Network Applications

Principals of Network Applications

- To create a **network application**, we need to write a program that runs on **different end systems**, and **communicates over a network**.
- There is a **layer of abstraction** between **network applications** and **the network**; allowing for **rapid network application development**.
- There are different **application architectures** we can use to developer **network applications**.

Client-Server Architecture

- The **client-server architecture** consists of two entities, the **client** and the **server**.
- The **sever** is a **network host that is always on**, and has a **permanent IP address**.
- The **client** communicate with the **server** over a **network**, and can have a **dynamic IP address**.
- The clients do not **directly communicate**, they use the **server to communicate**.

Peer to Peer Architecture

- The **peer to peer architecture** has **no always-on server**.
- The **clients communicate directly**.

Process Communication

- A **process** is a **program running on a network host**.
- **Processes** in the **same host** use **inter-process communication** to communicate.
- **Processes** in **different hosts** use **network-communication** to communicate.
- The **server process** is a process that **waits to be contacted by clients**.
- The **client process** is a process that **initiates communication with the server**.

Sockets

- One way **two process** can **connect over a network** is with **sockets**.
- A **socket** is a **structure within a network host** that **serves as an endpoint** for **sending and receiving data**.
- In order for **network hosts to communicate**, they must each have a **unique Internet Protocol Address (IP Address)**.
- The **operating system** uses the **ports** of the **server and client** to make sure the information ends up in the **correct place**.

Application Layer Protocols

- **Application Layer Protocols** define how **application processes** running on **different end systems communicate**.
- The **protocols** define **the message syntax, semantics, and rules**.
- The **type of transport** an **application** uses depends on what is **important to the application**; such as **data integrity, throughput, timing, etc**.

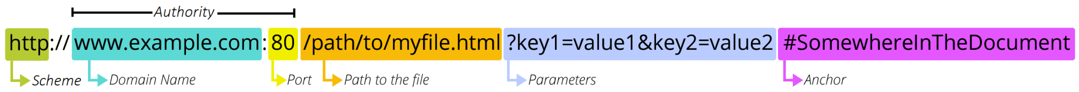
Internet Transport Services

- There are **two internet transport protocol services**:
  1. **User Datagram Protocol (UDP)** is a **transport protocol** that is **fast but unreliable**. UDP has **no confirmation** that the **packets were delivered**; upd is **not connection-oriented**.
  2. **Transport Control Protocol (TCP)** is a **transport protocol** that is **reliable, but has more overhead**. TCP has **flow control, congestion control, and is connection oriented**.
- **TCP and UDP** have **no encryption** by default.
- To **secure data** being transferred with **TCP** we use a **Secure Socket Layer (SSL)** which provides **encryption, data integrity, and end-point authentication**.
  - **SSL** is a part of the **application-layer**.

The World Wide Web

Introduction

- The **Word Wide Web or Web** is the world's dominant **software platform**.
- The **web** is an information space where **documents and other resources** can be **accessed through the internet** using a **web browser**.
- A **web page** is a **hypertext document** that is delivered by a **web server**.
- A **website** consists of **many webpages** linked together under a common **host**.
- **Web resources** can be accessed through a **Uniform Resource Locator (URL)**.


Authority
http://www.example.com:80/path/to/myfile.html?key1=value1&key2=value2#SomewhereInTheDocument
Scheme
Domain Name
Port
Path to the file
Parameters
Anchor

Hypertext Transfer Protocol

- The **web** uses the **Hypertext Transfer Protocol (HTTP) suite** to transfer data over the **internet**.
- **HTTP** uses **TCP** to facilitate the actual data transfer as follows:
  1. The **client** initiates the **TCP** connection with the **server** (typically on port 80).
  2. The **server** accepts the **TCP** connection from the **client**.
  3. **HTTP messages** are exchanged between the **browser** and the **server**.
  4. The **TCP** connection is **closed**.
- **HTTP** is **stateless**.
- There are **two** types of **HTTP connections**:
  1. **Persistent HTTP** is a connection where multiple files can be sent over a **single TCP connection** between the **client** and the **server**.
  2. **Non-Persistent HTTP** is where each file requires a **separate TCP connection** between the **client** and the **server**.

Hypertext Transfer Protocol Versions

- **HTTP 1.0** key features:
    1. The concept of headers and request methods were introduced.
    2. Version information is include in requests.
    3. It allowed for a single request / response for every TCP connection.
    4. Status codes were introduced.
    5. The content-type header made it possible to send different file types.
- **HTTP 1.1** key features:
    1. Allows for multiple requests with a single TCP connection via the keep-alive header.
    2. The upgrade header was introduced to allow the server and the client to switch communication protocols.
    3. Support for chunk transfers was introduced allowing for streaming content dynamically.
- **HTTP 2.0** key features:
    1. The protocol has switched to a binary protocol (no more plain text requests).
    2. Introduced push servers, allowing the server to push common resources before the client requests them.
    3. Introduced multiplexing interleaving the requests and responses without head-of-line blocking over a single TCP connection.

