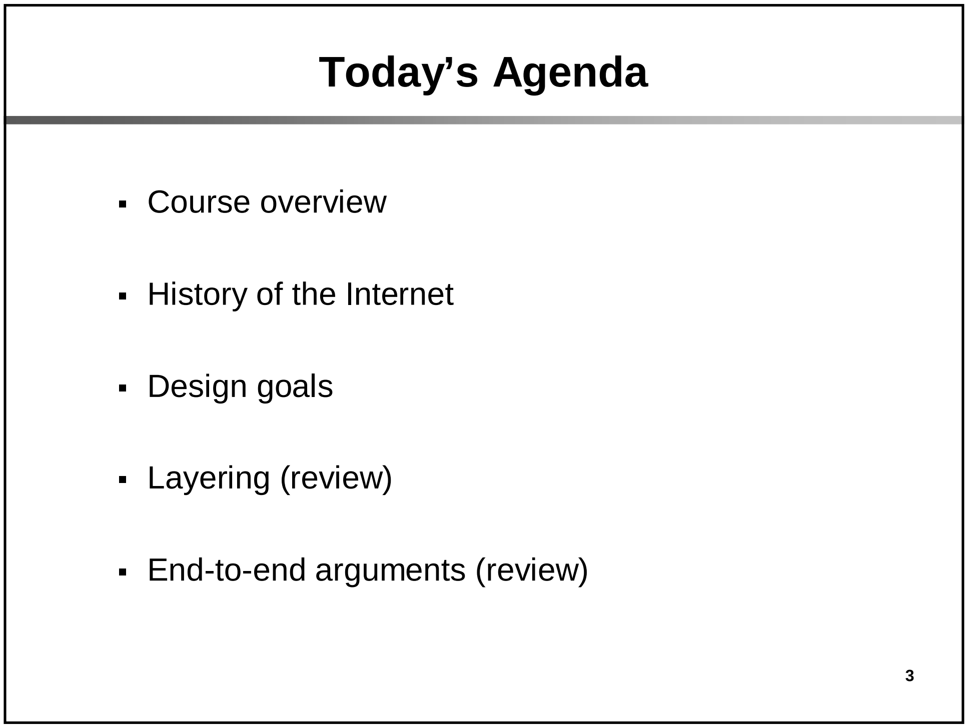
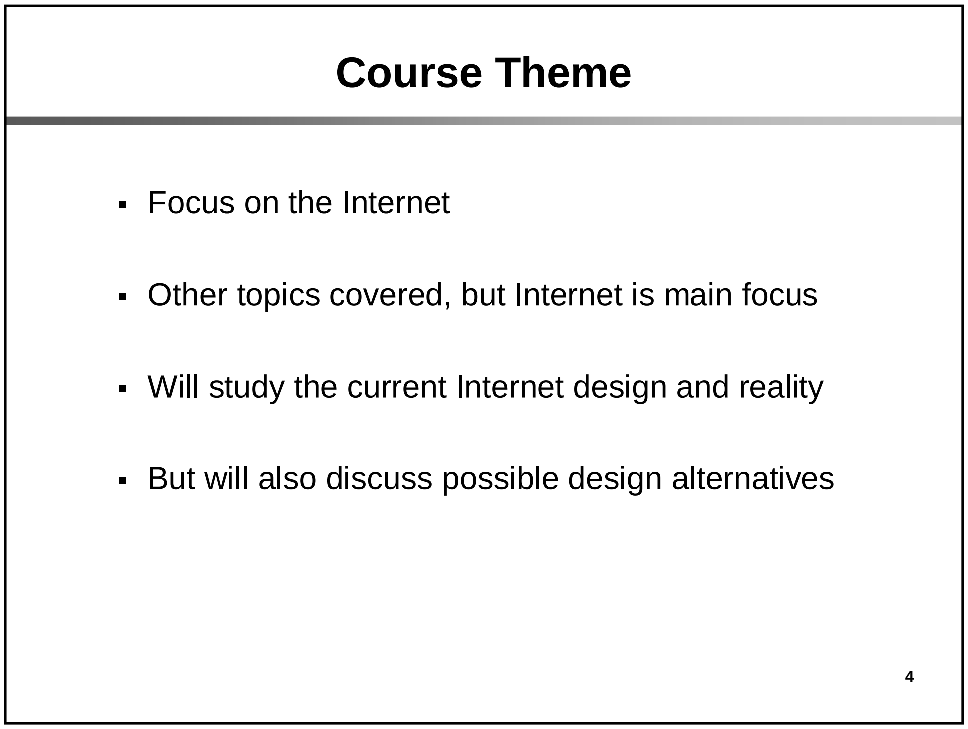
CS 268: Lecture 4 (Internet Architecture & E2E Arguments)



Today’s Agenda

- Course overview
- History of the Internet
- Design goals
- Layering (review)
- End-to-end arguments (review)



Course Theme

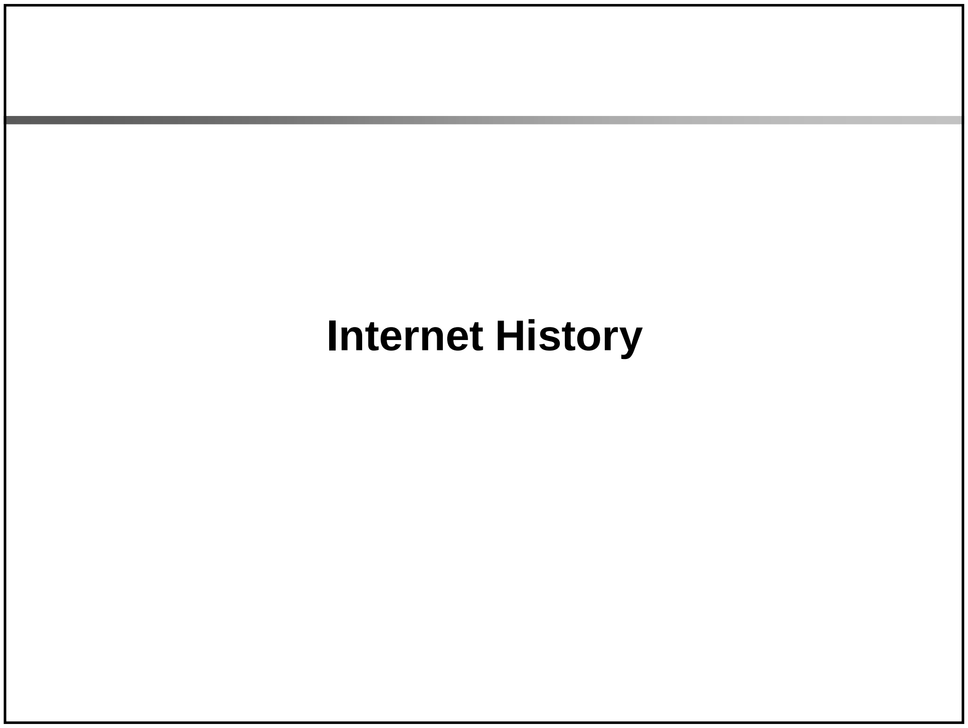
- Focus on the Internet
- Other topics covered, but Internet is main focus
- Will study the current Internet design and reality
- But will also discuss possible design alternatives

