Computer Networks

Application Layer

Examples of network applications

• e-mail
• web
• instant messaging
• remote login
• P2P file sharing
• multi-user network games
• streaming stored video clips

• social networks
• voice over IP
• real-time video conferencing
• grid computing
Creating a network app

write programs that
  – run on (different) end systems
  – communicate over network
  – e.g., web server software communicates with browser software

No need to write software for network-core devices
  – Network-core devices do not run user applications
  – applications on end systems allows for rapid app development, propagation

Application architectures

• Client-server
  – Including data centers / cloud computing
• Peer-to-peer (P2P)
• Hybrid of client-server and P2P
Client-server architecture

**server:**
- always-on host
- permanent IP address
- server farms for scaling

**clients:**
- communicate with server
- may be intermittently connected
- may have dynamic IP addresses
- do not communicate directly with each other

---

**Google Data Centers**

- Estimated cost of data center: $600M
- Google spent $2.4B in 2007 on new data centers
- Each data center uses 50-100 megawatts of power
Pure P2P architecture

- no always-on server
- arbitrary end systems directly communicate
- peers are intermittently connected and change IP addresses

Highly scalable but difficult to manage

Hybrid of client-server and P2P

Skype
- voice-over-IP P2P application
- centralized server: finding address of remote party:
- client-client connection: direct (not through server)

Instant messaging
- chatting between two users is P2P
- centralized service: client presence detection/location
  - user registers its IP address with central server when it comes online
  - user contacts central server to find IP addresses of buddies
Processes communicating

**Process:** program running within a host.
- within same host, two processes communicate using **inter-process communication** (defined by OS).
- processes in different hosts communicate by exchanging **messages**

**Client process:** process that initiates communication

**Server process:** process that waits to be contacted

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

**Sockets**

- process sends/receives messages to/from its **socket**
- socket analogous to door
  - sending process shoves message out door
  - sending process relies on transport infrastructure on other side of door which brings message to socket at receiving process

- API: (1) choice of transport protocol; (2) ability to fix a few parameters
Addressing processes

• to receive messages, process must have *identifier*
• host device has unique 32-bit IP address
• **Exercise:** use `ipconfig` from command prompt to get your IP address (Windows)

• **Q:** does IP address of host on which process runs suffice for identifying the process?
  – **A:** No, many processes can be running on same host.

• *Identifier* includes both IP address and port numbers associated with process on host.

• Example port numbers:
  – HTTP server: 80
  – Mail server: 25

Application-layer protocol defines

• Types of messages exchanged,
  – e.g., 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

**Public-domain protocols:**
• defined in RFCs
• allows for interoperability
  – e.g., HTTP, SMTP, BitTorrent

**Proprietary protocols:**
• e.g., Skype, ppstream
What transport service does an app need?

Data loss
- some apps (e.g., audio) can tolerate some loss
- other apps (e.g., file transfer, telnet) require 100% reliable data transfer

Timing
- some apps (e.g., Internet telephony, interactive games) require low delay to be “effective”

Throughput
- some apps (e.g., multimedia) require a certain minimum amount of throughput to be “effective”
- other apps (“elastic apps”) make use of whatever throughput they get

Security
- Encryption, data integrity, ...

Transport service requirements of common apps

<table>
<thead>
<tr>
<th>Application</th>
<th>Data loss</th>
<th>Throughput</th>
<th>Time Sensitive</th>
</tr>
</thead>
<tbody>
<tr>
<td>file transfer</td>
<td>no loss</td>
<td>elastic</td>
<td>no</td>
</tr>
<tr>
<td>e-mail</td>
<td>no loss</td>
<td>elastic</td>
<td>no</td>
</tr>
<tr>
<td>Web documents</td>
<td>no loss</td>
<td>elastic</td>
<td>no</td>
</tr>
<tr>
<td>real-time audio/video</td>
<td>loss-tolerant</td>
<td>audio: 5kbps-1Mbps, video:10kbps-5Mbps</td>
<td>yes, 100’s msec</td>
</tr>
<tr>
<td>stored audio/video</td>
<td>loss-tolerant</td>
<td>same as above</td>
<td>yes, few secs</td>
</tr>
<tr>
<td>interactive games</td>
<td>loss-tolerant</td>
<td>few kbps up</td>
<td>yes, 100’s msec</td>
</tr>
<tr>
<td>instant messaging</td>
<td>no loss</td>
<td>elastic</td>
<td>yes and no</td>
</tr>
</tbody>
</table>
Internet transport protocols services

TCP service:
- connection-oriented: setup required between client and server processes
- reliable transport between sending and receiving process
- flow control: sender won’t overwhelm receiver
- congestion control: throttle sender when network overloaded
- does not provide: timing, minimum throughput guarantees, security

UDP service:
- unreliable data transfer between sending and receiving process
- does not provide: connection setup, reliability, flow control, congestion control, timing, throughput guarantee, or security

Q: why bother? Why is there a UDP?

Internet apps: application, transport protocols

<table>
<thead>
<tr>
<th>Application</th>
<th>Application layer protocol</th>
<th>Underlying transport protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>e-mail</td>
<td>SMTP [RFC 2821]</td>
<td>TCP</td>
</tr>
<tr>
<td>remote terminal access</td>
<td>Telnet [RFC 854]</td>
<td>TCP</td>
</tr>
<tr>
<td>Web</td>
<td>HTTP [RFC 2616]</td>
<td>TCP</td>
</tr>
<tr>
<td>file transfer</td>
<td>FTP [RFC 959]</td>
<td>TCP</td>
</tr>
<tr>
<td>streaming multimedia</td>
<td>HTTP (eg Youtube), RTP [RFC 1889]</td>
<td>TCP or UDP</td>
</tr>
<tr>
<td>Internet telephony</td>
<td>SIP, RTP, proprietary (e.g., Skype)</td>
<td>typically UDP</td>
</tr>
</tbody>
</table>