ATTRIBUTION

These slides are released under an Attribution-NonCommercial-ShareAlike 3.0 Unported (CC BY-NC-SA 3.0) Creative Commons license
AGENDA

Today:
- Chapter 2 (DNS), 3, and first few pages of 4 of “Network Programming with Go”

Next week:
- In class demo/exercises focused on Go’s “net” package

TODO

1. Project 1 due Tuesday Jan 11 at 5pm
A REQUEST

I am once again asking to please stop emailing me about the wait list and class size. It is out of my control!
Outline

1. Performance
2. Layering
3. Addressing
4. DNS
IP VERSION 4 (IPV4)

Binary: 11000000 . 10101000 . 00000001 . 00001010
Decimal: 192 . 168 . 1 . 10
CLASS-BASED ADDRESSING (NOT REALLY USED ANYMORE)

- Most significant bits determines “class” of address

<table>
<thead>
<tr>
<th>Class</th>
<th>Network Bits</th>
<th>Host Bits</th>
<th>Number of Networks</th>
<th>Number of Hosts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class A</td>
<td>0</td>
<td>14, 16</td>
<td>127</td>
<td>16M</td>
</tr>
<tr>
<td>Class B</td>
<td>1, 0</td>
<td>21, 8</td>
<td>16K</td>
<td>64K</td>
</tr>
<tr>
<td>Class C</td>
<td>1, 1, 0</td>
<td></td>
<td>2M</td>
<td>254</td>
</tr>
</tbody>
</table>

- Special addresses
  - Class D (1110) for multicast, Class E (1111) experimental
  - 127.0.0.1: local host (a.k.a. the loopback address)
  - Host bits all set to 0: network address
  - Host bits all set to 1: broadcast address
CLASS-BASED ADDRESSING EXAMPLES

<table>
<thead>
<tr>
<th>Network ID</th>
<th>First octet</th>
<th>Second octet</th>
<th>Third octet</th>
<th>Fourth octet</th>
<th>Host ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 bits</td>
<td>Network 10</td>
<td>Host 1</td>
<td>Host 2</td>
<td>Host 3</td>
<td>24 bits</td>
</tr>
<tr>
<td>16 bits</td>
<td>Network 172</td>
<td>Network 16</td>
<td>Host 1</td>
<td>Host 2</td>
<td>16 bits</td>
</tr>
<tr>
<td>24 bits</td>
<td>Network 192</td>
<td>Network 168</td>
<td>Network 1</td>
<td>Host 2</td>
<td>8 bits</td>
</tr>
</tbody>
</table>
ADDRESSING EXAMPLE

![Diagram of IP address structure with binary representation and network and host ID calculations.]
IP ADDRESS PROBLEM (1991)

• Address space depletion
  • In danger of running out of classes A and B

• Why?
  • Class C too small for most organizations (only ~250 addresses)
  • Very few class A – very careful about giving them out (who has 16M hosts anyway?)
  • Class B – greatest problem
CIDR

- Classless Inter-Domain Routing (1993)
  - Networks described by variable-length prefix and length
  - Allows arbitrary allocation between network and host address
  - e.g. 10.95.1.2 contained within 10.0.0.0/8:
    - 10.0.0.0 is network and remainder (95.1.2) is host
  - Pro: Finer grained allocation; aggregation
  - Con: More expensive lookup: longest prefix match
SUBNETS AND NETMASKS

Prefix length = L bits

32 - L bits

Network

Host

Subnet mask

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0
192.168.156.97/19

Network ID: 192.168.128.0
Host ID: 0.0.28.97
ADDRESS AGGREGATION EXAMPLE

Advertise 212.56.132.0/22

Customer A
212.56.132.0/24

Customer B
212.56.133.0/24

Customer C
212.56.134.0/24

Customer D
212.56.135.0/24

212.56.132.0/24 11010100.00111000.100001\textbf{00}.00000000
212.56.133.0/24 11010100.00111000.100001\textbf{01}.00000000
212.56.134.0/24 11010100.00111000.100001\textbf{10}.00000000
212.56.135.0/24 11010100.00111000.100001\textbf{11}.00000000

Common Prefix: 11010100.00111000.100001\textbf{00}.00000000
^^^^^^^^ 22 bits in common  ^^
Routing
• Packet to 10.1.1.6
• Matches 10.1.0.0/23