Topics

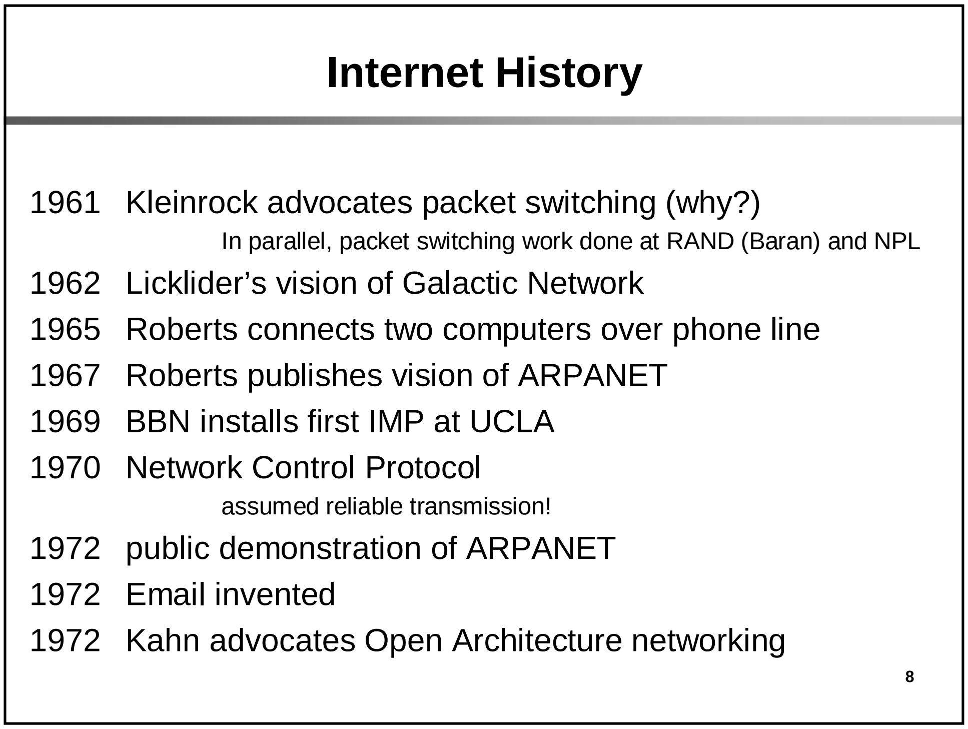
- General Internet background (review)
- TCP/IP (historical)
- TCP congestion control
- Beyond TCP
- Router Support for congestion control
- Intradomain routing
- Interdomain routing
- Multicast routing
- QoS: Intserv and DiffServ
- Mobility

Topics Continued

- Security: crypto
- Security: robust protocols
- Security: malware
- Web
- Overlay networks
- P2P-style overlays
- Distributed Computing
- Wireless
- Sensornets (2)
- Perspectives on Internet Architecture
- Alternatives to the Internet Architecture (2)



Internet History



Internet History

1961 Kleinrock advocates packet switching (why?)

In parallel, packet switching work done at RAND (Baran) and NPL

1962 Licklider’s vision of Galactic Network

1965 Roberts connects two computers over phone line

1967 Roberts publishes vision of ARPANET

1969 BBN installs first IMP at UCLA

1970 Network Control Protocol

assumed reliable transmission!

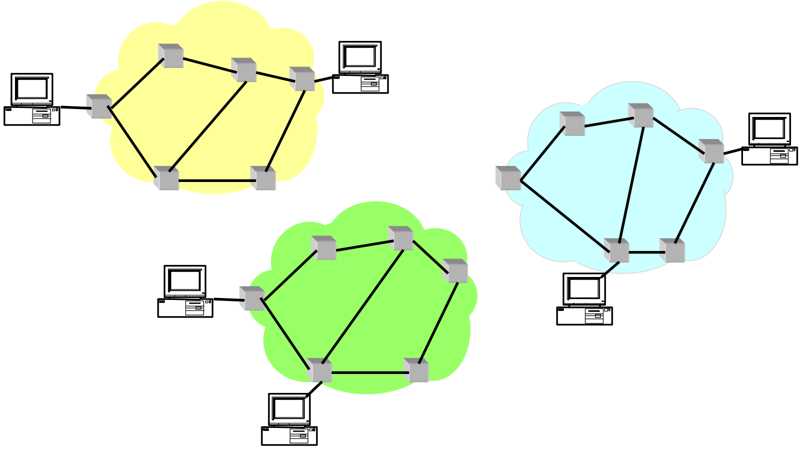
1972 public demonstration of ARPANET

1972 Email invented

1972 Kahn advocates Open Architecture networking

The Problem

- Many different packet-switching networks
- Only nodes on the same network could communicate



