Project 3a is out!

Goal: implement a basic network firewall

- We give you the VM & framework.
- You implement the firewall logic.
Get started early
What Is Firewall?

Blocks malicious traffic

Blocks unauthorized traffic
1. Decode the packet
2. Check the firewall rules
3. Pass or drop the packet
Packets on wire look like this…

```
52 54 00 12 35 02 18 ba dd c0 ff ee 08 00 45 00
00 ac 06 5a 40 00 40 06 72 a0 0a 00 02 0f 5b bd
59 86 b0 90 00 50 ca af 9a 10 00 00 fa 02 50 18
39 08 d2 5b 00 00 47 45 54 20 2f 76 30 2f 73 6f
75 72 63 65 73 20 48 54 54 50 2f 31 2e 31 0d 0a
48 6f 73 74 3a 20 76 69 64 65 6f 73 65 61 72 63
68 2e 75 62 75 6e 74 75 2e 63 6f 6d 0d 0a 55 73
65 72 2d 41 67 65 6e 74 3a 20 55 6e 69 74 79 20
56 69 64 65 6f 20 4c 65 6e 73 20 52 65 6d 6f 74
65 20 53 63 6f 70 65 20 76 36 2e 38 2e 30 0d 0a
43 6f 6e 65 63 74 69 6f 6e 3a 20 4b 65 65 70
2d 41 6c 69 76 65 0d 0a 0d 0a
```

and your firewall should decode this.
Firewall rules

Type 1: a combination of

- Protocol (TCP/UDP/ICMP)
- IP address or country (e.g., Canada)
- Port number

Type 2: domain names

- E.g., block DNS queries for *.facebook.com
NO CHEATING
WE RUN COPY CHECKER
Questions?

- General questions
  - Ask your favorite GSI

- Project-specific questions
  - Sangjin Han (main)
  - Steve Wang
  - Kaifei Chen
  - Aurojit Panda
DNS and the Web (wrap up) + Link Layer

EE 122, Fall 2013
Sylvia Ratnasamy
http://inst.eecs.berkeley.edu/~ee122/

Material thanks to Ion Stoica, Scott Shenker, Jennifer Rexford, Nick McKeown, and many other colleagues
Announcements (1)

- Midterm solutions now posted
- We will accept regrade requests received by 5pm, Nov 11
- Regrade process if we clearly made a mistake:
  - e.g., total is incorrect; correct selection in multiple choice, etc.
  - Bring it to the attention of your TA/me when you look over your exam
  - If your TA/I agree, we’ll correct your score immediately
- Regrade process if you disagree with our assessment
  - submit a <1-page request, explaining your point
  - we will regrade your entire exam
  - process also described on the course webpage
- We’ll return your exams after Nov 11
Announcements (2)

- Midterm grades
Last Time

- Three approaches to improving content delivery
  - Compensate for TCP’s weaknesses
  - Caching and replication
  - Exploit economies of scale
HTTP Performance

- Most Web pages have multiple objects
  - e.g., HTML file and a bunch of embedded images

- How do you retrieve those objects (naively)?
  - One item at a time

- New TCP connection per (small) object → Slow!
  - Minimum of 2RTTs per object
Improving HTTP Performance: **Concurrent Requests & Responses**

- Use multiple connections *in parallel*
Improving HTTP Performance:

**Persistent Connections**

- Maintain TCP connection across multiple requests (and even user “sessions”)
  - Amortize overhead of connection set-up and tear-down
  - Allow TCP to learn more accurate RTT estimate
  - Allow TCP congestion window to increase

- Default in HTTP/1.1
Improving HTTP Performance:

**Pipelined Requests & Responses**

- Batch requests and responses to reduce the number of packets.
- Multiple requests can be contained in one TCP segment.
Scorecard: Getting $n$ Small Objects

Time dominated by latency

- One-at-a-time: $\sim 2n$ RTT
- M concurrent: $\sim 2[n/m]$ RTT
- Persistent: $\sim (n+1)RTT$
- Pipelined: $\sim 2$ RTT
- Pipelined/Persistent: $\sim 2$ RTT first time, RTT later
Scorecard: Getting $n$ Large Objects

*Time dominated by bandwidth*

$(F$ is object size, $B$ is bandwidth)

- One-at-a-time: $\sim nF/B$
- M concurrent: $\sim [n/m] F/B$
  - assuming shared with large population of users
  - and each TCP connection gets the same bandwidth
- Pipelined and/or persistent: $\sim nF/B$
- The only thing that helps is getting more bandwidth..
Improving HTTP Performance: Caching

