CE693: Adv. Computer Networking

L-2 Design Considerations

Fall 1391

Acknowledgments: Lecture slides are from the graduate level Computer Networks course thought by Srinivasan Seshan at CMU. When slides are obtained from other sources, a a reference will be noted on the bottom of that slide. A full list of references is provided on the last slide.

Lecture: Design Considerations

- How to determine split of functionality
  - Across protocol layers
  - Across network nodes
- Assigned Reading
  - [SRC84] End-to-end Arguments in System Design
  - [Cla88] Design Philosophy of the DARPA Internet Protocols
- Optional Reading
  - [CT90] Architectural Considerations for a New Generation of Protocols
  - [Clark02] Tussle in Cyberspace: Defining Tomorrow’s Internet

Outline

- Design principles in internetworks
- IP design

Goals [Clark88]

0 **Connect existing networks**

initially ARPANET and ARPA packet radio network

1. Survivability

ensure communication service even in the presence of network and router failures

2. Support multiple types of services
3. Must accommodate a variety of networks
4. Allow distributed management
5. Allow host attachment with a low level of effort
6. Be cost effective
7. Allow resource accountability

Goal 0: Connecting Networks

- How to internetwork various network technologies
  - ARPANET, X.25 networks, LANs, satellite networks, packet networks, serial links…
- Many differences between networks
  - Address formats
  - Performance – bandwidth/latency
  - Packet size
  - Loss rate/pattern/handling
  - Routing

Challenge 1: Address Formats

Challenge 1: Address Formats

- Map one address format to another?

Challenge 1: Address Formats

- Map one address format to another?
  - Bad idea → many translations needed

Challenge 1: Address Formats

- Map one address format to another?
  - Bad idea → many translations needed
- Provide one common format

Challenge 1: Address Formats

- Map one address format to another?
  - Bad idea → many translations needed
- Provide one common format
  - Map lower level addresses to common format

Challenge 2: Different Packet Sizes

- Define a maximum packet size over all networks?
  - Either inefficient or high threshold to support
- Implement fragmentation/re-assembly
  - Who is doing fragmentation?
  - Who is doing re-assembly?

Gateway Alternatives

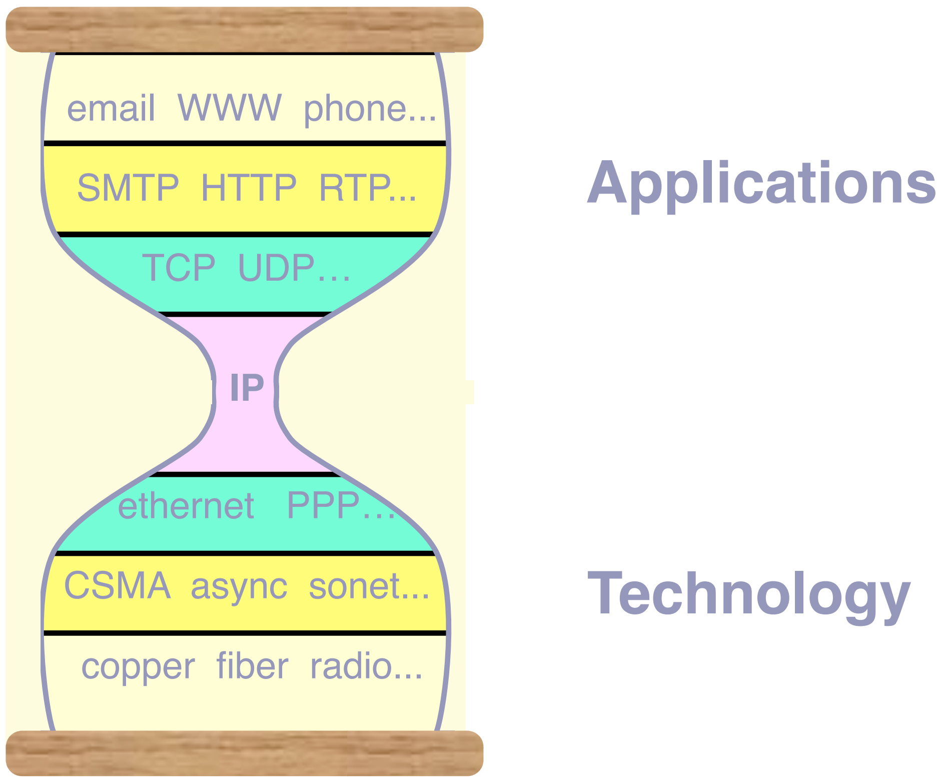
- Translation
  - Difficulty in dealing with different features supported by networks
  - Scales poorly with number of network types (N^2 conversions)
- Standardization
  - "IP over everything" (**Design Principle 1**)
  - Minimal assumptions about network
  - Hourglass design

IP Standardization

- Minimum set of assumptions for underlying net
  - Minimum packet size
  - Reasonable delivery odds, but not 100%
  - Some form of addressing unless point to point
- Important non-assumptions:
  - Perfect reliability
  - Broadcast, multicast
  - Priority handling of traffic
  - Internal knowledge of delays, speeds, failures, etc
- Also achieves Goal 3: Supporting Varieties of Networks

