Chapter 2

ROHC

2.1 Overview

Only RFC 3095 [2] which proposes four profiles: IP/UDP, IP/UDP/RTP, IP/ESP, and uncompressed is discussed in this document. Both IPv4 and IPv6 are considered in this RFC. Other profiles such as IP/TCP [9] are under working. Instead of regular IP header, special ROHC header is placed in the front of data payload over the last link. Under the consumption of transporting over unreliable link, such as radio link, many policies are designed for repairing the impairment. That is why ROHC is more robust than other header compression methods.

ROHC comprises two components, compressor and decompressor as shown in Figure 2.1. Both of them have to maintain a context space to record information of each session, such as static fields and possible reference values from previous headers in the packet stream. Context identification (CID) is used to identify and index each session item in the context. Each incoming packet is inspected the IP version, source IP address, destination IP address, source port, destination port, and profile (UDP, RTP, or ESP) to determine which CID the packet should use. If all above fields are identical to some session item already existing in the context, the packet can be compressed or decompressed by using the information indexed by the corresponding CID. If not, a new session item is created into context and records the

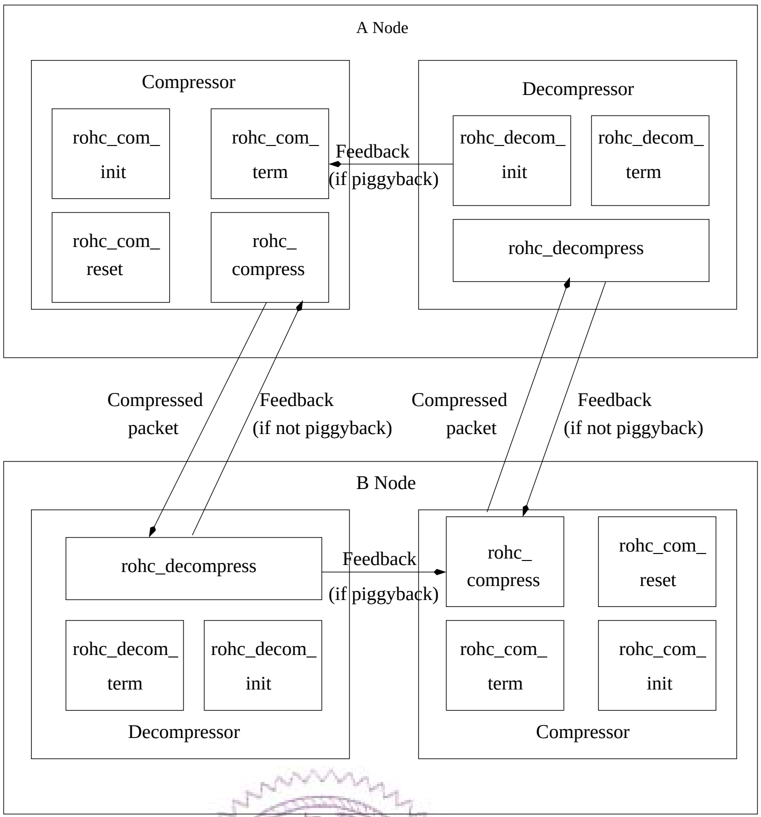

Figure 2.1: Software Architecture

header information of the packet. Because limited memory resource, only fixed numbers of session item can be kept in the context. Least Recently Used (LRU) scheduling algorithm is used to replace session item if the context has already contained predefined maximum number of item. In the following subsections, it is present how to utilize context information to compress or decompress packets.

2.2 Design and Implementation

2.2.1 States and Modes

There are both three states that ROHC compressor and decompressor may stay in. *Initialization and Refresh (IR) State, First Order (FO) State, and Second Order (SO) State* are maintained in compressor. Corresponding to three compressor states are *No Context State, Static Context State, and Full Context State* maintained in decompressor.

Compressor in IR state initializes/re-initializes context data of decompressor by sending complete header information which all fields are in **uncompressed** form plus some additional ROHC information. When compressor enters into FO state, static part of header will not be sent anymore. Only dynamic fields are sent to correct dynamic header information stored in decompressor's context. Finally in SO state, packets carry the least header information which contained no static and dynamic header fields actually. Decompressor must restore the original header by context established before. At decompressor side, decompressor stays in No Context state at begin or when error occurs to decompress static fields. When decompressor believes it has correct static fields information of header, it will enter into Static Context State. Finally, decompressor keeps all correct header information and goes to Full Context state.

