Advanced Research Topics in Networked Systems

Qiao Xiang, Congming Gao, and **Lu Tang**

https://sngroup.org.cn/courses/ans-xmuf23/index.shtml

9/12/2023

This deck of slides are heavily based on CPSC 433/533 at Yale University, by courtesy of Dr. Y. Richard Yang.

Outline

- *Administrative trivia's*
- What is a network protocol?
- A brief introduction to the Internet: past and present
- Summary

Personnel

- Instructor
  - Qiao Xiang, qiaoxiang@xmu.edu.cn
    - office hours: by appointment
  - Congming Gao, email
  - Lu tang, email

- Teaching assistant
  - Rulan Yang, email
  - Jinghui Jiang, email

Instructor: Qiao Xiang

- Joined XMU as a professor in January 2021
- Research: Computer Networks and Systems
- Previously,
  - Research assistant professor, Yale University, US., 2019-2020
  - Postdoctoral fellow, Yale University, US. 2016-2018
  - Postdoctoral fellow, McGill University, Canada, 2014-2015
  - Ph.D. in Computer Science, Wayne State University, US, 2014
  - B.E. in Information Security and B.Econ., NKU, 2007

Instructor: Congming Gao

- Joined XMU as a professor this January
- Research: Computer Networks, Computer Systems
- Previously,
  - Research assistant professor, Yale University, US., 2019-2020
  - Postdoctoral fellow, Yale University, US. 2016-2018
  - Postdoctoral fellow, McGill University, Canada, 2014-2015
  - Ph.D. in Computer Science, Wayne State University, US, 2014
  - B.E. in Information Security and B.Econ., NKU, 2007

Instructor: Lu Tang

- Joined XMU as a professor this January
- Research: Computer Networks, Computer Systems
- Previously,
  - Research assistant professor, Yale University, US., 2019-2020
  - Postdoctoral fellow, Yale University, US. 2016-2018
  - Postdoctoral fellow, McGill University, Canada, 2014-2015
  - Ph.D. in Computer Science, Wayne State University, US, 2014
  - B.E. in Information Security and B.Econ., NKU, 2007

What are the Goals of this Course?

- Learn design principles and techniques of:
  - the Internet infrastructure (Internet service provider, data center, cloud)
  - large-scale networked systems

- How to do high-quality system research
  - go through the complete cycle of a research paper, including identifying a problem, making a proposal, iterate different designs, fast prototyping, comprehensive experiments, paper writing and even paper review and TPC meeting

Advanced Research Topics in Networked Systems vs. Advanced Network Technology

ARTNS:

- Bilingual:
  - English in slides / assignments / projects
  - Chinese in lecture / discussions
- More emphasis on system
- More emphasis on research
- Lab and project closely related to your own research area

What Do You Need To Do?

- Please go to the class website to fill out the class background survey
  - help us better understand your background
  - help us determine the depth, topics, and the details of lectures
  - suggest topics that you want to be covered (if you think of a topic later, please send me an email)

Workload

- Attendance (10%)
- 2 written assignments (5%+5%)
  - WA1: mock PC (review + discussion) (2 weeks)
  - WA2: distributed algorithms (2 weeks)
- 2 lab assignments (15%+15%)
  - LA1: P4 tutorial (3 weeks)
    - bmv2 as a baseline, real switch for bonus
  - LA2: experiment (3 weeks)
    - a systematic experiment study including methodology, dataset, figures and results analysis
    - the specifics of the experiment is decided by your advisor

Workload

- 2 class projects (20%+30%)
  - P1: reproducing via LLM (4 weeks)
    - reproduce one paper by prompt engineering ChatGPT
    - which paper to reproduce is decided by your advisor
  - P2: research paper (1-3 students per team, going through the whole 16-week semester)
    - the complete process of producing a 6 to 12-page research paper including proposal, design, implementation, experiment and writing
    - team formation and topic are decided by your advisor

How to Succeed in this Class?

- Engage in lectures
  - Questions are highly encouraged
- Push the instructors and your advisor
- Read references / online materials
- Apply the principles / techniques you learned in lectures to assignments and the project
- Do not procrastinate assignments and the project
  - For lab assignments and projects, follow the timeline of checkpoints to avoid the deadline panic

Grading

Class Participation	10%
Written Assignments	10%
Lab Assignments	30%
Course Projects	20%+30%

- Grades are important, but you do not need worry too much about them
- More important is what you realize/learn than the grades !!

Questions?

Outline

- Administrative trivia's
- *Course scope*

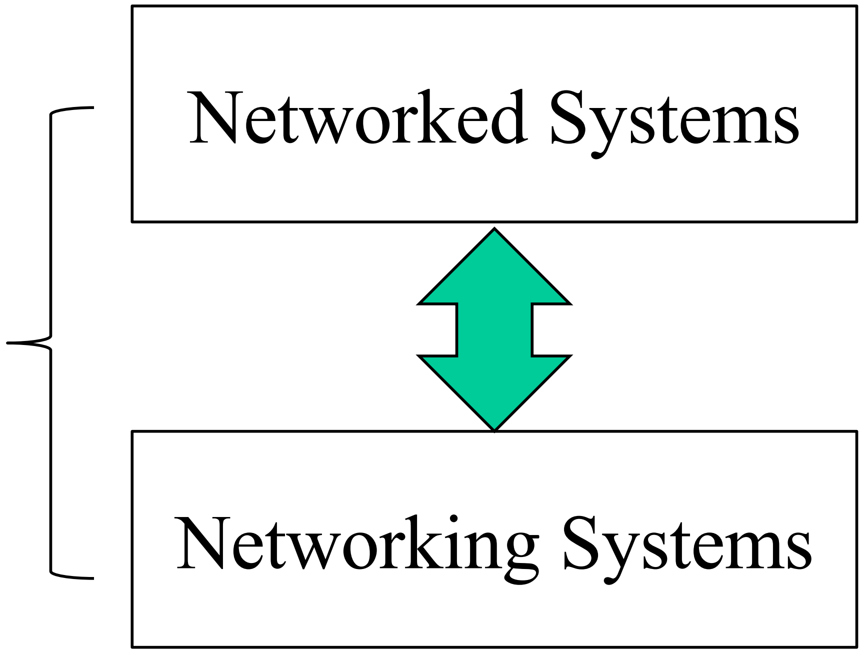
Scope: Networked Systems

- Networked systems: higher level computer system services built on top of networking services

- Networked systems include some of the most important, pervasive computer systems, e.g.,
  - Web, IM, multimedia
  - Large-scale data analytics (e.g., Hadoop, Spark, …)
  - Data store, pub/sub system (e.g., Raft, Kafka)
  - …

Networking Systems and Networked Systems

Share many common challenges and techniques

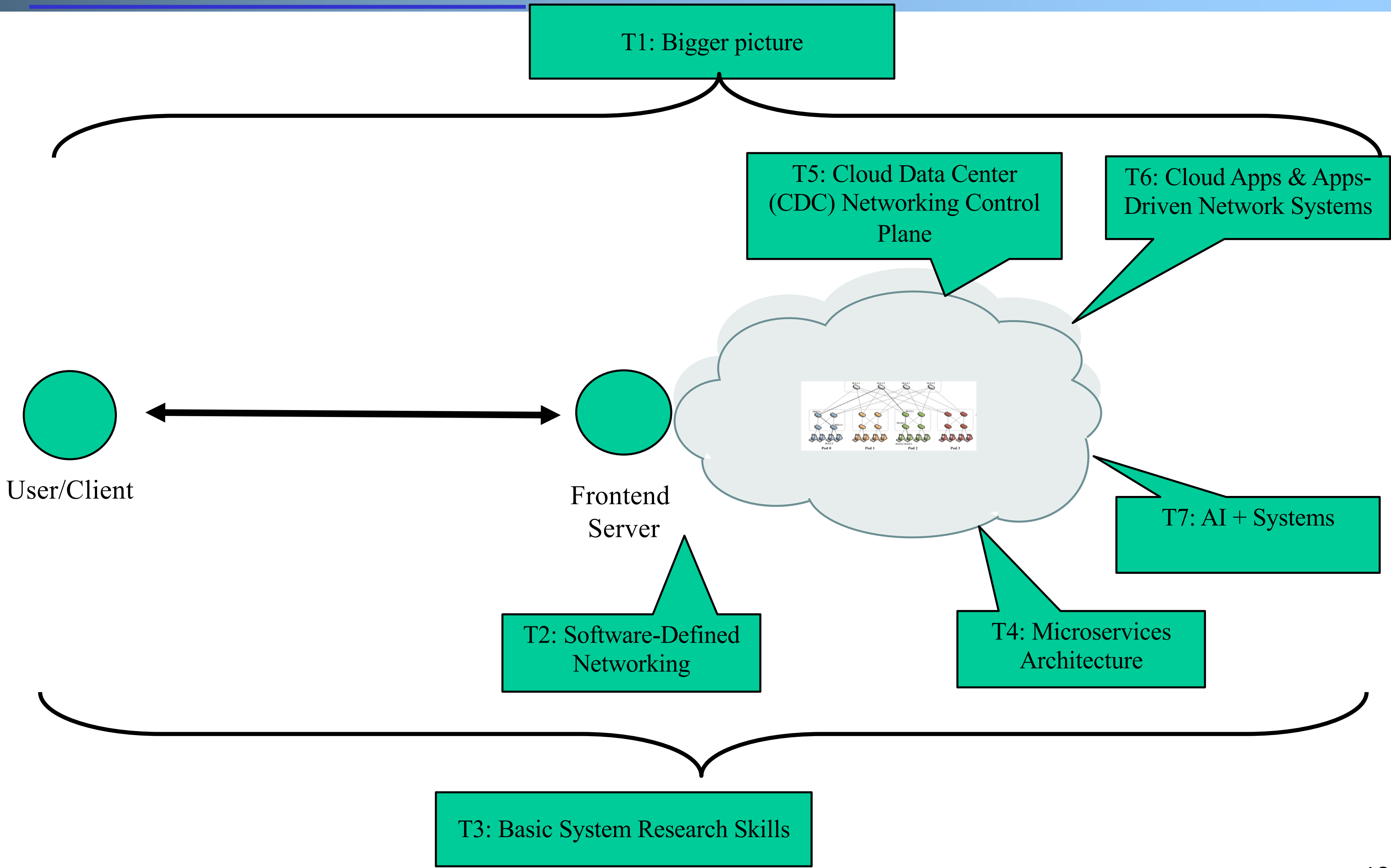


- Depend on networking systems capabilities
- Work around networking systems "limitations"

- Designed to support target networked systems

One goal of this class is to integrate both networking systems and networked systems

Networked Systems Topics We Plan to Cover



Common Design Goals

Global Rank	Domain	Monthly visits (billions)	Parent
1	Google.com	60.49	Alphabet Inc
2	Youtube.com	24.31	Alphabet Inc
3	Facebook.com	19.98	Facebook, Inc
4	Baidu.com	9.77	Baidu, Inc
5	Wikipedia.org	4.69	Wikimedia Foundation
6	Twitter.com	3.92	Twitter, Inc
7	Yahoo.com	3.74	Verizon Comm. Inc
8	pornhub.com	3.36	Mindgeek
9	Instagram.com	3.21	Facebook, Inc
10	xvideos.com	3.19	WGCZ Holding

- Scale to massive scale
- Reduce latency/tail latency
- Drive down costs
- Support large-scale resource sharing across users (multi-tenants), apps and servers
- Integrate next-generation hardware, e.g., accelerator, NIC, programmable switch, RDMA fabric, …
- Improve software development productivity

Fundamental Challenge Achieving Goals: Complexity

- **Complexity** arises from design strategies intended to create
  - Scalability
    - to handle the size and complexity of a system as a whole
  - Efficiency
    - to handle resource scarcity
  - Reliability
    - to handle component failures
  - Modularity
    - to allow reuse of components
  - Evolvability
    - to allow reuse of components in time

Complexity

"Developers who have worked at the small scale might be asking themselves why we need to bother when we could just use some kind of out-of the-box solution. For small-scale applications, this can be a great idea. We save time and money up front and get a working and serviceable application. The problem comes at larger scales—there are no off-the-shelf kits that will allow you to build something like Amazon... There's a good reason why the largest applications on the Internet a all bespoke creations: no other approach can create massively scalable applications within a reasonable budget."

http://www.evontech.com/symbian/55.html

Cloud Advantages

- Single administrative domain, no need to be compatible with outside world when inside itself
- Tiny round trip times (microseconds)
- Control both networks and end hosts
- Economy of scale—huge buyers that can drive market

Course Topic 1: Bigger Picture

- Network Physical Infrastructure
- Layering Architecture
  - Layering basic concepts and design principle
  - Application layer (optional)
    - RFC5246 (TLS1.2), RFC8446 (TLS1.3)
    - RFC7540 (HTTP/2)
    - RFC900 (QUIC)
    - HTTP/3 draft
  - Transport layer (optional)
    - RFC793 (TCP)
    - RFC5681, RFC5682 (TCP congestion control)
    - RFC8312 (Cubic), BBR, MP-TCP

Course Topic 2: Software-Defined Networking (SDN) Architecture

- OpenFlow SDN (OF1.3)
- Programmable ASIC: RMT, P4
- Network operating systems: Andromeda/Onix/Orion'21 (Google), SONIC (Microsoft), FBOSS (FB)

