

# NETWORK BASICS

COMP 30650: NETWORKS AND INTERNET SYSTEMS

Dr. Gavin McAra

Email: [gavin.mcardle@ucd.ie](mailto:gavin.mcardle@ucd.ie)

Office Hours: Thursday 13:00 - 14:00

# RECAP

## Goals and Motivations for COMP 30650

- Fundamental understanding of what happens when you click a link, send an email, watch a video, etc.

## Use of Computer Networks

- Business, Home and mobile
- Centres on Communication and Resource Sharing
- Diverse uses

## Considerations for Designing Computer networks

- What the network needs to be able to do to deliver content
- 

# TODAY'S PLAN

## Network Hardware

- Components of networks
- Types of network


## Network Software

- Layers & Protocols



# LECTURE 2: NETWORK BASICS - OUTCOME

## Learning Outcomes

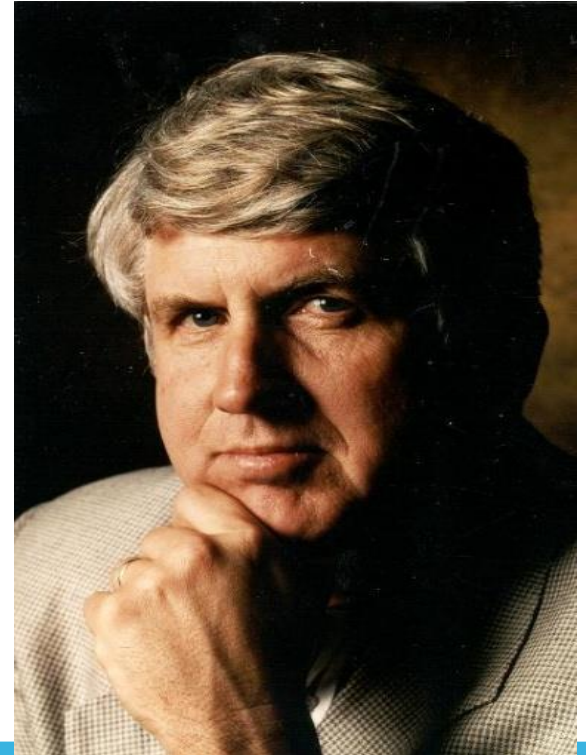
- Better understanding of the components of a computer network
  - Better understanding of the benefits of different types of Network Topology
  - of communication via a network Appreciate the complexity
  - Begin to understand the protocol stack
- 

# THE VALUE OF CONNECTIVITY

## “Metcalfe’s Law” ~1980:

- Large networks are relatively more valuable than small ones
- **Metcalfe's law** states that the value of a telecommunications network is proportional to the square of the number of connected users of the system ( $n^2$ ).

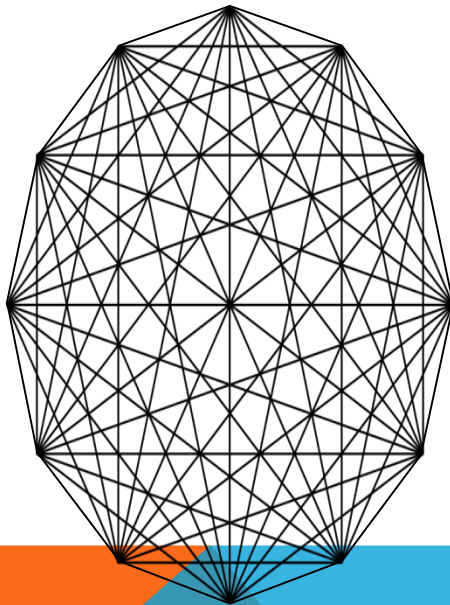
Bob Metcalfe



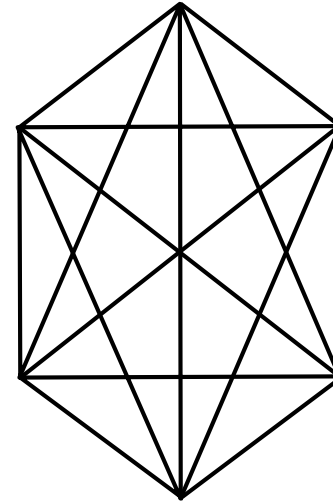
: © 2009 IEEE

# THE VALUE OF CONNECTIVITY

**Example: left network has much more connectivity despite only having twice as many nodes**



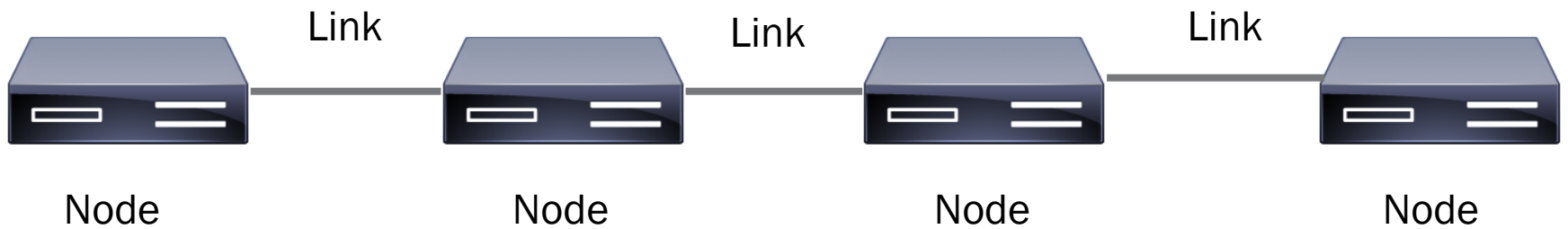
VS



$$x = n(n - 1) / 2$$


# NETWORK COMPONENTS

Is this a network?



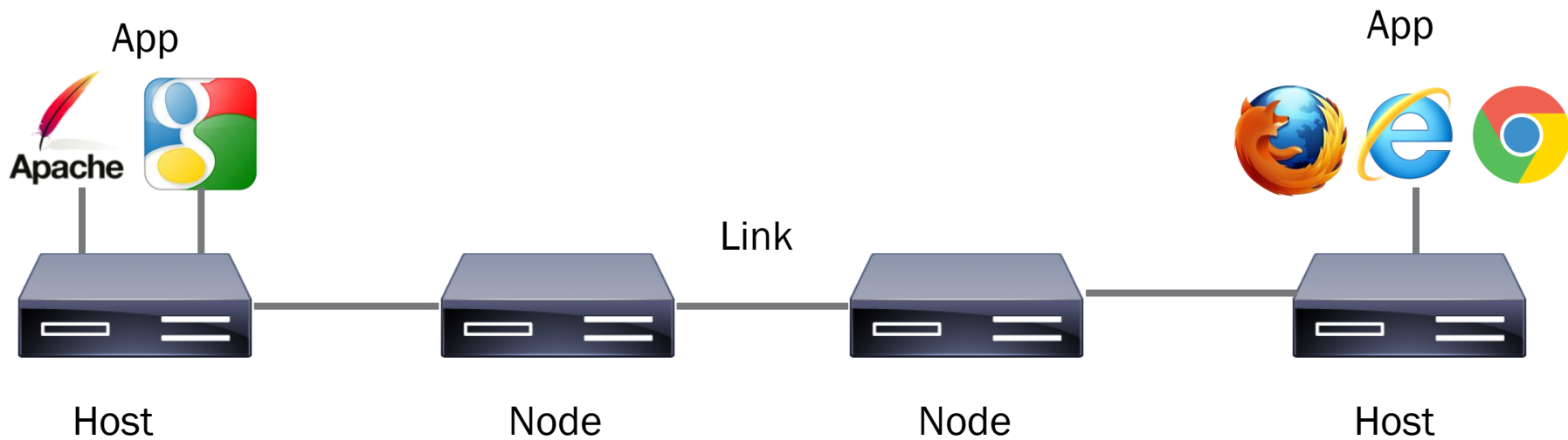
# A NETWORK

## Definitions of a network

- An interconnected collection of autonomous **nodes**
    - Interconnected = able to exchange information
  - A set of **nodes** connected by media **links** node
    - Any device capable of sending &/or receiving data to &/or from other nodes in the network
  - A connected collection of hardware and software that permits information exchange and resource sharing
    - Information = data, text, audio, video, images,
    - Resources = printers, memory, link bandwidth
- 

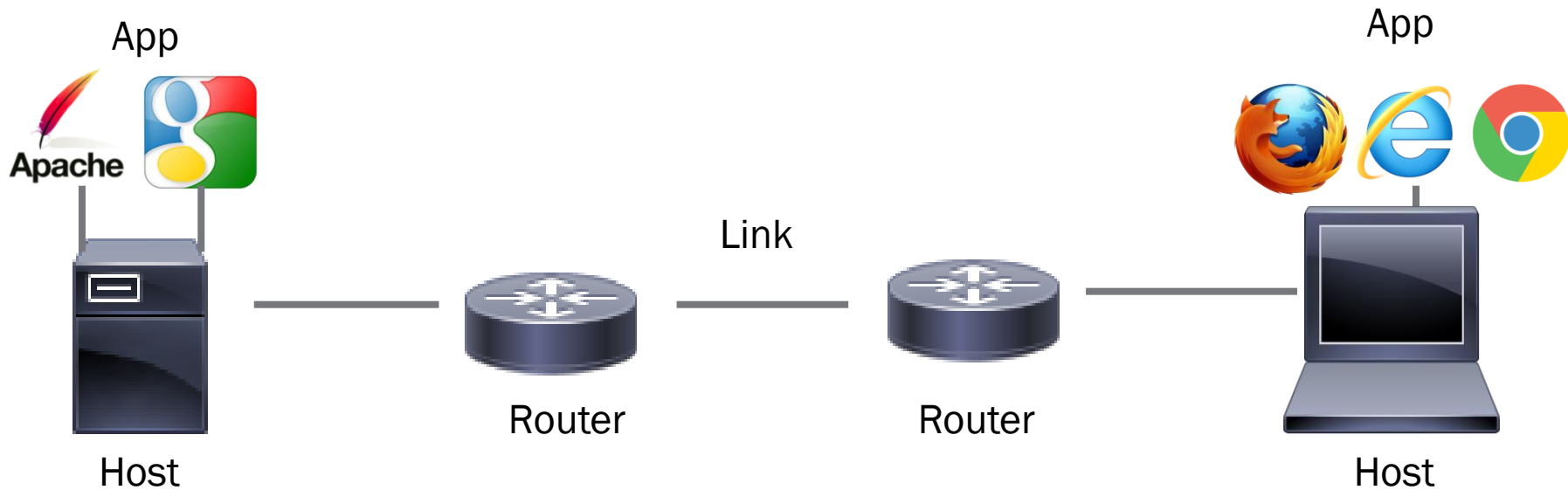


# NETWORK COMPONENTS



# NETWORK COMPONENTS

Nodes and links



# NETWORK COMPONENTS

Component	Function	Example
<b>Application</b> , app, user	Uses the network and gets service from it	Skype, web browser, Gmail
<b>Host</b> , end system, end device, node, source, sink	Supports applications	Laptop, server, PC, mobile phone
<b>Router</b> , switch, node hub, intermediate system	Relays messages between links	Access point, modem
<b>Link</b> , channel	Connects nodes	Wires, fibre optics, wireless

