TCP Slow Start and Congestion Avoidance lower the data throughput drastically when segment loss is detected. Fast Retransmit and Fast Recovery have been designed to speed up the recovery of the connection, without compromising its congestion avoidance characteristics.

Fast Retransmit and Recovery detect a segment loss via duplicate acknowledgements. When a segment is lost, TCP at the receiver will keep sending ack segments indicating the next expected sequence number. This sequence number would correspond to the lost segment. If only one segment is lost, TCP will keep generating acks for the following segments. This will result in the transmitter getting duplicate acks (i.e. acks with the same ack sequence number).

TCP Connection begins with slow start. The congestion window grows from an initial 512 bytes to 70000 bytes.

Loss of a TCP segment

TCP segment (start sequence number = 100000) is transmitted

TCP segment (start sequence number = 100512) is transmitted
### Module Interfaces (TCP Fast Retransmit and Recovery)

<table>
<thead>
<tr>
<th>Client Node</th>
<th>Internet</th>
<th>Server Node</th>
</tr>
</thead>
</table>

#### TCP Segment (start sequence number = 101024)
- TCP segment (start sequence number = 101024) is transmitted
- TCP segment (start sequence number = 101536) is transmitted
- TCP segment (start sequence number = 102048) is transmitted
- TCP segment (start sequence number = 102560) is transmitted
- TCP segment (start sequence number = 103072) is transmitted
- TCP segment (start sequence number = 103584) is transmitted

#### TCP Segment (start sequence number = 100000)
- TCP segment (start sequence number = 100000) is delivered to the receiver

#### TCP Segment (start sequence number = 100512)
- TCP segment (start sequence number = 100512) is lost due to congestion in the network.

#### TCP Segment (start sequence number = 101024)
- TCP Segment with start sequence number 101024 is received. TCP realizes that a segment has been missed. TCP buffers the out of sequence segment as TCP cannot deliver out of sequence data to the application.
- TCP sends an acknowledgement to the Sender with the next expected sequence number set to 100512.

#### TCP Segment (start sequence number = 101536)
- TCP receives the next segment. This and the following out of sequence segments will be buffered by TCP.
- TCP sends another acknowledgement with the next expected sequence number still set to 100512. This is a duplicate acknowledgement.

#### TCP Segment (start sequence number = 102048)
- TCP keeps acknowledging the received segments with the next expected sequence number as 100512.
Module Interfaces (TCP Fast Retransmit and Recovery)

<table>
<thead>
<tr>
<th>Client Node</th>
<th>Internet</th>
<th>Server Node</th>
<th>EventStudio System Designer 6</th>
</tr>
</thead>
</table>

Fast Retransmit

TCP receives duplicate acks and it decides to retransmit the segment, without waiting for the segment timer to expire. This speeds up recovery of the lost segment.

- **ACK**
  - ack_num = 100512

Client receives acknowledgement to the segment with starting sequence number 100512.

- **ACK**
  - ack_num = 100512

First duplicate ack is received. TCP does not know if this ack has been duplicated due to out of sequence delivery of segments or the duplicate ack is caused by lost segment.

- **ACK**
  - ack_num = 100512

Second duplicate ack is received.

- **ACK**
  - ack_num = 100512

Third duplicate ack is received. TCP now assumes that duplicate acks point to a segment that has been lost.

TCP retransmits the missing segment i.e. the segment corresponding to the ack sequence number in the duplicate acks.

Fast Recovery

TCP Segment

- **ACK**
  - seq_num = 100512

Yet another ack is received, this will further inflate the congestion window.

- **ACK**
  - seq_num = 100512

Finally, the retransmitted segment is delivered to the server.

- **ACK**
  - seq_num = 100512

Now TCP acknowledges all the segments that it had buffered.

- **ACK**
  - seq_num = 100512

The cummulative TCP ack is delivered to the client.

Congestion Avoidance

Client closes TCP connection

Client to server TCP connection release

- **FIN**

Client sends a TCP segment with the FIN bit set in the TCP header.
Module Interfaces (TCP Fast Retransmit and Recovery)

Client Node | Internet | Server Node | EventStudio System Designer 6

Server receives the FIN
Server responds back with ACK to acknowledge the FIN
Client receives the ACK

Server to client TCP connection release

FIN
FIN is sent out to the client to close the connection
Client receives FIN
Client sends ACK
Server receives the ACK

ACK

ACK

This sequence diagram was generated with EventStudio System Designer (http://www.EventHelix.com/EventStudio).