Data Networking

Layering and Protocol Design for Milestone 3

Alexander Bunte (2518686)
Jonas Bürse (2519546)
Stefan Tombers (2521218)

September 20, 2011

1 Introduction

The goal of this project is to transfer textfiles as sequence of characters through a network with reliable data transfer. In this paper we will present our layering and protocol design of milestone 3.

2 Layering and Protocol Design

Beside the two layers, physical and application layer, cnet provides we implement three additional layers between them. We call them *link layer*, *network layer*, and *transport layer*, from bottom to top. Data packets in the link layer are called *frame*, in the network layer *datagram* and in the transport layer *segment*. To data packets from the application layer we refer to as *messages*. Figure 1 gives an overview how the different layers call each other.

2.1 Link Layer

The link layer provides basic "point2point" connections between neighboring nodes. It can handle the transmission of arbitrary large packets (limited by the buffer size, which is large enough to handle all messages in this scenario) independently of the MTU of the link because it has the ability of fra(g)mentation. This means that data from the upper layer is split into several frames, which fit to the MTU, and put together on the receiver side. Figure 2 illustrates the receiver side of the link layer. The ordering of the frames is checked by sequence numbers and identification numbers (ID). We use an additional *isLast* flag to identify when a datagram ends. This mechanism is guarded by the fact that no reordering of frames on one link occur. This layer also guarantees that all datagrams which are handed to the upper layer are *free of corruption*. Corrupt frames are recognized by computing a checksum and all frames with errors are dropped. We dropped the idea of implementing error correction due to the low rate of corruption. Finally, the data in this layer are handed to the physical layer in a first come first serve fairness.

2.2 Network Layer

The network layer has two main tasks: build and maintain a routing / forwarding table as well as route datagrams through the network. The building and maintaining of the routing table is done in a distributed fashion using the *distance vector algorithm*. Initially only the costs to the direct neighbors are known and all other costs are infinite. The minimal costs are broadcasted to the direct neighbors. If a node receives information of a lower cost path, it updates its forwarding table. If the minimal route cost changes it broadcasts this again to its neighbors. These data are send in a reliable fashion using acknowledgments. This ensures that the algorithm terminates with low cost routs for every node.

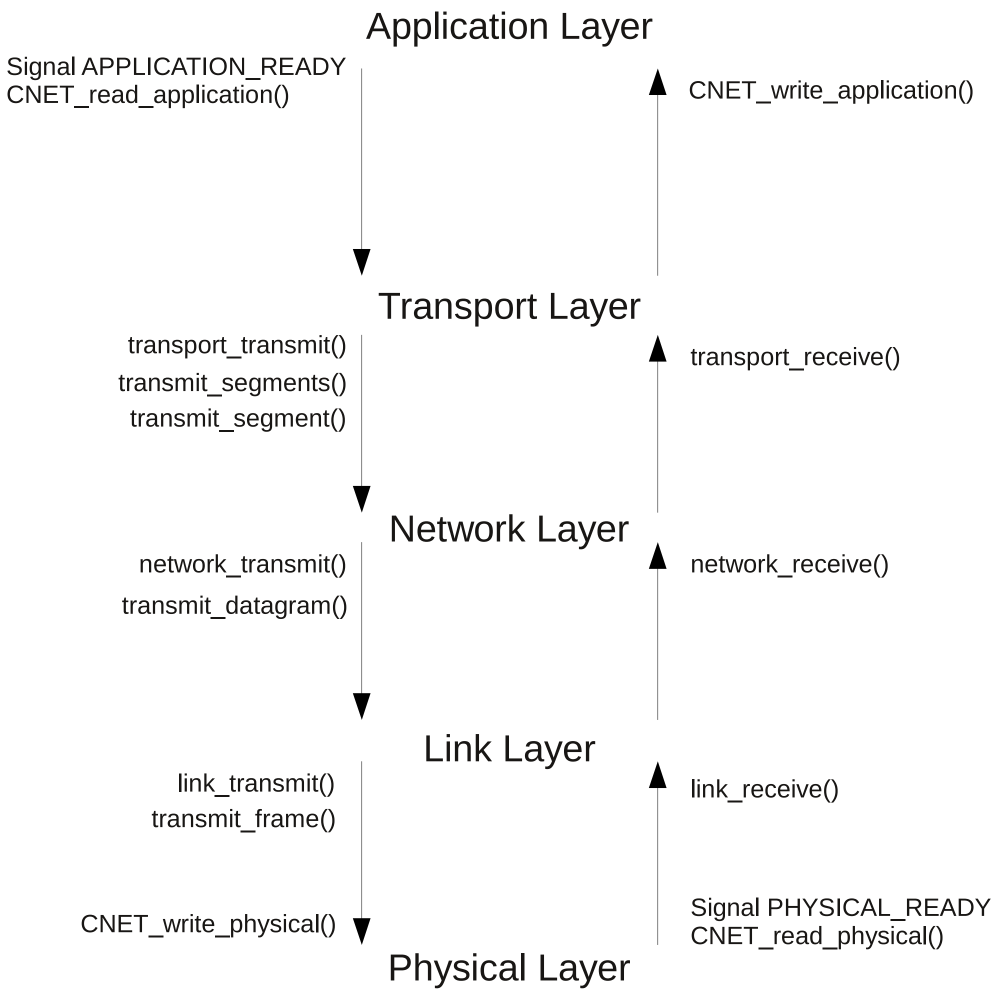

Figure 1: Overview Layers. Overview of the call hierarchy of the different layers.

