COMP 249 Advanced Distributed Systems
Multimedia Networking

The Multimedia Control Protocol RTCP

Kevin Jeffay
Department of Computer Science
University of North Carolina at Chapel Hill
jeffay@cs.unc.edu
October 5, 1999

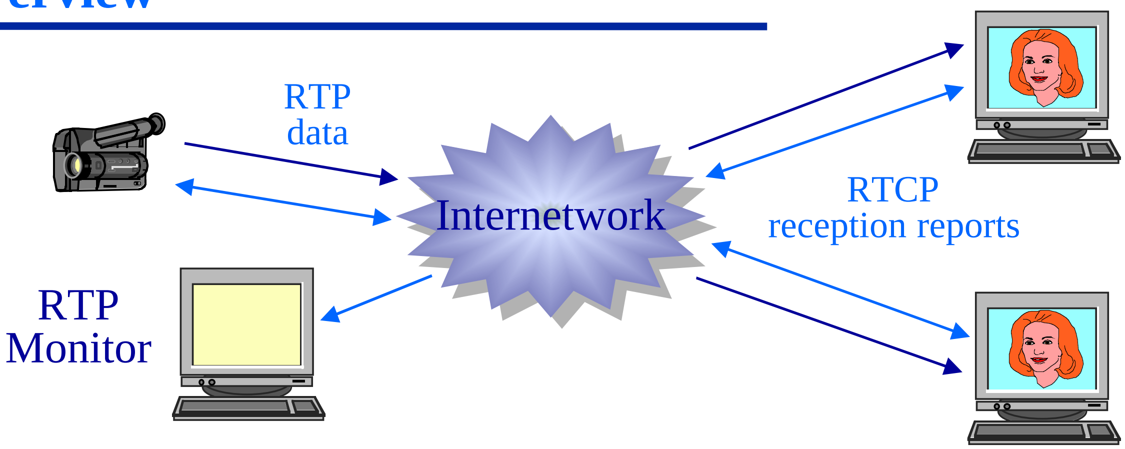
http://www.cs.unc.edu/~jeffay/courses/comp249f99

The Multimedia Transport Protocol RTP
Outline

- RTP concepts
  - Entities and abstractions
- Protocol definition
  - Header format and packet structure
- Developing interoperable applications with RTP
  - RTP profiles
- Quality-of-service monitoring and reporting
  - Real-time control protocol RTCP

The Real-Time Control Protocol RTCP

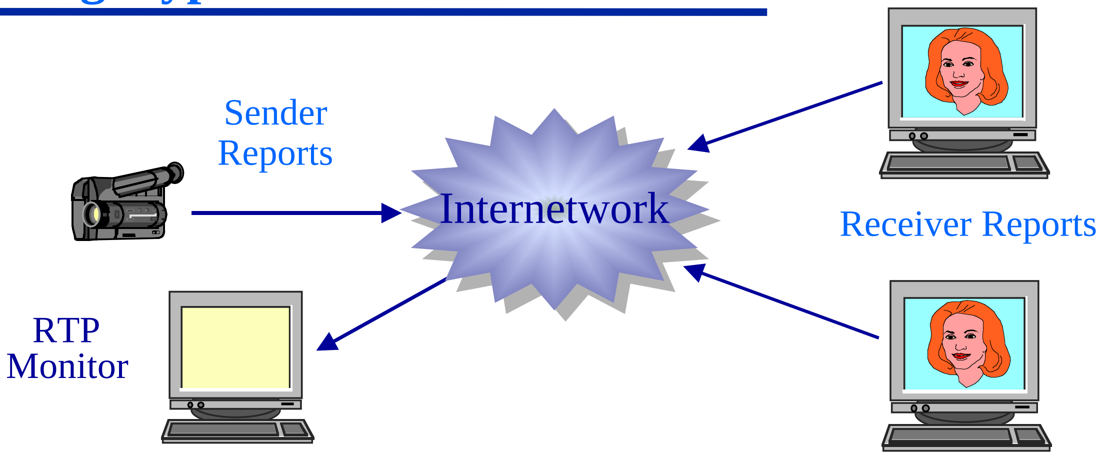
Overview



- Senders & receivers periodically generate reports of various session statistics and multicast to the group
- RTCP enables...
  - » Diagnosis of faults in the multicast distribution tree
  - » Congestion control
  - » Third party performance monitoring & logging
  - » (Simple) conference control

The Real-Time Control Protocol RTCP

Message types



- Sender reports (SR)
  - » cumulative frame & byte counts
  - » wall clock/timestamp values
- Receiver reports (RR)
  - » frame loss/Frame delivery rate
- Source description (SDES) items
  - » useful ASCII text strings (user & host name of participant, e-mail address, notes, ...)
- “Bye” message
  - » used to update participant’s *SSRC* tables

