An Implementation of ATM Signalling Protocol

KookJin Nam

September 1995

1 Introduction

In this document, some implementation issues, s/w structure and simulation result of an ATM signalling protocol server are described.

The implementation of signalling protocol server is based on ITU-T Q.2931 and Q.2110(SSCOP), and the compatibility with ATM Forum UNI is considered as well.

Currently, it works rather as a simulator. It is to generate exceptional cases as mush as possible to test and investigate the results of experiment. Porting of this server program to a real envionment is straightforward. For this, basic call connection with Fore System ASX-100 has been tested in the bases of EATM card[1] and device driver[2] in Linux environment.

Section 2 reviews general ATM protocol structures based on ITU-T recommendations, Q.2931 and Q.2110.

Section 3 describes program execution environment, implementation issues and data structures.

Finally experimental result of Q.2931 and SSCOP are described in section 4.

2 ATM Signalling Protocol

2.1 Overview of Signalling Protocol

Today, signalling protocol of ATM is being established in the wide range of the organizations in the world: ITU-T, ATM Forum, equipment venders and universities. At the moment, ITU-T and ATM Forum are leading the international standards. As a vender specification SPANS (Simple Protocol for ATM Network Signalling) of FORE SYSTEMS is being used in their products. In addition to these, CMAP (Connection Management Access Protocol) of Washington university and MSNL (MultiService Network Layer) of Cambridge university have been developed. In this document, Q.2931 of ITU-T has been based on the implementation of the point-to-point connection and Q.2971 is being considered as a point-to-multipoint call Connection control.

Protocol reference model of B-ISDN of ITU-T is shown in figure 1.

This protocol model consists of three planes: user plane, control plane and management plane. The management plane provides two functions: plane management and layer management. The plane management performs general management functions in the system and the layer management provides resource related functions and handles OAM information. Call control and connection control functions are performed in the control plane, and user information flow control is provided by the user plane.

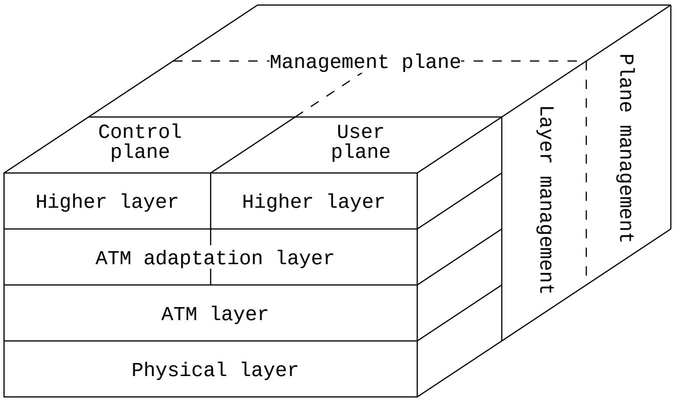

Figure 1: B-ISDN Protocol reference model

2.2 ITU-T Signalling Protocol Stack

The protocol stack that is defined by ITU-T is shown in figure 2. Mostly the layers below of CPCS are supported by ATM device driver. Therefore, signalling of ITU-T usually say SSCOP and Q.2931. The Q.2931 layer manages all ATM call related stuffs and the SSCOP takes charge of reliable data transfer. All messages of Q.2931 are transferred and received through SSCOP, and SSCOP utilizes the service of CPCS and SAR protocols which provide an unassured information transfer.

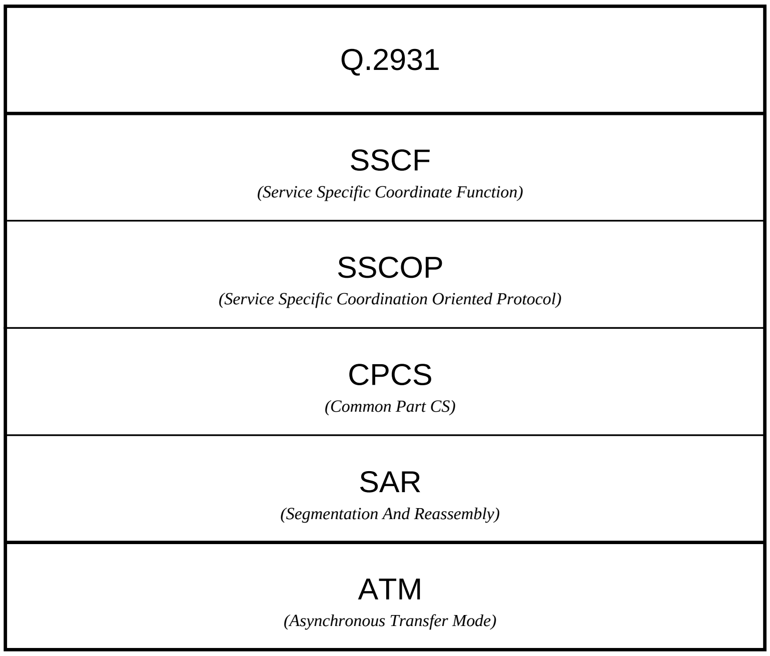

Figure 2: ATM signalling protocol stack

2.3 Q.2931

2.3.1 Call States

In the ITU-T Q.2931, 11 call connection states are defined as shown in figure 3. In this figure, the call state of BGN "release indication" has not been defined because it does not actually exist. This state transition diagram presents overall call connections of the ITU-Q.2931 even if there are still many detailed actions that cannot be described in this diagram.

