DNS and HTTP

CSE 461 Section (Week 4)
Addressing So Far

- Port numbers for applications
- MAC addresses for hardware
- IP addresses for a way to send data in a smart, routable way
Problems with MACs/IPs/Ports

• Humans (and dogs!) are bad at remembering strings of numbers
• We need a human-friendly naming system!
Requirements for Human-Readable Naming System

- What do we need?
  - As short as possible
  - Easy to memorize (i.e., not arbitrary)
  - Unique
  - Customizable
  - Hierarchical
  - Reflect organizational structure
  - A way to quickly translate to and from the existing, computer-friendly addressing systems
  - Ideally, we’d like to address specific resources as well
• Human-readable “domain names” map to IP addresses (names < 254 characters)
• A human can type www.google.com into their browser, and the browser will (somehow) know to go to 173.194.33.179
• But how might this be done?
  – Some sort of hash (not really practical)
  – A file of all of the mappings
  – Separate servers to provide the mappings
In the Beginning...

- All domain name/IP address mappings stored in /etc/hosts
- Live demo
- But this sucked... why?
  - Doesn’t scale to large number of domain names
  - Not authoritative
  - Errors common
DNS Servers Are the Answer!

• Systems keep a small cache of mappings they know
• When a domain name is used that isn’t in the cache, the system queries a name server
• Simple communication on port 53
• Database is distributed
• Hierarchical namespace: it’s name servers all the way down
Resolving a Domain Name

• If I type shop.spacex.com, what happens?
  – Check /etc/hosts
  – Check local DNS server
  – Go down hierarchy and ask:
    • Ask . DNS root server
    • Ask .com TLD (Top Level Domain) server
    • Ask spacex.com’s NS
    • Send HTTP request to the IP address obtained
Domain Names in Practice

• Who’s purchased a domain name before?
• Name registrars
• ICANN
  – Internet Corporation for Assigned Names and Numbers
• Propagation delays
Multiple IP Addresses and Aliasing

• DNS servers can return different IP address results for the same domain name
• Why is this useful?
• Demo

• Also, multiple domain names can map to one IP address
• Why is this useful?
• Demo
Attacks and Other Fun

• What are some ways this system can break?
  – DoS attacks on DNS server
    • Done before, in 2002 and 2007
    • Not much impact due to filtering and caching
  – Return incorrect IP address to a DNS request
  – Could even return the IP of our own server!
    • Commonly done by ISPs
  – Compromise root servers

• Google DNS
HTTP Request Types

• GET
  – Page, please

• HEAD
  – Metadata, please

• POST
  – Here’s some data—update what you have

• PUT
  – Here’s some data—create a new resource

• CONNECT
  – Make a tunnel for me through your proxy!
CONNECT and Tunneling

• CONNECT requests say, “give me a tunnel!”
• Used for communication via proxies
• It’s a virtual connection—it means “forward all of the packets that I send to the destination, and vice-versa
• You don’t *have* to use CONNECT with proxies...
• But you do when HTTPS/SSL is used... why?
HTTP Headers: Host

• Specifies the hostname (and, optionally, the port) to which to send a request

• Example:
GET /Passport.aspx?popup=1 HTTP/1.1
Host: www.bing.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:34.0) Gecko/20100101
Firefox/34.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
HTTP Headers: Connection

• Standard header for keep-alives
  – “Connection: keep-alive” is sent to keep a connection alive
  – If you don’t want to keep it alive, send “Connection: close”
  – We’ll rewrite packets to do this for the proxy project
• Also used to switch from HTTP /1.1 to HTTP/2
  – Client sends “Connection: upgrade” to do this
• Example:
  GET /ajax/libs/jquery/1.7.1/jquery.min.js
  HTTP/1.1
  Host: ajax.googleapis.com
  Connection: keep-alive
  Accept: */*
Buffering Requests

• When writing a proxy, you may want to buffer requests
• This is a good idea to keep your network traffic and thread consumption reasonable
TCP Streams

• UDP sends a string of messages; no concept of “connection”

• TCP is like a direct pipe from one node to another
  – No concept of message boundaries
  – Easy to associate data with the data that preceded it
  – Gives you guarantees that data you send will arrive