

## **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>

1

## **The Multimedia Transport Protocol RTP**

### **Outline**

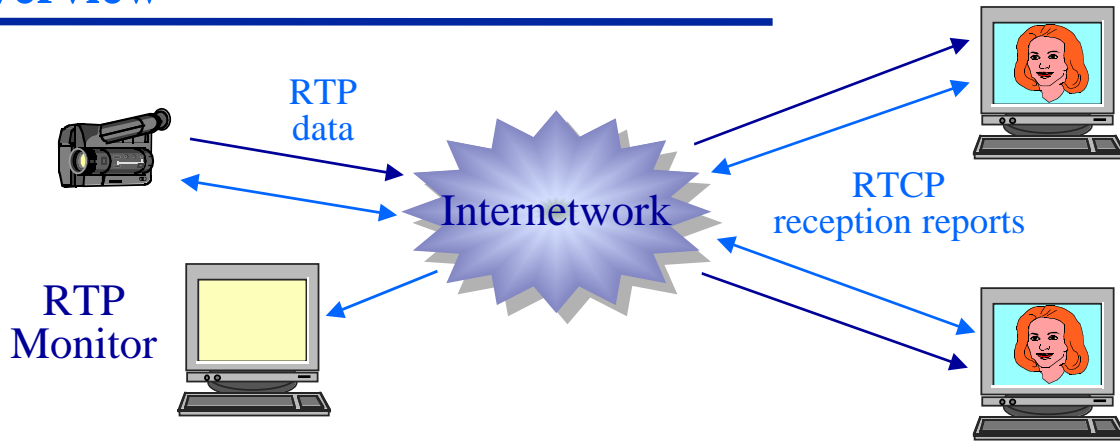
---

- ◆ 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

2

# The Real-Time Control Protocol RTCP

## Overview

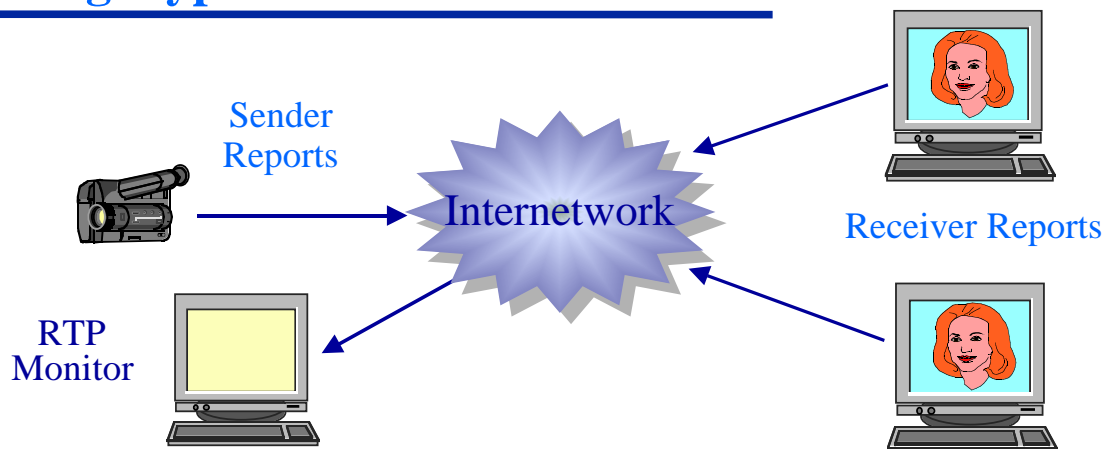


- ◆ Senders & receivers periodically generate reports of various session statistics and multicast to the group
- ◆ RTCP enables...
  - » Diagnosis of faults in the multicast distribution tree
  - » Congestion control
  - » Third party performance monitoring & logging
  - » (Simple) conference control