IP Hourglass

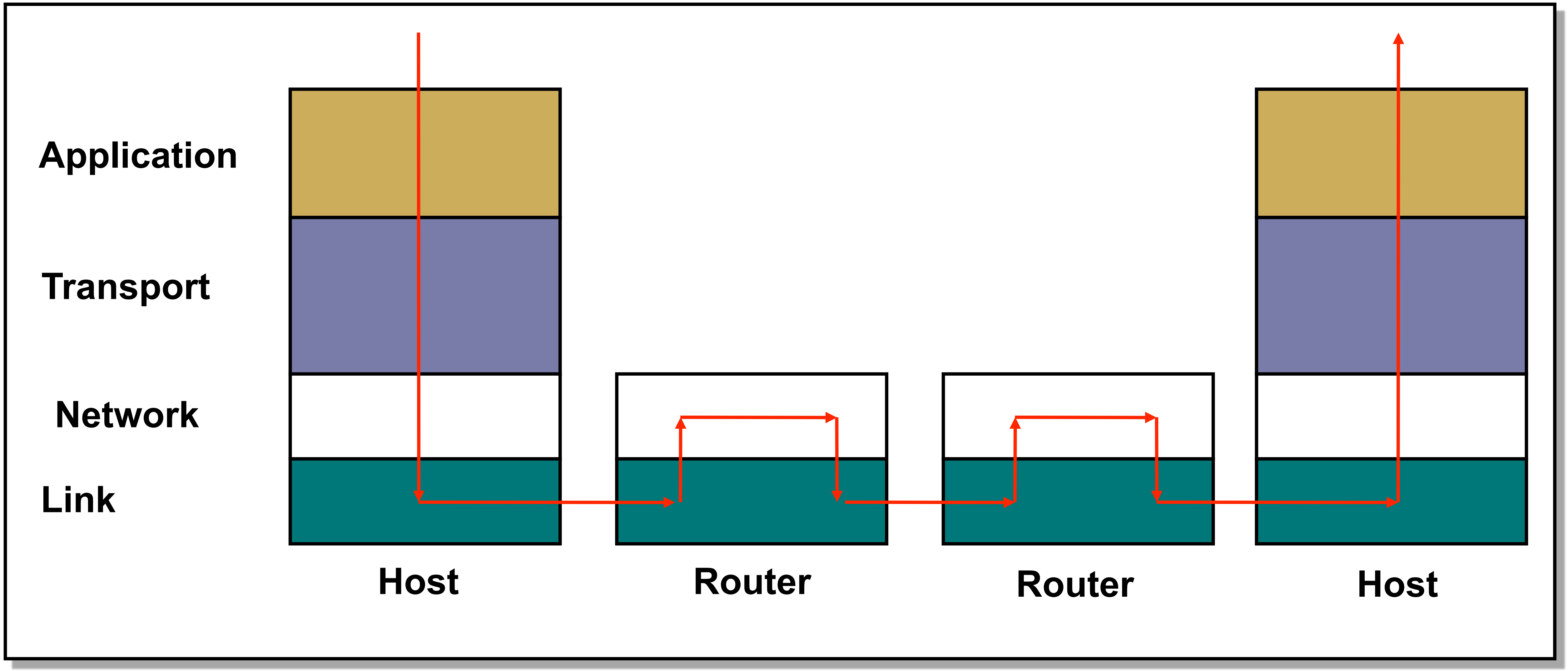
- Need to interconnect many existing networks
- Hide underlying technology from applications
- Decisions:
  - Network provides minimal functionality
  - “Narrow waist”



Tradeoff: **No assumptions, no guarantees.**

IP Layering (Principle 2)

- Relatively simple

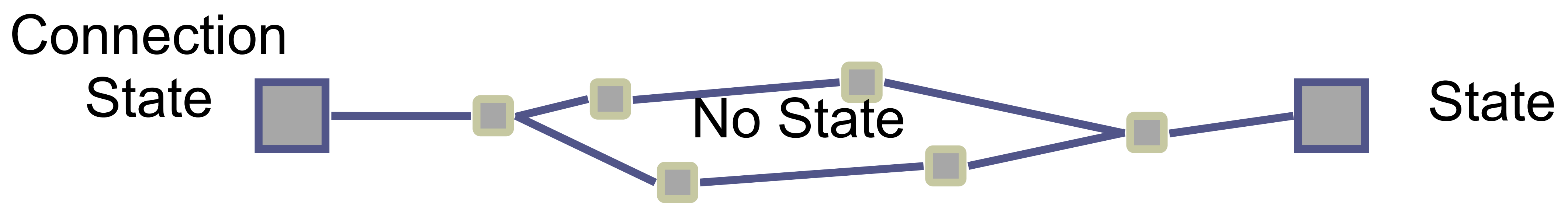


Survivability

- If network disrupted and reconfigured
  - Communicating entities should not care!
  - No higher-level state reconfiguration
- How to achieve such reliability?
  - Where can communication state be stored?

	Network	Host
Failure handing	Replication	"Fate sharing"
Net Engineering	Tough	Simple
Switches	Maintain state	Stateless
Host trust	Less	More

Principle 3: Fate Sharing



- Lose state information for an entity if and only if the entity itself is lost.
- Examples:
  - OK to lose TCP state if one endpoint crashes
    - NOT okay to lose if an intermediate router reboots
  - Is this still true in today's network?
    - NATs and firewalls
- Survivability compromise: Heterogeneous network → less information available to end hosts and Internet level recovery mechanisms

Principle 4: Soft-state

- Soft-state
  - Announce state
  - Refresh state
  - Timeout state
- Penalty for timeout – poor performance
- Robust way to identify communication flows
- Helps survivability

