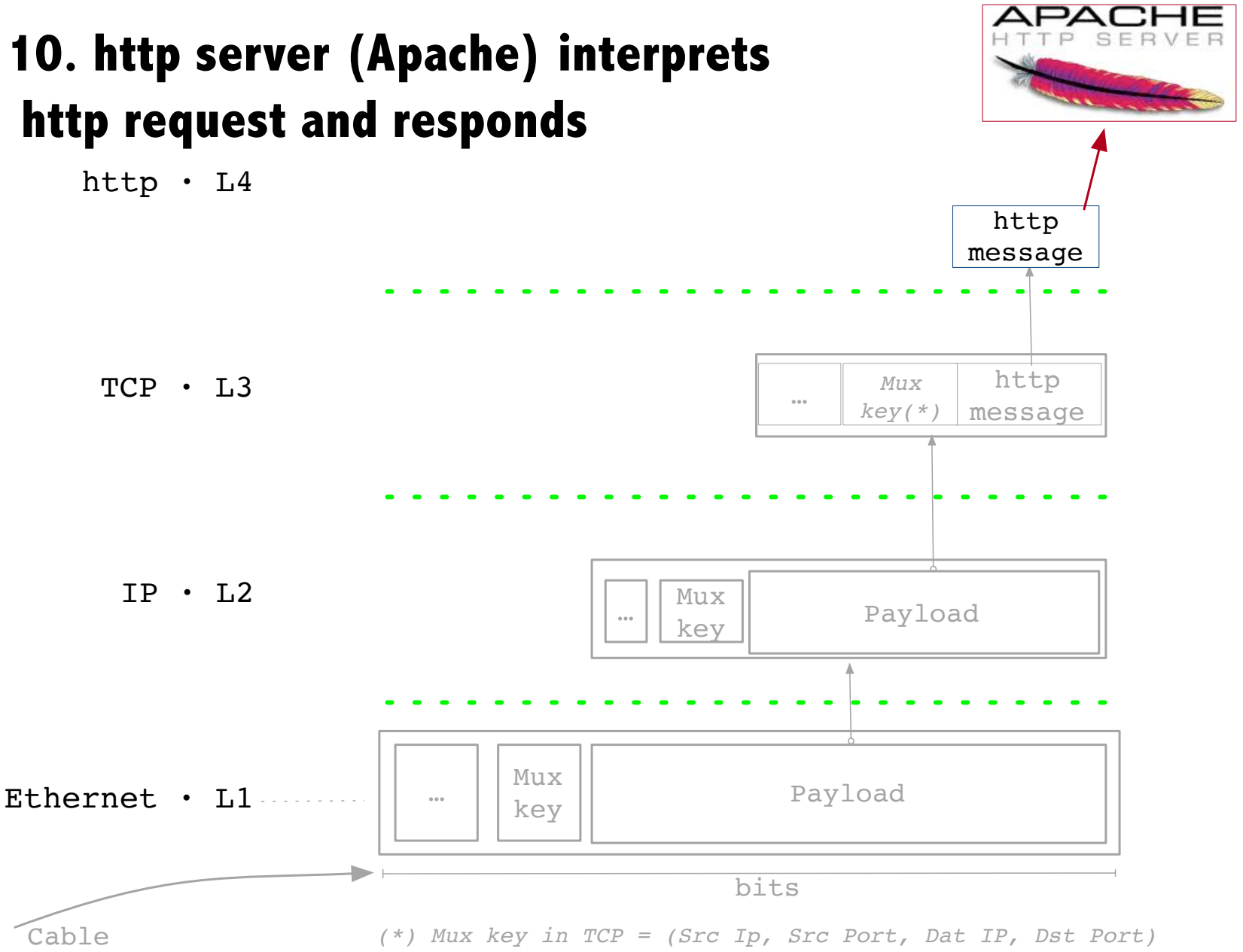
6. IP hands the TCP segment to TCP protocol



