CSE 486/586 Distributed Systems
The Internet in 2 Hours: The Second Hour

Steve Ko
Computer Sciences and Engineering
University at Buffalo

Recap

- The Internet
  - A network of networks
  - A case study as a distributed system
- Protocol
  - An agreement between multiple parties
  - Syntax & semantics
- Design a system
  - Why, what, and how
- The Internet
  - Connecting by layering

Layering: A Modular Approach

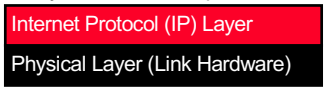
- Sub-divide the problem
  - Each layer relies on services from layer below
  - Each layer exports services to layer above
  - Each layer is designed to solve a specific, narrow set of problems
- Interface between layers defines interaction
  - Hides implementation details
  - Layers can change without disturbing other layers
- "The" computer science approach
  - ISA, OS, networking…

Back to Our How Question

- Question: how to send data from one machine to another that can be in different types of networks?
  - How to transfer data from one technology to another
  - How to ultimately deliver data from one machine to another
- How to transfer data from one technology to another
  - Designing a common interface that all hardware can translate to

Back to Our How Question

- Delivering data from one machine to another needs:
  - Addressing
  - Routing
- So in the end, the "connectivity" layer was designed to have those.
  - Interface for hardware
  - Addressing
  - Routing
- These two layers are the core part of the Internet



- Question: are we done?

Going back to the What Question

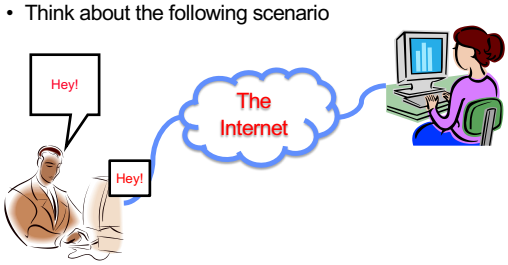
- What
  - Internet communication must continue despite loss of networks or gateways.
  - The Internet must support multiple types of communications service.
  - The Internet architecture must accommodate a variety of networks.
  - The Internet architecture must permit distributed management of its resources.
  - The Internet architecture must be cost effective.
  - The Internet architecture must permit host attachment with a low level of effort.
  - The resources used in the Internet architecture must be accountable.

Challenges in Layering

- What to put in a layer? How many layers do we need?
- Let's take an example: survivability
  - Protection of a conversation despite failures
  - Network hardware can fail (e.g., data loss) or behave erroneously (e.g., data duplication or reordering).
- Where does this go?
  - Should it be handled by the existing layers? If so, which one?
  - Should it have a different layer?
- This is an abstract question, so we need to bring it down to something more concrete.

So, How to Protect a Conversation?

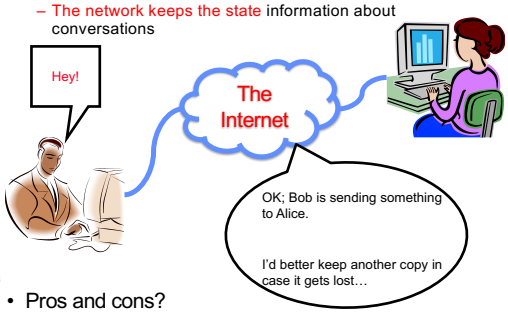
- Think about the following scenario



- Assume the Internet has two things: link hardware (physical layer) and addressing and routing (IP layer)
- A possible approach?

Two Approaches to Survivability

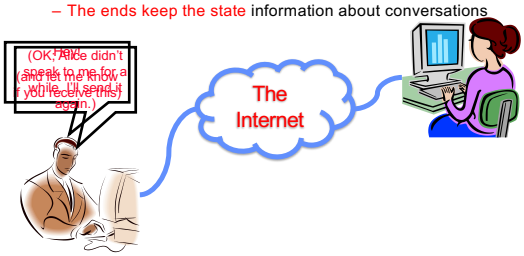
- Approach 1: "stateful" network
  - The network keeps the state information about conversations



- Pros and cons?
- Different approach?

Two Approaches to Survivability

- Approach 2: "stateless" network
  - The ends keep the state information about conversations

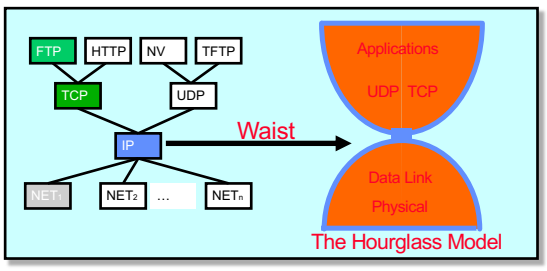


- Pros and cons?

Two Approaches to Survivability

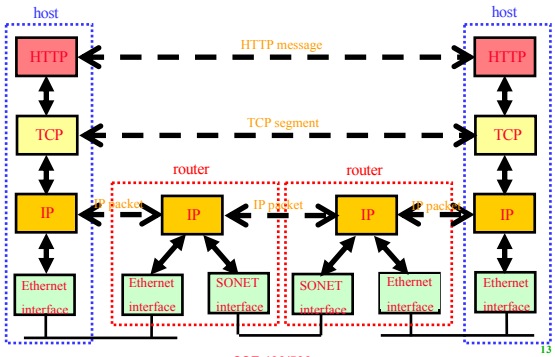
- Stateless networks' principle: fate-sharing
  - The conversation shares the same fate with the "ends."
  - *"it is acceptable to lose the state information associated with an entity if, at the same time, the entity itself is lost."*
- Advantages
  - Fate-sharing protects against any number of intermediate network failures
  - Fate-sharing is much easier to engineer.
- The result: a "best-effort" network
  - The IP (Internet Protocol) layer doesn't really provide anything other than "best-effort" delivery (i.e., addressing and routing).
  - The end hosts provide conversation protection mechanisms.
- Q: is today's Internet still stateless?

