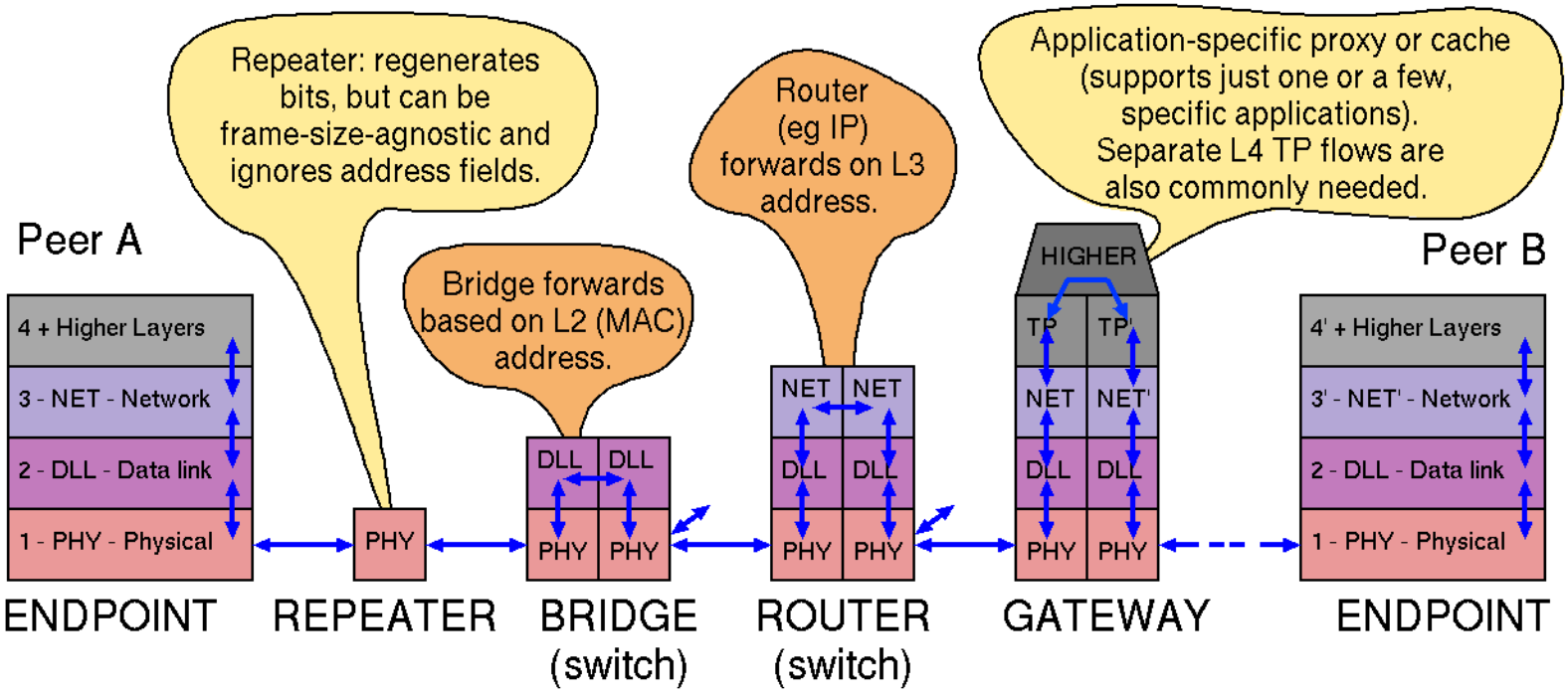
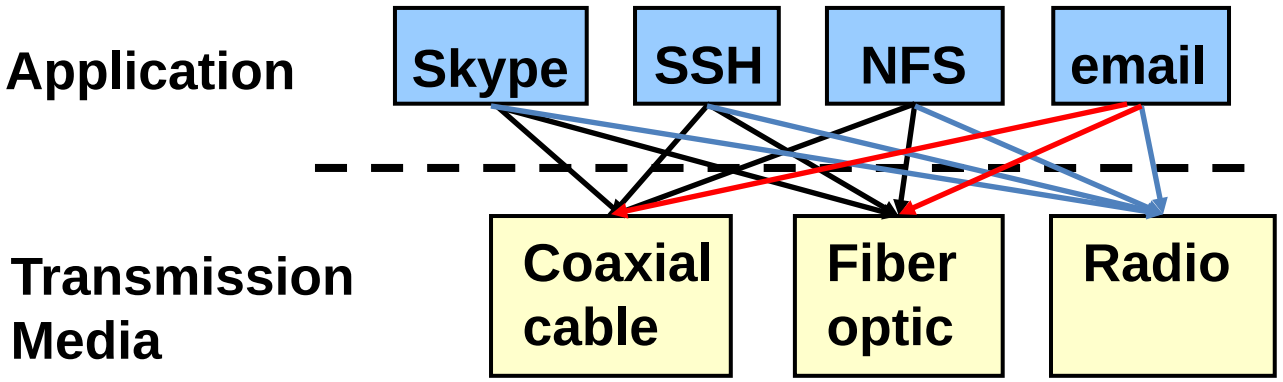
Topic 2 – Architecture and Philosophy



Topic 2 – Architecture and Philosophy

- Abstraction
- Layering
- Layers and Communications
- Entities and Peers
- What is a protocol?
- Protocol Standardization
- The architects' process
  - How to break system into modules
  - Where modules are implemented
  - Where is state stored
- Internet Philosophy and Tensions

A Multitude of Apps Problem

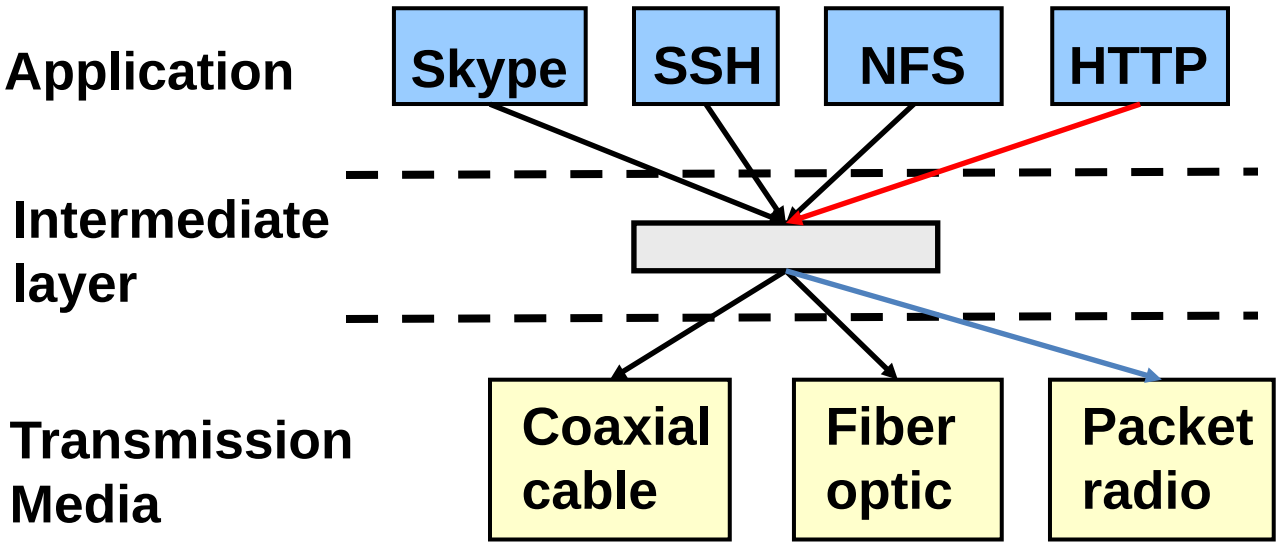


- Re-implement every application for every technology?
- No – ridiculous quadratic cost!

But how does layered design avoid this?

Solution: Intermediate Layers

- Introduce intermediate layers that provide set of abstractions for various network functionality and technologies
  - A new app/media implemented only once
  - Variation on “add another level of indirection”



Abstraction Concept

A mechanism for breaking down a problem

What not *how:* Specification *versus* implementation
(eg OO-language class definition).

Allows replacement of a component implementation without needing other parts to be changed: a consistent *interface*.