Hypertext Transfer Protocol Requests

- **HTTP requests** are the **messages** used to communicate over the **HTTP protocol**.
- There are **two main parts** of the request:
    1. **The Header** is the field of an **HTTP request or response** that passes **additional context and metadata** about the request.
    2. **The Body** is the field of an **HTTP request or response** that passes the **target data**.
- There are two **8 types of HTTP request methods**:
    1. The **GET method** is used to **retrieve information** from the server.
    2. The **HEAD method** is used to **retrieve information** from the server, but it transfers the status line and the header only.
    3. The **POST method** is used to **send data** to the server.
    4. The **PUT method** is used to **replace data** on the server.
    5. The **DELETE method** is used to **delete data** on the server.
    6. The **CONNECT method** is used to **establish a tunnel to the server**.
    7. The **OPTIONS method** is used to **describe the communication options** for the target resource.
    8. The **TRACE method** is used to **perform a message loop back test**.

Maintaining State over HTTP

- Since **HTTP requests have no state**, we use **cookies**.
- **Cookies** are **key-value pairs** that are **sent back-and-forth with each request** (similar to headers).

Web Cache (Proxy Servers)

- ISPs will use **cache proxy servers** to serve **cached data**, to lessen the load on the **origin server**.
- If the data is **not found on the cache proxy server**, the request will be **forwarded to the origin server**.
- **Cache servers** can reduce the response time for requests, and reduce the traffic on access links.
- A **conditional GET** request is a request that uses the **cached resource** if it is **up-to-date**. This uses the **If-modified-since** header.

Electronic Main

E-Mail Components

- There are **three components** to e-mail:
  1. **User Agents** - Software that **acts on behalf of a user** (composing, editing, sending, and displaying email messages).
  2. **Mail Servers** - Software that **transfers electronic mail**.
  3. **Simple Mail Transfer Protocol (SMTP) -** An **internet standard** for communicating **electronic mail**.
- A **mailbox** is a container that **store incoming messages** for the user.
- The **message queue** is a queue of **outgoing messages**.
- **SMTP** facilitates the **transfer of messages**.

Simple Mail Transfer Protocol (SMTP)

- **Simple Mail Transfer Protocol** uses the **TCP protocol** to reliably **transfer email messages** from the client to the server.
- **Unencrypted mail** typically uses **port 25**.
- There are **three phases of transfer**:
  1. The handshake phase.
  2. The message transfer phase.
  3. The closure phase.s
- **Commands** use ASCII text, and **responses** use status codes and phrases.
- **Messages** must be in **7-bit ASCII**.
- **Messages** have a **header** and a **body**.

Mail Access Protocols

- **Simple Mail Transfer Protocol (SMTP)** is a protocol that allows email clients to **deliver and store** messages on a **receiver's** server.
- **Internet Mail Access Protocol (IMAP)** is a protocol that allows email clients to **retrieve, delete, and store** messages on a mail server.
- **Hypertext Transfer Protocol (HTTP)** can be used to create a **web-interface** on top of **SMTP and IMAP**.

Domain Name System

Domain Name System

- The **Domain Name System (DNS)** is a **hierarchial** and **decentralized naming system** that is used to **identify computers** that are reachable over the **internet**.
- A **name server** is a single **server component** of the **domain name system**.
- The **DNS** allows computers to translate **domains** (ex google.com) to **IP addresses**.

DNS Root Name Servers

- **DNS root servers** are official, contact-of-last-resort servers that **cannot resolve names**. They are managed by the **Internet Corporation for Assigned Names and Numbers (ICANN)**.
- **DNS root servers** can direct requests to **Top-Level Domain (TLD) servers**.
- The internet **would not work** without **root servers**.
- **DNSSEC** is used to provide **DNS security** (authentication and message integrity).