When the call release fails, the call state goes to management state and RESTART message is sent to clear the virtual channel that has been allocated in this call. Upon receiving of the RESTART ACKNOWLEDGE, the call state returns to null state.

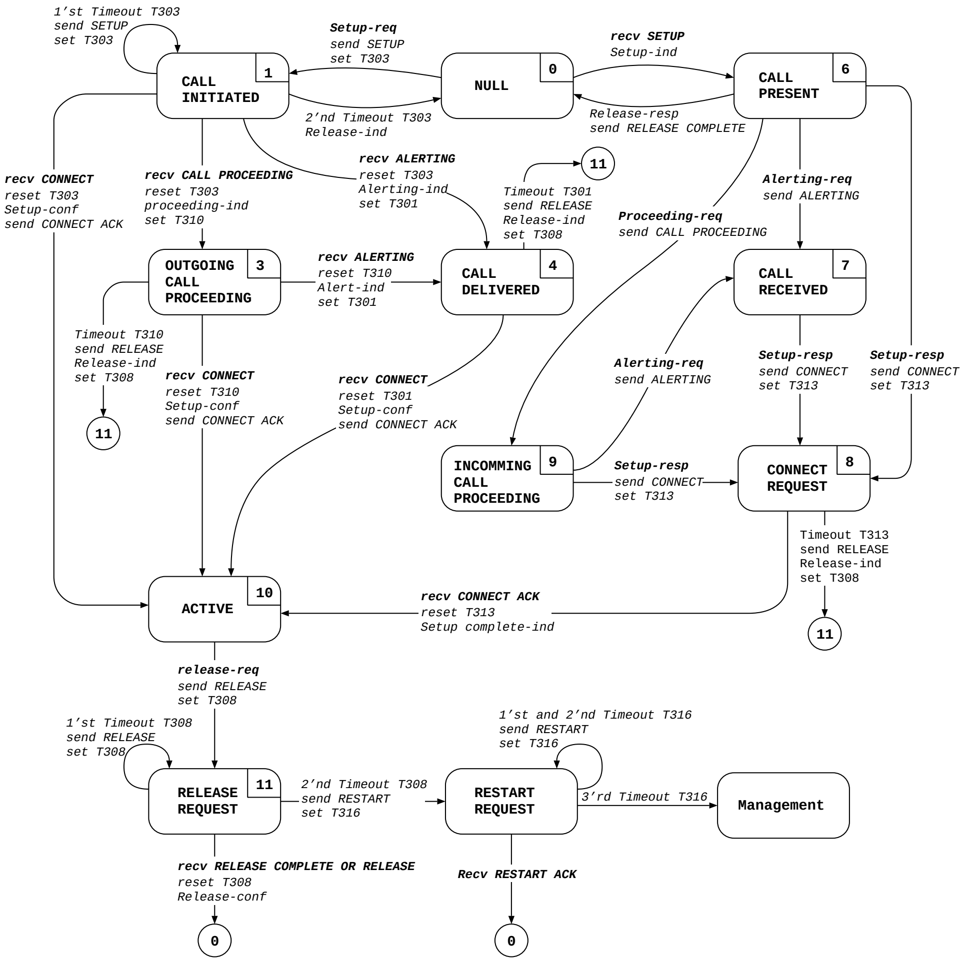

Figure 3: State Diagram of ITU-T Q.2931

2.3.2 Message format

The message and information element structure are shown in figure 4. The header of messages consists of 9 octets and that of information elements consists of 4 octets. The number of information elements depends on the types of messages and the length of content of information elements depends on the types of information elements.

