Transport Protocols
Transport Protocols

- Connection Oriented Transport Protocol Mechanisms
  - Logical connection
  - Establishment
  - Maintenance termination
  - Reliable
  - e.g. TCP
Reliable Sequencing Network Service

- Assume arbitrary length message
- Assume virtually 100% reliable delivery by network service
  - e.g. reliable packet switched network using X.25
  - e.g. frame relay using LAPF control protocol
  - e.g. IEEE 802.3 using connection oriented LLC service
- Transport service is end-to-end protocol between two systems on same network
Issues in a Simple Transport Protocol

- Addressing
- Multiplexing
- Flow Control
- Connection establishment and termination
Addressing

• Target user specified by:
  — User identification
    • Usually host, port
      — Called a socket in TCP
    • Port represents a particular transport service (TS) user
  — Transport entity identification
    • Generally only one per host
    • If more than one, then usually one of each type
      — Specify transport protocol (TCP, UDP)
  — Host address
    • An attached network device
    • In an internet, a global internet address
  — Network number
Finding Addresses

• Four methods
  — Know address ahead of time
    • e.g. collection of network device stats
  — Well known addresses
  — Name server
  — Sending process request to well known address
Multiplexing

- Multiple users employ same transport protocol
- User identified by port number or service access point (SAP)
- May also multiplex with respect to network services used
  - e.g. multiplexing a single virtual X.25 circuit to a number of transport service user
    - X.25 charges per virtual circuit connection time
Flow Control

• Longer transmission delay between transport entities compared with actual transmission time
  —Delay in communication of flow control info

• Variable transmission delay
  —Difficult to use timeouts

• Flow may be controlled because:
  —The receiving user can not keep up
  —The receiving transport entity can not keep up

• Results in buffer filling up
Coping with Flow Control
Requirements (1)

• Do nothing
  —Segments that overflow are discarded
  —Sending transport entity will fail to get ACK and will retransmit
    • Thus further adding to incoming data

• Refuse further segments
  —Clumsy
  —Multiplexed connections are controlled on aggregate flow
Coping with Flow Control
Requirements (2)

• Use fixed sliding window protocol
  — Works well on reliable network
    • Failure to receive ACK is taken as flow control indication
    — Does not work well on unreliable network
      • Can not distinguish between lost segment and flow control

• Use credit scheme
Credit Scheme

- Greater control on reliable network
- More effective on unreliable network
- Decouples flow control from ACK
  - May ACK without granting credit and vice versa
- Each octet has sequence number
- Each transport segment has seq number, ack number and window size in header
Use of Header Fields

- When sending, seq number is that of first octet in segment
- ACK includes
  - ack number $AN=i$
  - window number $W=j$
- All octets through seq. num. $SN=i-1$ acknowledged
  - Next expected octet is $i$
- Permission to send additional window of $W=j$ octets
  - i.e. octets through $i+j-1$
Credit Allocation

Transport Entity A

...1000 1001 2400 2401...
A may send 1400 octets

SN = 1001
SN = 1201
SN = 1401

...1000 1001 1601 2401...
A shrinks its transmit window with each transmission

SN = 1601
SN = 1801

...1600 1601 2001 2601...
A adjusts its window with each credit

SN = 2001
SN = 2201
SN = 2401

...1600 1601 2600 2601...
A exhausts its credit

AN = 2601 \cdot W = 1400

...2600 2601 4000 4001...
A receives new credit

Transport Entity B

...1000 1001 2400 2401...
B is prepared to receive 1400 octets, beginning with 1001

SN = 1001
SN = 1201
SN = 1401

...1600 1601 2601...
B acknowledges 3 segments (600 octets), but is only prepared to receive 200 additional octets beyond the original budget (i.e., B will accept octets 1601 through 2600)

SN = 1601
SN = 1801

...1600 1601 2001 2601...

