COMP 204 – Principles of Computer Networks

Week 2
© 2011 Alex Elbert, Todd Whittaker

Agenda

• Review this week’s learning outcomes
• Presentation of this week’s material
• Introduce homework problems
• Q & A session
This Week’s Outcomes

- Explain the functions of common TCP/IP application layer programs.
- Explain how protocols ensure a common language among communications endpoints.
- Compare and contrast TCP and UDP transport layer protocols.
- Simulate the key TCP functions of initiation, termination, acknowledgement, and re-transmission.

Review – OSI model

<table>
<thead>
<tr>
<th>Layer</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>All People Seem To Need Data Processing (they really do – for exam purposes)</td>
</tr>
<tr>
<td>Presentation</td>
<td>Each layer provides a different level of abstraction</td>
</tr>
<tr>
<td>Session</td>
<td>Each layer has a well-defined function</td>
</tr>
<tr>
<td>Transport</td>
<td>Layer boundaries are chosen to minimize the information flow between layer boundaries</td>
</tr>
<tr>
<td>Network</td>
<td>The number of layers is kept small enough to be feasible</td>
</tr>
<tr>
<td>Data Link</td>
<td></td>
</tr>
<tr>
<td>Physical</td>
<td></td>
</tr>
</tbody>
</table>
Review - OSI versus TCP/IP

- Application
- Presentation
- Session
- Transport
- Network
- Data Link
- Physical

- Application
- Transport
- Internet
- Network Access

Chapter 3

Chapter 4

FRANKLIN UNIVERSITY
Chapter 3

• Application layer (7 on OSI model)

Application Layer

• Applications vs. Services vs. Protocols
  – User applications – programs that someone runs.
  – Services – processes that the operating system runs to provide services to applications or other parts of the OS.
  – Application protocols – the language defined by an application for exchanging data with other applications or services.
Application Layer

• Applications vs. Services vs. Protocols
  – User applications – programs that someone runs.
  – Services – processes that the operating system runs to provide services to applications or other parts of the OS.
  – Application protocols – the language defined by an application for exchanging data with other applications or services.
Application Layer

- Applications vs. Services vs. Protocols
  - User applications – programs that someone runs.
  - Services – processes that the operating system runs to provide services to applications or other parts of the OS.
  - Application protocols – the language defined by an application for exchanging data with other applications or services.

Application Layer Protocols

- Rules for exchanging data between applications.
- How is data structured in the messages?
- What kind of messages (errors, requests, responses, etc.)?
Applications and Services

It’s still “client-server” even if the application and the service run on the same machine.
Applications and Services

![Diagram showing peer-to-peer networking with applications and services running on both peers.]

- Peer 1: Runs the application and the service
- Peer 2: Runs the application and the service

Often employs a “centralized” service for naming or tracking purposes.
Book examples

- DNS (domain name system) resolution
- Web servers and HTTP
- E-mail servers and SMTP (simple mail transfer protocol), POP (post office protocol), IMAP (internet message access protocol)
- DHCP (dynamic host configuration protocol) clients and servers

DNS – Background

- Original Internet had flat namespace
- INTERNIC approved names which has the following shortcomings:
  - Central authority
    - Big workload
    - Cannot delegate
  - Does not accommodate large sets
    - Name conflict
    - Difficult to maintain translation file
DNS – Design Goals

- Decentralized naming mechanism
- Ability to delegate the authority
- Distributing translation responsibility
- Similar to organizational structure

DNS – Features

- Distributed system
- Efficient – most names resolve locally
- Reliable – no single point of failure
- Servers are called name servers
- Clients are called name resolver
DNS – Features

- Conceptually each level knows next level down
- Each server has naming authority
- Each subdomain would need its own server
- In reality multiple levels are served by single server

Mapping Domain Names

```
        Root
       /   \
    edu   com  gov
    /\     /\    /
   osu  franklin
     /  \
    cis
```
DNS – Hierarchical Design

- Hierarchical naming scheme
- Partitioned at the top
- Authority for names in subdivisions is delegated
- Similar to phone numbers
  - Area code, exchange, subscriber number
- Names in partition do not conflict