IR state is the lowest level state because headers are not be compressed. And SO state is the highest level state in which compressor sends the least bits of header information to network. Equally, No Context State is the lowest level and Full Context state is the highest level state of decompressor. ROHC can perform good performance when compressor and decompressor stay in higher level state for much longer period than lower level ones.

Three operating modes in compressor and decompressor are *Unidirectional mode(U-mode), Bidirectional Optimistic mode (O-mode), and Bidirectional Reliable mode (R-mode).* Initially, compressor must operate in U-mode. After receiving feedback from decompressor,

2.2.1. States and Modes

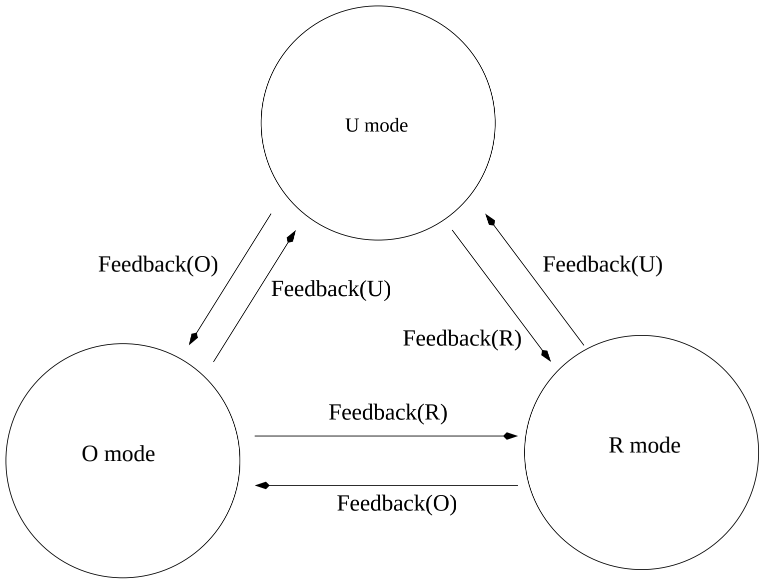

Figure 2.2: ROHC mode transition

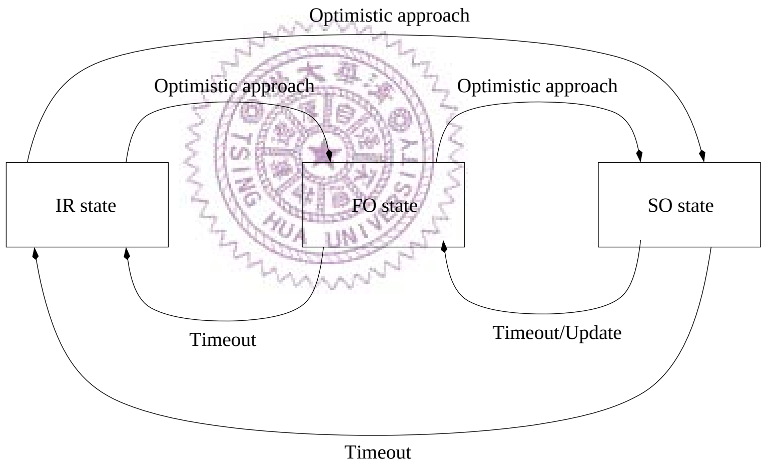

Figure 2.3: ROHC state transition in U-mode

compressor can transit to other mode indicated on the feedback, i.e. mode transition is triggered by decompressor as shown in Figure 2.2.

