Assignment 7: TCP and Congestion Control

Due the week of October 29/30, 2015

I'd like to complete our exploration of TCP by taking a close look at the topic of congestion control in TCP. To prepare for this, you should read the discussion of congestion control presented in §6.1 - §6.4 or our text. There are several additional readings.

The early part of the readings from the text together with some of the material in §6.4.2 of the text describe techniques that can be used in individual routers to address congestion issues. The RFC "Recommendations on Queue Management and Congestion Avoidance in the Internet" discusses the relationships between and significance of these techniques.

At the end of §6.4, Peterson and Davie discuss a proposed alternative to the standard TCP congestion control mechanisms. This scheme, known as TCP Vegas, was actually first described in a paper authored by Peterson and a student named Lawrence Brakmo. The paper, *TCP Vegas: New Techniques for Congestion Detection and Avoidance*, will be one of our readings this week. The techniques the paper describes are interesting in their own right. In addition, by providing a contrast to the techniques employed by standard TCP implementations, learning about TCP Vegas will help us understand current TCP congestion control mechanisms more fully.

Exercises

1. Complete problem 6.14 part (a) from the text. Stop when you get to wall clock time 10. For each tick of the wall clock show the value of the virtual clock used to determine A_i, which packets arrive, the F_i value assigned to each packet that arrived, the packet (if any) sent, and the packets in the queue at each node. Identify packets by host/arrival time.

   As you work this problem, think about what data structures would be used to implement the process efficiently and come prepared to discuss such an implementation.

2. This is another question borrowed from the Cornell instructor who produced one of the traces we looked at using Wireshark. The trace file mentioned can be found in ~tom/shared/336/traces.

   Load the trace file trace4.cap in Wireshark and answer the following questions. For this trace a stream of TCP packets were sent from orkid to orchid for 1 second. The delay from orkid to orchid was set to be 10 ms using NistNet. The complete Ethernet bandwidth of 10 Mbps was available between the two nodes.

   (a) Plot the time sequence graph and zoom in to the first 200ms of the trace. What do you see? Give approximate sizes for the first 5 send windows and explain your observation briefly in words.

   (b) Plot the time sequence graph and look at the complete trace. The curve is not a straight line, as you would have observed in other traces. Instead we have 4 similarly shaped segments in the curve. What events do you believe caused each of these kinks in the curve? Does your explanation precisely predict the number of packets sent in each burst? If not, identify the anomalies you can't quite explain.

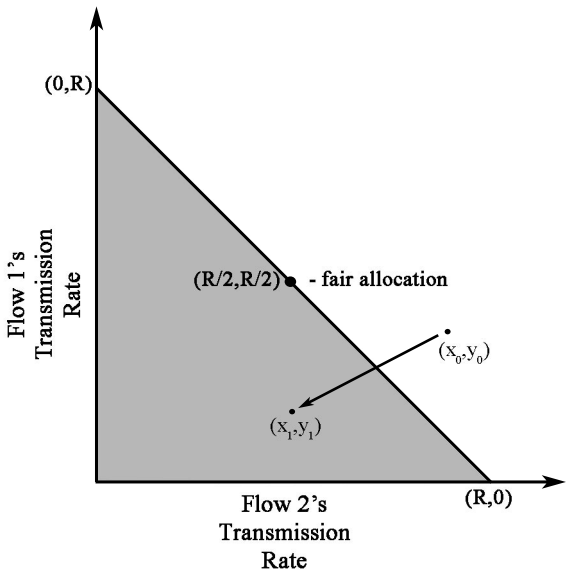

Figure 1: Configurations of two sender feedback ssytem.

3. The text states that "It has been shown that additive increase/multiplicative decrease is a necessary condition for a congestion-control mechanism to be stable." For this question, I would like you to investigate some alternatives to additive increase/multiplicative decrease to help understand its advantages.

   Additive increase and multiplicative decrease are two examples of linear window size update mechanisms. In general, If W_i is the congestion window size at time i, after the window size is adjusted by a linear mechanism, we will have $W_{i+1} = AW_i + B$ for some constants A and B. In the case of additive increase/multiplicative decrease, the adjustment made when congestion is detected uses a value of A between 0 and 1 and sets B to zero. On the other hand, when a full window's worth of packets is acknowledged, the update formula uses 1 as the value of both A and B.

   To understand the behavior of such a scheme, consider the simple case where just two computers using this approach are sharing a single bottleneck link. Assuming that the capacity of the bottleneck link is R, the diagram shown in Figure 1 can be used to illustrate the way in which the update process will adjust these two flows of packets.

   In this figure, the point (x,y) is used to represent the state of the system when the flow from the first computer has a window size that allows it to transmit at the rate y and the flow from the second computer has a window size that allows it to transmit at rate x. The transmission rate of a station using TCP congestion control will equal its window size divided by the round trip time. Assuming the units used to measure time are such that RTT = 1, it is fair to treat x and y interchangeably as rates or window sizes. We will take this liberty in the following explanation.