- Hierarchical naming does not imply physical location
- DNS allows computers to be named by logical association, not physical
- Each sub-tree of the name space is called domain
- Part that the name server is responsible for is called a zone
- If the name server does not delegate the responsibility for managing its domain, then domain is the same as the zone
- Mapping information is stored in a zone file
DNS – Name Servers

- **Primary server** stores, maintains and updates the zone file on its own
- **Secondary server** downloads the zone file from the **primary server** during the process called **zone transfer**
- Both servers are **authoritative** for that zone
- Done for redundancy purposes

DNS – Basics

- Two major concepts
  - Name syntax and rules for delegation
  - Implementation of distributed mapping system
- Each level is a label
- Labels are separated by dots
DNS – Example

• Highest level is last
  cs.franklin.edu

• Three labels
  • cs
  • franklin
  • edu

• Three domains
  • cs.franklin.edu
  • franklin.edu
  • edu
DNS – Syntax

- Standard only specifies syntax
- Any organization could choose own values for labels
- Usually follow official Internet domain system
- Organizational or Geographical

DNS – Top Level Domains

- Top level organizational domains
  - COM – commercial organizations
  - EDU – educational institutions
  - GOV – federal government institutions
  - MIL – United States military
  - NET – major network support centers
  - ORG – other organizations
  - INT – international organizations
DNS – Top Level Domains

- Geographical domains
- Two letter country codes (ISO 3166)
- 239 official country codes (as of 2/26/01)
  - US – United States
  - CA – Canada
  - DE – Germany
  - JP – Japan
  - LI – Liechtenstein
  - TW – Taiwan, Province of China

DNS – US Domain Names

- US Domain subdivided
  - FED – Federal government agencies
  - DNI – Distributed national institutes
  - 50 state codes
    - CI.<city>.<state>.US
    - CO.<county>.<state>.US
    - <school>k12.<state>.US
    - <school>.cc.<state>.US
DNS – Examples

- City of Columbus
  - WWW.CI.Columbus.OH.US
- Dublin City Schools
  - WWW.Dublin.K12.OH.US
- Franklin County
  - WWW.CO.Franklin.OH.US
- Delaware County
  - WWW.CO.Delaware.OH.US

DNS – Protocol

- A given name may map to more than one item in the domain system.
- Must specify what type of resolution is desired
- May need IP address or mail exchanger
DNS – Query Types

• Queries and Resource records contain type

<table>
<thead>
<tr>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Host address to IP</td>
</tr>
<tr>
<td>MX</td>
<td>Mail Exchanger</td>
</tr>
<tr>
<td>NS</td>
<td>Name of authoritative server for domain</td>
</tr>
<tr>
<td>CNAME</td>
<td>An alias in case multiple services are running on a single server</td>
</tr>
</tbody>
</table>

DNS – Resolution

• Two methods of resolving names
  – Recursive – Asking name server for complete resolution. The client asks the name server for resolution, if the server has the necessary information its returned. Otherwise, the server queries other servers to find out the answer.
  – Iterative – Contacting names servers one at a time. The client asks the name server for resolution, if the server has the necessary information it is returned. Otherwise, the name of the server that might have the information is returned, so the client can re-query it.
DNS – Resolution

• Must be efficient
• What if always start at root
  – Most translations are local
  – Root would be overloaded
  – Root failure would cause system failure
  – Generate lots of network traffic

DNS – Caching

• Name servers use name caching
  – Contains recently used names and where mapping was obtained
• Server checks cache
  – Sends non-authoritative response
  – Includes server and its IP address
• Client decides whether to use or revalidate
DNS – Caching

• Each cache entry has a time limit
• Entry is deleted when expires and mapping is requested again
• Time limit is set by original server
• Long time limits can be sent for static binding, short for dynamic bindings

DNS – Caching

• Some OS download local name server database (including time limits)
• Name resolution requires no traffic
• Must check with name server for updates, expired entries
DNS – Utilities

• nslookup
  – Used to lookup a translation from name to address
• named
  – Unix name server

Chapter 4

• Transport layer (4 on OSI model)
Transport layer

- Tracks communication between apps on sender and receiver.
- Segments data for multiplexing and flow control (sender).
- Reassembles segments into streams (receiver).
- Error recovery, open and close sessions.