Course Topic 3: Research Skills

Basic principles and skills on

- using latex
- reading literature
- identifying problem and your own idea
- orchestrating the execution of a research idea
- writing your own paper
- preparing rebuttal
- publicizing your work
- attending conferences

Course Topic 4: Microservice Architecture

- Microservices basic concepts (containers, pods, deployment, service)
- Foundation of realizing containers: namespace, veth, ip, iptables
- Microservice architecture control plane:
  - Cluster control: consistent, replicated log protocol, etcd, k8s internals scheduler
  - Networking control (flannel; calico; weave; istio)

Course Topic 5: Cloud Data Center Networking Systems Design & Analysis (Part 1; Control)

- Classical cloud DC designs: Basic Clos, Facebook topology, Microsoft VL2, Google Juniper, DC evolving
- DC WAN control: Google B4'13/B4 and after'18, Microsoft SWAN
- Peering control: BGP background, Espresso, edge fabric

Course Topic 5: Cloud Data Center Networking Systems Design and Analysis (Part 2/Analysis)

- Data path analysis (HSA, VeriFlow, APKeep)
- Control-path analysis (minesweeper)
- Data center traffic analysis in Microsoft, Facebook

Course Topic 6: Cloud Applications & Application-Driven Networking Systems

- Data analytics (DA) programming model
  - MapReduce, Noria, Spark
  - Spark perf measurements (DA analysis)
- DC cluster compute scheduling
  - Delayed scheduling, YARN, Mesos, Borg, DRF
- Low latency, high-tput DC transport scheduling: DCTCP, RDMA, DCQCN; TIMELY; HPCC'20; On-Ramp'21
- Coflow scheduling: Coflow scheduling: Sincronia'18
- RPC scheduling: gRPC, eRPC, nanoPU
- File systems

Course Topic 7: AI + Systems

- High-performance machine learning systems
- Large language models
- LLM for X
  - X: networking, system, software engineering, …

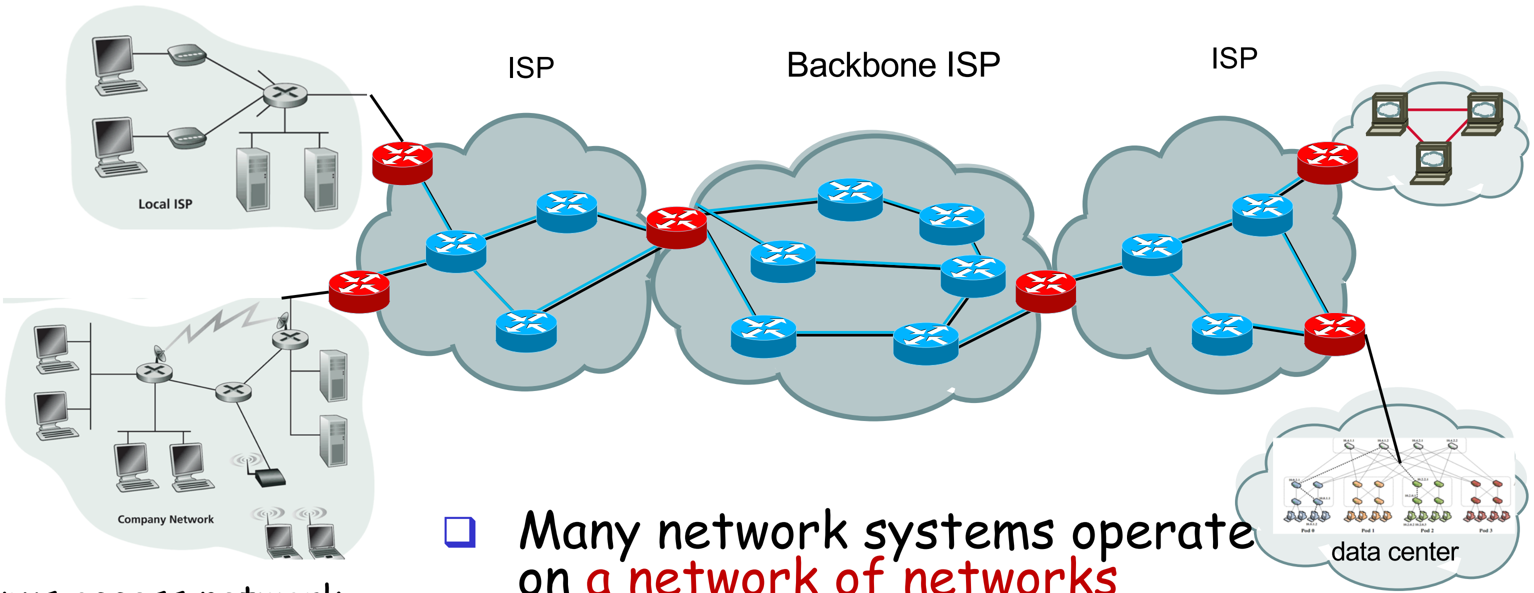
Outline

- Administrative trivia's
- Course scope
- *Bigger picture*
  - Physical infrastructure

Network System Physical Infrastructure

Residential access network

- Cable, Fiber, DSL, Wireless, Cellular

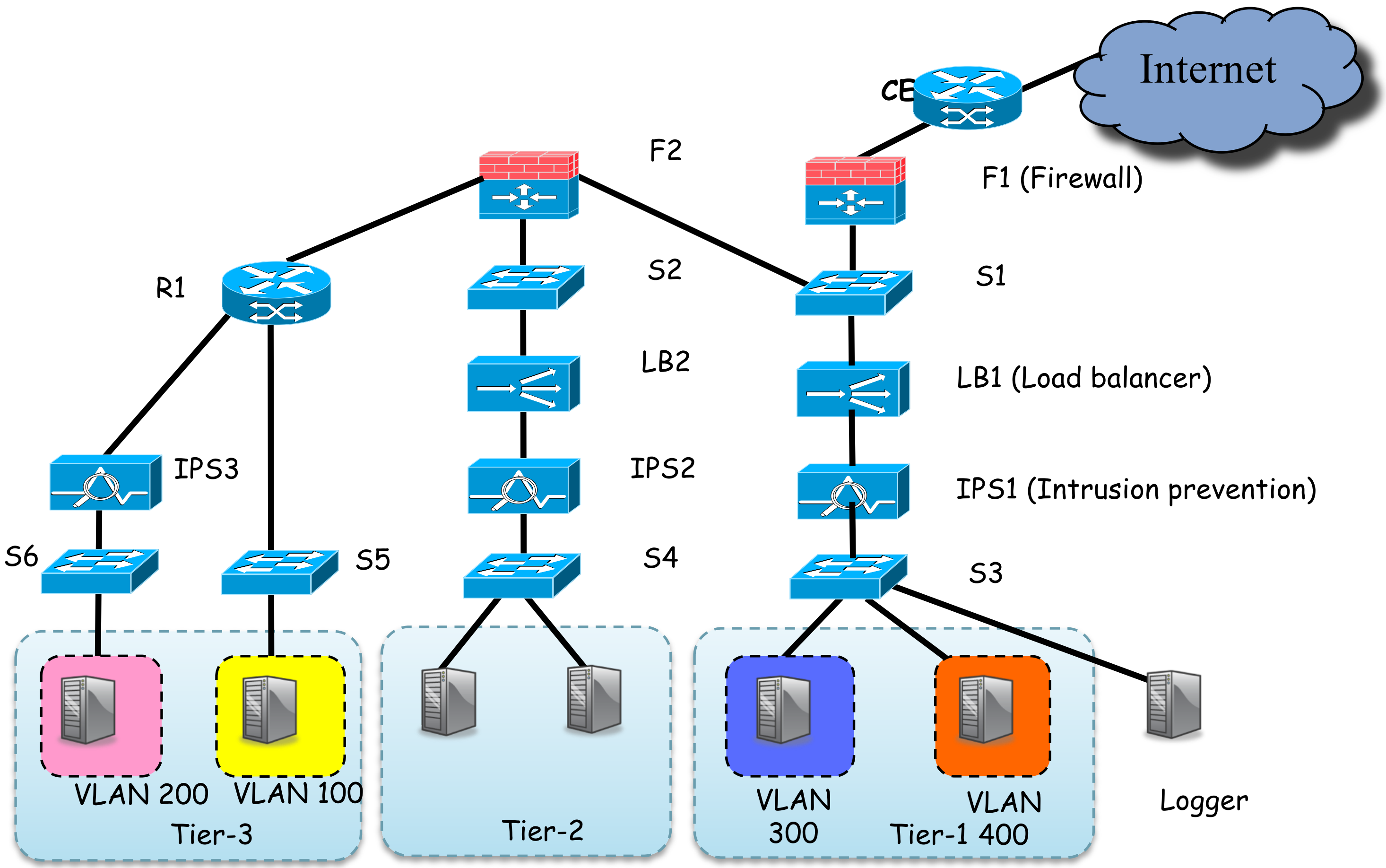


Campus access network, e.g.,

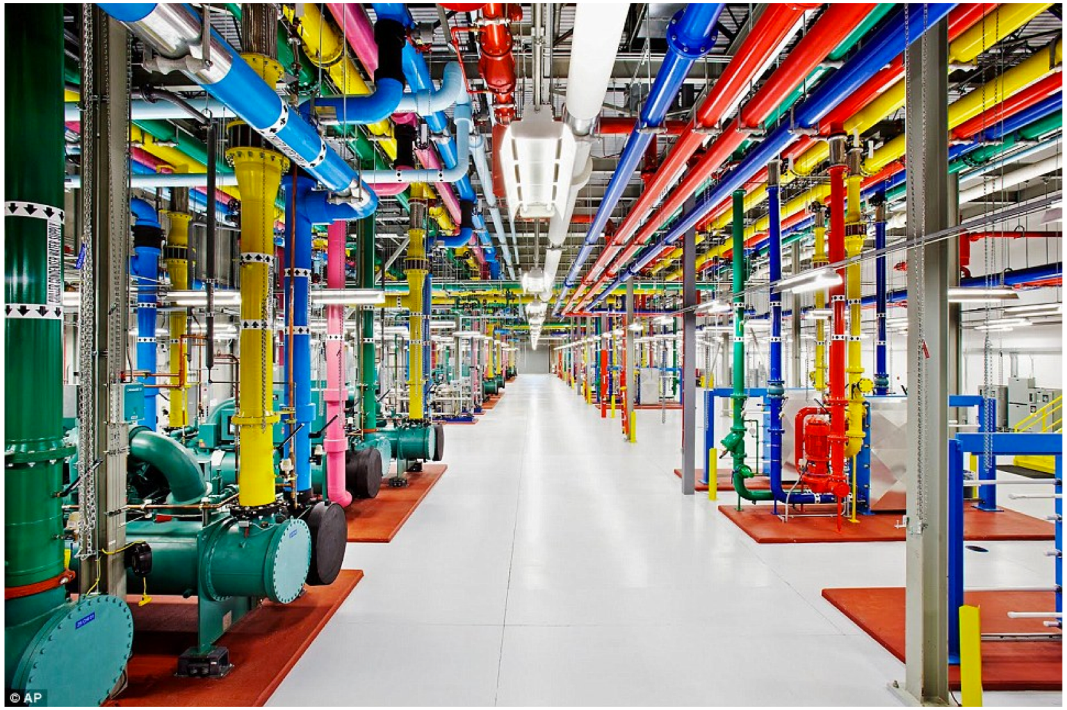
- Ethernet
- Wireless

- Many network systems operate on a network of networks
  - Each individually administrated network is called an Autonomous System (AS)