   The shaded triangle includes the points corresponding to all states for which the sum of the packets transmitted by the two flows does not exceed R, the capacity of the bottleneck link. The hypotenuse of this triagle corresponds to the set of states in which the link is maximally utilized

without exceeding its capacity. Different points on this line, however, represent different degrees of fairness. At the point (0,R), the link is fully utilized, but only flow 1 is allowed to send packets. Similarly, at point (R,0) only flow 2 can transmit. The midpoint of the hypotenuse, $(R/2, R/2)$ is therefore the optimal operating state for the system. At this point, the link would be fully utilized and the allocation of bandwidth to the two senders would be fair.

A feedback mechanism like additive increase/multiplicative decrease aims to move the system from one state to another with the goal of approaching this optimal state as closely and as quickly as possible. For example, if the system began in the state corresponding to the point (x_0, y_0) shown in the figure, packets would eventually be lost (since the sum of the flows exceed the link capacity). When the flows detected this loss, they would update their window sizes yielding new window sizes $x_1 = Ax_0 + B$ and $y_1 = Ay_0 + B$. This would correspond to the transition of the system state from (x_0, y_0) to (x_1, y_1) shown in Figure 1. The overall behavior of the systems would be determined by the complete series of such transitions it performed.

(a) One possible alternative to additive increase/multiplicative increase would be additive increase/ additive decrease. In this case, when a packet is lost, the window would be updated using the formula $W_{i+1} = W_i - B_d$. When a full window of packets is sent without loss, the update performed would be $W_{i+1} = W_i + B_i$. Typically, the constants would be selected so that $B_d > B_i$.

Assuming that the round-trip times for the two flows are the same and that when packets are lost both flows suffer and detect loss, all the states (x_i, y_i) that can ever be reached from some initial state (x_0, y_0) fall on a line that passes through (x_0, y_0). Find the slope and y-intercept of this line.

(b) Another alternative scheme would be to use multiplicative increase and multiplicative decrease. In this case, when congestion is detected, the window size update would be computed using $W_{i+1} = A_d W_i$, and after a full window of packets was delivered without loss, the update would be $W_{i+1} = A_i W_i$ with $0 < A_d < 1 < A_i$ and $A_i A_d < 1$.

Under the same assumptions about round-trip time and loss detection made in part (a), show that all states (x_i, y_i) that can be reached from a given initial state (x_0, y_0) using this scheme fall on a line that passes through (x_0, y_0). Find the slope and y-intercept of this line.

(c) Assuming a fixed round-trip time, the congestion window size used by a flow is proportional to the rate at which it sends data and is therefore proportional to its throughput. In situations, like those described in parts (a) and (b), where one flow's window size is a simple function of the other's, it becomes possible to express the fairness of the state of the system as a function of the window size of one flow. Please derive such a formula for fairness as a function of flow 2's window size (i.e. fairness as a function of "x_i") for both:

- additive increase/additive decrease, and
- multiplicative increase/multiplicative decrease

using Jain's fairness index as the measure of fairness.

Determine how each of these functions behaves as the window sizes change? That is, does the fairness index increase, decrease or remain the same as the window sizes of the two flows are increased?

(d) Given your observations concerning the behavior of the fairness index from part (c), explain qualitatively how the fairness index will behave as a function of time when the additive increase/ multiplicative decrease procedure actually incorporated in TCP is used to adjust window sizes.

(e) For all of the subparts of this question, we have assumed that the round-trip times associated with the two flows are equal. This is unrealistic for the Internet where connection lengths may vary greatly. Suppose that flow 1's connection's round-trip time is three times as long as flow 2's. In this case, each time flow 1 succeeds in delivering one window's worth of packets, flow 2 will deliver three windows worth of packets. Therefore flow 2 will apply the "increase" part of any window size adjustment algorithm three times as often as flow 1.

How would the results you obtained for the additive increase/additive decrease combination in parts (a) and (c) change in this case? In your answer, assume that flow 2 applies both the "increase" and "decrease" adjustments 3 times as often as flow 1. Would the set of states the system could reach still be a line? If so, what would its slope be? What value will the fairness index for the system approach as the window size grows?

What does this suggest about the fairness of additive increase/multiplicative decrease in TCP when flows with different round-trip times interact?

(f) If you have been paying attention carefully all semester, you should realize that this is not the first time we have encountered the use of additive-increase/multiplicative-decrease in a feedback mechanism. What other protocol proposal that we have studied used this technique? Is there a similar argument supporting the assumption that the use of AIMD in this other context will provide convergence to fairness?

4. The diagram from our textbook shown below illustrates the adjustments made to the TCP congestion window under the additive increase/multiplicative decrease strategy (AIMD). This diagram is an idealized representation of actual TCP behavior. Among other things, it suggests that each computer immediately resumes additive increase after applying multiplicative decrease due to a packet loss. This approximates actual behavior when all loses are detected using duplicate ACKs, but is far from accurate when timeouts occur. For this problem, however, I want you to assume that it does accurately reflects the behavior of a single TCP connection.

