Communication Networks 2
Exercise 2 - TCP and UDP

Multimedia Communications Lab
TU Darmstadt

Problem 1 Flow Control vs Cogestion Control

Describe the concept of Flow Control. Also describe the concept of Congestion Control. What are the differences between both principles?

Solution:

a) **Flow Control:** Protect slow receiver from getting flooded by a fast sender.

b) **Congestion Control:** Protect the network from getting overloaded

See Computer Networks, 5th Edition, P 281ff

Problem 2 Rate Limit

How could a TCP sender limit the rate at which it sends traffic into its connection?

Solution:
Each side of a TCP connection consists of a receiver buffer, a send buffer and several variables (Last Byte Read, Last Byte Acknowledged, rwnd, and so on). The congestion control mechanism operating at the sender keeps track of an additional variable, the **congestion window**. The congestion window, denoted *cwnd* imposes constraint on the rate at which a TCP sender can send traffic into the network. Specifically, the amount of unacknowledged data at the sender may not exceed the $cwnd$. Thus, the congestion window limits the amount of unacknowledged data on the sender buffer and therefor indirectly limits the sender's send rate.

Consider a connection for which loss and packet transmission delays are negligible. Then, roughly at the beginning of every RTT, the constraint permits the sender to send $cwnd$ bytes of data into the connection.

Problem 3 Cogestion Indication

How could a TCP sender perceive that there is congestion on the path between itself and the destination?

Solution:
We define a loss event at a TCP sender at the occurence of either a timeout or the receipt of three duplicate ACKs from the receiver. When there is excessive congestion on the path from the sender to the receiver, then one or more router buffers overflow causing a datagram to be dropped. The dropped datagram, in turn, results in a loss event at the sender - either a timeout or the receipt of three duplicate ACKs - which is taken by the sender to be an indication of congestion on the sender to receiver path.

Problem 4 Bandwith Probing

How does TCP probe for the maximal available bandwidth?

Solution:
After a slow start phase with an exponential increase of the $cwnd$, TCP probes by additive increase for the maximum possible throughput. TCP increases the $cwnd$ by $1MSS$ every RTT until congestion occurs. This situation is indicated by a triple acknowledgment of the same segment. In this case, the $cwnd$ is divided by two in order to reduce the throughput.

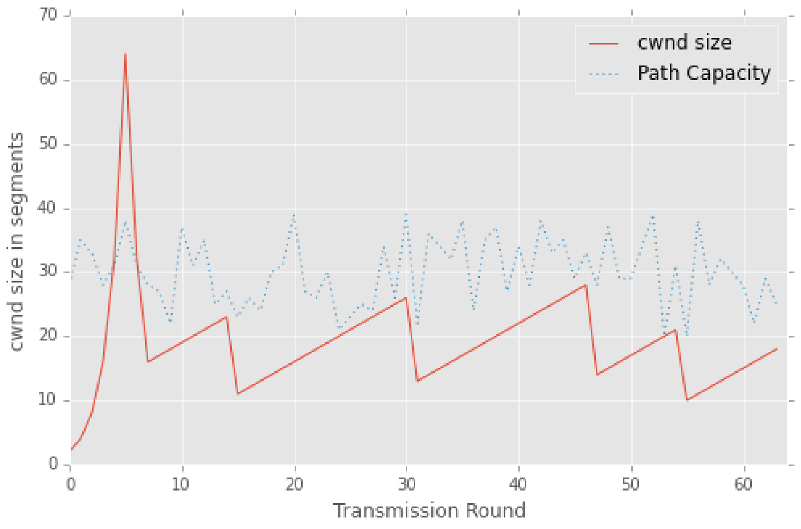

Figure 1: Probing of TCP's congestion window

This behavior is shown in Figure **??**. The Path capacity indicates the borderline at which a triple acknowledgment of the same segment arrives at the TCP sender.

Problem 5 Average Throughput

What is the average throughput of a TCP connection in steady state? (Assume no slow start is happening and a constant path capacity)

Solution:
In steady state, the rate at which TCP sends data is a function of the congestion window and the current RTT. When the window size is w bytes and the round trip time is RTT seconds, then TCP's transmission rate is roughly $\frac{w}{RTT}$. However, as TCP probes for additional bandwidth by additive increase until a duplicate ACK occurs and reacts to those duplicate ACK by halving the window size. Denoted by W, the value of w when a duplicate ACK occurs, the transmission rate of the TCP sender ranges linear from $\frac{1}{2}\frac{W}{RTT}$ to $\frac{W}{RTT}$.

These assumptions lead to the highly simplified macroscopic model for the steady state behavior of TCP. In average the send rate of a connection is $rate = \frac{3}{4}\frac{W}{RTT}$

Problem 6 Cogestion Control Phases

What are the three states of TCP congestion control?

