# Statistical Multiplexing; Layered Network Architecture

**Qiao Xiang**, Congming Gao, Qiang Su

https://sngroup.org.cn/courses/cnns-xmuf25/index.shtml

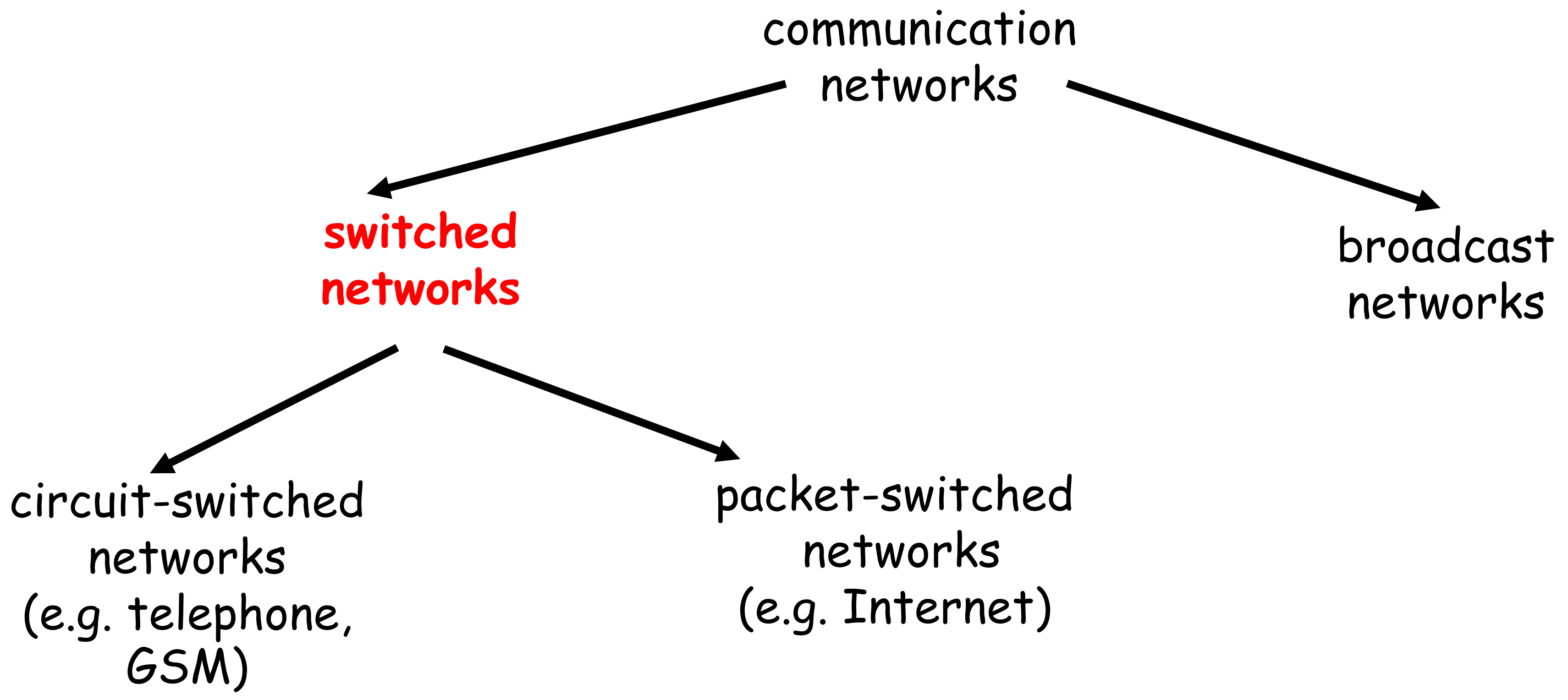
09/09/2025

## Outline

- Admin. and recap
- A taxonomy of communication networks
  - circuit switching vs. packet switching
  - datagram and virtual circuit packet switched networks
- Layered network architecture

## Recap: A Taxonomy of Comm. Networks

- Basic question: how are data (the bits) transferred through communication networks?


communication networks
switched networks
broadcast networks
circuit-switched networks (e.g. telephone, GSM)
packet-switched networks (e.g. Internet)

## Recap: Circuit Switching vs. Packet Switching

| | circuit switching | packet switching |
|---|---|---|
| resource usage | use a single partition bandwidth | use whole link bandwidth |
| reservation/setup | need reservation (setup delay) | no reservation |
| resource contention | busy signal (session loss) | congestion (long delay and packet losses) |
| charging | time | packet |
| header | no per-pkt header | per packet header |
| fast path processing | fast | per packet processing |

## Queueing Theory

- Strategy:
  - model system state
    - if we know the fraction of time that the system spends at each state, we can get answers to many basic questions: how long does a new request need to wait before being served?
- System state changes upon events:
  - introduce state transition diagram
  - focus on equilibrium: state trend neither growing nor shrinking (key issue: how to define equilibrium)

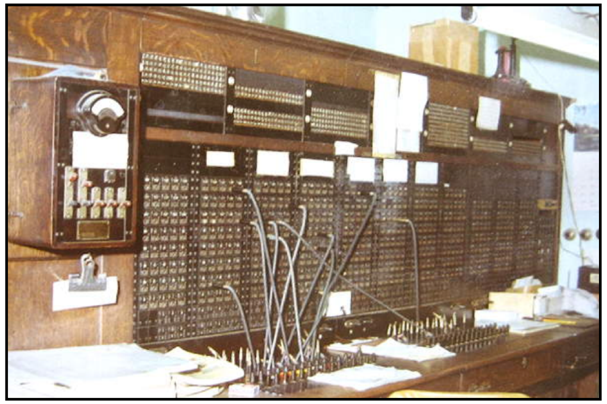
- Our approach: We are not interested in extremely precise modeling, but want quantitative intuition

## Warm up: Analysis of Circuit-Switching Blocking (Busy) Time

- Assume a link has only a finite number of N circuits

- Objective: compute the percentage of time that a new session (call) is blocked

- Analogy in a more daily-life scenario?
- Key parameters?



# Analysis of Circuit-Switching Blocking (Busy) Time

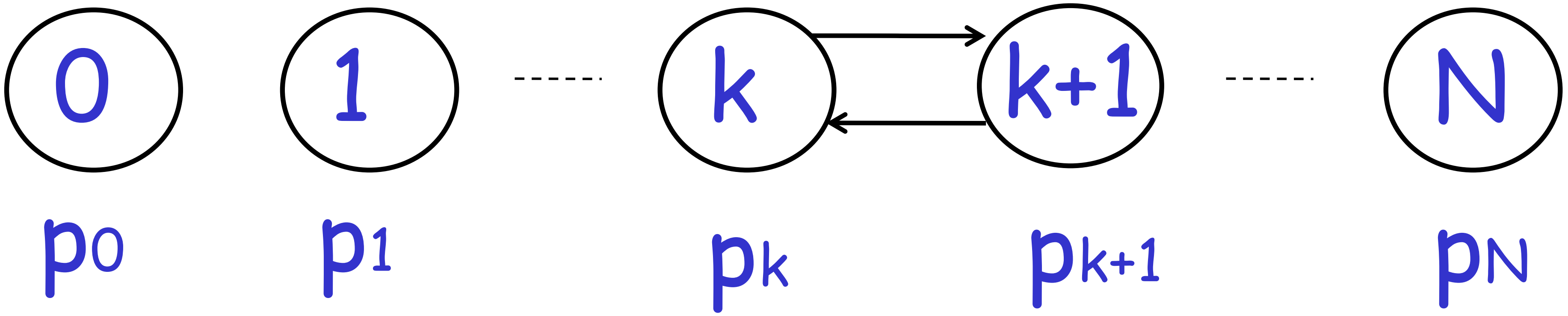
- Consider a simple arrival pattern
  - client requests arrive at a rate of $\lambda$ (lambda/second)
  - service rate: each call takes on average $1/\mu$ second

- Arrival and service patterns: memoryless (Markovian)
  - During a small interval $\Delta t$, the number of expected new arrivals is: $\lambda \Delta t$
  - During a small interval $\Delta t$, the chance (fraction) of a current call finishes is: $\mu \Delta t$

- This model is also called an M/M/N model

## Analysis of Circuit-Switching Blocking (Busy) Time: State

system state: # of busy lines



Q: How to characterize equilibrium?

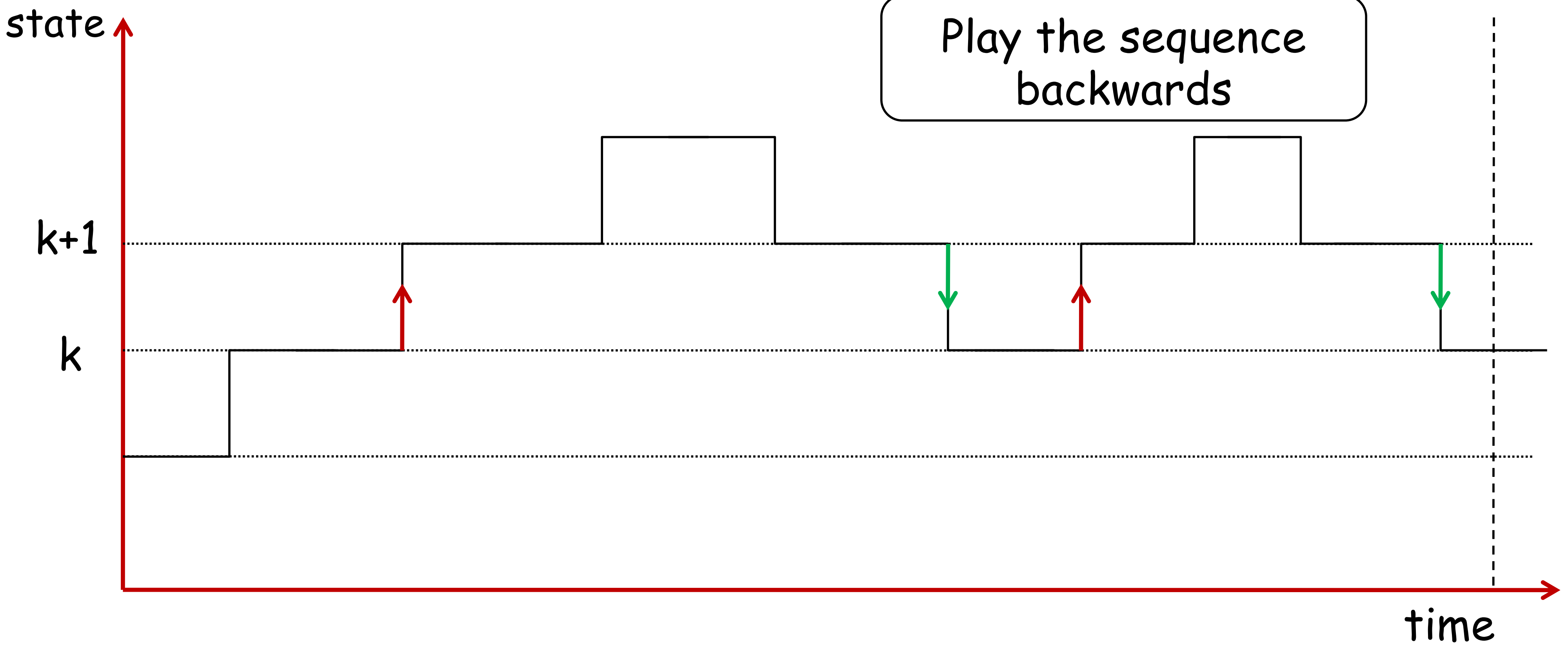
# Equilibrium = Time Reversibility [Frank Kelly]

