RELIABLE TRANSPORT PROTOCOL

PROTOCOL SPECIFICATION

October 2014

Prepared for
CS 3251 - Programming Assignment 2
Georgia Institute of Technology

by
Max Kim & Siu-Lok Tsui

Protocol Implementation Capabilities

- Pipelining
  - RTP supports a Selective Repeat Automatic Repeat-Request protocol. A window size argument is supported for pipelined transfer that the receiver uses to notify the sender of an appropriate window size in order to ensure flow control. However, rather than using an acknowledgement number to confirm the receipt of packets, RTP has a NACK flag that it can use to request for the retransmission of specific packets. The first packet sent in a stream will have the BEG bit set to 1, which denotes that it is meant to be the first packet in the stream. Then, the receiver can use this first packet and combine it with the window size/packet offset field with a NACK signal to ask the receiver for a specific packet that has been either lost or corrupted. If the sender successfully receives all of the packets, it will simply send an ACK back to the sender to notify it of the successful transfer.

- Bi-directional transfer
  - Once a connection is established, RTP supports the transfer of data between both the client and server in both directions.

- Lost packets
  - If no ACK is received after the retransmit timer expires, the sender will simply resend the packets that haven't been acknowledged. The retransmit timer is calculated by comparing the NTP timestamp in the packet with the receiver's own NTP time. The difference in this time will be added to a variable flat time in order to calculate the estimated round trip time. This variable flat time can be changed to allow the retransmit timer to be more or less lenient, and also to allow for slight differences in the end points' NTP times.

  - Selective repeat is implemented for pipeline operations, making it so that only the packets that need to be resent are repeated. The receiver can utilize the received packet's sequence numbers, as well as the first packet that has the BEG flag, in order to determine if any packets were lost and to request the retransmission of lost packets. The sender will also only be limited to sending a window size of packets that the receiver advertised it is willing to accept, allowing it to determine if a packet at the near the end of the transmission was lost.

- Corrupted packets
  - In the header field, the RTP checksum field will utilize the Cyclic Redundancy Check. Both the sender and receiver will divide the entire packet's binary data by a keyword, in this case 0x04C11DB7 for RTP. The sender will place the remainder in this checksum field, and when the receiver receives the packet it will compare the remainder it calculates with the remainder in this field in order to verify the

data’s integrity. This standardized keyword was set by IEEE 802.3, and will be used to ensure that the same keyword is always used to verify the data.

- Duplicate packets
    - Duplicate packets are simply discarded by the receiver if the sequence number repeats.

- Out-of-order packets
    - For pipelined operations, the out-of-order packets will be re-ordered while in the buffer by utilizing the sequence numbers.

RTP High-Level Description

In this day and age, TCP has become outgrown by the network that it sustains. RTP is a modern transport protocol focused on reliability that aims to remedy the shortcomings of TCP. The protocol is connection oriented and requires a connection to be established between the sender and receiver before any data packets can be transmitted. Applications that utilize RTP will first establish a connection, and then encapsulate buffers of data into RTP packets. While similar in some ways to the TCP that it is aiming to replace, RTP will include a few notable differences. The RTP header, connection state progression, and algorithms utilized to ensure reliable delivery differ in various ways from TCP.

The RTP header contains various changes and improvements. One of the largest changes involves the removal of the acknowledgement number field from the header. Rather than confirming the receipt of individual packets, the endpoints will simply use a single ACK flag to denote the receipt of a whole window of packets without any errors, such as corrupted packets, lost packets, or the receipt of duplicate packets. If a receiver does detect an error in a given window of packets based on the sequence number or checksum, a NACK flag will be set in the header. If the NACK flag is set in a receiver’s response, the window size header field will instead be used to notify the sender which recently sent packet should be resent. Although this change adds complexity in the development of the software for processing RTP packets, it minimizes the header overhead per packet. Several fields have also been removed from the RTP header in order to maximize efficiency. The data offset, urgent and and push flags, urgent pointer, and options field have all been removed in favor of optimizing the size of the header. Despite the removal of all these fields, the header fields for the destination and source port numbers, window size, and checksum have all been increased to 32 bits to better accommodate the growth of the modern network that sustains much more data than ever before. A new addition to the header is a header field that contains a portion of a Network Time Protocol timestamp. The primary purpose of the NTP timestamp is for a receiver to determine an estimated time for a packet from a client to arrive. When a packet is first received, a server can crosscheck the NTP timestamp in the packet with its own NTP time to calculate a timeout before closing the connection. During the connection, both endpoints can also utilize the NTP timestamps to determine retransmit timers before assuming that a packet is lost. An in-depth explanation of the details regarding the integration of the NTP timestamp in the RTP protocol is provided later in this report.