● Why does caching work?
  ● Exploits *locality of reference*

● How well does caching work?
  ● Very well, up to a limit
  ● Large overlap in content
  ● But many unique requests
Improving HTTP Performance:
Caching: How

- Modifier to GET requests:
  - `If-modified-since` – returns “not modified” if resource not modified since specified time

```
GET /~ee122/fa13/ HTTP/1.1
Host: inst.eecs.berkeley.edu
User-Agent: Mozilla/4.03
If-modified-since: Sun, 27 Oct 2013 22:25:50 GMT
<CRLF>
```

- Client specifies “if-modified-since” time in request
- Server compares this against “last modified” time of resource
- Server returns “Not Modified” if resource has not changed
- …. or a “OK” with the latest version otherwise
Improving HTTP Performance:

Caching: How

- **Modifier to GET requests:**
  - `If-modified-since` – returns “not modified” if resource not modified since specified time

- **Response header:**
  - `Expires` – how long it’s safe to cache the resource
  - `No-cache` – ignore all caches; always get resource directly from server
Improving HTTP Performance:

Caching: Where?

- Options
  - Client
  - Forward proxies
  - Reverse proxies
  - Content Distribution Network
Baseline: Many clients transfer same information
- Generate unnecessary server and network load
- Clients experience unnecessary latency
Improving HTTP Performance: Caching with “Reverse Proxies”

- Cache documents close to server → decrease server load
- Typically done by content provider
Improving HTTP Performance:
Caching with “Forward Proxies”

- Cache documents close to clients
  → reduce network traffic and decrease latency
- Typically done by ISPs or enterprises
Improving HTTP Performance:

Content Distribution Networks

- Caching and replication as a service
- Large-scale distributed storage infrastructure (usually) administered by one entity
  - *e.g.*, Akamai has servers in 20,000+ locations
- Combination of (pull) caching and (push) replication
  - **Pull**: Direct result of clients’ requests
  - **Push**: Expectation of high access rate
- Also do some processing
  - Handle *dynamic* web pages
  - *Transcoding*
Improving HTTP Performance: CDN Example – Akamai

- Akamai creates new domain names for each client
  - e.g., \texttt{a128.g.akamai.net} for \texttt{cnn.com}

- The client content provider modifies its content so that embedded URLs reference the new domains.
  - “Akamaize” content
    - e.g.: \texttt{http://www.cnn.com/image-of-the-day.gif} becomes \texttt{http://a128.g.akamai.net/image-of-the-day.gif}

- Requests now sent to CDN’s (i.e., Akamai’s) infrastructure…
Cost-Effective Content Delivery

- Examples:
  - Web hosting companies
  - CDNs
  - Cloud infrastructure

- Common theme: multiple sites hosted on shared physical infrastructure
  - efficiency of statistical multiplexing
  - economies of scale (volume pricing, etc.)
  - amortization of human operator costs
Data Link Layer
Point-to-Point vs. Broadcast Media

- **Point-to-point:** dedicated pairwise communication
  - E.g., long-distance fiber link
  - E.g., Point-to-point link between Ethernet switch and host

- **Broadcast:** shared wire or medium
  - Traditional Ethernet (pre ~2000)
  - 802.11 wireless LAN
Given a shared broadcast channel
- Must avoid having multiple nodes speaking at once
- Otherwise, collisions lead to garbled data
- Need algorithm that determines which node can transmit

Three classes of techniques
- Channel partitioning: divide channel into pieces
- Taking turns: scheme for trading off who gets to transmit
- Random access: allow collisions, and then recover
“Taking Turns” MAC protocols

Polling

- Master node “invites” slave nodes to transmit in turn

- Concerns:
  - Polling overhead
  - Latency
  - Single point of failure (master)

Token passing

- Control token passed from one node to next sequentially

- Node must have token to send

- Concerns:
  - Token overhead
  - Latency
  - At mercy of any node
None of these are the “Internet way”…

- What’s wrong with
  - TDMA
  - FDMA
  - Polling
  - Token passing

- Turn to random access
  - Optimize for the common case (no collision)
  - Don’t avoid collisions, just recover from them
  - Should sound familiar…
Random Access MAC Protocols
Random Access MAC Protocols

- When node has packet to send
  - Transmit at full channel data rate
  - No *a priori* coordination among nodes
- Two or more transmitting nodes $\Rightarrow$ collision
  - Data lost
- Random access MAC protocol specifies:
  - How to detect collisions
  - How to recover from collisions
- Examples
  - ALOHA and Slotted ALOHA
  - CSMA, CSMA/CD, CSMA/CA (wireless, covered later)
Where it all Started: AlohaNet