- Statistically cannot distinguish

$$\boxed{\#f_{k\to k+1} = \#f_{k+1\to k}}$$

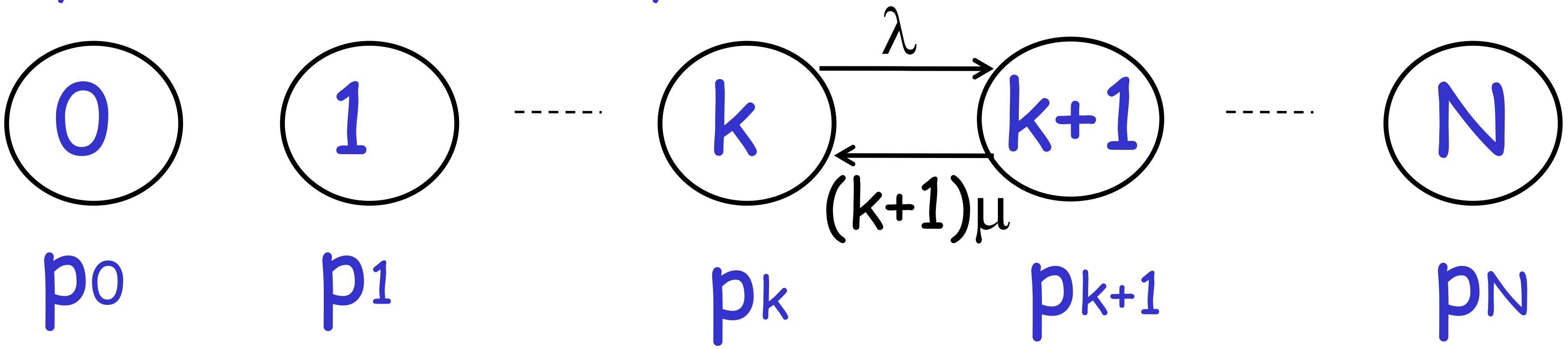
$$\boxed{\#b_{k\to k+1}, \quad \#b_{k+1\to k}}$$

Play the sequence backwards



# Analysis of Circuit-Switching Blocking (Busy) Time: Sketch

system state: # of busy lines



at equilibrium (time resersibility) in one unit time:

#(transitions k $\rightarrow$ k+1) = #(transitions k+1 $\rightarrow$ k)

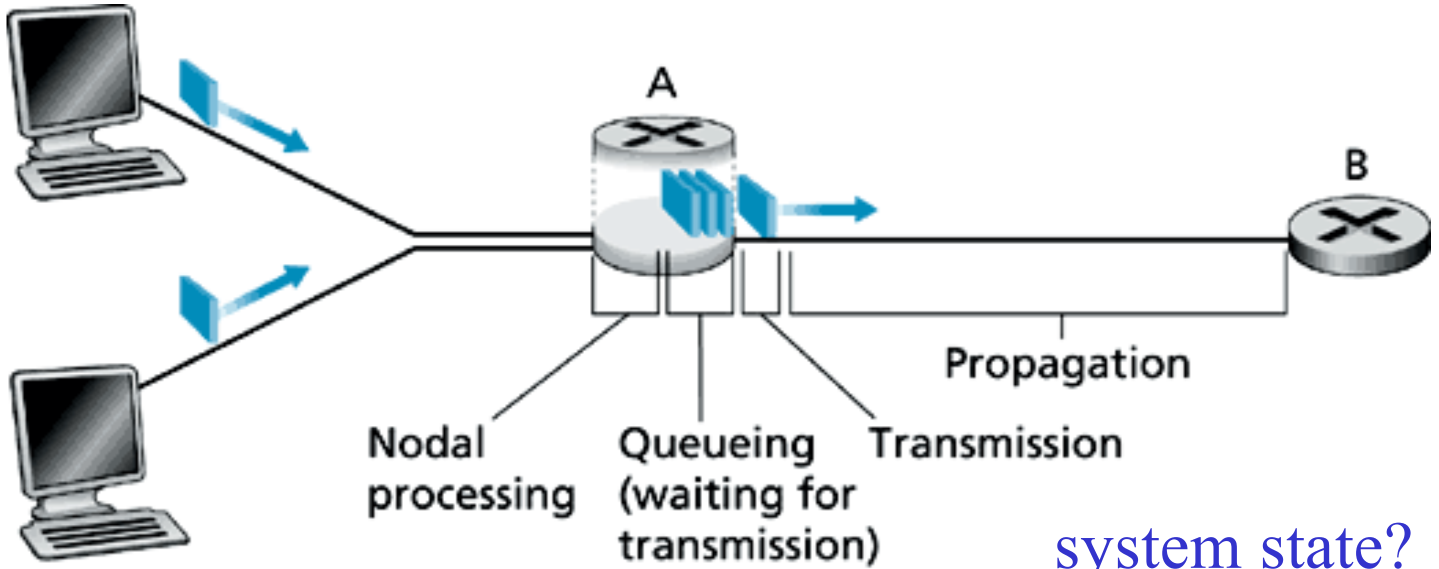
$$\boxed{p_k \lambda = p_{k+1}(k+1)\mu}$$

$$\boxed{p_{k+1} = \tfrac{1}{k+1}\tfrac{\lambda}{\mu} p_k = \tfrac{1}{(k+1)!}\left(\tfrac{\lambda}{\mu}\right)^{k+1} p_0}$$

$$\boxed{p_0 = \frac{1}{1 + \frac{1}{1!}\frac{\lambda}{\mu} + \frac{1}{2!}\left(\frac{\lambda}{\mu}\right)^2 + \ldots + \frac{1}{N!}\left(\frac{\lambda}{\mu}\right)^N}}$$

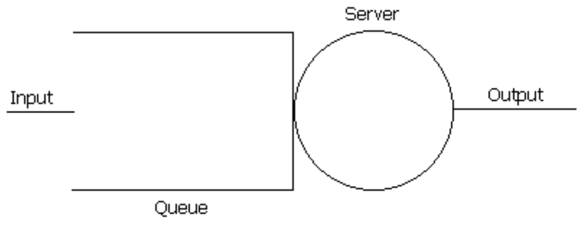
# Queueing Analysis: Packet Switching Delay

- Four types of delay at each hop
  - nodal processing delay: check errors & routing
  - queueing: time waiting for its turn at output link
  - transmission delay: time to pump packet onto a link at link speed
  - propagation delay: router to router propagation
- The focus is on queueing and transmission delay

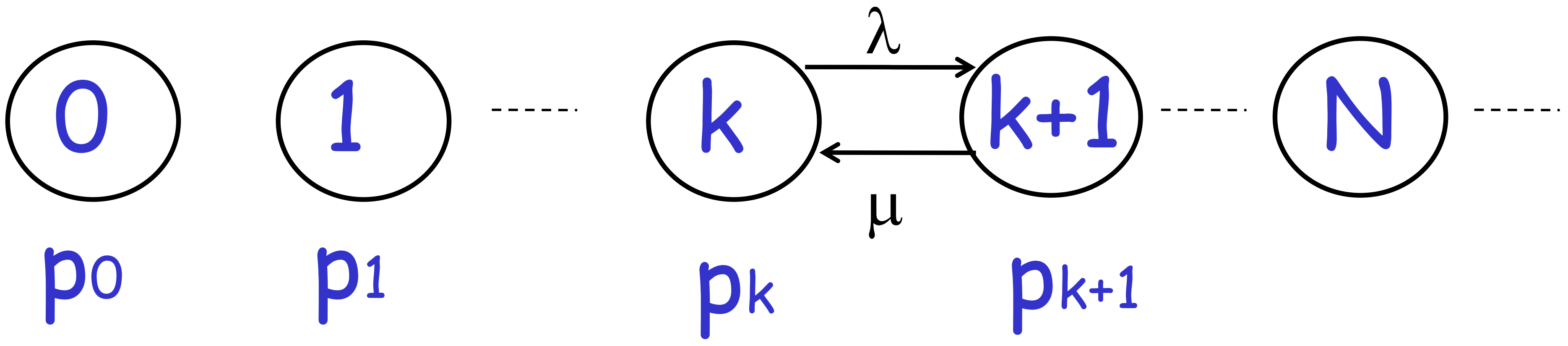


system state?

# Packet Switching Delay



system state: #packets in queue



at equilibrium (time reversibility) in one unit time:

#(transitions k → k+1) = #(transitions k+1 → k)

$$\boxed{p_k \lambda = p_{k+1} \mu}$$

$$\boxed{\sum_{\mathrm{k=0}}^{\infty} p_k = 1}$$

$$\boxed{p_{k+1} = \tfrac{\lambda}{\mu} p_k = \left(\tfrac{\lambda}{\mu}\right)^{k+1} p_0 = \rho^{k+1} p_0}$$

$$\boxed{p_0 = 1 - \rho}$$

$$\boxed{\rho = \tfrac{\lambda}{\mu}}$$

## Summary: Queueing Theory

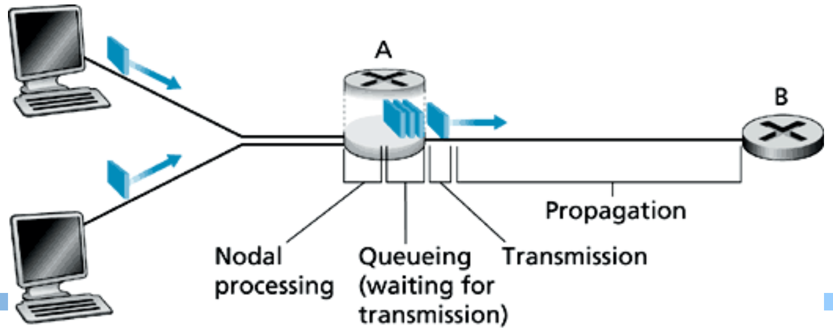
- Model system state
- Introduce state transition diagram
- Focus on equilibrium: state trend neither growing nor shrinking

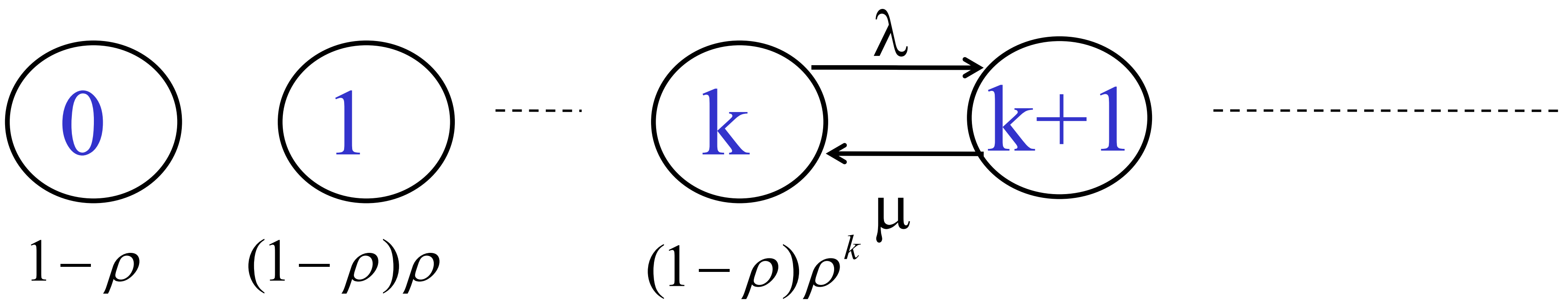
## Example

- Assume requests (packets) come in at a rate of one request per 50 ms $\lambda = 1/50\text{ms} = 20/\text{s}$
- Each request (packet) takes on average 20 ms $1/\mu = 20$ ms, $\mu = 50/\text{s}$

- What is the fraction of time that the system is empty? $P_0$
- What is the chance that a packet newly arrived needs to wait for 3 early packets? $P_3$

# Analysis of Delay (cont’)





- Average queueing delay:

$$\sum_{k=0}^{\infty} p_k \cdot k \cdot \frac{1}{\mu} = \sum_{k=0}^{\infty} \rho^k \,(1-\rho)\, k \frac{1}{\mu}$$

