# EE 122 Final Note Sheet

## Domain Name System (DNS)

### Internet Names & Addresses

- Machine Addresses: e.g., 169.229.131.101
  - reouter-usable labels for machines
  - conforms to network structure (the "where")
- Machine Names: e.g., instr.eecs.berkeley.edu
  - human-usable labels for machines
  - conforms to organizational structure ("the who")
- The Domain Name System (DNS) is how we map from one to the other.

### Goals of DNS

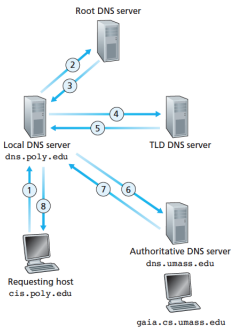
- No naming conflicts (uniqueness)
- Scalable (many names and frequent updates)
- Distributed, autonomous administration
  - Ability to update my own machines names
  - Don't have to track all updates
- Highly available
- Lookups are fast

### DNS Hierarchy

Made up of three intertwined hierarchies

- Hierarchical namespace, as opposed to the original flat namespace.
- Hierarchically administered, as opposed to centralized
- (Distributed) Hierarchy of servers, as opposed to centralized storage.

### Domain Resolution




- Recursive query, asks the server to get you the answer.
- Iterative quere, asks the server to give you the next server to check.

### DNS Records

DNS info is stored as resource records (RRs)

- Address: name = hostname, value = IP Address
- Name Server: name = domain, value = name of dns server for domain
- Canonical NAME: name = hostname, value = canonical name
- Mail eXchanger: name = domain in email address, value = canonical name of mail server

### DNS Protocol

Client-Server interaction is on UDP Port 53

### DNS Caching

- DNS servers cache responses to queries
- Responses include a TTL field and the server deletes the cached entry after the TTL expires
- Negative Caching: misspellings like cnn.comm take a long time to fail the first time. Some servers implement a cache of mistakes, but it is optional and not widely implemented.

## Hyper Text Transfer Protocol (HTTP)

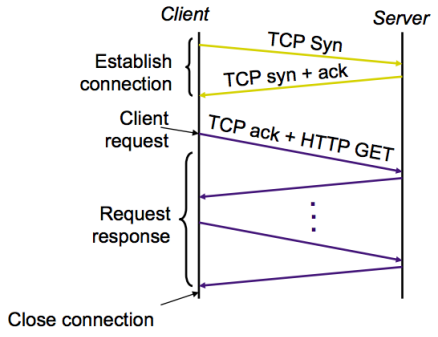
### Uniform Record Locator (URL)

Web content is named using URLs, URLs use DNS hostnames

### HTTProtocol

- Stateless
  - Each request-response is treated independently, servers do not retain state
  - Good: This improves server-side scalability, failure handling
  - Bad: Some application need persistent state (ex, shopping carts, user profiles, ect.)
- HTTP runs over TCP on port 80
- Synchronous request/reply protocol

### Steps in an HTTP Request/Response




### Client-to-Server Communication

- HTTP Request Message
  - Request line: method, resource, and protocol version
  - Request headers: provid information or modify request
  - Body: optional data (ex, "POST" data to the server)
- HTTP Response Message
  - Status line: Protocol version, status code, status phrase
  - Response headers: provide information
  - Body: optional data

### HTTP Performance

Most web pages have multiple objects. Naively, this would be one tcp connection per (possibly small) object.

- Concurrent Requests and Responses
  - Uses multiple connects in parallel
  - Doesn't necessarily maintain the order of responses.
- Persistent Connections
  - Maintain TCP connection across multiple requests. (including transfers from the current page.) Either the cliet or the server can tear down the connection.
  - This avoids overhead of connection set-up and tear-down, creates a more accurate RTT estimate, allows TCP congestion window to increase. (Thereby being able to use previosly discovered bandwidth.)
  - default in HTTP/1.1
- Pipelined Requests
  - Batch requests and responses to reduce the number of packets
  - Multiple requests can be contained in one TCP segment
- Getting n Large Objects

| Connection Style | latency | bandwidth |
|---|---|---|
| One-at-a-time | 2n RTT | nF/B |
| M concurrent | [n/m] RTT | [n/m] F/B |
| Persistent | (n+1) RTT | nF/B |
| Pipelined | 2 RTT | nF/B |
| Pipelined/Persistent | 2 RTT then RTT later | nF/B |

- Caching
  - Adds the modifier, "If-modified-since", os if the resource hasn't changed, it simply returns "not modified"
  - The response header has field, "Expires" that tells it how long it is safe to cache and "No-cache" which will ignore caches and always get the resource.
  - Client Caching: on your computer.

  - Forward Proxies: Done by ISPs to reduce network traffic and lower latency
  - Reverse Proxies: Done close to the server to reduce server load. Usually done by the content provider.
- Replication
  - Replicates popular web sites across multiple machines. This spreads load on servers, places content closer to clients, and helps when content isn't cacheable. To account for locationality, DNS returns different addresses based on client's geo location, server load, ect.

## Physical Layer

### Properties of a Desired Link

- Fast (High Rate)
- Reliable: Very low probability of undetected error

### Inside the Link Layer

- MAC - Medium Access Control: manages many different connections talking simultaneously. (How to share limited space)
  - Time Division Multiplexing - taking turns in the Time Domain
  - Frequency Division Multiplexing - takes turns in Frequency Domain
  - Spacial Division Multiplexing - Chooses a particular direction to listen to signals (cell phones)
  - Aloha Protocol - If you have data to transmit, send it. 2/3rds of bandwidth is lost, but it turns out to be bettern than making everyone wait for a turn to send data.
- PHY - "Physical Layer", gets bits from point A to point B. This operates on 3 levels,
  - Bits - dealing with individual bits of data
    Bits are packaged into frames, with codes for error detection and correction
  - Signals - operates on the signal level (Discrete Time Sampling)
    Modulation, translates the bits into analog values and vice versa
    Codes, Hash functions used to verify frames, add redundancy (Reid Soloman)
  - Hardware - Antennas/ Laser Diodes
    Timers/Clocks, Mixing/Demixing, Amplification, Filtering, Antennas

### Raw Resource(s) in Wireless

In Wireless (802.11n)

- W, Bandwidth (MHz) typically 20MHz for wifi
- k, Number of antennas
- Signal to Noise Ratio (SNR)
- Acheivable Bits/second $= k * W \log_2(1 + SNR)$

---