**Chapter 5. RTP Control Protocol**

• Components of RTCP

• Transport of RTCP Packets

• RTCP Packet Formats

• Security and Privacy

• Packet Validation

• Participant Database

• Timing Rules

There are two parts to RTP: the data transfer protocol, which was described inHTUChapter 4UTH, and an associated control protocol, which is described in this chapter.

The control protocol, RTCP, provides for periodic reporting of reception quality,participant identification and other source description information, notification on

changes in session membership, and the information needed to synchronize mediastreams.

This chapter describes the uses of RTCP, the format of RTCP packets, and the timingrules used to scale RTCP over the full range of session sizes. It also discusses theissues in building a participant database, using the information contained in RTCPpackets.

**Components of RTCP**

An RTCP implementation has three parts: the packet formats, the timing rules, and

the participant database.

There are several types of RTCP packets. The five standard packet types are

described in the section titled HTURTCP Packet FormatsUTH later in this chapter, along withthe rules by which they must be aggregated into compound packets for

transmission. Algorithms by which implementations can check RTCP packets forcorrectness are described in the section titled HTUPacket ValidationUTH.

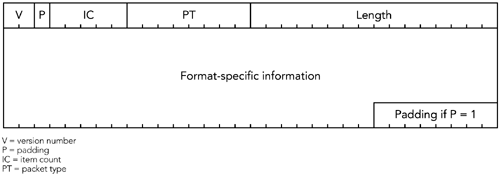
The compound packets are sent periodically, according to the rules described in thesection titled HTUTiming RulesUTH later in this chapter. The interval between packets isknown as the Treporting intervalT. All RTCP activity happens in multiples of the

reporting interval. In addition to being the time between packets, it is the time overwhich reception quality statistics are calculated, and the time between updates ofsource description and lip synchronization information. The interval varies

according to the media format in use and the size of the session; typically it is on theorder of 5 seconds for small sessions, but it can increase to several minutes for verylarge groups. Senders are given special consideration in the calculation of thereporting interval, so their source description and lip synchronization information issent frequently; receivers report less often.

Each implementation is expected to maintain a participant database, based on theinformation collected from the RTCP packets it receives. This database is used to fillout the reception report packets that have to be sent periodically, but also for lipsynchronization between received audio and video streams and to maintain sourcedescription information. The privacy concerns inherent in the participant databaseare mentioned in the section titled Security and Privacy later in this chapter. TheParticipant Database section, also in this chapter, describes the maintenance of theparticipant database.

**Transport of RTCP Packets**

Each RTP session is identified by a network address and a pair of ports: one for RTPdata and one for RTCP data. The RTP data port should be even, and the RTCP portshould be one above the RTP port. For example, if media data is being sent on UDPport 5004, the control channel will be sent to the same address on UDP port 5005.

All participants in a session should send compound RTCP packets and, in turn, will

receive the compound RTCP packets sent by all other participants. Note that

feedback is sent to all participants in a multiparty session: either unicast to a

translator, which then redistributes the data, or directly via multicast. The

peer-to-peer nature of RTCP gives each participant in a session knowledge of allother participants: their presence, reception quality, and—optionally—personaldetails such as name, e-mail address, location, and phone number.

**RTCP Packet Formats**

Five types of RTCP packets are defined in the RTP specification: receiver report (RR),sender report (SR), source description (SDES), membership management (BYE),and application-defined (APP). They all follow a common structure—illustrated inHTUFigure 5.1UTH—although the format-specific information changes depending on thetype of packet.

**Figure 5.1. The Basic RTCP Packet Format**

The header that all five packet types have in common is four octets in length,

comprising five fields:

1. T**Version number (V)**T. The version number is always 2 for the current

version of RTP. There are no plans to introduce new versions, and previousversions are not in widespread use.

2. T**Padding (P)**T. The padding bit indicates that the packet has been padded out

beyond its natural size. If this bit is set, one or more octets of padding havebeen added to the end of this packet, and the last octet contains a count ofthe number of padding octets added. Its use is much the same as the

padding bit in RTP data packets, which was discussed in HTUChapter 4UTH, RTPData Transfer Protocol, in the section titled Padding. Incorrect use of thepadding bit has been a common problem with RTCP implementations; thecorrect usage is described in the sections titled HTUPacking Issues and PacketValidationUTH later in this chapter.

3. T**Item count (IC)**T. Some packet types contain a list of items, perhaps in

addition to some fixed, type-specific information. The item count field is usedby these packet types to indicate the number of items included in the packet(the field has different names in different packet types depending on its use).

Up to 31 items may be included in each RTCP packet, limited also by themaximum transmission unit of the network. If more than 31 items are

needed, the application must generate multiple RTCP packets. An item countof zero indicates that the list of items is empty (this does not necessarilymean that the packet is empty). Packet types that don&apos;t need an item countmay use this field for other purposes.

4. T**Packet type (PT)**T. The packet type identifies the type of information carried

in the packet. Five standard packet types are defined in the RTP specification;other types may be defined in the future (for example, to report additionalstatistics or to convey other source-specific information).

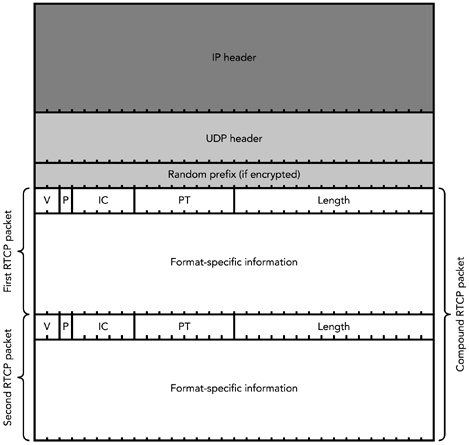
5. T**Length**T. The length field denotes the length of the packet contents following

the common header. It is measured in units of 32-bit words because all RTCPpackets are multiples of 32 bits in length, so counting octets would only allowthe possibility of inconsistency. Zero is a valid length, indicating that thepacket consists of only the four-octet header (the IC header field will also bezero in this case).