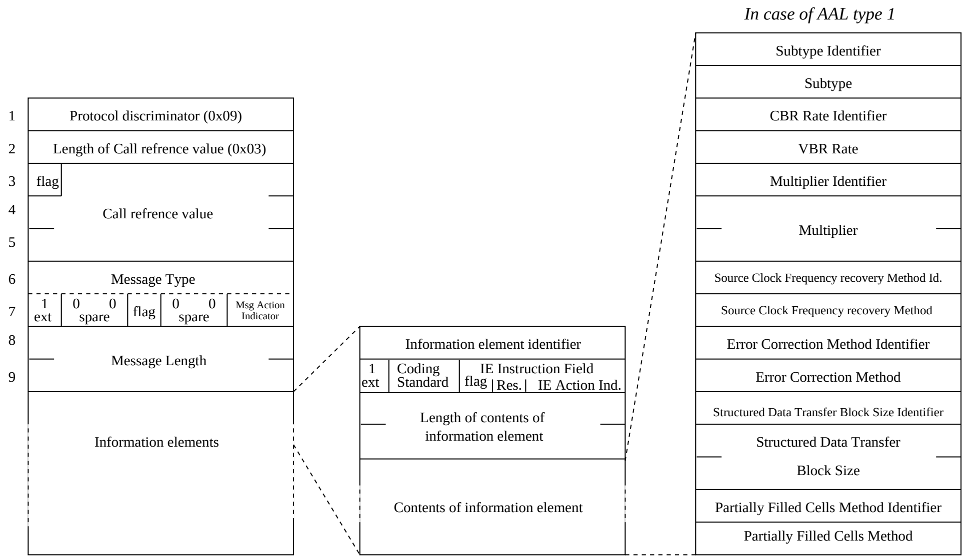

Figure 4: Message structure

2.3.3 Differences between ATM Forum UNI 3.1 and ITU-T Q.2931

There are some differences between ITU-T Q.2931 and ATM Forum UNI. In this section, these differences are mentioned in the view of implementation, not in editorial view.

Followings are not supported in ATM Forum UNI.

- ALERTING[1] and NOTIFY messages.
- "End-to-end transit delay" information element in CONNECT message.
- "Notification indicator" information element in all messages.
- "OAM traffic descriptor" information element in SETUP message.
- Overlap receiving of SETUP message. [2]
- 64 kbit/s based ISDN circuit-mode services.
- Restart of all virtual channels in a virtual path.

Followings are not supported in ITU-T Q.2931.

- Point-to-multipoint call connection, therefore "endpoint reference" information element in all messages.[3]
- in "ATM traffic descriptor" information element "sustainable cell rate" values.

Followins are the differences between ITU-T Q.2931 and ATM Forum UNI.

- In Atm Forum UNI, "called party number" and "connection identifier" information elements in SETUP message are mandatory in the network-to-user direction. However, these are optional, in ITU-T Q.2931.

[1]Therefore, "call received" and "call delivered" state do not exist in ATM Forum.

[2]therefore, "broadband sending complete" information element in SETUP message is optional information element in ATM Forum UNI

[3]Point-to-multipoint call connection is defined in ITU-T Q.2971.

- In ATM Forum UNI, "broadband sending complete" element is optionally included. However, it is mandatory, in ITU-T Q.2931.

2.4 SSCOP

2.4.1 State Transition Diagram

A simplified state transition diagram of SSCOP is described in the figure 5. In this diagram incoming connection pending, incoming resyncronization pending, recovery response pending and incoming recovery pending state are not defined. In the implementation viewpoint, in case of SSCOP user's responses are deterministic, these states can be consider as transition states.