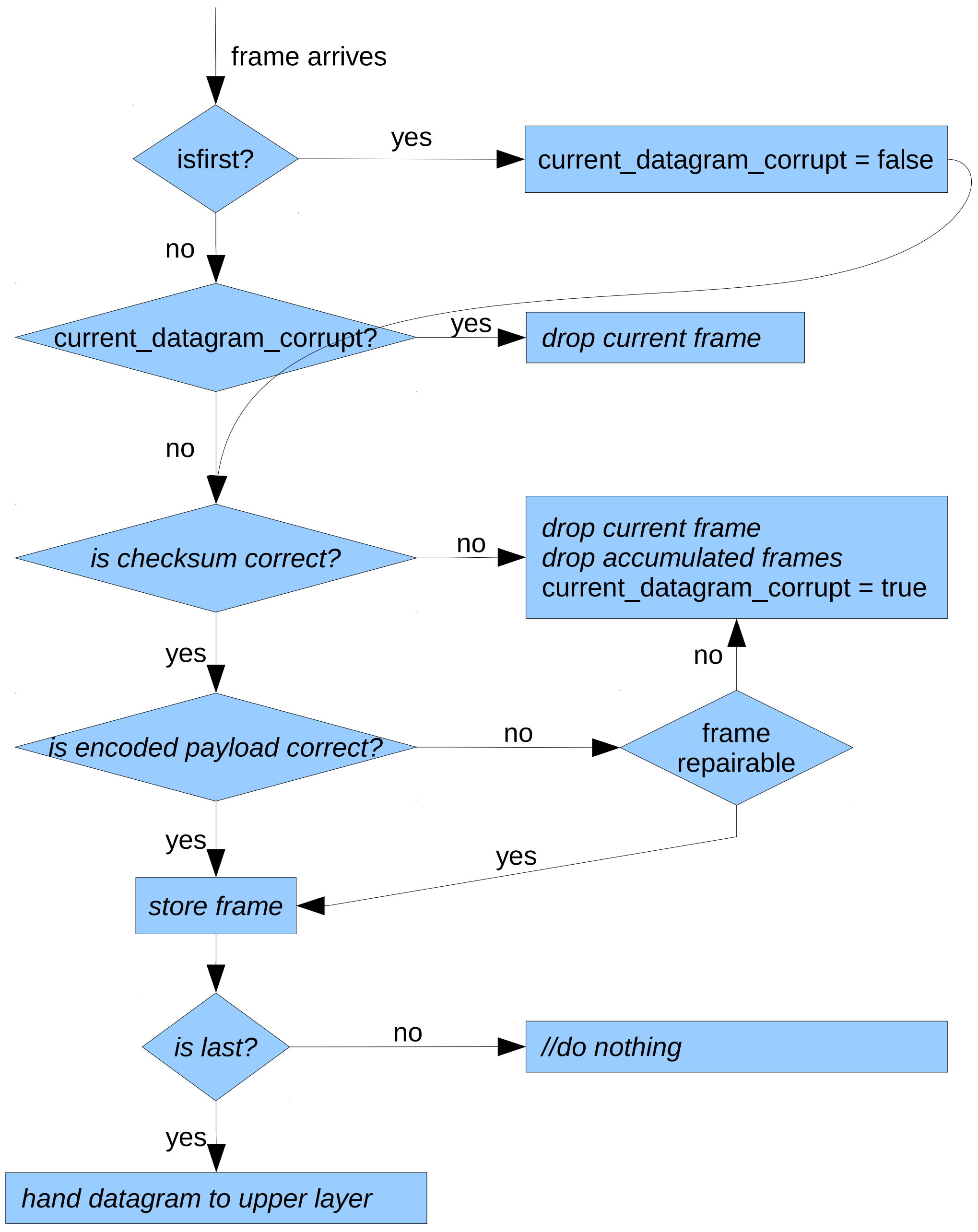

Figure 2: Receiver side of link layer. The flow graph shows how data on the receiver side of the link layer are handled.

The forwarding table is build from the routing table by choosing the neighbor with the minimal costs. If a datagram arrives at a node, the network layer decides if it is the final destination of the datagram or if the datagram needs to be forwarded on the basis of the forwarding table. On each hop a datagram takes, the hop counter is decreased to avoid that a datagram travels too long in the network.

2.3 Transport Layer

The main task of the transport layer is to provide reliable data transfer and congestion control. Reliable data transfer means that only corruption free messages are delivered in order to the application layer. To provide this service, the transport layer uses cumulative acknowledgments, sequence numbers in form of offsets and an isLast flag to identify the end of a message. For each received or send message from a host a connection is created. This contains a buffer for incoming data, cyclic sequence numbers and information to detect if messages are received complete. A window is used to send multiple messages without waiting for each acknowledgment. To increase performance we use cumulative acknowledgments where the acknowledgment number is the next offset the connection is waiting for. To provide this in an efficient way we buffer a limited amount of reordered messages.

Additionally this layer provides congestion control using the Tahoe algorithm explained in the lecture using a slow start and congestion avoidance phase with additive increase and multiplicative decrease.

Due to smaller MTU than the size of the messages we additionally segment messages on this layer to decrease the number of resend messages in case of corruption. If a frame on the link layer gets corrupt, we need to resend the whole segment the whole path the frames are reached previously without errors. If the segments are smaller, the traffic becomes lower.

3 Results

FIXME Analysis of milestone 3.