Reliable Data Transfer Protocol

By

Svyatoslav Kucheryavykh

Russell Clark and Matt Sanders

November 4, 2015

CS 3251

Georgia Tech

RxP Capabilities

1) Pipelined – RxP is a pipelined and supports Selective Repeat protocol. The sender will keeps sending packets until window size, given by receiver, is achieved. Receiver will send ACK acknowledging a packet with ACKs sequence number and all packets before it. If packet is not acknowledged sender will resend them. Receiver can also send a NACKs with the sequence number of expected or corrupted packet to indicate a particular packet that it wants to resend, without waiting for timeout.

2) Lost packets are handled with a timeout. If ACK for a particular packet is not received, the packet will be resend. Also receiver can send a NACK before the timeout with expected packet serial number to indicate that it has not received it.

3) Corrupted packets will be handled with checksum. MD5 hash will be used to compute the checksum of a packet and then it will be compared with hash computed at the receiver.

4) Duplicate packets will be discarded. Every packet will have a sequence number and if a packet will a certain number is already received, it will be discarded. ACK will also be sent.

5) Out of order packets will be reordered based on their sequence numbers.

6) Bi-directional transfer will be available once connection between client and server is established. Then both can act as a sender and as a receiver. Client will be able to select an option to get a file from the server or send his own.

7) Checksum will be a hash produced from MD5 using a packet. This checksum will then be compared to a new hash performed at the receiver. If the two hash values don't match, the packet will not be acknowledged and will have to be resend.

RxP High-Level Description

RxP is reliable, connection oriented transfer protocol. Before any data transfer between client and server can begin, a 4-way handshake must be performed. Client will send a SYN message to the client requesting a connection. Server has a challenge ready that it will send back to the client to do. In this protocol the challenge will be a random character string that server expects client to hash. The hash used is MD5. If no response to the challenge is received by the server, then nothing happens. Since server did not allocate resources for the potential client, a possible server denial attack can be avoided. In case that the challenge is successfully received and is verified to be correct, the server will send an ACK and establish a connection. Then once

the connection is established, client can choose to receive or send data to the server. The data is send with multiple packets, with total window size determined by the receiver. Each packet will have a sequence number that determines its position in a series of packets. Receiver will send ACK with a sequence number of a packet that is acknowledged and all the packets before it. The data will be send in a stream of data (bytestream). Each message will be broken down into packets and have an end packet that will be have ACK turned on.

Every packet will have a header that contains source and destination port number, sequence number, window size, checksum, tags such as ACK, SYN, and FIN, and option field.

RxP Header

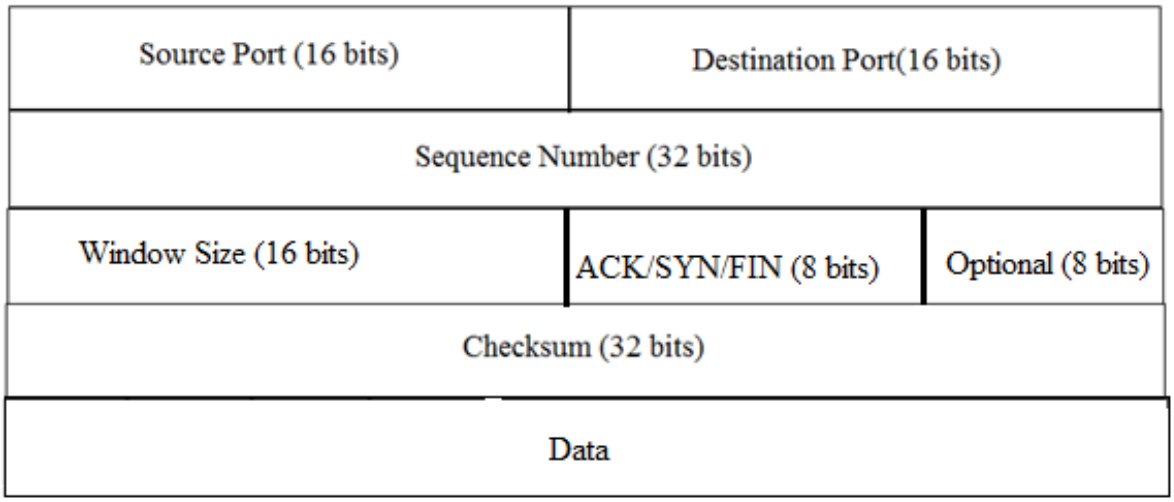

Figure 1. RxP Header. Total size is 16 bytes

RxP header contains Source Port (0-15) and Destination Port (16-31). Then comes the Sequence Number (32-63) that determines the packet's sequence number in a stream. It also shows which packets to resend and acknowledge if NACK/ACK field is selected. Window Size (64-95) determines the size of the window. It tells Sender how much space receiver has in its buffer. Then comes Checksum (96-127). It is created using MD5 hash and compared the the hash created at destination to make sure no data got corrupted. Flags ACK, NACK, SYN, FIN (128-131) determine the "type" of the packet and help transition through the states. Optional (132 - 159) field is for potential new future features.