Similar to TCP, RTP is a connection oriented transport protocol, in which two endpoints must first establish a communication session before exchanging data. However, there exist potential pitfalls in TCP connection establishment that RTP aims to eliminate. Primarily, there exists an exploit in the creation of new connections in TCP that renders it vulnerable to denial of service attacks. The three way handshake of TCP creates this vulnerability as a server must wait to receive an ACK from a client after responding with a SYN, ACK before establishing a connection. If the client does not respond, then the server will maintain these unused connections until the connection times out, causing the server's resources to be wasted in the meantime. RTP aims to minimize the impact of malevolent connection attempts by strictly managing open connections. Rather than having the server wait for a final ACK from the client after responding with a SYN, ACK, in RTP the server will simply expect data packets after it responds to the client with an ACK. It will use the NTP timestamp in the packet received from the client to determine an estimated round trip time for a response, and if no following data packets are received from the client before this estimated time elapses, the connection will timeout and close. In this way, the server is much more resilient to denial of service attacks. With this method, a receiver strictly manages open connections that it is receiving data from, and will close the connection with a FIN packet on unused connections.

The state transition has also been simplified in connection establishment. One major is change is that there is less handshaking required to establish and close a connection. Once the initial two way handshake is completed, the connection is immediately established and the receiver will be ready to accept data upon sending its response to the sender. In closing connections, endpoints can simply send a FIN signal and close the connection on their end immediately. Due to the stringent nature of keeping a connection open, both endpoints will also simply close a connection if no data is transferred within a given period of time, making it so that the receipt of the FIN signal does not have to be acknowledged for the connection to close.

To ensure the reliable delivery of data, RTP utilizes sequence numbers, retransmit timers, and Cyclic Redundancy Checks. The exact use of these features is outlined further in this report.

RTP Header Structure and Fields

The RTP header structure and fields are as minimal as possible for simplicity while maintaining reliability. The fields are illustrated (Figure 1) and described in details below.

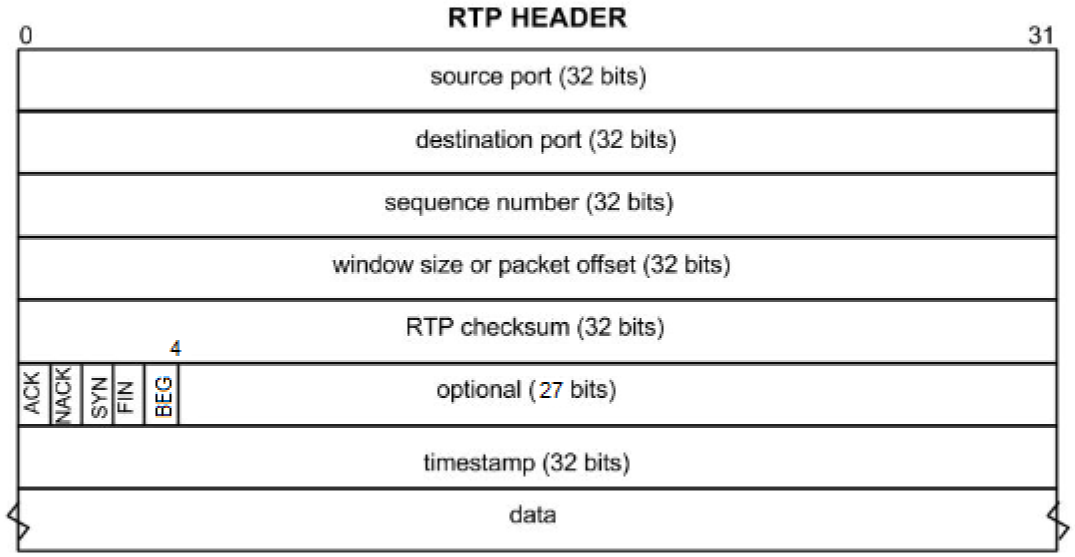

Figure 1. Reliable Transport Protocol Header Structure

Source port	bit 0-31	Source port of the packet
Destination port	bit 32-63	Destination port of the packet
Sequence number	bit 64-95	Sequence number of the packet
Window size ***or***	*bit 96-127*	Sliding window size
Packet offset	*bit 96-127*	Identify packet to be resent
RTP checksum	bit 128-159	Packet checksum
Control flags	bit 160-164	ACK, NACK, SYN, FIN, BEG bits
Optional	bit 165-191	Reserved for future implementation, etc.
Timestamp	bit 192-223	Parsed NTP timestamp to estimate RTT
Data (if any)	*variable*	

State Machine Diagrams

Figure 1 below displays the simplified state machine diagram of an RTP connection.

