CO318 - Systems and Network Programming Lab and Project II

Project II: Developing a reliable data transfer protocol for computer networks and investigating its performance

Group 18: E/10/049, E/10/170

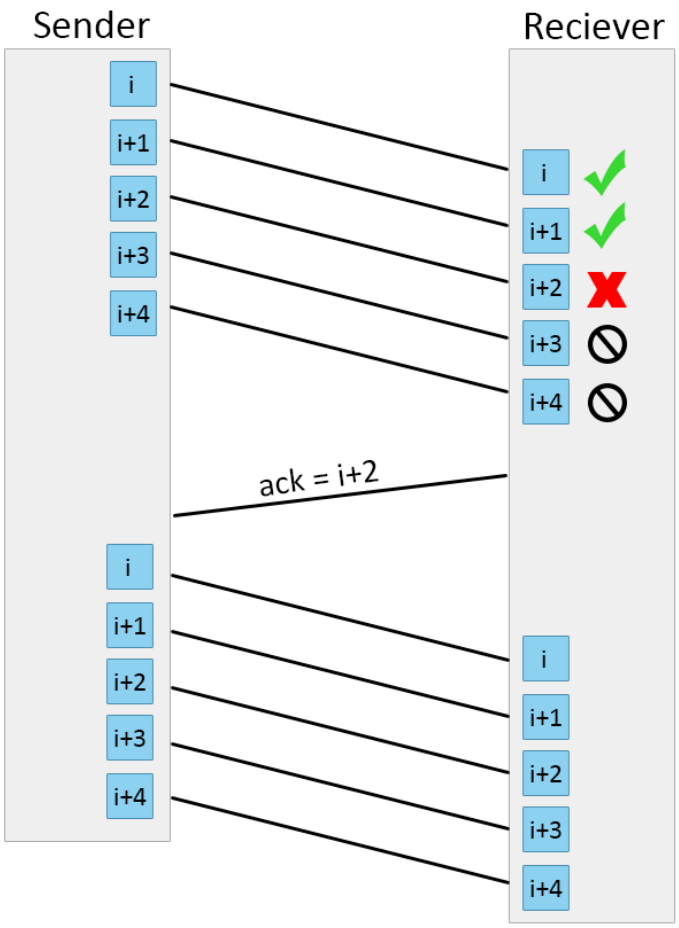
Project in brief

In this project we had to implement a given reliable data transfer protocol for computer networks and later improves the protocol for better error-control approach. Then we had to investigate the performance of both methods. Simply we created a simulator for the reliable data transfer protocol, which can be used for investigating its performance

Part 1

In part 1, we implemented the following protocol,

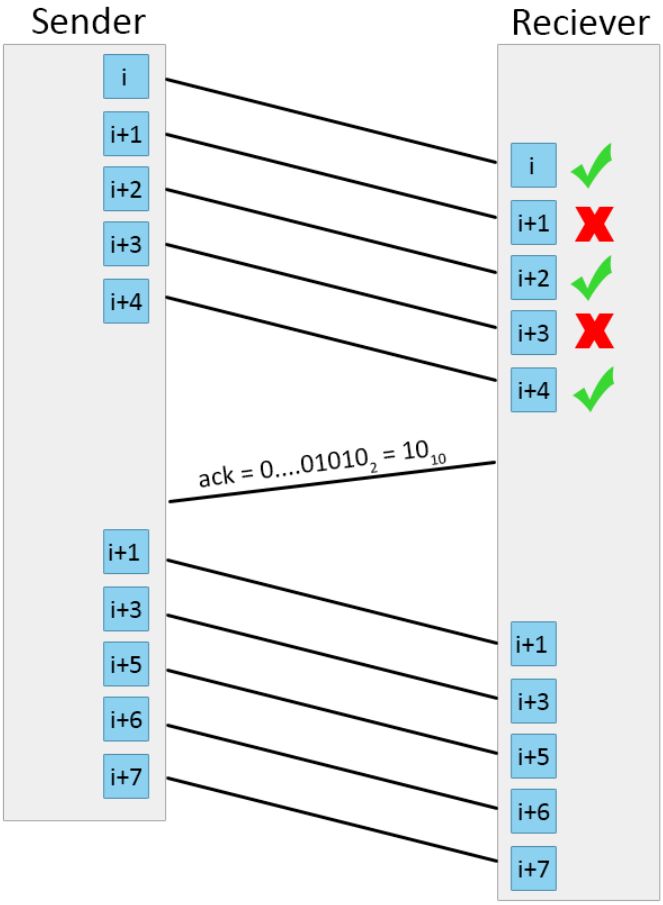
- The data will be sent as packets. Each packet has 100 bytes.
- Set of packet is called as a window. Here one window can contain one to five packets.
- Each packet has a sequence number in the range of 0 to 7.
- The sender continuously sends *W* packets corresponding to the current window, and then waits for an acknowledgement.
- The receiver receives *W* packets. And check whether the packets are damaged. If receiver found a damaged packet he discards all the remaining packets and sends the sequence number of the damaged packet to the sender as the acknowledgement.
- If there are no damaged packets then the acknowledgement will be the next expected sequence number.
- The sender receives the acknowledgement and then moves its window starting from the packet with the received sequence number. Then sends the new window.
- This process continues until all the packets reach the destination without damages.



Part 2

In the part 2 we had to propose an improvement on the error-control approach in part 1 protocol. We proposed the following method.

- In part 1, when the receiver receives a damaged packet, he discards all the remaining packets. This is a waste of packets. In our method we keep all the good packets and send the acknowledgement to get only the damaged packets.
- In this method, acknowledgement had to represent which packets in the window damaged. We sent an integer as an acknowledgement and when decoded as binary, it gives the packets which were damaged.



Performance study

We get the time it take to transfer 1MB file from sender to receiver. Our program calculates the transfer time and gave it in the standard output. Unfortunately we couldn't use an automated testing method so we get the average data rate out of about 3 to 5 results per each case.

Part 1		Window size				
		1	2	3	4	5
Error ratio	0	3550	5468	7218	8212	8579
	0.1	3213	4643	6187	6227	6828
	0.2	2843	3852	5060	5299	5073
	0.3	2460	3300	3918	4046	3665
	0.4	2142	2661	3002	2875	2706
	0.5	1792	2062	2280	2167	1935

Data rate (KB/s) with window size and error ratio in part 1

In part one, we have better data rate when the window size is high and the error ratio is low.

Part 2		Window size				
		1	2	3	4	5
Error ratio	0	3533	4787	7560	8337	8959
	0.1	3208	4238	6745	7736	8204
	0.2	2755	3778	6037	7672	7195
	0.3	2449	3721	5307	6075	6387
	0.4	2012	3217	4507	5081	5525
	0.5	1746	2638	3844	4305	4661

Data rate (KB/s) with window size and error ratio in part 2

In part two also, we have better data rate when the window size is high and the error ratio is low.

		Difference				
		1	2	3	4	5
Error ratio	0	-17	-681	342	125	380
	0.1	-5	-405	558	1509	1376
	0.2	-88	-74	977	2373	2122
	0.3	-11	421	1389	2029	2722
	0.4	-130	556	1505	2206	2819
	0.5	-46	576	1564	2138	2726

Difference between Data rate (KB/s) between part 1 and part 2

When we take the difference between two methods,

- Part 2 has much better data rate than part 1 when the error ratio is high and window size is high.
- When the window size is one, there is not much difference between two methods.
- We get an unusual situation when window size is two and error ratio is low. Part one had a slightly better data rate.
- Overall, **method two achieved better data rate** especially when the error ratio is high. So we can say the method we implemented in part two is successful.