Vertical versus *Horizontal*

Network abstractions tend to be stacked vertically in layers.

The communications paths run horizontally throughout the system.

Computer System Modularity

Partition system into modules using abstractions.

- Aims:
  - ***Hide*** internal implementation, which can be freely replaced or improved.
  - **Extend** external system functionality by adding new modules that use the existing interfaces.
- Well-defined interfaces facilitate these two operations.
- Examples:
  - a library module encapsulating set of functionality, such as alternative image compression algorithms;
  - a programming language + compiler abstracts details of a particular CPU, such as the number of registers.

Computer System Modularity (cnt'd)

- Modular design with fixed interfaces has **many benefits**:
  - Modular reuse saves engineering costs
  - Revision control
  - Contractual boundary
  - Formal specification of the interface
  - Ease of documentation, understanding, etc ...

- **But can impair/restrict performance!**
  - An example from graphics:
    - the display adaptor can now shade ovals using hardware acceleration, but
    - The API (interface) only supports dots, lines and blocks.
    - The software above the interface must still manually shade the ovals, one pixel at a time.

Network System Modularity

Like software modularity, but:

- Implementation is distributed across many machines (switches, routers and hosts)
- Must decide:
  - How to abstract the system into modules:
    - Typically **Vertical protocol layering**
  - Where functionality is implemented
    - Mostly at the connection endpoints (**end-to-end** principle) or
    - Repeated at many intermediate points (if unavoidable) ‘stateless’
  - Where state is stored
    - **Stateless,** or
    - **Fate-sharing**
  - eg. NFS and some optimistic concurrency said to be ‘stateless’.

Vertical Layering Approach

- This is the primary approach
- A restricted form of abstraction:
  - system functions are divided into layers, one built upon another
- Often called a *stack*, but **not** a data structure!

Let's talk about protocol stacks ...

Layers in Comms Protocols

- Interaction only between adjacent layers
- *layer n* uses services provided by *layer n-1*
- *layer n* provides service to *layer n+1*
- Bottom layer (L1 - PHY) is physical media:
  - Radio or near-field induction
  - Light: fibre optic cables, laser beams, infra-red TV remote control …
  - Wires (coax, twisted pair)
  - Other: sound, quantum entanglement …
- Top layer (7? APP) is user application: eg web browser.

Entities

Entities usually do something useful

- Encryption – Routing – Error correction – Reliable Delivery
- Nothing at all is also reasonable

Examples of entities in the middle

- IP Router or Ethernet switch
- Mobile Phone Cell Tower
- Protocol layer - HTTP cache or proxy
- Person translating French to English

Examples of entities at the end

- Protocol layer providing encryption (end-to-end)
- HTTP end-point
- Person speaking French

[Not all communication is end-to-end, especially O&M (operations and management/maintenance).]

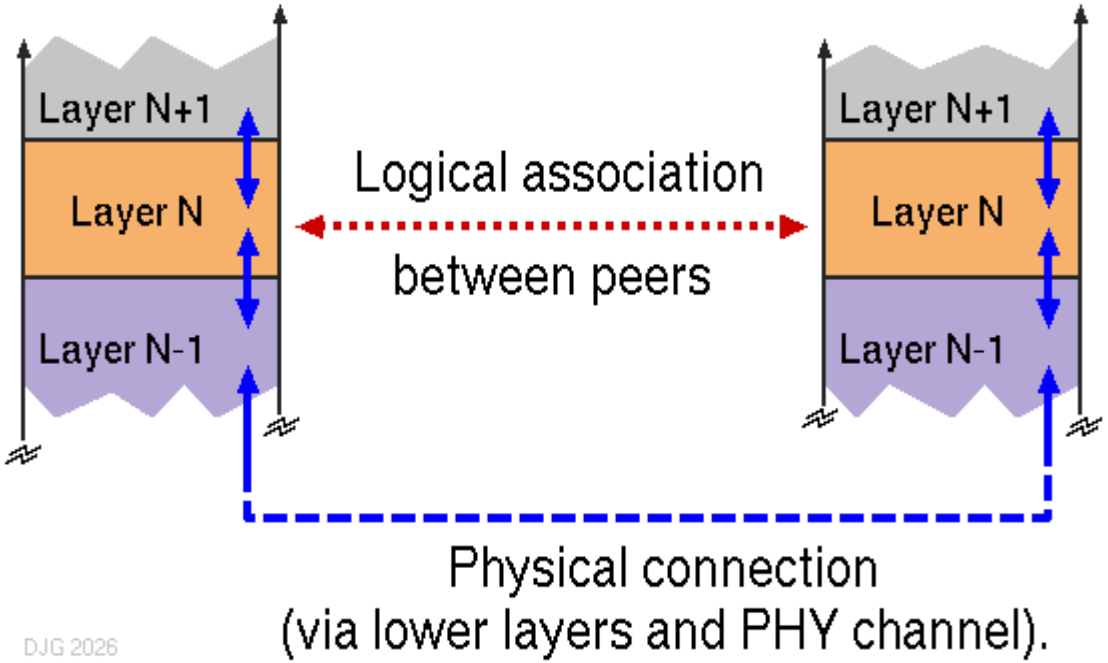
Entities and Peers

Entity – a *thing* (an independent existence).

Entities physically *interact* with the entities in layers above and below (eg. subroutine call, likely in-lined).

Entities *communicate* with *associated peer* entities

- same level but different place (eg. different box or remote host machine).



Communication between logical peers is provided by entities in the lower layers.

Interfaces and Protocols

Often we might casually use these two terms interchangeably.

An interface can be the point of connection between:

1) two vertically adjacent protocol layers, or

2) two physical components with a horizontal link between them (a link).

An interface has a defined set of primitive operations, such as send(), stop(), sleep(), read(), write(), reset(), change wire to 3.3 volts, …

A protocol is the set of allowable orderings that these primitives can be invoked in and the associated meaning.

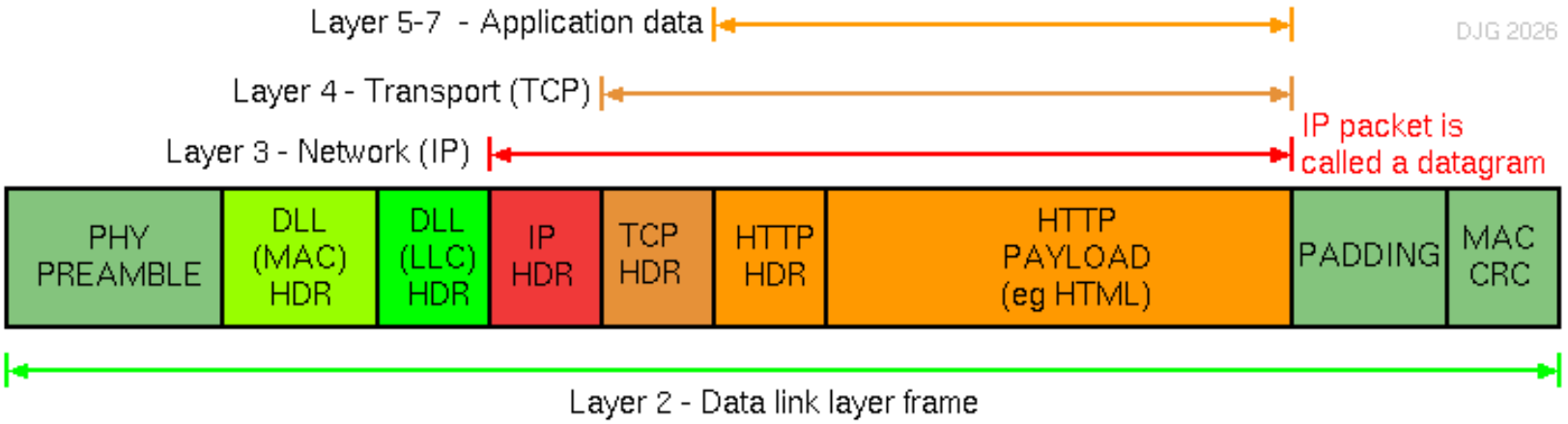
The primitives of one layer can map/tie-up with the protocols of the layer below.

