Layered Network Architecture

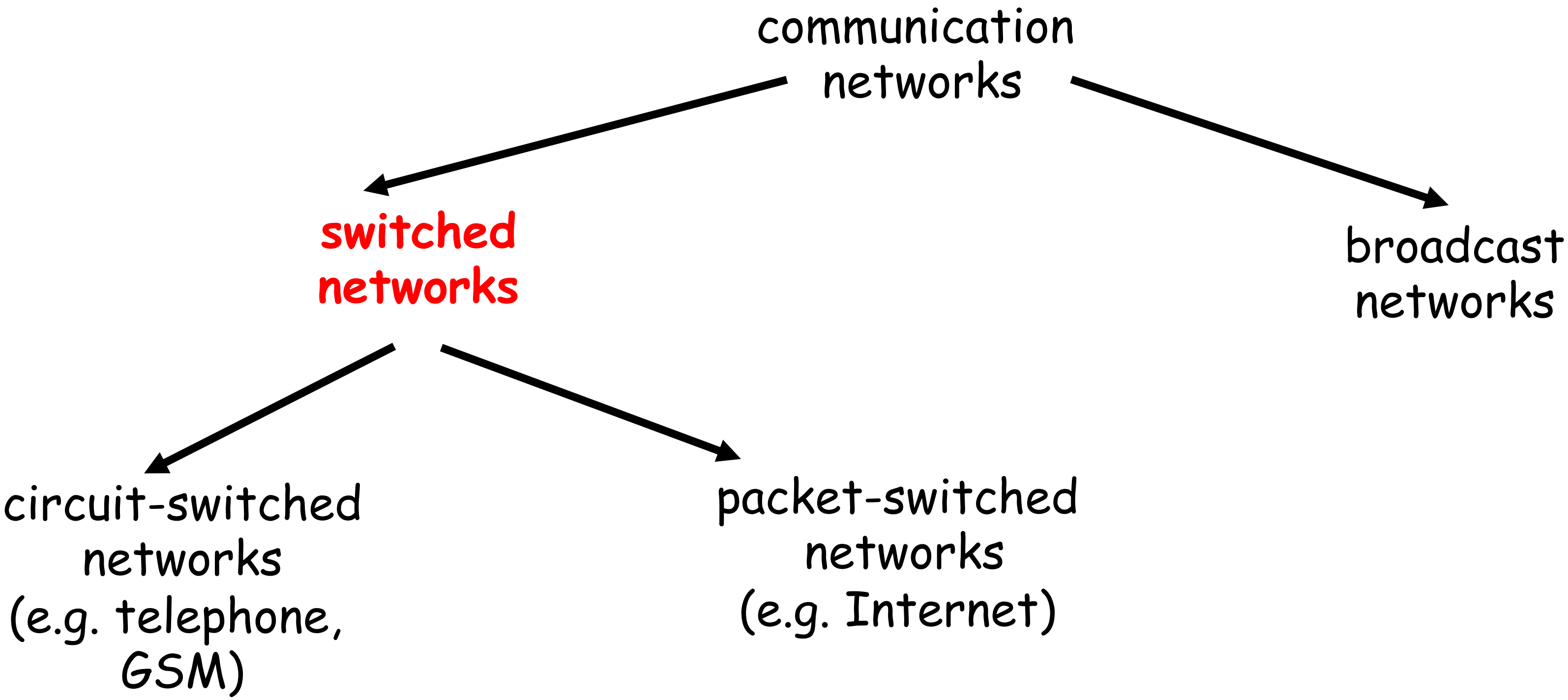
Reading: KR 1.5

Admin.

- Readings from the textbook and additional suggested readings (highly recommended) are linked on the syllabus page

- Assignment 1

Recap: A Taxonomy of Switched Networks



- Circuit switching: dedicated circuit per call/session:
  - E.g., telephone, GSM High-Speed Circuit-Switched Data (HSCSD)
- Packet switching: data sent thru network in discrete "chunks"
  - E.g., Internet, General Packet Radio Service (GPRS)

