Internet/Networking Overview

Notes for CSC 580

Internet Overview

- Basic idea: Network of networks
- Internet (protocol) vs. “The Internet”

Packet Switching

- Each packet sent independently
  - Different pieces can be routed separately
  - Not dependent on a fixed “switched” connection, so can “re-route” easily to avoid trouble spots
- Postcard analogy

Packet size determined by MTU (Maximum Transmission Unit)

Some Internet History

- ARPA (Advanced Research Projects Agency) experiment to test ideas of “packet switched networks”
- 1969: First node goes on-line (UCLA)
- 1970’s: Maturing and apps (e-mail in 1972)
- 1980’s: Widespread in academic, military, and research communities
  - 1985: NSFNET
- 1990’s: The web and privatization

Some Web History

- 1990: Tim Berners-Lee at CERN defined:
  - HTTP (transfer)
  - HTML (presentation)
  - URLs (reference)
- 1993: Mosaic released by NCSA
- December 1994: Netscape appears
  - Improvements in efficiency/caching
  - Integrated encryption/SSL to enable secure connections
  - Portable with attractive / easy-to-use user interface

Network Protocols

- A network protocol provides syntactic and semantic rules for communication.
  - Often defined in terms of state machines
  - Standards allow service-based interoperability
    - Internet RFCs (TCP/IP, DNS, …)
    - IEEE standards (Ethernet, etc.)
- Protocols can be in hardware or software
  - Ethernet access protocol often in hardware
  - HTTP and other high-level usually in software
**Network Layers**

- **Layered Model:**
  - Each layer uses only the layer directly below it
  - Benefit: Different issues to address at different levels of abstraction

<table>
<thead>
<tr>
<th>Layer</th>
<th>OSI Model</th>
<th>IP (or TCP/IP or Internat) Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Application</td>
<td>Examples: HTTP, FTP, SMTP (E-mail)...</td>
</tr>
<tr>
<td>6</td>
<td>Presentation</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Session</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Transport</td>
<td>Examples: TCP, UDP</td>
</tr>
<tr>
<td>3</td>
<td>Network</td>
<td>IP (Internet Protocol)</td>
</tr>
<tr>
<td>2</td>
<td>Data Link</td>
<td>Transmission media (ethernet, token ring, …)</td>
</tr>
<tr>
<td>1</td>
<td>Physical</td>
<td></td>
</tr>
</tbody>
</table>

Next: We look at issues and vulnerabilities with each layer.

**Link Layer**

- For directly-connected systems to communicate
- **Example 1: Ethernet**
  - Ethernet cards IDed by “MAC address” (48 bits)
  - E.g., sending from 00:02:d1:61:9c:f1 to 00:d0:15:38:2c:d0
  - Packet:

```
00 02 d1 61 9c f1 08 00 00 d0 15 38 2c d0 20 00 57 00 ... Data to be transmitted ...
```

- **Example 2: PPP (Point-to-Point Protocol)** link layer uses HDLC (High-level Data Link Control)
  - Only two endpoints, so no addressing is necessary!
  - Includes error detection for flaky links

**Link Layer Issues**

- **Topology**
  - Bus
  - Ring
  - Hub/Concentrator
  - Crossbar/switch
  - Star
  - Tree

**Link Layer Threats and Vulnerabilities**

- **Message Interception**
  - All systems on a LAN “see” all traffic
  - Usually ignore all but to them (based on MAC addr)
  - However: Interfaces can be put into “promiscuous mode”
  - Ethernet evolution 1: Hubs
    - Star topology, but all traffic still to all hosts
  - Ethernet evolution 2: Switches – star/tree topology
    - Switch remembers which MAC addresses are connected to which ports, and sends traffic only to addressed host

