RTP library implementation

Design Specification v 1.0

Venkat Srivathsan <sv2117@columbia.edu>
Working under Dr. Henning Schulzrinne
Columbia University
8.13.2004

introduction

The RTP (Real-time protocol) library is written is GNU C, for the UNIX platform. It does support multicast, but supports delivery of RTP packets to multiple unicast addresses.

The library is broadway classified into three modules, as

1. Network Module

   Files: networks.c, networks.h

   This module contains udp socket programming functions.

2. RTP module

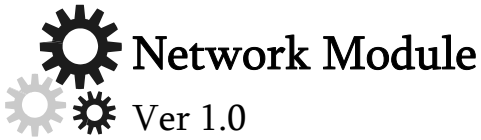
   Files: rtp.c, rtp.h

   This module contains rtp data structures and carries out the basic rtp packet send and receive.

3. RTCP module

   Files: rtcp.c, rtcp.h

   This module contains rtcp data structures and carries out rtcp functions; controls the flow of data.


Network Module
Ver 1.0

data structures

It does not declare any new data structures.

```
typedef int udp_socket;     // udp_socket is 'typedef'ed to int to be used
                               as a socket descriptor.
```

functions

It defines the following 4 functions

```
udp_socket* UDP_connection_Create();
```

arguments: *none*
returns: *socket descriptor pointer*

This function creates a udp socket and returns a pointer to the socket descriptor.

```
int UDP_connection_Delete(udp_socket* s);
```

arguments: *socket descriptor pointer*
returns: *int*[#]

This function accepts a pointer to a udp socket descriptor and closes the socket.

```
int UDP_connection_Bind(udp_socket* s, struct sockaddr_in addr);
```

arguments: *socket descriptor pointer*
socket address
returns: *int*[#]

This functions binds the socket s to the address addr

```
int UDP_connection_Send(rtp_packet pkt, udp_socket* s, struct
                                sockaddr_in);
```

arguments: *rtp packet*
socket descriptor pointer
socket address
returns: *int*[#]

This function creates a udp socket and returns a pointer to the socket descriptor.

[#] *The integers returned here denote error messages whose values are yet to be set.*

RTP Module

Ver 1.3

data structures

```
// RTP data header
typedef struct
{
        unsigned int version:2;         // protocol version
        unsigned int p:1;               // padding flag
        unsigned int x:1;               // extension flag
        unsigned int cc:4;              // CSRC count
        unsigned int m:1;               // marker
        unsigned int pt:7;              // payload type

        u_int16 seq;                    // sequence number
        u_int32 timestamp;              // timestamp
        u_int32 ssrc;                   // synchronization source
        u_int32 *csrc;                  // optional CSRC list
}rtp_header;
```

```
// RTP header extension
typedef struct
{
        u_int16 userdefined;            // user defined field
        u_int16 length;                 // length
}rtp_header_extension;
```

```
// RTP packet
typedef struct
{
        rtp_header header;              // rtp header
        rtp_header_extension ext;       // rtp extension
        char* payload;                  // payload
        long payload_len;               // payload length

}rtp_packet;
```

```
// list of unicast addresses here data is to be sent
struct sdl;

typedef struct sld
{
        udp_socket *s;                  // socket descriptor
        struct sockaddr_in addr;        // send address parameters
        struct sld *next;               // pointer to next

}send_address_list;
```

```
// RTP context
typedef struct
{
        u_int32 id;                     // context id
        u_int32 ssrc;                   // SSRC number
        u_int32 sent_packet_count;      // number of packets sent
        u_int32 sent_octet_count;       // number of bytes sent

        u_int32 version;                // version
        u_int32 marker;                 // marker flag
        u_int32 csrc_length;            // CSRC length
        u_int32* csrc;                  // CSRC list
        u_int32 pt;                     // payload type

        u_int32 ini_timestamp;          // initial timestamp
        u_int32 timestamp;              // current timestamp
        u_int32 time_elapse;            // total time in operation

        u_int16 ini_seq;                // initial sequence number
        u_int16 seq;                    // current sequence number

        udp_connection conn;            // udp connection parameters

}rtp_context;
```

```
// list of send addresses and receive parameters for each RTP context
typedef struct
{
        send_address_list *send;          // send address(es) parameters

        udp_socket* s;                    // socket descriptor for recv
        struct sockaddr_in recv_addr;     // receive address parameters

}udp_connection;
```

functions

It defines the following functions

---

int create_RTP_session(context *cid);

arguments: *context id*
returns: *error value*

Creates an new RTP session with session id cid, created and passes to the function by the user.