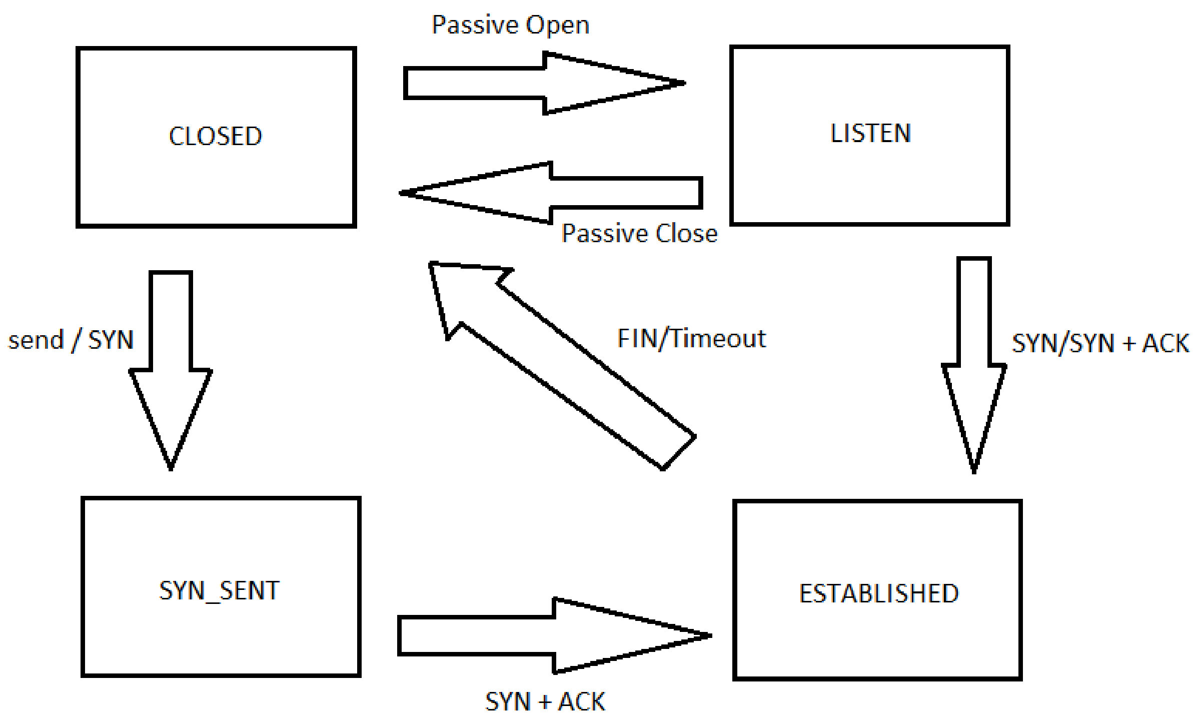

Figure 2. Reliable Transport Protocol Finite State Machine Diagram

Figure 2 below displays an example endpoint state transition that shows the successful transfer and acknowledgement of data as well as the use of a NACK signal from the server to request the retransmission of a lost packet.