- Transmission delay: $\boxed{S = \frac{1}{\mu}}$

- Queueing + transmission:

## Delay

$$\boxed{\rho = \frac{\lambda}{\mu}}$$

$$\boxed{S = \frac{1}{\mu}}$$

$$\boxed{\text{average queueing delay: } w = S\frac{\rho}{1-\rho}}$$

$$\boxed{queueing + trans = S\frac{\rho}{1-\rho} + S = S\frac{1}{1-\rho}}$$

For a demo of M/M/1, see:
http://www.dcs.ed.ac.uk/home/jeh/Simjava/queueing/mm1_q/mm1_q.html

# Queueing Delay as a Function of Utilization

Assume:

R = link bandwidth (bps)

L = packet length (bits)

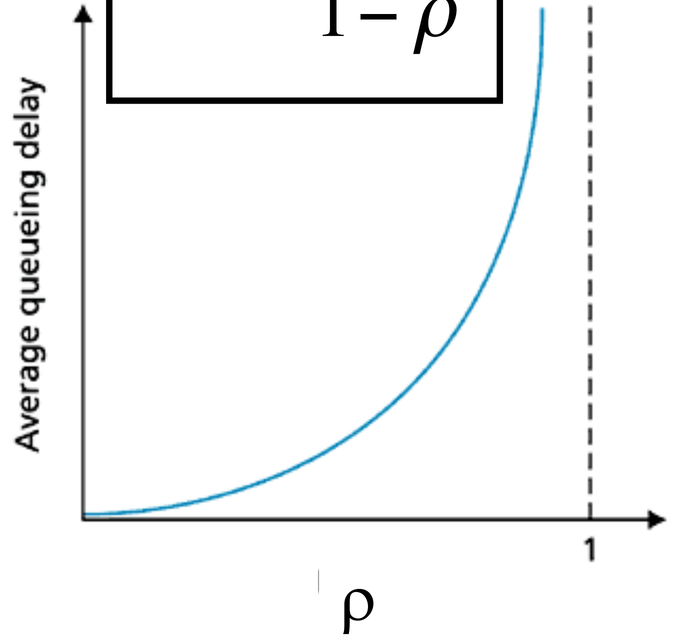
S = L / R

a = average packet arrival rate (pkt/sec)

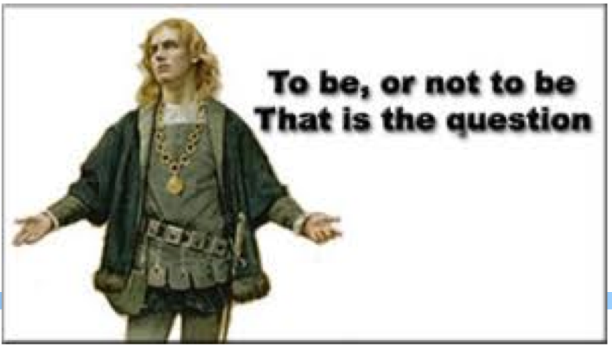
$$utilization: \ \rho = \frac{a}{1/S} = aS$$

$$w = S\frac{\rho}{1-\rho}$$

- ρ ~ 0: average queueing delay small
- ρ -> 1: delay becomes large
- ρ > 1: more "work" arriving than can be serviced, average delay infinite !

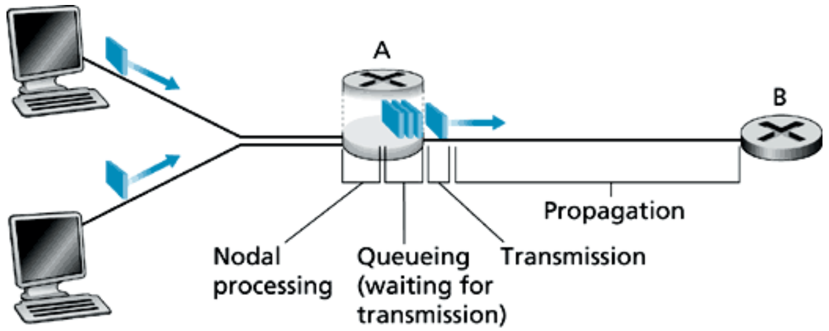


# Statistical Multiplexing



A simple model to compare bandwidth efficiency of
- reservation/dedication (aka circuit-switching) vs
- no reservation (aka packet switching)

setup
- a single bottleneck link with service rate $\mu$
- $n$ flows; each flow has an arrival rate of $\lambda/n$



- no reservation: all arrivals into the single link, the queueing delay + transmission delay:

$$\boxed{S\frac{1}{1-\rho}}$$

- reservation: each flow uses its own reserved (sub)link with rate $\mu$ /n, the queueing delay + transmission delay:

For each flow i:

$$\rho_i = \frac{\lambda/n}{\mu/n} = \rho \quad \Rightarrow \quad \boxed{\bigcirc\!\!\!\!n\; S\frac{1}{1-\rho}}$$

$$S_i = \frac{1}{\mu/n} = nS$$

## Summary: Packet Switching vs. Circuit Switching

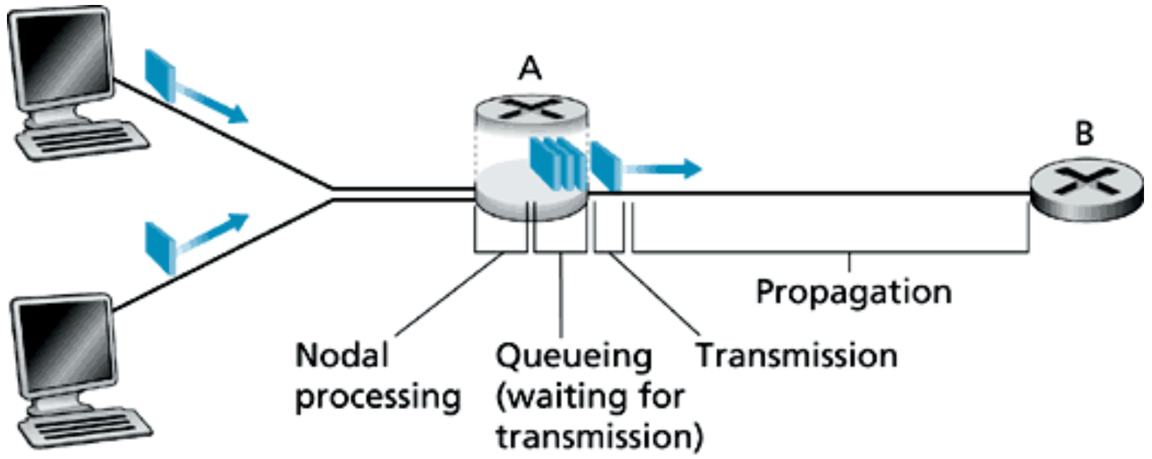
- Advantages of packet switching over circuit switching
  - most important advantage of packet-switching over circuit switching is statistical multiplexing, and therefore more efficient bandwidth usage
- Disadvantages of packet switching
  - potential congestion: packet delay and high loss
    - protocols 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
- *A taxonomy of communication networks*
  - circuit switched networks
  - packet switched networks
  - circuit switching vs. packet switching
  - *datagram and virtual circuit packet switched networks*

## A Taxonomy of Packet-Switched Networks According to Routing

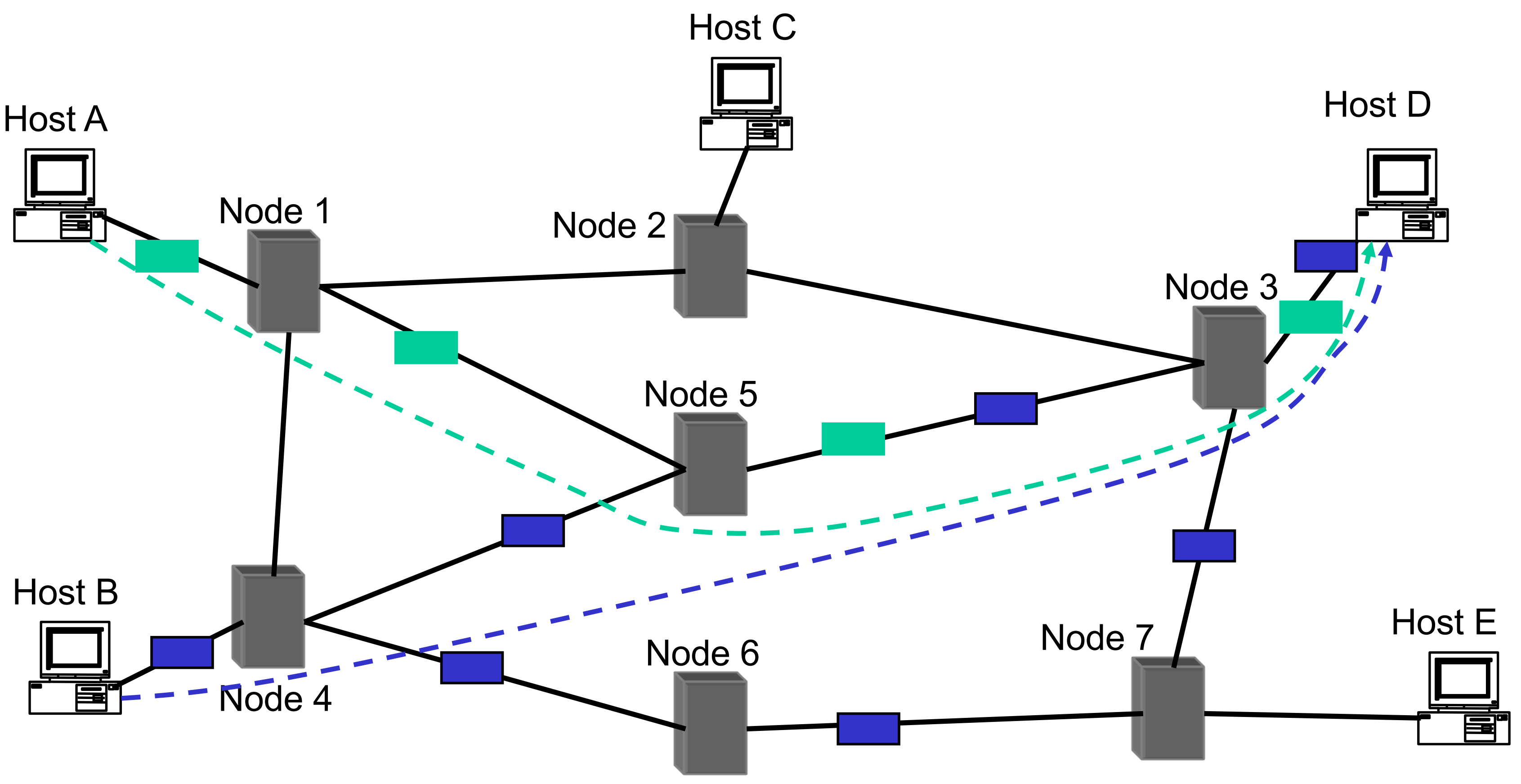
- Two types of packet switching
  - datagram network
    - each packet of a flow is switched **independently**
  - virtual circuit network:
    - all packets from one flow are sent along a **pre-established** path (= virtual circuit)