How state transition works is tight coupled with operating mode. Figure 2.3 2.4 2.5

2.2.1. States and Modes

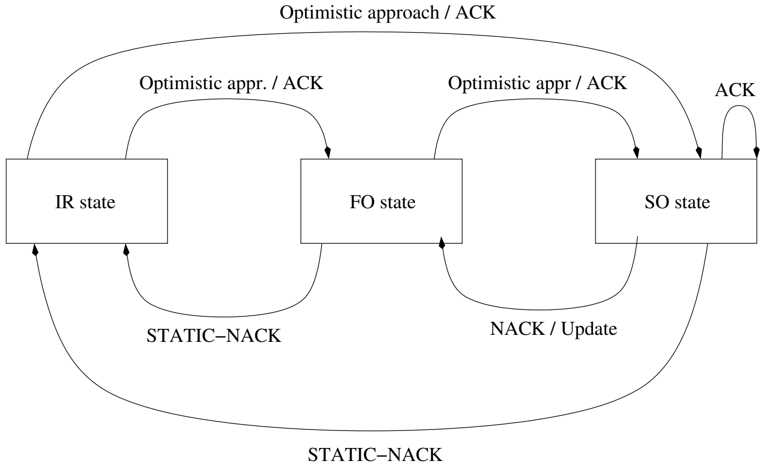

Figure 2.4: ROHC state transition in O-mode

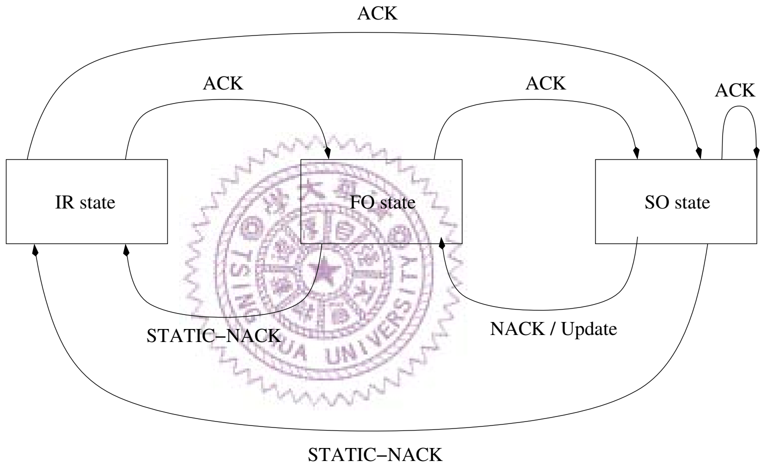

Figure 2.5: ROHC state transition in R-mode

show the state transition in each mode. In U-mode, compressor does not expect any feedback from decompressor. Slow-start algorithm proposed in IPHC [7] is adopted to implement U-mode state transition. Similar transition policy is similar in O-mode and R-mode. Difference between them is caused by frequency of sending feedback. R-mode sends feedback more frequently to assure reliable compressing.

2.2.2 ROHC Packet Format

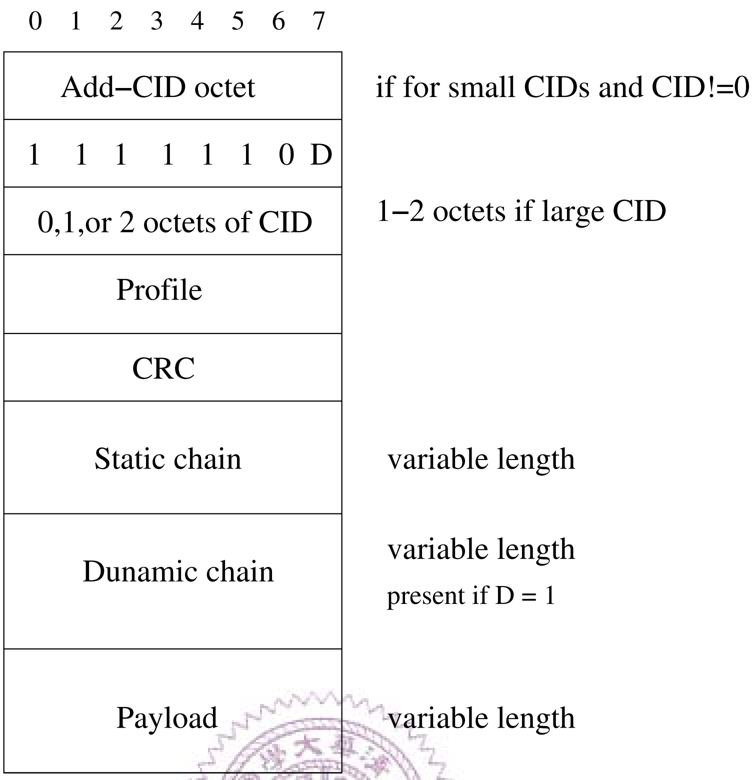

Figure 2.6: ROHC IR packet structure

Table 2.1 shows the classification of header fields. Figure 2.6 2.7 2.8 shows IR packet, IR-DYN packet, and general compressed packet structure separately. Static chain and dynamic chain in Figure 2.6 2.7 are composed of static fields and dynamic fields in Table 2.1. IR packets are sent when compressor is in the IR state and IR-DYN packets are sent in FO state.

