INST346 HW03

Transport Protocols, TCP/UDP, and Reliable Transfer

Cody Buntain

October 16, 2017

1 The Transport Layer

1. Suppose the network layer provides the following service. The network layer in the source host accepts a segment of maximum size 1,200 bytes and a destination host address from the transport layer. The network layer then guarantees to deliver the segment to the transport layer at the destination host. Suppose many network application processes can be running at the destination host.

   a) Design the simplest possible transport-layer protocol that will get application data to the desired process at the destination host. Assume the operating system in the destination host has assigned a 4-byte port number to each running application process.

   Solution Since the network layer in this case handles all of our reliable transmission, the transport protocol must only handle application multiplexing and demultiplexing. To perform this mux/demux, we construct a protocol that requires a destination address *and port* from the application layer. We then embed this destination port in a 4-byte header specific to our protocol. This four bytes of header is subtracted from the 1,200 byte MSS we can send, so our simple protocol can handle at most 1,196 bytes of application data in a single segment.

   On the receiver, we simply extract the 4-byte port number from our protocol's header and resolve this number to the proper application using the OS's 4-byte-number-to-application mapping.

   b) Modify this protocol so that it provides a "return address" to the destination process.

   Solution We simply add an additional 4-byte field our protocol's header that contains the source port, which identifies the application sending the data. This source port then operates as the return address for communication between two applications. This source port/return address is populated by the sender's OS when it creates its local 4-byte-number-to-application mapping.

2. Why do you think voice and video traffic are often sent over TCP rather than UDP in todays Internet?

   Solution HTTP, which runs on top of TCP, is already so ubiquitous on the Internet that virtually all security appliances (i.e., firewalls) are built to support HTTP traversal.

As a result, it is simpler to rely on HTTP/TCP for a particular application than designing a new protocol that many firewalls may or may not support.

At a more fundamental level, firewalls can much more easily track connections in TCP than in UDP and are thus much more likely to support TCP traversal and block UDP traffic.

TCP also provides congestion control, which supports a fair and equitable division of network resources. As a result, if many users are going to use a particularly bandwidth-intensive application, to provide a good experience for many users, TCP would be more ideal.

3. Suppose a process in Host C has a UDP socket with port number 6789. Suppose both Host A and Host B each send a UDP segment to Host C with destination port number 6789.

   a) Will both of these segments be directed to the same socket at Host C? If so, how will the process at Host C know that these two segments originated from two different hosts?

   Solution Yes, both segments will be directed to the same socket. For each received segment, at the socket interface, the operating system will provide the process with the IP addresses to determine the origins of the individual segments.

   b) How does this process differ in TCP?

   Solution In TCP, these packets will only be directed to the same socket *if* they are both being sent by systems initiating new connections. If either host is already participating in a TCP connection to port 6789 on the receiver, TCP's three-way handshake and welcome/client socket mechanisms will split the connections into separate sockets, one for each host.

2 Principles of Reliable Data Transfer

1. In our RDT protocols, why did we need to introduce checksums?

   Solution To detect packet corruption during transmission.

2. In our RDT protocols, why did we need to introduce sequence numbers?

   Solution Sequence numbers are required for a receiver to determine whether an arriving packet contains new data or is a retransmission, to support re-ordering, and provide some information about potentially dropped packets.

3. In our RDT protocols, why did we need to introduce acknowledgements?

   Solution Acknowledgements of some form are necessary to provide feedback to the sending host, so the sending host can know whether packets were successfully received.

4. In our RDT protocols, why did we need to introduce timers?

   Solution Timers were introduced to detect lost packets. If the ACK for a transmitted packet is not received within the duration of the timer for the packet, the packet (or its ACK or NACK) is assumed to have been lost. Hence, the packet is retransmitted.

5. Suppose that the roundtrip delay between sender and receiver is constant and known to the sender. Would a timer still be necessary in protocol RDT 3.0, assuming that packets can be lost? Explain.

   Solution A timer would still be necessary as we must still detect when packet loss occurs. If the round trip time is known, the only advantage will be the sender knows for sure that either the packet or its ACK has been lost. In the real world, we have to estimate these values, which results in duplicate transmissions for prematurely expired timers.

6. Consider the cross-country pipeline motivating example shown in section 3.4.2. Suppose that the size of a packet is 1,500 bytes, including both header fields and data.

   a) How big would the window size have to be (i.e., how many packets should be in flight simultaneously) for the 1Gbps channel utilization to be greater than 98 percent?