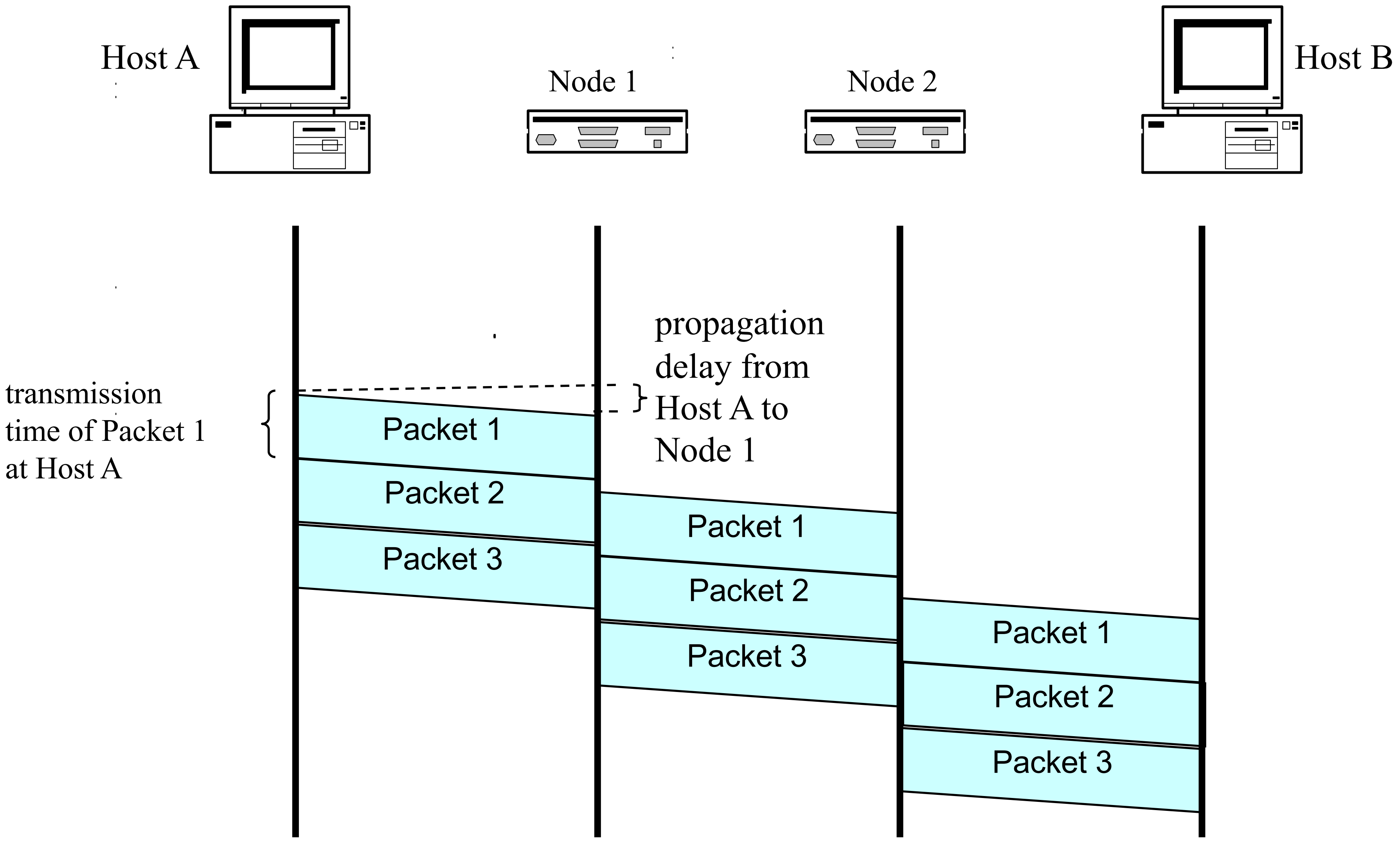
## Datagram Packet Switching

- Commonly when we say packet switching we mean datagram switching
- Example: IP networks
- Each packet is independently switched
  - each packet header contains *complete destination address*
  - receiving a packet, a router looks at the packet's destination address and *searches* its current routing table to determine the possible next hops, and pick one
- Analogy: postal mail system

# Datagram Packet Switching



# Timing Diagram of Datagram Switching



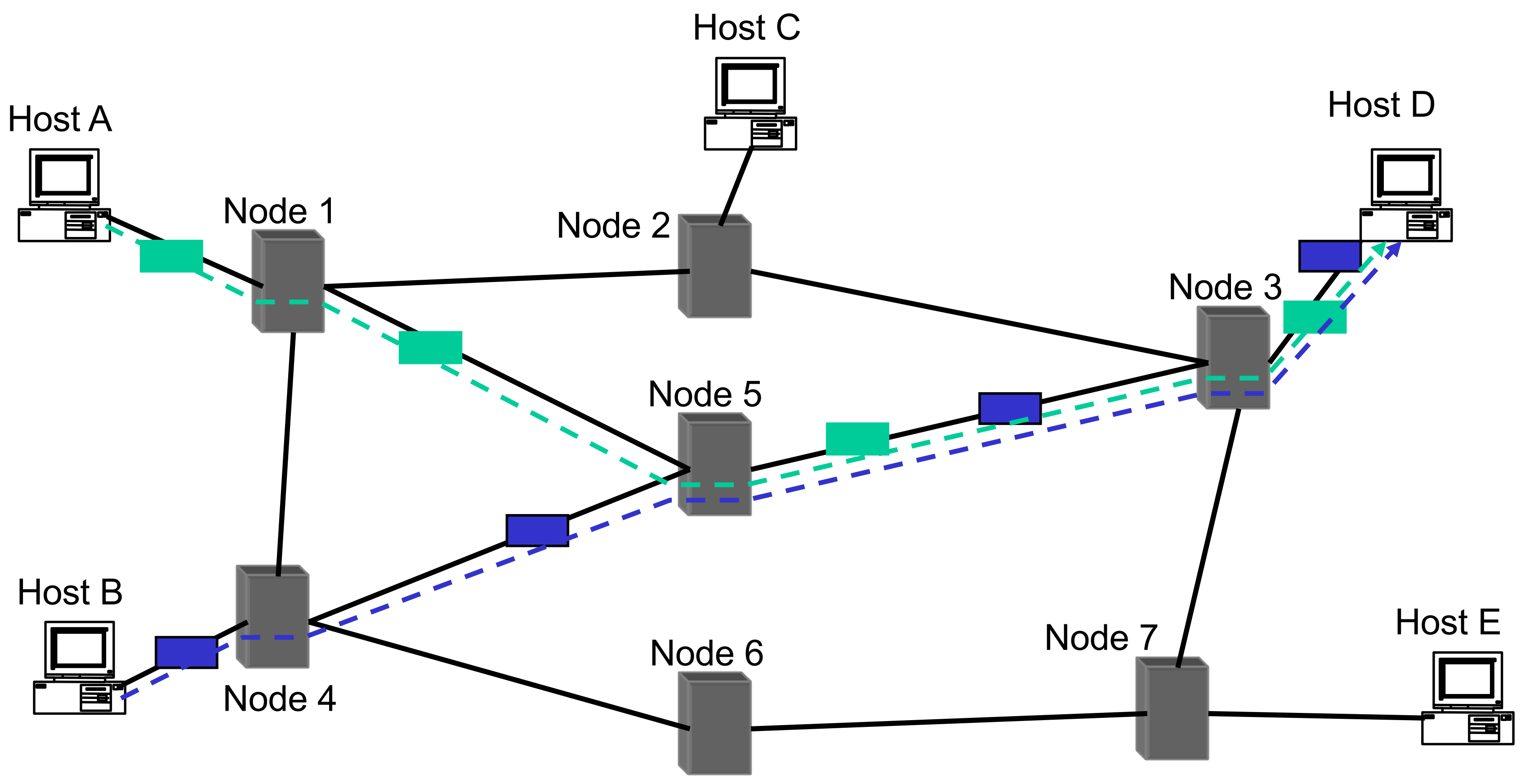
# Virtual-Circuit Packet Switching

- Example: Multiple Label Packet Switching (MPLS) in IP networks

- Hybrid of circuit switching and datagram switching
  - fixed path determined at *virtual circuit setup time*, remains fixed thru flow
  - Implementation:
    1. each packet carries a short local, tag (virtual-circuit (VC) #); tag determines next hop

| Incoming VC# | Outgoing Interface | QoS |
|---|---|---|
| 12 | 2 | |
| 16 | 3 | |
| 20 | 3 | |
| … | | |

