Internet Technology

03. Application layer protocols

Paul Krzyzanowski

Rutgers University

Spring 2016

Today we'll examine

- DNS: Domain Name System
- HTTP: Hypertext Transfer Protocol
- FTP: File Transfer Protocol

Domain Name System

How are IP addresses assigned?

IP addresses are distributed hierarchically

- Internet Assigned Numbers Authority (IANA) at the top
 - IANA is currently run by ICANN
 - Internet Corporation for Assigned Names and Numbers



IANA

How are machine names assigned?

- Early ARPANET
 - Globally unique names for each machine (e.g., UCBVAX)
 - Kept track at the Network Information Center at the Stanford Research Institute (SRI NIC)
- That doesn't scale!
- A domain hierarchy was created in 1984 (RFC 920)
 - Domains are administrative entities: divide name management
 - Tree-structured global name space
 - Textual representation of domain names www.cs.rutgers.edu



Top Level Domains (TLDs)

ccTLD

Country-code domains ISO 3166 codes

e.g., .us, .de, .ca, .es

IDN ccTLD Internationalized country-code domains

e.g., السعودية. , .рф

gTLD Generic top-level domains

e.g., .biz, .com, .edu, .gov, .info, .net, .org

There are currently 1,239 top-level domains

Each top-level domain has an administrator assigned to it

Assignment is delegated to various organizations by the Internet Assigned Numbers Authority (IANA)

See http://www.iana.org/domains/root/db for the latest count

Shared registration

- **Domain name registry**: this is the database
 - Keeps track of all domain names registered in a top-level domain
- Domain name registry operator: this is the company that runs the db
 - NIC = Network Information Center organization that keeps track of the registration of domain names under a top-level domain
 - keeps the database of domain names
- **Domain name registrar**: this is the company you use to register
 - Company that lets you register a domain name

Shared registration

- Until 1999: Network Solutions Inc. operated the .com, .org, .net registries
- Now
 - Multiple domain registrars provide domain registration services
 - Around 1,000 of these companies each is accredited by the ICANN
 - 2,124 as of February 2016, including 701 unique DropCatch.com registrars
- The registrar you choose becomes the designated registrar for your domain
 Maximum period of registration for a domain name = 10 years
- The registry operator keeps the central registry database for the top-level domain
- Only the designated registrar can change information about domain names
 - A domain name owner may invoke a domain transfer process

Example

- Namecheap is the designated registrar for poopybrain.com
- VeriSign is the registry operator for the .com gTLD

See https://www.icann.org/registrar-reports/accredited-list.html for the latest list of registrars

The problem

Every device connected to the Internet has a unique Internet Protocol (IP) address

How do you resolve user-friendly machine names to IP addresses?

www.cs.rutgers.edu \longrightarrow 128.6.4.24

Original solution

Through the 1980s

- Search /etc/hosts file for machine name (see RFC 606)
- File periodically downloaded from Network Information Center (NIC) at the Stanford Research Institute (SRI)
- This was not sustainable with millions of hosts on the Internet
 - A lot of data
 - A lot of churn in the data
 - new hosts added, deleted, addresses changed
 - Maintenance
 - Traffic volume

Solution doesn't scale!

DNS: Domain Name System

- Distributed database
 - Hierarchy of name servers
- DNS is an application-layer protocol
 - Name-address resolution is handled at the edge
 - The network core is unaware of host names

DNS provides

- Name to IP address translation
- Aliasing of names (called canonical names)
- Identification of name servers
- Mail server names
- Load distribution:
 - Multiple name servers that can handle a query for a domain
 - Caching
 - Ability to provide a set of IP addresses for a name



Authoritative DNS server

- An authoritative name server is responsible for answering queries about its zone
 - Configured by the administrator
- Zone = group of machines under a node in the tree E.g., rutgers.edu

A DNS server returns answers to queries

Key data that a DNS server maintains (partial list)

Information	Abbreviation	Description
Host	A	Host address (name to address) Includes name, IP address, time-to-live (TTL)
Canonical name	CNAME	Name for an alias
Mail exchanger	MX	Host that handles email for the domain
Name server	NS	Identifies the name server for the zone: tell other servers that yours is the authority for info within the domain
Start of Zone Authority	SOA	Specifies authoritative server for the zone. Identifies the zone, time-to-live, and primary name server for the zone

