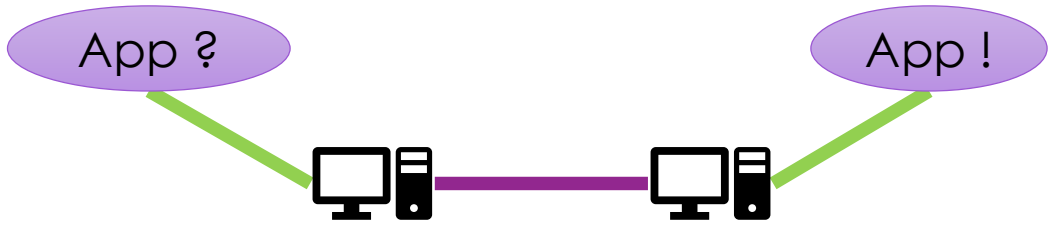
COMP3310/6331 – #9

WANs and Layers

Dr Markus Buchhorn: *markus.buchhorn@anu.edu.au*

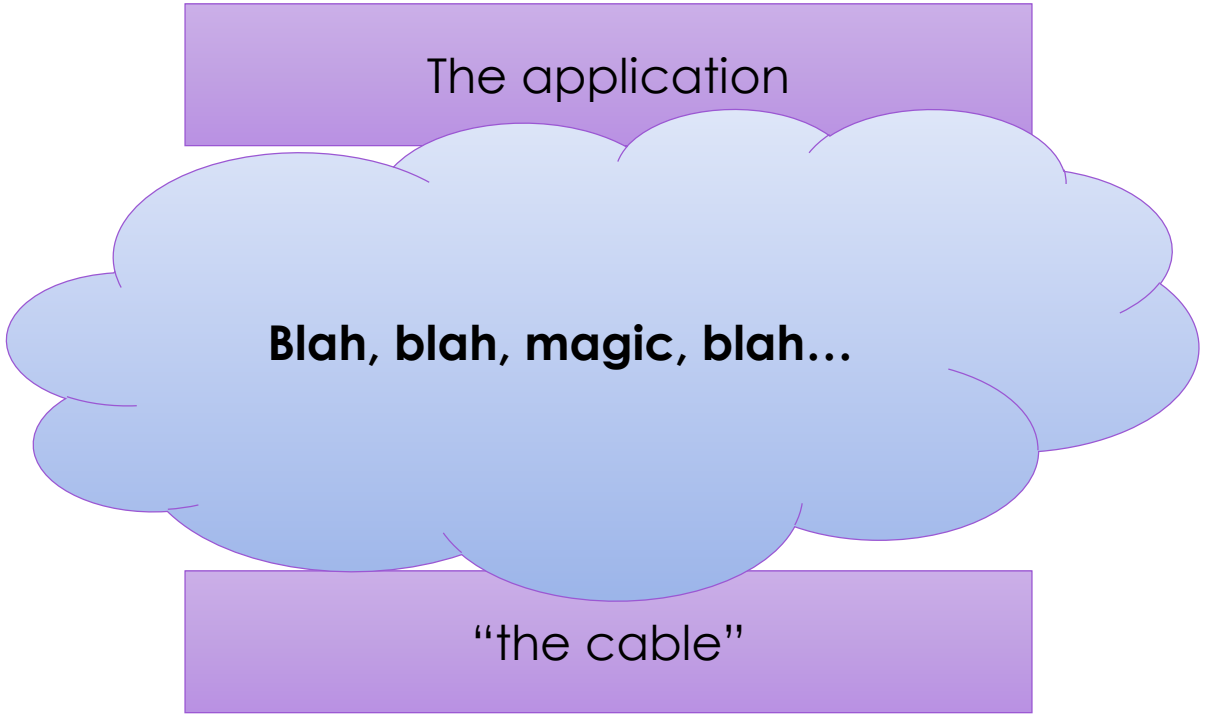
Remember this?