Principle 5: End-to-End Argument

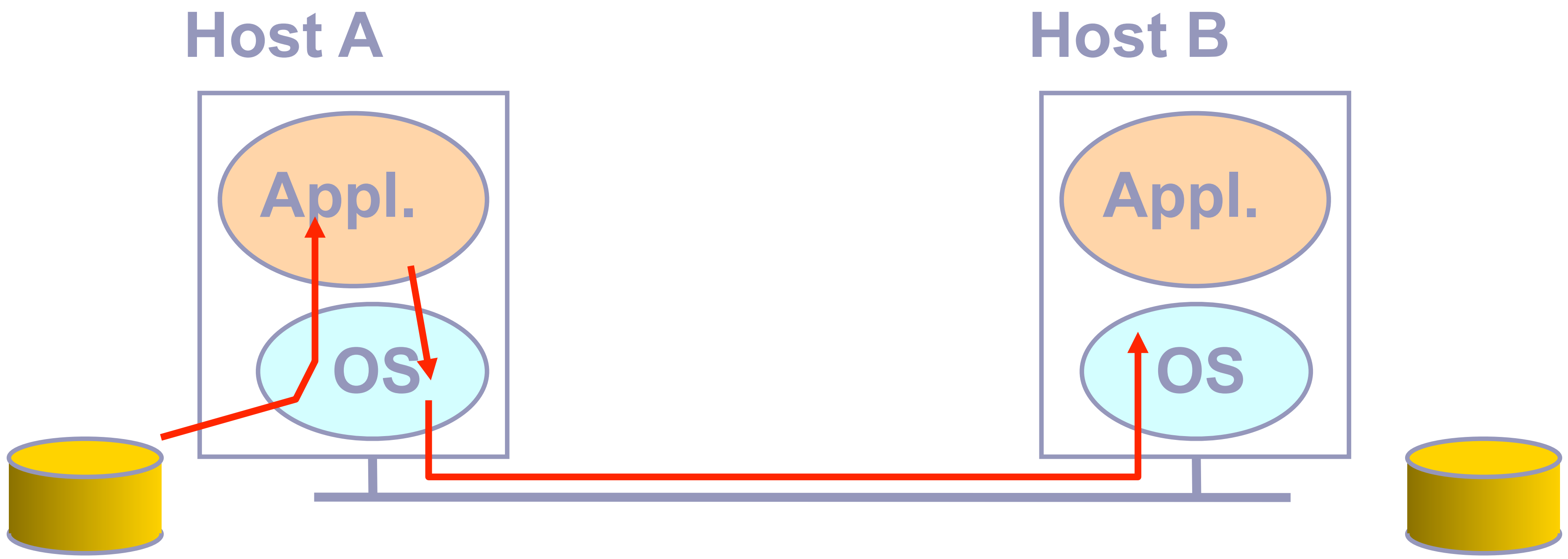
- Deals with where to place functionality
  - Inside the network (in switching elements)
  - At the edges
- Argument
  - There are functions that can only be correctly implemented by the endpoints – do not try to completely implement these elsewhere
  - Guideline not a law