7. TCP deencapsulates Payload
8. Interprets mux key
9. Hands payload to http



10. http server (Apache) interprets http request and responds



21

Attaining scalability and preserving connectivity

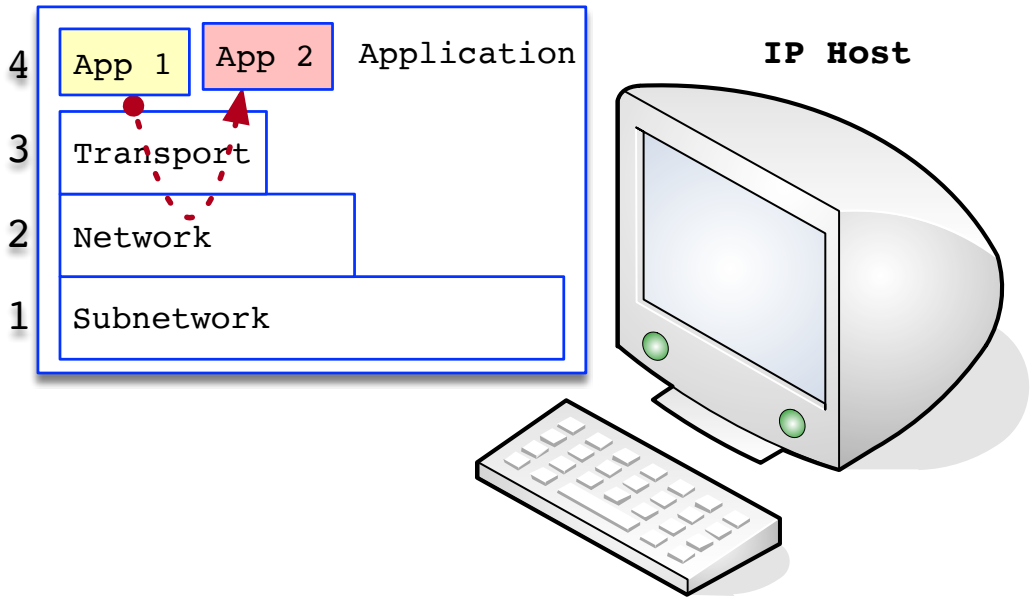
A story about most of the course

How to make networks bigger (Metcalf's Law)

- Scalable connectivity
- Design and build networks that get larger and at the same time preserve the communication capacity amongst the hosts
- Recall: all network technologies have a threshold size beyond which it does not properly scale

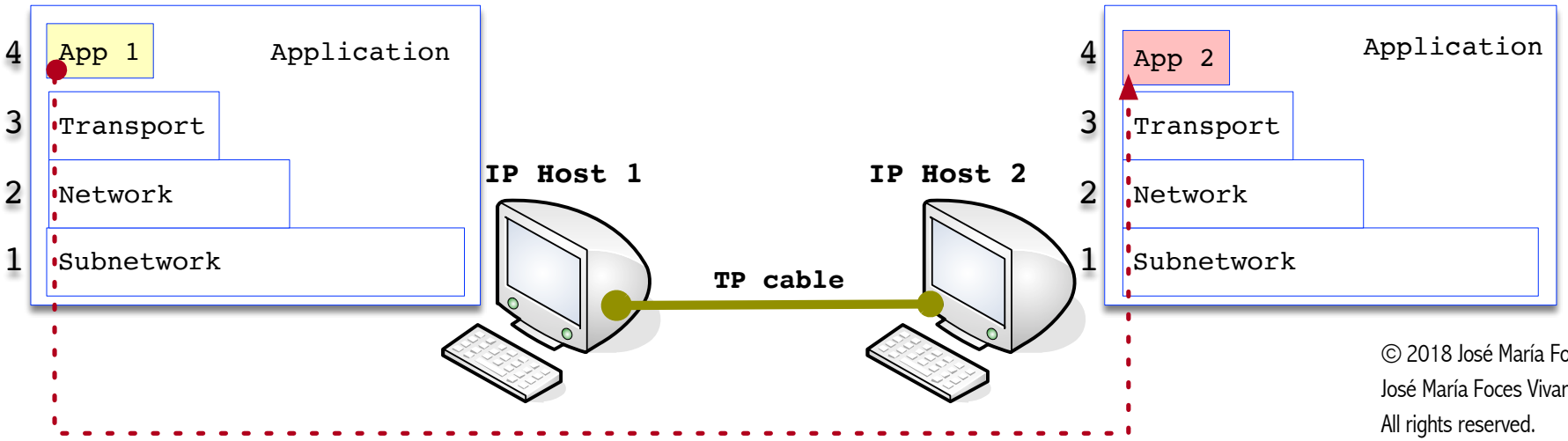
Attaining scalable connectivity

- A network technology is **scalable** if it can grow to huge sizes still preserving its communication functionality
- We wish that the connectivity offered by a network be scalable, i.e., that it can grow as needed, at least within some affordable limits
- The smallest conceivable network: comprised of only one host
  - The *loopback interface* at the network layer permits communication of two applications as though they were at different hosts



