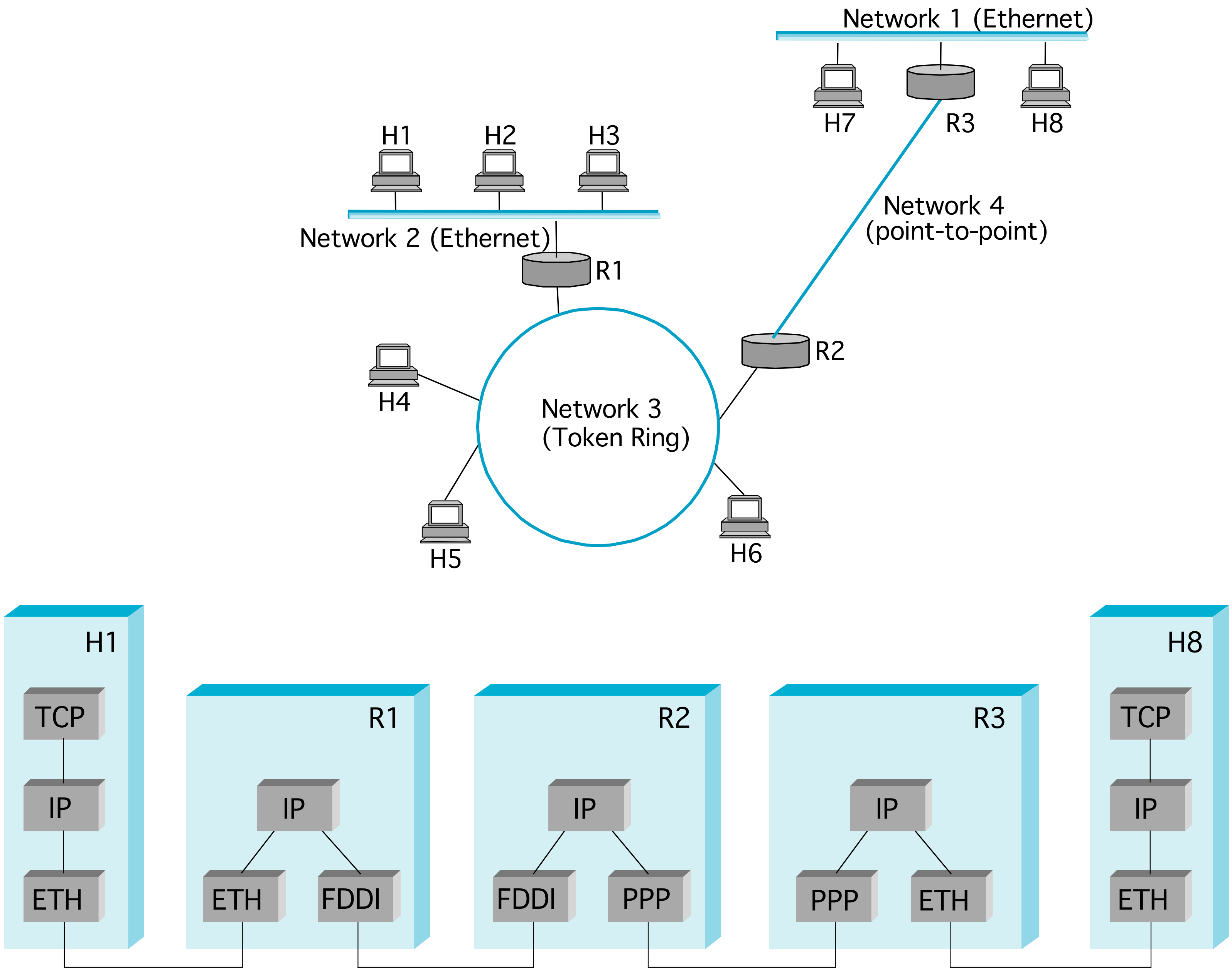
CSC 525:
Computer Networks

"Design Principles of the DARPA Internet Protocols"

- Why IP
  - A number of different network technologies developed in early 70's
  - Different media: copper, radio, satellite
  - Different protocol designs
  - Under different administrative control
  - The need for inter-connecting different nets.

Fundamental Goal

- To develop an effective technique for multiplexed utilization of all existing networks
  - no centralized control
  - no changes to individual networks
- Result
  - IP-based packet switched network consisting of distinct networks with store-and-forward gateways (routers) between them.
  - TCP/IP


Network 1 (Ethernet)
H7
R3
H8
H1
H2
H3
Network 4
(point-to-point)
Network 2 (Ethernet)
R1
R2
H4
Network 3
(Token Ring)
H5
H6
H1
TCP
IP
ETH
R1
IP
ETH
FDDI
R2
IP
FDDI
PPP
R3
IP
PPP
ETH
H8
TCP
IP
ETH

Second Level Design Goals

In the order of importance

- continued operation despite partial failures
- support multiple types of communication services
- accommodate a variety of networks
- distributed management of resources
- cost effective, performance
- low cost attachment
- resource accountability

What else would you add?