Following the RTCP header is the packet data (the format of which depends on thepacket type) and optional padding. The combination of header and data is an RTCPpacket. The five standard types of RTCP packets are described in the sections thatfollow.

RTCP packets are TneverT transported individually; instead they are always groupedtogether for transmission, forming compound packets. Each compound packet isencapsulated in a single lower-layer packet—often a UDP/IP packet—for transport.

If the compound packet is to be encrypted, the group of RTCP packets is prefixed bya 32-bit random value. The structure of a compound packet is illustrated in HTUFigure5.2UTH.

**Figure 5.2. Format of a Compound RTCP Packet**

A set of rules governs the order in which RTCP packets are grouped to form a

compound packet. These rules are described later in the chapter, in the sectiontitled HTUPacking IssuesUTH, after the five types of RTCP packets have been described inmore detail.

**RTCP RR: Receiver Reports**

One of the primary uses of RTCP is reception quality reporting, which is

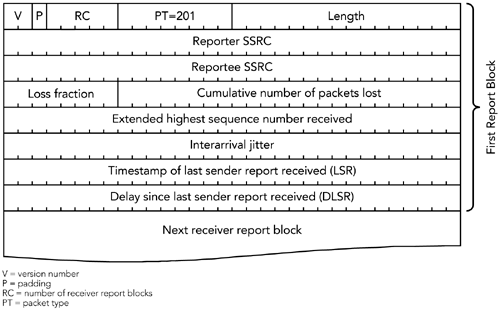
accomplished through RTCP receiver report (RR) packets, which are sent by allparticipants who receive data.

**THE RTCP RR PACKET FORMAT**

A receiver report packet is identified by a packet type of 201 and has the format

illustrated in HTUFigure 5.3UTH. A receiver report packet contains the SSRC

(synchronization source) of the participant who is sending the report (the TreporterTTSSRCT) followed by zero or more report blocks, denoted by the RC field.

**Figure 5.3. Format of an RTCP RR Packet**

Many RTCP packet types have a structure similar to that of receiver reports, with

a list of items following the fixed part of the packet. Note that the fixed part of the

packet remains even if the list of items is empty, implying that a receiver report

with no report blocks will have RC=0, but Length=1, corresponding to the

four octet fixed RTCP header plus the four octet reporter SSRC.

T

- -

T T T

Each report block describes the reception quality of a single synchronization sourcefrom which the reporter has received RTP packets during the current reportinginterval. A total of 31 report blocks can be in each RTCP RR packet. If there are morethan 31 active senders, the receiver should send multiple RR packets in a compoundpacket. Each report block has seven fields, for a total of 24 octets.

The TreporteeT TSSRCT identifies the participant to whom this report block pertains. Thestatistics in the report block denote the quality of reception for the reportee

synchronization source, as received at the participant generating the RR packet.

The Tcumulative number of packets lostT is a 24-bit TsignedT integer denoting thenumber of packets expected, less the number of packets actually received. Thenumber of packets expected is defined to be the extended last sequence numberreceived, less the initial sequence number received. The number of packets

received includes any that are late or duplicated, and hence may be greater than thenumber expected, so the cumulative number of packets lost may be negative. Thecumulative number of packets lost is calculated for the entire duration of the session,not per interval. This field saturates at the maximum positive value of T0x7FFFFFT ifmore packets than that are lost during the session.

The Textended highest sequence numberT received in the RTP data packets from thissynchronization source is calculated as discussed in HTUChapter 4UTH, RTP Data TransferProtocol, in the section titled Sequence Number. Because of possible packet

Many of the RTCP statistics are maintained for the duration of the session, rather

than per reporting interval. If an SSRC collision occurs, however, or if there is a

very large gap in the sequence number space, such that the receiver cannot tell

whether the fields may have wrapped, then the statistics are reset to zero.

reordering, this is not necessarily the extended sequence number of the last RTPpacket received. The extended highest sequence number is calculated per session,not per interval.

The Tloss fractionT is defined as the number of packets lost in this reporting interval,divided by the number expected. The loss fraction is expressed as a fixed-pointnumber with the binary point at the left edge of the field, which is equivalent to theinteger part after multiplying the loss fraction by 256 (that is, if 1/4 of the packetswere lost, the loss fraction would be 1/4 x 256 = 64). If the number of packetsreceived is greater than the number expected, because of the presence of

duplicates, making the number of packets lost negative, then the loss fraction is setto zero.

The Tinterarrival jitterT is an estimate of the statistical variance in network transittime for the data packets sent by the reportee synchronization source. Interarrivaljitter is measured in timestamp units, so it is expressed as a 32-bit unsigned integer,like the RTP timestamp.

To calculate the variance in network transit time, it is necessary to measure thetransit time. Because sender and receiver typically do not have synchronized clocks,however, it is not possible to measure the absolute transit time. Instead the relativetransit time is calculated as the difference between a packet&apos;s RTP timestamp andthe receiver&apos;s RTP clock at the time of arrival, measured in the same units. Thiscalculation requires the receiver to maintain a clock for each source, running at thesame nominal rate as the media clock for that source, from which to derive theserelative timestamps. (This clock may be the receiver&apos;s local playout clock, if thatruns at the same rate as the source clocks.) Because of the lack of synchronizationbetween the clocks of sender and receiver, the relative transit time includes anunknown constant offset. This is not a problem, because we are interested only inthe variation in transit time: the difference in spacing between two packets at thereceiver versus the spacing when they left the sender. In the following computationthe constant offset due to unsynchronized clocks is accounted for by the

subtraction.

If TSTBiB is the RTP timestamp from packet TiT, and TRTBiB is the time of arrival in RTP

timestamp units for packet TiT, then the relative transit time is (TRTBiB – TSTBiB), and for twopackets, TiT and TjT, the difference in relative transit time may be expressed as

Note that the timestamps, TRTBxB and TSTBxB, are 32-bit unsigned integers, whereas TDT(TiT,TjT)is a signed quantity. The calculation is performed with modulo arithmetic (in C, thismeans that the timestamps are of type Tunsigned intT, provided that

Tsizeof(unsigned int) == 4T).