There are several compressed packet types in ROHC. Figure 2.8 only shows the general form. They can be simply classified into three basic types that type 0, type 1, and type 2. Table 2.2 shows detail of compressed packets that can be used in each kind of profile. Alphabets in front of type number stand for what operation mode of compressor/decompressor.

Type 0 packets carry the least information which only contains *Sequence Number (SN)*

2.2.2. ROHC Packet Format

0 1 2 3 4 5 6 7	
Add−CID octet	if for small CIDs and CID!=0
1 1 1 1 1 0 0 0	
0,1,or 2 octets of CID	1−2 octets if large CID
Profile	
CRC	
Dunamic chain	variable length
Payload	variable length

Figure 2.7: ROHC IR-DYN packet structure

Protocol	Static Fields	Dynamic Fields
IPv4	Version=4, Protocol, Source Address, Destination Address	Type of Service, Time to Live, Identification, DF, RND, NBO, Generic extension header list
IPv6	Version=6, Flow Label, Next Header, Source Address, Destination Address	Traffic Class, Hop Limit, Generic extension header list
UDP	Source Port, Destination Port	Checksum
RTP	SSRC	Version, P , CC, M, PT, Sequence Number, Timestamp, Generic CSRC list, X, TIS, TSS, TS Stride, Time Stride
ESP	SPI	Sequence Number

Table 2.1: Header Classification

of original header excluding ROHC basic header. All the other information must be derived from context by CID value and SN when decompressor receives such packets. For example, the increasing rate of identification (IP-ID) and SN is the same. Decompressor can get the

2.2.2. ROHC Packet Format

0 1 2 3 4 5 6 7	
Add–CID octet	
first octet of base header	with type indication
0,1,or 2 octets of CID	1–2 octets if large CID
remainder of base header	variable length
extension	variable length
IP–ID of outer IPv4 header	2 octets, if value(RND2)=1
AH data for outer list	variable length
GRE checksum	2 octets, if GRE flag C=1
IP–ID of inner IPv4 header	2 octets, if value(RND1)=1
AH data for inner list	variable length
GRE checksum	2 octets, if GRE flag C=1
UDP checksum	2 octets, if context(UDP checksum) != 0

Figure 2.8: ROHC general compressed packet structure

Profile	Type 0	Type 1	Type 2
UDP and ESP	R-0, R-0-CRC, UO-0	R-1, UO-1	UOR-2
RTP	R-0, R-0-CRC, UO-0	R-1, R-1-ID, R-1-TS, UO-1, UO-1-ID, UO-1-TS	UOR-2, UOR-2-ID, UOR-2-TS

Table 2.2: Detail of Compressed Header

offset by subtracting SN from IP-ID when in the lower level state. If IP-ID $<$ SN, the IP-ID offset value is (IP-ID + 2^{16} - SN). Then decompressor can easily derive IP-ID value from SN in higher level state. Because UDP itself doest not contain SN field, ROHC maintain an SN value for UDP profile.

Type 2 packets carry the most information among compressed ones. In this document, it is designed that type 2 packets can be sent in FO and SO state. Type 0 and type 1 packets can only be sent in SO state.

2.2.3 Window-Based LSB (W-LSB) Encoding

WLSB encoding [10] [2] is a compression skill suitable for SN,IP-ID offset, and TS-SCALED fields in this implementation. The greatest benefit of WLSB encoding is to resist out-of-order and packet lost problems which are often happen in packet-switching network.

In the light of a single header field, compressor/decompressor records the field values of several packets sent before to be references. According to reference_c value, compressor can send fewer bits (LSB) of field to decompressor. After receiving LSB, decompressor can restore original field value according to reference_d.

Compressor needs to maintain a sliding window of reference value (reference_c), but decompressor needs not. All the reference_c values in the window are possible to be reference_d which decompressor uses now, and compressor calculate the LSB according to the sliding window. It is assured to get the correct result of new field value as long as reference_d value is covered by sliding window.

