Project description

In this project, we will implement a transport protocol that provides some reliability services on top of the unreliable UDP. This will be done by augmenting UDP with the GBN protocol. You are required to implement a special GBN sender and receiver whose details are specified in the following subsections.

Sender

The sender script should be called with three arguments: *filename, receiver IP address*, and *receiver port*. Other variables like the *maximum segment size* (*MSS*), *window size N*, and *time-out interval* should be statically defined in your script.

The *filename* argument is the name of a file residing on the sender side, and it is required to send the file to the receiver. Other arguments are self-explanatory.

When the sender script is called, it proceeds with the following steps:

1. Read the file and divide it into K chunks of size less than or equal to the *MSS*. The structure of the sender packet is shown in Figure 1. The packet ID is a unique identifier for each packet starting from 0. The file ID is a unique identifier for each file being sent. The value of the trailer bits should be set to 0x0000 by the sender for all file chunks except the last one. The packet of the last file chunk should have a trailer value of 0xFFFF. This is done by the sender to indicate the end of the file being sent so that the receiver knows when to stop waiting for more bytes of the same file.

Packet ID (16 bits)	File ID (16 bits)	Application Data	Trailer (32 bits)

Timeout event: If a timeout occurs, the sender resends all the transmitted packets but has not yet been acknowledged.

For example, suppose that the window size is 4, packets 0, 1, 2, and 3 have been transmitted, and the received ACK is 2. Go-Back-N uses cumulative acknowledgments, so the received ACK number implies that all packets up to packet 2 have been correctly received. In this case, the sender slides its window forward such that it includes packets 3, 4, 5, and 6 and transmits packets 4, 5, and 6.
The sender terminates on receiving an acknowledgment for the last packet in the file.

Receiver

Running the receiver program deals with only one event: *The reception of a packet*. The receiver parses the received packet separating the header and trailer information from the application data and indicates via the user interface that a file is being received. If the packet ID is the expected packet to be received, the application data is stored. Otherwise, the data is discarded. Afterward, the receiver sends an acknowledgment packet to the sender with the ID of the last correctly received packet. When the last packet has been received, the receiver then acknowledges the packet, writes the data to a new file, and indicates via the user interface that the file reception is complete.

Simulated loss

As the network is small in our experiments, there is almost no packet loss. So, to test the implemented protocol, packet loss will need to be simulated by randomly dropping some packets at the receiver and assuming that they were lost. Thus, for each received packet at the receiver, generate a random number that decides whether the packet is lost or not. The simulated loss rate should be in the range of 5% to 15%. Note that we will assume that there are

no errors introduced by the channel to the received packets, so there is no checksum verification needed.