Network, Transport, and Application Layers

- TPDU (Transport Protocol Data Unit)
- Transport service provider
- Transport service user

Connection Service Primitives

<table>
<thead>
<tr>
<th>Primitive</th>
<th>Packet sent</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>LISTEN</td>
<td>(none)</td>
<td>Block until some process tries to connect</td>
</tr>
<tr>
<td>CONNECT</td>
<td>CONNECTION REQ.</td>
<td>Actively attempt to establish a connection</td>
</tr>
<tr>
<td>SEND</td>
<td>DATA</td>
<td>Send information</td>
</tr>
<tr>
<td>RECEIVE</td>
<td>(none)</td>
<td>Block until a DATA packet arrives</td>
</tr>
<tr>
<td>DISCONNECT</td>
<td>DISCONNECTION REQ.</td>
<td>This side wants to release the connection</td>
</tr>
</tbody>
</table>

Simple Connection Management

Nesting
Berkeley Sockets

<table>
<thead>
<tr>
<th>Primitive</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOCKET</td>
<td>Create a new communication end point</td>
</tr>
<tr>
<td>BIND</td>
<td>Attach a local address to a socket</td>
</tr>
<tr>
<td>LISTEN</td>
<td>Announce willingness to accept connections; give queue size</td>
</tr>
<tr>
<td>ACCEPT</td>
<td>Block the caller until a connection attempt arrives</td>
</tr>
<tr>
<td>CONNECT</td>
<td>Actively attempt to establish a connection</td>
</tr>
<tr>
<td>SEND</td>
<td>Send some data over the connection</td>
</tr>
<tr>
<td>RECEIVE</td>
<td>Receive some data from the connection</td>
</tr>
<tr>
<td>CLOSE</td>
<td>Release the connection</td>
</tr>
</tbody>
</table>

Environments

Data Link

Transport
**Addressing**

- TSAP (Transport Service Access Point)
- NSAP (Network Service Access Point)

**Initial Connection Protocol**

**Forbidden Region**

**Three-way Handshake**
Abrupt Disconnect with loss of Data

Two-army Problem

Connection Teardown Scenarios

Buffer Management

Fixed sized buffers
Variable sized buffers
Circular buffer
Deadlock – Dropped Window Advertisement

Crash Recovery

A – Acknowledge
W – Write
C – Crash

State Machine for Simple Transport Protocol
Graphical Version of FSM

UDP
- User datagram protocol
  - multiplexes and demultiplexes
- Segments
  - 8 byte header
  - data

RCP over UDP
- Stubs
- Argument marshaling
  - pointers (length may not be known)
  - argument types may not be known (printf)
  - globals

Real-Time Transport Protocol
- Transport protocol that runs in the application layer
- Multimedia
RTP Header

- Timestamp
- Jitter
- RTCP – Real-time Transport Control Protocol
  - No data, just control

Sample Assigned Ports

<table>
<thead>
<tr>
<th>Port</th>
<th>Protocol</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>FTP</td>
<td>File transfer</td>
</tr>
<tr>
<td>23</td>
<td>Telnet</td>
<td>Remote login</td>
</tr>
<tr>
<td>25</td>
<td>SMTP</td>
<td>E-mail</td>
</tr>
<tr>
<td>69</td>
<td>TFTP</td>
<td>Trivial file transfer protocol</td>
</tr>
<tr>
<td>79</td>
<td>Finger</td>
<td>Lookup information about a user</td>
</tr>
<tr>
<td>80</td>
<td>HTTP</td>
<td>World Wide Web</td>
</tr>
<tr>
<td>110</td>
<td>POP-3</td>
<td>Remote e-mail access</td>
</tr>
<tr>
<td>119</td>
<td>NNTP</td>
<td>USENET news</td>
</tr>
</tbody>
</table>

TCP Byte Stream

- Receiver doesn’t know how sender sent the data

- Urgent data – generate a signal on the receiver

TCP Header

- Segment – 20 byte header + options + data
- MTU – Maximum Transfer Unit
- Window specifies # of bytes that may be sent starting from acknowledge
Sample Options

- Window scale
  - fast links with long delay may be idle most of the time
- Selective retransmit (rather than go back n)

TCP Checksum

- Checksums header + data + pseudo-header
- Pseudo-header:

<table>
<thead>
<tr>
<th>Source address</th>
<th>Destination address</th>
<th>00000000</th>
<th>Protocol = 6</th>
<th>TCP segment length</th>
</tr>
</thead>
</table>

TCP Connection

(a)

(b)