Summary: Packet Switching vs. Circuit Switching

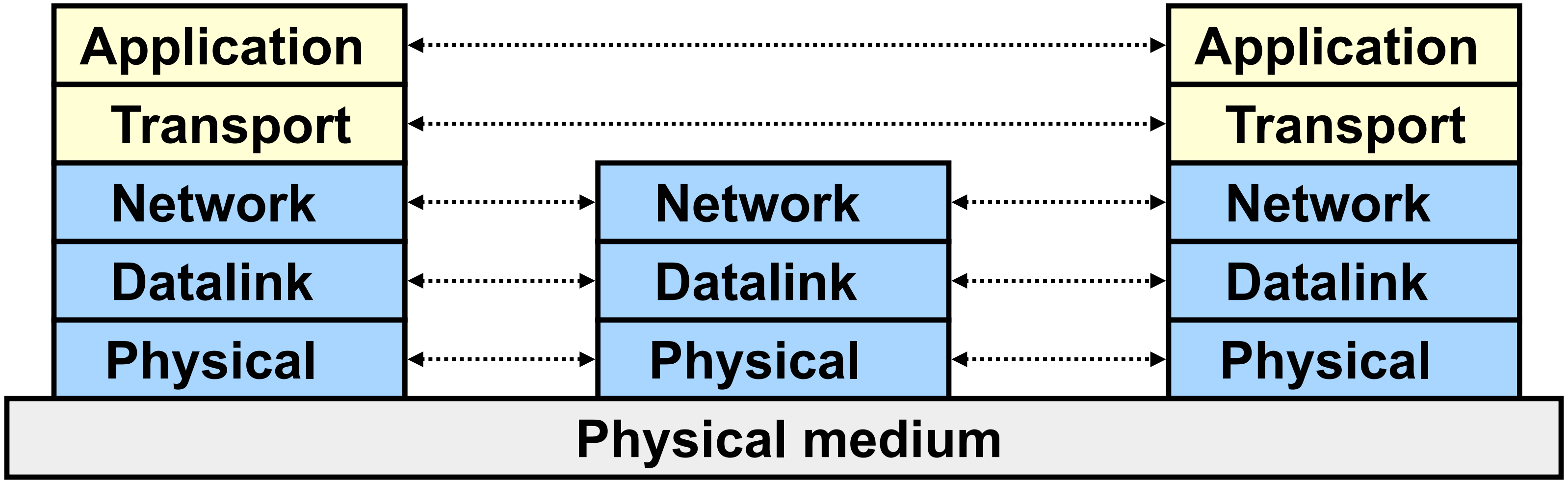
- Advantages of packet switching over circuit switching
  - Most important advantage of packet-switching over circuit switching is statistical multiplexing, and therefore efficient bandwidth usage
- Disadvantages of packet switching
  - potential congestion: packet delay and high loss
    - Protocols needed for reliable data transfer, congestion control
    - It is possible to guarantee quality of service (QoS) in packet-switched networks and still gain statistical multiplexing, but it adds much complexity
  - Packet header overhead
  - Per packet processing overhead

Outline

- Admin. and recap
- Layered network architecture
  - *What is layering?*
  - Why layering?
  - How to determine the layers?
  - ISO/OSI layering and Internet layering

What is Layering?

- A technique to organize a networked system into a succession of logically distinct entities, such that the service provided by one entity is solely based on the service provided by the previous (lower level) entity.



Outline

- Admin. and recap
- A taxonomy of communication networks
- Layered network architecture
  - What is layering?
  - *Why layering?*

Why Layering?

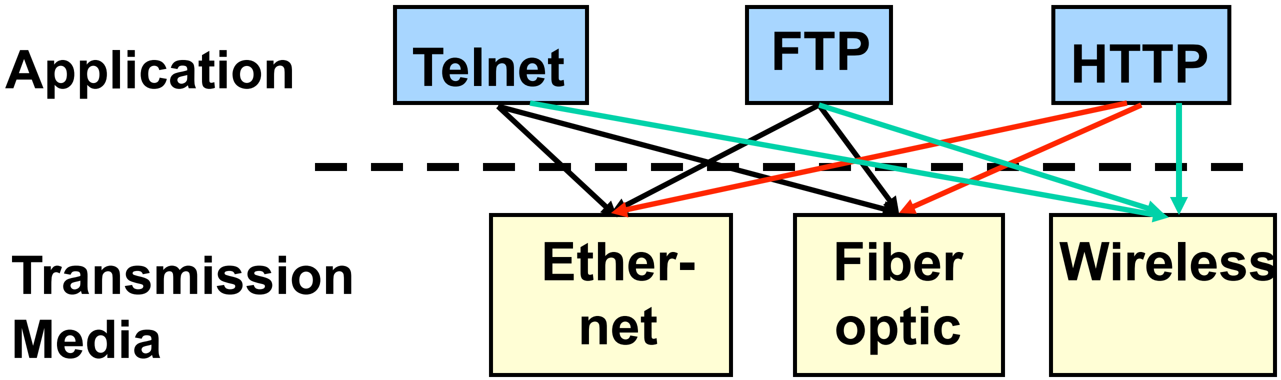
Networks are complex !

- Many "pieces":
  - Hardware
    - Hosts
    - Routers
    - Links of various media
  - Software
    - Applications
    - Protocols

Dealing with complex systems:
explicit structure allows identification of the relationship among a complex system's pieces

Modularization eases maintenance, updating of system:
- Change of implementation of a layer's service transparent to rest of system, e.g., changes in routing protocol doesn't affect rest of system

