**Application Layer (Principle, Web)**

* IPC (Interprocess Communication) 进程通信
  + Processes talk to each other through IPC
  + Single machine: shared memory
  + Across machines: message passing 消息交换
* Sockets 套接字
  + Process sends/receives messages to/from its socket 进程间通信利用socket
* Addressing processes 寻址进程
  + To receive messages, process must have identifier 每个进程必须拥有标识符
  + Identifier = IP address (32-bit) + port numbers
* Client-server architecture
  + Server
    - Always-on host
    - Permanent IP address
    - Static port conventions (http: 80, email: 25, ssh: 22)
    - Data centres for scaling 利用大量服务器实现可扩展性
    - May communicate with other servers to respond
  + Client
    - May be intermittently connected 间歇性接入网络
    - May have dynamic IP address
    - Do not communicate directly with each other
* P2P architecture
  + No always-on server
  + Arbitrary end systems (peers) directly communicate 任意端系统可以直接通讯
  + Symmetric responsibility (间歇性接入网络, 可以改变IP地址)
  + Often used for: file sharing, games, video distribution/chat. In general, ‘distributed systems’.
* P2P: Pros and Cons
  + Self scalability, speed, reliability, geographic distribution (高度可伸缩)
  + Decentralized control, distributed algorithms are complex (难于管理)
* App-layer protocol defines 应用层协议
  + Types of messages exchanged
  + Message syntax
  + Message semantics
  + Rules
* What transport service does an app need? 应用需求与传输层服务
  + Data integrity: tolerance of data loss
  + Timing: require low delay to be “effective”
  + Throughput: require minimum amount of throughput to be “effective” vs “elastic apps”
  + Security
* Internet transport protocol services （Internet提供的传输服务）
  + TCP service
    - Reliable transport
    - Flow control: sender won’t overwhelm receiver
    - Congestion control: throttle sender when network overloaded
    - Does not provide: time, min throughput guarantee, security
    - Connection-oriented: c/s processes setup is required
  + UDP service
    - Unreliable data transfer
    - Does not provide: all these above …
* Web and HTTP
  + Web page consists of objects
  + Web page consists of base HTML-file which includes several referenced objects (URL)
  + URL - protocol://host-name[:port]/directory-path/resource
* HTTP (hypertext transfer protocol) 超文本传输协议
  + Web’s application layer protocol
  + client/server model
  + Uses TCP
    - Client initiates TCP connection to server, port 80 (服务器在80端口等待客户请求, 浏览器发起到服务器的tcp链接, 创建套接字socket)
    - Server accepts TCP connection from client
    - HTTP messages exchanged between browser (client) and Web server (server)
    - TCP connection closed
  + HTTP is “stateless”
    - Server maintains no information about past client requests
* Uploading form input
  + POST method - input is upload to server in entity body
  + GET method - input is uploaded in URL
* User-server state: cookies
  + Four components
    - Cookie header line of HTTP response message
    - Cookie header line in next HTTP request message
    - Cookie file kept on user’s host, managed by user’s browser
    - Back-end database at Web site
  + Third party cookies
* Performance of HTTP
  + PLT (Page Load Time) as the metric (from click until user sees page)
  + Depends on many factors (e.g. content/structure, protocols involved, bandwidth, RTT)
* How to import PLT
  + Reduce content size for transfer (smaller images, compression)
  + Change HTTP to make better use of available bandwidth (persistent connections and pipelining)
  + Change HTTP to avoid repeated transfers of the same content (caching and web-proxies)
  + Move content closer to the client (CDNs)
* Non-persistent HTTP 非持久性连接
  + At most one object sent over TCP connect. I.e. connect then closed.
  + RTT: time for small packet to travel from client to server and back
  + Response time: 2RTT + file transmission time
  + HTTP/1.0 (fairly poor PLT)
* Persistent HTTP 持久性连接
  + Server leaves TCP connection open after sending response
  + Subsequent HTTP messages between same client/server are sent over the same TCP connection
  + Persistent without pipelining 无流水的持久性连接
    - Client issues new request only when previous response has been received
    - One RTT for each referenced object
  + Persistent with pipelining 带有流水机制的持久性连接
    - Client sends requests as soon as it encounters a referenced object
    - As little as one RTT for all the referenced objects
    - HTTP/1.1
* Web caches (proxy server) Web 缓存/代理服务器技术
  + Goal: satisfy client request without involving origin server
    - Why?
      * Reduce response time for client request
      * Reduce traffic on an institution’s access link
      * Internet dense with caches: enables “poor” content providers to effectively deliver content 在大范围内internet实现有效的内容分发
  + User sets browser: Web accesses via cache
  + Browser sends all HTTP requests to cache
  + Cache acts as both client and server
  + Typically cache is installed by ISP
* Conditional GET
  + Goal: don’t send object if cache has up-to-date cached version
  + Cache: specify date of cached copy in HTTP request
  + Server: response contains no object if cached copy is up-to-date
* HTTPS
  + HTTP over a connection encrypted by TLS (Transport Layer Security)
  + Provides: authentication & bidirectional encryption 双向加密
* HTTP/2
  + Better content structure
  + Improvements