Connection-Oriented Transport Mechanisms

- Two basic types of transport service:

  **Connection-oriented**
  - Establishment, maintenance and termination of a logical connection between TS users
  - Has a wide variety of applications
  - Most common
  - Implies service is reliable

  **Connectionless or datagram service**
Transport Protocols

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

Reliable Sequencing Network Service

- Issues:
  - Addressing
  - Multiplexing
  - Flow control
  - Connection establishment/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

Multiplexing

- Multiple users employ the same transport protocol and are distinguished by port numbers or service access points

Upward multiplexing
- Multiplexing of multiple connections on a single lower-level connection

Downward multiplexing
- Splitting of a single connection among multiple lower-level connections
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

Alternatives to Flow Control

Requirements

<table>
<thead>
<tr>
<th>Do nothing</th>
<th>Refuse to accept further segments from the network service</th>
</tr>
</thead>
<tbody>
<tr>
<td>Segments that overflow the buffer are discarded</td>
<td>Relies on network service to do the work</td>
</tr>
<tr>
<td>Sending transport entity will retransmit</td>
<td></td>
</tr>
</tbody>
</table>

Receiving transport entity can:

- Use a fixed sliding window protocol
  - With a reliable network service this works quite well
- Use a credit scheme
  - A more effective scheme to use with an unreliable network service
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 in its header:
  - sequence number,
  - acknowledge number and
  - window size

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 $i+j-1$
Assume 200 octets of data in each segment, and original credit of 1400

Transport Entity A

- A may send 1400 octets

- A shrinks its transmit window with each transmission

- A exhausts its credit

Transport Entity B

- B is prepared to receive 1400 octets, beginning with 1001

- 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)

- B acknowledges 5 segments (1000 octets) and restores the original amount of credit

- B acknowledges 5 segments (1000 octets) and restores the original amount of credit

Figure 15.1 Example of TCP Credit Allocation Mechanism

Figure 15.2 Sending and Receiving Flow Control Perspectives
Connection Establishment and Termination

• Serves three main purposes:
  — Allows each end to assure that the other exists
  — Allows exchange or negotiation of optional parameters
  — Triggers allocation of transport entity resources
• Connection establishment is by mutual agreement

![Simple Connection State Diagram](image)
What to do 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

Unreliable Network Service

Examples:

- Internetwork using IP
- Frame relay network using only the LAPF core protocol
- IEEE 802.3 LAN using the unacknowledged connectionless LLC service

- Segments are occasionally lost and may arrive out of sequence due to variable transit delays
Problems

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

Ordered Delivery

- With an unreliable network service it is possible that segments may arrive out of order
- Solution to this problem is to number segments sequentially
  - TCP uses scheme where each data octet is implicitly numbered
Retransmission Strategy

- Events necessitating retransmission:
  - Sending transport does not know transmission was unsuccessful
  - Receiver acknowledges successful receipt by returning a segment containing an acknowledgment number

- Segment may be damaged in transit but still arrives at its destination
- Segment fails to arrive

- No acknowledgment will be issued if a segment does not arrive successfully
  - Resulting in a retransmit
- A timer needs to be associated with each segment as it is sent
- If timer expires before acknowledgment is received, sender must retransmit

Cont.
Table 15.1
Transport Protocol Timers

<table>
<thead>
<tr>
<th>Timer</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retransmission timer</td>
<td>Retransmit an unacknowledged segment</td>
</tr>
<tr>
<td>MSL (maximum segment lifetime) timer</td>
<td>Minimum time between closing one connection and opening another with the same destination address</td>
</tr>
<tr>
<td>Persist timer</td>
<td>Maximum time between ACK/CREDIT segments</td>
</tr>
<tr>
<td>Retransmit-SYN timer</td>
<td>Time between attempts to open a connection</td>
</tr>
<tr>
<td>Keepalive timer</td>
<td>Abort connection when no segments are received</td>
</tr>
</tbody>
</table>

Duplicate Detection

- Receiver must be able to recognize duplicates
- Segment sequence numbers help
- Complications arise if:
  - A duplicate is received prior to the close of the connection
    - Sender must not get confused if it receives multiple acknowledgments to the same segment
    - Sequence number space must be long enough
  - A duplicate is received after the close of the connection
Incorrect Duplicate Detection

