CSE 461: Computer Networks

John Zahorjan—zahorjan@cs
Justin Chan—jucha@cs
Rajalakshmi Nandkumar—rajaln@cs
CJ Park—cjparkuw@cs
Course Staff

Justin Chan

Rajalakshmi Nandakumar

Chunjong Park
Grading

• Assignments/Projects/Homeworks: 55%
• Midterm: 15%
• Final: 30%
Reading Material:

- *Computer Networking: A Top-Down Approach*  
  Kurose, Ross  

- Other networking books would be fine as well

- There is a lot of information available online  
  (but it’s much harder to read a paragraph here and there than a book)
Administration

• Office hours
  • Opportunity to have more persona interactions with both me and the TAs.

• Course Resources
  • Mailing list: one-way communication
  • Dropbox: Homework
  • GoPost Forum: Back and forth discussions on class content
  • Gradebook: Grades will be posted here

• Slides
  • Customized, department-communal slides
Late Policy

• There is a policy on the course web
• We understand there can be unusual circumstances...
Sections

• Start tomorrow
CSE 461: Computer Networks
Our Goals

- We’ll spend most of our time studying our the Internet is built
- The Internet consists of hardware and software
  - NIC, switches, routers, hosts, WiFi, Ethernet, ...
  - DNS, TCP, IP, BGP, etc.
- The Internet is an implementation of a (many) distributed algorithm(s)
  - So are distributed applications
Our Goals (cont.)

• The Internet must confront a number of problems inherent in distributed systems
  • The major complication is that each agent can observe only its own state
  • It must infer the state of other agents based on what it knows about how they act
    • What protocol they run
  • The possibility of errors adds a significant level of difficulty
Today

• We start with a sweeping overview of the Internet

• To keep it at an appropriate level, I simplify most everything
  • The actual Internet is more flexible/general than what I show, but...
  • The key ideas shown here are the key ideas

• The survey is “bottom up”
• The course material is “top down”
Basic Concepts and Terminology - Link

0110100

Destination
*demodulates*

Link
*carries modulated signal*

Source
*modulates*

011010011101000
Basic Concepts and Terminology - Performance

Bandwidth (bits per sec)

Latency (milliseconds)
Basic Concepts and Terminology - Framing

- Frames provide boundaries so the receiver can know when the source has something to say and when not.
- Frames boundaries are useful when there are errors. A frame is corrupt, but not the entire data stream.
The *frame header* contains information the sender wants to transmit to the receiver that is not part of the data stream.

In this example, we’re transmitting sequence numbers:

- Why?

Headers are communication between the sending and receiving protocol implementations.

Data is communication between sending and receiving protocol clients (apps).
• A local area network (often just called “a network”) is an ensemble of nodes who can hear each other’s transmissions
  • When red transmits, orange, yellow and green all hear the bits
• Now need more information in the header
  • Source address (name)
  • Destination address
  • (This description corresponds to MAC addresses)
• Note that the addresses are just unique names
  • I know my name is 183449338302233928288.
  • When I see a frame addressed to that name, I act on it.
  • If I see a frame addressed to any other name, I ignore it.
• This works because every transmission is sent to every destination
Routers sit on two or more networks
Each LAN can achieve host to host delivery within the LAN as always
The router notices when a “packet” sent in the solid network is destined for the hash network
  - It copies the packet onto the other network
  - It doesn’t copy packets that don’t traverse networks
Basic Concepts and Terminology - Internet
Basic Concepts and Terminology - Routing

source

destination
Basic Concepts and Terminology - IP Address

- MAC addresses are just UIDs
  - No useful structure
- To route efficiently, we need addresses that have some locality
  - That’s what IP addresses are for
- IP addresses name network interface cards (roughly, hosts)
- The IP address space is global (mostly)
- Addresses that are “similar to” each other are located in the same LAN
  - The lower left LAN has a “gateway”
An Internet packet contains a destination IP address.

How can the Internet support the source application naming the destination application?