Example: Reliable File Transfer

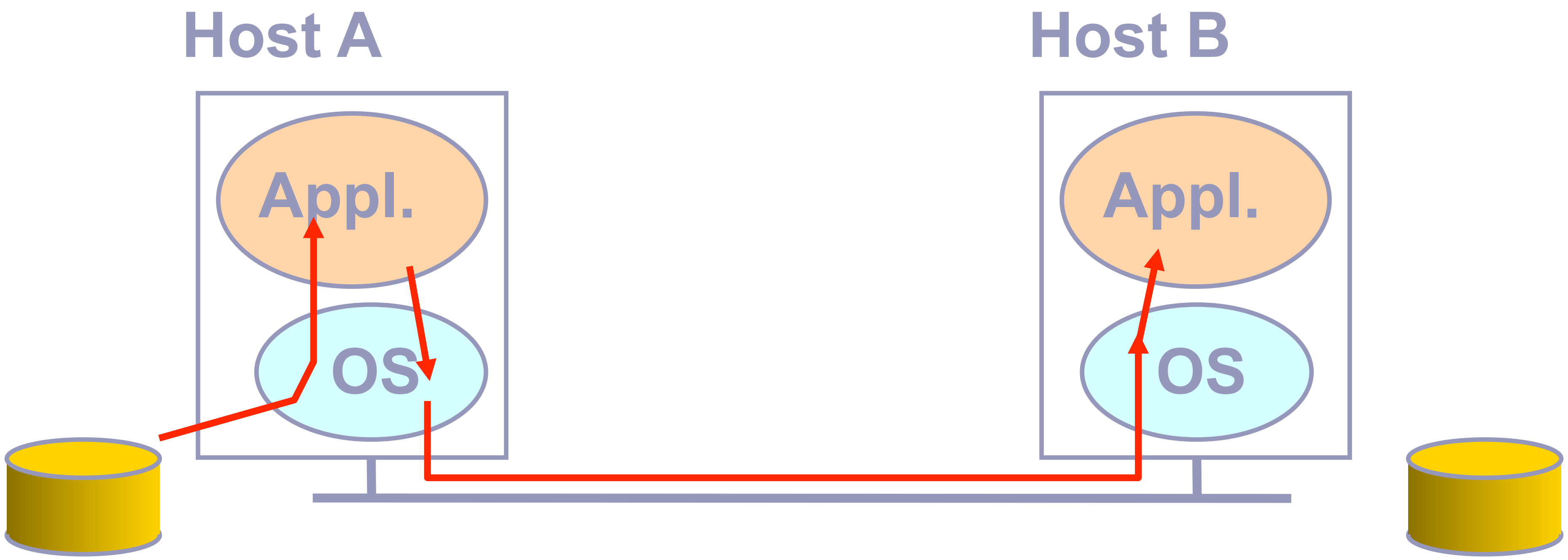
Example: Reliable File Transfer

Example: Reliable File Transfer

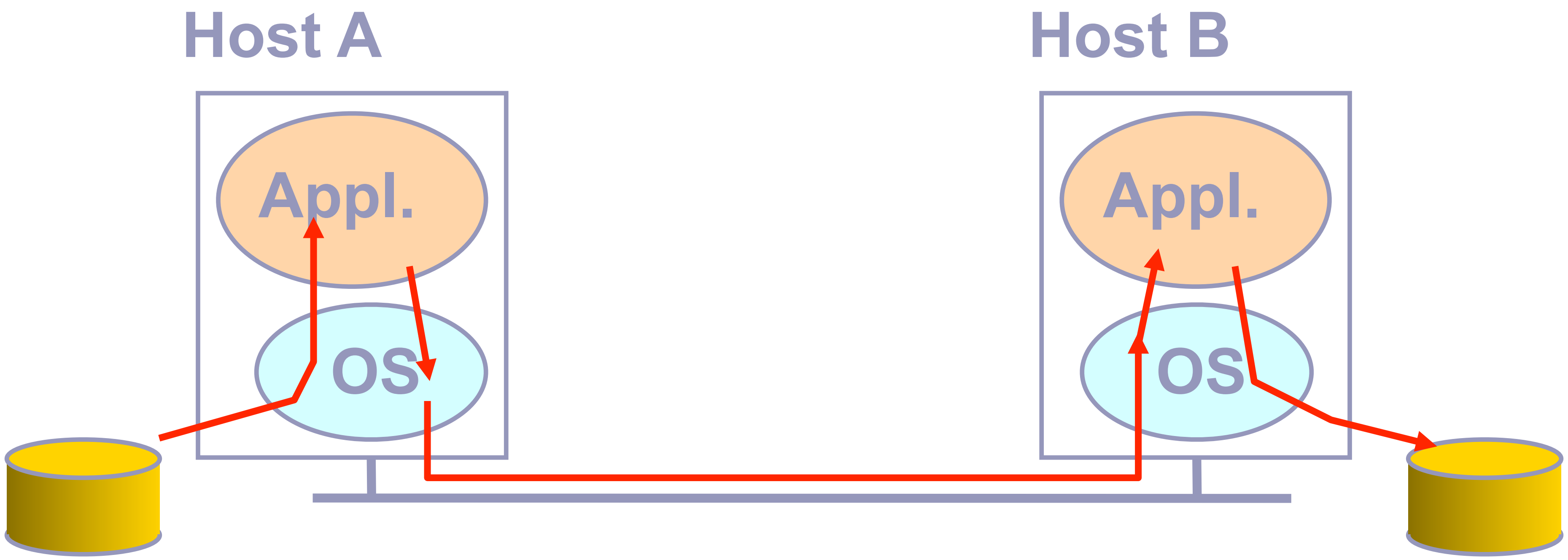
Example: Reliable File Transfer



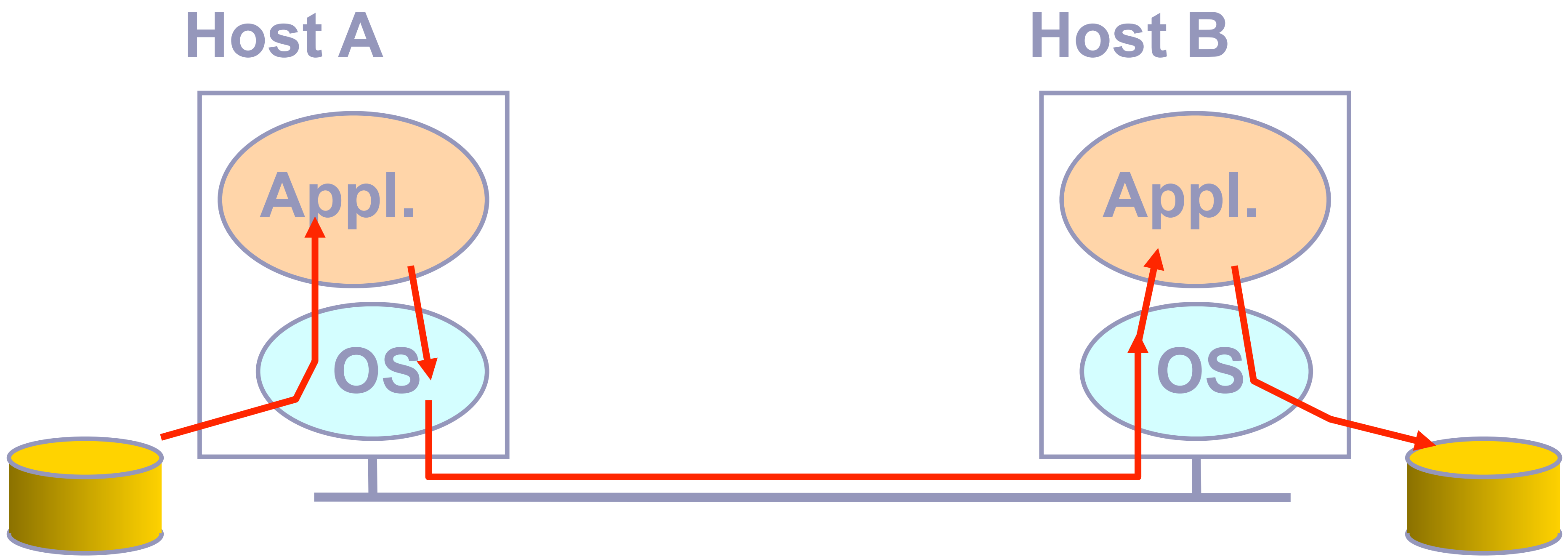
Example: Reliable File Transfer



Example: Reliable File Transfer