SN = 2001
SN = 2201
SN = 2401

...2600 2601 4000 4001...
B acknowledges 5 segments (1000 octets) and restores the original amount of credit
Sending and Receiving Perspectives

(a) Send sequence space

Data octets so far acknowledged
Octets not yet acknowledged
Window of octets that may be transmitted

Initial Sequence Number (ISN)
Last octet acknowledged (AN – 1)
Last octet transmitted
Window shrinks from trailing edge as segments are sent
Window expands from leading edge as credits are received

(b) Receive sequence space

Data octets so far acknowledged
Octets not yet acknowledged
Window of octets that may be accepted

Initial Sequence Number (ISN)
Last octet acknowledged (AN – 1)
Last octet received
Window shrinks from trailing edge as segments are received
Window expands from leading edge as credits are sent
Establishment and Termination

- Allow each end to now the other exists
- Negotiation of optional parameters
- Triggers allocation of transport entity resources
- By mutual agreement
Connection State Diagram

- **CLOSED**
  - Active Open: send SYN
  - Passive Open
  - Close

- **SYN SENT**
  - Receive SYN

- **LISTEN**
  - Receive SYN: Send SYN
  - Send FIN

- **ESTAB**
  - Close
  - Receive FIN

- **FIN WAIT**
  - Receive FIN

- **CLOSE WAIT**
  - Close: Send FIN

- **CLOSED**
  - Close
  - Send FIN

---

Legend:

- Event
- Action
- State
Connection Establishment

(a) Active/Passive Open

(b) Active/Active Open
If TS is Not Listening

• Three things can happen
  — Reject with RST (Reset)
  — Queue request until matching open issued
  — Signal transport service (TS) user to notify of pending request
• May replace passive open with accept
Termination

- Either or both sides
- By mutual agreement
- Abrupt termination
- Or graceful termination
  — Close wait state must accept incoming data until FIN received
Graceful Degradation: Consider the Side Initiating Termination

- TS user Close request
- Transport entity sends FIN, requesting termination
- Connection placed in FIN WAIT state
  - Continue to accept data and deliver data to user
  - Not send any more data
- When FIN received, inform user and close connection
Now consider side not Initiating termination

- FIN received
- Inform TS user Place connection in CLOSE WAIT state
  — Continue to accept data from TS user and transmit it
- TS user issues CLOSE primitive
- Transport entity sends FIN
- Connection closed

- All outstanding data is transmitted from both sides
- Both sides agree to terminate
Unreliable Network Service

• E.g.
  — internet using IP,
  — frame relay using LAPF
  — IEEE 802.3 using unacknowledged connectionless LLC

• Segments may get lost
• Segments may arrive out of order
Problems

• Ordered Delivery
• Retransmission strategy
• Duplication detection
• Flow control
• Connection establishment
• Connection termination
• Crash recovery
Ordered Delivery

- Segments may arrive out of order
- Number segments sequentially
- TCP numbers each octet sequentially
- Segments are numbered by the first octet number in the segment
Retransmission Strategy

- Segment damaged in transit
- Segment fails to arrive
- Transmitter does not know of failure
- Receiver must acknowledge successful receipt
- Use cumulative acknowledgement
- Time out waiting for ACK triggers re-transmission
Timer Value

- Fixed timer
  - Based on understanding of network behavior
  - Can not adapt to changing network conditions
  - Too small leads to unnecessary re-transmissions
  - Too large and response to lost segments is slow
  - Should be a bit longer than round trip time

- Adaptive scheme
  - May not ACK immediately
  - Can not distinguish between ACK of original segment and re-transmitted segment
  - Conditions may change suddenly
Duplication Detection

• If ACK lost, segment is re-transmitted
• Receiver must recognize duplicates
• Duplicate received prior to closing connection
  — Receiver assumes ACK lost and ACKs duplicate
  — Sender must not get confused with multiple ACKs
  — Sequence number space large enough to not cycle within maximum life of segment
• Duplicate received after closing connection
Incorrect Duplicate Detection
Flow Control