<table>
<thead>
<tr>
<th>Destination</th>
<th>Next Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>127.0.0.1</td>
<td>loopback</td>
</tr>
<tr>
<td>Default or 0/0</td>
<td>10.1.16.1</td>
</tr>
<tr>
<td>10.1.8.0/24</td>
<td>interface1</td>
</tr>
<tr>
<td>10.1.2.0/23</td>
<td>interface2</td>
</tr>
<tr>
<td><strong>10.1.0.0/23</strong></td>
<td><strong>10.1.2.2</strong></td>
</tr>
<tr>
<td>10.1.16.0/24</td>
<td>interface3</td>
</tr>
</tbody>
</table>
Route Table Example 2 (R1)

- Packet to 10.1.1.6
- Matches 10.1.1.4/30
- Longest prefix match

Routing table at R1

<table>
<thead>
<tr>
<th>Destination</th>
<th>Next Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>127.0.0.1</td>
<td>loopback</td>
</tr>
<tr>
<td>Default or 0/0</td>
<td>10.1.2.1</td>
</tr>
<tr>
<td>10.1.0.0/24</td>
<td>interface1</td>
</tr>
<tr>
<td>10.1.1.0/24</td>
<td>interface2</td>
</tr>
<tr>
<td>10.1.2.0/23</td>
<td>interface3</td>
</tr>
<tr>
<td><strong>10.1.1.4/30</strong></td>
<td><strong>10.1.1.101</strong></td>
</tr>
</tbody>
</table>
Routing table at H1

<table>
<thead>
<tr>
<th>Destination</th>
<th>Next Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>127.0.0.1</td>
<td>loopback</td>
</tr>
<tr>
<td>Default or 0/0</td>
<td>10.1.1.1</td>
</tr>
<tr>
<td>10.1.1.0/24</td>
<td>interface1</td>
</tr>
<tr>
<td>10.1.1.4/30</td>
<td>interface2</td>
</tr>
</tbody>
</table>
• IP addresses identify a *machine*

• Actually they identify a network interface on a machine

• How to identify different programs on the machine?

• Process ID/PID? (no... why not?)

• Instead we use a port (which is a 16-bit number)

• 0-1024 reserved for the OS, you can use 1025-65535
NETWORK ADDRESS TRANSLATION

10.0.0.2

10.0.0.3

Network address translation

10.0.0.3:50926

1.2.3.4:50926

Internet
LOAD BALANCING VIA NETWORK ADDRESS TRANSLATION

Diagram showing network configuration with local network (192.168.100.3, 192.168.100.4, 192.168.100.5) connecting to a router/NAT device with a default gateway (192.168.1.1) and public IP address (145.12.131.7). The diagram illustrates the translation of private IP addresses to a public IP address through a router. 

LOAD BALANCING VIA NETWORK ADDRESS TRANSLATION

### NAT Translation Table

<table>
<thead>
<tr>
<th>Private IP Addr &amp; Port</th>
<th>Public IP Addr &amp; Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.100.3, 3855</td>
<td>145.12.131.7, 6282</td>
</tr>
<tr>
<td>……</td>
<td>……</td>
</tr>
</tbody>
</table>

#### Step 1
Source: 192.168.100.3, 3855  
Dest: 209.131.36.158, 80  
(www.yahoo.com)

#### Step 2
Map private IP & port to public IP & port

#### Step 3
Source: 145.12.131.7, 6282  
Dest: 209.131.36.158, 80  
(www.yahoo.com)

Please fetch http://www.yahoo.com

Router/NAT Device

Default Gateway  
192.168.1.1  
(Public IP Address)

145.12.131.7  
(To Yahoo)
LOAD BALANCING VIA NETWORK ADDRESS TRANSLATION

Step 5: map public IP & port back to private IP & port

NAT translation table

<table>
<thead>
<tr>
<th>Private IP &amp; Port</th>
<th>Public IP &amp; Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.100.3, 3855</td>
<td>145.12.131.7, 6282</td>
</tr>
</tbody>
</table>

Step 4
Source: 209.131.36.158, 80
Dest: 145.12.131.7, 6282

Step 6
Source: 209.131.36.158, 80
Dest: 192.168.100.3, 3855

Router/NAT Device

Default Gateway
192.168.1.1

145.12.131.7
(Public IP Address)

From Yahoo

Networks use **Border Gateway Protocol (BGP)** to announce reachability.

