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

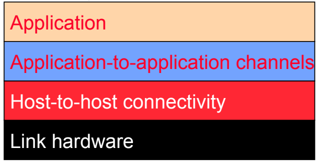
Steve Ko
Computer Sciences and Engineering
University at Buffalo

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
- Interface between layers defines interaction
  - Hides implementation details
  - Layers can change without disturbing other layers
- "The" computer science approach
  - ISA, OS, networking…


Application
Application-to-application channels
Host-to-host connectivity
Link hardware

Challenges in Layering

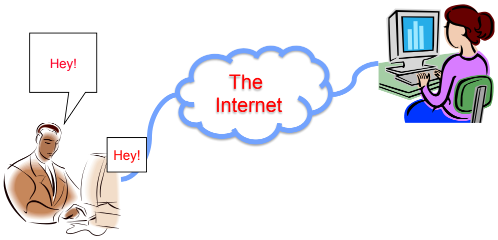
- What to put on top of physical networks?
- Assumption (for the sake of the discussion):
  - Packet switching (a conversation is divided into smaller units called packets).
- Basic things for enabling a conversation between remote hosts:
  - Addressing (where do I send a msg?)
  - Routing (how do I reach that address?)
- Most importantly, survivability
  - Protection of a conversation *as long as* there's a physical path between entities communicating and they are alive.
- What are some of the threats that disrupt a conversation?
  - Packet loss, out-of-order delivery, duplicate packets, etc.

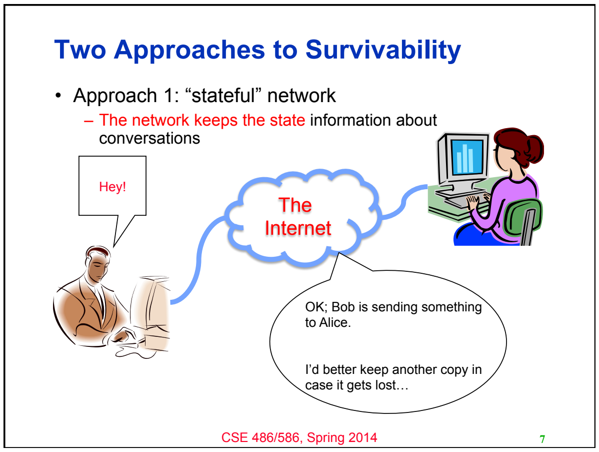
We Must Ask Ourselves…

- In a conversation, there are two components involved
  - Hosts
  - Network
- So, one more question: where do we want to put the functionalities? More specifically, what would be a good network/host division of labor?
- Addressing and routing?
  - Yeah, probably in the network
- What about conversation protection mechanisms?
  - The network or hosts?

So, How to Protect a Conversation?

- Think about the following scenario

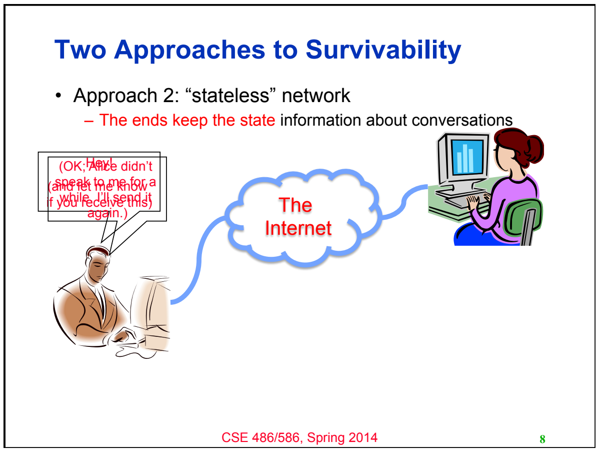




Two Approaches to Survivability

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

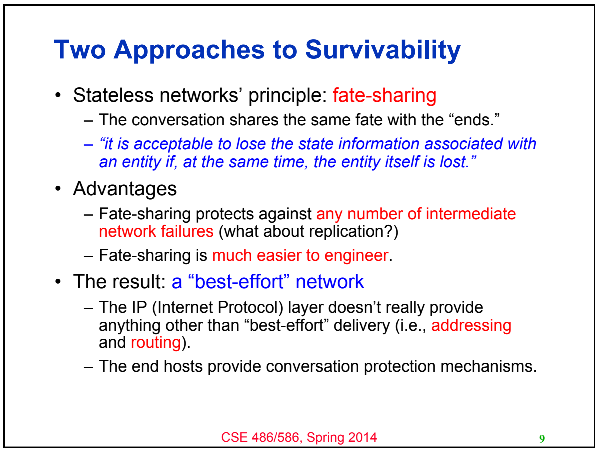
Hey!
The Internet
OK; Bob is sending something to Alice.
I’d better keep another copy in case it gets lost…



Two Approaches to Survivability

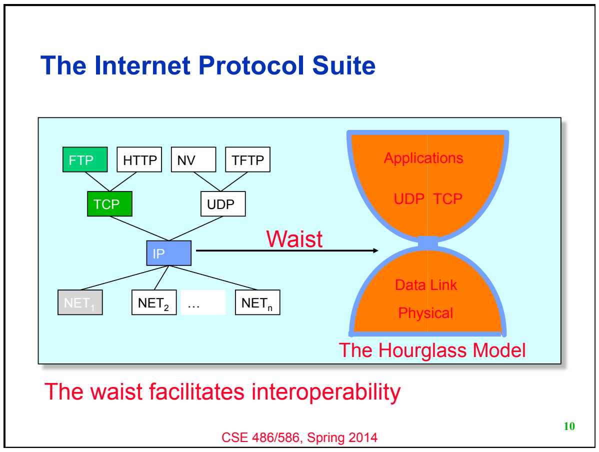
- Approach 2: “stateless” network
  - The ends keep the state information about conversations

