**Application Layer (CDN) + Transport Layer Part 1**

* Video Streaming:
  + Video traffic; solution: distributed, application-level infrastructure
  + Multimedia: video
    - CBR (constant bit rate)
    - VBR (variable bit rate)
      * Spatial coding: use redundancy within images
      * Temporal coding: use redundancy between images
  + Streaming multimedia: DASH (Dynamic, Adaptive Streaming over HTTP)
    - Server: divides video file into multiple chunks; each chunk stored, encoded at different rates; manifest file (provides URLs for different chunks)
    - Client: periodically measures server-to-client bandwidth; consulting manifest, requests one chunk at a time
* CDN (Content Distribution Networks): store/serve multiple copies of videos at multiple geographically distributed sites (enter deep & bring home)
  + Pull caching: direct result of clients’ requests
  + Push replication: expectation of high access rate
* Transport services and protocols
  + Provide logical communication between app processes running on different hosts
  + Transport protocols run in end systems
    - Send side: breaks app messages into segments passes to network layer
    - Receive side: reassembles segments into messages passes to app layer
  + 传输层vs网络层
    - 网络层：提供主机之间的逻辑通信机制
    - 传输层：提供应用进程之间的逻辑通信机制
* Multiplexing/demultiplexing 多路复用/分用
  + Multiplexing at sender: handle data from multiple sockets, add transport header
  + Demultiplexing at receiver: use header info to deliver received segments to correct socket
    - Datagram (source\_IP, dest\_IP, segment (source\_port#, dest\_port#))
    - Direct to socket
  + Connectionless demux 无连接分用
    - Created socket (host-local port #)
    - Identified by Datagram (dest\_IP, dest\_port#) send into UDP socket
    - Receives UDP segment: checks dest\_Port #, directs UDP segment to socket
    - IP datagrams with same dest, port #, but different source IP addresses and/or source port numbers will be directed to same socket dest
  + Connection-oriented demux 面向连接的分用
    - TCP socket identified by 4-tuple: (source\_IP, source\_port#, dest\_IP, dest\_port#)
    - Receiver uses all 4 values to direct segment to socket
    - Server host may support many simultaneous TCP sockets
    - Web servers have different sockets for each connecting client
      * Non-persistent HTTP will have different socket for each request
* UDP (src\_port#, dest\_port#, length, checksum, messagel) 基于Internet IP协议
  + “Best effort” -- may lost, delivered out-of-order
  + Connectionless
    - No handshaking between UDP sender/receiver
    - Each UDP segment handled independently of others
* Why UDP?
  + No connection establishment
  + Simple: connectionless
  + Small header size
  + No congestion control: as fast as desired
* UDP checksum
  + Sender
    - 将段的内容视为16-bit 整数
    - “校验和”计算：计算所有整数的和，进位加在和的后面，将得到的值按位求反，得到校验和
    - 发送方将校验和放入校验和字段
  + Receiver
    - 计算所收到端的校验和
    - 将其与校验和字段进行对比
      * 不相等：检验出错误
      * 相等：没有检测出错误（但有可能有错误）
* rdt (reliable data transfer) 可靠数据传输
  + rdt1.0: reliable transfer over a reliable channel (no bits error, no loss of packets)
  + rdt2.0: channel with bit errors (may flip bits in packet, checksum to detect bit errors)
    - ACK/NAK
    - Stop and wait
  + rdt2.1: if ACK/NAK corrupted (seq# (0, 1) added to packet) duplicates return ACK
  + rdt2.2: a NAK-free protocol (ACK only) returns (ACK, seq#) check duplicates