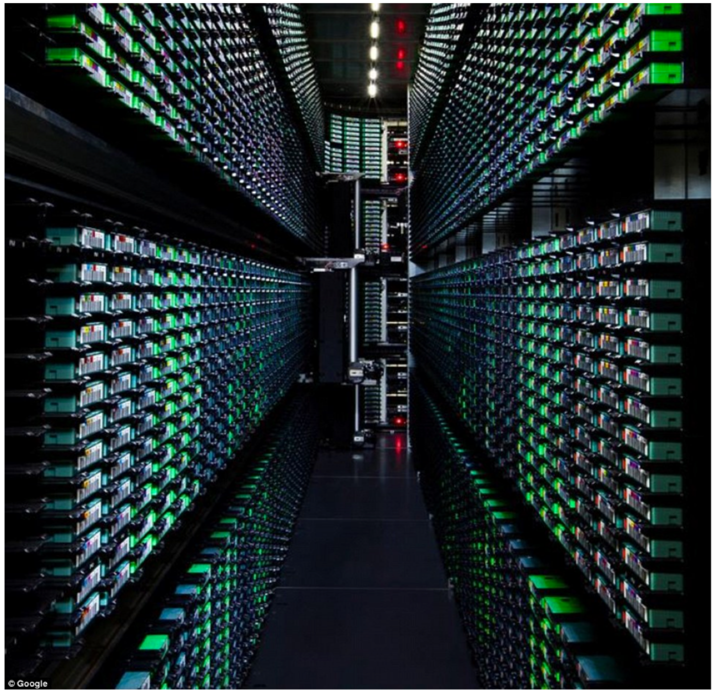
Campus/Enterprise Network



Data Center Network

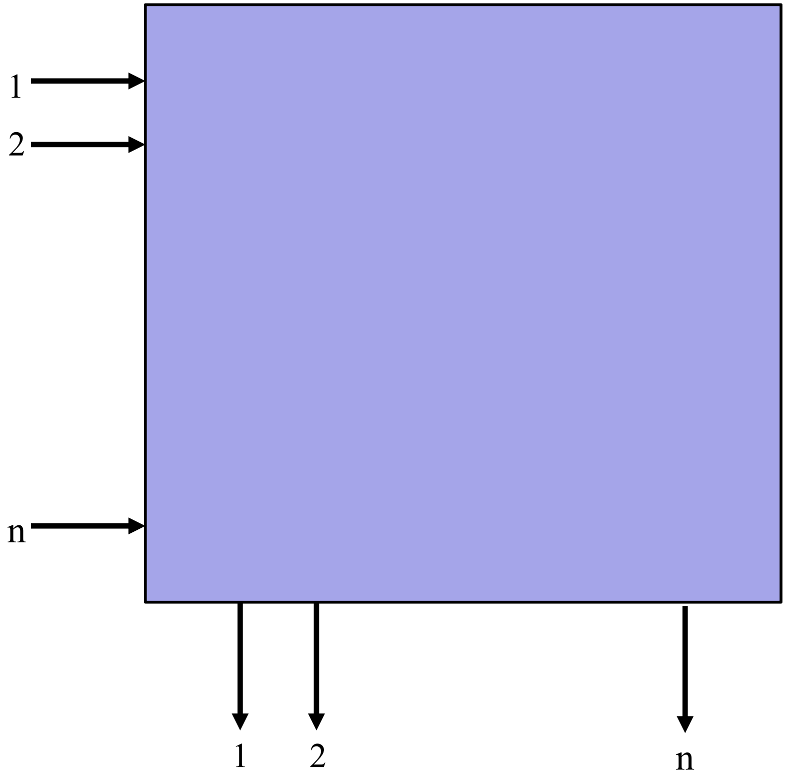


http://www.dailymail.co.uk/sciencetech/article-3369491/Google-s-plan-world-Search-engine-build-half-billion-dollar-data-center-US.html



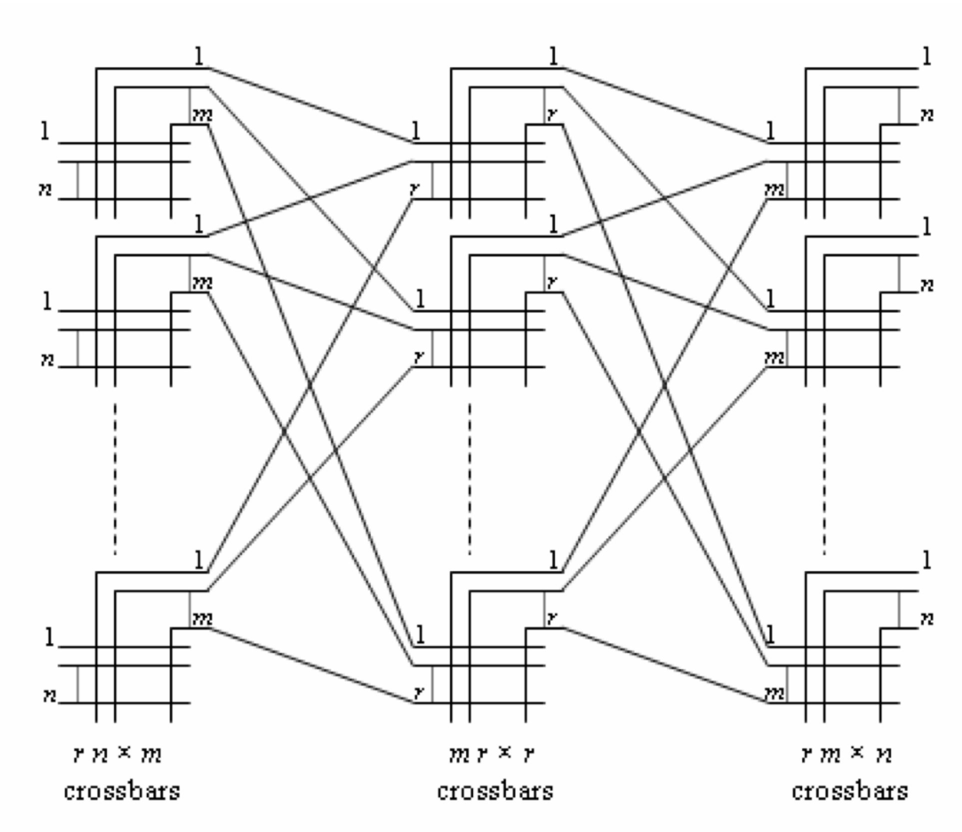
http://www.dailymail.co.uk/sciencetech/article-3369491/Google-s-plan-world-Search-engine-build-half-billion-dollar-data-center-US.htı

Basic Data Center Network Infrastructure Design Goals



- High/non-blocking connection (bi-sect bandwidth)
- Low cost/feasible construction (using existing networking devices)

Data Center Network Infrastructure Intuition: Clos Networks



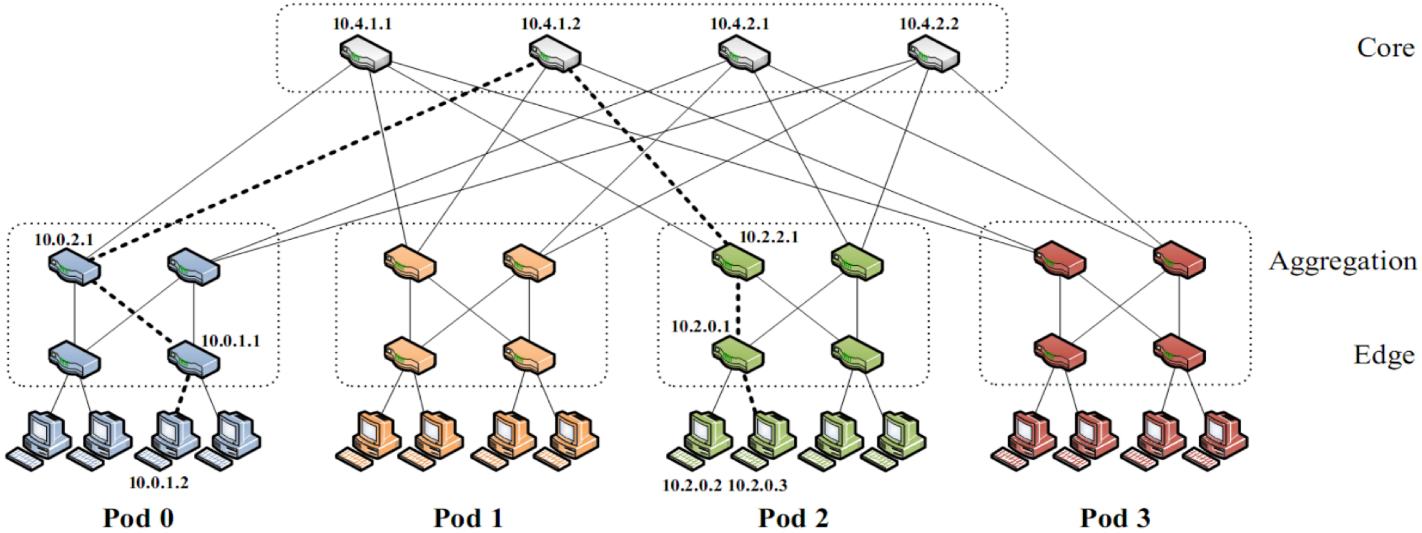
https://en.wikipedia.org/wiki/Clos_network

Q: How big is m so that each new call can be established w/o moving current calls (see note for analysis)?

Problem to think about: If you can move existing calls, it is only m >= n. See note for analysis.

Data Center Network Infrastructure using Fat-tree Networks

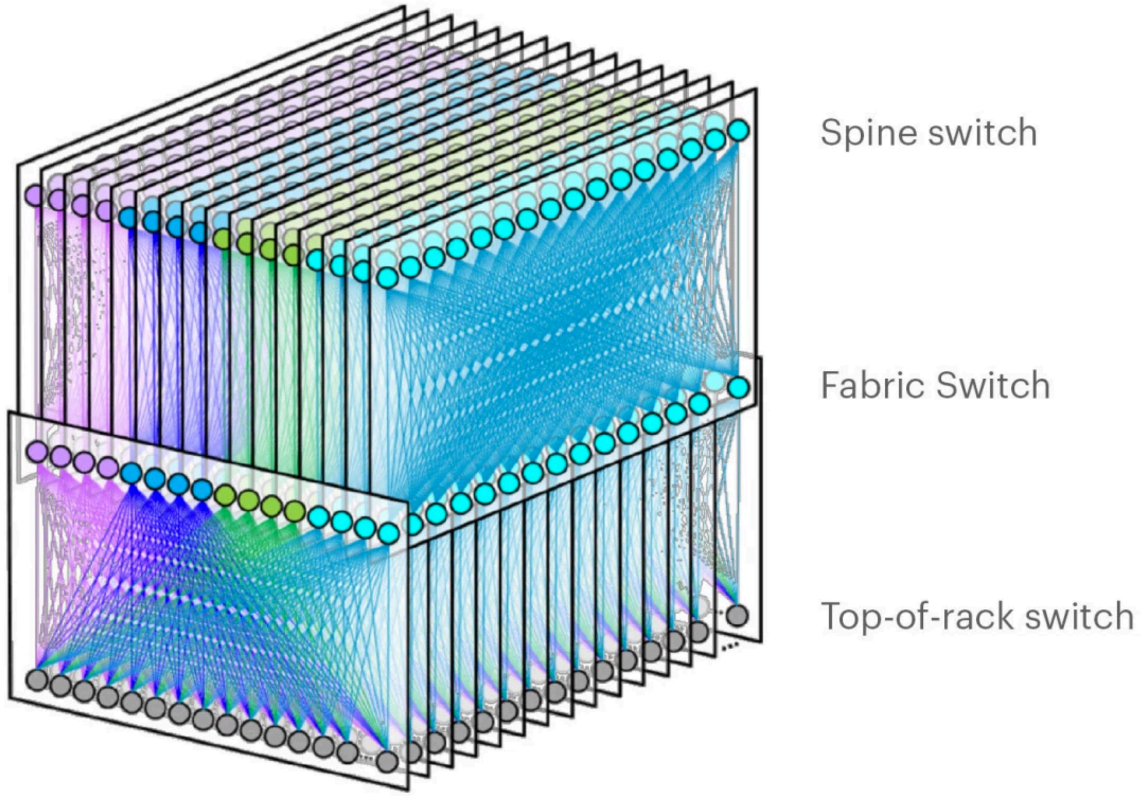
- K-ary fat tree: three-layer topology (edge, aggregation and core)
  - k pods w/ each pod consists of $(k/2)^2$ servers & 2 layers of k/2 k-port switches
    - each edge switch connects to k/2 servers & k/2 aggr. switches
    - each aggr. switch connects to k/2 edge & k/2 core switches
  - $(k/2)^2$ core switches: each connects to k pods



Question to think: How large a network can k-ary support using k-port switches?

http://www.cs.cornell.edu/courses/cs5413/2014fa/lectures/08-fattree.pdf

Data Center Network



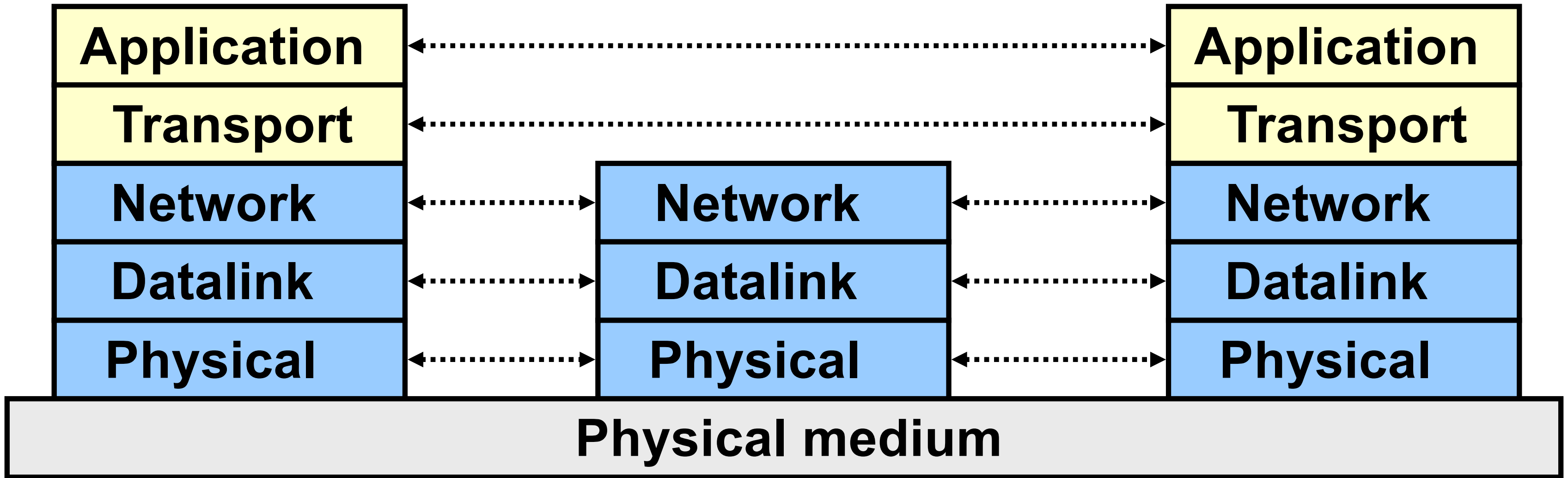
https://engineering.fb.com/2019/03/14/data-center-engineering/f16-minipack/

Outline

- Administrative trivia's
- Course scope
- *Bigger picture*
  - Physical infrastructure
  - *Basic network system architecture: the layering architecture*

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

- Administrative trivia's
- Course scope
- *Bigger picture*
  - Physical infrastructure
  - Basic network system architecture: the layering architecture
    - Why layering?

