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**.

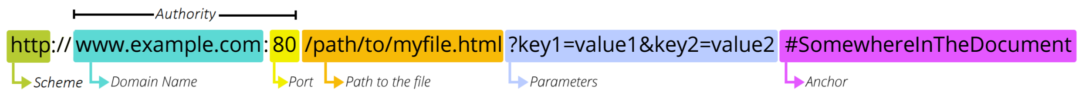
Internet Transport Services

- 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 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.