Fredrik Nordlund <fnordl@kth.se> Prakash Rajagopalan<prakashr@kth.se>

# IK2213 RDUP Report

## Introduction

We all know that most extensively used transport protocols are TCP (transmission control protocol) and UDP (user datagram protocol). They are transport protocols that run over IP links, and they define two different ways to send data from one point to another over an IP network path. TCP running over IP is written TCP/IP; UDP in the same format is UDP/IP. TCP ensures all the data gets from point to point, it is called a "reliable" protocol. In UDP's case, that reliability is left to the user, so UDP in its native form is known as an "unreliable" protocol.There are indeed options out there in the form of a variety of "reliable UDP" protocols, and we are going to implement one such in this assignment.Using Reliable UDP transports ,one that has the benefit of high-capacity throughput and minimal overhead.

## Problem

The problem is to design and implementation a reliable transport protocol that runs on top of UDP providing Sliding window flow control,detection and ARQ-based retransmission of lost packets,detection and reordering of packets that arrive out of order.The RUDP is implemented using event-driven user-space scheduler .

We satisfy the basic requirement on this assignment.

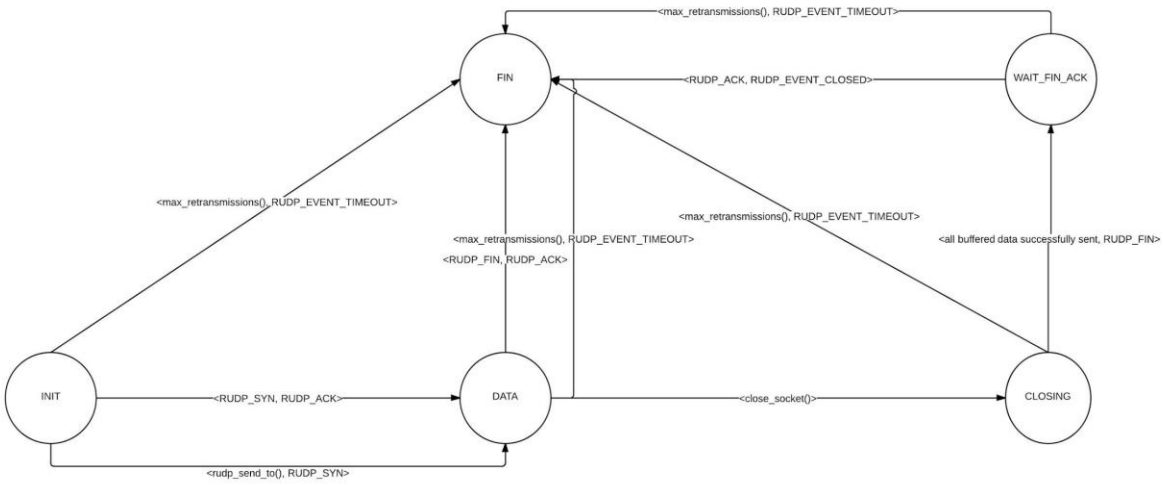
## Solution

The RUDP is implemented as a single system.For state management, RUDP socket maintains a list of states associated with the socket,in addition to function pointers for event handler functions which can be registered by applications. We logically separate sender and receiver states, although sender and receiver can exist in the same state when both parties exchange data. Within a sender state, we maintain a sliding window of transmitting packets. A packet list is implemented to store packet that have not yet been transmitted. Within a receiver state, we maintain the sequence number of the last packet received. We utilize two types of events in RUDP .one which is triggered when data is received on a RUDP socket, and another which is triggered when we detect packet loss.We support two other events here :RUDP_EVENT_TIMEOUT, which indicates that that a packed has been retransmitted more than RUDP_MAXRETRANS times, and RUDP_EVENT_CLOSE which indicates that an RUDP socket has been closed.An application can register to these events using RUDP Api's.