Several symbols are defined as bellow:

V: field value of input packet now

Vmax: The maximum value in the sliding window.

Vmin: The maximum value in the sliding window.

R: Range, R = max (|V - Vmax| , |V - Vmin|)

K: LSB number, K = ceiling (log_2 (2 * R + 1))

Three examples are given to show how to WLSB encoding.

Ex1. Without packet lost and mis-ordering:

Suppose that packets with SN values 279, 280, 281, 282, and 283 have already sent. Only 279 and 283 become to reference values. So sliding window is {279, 283} now. Consider next input packet with SN value 284:

New Value	Vmax	Vmin	R	K
284	283	279	max (|284 - 283| , |284 - 279|) = 5	4

$284_{10} = (10001\mathit{1100})_2$

Because LSB number is 4, compressor sends least significant 4 bits of 284 (1100) to decompressor. If reference_d is 283($10001\mathit{1011}_2$), decompressor replaces the last 4 bits with receiving LSB and gets the value $10001\mathit{1100}_2$. Consider the (K+1)*th* bit from tail, both $1000\mathit{0}1100_2$ (268_{10}) and $1000\mathit{1}1100_2$ (284_{10}) could be the right value. The value which is most close to reference_d is picked. Finally 284_{10} is selected to be the new SN value by decompressor. If reference_d is 279($10001\mathit{0111}_2$), the same result is obtained. After decompressor successes to restore SN, reference_d is changed to new value 284.

Ex2. Packet lost without mis-ordering:

Suppose that packets with SN values 279, 280, 281, 282, and 283 have already sent. Only 279 and 283 become to reference values. So sliding window is {279, 283} now. Subsequence packets 284 to 289 are all lost. Consider next input packet with SN value 290:

New Value	Vmax	Vmin	R	K
290	283	279	max (|290 - 283| , |290 - 279|) = 11	5

$290_10 = (1001\mathit{00010})_2$

The same as above, correct result is obtained.

Ex3. Packet mis-ordering:

Suppose that packets with SN values 279, 280, 281, 282, and 283 have already sent. Only 279 and 283 become to reference values. So sliding window is {279, 283} now. Consider next input packet with SN value 278:

New Value	Vmax	Vmin	R	K
278	283	279	max (|278 - 283| , |278 - 279|) = 5	4

$278_{10} = (10001\mathit{0110})_2$

The same as Ex1, correct result is obtained.

2.2.4 RTP Fields

How to compress RTP Timestamp field [11] is described in this section. RTP timestamp stands for the time when first octet of multimedia instance was sampled. Many audio applications execute 8k Hz sample rate and each voice frame contains 20ms audio data. There is one voice frame in one RTP payload. In the other words, a single voice frame will be sampled 160 times, but timestamp only records the first octet sample time. So RTP timestamp increases 160 per RTP packet.

$$\frac{20ms}{1s} = \frac{TS\_SCALED}{8000}$$

=> TS_SCALED = 160

ROHC utilizes down-scale approach to compress timestamp (TS):

TS = TS_SCALED * TS_STRIDE + TS_OFFSET

In above example, TS_STRIDE is 160. TS_OFFSET is the remainder of TS_now/TS_STRIDE. Compressor only sends TS_SCALED value to decompressor with the same approach of IP-ID compressing described in section 2.2.2 and WLSB encoding when TS_SCALED slope changes.

Unfortunately, not every RTP multimedia codec have fixed TS_STRIDE. All video codec and some audio codec perform non-regular TS_SCALED interval timestamp and just can be compressed by WLSB encoding. This property is relevant to multimedia payload type which is recorded in RTP PT field. Table 2.3 [12] shows relationships between PT value and each codec.

Considering about Table 2.4 [13], each fields in the table is explain as below:

Sample/Frame: This field shows whether the codec is sample-based audio encoding or frame-based audio encoding. Sample-based audio encoding has the property that there is fixed number of bits in each audio frame. Frame-based audio encoding makes a fixed-length data block into a compressed data block. The compressed data block usually has fixed length, too. This encoded data block is called *frame* and several frames can reside in one RTP packet.

Sampling rate: This field shows how many samples in one second.