- Credit allocation
- Problem if $AN=i$, $W=0$ closing window
- Send $AN=i$, $W=j$ to reopen, but this is lost
- Sender thinks window is closed, receiver thinks it is open
- Use window timer
- If timer expires, send something
  — Could be re-transmission of previous segment
Connection Establishment

- Two way handshake
  - A send SYN, B replies with SYN
  - Lost SYN handled by re-transmission
    - Can lead to duplicate SYNs
    - Ignore duplicate SYNs once connected
- Lost or delayed data segments can cause connection problems
  - Segment from old connections
  - Start segment numbers fare removed from previous connection
    - Use SYN i
    - Need ACK to include i
    - Three Way Handshake
Two Way Handshake:
Obsolete Data Segment
Two Way Handshake:
Obsolete SYN Segment

A

SYN_i

Connection closed

B

Obsolete SYN_i arrives

SYN_k

B responds; A sends new SYN

SYN_j

B discards duplicate SYN

SN \geq k

B rejects segment as out of sequence
Three Way Handshake: State Diagram

SV = state vector
MSL = maximum segment lifetime
Three Way Handshake: Examples

**A**

- $SYN_i$
- $SYN_j, AN = i + 1$
- $SN = i + 1, AN = j + 1$

(a) Normal operation

**B**

- A initiates a connection
- B accepts and acknowledges
- A acknowledges and begins transmission

(b) Delayed SYN

- Obsolete SYN arrives
- B accepts and acknowledges
- A rejects B’s connection

(c) Delayed SYN, ACK

- A initiates a connection
- Old SYN arrives at A; A rejects
- B accepts and acknowledges
- A acknowledges and begins transmission
Connection Termination

- Entity in CLOSE WAIT state sends last data segment, followed by FIN
- FIN arrives before last data segment
- Receiver accepts FIN
  - Closes connection
  - Loses last data segment
- Associate sequence number with FIN
- Receiver waits for all segments before FIN sequence number
- Loss of segments and obsolete segments
  - Must explicitly ACK FIN
Graceful Close

• Send $FIN_i$ and receive $AN_i$
• Receive $FIN_j$ and send $AN_j$
• Wait twice maximum expected segment lifetime
Failure Recovery

- After restart all state info is lost
- Connection is half open
  - Side that did not crash still thinks it is connected
- Close connection using persistence timer
  - Wait for ACK for (time out) * (number of retries)
  - When expired, close connection and inform user
- Send RST i in response to any i segment arriving
- User must decide whether to reconnect
  - Problems with lost or duplicate data
TCP & UDP

• Transmission Control Protocol (TCP)
  — Connection oriented
  — RFC 793

• User Datagram Protocol (UDP)
  — Connectionless
  — RFC 768
TCP Services

- Reliable communication between pairs of processes
- Across variety of reliable and unreliable networks and internets
- Two labeling facilities
  - Data stream push
    - TCP user can require transmission of all data up to push flag
    - Receiver will deliver in same manner
    - Avoids waiting for full buffers
  - Urgent data signal
    - Indicates urgent data is upcoming in stream
    - User decides how to handle it
TCP Header

<table>
<thead>
<tr>
<th>Bit:</th>
<th>0</th>
<th>4</th>
<th>10</th>
<th>16</th>
<th>31</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Source Port</td>
<td>Destination Port</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sequence Number</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Acknowledgment Number</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data offset</td>
<td>Reserved</td>
<td>U</td>
<td>A</td>
<td>C</td>
<td>P</td>
</tr>
<tr>
<td></td>
<td>Window</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Checksum</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Urgent Pointer</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Options + Padding</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Items Passed to IP

- TCP passes some parameters down to IP
  - Precedence
  - Normal delay/low delay
  - Normal throughput/high throughput
  - Normal reliability/high reliability
  - Security
TCP Mechanisms (1)