3

# 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

4

# The Real-Time Control Protocol RTCP

## Mechanics

---

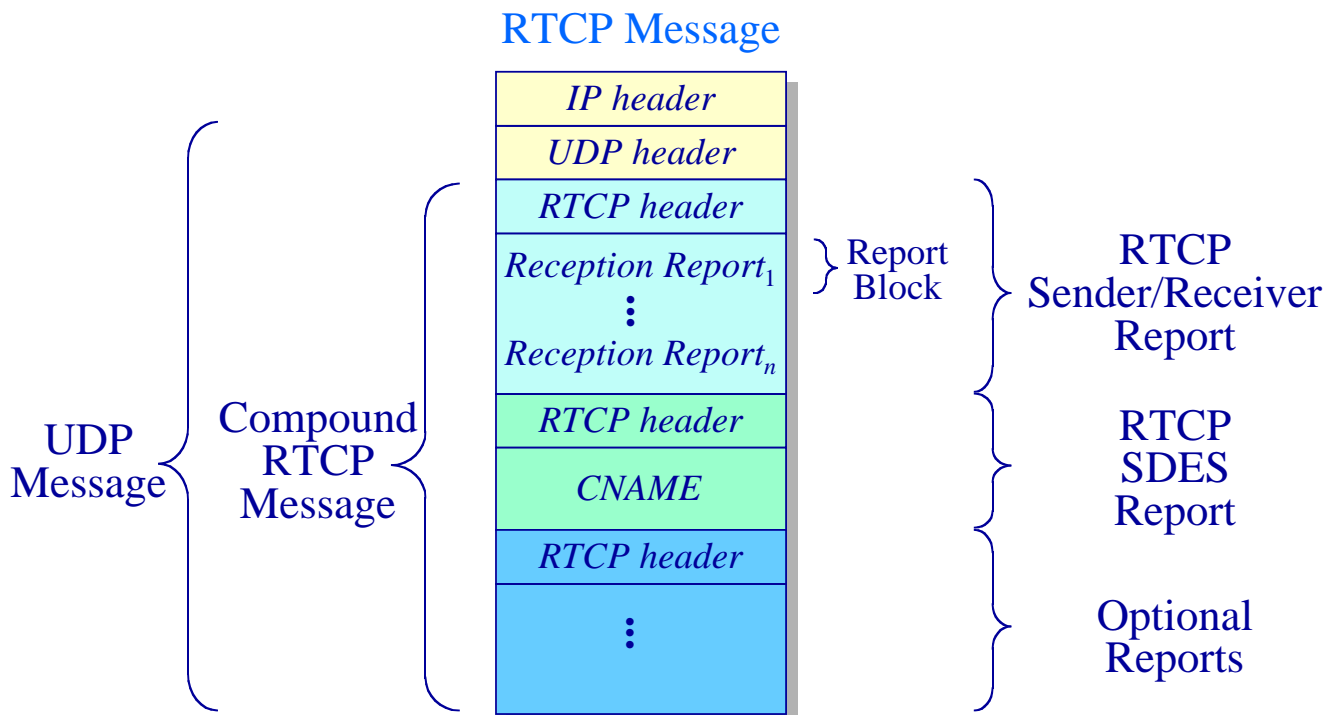
- ◆ RTCP messages are “stackable”
  - » To amortize header overhead, multiple RTCP messages can be combined and sent in a *compound RTCP message*
- ◆ RTCP messages are always sent in (at least) pairs
  - » Messages must always contain a sender/receiver report and a source description message containing 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

