Application layer

Reading: RFC 3117
Kurose-Ross chapter 2

Socket Programming

Socket, Port and IP address

Socket interaction: TCP

- Server (running on hostid)
  - create socket, port=x, for incoming request: socket(); bind()
  - wait for incoming connection request: ConnSock = accept()
  - read request from ConnSock
  - write reply to ConnSock
  - close ConnSock

- Client
  - create socket, connect to hostid, port=x: ClientSock = socket(); connect()
  - write request to ClientSock
  - read reply from ClientSock
  - close ClientSock
Socket interaction: UDP

Server (running on hostid)

- create datagram socket, port=x, for incoming request: ServSock = socket(); bind()
- read request from ServSock
- write reply to ServSock, specifying host and port number

Client

- create datagram socket, ClientSock = socket()
- Create, address (hostid, port=x) datagram request, send using ClientSock
- read reply from ClientSock
- close ClientSock

Aside: Retransmission on error – use idempotent operations.

Application Protocols

Design and operation

App Protocol Design Issues

- Dialog control – whose turn to “talk” (session layer issue); asynchrony; parallelism
- Data representation – network standard encoding (presentation-layer issue)
- Security – authentication, privacy
- Transport-layer – connection/connectionless
- Framing of messages
- Error/status reporting
- Syntax and semantics of message
- State maintenance – client, server, both

Reference: RFC 3117

Application protocol examples

- Telnet
- HTTP
- SMTP
- MIME
- POP3
- IMAP
- FTP
- DNS
- BOOTP
- DHCP
Telnet Design

- Dialog: Asynchronous
- Representation: raw bytes; IAC byte-stuffed; CRLF
- Security: Nil
- Transport-layer: TCP
- Framing: Byte-by-byte
- Error reporting: Minimal
- Syntax: IAC-escaped commands
- State: Server: Logged in “shell”

HTTP Design

- Dialog: Command-reply; pipelined commands (v1.1)
- Representation: MIME objects
- Security: HTTPS provided by SSL
- Transport-layer: TCP
- Framing: HTTP/1.0: connection; HTTP/1.1: length header in MIME object
- Error reporting: 3-digit error codes
- Syntax: ASCII commands and parameters; CRLF; MIME objects (headers and data)
- State: Client maintains state; stateless server (cookies)

SMTP Design

- Dialog: Take turns
- Representation: ASCII text, CRLF
- Security: Minimal
- Transport-layer: TCP
- Framing: CRLF; CRLF “.” CRLF
- Error reporting: theory of error codes; human-readable text message
- Syntax: four-letter commands; ASCII text parameters; CRLF
- State: Both: short-term state (e.g. recipient list); long-term (e-mail queues)

MIME

- Is not a protocol but is used in SMTP and other protocols to address certain issues:
- Data typing: MIME types
- Representation: ASCII text or binary data
- Security: nil
- Framing: external to MIME objects; some protocols add a length header
- Error reporting: not applicable
- Syntax: headers in ASCII text (mail format); blank line; data object encoded according to header
### POP3 Design
- **Dialog**: Take turns
- **Representation**: ASCII text (email)
- **Security**: Secure authorisation option
- **Transport-layer**: TCP
- **Framing**: CRLF; CRLF “.” CRLF
- **Error reporting**: +OK -ERR
- **Syntax**: ASCII text commands and parameters
- **State**: Both (per session: protocol stage; authorised user; items marked for deletion)

### IMAP Design
- **Dialog**: Pipelined commands
- **Representation**: ASCII text
- **Security**: Authentication option; protection option
- **Transport-layer**: TCP
- **Framing**: CRLF; continuation flag
- **Error reporting**: OK NO BAD
- **Syntax**: ASCII commands and parameters
- **State**: Both: Per session (authenticated user; selected folder); Server: folders and items status maintained between sessions

### FTP Design
- **Dialog**: Take turns; out-of-band data
- **Representation**: Text files CRLF; binary files
- **Security**: Nil: Passwords in plain text
- **Transport-layer**: TCP
- **Framing**: CRLF; connection “blasting” for files
- **Error reporting**: 3-digit codes; human readable text
- **Syntax**: ASCII commands and parameters
- **State**: Both: per session (authorised user)

### DNS: iterated queries
- **_recursive query**: puts burden of name resolution on contacted name server
  - **heavy load?**
- **iterated query**: contacted server replies with name of server to contact: “I don’t know this name, but ask this server”
DNS Design
- Dialog: Query-response
- Representation: RRs; 16-bit MSB first
- Security: Nil
- Transport-layer: UDP or TCP
- Framing: Datagram; RR counts
- Error reporting: Error flag bits
- Syntax: Binary data
- State: Stateless protocol (query-response)