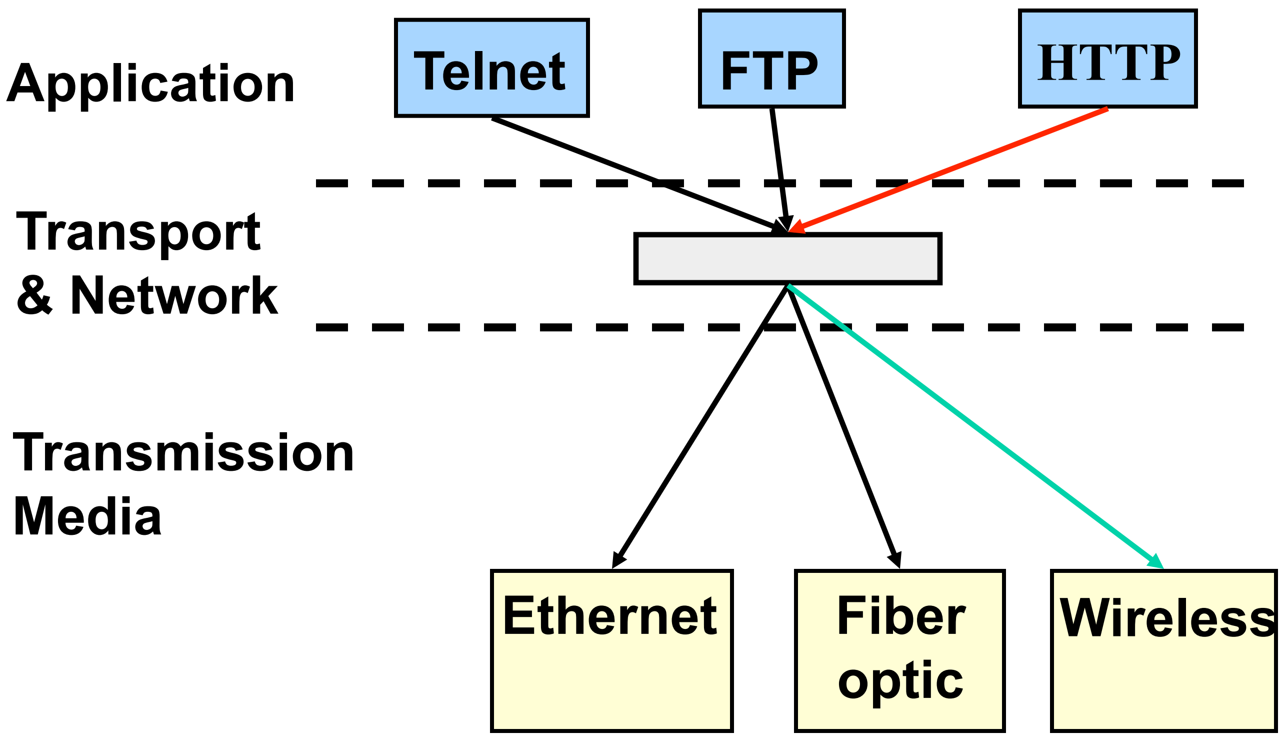
An Example: No Layering



- No layering: each new application has to be re-implemented for every network technology !

An Example: Benefit of Layering

- Introducing an intermediate layer provides a common abstraction for various network technologies



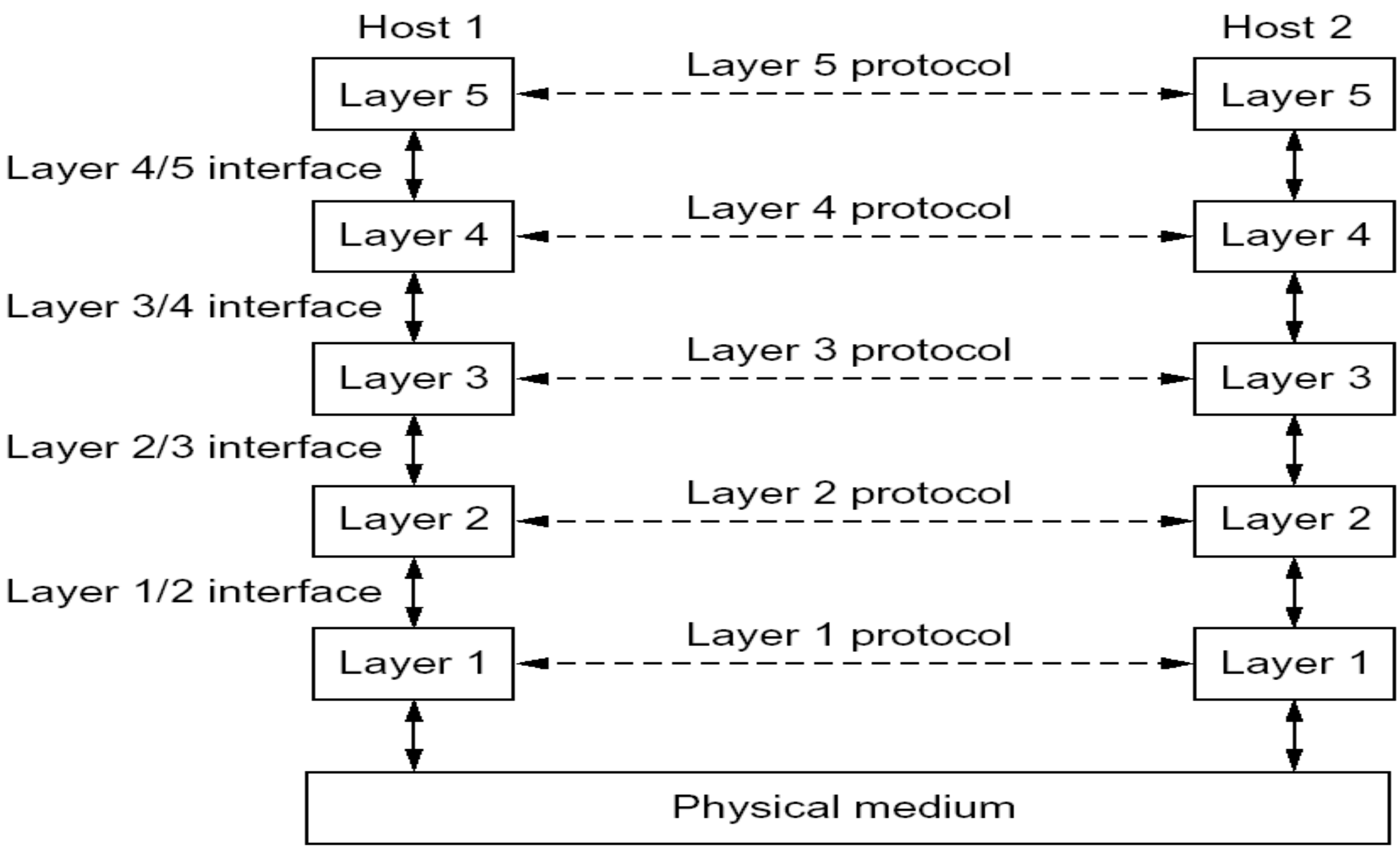
Discussion: potential disadvantages of layering