The interarrival jitter is calculated as each data packet is received, using the

difference in relative transit times TDT(TiT,TjT) for that packet and the previous packetreceived (which is not necessarily the previous packet in sequence number order).

The jitter is maintained as a moving average, according to the following formula:

Whenever a reception report is generated, the current value of TJTBiB for the reporteeSSRC is included as the interarrival jitter.

The Tlast sender reportT (TLSRT) timestamp is the middle 32 bits out of the 64-bit NTP(Network Time Protocol) format timestamp included in the most recent RTCP SRpacket received from the reportee SSRC. If no SR has been received yet, the field isset to zero.

The Tdelay since last sender reportT (TDLSRT) is the delay, expressed in units of

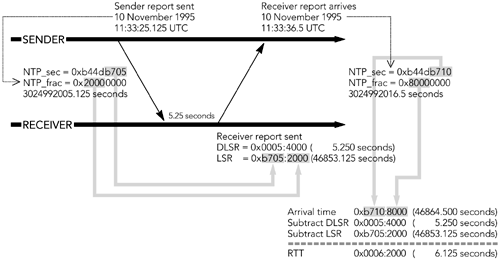
1/65,536 seconds, between receiving the last SR packet from the reportee SSRCand sending this reception report block. If no SR packet has been received from thereportee SSRC, the DLSR field is set to zero.

**INTERPRETING RR DATA**

The reception quality feedback in RR packets is useful not only for the sender, butalso for other participants and third-party monitoring tools. The feedback providedin RR packets can allow the sender to adapt its transmissions according to thefeedback. In addition, other participants can determine whether problems are localor common to several receivers, and network managers may use monitors thatreceive only the RTCP packets to evaluate the performance of their networks.

A sender can use the LSR and DLSR fields to calculate the round-trip time between

it and each receiver. On receiving an RR packet pertaining to it, the sender subtracts

the LSR field from the current time, to give the delay between sending the SR andreceiving this RR. The sender then subtracts the DLSR field to remove the offsetintroduced by the delay in the receiver, to get the network round-trip time. Theprocess is shown in HTUFigure 5.4UTH, an example taken from the RTP specification. (Notethat RFC 1889 contains an error in this example, which has been corrected in thenew version of the RTP specification.)

**Figure 5.4. Sample Round-Trip Time (RTT) Computation.**

Note that the calculated value is the TnetworkT round-trip time, and it excludes anyprocessing at the endpoints. For example, the receiver must buffer the data tosmooth the effects of jitter before it can play the media (see HTUChapter 6UTH, MediaCapture, Playout, and Timing).

The round-trip time is important in interactive applications because delay hindersinteractivity. Studies have shown that it is difficult to conduct conversations whenthe total round-trip time exceeds about 300 millisecondsHTPU61UTPH (this number is

approximate and depends on the listener and the task being performed). A sendermay use knowledge of the round-trip time to optimize the media encoding—forexample, by generating packets that contain less data to reduce packetizationdelays—or to drive the use of error correction codes (see HTUChapter 9UTH, Error

Correction).

The fraction lost gives an indication of the short-term packet loss rates to a receiver.

By watching trends in the reported statistics, a sender can judge whether the loss isa transient or a long-term effect. Many of the statistics in RR packets are cumulativevalues, to allow long-term averaging. Differences can be calculated between anytwo RR packets, making measurements over both short and long periods possibleand giving resilience to the loss of reports.

For example, the packet loss rate over the interval between RR packets can bederived from the cumulative statistics, as well as being directly reported. The

difference in the cumulative number of packets lost gives the number lost duringthat interval, and the difference in the extended last sequence numbers gives thenumber of packets expected during the interval. The ratio of these values is thefraction of packets lost. This number should be equal to the fraction lost field in theRR packet if the calculation is done with consecutive RR packets, but the ratio alsogives an estimate of the loss fraction if one or more RR packets have been lost, andit can show negative loss when there are duplicate packets. The advantage of thefraction lost field is that it provides loss information from a single RR packet. This isuseful in very large sessions, in which the reporting interval is long enough that twoRR packets may not have been received.

Loss rates can be used to influence the choice of media format and error protectioncoding used (see HTUChapter 9UTH, Error Correction). In particular, a higher loss rateindicates that a more loss-tolerant format should be used, and that, if possible, thedata rate should be reduced (because most loss is caused by congestion; see

HTUChapter 2UTH, Voice and Video Communication over Packet Networks, and HTUChapter 10UTH,Congestion Control).

The jitter field may also be used to detect the onset of congestion: A sudden

increase in jitter will often precede the onset of packet loss. This effect depends onthe network topology and the number of flows, with high degrees of statisticalmultiplexing reducing the correlation between increased jitter and the onset ofpacket congestion.

Senders should be aware that the jitter estimate depends on packets being sentwith spacing that matches their timestamp. If the sender delays some packets, thatdelay will be counted as part of the network jitter. This can be an issue with video,where multiple packets are often generated with the same timestamp but are

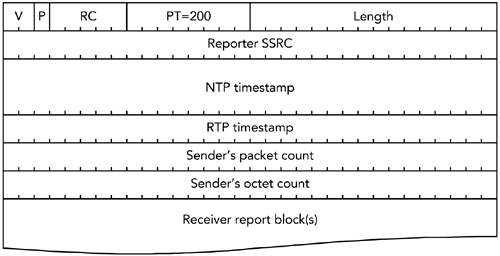
spaced for transmission rather than being sent as a burst. This is not necessarily aproblem, because the jitter measure still gives an indication of the amount of bufferspace that the receiver will require (because the buffer space needs to

accommodate both the jitter and the spacing delay).

**RTCP SR: Sender Reports**

In addition to reception quality reports from receivers, RTCP conveys sender report(SR) packets sent by participants that have recently sent data. These provide

information on the media being sent, primarily so that receivers can synchronizemultiple media streams (for example, to lipsync audio and video).