- For example: reset() might be mapped to ‘send the bits 0101 0010’

Packet Embedding

In Computer Networks we generally see higher-layer information embedded within lower-layer information

- Such embedding is often protocol layering.
- Higher-layer information is often recovered and sent to the layer above by stripping off headers and trailers of the current layer.
- Eg. an IP entity only looks at the IP headers and passes 'up' the IP payload.
- Refer to the C/C++ IP protocol tick as a (fairly) good example.



BUT literal embedding (nesting) is not the only form of layering

Layering is to help describe or understand a communications system. It should **NOT strictly** determine the implementation strategy.

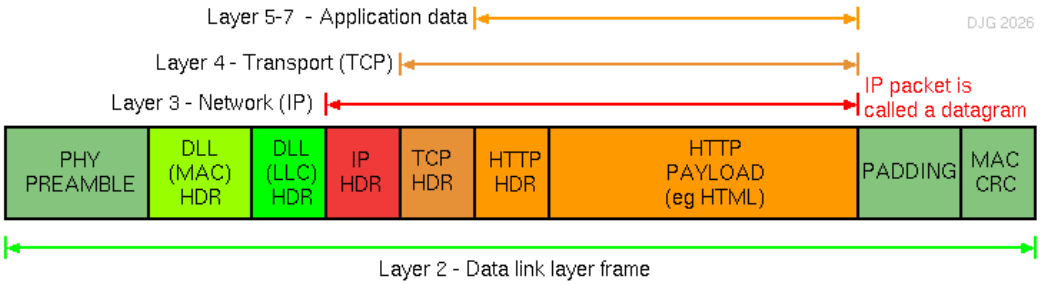
Principle Networking Layers

Layer 1 – Physical layer – carries the data (copper/fibre)
- a mechanism for delimiting bit cell boundaries and recovering the transmit clock is also needed (discussed later).

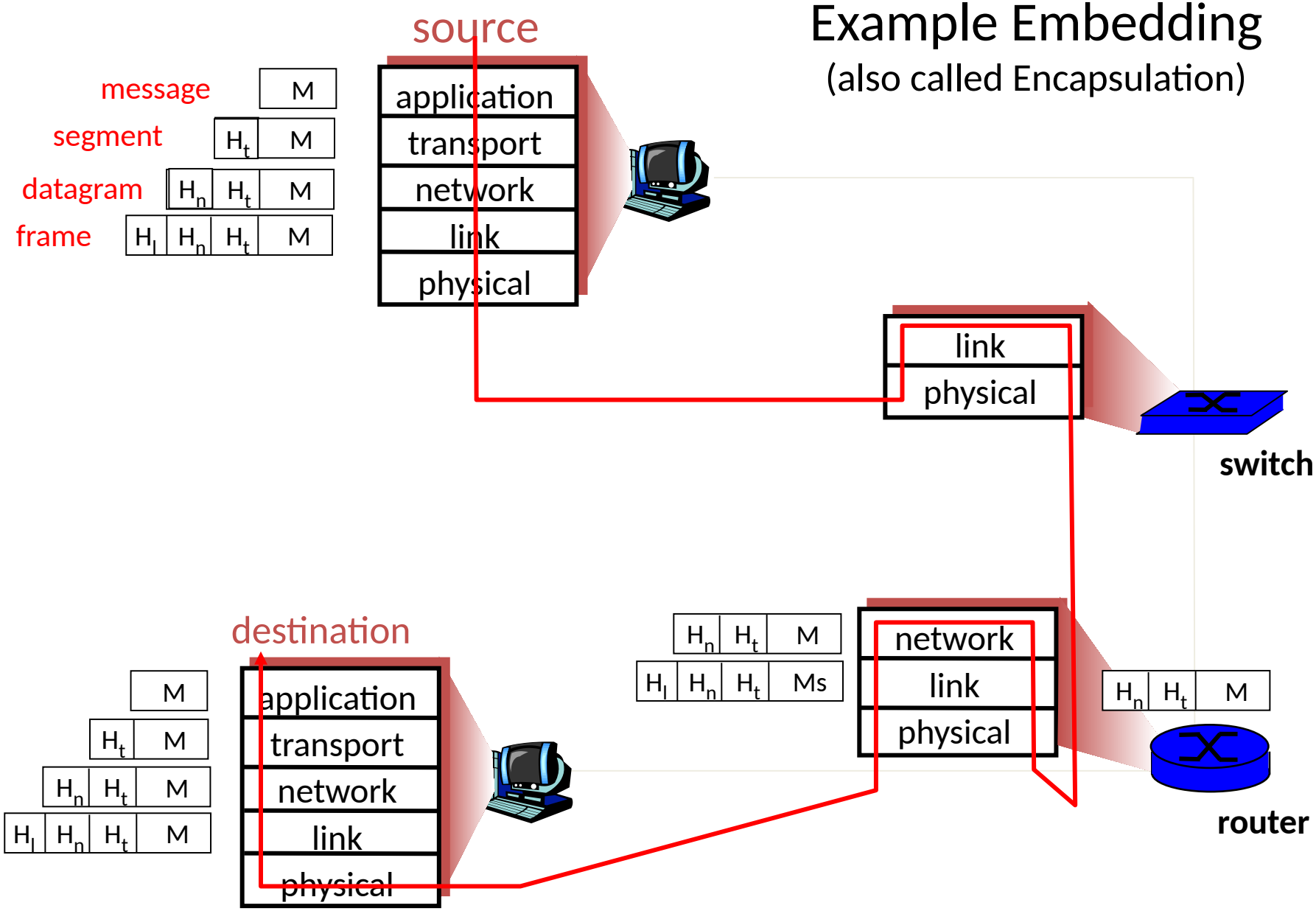
Layer 2 – Data link layer (for packet network):
- delimit the packet boundaries and add some check bits (CRC)
- provide media access control (MAC) for shared medium (radio)
- MAC not needed for a dedicated cable or TDM channel
- Eg. Ethernet

Layer 3 – Network layer
- Provide network-wide addressing and routing
- Eg. IP

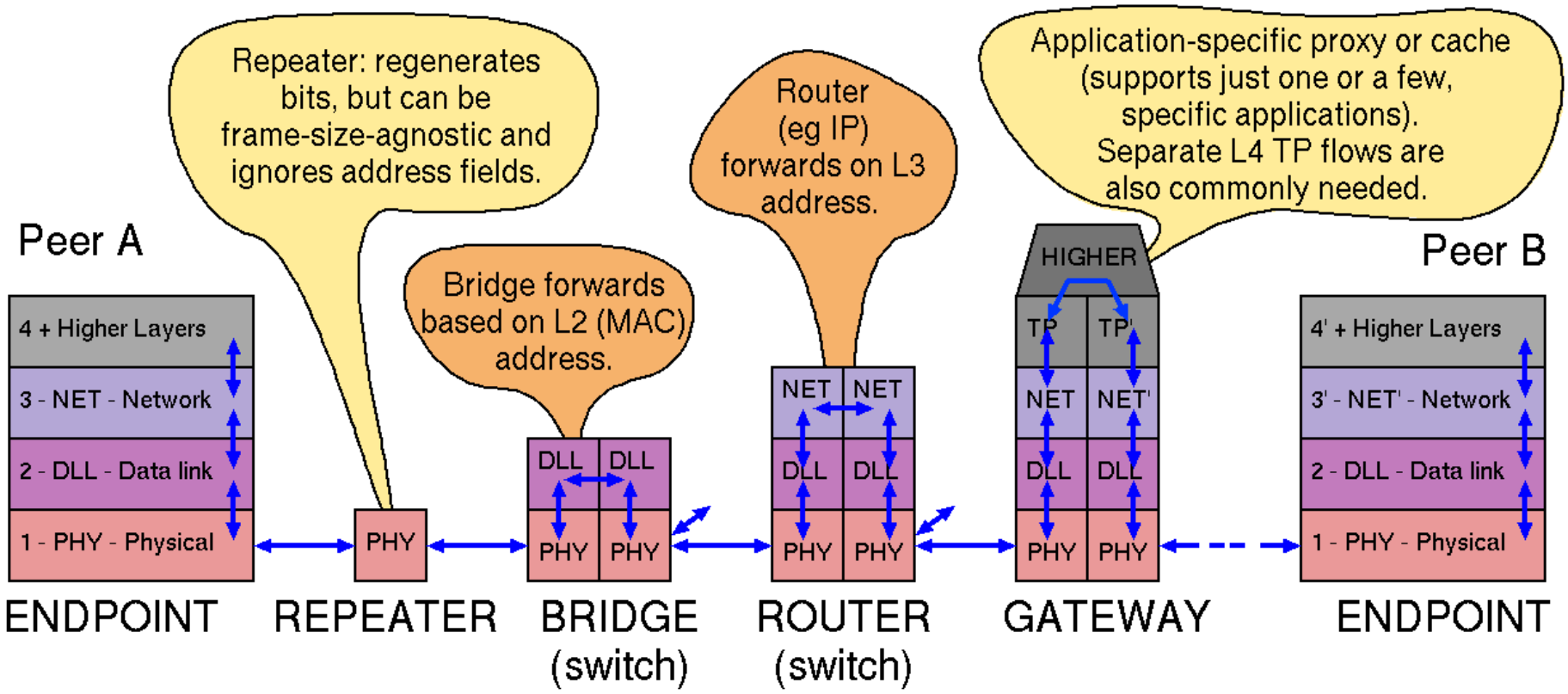
Layer 4 – Transport layer
- Ensure reliability and rate matching (flow control)
- May use retransmissions and ACK/NAK messages
- Eg. TCP