Finding your way

- How do you find the DNS Server for rutgers.edu?
 - That's what the domain registry keeps track of
 - When you register a domain, you supply the addresses of at least two DNS servers that can answer queries for your zone
- So how do you find it?
 - Start at the root

Root name servers

- The root name server answers can return a list of authoritative name servers for top-level domains
- 13 root name servers
 - A.ROOT-SERVERS.NET, B.ROOT-SERVERS.NET, ...
 - Each has redundancy (via anycast routing or load balancing)



Download the latest list at http://www.internic.net/domain/named.root

DNS Queries

- Iterative (non-recursive) name resolution
 - DNS server will return a definitive answer or a referral to another DNS server
 - *referral* = reference to a DNS server for a lower level of the queried namespace
 - Server returns intermediate results to the client
 - 1. Send query to a root name server
 - 2. Send query to a edu name server
 - 3. Send query to a rutgers name server
 - Advantage: stateless
- Recursive DNS name resolution
 - Name server will take on the responsibility of fully resolving the name
 - May query multiple other DNS servers on your behalf
 - DNS server cannot refer the client to a different server
 - Disadvantage: name server has more work; has to keep track of state
 - Advantages: Caching opportunities, less work for the client!

Most top-level DNS servers only support iterative queries

DNS Resolvers: local name server

DNS Resolver

- Not really a part of the DNS hierarchy
- Acts as an intermediary between programs that need to resolve names and the name servers
- A resolver is responsible for performing the full resolution of the query
- Where are they?
 - Local system has one: that's what applications contact
 - Local cache; may be a process or a library
 - On Linux & Windows, these are limited DNS servers (called stub resolvers): they are not capable of handling referrals and expect to talk with a name server that can handle recursion (full resolution)
 - ISPs (and organizations) run them on behalf of their customers
 - Including a bunch of free ones (OpenDNS, Google Public DNS)
- Resolvers cache past lookups not responsible for zones

Using a DNS resolver

To look up a name:

- Send a DNS query to the local resolver (recursion requested)

Local resolver

- If the local resolver has cached results, it can return the answer
- Otherwise, consult a local hosts file (e.g., /etc/hosts) to return locallyconfigured name→address mappings
- Otherwise contact a DNS server that the client knows about this is typically another resolver that is provided by the ISP
 - The local system is configured with one or more addresses of external name servers

ISP Resolver

- Check cache
- Check a locally-configured zone file (if any). If the desired data is there, return an authoritative answer
- Otherwise, do an iterative set of queries to traverse the hierarchy to find the desired name server and get results

DNS Resolvers in action



Local stub resolver:

- check local cache
- check local hosts file
- send request to external resolver

E.g., on Linux: resolver is configured via the /etc/resolv.conf file

External resolver

- DNS server that accepts recursion
- Running at ISP, Google Public DNS, OpenDNS, etc.

Sample query

- Rutgers registered rutgers.edu with the .edu domain
 - educause.net is the domain registry for the .edu gTLD

 The root name server contains addresses for the name servers of all the top-level domains

 The local name server is provided the list of addresses of root name servers

Sample Query

Submit query to a local DNS resolver:

- Send query(cs.rutgers.edu) → root name server root name servers identify authoritative servers for top-level domains send query to c.root-servers.net: 192.33.4.12
- 2. Receive referral to a list of DNS servers for edu a.edu-servers.net: 192.5.6.30 g.edu-servers.net: 192.42.93.30
- 3. Send query(cs.rutgers.edu) → edu name server send query to g.edu-servers.net: 192.41.162.32
- 4. Receive referral to rutgers.edu name servers:
 - ns87.a0.incapsecuredns.net 192.230.121.86
 - ns8.a1.incapsecuredns.net. 192.230.122.7
 - ns124.a2.incapsecuredns.net 192.230.123.123
- query(cs.rutgers.edu) → rutgers name server send query to 192.230.122.7
- 6. The rutgers name server returns