The Real-Time Control Protocol RTCP

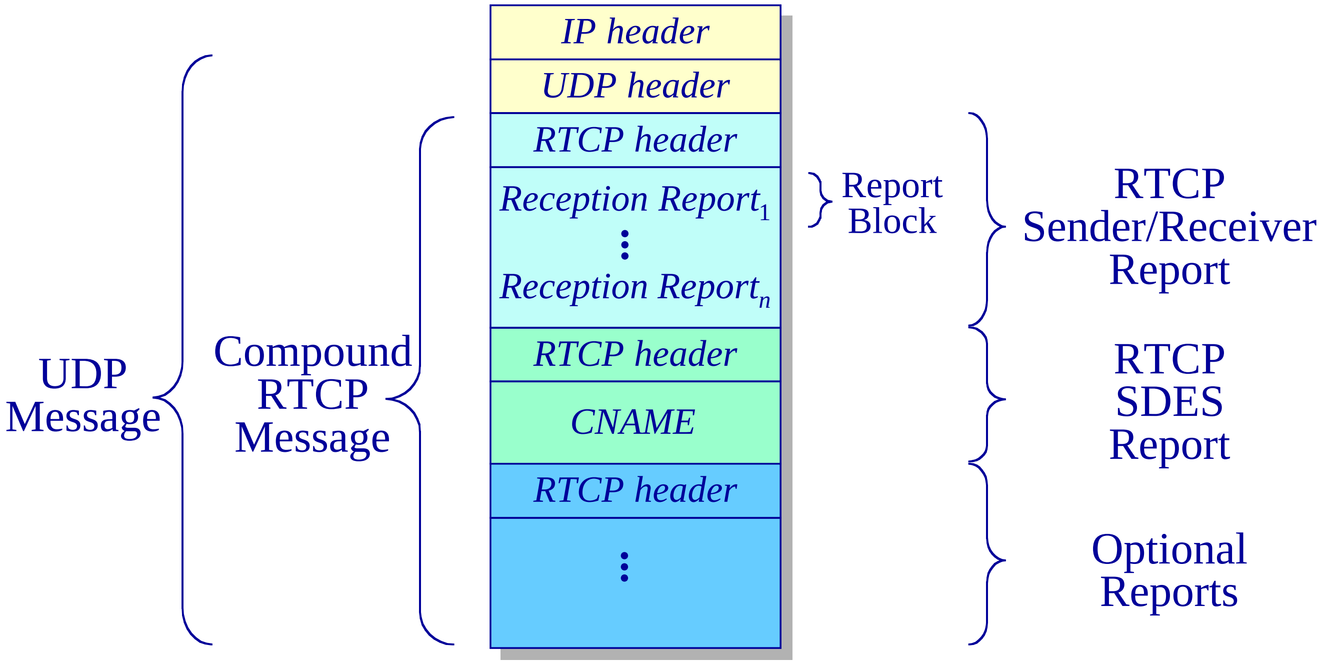
Mechanics

- RTCP messages are "stackable"
  - » To amortize header overhead, multiple RTCP messages can be combined and sent in a *compound RTCP message*
- RTCP messages are always sent in (at least) pairs
  - » Messages must always contain a sender/receiver report and a source description message containing the *canonical name* (CNAME) of the participant
- RTCP messages are sent periodically with a period set to ensure that control messages consume no more than 5% of the session bandwidth
  - » Much of the contents of sender & receiver reports are included so that participants can compute the RTCP sending interval

The Real-Time Control Protocol RTCP

Message encapsulation

RTCP Message



IP header

UDP header

RTCP header

Reception Report$_1$

⋮

Reception Report$_n$

RTCP header

CNAME

RTCP header

⋮

UDP Message

Compound RTCP Message

Report Block

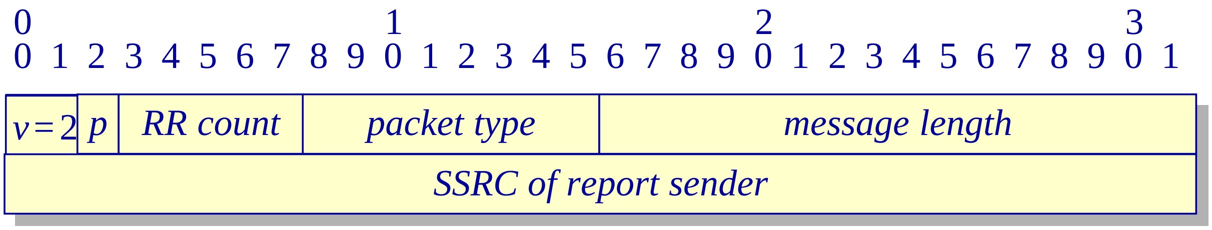
RTCP Sender/Receiver Report

RTCP SDES Report

Optional Reports

The Real-Time Control Protocol RTCP

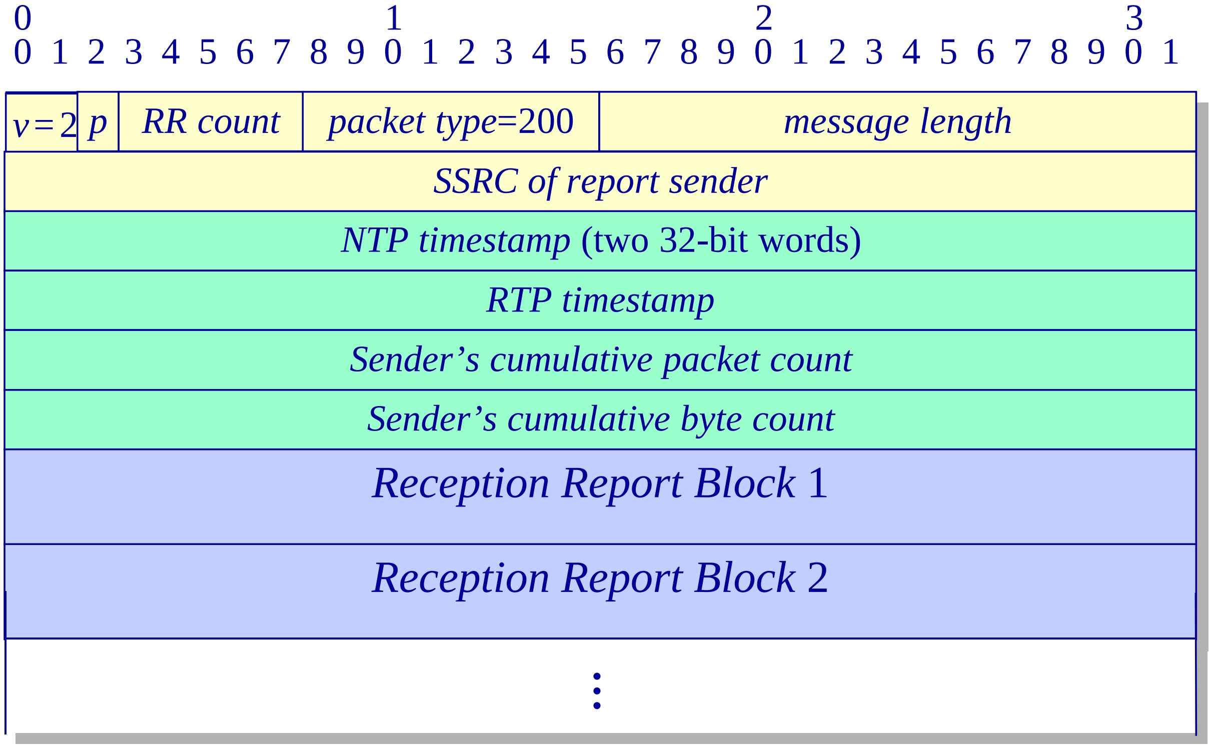
Common sender/receiver report message header



