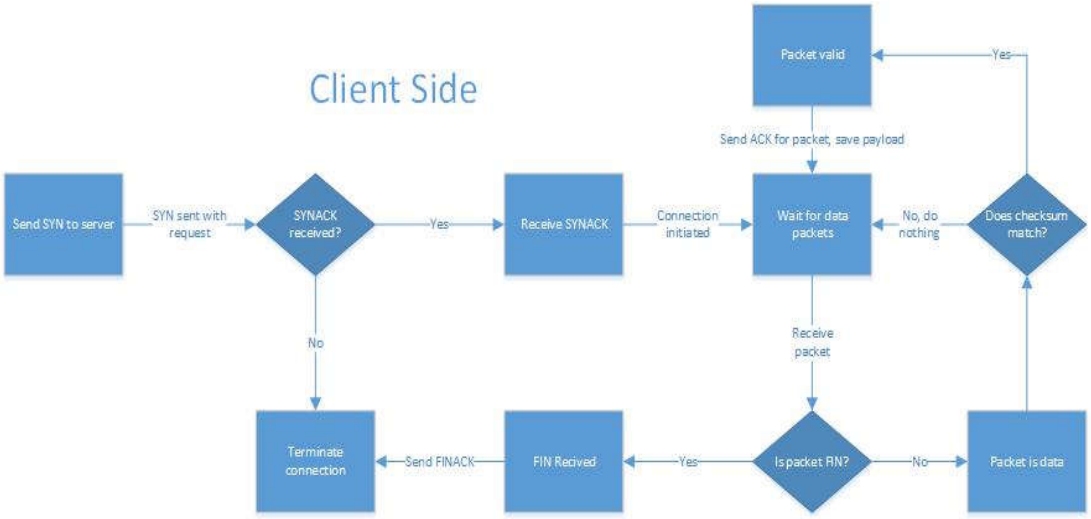
RTP Protocol Design Documentation

Cameron Gallahue and Bill Feng

1) High Level Analysis
   a) In order to send data via RTP, several pieces of information must be known. We must know where to send the data (IP address and port number), what data to send, and how much data can be sent at once (the window size in packets). The actual functionality of RTP runs as follows:
      i) The server is set up to listen for incoming packets on a particular UDP port number.
      ii) The client sends a connection request packet to the server, containing the client's IP and port number, as well as the specific data requested.
      iii) The server accepts the connection, and divides up the requested data into packets.
      iv) All of these packets are added into the queue, and as many are sent at once as the maximum window size allows. The client acknowledges each packet as it is received, and the server shifts the sender window over to continue sending more packets in its queue. As each packet has its own destination IP and port number, the server can service more than one connection at a time by simply adding the packets for each new request to the end of the sending queue.
      v) The client then closes the connection, which sends a message to the server saying that it wants to close the connection. The server then sends an acknowledgement of the connection closing, and closes the connection. The client then closes the connection once this packet is received.
   b) RTP Header Structure
      i) ip_src: the source IP address
      ii) sPort: the RTP source port number
      iii) ip_dest: the destination IP address
      iv) dPort: the RTP destination port number
      v) sPort_udp: the source port number used by UDP
      vi) dPort_udp: the destination port number used by UDP
      vii) seqn: the sequence number of the packet
      viii) ackn: the ACK number of the packet
      ix) SYN: Boolean representing initiation of a connection
      x) ACK: Boolean representing the acknowledgement of a packet
      xi) BEG: Boolean, used to indicate the first packet of a file being transferred
      xii) FIN: Boolean, used to indicate closing a connection or end of file
      xiii) GET: Boolean used when the payload is a request for data
      xiv) POS: Boolean used when the payload is the client posting a file to the server
      xv) TER: Boolean used for notifying the other host that file transfer has completed
      xvi) Timestamp: the current timestamp of the packet used to determine timeout
      xvii) RWND: receive buffer window for the sender host of the packet
      xviii) Checksum: the checksum value of the packet
      xix) Length: the total length of the packet
   c) Finite State Machine Diagrams: The next page shows the general FSM for the client and server. The page after shows the getFile client and server FSM are shown. postFile client FSM is extremely similar with getFile server FSM, with the addition of a name packet. For single packet transfer (database queries), a simple wait-for-ack procedure is used, if ACK is not received within a certain time, then the packet is resent.

Client Side



Server Side