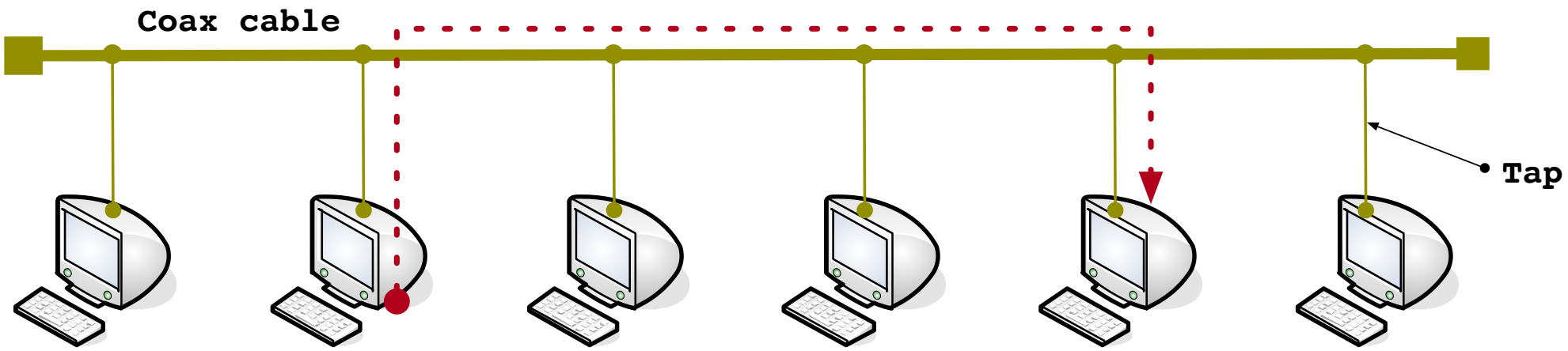
Two directly-connected hosts

- Two hosts are directly connected by means of an Ethernet TP (Twisted Pair) cable, for example
- This simplistic scheme works fine
- Resulting network is scalable because hosts can always communicate
- The TP cable and the Ethernet protocol form a so-called **point-to-point link**



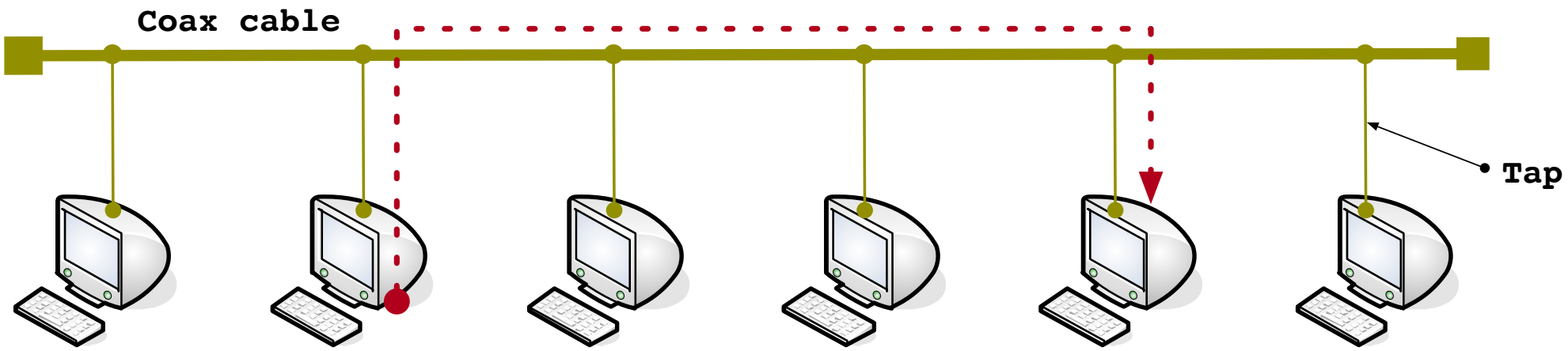
Many hosts connected to one coax cable

- Assume Ethernet technology
- Coax cable and Ethernet form a **Multiple Access link**
- **Half duplex:** only one communication at any specific time
- Scales well if number of hosts and utilization are *moderate*