Finds an empty slot in the rtp_context_list array and uses its index as the cid value.

Initializes all parameters by calling *init_RTP_context*

---

int init_RTP_context(context* cid);

arguments: *context id*
returns: *error value*

Initialises context values to typical values and initial timestamp and seq number to random number, by calling *random_number*

---

int delete_RTP_session(context* cid);

arguments: *context id*
returns: *error value*

Deletes the context with id cid by setting to NULL, the index *cid in the context list.

int open_RTP_connextion(context *cid);

arguments: *context id*
returns: *error value*

Takes a context id as argument and creates and binds a socket to the receive address. The receive parameters should already be loaded by the user by calling *set_RTP_receive_addr*

---

int close_RTP_connection(context *cid, u_int8 *reason);

arguments: *context id*
returns: *error value*

Closes the RTP connection related to context id cid (and sends BYE? Havnt designed RTCP yet)

---

int set_RTP_session_stampRate(context* cid,u_int32 pt, u_int32 usec);

arguments: *context id*
payload type
timestamp change rate in microseconds
returns: *error value*

Sets the timestamp change rate.

---

int get_RTP_session_stampRate(context* cid, u_int32 pt, u_int32* usec);

arguments: *context id*
payload type
timestamp change variable pointer
returns: *error value*

Sets usec to the current stamp rate.

int set_RTP_receive_addr(context *cid, u_int8 *addr, u_int16 port);

arguments: *context id*
ip address in a string
port
returns: *error value*

Loads the receiver parameters and sets up the receive parameters for *open_RTP_connection*

---

int get_RTP_receive_addr(context *cid, u_int8 *addr, u_int16 *port);

arguments: *context id*
pointer to address string
pointer to port
returns: *error value*

Returns the receive address parameters in *addr* and *port*

---

int add_RTP_send_addr(context *cid, u_int8 *addr, u_int16 port);

arguments: *context id*
ip address in a string
port
returns: *error value*

Adds a unicast send address. It creates a unique socket and binds it to the sending address.

---

int add_RTP_send_address_list(context* cid, send_address_list* address);

arguments: *context id*
send address parameters
returns: *error value*

Adds *address* to the linked list of sending addresses, in context cid.

int remove_RTP_send_addr(context *cid, char *addr, u_int16 port);

arguments: *context id*
ip address in a string
port
returns: *error value*

Removes from the send address linked list the entry with ip address *addr* and port *port*.

---

int remove_from_RTP_send_address_list(context* cid,
send_address_list* address);

arguments: *context id*
send address parameters
returns: *error value*

Removes *address* from the send_address linked list.

---

int send_RTP_packet(context *cid, u_int32 tsinc, u_int8 marker, u_int16
pt, u_int8* payload, u_int32 payload_length);

arguments: *context id*
timestamp increment
marker
payload type
payload
payload length
returns: *error value*

Calculates the padding, timestamp and the next sequence number, generates the rtp header and sends the packet. Updates the context for packet count and octet count.

int receive_RTP_packet(context *cid, udp_socket* s, rtp_packet* pkt, u_int32* pkt_length, struct sockaddr_in *addr);

arguments: *context id*
timestamp increment
marker
payload type
payload pointer
payload length pointer
receive address parameters
returns: *error value*

Receives rtp packets on socket s and returns to the user, the rtp packet received, the packet length and the remote address parameters.

---

int build_RTP_header(rtp_packet* pkt, u_int16 seq, u_int32 timestamp, u_int32 marker, u_int32 pt, u_int32 ext, u_int8 padding);

arguments: *rtp packet pointer*
sequence no
timestamp
marker
payload type
extension
padding
returns: *error value*

Builds the rtp packet header with the supplied parameters.

---

int random_number();

arguments: *none*

returns: *random number*

generates a random number using the RFC specified algorithms.

RTCP Module

Ver 1.0

Design in progress…