**Network Layer Issues**

- **IP Packets**

```
<table>
<thead>
<tr>
<th>Version</th>
<th>IHL</th>
<th>Total Length</th>
<th>Identification</th>
<th>Label</th>
<th>Flags</th>
<th>Fragment Offset</th>
<th>Protocol</th>
<th>Header Checksum</th>
<th>IP Source Address</th>
<th>IP Destination Address</th>
<th>Options (usually empty)</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

- Everything is just bits being transmitted
  - Can be captured/trapped
  - Header checksum is not cryptographic – for detecting transmission errors, not tampering

**IP Addresses**

- Network (IP) layer concerned with addressing and routing
  - Address usually obtained from DNS – an application-layer protocol!
  - Current version of IP protocol: IPv4
    - Addresses are 32-bit values
    - IP addresses are for interfaces, not computers
    - Given as 4 bytes in “dotted notation” (e.g., 129.120.61.48)
    - Addresses divided into network and host parts
      - Class C example: 129.120.61 is net addr, 48 is host addr
      - Hosts w/same network addr can talk directly to each other (LAN)
    - Net address notations:
      - Subnet mask: 129.120.61.255
      - Net addr notation:
        - Subnet mask: 129.120.61.255
        - Net addr notation:
          - Subnet mask: 129.120.61.0
          - Net addr notation:
            - Subnet mask: 129.120.61.0
  - See with “ifconfig” in Unix/Linux ; “ipconfig” in Windows
  - Next generation IP: IPv6
    - Addresses are 128-bit values – huge address space
ARP: Finding the right host on a subnet

• **Problem:**
  - Ethernet works on MAC addresses (doesn’t understand IP)
  - IP works on IP addresses (doesn’t understand Ethernet)
  - How do we get a packet to the right host on a LAN/subnet?

• **Answer:** The Address Resolution Protocol (ARP)
  - Example: Host 10.1.1.42 wants to send to 10.1.1.92
    - But! Only knows IP address, not MAC address
    - So: Broadcasts an ARP message on Ethernet saying “Who has 10.1.1.92?”
    - 10.1.1.92 responds with “I have 10.1.1.92. My MAC is 00:02:2d:9a:27:72”
    - Now 10.1.1.42 sends over Ethernet to this MAC

When ARP goes bad: ARP Spoofing

• **Performance:** Hosts keep an “ARP Table” of known IP address <-> MAC mappings
  - Doesn’t have to ask if MAC address known
  - Updates table with each “I have a.b.c.d” message
  - Expires mappings regularly (in case IP moves)

• **ARP spoofing:** To sniff on a switched Ethernet
  - Attacker (on same LAN) sends out “I have a.b.c.d” messages for target
    - Packets then sent to the attacker rather than the destination (which could be the gateway router)
  - Attacker can then forward packets so no disruption – just monitoring

ARP Spoofing Countermeasures

• **Static ARP tables**
  - Sensitive subnets should use static ARP tables
  - Mappings don’t expire
  - Mappings are hard-coded to be genuine by the administrator
  - Not perfect: MAC address spoofing still possible!

• **Possible future directions:**
  - A better solution is still an unresolved research issue
  - Some suggest authenticated ARP
    - Uses digital signatures (PK Crypto), so slow – and ARP needs to be very low overhead!

Subnet-to-subnet Communication

Gateways and Routers

• Router “lives on” multiple subnets
  - Local address on each
  - Can be more than 2 NICs / subnets

• Routing tables say what goes where
  - See with “/sbin/route” in Linux/Unix
  - See with “route print” in Windows

Sample simplified host routing table:

<table>
<thead>
<tr>
<th>Destination</th>
<th>Gateway</th>
<th>Genmask</th>
<th>Iface</th>
</tr>
</thead>
<tbody>
<tr>
<td>129.12.60.0</td>
<td>0.0.0.0</td>
<td>255.255.254.0</td>
<td>eth0</td>
</tr>
<tr>
<td>0.0.0.0</td>
<td>129.12.61.1</td>
<td>0.0.0.0</td>
<td>eth0</td>
</tr>
</tbody>
</table>

Sample simplified router routing table:

<table>
<thead>
<tr>
<th>Destination</th>
<th>Gateway</th>
<th>Genmask</th>
<th>Place</th>
</tr>
</thead>
<tbody>
<tr>
<td>129.12.60.0</td>
<td>129.12.61.1</td>
<td>255.255.254.0</td>
<td>k190</td>
</tr>
<tr>
<td>72.1.19.0</td>
<td>72.1.19.2</td>
<td>0.0.0.0</td>
<td>k190</td>
</tr>
</tbody>
</table>

Network Layer Topology

• How are subnets connected together?
  - Earlier discussion was physical link topology – now logical links

• Physical layer considerations:
  - Point-to-point: Direct connections of two endpoints
    - Protocols: PPP (point-to-point protocol – typically over serial/phone lines) and PPoE (point-to-point over Ethernet – used by a lot of DSL)
    - Broadcast: Sent out to “whoever gets it” (e.g., wireless)
    - Similar issue on ethernet: switches vs. hubs

• Interconnection issues:
  - Ownership: Who owns pieces of the network?
  - Control: Sub-network administration
