The Application Layer

Michael Brodskiy
Professor: E. Bernal Mor

September 18, 2023

- Network Applications
  - Social networking
  - Web
  - Text messaging
  - E-mail
  - Multi-user network games
  - Streaming stored video (YouTube, Hulu, Netflix)
  - P2P File Sharing
  - And many more
- Creating Network Applications
  - Write programs that:
    - Run on (different) end systems
    - Communicate over network
    - For example, web server software communicates with browser software
  - No need to write software for network-core devices (intermediate nodes)
    - Network-core devices do not run user applications
    - Applications on end systems allow for rapid application development and propagation
- Application Architecture
  - Network architecture — a set of layers and protocols
    - It is fixed, and provides the network application developer with specific set of services

  - Application Architecture — define how the application is structured over various end systems
    * Designed by the application developer
    * Predominant architectural paradigms
      · Client-server
      · Peer-to-peer (P2P)

- Client-server Architecture
  - Server
    * Always-on host
    * Permanent IP-address (like ID)
    * Often in data centers, for scaling
  - Clients
    * Contact, communicate with server
    * May be intermittently connected
    * May have dynamic IP addresses
    * Do not communicate directly with each other
  - Examples: HTTP, IMAP, SFTP

- Peer-Peer (P2P) Architecture
  - No always-on server
  - Arbitrary end systems directly communicate
  - Peers request service from other peers, provide service in return to other peers
    * Self scalability — new peers bring new service capacity, as well as new service demands
  - Peers are intermittently connected and change IP addresses
    * Complex management
  - Example: P2P File Sharing

- Process Communication
  - Process — program running within a host
    * Within same host, two processes communicate using inter-process communication, defined by OS (Operating System)
    * Processes in different hosts communicate by exchansing messages
  - Client process — process that initiates communication
  - Server process — process that wants to be contacted

  - Note: applications with P2P architectures have client processes 7 server processes

- Sockets
  - Process send/receives messages to/from its socket
  - Socket analogous to door
    * Sending process shoves message out the door
    * Sending process relies on transport infrastructure on other side of door to deliver message to socket at receiving process

- Addressing Processes
  - To receive messages, a process must have an identifier
  - Host device has a unique IP address
  - Identifier includes both IP address and port numbers associated with process on host
    * HTTP server: 80
    * Mail server: 25
  - To send HTTP message to gaia.cs.umass.edu web server:
    * IP address: 128.119.245.12
    * Port number: 80

- An Application Layer Protocol Defines:
  - Types of messages exchanged
    * Example: request, response
  - Message syntax
    * What fields in messages & how fields are delineated
  - Message semantics
    * Meaning of information in fields
  - Rules for when and how processes send & respond to messages

- Application Layer Protocols can be
  - Open protocols
    * Defined in RFCs, everyone has access to protocol definition
    * Allows for interoperability
    * Example: HTTP, SMTP
  - Proprietary protocols

        * Example: Skype

- Transport Layer Services for Applications
    - Transport layer is on the other side of the "door"
    - There are multiple Transport-layer protocols that provide different services
    - The application developer must choose a Transport-layer protocol, depending on the services needed by the application
        * Examples: priority mail, express mail, certified mail
    - A Transport-layer protocol can provide a different array of services

- Transport Services
    - Data integrity/reliable transport
        * Some apps (*e.g.* file transfer, web transactions) require 100% reliable data transfer
        * Other apps (*e.g.* audio) can tolerate some loss
    - Timing
        * Some apps (*e.g.* Internet telephony, interactive games) require low delay to be "effective"
    - Throughput
        * Some apps (e.g., multimedia) require minimum amount of throughput to be "effective"
        * Other apps ("elastic apps") make use of whatever throughput they get
    - Security
        * Encryption, data integrity, ...

- Internet Transport Protocol Services
    - TCP Service
        * Reliable transport between sending and receiving processes
        * Flow control — sender will not overwhelm receiver
        * Congestion control — throttle sender when network overloaded
        * Does not provide timing, minimum throughput guarantee, security
        * Connection-oriented service: setup required between client and service processes
    - UDP Service:
        * Unreliable data transfer between sending and receiving process
        * Does not provide reliability, flow control, congestion control, timing, throughput guarantee, security, or connection setup

        * Connectionless service: no setup required

- Vanilla TCP & UDP sockets
  - No encryption
  - Clear text passwords sent into socket traverse Internet in clear text
- Transport Layer Security (TLS)
  - Provides encrypted TCP connections
  - Data integrity
  - End-point authentication
  - TSL implemented in Application Layer
    * Applications use TLS libraries, that use TCP in turn
  - TLS socket API
    * Clear text sent into socket traverse Internet encrypted
- Designing Network Applications
  - It is a complex process
  - Requires knowledge of programming, software engineering, and networking
  - From a networking point of view, there are two major decisions:
    1. Type of application (aka Application Architecture)
       * Client-server vs. peer-to-peer
    2. Services requested to the Transport Layer
       * *E.g.* reliable vs. unreliable data transfer