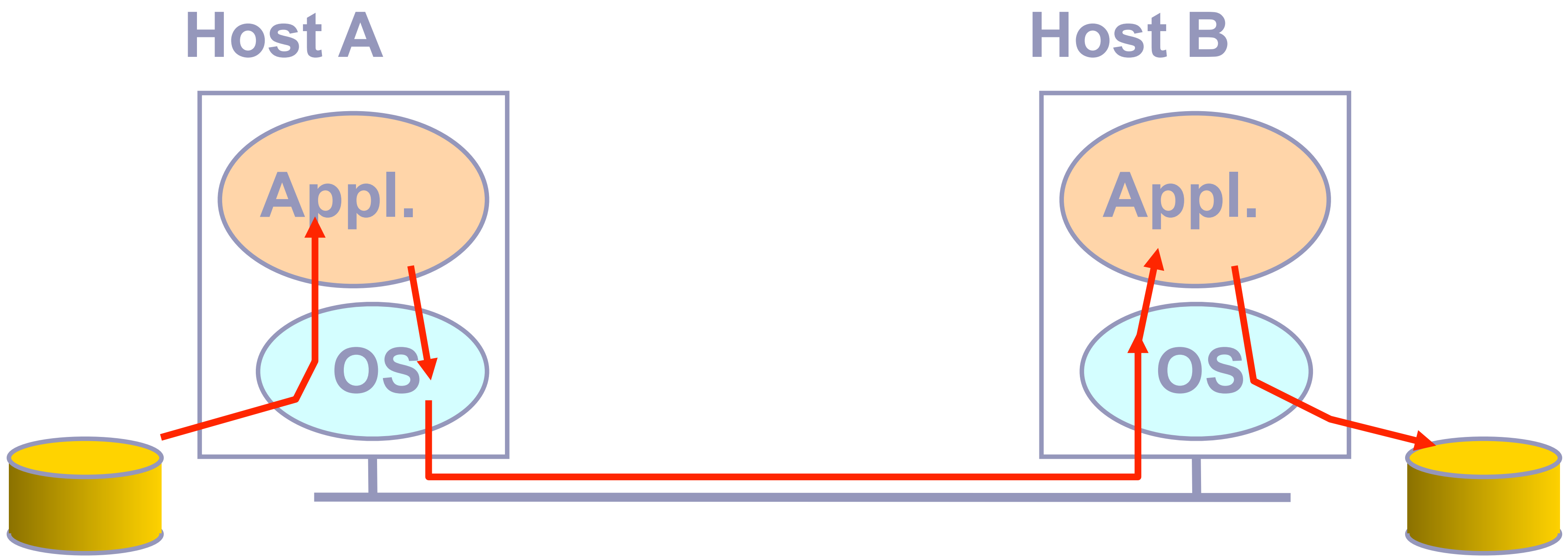


Example: Reliable File Transfer



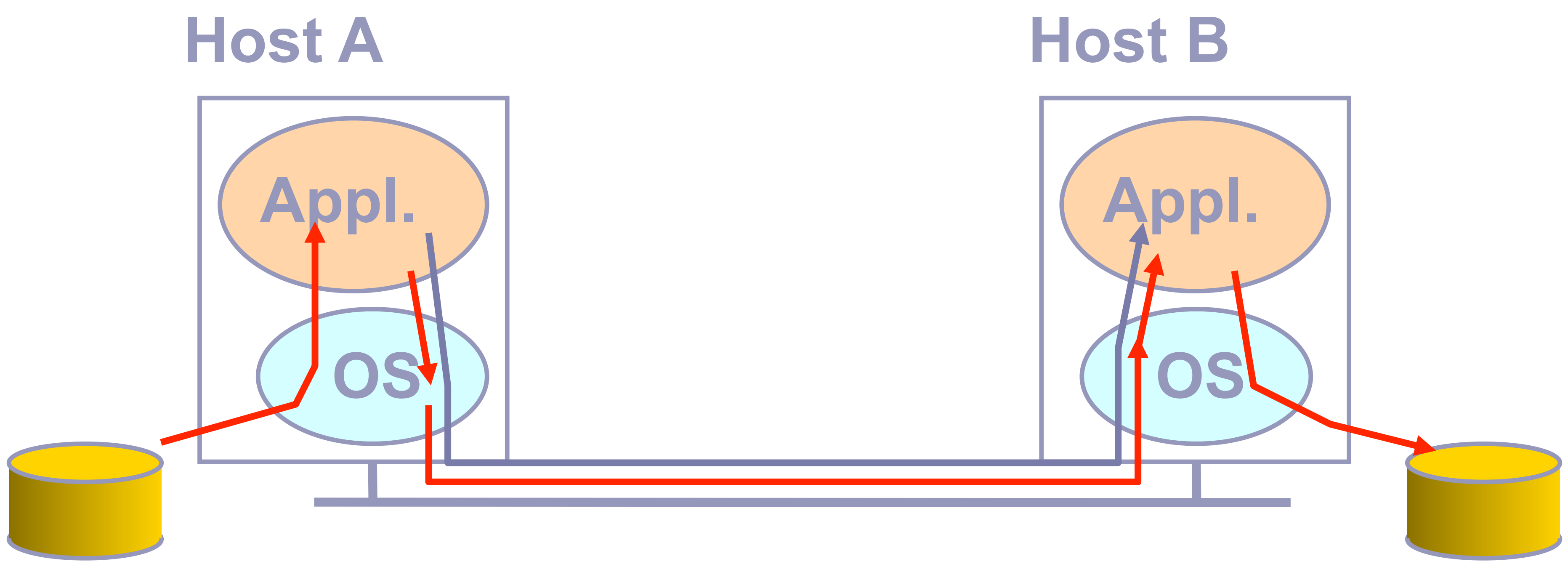
- Solution 1: make each step reliable, and then concatenate them

Example: Reliable File Transfer



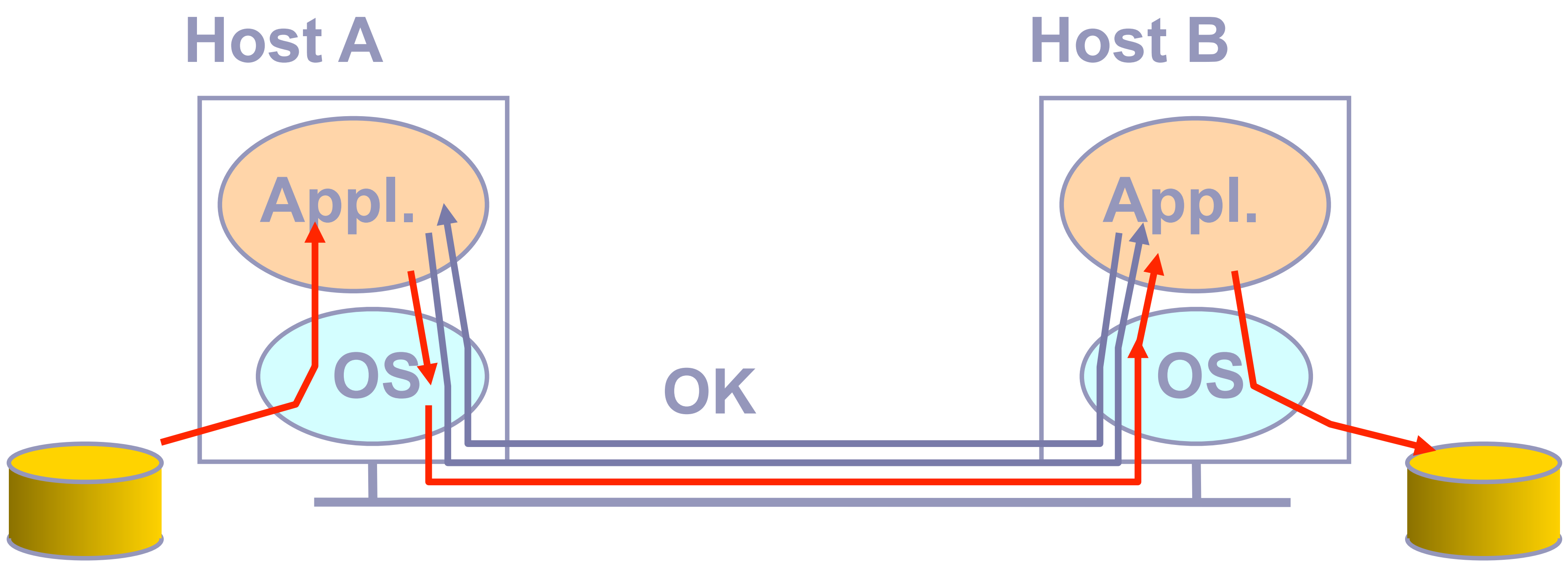
- Solution 1: make each step reliable, and then concatenate them
- Solution 2: end-to-end check and retry