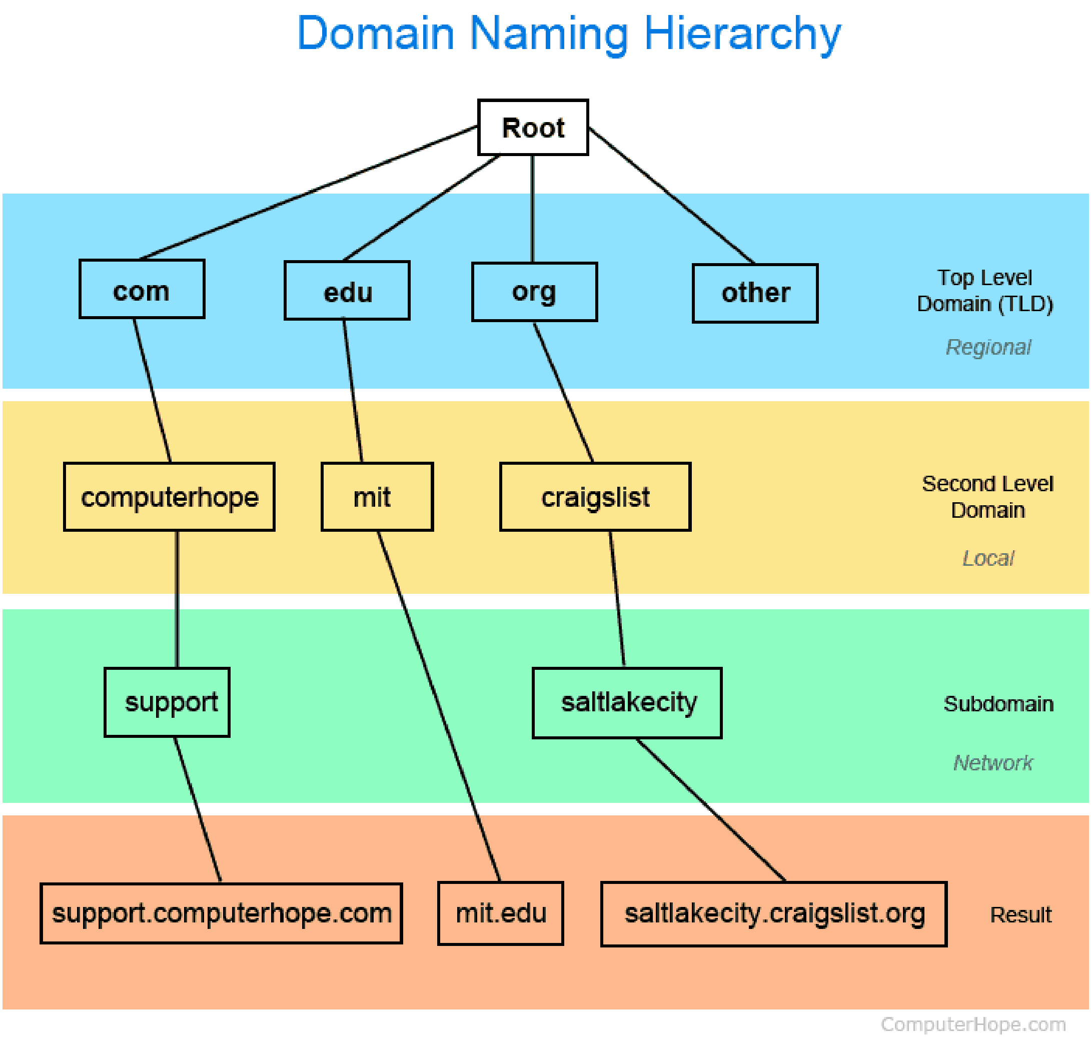
Top-Level Domain Servers

- **Top-Level Domain servers** are servers that are responsible for .com, .org, .net, .ca, etc.
- **Authoritative DNS servers** are **DNS servers** that are owned by **organizations**, and provide **authoritative hostname to IP mappings** for organizations named hosts.

Local DNS Servers

- **Local DNS servers** are **DNS servers** that are owned and operated by **internet service provides, companies, and universities**.
- When you make a **DNS request**, it goes to your **local DNS server**, if your **local DNS server does not have the mapping**, it will act as a proxy up the **DNS** hierarchy.

DNS Hierarchy



Types of DNS Name Resolution

- There are **two types** of **DNS name resolution**:

  1. **Iterated query** is where the **local DNS server contacts other DNS servers**, and **if the server cannot resolve the domain** the other DNS server **redirect the local DNS server**. This process continues until the **domain is resolved, or it is discovered that the domain does not exist**.
  2. **Recursive query** is where the **local DNS server contacts one DNS server**, if the DNS server **cannot resolve the domain, it will forward the request**. This process continues until the **domain is resolved, or it is discovered that the domain does not exist**.

- **Iterated queries** put a **heavy-load** on **local DNS servers**.

- **Recursive queries** put a **heavy-load** on **non-local DNS servers**.

- With **recursive queries** each time a new **domain is resolved** it can be **cached** on **all of the servers** that **forwarded the request**, whereas with **iterative queries** it will **only be cached on the local DNS server**.

