CS 352
The Application Layer

Lecture 3.1, Spring 2020
http://www.cs.rutgers.edu/~sn624/352

Srinivas Narayana
Application-layer Protocol

- Types of messages exchanged,
  - e.g., request, response
- **Message format:**
  - Syntax: what fields in messages & how fields are delineated
  - Semantics: meaning of information in fields
- **Actions:** when and how processes send & respond to messages

Public-domain protocols:
- defined in RFCs
- allows for interoperability
- e.g., HTTP, SMTP

Proprietary protocols:
- e.g., Skype
Application Addresses

• We usually think of an application executing on a single endpoint
• However, applications can reside on, say, 2 different endpoints connected by a network
• In order to communicate, need to identify the communicating parties
  • Telephone network: phone number (10 digits)
• Computer network: IP address
  • IPv4 (32 bits) 128.6.24.78
  • IPv6 (128 bits) 2001:4000:A000:C000:6000:B001:412A:8000
• Suppose there is more than one networked program executing on a host
  • In addition to host address, we need one more address
  • “Which Program to talk to?”
• The identity for an application: port number (+ IP addr)
A **socket** is the door between OS/network and the application process.

The application’s **programming interface** to the network.
An app-layer connection is a 4-tuple

Connection := (IP_S, Port_S, IP_D, Port_D)
(S = source, D = destination)
App-layer connections

- A small demo
Recall: Apps rely on services by lower layers

Within the operating system of respective entity
Common Architectures of Applications
Client-server architecture

Server:
- always-on endpoint
- “permanent” IP address
- server farms ("data centers") for scaling

Clients:
- communicate with server
- may be intermittently connected
- may have dynamic IP addresses
- do not communicate directly with each other

- The web (HTTP) works this way.
- Many mobile apps work this way (e.g., Instagram)
Peer-to-peer (P2P) architecture

- Peers:
  - Intermittently connected hosts
  - Directly talking to each other

- Little to no reliance on always-up servers
  - Examples: BitTorrent, Skype

- Today, many applications use a hybrid model
  - Example: Skype “supernodes”
Going forward: A few applications

• Domain Name System

• The web: HTTP

• Mail

• File transfer
CS 352
Domain Name System

Lecture 3.2, Spring 2020
http://www.cs.rutgers.edu/~sn624/352

Srinivas Narayana
“You have my name. Can you lookup my address?”
Domain Name System (DNS)

• Problem statement:
  • Average brain can easily remember 7 digits for a few names
  • On average, IP addresses have 12 digits
  • We need an easier way to remember IP addresses

• Solution:
  • Use alphanumeric names to refer to hosts. Called host names or domain names
    • Example: cs.rutgers.edu
  • We need a directory: add a service to map between alphanumeric host names and binary IP addresses
  • We call this process Address Resolution
Types of Directories

• Directories map a name to an address

• Simplistic designs
  • Central directory
  • Ask everyone (e.g., flooding)
  • Tell everyone (e.g., push to a file like /etc/hosts)

• Scalable distributed designs
  • Hierarchical namespace (e.g., Domain Name System (DNS))
  • Flat name space (e.g., Distributed Hash Table)
Simple DNS

- What if every host has a local directory?
- `/etc/hosts.txt`
  - How things worked in the early days of the Internet!
- What if hosts moved around? How do you keep this up to date?
Simple DNS

Key idea: Implement a server that looks up a table.

Will this scale?

- Every new host needs to be entered in this table
- Performance: can the server serve billions of Internet users
- Failure: what if the server or the database crashes?
- How to secure this server?

<table>
<thead>
<tr>
<th>DOMAIN NAME</th>
<th>IP ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>cs.rutgers.edu</td>
<td>128.6.4.2</td>
</tr>
<tr>
<td><a href="http://www.google.com">www.google.com</a></td>
<td>74.125.225.243</td>
</tr>
<tr>
<td><a href="http://www.princeton.edu">www.princeton.edu</a></td>
<td>128.112.132.86</td>
</tr>
</tbody>
</table>

QUERY | STD QUERY | cs.rutgers.edu

RESPONSE | STD QUERY | 128.6.4.2

<Client IP, CPort, DNS server IP, 53>

<DNS server, 53, Client IP, Cport>
RFC 1034: Distribution through hierarchy enables scaling
DNS Protocol

- Client and Server
- Client connects to Port 53 on server
- Assume DNS server IP known
- Two types of messages
  - Queries
  - Responses
- Type of Query (OPCODE) methods
  - Standard query (0x0)
    - e.g., Request IP address for a given domain name
  - Updates (0x5)
    - Provide a binding of IP address to domain name
- Each type has a common message format that follows the header
DNS Protocol

- When client wants to know an IP address for a host name
  - Client sends a DNS query to the “local” name server in its network
  - If name server contains the mapping, it returns the IP address to the client
  - Otherwise, the name server forwards the request to the root name server
  - The request works its way down the tree toward the host until it reaches a name server with the correct mapping
Example