- Norm Abramson left Stanford in 1970 *(so he could surf!)*
- Set up first data communication system for Hawaiian islands
- Central hub at U. Hawaii, Oahu
Aloha Signaling

- Two channels: random access, broadcast
  - Sites send packets to hub (random-access channel)
    - If not received (due to collision), site resends
  - Hub sends packets to all sites (broadcast channel)
    - Sites can receive even if they are also sending

- Questions:
  - When do you resend? Resend with probability $p$
  - How does this perform? Need a clean model….
Slotted ALOHA

Model/Assumptions
- All frames same size
- Time divided into equal slots (time to transmit a frame)
- Nodes are synchronized
- Nodes begin to transmit frames only at start of slots
- If multiple nodes transmit, nodes detect collision

Operation
- When node gets fresh data, transmits in next slot
- No collision: success!
- Collision: node retransmits with probability $p$ until success
Slot-by-Slot Example
Efficiency of Slotted Aloha

- Suppose N stations have packets to send
  - Each transmits in slot with probability $p$

- Probability of successful transmission:
  - by a particular node $i$: $S_i = p \cdot (1-p)^{(N-1)}$
  - by any of N nodes: $S = N \cdot p \cdot (1-p)^{(N-1)}$

- What value of $p$ maximizes prob. of success:
  - For fixed $p$, $S \rightarrow 0$ as $N$ increases
  - But if $p = 1/N$, then $S \rightarrow 1/e = 0.37$ as $N$ increases

- Max efficiency is only slightly greater than 1/3!
Improving on Slotted Aloha

- Fewer wasted slots
  - Need to decrease collisions and empty slots

- Don’t waste full slots on collisions
  - Need to decrease time to detect collisions

- Avoid need for synchronization
  - Synchronization is hard to achieve
  - And Aloha performance drops if you don’t have slots!
CSMA (Carrier Sense Multiple Access)

- CSMA: listen before transmit
  - If channel sensed idle: transmit entire frame
  - If channel sensed busy, defer transmission

- Human analogy: don’t interrupt others!

- Does this eliminate all collisions?
  - No, because of nonzero propagation delay
CSMA Collisions

Propagation delay: two nodes may not hear each other’s before sending.

Would slots hurt or help?

CSMA reduces but does not eliminate collisions

Biggest remaining problem?

Collisions still take full slot!
CSMA/CD (Collision Detection)

- CSMA/CD: carrier sensing, deferral as in CSMA
  - Collisions detected within short time
  - Colliding transmissions aborted, reducing wastage

- Collision detection easy in wired (broadcast) LANs
  - Compare transmitted, received signals

- Collision detection difficult in wireless LANs
  - next lecture
CSMA/CD Collision Detection

B and D can tell that collision occurred.

Note: for this to work, need restrictions on minimum frame size and maximum distance.

Why?
Limits on CSMA/CD Network Length

- Latency depends on physical length of link
  - Time to propagate a packet from one end to the other
- Suppose A sends a packet at time $t$
  - And B sees an idle line at a time just before $t+d$
  - ... so B happily starts transmitting a packet
- B detects a collision, and sends jamming signal
  - But A can't see collision until $t+2d$
A needs to wait for time $2d$ to detect collision
- So, A should keep transmitting during this period
- ... and keep an eye out for a possible collision

Imposes restrictions. E.g., for 10 Mbps Ethernet:
- Maximum length of the wire: 2,500 meters
- Minimum length of a frame: 512 bits (64 bytes)
  - 512 bits = 51.2 $\mu$sec (at 10 Mbit/sec)
  - For light in vacuum, 51.2 $\mu$sec $\approx$ 15,000 meters vs. 5,000 meters “round trip” to wait for collision

What about 10Gbps Ethernet?
Performance of CSMA/CD

- Time wasted in collisions
  - Proportional to distance $d$
- Time spend transmitting a packet
  - Packet length $p$ divided by bandwidth $b$
- Rough estimate for efficiency (K some constant)
  $$E \sim \frac{p}{b} + Kd$$

Note:
- For large packets, small distances, $E \sim 1$
- As bandwidth increases, $E$ decreases
- That is why high-speed LANs are all switched
Recap: Key Ideas of Random Access

1. **Carrier sense**
   - Listen before speaking, and don’t interrupt
   - Checking if someone else is already sending data
   - … and waiting till the other node is done

2. **Collision detection**
   - If someone else starts talking at the same time, stop
     - *But make sure everyone knows there was a collision!*
   - Realizing when two nodes are transmitting at once
   - …by detecting that the data on the wire is garbled

3. **Randomness**
   - *Don’t start talking again right away*
   - Waiting for a random time before trying again
Bob Metcalfe, Xerox PARC, visits Hawaii and gets an idea!