# LINKS

## Types of links

- Full-Duplex
  - Bidirectional
- Half-Duplex
  - Bidirectional
- Simplex
  - Unidirectional



# WIRELESS LINKS

## Message is broadcast

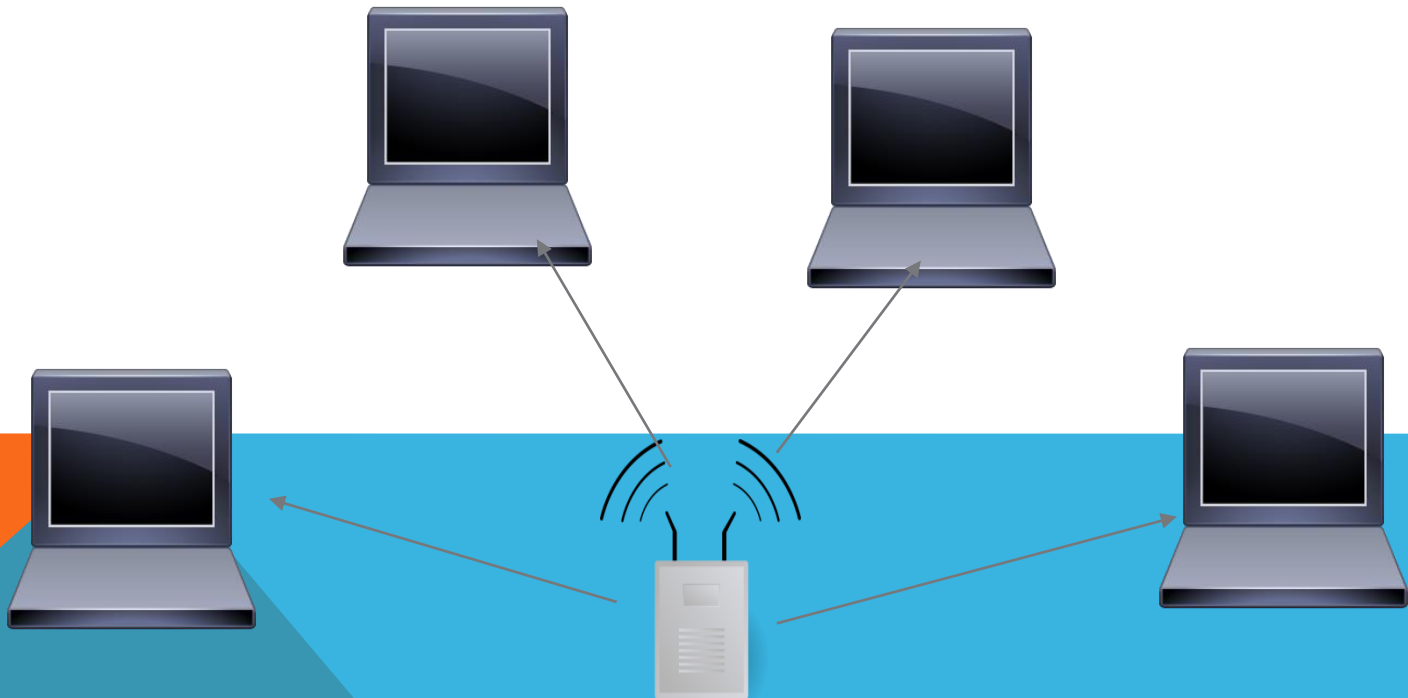
- Received by all nodes in range
- Not a good fit with our model



# WIRELESS LINKS

## Message is broadcast

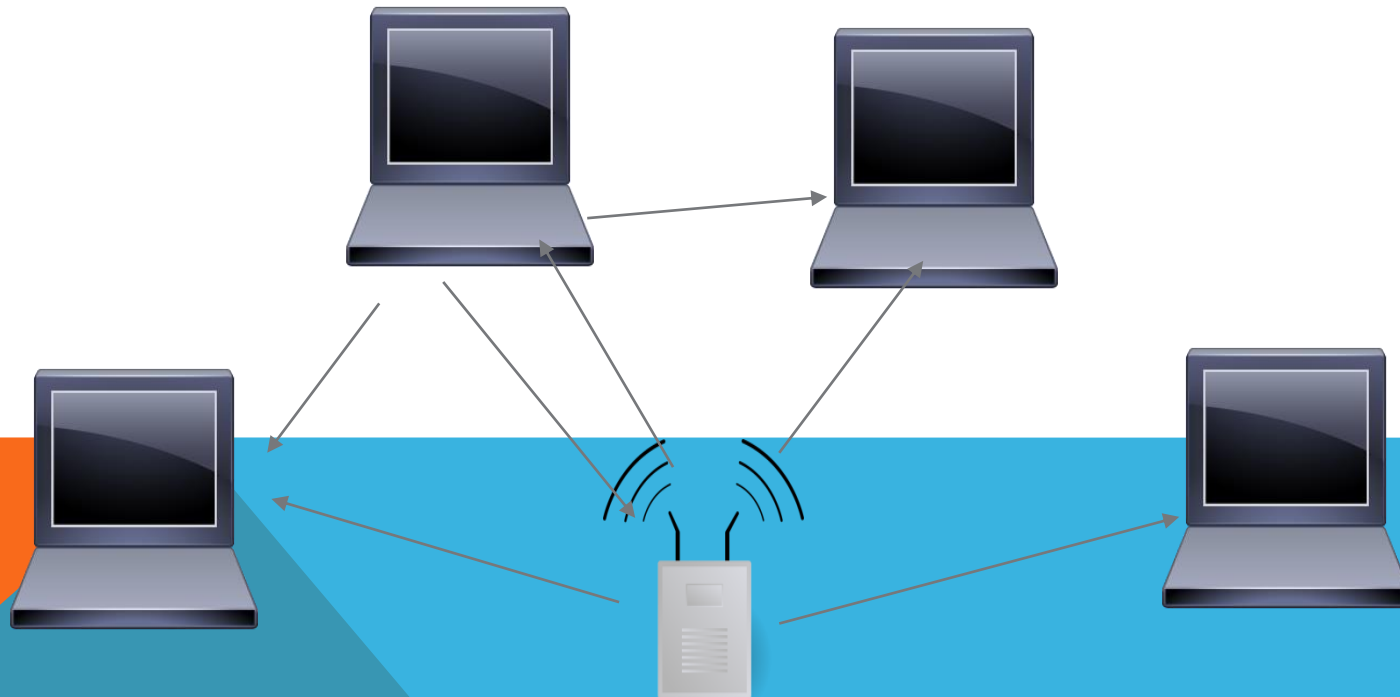
- Received by all nodes in range
- Not a good fit with our model



# WIRELESS LINKS

## Message is broadcast

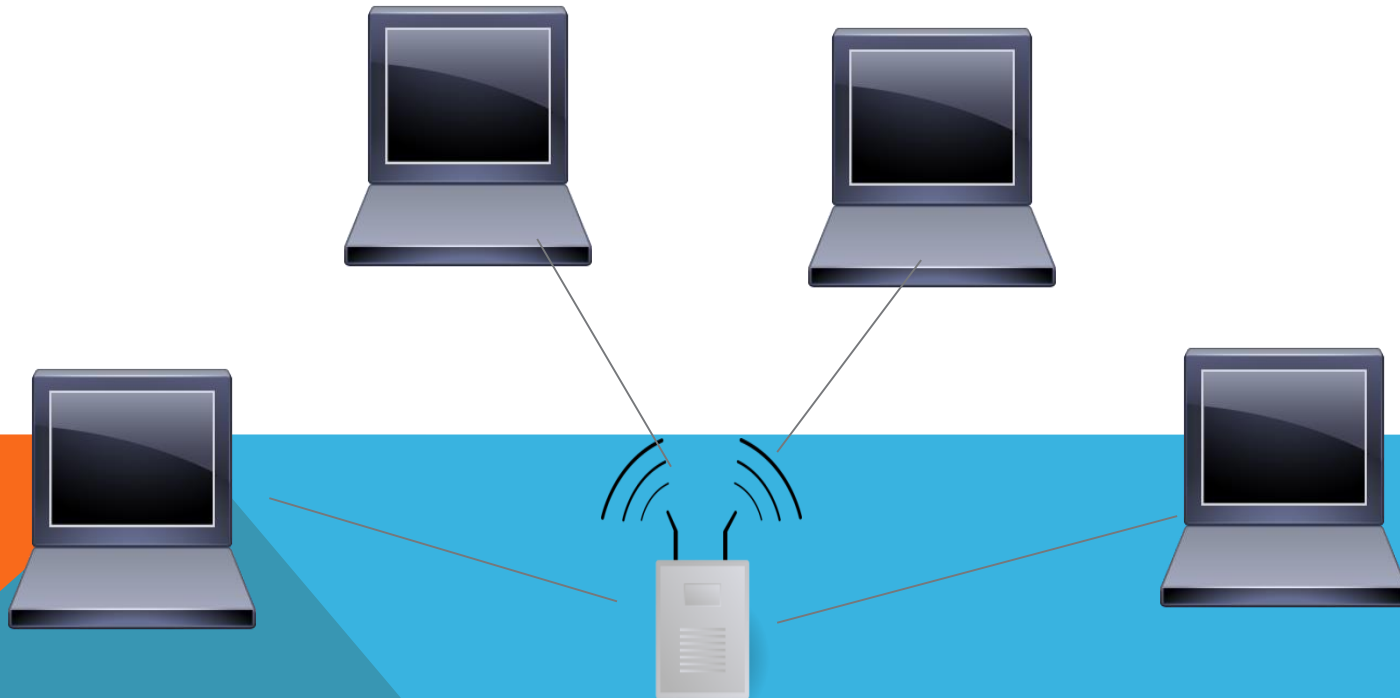
- Received by all nodes in range
- Not a good fit with our model