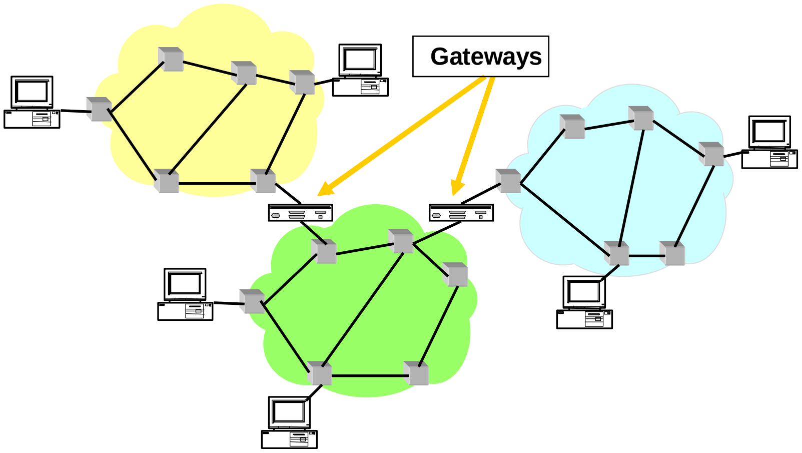
9

Kahn’s Ground Rules

- Each network is independent and must not be required to change
- Best-effort communication
- Boxes (routers) connect networks
- No global control at operations level

10

Solution



Question

- Kahn imagined there would be only a few networks (~20) and thus only a few routers
- He was wrong
- Why?

History Continued

1974	Cerf and Kahn paper on TCP/IP
1980	TCP/IP adopted as defense standard
1983	Global NCP to TCP/IP flag day
198x	XNS, DECbit, and other protocols
1984	Janet
1985	NSFnet (picks TCP/IP)
198x	Internet meltdowns due to congestion
1986+	Van Jacobson saves the Internet (BSD TCP)
1988	Deering and Cheriton propose multicast
199x	QoS rises and falls
199x	ATM rises and falls (as an internetworking layer)
1994	Internet goes commercial
200x	The Internet boom and bust
2001	Ion Stoica gets Ph. D.!

Internet Design Goals

Goals (Clark'88)

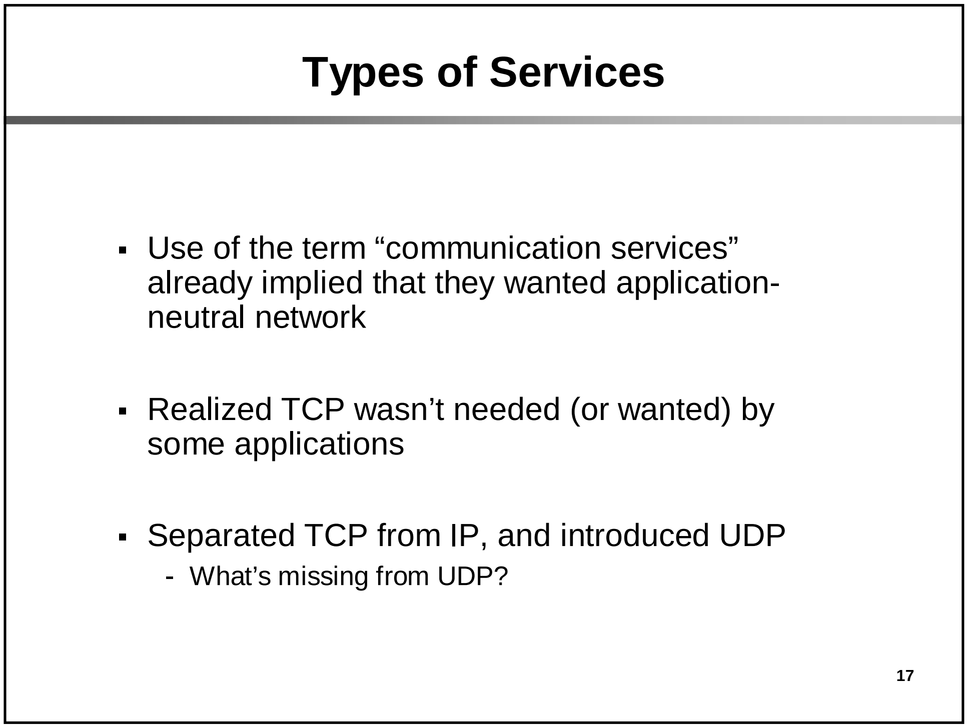
1. **Connect existing networks**
2. Robust in face of failures (not nuclear war…)
3. Support multiple types of services
4. Accommodate a variety of networks
5. Allow distributed management
6. Easy host attachment
7. Cost effective
8. Allow resource accountability

Robust

1. As long as the network is not partitioned, two endpoints should be able to communicate
2. Failures (excepting network partition) should not interfere with endpoint semantics (why?)

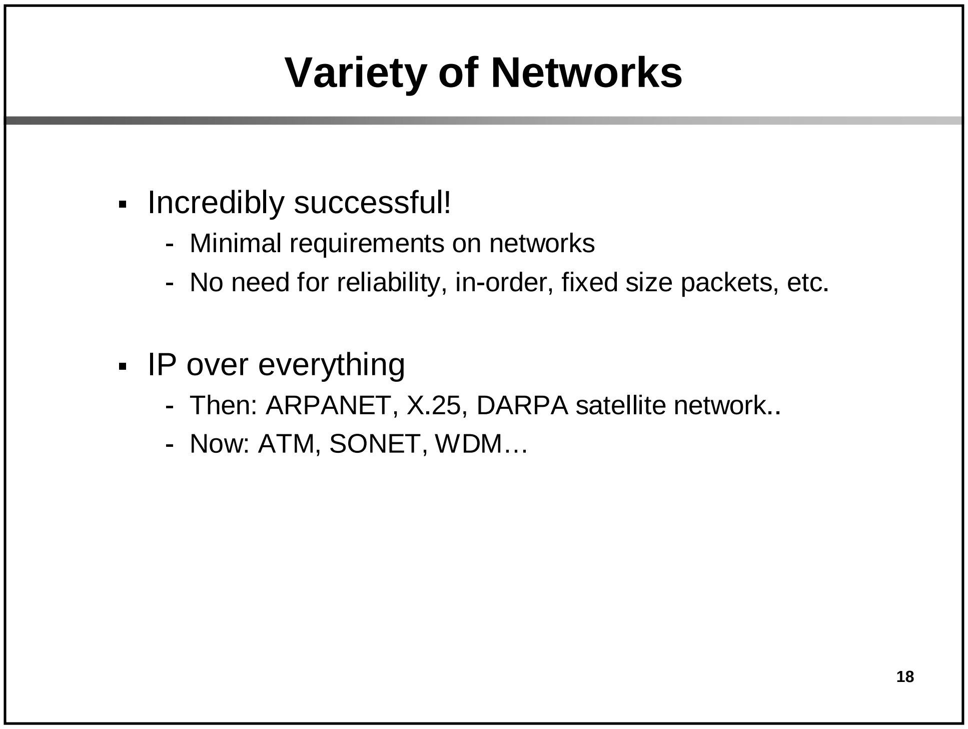
- Maintain state only at end-points
  - Fate-sharing, eliminates network state restoration
  - stateless network architecture (no per-flow state)

- Routing state is held by network (why?)
- No failure information is given to ends (why?)