Conversations

- One computer can simultaneously be sending/receiving data on many applications.
- How is this done?
  - Ports
  - Segmentation
## Ports

- Each application is assigned a *port* for communication. Just a number.

<table>
<thead>
<tr>
<th>Port</th>
<th>Keyword</th>
<th>Unix Keyword</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Echo</td>
<td>echo</td>
<td>Echo</td>
</tr>
<tr>
<td>20</td>
<td>FTP-DATA</td>
<td>ftp-data</td>
<td>File Transfer Protocol (data)</td>
</tr>
<tr>
<td>21</td>
<td>FTP</td>
<td>ftp</td>
<td>File Transfer Protocol</td>
</tr>
<tr>
<td>43</td>
<td>NICNAME</td>
<td>whois</td>
<td>Who is</td>
</tr>
<tr>
<td>53</td>
<td>DOMAIN</td>
<td>nameserver</td>
<td>Domain Name Server</td>
</tr>
<tr>
<td>80</td>
<td>HTTP</td>
<td>http</td>
<td>Web servers</td>
</tr>
</tbody>
</table>

This program... ...is tied to this port.
Ports

- “Well known” ports are 0 – 1023
- “Registered” ports are 1024 – 49151
- “Dynamic” ports are 49152 – 65535
- Where the heck do these weird numbers come from?

Ports

- Binary (base 2 number system).
  - Fixed width field consisting of all 1’s and 0’s.

<table>
<thead>
<tr>
<th>Bin</th>
<th>Dec</th>
<th>Hex</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0001</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0010</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>0011</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>0100</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>0101</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>0110</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>0111</td>
<td>7</td>
<td>7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bin</th>
<th>Dec</th>
<th>Hex</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>1001</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>1010</td>
<td>10</td>
<td>A</td>
</tr>
<tr>
<td>1011</td>
<td>11</td>
<td>B</td>
</tr>
<tr>
<td>1100</td>
<td>12</td>
<td>C</td>
</tr>
<tr>
<td>1101</td>
<td>13</td>
<td>D</td>
</tr>
<tr>
<td>1110</td>
<td>14</td>
<td>E</td>
</tr>
<tr>
<td>1111</td>
<td>15</td>
<td>F</td>
</tr>
</tbody>
</table>
Ports

• Binary (base 2 number system).
  – What is this $10101011_2$ in decimal? Hex?
    • $1 \times 2^7 + 0 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 171_{10} = AB_{16}$.

• “Well known” ports are 0 – 1023
• “Registered” ports are 1024 – 49151
• “Dynamic” ports are 49152 – 65535
• Where the heck do these weird numbers come from?
Socket

• Combination of an internet name or number and a port is called a *socket*.
  – Ex: www.franklin.edu:80
  – Ex: cs.franklin.edu:22
  – Ex: 127.0.0.1:7
Segmentation

• How do we share the network, especially when sending long messages?
  – Long messages tie up the network for a very long time – and other problems...

  Application 1's message

  Application 2's message

  Application 3's message

Segmentation

• How do we share the network, especially when sending long messages?
  – Multiplex the network by breaking up long messages into multiple *segments*.
Segmentation

• How do we share the network, especially when sending long messages?
  – Interleave the sending of the segments

Segmentation

• How do we share the network, especially when sending long messages?
  – What has to happen on the other end?
Flow and error control

- Flow control determines the speed of transmission such that the network isn’t flooded.
- Error control is resending segments that are lost or corrupted in transmission.

UDP

- Remember this diagram?

![Layer Diagram]
UDP

• Remember this diagram?

What is a “header”?

