Computer Networks - CS 204

Rishit Saiya - 180010027, Assignment - 2

April 15, 2020

1 Congestion Avoidance and Control

1. (a) **Conservation Principle**
   By 'conservation of packets' we mean that for a connection 'in equilibrium', i.e., running stably with a full window of data in transit. A new packet isn't put into the network until an old packet leaves. The physics of flow predicts that systems with this property should be robust in the face of congestion.

   The existence of stationary modes in queuing systems at a stationary entering flow induce some conservation laws between the quantities describing a state of the system.

   (b) **Ensuring TCP has followed Conservation Principle**
   TCP is based on a principle of 'Conservation of Packets' , i.e. if the connection is running at the available bandwidth capacity then a packet is not injected into the network unless a packet is taken out well.

   TCP implements this principle by using the acknowledgments to clock outgoing packets because an acknowledgment means that a packet was taken off the wire by the receiver. It also maintains a congestion window (CWD) to reflect the network capacity.

2. **Relation between Time Out and RTT**
   An underestimate of the RTT will result in an RTO computation that is less than the next measured RTT, causing unnecessary timeouts and retransmissions.

   Also, after several observations, we observed that a constant value as in multiple of RTT was too inflexible and and it failed to respond when the variance in RTT was high which is very common at heavy loads.

   Instead author has suggested to estimate the Retransmissions Timeouts in terms of RTT and not 2 * RTT so as to reduce the time lost in the retransmissions.

3. **Timeouts in Retransmissions**
   A TCP host must implement Karn's algorithm and Jacobson's algorithm for computing Retransmission Timeout (RTO).

- Jacobson's algorithm for computing the smoothed round- trip (RTT) time incorporates a simple measure of the variance.
- Karn's algorithm for selecting RTT measurements ensures that ambiguous round-trip times will not corrupt the calculation of the smoothed round-trip time.

4. (a) **Behaviour of TCP with and without Slow Start**
   Slow Start incorporates the fact that the maximum Bandwidth Delay Product is utilised till the packet loss has occurred. Till then we have an exponential increase in the graph of Sequence Number vs. Time. When the packet loss occurs, the bandwidth essentially reduces to half its value and then reiterates the same procedure of exponentially increasing the outgoing packets.

   Whilst the case where slow start isn't used, there will be a huge transmission of packets in the start itself manifesting to the fact that, if there is a loss of data packet in the process, duration of retransmission time will increase tremendously.

   (b) **Slow Start in TCP**
   TCP Slow Start is part of the congestion control algorithms put in place by TCP to help control the amount of data flowing through to a network. This helps to regulate the case where too much data is sent to a network and the network is incapable of processing that amount of data manifesting in network congestion.

5. There is no legitimate question.

6. **The Combined Slow Start and Congestion Avoidance Algorithm.**
   While algorithms at the transport endpoints can ensure the network capacity isn't exceeded, they cannot ensure fair sharing of that capacity. Only in gateways, at the convergence of flows, enough information to control sharing and fair allocation exists. Thus, we view the gateway 'congestion detection' algorithm as the next big step.

   The goal of this algorithm is to send a signal to the end nodes as early as possible, but not so early that the gateway become stacked with traffic. Since we plan to continue using packet drops as a congestion signal, gateway 'self protection' from a mis-behaving host should fall-out for free: That host will simply have most of its packets dropped as the gateway tries to tell it that it's using more than its fair share. Thus, like the endnode algorithm, the gateway algorithm should reduce congestion even if no endnode is modified to do congestion avoidance. And nodes that do implement congestion avoidance will get their fair share of bandwidth and a minimum number of packet drops.

   Since congestion grows exponentially, detecting it early becomes essential. If detected early, small adjustments to the senders' windows will cure it, otherwise massive adjustments are necessary to give the net enough spare capacity to pump out the backlog.

   Explaining mathematically, it comes to following relations:

   The sender keeps two state variables for congestion control: a slow- start/congestion window, cwnd, and a threshold size, ssthresh, to

switch between the two algorithms. The sender's output routine always sends the minimum of cwnd and the window advertised by the receiver. On a timeout, half the current window size is recorded in ssthresh (this is the multiplicative decrease part of the congestion avoidance algorithm), then cwnd is set to 1 packet (this initiates slow-start). When new data is acked, the sender does

```
if(cwnd < ssthresh)
    cwnd += 1
else
    cwnd += 1/cwnd
```

Thus slow-start opens the window quickly to what congestion avoidance thinks is a safe operating point (half the window that got us into trouble), then congestion avoidance takes over and slowly increases the window size to probe for more bandwidth becoming available on the path. Sending packets smaller than the maximum transmission unit for the path lowers efficiency,the implementor must take care that the fractionally sized cwnd does not result in small packets being sent.

2 A Protocol for Packet Network Intercommunication

1. (a) **Framing**
   Framing is a function of the data link layer. Framing is a point-to-point connection between two computers or devices consists of a wire in which data is transmitted as a stream of bits. However, these bits must be framed into discernible blocks of information.

   (b) **Necessity of Framing**
   It provides a way for a sender to transmit a set of bits that are meaningful to the receiver. Ethernet, token ring, frame relay, and other data link layer technologies have their own frame structures. Frames have headers that contain information such as error-checking codes.