Types of Services

- Use of the term “communication services” already implied that they wanted application-neutral network
- Realized TCP wasn’t needed (or wanted) by some applications
- Separated TCP from IP, and introduced UDP
  - What’s missing from UDP?



Variety of Networks

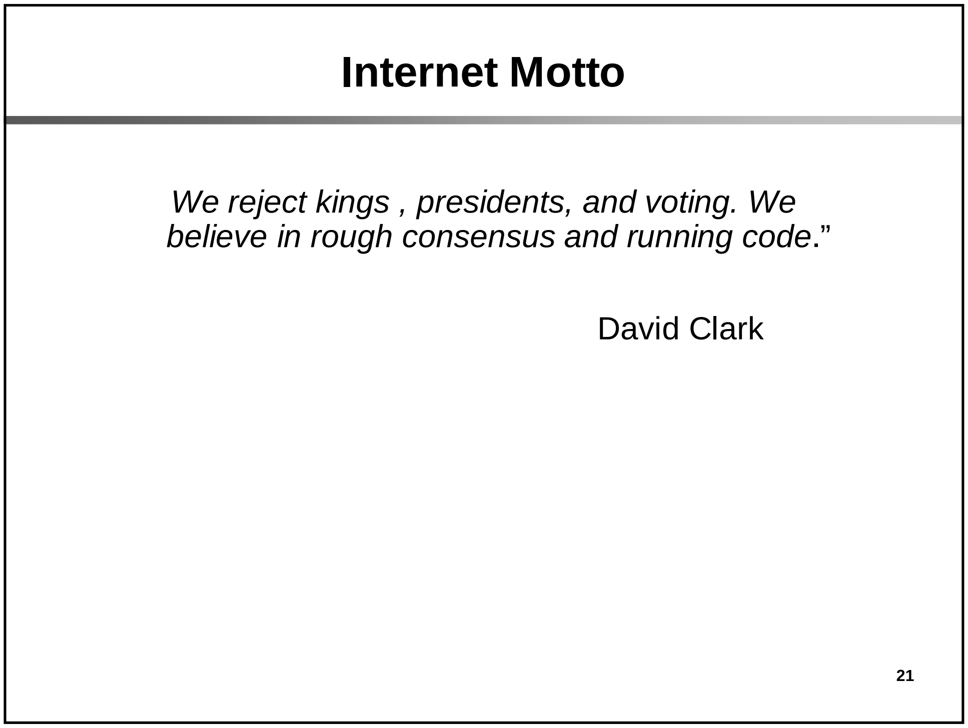
- Incredibly successful!
  - Minimal requirements on networks
  - No need for reliability, in-order, fixed size packets, etc.
- IP over everything
  - Then: ARPANET, X.25, DARPA satellite network..
  - Now: ATM, SONET, WDM…

Host Attachment

- Clark observes that the cost of host attachment may be somewhat higher because hosts have to be smart
- But the administrative cost of adding hosts is very low, which is probably more important

Why Datagrams?

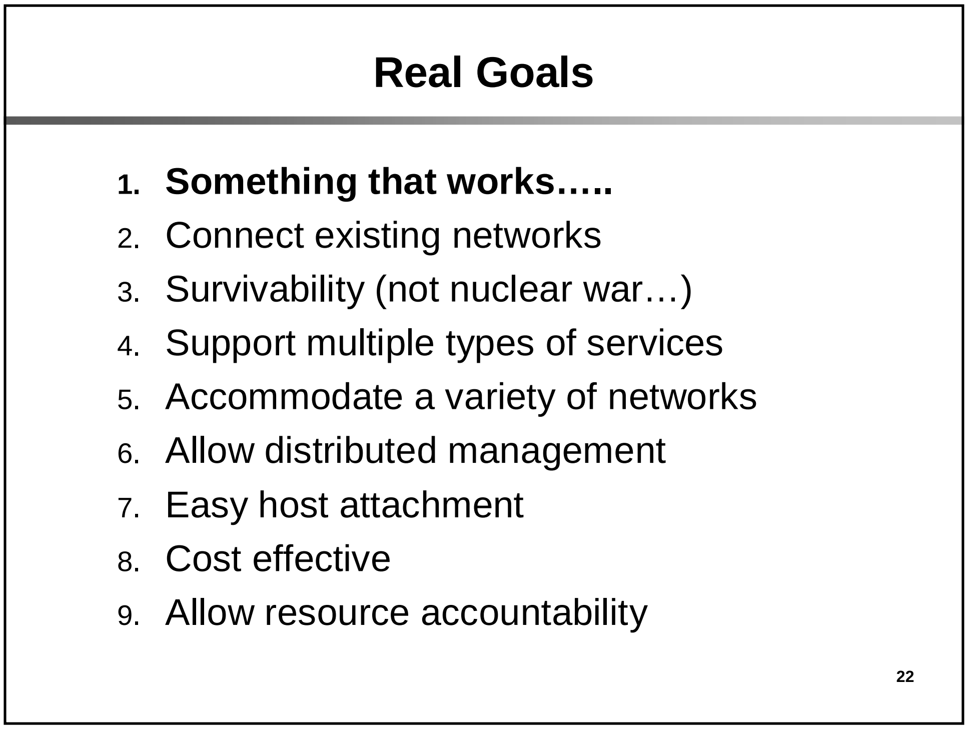
- No connection state needed
- Good building block for variety of services
- Minimal network assumptions



Internet Motto

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

David Clark



Real Goals

1. **Something that works…..**
2. Connect existing networks
3. Survivability (not nuclear war…)
4. Support multiple types of services
5. Accommodate a variety of networks
6. Allow distributed management
7. Easy host attachment
8. Cost effective
9. Allow resource accountability

Questions

- What priority order would a commercial design have?
- What would a commercially invented Internet look like?
- What goals are missing from this list?
- Which goals led to the success of the Internet?