Ms/frame: This field shows how many milliseconds the audio frame covers in the frame-based audio encoding.

Default ms/packet: This field shows how many milliseconds a RTP packet covers.

Take G728 encoding as example to show how to determine if there exists a fixed TS_SCALE value for the encoding method. G728 encoding has the properties of 8k Hz sampling rate, 2.5ms audio data in a frame, and 20ms audio data in a RTP payload. So there are 8 frames in a RTP payload. The timestamp interval is 160 caculated as above.

Consider about G722 encoding:

$$\frac{20ms}{1s} = \frac{TS\_SCALED}{16000}$$

=> TS_SCALED = 320

In this implementation, only G721, G722, G726-40, G726-32, G726-24, G726-16, G728, G729, G729D, G729E, GSM, GSM-EFR, LPC, QCELP encodings mapped to 2, 3, 7, 9, 12, 15, and 18 PT values can be directly determined non-trivial TS_SCALED value in the same way. To contrast Table 2.3 with Table 2.4, not all down-scalable audio encodings are mapped to fixed PT values. If PT value falls in dynamic or reserved part, a predefined threshold is used to determine if non-trivial TS_SCALED value exists or not, i.e. if continuous RTP packets over than the threshold value have the same TS_SCALED value, it is believed there exists non-trivial TS_SCALED value in this RTP multimedia encoding.

2.3 Performance Analysis

Table 2.5 shows the number of header bytes generated by each protocol.

Because carrying ROHC base header additionally, IR packet size is larger than the original packet. Therefore, compressor and decompressor stay in higher state much longer time than in lower state. After compressor and decompressor accomplish fully information about the session, type 0 packet will be transferred most frequently. As Figure 2.9 ˜ 2.14 show, original packet header can be reduced to type 0 packet header with length between 2 and 4 bytes. Another cost of ROHC is feedback header with length 3-6 bytes in normal case and one feedback is sent at most for one compressed packet.

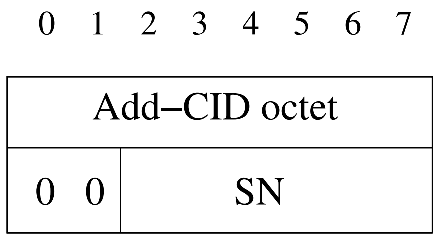

Figure 2.9: R-0 packet with small CID

PT	Codec	Multimedia	Audio channels	Clock rate(Hz)	References
0	PCMU	Audio	1	8000	RFC 3551
1	1016	Audio	1	8000	RFC 3551
2	G721	Audio	1	8000	RFC 3551
3	GSM	Audio	1	8000	RFC 3551
4	G723	Audio	1	8000	
5	DVI4	Audio	1	8000	RFC 3551
6	DVI4	Audio	1	16000	RFC 3551
7	LPC	Audio	1	8000	RFC 3551
8	PCMA	Audio	1	8000	RFC 3551
9	G722	Audio	1	8000	RFC 3551
10	L16	Audio	1	44100	RFC 3551
11	L16	Audio	2	44100	RFC 3551
12	QCELP	Audio	1	8000	
13	CN	Audio	1	8000	RFC 3389
14	MPA	Audio	1	90000	RFC 2250,3551
15	G728	Audio	1	8000	RFC 3551
16	DVI4	Audio	1	11025	
17	DVI4	Audio	1	22050	
18	G729	Audio	1	8000	
19	Reserved				
20–24					
25	CellB	Video		90000	RFC 2029
26	JPEG	Video		90000	RFC 2435
27					
28	nv	Video		90000	RFC 3551
29–30					
31	H261	Video		90000	RFC 2032
32	MPV	Video		90000	RFC 2250
33	MP2T	Audio/Video		90000	RFC 2250
34	H263	Video		90000	
35–71					
72–76	Reserved				RFC 3550
77–95					
96–127	Dynamic				RFC 3551

PT	Codec	Multimedia	Audio channels	Clock rate(Hz)	References
dynamic	GSM-HR	Audio	1	8000	
dynamic	GSM-EFR	Audio	1	8000	
dynamic	L8	Audio	Variable	variable	
dynamic	RED	Audio			
dynamic	VDVI	Audio	Variable		
dynamic	BT656	Video		90000	
dynamic	H263-1998	Video		90000	
dynamic	MP1S	Video		90000	
dynamic	MP2S	Video		90000	
dynamic	BMPEG	Video		90000	