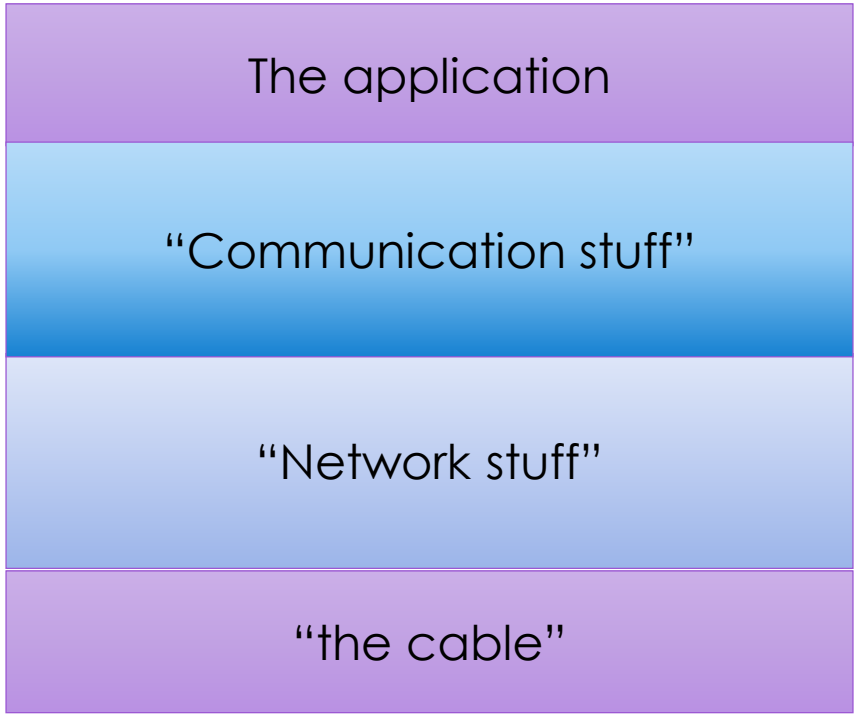
An application talks to an operating system

- which talks to the computer hardware
- which talks to a “cable”
- which talks to another computer’s hardware
- which talks to that operating system
- which talks to an application over there

Or another way



Which now becomes



And more!!

Going Wide-Area Networking

- Introducing the **Network Layers** model!
- ***Why***?
  - Because it's good for you! Lots of examinable concepts!

- No really, **why**?
- Many LANs are user-friendly, run at high speed, with large address spaces
  - Can send information efficiently from A to B

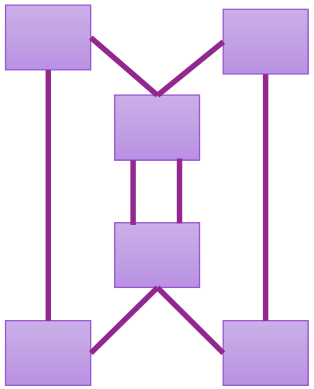
- Why not go global?

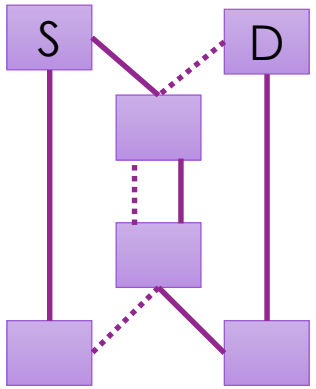
Because: Scaling

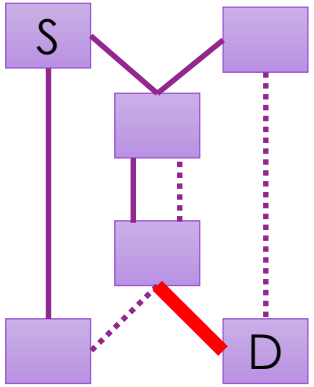
- Address tables and management don't scale globally
  - Billions of devices – and every switch needs to store them all?
- Updates take a long time to propagate
  - Long delays, depending on the paths
  - Topology changes happen a lot
- Broadcasts have to be sent globally
  - E.g. whenever a new device connects
  - Spanning Tree won't converge in time

Because: Traffic control

- LANs organise themselves for simplicity – not optimisation
- Spanning Tree doesn't guarantee the optimal topology
  - Sometimes people do know better
- Network traffic costs money, needs to consider politics







Because: Which LAN?

- Many LAN choices
  - 802.3 Ethernet, 802.11 WiFi, xDSL, 4G, … each fit-for-purpose (wired/wireless)

- Different LANs don't mix. Mismatched behaviours with different
  - Address schemes
  - Service models (frames, cells, circuits)
  - Security models
  - Frame sizes
  - Performance
  - Prioritisation mechanisms

- Don't want to write applications tuned to different LAN types
- Don't want to buy boxes that translate between every possible combination
  - (Many, expensive) single points of failure

Solving these

- Want to communicate across networks: aka Inter-network
- Take the **LAN** to the **WAN**

- Scaling problems
  - *Use a hierarchy of connections, addresses and aggregate/group*
- Traffic control
  - *Optimise routing with more information, and support prioritisation*
- Which LAN?
  - *All of them. Put a common **layer** across the top*

Fundamental theorem of software engineering

- *"We can solve any problem by introducing an extra ~~level of indirection~~ layer"*
- *"…except the problem of too many ~~levels of indirection~~ layers…"*
  - Attributed to David J Wheeler et al.

Just one layer?