5

# The Real-Time Control Protocol RTCP

## Message encapsulation

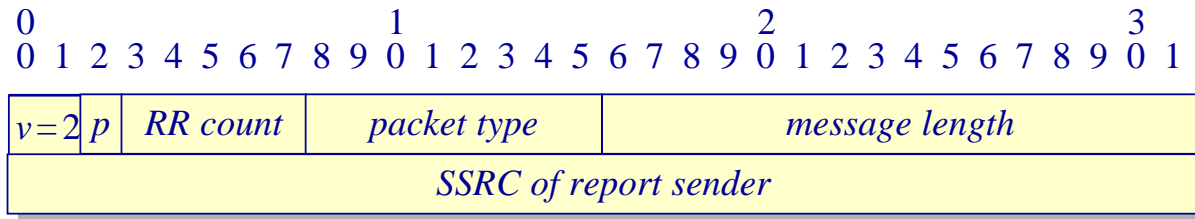
---



6

# The Real-Time Control Protocol RTCP

## Common sender/receiver report message header

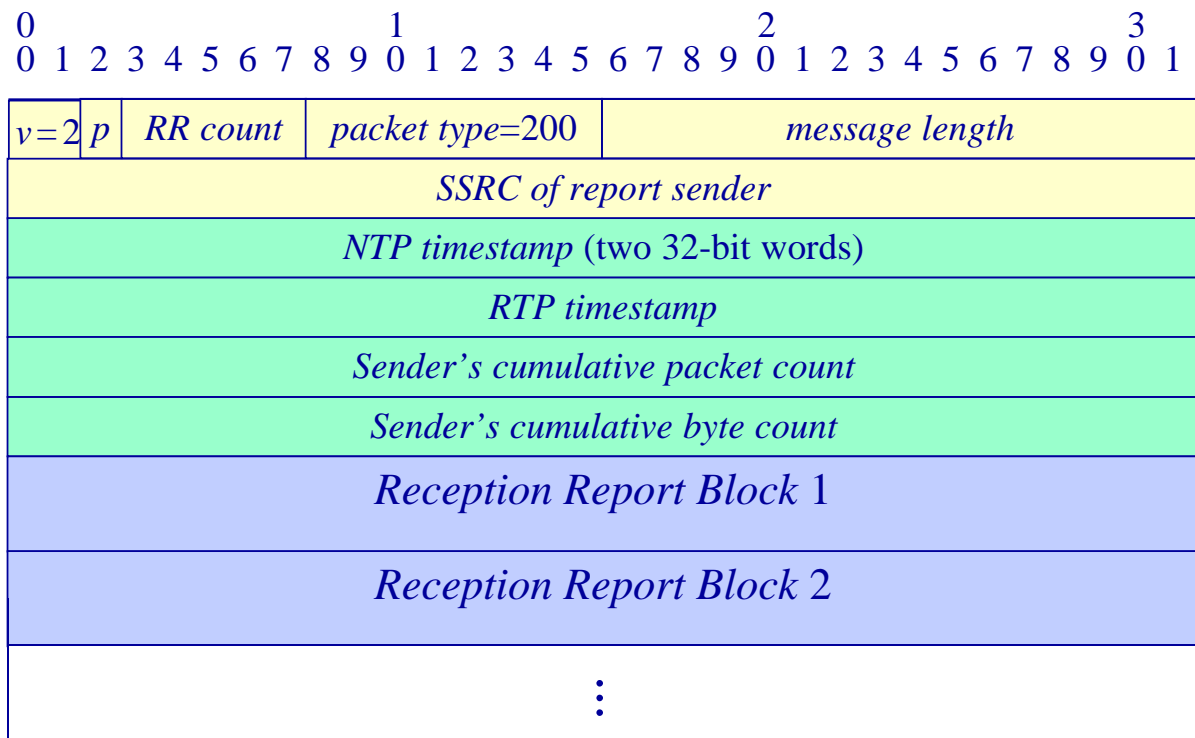


- ◆ All report messages have the same 8 byte header
  - » version number (same as RTP)
  - » padding indicator
  - » reception report count (5 bits)
  - » RTCP message type (8 bits)
  - » RTCP message length (16 bits)
  - » *SSRC* for the sender of this report (32 bits)