ISO/OSI Concepts

- ISO – International Standard Organization
- OSI – Open System Interconnection

- Service – says what a layer does
- Interface – says how to access the service
- Protocol – says how the service is implemented
  - A set of rules and formats that govern the communications between two peers

An Example of Layering

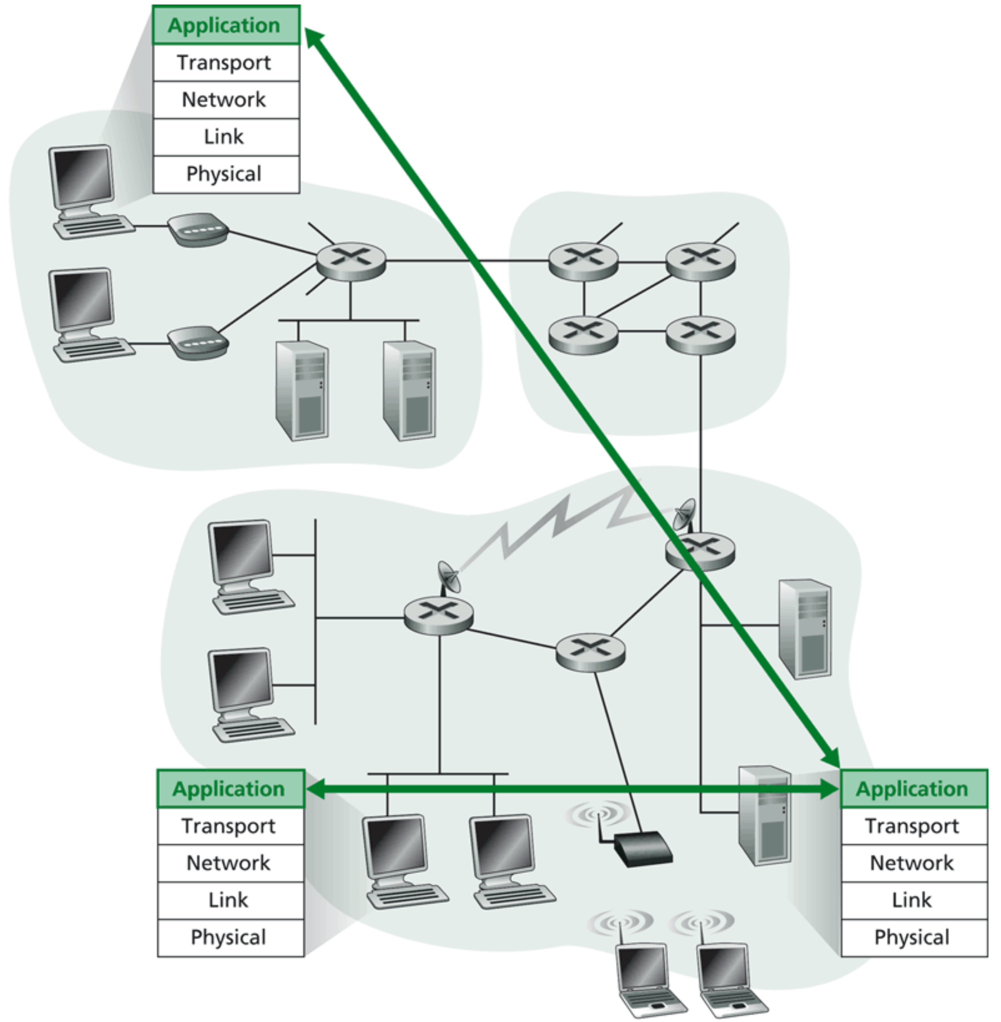


Layering -> *Logical* Communication

E.g.: application

- Provide services to users

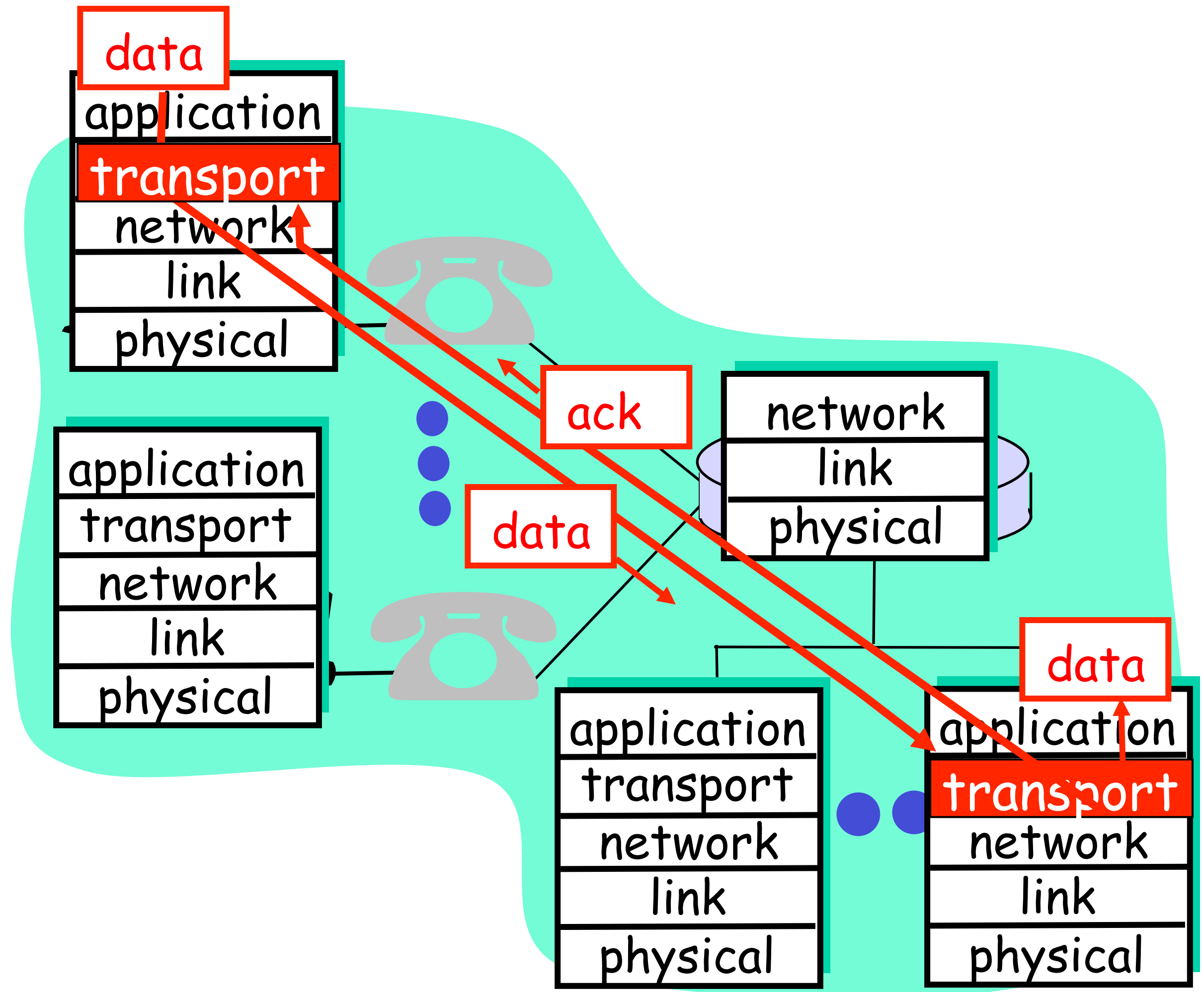
- Application protocol:
  - Send messages to peer
  - For example, HELO, MAIL FROM, RCPT TO are messages between two SMTP peers