**THE RTCP SR PACKET FORMAT**

A sender report packet is identified by a packet type of 200 and has the format

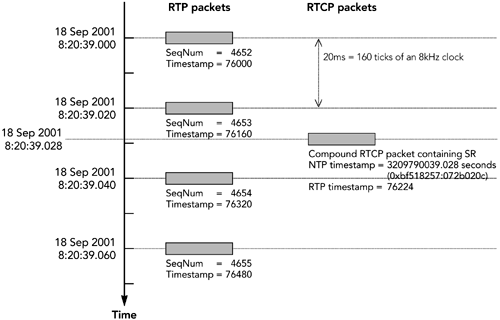
illustrated in HTUFigure 5.5UTH. The payload contains a 24-octet sender information blockfollowed by zero or more receiver report blocks, denoted by the RC field, exactly asif this were a receiver report packet. Receiver report blocks are present when thesender is also a receiver.

**Figure 5.5. Format of an RTCP SR Packet**

The NTP TtimestampT is a 64-bit unsigned value that indicates the time at which thisRTCP SR packet was sent. It is in the format of an NTP timestamp, counting secondssince January 1, 1900, in the upper 32 bits, with the lower 32 bits representingfractions of a second (that is, a 64-bit fixed-point value, with the binary point after32 bits). To convert a UNIX timestamp (seconds since 1970) to NTP time, add2,208,988,800 seconds.

|  |
| --- |
| Although the NTP timestamp in RTCP SR packets uses the format of an NTP  timestamp, the clock does not have to be synchronized with the Network Time  Protocol or have any particular accuracy, resolution, or stability. For a receiver to  synchronize two media streams, however, those streams must be related to the  Tsame clockT. The Network Time ProtocolHTPU5UTPH is occasionally useful for synchronizing  the sending clocks, although it is needed only if the media streams to be  synchronized are generated by different systems. These issues are discussed  further in HTUChapter 7UTH, Lip Synchronization. |
|
|  |

The TRTPT TtimestampT corresponds to the same instant as the NTP timestamp, but itis expressed in the units of the RTP media clock. The value is generally TnotT the same

as the RTP timestamp of the previous data packet, because some time will haveelapsed since the data in that packet was sampled. HTUFigure 5.6UTH shows an example ofthe SR packet timestamps. The SR packet has the RTP timestamp corresponding tothe time at which it is sent, which does TnotT correspond to either of the surroundingRTP data packets.

**Figure 5.6. Use of Timestamps in RTCP SR Packets**

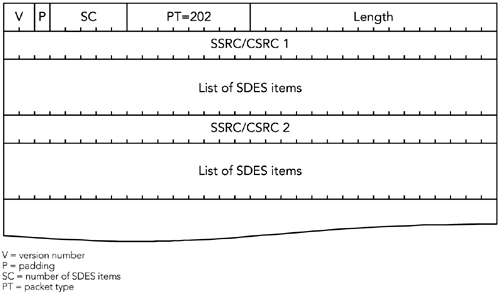
The Tsender&apos;s packet countT is the number of data packets that this synchronizationsource has generated since the beginning of the session. The Tsender&apos;s octet countT isthe number of octets contained in the payload of those data packets (not includingthe headers or any padding).

The packet count and octet count fields are reset if a sender changes its SSRC (forexample, because of a collision). They will eventually wrap around if the sourcecontinues to transmit for a long time, but generally this is not a problem.

Subtraction of an older value from a newer value will give the correct result if 32-bitmodulo arithmetic is used and no more than 2P32P counts occurred in between, evenif there was a wrap-around (in C, this means that the counters are of type TunsignedintT, as long as Tsizeof(unsigned int) == 4T). The packet and octet counts enablereceivers to calculate the average data rate of the source.

**INTERPRETING SR DATA**

From the SR information, an application can calculate the average payload data rateand the average packet rate over an interval without receiving the data. The ratio of

the two is the average payload size. If it can be assumed that packet loss is

independent of packet size, the number of packets received by a particular receiver,multiplied by the average payload size (or the corresponding packet size), gives theapparent throughput available to that receiver.

The timestamps are used to generate a correspondence between media clocks anda known external reference (the NTP format clock). This makes lip synchronizationpossible, as explained in HTUChapter 7UTH.

**RTCP SDES: Source Description**

RTCP can also be used to convey source description (SDES) packets that provideparticipant identification and supplementary details, such as location, e-mail

address, and telephone number. The information in SDES packets is typicallyentered by the user and is often displayed in the graphical user interface of anapplication, although this depends on the nature of the application (for example, asystem providing a gateway from the telephone system into RTP might use theSDES packets to convey caller ID).

**THE RTCP SDES PACKET FORMAT**

Each source description packet has the format illustrated in HTUFigure 5.7UTH and usesRTCP packet type 202. SDES packets comprise zero or more lists of SDES items, theexact number denoted by the SC header field, each of which contains information ona single source.

**Figure 5.7. Format of an RTCP SDES Packet**

It is possible for an application to generate packets with empty lists of SDES items,in which case the SC and length fields in the RTCP common header will both be zero.

In normal use, SC is equal to one (mixers and translators that are aggregatingforwarded information will generate packets with larger lists of SDES items).

Each list of SDES items starts with the SSRC of the source being described, followedby one or more entries with the format shown in HTUFigure 5.8UTH. Each entry starts witha type and a length field, then the item text itself in UTF-8 format.HTPU13UTPH The length fieldindicates how many octets of text are present; the text is TnotT null-terminated.

**Figure 5.8. Format of an SDES Item**

The entries in each SDES item are packed into the packet in a continuous manner,with no separation or padding. The list of items is terminated by one or more nulloctets, the first of which is interpreted as an item of type zero to denote the end ofthe list. No length octet follows the null item type octet, but additional null octetsmust be included if needed to pad until a 32-bit boundary is reached. Note that thispadding is separate from that indicated by the P bit in the RTCP header. A list withzero items (four null octets) is valid but useless.

Several types of SDES items are defined in the RTP specification, and others may bedefined by future profiles. Item type zero is reserved and indicates the end of the listof items. The other standard item types are CNAME, NAME, EMAIL, PHONE, LOC,TOOL, NOTE, and PRIV.