BOOTP Design
- Dialog: Query-response
- Representation: Binary/text data; MSB first
- Security: Nil
- Transport-layer: UDP
- Framing: Fixed-size Datagram
- Error reporting: Nil – discard packet
- Syntax: Fixed fields (RFC1497: tagged fields)
- State: Stateless protocol (query-response)

DHCP obtaining IP address

<table>
<thead>
<tr>
<th>Client</th>
<th>Server 1</th>
<th>Server 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DHCPDISCOVER</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DHCPREQUEST</td>
<td>Commit Lease</td>
</tr>
<tr>
<td></td>
<td>DHCPACK</td>
<td>Offer Declined</td>
</tr>
<tr>
<td></td>
<td>DHCPRELEASE</td>
<td>Discard Lease</td>
</tr>
</tbody>
</table>

DHCP Design
- Dialog: Query-response
- Representation: Binary/text data; MSB first
- Security: Nil
- Transport-layer: UDP
- Framing: Datagram
- Error reporting: DHCPNAK message
- Syntax: Fixed fields; tagged fields (RFC1497)
- State: Server maintains IP lease data
Secure Sockets Layer (SSL)

- A protocol widely used on the Web
- Operates between the application and transport layers
- Operations of SSL
  - Negotiation for PKI
    - Server and browser negotiate to select cryptographic algorithm and create a session secret key.
  - Communications
    - Encrypted by using the key that was negotiated.

Security goals

- Secrecy
- Authentication
- Non-repudiation
- Integrity

Approaches

- Secret key
  - Alice and Bob share a secret k
  - Public algorithms E (encrypt), D (decrypt)
  - \( P \rightarrow E_k(P) \rightarrow D_k(E_k(P)) \)
- Public key
  - Bob creates a pair of keys \( E_b, D_b \)
    - Different but mathematically related
  - Public algorithms \( E, D \) require key pair
  - \( P \rightarrow E_{E_b}(P) \rightarrow D_{D_b}(E_{E_b}(P)) \)
Network layer

1: Introduction to TCP/IP, IP design
2: IP addressing, Address resolution
3: IP Routing

IP Datagram

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>Version of IP protocol</td>
</tr>
<tr>
<td>IHL</td>
<td>Internet Header Length</td>
</tr>
<tr>
<td>DS service type</td>
<td>DS service type of the packet</td>
</tr>
<tr>
<td>Total Length</td>
<td>Total length of the IP datagram</td>
</tr>
<tr>
<td>Identification</td>
<td>Identification of the datagram</td>
</tr>
<tr>
<td>Flags</td>
<td>Flags of the datagram</td>
</tr>
<tr>
<td>Fragment offset (13)</td>
<td>Fragment offset for MTU</td>
</tr>
<tr>
<td>Time to Live (TTL)</td>
<td>Time to Live (TTL) of the datagram</td>
</tr>
<tr>
<td>Protocol</td>
<td>Protocol of the datagram</td>
</tr>
<tr>
<td>Header Checksum</td>
<td>Header Checksum of the datagram</td>
</tr>
<tr>
<td>Source Address</td>
<td>Source address of the datagram</td>
</tr>
<tr>
<td>Destination Address</td>
<td>Destination address of the datagram</td>
</tr>
<tr>
<td>Options</td>
<td>Options in the datagram</td>
</tr>
<tr>
<td>Data</td>
<td>Data of the datagram</td>
</tr>
</tbody>
</table>

IP Fragmentation

- Fragmentation: Division of packet into smaller units to accommodate a protocol’s MTU.
- Each fragment has its own header.
- Fragment can be further fragmented.
- Datagram fragmented at source or any other router in the path.
- Reassembly done only at destination.
  - Why??

Address Classes

- Class A
  - Octet
  - Net ID
  - Host ID

- Class B
  - Octet
  - Net ID
  - Host ID

- Class C
  - Octet
  - NetID
  - HostID
Subnetting

- Subnet
  - Division of a single class A, B, or C network into smaller pieces.
  - Each piece: A physical network in TCP/IP environment.
  - Uses IP address derived from single network ID.
  - Result: Single network (Single Netid) divided into smaller subnets.
    - Each subnet has different network ID.