TCP Connection State

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLOSED</td>
<td>No connection is active or pending</td>
</tr>
<tr>
<td>LISTEN</td>
<td>The server is waiting for an incoming cell</td>
</tr>
<tr>
<td>SYN RCVD</td>
<td>A connection request has arrived; wait for ACK</td>
</tr>
<tr>
<td>SYN SENT</td>
<td>The application has started to open a connection</td>
</tr>
<tr>
<td>ESTABLISHED</td>
<td>The normal data transfer state</td>
</tr>
<tr>
<td>FIN WAIT 1</td>
<td>The application has said it is finished</td>
</tr>
<tr>
<td>FIN WAIT 2</td>
<td>The other side has agreed to release</td>
</tr>
<tr>
<td>TIMED WAIT</td>
<td>Wait for all packets to die off</td>
</tr>
<tr>
<td>CLOSING</td>
<td>Both sides have tried to close simultaneously</td>
</tr>
<tr>
<td>CLOSE WAIT</td>
<td>The other side has initiated a release</td>
</tr>
<tr>
<td>LAST ACK</td>
<td>Wait for all packets to die off</td>
</tr>
</tbody>
</table>
TCP Connection Establishment

Heavy solid – normal client
Heavy dashed – normal server
Light – unusual events

Window Management in TCP

Window

- Receive window is 0
  - urgent packets
  - 1-byte packet to recover lost window announcement
- Senders may buffer data
  - 1 character / segment, 40 bytes of headers
  - Nagle: send 1 character, buffer until ACK
    - good balance in many cases
    - bad for things like mice (erratic jumps over the network)

Silly Window Syndrome

receiver consumes 1 byte at a time
TCP Congestion Control

- Three parameters
  - Flow control window
  - Threshold
  - Congestion window
- Flow control bounds congestion window
- Threshold starts at 64K
- Congestion window starts at 1 and grows
  - "slow start" until threshold: +1 per segment
  - linear once it passes threshold: +1 per batch
- On timeout
  - threshold = congestion / 2
  - congestion = 1

TCP Timer Management

Probability Density for ACK arrival

Example Illustrating Threshold and Congestion Window
Timers

- Estimating RTT \( RTT = \alpha RTT + (1 - \alpha)M \)
  \( \alpha = 7/8 \)
- Timeout \( \beta RTT \) (\( \beta = 2 \))
- Dynamic estimate of variance
  \( D = \alpha D + (1 - \alpha)(RTT - M) \)
- Karn’s algorithm
  - don’t update RTT for retransmitted segment
- Persistence time
  - dropped window advertisements
- Keepalive timer
- TIMED WAIT

Wireless TCP and UDP

- Jacobson’s rule no longer holds
- Indirect TCP: split TCP into two connections
  ![Wireless TCP Diagram]
  - Alternate: base station becomes active in making the last kilometer reliable

Transactional TCP

- SYN includes data
  ![Transactional TCP Diagram]
  - SYN includes data
  - SYN, ACK
  - FIN
  - ACK(FIN)

Performance Issues

- Problems
- Measurement
- System design
- Fast TPDU processing
- Newer protocols
**Bandwidth-delay product**

- bandwidth times RTT

**Measurement**

- Large enough sample size
  - multiple TPDUs
  - statistical significance
- Sampling times
- Coarse grained clocks
- Check for external events
- Watch for caching
- Understand what you are measuring
- Watch out for extrapolation

**Extrapolation**

**System Design**

- Rule #1: CPU speed is more important than network speed
  - protocol processing is the bottleneck
- Rule #2: Reduce packet count
  - per packet overhead
  - pipelined processors
  - Nagle and Clark's silly window syndrome
- Rule #3: Minimize context switches (*)
- Rule #4: Minimize copies
- Rule #5: Bandwidth is easy, latency is hard
- Rule #6: Congestion avoidance is better than detection and recover
- Rule #7: Avoid timeouts
Context Switches

User process running at the time of the packet arrival
Network manager
Receiving process

User space
Kernel space

Fast TPDU processing

- Know what the fast path is

TCP and IP fields that remain constant in a stream

<table>
<thead>
<tr>
<th>Source port</th>
<th>Destination port</th>
<th>VER</th>
<th>IHL</th>
<th>TOS</th>
<th>Total length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence number</td>
<td>Acknowledgement number</td>
<td>Identification</td>
<td>Fragment offset</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Len</td>
<td>Unused</td>
<td>Window size</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Checksum</td>
<td>Urgent pointer</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Timer Management

- Timer wheel
Gigabit Networks

- Sequence number wraparound
- Communication speeds are improving faster than processor speeds
- Go back n is bad when bandwidth-delay product is large
- Gigabit lines are delay limited rather than bandwidth limited (see graph)
- Variance may be more important than bandwidth

Design for speed, not bandwidth optimization