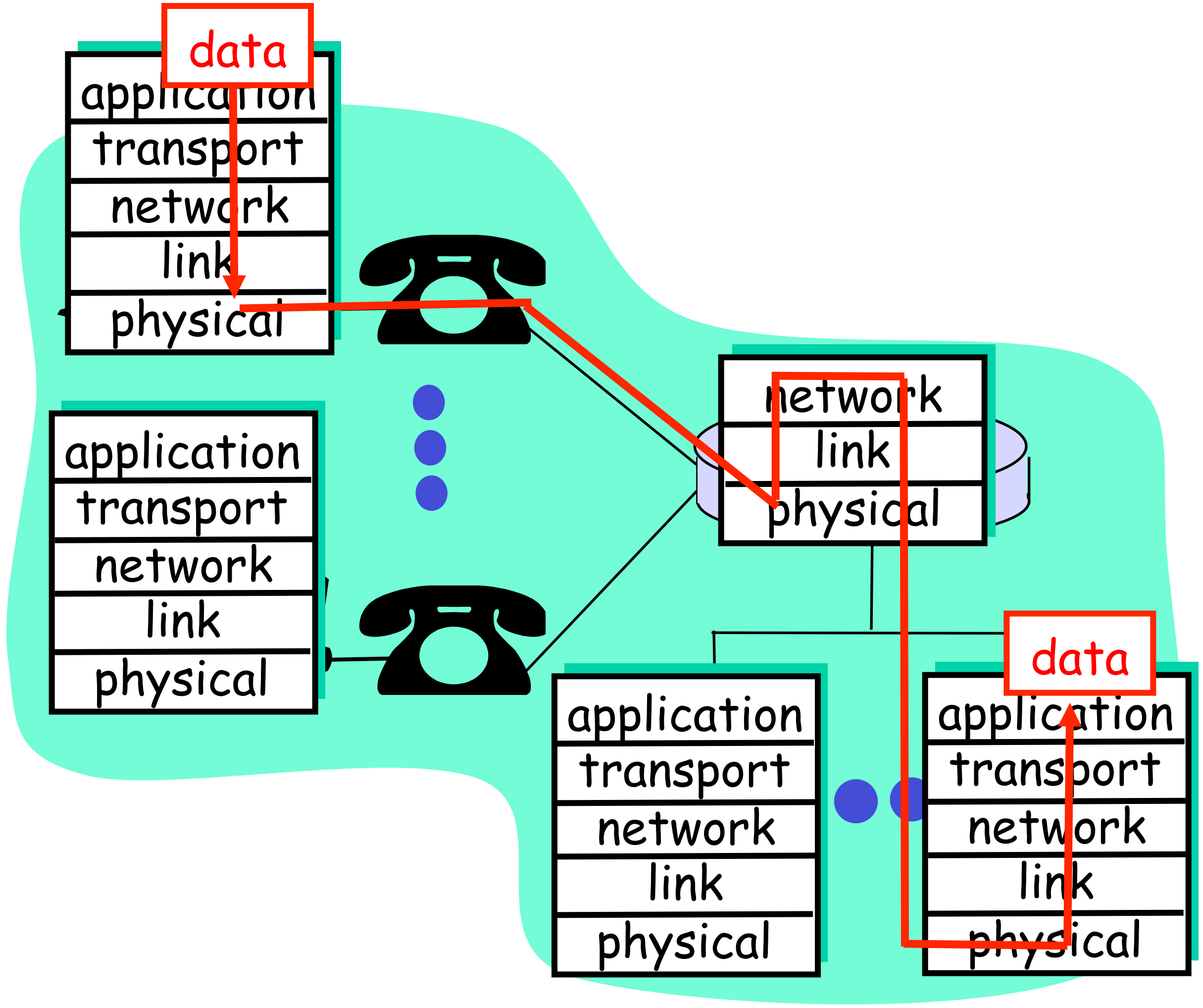
Layering: *Logical* Communication

E.g.: transport

- Take msg from app
- Transport protocol
  - Add control info to form "datagram"
  - Send datagram to peer
  - Wait for peer to ack receipt; if no ack, retransmit



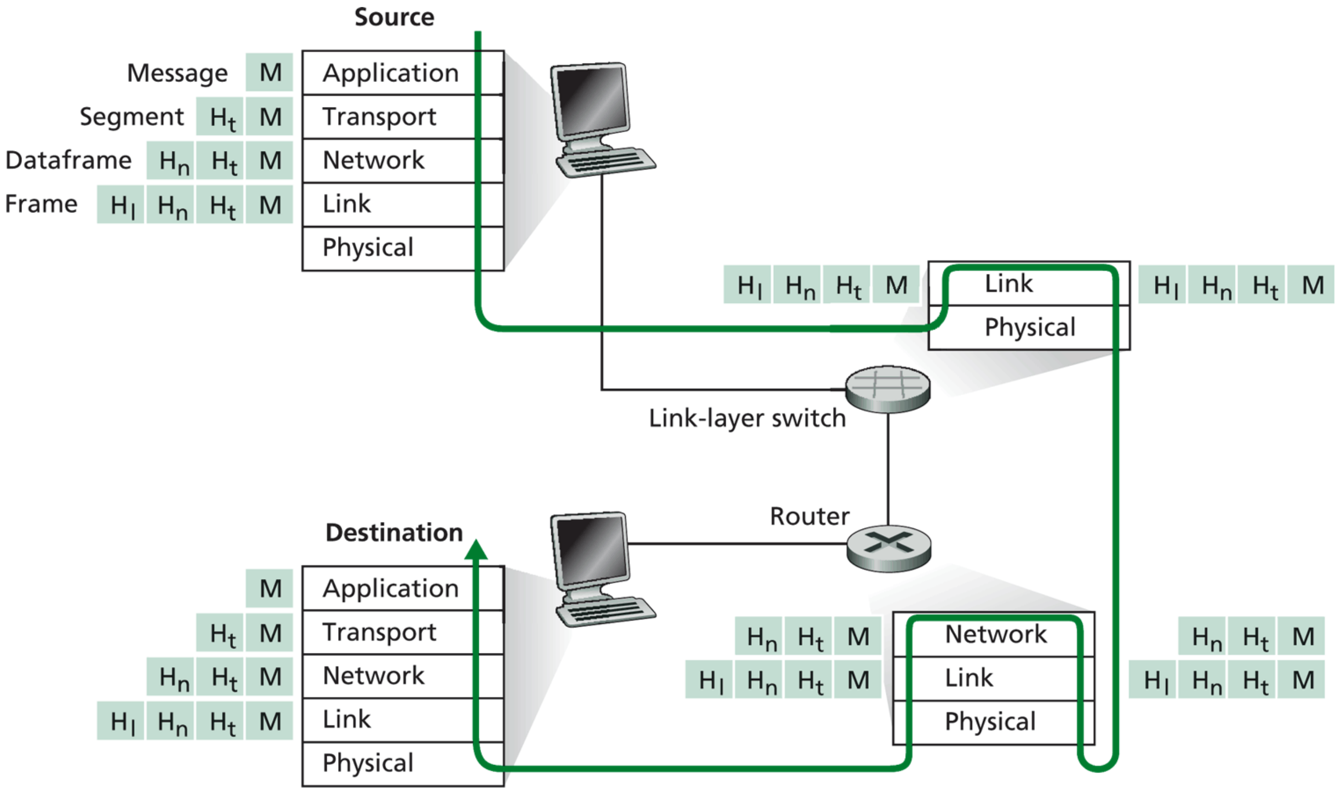
Layering: *Physical* Communication



Protocol Layering and Data

Each layer takes data from above

- Adds header information to create new data unit
- Passes new data unit to layer below



Outline

- Review
- A taxonomy of communication networks
- Layered network architecture
  - What is layering?
  - Why layering?
  - *How to determine the layers?*

Key design issue:

How do you *divide* functionalities among the layers?