- Each network talks just with its neighbors.
- Goal is to get a packet to the destination network.
  - It is up to that destination network to get individual packets to their ultimate destination.
  - Back-to-back packets from the same “connection” might take different paths!
  - Might arrive out of order too!
Outline

1. Performance
2. Layering
3. Addressing
4. DNS
DNS HOSTNAME VERSUS IP ADDRESS

- **DNS host name** (e.g. www.cs.ucsd.edu)
  - Mnemonic name appreciated by humans
  - Variable length, full alphabet of characters
  - Provides little (if any) information about location

- **IP address** (e.g. 128.112.136.35)
  - Numerical address appreciated by routers
  - Fixed length, decimal number
  - Hierarchical address space, related to host location
MANY USES OF DNS

• Hostname to IP address translation
• IP address to hostname translation (reverse lookup)
• Host name aliasing: other DNS names for a host
• Alias host names point to canonical hostname
• Email: Lookup domain’s mail server by domain name
ORIGINAL DESIGN OF DNS

• Per-host file named `/etc/hosts` (1982)
  • Flat namespace: each line = IP address & DNS name
  • SRI (Menlo Park, California) kept the master copy
  • Everyone else downloads regularly
• But, a single server doesn’t scale
  • Traffic implosion (lookups and updates)
  • Single point of failure
• Need a distributed, hierarchical **collection** of servers
DNS: GOALS AND NON-GOALS

- A wide-area distributed database
- Goals:
  - Scalability; decentralized maintenance
  - Robustness
  - Global scope
    - Names mean the same thing everywhere
  - Distributed updates/queries
  - Good performance
**DOMAIN NAME SYSTEM (DNS)**

- Hierarchical name space divided into contiguous sections called **zones**
  - Zones are distributed over a collection of DNS servers

- Hierarchy of DNS servers:
  - **Root** servers (identity hardwired into other servers)
  - **Top-level domain (TLD)** servers
  - **Authoritative** DNS servers

- Performing the translations:
  - **Local DNS servers** located near clients
  - **Resolver** software running on clients
DNS IS HIERARCHICAL

- Hierarchy of namespace matches hierarchy of servers
- Set of nameservers answers queries for names within zone
- Nameservers store names and links to other servers in tree
DNS ROOT NAMESERVERS

- 13 root servers

A Verisign, Dulles, VA
B USC-ISI Marina del Rey, CA
C Cogent, Herndon, VA
D U Maryland College Park, MD
E NASA Mt View, CA
F Internet Software Consortium, Palo Alto, CA
G US DoD Vienna, VA
H ARL Aberdeen, MD
I Autonomica, Stockholm
J Verisign
K ICANN Los Angeles, CA
L M WIDE Tokyo
TLD AND AUTHORITATIVE SERVERS

- **ftp://ftp.internic.net/domain/named.root**
- **Top-level domain (TLD) servers**
  - Responsible for com, org, net, edu, etc, and all top-level country domains: uk, fr, ca, jp
  - Network Solutions maintains servers for com TLD
  - Educause non-profit for edu TLD
- **Authoritative DNS servers**
  - An organization’s DNS servers, providing authoritative information for that organization
  - May be maintained by organization itself, or ISP
## COMMON TLDS

<table>
<thead>
<tr>
<th>Domain</th>
<th>Intended use</th>
<th>Start date</th>
<th>Restricted?</th>
</tr>
</thead>
<tbody>
<tr>
<td>com</td>
<td>Commercial</td>
<td>1985</td>
<td>No</td>
</tr>
<tr>
<td>edu</td>
<td>Educational institutions</td>
<td>1985</td>
<td>Yes</td>
</tr>
<tr>
<td>gov</td>
<td>Government</td>
<td>1985</td>
<td>Yes</td>
</tr>
<tr>
<td>int</td>
<td>International organizations</td>
<td>1985</td>
<td>Yes</td>
</tr>
<tr>
<td>mil</td>
<td>Military</td>
<td>1985</td>
<td>Yes</td>
</tr>
<tr>
<td>net</td>
<td>Network providers</td>
<td>1985</td>
<td>No</td>
</tr>
<tr>
<td>org</td>
<td>Non-profit organizations</td>
<td>1985</td>
<td>No</td>
</tr>
<tr>
<td>aero</td>
<td>Air transport</td>
<td>2001</td>
<td>Yes</td>
</tr>
<tr>
<td>biz</td>
<td>Businesses</td>
<td>2001</td>
<td>No</td>
</tr>
<tr>
<td>coop</td>
<td>Cooperatives</td>
<td>2001</td>
<td>Yes</td>
</tr>
<tr>
<td>info</td>
<td>Informational</td>
<td>2002</td>
<td>No</td>
</tr>
<tr>
<td>museum</td>
<td>Museums</td>
<td>2002</td>
<td>Yes</td>
</tr>
<tr>
<td>name</td>
<td>People</td>
<td>2002</td>
<td>No</td>
</tr>
<tr>
<td>pro</td>
<td>Professionals</td>
<td>2002</td>
<td>Yes</td>
</tr>
<tr>
<td>cat</td>
<td>Catalan</td>
<td>2005</td>
<td>Yes</td>
</tr>
<tr>
<td>jobs</td>
<td>Employment</td>
<td>2005</td>
<td>Yes</td>
</tr>
<tr>
<td>mobi</td>
<td>Mobile devices</td>
<td>2005</td>
<td>Yes</td>
</tr>
<tr>
<td>tel</td>
<td>Contact details</td>
<td>2005</td>
<td>Yes</td>
</tr>
<tr>
<td>travel</td>
<td>Travel industry</td>
<td>2005</td>
<td>Yes</td>
</tr>
<tr>
<td>xxx</td>
<td>Sex industry</td>
<td>2010</td>
<td>No</td>
</tr>
</tbody>
</table>
LOCAL NAME SERVERS