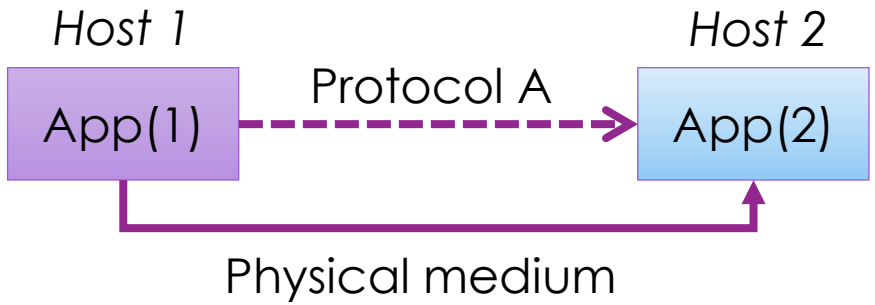
- Networks have complex requirements
  - Find a path
  - Make connections
  - Transfer communications (bits) reliably
  - Ensure security
  - Share bandwidth, globally and diverse paths
  - Allow for hosts to come/go
  - Deal with topology changes
  - …
- Networks don't really care what you're using it for!

Why stop at networks?

- Applications need functionality too
  - Find/advertise resources
  - Connect to other machines
    - One or many
  - Exchange application-specific messages
  - Adapt to capabilities
    - Devices, software, …
  - Maintain state of a connection
  - Expect sufficient reliability
  - Expect trustworthiness
  - Maximise performance, minimise delays
- Simplify: let's modularise/layer things…

Layers upon layers

- Use **layers** to divide (allocate) functionality
- Use **protocols** to exchange information within a layer
- User **services** from lower layers to build on

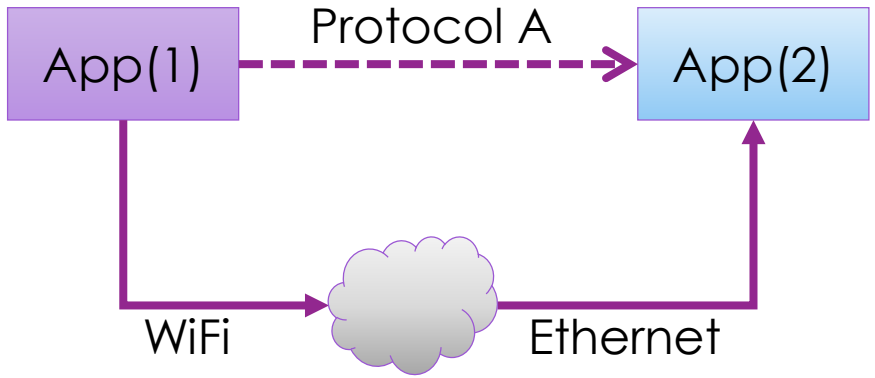


e.g.
Copper medium and
Ethernet protocol

Simple!

Layers upon layers

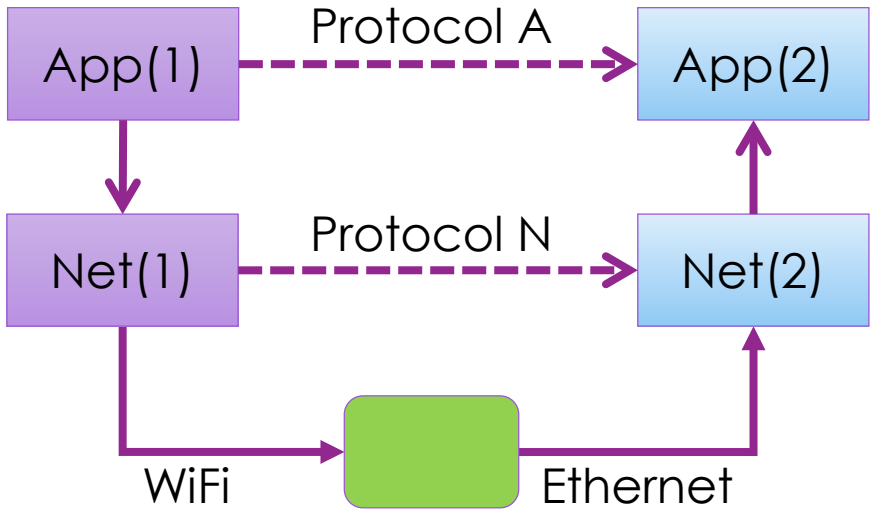
- Use **layers** to divide (allocate) functionality
- Use **protocols** to exchange information within a layer
- User **services** from lower layers to build on



Protocol is not Ethernet nor Wifi?