Example: Reliable File Transfer



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

Example: Reliable File Transfer



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

E2E Example: File Transfer

E2E Example: File Transfer

- Even if network guaranteed reliable delivery

E2E Example: File Transfer

- Even if network guaranteed reliable delivery
  - Need to provide end-to-end checks

E2E Example: File Transfer

- Even if network guaranteed reliable delivery
  - Need to provide end-to-end checks
  - E.g., network card may malfunction

E2E Example: File Transfer

- Even if network guaranteed reliable delivery
  - Need to provide end-to-end checks
  - E.g., network card may malfunction
  - The receiver has to do the check anyway!

E2E Example: File Transfer

- Even if network guaranteed reliable delivery
  - Need to provide end-to-end checks
  - E.g., network card may malfunction
  - The receiver has to do the check anyway!
- Full functionality can only be entirely implemented at application layer; no need for reliability from lower layers

E2E Example: File Transfer

- Even if network guaranteed reliable delivery
  - Need to provide end-to-end checks
  - E.g., network card may malfunction
  - The receiver has to do the check anyway!
- Full functionality can only be entirely implemented at application layer; no need for reliability from lower layers

E2E Example: File Transfer

- Even if network guaranteed reliable delivery
  - Need to provide end-to-end checks
  - E.g., network card may malfunction
  - The receiver has to do the check anyway!
- Full functionality can only be entirely implemented at application layer; no need for reliability from lower layers

- Does FTP look like E2E file transfer?

E2E Example: File Transfer

- Even if network guaranteed reliable delivery
  - Need to provide end-to-end checks
  - E.g., network card may malfunction
  - The receiver has to do the check anyway!
- Full functionality can only be entirely implemented at application layer; no need for reliability from lower layers
- Does FTP look like E2E file transfer?
  - TCP provides reliability between kernels not disks

E2E Example: File Transfer

- Even if network guaranteed reliable delivery
  - Need to provide end-to-end checks
  - E.g., network card may malfunction
  - The receiver has to do the check anyway!
- Full functionality can only be entirely implemented at application layer; no need for reliability from lower layers
- Does FTP look like E2E file transfer?
  - TCP provides reliability between kernels not disks

E2E Example: File Transfer

- Even if network guaranteed reliable delivery
  - Need to provide end-to-end checks
  - E.g., network card may malfunction
  - The receiver has to do the check anyway!
- Full functionality can only be entirely implemented at application layer; no need for reliability from lower layers
- Does FTP look like E2E file transfer?
  - TCP provides reliability between kernels not disks
- Is there any need to implement reliability at lower layers?

Discussion

- Yes, but only to improve performance
- If network is highly unreliable
  - Adding some level of reliability helps performance, not correctness
  - Don’t try to achieve perfect reliability!
  - Implementing a functionality at a lower level should have minimum performance impact on the applications that do not use the functionality

Examples

- What should be done at the end points, and what by the network?
  - Reliable/sequenced delivery?
  - Addressing/routing?
  - Security?
  - What about Ethernet collision detection?
  - Multicast?
  - Real-time guarantees?

Goal 2: Types of Service

- **Principle 6**: network layer provides one simple service: best effort datagram (packet) delivery
  - All packets are treated the same

- Relatively simple core network elements
- Building block from which other services (such as reliable data stream) can be built
- Contributes to scalability of network

- No QoS support assumed from below
  - In fact, some underlying nets only supported reliable delivery
    - Made Internet datagram service less useful!
  - Hard to implement without network support
  - QoS is an ongoing debate…

Types of Service

- TCP vs. UDP
  - Elastic apps that need reliability: remote login or email
  - Inelastic, loss-tolerant apps: real-time voice or video
  - Others in between, or with stronger requirements
  - Biggest cause of delay variation: reliable delivery
    - Today's net: ~100ms RTT
    - Reliable delivery can add seconds.
- Original Internet model: "TCP/IP" one layer
  - First app was remote login…
  - But then came debugging, voice, etc.
  - These differences caused the layer split, added UDP

Goal 4: Decentralization

- **<u>Principle 7:</u>** Each network owned and managed separately
  - Will see this in BGP routing especially
- **<u>Principle 7':</u>** Be conservative in what you send and liberal in what you accept
  - Unwritten rule
- Especially useful since many protocol specifications are ambiguous
- E.g. TCP will accept and ignore bogus acknowledgements

The “Other” goals

5. Attaching a host
   - Host must implement hard part ☹ → transport services
     - Not too bad

6. Cost effectiveness
   - Packet overhead less important by the year
   - Packet loss rates low
   - Economies of scale won out
   - Internet cheaper than most dedicated networks

- But…

7. Accountability

- Huge problem
- Accounting
  - Billing? (mostly flat-rate. But phones have become that way also - people like it!)
  - Inter-ISP payments
    - Hornet's nest. Complicated. Political. Hard.
- Accountability and security
  - Huge problem.
  - Worms, viruses, etc.
    - Partly a host problem. But hosts very trusted.
  - Authentication
    - Purely optional. Many philosophical issues of privacy vs. security.
  - Greedy sources aren't handled well

Other IP Design Weaknesses

- Weak administration and management tools
- Incremental deployment difficult at times
  - Result of no centralized control
  - No more “flag” days
  - Are active networks the solution?