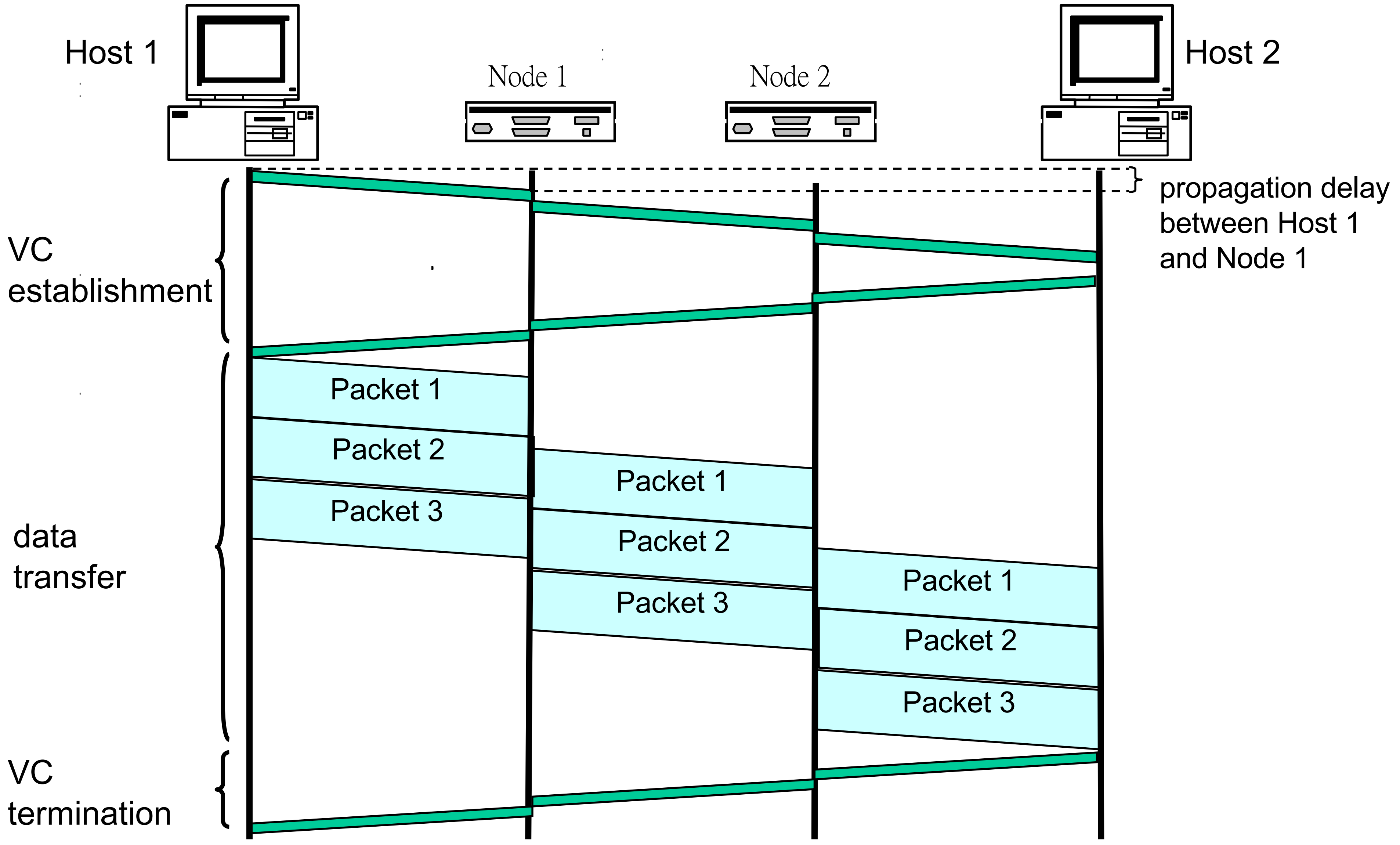
## Virtual-Circuit Switching



## Virtual-Circuit Packet Switching

- Three phases
  1. VC establishment
  2. Data transfer
  3. VC disconnect

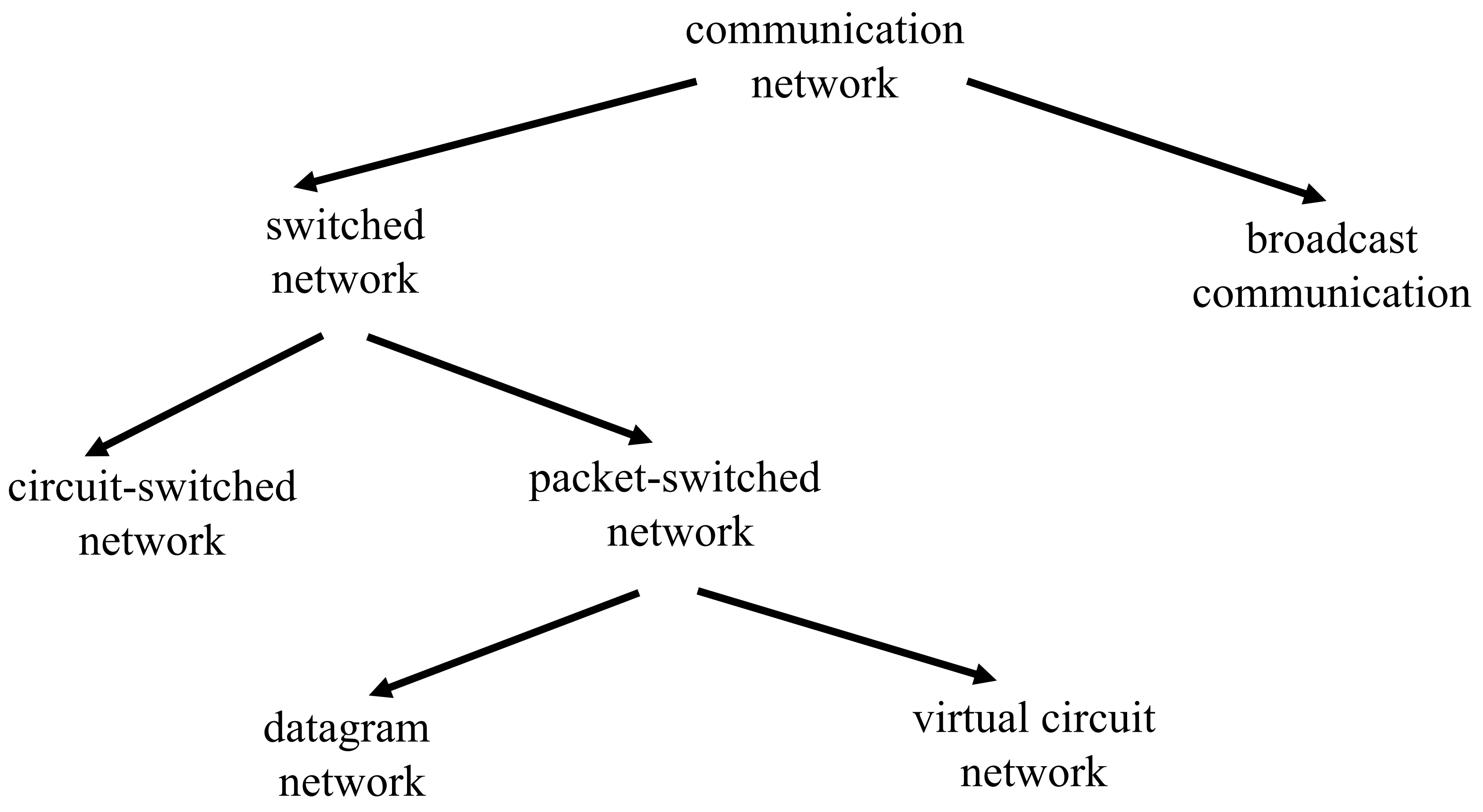
## Timing Diagram of Virtual-Circuit Switching



## Discussion: Datagram Switching vs. Virtual Circuit Switching

- What are the benefits of datagram switching over virtual circuit switching?
- What are the benefits of virtual circuit switching over datagram switching?

## Summary of the Taxonomy of Communication Networks



## Summary of Progress

- We have seen the hardware infrastructure, the basic communication scheme, a next key question is how to develop the software system.

## Summary of Progress

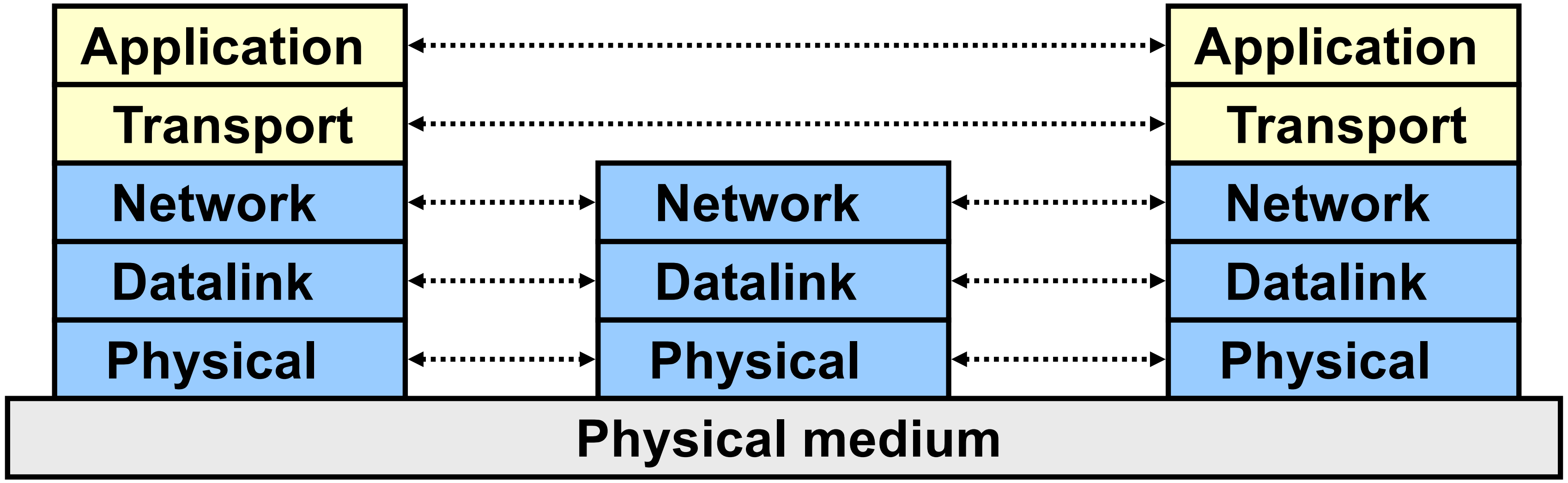
- We have seen the hardware infrastructure, the basic communication scheme, a next key question is how to develop the software system.

## Outline

- Admin. and recap
- A taxonomy of communication networks
- 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
    - infrastructure

Dealing with complex systems:

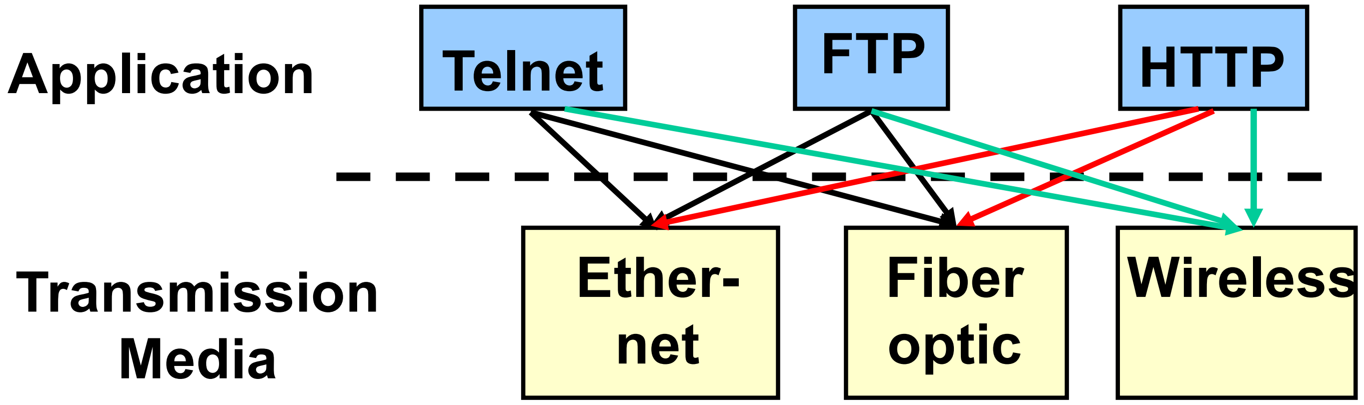
explicit structure allows identification of the relationship among a complex system's pieces