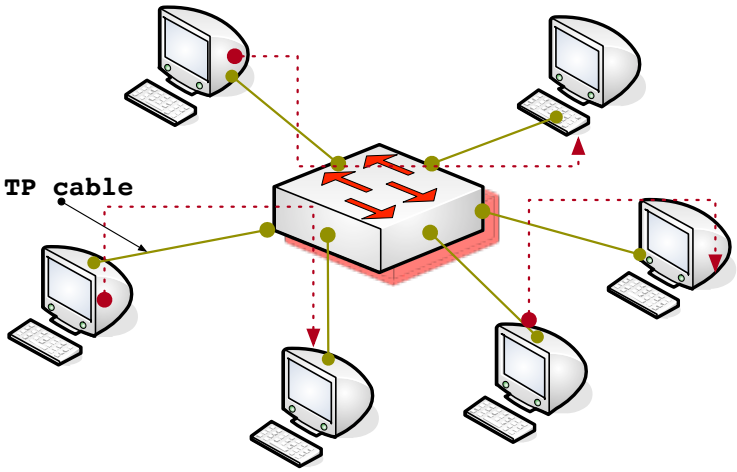
Many hosts connected to one coax cable

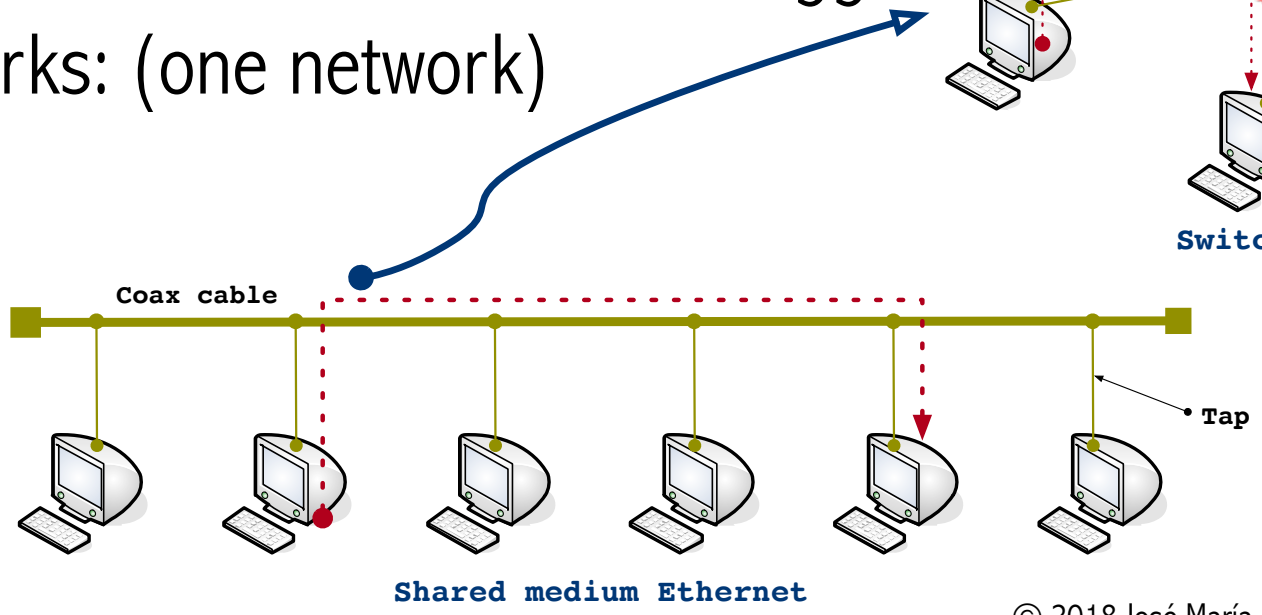
- Scales well if number of hosts and utilization is *moderate*
  - Number of of collisions small
- The resulting connection scheme forms a **single network**
- What can we do if number of hosts or utilization are high?
  - **Switching**



