CS3102 P2: Practical Report

Reliable Data Transfer Using UDP

University of
St Andrews

190010906

01 April 2022

1 Introduction

This report cover the design and implementation of a connetion-oritented, reliable, unicast, transport protocol,built on top of UDP.

The protocol in question is called RDT - Reliable Data Transport

2 Design

Given the scope of the practical, simplicity was the main goal when considering the design of the RDT protocol. This section will discuss the design decisions and rationale behind them. The two attempted extension features, checksums and adaptive re-transmission timeouts, are also detailed here.

2.1 Packet Structure

RDT packets (see Figure 1) are composed of a constant 12 byte header and an optional data segment. The size of the data segment ranges from 0 to 1300 bytes, with 1300 bytes used as the maximum size so as not to interfere with the operation of `slurpe-3`, which was used for testing. **Theoretical maximum size?**.

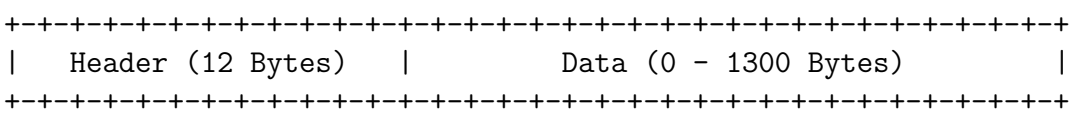

Figure 1: RDT Packet Structure

The RDT header (see Figure 2) is comprised of the following fields: a 32-bit `sequence` field, used for **what?**; a 16-bit `type` field, used to denote the packet function (see Figure); a 16-bit `checksum` field, calculated over the header and data segment to **what?**; a 16-bit `size` field denoting the size of the data segment (in bytes); and a 16-bit `padding` field to ensure 32-bit word alignment.

Several factors influenced the RDT header design. As the **C library function** `ftell` (used to calculating file sizes in `RdtClient.c`) returns 32-bit `long` values, a 32-bit `sequence` field was required to support the transmission of large files/amounts of data. The given implentation of the IPv4 header checksum used returns a `uint16_t` value, thus necessitating a 16-bit field. As a maximum data segment size of 1300 bytes was required, at least 11 bits were required for the `size` field, however 16 bits were used for alignment. For the remaining `type` and `padding` fields, there were no other considerations for field size other than 32-bit alignment.

A single type field was chosen, rather than a set TCP-style flags, for simplicity. Given the minimal nature of the RDT protocol, it was faster simpler to enumerate all packet types (see Figure), rather than testing multiple flags.

The type field supports the following types: `SYN` (0) and `SYN ACK` (1), used for the connection handshake; `DATA` (2) and `ACK` (3), used for sending and acknowledging data

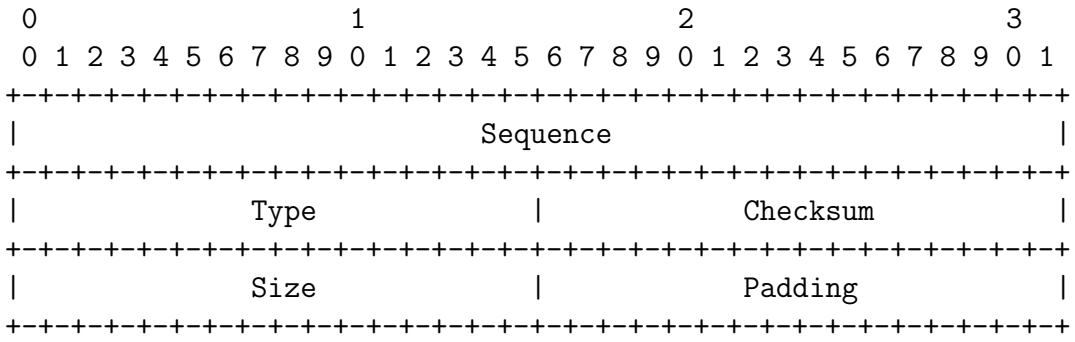

Figure 2: RDT Header

segments; `FIN` (4) and `FIN ACK` (5), used for graceful connection termination; and `RST` (6), used for abrupt connection termination.

2.2 Connection Management

The operation of the RDT protocol can be modelled by the FSM in Figure 3. For connection management, a two-way handshake is used. As RDT only supports uni-directional communication, a two-way handshake is adequate for establishing and terminating connections.

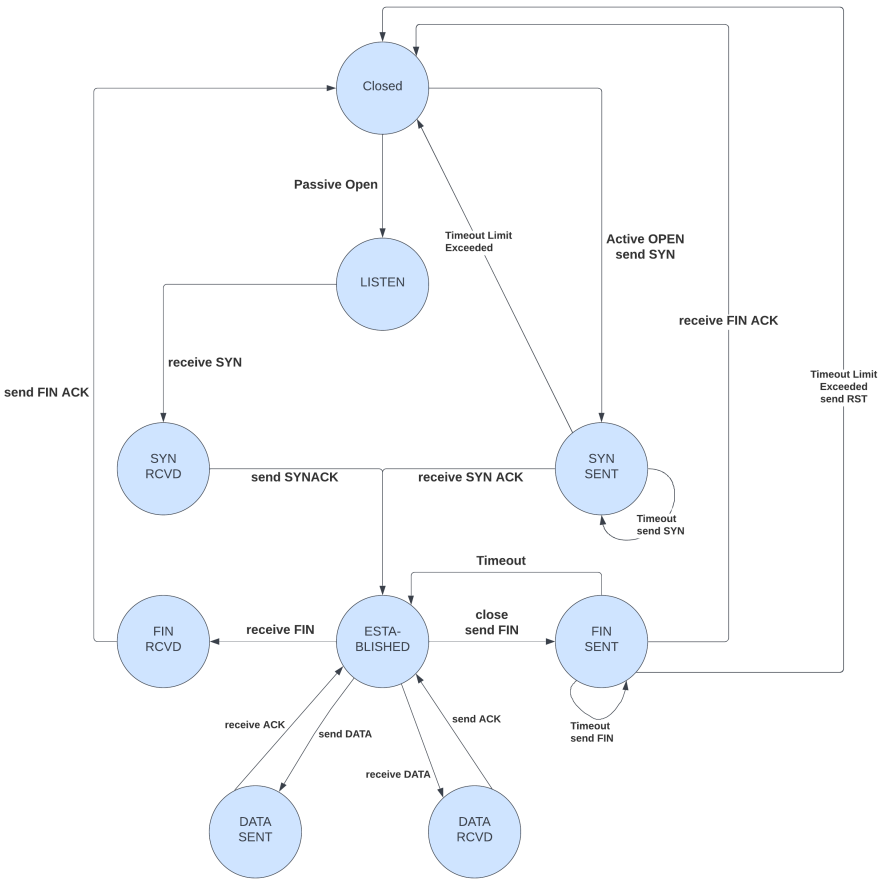

Figure 3: Finite State Machine

200 ms is used initially for the handshake RTO, as packet is only 12 bytes.

Server timeouts not required.

3 Testing

This section will detail how RDT was tested to validate correct operation.

3.1 Methodology

To test the ability of RDT to deliver packets in a reliable and ordered manner, two test programs were created. The latter was run on **pc** and the former on **pc**. Slurpe was place in the middle. A file was transmitted from A to B. Decoded with SHA.

4 Analysis

- Size of header vs size of packet

- Bandwidth utilization

5 RDT Packet Data Size

Several experiments were carried out to measure the effect of varying the RDT packet size.

The maximum packet size supported by UDP datagrams is 65,507 for IPv4 (this analysis will not consider IPv6), therefore given a fixed header size of