# WIRELESS LINKS

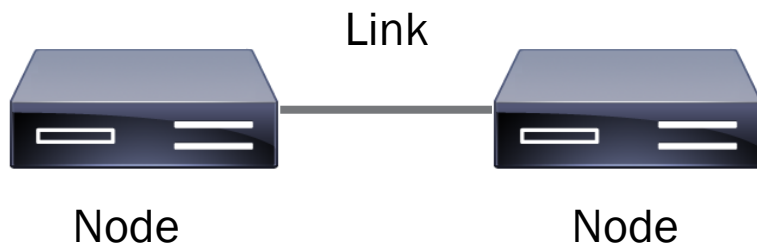
## Message is broadcast

- Received by all nodes in range
- Not a good fit with our model

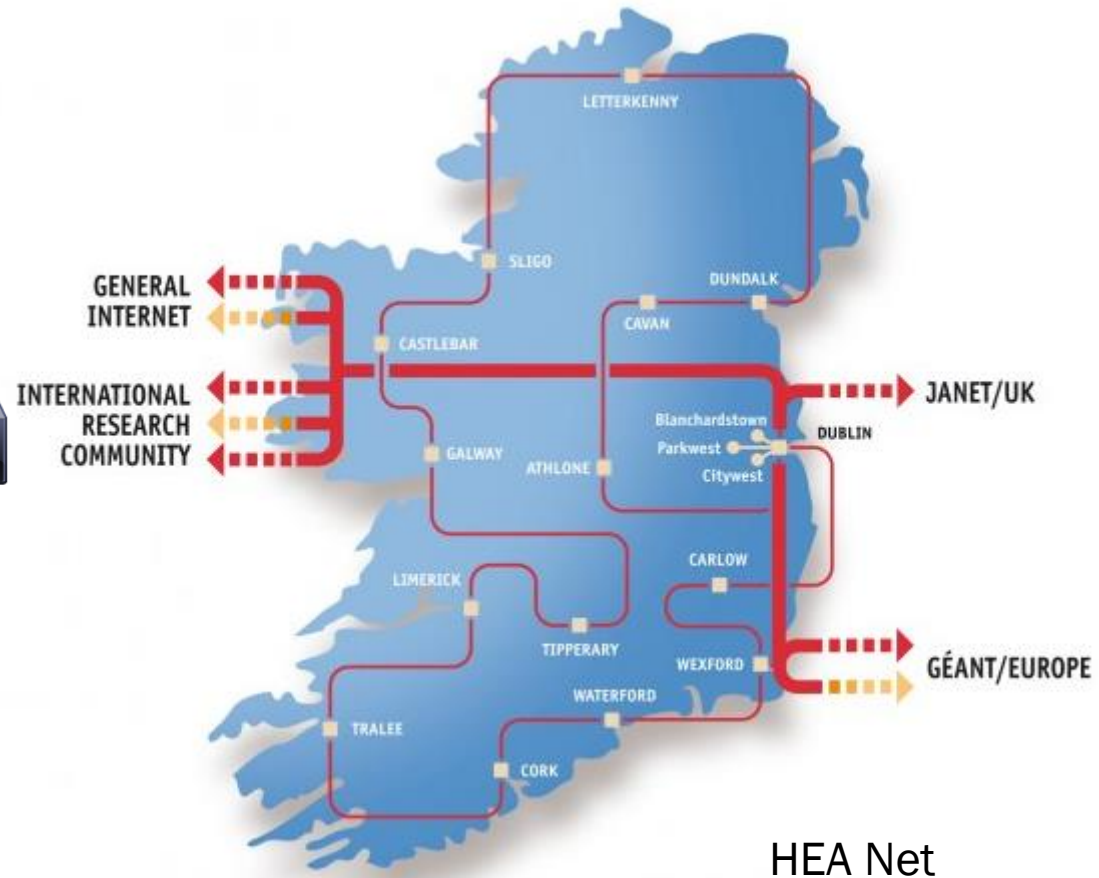
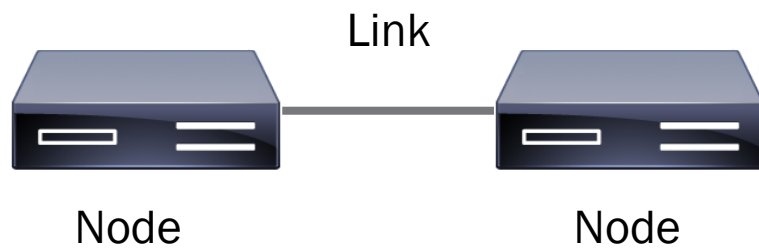




# NETWORK SCALE



# NETWORK SCALE



# NETWORK SCALE

Scale	Type	Example
Vicinity	Personal Area Network (PAN)	Bluetooth
Building	Local Area Network (LAN)	Wi-Fi, Ethernet
City	Metropolitan Area Network (MAN)	Cable, DSL
Country	Wide Area Network (WAN)	Large ISP
Planet	The Internet (network of networks)	The Internet

# INTERNETWORKS

**A network is an interconnected collection of autonomous devices**

- interconnected = able to exchange information

**An internetwork or an internet is what you get when you join networks together**


- Combine networks together

The Internet is an internet we all use and access using a specific protocol called Internet Protocol (IP)

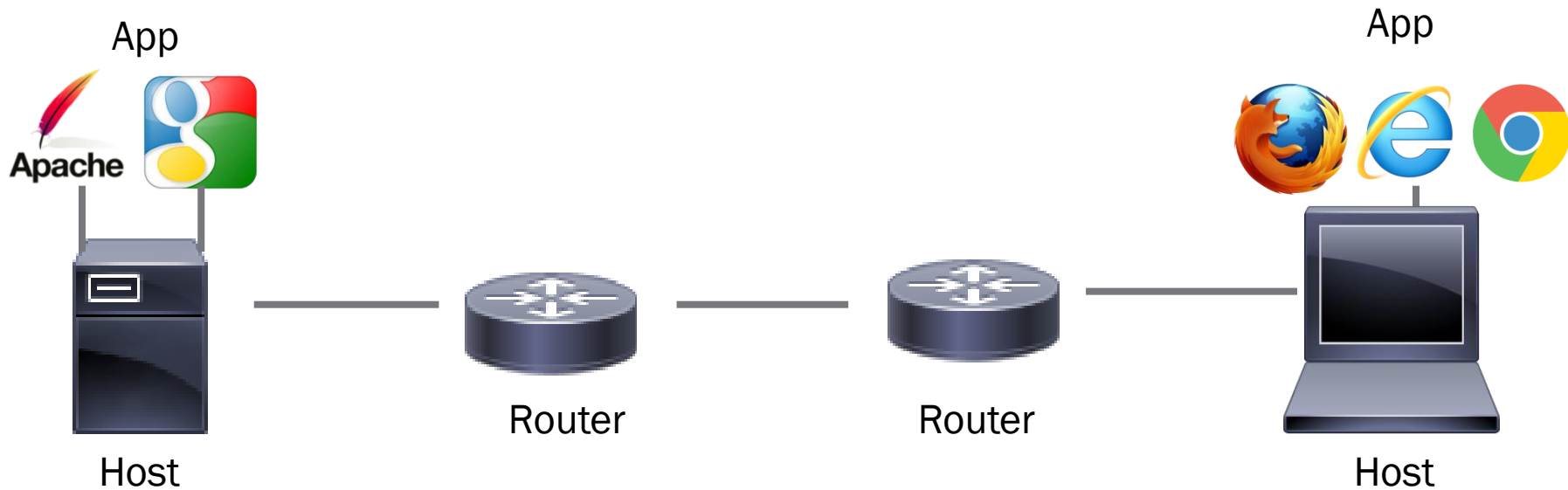


# EXAMPLE NETWORKS

Components & models apply to most networks

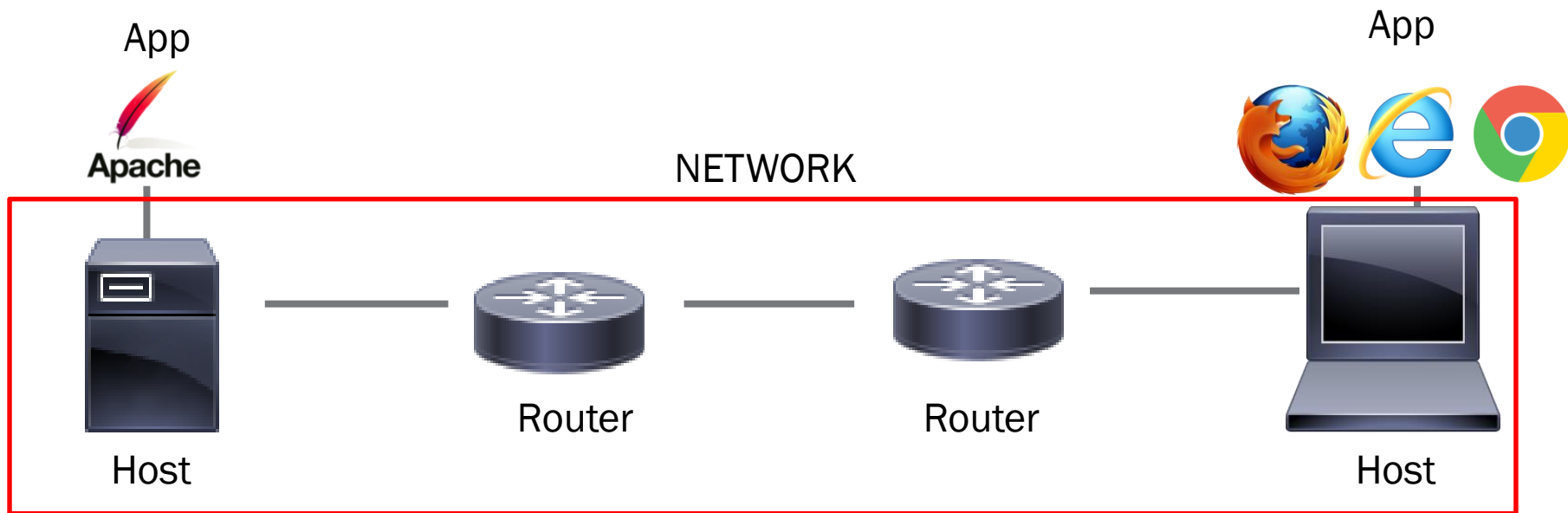
- Wifi-802.11
  - Ethernet
  - Internet Service Provider (ISP)
  - Cable/DSL networks
  - Mobile Phone network
  - Bluetooth
  - Telephone
  - Satellite
- 