The End-to-End Arguments

The function in question can completely and correctly be implemented only with the knowledge and help of the application standing at the endpoints of the communication systems. Therefore, providing that questioned function as a feature of the communications systems itself is not possible.

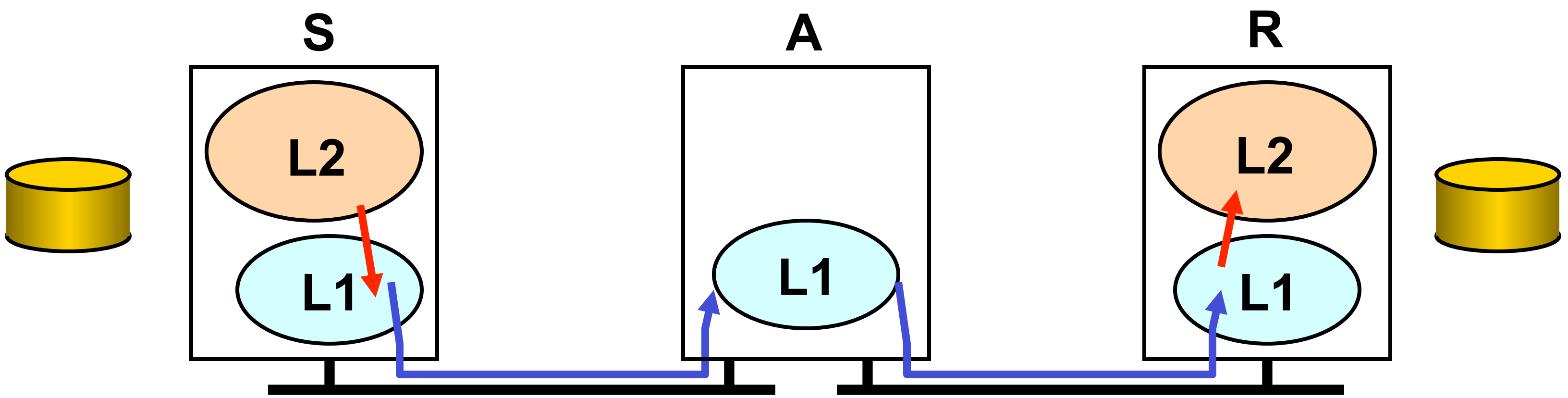
J. Saltzer, D. Reed, and D. Clark, 1984

What does End-to-End Arguments Mean?

- The application knows the requirements best, place functionalities as high in the layer as possible

- Think twice before implementing a functionality at a lower layer, even when you believe it will be useful to an application

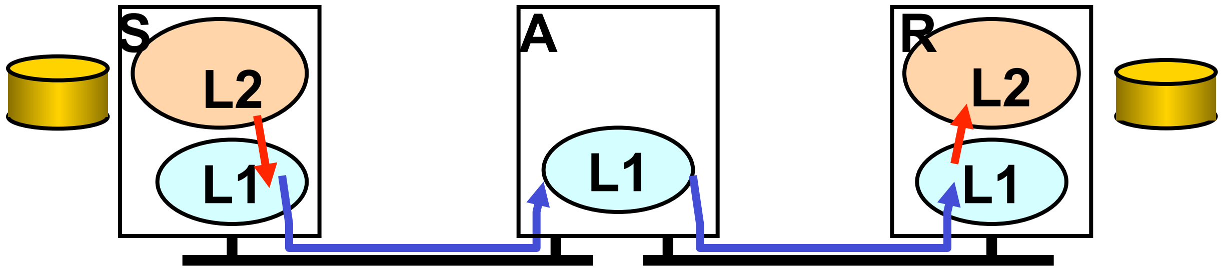
Example: Where to Provide Reliability ?



- Solution 1: the network (lower layer L1) provides reliability, i.e., each hop provides reliability
- Solution 2: the end host (higher layer L2) provides reliability, i.e., end-to-end check and retry

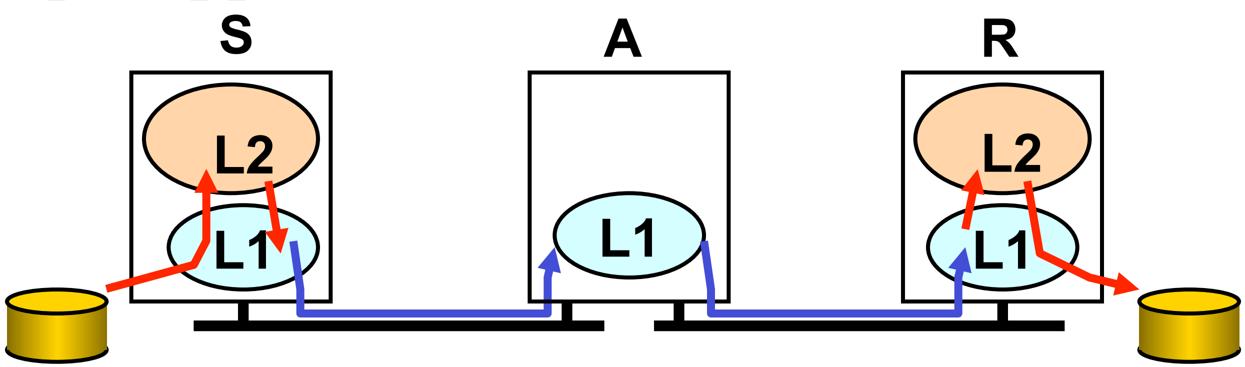
Reasons for Implementing Reliability at Higher Layer