- layered reference model for discussion

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

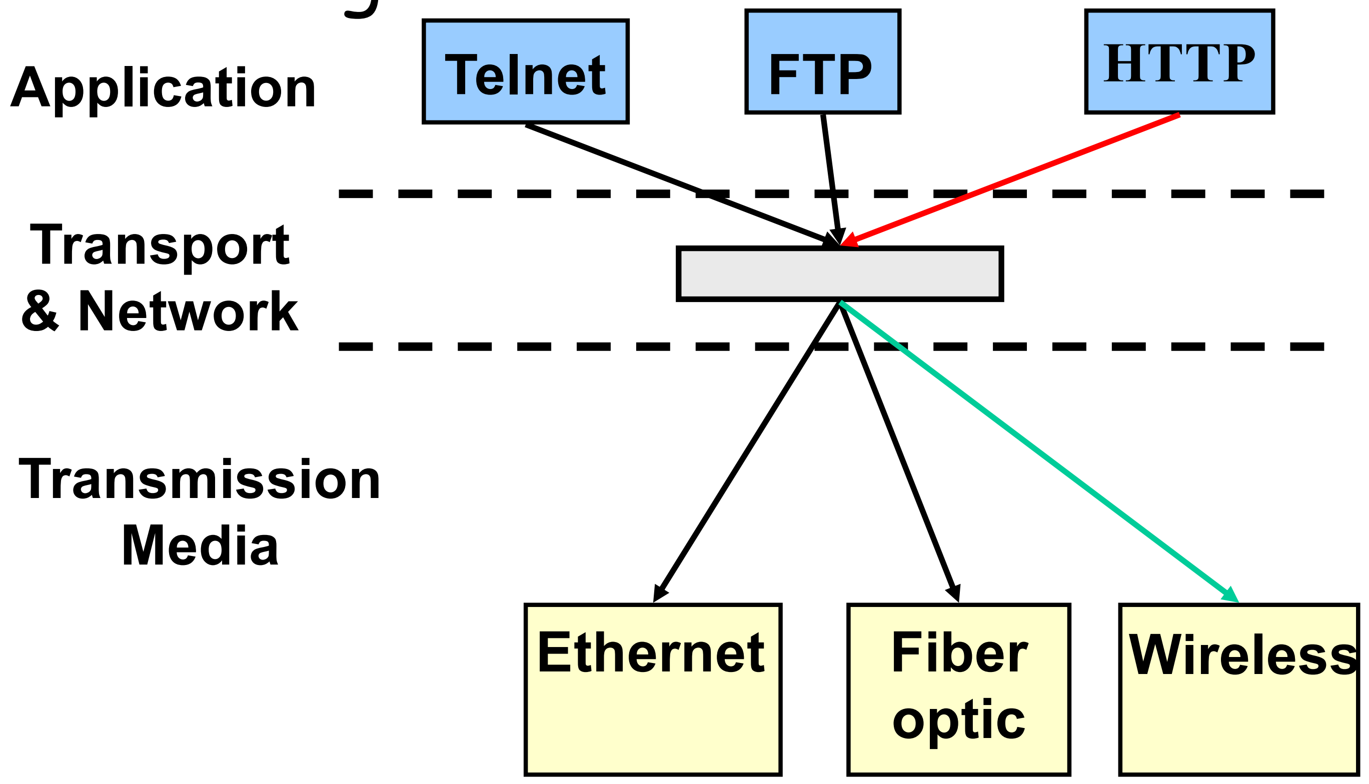
## An Example: No Layering



- 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 network technologies

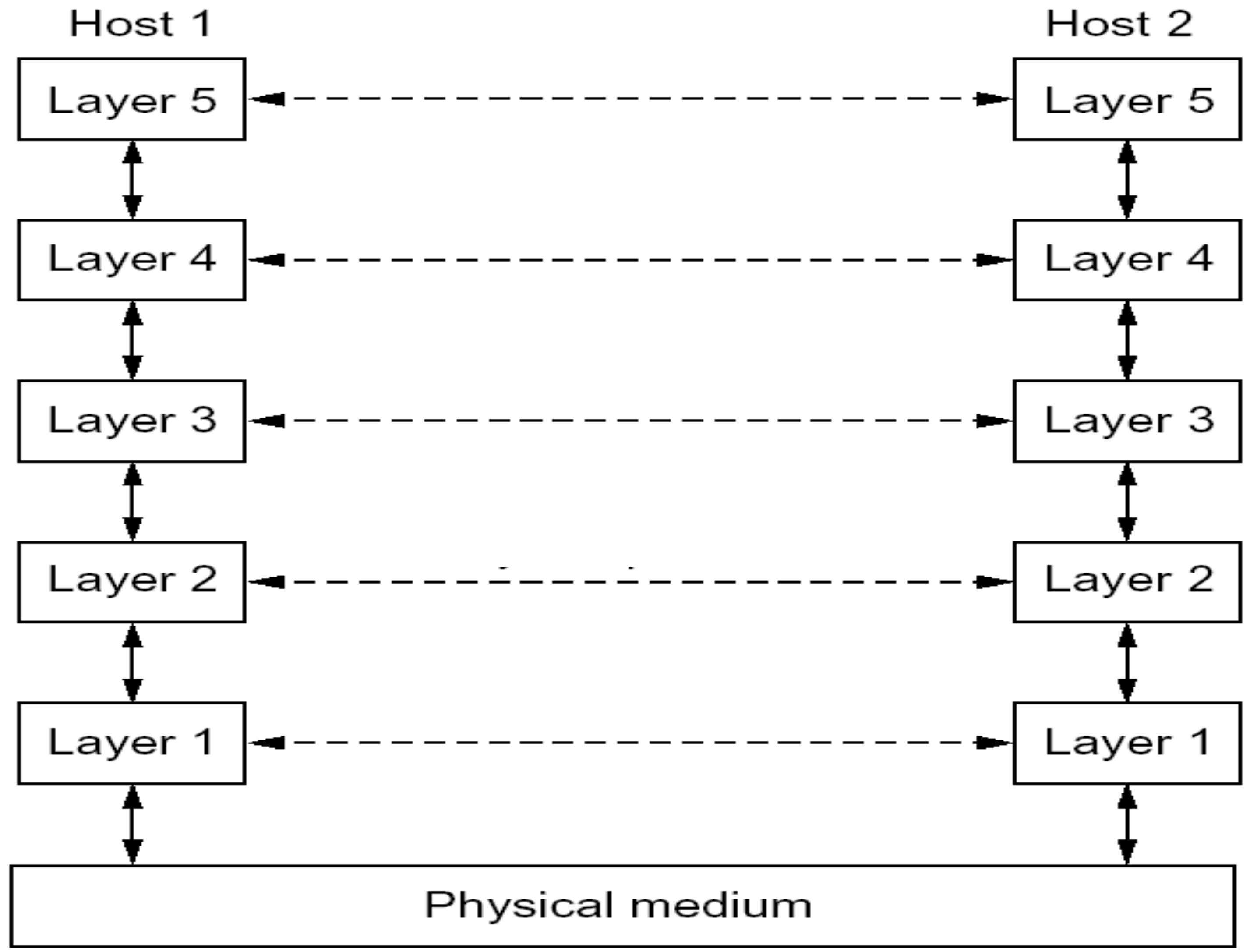


# ISO/OSI Concepts

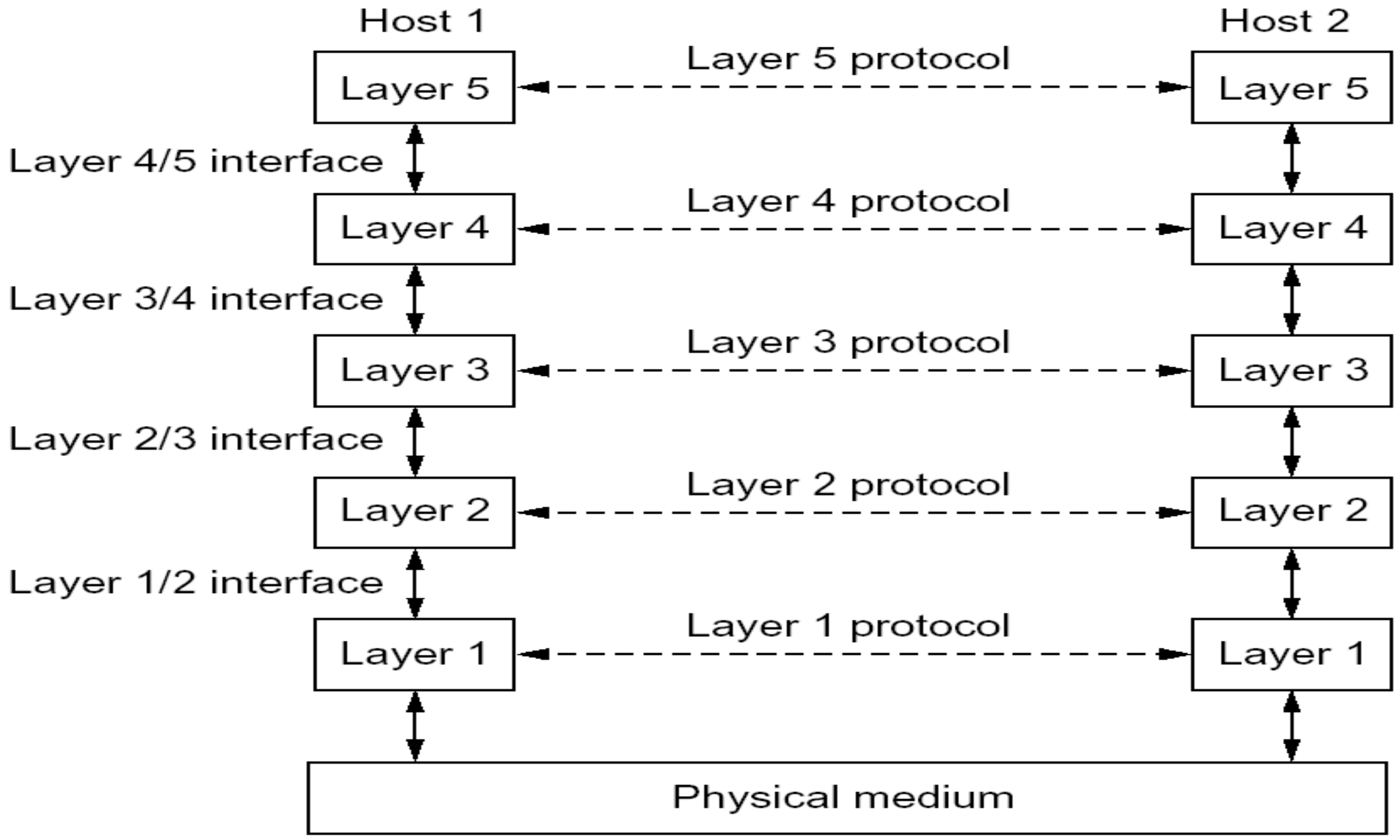
- ISO - International Standard Organization
- OSI - Open System Interconnection