Example Embedding
(also called Encapsulation)



Peering in the OSI Reference Model

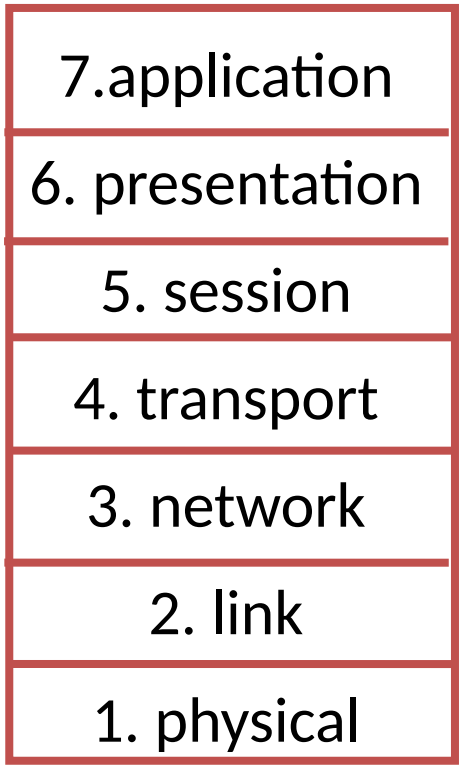


PHYs and DLLs can also vary on a per-hop basis (not shown primed though).

DJG 2026

Topping the OSI reference model

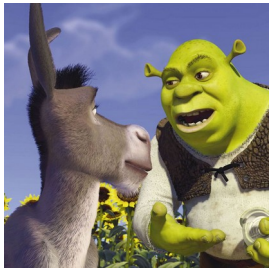
- 6. presentation: allow applications to interpret meaning of data, e.g., encryption, compression, machine-specific conventions
- *5. session:* synchronization, checkpointing, recovery of data exchange
- Internet stack "missing" these layers!
  - these services, *if needed*, must be implemented in application (or middleware…)

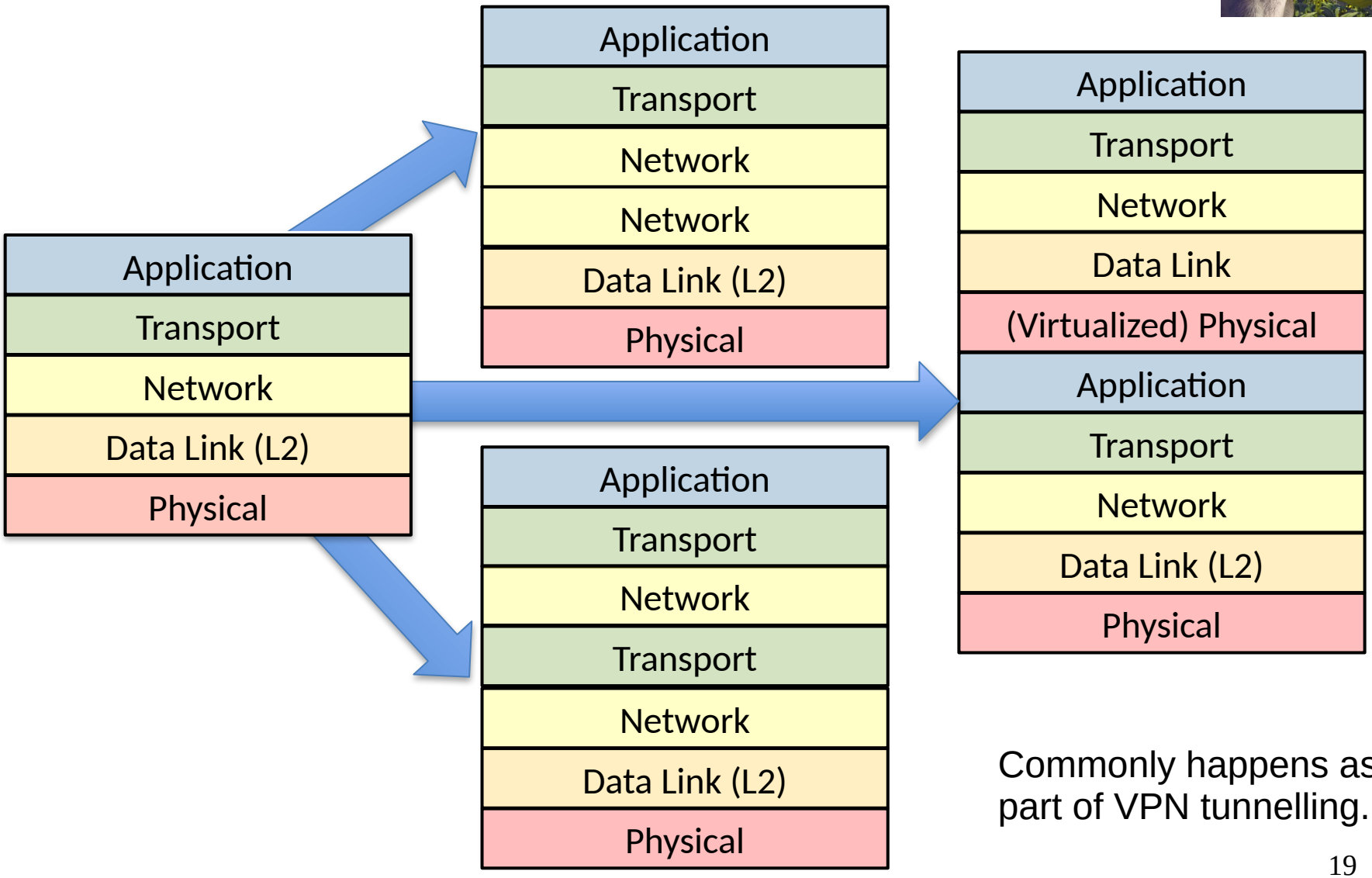


The seven-layer cake!

The operating system 'application' (eg. web browser) is not quite the same concept as a real user 'application' (eg. banking).

Layers on Layers examples





Commonly happens as part of VPN tunnelling.