From shared cable to switching

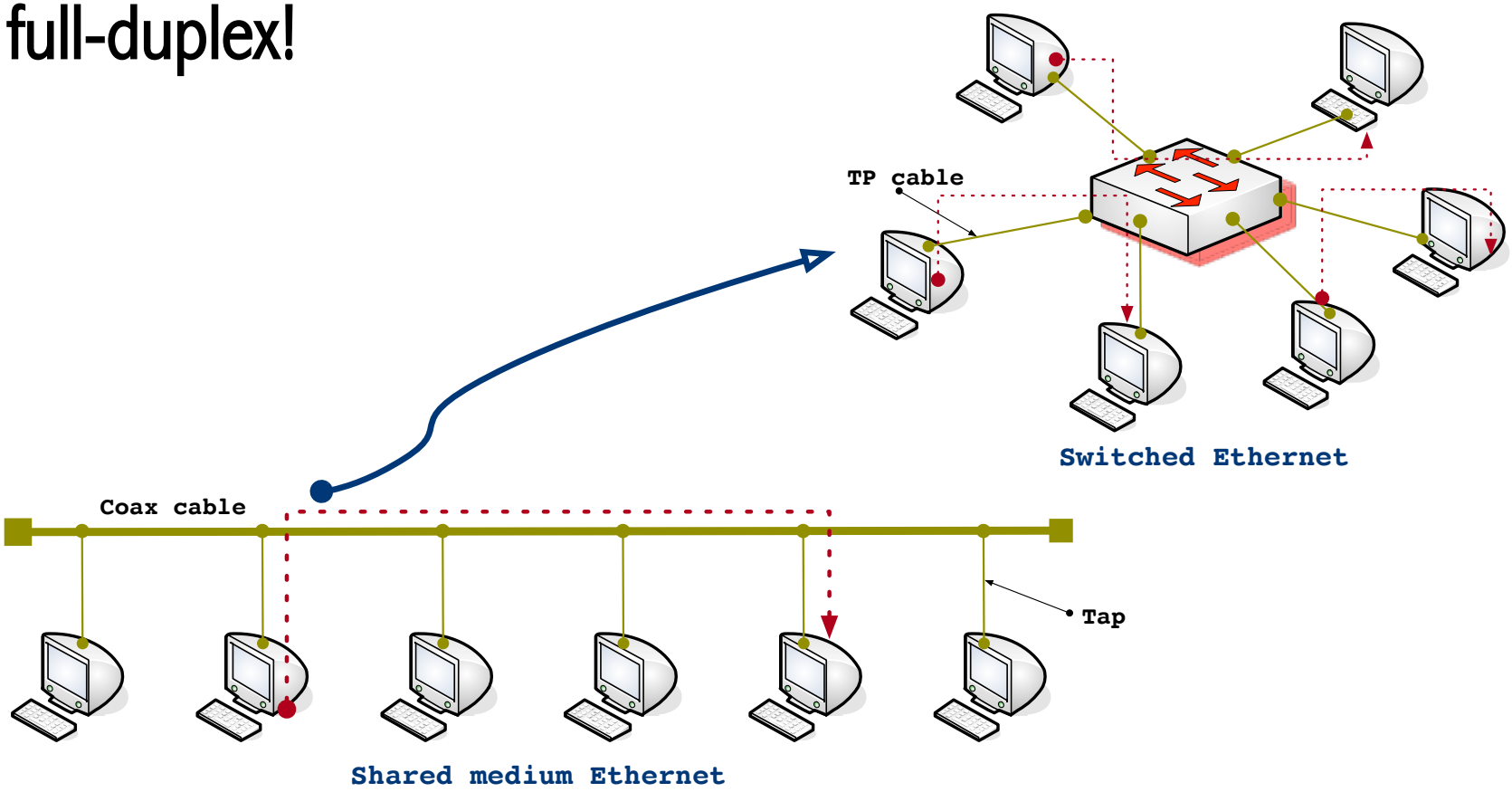
- New networking equipment: **switch**
  - Accepts a number of point-to-point hosts
  - It's a Store-and-forward device
- Can be interconnected to form bigger networks: (one network)