Why Layering?

Networks are complex !

- many "pieces":
  - hardware
    - hosts
    - routers
    - links of various media
  - software
    - applications
    - infrastructure

Dealing with complex systems:

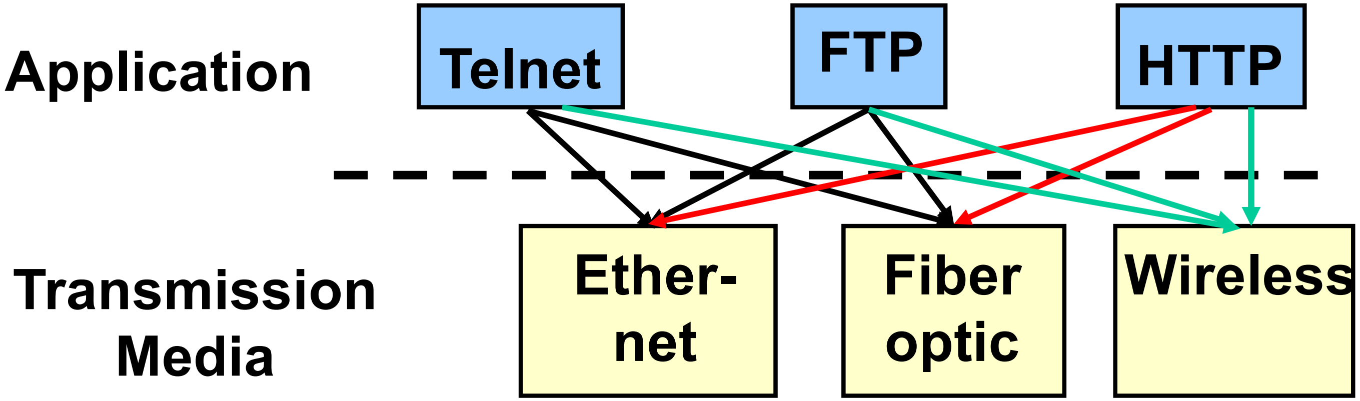
explicit structure allows identification of the relationship among a complex system's pieces

- 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

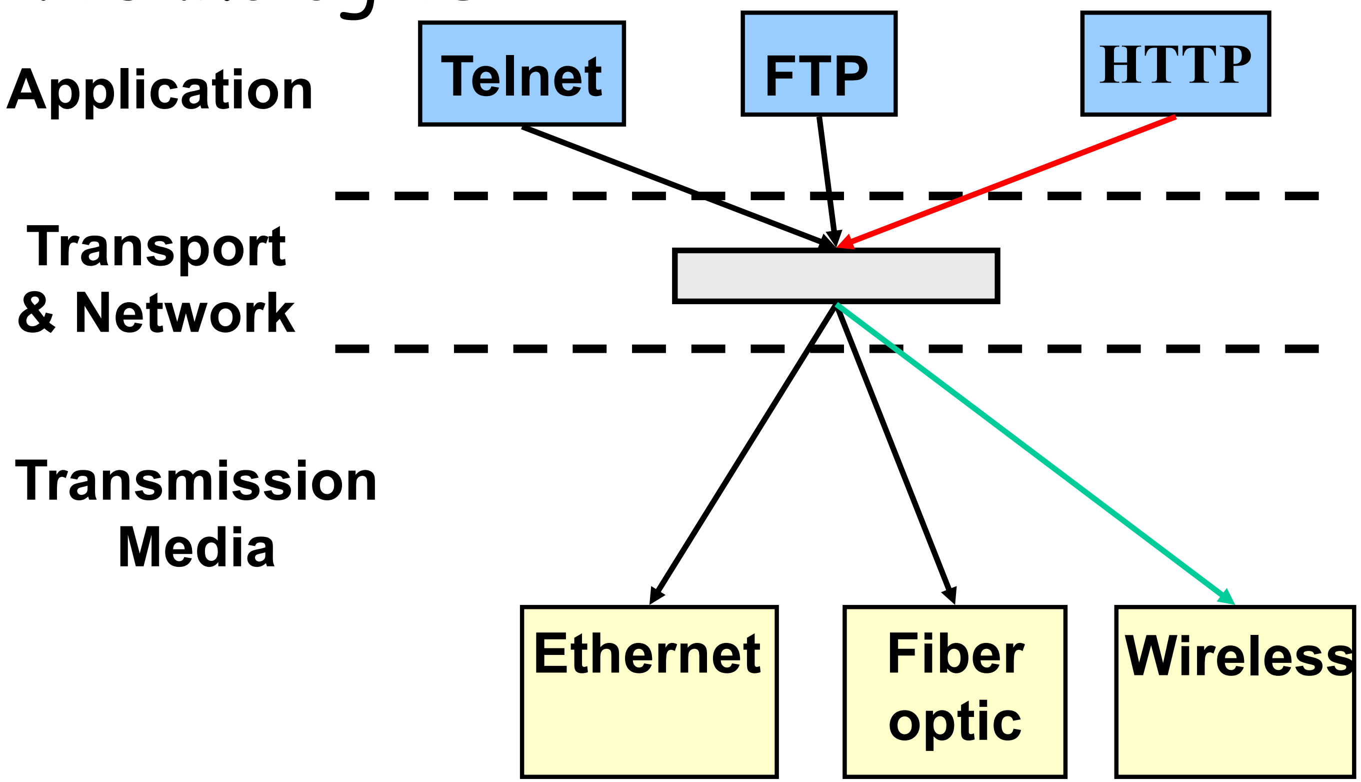
An Example: No Layering



- 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

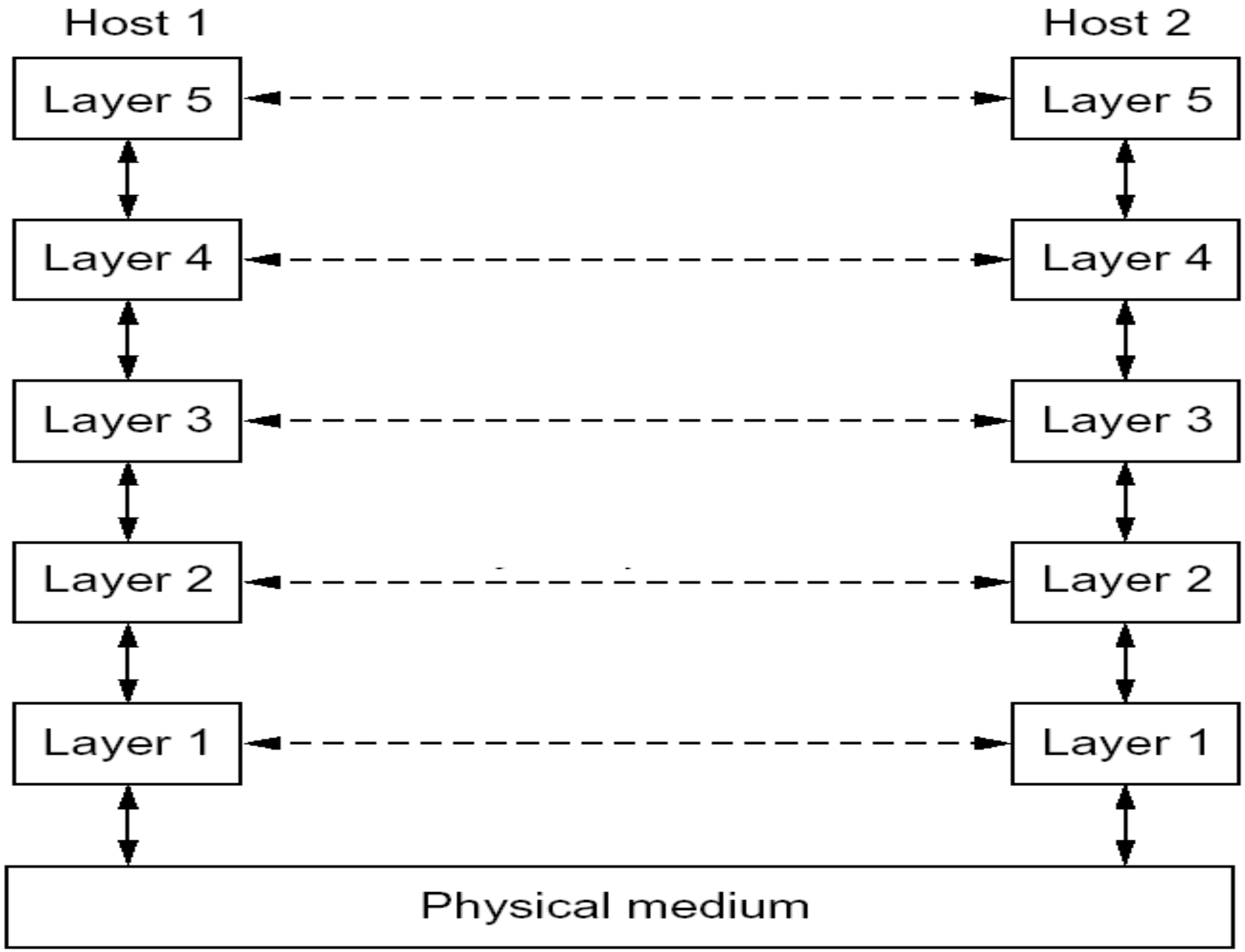


ISO/OSI Concepts

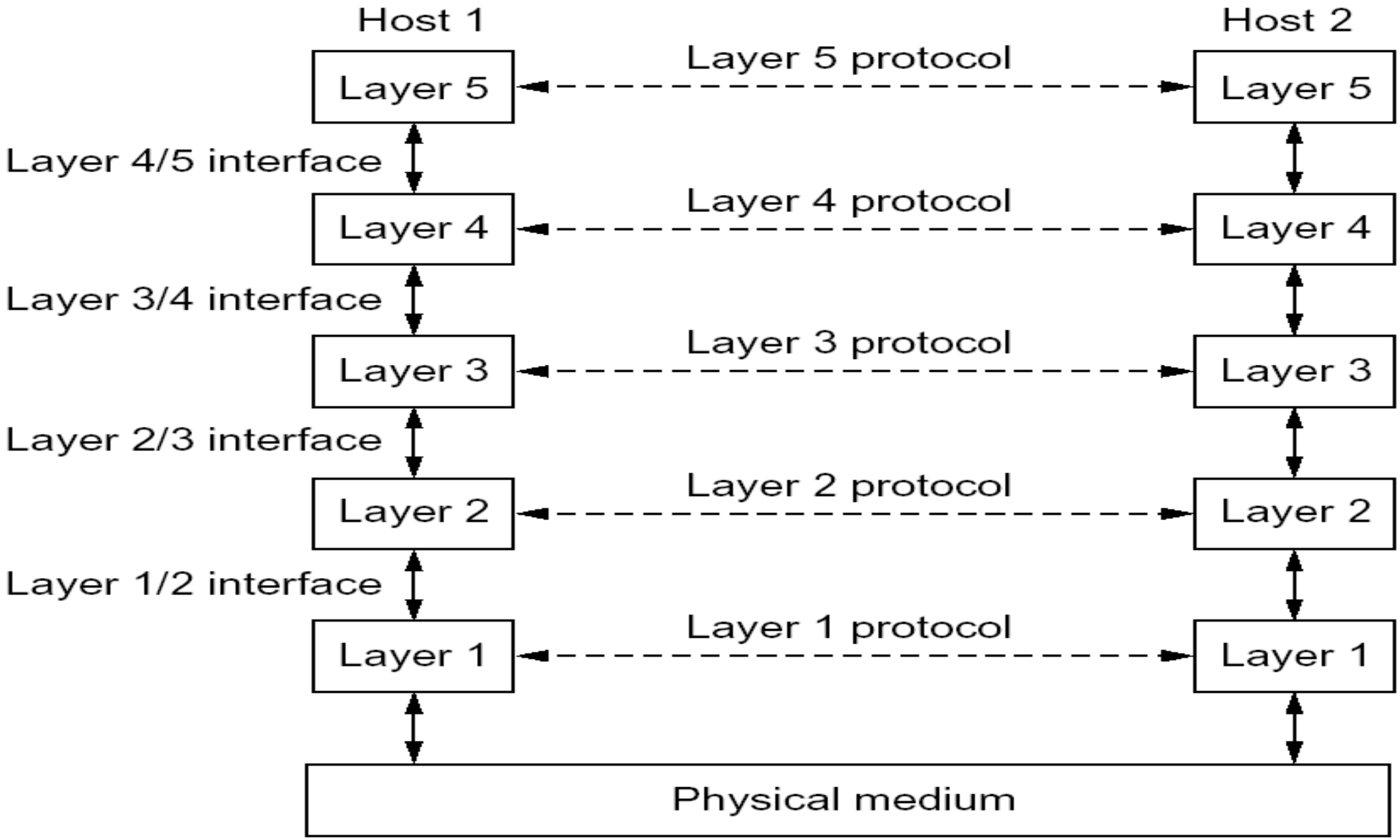
- ISO - International Standard Organization
- OSI - Open System Interconnection