   Solution For this problem, you can use the link utilization equation in the book, section 3.4.2, page 219, which we want to be greater than or equal to 0.98 (or 98%). We must modify this equation to allow for sending multiple packets, so we add a factor w for our window size:

$$U_{sender} = \frac{w \cdot L/R}{RTT + L/R} \geq 0.98 \tag{1}$$

   We know $L = 1,500$ bytes $= 12,000$ bits from the question and $R = 1$ Gbps and $RTT = 30$ milliseconds from the book (observant students may have noted here that 30 milliseconds is not a good estimate of the RTT between LA and NYC, but we'll follow the book here anyway). After putting all the values in this equation into consistent units, we can solve for w, the window size:

$$\begin{aligned}
U_{sender} &= \frac{w \cdot 12,000 \text{ bits}/1,000,000,000 \text{ bits / second}}{0.03 \text{ seconds} + 12,000 \text{ bits}/1,000,000,000 \text{ bits / second}} \geq 0.98 && (2)\\
&\Rightarrow \frac{w \cdot 0.000012 \text{ seconds}}{0.03 \text{ seconds} + 0.000012 \text{ seconds}} \geq 0.98 && (3)\\
&\Rightarrow \frac{w \cdot 0.000012 \text{ seconds}}{0.030012 \text{ seconds}} \geq 0.98 && (4)\\
&\Rightarrow w \cdot 0.000012 \text{ seconds} \geq 0.02941176 \text{ seconds} && (5)\\
&\Rightarrow w \geq 2,450.98 && (6)
\end{aligned}$$

   So our window size w must be at least 2,451 packets wide to achieve 0.98 link utilization.

   b) Assume the 100th packet in your window was corrupted. After receiving a negative ACK, how many packets would be retransmitted by the sender using i) go-back-n, and ii) selective repeat?

   Solution For GBN, we would have to transfer ALL packets after the 100th packet, which would be 2,351 packets. In SR, however, we would only need to retransmit the single packet that was corrupted.

c) Suppose your protocol only had room for 12-bit sequence numbers. Would you use go-back-n or selective repeat as your pipeline approach? Why?

d) Given only 12 bits, with SR, our window size would be restricted to $2^{11} = 2,048$ bytes (because SR's window size is restricted to half of our sequence number space, or $2^{12}/2 = 2^{11}$). 2,048 is less than the window size we need to maintain 98% utilization, so i would use GBN here, which allows for a larger window.

3 Fundamentals of TCP

1. Answer each of the following as **True** or **False**:

   a) Host A is sending Host B a large file over a TCP connection. Assume Host B has no data to send to Host A. Host B will not send acknowledgements to Host A because Host B cannot piggyback the acknowledgements on data.

   Solution False

   b) The size of the TCP `rwnd` never changes throughout the duration of the connection.

   Solution False

   c) Suppose Host A is sending Host B a large file over a TCP connection. The number of un-ACKed bytes that A sends cannot exceed the size of the receive buffer.

   Solution True except for when the receive buffer has an advertised size of 0, in which case, Host A will send 1 byte to avoid deadlock.

   d) Suppose Host A is sending Host B a large file over a TCP connection. If the sequence number for a segment of this connection is m, then the sequence number for the subsequent segment must be $m + 1$.

   Solution False

   e) The TCP segment has a field in its header for the receive window.

   Solution True

   f) Suppose the last `SampleRTT` in a TCP connection is equal to 1 second. The current value of TimeoutInterval for the connection will necessarily by ≥ 1 sec.

   Solution False, as SampleRTT and EstimatedRTT could be set such that EstimatedRTT is much lower than SampleRTT, so 0.875 x EstimatedRTT + 0.125 SampleRTT could be much less than (1-4DevRTT), which would create a TimeoutInterval < 1.

   g) Suppose Host A sends one segment with the sequence number 38 and 4 bytes of data over a TCP connection to Host B. In this same segment, the acknowledgement number is necessarily 42.

   Solution False

2. Suppose Host A and Host B negotiate a new TCP connection. Host A then sends two TCP segments back to back to Host B over this new connection. The first segment has sequence number 90; the seconds has sequence number 110.

   a) How much data is in the first segment?

Solution 20 bytes

b) Suppose the first segment is lost but the second segment arrives at B. In the acknowledgement that Host B sends to Host A, what will be the acknowledgement number?

Solution Host B will send an ACK for sequence number 90.

3. Say Host A wants to send a large file of L bytes to Host B. Assume an MSS of 1,460 bytes.

a) What is the maximum size L of this file such that the TCP sequence numbers are not exhausted? Recall that the TCP sequence number field has 4 bytes.

Solution TCP has a 4-byte sequence number space, meaning it can uniquely identify 32 bits worth of bytes. That makes for a max of 2^{32} bytes $\approx$ 4 gigabytes.

b) For the L you obtained, how long does it takes to transmit the file? Assume that a total of 40 bytes of transport and network header are added to each segment before the resulting packet is sent out over a 155 Mbps link. Ignore flow control and congestion control so A can pump out the segments back to back and continuously.

Solution We have a data size of 4,294,967,296 bytes, and with an MSS of 1,460 bytes, this results in 2,941,759 packets. We then account for the additional 40 bytes of header space to create the 1,500-byte MTU. This translates into 4,412,638,500 bytes to send along this 155 Mbps link. Converting 155 Mbps to bytes yields 19,375,000 bytes per second.

Dividing 4,412,638,500 bytes by this 19,375,000 bytes per second yields 227.75 seconds.