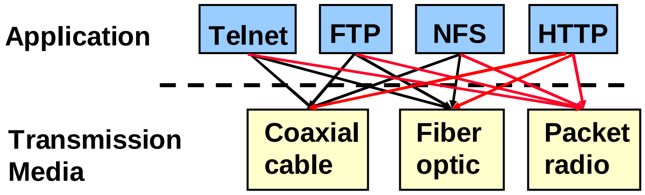
Layering and other General Mutterings about Internet Architecture

Repeats122 material

The Big Question

- Many different network styles and technologies
  - circuit-switched vs packet-switched, etc.
  - wireless vs wired vs optical, etc.
- Many different applications
  - ftp, email, web, P2P, etc.
- How do we organize this mess?

The Problem



Application

Telnet FTP NFS HTTP

Transmission Media

Coaxial cable

Fiber optic

Packet radio

- Do we re-implement every application for every technology?
- Obviously not, but how does the Internet architecture avoid this?

Architecture

- Architecture is not the implementation itself
- Architecture is how to “organize” implementations
  - what interfaces are supported
  - where functionality is implemented
- Architecture is the modular design of the network

Software Modularity

Break system into modules:

- Well-defined interfaces gives flexibility
  - can change implementation of modules
  - can extend functionality of system by adding new modules
- Interfaces hide information
  - allows for flexibility
  - but can hurt performance

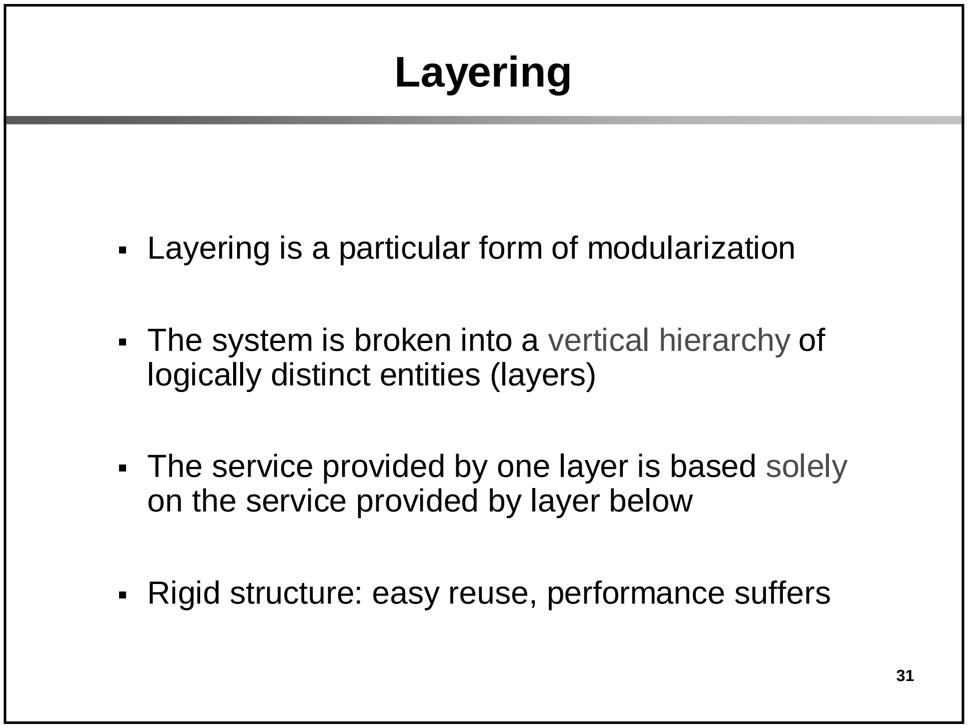
Network Modularity

Like software modularity, but with a twist:

- Implementation distributed across routers and hosts
- Must decide both:
  - how to break system into modules
  - where modules are implemented
- Lecture will address these questions in turn

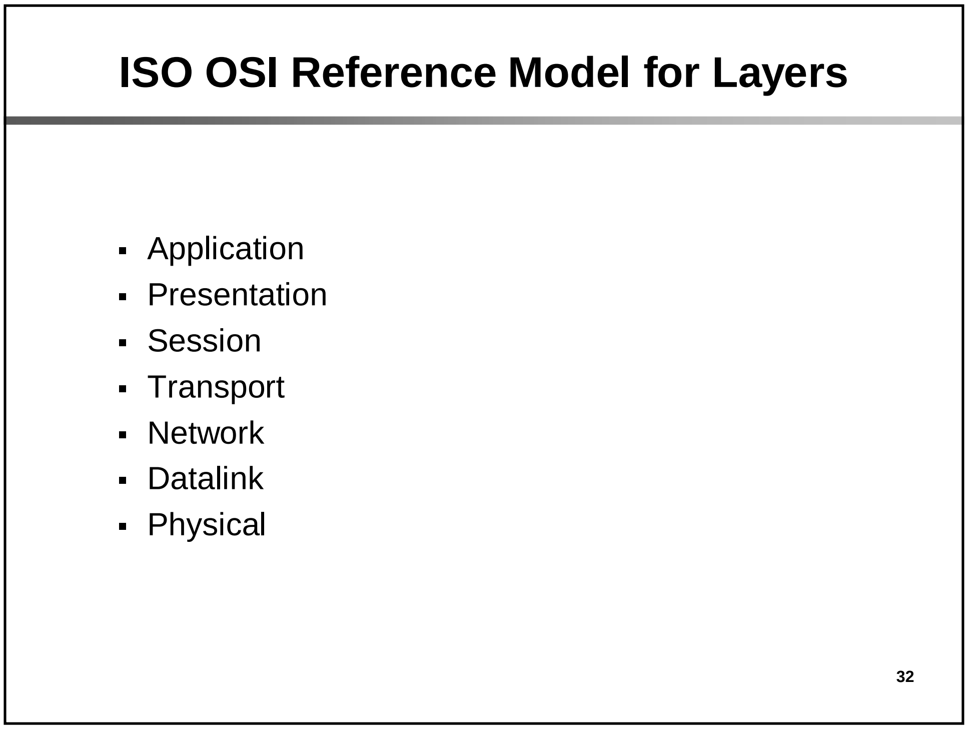
Two Aspects to Architecture

- Layering
  - how to break network functionality into modules
- The End-to-End Argument
  - where to implement functionality



Layering

- Layering is a particular form of modularization
- The system is broken into a vertical hierarchy of logically distinct entities (layers)
- The service provided by one layer is based solely on the service provided by layer below
- Rigid structure: easy reuse, performance suffers



ISO OSI Reference Model for Layers

- Application
- Presentation
- Session
- Transport
- Network
- Datalink
- Physical

Where Do These Fit?