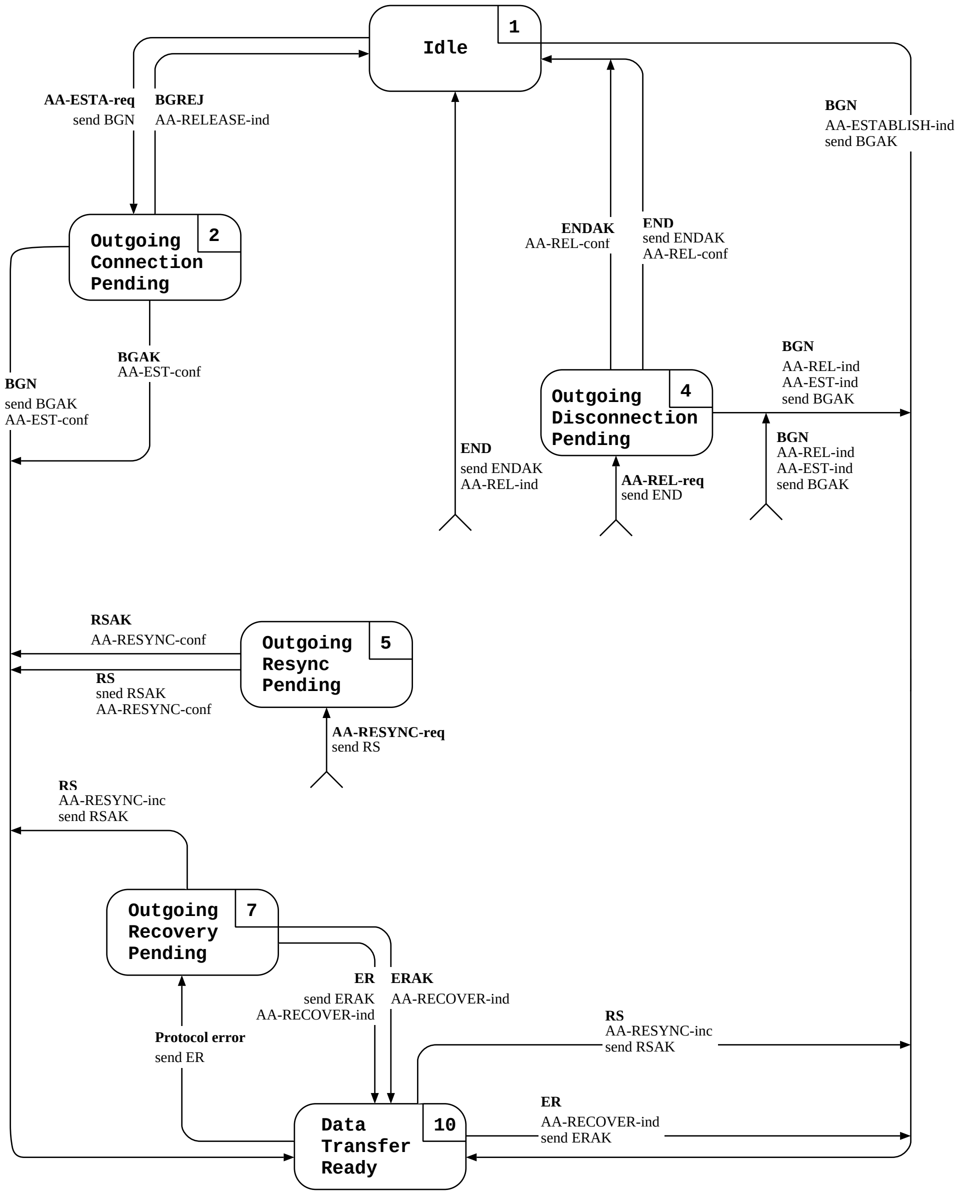

Figure 5: SSCOP states transition diagram

2.4.2 Functions of SSCOP

The SSCOP performs the following functions.

- Sequence Integrity
  This function preserves the order of SSCOP SDUs that submitted for transfer by SSCOP
- Error Correction by Selective Retransmission
  Through a sequencing mechanism, the receiving SSCOP entity can detect missing SSCOP SDUs, This function corrects sequence error through retransmission.
- Flow Control
  This function allows an SSCOP receiver to control the rate at which the peer SSCOP transmitter entity may send information.
- Error Reporting to Layer Management
  This function indicates to layer management errors which have occurred
- Keep Alive
  This function verifies that the two peer SSCOP entities participating in a connection are remainning in link connection established state even in the case of a prolonged absence of data transfer
- Local Data Retrieval
  This function allows the local SSCOP user to retrieve in-sequence SDUs which have not yet been released by the SSCOP entity.
- Connection Control
  This function performs the establishment, release, and resynchronization of an SSCOP connection. It also allows the transmission of variable length user-to-user information without a guarantee of delivery.
- Transfer of User-Data
  This function is used for the conveyance of user data between users of the SSCOP. SSCOP supports both assured and unassured data transfer.
- Protocol Error Detection and Recovery
  This function detects and recovers from errors in the operation of the protocol.
- Status Reporting
  This function allows the transmitter and receiver peer entities to exchange status information.

3 Implementation and Environment

3.1 Environment

As a develop environment of the signalling protocol in UNIX, two types of entity are supposed as shown in figure 6: User entiry and Network entiry.

The user entity consists of two types of processes: ATM protocol server and user applications (figure 6). The ATM protocol server handles all the signalling procedures and supports a lot of user processes' signalling requests. Between these two processes, there is an ATM library and the user can connect to a peer User entity using the functions that are provided by the ATM library. The communication between the ATM library and event handler of the ATM protocol server is performed by sockets communication. The Event handler handles the events of the users' requests

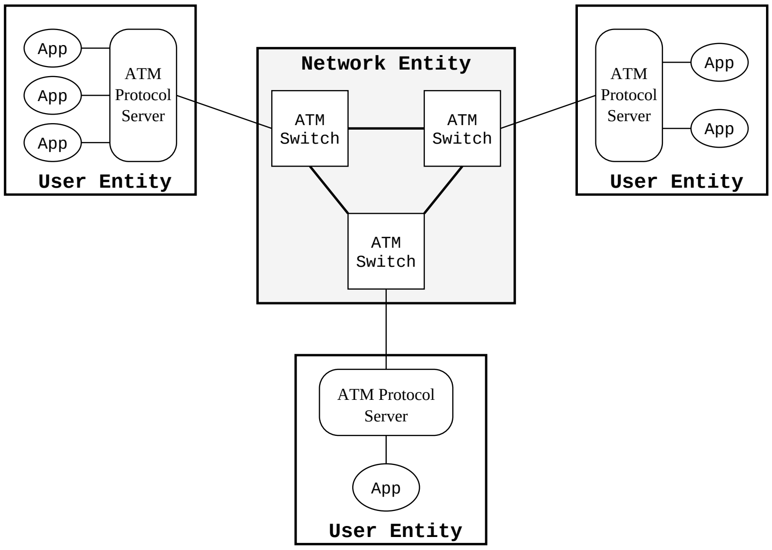

Figure 6: Environment of implementation

and the cells come from the Network entity and are activated by a synchronous I/O multiplexing function.