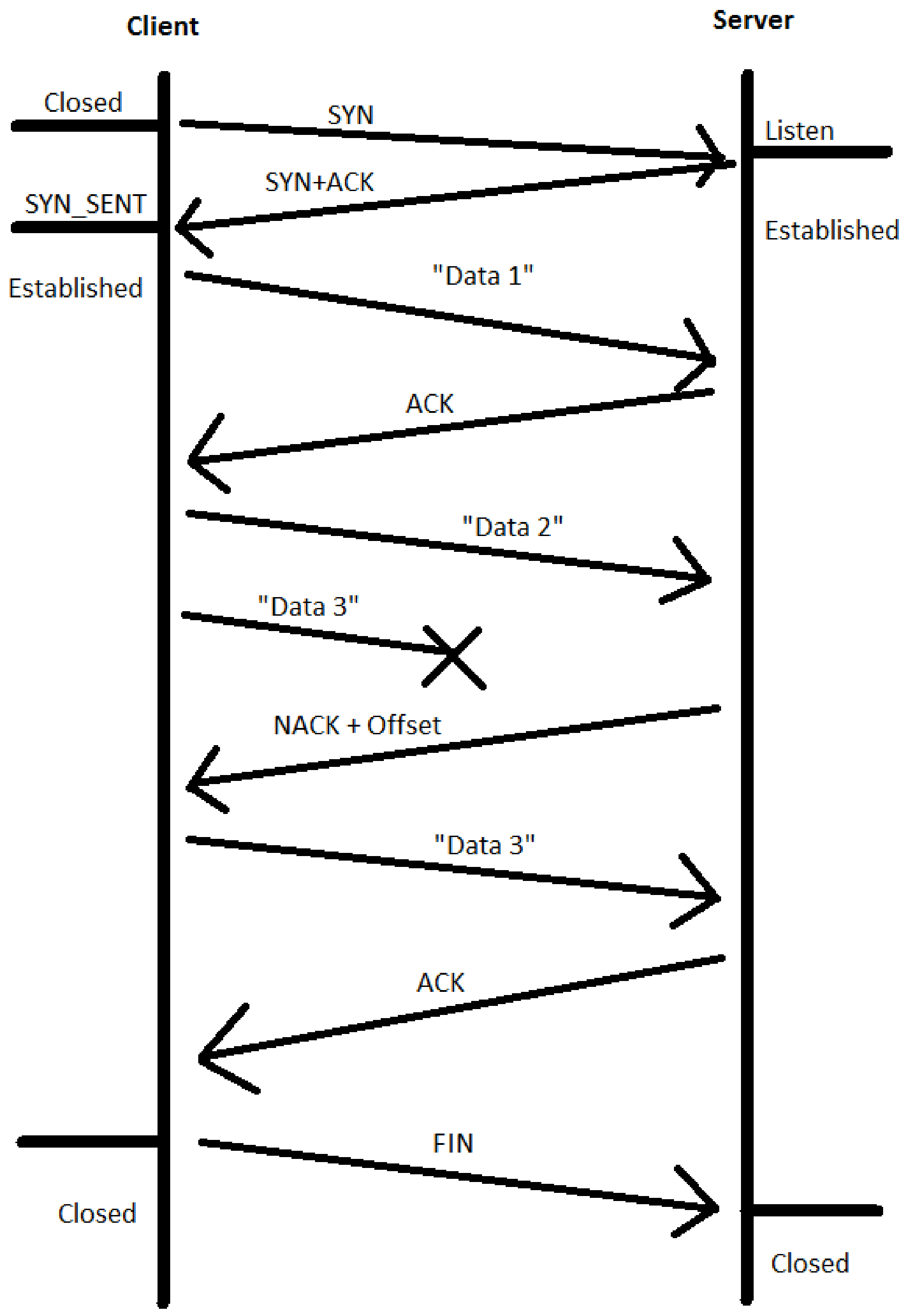

Figure 3. Reliable Transport Protocol Example Endpoint State Transition

Programming Interface

The RTP application interface provides for calls on the RTP to manage the connections between endpoints and to send and receive data. The OPEN function will be used to initialize a connection between two endpoints, which requires a two way handshake before data can be transmitted involving the sending of a SYN packet and the response of a SYN, ACK packet. The CLOSE function provides a method to close a connection when either endpoint wishes to close a connection, or if a connection times out due to the passing of time without the receipt of any data from a new connection. The SEND method provides the functionality of encapsulating a data buffer in RTP packets and sending them to a receiver on an open connection. The RECEIVE method signifies that an endpoint is prepared to receive data on a connection, and allows the receiver to send the sender information relevant for the reliable transfer of data, including a window size and ACK/NACK responses.

Algorithms

- Cyclic Redundancy Check
    - RTP will utilize the Cyclic Redundancy Check in order to verify the integrity of data. An entire RTP packet is considered as one binary message by the sender and receiver. The sender first calculates the remainder of dividing this message by the CRC-32 keyword standardized by IEEE 802.3, 0x04C11DB7, and then places this remainder in the checksum field of the header. After the receiver receives this packet, it will also separately calculate the remainder of dividing the entire packet by the keyword. If the remainder calculated matches the one in the header, then the receiver has verified the integrity of the data. If the remainders differ, then the receiver will consider the entire packet corrupted and will send a NACK to the sender so that it will resend the packet. This algorithm was chosen because of its mathematical simplicity and its ability to easily detect errors. More information on the CRC used can be found in Reference 1.
- NTP Timestamp
    - All packets sent through RTP will have a header field for the NTP timestamp to denote the time at which the packet was sent. A 64 bit NTP timestamp consists of two halves. The first half contains a 32 bit representation of the number of seconds that have passed since the era epoch began, with the current era having begun on January 1, 1990. The second half contains a 32 bit number that specifies the fraction of the second currently in the first half that has elapsed. The most significant bit represents ½ of the second, and each less significant bit represents 1 / (2^n) of the second, with n being the distance of that bit from the most significant bit. For example, the next significant bit after the most significant bit represents ¼ of a second. Rather than using the entire timestamp, RTP will use the least 16 significant bits of the first half and the most 16 significant bits of the second half. This allows a span of 65536 seconds and a resolution of 1/65536 of a second. Due to the low latency nature of the internet, this is more than enough to determine an estimated round trip time. An

endpoint can simply calculate the difference between this received timestamp and its own NTP time to generate an up to date and relatively accurate estimated RTT. It can also add a variable flat time value in order to increase the amount of leniency before an endpoint resend a packet or closes a connection. More information on the NTP timestamp used can be found in Reference 2.

References

1. http://www.xilinx.com/support/documentation/application_notes/xapp209.pdf
2. http://www.eecis.udel.edu/~mills/ntp/html/warp.html