Trade-offs

- But not all the goals can be satisfied equally
  - Which goals win and which lose in the trade-off?
- The order of the goals is essential
  - Very strong focus on the first three
    - Survive network and router failures
    - Provide different types of services
    - Accommodate a variety of networks
  - A different order would produce a different design.
    - E.g., accounting barely works in the Internet.

Survivability

- Links and Routers may fail and stop working
  – (side note: design did not anticipate misbehavior)
- Two ends can continue communication
  – Despite faults at any intermediate point
  – Mask any transient failures (e.g. route changes)
- A network with only one failure mode: complete partition

Achieving Survivability

- Implications of storing network states
  - Any state stored at intermediate nodes must be replicated (since node may fail)
  - Difficult (at best) to design/implement.
- Solution: States are only stored at edges (hosts)
  - Stateless packet switches/routers (middle of network)
  - Consider TCP states: seq#, ack#, window size, etc are all stored at the hosts. No TCP information is stored in the intermediate routers.
  - Fate-Sharing: the fate of the communication is only shared with the fate of the two ends.

Soft states vs. Hard states.

- What if we do have to maintain some states in the network?
  - E.g., to provide Quality of Service guarantees.
- Soft states: periodic refresh regardless of whether there has been any change.
- Hard states: only update when there's a change.
- Another alternative: carry states in packets.
- Pros and cons?

Heterogeneity: Above IP

- To provide different types of services
  - TCP: reliable delivery
  - Debugging: low complexity, no reliability
  - Voice: delay bound, reliability sometimes hurts
- Leave the network layer simple
  - Split TCP (transport) and IP (network)
  - Connectionless datagram as basic building block
  - No assumption on subnet QoS capability
  - Build multiple services (transport protocols) at the hosts

Heterogeneity: Below IP

- To accommodate a variety of networks
- Least assumption on what subnets can do
  - Transport a packet
  - Reasonable packet size
  - Reasonable (not perfect) reliability
- Not assumed
  - Reliable or sequential delivery
  - Network broadcast or multicast
  - Priority or services
  - Failures, speeds, or delays
- Implement functionalities at the hosts

Other Goals

- Distributed Management
  - Success in allowing multiple domains and diverse routing policies.
  - Not very successful as the networks become large and complex.
- Performance
  - Header size, end-to-end retransmissions, routing table lookup.
- Adding Hosts and Nets
  - Speaking IP
  - Host software is complex
  - Relies on correct implementations/behaviors at the host
- Accounting
  - Challenging in the datagram model

Summary

- Identified and Prioritized Goals
  - Top three goals very successful
  - Bottom goals less successful
- Building Block: Datagram
  - Very effective for top goals (survivability, type of services, variety of networks)
  - Suggests that "flows" might be better for different priorities of goals
  - Suggests periodic messages: Soft-State

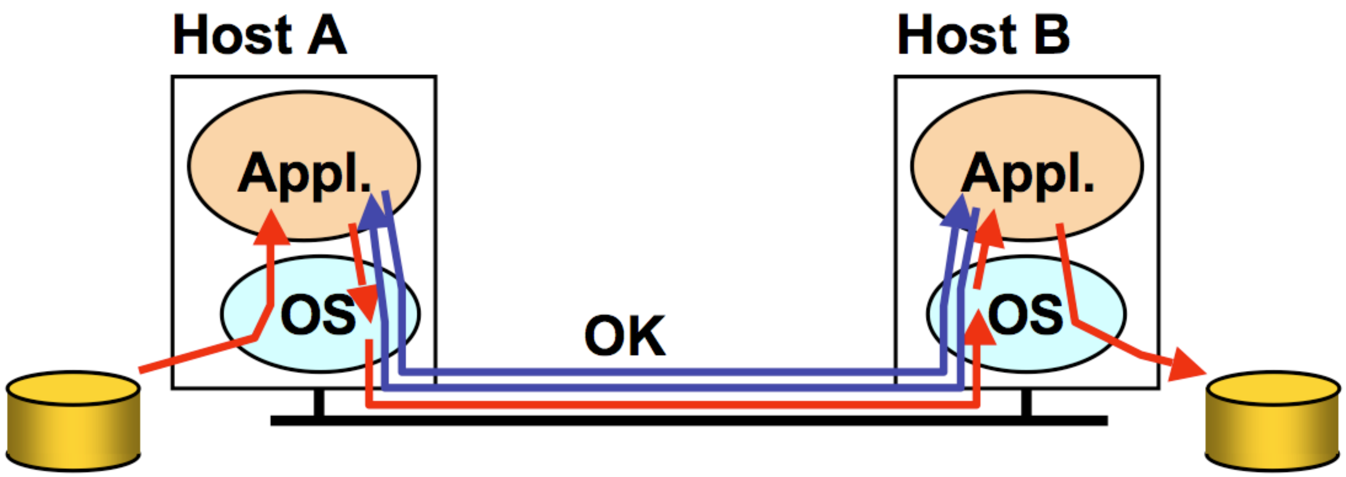
"End-to-End Arguments in System Design"

- Fact: layered network architecture
  - why layers: divide and conquer
- Downside of layering: potential duplication of functionalities across layers
  - example: how many layers are concerned with data transmission reliability?
- What functionality to put into which layer?