- Do not strictly belong to hierarchy
- Each ISP (or company, or university) has one
  - Also called default or caching name server
- When host makes DNS query, query is sent to its local DNS server
  - Acts as proxy, forwards query into hierarchy
  - Does work for the client
DNS RESOURCE RECORDS

- DNS is a distributed database storing **resource records**
- Resource record includes: (name, type, value, time-to-live)

<table>
<thead>
<tr>
<th>Type</th>
<th>Meaning</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOA</td>
<td>Start of authority</td>
<td>Parameters for this zone</td>
</tr>
<tr>
<td>A</td>
<td>IPv4 address of a host</td>
<td>32-Bit integer</td>
</tr>
<tr>
<td>AAAA</td>
<td>IPv6 address of a host</td>
<td>128-Bit integer</td>
</tr>
<tr>
<td>MX</td>
<td>Mail exchange</td>
<td>Priority, domain willing to accept email</td>
</tr>
<tr>
<td>NS</td>
<td>Name server</td>
<td>Name of a server for this domain</td>
</tr>
<tr>
<td>CNAME</td>
<td>Canonical name</td>
<td>Domain name</td>
</tr>
<tr>
<td>PTR</td>
<td>Pointer</td>
<td>Alias for an IP address</td>
</tr>
<tr>
<td>SPF</td>
<td>Sender policy framework</td>
<td>Text encoding of mail sending policy</td>
</tr>
<tr>
<td>SRV</td>
<td>Service</td>
<td>Host that provides it</td>
</tr>
<tr>
<td>TXT</td>
<td>Text</td>
<td>Descriptive ASCII text</td>
</tr>
</tbody>
</table>
Most queries and responses are UDP datagrams

Two types of queries:

- **Recursive**: Nameserver responds with answer or error
  
  ![Recursive Query Diagram]
  
  Client → Nameserver: www.ucsd.edu?
  
  Answer: www.ucsd.edu A 132.239.180.101

- **Iterative**: Nameserver may respond with a referral
  
  ![Iterative Query Diagram]
  
  Client → Nameserver: www.ucsd.edu?
  
  Referral: .edu NS a.edu-servers.net
ITERATIVE LOOKUP

1: noise.cs.uchicago.edu
10: 128.135.24.19

2: query
3: edu
4: query
5: uchicago.edu
6: query
7: cs.uchicago.edu
8: query
9: 128.135.24.19

Originator: filts.cs.vu.nl
Local resolver (cs.vu.nl)
Root name server (a.root-servers.net)
Edu name server (a.edu-servers.net)
uchicago name server
uchicago cs name server
DNS CACHING

• Performing all these queries takes time
  • And all this before actual communication takes place
• Caching can greatly reduce overhead
  • The top-level servers very rarely change
    • Popular sites visited often
  • Local DNS server often has the information cached
• How DNS caching works
  • All DNS servers cache responses to queries
  • Responses include a time-to-live (TTL) field
    • Server deletes cached entry after TTL expires
JULIA EVAN’S GUIDE TO DIG

**dig** makes DNS queries!

$ dig google.com

google.com 208 IN A
ip address → 172.217.13.110

**dig** makes a reverse DNS query - find which domain resolves to an IP!

**dig** +trace domain

traces how your domain gets resolved, starting at the root nameservers

**dig** +short domain

Usually dig prints lots of output! With +short it just prints the IP address/value of the DNS record

**dig** @ 8.8.8.8 domain

dig @ server lets you pick which DNS server to query! Useful to check if your system DNS is misbehaving 😞

**dig** TYPE domain.com

this lets you choose which DNS record to query for!

types to try: SRV default

**dig** -x 172.217.13.174

Julia Evans @bozk
UC San Diego