Recap: what is a protocol?

human protocols:

- "what's the time?"
- "I have a question"
- introductions

… specific msgs sent

… specific actions taken when msgs received, or other events

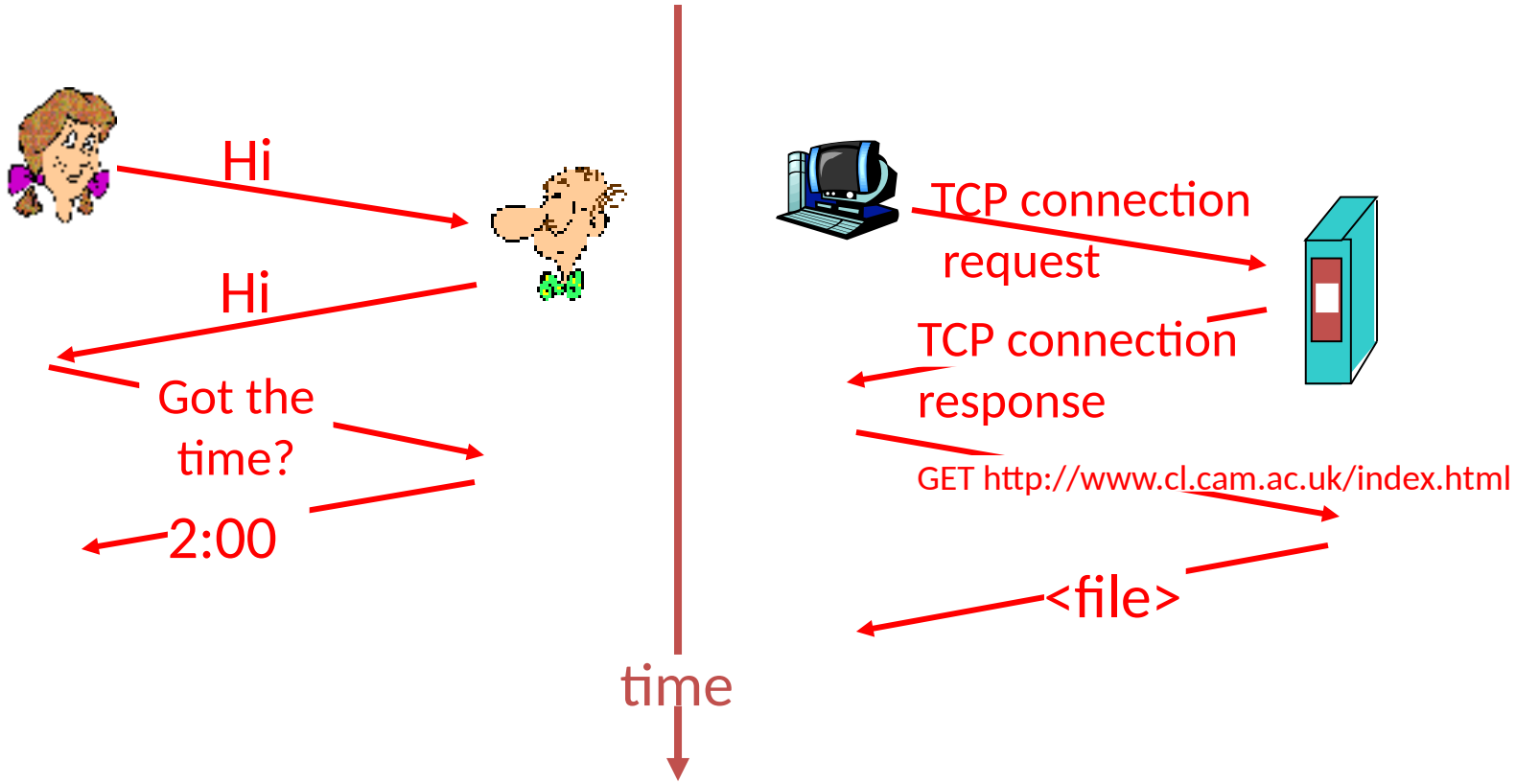
network protocols:

- machines rather than humans
- all communication activity on Internet governed by protocols

Protocols define format, order of msgs sent and received among network entities, and state updates on msg transmission, receipt

Recap: what is a protocol?

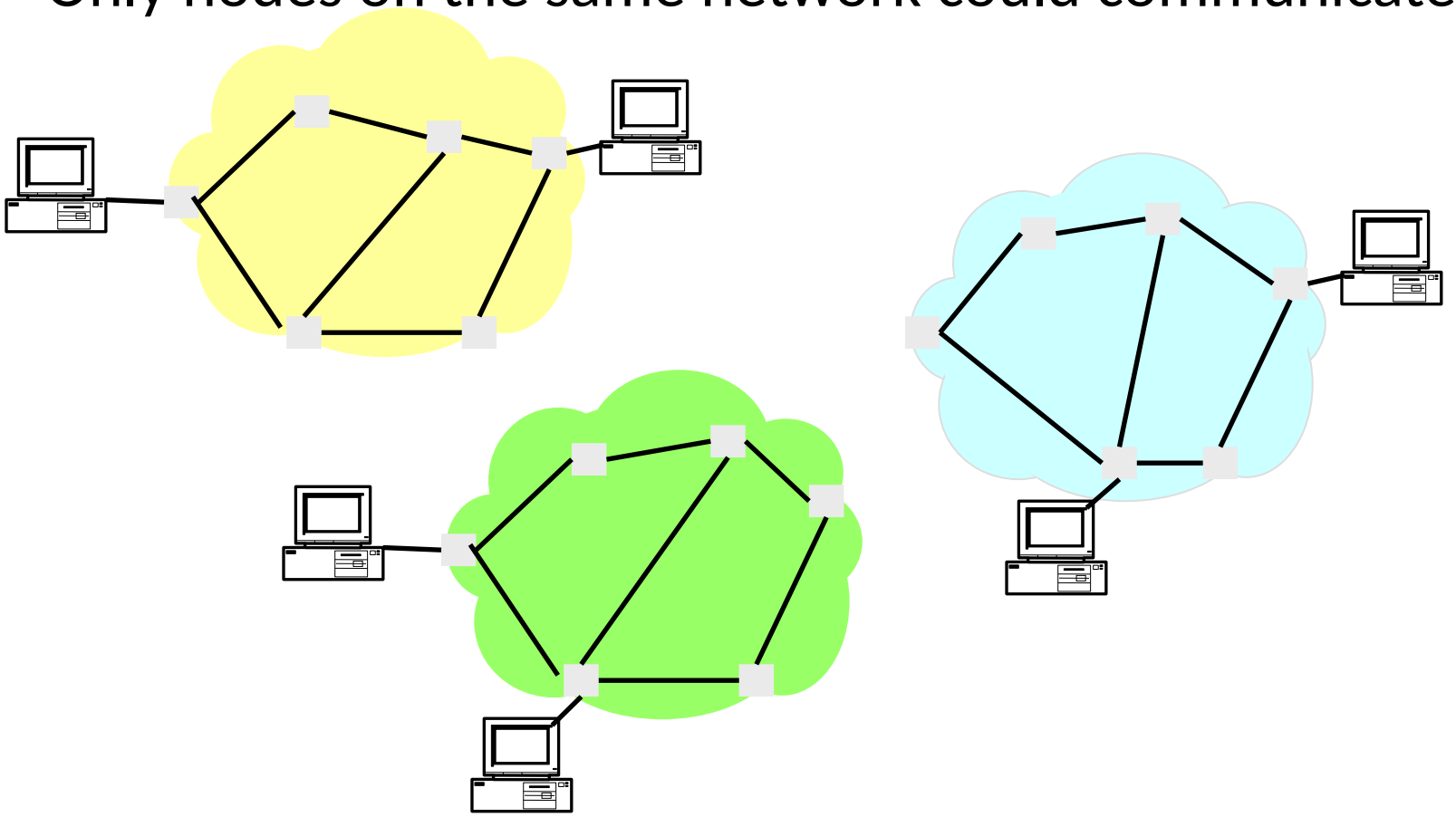
a human protocol and a computer network protocol:



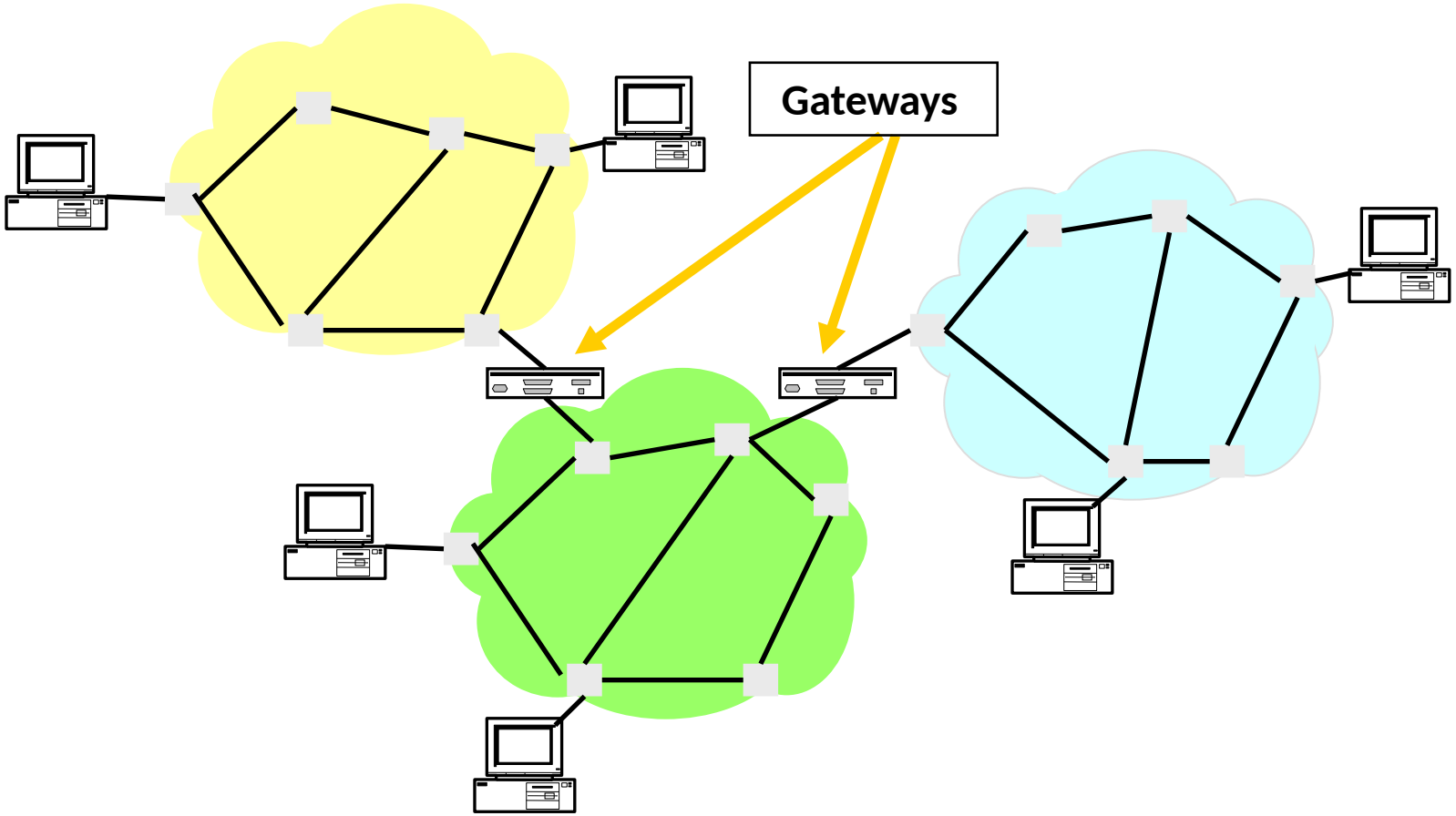
Q: Other human protocols?

Incompatible Standards Problem

- Many different packet-switching networks
- Each with its own Protocol
- Only nodes on the same network could communicate



INTERnet Solution



A gateway is generally a protocol convertor.
Today, gateways mostly used just between wildly different networks types.
NAT and firewall-like ‘gateways’ will be discussed later on.

Internet Design Goals (Clark ‘88)

- **Interconnect existing networks**
- Robust in face of failures
- Support multiple types of delivery services
- Accommodate a variety of networks
- Allow distributed management
- Easy host attachment
- Cost effective
- Allow resource accountability

Real Goals

Internet Motto

"We reject kings, presidents, and voting. We believe in rough consensus and running code" – David Clark

- **Build something that works!**
- Connect existing networks
- Robust in face of failures
- Support multiple types of delivery services
- Accommodate a variety of networks
- Allow distributed management
- Easy host attachment
- Cost effective
- Allow resource accountability