  - Boundary: Separation of separate domains of control

Topology Example

- Corporate Network
- DMZ
- Internet
- Controlled Access Point

But: What about public servers?
- Must make sure “controlled access” is only way in! (Modems, wireless, …)
- Firewall/border security: “A crunchy shell around a soft, chewy center” (Cheswick)
About DMZs

- **Important**: What is called a “DMZ” on many home routers is not what a network professional means by a “DMZ”!!!

- A real DMZ:
  - Hosts isolated in a separate subnet so traffic does not enter the internal network, even if public hosts broken in to.
  - Can provide gateways or “bastion hosts” that are connected to both internal and external networks (safe stepping stones).

- A “home router DMZ”:
  - All traffic (all ports) is routed to one particular internal-network system – makes an internal host “public” for receiving connections
  - Actually lets traffic into the internal network, so if someone breaks into the “DMZ host” they have full access to your network!

Faking Network Layer Topology

- Overlay networks
  - Idea: Use connections over a network on top of a network
  - P2P can be viewed as overlay networks
  - Anonymity networks (like Tor) are overlay networks
  - PlanetLab overlay for networking research

- PPP: Point-to-Point Protocol
  - Standard used for dial-up connections
  - One host on each side of a link
  - Originally for sending network packets over serial connections

Network Layer Attacks

Field Tampering

- **Attack type 1**: Put invalid data in fields
  - Example 1: Ping of Death
    - “Too large” ping packet crashes machine
  - Example 2: LAND Attack
    - Specially crafted packet with both source and destination set to victim address, with fields that make machine lock up
  - Example 3: Jolt Attack (and Teardrop)
    - Invalid fragmentation of packets that destination can’t reassemble, so machine freezes waiting for more

IP Spoofing

- **Smurf Attack** (Simplified)
  - Attacker
  - Intermediaries
  - Victim
  - Fake ping packet with src 209.12.17.35 and dest 123.45.67.89
  - Many (up to 254) ping responses to Victim
  - Works particularly well when Attacker-Intermediaries connection is lower bandwidth than Intermediaries-Victim

- **IP Spoofing Countermeasures**
  - Filter out broadcast messages at gateway
    - Doesn’t work if intermediary inside border!
  - In general: Filter out LAN-only messages across border
  - Egress filtering
    - Only let out packets with appropriate source addr
    - Doesn’t stop you from being an intermediary or a victim – think of it as “being a good netizen”
Fragmentation issues

- Fragmentation: Breaking up long IP packets to fit in a particular type of low-level link.
  - Example: Slow PPP might use maximum packet length of ≈ 500 bytes for responsiveness vs. typical Ethernet length 1468 bytes.

- Security issues:
  - Using fragmentation to avoid an Intrusion Detection System
  - Break up a “signature” into multiple fragments
  - How are overlapping packets re-assembled?

<table>
<thead>
<tr>
<th>Fragment 1:</th>
<th>Fragment 2:</th>
<th>Reassembly:</th>
</tr>
</thead>
<tbody>
<tr>
<td>default</td>
<td>default</td>
<td>default.id (No signature match!)</td>
</tr>
<tr>
<td>default.id</td>
<td>default.id</td>
<td></td>
</tr>
</tbody>
</table>

Fragmentation issues – Cont’d

- Solutions?
  - Try every possible packet reassembly
  - Problem: $n$ fragments gives $2^n$ reassemblies
  - Know how major OSes work and try those assemblies

- Problem: What if a new machine or new network stack?
  - Reassemble packets at firewall
  - Only a consistent reassembled packet stream seen inside
  - Problems:
    - Difficult to keep up with a very high bandwidth connection at the gateway
    - Doesn’t protect from internal attacks

Transport Layer

- IP provides little beyond basic routing
  - Packets may be lost
  - Packets may arrive out of order
  - Errors may occur in packets
  - One address per machine — no way of distinguishing different users/services

- Transport layer
  - UDP: User Datagram Protocol
    - Adds “ports” to distinguish users/services
  - TCP: Transmission Control Protocol
    - In addition to ports adds: error detection w/packet retransmission, packet re-ordering, and sessions (simulated connections)

Transport Layer Issues

UDP Packets

- UDP adds connection distinguishers (ports) to IP:

<table>
<thead>
<tr>
<th>Source Port</th>
<th>Destination Port</th>
<th>Length</th>
<th>Checksum</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>160</td>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>

Some common UDP protocols:
- Domain Name Service (DNS) – port 53