Generally, it is known that user and network part signalling protocol are exactly same. It is right in the view of protocol description, but in the view of implementation network part program has more complicated date structure because it should handle more than one port and cell switching between the ports. In compare with this, user part program only needs to handle one port. However, user part program has another burden to provide ease to use APIs for application programs.

3.2 Implementation issues

Basically, signalling server is implemented to communicate with both of the ATM Forum UNI and the ITU-T Q.2931. In principle, the communication between ATM Forum UNI and ITU Q.2931 has some conflicts as described in section 2.3.3. To avoid these conflicts, restrictions of both of the specifications have been applied in sending of messages, however, in receiving messages, if the messages can be applied one of the two specifications, these are not treated as erroneous messages. For example, if the "sending complete" information element is not included in the SETUP message, ITU-T Q.2931 will reject this call connection request with a cause #96 "mandatory information element is missing." However, in case of ATM Forum UNI, it will not. Therefore, in this signalling server, sending the complete information element is treated as a mandatory information element in making SETUP message, but it is treated as optional information element in receiving.

However, the inter-operability with ITU Q.2931 or ATM Forum UNI mostly depends on how the user uses the signalling server. As an another example, upon receipt of SETUP message, the signalling server notifies it to the user and the user will select one of the following cases depending on its characteristics of application.

1. sending CALL PROCEEDING, ALERTING and CONNECT message
2. sending CALL PROCEEDING, and CONNECT message
3. sending ALERTING and CONNECT message
4. sending CONNECT message

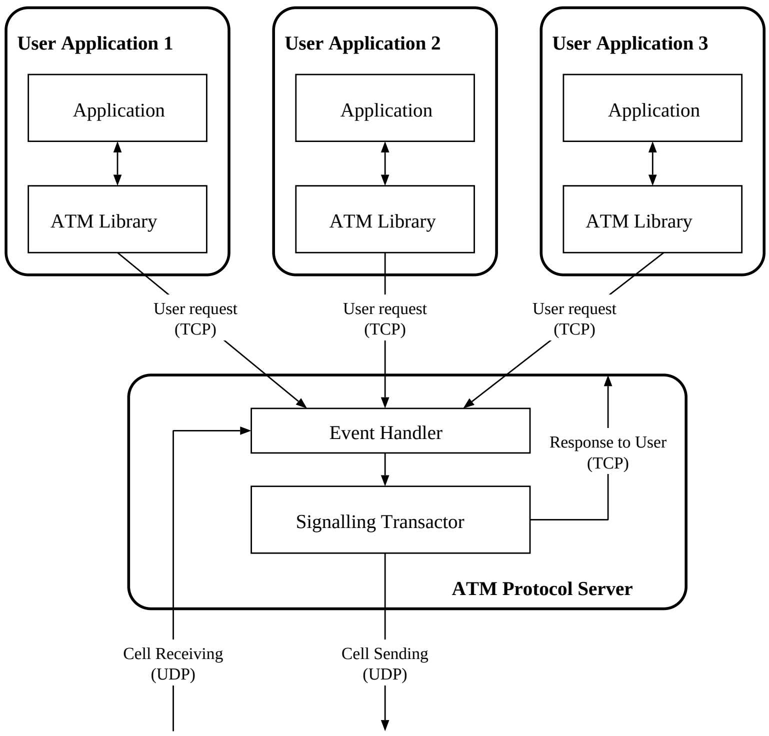

Figure 7: Relation of user application and ATM protocol server

As the response of the SETUP message, if the user selects case 1 or 3, it will be treated as an error in the ATM Forum UNI.

Followings are not supported in this version of signalling server.

- Point-to-multipoint connection.
- Interworking with N_ISDN.
- Overlap receiving of SETUP message.
- Streaming mode service in CPCS.

3.3 S/W Structure

In the experimental environment, the signalling server runs as a simulator in the UNIX network environment.

The signalling server operates in two modes for the test: user and network mode.

As shown in figure 8, only one network mode signalling server exists and all the others act as a user mode.

The physical connections between the user and the network entity are described by the data structure of Port and two types of Port are defined in this: UNI and NNI port. In figure 8, the network entity consists of 3 UNI port and 1 NNI port and the user entities consist of one UNI. Also, it presents that three call connections exist in the whole environment, each user entity handles two calls and network entity handles 6 calls. Every call handled by the signalling server is identifyed by CRV(call reference value). The Port data structure is as follows.