- All report messages have the same 8 byte header
  - » version number (same as RTP)
  - » padding indicator
  - » reception report count (5 bits)
  - » RTCP message type (8 bits)
  - » RTCP message length (16 bits)
  - » *SSRC* for the sender of this report (32 bits)

The Real-Time Control Protocol RTCP

Sender reports — packet format



The Real-Time Control Protocol RTCP

Reception report blocks

- Each sender and receiver report should contain a reception report block for each synchronization source heard from since the last RTCP report
- Contents:
  - » source identifier for the block (*SSRC*)
  - » fraction of RTP packets from this source lost since the last report
  - » cumulative number of lost packets
  - » extended highest sequence number received
  - » estimated average RTP packet interarrival time jitter
  - » last *SR* timestamp received from this source
  - » delay since receiving the last *SR* report from this source

RTCP Reception Report Blocks

Loss calculation

- The number of lost packets is expressed as a fraction

$$\textit{fraction lost} = \frac{\textit{number of packets lost}}{\textit{number of packets expected}}$$

» where:

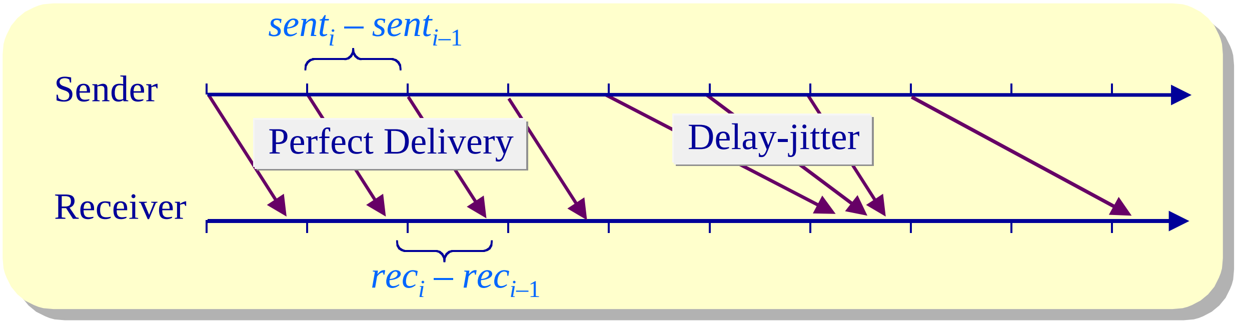
$$\textit{nbr of packets lost} = \textit{nbr packets expected} - \textit{nbr packets received}$$

$$\textit{number of packets expected} = \textit{EHSNR} - \textit{initial sequence number}$$

$$\begin{aligned} \textit{EHSNR} &= \textit{extended highest sequence number received} \\ &= \textit{number of sequence number cycles} \times 2^{16} \\ &\qquad + \textit{last sequence number received} \end{aligned}$$

RTCP Reception Report Blocks

Jitter calculation



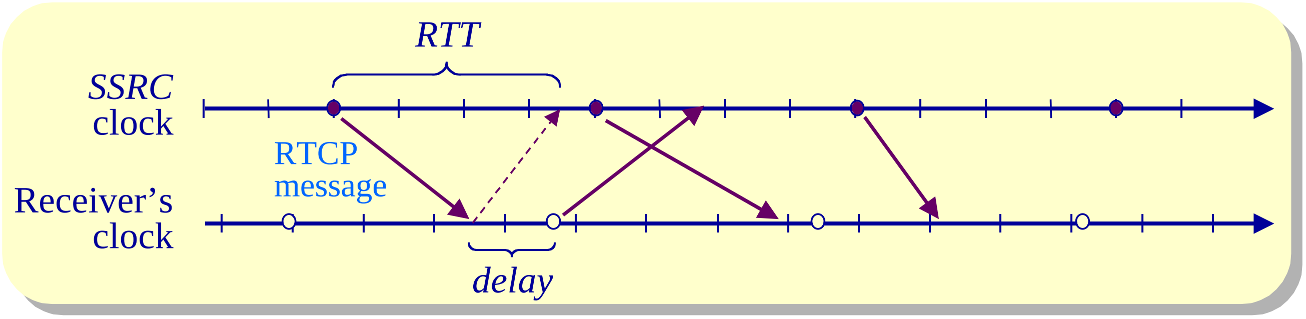
- Interarrival time jitter is an estimate of the statistical variance of RTP data packet interarrival times
  - » the smoothed mean absolute value of the difference between the sending interval at a source and the interarrival time at a receiver

$$jitter_{new} = jitter_{old} + \frac{instantaneous\ jitter - jitter_{old}}{16}$$

$$instantaneous\ jitter = \left|(rec_i - rec_{i-1}) - (sent_i - sent_{i-1})\right|$$

