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**.