- Network Time Protocol (NTP) – port 123
- “Discoverable” services (IPP, Rendezvous, …)
- Streaming/multicast transmissions

TCP Packets

- TCP adds “sessions” or “connections” to the bare IP protocol:

<table>
<thead>
<tr>
<th>Source Port</th>
<th>Destination Port</th>
<th>Sequence Number</th>
<th>Acknowledgment Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>15</td>
<td>16</td>
<td>21</td>
</tr>
</tbody>
</table>

Data Offset | Flags | Window | Checksum | Urgent Pointer | Options | Flags:
|------------|-------|--------|----------|----------------|---------|----------|

URG: Urgent ptr valid
RST: Reset flag
ACK: ACK valid
SYN: Synchronize seq #s
PSH: Push function
FIN: Finish of connection

The 3-way handshake

Labels below give (Flags, Seq#, Ack#, Flags):

Client

(SYN, C-Seq, 0)

(SYN|ACK, S-Seq, C-Seq+1)

Client

(ACK, C-Seq+1, S-Seq+1)

Server

- To establish connection, client must prove that it received the SYN|ACK packet!
- SYN|ACK packet routed to system with source address from first SYN packet
  - Since based on routing, only secure back to the subnet of the source

Transport Layer Issues

UDP Packets

- UDP adds connection distinguishers (ports) to IP:

<table>
<thead>
<tr>
<th>Source Port</th>
<th>Destination Port</th>
<th>Length</th>
<th>Checksum</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>160</td>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>

Some common UDP protocols:
- Domain Name Service (DNS) – port 53
- Network Time Protocol (NTP) – port 123
- “Discoverable” services (IPP, Rendezvous, …)
- Streaming/multicast transmissions

TCP Packets

- TCP adds “sessions” or “connections” to the bare IP protocol:

<table>
<thead>
<tr>
<th>Source Port</th>
<th>Destination Port</th>
<th>Sequence Number</th>
<th>Acknowledgment Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>15</td>
<td>16</td>
<td>21</td>
</tr>
</tbody>
</table>

Data Offset | Flags | Window | Checksum | Urgent Pointer | Options | Flags:
|------------|-------|--------|----------|----------------|---------|----------|

URG: Urgent ptr valid
RST: Reset flag
ACK: ACK valid
SYN: Synchronize seq #s
PSH: Push function
FIN: Finish of connection

The 3-way handshake

Labels below give (Flags, Seq#, Ack#, Flags):

Client

(SYN, C-Seq, 0)

(SYN|ACK, S-Seq, C-Seq+1)

Client

(ACK, C-Seq+1, S-Seq+1)

Server

- To establish connection, client must prove that it received the SYN|ACK packet!
- SYN|ACK packet routed to system with source address from first SYN packet
  - Since based on routing, only secure back to the subnet of the source
**SYN Issues – Predictability**

- Sequence numbers should be unpredictable
  - Most systems today select random values that meet some necessary conditions
- Otherwise:

  ![Diagram](null)

  Data pretending to come from fake Src

  Particularly dangerous when "fake Src" is a trusted IP address

**SYN Flooding – Solutions**

- SYN cookies
  - Basic idea: Use cryptography to avoid saving state
  - Specifically: Store info in Seq # to verify upon ACK
  - Time: Increments every 64 seconds
  - MSS = Maximum Segment Size (must be remembered!)
  - Cryptograph hash w/secret gives unpredictability
    - Only the server and the receiver of the seq# can reproduce seq#
  - Not perfect: Limited MSS options, 24-bits can be brute-forced, ...
- Router solutions (protect hosts without modifying hosts)
  - Rate limiting/shaping, Cisco router "TCP Intercept" feature, ...

**Transport Layer Protection**

SSL/TLS

- Originally designed to protect web browser to web server
  - Invented by Netscape
  - Generic TCP protection
  - Authentication: Supports server and client certificates
  - Confidentiality: Symmetric encryption after key establishment
  - Integrity: All packets protected with a MAC
- Later versions (SSL v2.1) referred to as TLS
  - TLS incorporated within application-layer protocols now in addition to in a sub-application layer
    - Example 1: IMAP (mail) can be either a separate SSL protected service/port (imap: port 993) or negotiated after plaintext startup in standard IMAP (port 143)
    - Example 2: LDAP with similar options (ldap is port 389, ldaps is port 636)

**Application Layer**

- Task-specific
  - Protocols for sending e-mail (SMTP), getting web pages (HTTP), secure shell (SSH), ...
  - Can do things that only knowledge of task can accomplish
- Security either provided in application-specific ways (e.g., PGP for e-mail) or by relying on lower-level protections (SSL/TLS, IPSec, …)
### Application Layer – Example (HTTP)

- Opening message has request and HTTP version
- Content is “media”, so MIME types make sense
- More on specific applications in later classes…