RTCP Reception Report Blocks

Round trip time calculation



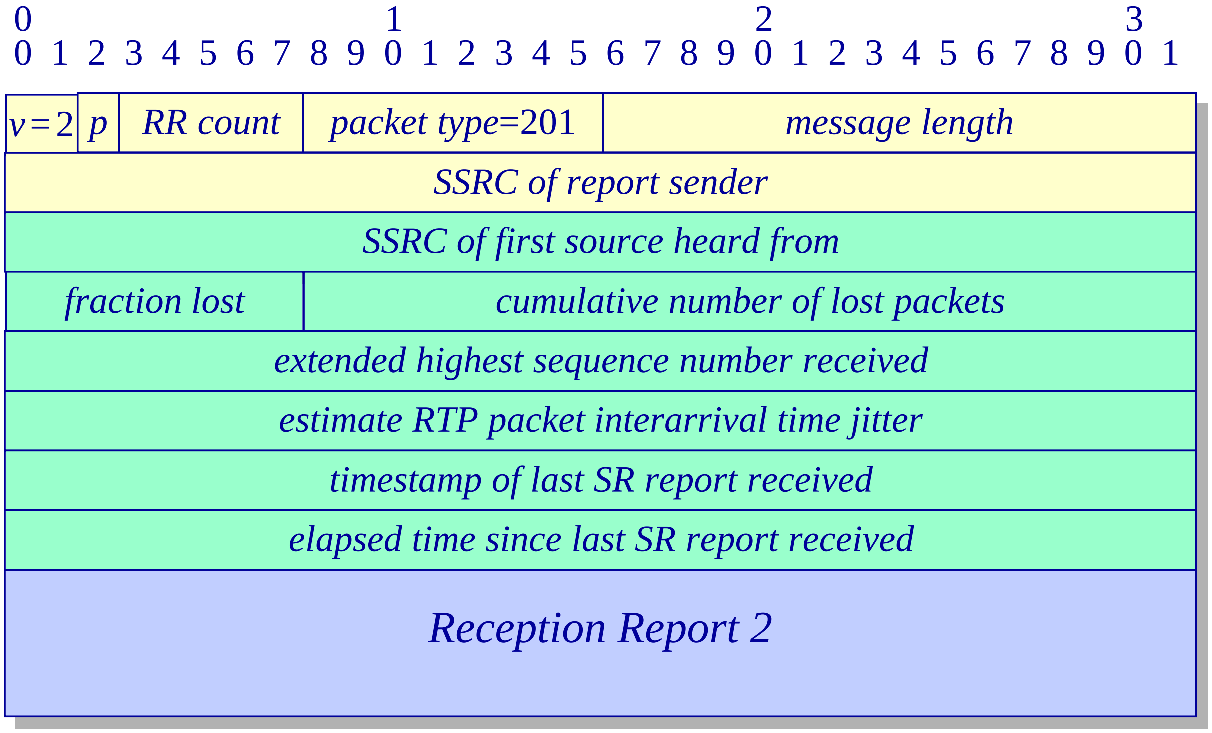
- The *last-SR-timestamp received* and *delay-since-receiving-last-SR-report* fields in the reception report block are used to compute an estimate of the round-trip time from the receiver to a synchronization source

$$estimated\ round\text{-}trip\text{-}\ time = RR\ received - SR\ sent - delay$$

$$\begin{aligned} RR\ received &= time\ a\ source\ received\ this\ reception\ report \\ SR\ sent &= last\ SR\ timestamp\ received\ field \\ delay &= delay\ since\ last\ SR\ report\ field \end{aligned}$$

