Reliable Transport Protocol

INF-2300 - Assignment 3

Deadline: Wednesday 30. October 2024

Goals

The goal of this assignment is teach you about the algorithms that ensure reliable data transfer from the transport layer in the OSI-stack model to the application layer above. You will implement one of these algorithms and learn about their strengths and weaknesses, as well as how they are vital for communication over an unreliable channel.

Reliable Data Transport

We are used to thinking of network communication channels (like sockets) as a continuous stream of bytes, much like reading a file from disk. This is only possible because the transport layer guarantees reliable data transfer from the network layer below. In reality, the network layer is a messy place. Data is fragmented, sent via different routes and possibly lost or even corrupted in transit. Each fragment probably won't even arrive in the order in which it was sent. There are three main algorithms which provide reliable data transfer, all covered in the course syllabus: Go-Back-N, Alternating Bit and Selective Repeat. You are encouraged to implement either Go-Back-N (GBN) or Selective-Repeat. GBN is the default mode in which TCP[1] operates.

Messy communication channels

There are three main problems that can arise when sending data over the network.

- **Packet loss:** a packet is irretrievably lost. This happens when a router's buffer is full and it receives more data. It will then either drop incoming data, or delete some of the data it already had.
- **Latency:** a packet is delayed en route and arrives out of order. Data from the application layer, such as a file, is split into multiple packages which are sent separately. These might be routed differently through the network and there is no guarantee that they arrive in the order that they were sent.
- **Corruption:** a packet is modified for some reason. This could happen because of interference, cosmic rays, etc.

Precode

The precode is essentially a simulator which uses classes to represent the various layers of the OSI stack and the connections between them. It simulates packets being sent across a network channel which might at arbitrary times drop packets, change them, or have them arrive out of order. You will need to handle all of these three cases.

[1] Transmission Control Protocol

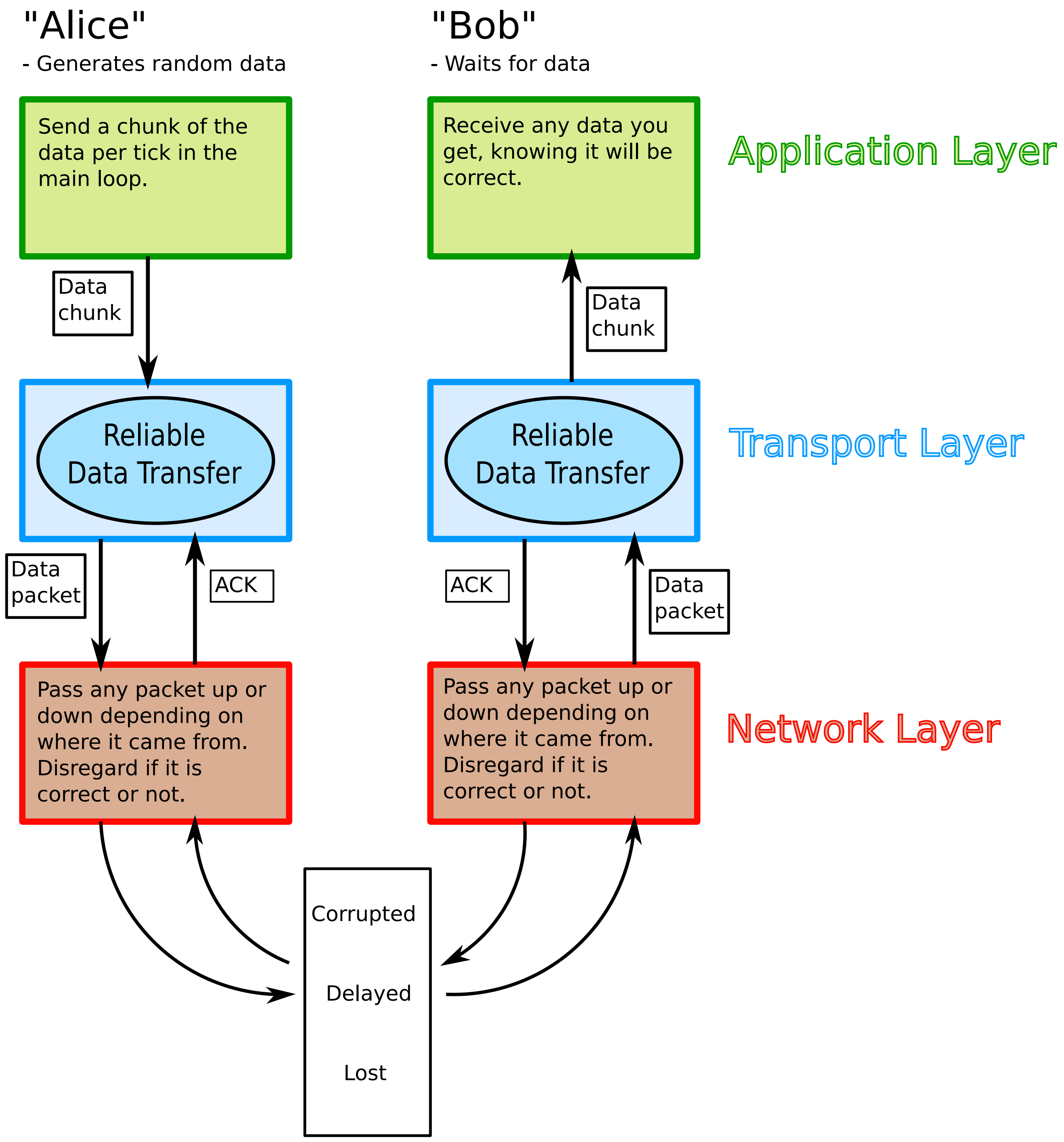
"Alice"
- Generates random data
"Bob"
- Waits for data
Send a chunk of the data per tick in the main loop.
Receive any data you get, knowing it will be correct.
Application Layer
Data chunk
Data chunk
Reliable Data Transfer
Reliable Data Transfer
Transport Layer
Data packet
ACK
ACK
Data packet
Pass any packet up or down depending on where it came from. Disregard if it is correct or not.
Pass any packet up or down depending on where it came from. Disregard if it is correct or not.
Network Layer
Corrupted
Delayed
Lost

Figure 1: A diagram showing the main data flow of the simulator. We build and connect two OSI stacks named 'Alice' and 'Bob', Alice will attempt to send data to Bob. The data passes through the Transport layer on her side, on through the network layer and back up to Bob's application layer. The problem is that the network layers will remove, change and delay some of these packets. Any ACK Bob might send back might also be delayed or lost.

The simulator will only change the 'data' attribute in the Packet class. This means an ACK is never corrupted. Any attributes you choose to add to the Packet class are never touched. The simulator only tries to send data one way. Bob never sends data to Alice, but if it works one way, it should (in theory) work both ways. Figure 1 shows the main data flow in the simulator.

Requirements

You will implement a reliable transfer protocol in the transport layer of the OSI stack. Do not modify the application or network layers.

Resources

- Reliable Data-Transfer Animation:
  http://www.ccs-labs.org/teaching/rn/animations/gbn_sr/

Hand-In

Hand in your solution and report by uploading them to Canvas. You must deliver two files: your report PDF and a compressed archive with your code. Don't forget to update your README!

Good luck!