A: 128.6.4.2 *address* MX: dragon.rutgers.edu *domain name for email*

Caching

- Starting every query at the root would place a huge load on root name servers
- A name server can be configured to cache results of previous queries
 - Save query results for a *time-to-live* amount of time
 - The time-to-live value is specified in the domain name record by an authoritative name server
- Caching name servers are recursive name servers

The DNS Query Protocol

DNS Records

- DNS servers store resource records (RRs)
- Format
 - Name, value, type of record, TTL (time to live)
- Common types
 - Address: A
 - Name: hostname
 - Value: IP address
 - Name Server: NS
 - Name: domain (rutgers.edu)
 - Value: hostname of authoritative name server for the domain

- Canonical name: CNAME
 - Name: alias hostname
 - Value: real hostname
- Mail Exchanger: MX
 - Name: hostname
 - Value: mail server for hostname

DNS Protocol

- DNS is a service that listens to requests on TCP or UDP port 53
- Protocol consists of query and reply messages
 - Both messages have the same format and header



DNS Protocol

- DNS is a service that listens to requests on TCP or UDP port 53
- Protocol consists of *query* and *reply* messages
 - Both messages have the same format and header



DNS Queries

- Questions field contains a sequence or DNS queries
- Query name
 - Encoded form of the name for which we want an address
- Query type
 - 1 = IP address, 2 = name server, 0x0f = mail server, ...
- Query class
 - 1 = Internet addresses, 2 = CSNET

QNAME (variable)		
QTYPE	QCLASS	
(16 bits)	(16 bits)	

Reverse DNS

- What if we have an IP address and want the name?
- Special domain for reverse lookups
 - in-addr.arpa
 - ARPA = Address & Routing Parameter Area, not Advanced Research Projects Agency (e.g., ARPANET)

www.cs.rutgers.edu \rightarrow 128.6.4.24

 $24.4.6.128.in\text{-}addr.arpa \rightarrow www.cs.rutgers.edu$

Setting up reverse DNS

- Different query path than regular DNS queries
- On a DNS server
 - Configure PTR (pointer) records that map IP addresses to names
- Let the world find out
 - ISP allocated IP addresses to you
 - You tell the ISP what DNS servers are responsible for reverse DNS entries
- Example query path
 - DNS resolver contacts root servers
 - Root server refers to ARIN (North American IP registry) RDNS server
 - ARIN refers to local ISP RDNS server, which refers to your server

Root server \rightarrow RIR (e.g., ARIN) DNS server \rightarrow ISP DNS server

Web and HTTP

HTTP Basics

- HTTP: Hypertext Transfer Protocol (RFC 2616)
 - Web's application-layer protocol
 - Client-server model
 - TCP-based protocol
 - Client connects to port 80 on the server
 - HTTP messages are exchanged
 - Client closes the connection
- HTTP is stateless
 - Server does not store state on previous requests
 - Simplifies design
 - Easier failure recovery
 - Simplifies load balancing



URLs

- Requests for objects are URLs
- URL = Uniform Resource Locator



http://box.pk.org:8080/secret/demo/mystuff.html

Types of connections

- Non-persistent HTTP (HTTP 1.0)
 - At most one object is sent over a TCP connection
 - Request/response
- Persistent HTTP (HTTP 1.1)
 - Multiple objects can be sent over a single connection
Non-persistent HTTP

- www.pk.org/index.html is one file that references:
 - Five CSS (cascading style sheet) files
 - Four image files



Non-persistent HTTP: Response time

- Round-trip time (RTT)
 - Time for a small packet to travel from the client to the server & back to the client
- Response time
 - One RTT to initiate the connection
 - One RTT for request & start of response
 - File transmission time
- Total time =

objects x (2xRTT + transit_time)



Persistent HTTP: Response time

- Server leaves connection open after sending response
 - Subsequent HTTP messages are sent over the same open connection
 - One RTT for each referenced object once the connection is set up
- Response time
 - One RTT to initiate the connection
 - One RTT for request & start of response per object
 - File transmission time per object
- Total time_{persistent} = RTT + # objects × (RTT + transit_time)
- Versus Total time_{non-persistent} = # objects × (2×RTT + transit_time)



HTTP Request Message

- Two classes of messages: request & response
- HTTP request messages are human-readable ASCII text