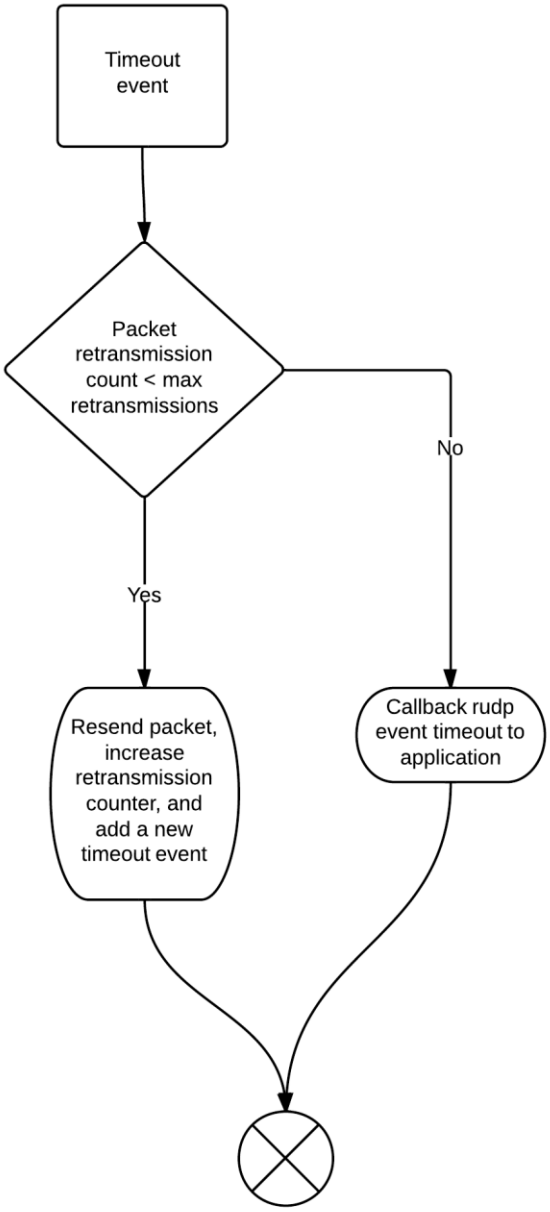
RUDP relies upon sequence numbers to provide reliability. An RUDP sequence number is an unsigned 32-bit integer, which is transmitted as part of RUDP header. When we send a SYN to initiate a RUDP session, a random sequence number is generated for the SYN packet. Subsequent packets are sent with incremented sequence numbers. ACK packets have a sequence number which is 1 greater than the sequence number of the packet they acknowledge. Also ,RUDP sender state maintain a sliding window of transmitted but unacknowledged packets. The size of the sliding window is defined by RUDP_WINDOW. Upon receiving an ACK packet, if the ACK packet is intended to acknowledge the

item sent, we remove this item from the list and increment the window size by 1, creating space in the window for new packets to be sent. When a non-ACK packet is sent in RUDP, a timer event is registered to occur after RUDP_TIMEOUT milliseconds. If the timeout event occurs, the packet associated with it will be retransmitted, unless the packet has already been retransmitted RUDP_MAXRETRANS times, in which case we will trigger a RUDP_EVENT_TIMEOUT event. When we receive an ACK, the timeout event for the packet being acknowledged is canceled. In RUDP, timeout events represent the detection of packet loss. When an application calls rudp_close on an RUDP socket, we wait until all queued data has been successfully transmitted, after which we send a FIN message. When a corresponding ACK has been received for the FIN message, we consider the sender state to be complete. Similarly, we consider a receiver state to be complete after it has sent the ACK for FIN.Once all sessions on the socket are complete, we close the UDP socket, and if the application has registered an RUDP event handler, we send a RUDP_EVENT_CLOSE event.

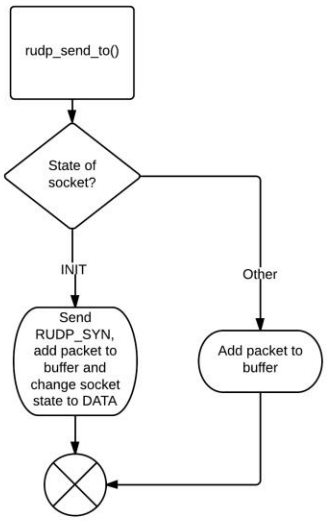
## State Diagrams



Fredrik Nordlund <fnordl@kth.se> Prakash Rajagopalan<prakashr@kth.se>



Fredrik Nordlund <fnordl@kth.se> Prakash Rajagopalan<prakashr@kth.se>



Fredrik Nordlund <fnordl@kth.se> Prakash Rajagopalan<prakashr@kth.se>