- IP
- TCP
- Email
- Ethernet

Layering Solves Problem

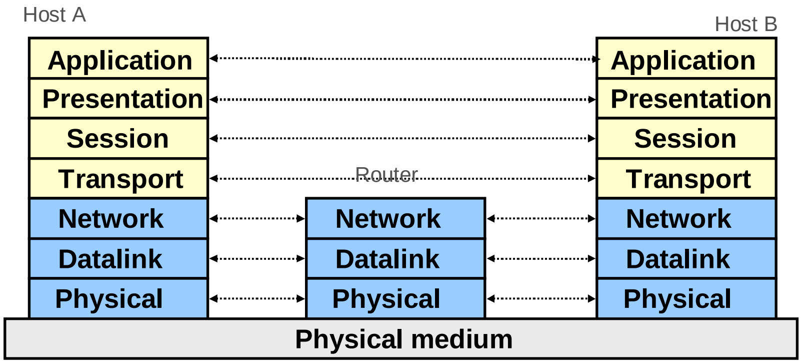
- Application layer doesn’t know about anything below the presentation layer, etc.
- Information about network is hidden from higher layers
- This ensures that we only need to implement an application once!

OSI Model Concepts

- Service: what a layer does
- Service interface: how to access the service
  - interface for layer above
- Peer interface (protocol): how peers communicate
  - a set of rules and formats that govern the communication between two network boxes
  - protocol does not govern the implementation on a single machine, but how the layer is implemented between machines

Who Does What?

- Seven layers
  - Lower three layers are implemented everywhere
  - Next four layers are implemented only at hosts



Host A

Host B

Router

Application

Presentation

Session

Transport

Network

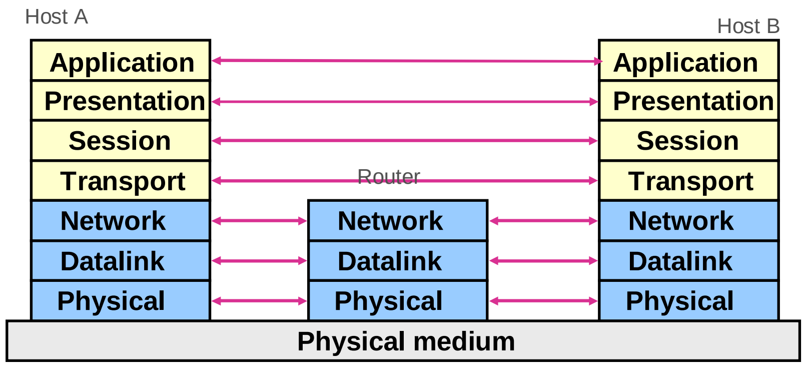
Datalink

Physical

Physical medium

Logical Communication

- Layers interacts with corresponding layer on peer

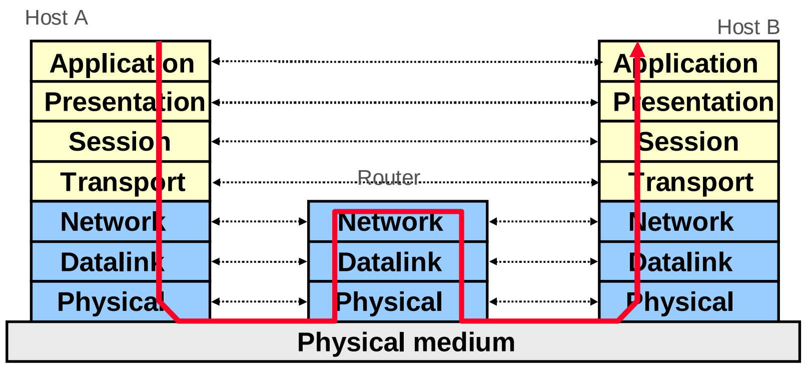


Host A	Router	Host B
Application		Application
Presentation		Presentation
Session		Session
Transport		Transport
Network	Network	Network
Datalink	Datalink	Datalink
Physical	Physical	Physical

Physical medium

Physical Communication

- Communication goes down to physical network, then to peer, then up to relevant layer

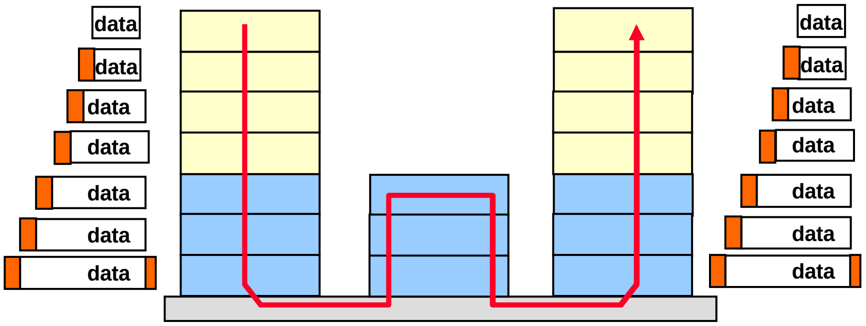


Host A	Router	Host B
Application		Application
Presentation		Presentation
Session		Session
Transport		Transport
Network	Network	Network
Datalink	Datalink	Datalink
Physical	Physical	Physical