Table 2.3: RTP Payload Type

Encoding	Sample/Frame	bits/sample	Sampling rate	Ms/frame	Default ms/packet
DVI4	Sample	4	var.		20
G722	Sample	8	16000		20
G723	Frame	N/A	8000	30	30
G726-40	Sample	5	8000		20
G726-32	Sample	4	8000		20
G726-24	Sample	3	8000		20
G726-16	Sample	2	8000		20
G728	Frame	N/A	8000	2.5	20
G729	Frame	N/A	8000	10	20
G729D	Frame	N/A	8000	10	20
G729E	Frame	N/A	8000	10	20
GSM	Frame	N/A	8000	20	20
GSM-EFR	Frame	N/A	8000	20	20
L8	Sample	8	var.		20
L16	Sample	16	var.		20
LPC	Frame	N/A	8000	20	20
MPA	Frame	N/A	var.	var.	
PCMA	Sample	8	var.		20
PCMU	Sample	8	var.		20
QCELP	Frame	N/A	8000	20	20
VDVI	Sample	var.	var.		20

Table 2.4: Voice Encoding

Protocol	IPv4	IPv6	UDP	RTP	ESP
Length (Bytes)	20	40 (at least)	8	12 (at least)	8 (at least)

Table 2.5: Original Header Length

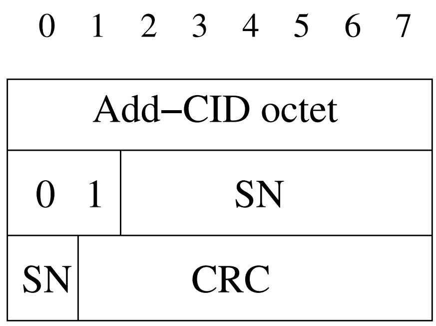

Figure 2.10: R-0-CRC packet with small CID

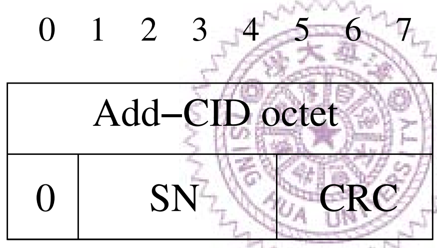

Figure 2.11: UO-0 packet with small CID

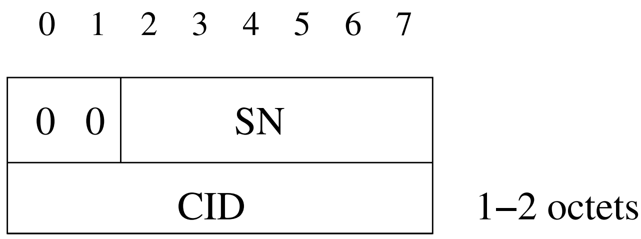

Figure 2.12: R-0 packet with large CID

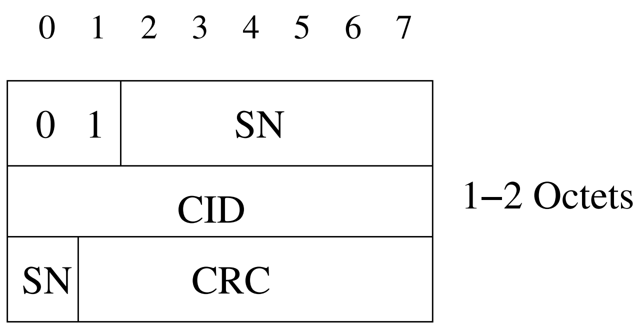

Figure 2.13: R-0-CRC packet with large CID

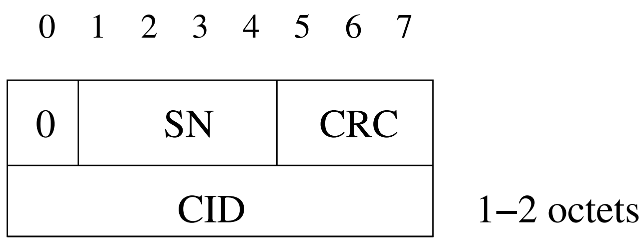

Figure 2.14: UO-0 packet with large CID