UDP

• UDP header

- Source port (16)
- Destination port (16)
- Length (16)
- Checksum (16)

8 bytes total

32 bits wide

• \(2^{16}\) is 65536, therefore max UDP segment is 64K (where “1K” is 1024)
UDP

- UDP header

<table>
<thead>
<tr>
<th>Source port (16)</th>
<th>Destination port (16)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length (16)</td>
<td>Checksum (16)</td>
</tr>
</tbody>
</table>

32 bits wide

8 bytes total

- If a port identifies a program on a machine, how does the segment get to the machine?

That’s a problem for a different layer! We’re at layer 4 (transport) with UDP, but machines are at layer 3. So, next week!

- If a port identifies a program on a machine, how does the segment get to the machine?
UDP

• UDP
  – Considered to be “unreliable” delivery.
    • Packets are just sent out on the network.
    • No acknowledgement of receipt, retransmissions.
    • Segments can arrive out of order.
  – Very low overhead
    • Only 8 bytes in the header.
    • No acknowledgement or retransmissions.

Very much like the standard US postal service!
What does all this extra stuff give us?
TCP

- Connection-oriented protocol
  - Reliable: lost or mangled packets are retransmitted.
  - Sequenced: messages can be reassembled into the same stream in the same order as was sent.
  - Flow and congestion control: window size can be changed to allow more or less data at a time.

TCP Flags

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source port (16)</td>
<td>Destination port (16)</td>
</tr>
<tr>
<td>Sequence Number (32)</td>
<td>Acknowledgement Number (32)</td>
</tr>
<tr>
<td>Header length (4)</td>
<td>Reserved (6)</td>
</tr>
<tr>
<td>TCP flags (6)</td>
<td>Window (16)</td>
</tr>
<tr>
<td>Checksum (16)</td>
<td>Urgent (16)</td>
</tr>
<tr>
<td>Options (0 or 32)</td>
<td></td>
</tr>
</tbody>
</table>

**URG**: urgent field significant

**ACK**: Acknowledgement field significant

**SYN**: Synchronize sequence numbers

**RST**: Reset the connection

**FIN**: No more data from sender

**PSH**: Push function
TCP

- Establishing a conversation
  - 3-way handshake required
    - Initiator opens the connection with SYN
    - Responder ACKs and sends own SYN
    - Initiator ACKs
TCP

• Nicely ending a conversation
  – 3-way handshake required
    • Initiator sends FIN
    • Responder sends ACK
    • Responder sends FIN
    • Initiator sends ACK

• Quickly ending a conversation
  – Send RST.
    • This is what Comcast did.
    • Net neutrality issue.
TCP

• Windowing

This gets very slow as we acknowledge every segment one at a time.

TCP

• Windowing

Getting better...
TCP

• Windowing

Now we’re talking!

TCP

• Retransmission
  – When a segment is sent, the sender will also put it in a retransmit queue.
  – When an ACK is received, the sender will remove all ACKed segments from the queue.
  – If an ACK is not received after a set amount of time, or if the ACK numbers don’t match, the sender will retransmit all segments in the queue.
TCP

• Windowing
  – The amount of data a host can receive before an ACK is the window size.
  – Window sizes can be adjusted on the fly in case packets get lost (smaller) or if the receiver can handle the data faster (larger)

This Week’s Outcomes

• Explain the functions of common TCP/IP application layer programs.
• Explain how protocols ensure a common language among communications endpoints.
• Compare and contrast TCP and UDP transport layer protocols.
• Simulate the key TCP functions of initiation, termination, acknowledgement, and re-transmission.
Self Quiz

• Compare and contrast client/server and peer-to-peer architectures.
• What is the differences between a user application, a service, and an application protocol?
• TCP and UDP are at what layer in OSI?
• How is a network shared effectively?

Self Quiz

• How does TCP achieve connection oriented communications?
• Why is UDP considered “unreliable”? What makes it so?
• How are TCP sessions initiated? Closed?
• How does a sender know to retransmit in TCP? In UDP?
Due this week

- Proctor forms
- Homework 1
- Lab 1
- Participation 2

Due next week

- Homework 2
- Participation 3
Next week

• Chapter 5
• OSI Layer 3
• Internet Protocol (IP)
• Addressing and routing

Q & A

• Questions, comments, concerns?