Physical medium

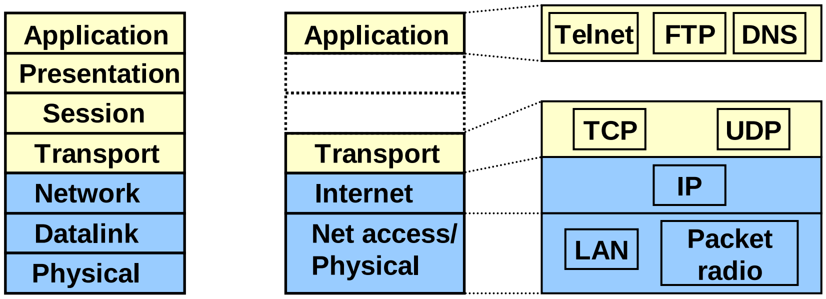
Encapsulation

- A layer can use only the service provided by the layer immediate below it
- Each layer may change and add a header to data packet



OSI vs. Internet

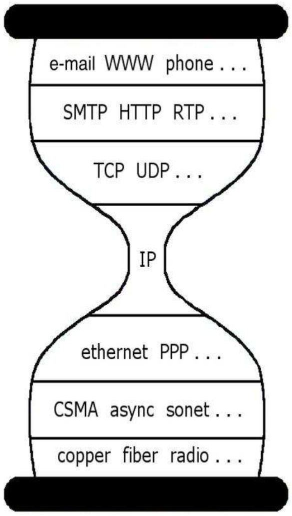
- OSI: conceptually define services, interfaces, protocols
- Internet: provide a successful implementation



OSI (formal)

Internet (informal)

Hourglass



41

Implications of Hourglass

A single Internet layer module:

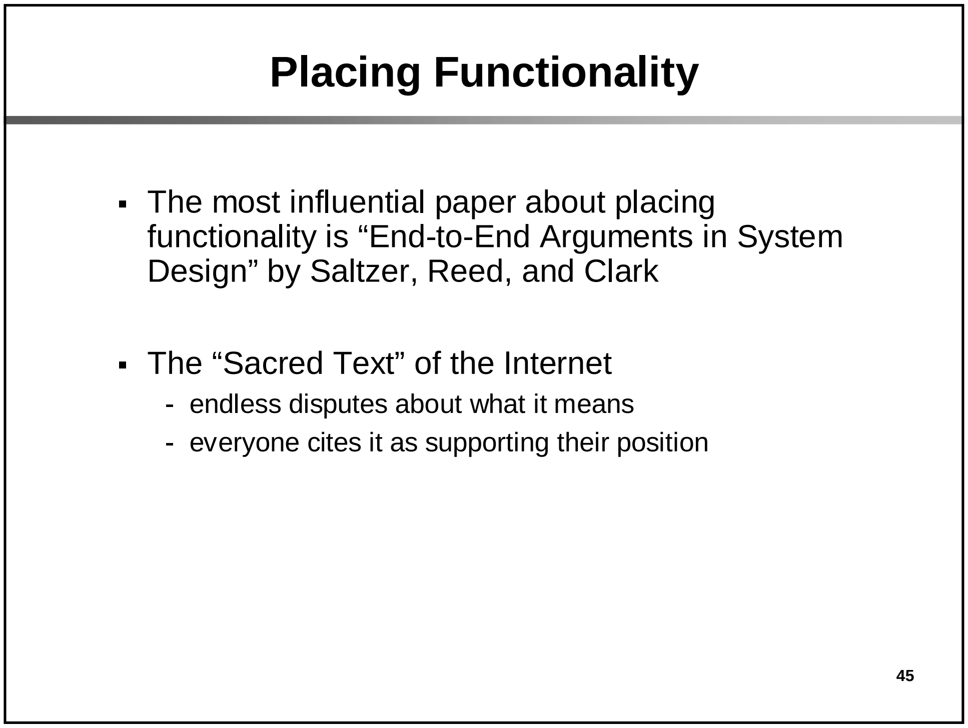
- Allows all networks to interoperate
  - all networks technologies that support IP can exchange packets
- Allows all applications to function on all networks
  - all applications that can run on IP can use any network
- Simultaneous developments above and below IP

42

Back to Reality

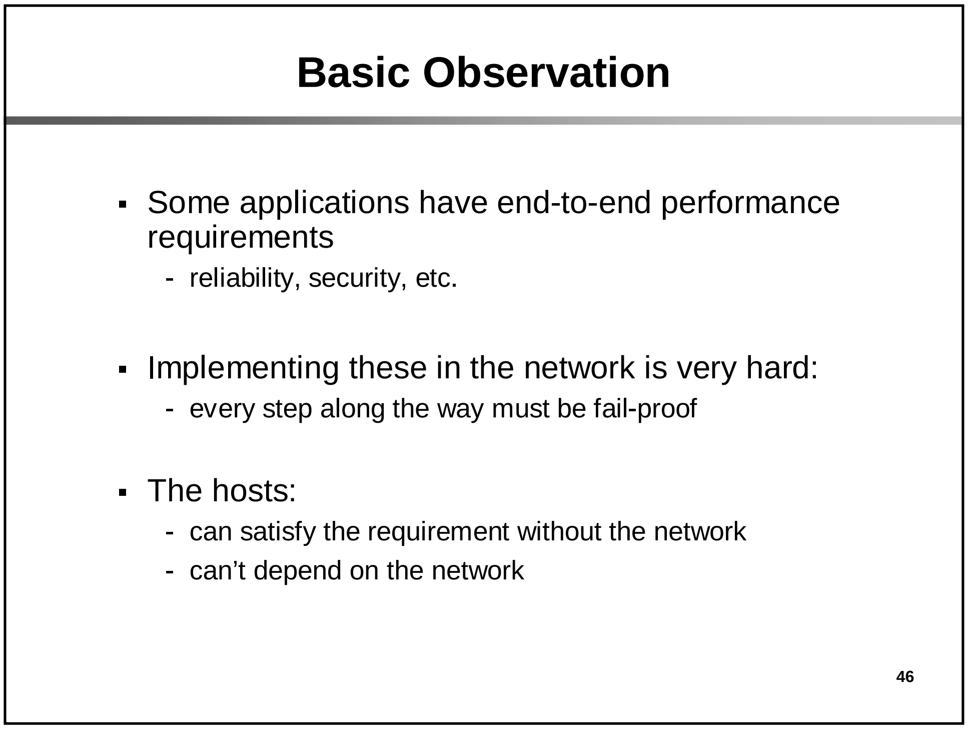
- Layering is a convenient way to think about networks
- But layering is often violated
  - Firewalls
  - Transparent caches
  - NAT boxes
  - ....
- What problems does this cause?
- What is an alternative to layers?