Changes Over Time

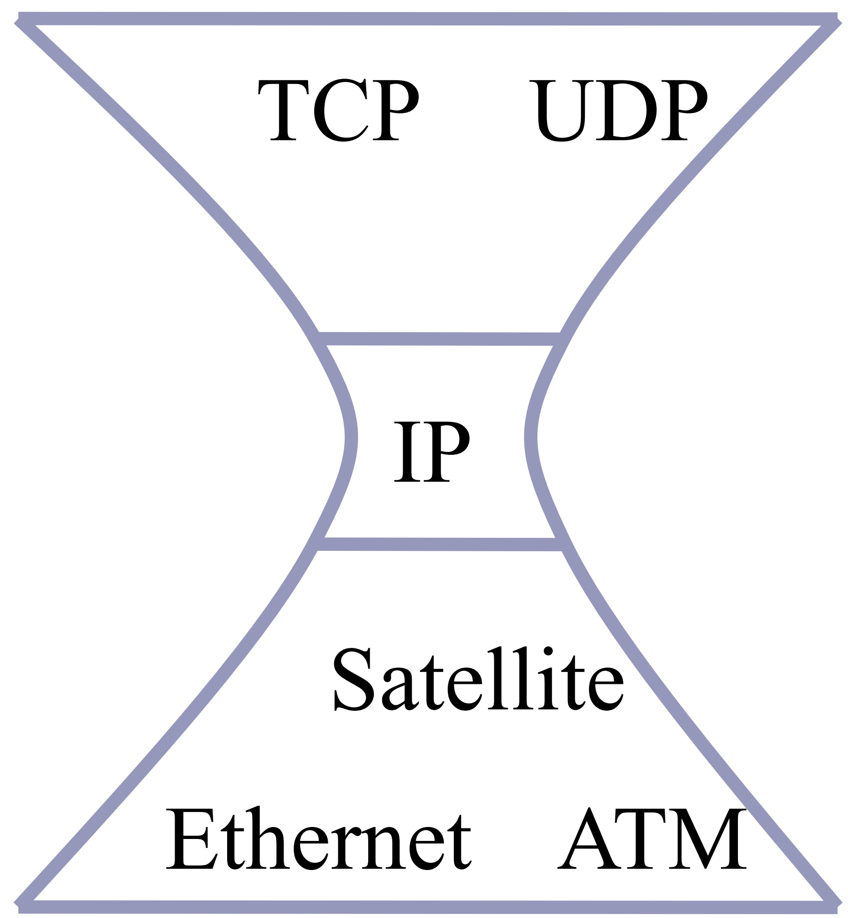
- Developed in simpler times
  - Common goals, consistent vision
- With success came multiple goals – examples:
  - ISPs must talk to provide connectivity but are fierce competitors
  - Privacy of users vs. government's need to monitor
  - User's desire to exchange files vs. copyright owners
- Must deal with the tussle between concerns in design

New Principles?

- Design for variation in outcome
  - Allow design to be flexible to different uses/results
- Isolate tussles
  - QoS designs uses separate ToS bits instead of overloading other parts of packet like port number
  - Separate QoS decisions from application/protocol design
- Provide choice → allow all parties to make choices on interactions
  - Creates competition
  - Fear between providers helps shape the tussle

Summary: Internet Architecture

- Packet-switched datagram network
- IP is the "compatibility layer"
  - Hourglass architecture
  - All hosts and routers run IP
- Stateless architecture
  - no per flow state inside network



Summary: Minimalist Approach

- Dumb network
  - IP provide minimal functionalities to support connectivity
    - Addressing, forwarding, routing
- Smart end system
  - Transport layer or application performs more sophisticated functionalities
    - Flow control, error control, congestion control
- Advantages
  - Accommodate heterogeneous technologies (Ethernet, modem, satellite, wireless)
  - Support diverse applications (telnet, ftp, Web, X windows)
  - Decentralized network administration

Summary

- Successes: IP on everything!

- Drawbacks…

  but perhaps they're totally worth it in the context of the original Internet. Might not have worked without them!

"This set of goals might seem to be nothing more than a checklist of all the desirable network features. It is important to understand that these goals are in order of importance, and **an entirely different network architecture would result if the order were changed**."

Goals [Clark88]

0 **Connect existing networks**

- initially ARPANET and ARPA packet radio network

1. Survivability
   - ensure communication service even in the presence of network and router failures
2. Support multiple types of services
3. Must accommodate a variety of networks
4. Allow distributed management
5. Allow host attachment with a low level of effort
6. Be cost effective
7. Allow resource accountability

Outline

- Design principles in internetworks
- IP design

Fragmentation

- IP packets can be 64KB
- Different link-layers have different MTUs
- Split IP packet into multiple fragments
  - IP header on each fragment
  - Various fields in header to help process
  - Intermediate router may fragment as needed
- Where to do reassembly?
  - End nodes – avoids unnecessary work
  - Dangerous to do at intermediate nodes
    - Buffer space
    - Multiple paths through network

Fragmentation is Harmful

- Uses resources poorly
  - Forwarding costs per packet
  - Best if we can send large chunks of data
  - Worst case: packet just bigger than MTU
- Poor end-to-end performance
  - Loss of a fragment
- Reassembly is hard
  - Buffering constraints