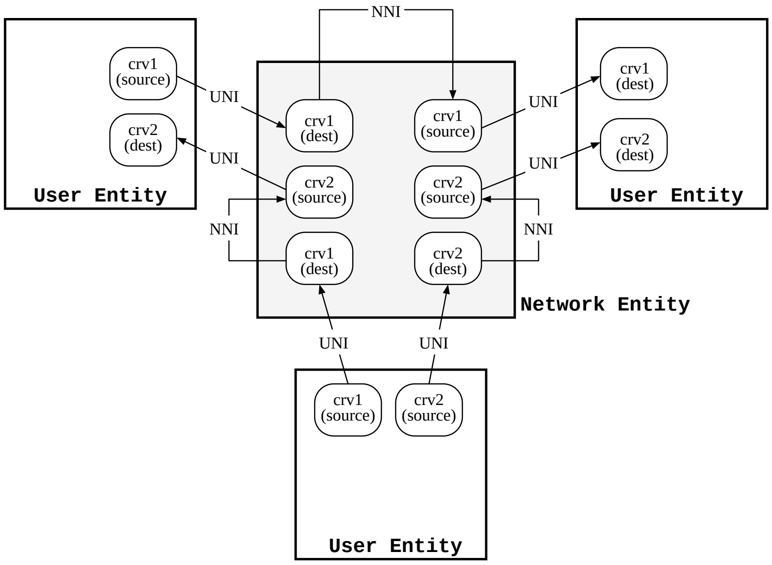

Figure 8: Call connections between user and network entity

```
struct Port {
    Port                prev;
    Port                next;
    int                 port_type;      /* NNI or UNI */
    int                 last_crv;       /* CRV for allocating */
    AtmCall             atm_call_head;  /* ATM calls handled by this port */
    SscopConn           sscop;          /* Sscop handler */
    struct sockaddr_in  pcon;           /* Physical connection ID */
    Q2931Address        *addr;          /* Address of this port */
    int                 tid;            /* Timer id */
    int                 state;          /* State of global CRV */
    int                 address_request_rc;
    int                 cpcs_buf_len;
    u_char              cpcs_buf[MAX_CPCS_DATA_SIZE];
};

struct AtmCall {
    AtmCall             prev;
    AtmCall             next;
    Port                port;           /* Port of this call */
    CallStateValues     state;          /* Call state */
    CallDirection       dir;            /* Call direction */
    int                 crv;            /* Call reference value */
    int                 vpi;            /* VPCI */
    int                 vci;            /* VCI */
    int                 tid;            /* Timer id */
    int                 listen_type;
    int                 release_cond;
    int                 listen_cond;
    int                 listen_time;    /* Timeout for receiving SETUP */
    int                 setup_rc;       /* SETUP retry count */
    int                 release_rc;     /* RELEASE retry count */
    int                 restart_rc;     /* RESTART retry count */
    Q2931Message        setup_mesg;     /* SETUP message for retransmission */
    Q2931Message        release_mesg;   /* RELEASE message for retransmission */
    list                user_elements;
    u_int               route_id_dir;   /* Used for NNI connection */
    u_int               route_id;       /* Used for NNI connection */
};
```

```
typedef struct _SscopConn {
    int         state;          /* SSCOP state */
    int         clear_buffers;
    int         credit;
    int         vt_cc;          /* VT(CC) Connection control state */
    u_char      vt_sq;          /* VT(SQ) Transmitter connection sequence state */
    u_char      vr_sq;          /* VR(SQ) Receiver connection sequence state */
    u_int       vt_s : 24;      /* VT(S) Send state */
    u_int       vt_ps : 24;     /* VT(PS) Poll send state */
    u_int       vt_a : 24;      /* VT(A) Acknowledge state variable */
    u_int       vt_pa : 24;     /* VT(PA) Poll acknowledge state */
    u_int       vt_ms : 24;     /* VT(MS) Maximum send state */
    u_int       vr_mr : 24;     /* VR(MR) Maximum acceptable receive state */
    u_int       vr_r : 24;      /* VR(R) Receive state */
    u_int       vr_h : 24;      /* VR(H) Highest expected state */
    int         vt_pd;          /* VT(PD) Poll Data state */
    SscopBuf    rcv_buf_head;   /* Buffers for receiving */
    SscopBuf    trn_buf_head;   /* Buffers for transmitting */
    SscopNrdBuf nrd_buf_head;   /* Buffers that is not ready to transmit */
  /* Timers' IDs */
    int         cc_tid;
    int         no_response_tid;
    int         keep_alive_tid;
    int         poll_tid;
    int         idle_tid;
} *SscopConn;
```

Data structure constructing Q.2931 messages is the same as that is used in Vince and its data structure is shown in figure 9. This message data structure is based on the Q.2931 message structure in figure 4. The elements and octet groups are connected by single linked lists.