Shared wired medium
- coax cable
Evolution

- Ethernet was invented as a broadcast technology
  - Hosts share channel
  - Each packet received by all attached hosts
  - CSMA/CD for media access control

- Current Ethernets are “switched”
  - Point-to-point links between switches; between a host and switch
  - No sharing, no CSMA/CD
  - (Next lecture) uses “self learning” and “spanning tree” algorithms for routing
Ethernet: CSMA/CD Protocol

- **Carrier sense**: wait for link to be idle
- **Collision detection**: listen while transmitting
  - No collision: transmission is complete
  - Collision: abort transmission & send jam signal
- **Random access**: *binary exponential back-off*
  - After collision, wait a random time before trying again
  - After m\(^{th}\) collision, choose K randomly from \{0, …, 2\(^m\)-1\}
  - … and wait for K*512 bit times before trying again
  - If transmission occurring when ready to send, wait until end of transmission (CSMA)
**Ethernet Frame Structure**

- **Encapsulates IP datagram**

- **Preamble**: 7 bytes with a particular pattern used to synchronize receiver, sender clock rates

- **Addresses**: 6 bytes: frame is received by all adapters on a LAN and dropped if address does not match

- **Type**: 2 bytes, indicating higher-layer protocol (e.g., IP, Appletalk)

- **CRC**: 4 bytes for error detection

- **Data payload**: maximum 1500 bytes, minimum 46 bytes
MAC Addresses
Medium Access Control Address

- **MAC address**
  - Numerical address associated with an adapted
  - Flat name space of 48 bits (e.g., 00-15-C5-49-04-A9 in HEX)
  - Unique, hard-coded in the adapter when it is built

- **Hierarchical Allocation**
  - Blocks: assigned to vendors (e.g., Dell) by the IEEE
    - First 24 bits (e.g., 00-15-C5-**-**-**)
  - Adapter: assigned by the vendor from its block
    - Last 24 bits

- **Broadcast address** (FF-FF-FF-FF-FF-FF-FF)
  - Send the frame to *all* adapters
MAC Address vs. IP Address

- **MAC addresses** (used in link-layer)
  - **Hard-coded** in read-only memory when adapter is built
  - Like a social security number
  - **Flat** name space of 48 bits (e.g., 00-0E-9B-6E-49-76)
  - Portable, and can stay the same as the host moves
  - Used to get packet between interfaces on same network

- **IP addresses**
  - **Configured**, or learned dynamically
  - Like a postal mailing address
  - **Hierarchical** name space of 32 bits (e.g., 12.178.66.9)
  - Not portable, and depends on where the host is attached
  - Used to get a packet to destination IP subnet
Two protocols used to bootstrap communication: DHCP and ARP
Who Am I: Acquiring an IP Address

- Dynamic Host Configuration Protocol (DHCP)
  - Broadcast “I need an IP address, please!”
  - Response “You can have IP address 1.2.3.4.”
Who Are You: Discovering the Receiver

- Address Resolution Protocol (ARP)
  - Broadcast “who has IP address 1.2.3.6?”
  - Response “0C-C4-11-6F-E3-98 has 1.2.3.6!”
Dynamic Host Configuration Protocol

- DHCP configures several aspects of hosts
  - Most important: temporary IP address (lease)
  - But also: local DNS name server, gateway router, netmask

- DHCP server does the allocation
  - Multiplexes block of addresses across users

- DHCP protocol:
  - Broadcast (at layer 2) a server-discovery message
  - Server(s) sends a reply offering an address
Response from the DHCP Server

- DHCP “offer” message from the server
  - Informs the client of various configuration parameters (proposed IP address, mask, gateway router, DNS server, ...)
  - Lease time (duration the information remains valid)
Response from the DHCP Server

- **DHCP “offer” message from the server**
  - Informs the client of various configuration parameters (proposed IP address, mask, gateway router, DNS server, ...)
  - Lease time (duration the information remains valid)

- **Multiple servers may respond**
  - Multiple DHCP servers on the same broadcast network

- **Client accepts one of the offers**
  - Client sends a DHCP “request” echoing the parameters
  - The DHCP server responds with an “ACK” to confirm
  - … and the other servers see they were not chosen
Dynamic Host Configuration Protocol

- **arriving client**
- **DHCP server 203.1.2.5**

**Why all the broadcasts?**

- DHCP discover **(broadcast)**
- DHCP offer **(broadcast)**
- DHCP request **(broadcast)**
- DHCP ACK **(broadcast)**
DHCP Uses “Soft State”