In the context of the Internet

Applications

…built on…

Reliable (or unreliable) L4 transport

…built on…

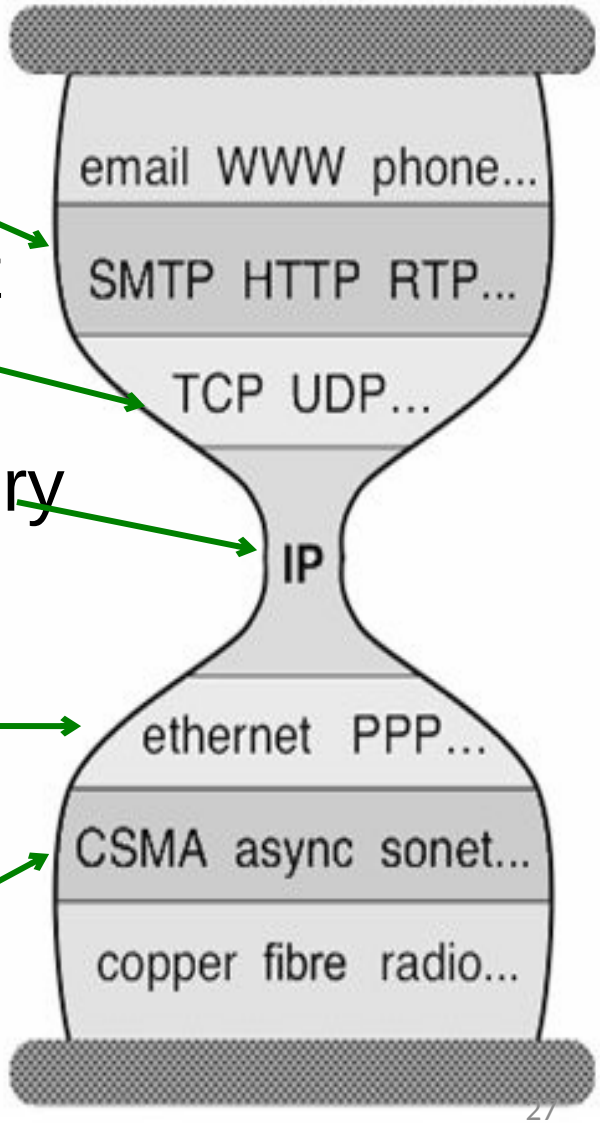
Best-effort L3 global datagram delivery

…built on…

Best-effort L2 local frame delivery

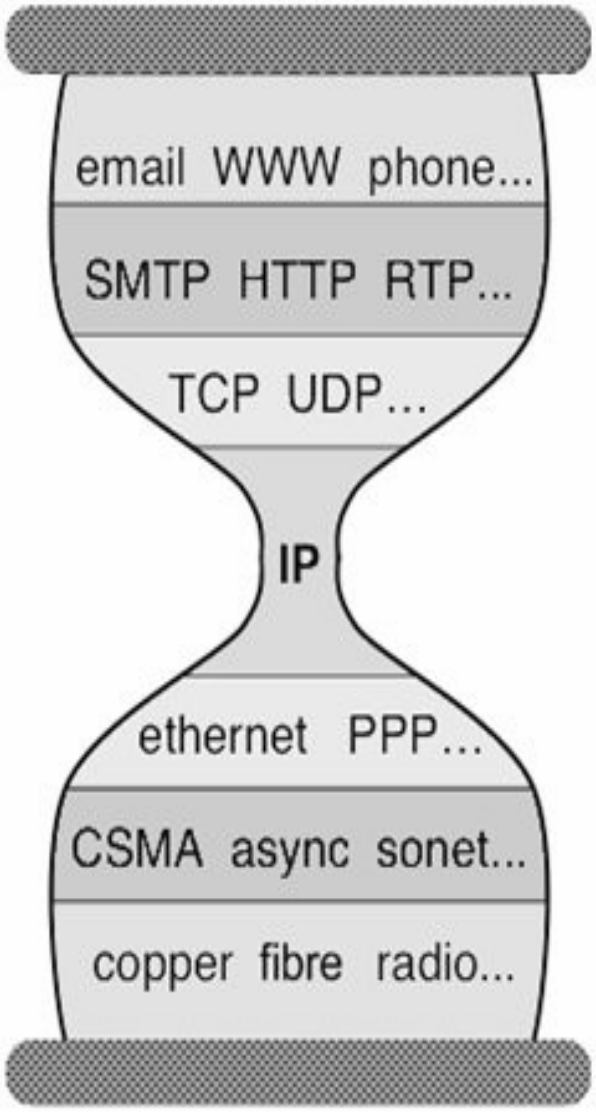
…built on…

L1 Physical transfer of bits



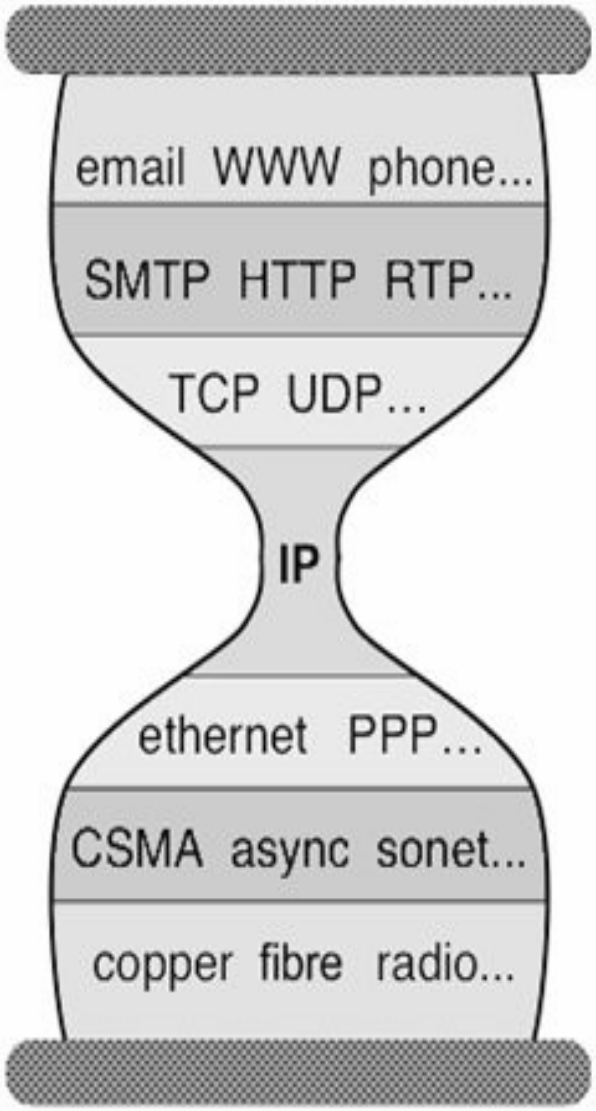
Narrow waist

- Each layer:
  - Depends on layer below
  - Supports layer above
  - Independent of others
- Alternatives within layer
  - Interfaces differ somewhat
  - Components pick which lower-level protocol to use
- But only one IP layer
  - Narrow waist, unifying protocol



Layering Crucial to Internet's Success

- Reuse
- Hides underlying detail
- Innovation at each level can proceed in parallel
- Pursued by very different communities



What are some of the drawbacks of protocols and layering?

Drawbacks of Layering

- Layer N may duplicate lower-layer functionality
  - e.g., error recovery to re-transmit lost data
- Information hiding may hurt performance
  - e.g., packet loss due to corruption vs. congestion
- Headers start to get really big
  - e.g., typical TCP+IP+Ethernet is 54 bytes
- Layer violations when the gains too great to resist
  - e.g., TCP-over-wireless
- Layer violations when network doesn't trust ends
  - e.g., firewalls

Placing Network Functionality

- Hugely influential paper: “End-to-End Arguments in System Design” by Saltzer, Reed, and Clark (‘84)
  - articulated as the “End-to-End Principle” (E2E)
- Endless debate over what it means
- Everyone cites it as supporting their position

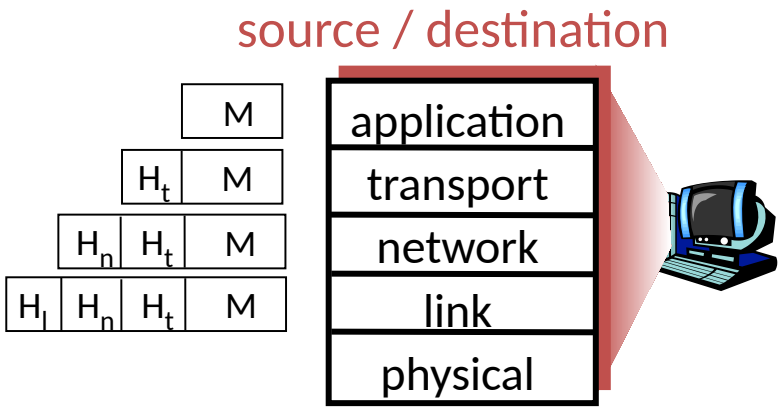
  (regardless of the position!)

Basic Observation

- Some application requirements can (ultimately) only be correctly implemented **end-to-end**
  - reliability, security, *etc.*
- Implementing these in the network is hard
  - every step along the way must be fail-proof
- Hosts
  - **Can** satisfy the requirement without network's help
  - **Will/must** do so, since they can't really rely on the network.

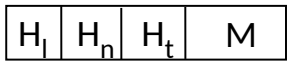
What Gets Implemented on Host?

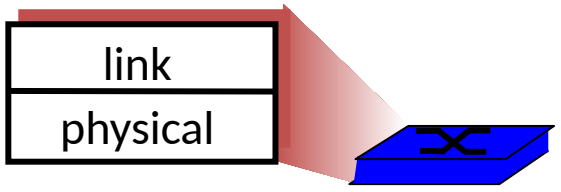
- Bits that arrive on PHY (wire/fibre/radio/...) must make it up to application
- Therefore, all layers must exist at the host



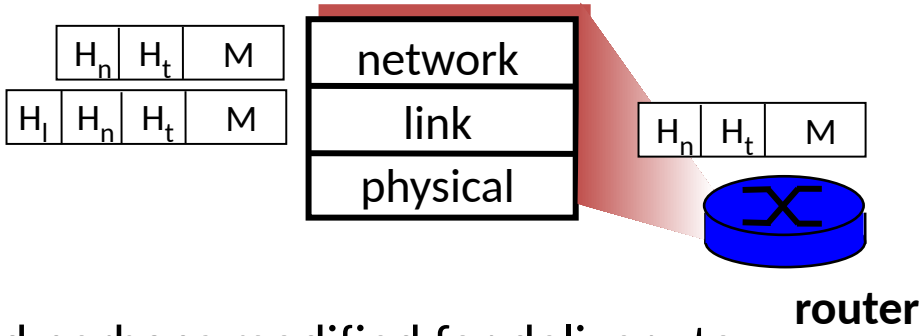
What's Implemented on Switches?

- L2 switches are much like routers, except they don't participate in global delivery, just local delivery
- They only need to support Physical and Datalink:
  - Don't need to support Network layer (L3)
- We won't otherwise focus on the router/switch distinction
  - Almost all boxes support network layer these days



