The “Rocket”-TCP Protocol

Yingzhi Hao

Shalin Patel

Abdullah Siddiki

(Last Name Alphabetic Order)

Content:

Overview

Segment Structure Overview

Our implementation simulates a TCP-like protocol. This protocol has the ability to reliably transfer data over a channel that simulates various errors, including loss, corruption, and swapping, and the protocol maintain fairness when the channel is congested. We construct segments to send over the channel that contain a portion of the data and header information that is processed to ensure reliable data transfer. Each segment is a python bytearrary object. This segment consists the following fields: 1-byte check-sum number, 1-byte acknowledgement number, 1-byte sequence number, and 256-byte data field. In the following sections we will explain how we process each of these header fields to ensure data is sent and received properly.

Check-Sum (1 Byte)	ACK (1 Byte)	Seq-Number (1 Byte)	Data Field (256 Byte)

Table 1. Overview of the Segment's Structure

Check-Sum

The check-sum value is calculated at the sender side by doing XOR calculation over every byte of the ACK, Seq-Number and Data Field. The calculated value is put into the segment's check-sum field and sent to the receiver. The receiver extracts the check-sum value from the segment, and uses the inverted check-sum value to do a XOR calculation over every byte in the received segment. If the calculated result is $11111111_{(2)}$, then random bit-corruption has not occurred during this transmission.

ACK

The acknowledgement number is the next expected value of the sending data sequence. It is the received segment's sequence number plus the size of the data field (256). This number wraps around a range from 0 to 255.

Seq-Number

The sequence number is byte-stream number of the first byte in the segment. This number wraps around a range from 0 to 255.

Data Field

The data field contains the actual payload of a segment.

Running Logic

The sender prepares the check-sum, acknowledgement number and sequence number and wrap these data with the payload. The packaged segment is then put into the connected channel. Meantime, the sending-socket timer is started. If a timeout occurred, the sender resends the segment. If the sender gets an ACK segment from the receiver before timeout, the ACK segment will be checked to see if the package is corrupted or the ACK is incorrect, which both also make the sender resend the segment again. If the ACK is uncorrupted and correct, the sender will prepare a new segment until reaches the end of the input data-stream.

The receiver receives a segment from the channel. It uses the check-sum value to check if any corruption has occurred during transmission. If a corruption is detected, or the ACK number is not the expected value, the receiver will send an ACK segment with the previous ACK number inside to ask for a retransmission of the segment. Otherwise, the receiver calculates a new ACK value, wrap it into a segment and send the segment to ask for new data. The receiver also starts a timer when it sends an ACK segment to the sender. If a timeout occurs when the receiver is waiting for the sender’s segment, the receiver will retransmit the ACK segment again.

State Machine

Sender:

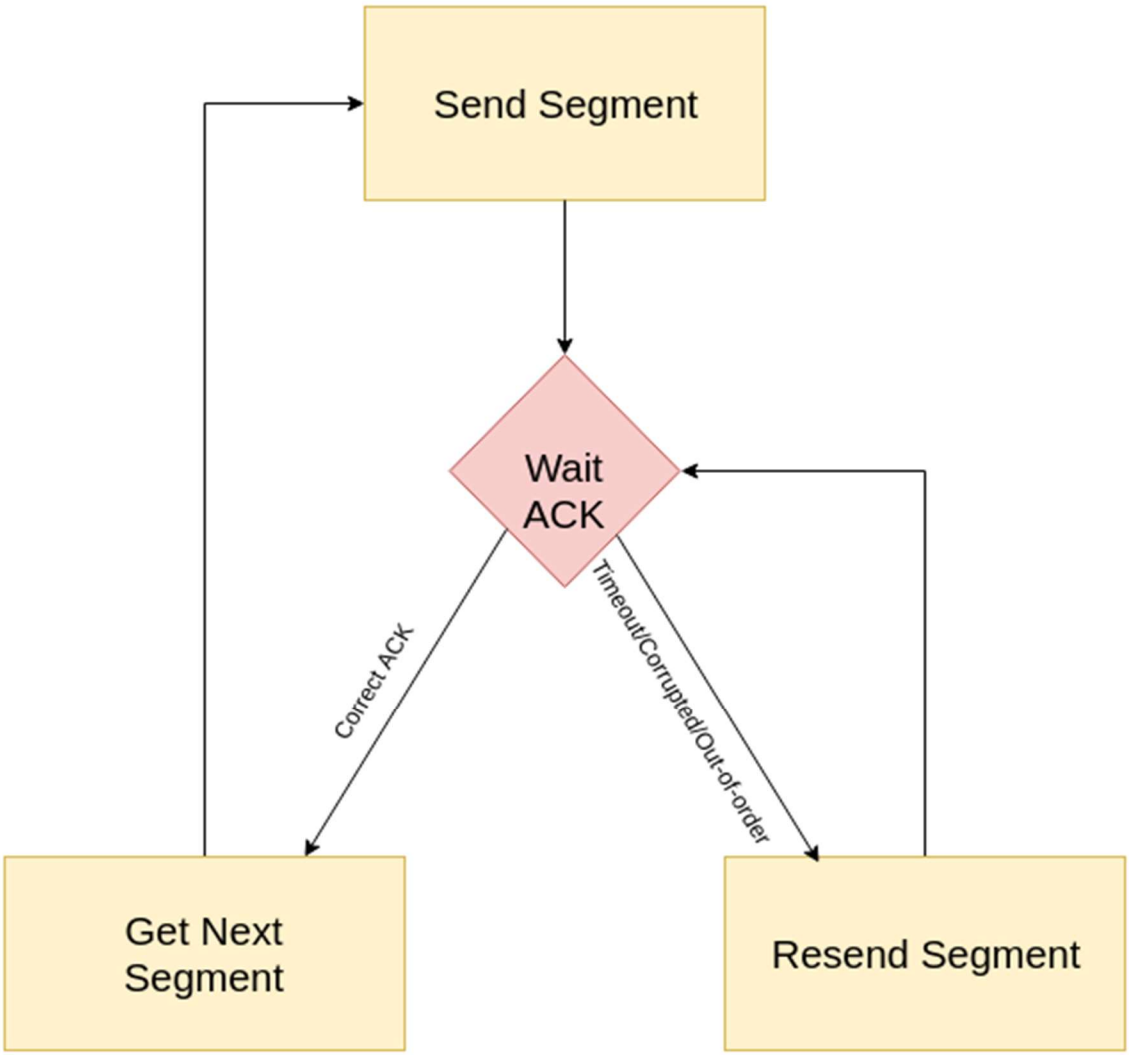

Figure 1. Sending Mechanism

Receiver:

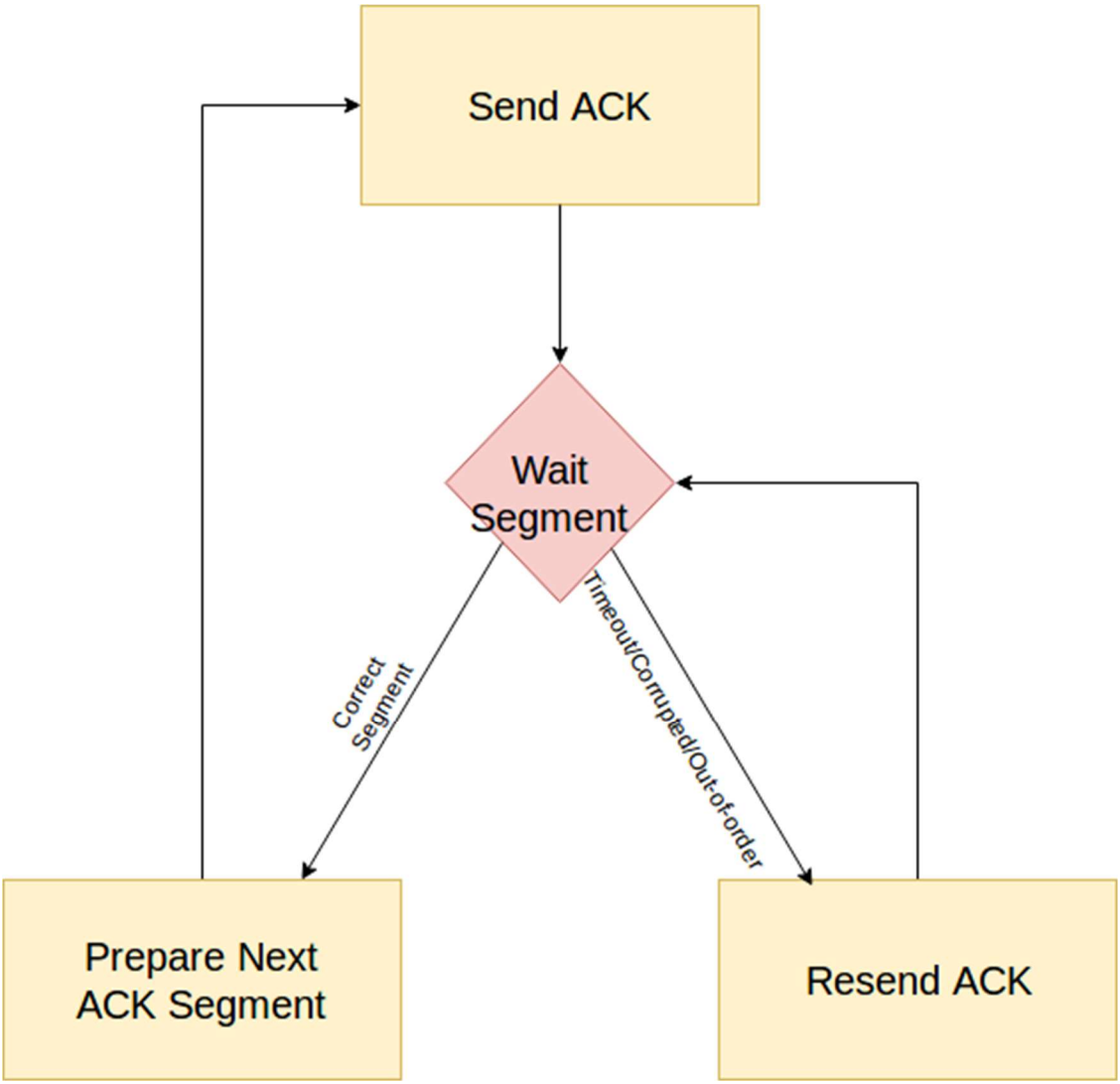

Figure 2. Receiving Mechanism

Reliable Data Transfer

Packet Corruption

A packet may experience corruption during transmission where a single bit or all bits in the segment may be flipped. Such corruption is detected by check the check-sum value of the received segment. If the calculated result is $11111111_{(2)}$, then random bit-corruption has not occurred during this transmission. Otherwise a corruption has occurred.

Packet Swap

Packet swap (out-of-order arrival of packets) is handled by the receiver. The receiver compares the sequence number of the received packet with the expected sequence number. When a packet is successfully received (in-order and not corrupted), the expected sequence number is computed as the sequence number of the received packet plus the segment size. If the received packet's sequence number is not equal to the expected sequence number, then there is a packet swap. We send an ACK to the sender with the sequence number of the packet we are expecting.

Packet Loss

Lost packets are handled using the ACK and sequence number fields of the header information of the segment. The receiver waits for a specified time before it begins to retransmit the ACK of the previous received and checked segment. The sender will continually send the next packet unless the ACK of the expected packet is received, which is the sequence number of the next segment. The rate at which duplicate ACKs from the receiver and segments from the sender are transmitted are modified based on the perceived loss, which will be discussed in the next section regarding fairness. We covered the edge case of the first transmitted data never being received by sending a special ACK telling the sender the first segment was never received to avoid the receiver thinking the first non-dropped packet is the first packet of data sent. This was covered since we did not implement handshake functionality, meaning there is no expected ACK on the sender side.

Fairness

Throughput Throttling

The protocol's transmission throughput is controlled by changing the timeout interval between sending out duplicate packages. On the sender side, this is achieved by controlling the timeout of the sending socket. On the receiver side, this is achieved by controlling the timeout of the receiving socket. Since a duplicate package is sent when a timeout occurs, a longer timeout means less segments are sent into the channel per unit time, which reduces the transmission rates and thus, can be used to achieve fairness. In this protocol, an occurrence of timeout, corrupted package or duplicated package is all considered as a congestion happening in the channel. Once

these situations occur, the sender or receiver need to retransmit their segments, and a counter will start. If the counter reaches three, the timeout value will be reduced to half, which reduces the transmission rate. A success transmission is considered as a positive sign of the channel. Thus, the timeout value will be decreased linearly every time when the sender gets a correct and uncorrupted ACK, which increases the transmission rate.