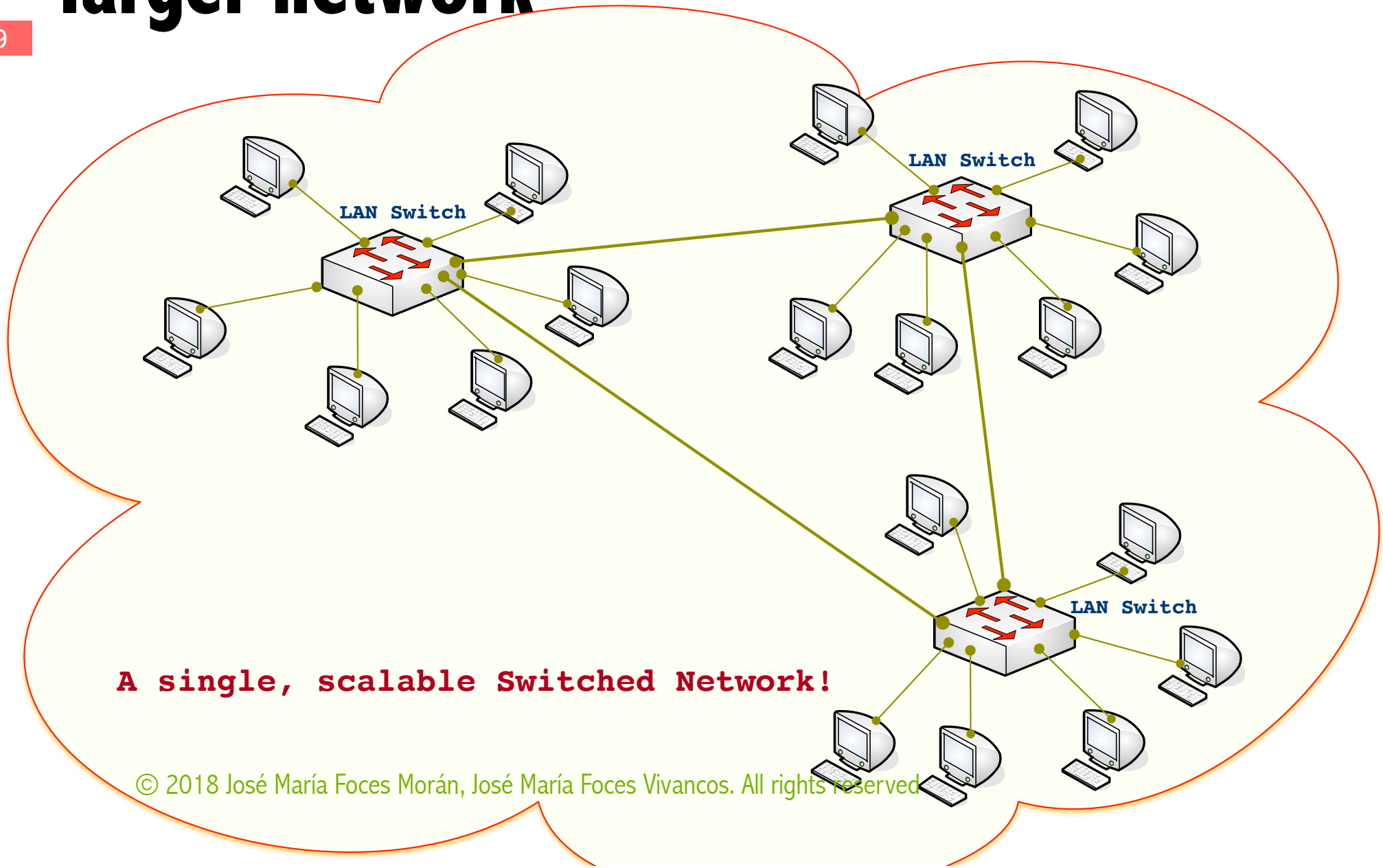


From shared cable to switching

- Switch permits several simultaneous flows: full-duplex!