- The lower layer cannot completely provide the functionality
  - The receiver has to do the check anyway!
  - L1 may ack and then crash, then the function fails
  - If L2 crash; the application crashes also->fate sharing
- Implementing it at lower layer increases complexity, cost and overhead at lower layer
  - Shared by all upper layer applications → everyone pays for it, even if you do not need it
- The upper layer
  - Knows the requirements better and thus may choose a better approach to implement it



Reasons for Implementing Reliability at Lower Layer

- Improve performance, e.g., if high cost/delay/… on a link local reliability
  - Improves efficiency
  - Reduces delay
- Share common code, e.g., reliability is required by multiple applications

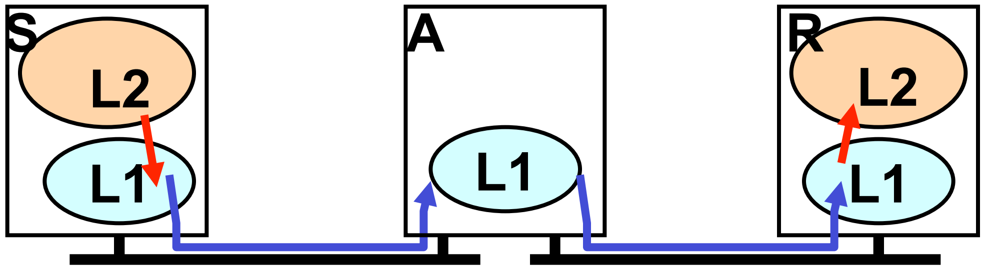


Summary: End-to-End Arguments

- If a higher layer can do it, don't do it at a lower layer -- the higher the layer, the more it knows about the best what it needs
- Add functionality in lower layers *iff* it
  1. is used by and improves performance of a large number of (current and potential future) applications;
  2. does not hurt (too much) other applications, and
  3. does not increase (too much) complexity/overhead.
- Practical tradeoff, e.g.,
  - Allow multiple interfaces at a lower layer (one provides the function; one does not)

Examples

- We used reliability as an example
- Assume two layers (L1: network; L2: end-to-end). Where may you implement the following functions?
  - Security (privacy of traffic)
  - Quality of service (e.g., delay/bandwidth guarantee)
  - Flow control (e.g., not to overwhelm network links or receiver)



Challenges

- Challenges to build a good (networking) system: find the right balance between:

end-to-end arguments

performance ⟷ reuse, interoperability, implementation effort (apply layering concepts)

No universal answer: the answer depends on the goals and assumptions!

The Design Philosophy of the DARPA Internet

Goals

0. **Connect different networks**

1. Survivability in the face of failure
2. Support multiple types of services
3. Accommodate a variety of networks

4. Permit distributed management of resources
5. Be cost effective
6. Permit host attachment with a low level of effort
7. Be accountable

Survivability in the Face of Failure: Questions

- What does the goal mean?
- Why is the goal important?
- How does the Internet achieve this goal?
- Does the Internet achieve this goal (or in what degree does the Internet achieve this goal)?

Survivability in the Face of Failure

- Continue to operate even in the presence of network failures (e.g., link and router failures)
  - As long as the network is not partitioned, two endpoints should be able to communicate…moreover, any other failure (excepting network partition) should be transparent to endpoints
- Decision: maintain state only at end-points (fate-sharing)
  - Eliminate the problem of handling state inconsistency and performing state restoration when router fails
- Internet: stateless network architecture

Support Multiple Types of Service: Questions

- What does this goal mean?
- Why is the goal important?
- How does the Internet achieve this goal?
- Does the Internet achieve this goal (or in what degree does the Internet achieve this goal)?

Support Multiple Types of Service

- Add UDP to TCP to better support other types of applications
  - E.g., "real-time" applications
- This was arguably the main reason for separating TCP and IP
- Provide datagram abstraction: lower common denominator on which other services can be built: everything over IP
  - Service differentiation was considered (remember ToS?), but this has never happened on the large scale (Why?)

Support a Variety of Networks: Questions

- What does the goal mean?
- Why is this goal important?
- How does the Internet achieve this goal?
- Does the Internet achieve this goal (or in what degree does the Internet achieve this goal)?

Support a Variety of Networks

- Very successful
  - Because the minimalist service; it requires from underlying network only to deliver a packet with a "reasonable" probability of success
- …does not require:
  - Reliability
  - In-order delivery
- The mantra: IP over everything
  - Then: ARPANET, X.25, DARPA satellite network..
  - Now: ATM, SONET, WDM…

Other Goals

- Permit distributed management of resources
- Be cost-effective
- Permit host attachment with a low level of effort
- Be accountable