The Real-Time Control Protocol RTCP

Receiver reports

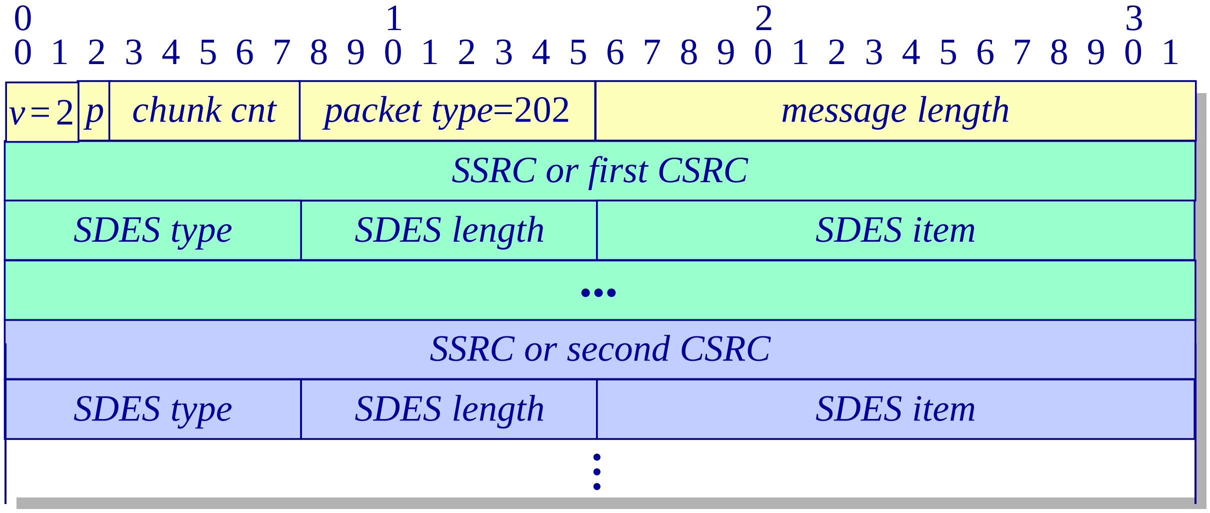


0 1 2 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

v=2	*p*	*RR count*	*packet type*=201	*message length*
SSRC of report sender				
SSRC of first source heard from				
fraction lost	*cumulative number of lost packets*			
extended highest sequence number received				
estimate RTP packet interarrival time jitter				
timestamp of last SR report received				
elapsed time since last SR report received				
Reception Report 2				

The Real-Time Control Protocol RTCP

Source description item (SDES) messages



0 1 2 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

v=2	*p*	*chunk cnt*	*packet type*=202	*message length*
SSRC or first CSRC				
SDES type	*SDES length*	*SDES item*		
•••				
SSRC or second CSRC				
SDES type	*SDES length*	*SDES item*		
⋮				

- An SDES message consists of one or more “chunks” of source description items
  - » CNAME (*user@host*)
  - » NAME
  - » EMAIL
  - » PHONE
  - » LOC
  - » TOOL
  - » NOTE
  - » ...

Source Description Item (SDES) Messages

BYE message

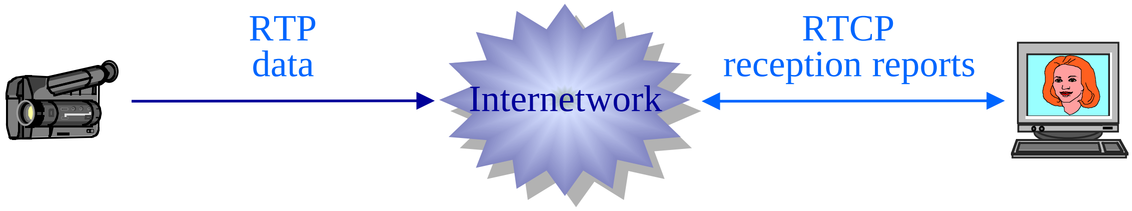
```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
```

<table>
<tr><td><i>v</i>=2</td><td><i>p</i></td><td><i>source cnt</i></td><td><i>packet type</i>=203</td><td><i>message length</i></td></tr>
<tr><td colspan="5"><i>SSRC/CSRC</i></td></tr>
<tr><td colspan="5"><i>SSRC/CSRC</i></td></tr>
<tr><td colspan="5">⋮</td></tr>
<tr><td colspan="3"><i>note length</i></td><td colspan="2"><i>reason for leaving</i></td></tr>
</table>

- The BYE message is used by participants to update their *SSRC* tables
  - » participants can optionally say why they are leaving (a cheap way of reporting errors in real-time)

The Real-Time Control Protocol RTCP

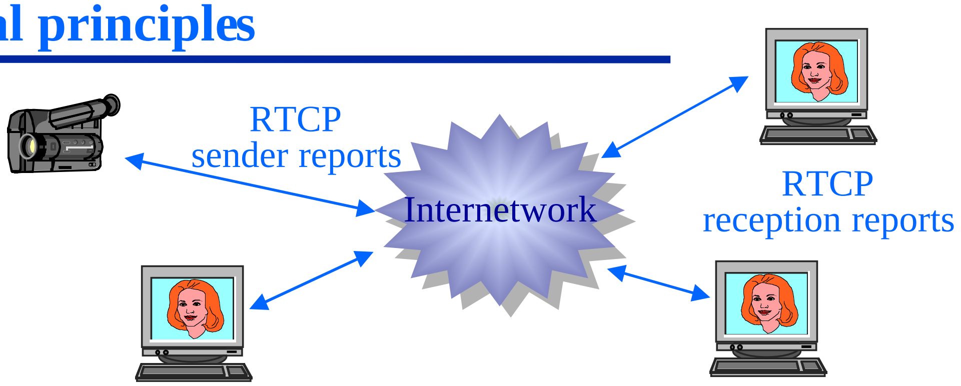
Scalability issues



- RTP data transmission is inherently scalable
- RTCP message transmissions are not!
  - » RTCP message bandwidth must be controlled
- Generic RTCP transmission guidelines:
  - » RTCP messages should consume no more than 5% of session bandwidth
  - » 25% of RTCP bandwidth should be allocated to senders
- The challenge: Each session participant must independently compute an RTCP sending interval that ensures transmission guidelines are met

Computing the RTCP Sending Interval

General principles



RTCP
sender reports

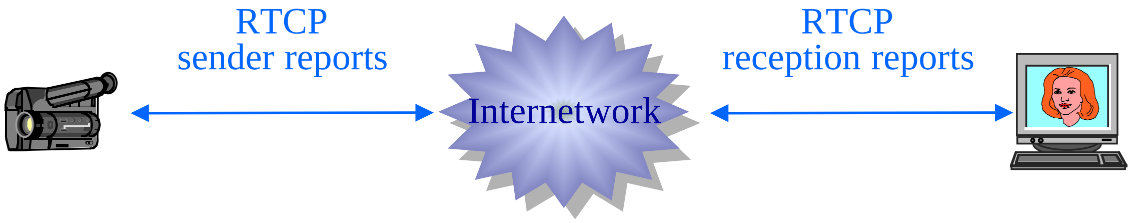
Internetwork

RTCP
reception reports

- Participants keep a running estimate of the session size
- For a given session bandwidth, session size, and RTCP packet size, compute a transmission delay
- Randomize the delay to avoid synchronization effects
  - » Randomize uniformly in the range [0.5, 1.5] x *computed delay*
- As other participants join and leave the group, adjust the delay accordingly