- "Application" is defined by the OS, and there are many
- Answer: The Internet is agnostic. It carries an uninterpreted ID (an integer), called a port
- The scope of the port name is the host (IP address)
Basic Concepts and Terminology- Berkeley Sockets

- Application creates a socket, which is an OS managed resource
  - Nothing to do with the Internet’s
- Application binds the socket to a port (a small integer in a restricted range)
- Incoming Internet packets give both the IP address of this node and a port number as the destination
- The OS looks for a socket bound to the port
  - If there is one, it puts the packet into the receive buffer of that socket
- When the application does a read from the socket, it fetches packets from the socket’s input buffer
A transport protocol carries uninterpreted bytes from a source application to a destination application.

Internet transport protocols are “end-to-end” – their implementations are in the ends hosts, not the hardware of Internet itself.
Basic Concepts and Terminology - UDP

- UDP is connectionless
- UDP is lossy
- UDP is packet-based
  - Provides app message framing
  - So long as msg isn’t too big

App Data  UDP header

UDP packet
Basic Concepts and Terminology - TCP

- TCP is connection based
- TCP is reliable
- TCP is stream-based
  - Reading from a stream is similar to reading from a file
### Internet Reference Model - Layering

- The classic OSI model has seven layers
- In practice, there are more like four

<table>
<thead>
<tr>
<th>Layer</th>
<th>Protocols</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>HTTP, SMTP, POP (project 0)</td>
</tr>
<tr>
<td>Transport</td>
<td>UDP, TCP</td>
</tr>
<tr>
<td>Network</td>
<td>IP</td>
</tr>
<tr>
<td>Link</td>
<td>Ethernet, WiFi</td>
</tr>
</tbody>
</table>
Basic Concepts and Terminology - Protocol

• A *protocol* is a set of rules governing how information is exchanged
  • It includes how information is encoded
  • It includes the definition of valid message exchange sequences

• The other end of a communication is presumed to be following the protocol
  • That allows each node to infer some information about the state of the other party/parties
Unrealistically Simple Example Protocol - USEP

• This protocol moves data from A to B, unreliably

• Sender:
  • Sends successive messages containing successive data
  • Each message contains a header
  • The header contains a *sequence number* – 0, 1, 2, ....

• Receiver:
  • Initializes a *next expected sequence number* variable to 0
  • When message arrives compares its seqno to next expected
    • seqno < next expected: ignore message
    • seqno == next expected: accept message; increment next expected
    • seqno > next expected: detect message loss(es); accept message; next expected = seqno + 1
USEP Questions

• Is the situation seqno < next expected possible?

• Is it possible to see the same seqno more than once?

• Why doesn’t the receiver just allocate a huge buffer and fill it with message contents as they arrive?
  • That is, allow messages to arrive in order 0, 3, 7, 4, 2, 1, 6, 5, for instance

• How does the receiver know when it has all the data?

• How does the receiver know when there’s a new sender wanting to start a new transfer?

• What does sender end up knowing about what data actually arrived?
Example Application Protocol – POP

• Client-server protocol
  • When client wants to start, it contacts the server

• Objective: simple, unreliable, data transfer

• Message format:

<table>
<thead>
<tr>
<th>Field</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>magic</td>
<td>16 bits</td>
</tr>
<tr>
<td>version</td>
<td>8 bits</td>
</tr>
<tr>
<td>command</td>
<td>8 bits</td>
</tr>
<tr>
<td>seqno</td>
<td>32 bits</td>
</tr>
<tr>
<td>session id</td>
<td>32 bits</td>
</tr>
<tr>
<td>data payload</td>
<td>variable</td>
</tr>
</tbody>
</table>

• Commands are: HELLO, DATA, ALIVE, and GOODBYE
• HELLO exchange sets up a new “session”
  • Client picks a UID for the session
    • How?

• Client sends successive data payloads
  • ALIVE responses reassure client

• GOODBYE exchanges allows clean shutdown’

• What happens if a message is lost?
Lost Message - Timeouts

• It is possible for the sender to know that a message was received
  • The sender receives a message that would have been sent only if its message was received (assuming the other end is following the protocol)
    • Example: If the client sends an HELLO, it knows it was received when it gets back an HELLO

• It isn’t possible to know that a message wasn’t received
  • Why can’t receiver send a message saying “I didn’t get it”?

• A common approach to guessing when a message is lost is a timeout
  • Send, wait for reply, if it doesn’t come “after a while,” act as though it was lost
POP State Diagrams

Server
(per-session state)

Client

GOODBYE in any state transitions to state CLOSED