Solution:

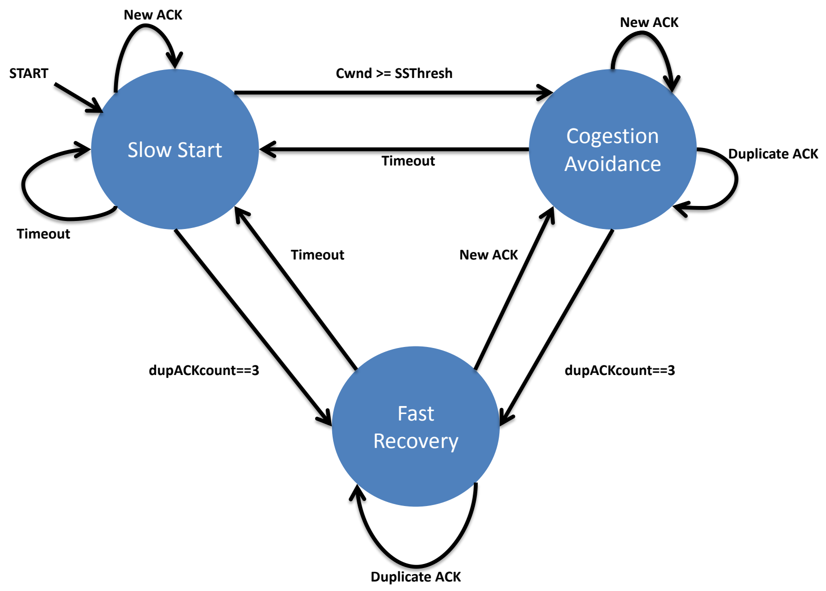

Figure 2: Finite State Machine Description of TCP congestion control

Problem 7 Slow Start

In slow start phase, what is the initial value of the congestion window? What is the increment rate for this window?

Solution:
When a TCP connection begins, the value of *cwnd* is typically initialized to a small value of 1 MSS, resulting in an initial sending rate of roughly MSS/RTT. Since the available bandwidth may be much larger than MSS/RTT, the TCP sender would like to find amount of available bandwidth quickly. Thus, in the **slow start** state, the value of cwnd begins at 1 MSS and increases by 1 MSS every time a transmitted segment is first acknowledged. This process results in a doubling of the sending rate every RTT. Thus, the TCP send rate starts slow but grows exponentially during the slow start phase.

Problem 8 Slow Start-Cont.

In slow start phase, when should the exponential growth of *cwnd* end?

Solution:
Slow start provides several answers to this question:

- if there is a loss events (i.e. congestion) indicated by a timeout, the TCP sender sets the value of *cwnd* to 1 and begins the slow start process anew. It also sets the value of a second state variable $ssthresh$ (shorthand for slow start threshold) to $cwnd/2$ - half of the value of the congestion windows when congestion was detected.

- The second way in which slow start may end is directly tied to the value of $ssthresh$. Since $ssthresh$ is half of the value, when congestion was last detected, it might be a bit reckless to keep doubling $cwnd$ when it reaches or surpasses the value of $ssthresh$. Thus, when the value of $cwnd$ equals $ssthresh$, slow start ends and TCP transitions into congestion avoidance mode.

- The final way in which slow start can end is if three duplicate ACKs are detected, in which case TCP performs a fast retransmit and enters fast recovery state.

Problem 9 Congestion Avoidance

In congestion avoidance phase, what is the initial value of the congestion window and what is the increment rate for this window?

Solution:
On entry to the congestion avoidance state, the value of $cwnd$ is approximately half its value when congestion was last encountered. Thus, rather than doubling the value of $cwnd$ each RTT, TCP adopts a more conservative approach and increases the value of $cwnd$ by just a single MSS every RTT.

Problem 10 Congestion Avoidance 2

In congestion avoidance phase, when should the congestion avoidance's linear increase (of 1 MSS per RTT) end?

Solution:
The congestions avoidance state behaves the same as slow start when a timeout occurs: The value of $cwnd$ is set to 1 MSS and the value of $ssthresh$ is updated to half the value of cwnd when the loss event occurred. Furthermore, TCP transitions to slow start. Recall, however, that a loss event also can be triggered by a triple duplicate ACK event. In this case, the network is continuing to deliver segments from sender to receiver. So TCP's behavior to this type of loss event should be less drastic, than within a timeout indicated loss. TCP halves the value of $cwnd$ and records the value of $ssthresh$ to be the half the value of cwnd when the triple duplicate ACK was received. The fast recovery state is then entered.

Problem 11 Fast Recovery

Wwhat happens in fast recovery phase? Please give a detailed explaination.

Solution:
In fast recovery state, the value of $cwnd$ is increased by 1 MSS for every duplicate ACK received for the missing segment that caused TCP to enter the fast recovery state. Eventually, when an ACK arrives for the missing segment, TCP enters the congestion avoidance state after deflating $cwnd$. If a timeout event occurs, fast recovery transitions to the slow start state after performing the same actions as in slow start and congestion avoidance: The value of cwnd is set to 1 MSS and the value of $ssthesh$ is set to $cwnd/2$ when the loss event occurred. Fast recovery is a recommended but not required component of TCP.