```plaintext
GET /index.html HTTP/1.0
Content-Length: 1423
Content-Type: text/html
Blah, blah, blah…
```

### Basic Network Security Tools

#### Firewalls
- Types of firewalls:
  - Stateless (“packet filters”)
    - Decisions made independently on a packet-by-packet basis
    - Good for blocking ports (“no incoming HTTP”) or blocking IP addresses/ranges (“blacklists”)
    - Simple and fast – included in many routers
  - Stateful
    - Keep information that relates packets to one another
    - Can track sessions and even related sessions (e.g., FTP control and data)
  - Application (or “proxies”)
    - Doesn’t forward packets at all – works at application layer
    - Best example: Web proxies
    - Allows content filtering as well as security

#### Intrusion Detection Systems (IDS)
- Categorization by location:
  - Host-based Intrusion Detection Systems (HIDS)
    - Many just watch system/audit logs for suspicious activity
    - Some with more sophisticated monitoring (pH: monitors system calls)
  - Network-based Intrusion Detection Systems (NIDS)
    - Watches all traffic at a certain point (can use a tap)
    - If just external access point, can miss insider attacks!
    - On switched networks: Use a “spanning port”
    - Difficulties with encrypted traffic

- Categorization by type:
  - Signature-based
    - Monitors traffic for known suspicious patterns
    - Advantages: Fast, few false positives
    - Drawbacks: Can’t detect novel attacks, must prioritize warnings
    - Keeping signatures up-to-date leads to subscription services
  - Anomaly-based
    - Tries to learn “typical activity” and flag anomalies
    - Anything unusual (including novel attacks) can be caught
    - Drawbacks: Slow and atypical behavior doesn’t necessarily mean bad behavior (too many false positives)
  - Snort and most commercial IDSs are signature-based
    - Sometimes with simple anomaly-based extensions