Networked Systems and Services, Fall 2017
Forward error correction assignment
Axel Wikström 014312988

My FEC implementation adds a header to the UDP packets to keep track of the data. The header contains a 8-byte sequence number followed by a data length byte. The data length byte is used to signal the end of the transmission. It is also used in the XOR scheme to determine the length of the last recovered B-packet. The data length byte is zero when there is more data to follow. The data length byte being greater than zero indicates the end of the transmission.

In the repeat scheme, the receiver simply keeps track of the next sequence number, and ignores all other packets that do not match it.

In the XOR scheme, the receiver uses a “receive window” with three buffers. When the A and B data packets of the current window have been determined, the window is forwarded to the next ABC group. All packets outside the window are ignored. The packet type (A, B or C) is determined from the sequence number.

The XOR scheme has a few corner cases when the transmission ends. If the last packet is an A-packet, it is simply repeated three times for added redundancy. When the last B packet does not fill up all 100 bytes, its length is encoded into its header of the B and C packets. When forming the data of the C packet by XORing, the missing bytes from B packet are replaced by zeroes. When recovering the last B packet from the A and C packets, its length can then be determined from the header of the C packet.

Program usage

`sender.py` reads the data from stdin. The probability of packet being dropped can be specified with the `--drop-chance` option.

`receiver.py` writes data to stdout by default, but a file can be specified with the `-o` option.

By default the repeat scheme is used. To use the XOR scheme, the `--use-xor` option can be passed to both programs.

The unit tests in `test.py` can be run with `python -m unittest`

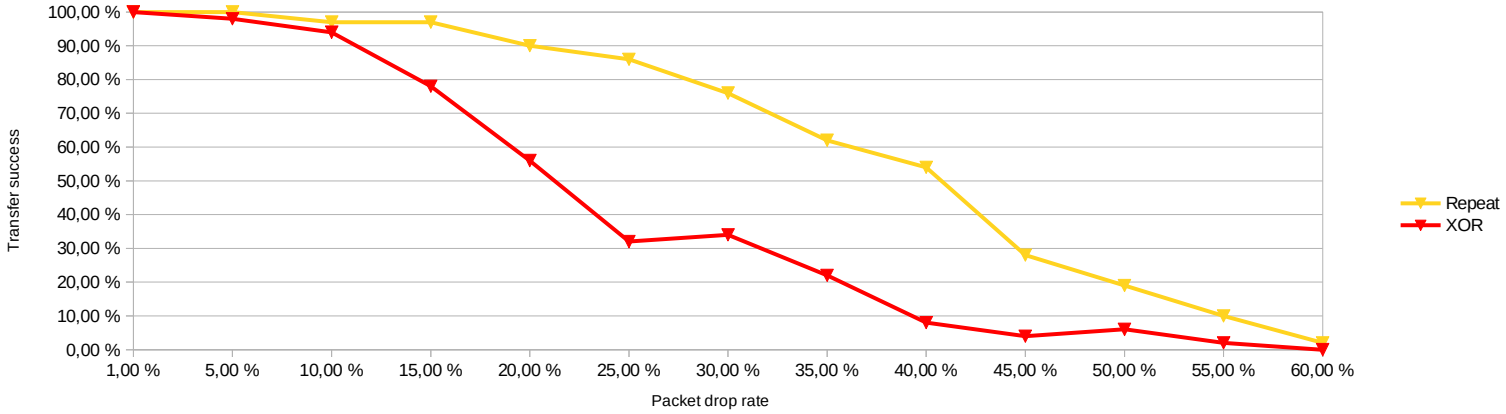
Benchmark

I wrote the `benchmark.py` utility to test the performance of the schemes.

A 1KiB file was transferred 50 times for each tested drop rate.

The table shows how much of the transfers succeeded with the specified rates.

Drop rate	**1,00 %**	**5,00 %**	**10,00 %**	**15,00 %**	**20,00 %**	**25,00 %**	**30,00 %**	**35,00 %**	**40,00 %**	**45,00 %**	**50,00 %**	**55,00 %**	**60,00 %**
Repeat	100,00 %	100,00 %	97,00 %	97,00 %	90,00 %	86,00 %	76,00 %	62,00 %	54,00 %	28,00 %	19,00 %	10,00 %	2,00 %
XOR	100,00 %	98,00 %	94,00 %	78,00 %	56,00 %	32,00 %	34,00 %	22,00 %	8,00 %	4,00 %	6,00 %	2,00 %	0,00 %



It can be seen that the repeat scheme performs better when the drop rate is increased. This seems natural, since the repeat scheme sends more redundant data. The code rate for the repeat scheme is 1/3, while it is 2/3 for the XOR scheme.