- 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



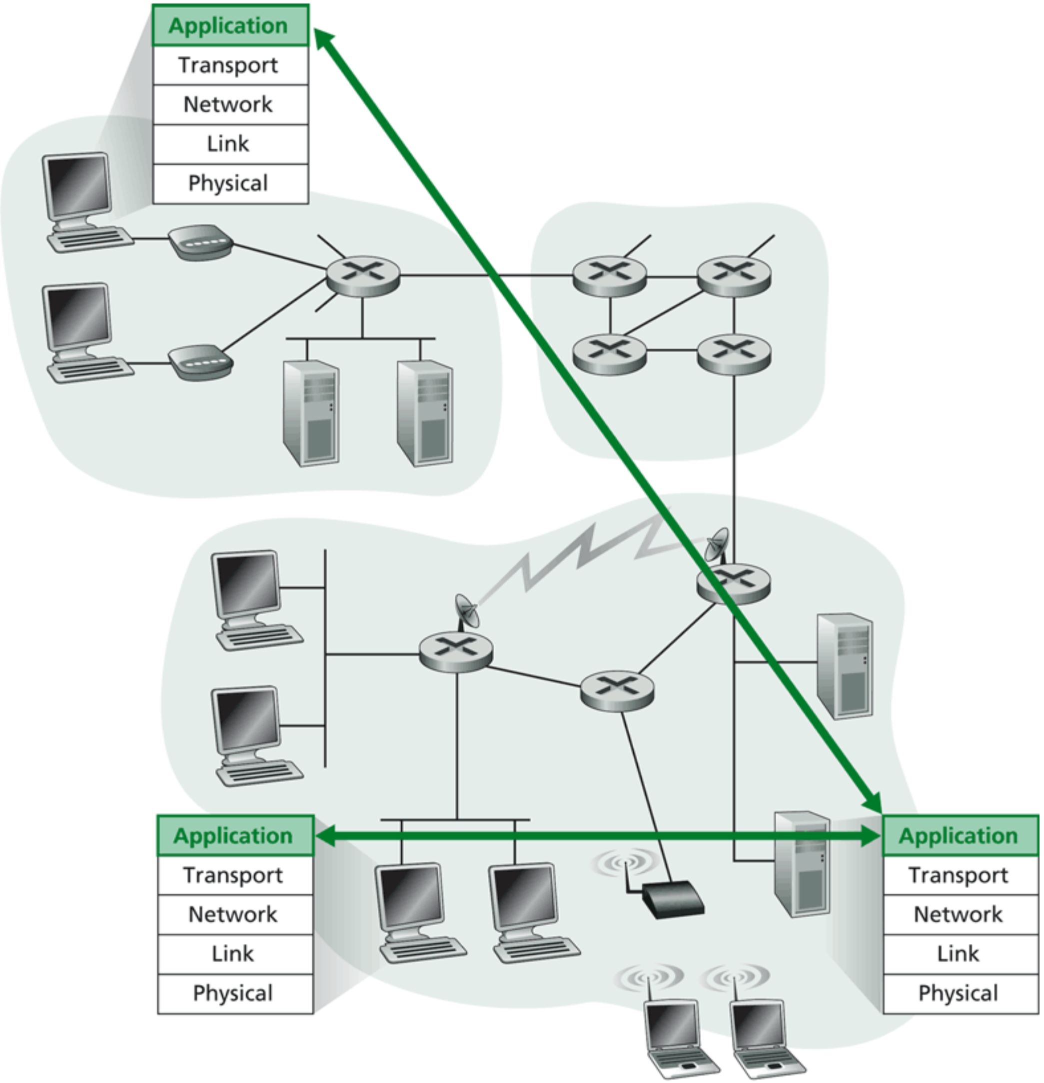
An Example of Layering



Layering -> *Logical* Communication

E.g.: application

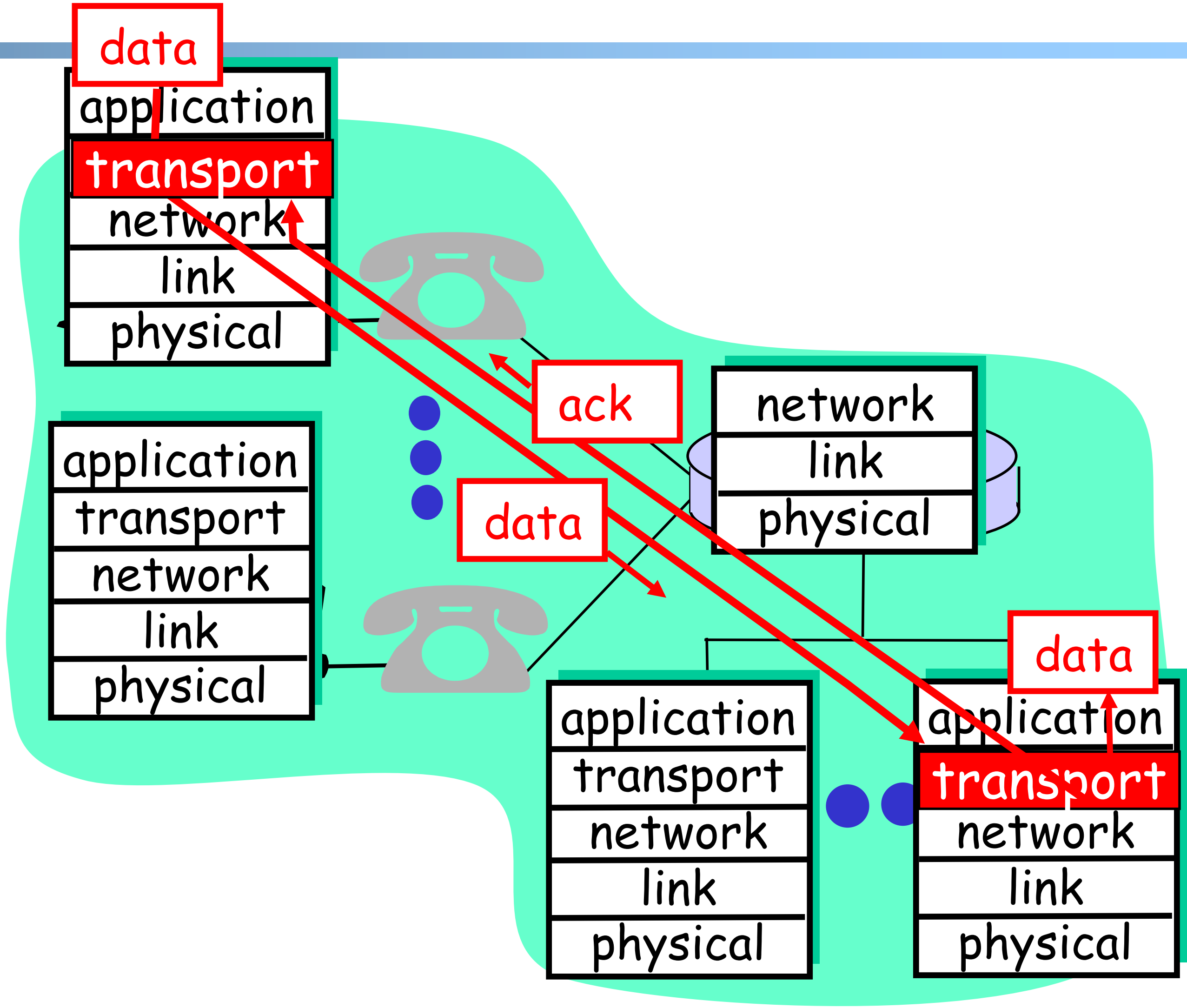
- provide services to users
- application protocol:
  - send messages to peer
  - for example, HELO, MAIL FROM, RCPT TO are messages between two SMTP peers