2. **TCP addressing in TCP segment transmissions**
   A receiving TCP is faced with the task of demultiplexing the stream of internetwork packets it receives and reconstructing the original messages for each destination process. Each operating system has its own internal means of identifying processes and ports.

   We assume that 16 bits are sufficient to serve as internetwork port identifiers. A sending process need not know how the destination port identification will be used. The destination TCP will be able to parse this number appropriately to find the proper buffer into which it will place arriving packets. We permit a large port number field to support processes which want to distinguish between many different message streams concurrently. In reality, we do not care how the 16 bits are sliced up by the TCP's involved.

3. (a) **'ES' and 'EM' flags in TCP segment**
ES flag in the TCP segment denotes the End of the Segment.

EM flag in the TCP segment denotes the End of the Message.

(b) **Role in Reassembly and Sequencing**
The splitting of messages into segments by the TCP and the potential splitting of segments into the smaller pieces by GATEWAYS creates the necessity for indicating to the destination TCP when the ES has arrived and when the EM has arrived. The flag field of the internetwork header is used for this purpose.

The ES flag is set by the source TCP each time it prepares a segment for transmission. If it should happen that the message is completely contained in the segment, then the EM flag would also be set. The EM flag is also set on the last segment of a message, if the message could not be contained in one segment. These 2 flags are used by the destination TCP, respectively, to discover the presence of a check sum for a given segment and to discover that a complete message has arrived.

The ES and EM flags in the internetwork header are known to the GATEWAY and are of special importance when packets must be split apart for propagation through the next local network.

4. (a) **TCB and RCB**
TCP coordinates the activities of transmission, reception, and retransmission for each TCP connection through a data structure shared by all processes. The data structure is known as a ***Transmission Control Block or TCB***.

TCP coordinates the activities of receiving at the receiving end based on the at least data destination descriptor parameter for each TCP connection through a data structure shared by all processes. The data structure is known as a ***Receive Control Block or RCB***.

(b) **Purpose of TCB and RCB**
The TCB contains information necessary to allow the TCP to extract and send the process data. The work flow is as follows:

In order to send a message, a process sets up its text in a buffer region in its own address space, inserts the requisite control information (described in the following list) in a transmit control block (TCB) and passes control to the TCP.

To receive a message in its address space, a process sets up a receive buffer, inserts the requisite control information in a receive control block (RCB) and again passes control to the TCP.

(c) **Typical Format of a TCB**
The various fields in the TCB are described as follows:

i. Source Address: This is the full net/HOST/TCP/port address of the transmitter.

ii. Destination Address: This is the full net/HOST/TCP/port of the receiver.

iii. Next Packet Sequence Number: This is the sequence number to be used for the next packet the TCP will transmit from this port.

iv. Current Buffer Size: This is the present size of the process transmit buffer.

v. Next Write Position: This is the address of the next position in the buffer at which the process can place new data for transmission.

vi. Next Read Position: This is the address at which the TCP should begin reading to build the next segment for output.

vii. End Read Position: This is the address at which the TCP should halt transmission.

viii. Number of Retransmissions/Maximum Retransmissions: These fields enable the TCP to keep track of the number of times it has retransmitted the data and could be omitted if the TCP is not to give up.

ix. Timeout/Flags: The timeout field specifies the delay after which unacknowledged data should be retransmitted. The flag field is used for semaphores and other TCP/process synchronization status reporting, etc.

x. Current Acknowledgment/Window: The current acknowledgment field identifies the first byte of data still unacknowledged by the destination TCP.

5. (a) **Difference between Connection and Association**

The term connection has a wide variety of meanings. It can refer to a physical or logical path between two entities or it can refer to the flow over the path or it can inferentially refer to an action associated with the setting up of a path, or it can refer to an association between two or more entities as well. This is with or without regard to any path between them. The relationship between two or more ports that are in communication, or are prepared to communicate is defined to be an association. Ports that are associated with each other are called associates.

It is clear that for any communication to take place between two processes, one must be able to address the other. The two important cases here are that the destination port may have a global and unchanging address or that it may be globally unique but dynamically reassigned. While in either case the sender may have to learn the destination address, given the destination name, only in the second instance is there a requirement for learning the address from the destination (or its representative) each time an association is desired.

Only after the source has learned how to address the destination can an association be said to have occurred. But this is not yet sufficient. If ordering of delivered messages is also desired, both TCP's must maintain sufficient information to allow proper sequencing. When this information is also present at both ends, then an association is said to have occurred.

Only when both partners are prepared to communicate with each other has an association occurred, and it is possible that neither partner may be able to verify that an association exists until some data flows between them.

(b) **Connection Free Protocol with Association**
In the ARPANET, the Interface Message Processors (IMP's) do not have to open and close connections from source to destination. The reason for this is that connections are, in effect, always open, since the address of every source and destination is never reassigned. When the name and the place are static and unchanging, it is only necessary to label a packet with source and destination to transmit it through the network. According to our literature and hypothesis, every source and destination forms an association.

In the case of processes, however, it is evident that port addresses are periodically being used and reused. Some ever present processes could be assigned fixed addresses which do not change (e.g., the networks assigned to Military uses). If we assumed that every TCP had an infinite supply of port addresses so that no old address would ever be reused, then any dynamically created port would be assigned the next unused address. In such a situation, there could never be any confusion by source and destination TCP as to the intended recipient or implied source of each message, and all ports would be associates.