Address Resolution Protocol (ARP)

- Map IP (Logical) address to a hardware (Physical) address.
  - Called Address resolution
- ARP uses local broadcast to obtain a hardware address.
- Address mappings are stored in cache for future reference.
- Two cases of resolution:
  - Local
  - Remote

Multimedia networks

KR: Kurose and Ross chapter 7
(KR3: 3rd ed)

Pulse Code Modulation

- Quantise pulses and represent as digital output
- Reconstruction is no longer exact
Compression

- **Lossless**
  - Original data can be exactly restored
  - Run-length coding
  - Lempel-Ziv algorithms, LZW
  - Huffman coding
  - Linear prediction

- **Lossy**
  - Relies on studies of human perception
  - Audio and photographs
  - MP3
  - JPEG
  - MPEG

Types of multimedia services

- **Streaming stored media**
- **Streaming live media**
- **Interactive media**
  - VoIP

QoS challenges (KR)

- End-to-end delay
- Jitter
- Packet resequencing
- Packet loss

Delayed play out

- Fixed delay

![Diagram showing packet generation, packet arrival, and missed playout over time.](After KR fig 7)
FEC

- Aim: To provide sufficient data to correct packet loss without retransmission
  - Redundant information (e.g. parity block every n blocks)
    - Increases data rate by \((n+1)/n\)
    - Loss may require n-1 packets delay to recover

RTP and RTCP

- RTP mixer
- RTP translator
- RTP in UDP
- RTCP QoS reports

SIP and SDP

- Establish VoIP session
- RTP used for transport
- Comparison with H.323

IP v6 and Network security protocols

COMP347 2006
Len Hamey
IPv6
- Addressing
- No ARP
- Extension headers
- No fragmentation

IP version 6
- Improved options
- Provision for protocol extension
- Autoconfiguration of addresses
- Renumbering of networks
- Resource allocation
  - Flow
  - Diffserv
- Support for very large packets

Security goals
- Integrity
- Availability
- Secrecy/privacy and confidentiality
- Authorisation
- Authentication
- Replay avoidance

IPSec
- AH
- ESP
- Security association
Security Association Parameters

- Sequence number counter
- Sequence counter overflow (flag)
- Anti-replay window
- AH authentication algorithm, keys, key lifetimes, etc
- ESP encryption and authentication algorithms, keys, initialisation values, key lifetimes, etc
- Lifetime of the SA (time or byte count)
- IPSec protocol mode
- Path MTU

Reference: S(CNIS) ch16; RFC 4301 p22-24

VPN

- Packets tunneled between routers
- Security parameters negotiated when the link is brought up

Basic NAT

<table>
<thead>
<tr>
<th>Web browser</th>
<th>Web server</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.0.11</td>
<td>149.22.35.11:80</td>
</tr>
<tr>
<td>192.168.0.11:1326 to 149.22.35.11:80</td>
<td>192.168.0.11:1326 to 149.22.35.11:80</td>
</tr>
<tr>
<td>192.168.0.1</td>
<td>137.111.11.25</td>
</tr>
<tr>
<td>192.168.0.11:1326 to 149.22.35.11:80</td>
<td>192.168.0.11:1326 to 149.22.35.11:80</td>
</tr>
<tr>
<td>192.168.0.32</td>
<td>149.22.35.11:80</td>
</tr>
<tr>
<td>137.111.11.26:1326 to 149.22.35.11:80</td>
<td>137.111.11.26:9723 to 149.22.35.11:80</td>
</tr>
<tr>
<td>137.111.11.26</td>
<td>137.111.11.26:9723</td>
</tr>
<tr>
<td>137.111.11.26:9723 to 149.22.35.11:80</td>
<td>149.22.35.11:80 to 192.168.0.11:1326</td>
</tr>
<tr>
<td>137.111.11.26:9723</td>
<td>137.111.11.26</td>
</tr>
</tbody>
</table>

NAT: Port address translation

<table>
<thead>
<tr>
<th>Web browser</th>
<th>Web server</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.0.11</td>
<td>149.22.35.11</td>
</tr>
<tr>
<td>192.168.0.11:1326 to 149.22.35.11:80</td>
<td>192.168.0.11:1326 to 149.22.35.11:80</td>
</tr>
<tr>
<td>192.168.0.1</td>
<td>137.111.11.25</td>
</tr>
<tr>
<td>192.168.0.11:1326 to 149.22.35.11:80</td>
<td>192.168.0.11:1326 to 149.22.35.11:80</td>
</tr>
<tr>
<td>192.168.0.32</td>
<td>149.22.35.11:80</td>
</tr>
<tr>
<td>137.111.11.26:1326 to 149.22.35.11:80</td>
<td>137.111.11.26:9723 to 149.22.35.11:80</td>
</tr>
<tr>
<td>137.111.11.26</td>
<td>137.111.11.26:9723</td>
</tr>
<tr>
<td>137.111.11.26:9723 to 149.22.35.11:80</td>
<td>149.22.35.11:80 to 192.168.0.11:1326</td>
</tr>
<tr>
<td>137.111.11.26:9723</td>
<td>137.111.11.26</td>
</tr>
</tbody>
</table>

to 149.22.35.11:80
Firewall
- Packet filtering
- Bastion host
- Application gateway
- SPI
- DMZ
- Deep packet inspection

Difficult protocols
- Involve additional connections
- May convey port numbers in an existing connection
- FTP
  - Passive mode
- SIP & RTP