FLOW CONTROL

Flow control balances the rate a producer creates data with the rate a consumer can use the data. Figure 4.5.1 shows unidirectional data transfer between a sender and a receiver; bidirectional data transfer can be deduced from the unidirectional process.

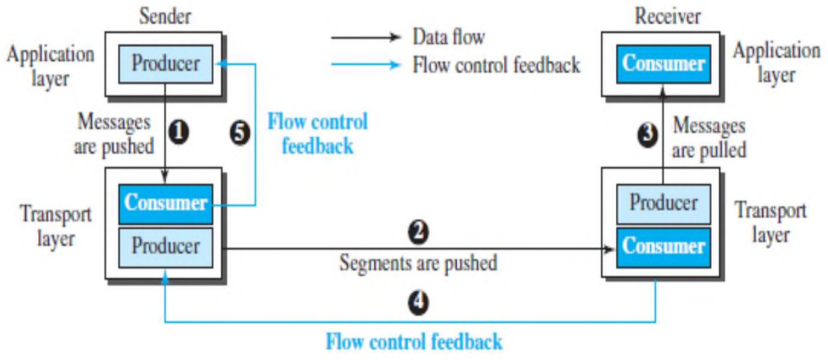

Fig4.5.1: Data flow and flow control feedbacks in TCP.
[Source : "Data Communications and Networking" by Behrouz A. Forouzan,Page-763]

The figure shows that data travel from the sending process down to the sending TCP, from the sending TCP to the receiving TCP, and from the receiving TCP up to the receiving process (paths 1, 2, and 3). Flow control feedbacks, are traveling from the receiving TCP to the sending TCP and from the sending TCP up to the sending process (paths 4 and 5). In other words, the receiving TCP controls the sending TCP; the sending TCP controls the sending process.Flow control feedback from the sending TCP to the sending process (path 5) is achieved through simple rejection of data by the sending TCP when its window is full.This means that our discussion of flow control concentrates on the feedback sent from the receiving TCP to the sending TCP (path 4).

Opening and Closing Windows

To achieve flow control, TCP forces the sender and the receiver to adjust their window sizes, although the size of the buffer for both parties is fixed when the connection is established. The receive window closes (moves its left wall to the right) when more bytes arrive from the sender; it opens (moves its right wall to the right) when more bytes are pulled by the process. We assume that it does not shrink (the right wall does not move to the left).The opening, closing, and shrinking of the send window is controlled by the receiver. The send window closes (moves its left wall to the right) when a new acknowledgment allows it to do so.

ERROR CONTROL

TCP is a reliable transport-layer protocol. This means that an application program that delivers a stream of data to TCP relies on TCP to deliver the entire stream to the application program on the other end in order, without error, and without any part lost or duplicated.

TCP provides reliability using error control. Error control includes mechanisms for detecting and resending corrupted segments, resending lost segments, storing out-of order segments until missing segments arrive, and detecting and discarding duplicated segments. Error control in TCP is achieved through the use of three simple tools: checksum, acknowledgment, and time-out.

Checksum

Each segment includes a checksum field, which is used to check for a corrupted segment.If a segment is corrupted, as detected by an invalid checksum, the segment is discarded by the destination TCP and is considered as lost. TCP uses a 16-bit checksum that is mandatory in every segment.

TCP uses acknowledgments to confirm the receipt of data segments. Control segments that carry no data, but consume a sequence number, are also acknowledged. ACK segments are never acknowledged.

CONGESTION CONTROL IN TCP

Congestion in a network may occur if the load on the network—the number of packets sent to the network is greater than the capacity of the network. Congestion control refers to the mechanisms and techniques to control the congestion.TCP uses a congestion window and a congestion policy that avoid congestion.If the network cannot able to deliver the data as fast as it is created by the sender, it must tell the sender to slow down.

Congestion policy in TCP

Slow Start Phase: starts slowly increment is exponential to threshold

1.Congestion Avoidance Phase: After reaching the threshold increment is by 1.

2.Congestion Detection Phase: Sender goes back to Slow start phase or Congestion avoidance phase.

Slow Start , exponential increase – In this phase after every RTT(round trip time) the congestion window size increments exponentially.

The slow-start algorithm is based on the idea that the size of the congestion window (cwnd) starts with one maximum segment size (MSS), but it increases one MSS each time an acknowledgment arrives. The sender starts with cwnd = 1. This means that the sender can send only one segment as in figure 4.5.2. After the first ACK arrives, the acknowledged segment is purged from the window, which means there is now one empty segment slot in the window. The size of the congestion window is also increased by 1 because the arrival of the acknowledgment is a good sign that there is no congestion in the network. The size of the window is now 2. After sending two segments and receiving two individual acknowledgments for them, the size of the congestion window now becomes 4, and so on.