Layers upon layers

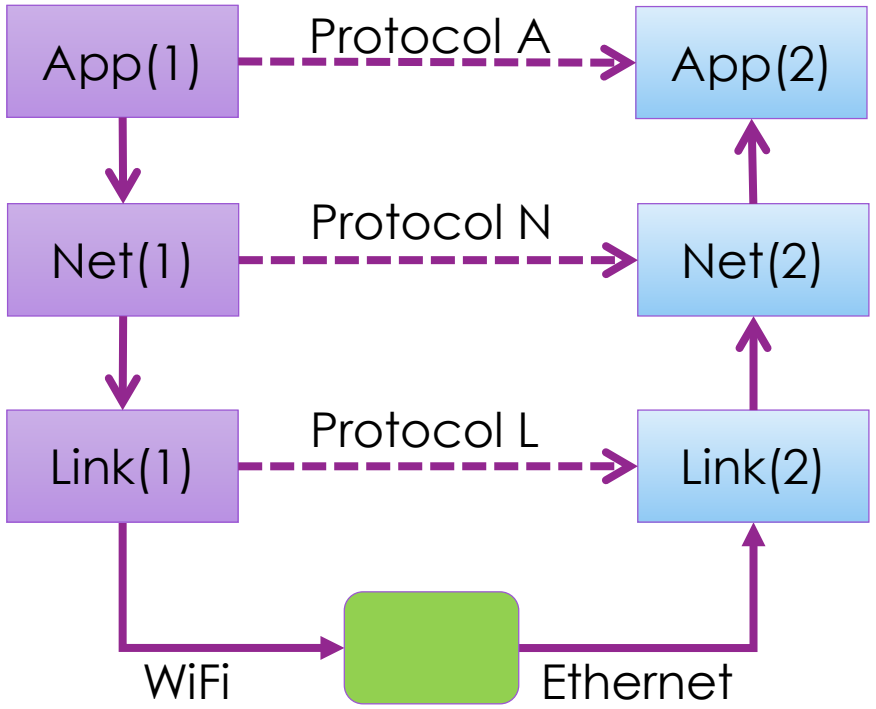
- Use **layers** to divide (allocate) functionality
- Use **protocols** to exchange information within a layer
- User **services** from lower layers to build on



Applications offload the networking details

Network layer provides a service, takes care of **everything?**

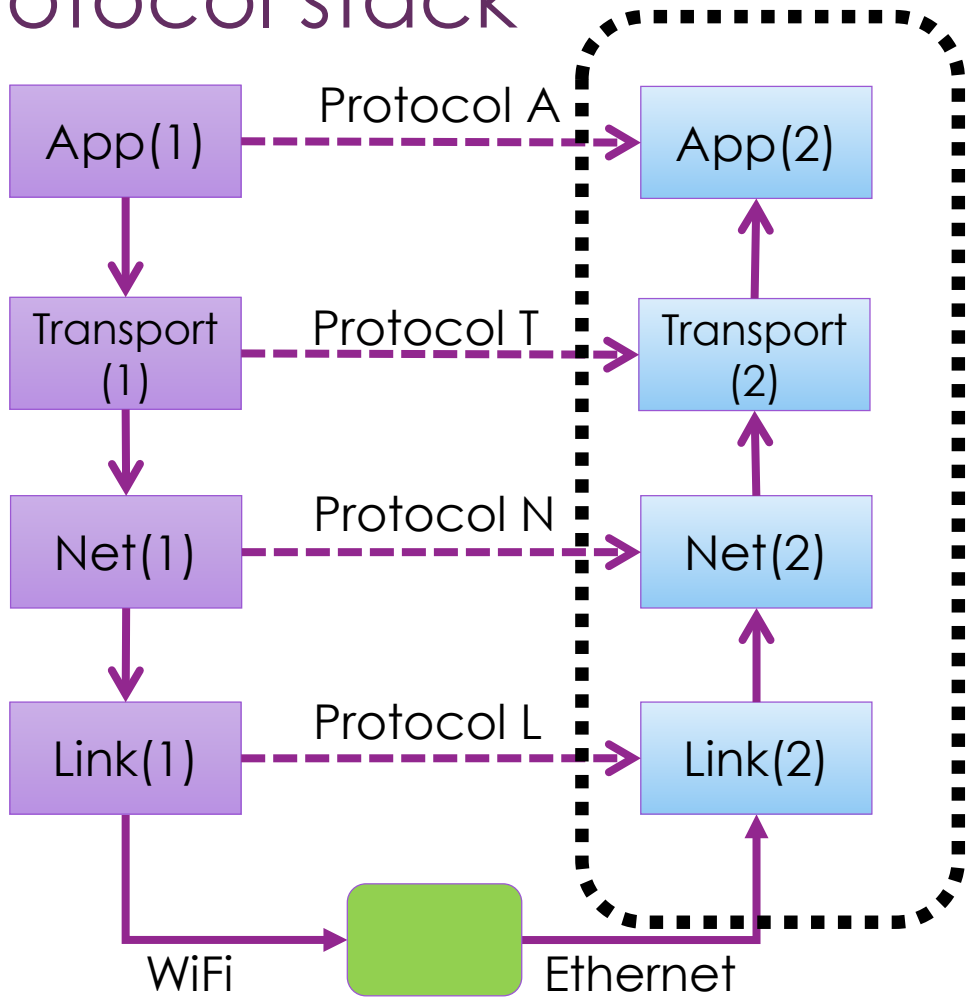
Layers upon layers



Network layer:
transmission, topology, …

Offload the Link details

A "protocol stack"