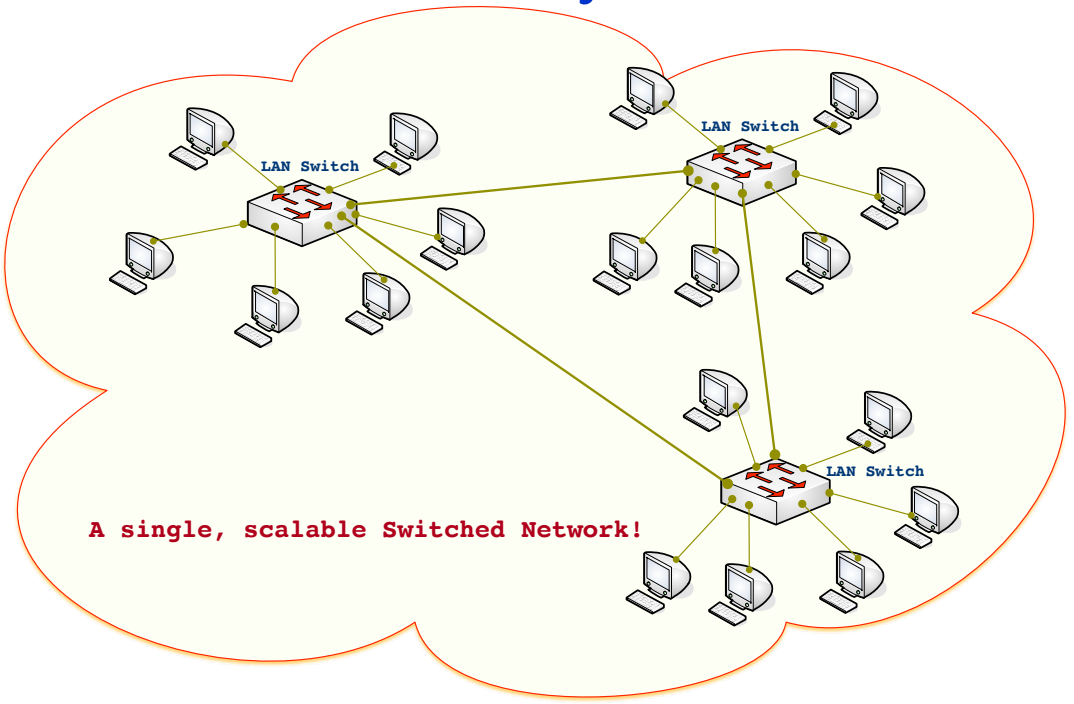


Switches can be connected to form *one* larger network



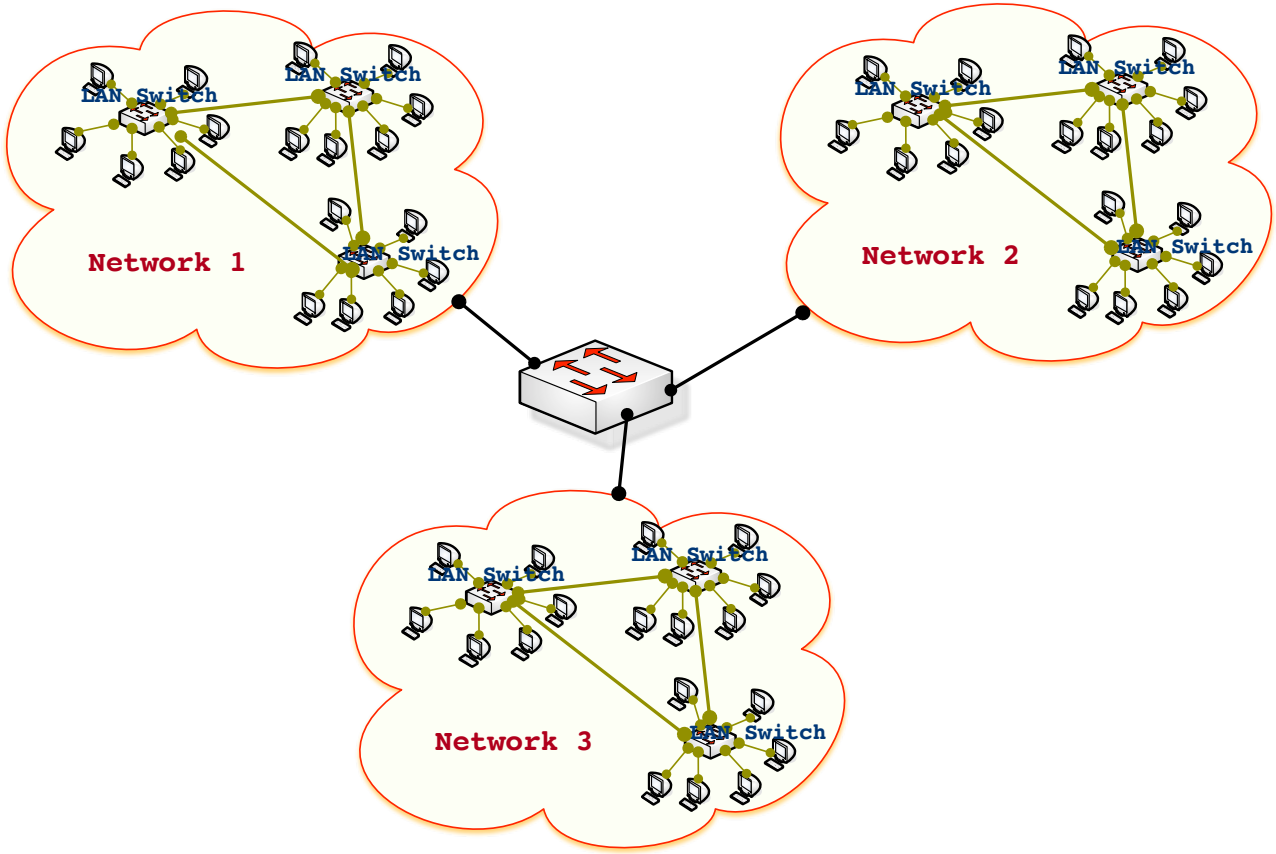
Scalability of a switched LAN

- A properly designed hierarchical switched LAN can house up to roughly 2500 hosts
- Depending on a host of factors
- Then how come the Internet, today has about 4000M of hosts?



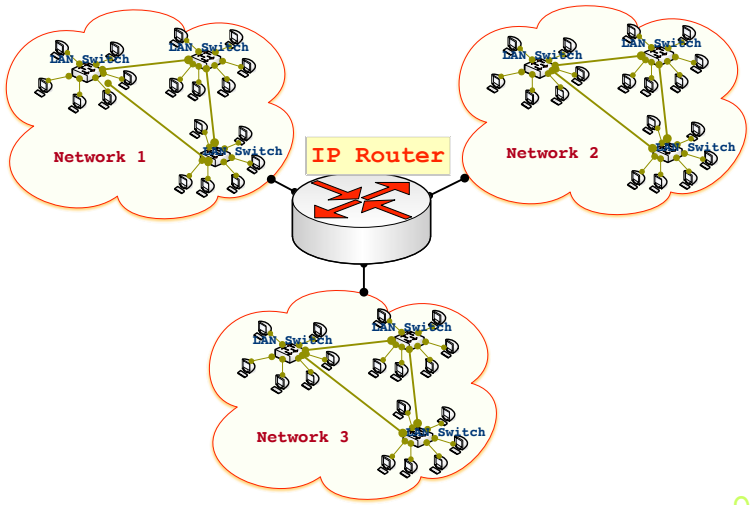
Scalability of a switched LAN

- The interconnection of three switched LANs results in a **single switched LAN**, a huge one, but still offering an aceptable communication capacity amongst the hosts?