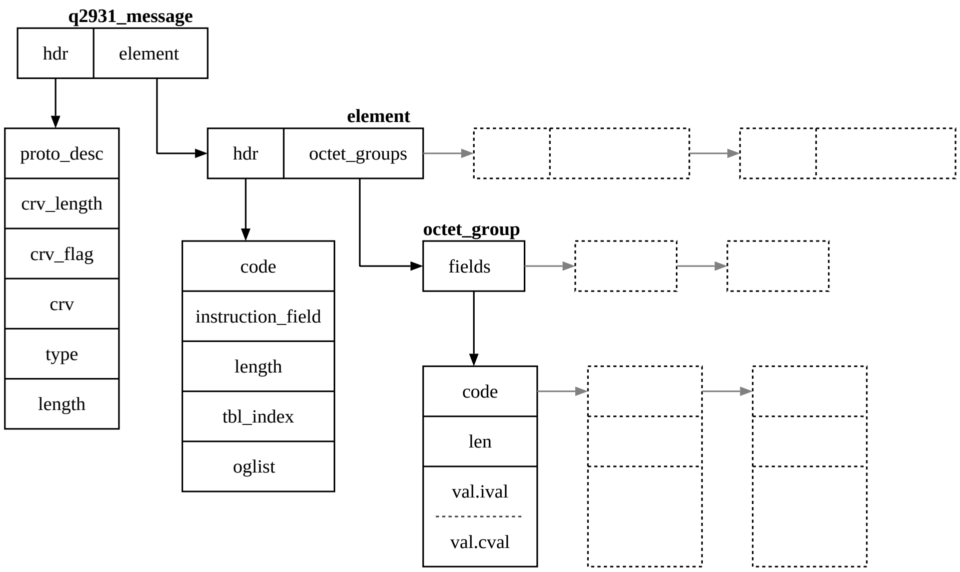

Figure 9: Data structure of message

4 Experiment and Simulation Results

Every sending and receiving messages of user entity can be monitored in the network entity. For this monitoring, some additional messages are defined in the signalling server.

- ADDRESS REQUEST

  This message is used to request network entity to allocate a new address. This is the first message sent by user entity and global call reference value is used for this.

- ADDRESS REQUEST ACKNOWLEDGE

  This message is sent to user entity as a response of ADRRESS REQUEST. Global call reference value is used for this.

- Broken message report

  This message is sent to network entity to report missing of message. This uses different value of protocol discriminator from Q.2931.

- Timeout report

  This message is used to report timeout of user entity.

Following is a case of normal call connection and release. The number after the name of message means call reference value and flag.

```
USER 1                      NETWORK 1                      NETWORK 2                   USER 2
|  ADDRESS REQUEST(0(0))       |                              |                              |
|----------------------------->|                              |                              |
|  ADDRESS REQUEST ACK(0(1))   |                              |                              |
|<-----------------------------|                              |                              |
|                              |                              |  ADDRESS REQUEST(0(0))       |
|                              |                              |<-----------------------------|
|                              |                              |  ADDRESS REQUEST ACK(0(1))   |
|                              |                              |----------------------------->|
|  SETUP(100(0))               |                              |                              |
|----------------------------->|                              |                              |
|  CALL PROCEEDING(100(1))     |                              |                              |
|<-----------------------------|                              |                              |
|                              |  SETUP(1(0))                 |                              |
|                              |------------------------------|                              |
|                              |                              |  SETUP(100(0))               |
|                              |                              |----------------------------->|
|                              |                              |  CALL PROCEEDING(100(1))     |
|                              |                              |<-----------------------------|
|                              |                              |  ALERTING(100(1))            |
|                              |                              |<-----------------------------|
|                              |  ALERTING(1(1))              |                              |
|                              |------------------------------|                              |
|  ALERTING(100(1))            |                              |                              |
|<-----------------------------|                              |                              |
|                              |                              |  CONNECT(100(1))             |
|                              |                              |<-----------------------------|
|                              |  CONNECT(1(1))               |                              |
|                              |------------------------------|                              |
|                              |                              |  CONNECT ACK(100(0))         |
|                              |                              |----------------------------->|
|  CONNECT(100(1))             |                              |                              |
|<-----------------------------|                              |                              |
|  CONNECT ACK(100(0))         |                              |                              |
|----------------------------->|                              |                              |
|                              |                              |                              |
|                              |                              |                              |
|  RELEASE(100(0))             |                              |                              |
|----------------------------->|                              |                              |
|  RELEASE COMPLETE(100(1))    |                              |                              |
|<-----------------------------|                              |                              |
|                              |  RELEASE(1(0))               |                              |
|                              |------------------------------|                              |
|                              |                              |  RELEASE(100(0))             |
|                              |                              |----------------------------->|
|                              |                              |  RELEASE COMPLETE(100(1))    |
|                              |                              |<-----------------------------|
```

Normally, CALL PROCEEDING and ALERTING message need not to send in call setup time. In this case, It is simplified as follows.

```
USER 1                      NETWORK 1                      NETWORK 2                    USER 2
|  SETUP(101(0))               |                              |                              |
|----------------------------->|                              |                              |
|                              |  SETUP(2(0))                 |                              |
|                              |------------------------------|                              |
|                              |                              |  SETUP(101(0))               |
|                              |                              |----------------------------->|
|                              |                              |  CONNECT(101(1))             |
|                              |                              |<-----------------------------|
|                              |                              |  CONNECT ACK(101(0))         |
|                              |                              |----------------------------->|
|                              |  CONNECT(2(1))               |                              |
|                              |------------------------------|                              |
|  CONNECT(101(1))             |                              |                              |
|<-----------------------------|                              |                              |
|  CONNECT ACK(101(0))         |                              |                              |
|----------------------------->|                              |                              |
```