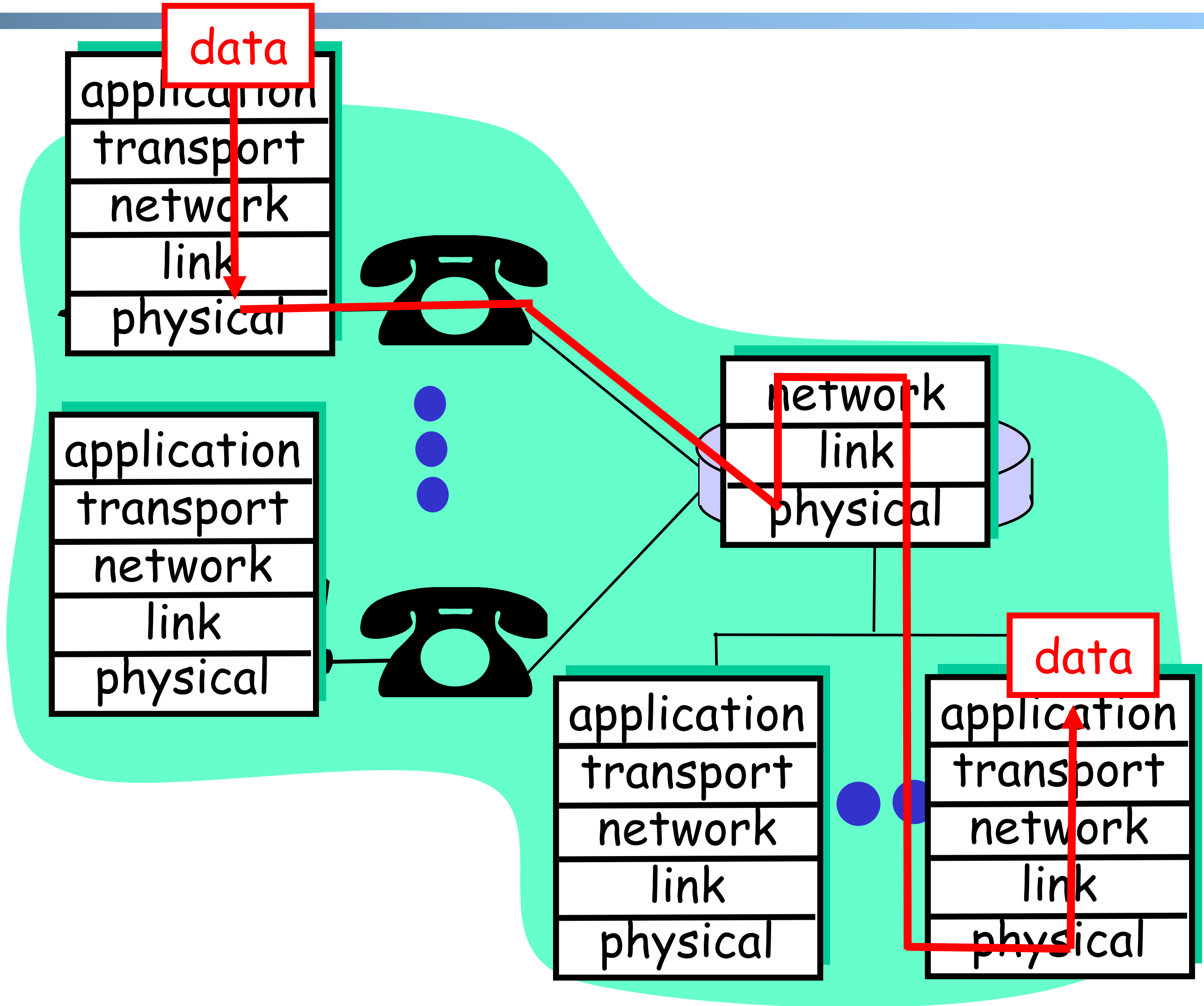
Layering: *Logical* Communication

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



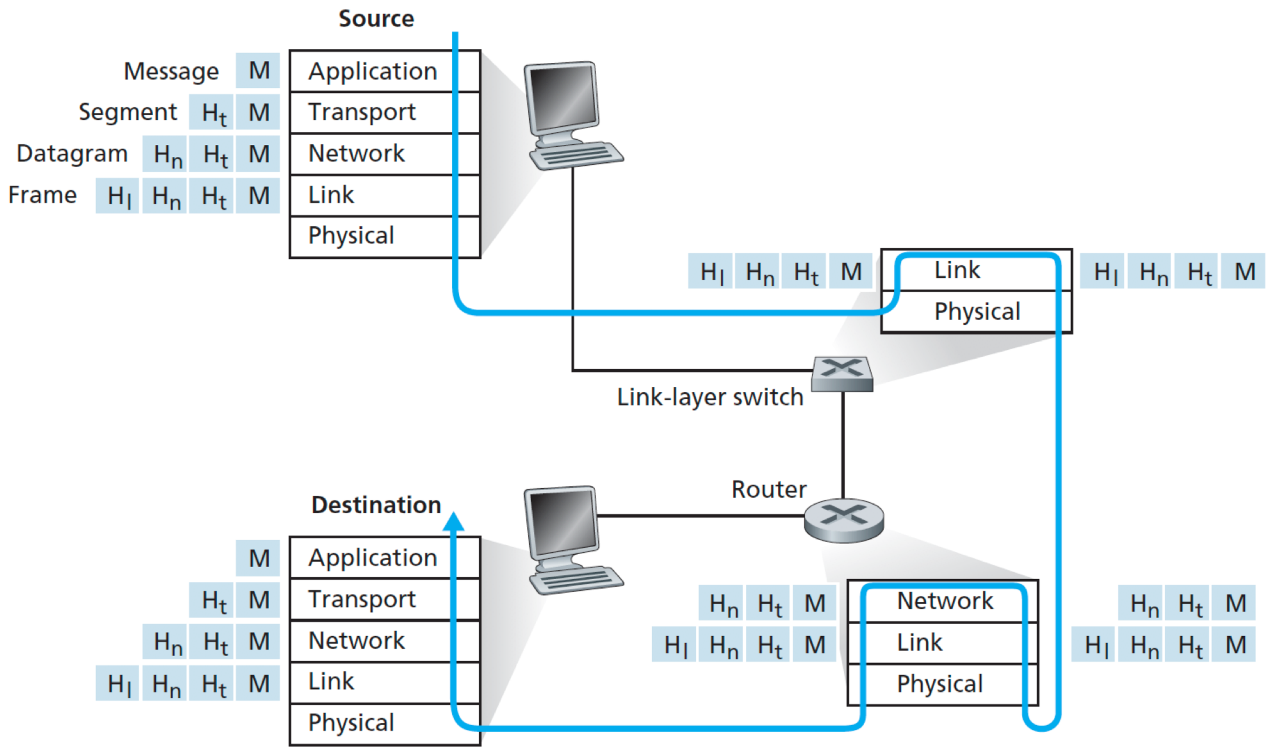
Layering: *Physical* Communication



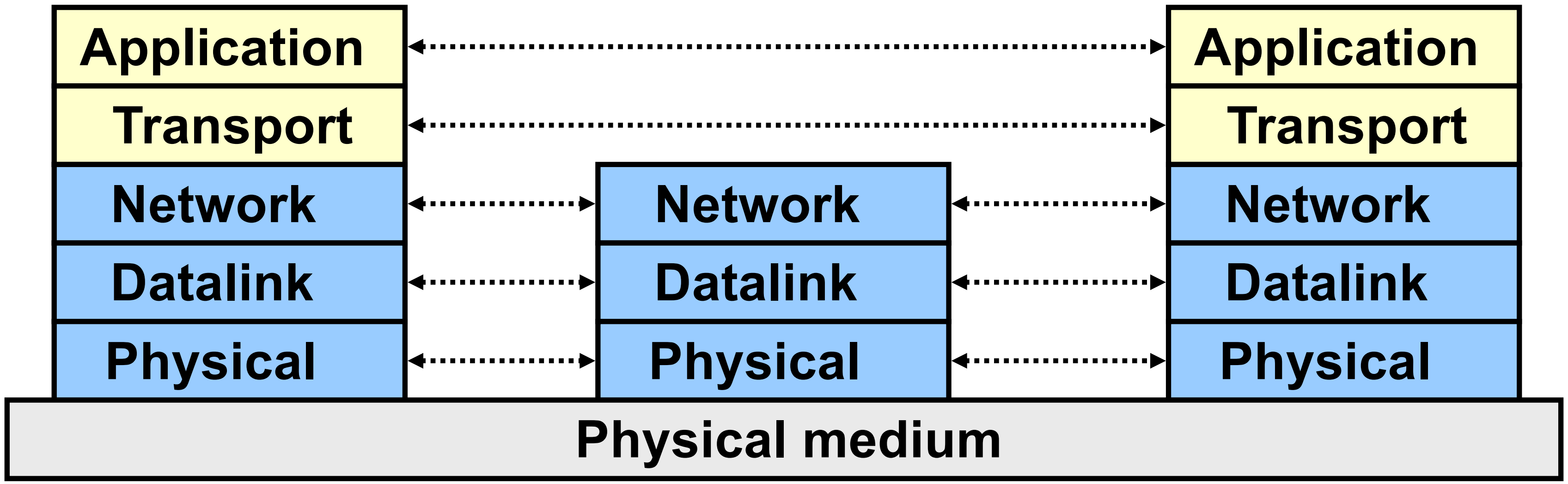
Protocol Layering and Meta Data

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

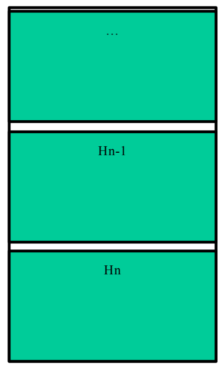


Packet as a Stack in a Layered Architecture

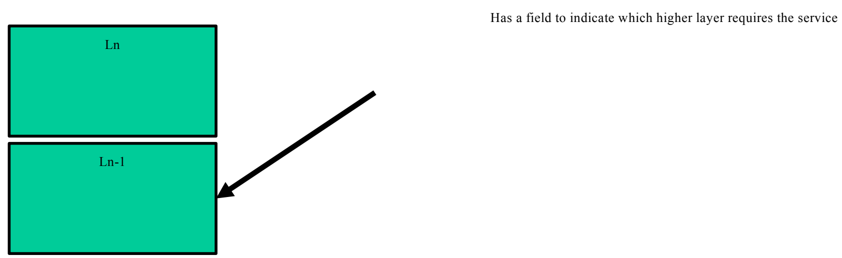


Some Implications of Layered Architecture

- A packet as a stack container



- Each layer needs multiplexing and demultiplexing to serve layer above



Key design issue:

How do you *divide* functionalities among the layers?

Outline

- Administrative trivia's
- Course scope
- *Bigger picture*
    - Physical infrastructure
    - Basic network system architecture: the layering 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.

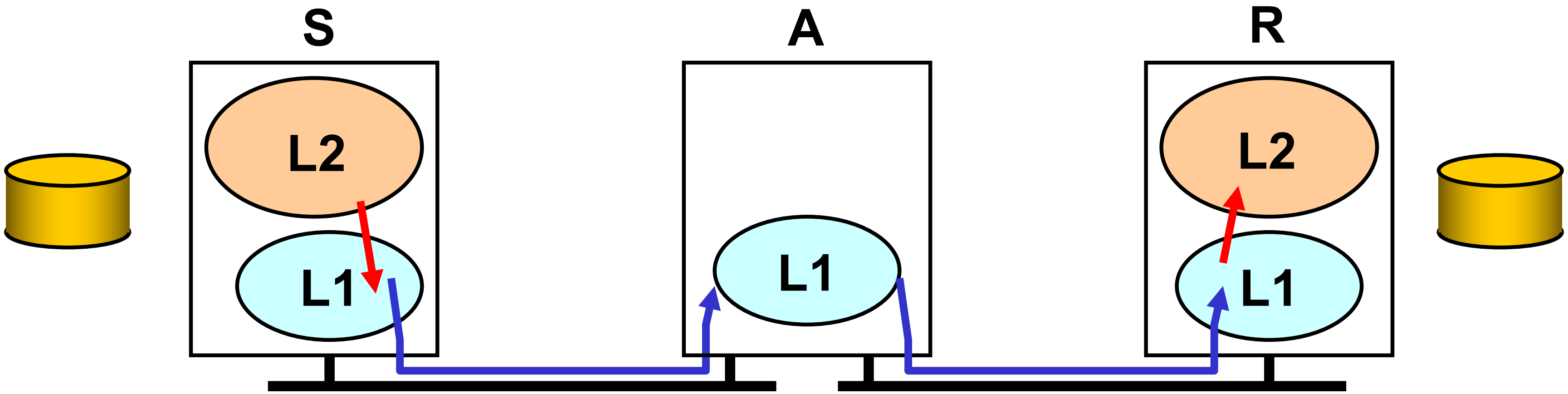
J. Saltzer, D. Reed, and D. Clark, 1984

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

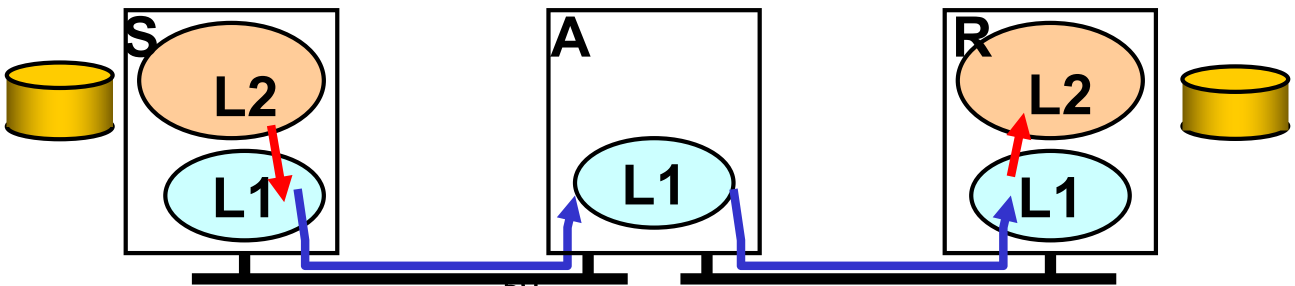
Example: Where to Provide Reliability ?



- 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

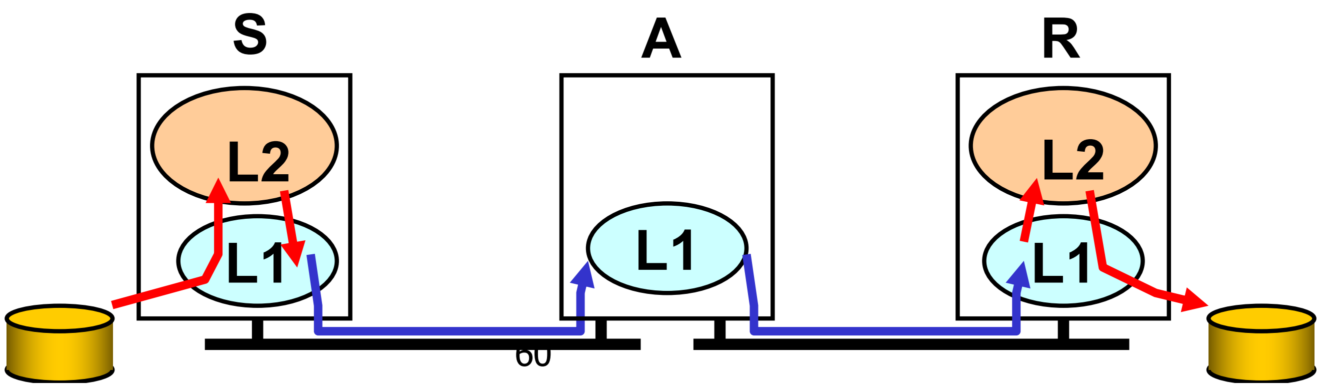
What are Reasons for Implementing Reliability at Higher Layer ?

- 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



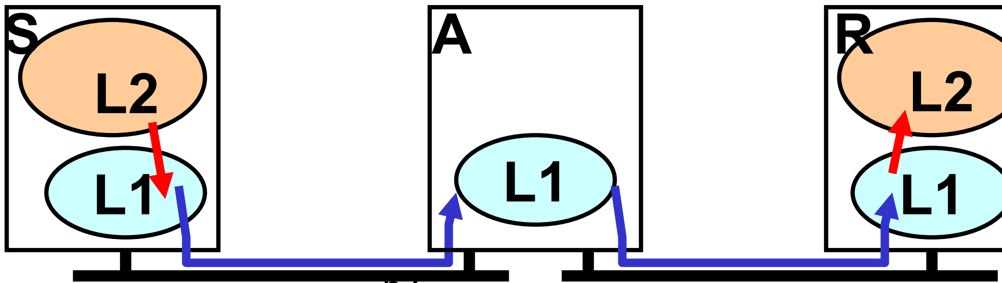
Summary: End-to-End Arguments

- 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

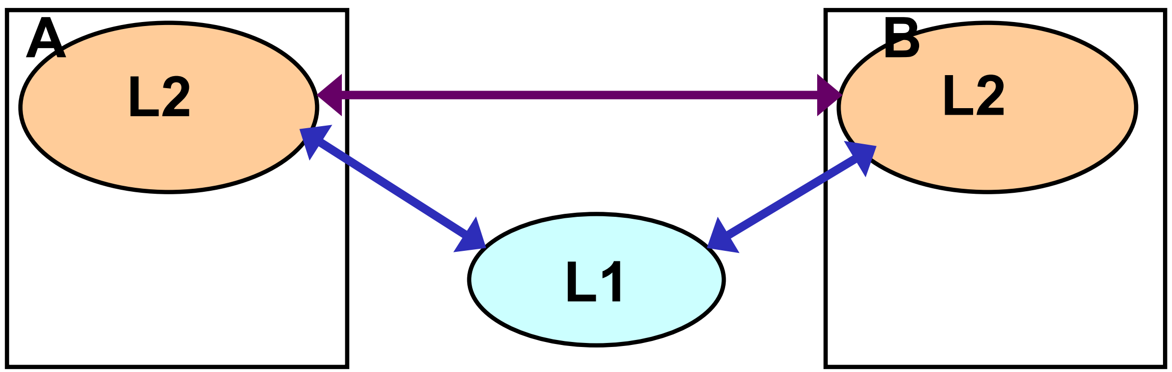
- We used reliability as an example

- 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:

end-to-end arguments

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

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

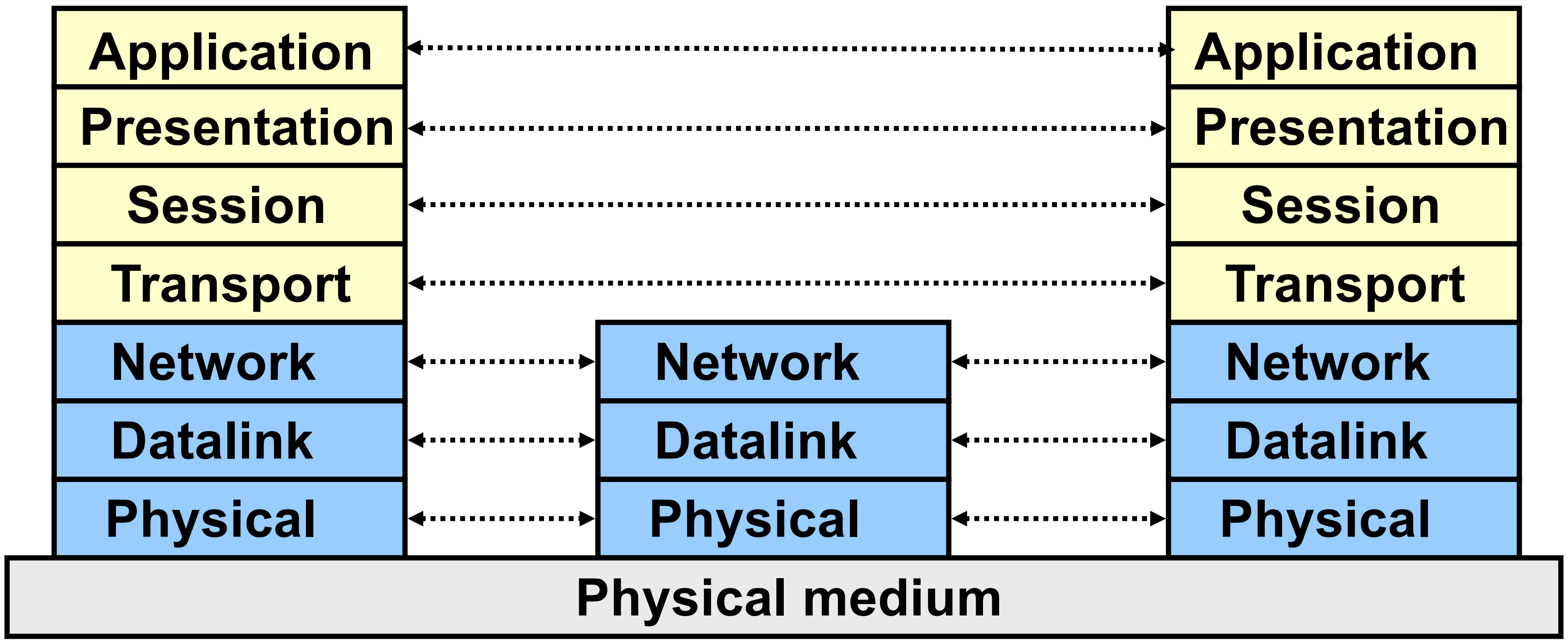
Question to Think: Limitations of Layering Architecture for Network Systems

Outline

- Administrative trivia's
- Course scope
- *Bigger picture*
  - Physical infrastructure
  - Basic network system architecture: the layering architecture
    - What is layering?
    - Why layering?
    - How to determine the layers?
    - *ISO/OSI layering and Internet layering*

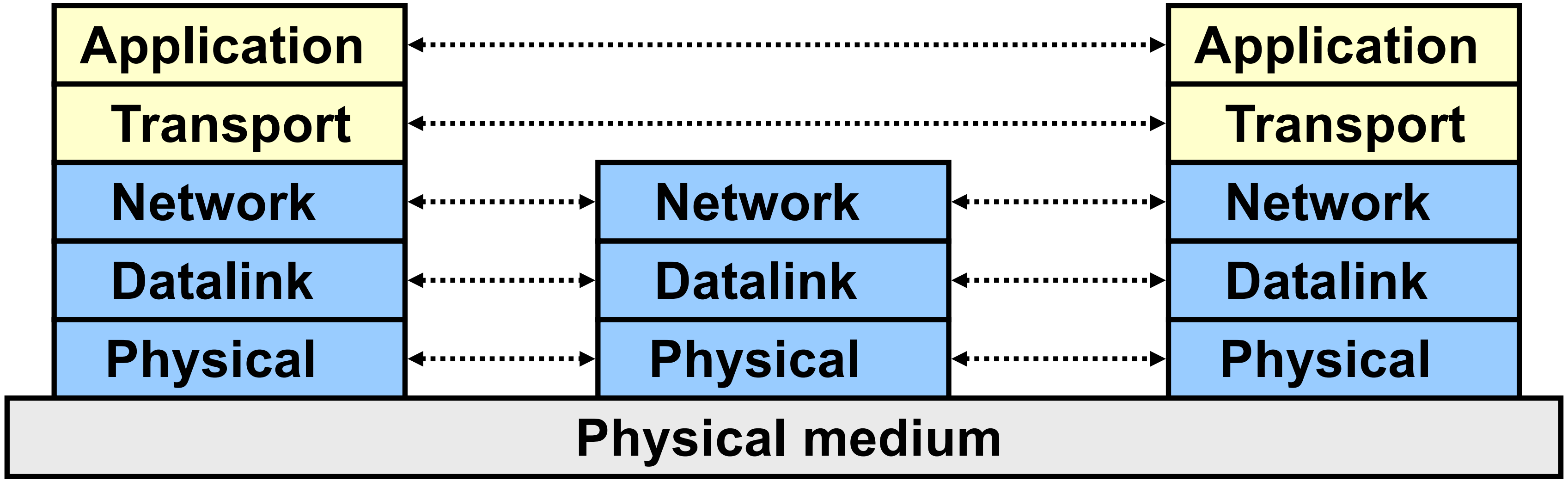
ISO/OSI Reference Model

- Seven layers
  - highest four layers are implemented in host



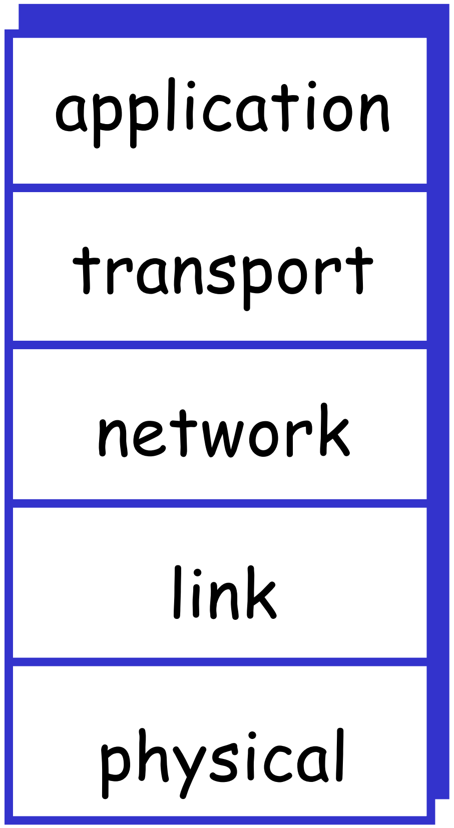
Internet Layering

- Five layers
  - highest two layers are implemented in host

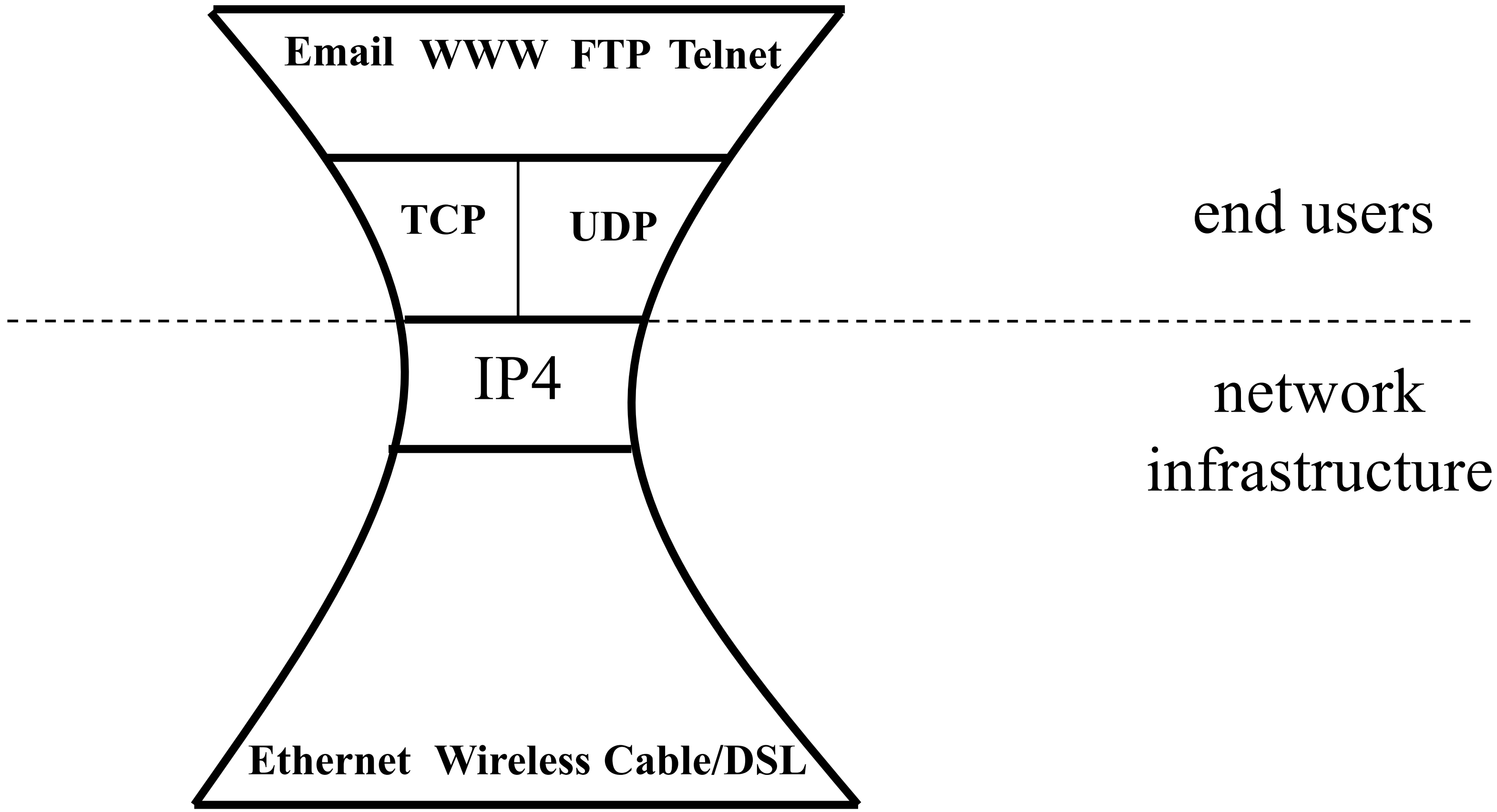


Internet Protocol Layers

- Five layers
  - Application: applications
    - ftp, smtp, http, p2p, IP telephony, blockchain, MapReduce, …
  - Transport: host-host data transfer
    - tcp (reliable), udp (not reliable)
  - Network: routing of datagram from source to destination
    - ipv4, ipv6
  - Link: data transfer between neighboring network elements
    - ethernet, 802.11, cable, DSL, …
  - Physical: bits "on the wire"
    - cable, wireless, optical fiber



The Hourglass Architecture of the Internet



Backup Slides

Why Network Systems: A Place to Build Systems

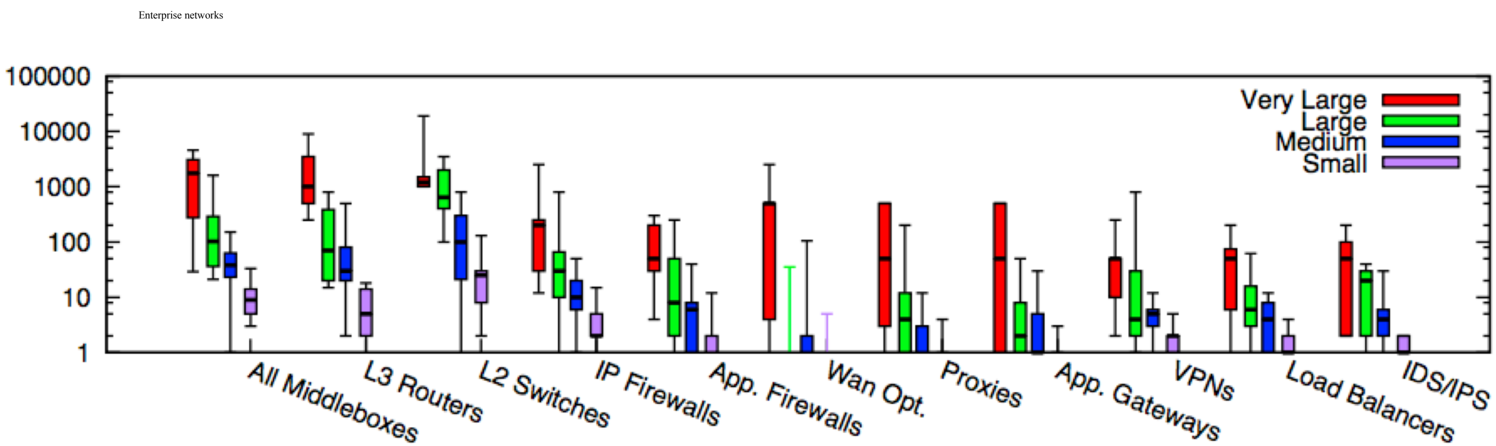
- Distributed systems
- Software engineering
- Operating systems
- Computer architecture
- …