RxP Finite State-Machine Diagram

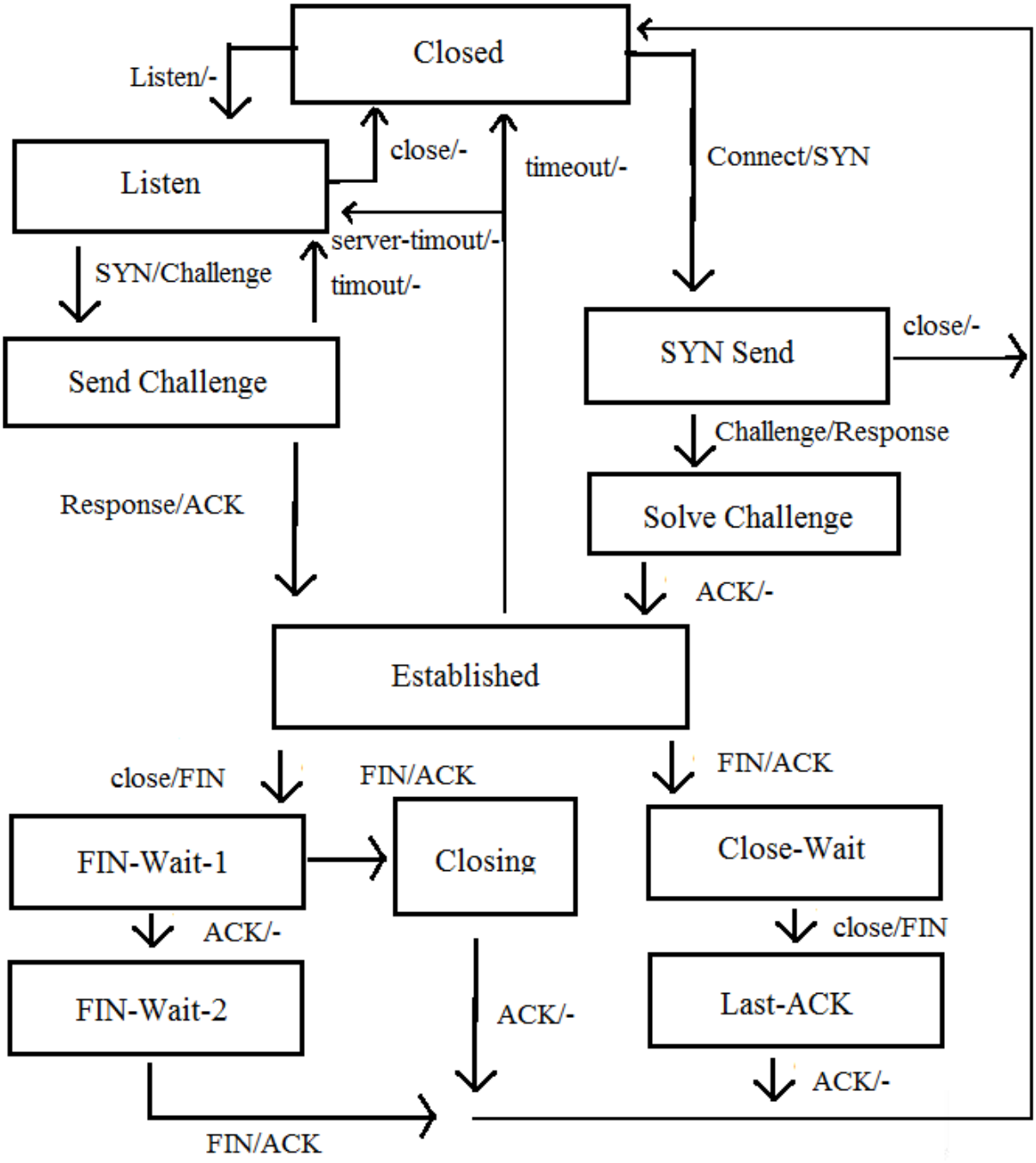

Figure 2. RxP Finite State-Machine Diagram.

Description of API

Socket() – creates an instance of RxP

bind(portN) – binds for a port number portN

connect(dstAddress) –connects to a particular address

listen(dstAddress) – Listen for clients.

send(msg) – sends a given msg to connected address.

receive() – receives data from established connection. Returns the sent msg

.close() – closes a connection.

setWindowSize(size) = sets the window size to size

Algorithms

MD5 – This hash function will be used to hash the file and turn it into a checksum. This function will also be used when doing the challenge the server sent to make sure the client is determined to connect.

Changes from the previous version

Clarified description of what happens with duplicate packets: They will be discarded but an ACK will be send afterwards anyways.

Added more methods like setWindowSize() and bind()

Modified description of the RxP to include bytestream description as well

Modified Header by decreasing size of window segment to 16 bits. Removed NACK. Made Optional section 8 bits long