**STANDARD SDES ITEMS**

The CNAME item (type = 1) provides a canonical name (CNAME) for each participant.

It provides a stable and persistent identifier, independent of the synchronizationsource (because the SSRC will change if an application restarts or if an SSRC

collision occurs). The CNAME can be used to associate multiple media streams froma participant across different RTP sessions (for example, to associate voice andvideo that need to be synchronized), and to name a participant across restarts of amedia tool. It is the only mandatory SDES item; all implementations are required tosend SDES CNAME items.

The CNAME is allocated algorithmically from the user name and host IP address ofthe participant. For example, if the author were using an IPv4-based application,the CNAME might be Tcsp@10.7.42.16T. IPv6 applications use the colon-separatednumeric form of the address.HTPU16UTPH If the application is running on a system with no

notion of user names, the host IP address only is used (with no user name or T@Tsymbol).

The use of private addresses and Network Address Translation (NAT) means that

IP addresses are no longer globally unique. For lip synchronization and other

uses of associated RTP sessions to operate correctly through Network Address

Translation, the translator must also translate the RTCP CNAME to a unique form

when crossing domain boundaries. This translation must be consistent across

multiple RTP streams.

As long as each participant joins only a single RTP session—or a related set of

sessions that are intended to be synchronized—the use of user name and host IPaddress is sufficient to generate a consistent unique identifier. If media streamsfrom multiple hosts, or from multiple users, are to be synchronized, then the

senders of those streams must collude to generate a consistent CNAME (whichtypically is the one chosen algorithmically by one of the participants).

The NAME item (type = 2) conveys the participant&apos;s name and is intended primarilyto be displayed in lists of participants as part of the user interface. This value istypically entered by the user, so applications should not assume anything about itsvalue; in particular, it should not be assumed to be unique.

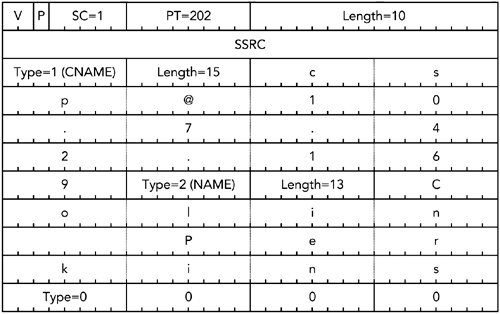
The EMAIL item (type = 3) conveys the e-mail address of a participant formatted asin RFC 822HTPU2UTPH—for example, HTUjdoe@example.comUTH. Sending applications shouldattempt to validate that the EMAIL value is a syntactically correct e-mail addressbefore including it in an SDES item; receivers cannot assume that it is a validaddress.

The PHONE item (type = 4) conveys the telephone number of a participant. The RTPspecification recommends that this be a complete international number, with a plussign replacing the international access code (for example, +1 918 555 1212 for anumber in the United States), but many implementations allow users to enter thisvalue with no check on format.

The LOC item (type = 5) conveys the location of the participant. Many

implementations allow the user to enter the value directly, but it is possible toconvey location in any format. For example, an implementation could be linked tothe Global Positioning System and include GPS coordinates as its location.

The TOOL item (type = 6) indicates the RTP implementation—the tool—in use by theparticipant. This field is intended for debugging and marketing purposes. It shouldinclude the name and version number of the implementation. Typically the user isnot able to edit the contents of this field.

The NOTE item (type = 7) allows the participant to make a brief statement aboutanything. It works well for a &quot;back in five minutes&quot; type of note, but it is not reallysuitable for instant messaging, because of the potentially long delay between RTCPpackets.

PRIV items (type = 8) are a private extension mechanism, used to define

experimental or application-specific SDES extensions. The text of the item beginswith an additional single-octet length field and prefix string, followed by a valuestring that fills the remainder of the item. The intention is that the initial prefixnames the extension and is followed by the value of that extension. PRIV items arerarely used; extensions can more efficiently be managed if new SDES item types aredefined.

The CNAME is the only SDES item that applications are required to transmit. Animplementation should be prepared to receive any of the SDES items, even if itignores them. There are various privacy issues with SDES (see the section titledHTUSecurity and PrivacyUTH later in this chapter), which means that an implementationshould not send any information in addition to the CNAME unless the user hasexplicitly authorized it to do so.

HTUFigure 5.9UTH shows an example of a complete RTCP source description packet

containing CNAME and NAME items. Note the use of padding at the end of the list ofSDES items, to ensure that the packet fits into a multiple of 32 bits.

**Figure 5.9. A Sample SDES Packet**

**PARSER ISSUES**

When implementing a parser for SDES packets, you should remember three

important points:

1. The text of SDES items is not null-terminated, implying that manipulating

SDES items in languages that assume null-terminated strings requires care.

In C, for example, SDES items should be manipulated with Tstrncpy()T,which allow strings up to a specified length to be copied (use of Tstrcpy()T isinappropriate because the text is not null-terminated). Care-less

implementations may be susceptible to buffer overflow attacks, which are aserious security risk.

2. The text of SDES items is in UTF-8 format; local character sets require

conversion before use. It is often necessary to query the locale in use on thesystem, and to convert between the system character set and UTF-8. Someapplications inadvertently generate SDES packets with the wrong characterset; an implementation should be robust to this mistake (for example, if theuse of an incorrect character set causes the UTF-8 parser to produce aninvalid Unicode character).

3. The text of SDES items may be entered by the user and cannot be trusted to

have safe values. In particular, it may contain metacharacters that haveundesirable side effects. For example, some user interface scripting

languages allow command substitution to be triggered by metacharacters,potentially giving an attacker the means to execute arbitrary code.

Implementers should take steps to ensure safe handling of SDES data intheir environment.