Computing the RTCP Sending Interval

Algorithm



RTCP
sender reports

Internetwork

RTCP
reception reports

- Case 1: *number of senders < 25% of session membership*
  - » Compute the average expected RTCP message interarrival time

$$\textit{average IAT of sender reports} = \frac{\textit{average RTCP packet size}}{0.25 \times \textit{RTCP bandwidth}}$$

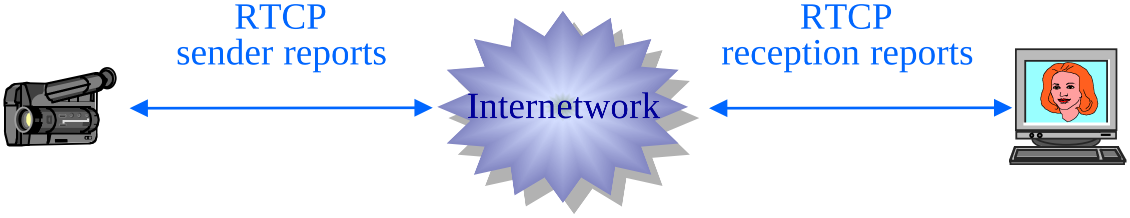
$$\textit{average IAT of receiver reports} = \frac{\textit{average RTCP packet size}}{0.75 \times \textit{RTCP bandwidth}}$$

  - » Average transmission delay is

number of peers x *average SR/RR IAT*

Computing the RTCP Sending Interval

Algorithm



- Case 1: *number of senders < 25% of session membership*
  - » Compute the "deterministic interval" T_d = MAX(T_{min}, *ave delay*)

    where T_{min} = 2.5 *secs* if the session is starting, and

    T_{min} = 5 *secs* otherwise
  - » Then delay for a duration $T = \dfrac{(0.5 + random()) \times T_d}{e - 1.5}$

Computing the RTCP Sending Interval

Algorithm

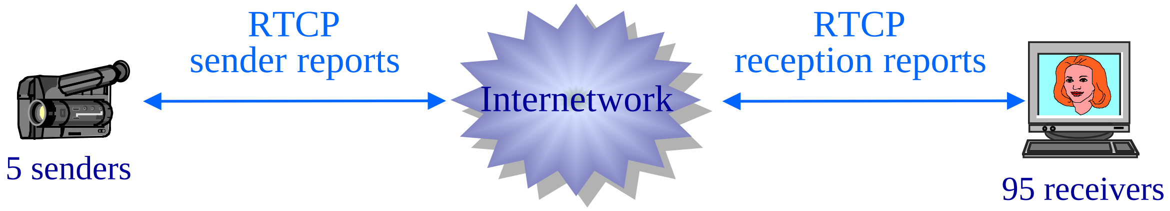
RTCP
sender reports
Internetwork
RTCP
reception reports

- Case 2: *number of senders > 25% of session membership*
  - » Average expected RTCP message interarrival time is simply

    $$\frac{\textit{average RTCP packet size}}{\textit{RTCP bandwidth}}$$
  - » The deterministic interval is T_d = MAX(T_{min}, $n \times$ *ave RTCP IAT*)
    where n is the total number of session participants
  - » T_{min} and the computed delay T are as before

Computing the RTCP Sending Interval

Example



RTCP sender reports

Internetwork

RTCP reception reports

5 senders

95 receivers

- For a 32 kbps DVI4 audio conference with 100 participants
  - » *senders* transmit every

$$\textit{number of senders} \times \frac{\textit{average RTCP packet size}}{0.25 \times \textit{RTCP bandwidth}} = \frac{5 \times 100\ \textit{bytes} \times 8\ \textit{bits/byte}}{0.25 \times 32\ \textit{kbps} \times 0.05}$$