- Connection establishment
  - Three way handshake
  - Between pairs of ports
  - One port can connect to multiple destinations
TCP Mechanisms (2)

- Data transfer
  - Logical stream of octets
  - Octets numbered modulo $2^{23}$
  - Flow control by credit allocation of number of octets
  - Data buffered at transmitter and receiver
TCP Mechanisms (3)

• Connection termination
  — Graceful close
  — TCP users issues CLOSE primitive
  — Transport entity sets FIN flag on last segment sent
  — Abrupt termination by ABORT primitive
    • Entity abandons all attempts to send or receive data
    • RST segment transmitted
Implementation Policy Options

- Send
- Deliver
- Accept
- Retransmit
- Acknowledge
Send

- If no push or close TCP entity transmits at its own convenience
  - Data buffered at transmit buffer
  - May construct segment per data batch
  - May wait for certain amount of data
Deliver

- In absence of push, deliver data at own convenience
  - May deliver as each in order segment received
  - May buffer data from more than one segment
Accept

- Segments may arrive out of order
  - In order
    - Only accept segments in order
    - Discard out of order segments
  - In windows
    - Accept all segments within receive window
Retransmit

- TCP maintains queue of segments transmitted but not acknowledged
- TCP will retransmit if not ACKed in given time
  — First only
  — Batch
  — Individual
Acknowledgement

- Immediate
- Cumulative
Congestion Control

- RFC 1122, Requirements for Internet hosts
- Retransmission timer management
  - Estimate round trip delay by observing pattern of delay
  - Set time to value somewhat greater than estimate
  - Simple average
  - Exponential average
  - RTT Variance Estimation (Jacobson’s algorithm)
Use of Exponential Averaging

(a) Increasing function

(b) Decreasing function
Jacobson’s RTO Calculation

(a) Increasing function

(b) Decreasing function
Exponential RTO Backoff

- Since timeout is probably due to congestion (dropped packet or long round trip), maintaining RTO is not a good idea.
- RTO increased each time a segment is re-transmitted.
- $RTO = q \times RTO$
- Commonly $q = 2$
  — Binary exponential backoff
Karn’s Algorithm

• If a segment is re-transmitted, the ACK arriving may be:
  — For the first copy of the segment
    • RTT longer than expected
  — For second copy

• No way to tell

• Do not measure RTT for re-transmitted segments

• Calculate backoff when re-transmission occurs

• Use backoff RTO until ACK arrives for segment that has not been re-transmitted
Window Management

- Slow start
  - $awnd = \text{MIN}([\text{credit, cwnd}])$
  - Start connection with $cwnd=1$
  - Increment $cwnd$ at each ACK, to some max

- Dynamic windows sizing on congestion
  - When a timeout occurs
  - Set slow start threshold to half current congestion window
    - $\text{ssthresh}=cwnd/2$
  - Set $cwnd = 1$ and slow start until $cwnd=ssthresh$
    - Increasing $cwnd$ by 1 for every ACK
  - For $cwnd \geq \text{ssthresh}$, increase $cwnd$ by 1 for each RTT
UDP

- User datagram protocol
  - RFC 768
- Connectionless service for application level procedures
  - Unreliable
  - Delivery and duplication control not guaranteed
- Reduced overhead
- e.g. network management
UDP Uses

• Inward data collection
• Outward data dissemination
• Request-Response
• Real time application
UDP Header

<table>
<thead>
<tr>
<th>Bit:</th>
<th>0</th>
<th>16</th>
<th>31</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Source Port</td>
<td>Destination Port</td>
<td></td>
</tr>
<tr>
<td>8 octets</td>
<td>Length</td>
<td>Checksum</td>
<td></td>
</tr>
</tbody>
</table>
Summary

- connection-oriented network and transport mechanisms and services
- TCP services, mechanisms, policies
- TCP congestion control
- UDP