7

# The Real-Time Control Protocol RTCP

## Sender reports — packet format



8

# 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

9

## 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:

*nbr of packets lost* = *nbr packets expected* – *nbr packets received*

*number of packets expected* = *EHSNR* – *initial sequence number*

*EHSNR* = *extended highest sequence number received*

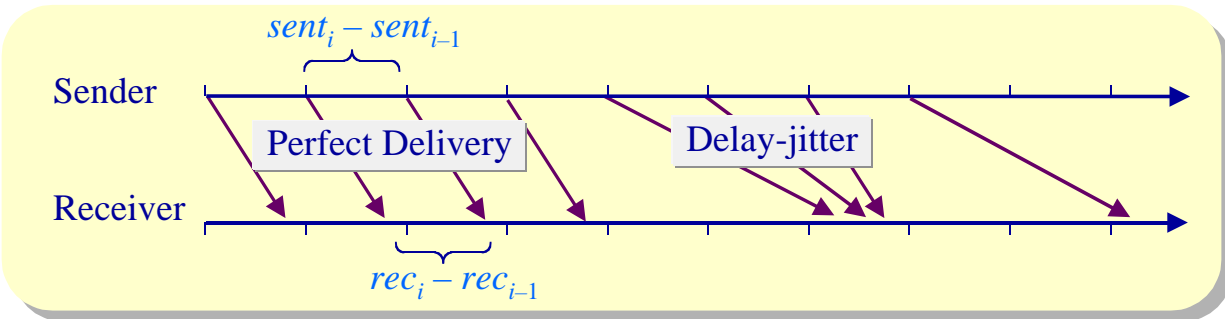
= *number of sequence number cycles* ×  $2^{16}$

+ *last sequence number received*

10

# 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

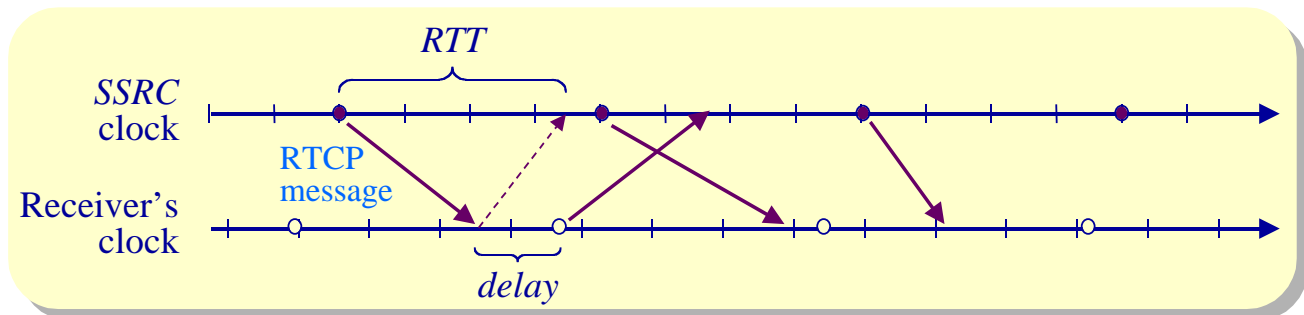
$$jitter_{new} = jitter_{old} + \frac{instantaneous\ jitter - jitter_{old}}{16}$$