# NETWORK BOUNDARIES



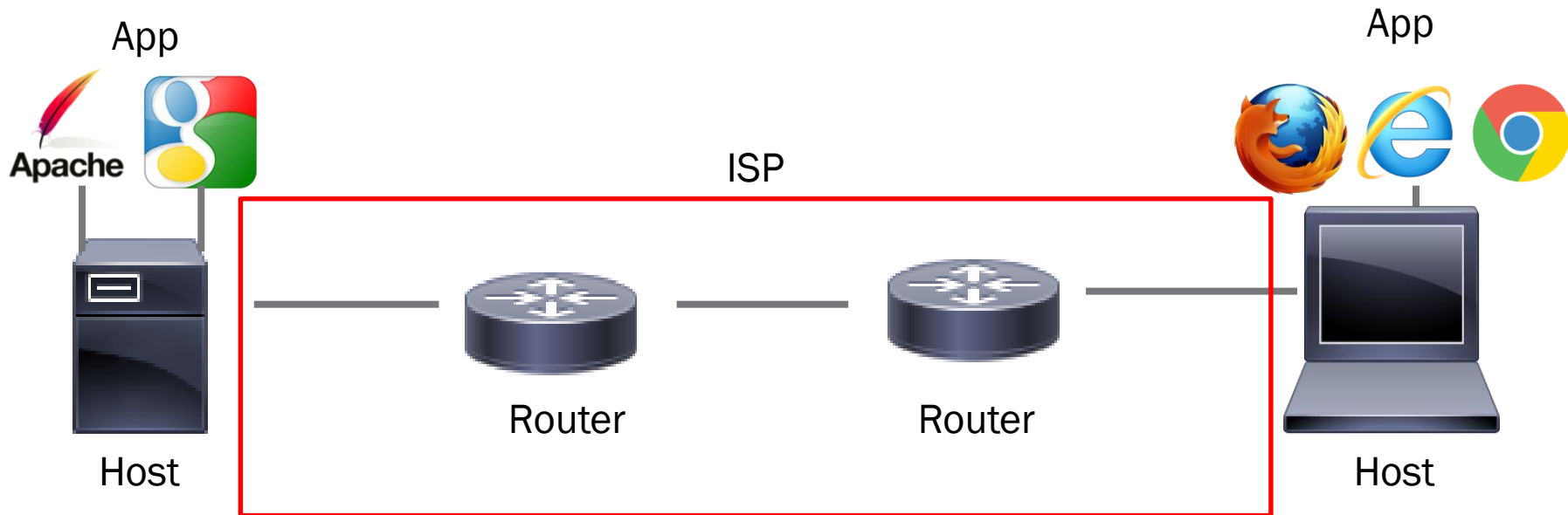
# NETWORK BOUNDARIES

Network



# NETWORK BOUNDARIES

Internet Service Provider

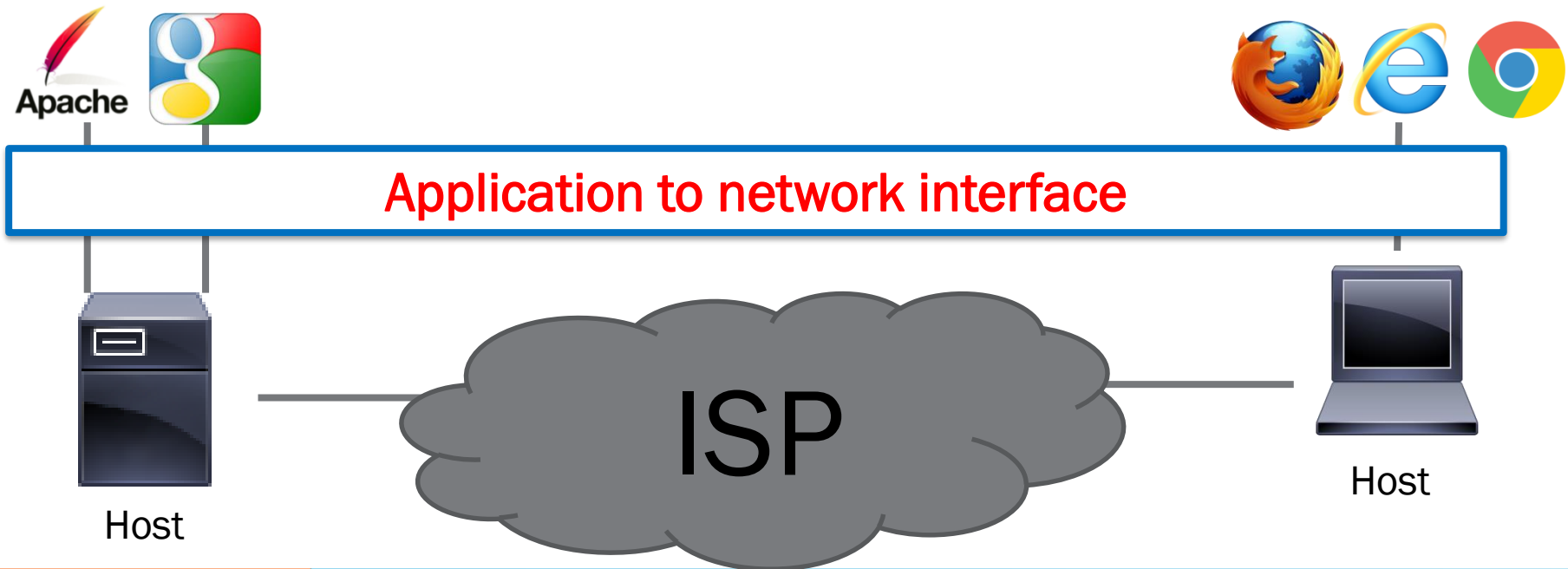




# KEY INTERFACES IN A NETWORK

## Network – Application Interface

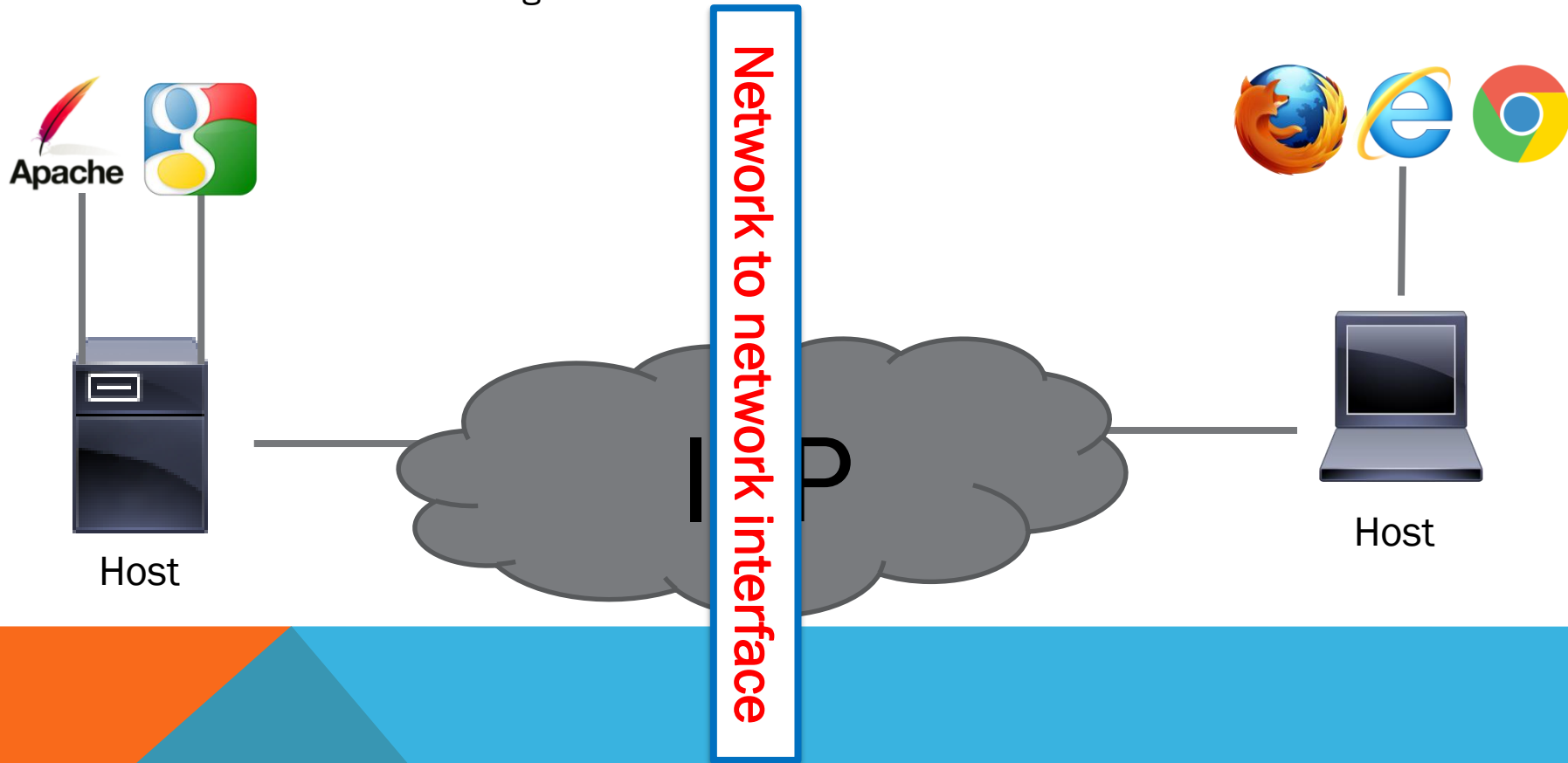
- Define how applications use the network



# KEY INTERFACES IN A NETWORK

## Network – Network Interface

- Define how nodes work together



# NETWORK TOPOLOGY

**Layout of connected devices on a network.**

- Think of it as the logical "shape" of the network wiring
- Logical means how it looks as a pure design concept, rather than how it actually looks physically



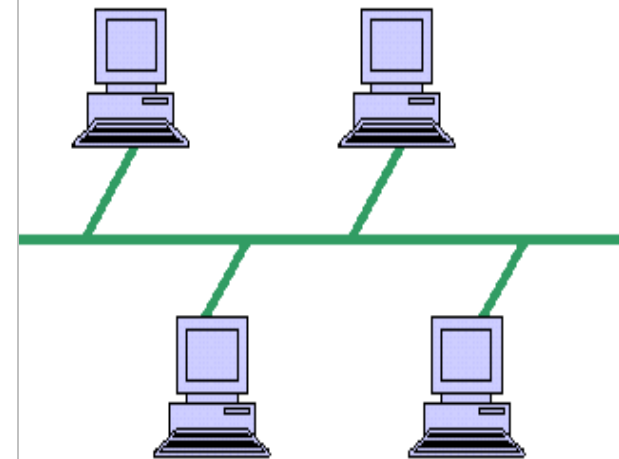
# BUS TOPOLOGY

## PROS

1. It is easy to set-up and extend bus network.
2. Cable length required for this topology is minimal
3. Bus topology is cheap.
4. Mostly used in small networks (e.g. LAN)

## CONS

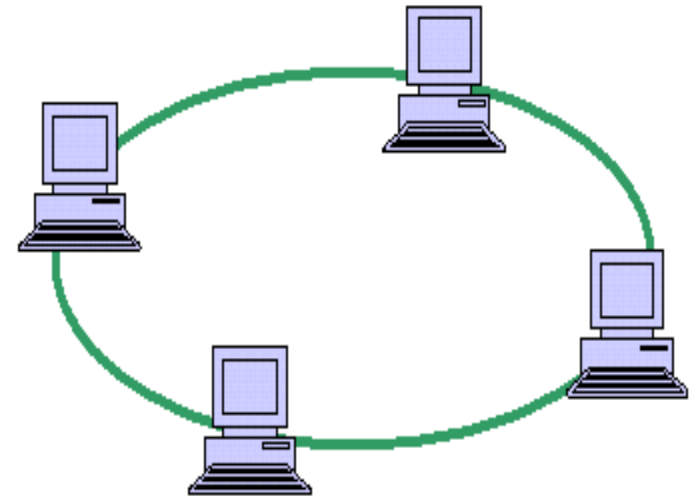
1. There is a limit on central cable length and number of nodes that can be connected.
2. Dependency on BUS cable
3. Proper termination is required to dump signals.
4. Efficiency of Bus network reduces, as the number of devices connected to it increases.
5. It is not suitable for networks with heavy traffic.
6. Security is very low because all the computers receive the sent signal from the source.