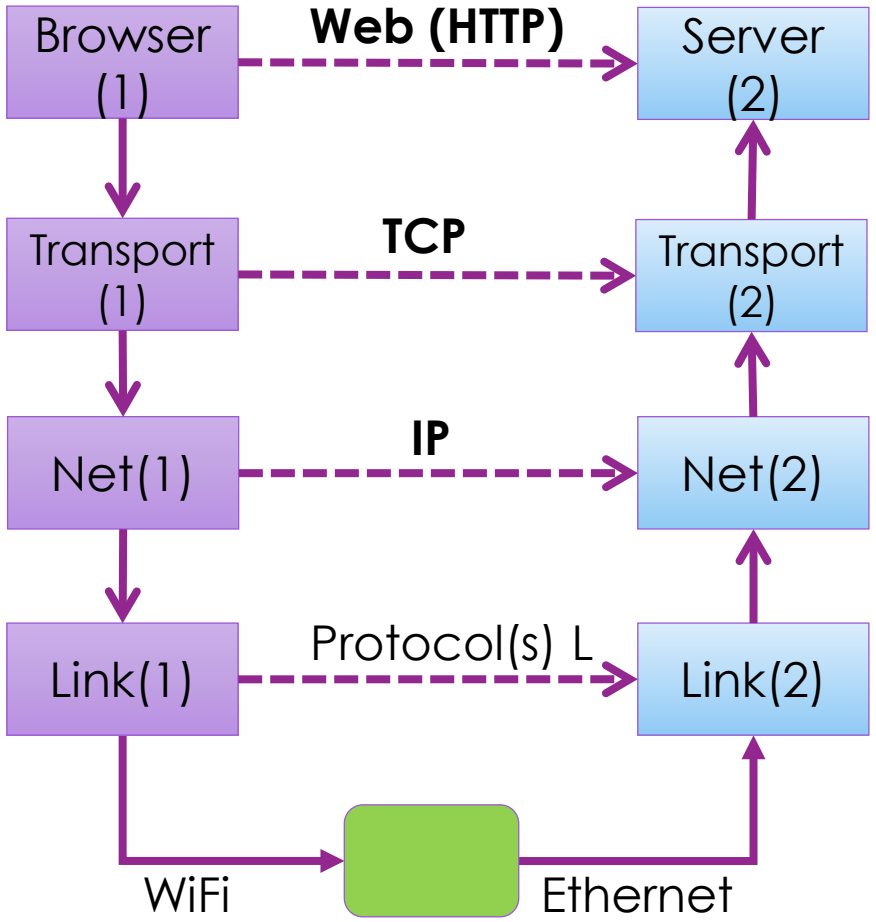


Offload more: reliability, app-muxing, security, performance, …

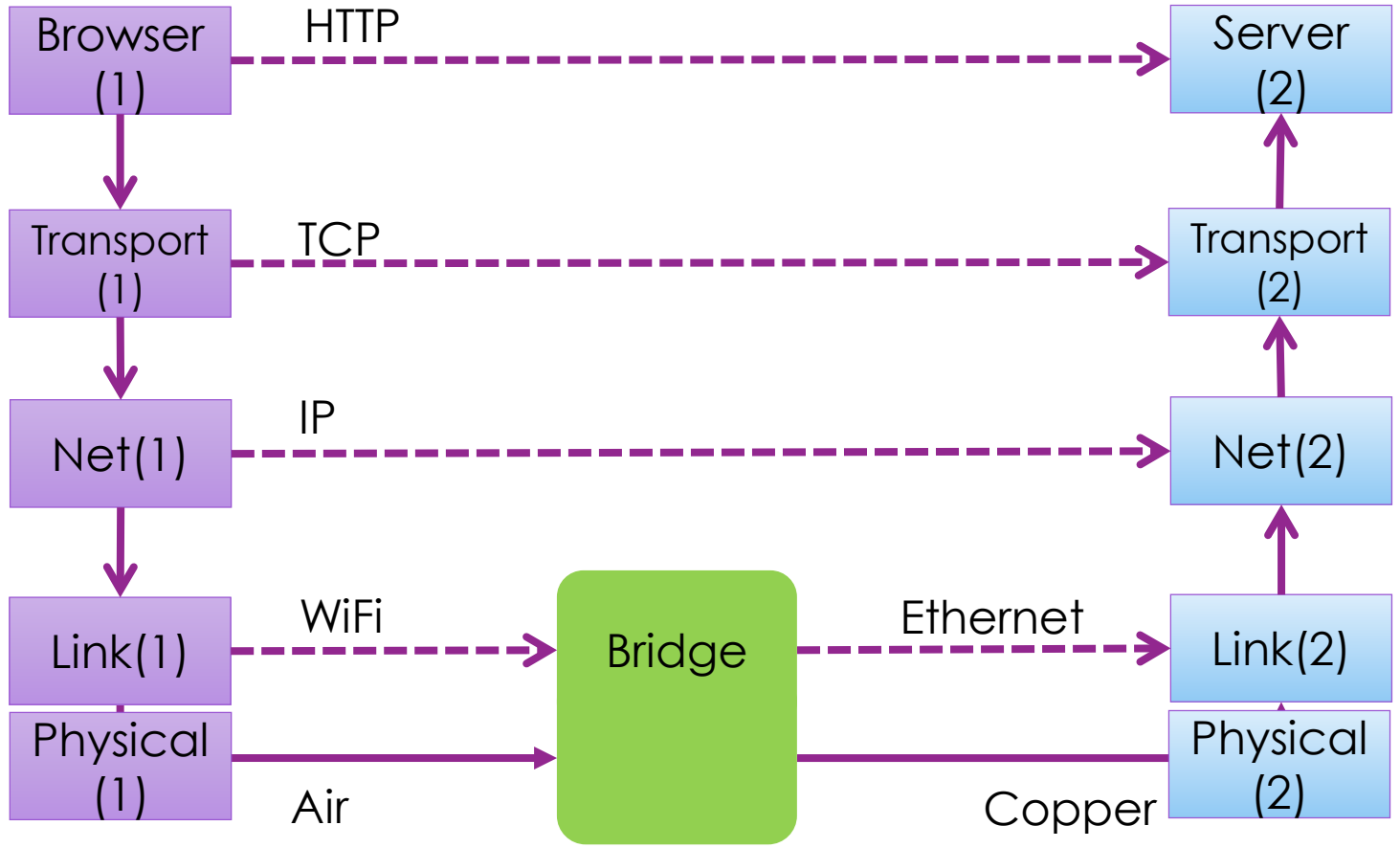
Each protocol exchanges Protocol Data Units (PDUs)

Each layer provides services through an API, exchanging Service Data Units (SDUs)