DNS Caching and Updating

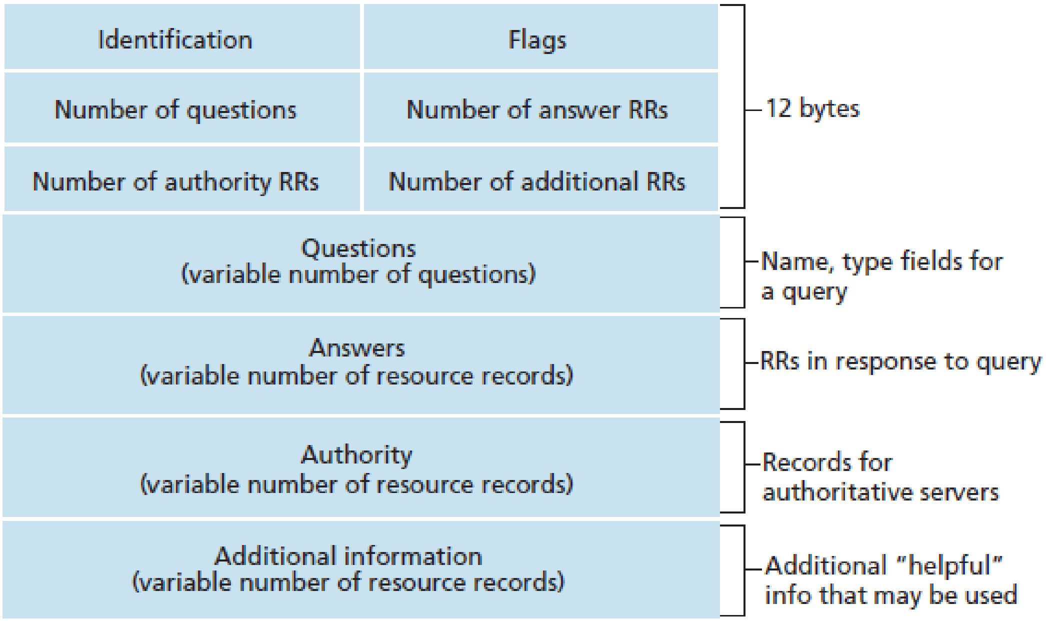
- One a **name server** learns a **mapping**, it **caches the mapping**.
- Each **cached entry** has a **timeout period (Time To Live — TTL)**. The TTL can be defined when setting up the domain.
- **Cached entries** may be **out-of-date** since they are only updated when the mapping **times out**.
- The delay it takes for of the **DNS servers** to update their **DNS mapping** is known as **propagation delay**.

Types of DNS Records

- **DNS records (RR records)** are stored with the following format: **(name, value, type, ttl)**.
- An **A Record** is an **IPv4 address mapping record (or DNS host record)**. It stores a **hostname** and it's corresponding **IPv4 address**.
- An **AAAA Record** is an **IPv6 address mapping record (or DNS host record)**. It stores a **hostname** and it's corresponding **IPv6 address**.
- A **Canonical Name Record (CNAME Record)** is a record to **alias hostnames**. It stores a **from hostname** and a **to hostname**.
- A **Mail Exchanger Record (MX Record)** is a record to specify a **SMTP email server** for the domain. It is used to **route outgoing** emails to an **email server**.
- A **Name Server Record (NS Record)** is a record that indicates a **DNS Zone** is **delegated to an Authoritative Name Server**, and provides the **address of the name server**.
- A **Reverse-Lookup Pointer Record (PTR Record)** allows a **DNS resolver** to provides an **IP address** and **receives a hostname** (Reverse DNS lookup).
- A **Certificate Record (CERT Record)** is a record that stores encryption certificates (PKIX, SPKI, PGP, etc).
- A **Service Location Record (SRV Record)** is a record that **specifies a service location** (similar to MX Records, but for other communication protocols).
- A **Text Record (TXT Record)** is a record that **carries machine-readable data** (DKIM, DMARC, sender policy, etc).
- A **State of Authority Record (SOA Record)** is a record that **appears at the beginning of a DNS zone file**. It is used to indicate the **authoritative name server** for the current **DNS zone**. Contact information for the domain administrator, and other meta-data is stored here.

DNS Protocol Messages

- **DNS query and reply** messages **have the same format**:


Identification
Flags
Number of questions
Number of answer RRs
Number of authority RRs
Number of additional RRs
12 bytes
Questions
(variable number of questions)
Name, type fields for a query
Answers
(variable number of resource records)
RRs in response to query
Authority
(variable number of resource records)
Records for authoritative servers
Additional information
(variable number of resource records)
Additional "helpful" info that may be used

- **Identification** is a **16-bit integer**, the **reply will respond with the same number**.
- **Flags** are: query or reply, recursion desired, recursion available, reply is authoritative.