# RING TOPOLOGY

## Pros

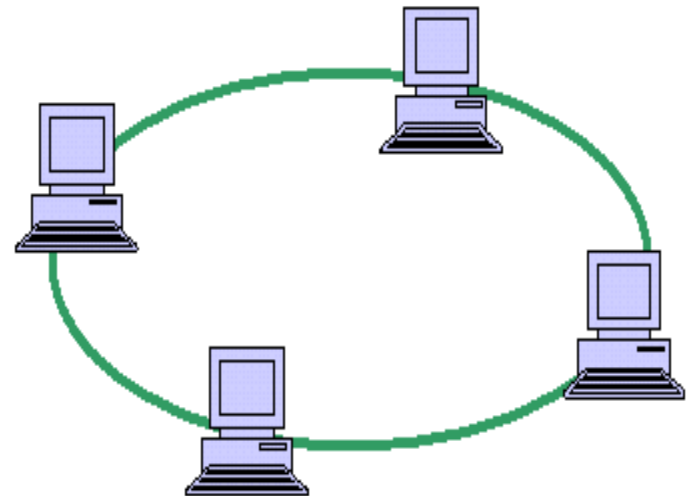
1. Very organized. Each node gets to send the data when it receives an empty token. This helps to reduce chances of collision. Also in ring topology all the traffic flows in only one direction at very high speed.
2. Even when the load on the network increases, its performance is better than BUS
3. There is no need for network server to control the connectivity between workstations.
4. Additional components do not affect the performance of network.
5. Each computer has equal access to resources.



# RING TOPOLOGY

## Cons

1. Each packet of data must pass through all the computers between source and destination – slow?
2. If one workstation or port goes down, the entire network gets affected.
3. Network is highly dependent on the wire.
4. Network cards are expensive as compared to Ethernet cards and hubs.



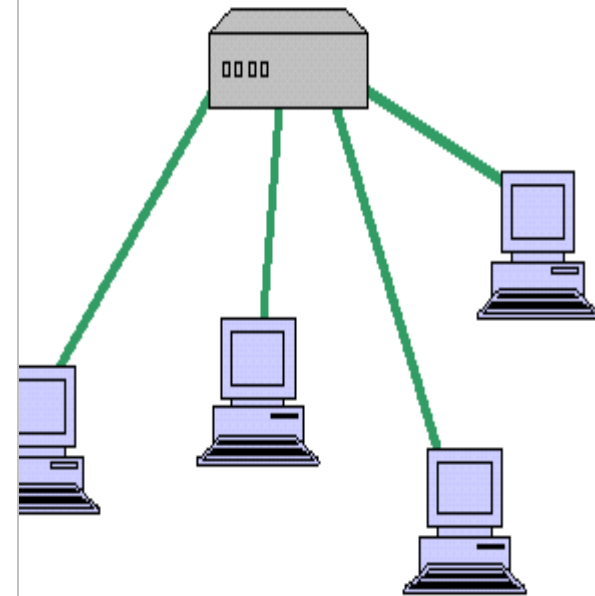
# STAR TOPOLOGY

## PROS

1. Better performance. A signal reaches the intended destination after passing through no more than 3-4 devices and 2-3 links.
2. Easy to connect/remove nodes or devices.
3. Centralized management - helps in monitoring the network.
4. Failure of one node or link doesn't affect the rest of network. At the same time its easy to detect the failure and troubleshoot it.

## CONS

1. Too much dependency on central device has its own drawbacks. If it fails whole network goes down.
2. The use of hub, a router or a switch as central device increases the overall cost of the network.
3. Performance depends on capacity of central device.



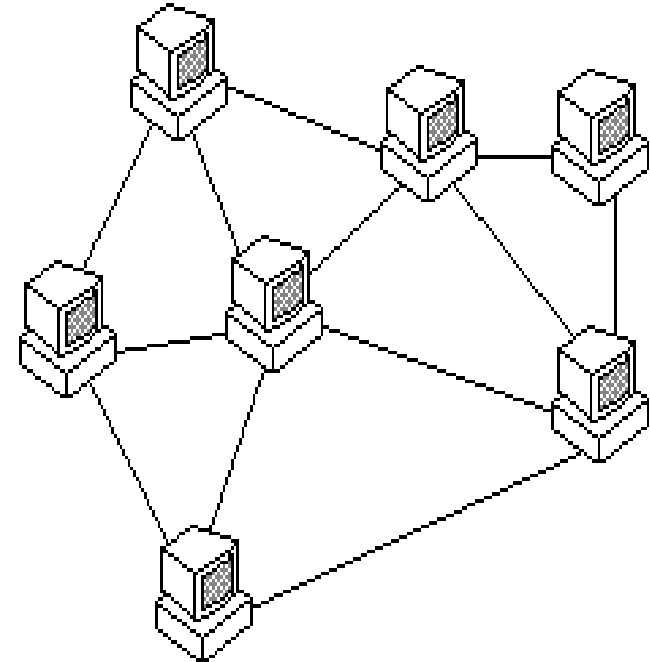
# MESH TOPOLOGY

## Pros

1. Data can be transmitted from different devices simultaneously. (high traffic).
2. Even if one of the components fails there is always an alternative present. So data transfer doesn't get affected.
3. Expansion and modification in topology can be done without disrupting other nodes.

## Cons

1. There are high chances of redundancy in many of the network connections.
2. High costs:  
Set-up and maintenance of this topology is very difficult.





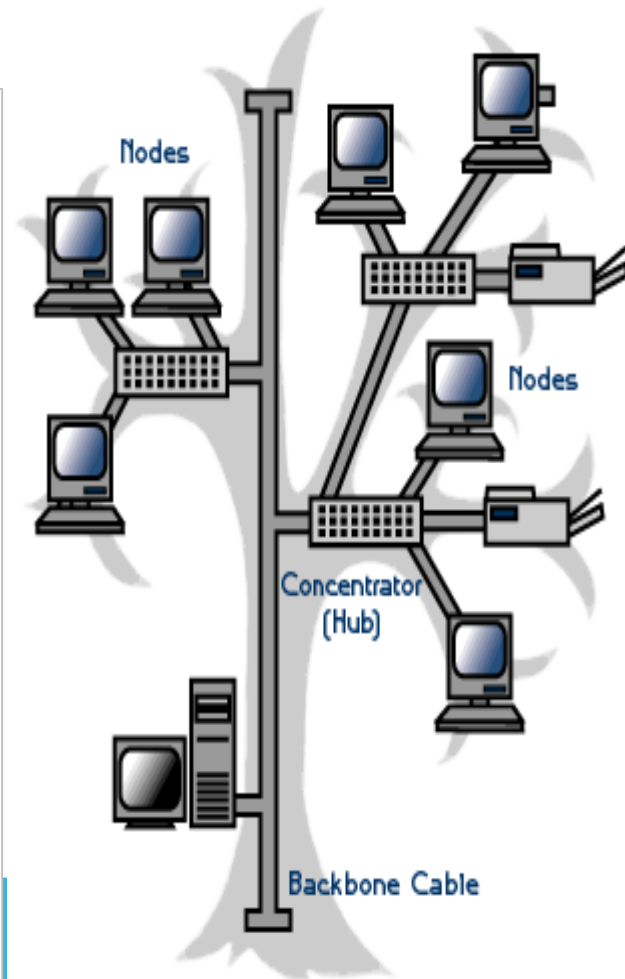
# TREE TOPOLOGY

## Pros

1. Expansion of Network is possible and easy.
2. Here, we divide the whole network into segments (star networks), which can be easily managed and maintained.
3. Error detection and correction is easy.
4. Each segment is provided with dedicated point-to-point wiring to the central hub.
5. If one segment is damaged, other segments are not affected.


## Cons

1. Relies heavily on the main bus cable, if it breaks whole network is crippled.
2. As more and more nodes and segments are added, the maintenance becomes difficult.
3. Scalability of the network depends on the type of cable used.



# NETWORKS FUNCTIONS

**The network does much for apps:**

- Make and break connections
  - Find a path through the network
  - Transfers information reliably
  - Transfers arbitrary length information
  - Send as fast as the network allows
  - Shares bandwidth among users
  - Secures information in transit
  - Lets many new hosts be added
- 

# NETWORKS FUNCTIONS

The network does much for apps:

- Make and break connections
- Find a path through the network
- Transfers information
- Transfers arbitrary information
- Send as fast as network allows
- Shares bandwidth among users
- Secures information in transit
- Lets many new hosts be added

Services

# NETWORKS NEED MODULARITY

The network does much for apps:

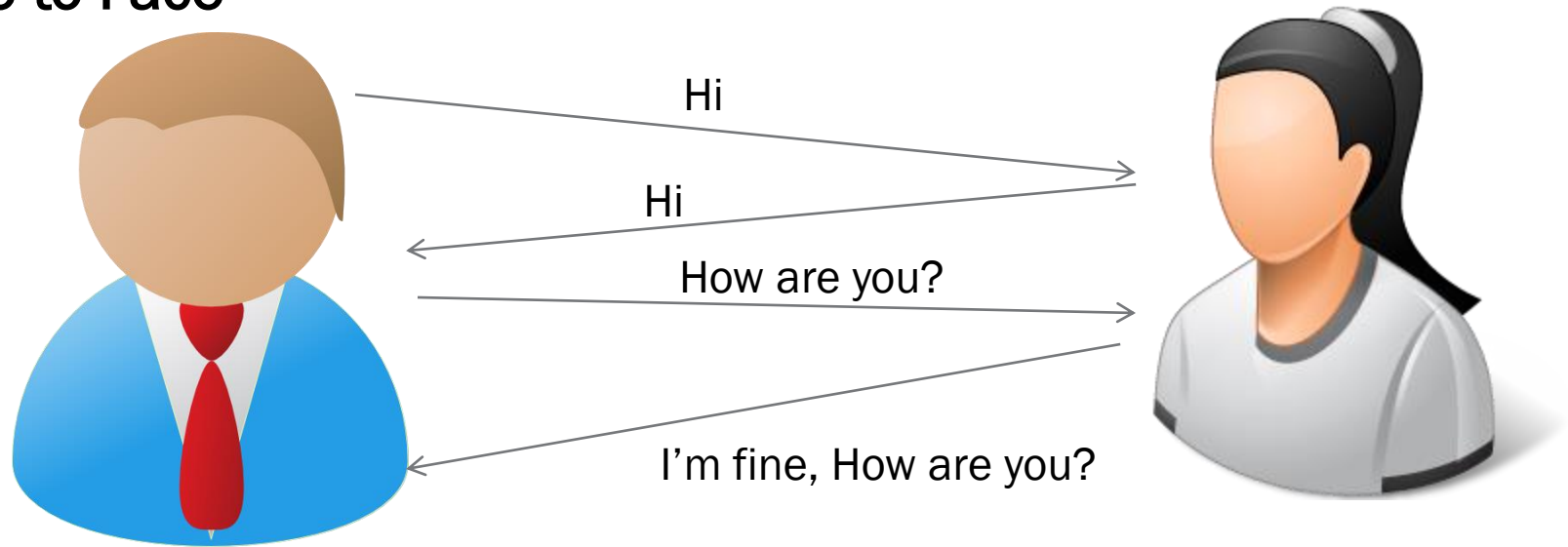
- Make and break connections
- Find a path through the network
- Transfers information
- Transfers arbitrary data
- Send as fast as possible
- Shares bandwidth
- Secures information
- Lets many new apps be built
- ...