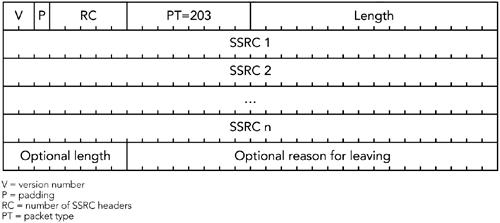
**RTCP BYE: Membership Control**

RTCP provides for loose membership control through RTCP BYE packets, whichindicate that some participants have left the session. A BYE packet is generatedwhen a participant leaves the session, or when it changes its SSRC—for example,because of a collision. BYE packets may be lost in transit, and some applications donot generate them; so a receiver must be prepared to time out participants whohave not been heard from for some time, even if no BYE has been received fromthem.

The significance of a BYE packet depends, to some extent, on the application. Italways indicates that a participant is leaving the RTP session, but there may also bea signaling relationship between the participants (for example, SIP, RTSP, or H.323).

An RTCP BYE packet does not terminate any other relationship between the

participants.

BYE packets are identified by packet type 203 and have the format shown in HTUFigure5.10UTH. The RC field in the common RTCP header indicates the number of SSRCidentifiers in the packet. A value of zero is valid but useless. On receiving a BYEpacket, an implementation should assume that the listed sources have left thesession and ignore any further RTP and RTCP packets from that source. It is

important to keep state for departing participants for some time after a BYE hasbeen received, to allow for delayed data packets.

**Figure 5.10. Format of an RTCP BYE Packet**

The section titled Participant Database later in this chapter further describes thestate maintenance issues relating to timeout of participants and RTCP BYE packets.

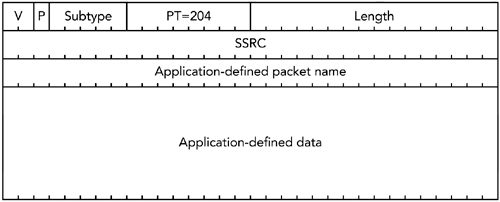
A BYE packet may also contain text indicating the reason for leaving a session,

suitable for display in the user interface. This text is optional, but an implementationmust be prepared to receive it (even though the text may be ignored).

**RTCP APP: Application-Defined RTCP Packets**

The final class of RTCP packet (APP) allows for application-defined extensions. It haspacket type 204, and the format shown in HTUFigure 5.11UTH. The Tapplication-definedpacket nameT is a four-character prefix intended to uniquely identify this extension,with each character being chosen from the ASCII character set, and uppercase andlowercase characters being treated as distinct. It is recommended that the packetname be chosen to match the application it represents, with the choice of subtypevalues being coordinated by the application. The remainder of the packet is

application-specific.

**Figure 5.11. Format of an RTCP APP Packet**

Application-defined packets are used for nonstandard extensions to RTCP, and for

experimentation with new features. The intent is that experimenters use APP as afirst place to try new features, and then register new packet types if the featureshave wider use. Several applications generate APP packets, and implementationsshould be prepared to ignore unrecognized APP packets.

**Packing Issues**

As noted earlier, RTCP packets are never sent individually, but rather are packed

into a compound packet for transmission. Various rules govern the structure ofcompound packets, as detailed next.

If the participant generating the compound RTCP packet is an active data sender,the compound must start with an RTCP SR packet. Otherwise it must start with anRTCP RR packet. This is true even if no data has been sent or received, in which casethe SR/RR packet contains no receiver report blocks (the RC header field is zero).

On the other hand, if data is received from many sources and there are too manyreports to fit into a single SR/RR packet, the compound should begin with an SR/RRpacket followed by several RR packets.

Following the SR/RR packet is an SDES packet. This packet must include a CNAMEitem, and it may include other items. The frequency of inclusion of the other

(non-CNAME) SDES items is determined by the RTP profile in use. For example, theaudio/video profileHTPU7UTPH specifies that other items may be included with every thirdcompound RTCP packet sent, with a NAME item being sent seven out of eight timeswithin that slot and the remaining SDES cyclically taking up the eighth slot. Otherprofiles may specify different choices.

BYE packets, when ready for transmission, must be placed as the last packet in acompound. Other RTCP packets to be sent may be included in any order. These

strict ordering rules are intended to make packet validation easier because it ishighly unlikely that a misdirected packet will meet these constraints.

A potentially difficult issue in the generation of compound RTCP packets is how to

handle sessions with larger numbers of active senders. If there are more than 31active senders, it is necessary to include additional RR packets within the compound.

This may be repeated as often as is required, up to the maximum transmission unit(MTU) of the network. If there are so many senders that the receiver reports cannotall fit within the MTU, the receiver reports for some senders must be omitted. In thatcase, reports that are omitted should be included in the next compound packetgenerated (requiring a receiver to keep track of the sources reported on in eachinterval).

A similar issue arises when the SDES items to be included within the packet exceed

the maximum packet size. The trade-off between including additional receiverreports and including source description information is left to the implementation.

There is no single correct solution.

Sometimes it is necessary to pad a compound RTCP packet out beyond its naturalsize. In such cases the padding is added to the last RTCP packet in the compoundonly, and the P bit is set in that last packet. Padding is an area where some

implementations are incorrect; the section titled Packet Validation later in thischapter discusses common problems.

**Security and Privacy**

Various privacy issues are inherent in the use of RTCP—in particular, source

description packets. Although these packets are optional, their use can exposesignificant personal details, so applications should not send SDES information

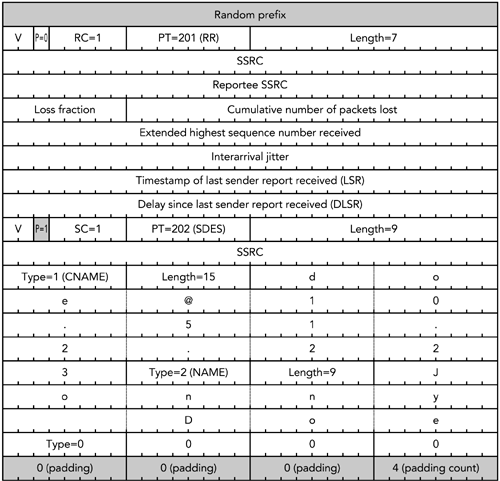
without first informing the user that the information is being made available.

The use of SDES CNAME packets is an exception because these packets are

mandatory. The inclusion of an IP address within CNAME packets is a potential issue.