Browser request for a URL (Uniform Resource Locator):

http://box.pk.org:12345/this/is/a/test.html

Creates an HTTP request Request line: GET, POST, HEAD, ... commands

GET /this/is/a/test.html HTTP/1.1 Host: box.pk.org:12345 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_2) AppleWebKit/536.26.17 (KHTML, like Gecko) Version/6.0.2 Safari/536.26.17 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8 Accept-Language: en-us Accept-Encoding: gzip, deflate Connection: keep-alive Carriage return, line feed (0x0d, 0x0a) indicates end of header ... and end of message in this case

headers

HTTP Response Message



Uploading form input

- HTTP POST method
 - Web pages may include form input
 - Input is uploaded to the server in the body of the request
- URL method
 - Parameter/value pairs are encoded in the URL (query string)
 - HTTP GET request is sent
 - Format
 - http://server/path/page?query_string
 - query_string is of the form item₀=value₀&item₁=value₁...

HTTP Methods

HTTP/1.0

• GET

- Request a resource

• POST

 Send data in the request message's body to the server

• HEAD

 Like GET, but only send the headers

HTTP/1.1

• GET, POST, HEAD

• PUT

 Uploads file to the path specified in the URL field

• DELETE

Deletes the file specified in the URL field

Some HTTP response codes

- 200 OK
 - Request succeeded; requested object is in the message
- 301 Moved Permanently
 - Requested object moved; new location specified in a Location: header in the list of headers
- 400 Bad Request
 - The server could not understand the request
- 404 Not Found
 - The requested content is not found on the server
- 505 HTTP Version Not Supported
 - Unsupported version

Try it out yourself

Talk to a server

Listen to a client

- Run
 - telnet cnn.com 80
- Type in a basic GET request
 - GET /index.html HTTP/1.1
 - Followed by an blank line
- Look at the response

- Run demo TCP server
 - java TCPServer
- Start a browser and connect to it:
 - http://localhost:12345/a/b/c
 - The server will print all the data it gets from the client

Keeping state: cookies

- HTTP is stateless
- Cookies provide a mechanism for web servers to store state
- Four parts to cookies:
 - 1. Cookie header line in the HTTP response message
 - 2. Cookie header line in subsequent HTTP request messages
 - 3. Cookie file stored on user's host & managed by browser
 - 4. Back-end database at the web server host
- Example
 - You visit an e-commerce site
 - When the site receives your request, it creates a unique ID and an entry in the database identified by that ID.
 - The HTTP response tells your browser to set a cookie. The cookie is sent with future messages to that server

Cookies in use



Maintaining state with cookies

- Cookies can help a server store & access
 - Shopping cart info
 - Login name, authorization credentials
 - Preferences (e.g., town name for weather)
 - Session state (e.g., web-based email)
 - History of web pages you visited on the site
- First-party cookies
 - Placed by the website you visit
- Third-party cookies
 - Placed by sites other than the one you visit mostly ads



Caching example



Caching example: improve access link

- Assume
 - Access link is now 100 Mbps
- Consequences
 - Utilization on LAN = 1.5%
 - Utilization on access link = 15%
 - Total delay =

Internet delay + access delay + LAN delay

 $= 2 \sec + m \sec + m \sec$

 But increasing the access link can be a costly upgrade



Caching example: add a caching proxy

- Assume
 - Access link remains at 15 Mbps
 - Install a caching proxy
 - Assume hit rate is 0.4 (40% hits)
- Consequences
 - 40% of requests satisfied by proxy (quick – e.g., 10 ms)
 - 60% have to go to outside servers
 - Use of access link reduced to 60%
 - Total average delay =

Internet delay + access delay + LAN delay

 $= 0.6^{*}(2.01 \text{ s}) + 0.4^{*}(10 \text{ ms}) + \text{puny ms}$ = < ~1.4 seconds



HTTP control for caching

Conditional GET

- Request an object BUT don't send it if the cache has an up-to-date version

HTTP Request

GET /index.html HTTP/1.1 Host: box.pk.org



HTTP control for caching

Next time you request the file, include two headers in your request

If-Modified-Since: time from Last-Modified

If-None-Match: value from Etag

HTTP Request

GET /index.html HTTP/1.1 Host: box.pk.org If-Modified-Since: Thu, 31 Jan 2013 01:18:12 GMT If-None-Match: "3c0549-17df-4d48b667f3d00"

HTTP Response

HTTP/1.1 304 Not Modified ←

Date: Mon, 11 Feb 2013 21:11:32 GMT Server: Apache/2.2.22 (Ubuntu) ETag: "3c0549-17df-4d48b667f3d00" Vary: Accept-Encoding This means the file was not modified since the cached copy.