Why Networked Systems: A Place to Apply Theory

- Algorithms and data structures
- Queuing theory
- Formal methods
- Cryptography
- Programming languages
- …

Complexity: Simple Forwarding to Network Functions

- Modern networks contain diverse types of equipment beyond simple routing/forwarding



Small: <=1k hosts; Medium: 1k-10k; Large: 10k-100k; Very Large: >=100k

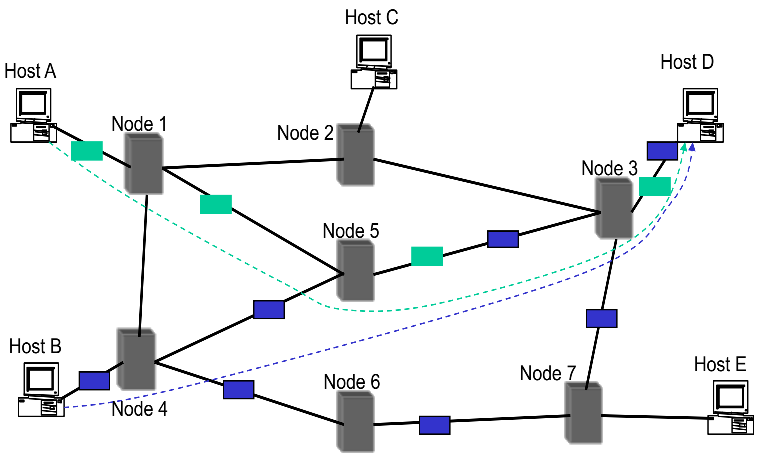
Source: [Sherry, et. al SIGCOMM'12]

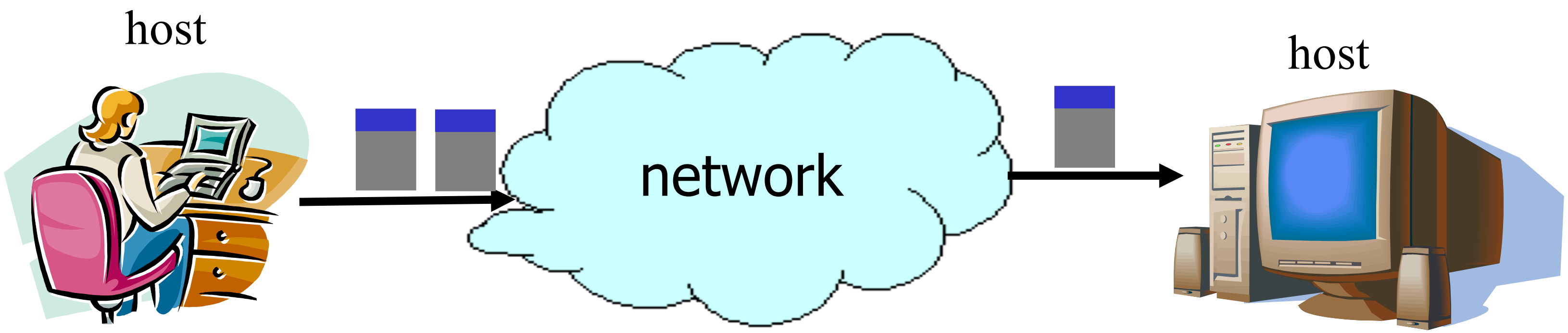
A Typical Summary of End-to-End Arguments

- 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)

Networked System Communication: Packet Switching Systems

- Packet switching
  - Divide messages into a sequence of packets
  - Headers with source and destination address

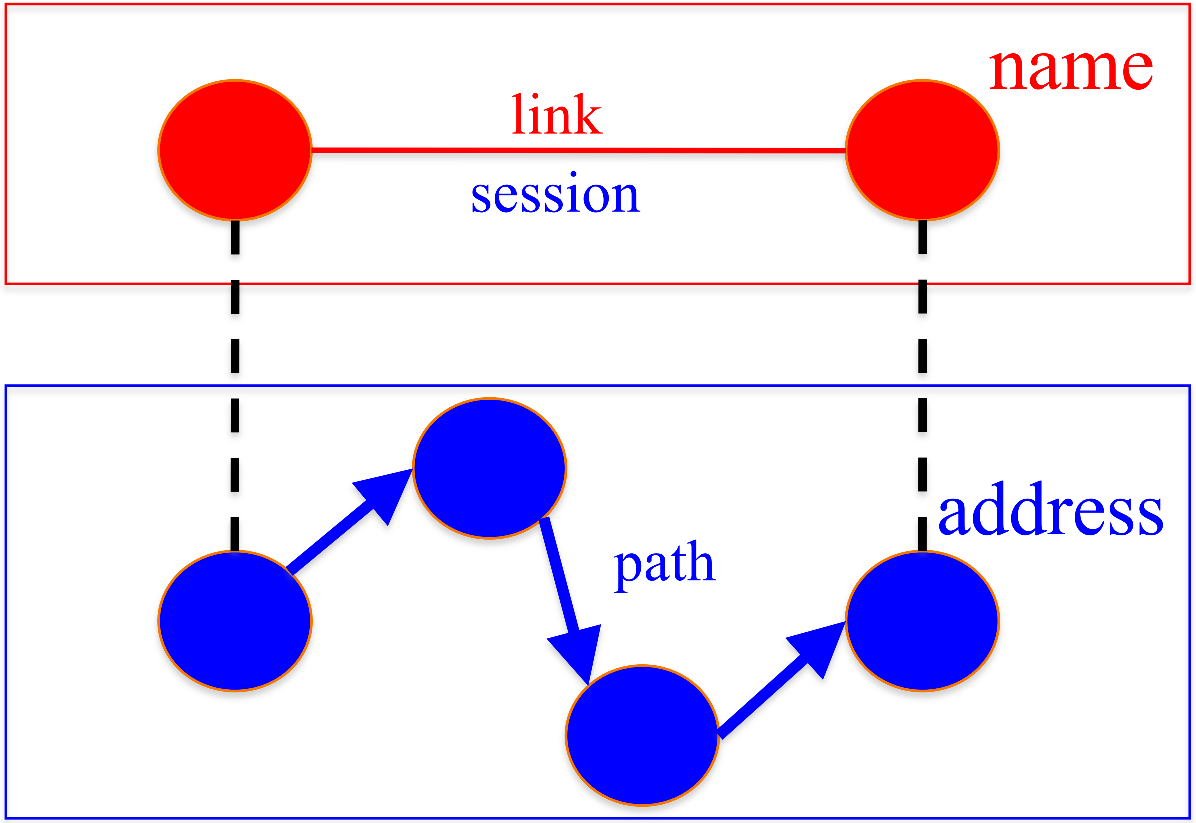




Why Packets?

- Statistical multiplexing
  - Data traffic is bursty
    - Logging in to remote machines
    - Exchanging e-mail messages
  - Don't want to waste bandwidth
    - No traffic exchanged during idle periods
- Packets can be delivered by most anything
  - RFC 1149: IP Datagrams over Avian Carriers

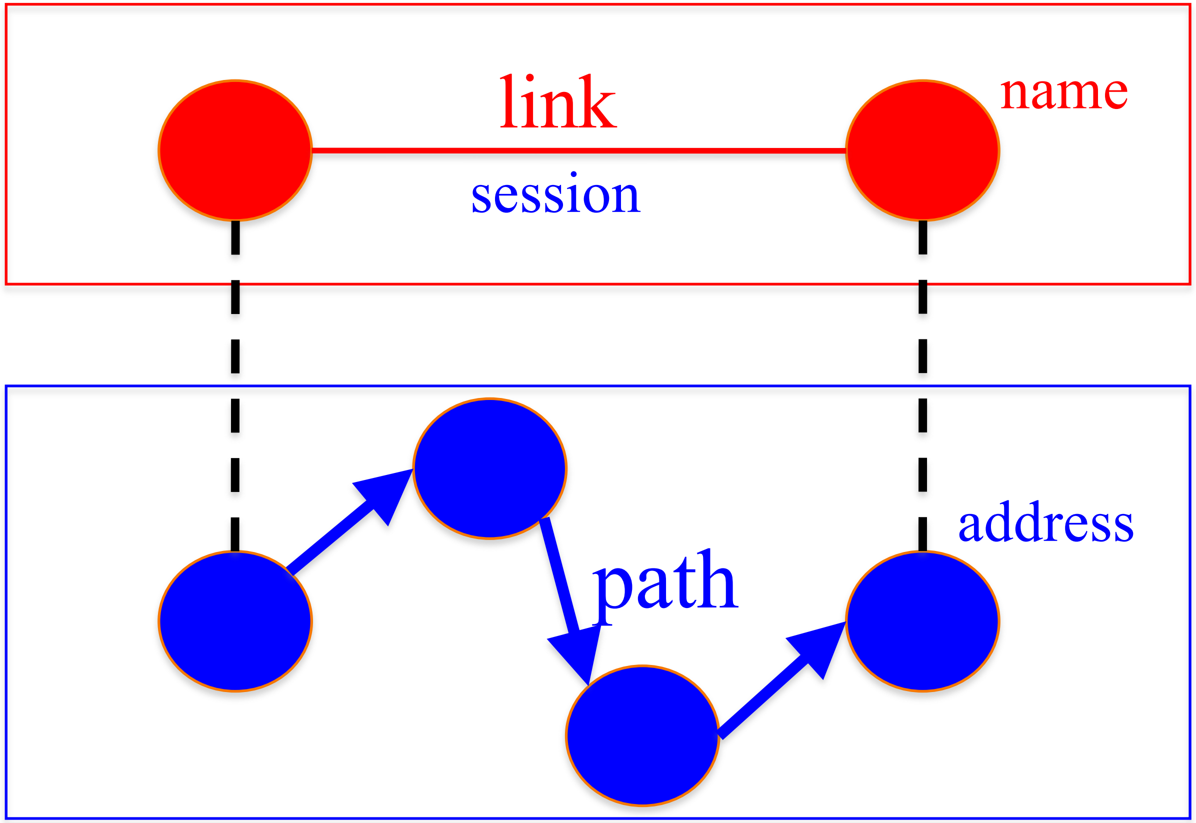
Supporting Layer: Directories to Map Name to Address



Types of Directories

- Simplistic designs
  - Ask everyone (e.g., flooding in ARP)
  - Tell everyone (e.g., pushing /etc/hosts)
  - Central directory
- Scalable distributed designs
  - Hierarchical namespace (e.g., DNS)
  - Flat name space (e.g., Distributed Hash Table)

Supporting Layer: Routing Mapping Flow/Pkt to Path



Path Computation

- Spanning tree (e.g., Ethernet)
  - One tree that connects every pair of nodes
- Shortest paths (e.g., OSPF, IS-IS, RIP)
  - Shortest-path tree rooted at each node
- Locally optimal paths (e.g., BGP)
  - Each node selects the best among its neighbors
- End-to-end paths (e.g., source routing)
  - Each node picks the best end-to-end path
- Load balancing path …

Networked Systems == Protocols?

SNMP WAP SIP PPP IPX RAFT MAC

LLDP FTP UDP

OSPF RTP ICMP IMAP IGMP HIP

EIGRP RED BGP HTTP ARP ECN

RIP IP MPLS TCP COAP

SMTP RTSP BFD CIDR

NNTP

SACK TLS NAT STUN

QUIC DNS SSH DHCP

POP VLAN LISP VTP TFTP ISIS

Networked Systems == Boxes?

Router

Label Switched Router

Load balancer

Serving Gateway

Switch

Repeater

Gateway

Intrusion Detection System

Bridge

Route Reflector

Deep Packet Inspection

DHCP server

Packet shaper

Firewall

Hub

NAT

PDN Gateway

Packet sniffer

WAN accelerator

DNS server

eNodeB

Proxy

Network Systems == Network Tools?

arpwatch

syslog

tcpdump

wget

nslookup

traceroute

trat

snort

nmap

whois

ipconfig

rancid

ntop

bro

net-snmp

dig

ping

iperf

wireshark

NDT

dummynet

mrtg

Network Systems == Network Tools?

arpwatch
syslog
tcpdump
wget
nslookup
traceroute
trat
snort
nmap
whois
ipconfig
rancid
ntop
bro
net-snmp
dig
ping
iperf
wireshark
NDT
dummynet
mrtg