However, the same information is available from the IP header of the packet. If theRTP packets pass through Network Address Translation (NAT), the translation of theaddress in the IP header that is performed should also be performed on the addressin the CNAME. In practice, many NAT implementations are unaware of RTP, so thereis a potential for leakage of the internal IP address.

The exposure of user names may be a greater concern—in which case applicationsmay omit or rewrite the user name, provided that this is done consistently amongthe set of applications using CNAME for association.

Some receivers may not want their presence to be visible. It is acceptable if thosereceivers do not send RTCP at all, although doing so prevents senders from usingthe reception quality information to adapt their transmission to match the receivers.

To achieve confidentiality of the media stream, RTCP packets may be encrypted.

When encrypted, each compound packet contains an additional 32-bit randomprefix, as illustrated in HTUFigure 5.12UTH, to help avoid plain-text attacks.

**Figure 5.12. Example of an Encrypted RTCP Packet, Showingthe Correct Use of Padding**

Security and privacy are discussed in more detail in HTUChapter 13UTH, Security

Considerations.

**Packet Validation**

It is important to validate whether received packets really are RTP or RTCP. Thepacking rules, mentioned earlier, allow RTCP packets to be rigorously validated.

Successful validation of an RTCP stream gives high assurance that the

corresponding RTP stream is also valid, although it does not negate the need forvalidation of the RTP packets.

HUListing 5.1UH shows the pseudocode for the validation process. These are the keypoints:

• All packets must be compound RTCP packets.

• The version field of all packets must equal 2.

• The packet type field of the first RTCP packet in a compound packet must be

equal to SR or RR.

• If padding is needed, it is added to only the last packet in the compound. The

padding bit should be zero for all other packets in the compound RTCPpacket.

• The length fields of the individual RTCP packets must total the overall length

of the compound RTCP packet as received.

Because new RTCP packet types may be defined by future profiles, the validationprocedure should not require each packet type to be one of the five defined in theRTP specification.

**Listing 5.1 Pseudocode for Packet Validation**

validate\_rtcp(rtcp\_t \*packet, int length)

rtcp\_t \*end = (rtcp\_t \*) (((char \*) packet) + length);

rtcp\_t \*r = packet;

int l = 0;

int p = 0;

// All RTCP packets must be compound packets

if ((packet-&gt;length+ 1) \* 4) == length) {

... error: not a compound packet

}

// Check the RTCP version, packet type, and padding of the first// in the compound RTCP packet...

if (packet-&gt;version != 2) {

...error: version number != 2 in the first subpacket

}

if (packet-&gt; p != 0) {

...error: padding bit is set on first packet in compound

}

if ((packet-&gt;pt != RTCP\_SR) &amp;&amp; (packet-&gt;pt != RTCP\_RR)) {

...error: compound packet does not start with SR or RR

}

// Check all following parts of the compound RTCP packet. The RTP

// version number must be 2, and the padding bit must be zero on// all except the last packet.

}

}

do {

if (p == 1) {

...error: padding before last packet in compound

}

if (r-&gt; p) {

p = 1;

if (r-&gt; version != 2) {

...error: version number != 2 in subpacket

}

l += (r-&gt;length + 1) \* 4;

r = (rtcp\_t \*) (((uint32\_t \*) r) + r-&gt;length + 1);

} while (r &lt; end);

// Check that the length of the packets matches the length of the// UDP packet in which they were received...

if ((l != length) || (r != end)) {

...error: length does not match UDP packet length

}

...packet is valid

One common implementation problem causes packets to fail their validity test:When you&apos;re padding compound RTCP packets beyond their natural length, youneed to ensure that the padding is added to only the last packet in the compound.

A common mistake has been to add the padding to the last packet, but to set the P

bit in the header of the first packet in the compound. The P bit must be set only inthe last packet.

It is possible to detect RTCP misdirected onto the RTP port via the packet type field.

The standard RTCP packets have packet type values with the high bit set; if they aremisdirected onto the RTP port, the high bit of the packet type field will fall into theplace of the M bit in the RTP header. With the top bit stripped, the standard RTCPpacket types correspond to an RTP payload type in the range 72 to 76. This range isreserved in the RTP specification and will not be used for valid RTP data packets, sodetection of packets in this range implies that the stream is misdirected. Similarly,RTP packets sent to the RTCP port may clearly be distinguished by their packet type,which will be outside the valid range for RTCP packet types.

**Participant Database**

Each application in an RTP session will maintain a database of information about theparticipants and about the session itself. The session information, from which theRTCP timing is derived, can be stored as a set of variables:

• The RTP bandwidth—that is, the typical session bandwidth, configured when

the application starts.

• The RTCP bandwidth fraction—that is, the percentage of the RTP bandwidth

devoted to RTCP reports. This is usually 5%, but profiles may define a meansof changing this (0% also may be used, meaning that RTCP is not sent).

• The average size of all RTCP packets sent and received by this participant.

• The number of members in the session, the number of members when this

participant last sent an RTCP packet, and the fraction of those who have sentRTP data packets during the preceding reporting interval.

• The time at which the implementation last sent an RTCP packet, and the next

scheduled transmission time.

• A flag indicating whether the implementation has sent any RTP data packets

since sending the last two RTCP packets.

• A flag indicating whether the implementation has sent any RTCP packets at

all.

In addition, the implementation needs to maintain variables to include in RTCP SRpackets:

• The number of packets and octets of RTP data it has sent.

• The last sequence number it used.

• The correspondence between the RTP clock it is using and an NTP-format

timestamp.

A session data structure containing these variables is also a good place to store the

SSRC being used, the SDES information for the implementation, and the file

descriptors for the RTP and RTCP sockets. Finally, the session data structure shouldcontain a database for information held on each participant.

In terms of implementation, the session data can be stored simply: a single

structure in a C-based implementation, a class in an object-oriented system. Withthe exception of the participant-specific data, each variable in the structure or classis a simple type: integer, text string, and so on. The format of the

participant-specific data is described next.