$$= 10\ \textit{secs}$$

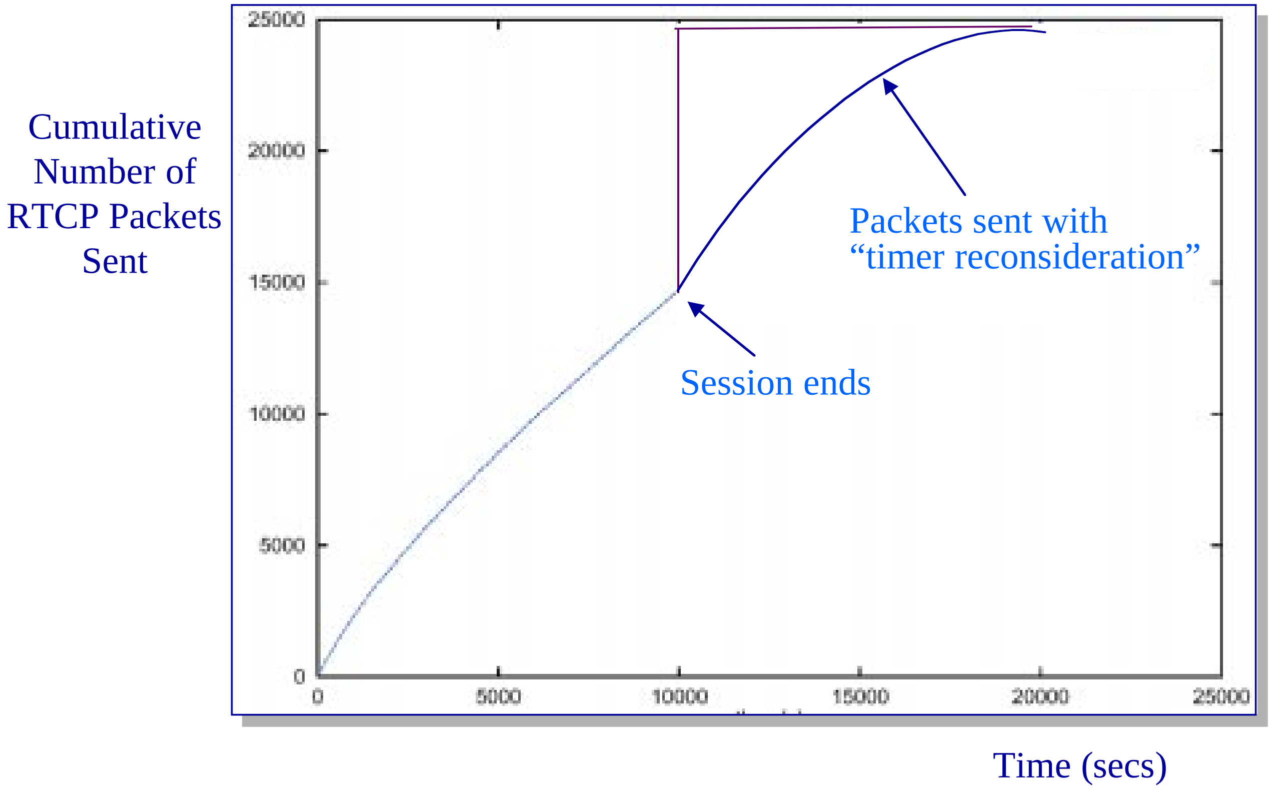
  - » *receivers* transmit every

$$\textit{number of receivers} \times \frac{\textit{average RTCP packet size}}{0.75 \times \textit{RTCP bandwidth}} = \frac{95 \times 100\ \textit{bytes} \times 8\ \textit{bits/byte}}{0.75 \times 32\ \textit{kbps} \times 0.05}$$

$$= 63\ \textit{secs}$$

RTCP Scalability Issues

BYE floods



Packets sent with “timer reconsideration”

Session ends

RTCP Scalability Issues

Timer reconsideration

- We want to slow down the RTCP transmission process when the session size is growing and speed up the process when the sesion size is shrinking
- Timer reconsideration:
  - » While waiting to send an RTCP message, update session size estimate
  - » When transmission timer expires, recompute *T*. If

$$\textit{time of last RTCP transmission} + T \le \textit{current time}$$

then transmit an RTCP message and calculate another delay, else, reschedule transmission for time

$$\textit{time of last RTCP transmission} + T$$

RTCP Scalability Issues

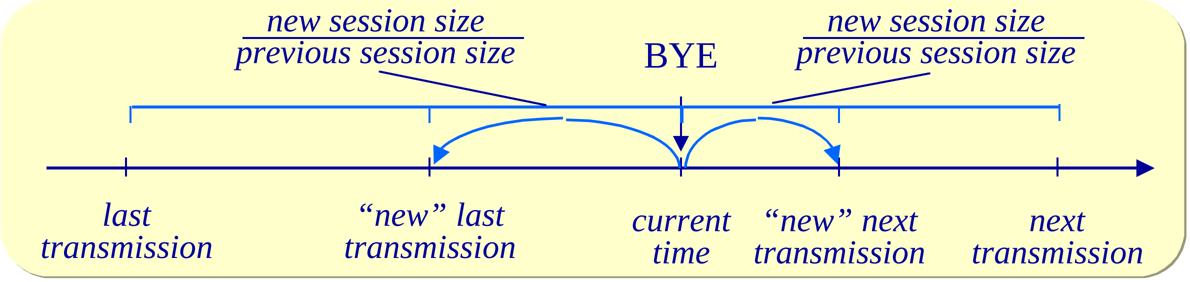
Reverse reconsideration

- On a receipt of a BYE message, if session size has decreased since last RTCP delay was computed, then next message will be sent at time:

$$\textit{current time} + \frac{\textit{new session size}}{\textit{previous session size}} \times \left(\textit{next RTCP transmission time} - \textit{current time} \right)$$

  - » We also update the time of the last (logical) RTCP transmission

$$\textit{current time} - \frac{\textit{new session size}}{\textit{previous session size}} \times \left(\textit{current time} - \textit{time of last RTCP transmission} \right)$$



RTCP Scalability Issues

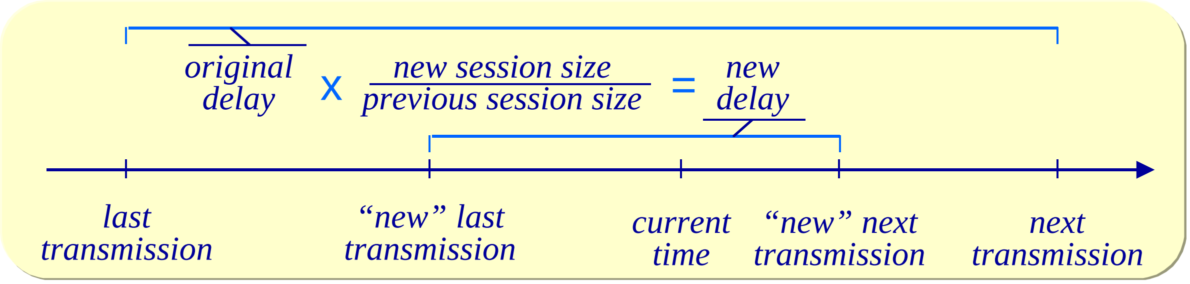
Reverse reconsideration

- On a receipt of a BYE message, if session size has decreased since last RTCP delay was computed, then next message will be sent at time

$$\textit{current time} + \frac{\textit{new session size}}{\textit{previous session size}} \times \left(\textit{next RTCP transmission time} - \textit{current time} \right)$$