- alternatively -

If the content has been modified at the server, then the content is sent as with a normal GET request.

Conditional GET

- Request a file from a server because it's not in your cache
 - Receive the file
 - Headers contain: Last-Modified and Etag
 - For caching, store both of those along with the file
- Next time you request the file, include two headers in your request
 - If-Modified-Since: <time from Last-Modified>
 - If-None-Match: <value from Etag>
- If the file has changed since you last requested it, the server will send back the new file. If not, the server will respond with a "304 Not Modified" code

More Optimizations

- Problem: Head-of-line blocking
 - One large (or slow) HTTP request can hold up all other requests from that client
- HTTP/1.x: Parallel connections
 - Open multiple TCP connections to the server
 - But:
 - Hard to deploy with proxies
 - Each connection takes time to open
 - Can use up a lot of connections extra server memory
 - Parallel connections typically limited to a small number (e.g., 4)
 - Can still lead to head-of-line blocking per connection
- HTTP/1.x: Pipelining
 - Send multiple HTTP requests without waiting for a response from each one
 - But:
 - The server still must send responses in the order requests were sent
 - Requests may be received quicker by the server but responses are still at risk of head-of-line blocking
 - Not supported or turned on in most browsers and proxies

More Optimizations

• HTTP/2 – Multiplexing

- Multiple request & response messages can be in flight at the same time
- Messages can be intermingled on one connection
- "Minification"
 - Reduce unnecessary characters form JavaScript & CSS
 - Merge multiple script files into one compressed file
- HTTP/2 header compression
 - Each HTTP header uses ~1400 bytes takes 7-8 round trips to move them to the client
- HTTP/2 server push
 - Server can push content give the client more than what it requested
 - Why send more data?
 - The browser has to get the first response, parse it, and make requests
 - But ... the server knows what a browser will need to render a web page
 - It can send the data before it's requested by the client

FTP: File Transfer Protocol

FTP: File Transport Protocol

- Transfer files between computers
- Client/server model
- Client: accepts commands from the user and initiates requests to get or put files on the server
- Defined in RFC 959
 - Original version RFC 765 June 1980
 - First proposal dates back to 1971



Separate data & control connections

- Client connects to an FTP server on TCP port 21
 - This is the command channel
 - Client port = some port \ge 1024 = N
- Commands are user requests and include authentication info
- When the server receives a command to transfer data, it initiates a TCP connection to the client on port *N*+1 from its local data port (20)
- After transferring one file, the server closes the data connection



Separation between control & data channels

Out of band control connection

Sample FTP Commands

- Sent as ASCII text over the control channel
- Access commands
 - USER: identify yourself
 - PASS: supply your password
 - CWD (CD): change working directory
 - CDUP (CD ..): change to parent
 - QUIT: log out
- Control commands
 - RETR (GET): retrieve a file
 - STORE (PUT): store a file
 - APPEND: append to a file
 - DELETE: delete a file
 - LIST (DIR): list files

- Error messages
 - Similar to HTTP:
 - Status code & text
 - 331 User name okay, need password.
 - 200 Command okay.
 - 230 User logged in, proceed.
 - 502 Command not implemented.
 - 125 Data connection already open; transfer starting.

Active vs. Passive FTP

- Not all clients can receive incoming connections
 - This was a pain with firewalls and NAT (network address translation)
- Passive mode FTP
 - Client initiates both connections to the server
 - The first connection (for commands) contacts the server on port 21
 - Originating port = $N, N \ge 1024$
 - Then the client then issues a PASV command
 - The server opens a random port $P \ge 1024$
 - Sends back the value *P* to the client as a response
 - The client then connects from port *N*+1 to port *P*
- Most browsers support only passive mode FTP

The end