- Future acknowledgments will resynchronize the protocol if an ACK/CREDIT segment is lost
- If no new acknowledgments are forthcoming the sender times out and retransmits a data segment which triggers a new acknowledgment
- Still possible for deadlock to occur

Flow Control
Connection Establishment

- Two way handshake
  - A sends SYN, B replies with SYN
  - Lost SYN handled by re-transmission
    - Can lead to duplicate SYN
  - Ignore duplicate SYN 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

SYNn

Connection closed

B

SYN j

Obsolete SYN i arrives

B responds; A sends new SYN

SYN j

B discards duplicate SYN

SV = i

B rejects segment as out of sequence

Three Way Handshake: State Diagram

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

Connection Termination

- Two-way handshake was found to be inadequate for an unreliable network service

- Out of order segments could cause the FIN segment to arrive before the last data segment
  - To avoid this problem the next sequence number after the last octet of data can be assigned to FIN
  - Each side must explicitly acknowledge the FIN of the other using an ACK with the sequence number of the FIN to be acknowledged
**Failure Recovery**

- When the system that the transport entity is running on fails and subsequently restarts, the state information of all active connections 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 in response to any 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: 0 4 8 16 31</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Port</td>
</tr>
<tr>
<td>Destination Port</td>
</tr>
<tr>
<td>Sequence Number</td>
</tr>
<tr>
<td>Acknowledgment Number</td>
</tr>
<tr>
<td>Data offset</td>
</tr>
<tr>
<td>Reserved</td>
</tr>
<tr>
<td>W C E</td>
</tr>
<tr>
<td>R K E</td>
</tr>
<tr>
<td>F E C</td>
</tr>
<tr>
<td>E S T N</td>
</tr>
<tr>
<td>Window</td>
</tr>
<tr>
<td>Checksum</td>
</tr>
<tr>
<td>Urgent Pointer</td>
</tr>
<tr>
<td>Options + Padding</td>
</tr>
<tr>
<td>+----------------------------------</td>
</tr>
</tbody>
</table>
```
TCP Mechanisms

- Can be grouped into:

  - Connection establishment
  - Always uses a three-way handshake
  - Connection is determined by host and port

  - Data transfer
  - Viewed logically as consisting of a stream of octets
  - Flow control is exercised using credit allocation

  - Connection termination
  - Each TCP user must issue a CLOSE primitive
  - An abrupt termination occurs if the user issues an ABORT primitive

Implementation Policy Options

- Send
- Deliver
- Accept
- Retransmit
- Acknowledge
**Send Policy**

- In the absence of both pushed data and a closed transmission window a sending TCP entity is free to transmit data at its own convenience.
- TCP may construct a segment for each batch of data provided or it may wait until a certain amount of data accumulates before constructing and sending a segment.
- Infrequent and large transmissions have low overhead in terms of segment generation and processing.
- If transmissions are frequent and small, the system is providing quick response.

**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 Policy

- If segments arrive out of order the receiving TCP entity has two options:

  **In-order**
  - Accepts only segments that arrive in order; any segment that arrives out of order is discarded
  - Makes for simple implementation but places a burden on the networking facility
  - If a single segment is lost in transit, then all subsequent segments must be retransmitted

  **In-window**
  - Accepts all segments that are within the receive window
  - Requires a more complex acceptance test and a more sophisticated data storage scheme

Retransmit Policy

- Retransmission strategies:

  **First-only**
  - Maintain one retransmission timer for entire queue
  - Efficient in terms of traffic generated
  - Can have considerable delays

  **Batch**
  - Maintain one retransmission timer for entire queue
  - Reduces the likelihood of long delays
  - May result in unnecessary retransmissions

  **Individual**
  - Maintain one timer for each segment in the queue
  - More complex implementation
**Acknowledge Policy**

- **Timing of acknowledgment:**

**Immediate**
- Immediately transmit an empty segment containing the appropriate acknowledgement number
- Simple and keeps the remote TCP fully informed
- Limits unnecessary retransmissions
- Results in extra segment transmissions
- Can cause a further load on the network

**Cumulative**
- Wait for an outbound segment with data on which to piggyback the acknowledgement
- Typically used
- Requires more processing at the receiving end and complicates the task of estimating round-trip time

---

**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
Summary

- Connection-oriented transport protocol mechanisms
  - Reliable sequencing network service
  - Unreliable network service
- UDP
- TCP
  - Services
  - Header format
  - Mechanisms
  - Implementation policy options