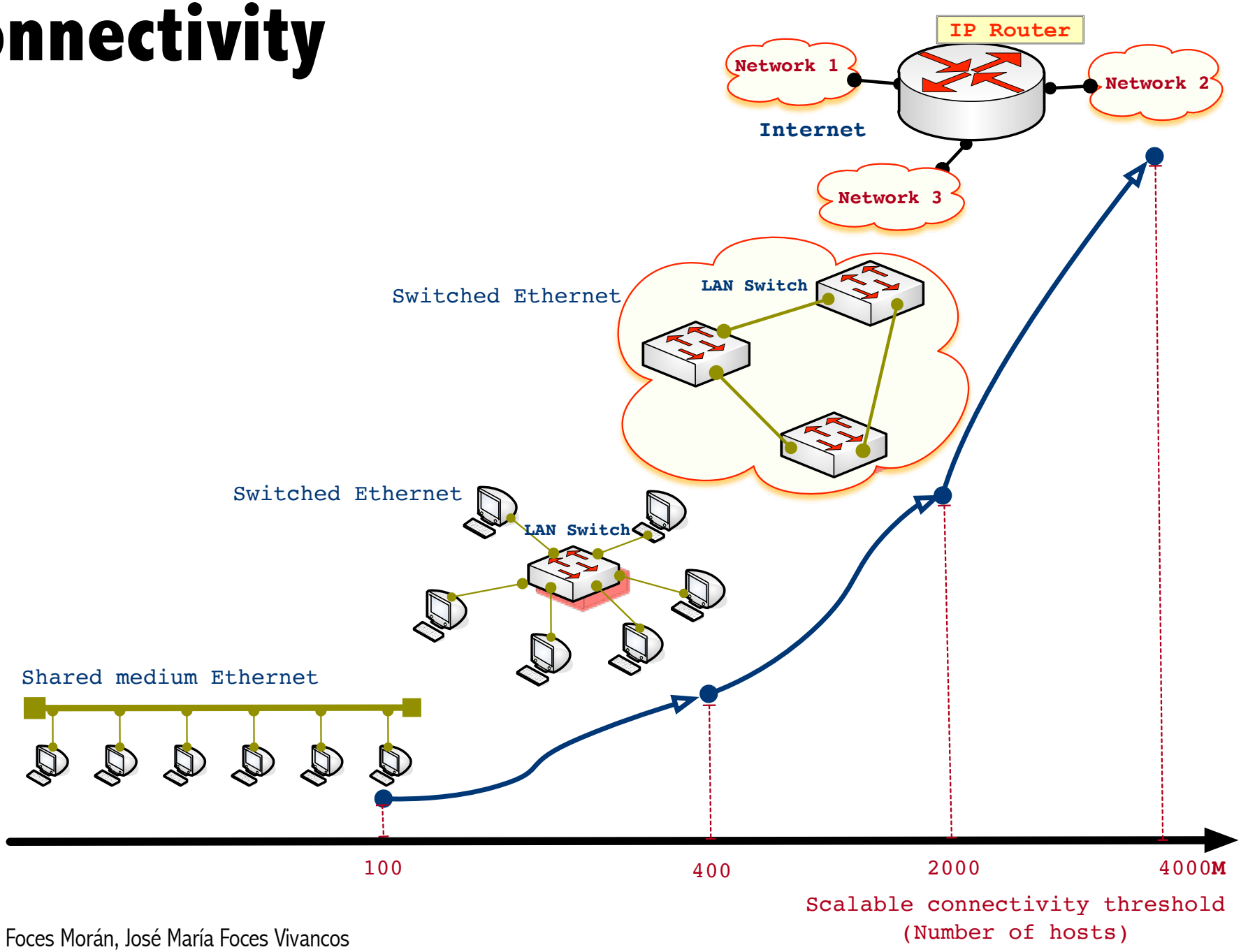


The scale of Internet, today

- The scale of Internet has been achieved thanks to:
  - Interconnecting networks
  - Internetwork
  - Using a single Internetwork Protocol: IP
  - And by using fast IP Routers



A story about scalable connectivity



End of Ch 1 Section 2