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

## An Example of Layering



## An Example of Layering

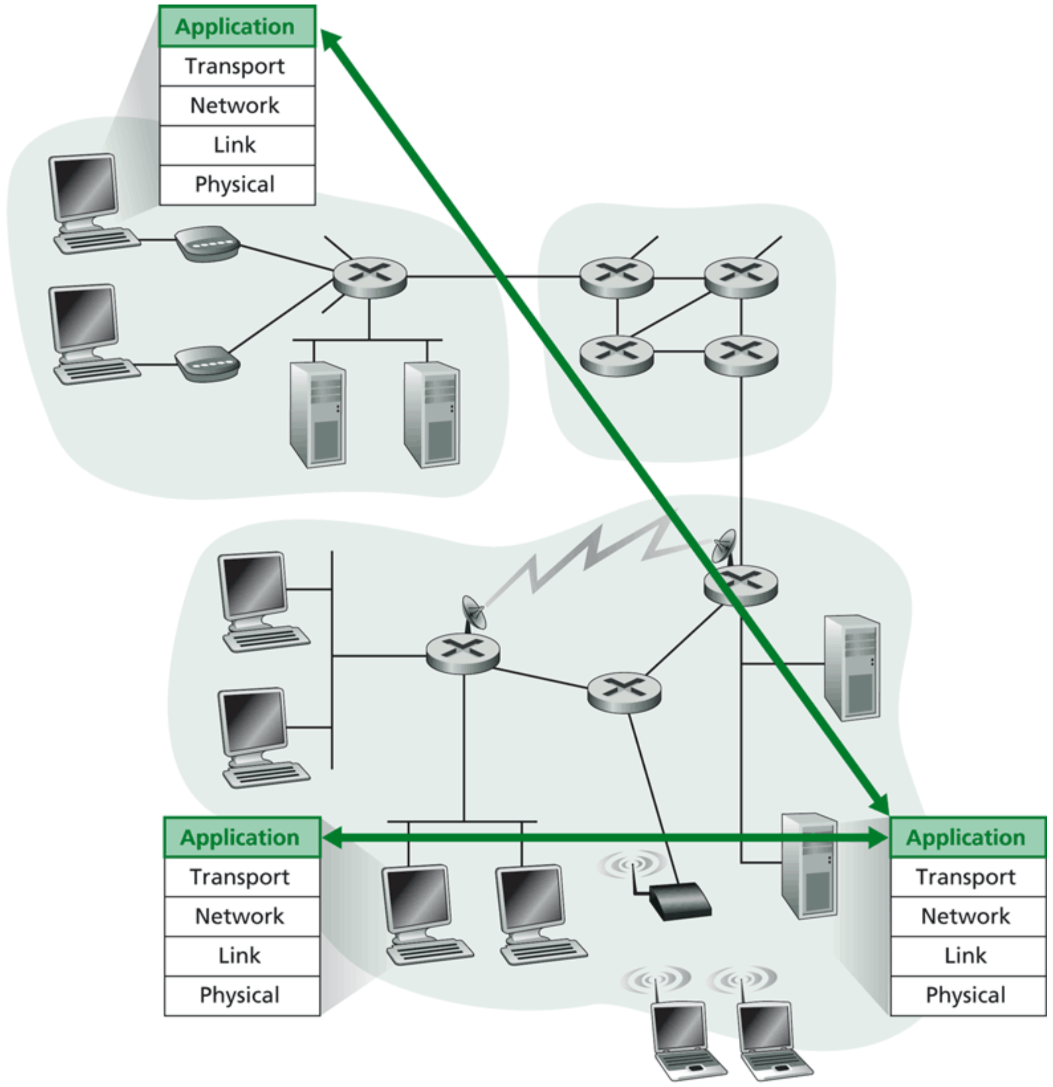


## Layering -> *Logical* Communication

E.g.: application

- provide services to users

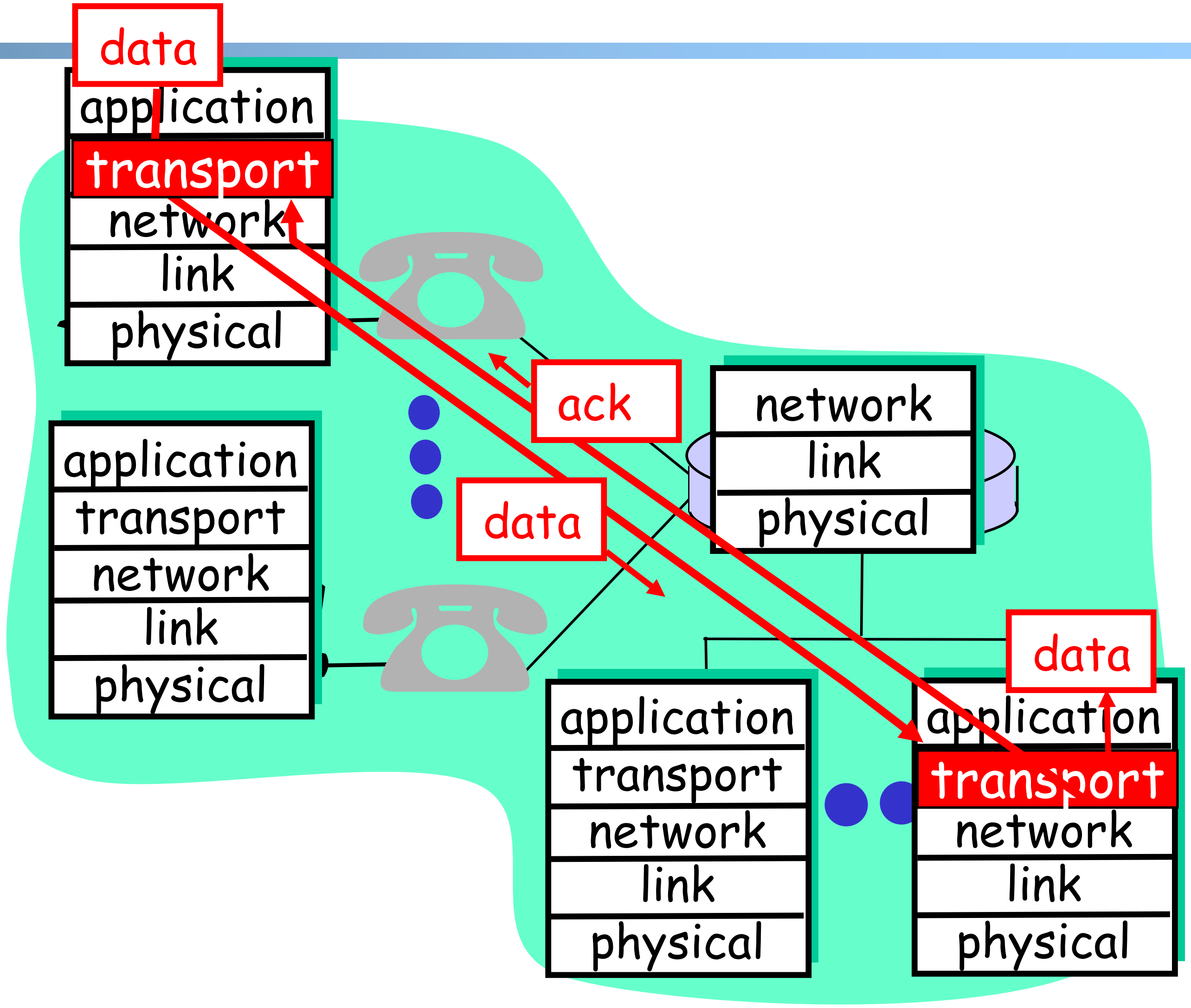
- application protocol:
  - send messages to peer
  - for example, HELO, MAIL FROM, RCPT TO are messages between two SMTP peers



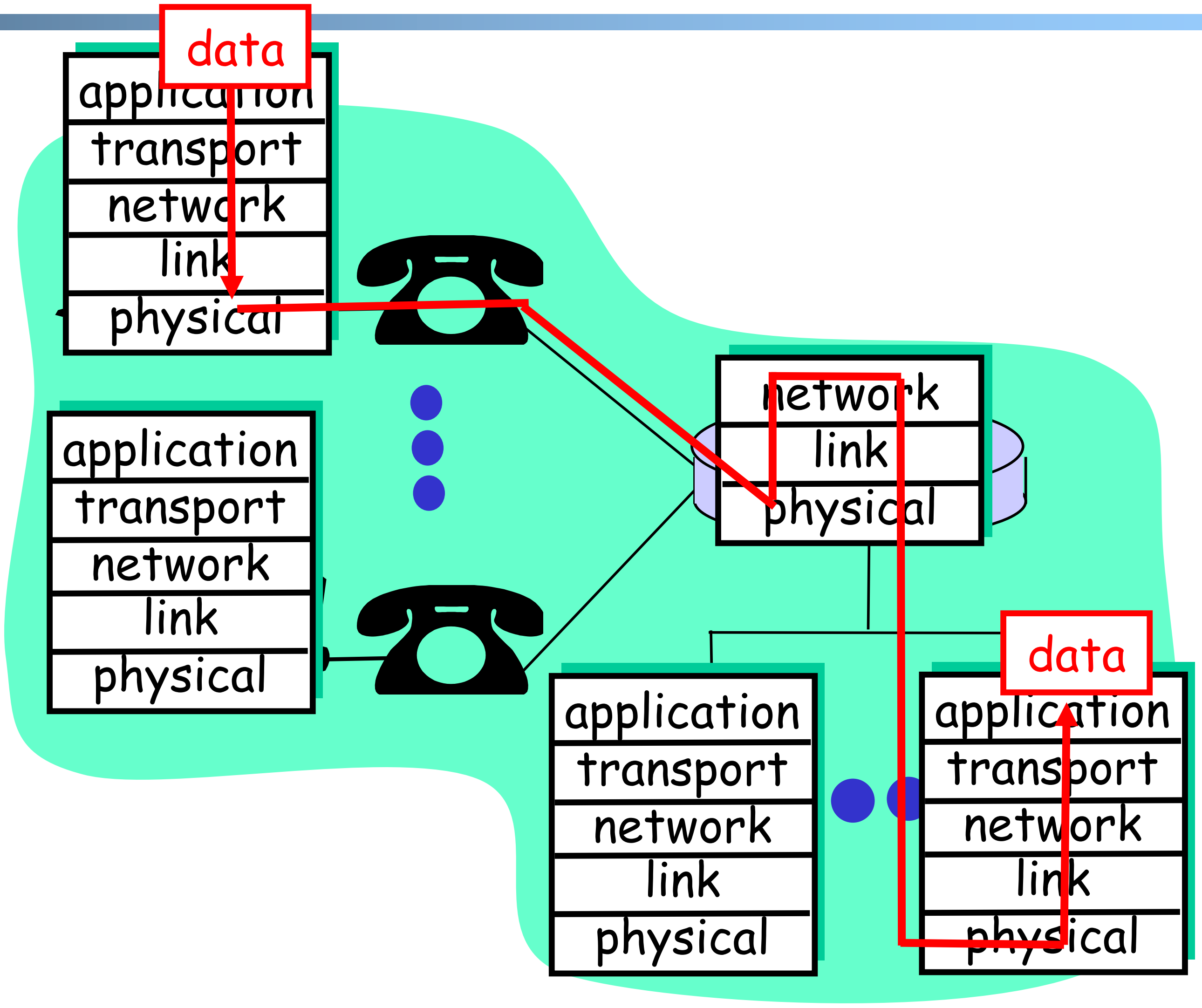
## Layering: *Logical* Communication

E.g.: transport

- Trans. msg for app
- Transport protocol
  - add control info to form "segment"
  - send segment to peer
  - wait for peer to ack receipt; if no ack, retransmit