- Soft state: if not refreshed state will be forgotten
  - Install state with timer, reset timer when refresh arrives
  - Delete state if refresh not received when timer expires
  - Allocation of address is “soft state” (renewable lease)

- Why does DHCP “lease” addresses?
  - Host might not release the address
    - E.g., host crashed, buggy client software
  - And you don’t want the address to be allocated forever
  - So if request isn’t refreshed, server takes address back
ARP
Adapters only understand MAC addresses
- Translate the destination IP address to MAC address
- Encapsulate the IP packet inside a link-level frame
Address Resolution Protocol

- Every node maintains an **ARP** table
  - <IP address, MAC address> pair

- Consult the table when sending a packet
  - Map destination IP address to destination MAC address
  - Encapsulate and transmit the data packet

- But: what if IP address **not** in the table?
  - Sender **broadcasts**: “Who has IP address 1.2.3.156?”
  - Receiver responds: “MAC address 58-23-D7-FA-20-B0”
  - Sender **caches** result in its ARP table
What if the destination is remote?

- Look up the MAC address of the first hop router
  - 1.2.3.48 uses ARP to find MAC address for first-hop router 1.2.3.19 rather than ultimate destination IP address

- How does the red host know the destination is not local?
  - Uses netmask (discovered via DHCP)

- How does the red host know about 1.2.3.19?
  - Also DHCP
Key Ideas in Both ARP and DHCP

- **Broadcasting**: Can use broadcast to make contact
  - Scalable because of limited size

- **Caching**: remember the past for a while
  - Store the information you learn to reduce overhead
  - Remember your own address & other host’s addresses

- **Soft state**: eventually forget the past
  - Associate a *time-to-live* field with the information
  - … and either refresh or discard the information
  - Key for *robustness* in the face of unpredictable change
## Taking Stock: Naming

<table>
<thead>
<tr>
<th>Layer</th>
<th>Examples</th>
<th>Structure</th>
<th>Configuration</th>
<th>Resolution Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>App. Layer</td>
<td><a href="http://www.cs.berkeley.edu">www.cs.berkeley.edu</a></td>
<td>organizational hierarchy</td>
<td>~ manual</td>
<td>DNS</td>
</tr>
<tr>
<td>Network Layer</td>
<td>123.45.6.78</td>
<td>topological hierarchy</td>
<td>DHCP</td>
<td>ARP</td>
</tr>
<tr>
<td>Link layer</td>
<td>45-CC-4E-12-F0-97</td>
<td>vendor (flat)</td>
<td>hard-coded</td>
<td></td>
</tr>
</tbody>
</table>
**Next Time**

- Walk through the steps in fetching a web page
  - End-to-end
  - Application layer to link layer

- With some missing pieces along the way
  - NAT
  - Middleboxes
Recap: Steps in reaching a Host

- First look up IP address
  - Search engines + DNS

- Need to know where local DNS server is
  - DHCP

- Also needs to know its own IP address
  - DHCP
Sending a Packet

- On same subnet:
  - Need MAC address of destination: ARP

- On some other subnet:
  - Need MAC address of first-hop router: ARP

- Need to tell whether destination is on same or other subnet?
  - Use the netmask: DHCP
Example: A Sending a Packet to B

How does host A send an IP packet to host B?

1. A sends packet to R.
2. R sends packet to B.
Example: A Sending a Packet to B

How does host A send an IP packet to host B?
Host A Decides to Send Through R

- Host A constructs an IP packet to send to B
  - Source 111.111.111.111, destination 222.222.222.222
- Host A has a gateway router R
  - Used to reach destinations outside of 111.111.111.0/24
  - Address 111.111.111.110 for R learned via DHCP
Host A Sends Packet Through R

- Host A learns the MAC address of R’s interface
  - ARP request: broadcast request for 111.111.111.110
  - ARP response: R responds with E6-E9-00-17-BB-4B
- Host A encapsulates the packet and sends to R
R Decides how to Forward Packet

- Router R’s adapter receives the packet.
- R extracts the IP packet from the Ethernet frame.
- R sees the IP packet is destined to 222.222.222.222.
- Router R consults its forwarding table.
  - Packet matches 222.222.222.0/24 via other adapter.

Two points:
- Routing table points to this port.
- Destination address is within mask of port’s address (i.e., local).
R Sends Packet to B

- Router R’ s learns the MAC address of host B
  - ARP request: broadcast request for 222.222.222.222
  - ARP response: B responds with 49-BD-D2-C7-56-2A
- Router R encapsulates the packet and sends to B