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

11

# 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

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

*RR received* = time a source received this reception report

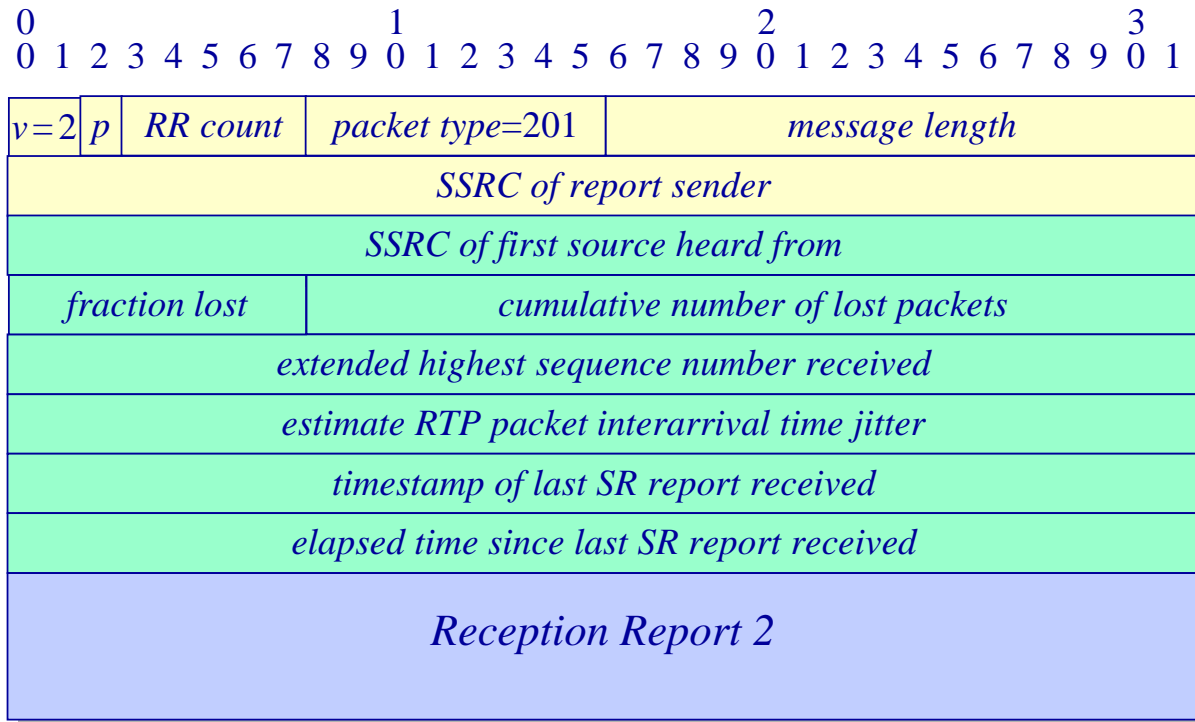
*SR sent* = last SR timestamp received field