If the called user does not want to receive a call request, the call is rejected with cause value #16 "normal call clearing."

```
USER 1                      NETWORK 1                      NETWORK 2                    USER 2
|  SETUP(107(0))               |                              |                              |
|----------------------------->|                              |                              |
|                              |  SETUP(8(0))                 |                              |
|                              |------------------------------|                              |
|                              |                              |  SETUP(107(0))               |
|                              |                              |----------------------------->|
|                              |                              |  RELEASE COMPLETE(107(1))    |
|                              |                              |<-----------------------------|
|                              |  RELEASE COMPLETE(8(1))      |                              |
|                              |------------------------------|                              |
|  RELEASE COMPLETE(107(1))    |                              |                              |
|<-----------------------------|                              |                              |
```

In case of missing of CONNECT message, SETUP message is retransmitted once more by the network and call is disconnected by the called user after T313 timeout with cause value #102 "recovery on timer expiry.".

```
USER 1                      NETWORK 1                      NETWORK 2                    USER 2
|  SETUP(102(0))               |                              |                              |
|----------------------------->|                              |                              |
|                              |  SETUP(3(0))                 |                              |
|                              |------------------------------|                              |
|  CALL PROCEEDING(102(1))     |                              |                              |
|<-----------------------------|                              |                              |
|                              |                              |  SETUP(102(0))               |
|                              |                              |----------------------------->|
|                              |                              |  CONNECT(102(1))             |
|                              |                              |          X-------------------|
|                              |  Timeout T303                |                              |
|                              |                              |  SETUP(102(0))               |
|                              |                              |----------------------------->|
|                              |                              |                              |
|                              |                              |  Timeout T313                |
|                              |                              |  RELEASE(102(1))             |
|                              |                              |<-----------------------------|
|                              |                              |  RELEASE COMPLETE(102(0))    |
|                              |                              |----------------------------->|
|                              |  RELEASE(3(1))               |                              |
|                              |------------------------------|                              |
|  RELEASE(102(1))             |                              |                              |
|<-----------------------------|                              |                              |
|  RELEASE COMPLETE(102(0))    |                              |                              |
|----------------------------->|                              |                              |
```

When call release fails with some reason, the resources that has been allocated to this call are released by RESTART and RESTART ACKNOWLEDGE message.

```
USER 1                      NETWORK 1                      NETWORK 2                    USER 2
|  RELEASE(110(0))             |                              |                              |
|----------------------------->|                              |                              |
```

```
|  RELEASE COMPLETE(110(1))    |                              |                              |
|<-----------------------------|                              |                              |
|                              |  RELEASE(11(0))              |                              |
|                              |------------------------------|                              |
|                              |                              |  RELEASE(110(0))             |
|                              |                              |----------------------------->|
|                              |                              |  RELEASE COMPLETE(110(1))    |
|                              |                              |         X--------------------|
|                              |  Timeout T308                |                              |
|                              |                              |  RELEASE(110(0))             |
|                              |                              |----------------------------->|
|                              |                              |  RELEASE COMPLETE(110(1))    |
|                              |                              |         X--------------------|
|                              |  Timeout T308                |                              |
|                              |                              |  RESTART(0(0))               |
|                              |                              |----------------------------->|
|                              |                              |  RESTART ACK(0(1))           |
|                              |                              |<-----------------------------|
```

Following is the result of monitoring the SSCOP for first few seconds.

```
   |  BGN(1,3842)           |
   |----------------------->|
   |  BGAK(3842)            |
   |<-----------------------|
   |  SD(0)                 |
   |----------------------->|
   |  SD(0)                 |
   |<-----------------------|
   |                        timer POLL
   |  POLL(1, 1)            |
   |<-----------------------|
timer POLL                  |
   |  POLL(1, 1)            |
   |----------------------->|
   |  STAT(1, 1 {})         |
   |<-----------------------|
   |  STAT(1, 1 {})         |
   |----------------------->|
   |                        timer POLL
   |  POLL(1, 2)            |
   |<-----------------------|
timer POLL                  |
   |  POLL(1, 2)            |
   |----------------------->|
   |  STAT(1, 2 {})         |
   |<-----------------------|
   |  STAT(1, 2 {})         |
   |----------------------->|
```

5 Conclusion

Brief discussion of ATM signalling protocol and implementation issues are described. At first, the network part of signalling server was supposed as just a copy of user part. As the implementation progressed, however, a bit of redundant data structure just for network part needs to be defined. this kind of redundancy has made the whole structure slightly more complicated.

At present, signalling server works as a simulator, though it does not provide complete simulation because many optional functions still need to be implemented and even more the main goal is not making the simulator. Therefore, Some of network entity functions are remained not implemented. As an example, when the call is released by RESTART message by user entity, the remote call connection still exists in active state. [4]

[4] In this case, The network entity has the responsibility to release the remote call connections.