  » We also update the time of the last (logical) RTCP transmission

$$\textit{current time} - \frac{\textit{new session size}}{\textit{previous session size}} \times \left(\textit{current time} - \textit{time of last RTCP transmission} \right)$$



$$\textit{original delay} \times \frac{\textit{new session size}}{\textit{previous session size}} = \textit{new delay}$$

last transmission

“new” last transmission

current time

“new” next transmission

next transmission

RTCP Scalability Issues

Sending BYE messages

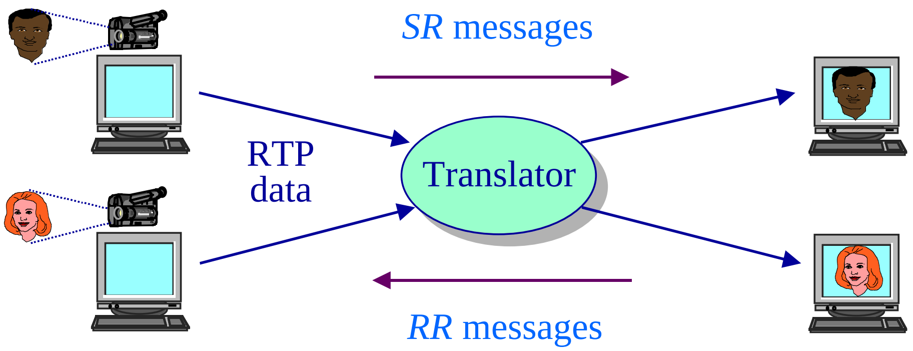
- For BYE message we “splurge” and allow bursts of RTCP messages
  - » Set *session size* to 1, *average packet size* to BYE message size and compute *T* as before
  - » Only increment the session size on receipt of BYE messages

- In the worst case BYE messages consume 5% of session bandwidth
  - » Other RTCP traffic consumes 5% of session bandwidth…
  - » Hence worst case is an additional 5% of session bandwidth consumed

The Real-Time Control Protocol RTCP

RTCP message processing in translators & mixers

- Translators that do not modify RTP data packets typically will not modify RTCP packets
- Translators that modify data packets must modify RTCP packets to so that the reported statistics reflect the performance of the modified stream



The Real-Time Control Protocol RTCP

Sender reports

```
0                                       1                                       2                                       3
0   1   2   3   4   5   6   7   8   9   0   1   2   3   4   5   6   7   8   9   0   1   2   3   4   5   6   7   8   9   0   1
```

v=2	*p*	*RR count*	*packet type*=200	*message length*
SSRC of report sender				
NTP timestamp (two 32-bit words)				
RTP timestamp				
Sender's cumulative packet count				
Sender's cumulative byte count				
Reception Report Block 1				
Reception Report Block 2				
⋮				

The Real-Time Control Protocol RTCP

Receiver reports

0 1 2 3

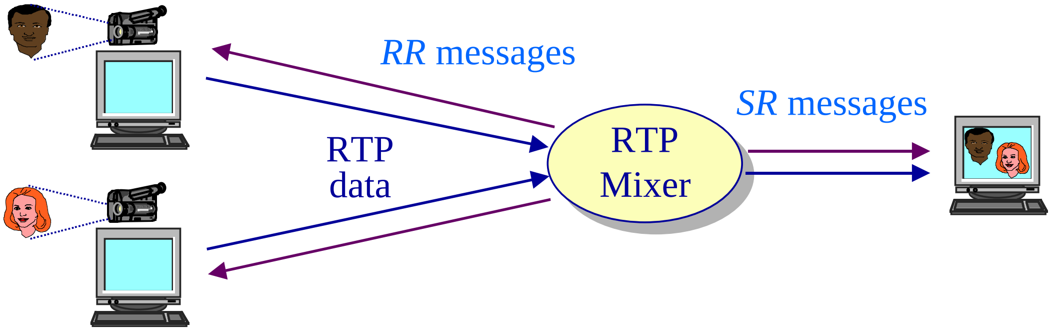
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

<table>
<tr><td><i>v</i>=2</td><td><i>p</i></td><td><i>RR count</i></td><td><i>packet type</i>=201</td><td><i>message length</i></td></tr>
<tr><td colspan="5"><i>SSRC of report sender</i></td></tr>
<tr><td colspan="5"><i>SSRC of first source heard from</i></td></tr>
<tr><td colspan="3"><i>fraction lost</i></td><td colspan="2"><i>cumulative number of lost packets</i></td></tr>
<tr><td colspan="5"><i>extended highest sequence number received</i></td></tr>
<tr><td colspan="5"><i>estimate RTP packet interarrival time jitter</i></td></tr>
<tr><td colspan="5"><i>timestamp of last SR report received</i></td></tr>
<tr><td colspan="5"><i>elapsed time since last SR report received</i></td></tr>
<tr><td colspan="5"><i>Reception Report 2</i></td></tr>
</table>

The Real-Time Control Protocol RTCP

RTCP message processing in translators & mixers

- Since mixers are synchronization sources, they generate their own RCTP packets
  - » Mixers generate *SR* sender information in exactly the same way sources do
  - » Mixers generate reception report blocks for sources in exactly the same way receivers do
- But all messages are only sent to one “cloud”


RR messages
RTP data
RTP Mixer
SR messages