Problem 12 Congestion Control vs Stop And Wait

There exists a stop-and-wait algorithm for reliable data transmissions which sends a segment and waits for its corresponding acknowledgment before sending the next segment. Discuss main disadvantages of this stop-and-wait algorithm in comparison to *cwnd* based approaches.

Solution:

Figure **??** shows the sequence diagram of a stop and wait algorithm for reliable data transmission. In this case, the sender sends a segment and then stops and wait. The sender waits until it receives an acknowledgment or a timeout occurs. In case of a timeout, the sender sends the segment again.

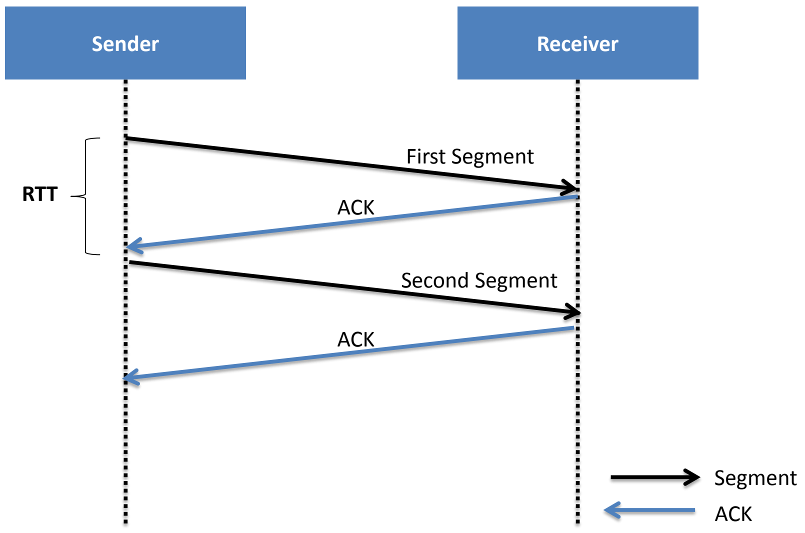

Figure 3: Sequence diagram of the stop-and-wait algorithm

If no loss event occurs, the stop and wait algorithm can send a segment every RTT. Thus, the send rate is $rate = \frac{seg\_size}{RTT}$. As the Sender cannot influence the RTT of the network, it has to increase the segment size to achieve a higher throughput. As each network has a certain, maximum segment size, the stop and wait algorithm might not achieve a good send rate.

Due to this reason, modern reliable data transmission algorithms rely on a pipelined approach. Those senders send up to N segments in parallel before waiting for acknowledgments of the transmitted data. Thus, the maximal achievable send rate is N times higher than for the stop and wait algorithm: $rate = N * \frac{seg\_size}{RTT}$

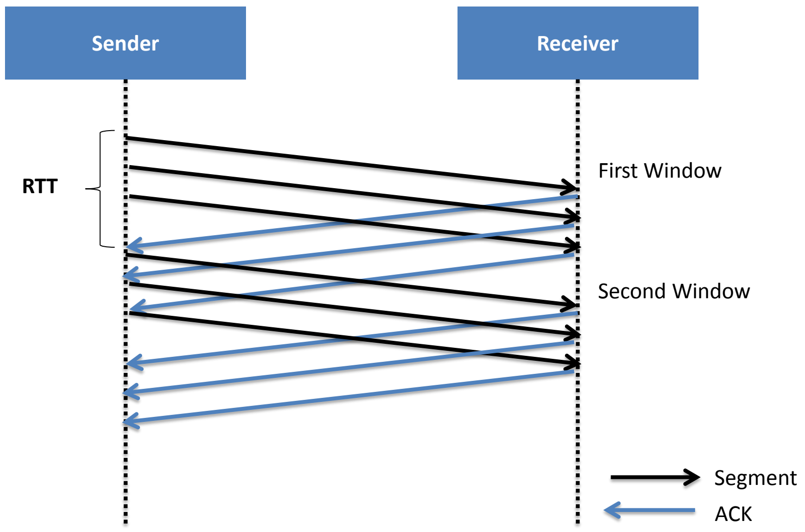

Figure 4: Sequence diagram of the window based approach used in TCP

This behavior is shown in Figure **??**. To achieve this behavior, the sender partitions it's send buffer two parts. The first part denotes the segments which were already sent but not acknowledged and the second part denotes the segments which could be sent in the future.

Problem 13 Congestion Control Simulation

Simulate the congestion control algorithm of TCP. It is sufficient to implement the two states slow start and congestion avoidance.

Solution:
Solution is available in tcp_congestion_control.py