*delay* = delay since last SR report field

12

# The Real-Time Control Protocol RTCP

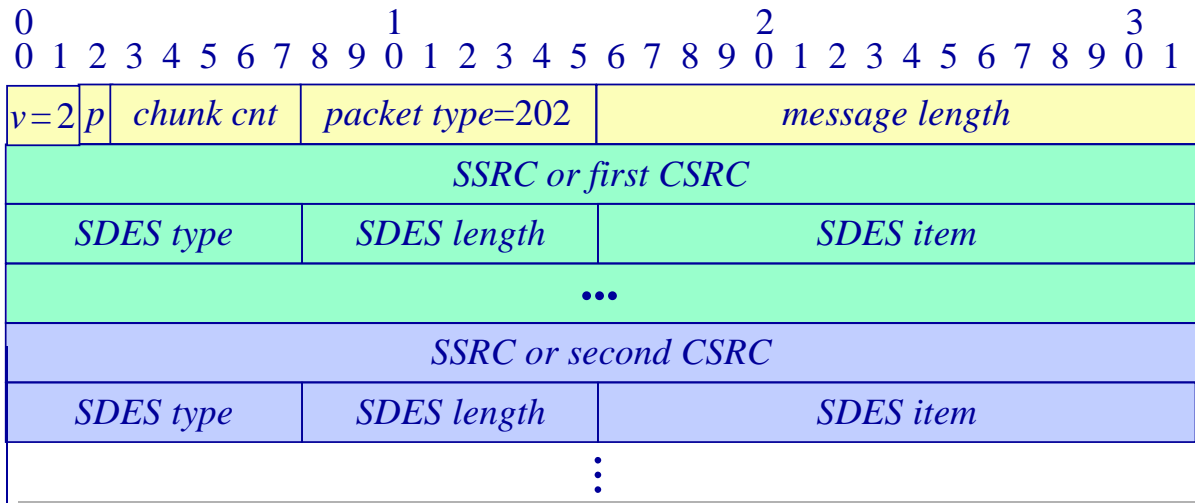
## Receiver reports



13

# The Real-Time Control Protocol RTCP

## Source description item (SDES) messages

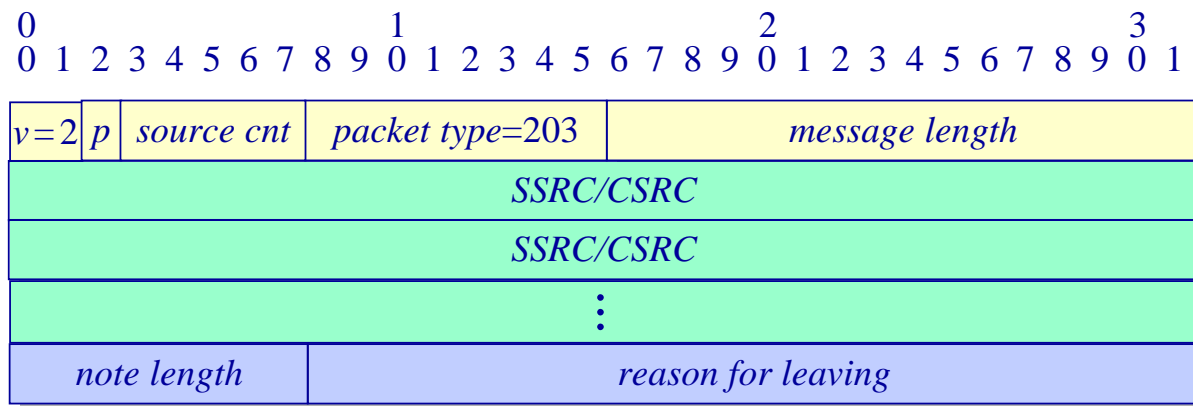


- ◆ An SDES message consists of one or more “chunks” of source description items
  - » CNAME (*user@host*)
  - » NAME
  - » EMAIL
  - » PHONE
  - » LOC
  - » TOOL
  - » NOTE
  - » ...

14

# Source Description Item (SDES) Messages

## BYE message



- ◆ 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)

15

# The Real-Time Control Protocol RTCP

## Scalability issues

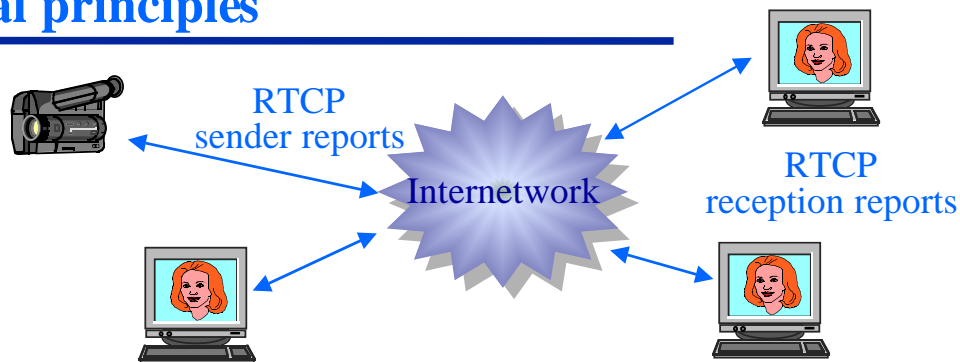


- ◆ RTP data transmission is inherently scalable
- ◆ RTCP message transmissions are not!
  - » RTCP message bandwidth must be controlled
- ◆ Generic RTCP transmission guidelines:
  - » RTCP messages should consume no more than 5% of session bandwidth
  - » 25% of RTCP bandwidth should be allocated to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

16

# Computing the RTCP Sending Interval

## General principles



- ◆ 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  $[0.5, 1.5] \times \text{computed delay}$
- ◆ As other participants join and leave the group, adjust the delay accordingly

17

# Computing the RTCP Sending Interval

## Algorithm



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

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

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

- » Average transmission delay is

$$\text{number of peers} \times \text{average SR/RR IAT}$$

18

# Computing the RTCP Sending Interval Algorithm

---



- ◆ Case 1: *number of senders < 25% of session membership*
  - » Compute the “deterministic interval”  $T_d = \text{MAX}(T_{min}, \text{ave delay})$  where  $T_{min} = 2.5 \text{ secs}$  if the session is starting, and  $T_{min} = 5 \text{ secs}$  otherwise
  - » Then delay for a duration  $T = \frac{(0.5 + \text{random}()) \times T_d}{e - 1.5}$