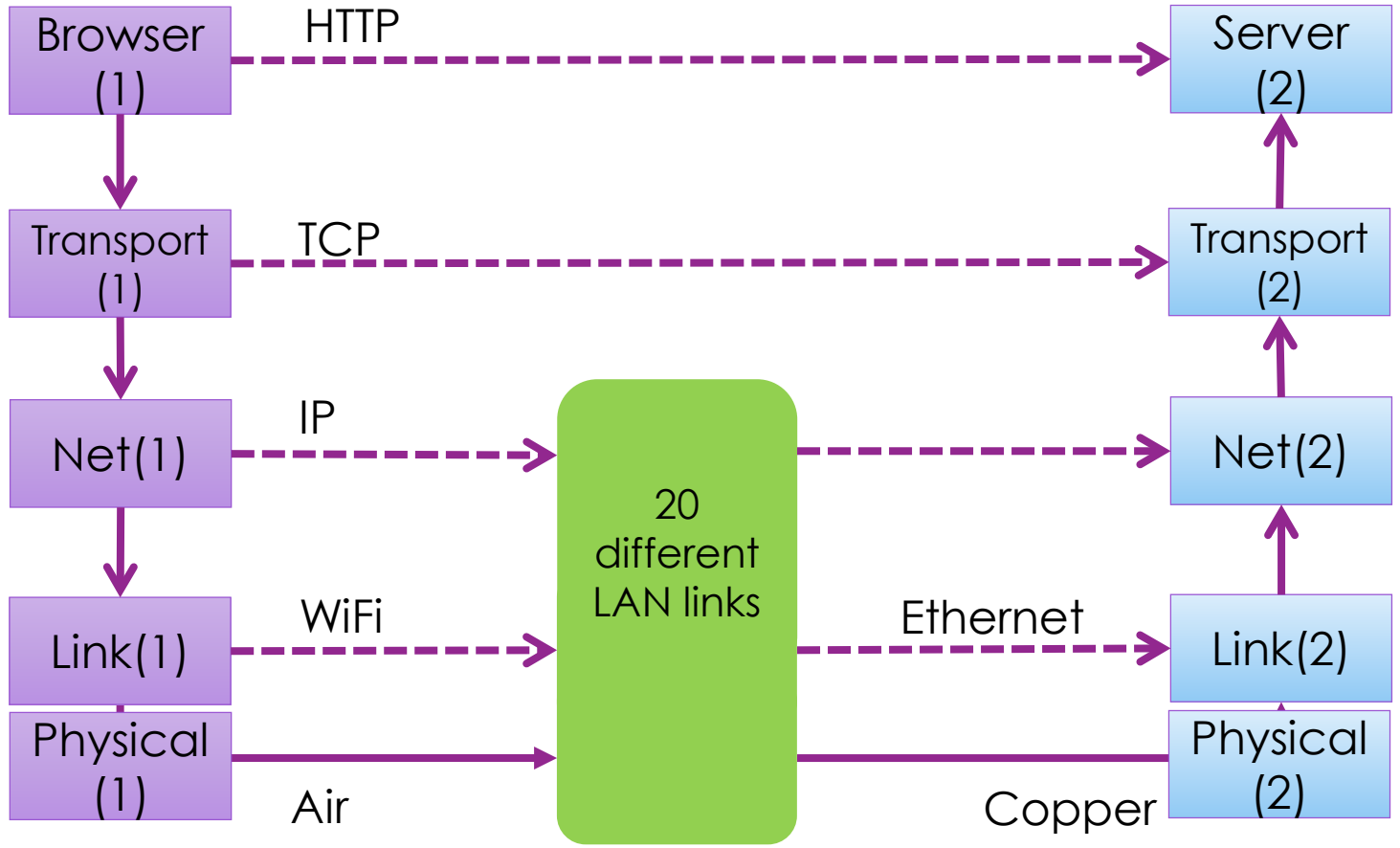
For example…


Browser (1)
Web (HTTP)
Server (2)
Transport (1)
TCP
Transport (2)
Net(1)
IP
Net(2)
Link(1)
Protocol(s) L
Link(2)
WiFi
Ethernet

Hiding **link** complexity



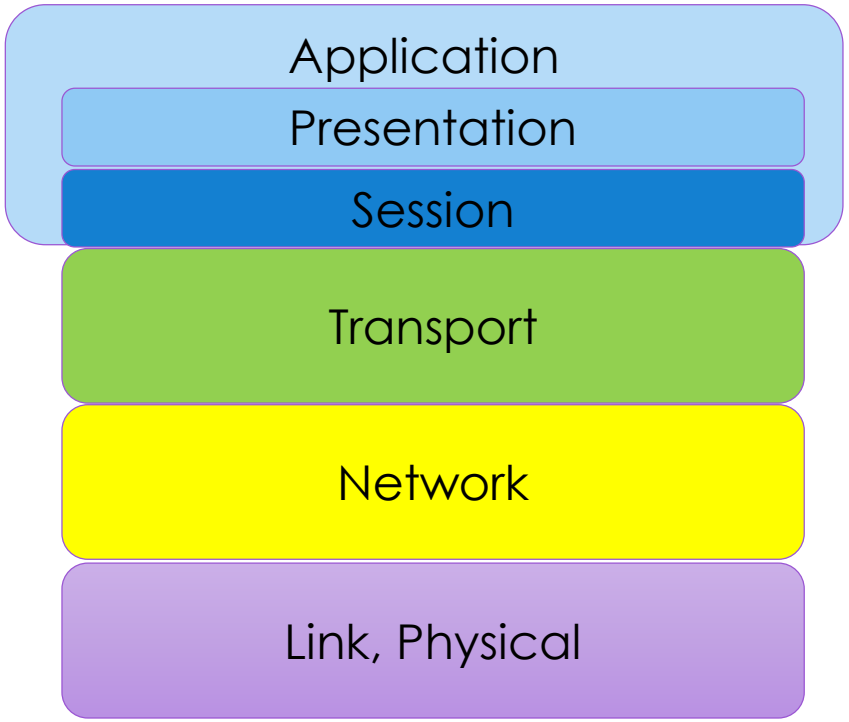
Hiding **path** complexity



Applications don't know nor care. Unless there is a performance question.

Role of each layer

- Each consumes services (functionality) from the layer below
- Each offers services (functionality) to layer above
- Which functions belong in each??



Reference Protocol Stacks

An OSI *(ISO+ITU-T/CCITT)* reference – great standard – almost never used

Layer	Name	Function
7	Application	Deliver functionality
6	Presentation	Convert information for application needs
5	Session	Combine diverse communications, maintain state
4	Transport	Ensure end-to-end performance
3	Network	Send packets over multiple links
2	Link	Transmit Frames
1	Physical	Modulation and encoding of bits

In reality

- There are protocols that span layers
  - For internetworking, some knowledge has to move up and down layers
- There are layers that have sublayers
  - E.g. Ethernet has MAC and LLC
    - *Logical Link Control; adds payload-muxing, flow-control and extra error-handling*
- There are people who think about this classification too much
  - It's not a rule
- But as a concept, and to guide protocol design – it's very useful
  - Think about what functionality you need/offer, and where it should be
  - *Success: Internet end-to-end principle – smart edges, dumb core (rules, not state)*

Internet "Reference" protocol Stack

Layer	Name	Function
7	Application	Deliver functionality (and the presentation/session)
4	Transport	Ensure end-to-end performance
3	Network	Send packets over multiple links
1,2	Physical, Link	Transmit Frames