Basic Observation

- Some applications have end-to-end performance requirements
  - Reliability, delay bound, security etc.
- Implementing these in the network is very hard
  - Every step along the way must be fail-proof
  - Sometimes network doesn't have enough information
- The hosts
  - Can satisfy the requirements without the network
  - Cannot depend on the network

Example: Reliable File Transfer



- Errors can occur at many levels
  - from sender disk to memory
  - from sender OS to sender line card
  - across network
  - from receiver line card to receiver OS
  - from receiver memory to disk

Reliable File Transfer (II)

- Make each step reliable and concatenate them
  - What happens if any router fails or misbehaves?
  - The receiver has to do the check anyway!
- End-to-end check and retry
  - Full functionality can be implemented entirely by the applications with no need for reliability from lower layers.

Conclusion

- Implementing perfectly reliable network
  - Doesn't reduce host implementation complexity
  - Does increase network complexity
  - Probably imposes overhead on all applications, even if they don't need it, sometimes can hurt other applications (e.g. VoIP)
  - Provides false sense of security
- However, in some cases improving network reliability can enhance performance
    - e.g., over very lossy links

What does it mean

- Think twice before putting functionality into the network
- If hosts can implement it correctly, do it at a lower layer only as a necessary performance enhancement.
  - Consider marginal gain at lower layer
  - Consider the impact on other applications and services.

Extended Version

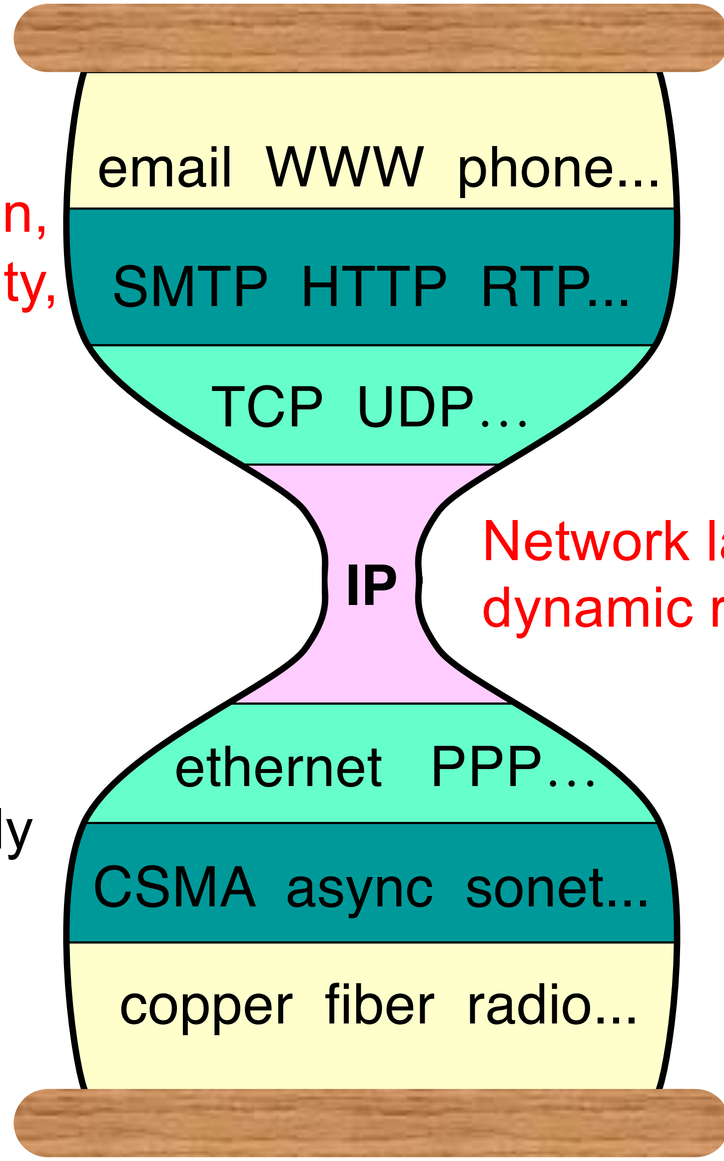
- Don't put application semantics in network
  - Leads to loss of flexibility
  - Cannot change old applications easily
  - Cannot introduce new applications easily
- Short-term application performance vs. long-term architectural flexibility
  - More functionality imposes more restrictions, but benefit known apps more.
  - Sometimes it's a tough call

The Erosion of the Architecture

- Layering and End-to-End principle are regularly violated by middle boxes in today's Internet:
  - Firewalls, NATs, ALG (application layer gateways)…
- Battle between architectural purity, commercial interests, and political pressures
  - ISPs
  - Vendors
  - Governments

The Result from the Design Principles

Transport layer: End to End communication, Multiplexing, Reliability, Congestion control, Flow control,



Network layer: global addressing, dynamic routing, best effort.

Data Link Layer: richly connected networks (many paths) with many types of links

So much for the principles

We reject kings, presidents and voting.
We believe in rough consensus and running code.
- Dave Clark