19

# Computing the RTCP Sending Interval Algorithm

---



- ◆ Case 2: *number of senders > 25% of session membership*
  - » Average expected RTCP message interarrival time is simply 
$$\frac{\text{average RTCP packet size}}{\text{RTCP bandwidth}}$$
  - » The deterministic interval is  $T_d = \text{MAX}(T_{min}, n \times \text{ave RTCP IAT})$  where  $n$  is the total number of session participants
  - »  $T_{min}$  and the computed delay  $T$  are as before

20

# Computing the RTCP Sending Interval

## Example



- ◆ For a 32 kbps DV14 audio conference with 100 participants

» *senders* transmit every

$$\begin{aligned} \text{number of senders} \times \frac{\text{average RTCP packet size}}{0.25 \times \text{RTCP bandwidth}} &= \frac{5 \times 100 \text{ bytes} \times 8 \text{ bits/byte}}{0.25 \times 32 \text{ kbps} \times 0.05} \\ &= 10 \text{ secs} \end{aligned}$$

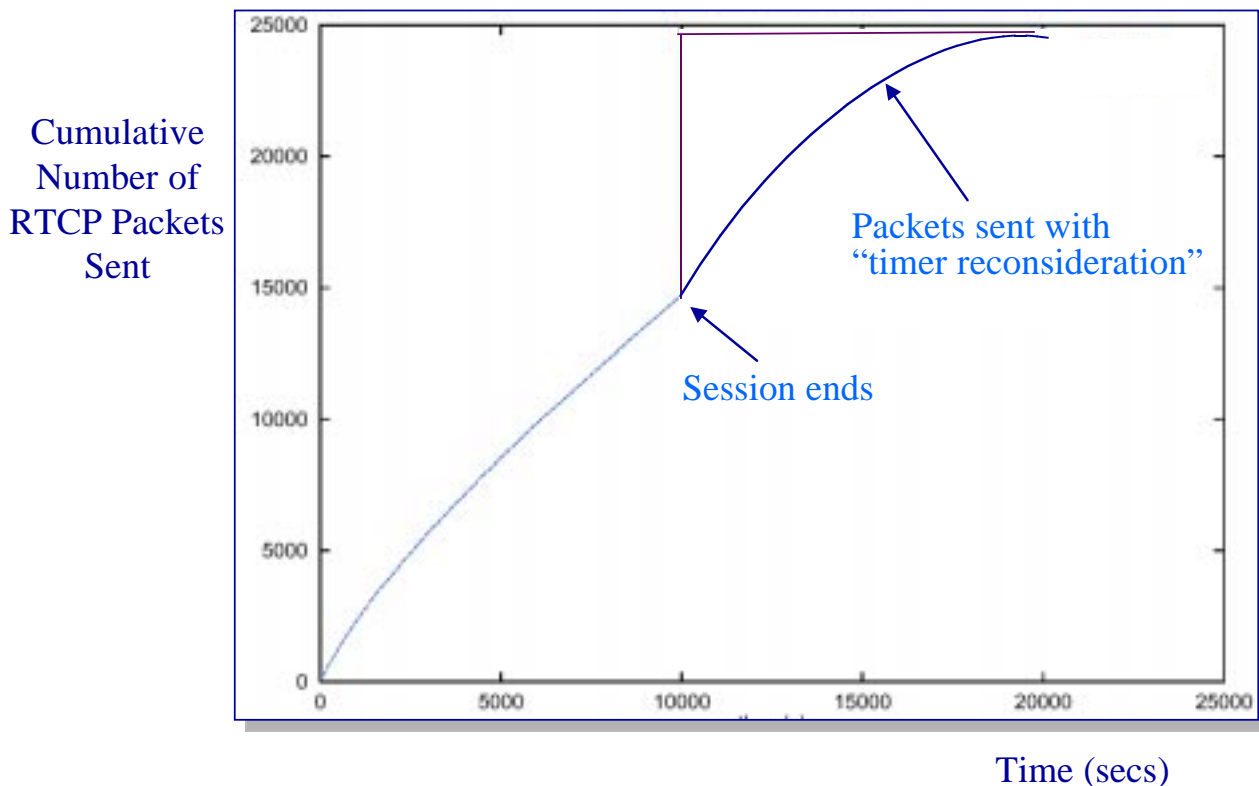
» *receivers* transmit every

$$\begin{aligned} \text{number of receivers} \times \frac{\text{average RTCP packet size}}{0.75 \times \text{RTCP bandwidth}} &= \frac{95 \times 100 \text{ bytes} \times 8 \text{ bits/byte}}{0.75 \times 32 \text{ kbps} \times 0.05} \\ &= 63 \text{ secs} \end{aligned}$$