We need a form of modularity and rules, to help manage complexity and support reuse



# PROTOCOLS

## Face-to-Face



## Radio Communication?

# PROTOCOLS

- A set of rules and formats that govern the communication between communicating parties
  - set of valid messages
  - What does a message mean
- A protocol is necessary for any function that requires cooperation between parties

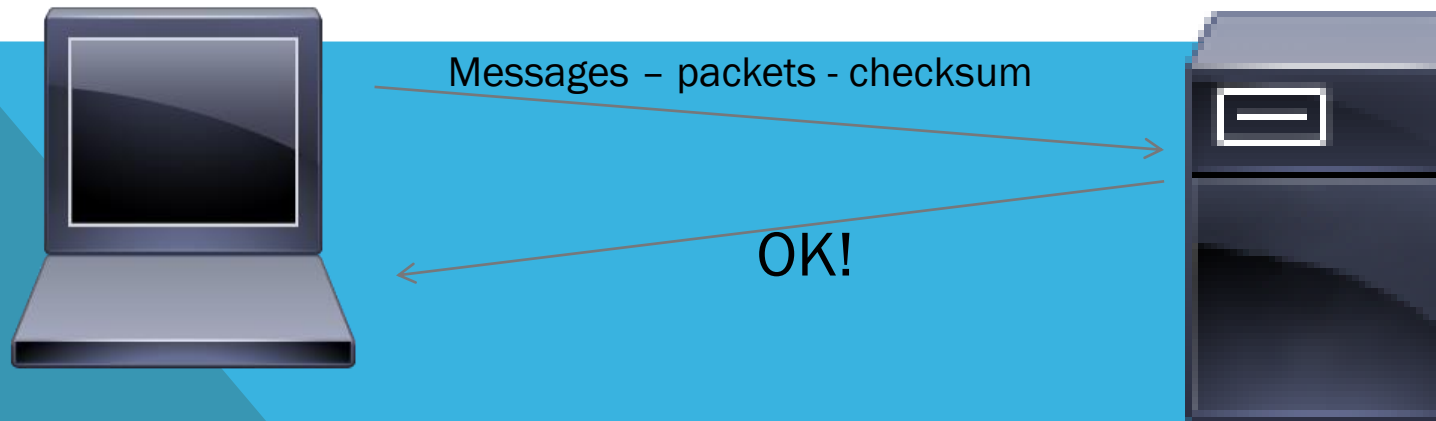


# PROTOCOLS

Example: Exchange messages over a network in the correct order and without loss or corruption.

## The Protocol

- send a message as a series of packets
- Send a checksum
- Receivers sends ok or not-OK message
- Sender waits for OK message
- If no response, resends message



# PROTOCOLS

## Syntax of a message

- What fields does it contain?
- in what format?

## Semantics of a message

- what does a message mean?
- for example, not-OK message means receiver got a corrupted message

## Actions to take on receipt of a message

- for example, on receiving not-OK message, retransmit the entire file





# PROTOCOL LAYERS

- A protocol effectively provides a service – e.g. reliable message transfer service to an application
  - Many services = many protocols!
  - Some protocols can use the services of other protocols as a step in their own execution.
    - packet transfer is one step in the execution of reliable message transfer protocol
- This dependency is called layering
  - reliable message transfer is layered above packet transfer protocol
  - like a subroutine like a subroutine



# PROTOCOL STACK

Protocols and layering is the main structuring method used to divide up network functionality

- Each instance of a protocol talks virtually to its peer using the protocol
- Each instance of a protocol uses only the services of the lower layer
  - once we define a service provided by a layer, we need know nothing more about the details of how the layer actually implements the service



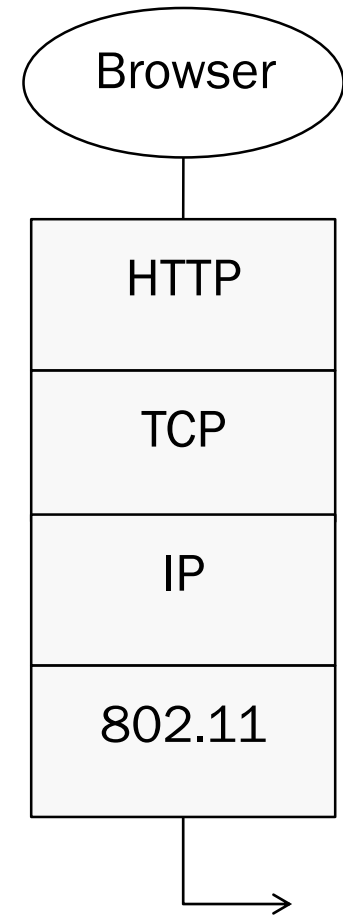
# PROTOCOLS AND LAYERS

## Protocols you've probably heard of:

- TCP, IP, 802.11, Ethernet, HTTP, SSL, DNS, ... and many more

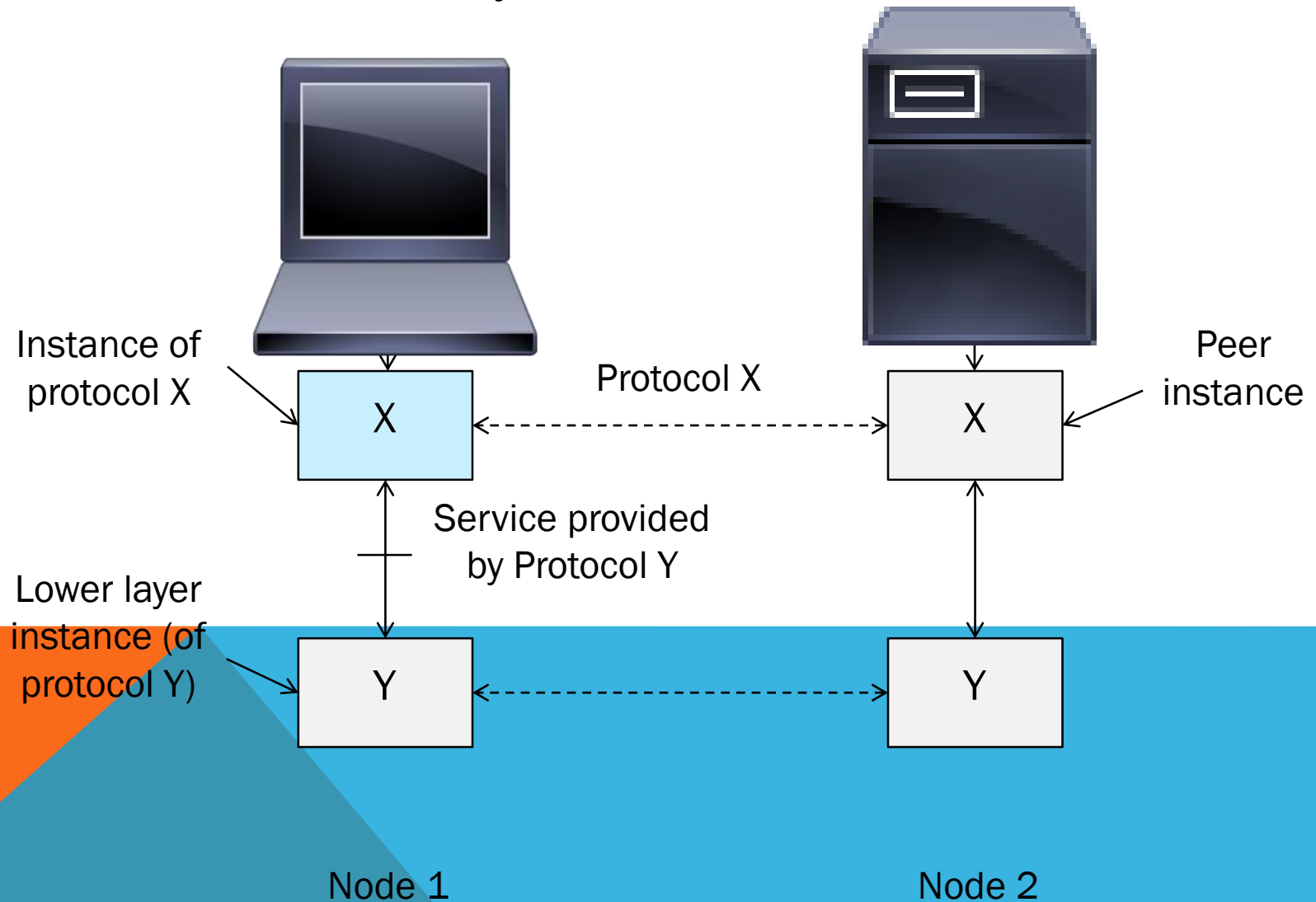
## An example protocol stack

- Used by a web browser on a host that is wirelessly connected to the Internet



# PROTOCOLS AND LAYERS

Protocols are horizontal, layers are vertical



# ENCAPSULATION

**Encapsulation is the mechanism used to effect protocol layering**

- Lower layer wraps higher layer content, adding its own information to make a new message for delivery
- Like sending a letter in an envelope; postal service doesn't look inside



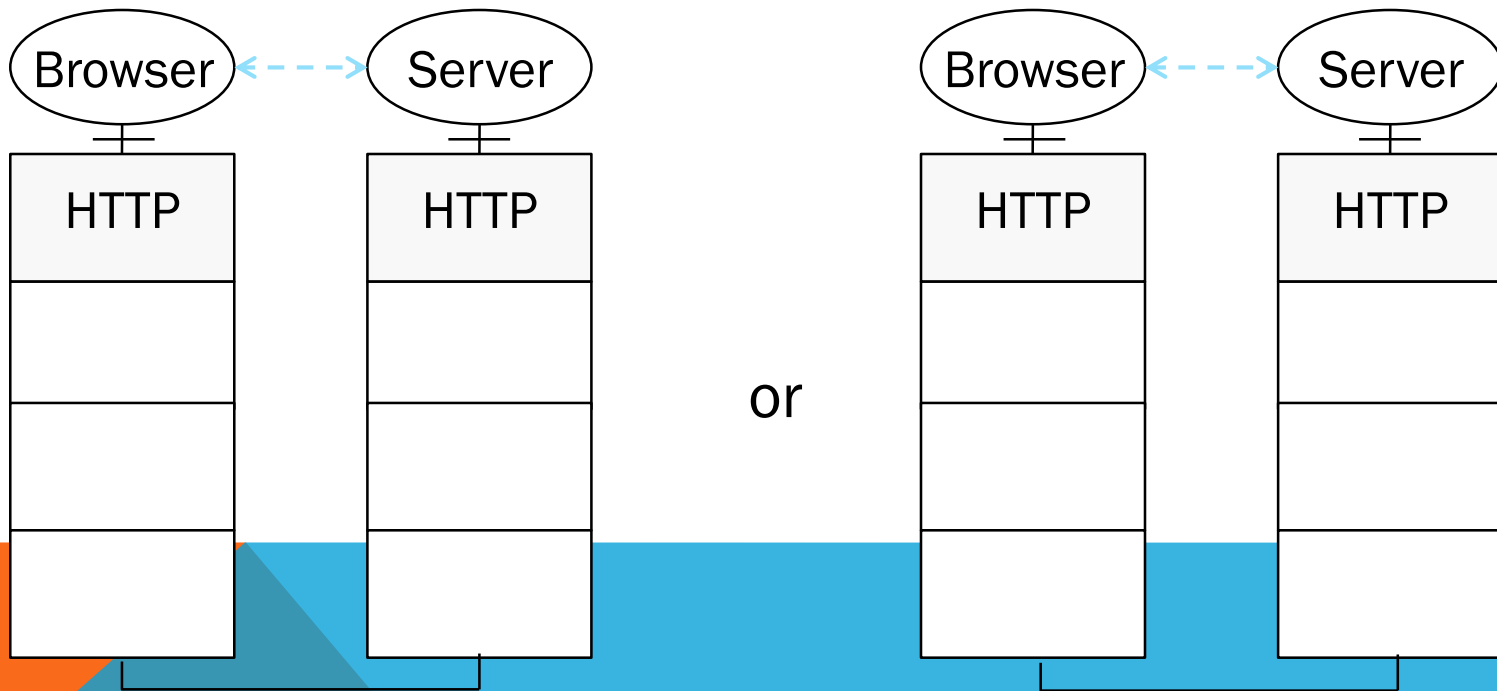
# ADVANTAGE OF LAYERING

- Breaks up a complex problem into smaller manageable pieces
  - can compose simple service to provide complex ones
  - HTTP is Java layered over TCP over IP (and uses DNS, ARP, DHCP, RIP, OSPF, BGP, PPP, ICMP)
- Abstraction of implementation details
  - separation of implementation and specification
  - can change implementation as long as service interface is maintained
- Can reuse functionality
  - upper layers can share lower layer functionality