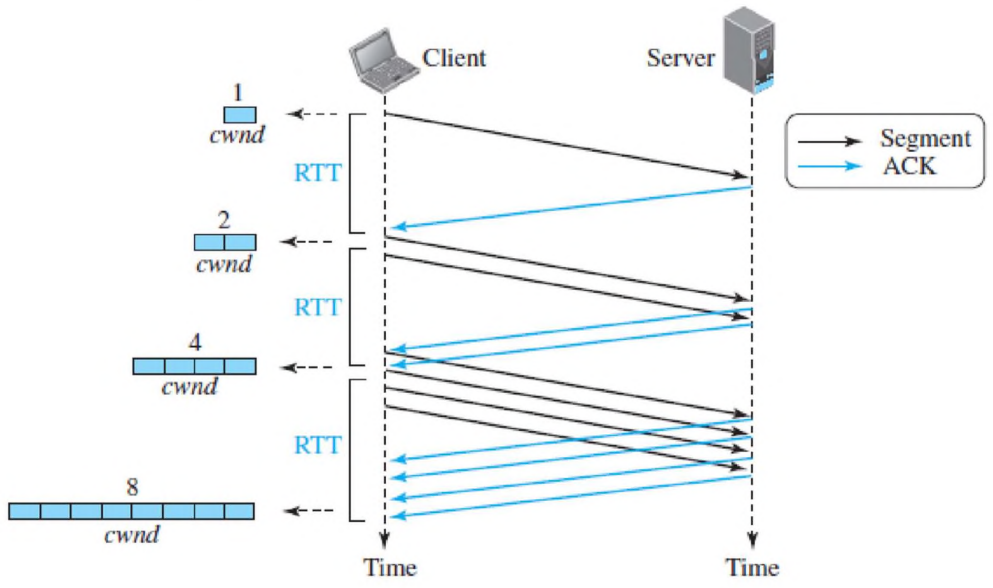

Fig4.5.2: Slow start exponential increase.

[Source : "Data Communications and Networking" by Behrouz A. Forouzan,Page-779]

Start	→	$cwnd = 1 \rightarrow 2^0$
After 1 RTT	→	$cwnd = cwnd + 1 = 1 + 1 = 2 \rightarrow 2^1$
After 2 RTT	→	$cwnd = cwnd + 2 = 2 + 2 = 4 \rightarrow 2^2$
After 3 RTT	→	$cwnd = cwnd + 4 = 4 + 4 = 8 \rightarrow 2^3$

Congestion Avoidance Phase : additive increase

To avoid congestion before it happens.This phase starts after the threshold value is denoted as ssthresh. The size of cwnd(congestion window) increases additive as shown in figure 4.5.3.When the size of the congestion window reaches the slow-start threshold in the case where cwnd =i, the slow-start phase stops and the additive phase begins.