Endless Arguments about End-to-End



Placing Functionality

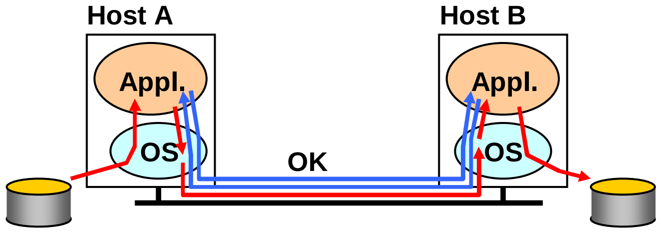
- The most influential paper about placing functionality is “End-to-End Arguments in System Design” by Saltzer, Reed, and Clark
- The “Sacred Text” of the Internet
  - endless disputes about what it means
  - everyone cites it as supporting their position



Basic Observation

- Some applications have end-to-end performance requirements
  - reliability, security, etc.
- Implementing these in the network is very hard:
  - every step along the way must be fail-proof
- The hosts:
  - can satisfy the requirement without the network
  - can’t depend on the network

Example: Reliable File Transfer



- Solution 1: make each step reliable, and then concatenate them
- Solution 2: end-to-end check and retry

Example (cont'd)

- Solution 1 not complete
  - What happens if any network element misbehaves?
  - The receiver has to do the check anyway!
- Solution 2 is complete
  - Full functionality can be entirely implemented at application layer with no need for reliability from lower layers
- Is there any need to implement reliability at lower layers?

Conclusion

Implementing this functionality in the network:

- Doesn't reduce host implementation complexity
- Does increase network complexity
- Probably imposes delay and overhead on all applications, even if they don't need functionality

- However, implementing in network can enhance performance in some cases
  - very lossy link

What the Paper Says

The function in question can completely and correctly be implemented only with the knowledge and help of the application standing at the end points of the communication system. Therefore, providing that questioned function as a feature of the communication system itself is not possible. (Sometimes an incomplete version of the function provided by the communication system may be useful as a performance enhancement.)

Conservative Interpretation

- “Don’t implement a function at the lower levels of the system unless it can be completely implemented at this level” (Peterson and Davie)
- Unless you can relieve the burden from hosts, then don’t bother

Radical Interpretations

- Don’t implement anything in the network that can be implemented correctly by the hosts
  - e.g., multicast
  - Makes network layer absolutely minimal
  - Ignores performance issues
- Don’t rely on anything that’s not on the data path
  - E.g., no DNS
  - Makes network layer more complicated

Moderate Interpretation

- Think twice before implementing functionality in the network
- If hosts can implement functionality correctly, implement it a lower layer only as a performance enhancement
- But do so only if it does not impose burden on applications that do not require that functionality

Extended Version of E2E Argument

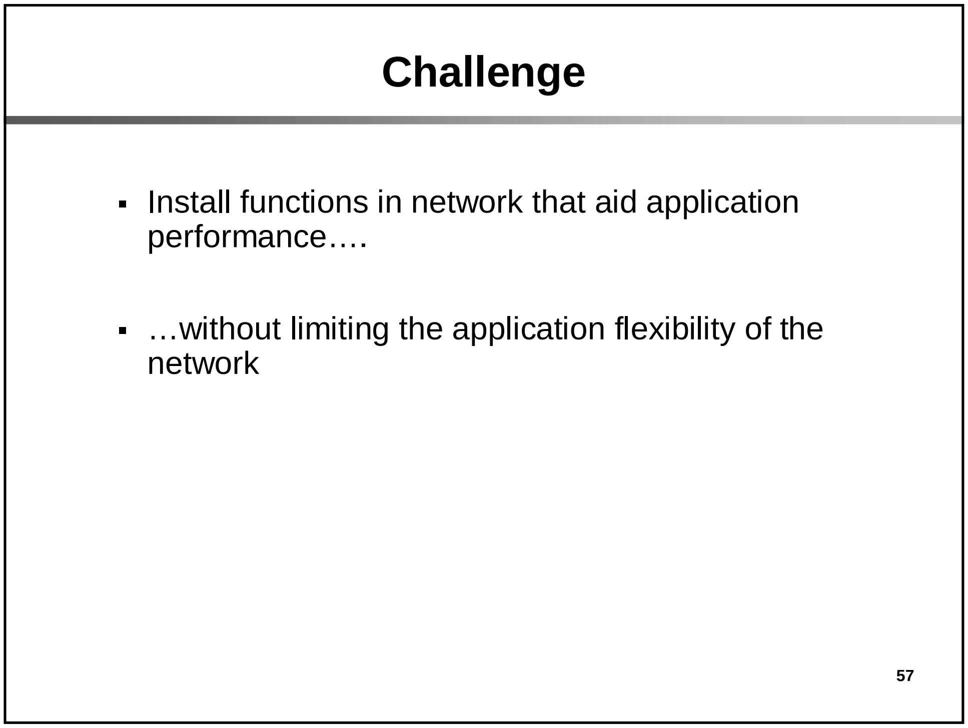
- Don’t put application semantics in network
  - Leads to loss of flexibility
  - Cannot change old applications easily
  - Cannot introduce new applications easily
- Normal E2E argument: performance issue
  - introducing more functionality imposes more overhead
  - subtle issue, many tough calls (e.g., multicast)
- Extended version:
  - short-term performance vs long-term flexibility

Do These Belong in the Network?

- Multicast?
- Routing?
- Quality of Service (QoS)?
- Name resolution? (is DNS “in the network”?)
- Web caches?

Back to Reality (again)

- Layering and E2E Principle regularly violated:
  - Firewalls
  - Transparent caches
  - Other middleboxes
- Battle between architectural purity and commercial pressures
  - extremely important
  - imagine a world where new apps couldn’t emerge



Challenge

- Install functions in network that aid application performance….
- …without limiting the application flexibility of the network