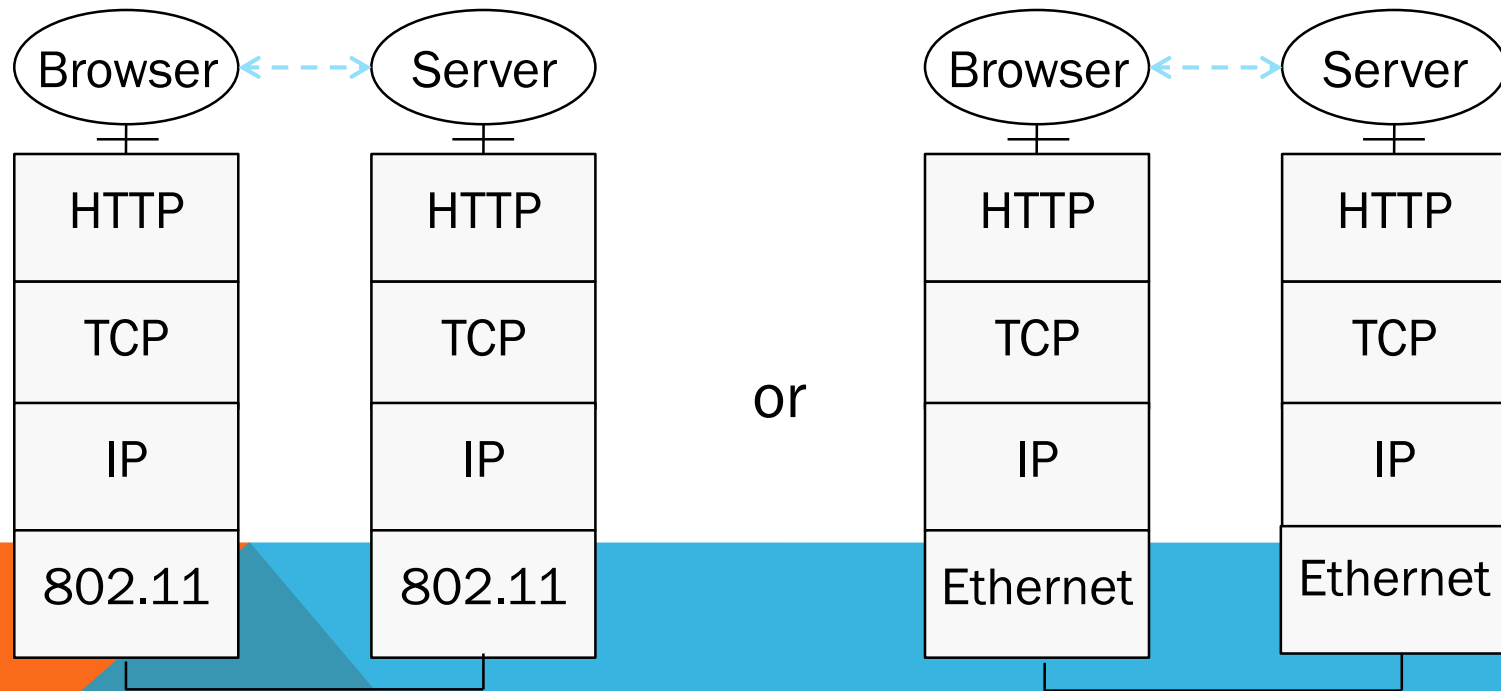
# ADVANTAGE OF LAYERING

Information hiding and reuse



# ADVANTAGE OF LAYERING (2)

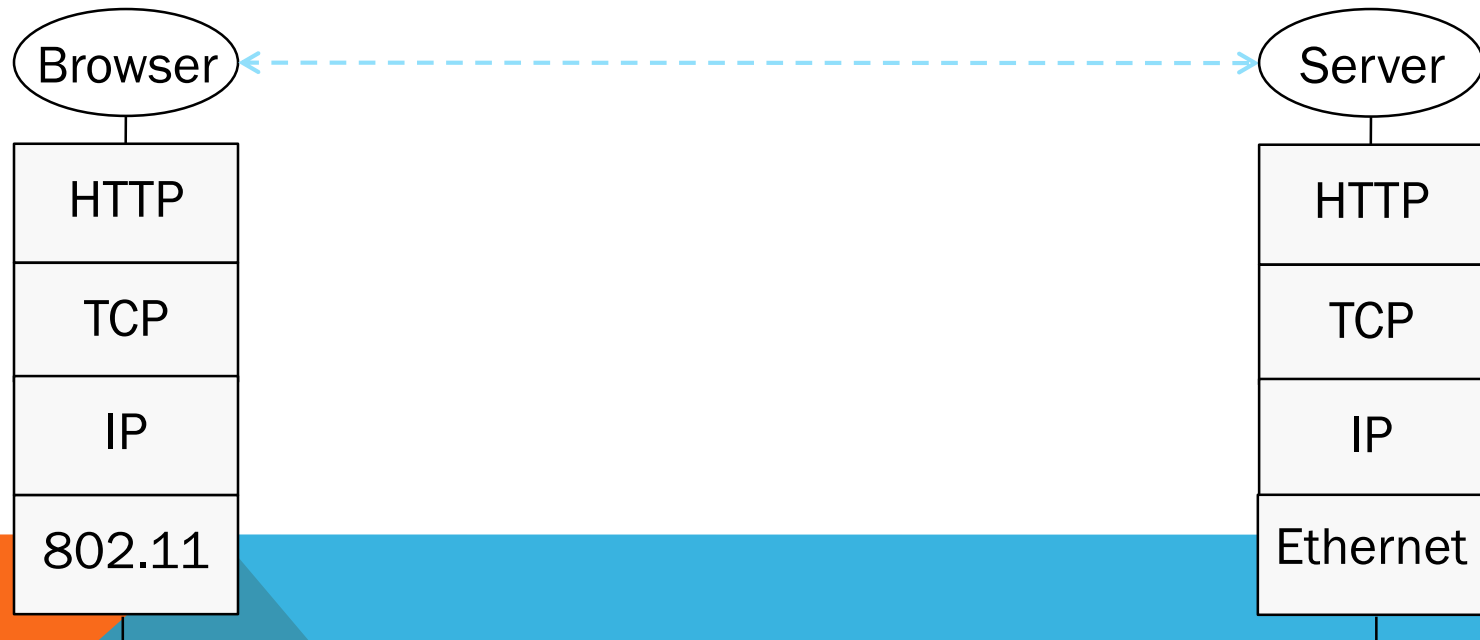
## Information hiding and reuse





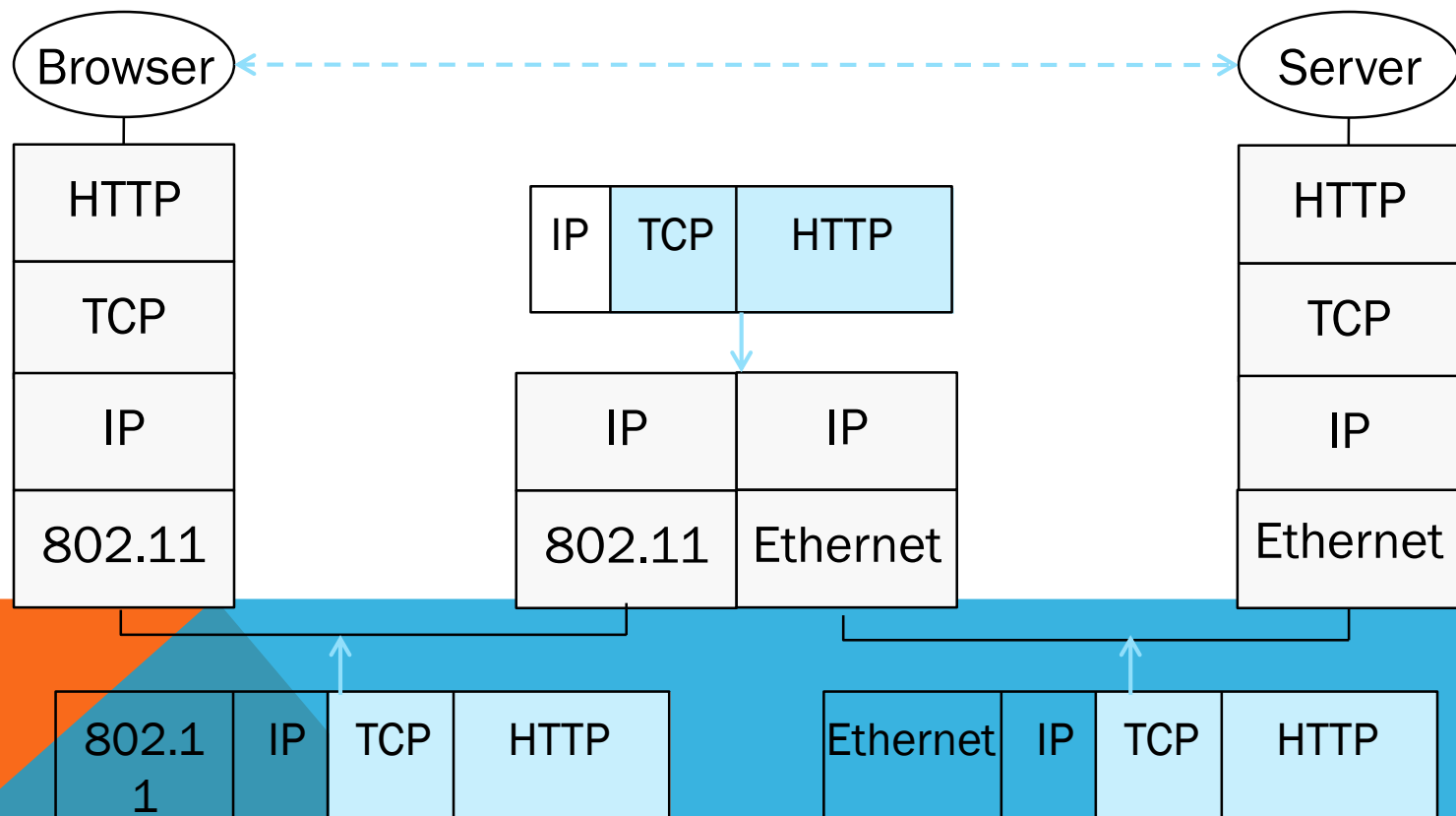
# ADVANTAGE OF LAYERING (3)