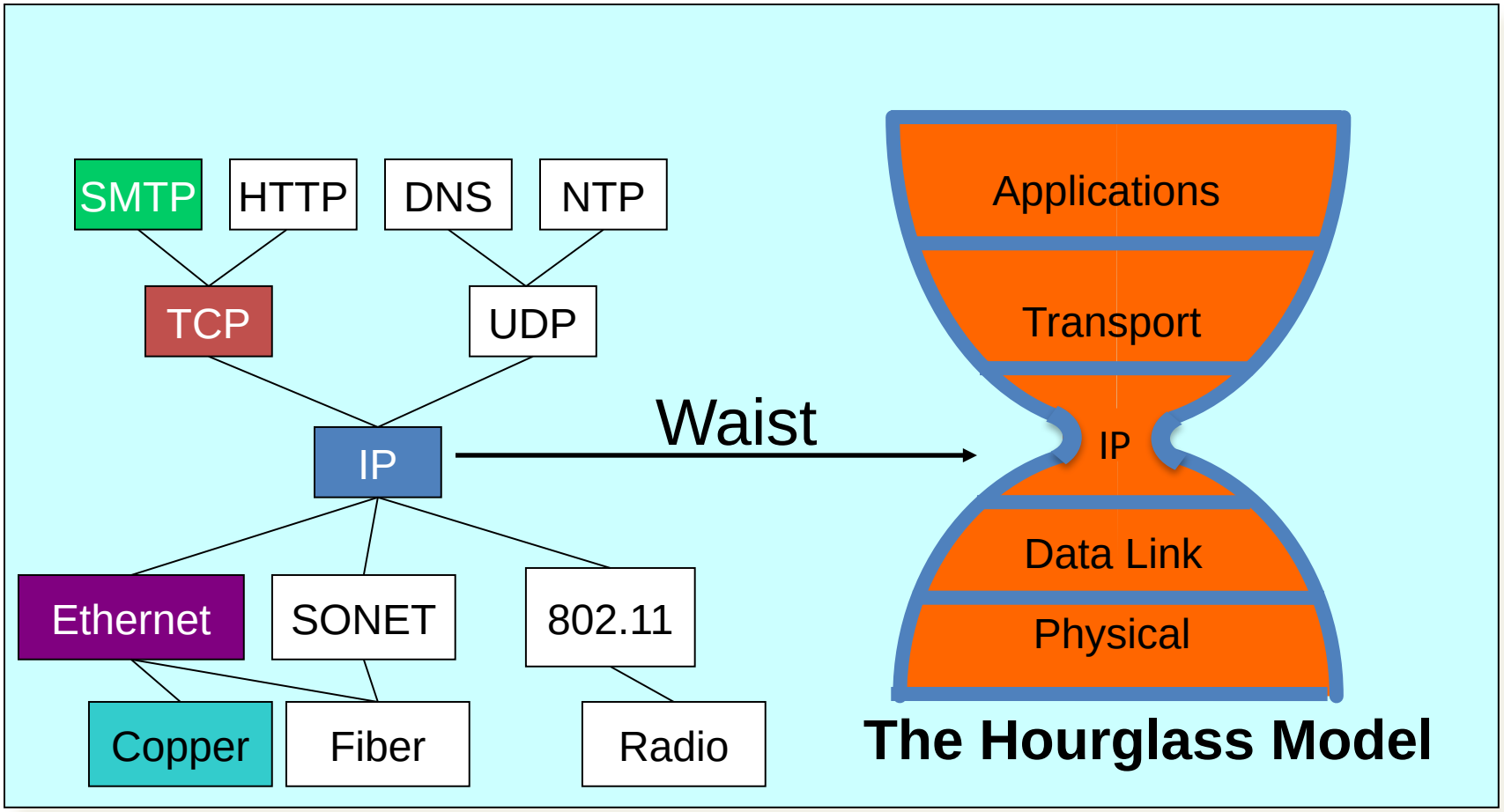


What Gets Implemented on a Router?



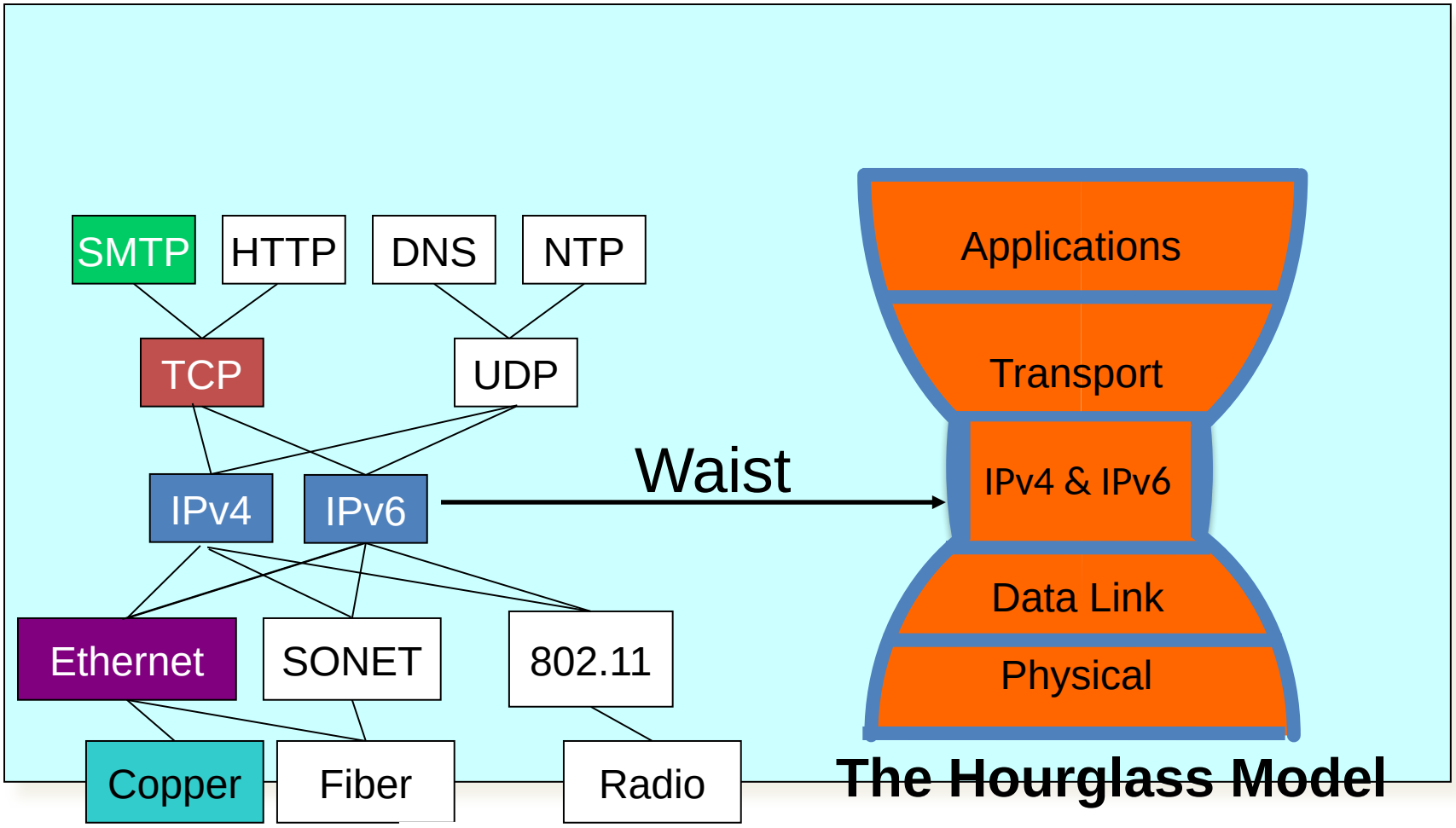
- Bits arrive on wire/phy:
  - Physical layer 1 necessary
- Packets must be comprehended and perhaps modified for delivery to next-hop:
  - Datalink layer 2 necessary
- Routers chose next hop based on network-layer address, which is global:
  - Network layer 3 necessary
- Routers don't support reliable delivery:
  - Transport layer 4 (and above) ***not*** supported
  - (A possible exception for real-time RTP traffic? Failed?)

The Ideal Internet *Hourglass*



There is just one network-layer protocol, **IP**.
The "narrow waist" facilitates interoperability.

The middle-age Internet *Hourglass*



There is just ~~one~~ TWO network-layer protocol, **IP**v4 + v6

The "narrow waist" facilitates interoperability(???)

Recap: Protocol Standardization

- All hosts must follow same protocol
  - Very small modifications can make a big difference
  - Or prevent it from working altogether
- This is why we have standards
  - Can have multiple implementations of protocol
- Internet Engineering Task Force (IETF)
  - Based on working groups that focus on specific issues
  - Produces “Request For Comments” (RFCs)
  - IETF Web site is ***https://www.ietf.org***
  - RFCs archived at ***https://www.rfc-editor.org***

Alternative to Standardization?

- Have one implementation used by everyone
- Open-source projects
  - Which has had more impact, Linux or POSIX?
- Or just sole-sourced implementation
  - Zoom, Signal, FaceTime, etc.

  which in turn create monopolies…
  - they limit/remove interoperability

Summary

- Engineering tools
  - Abstraction, Layering, Layers and Communications, Entities and Peers, Protocol as motivation, Examples of the *architects* process
  - Example: Internet Philosophy and Tensions
  - Decisions left un-made: Where to put stuff… What stuff needs putting…
- Standards
  - Political
  - Blind (and thus sometimes silly)
  - Good for user choice