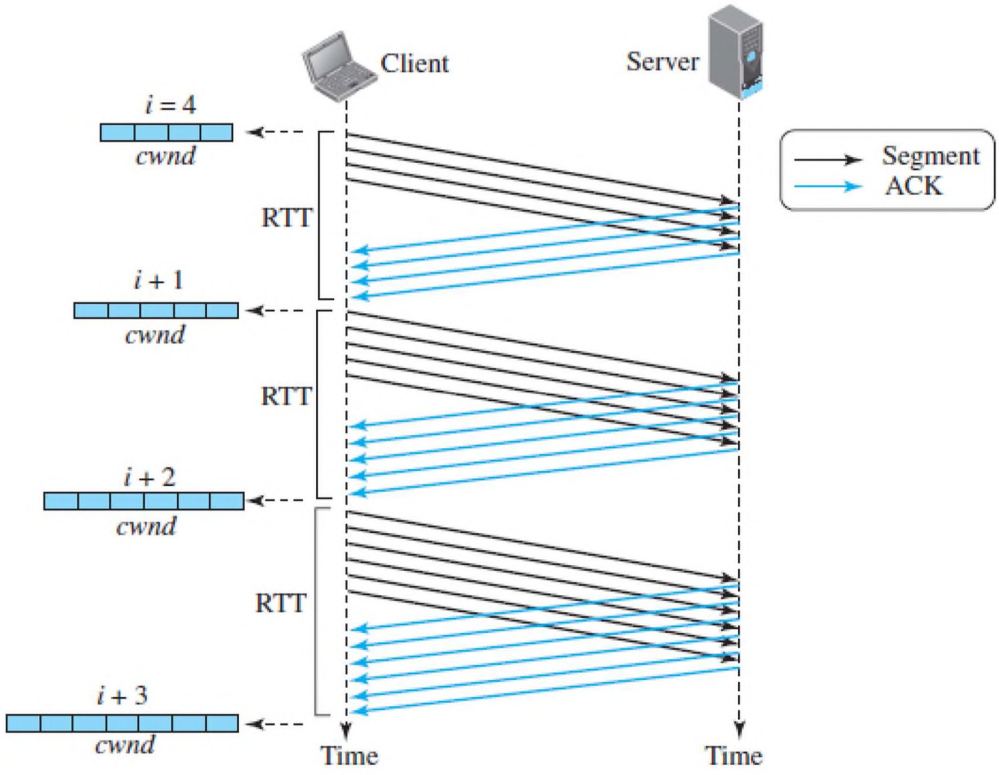

Fig4.5.3:TCP congestion avoidance.

[Source : "Data Communications and Networking" by Behrouz A. Forouzan,Page-780]

To avoid congestion before it happens.This phase starts after the threshold value is denoted as ssthresh. The size of cwnd(congestion window) increases additive as shown in figure 4.5.3.When the size of the congestion window reaches the slow-start threshold in the case where cwnd =i, the slow-start phase stops and the additive phase begins. In this algorithm, each time the whole "window" of segments is acknowledged, the size of the congestion window is increased by one. After each RTT cwnd = cwnd + 1.

Congestion Detection Phase : multiplicative decrement – If congestion occurs, the congestion window size is decreased. The only way a sender knows that congestion has occurred is the need to retransmit a segment. Retransmission is needed to recover a missing packet which is dropped by a router due to congestion.

Retransmission can occur in one of two cases: when the RTO timer times out or when three duplicate ACKs are received.

Case 1 : Retransmission due to Timeout . In this case congestion possibility is high.

A),ssthresh (Slow start threshold) is reduced to half of the current window size.

B).set cwnd = 1

C).start with slow start phase again.

Case 2 : Retransmission due to 3 Acknowledgement Duplicates .

In this case congestion possibility is less.

(a) ssthresh value reduces to half of the current window size.
(b) set cwnd= ssthresh
(c) start with congestion avoidance phase

Congestion Avoidance

TCP impose some methods to control congestion once it happens, instead of trying to avoid congestion.It is a prevention mechanism while congestion control is a recovery mechanism.

DECbit

DECbit means destination experiencing congestion bit. This mechanism was developed for use on the Digital Network Architecture (DNA), a connectionless network with a connection-oriented transport protocol.This mechanism could, therefore, also be applied toTCP and IP.It split the responsibility for congestion control between the routers and the end nodes.

Each router monitors the load it is experiencing and explicitly notifies the end nodes when congestion is about to occur. This notification is implemented by setting a binary congestion bit in the packets that flow through the router, hence the name DECbit.The destination host then copies this congestion bit into the ACK it sends back to the source. Finally, the source adjusts its sending rate so as to avoid congestion.

A router set this bit in a packet if its average queue length is greater than or equal to 1 at the time the packet arrives.This average queue length is measured over a time interval as , the last busy+idle cycle, plus the current busy cycle.The source records how many of its packets has set the congestion bit.If less than 50% of the packets had the bit set, then the source increases its congestion window byone packet. If 50% or more of the last window of packets had the congestion bit set, then the source decreases its congestion window to0.875 times the previous value.

Random Early Detection (RED)

RED provide congestion control at the router for TCP flows.RED was designed to work with TCP. Red notifies the sender by dropping packets. Packet dropping probability is increased as the average queue length increases. The moving average of the queue length is used to detect the long term congestion and allows short term bursts to arrive.

Properties of RED

1.RED drops packets before queue is full, in the hope of reducing the rates of some flows.

2.Drops packet for each flow roughly in proportion to its rate.

3.Red maintains average queue length.

4.Random drops desynchronize the TCP sources.

5.RED calculates the average que length using a weighted running average.

The Formula is as follows.

Average length = (1- Weight) x Average length + Weight x Sample length

Sample length is the queue length each time a packet arrives. The weight parameter is between 0 and 1.

RED has two queue length thresholds that trigger certain activity: MinThreshold and MaxThreshold.

When a packet arrives at the gateway, RED compares the current AvgLen with these two thresholds,according to the following rules:

ifAvgLen≤ MinThreshold

Then queue the packet

ifMinThreshold<AvgLen<MaxThreshold

calculate probability P

drop the arriving packet with probability P

ifMaxThreshold≤AvgLength

!drop the arriving packet

If the average queue length is smaller than the lower threshold, no action is taken, and if the average queue length is larger than the upper threshold, then the packet is always dropped. If the average queue length is between the two thresholds, then the newly arriving packet is dropped with some probability P.