Using information hiding to connect different systems



# ADVANTAGE OF LAYERING (4)

Using information hiding to connect different systems



# DISADVANTAGE OF LAYERING

## Adds overhead

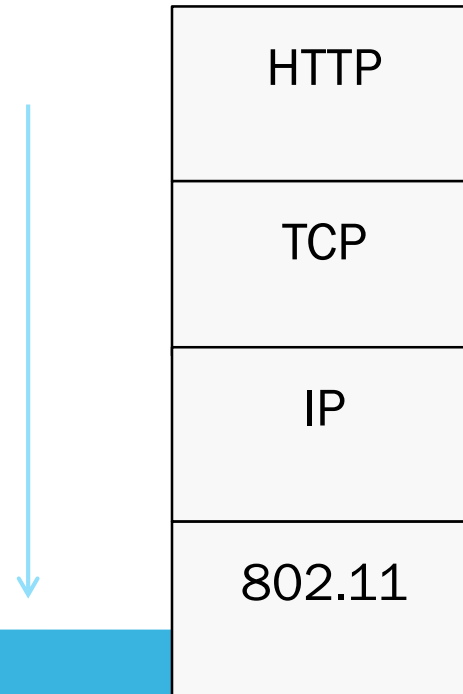
- But minor for long messages

## Hides information

- App might care whether it is running over wired or wireless!



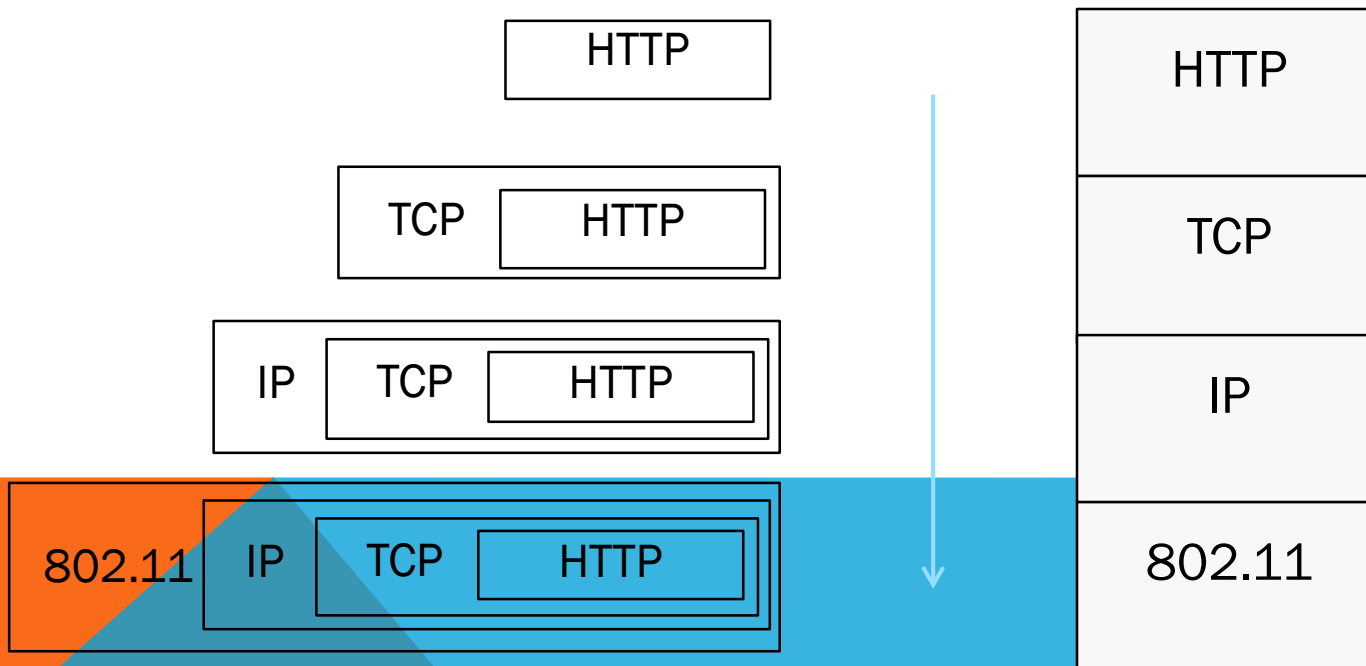
# ENCAPSULATION



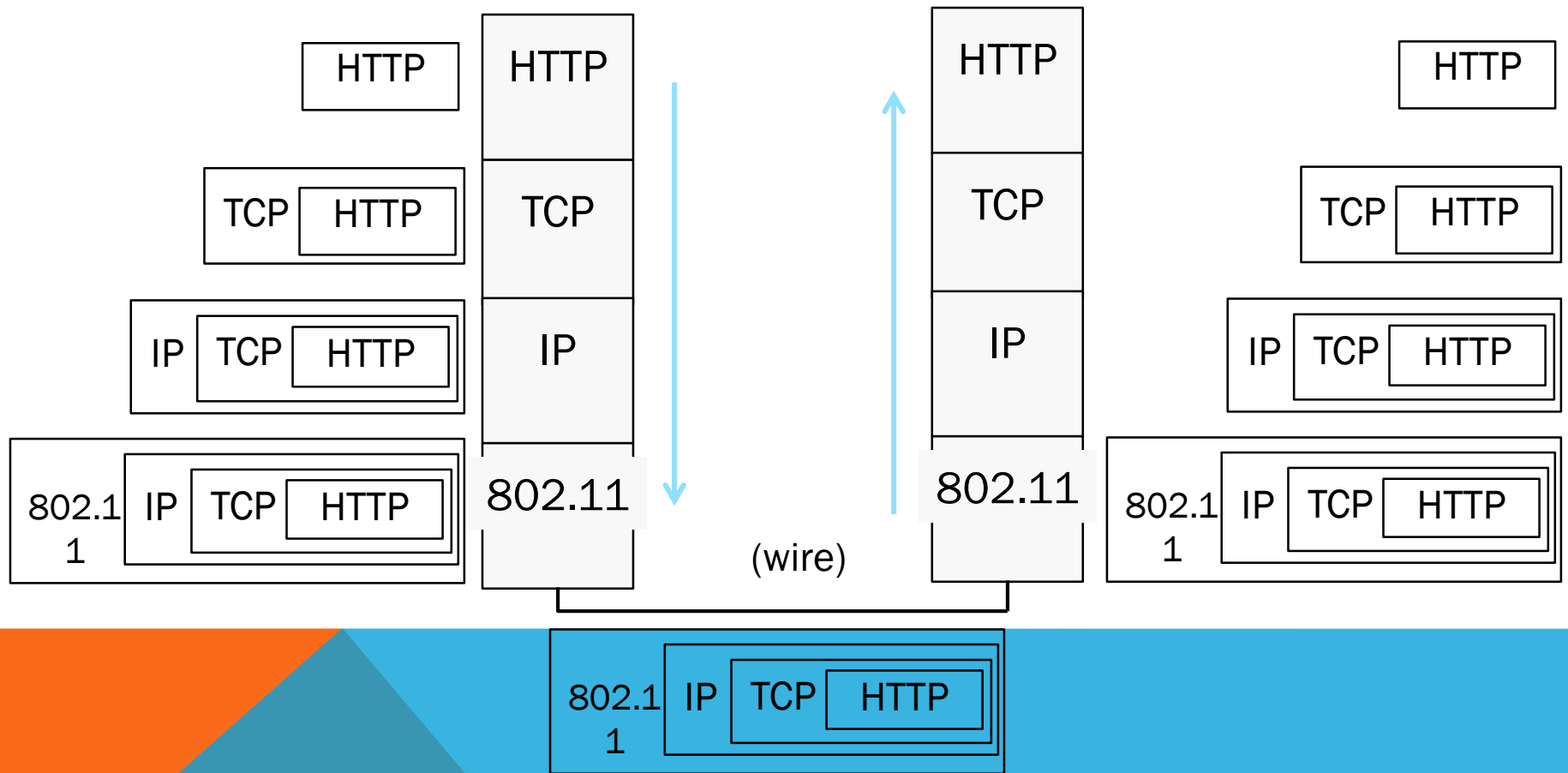
# ENCAPSULATION

Message “on the wire” begins to look like an onion

- Lower layers are outermost



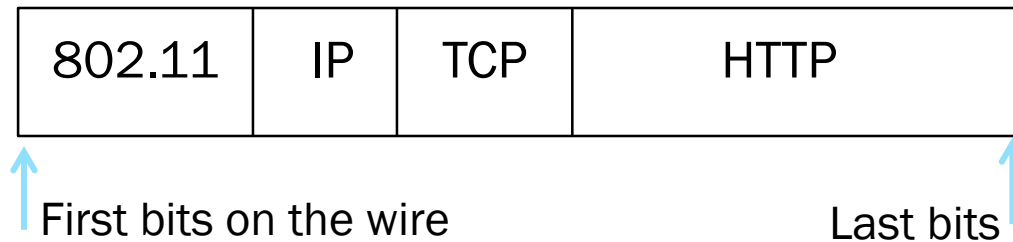
# ENCAPSULATION



# ENCAPSULATION

Normally draw message like this:

- Each layer adds its own header

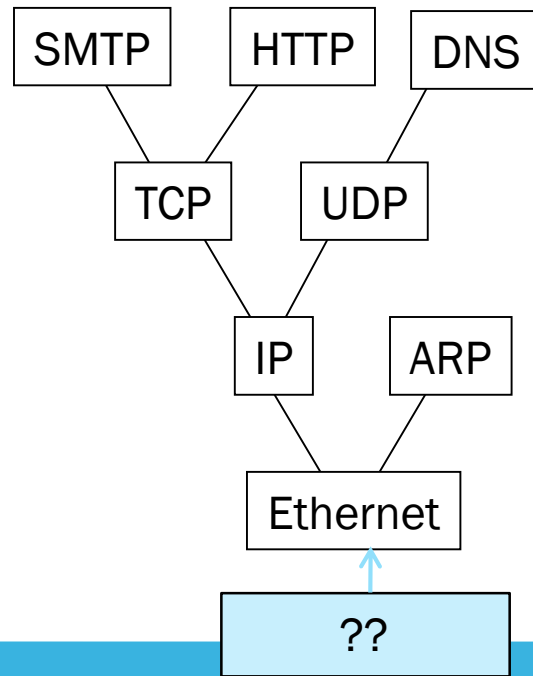


More involved in practice

- Trailers as well as headers, encrypt/compress contents
- Segmentation (divide long message) and reassembly

# DEMULTIPLEXING

Incoming message must be passed to the protocols that it uses





# DEMULTIPLEXING

Done with demultiplexing keys in the headers