To generate RTCP packets properly, each participant also needs to maintain statefor the other members in the session. A good design makes the participant databasean integral part of the operation of the system, holding not just RTCP-related

information, but all state for each participant. The per-participant data structuremay include the following:

Some implementations use less-than-perfect random number generators when

choosing their SSRC identifier. This means that a simple hashing function—for

example, using the lowest few bits of the SSRC as an index into a table—can lead

to unbalanced and inefficient operation. Even though SSRC values are supposed

to be random, they should be used with an efficient hashing function. Some have

suggested using the MD5 hash of the SSRC as the basis for the index, although

that may be considered overkill.

• SSRC identifier.

• Source description information: the CNAME is required; other information

may be included (note that these values are not null-terminated, and caremust be taken in their handling).

• Reception quality statistics (packet loss and jitter), to allow generation of

RTCP RR packets.

• Information received from sender reports, to allow lip synchronization (see

HUChapter 7UH).

• The last time this participant was heard from so that inactive participants can

be timed out.

• A flag indicating whether this participant has sent data within the current

RTCP reporting interval.

• The media playout buffer, and any codec state needed (see HUChapter 6UH,

Media Capture, Playout, and Timing).

• Any information needed for channel coding and error recovery—for example,

data awaiting reception of repair packets before it can be decoded (seeHUChapters 8UH, Error Concealment, and HU9UH, Error Correction).

Within an RTP session, members are identified by their synchronization sourceidentifier. Because there may be many participants and they may need to be

accessed in any order, the appropriate data structure for the participant database isa hash table, indexed by SSRC identifier. In applications that deal with only a singlemedia format, this is sufficient. However, lip synchronization also requires thecapability to look up sources by their CNAME. As a result, the participant databaseshould be indexed by a double hash table: once by SSRC and once by CNAME.

Participants should be added to the database after a validated packet has beenreceived from them. The validation step is important: An implementation does notwant to create a state for a participant unless it is certain that the participant is valid.

Here are some guidelines:

• If an RTCP packet is received and validated, the participant should be

entered into the database. The validity checks on RTCP are strong, and it isdifficult for bogus packets to satisfy them.

• An entry should not be made on the basis of RTP packets only, unless

multiple packets are received with consecutive sequence numbers. Thevalidity checks possible for a single RTP packet are weak, and it is possiblefor a bogus packet to satisfy the tests yet be invalid.

This implies that the implementation should maintain an additional, lightweighttable of probationary sources (sources in which only a single RTP packet has beenreceived). To prevent bogus sources of RTP and RTCP data from using too muchmemory, this table should be aggressively timed out and should have a fixed

maximum size. It is difficult to protect against an attacker who purposely generatesmany different sources to use up all memory of the receivers, but these precautionswill prevent accidental exhaustion of memory if a misdirected non-RTP stream isreceived.

Each CSRC (contributing source) in a valid RTP packet also counts as a participantand should be added to the database. You should expect to receive SDES

information for participants identified only by CSRC.

When a participant is added to the database, an application should also update thesession-level count of the members and the sender fraction. Addition of a

participant may also cause RTCP forward reconsideration, which will be discussedshortly.

Participants are removed from the database after a BYE packet is received or aftera specified period of inactivity. This sounds simple, but there are several subtlepoints.

There is no guarantee that packets are received in order, so an RTCP BYE may bereceived before the last data packet from a source. To prevent state from being torndown and then immediately reestablished, a participant should be marked as havingleft after a BYE is received, and its state should be held over for a few seconds (myimplementation uses a fixed two-second delay). The important point is that thedelay is longer than both the maximum expected reordering and the media playoutdelay, thereby allowing for late packets and for any data in the playout buffer forthat participant to be used.

Sources may be timed out if they haven&apos;t been heard from for more than five timesthe reporting interval. If the reporting interval is less than 5 seconds, the 5-secondminimum is used here (even if a smaller interval is used when RTCP packets arebeing sent).

When a BYE packet is received or when a member times out, RTCP reverse

reconsideration takes place, as described in the section titled HUBYE ReconsiderationUHlater in this chapter.

**Timing Rules**

The rate at which each participant sends RTCP packets is not fixed but varies

according to the size of the session and the format of the media stream. The aim isto restrict the total amount of RTCP traffic to a fixed fraction—usually 5%—of thesession bandwidth. This goal is achieved by a reduction in the rate at which eachparticipant sends RTCP packets as the size of the session increases. In a two-partytelephone call using RTP, each participant will send an RTCP report every fewseconds; in a session with thousands of participants—for example, an Internet radiostation—the interval between RTCP reports from each listener may be many

minutes.

Each participant decides when to send RTCP packets on the basis of the set of rulesdescribed later in this section. It is important to follow these rules closely, especiallyfor implementations that may be used in large sessions. If implemented correctly,RTCP will scale to sessions with many thousands of members. If not, the amount ofcontrol traffic will grow linearly with the number of members and will cause

significant network congestion.

**Reporting Interval**

Compound RTCP packets are sent periodically, according to a randomized timer.

The average time each participant waits between sending RTCP packets is known asthe reporting interval. It is calculated on the basis of several factors:

• **T he bandwidth allocated to RTCP**T. This is a fixed fraction—usually

5%—of the session bandwidth. The session bandwidth is the expected datarate for the session; typically this is the bit rate of a single stream of audio orvideo data, multiplied by the typical number of simultaneous senders. Thesession bandwidth is fixed for the duration of a session, and supplied as aconfiguration parameter to the RTP application when it starts.

The fraction of the session bandwidth allocated to RTCP can be varied by theRTP profile in use. It is important that all members of a session use the samefraction; otherwise state for some members may be prematurely timed out.

• **T he average size of RTCP packets sent and received**T. The average size

includes not just the RTCP data, but also the UDP and IP header sizes (thatis, add 28 octets per packet for a typical IPv4 implementation).

• **T he total number of participants and the fraction of those**

**participants who are senders**T. This requires an implementation to

maintain a database of all participants, noting whether they are senders(that is, if RTP data packets or RTCP SR packets have been received fromthem) or receivers (if only RTCP RR, SDES, or APP packets have been

received). The earlier section titled Participant Database explained this indetail.