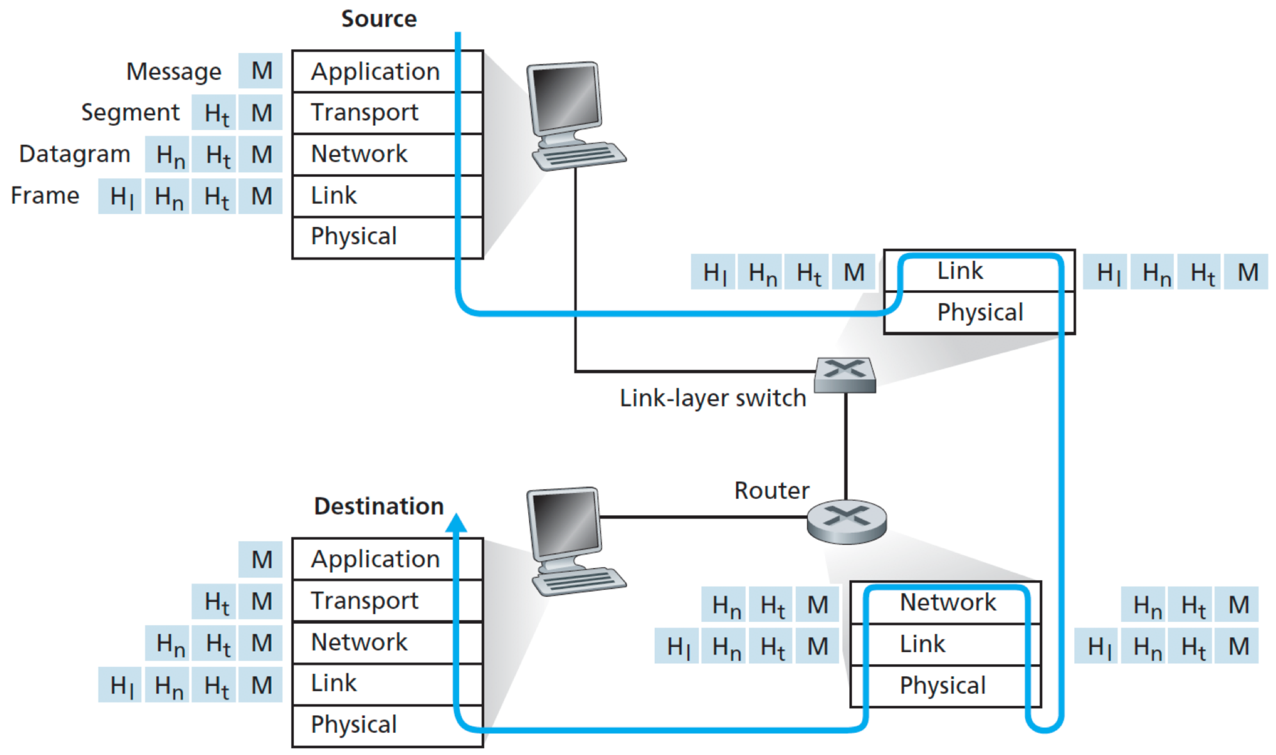
# Layering: *Physical* Communication



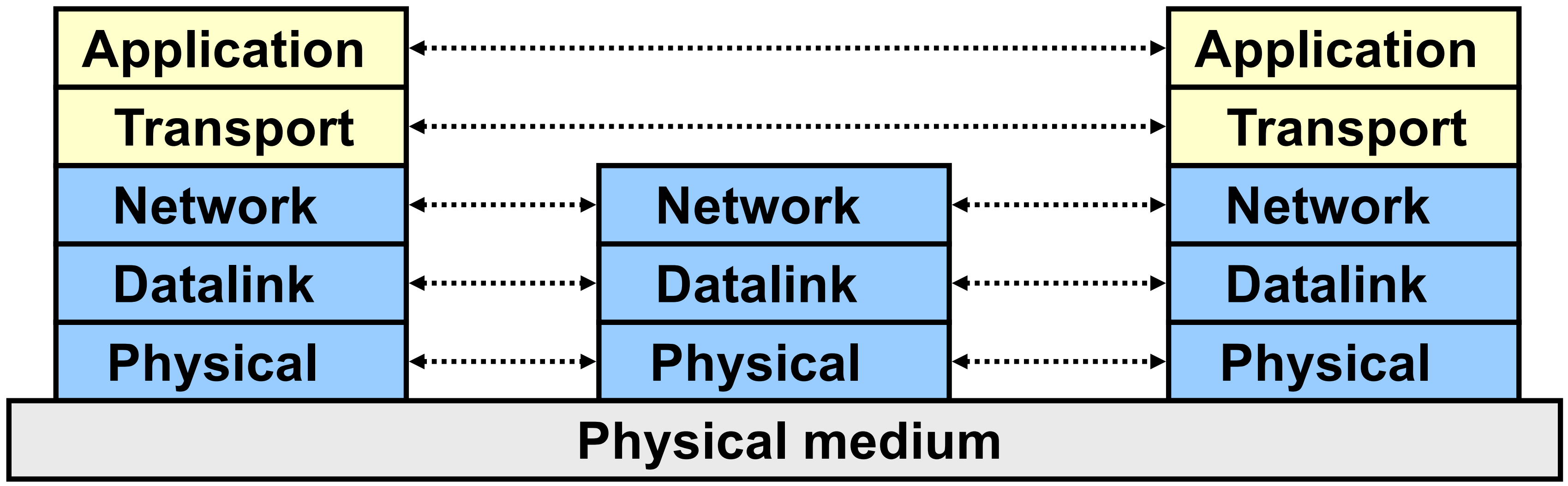
# Protocol Layering and Meta Data

Each layer takes data from above

- adds header (meta) information to its peer to create new data unit
- passes new data unit to layer below

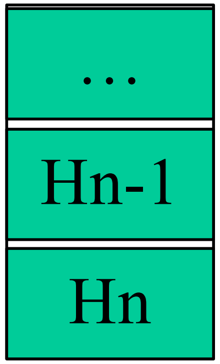


## Packet as a Stack in a Layered Architecture

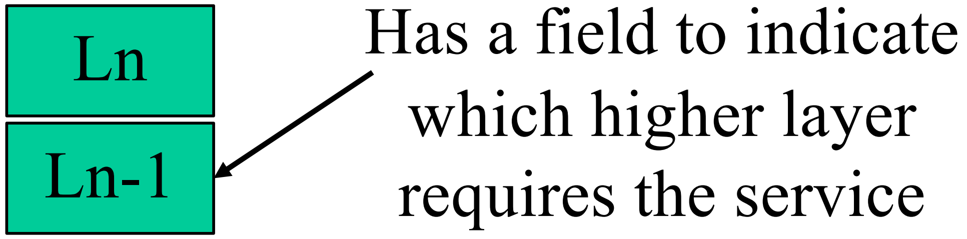


## Some Implications of Layered Architecture

- A packet as a stack container



- Each layer needs multiplexing and demultiplexing to serve layer above


Ln
Ln-1
Has a field to indicate which higher layer requires the service

## Key design issue:

How do you *divide* functionalities among the layers?

## Outline

- Admin. and recap
- A taxonomy of communication networks
- Layered network architecture
  - what is layering?
  - why layering?
  - ➢ *how to determine 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.*

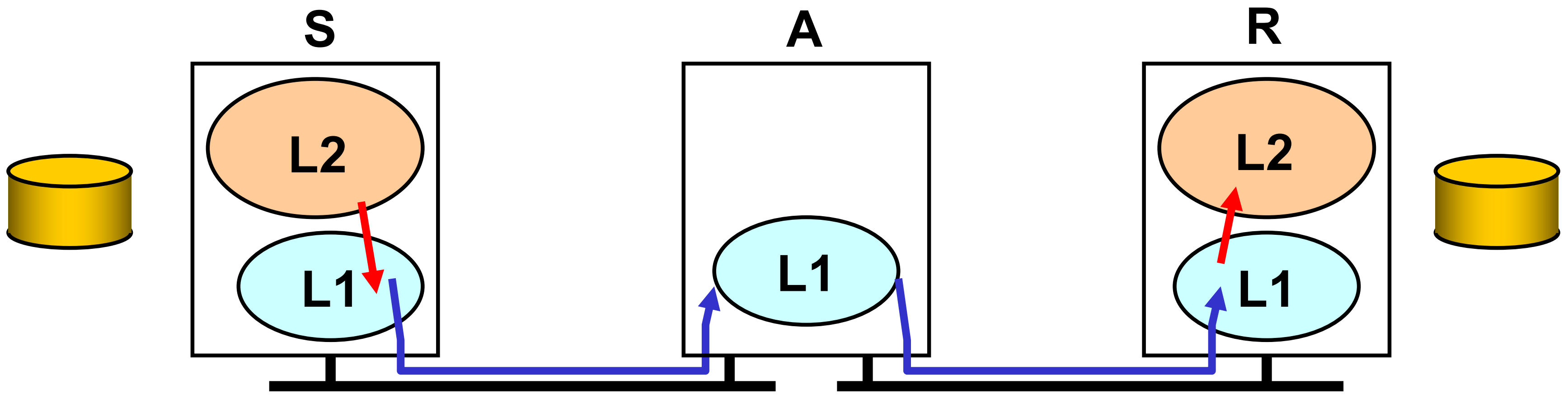
J. Saltzer, D. Reed, and D. Clark, 1984

## What does the 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

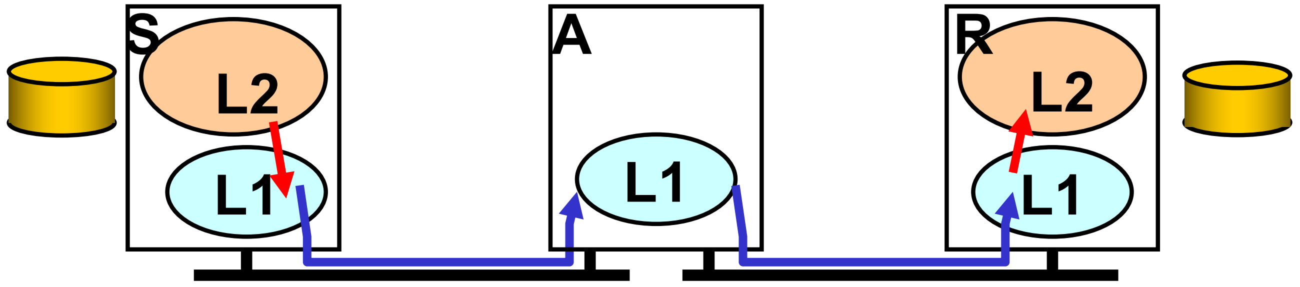
## Example: Where to Provide Reliability ?



- 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

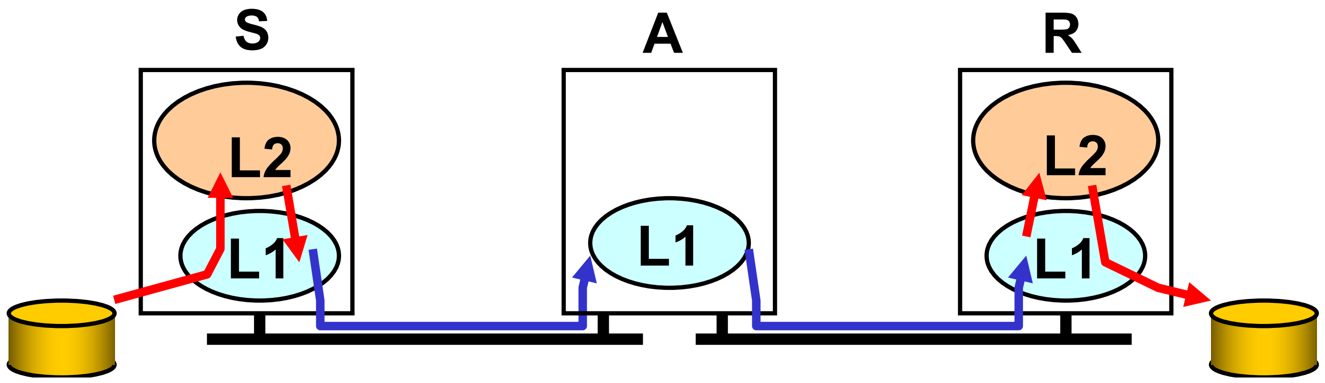
## What are Reasons for Implementing Reliability at Higher Layer ?

- The lower layer cannot completely provide the functionality
  - the receiver has to do the check anyway !
- 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



## Are There Reasons Implementing Reliability at Lower Layer ?

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



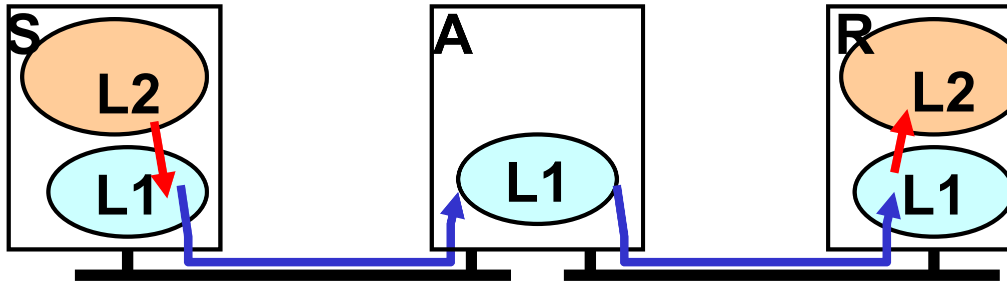
## Summary: End-to-End Arguments

- 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

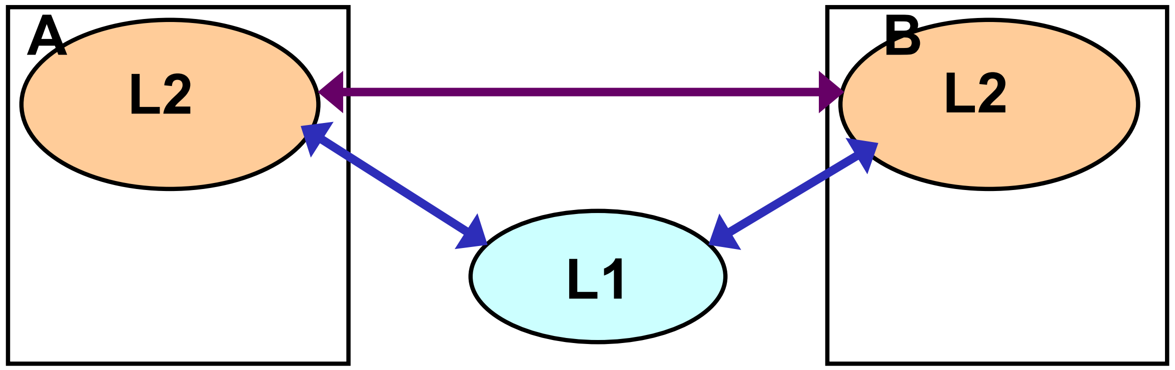
- We used reliability as an example

- 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)
  - congestion control (e.g., not to overwhelm network links or receiver)



## Example

- Consider the presence service in a social networking system: shows which contacts are online (e.g., skype)
  - implementing by end user's host app or through a third party service?



## Challenges

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

$$\begin{tikzcd}\text{performance} \arrow[r, leftrightarrow, "\text{end-to-end arguments}"] & \begin{array}{c}\text{reuse, interoperability,}\\ \text{implementation effort}\\ \text{(apply layering concepts)}\end{array}\end{tikzcd}$$

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

## Discussion: Limitations of Layered Architecture