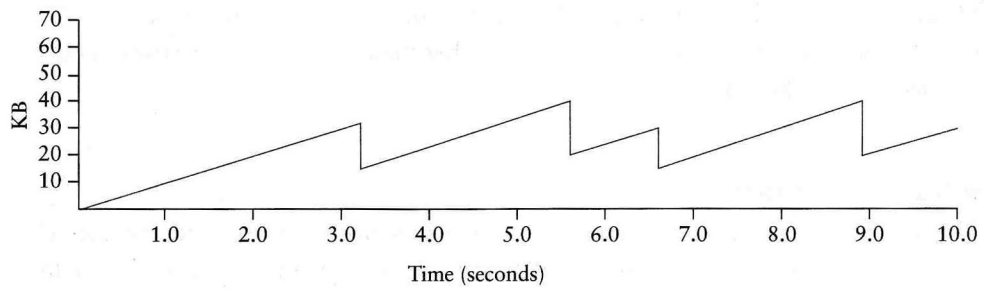

Figure 6.9 Typical TCP sawtooth pattern.

What I want you to explore in this question is what happens when we consider the traffic a router actually processes as many flows independently follow this sawtooth pattern in response to packets dropped by the router. In particular, suppose that the F distinct flows are sending packets through a single router.

We will assume that all of the flows send packets of the same size and that all flows have the same round trip time, RTT. We will also assume that variations in the round trip time due to changes

in queuing delay are negligible so that RTT can be considered constant. Given this assumption, we will measure all transmission rates in packets per RTT. This includes the individual flow rates, R_i, the combined transmission rate of all flows, $R = \sum_{i=1}^{F} R_i$, and the router's total capacity, C. Measured in this way, a flow's rate R_i will be equal to its congestion window size measured in packets. Therefore, assuming no packet loss, each R_i will increase by 1 every RTT.

Earlier in this assignment, you showed that under the assumption of equal RTT, the AIMD process is designed to adjust congestion windows in such a way that the system approaches fair allocation and full utilization. That is, we would hope that under AIMD, the system described would tend toward a point where $R_i = C/F$ for all i. This suggest that to make this interesting we should assume $C >> F$.

Important note:

> The goal of this problem and all of the simplifying assumptions made in its statement is to get a good approximate description of the idealized behavior of AIMD in certain scenarios. This means you should take the word "approximately" seriously whenever it appears.

(a) Suppose that at some time t_0 this system reaches the ideal point where $R_i = C/F$ for all i, and none of the router's buffers are occupied. No packets will need to be dropped, so over the next RTT, all flows will increase their sending rate by 1. As a result, at time $t_0 + RTT$, the combined rate R will equal $C + F$, and F of router's buffers will be full.

If the router has enough buffers to continue processing packets without any drops until time $t_0 + 2RTT$, what will R be at this time and how many packets will be buffered.[1]

(b) Now, assume instead that the router actually has only F packet buffers. As a result, immediately after $t_0 + RTT$ the router will begin dropping packets. Approximately how many packets will have to be dropped before the application of multiplicative decrease sufficiently reduces the combined flow rate so that the number of buffers required is $\leq F$. In your answer, assume that the router is lucky (or clever) and drops packets from as many distinct flows as possible (rather than dropping many packets from a few flows and none from others).

(c) Under the assumptions stated in part (b), approximately what will be the combined flow rate, R, at time $t_0 + 2RTT$.

(d) After the events described in part (b) and (c), the flows will resume AIMD and eventually reach the state where the combined rates of all flows exceed C leading to another, similar round of packet dropping. As long as all flows remain active, the system will repeat this cycle indefinitely. How long will each cycle last (measured in RTTs) and what will be the average number of packets dropped per RTT averaged over many such cycles?

(e) Give a formula for the average efficiency of a router operating in this cycle. Your answer should account for inefficiency due to both packet loss and periods where the combined rates of all flows are less than the router's capacity. Apply this formulas and your formula from part (d) to the case where $C = 100$ and $F = 10$.

(f) Recalling the advice in the introduction to this problem that you should take the word approximately seriously, consider how the results would differ if the flows passing through the router were all using TCP Vegas rather than TCP Reno.

[1] The assumption that RTT is fixed might become questionable at this point with many packets buffered, but please continue to make this assumption to keep things tractable.

Assume as we did in part (a) that the system magically arrives at a state where the flows are fairly sharing the full capacity of the router at time t_0. Also assume that all flows started when all the routers on the paths used were idle and that there are enough buffers in the router that no packets will have to be dropped. Finally, in the earlier parts of this problem, we simplified things by assuming that the router was lucky enough to spread dropped packets among the flows equally. With Vegas, you should instead assume that the router is lucky enough to spread any queuing delays equally among the flows. That is, you should assume that each flow will observe the same delivery rate.

Given these assumptions, what (if any) cycle would you expect the flow rates to follow in the future.

(g) Now, still assuming that we are using Vegas rather than Reno, suppose that at time $t_0 + \text{RTT}$ another flow (flow F+1) begins to transmit packets. Assuming the Vegas mechanism designed to end slow start before congestion begins moves flow F+1 into additive increase before its rate exceeds its fair share, will the system eventually reach a state where all the flows divide the capacity of the network fairly among the F+1 flows? If not, what will happen?