Path MTU Discovery

- Hosts dynamically discover minimum MTU of path
- Algorithm:
  - Initialize MTU to MTU for first hop
  - Send datagrams with Don't Fragment bit set
  - If ICMP "pkt too big" msg, decrease MTU
- What happens if path changes?
  - Periodically (>5mins, or >1min after previous increase), increase MTU
- Some routers will return proper MTU
- MTU values cached in routing table

IP Address Problem (1991)

- Address space depletion
  - In danger of running out of classes A and B
- Why?
  - Class C too small for most domains
  - Very few class A – IANA (Internet Assigned Numbers Authority) very careful about giving
  - Class B – greatest problem
    - Sparsely populated – but people refuse to give it back

IPv4 Routing Problems

- Core router forwarding tables were growing large
  - Class A: 128 networks, 16M hosts
  - Class B: 16K networks, 64K hosts
  - Class C: 2M networks, 256 hosts
- 32 bits does not give enough space encode network location information inside address – i.e., create a structured hierarchy

Solution 1 – CIDR

- Assign multiple class C addresses
- Assign consecutive blocks
- RFC1338 – Classless Inter-Domain Routing (CIDR)

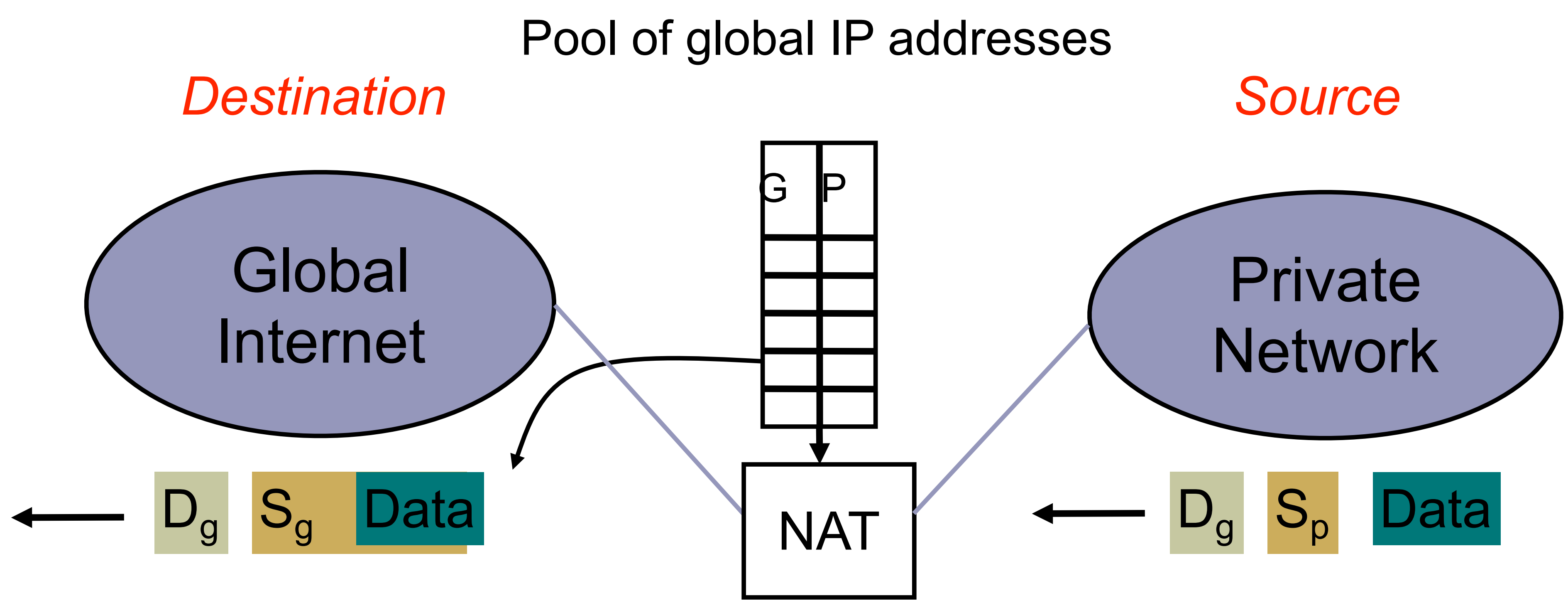
Classless Inter-Domain Routing

- Do not use classes to determine network ID
- Assign any range of addresses to network
  - Use common part of address as network number
  - e.g., addresses 192.4.16 - 196.4.31 have the first 20 bits in common. Thus, we use this as the network number
  - netmask is /20, /xx is valid for almost any xx
- Enables more efficient usage of address space (and router tables)

Solution 2 - NAT

- Network Address Translation (NAT)
- Alternate solution to address space
- Sits between your network and the Internet
- Translates local network layer addresses to global IP addresses
- Has a pool of global IP addresses (less than number of hosts on your network)

NAT Illustration



- **Operation: Source (S) wants to talk to Destination (D):**
  - Create S_g-S_p mapping
  - Replace S_p with S_g for outgoing packets
  - Replace S_g with S_p for incoming packets
- D & S can be just IP addresses or IP addresses + port #'s

Solution 3 - IPv6

- Scale – addresses are 128bit
  - Header size?
- Simplification
  - Removes infrequently used parts of header
  - 40byte fixed size vs. 20+ byte variable
- IPv6 removes checksum
  - Relies on upper layer protocols to provide integrity
- IPv6 eliminates fragmentation
  - Requires path MTU discovery
  - Requires 1280 byte MTU

IPv6 Changes

- TOS replaced with traffic class octet
- Flow
  - Help soft state systems
  - Maps well onto TCP connection or stream of UDP packets on host-port pair
- Easy configuration
  - Provides auto-configuration using hardware MAC address to provide unique base
- Additional requirements
  - Support for security
  - Support for mobility

Summary: IP Design

- Relatively simple design
  - Some parts not so useful (TOS, options)
- Beginning to show age
  - Unclear what the solution will be