• Host at cis.poly.edu wants IP address for gaia.cs.umass.edu
• Local DNS server
• Root DNS server
• TLD DNS server
• Authoritative DNS server
Query type

Iterative query:
• Contacted server replies with name of server to contact
• “I don’t know this name, but ask this server”
• Queries are iterative for the local DNS server
Query type

Recursive query:

- Puts burden of name resolution on the contacted name server

Problem: think about the root DNS server.
- Must it answer every DNS query?
DNS in action

- A small demo
CS 352
DNS Records

Lecture 3.3, Spring 2020
http://www.cs.rutgers.edu/~sn624/352

Srinivas Narayana
DNS records

**DNS:** distributed database storing resource records (RR)

RR format: \((\text{name}, \text{type}, \text{class}, \text{ttl}, \text{addr})\)

- **Type=A**
  - \text{name} is hostname
  - \text{value} is IP address

- **Type=AAAA**
  - \text{name} is hostname
  - \text{value} is IPv6 address

- **Type=NS**
  - \text{name} is domain (e.g. foo.com)
  - \text{value} is hostname of authoritative name server for this domain

- **Type=CNAME**
  - \text{name} is alias name for some "canonical" (the real) name
    - e.g., www.ibm.com is really servereast.backup2.ibm.com
  - \text{value} is canonical name

- **Type=MX**
  - \text{value} is name of mailserver associated with \text{name}

More complete info at [https://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml](https://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml)
DNS Record example

<table>
<thead>
<tr>
<th>NAME</th>
<th>Design.cs.rutgers.edu</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYPE</td>
<td>A</td>
</tr>
<tr>
<td>CLASS</td>
<td>IN</td>
</tr>
<tr>
<td>TTL</td>
<td>1 day (86400)</td>
</tr>
<tr>
<td>ADDRESS</td>
<td>192.26.92.30</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NAME</th>
<th>Cs.rutgers.edu</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYPE</td>
<td>NS</td>
</tr>
<tr>
<td>CLASS</td>
<td>IN</td>
</tr>
<tr>
<td>TTL</td>
<td>1 day (86400)</td>
</tr>
<tr>
<td>NSDNAME</td>
<td>Ns-lcsr.rutgers.edu</td>
</tr>
</tbody>
</table>

DNS serves as a general repository of information for the Internet!
DNS record types

• A small demo
DNS caching and updating records

• Once (any) name server learns a name to IP address mapping, it *caches* the mapping

  • Cache entries timeout (disappear) after some time

  • TLD servers typically cached in local name servers

  • In practice, root name servers aren’t visited often
DNS protocol: *query* and *reply* messages, both with same *message format*

**Message header**
- **QR = 0** for Query, 1 for response
- **Opcode = 0** standard
- **identification**: 16 bit # for query, reply to query uses same #
- **flags**:
  - Authoritative answer
  - recursion desired
  - recursion available
  - reply is authoritative
DNS protocol, messages

<table>
<thead>
<tr>
<th>QR</th>
<th>OPCODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>identification</td>
<td>flags</td>
</tr>
<tr>
<td>number of questions</td>
<td>number of answer RRs</td>
</tr>
<tr>
<td>number of authority RRs</td>
<td>number of additional RRs</td>
</tr>
<tr>
<td>questions (variable number of questions)</td>
<td></td>
</tr>
<tr>
<td>answers (variable number of resource records)</td>
<td></td>
</tr>
<tr>
<td>authority (variable number of resource records)</td>
<td></td>
</tr>
<tr>
<td>additional information (variable number of resource records)</td>
<td></td>
</tr>
</tbody>
</table>

Name, type fields for a query

RRs in response to query

Records for authoritative servers

Information about nameserver

Additional “helpful” info that may be used
Bootstrapping DNS

• How does a host contact the name server if all it has is the domain name and no (name server) IP address?

• IP address of at least 1 nameserver (usually, a local resolver) must be known a priori

• The name server may be bootstrapped “statically”, e.g.,
  • File /etc/resolv.conf in unix
  • Start -> settings-> control panel-> network ->TCP/IP -> properties in windows

• … or with another protocol!
  • DHCP: Dynamic Host Configuration Protocol (more on this later)
Summary of DNS

• Hostname to IP address translation via a global network of servers

• Use Multiple layers of indirection
  • Hierarchically scale
  • Good performance (load distribution)
  • Resilient to local transient failure

• Additional load distribution can happen at each level (e.g., TLD server)

• Uses caching all over for better performance

• DNS can be used to implement useful primitives atop domain names:
  • Example: Scaling large web services, e.g., google search
  • Domain-authoritative server will return an address from a pool of IP addresses, for example from Google’s server “farm”
Some themes and observations on DNS

• Request/response nature of the protocol

• How messages are structured: simple, text-based protocol
  • Similarly in HTTP, SMTP, FTP

• Caching is an effective method to improve performance