The Internet Protocol Suite



The waist facilitates interoperability

IP Suite: End Hosts vs. Routers



End-to-End Arguments

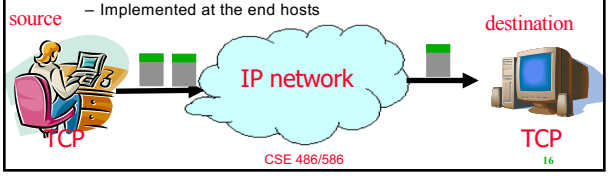
- If some functionality must be implemented end-to-end, then don't implement it in the network.
  - Exception: when there are clear performance improvements
- Helps resisting the tendency to put and hide complicated things in the lower layers
- Laid out in "*End-to-End Arguments in System Design*" by J.H. Saltzer, D.P. Reed and D.D. Clark (optional reading)
- A good rule of thumb in *any* system design, but still not something to follow blindly

CSE 486/586 Administrivia

- PA 1
  - Please try it out right away and see how far you can get.
  - We are not providing a lot of support and this is intentional. You have to use this to judge for yourself whether or not you want to stay in the course.
- Please use Piazza; all announcements will go there.
  - Please don't post as private (just for the instructors) unless it involves some private, sensitive information. Questions and answers benefit everybody in class.
  - Piazza is mainly a forum for students. Please help each other out by posting questions and relevant information as well as providing answers.
  - The teaching staff will mostly monitor Piazza activities, and not necessarily engage in actively answering questions.
  - If answers are incorrect, the teaching staff will make corrections.

TCP/IP

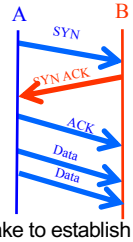
- IP "best-effort" network
  - The network knows the source and the destination.
  - A conversation is divided into packets.
  - Makes the best effort to deliver packets
  - Packet loss, corruption, out-of-order delivery, etc. could all happen.
- TCP (Transmission Control Protocol)
  - Handles the problems
  - Implemented at the end hosts



TCP

- An end-to-end protocol
- Protects conversations
  - Receiver is supposed to send an ack (acknowledgement) packet.
  - Packet loss → retransmission
  - Out-of-order delivery, duplicate packets → sequence numbers
  - Packet corruption → checksum
- Controls congestion
  - The network might be over-utilized
  - Prevents the network from collapsing (which was actually a concern in the late 80's)
- TCP is an abstraction: a reliable, byte-stream connection

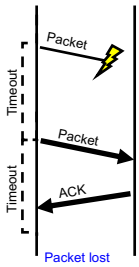
A (Very) Brief Overview of TCP



- Three-way handshake to establish connection
  - Host A sends a **SYN** (open) to the host B
  - Host B returns a SYN acknowledgment (**SYN ACK**)
  - Host A sends an **ACK** to acknowledge the SYN ACK
- Why 3-way instead of 2-way?
  - Reachability

Retransmission

- Timeout & retransmission to handle packet loss



The Dark Side of TCP

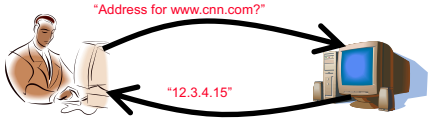
- There's overhead associated.
  - Connection establishment: 3-way handshake
  - Packet loss: retransmission timeout
  - Congestion control: doesn't utilize full bandwidth
- More importantly, some applications do not need these.
- Examples?
- So, enter UDP (User Datagram Protocol): exposes almost exactly what IP can give you.

Why Would Anyone Use UDP?

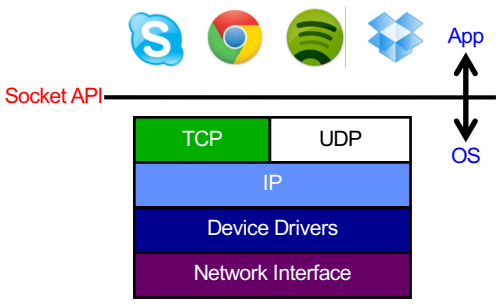
- Fine control over what data is sent and when
  - As soon as an application process writes
  - … UDP will package the data and send the packet
- No delay for connection establishment
  - UDP just blasts away without any formal preliminaries
  - … which avoids introducing any unnecessary delays
- No connection state
  - No allocation of buffers, parameters, sequence #s, etc.
  - … making it easier to handle many active clients at once
- Small packet header overhead
  - UDP header is only eight-bytes long

Popular Applications That Use UDP

- Multimedia streaming
  - Retransmitting lost/corrupted packets is not worthwhile
  - By the time the packet is retransmitted, it's too late
  - E.g., telephone calls, video conferencing, gaming
- Simple query protocols like Domain Name System
  - Overhead of connection establishment is overkill
  - Easier to have the application retransmit if needed
  - Will cover this in a separate lecture



What Applications See



Summary

- What to put on top of physical networks?
  - Layers providing survivability
- Where to put functionalities?
  - Fate-sharing & end-to-end arguments
  - IP layer doesn't provide much
  - TCP handles most of the survivability issues
- TCP & UDP: the two transport protocols of the Internet
- What interface do applications see?
  - Socket API
- Next: An introduction to Android programming

Acknowledgements

- These slides contain material developed and copyrighted by
  - Indranil Gupta at UIUC
  - Mike Freedman and Jen Rexford at Princeton