Hey!
(and let me know if you receive this)
(OK; Alice didn’t speak to me for a while. I’ll send it again.)
The Internet



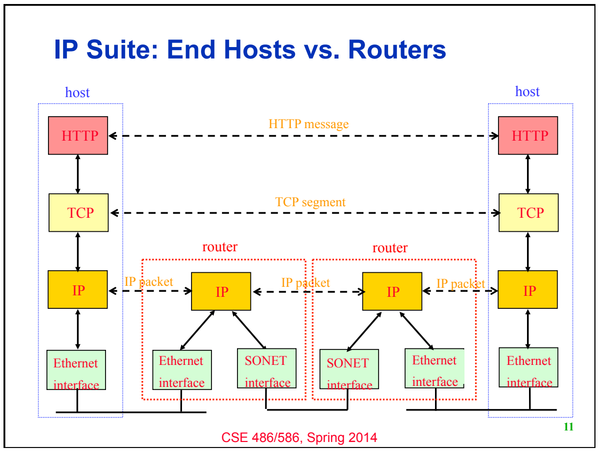
Two Approaches to Survivability

- 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 (what about replication?)
  - 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.

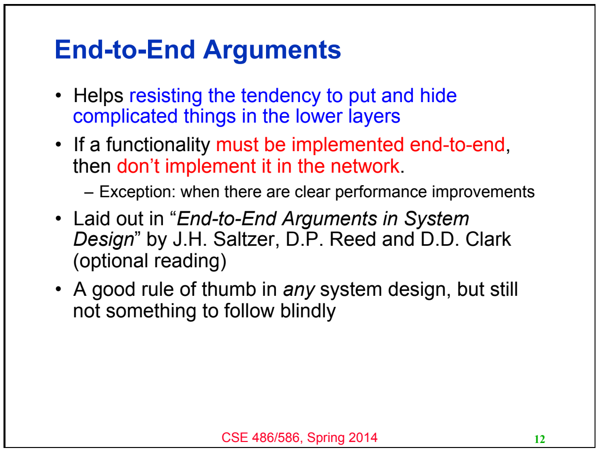


The Internet Protocol Suite

The waist facilitates interoperability



IP Suite: End Hosts vs. Routers



End-to-End Arguments

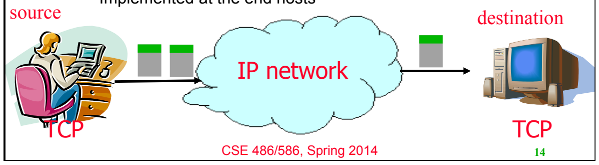
- Helps resisting the tendency to put and hide complicated things in the lower layers
- If a functionality must be implemented end-to-end, then don’t implement it in the network.
  - Exception: when there are clear performance improvements
- Laid out in “*End-to-End Arguments in System Design*” by J.H. 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

- Office hours
  - Tu: 9am - 12pm, W: 10am - 1pm, Th: 1pm - 4pm, F: 12pm - 3pm
- PA 1
  - Please try it out right away and see how far you can get.
- Please use Piazza; all announcements will go there.
- Please come to my office during the office hours!
  - Give feedback about the class, ask questions, etc.

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



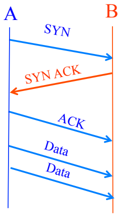
OK; Let's Think about It Together…

- Is this always a good thing?
- Is today's Internet still stateless?

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

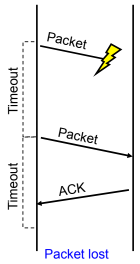
A (Very) Brief Overview of TCP



- 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

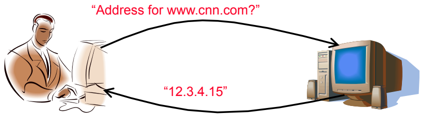
- There's overhead associated.
  - Connection establishment: 3-way handshake
  - Packet loss: retransmission timeout
  - Congestion control: doesn't utilize full bandwidth
- More importantly, some applications do not need these.
- Examples?
- So, enter UDP (User Datagram Protocol): exposes almost exactly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

Why Would Anyone Use UDP?

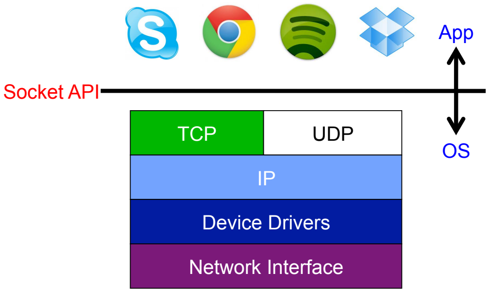
- Fine control over what data is sent and when
  - As soon as an application process writes
  - … UDP will package the data and send the packet
- No delay for connection establishment
  - UDP just blasts away without any formal preliminaries
  - … which avoids introducing any unnecessary delays
- No 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