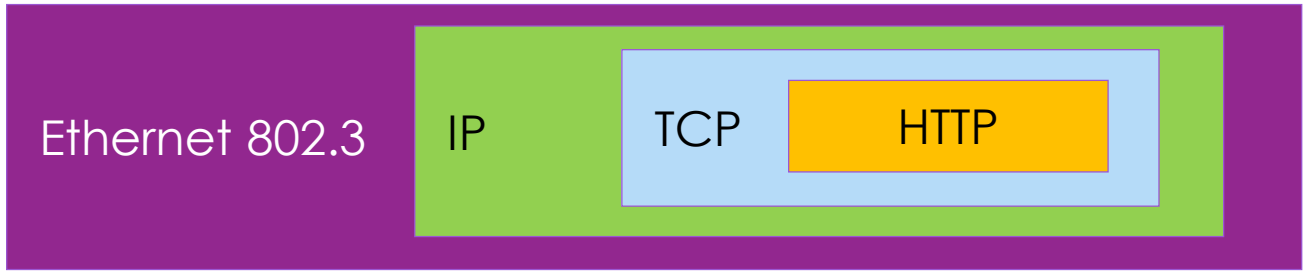
Layering considered harmful… (rfc3439)

Naming

Layer	What it transports (Protocol Data Unit)	How they connect
Application	Messages/Data	Proxy, gateway
Transport	Segments/Datagrams	
Network	Packets (!!)	Router
Link	Frames (cells, circuits)	Switch, Bridge
Physical	Bits	Hub (repeater)

Real world implementation

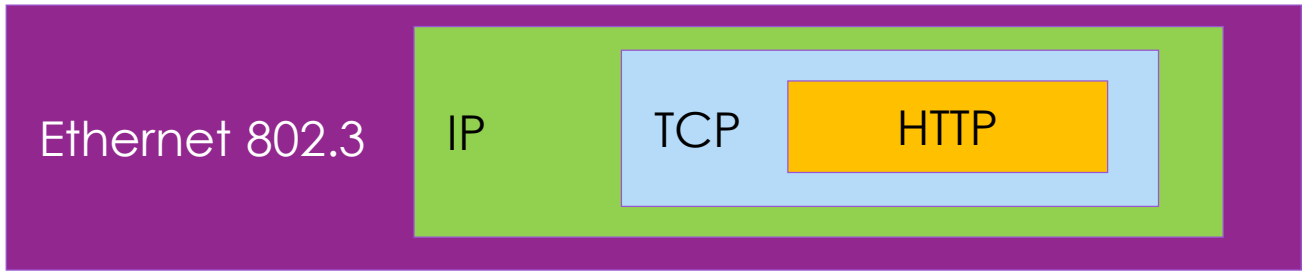
- Every protocol has a dictionary for what it sends
  - Every layer has a 'payload' to transmit
- Every payload is passed to a lower layer and is encapsulated
  - Think of a letter in an envelope
  - Add headers (and trailers), maybe encrypt, compress, segment
  - Repeat till it's sent



Preamble	Start of Frame	MAC dest	MAC src	802.1Q tag [opt]	Type / Length	Payload	Checksum
7 byte	1 byte	6 byte	6 byte	4 byte	2 byte	42-1500 byte	4 byte

“Demuxing” the “encapsulated”

- Host/Receiver ultimately gets the (whole) message
- Need to pass it to the process that needs it
  - Which one???



- Every layer has a ‘key’ <u>about</u> its payload (in its header)
  - **Ethernet** has an address, and Ether-type
  - **IP** has an address and Transport-type (tcp vs udp)
  - **TCP** has a port number
  - **Applications** have message keys (e.g. http url’s)

Meanwhile, out on the network

- Many devices watch traffic
  - To learn, to report, to manage
  - How does it scale?

- Ideally don't hold up the traffic
  - Minimal packet inspection
  - Only read the layer (headers) they are responsible for forwarding
    - Link, network

- Sometimes we need more, e.g. security, priority
  - Deep Packet Inspection, in real-time
    - Right down to the inner payload
    - Significant load, delays

Security, Priority

- Having DPI is considered harmful
- Lack of DPI can be considered negligent

- Good design works out how much to unpack, and when

- Firewalls and DMZs and VLANs and VPNs and …

- We'll come back to this a few times

Actual Protocol Stack?

Layer	Name	Function
10	**Money**	**Actually decides what happens**
9	**Religion**	**Arguing for the sake of it**
8	**Politics**	**Stomps on what you'd like to do**
7	Application	Deliver functionality
6	Presentation	Convert information for application needs
5	Session	Combine diverse communications, maintain state
4	Transport	Ensure end-to-end performance
3	Network	Send packets over multiple links
2	Link	Transmit Frames
1	Physical	Modulation and encoding of bits