21

## RTCP Scalability Issues

### BYE floods



22

# 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 session size is shrinking
- ◆ Timer reconsideration:
  - » While waiting to send an RTCP message, update session size estimate
  - » When transmission timer expires, recompute  $T$ . If

$$\text{time of last RTCP transmission} + T \leq \text{current time}$$

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

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

23

# RTCP Scalability Issues

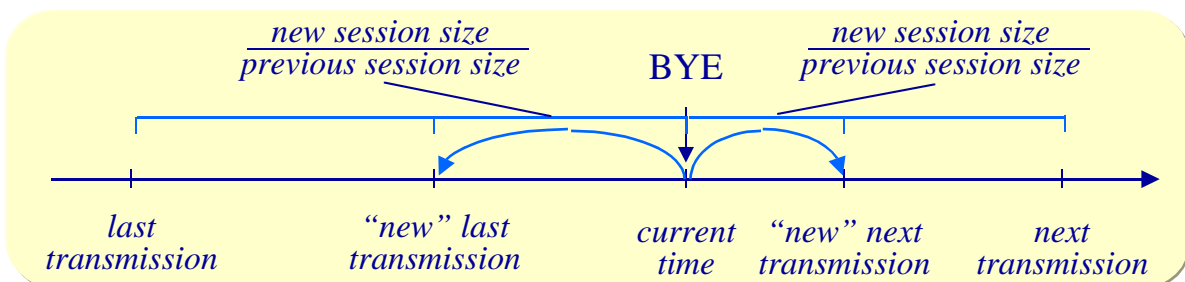
## Reverse reconsideration

- ◆ 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:

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

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

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



24

# RTCP Scalability Issues

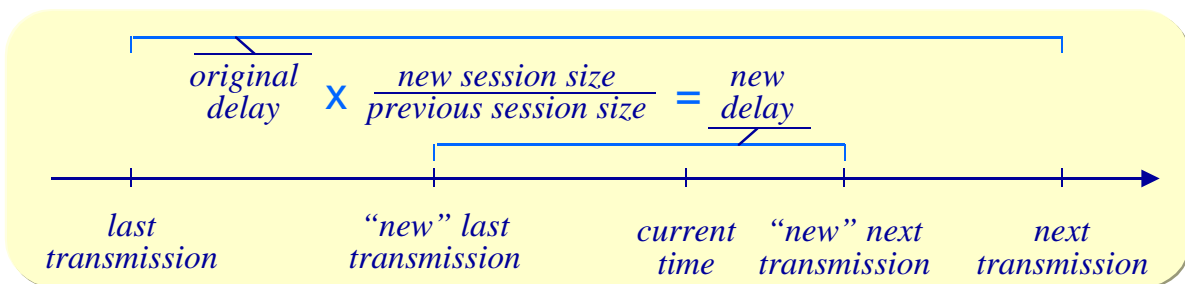
## Reverse reconsideration

- ◆ 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

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

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

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



25

# RTCP Scalability Issues

## Sending BYE messages

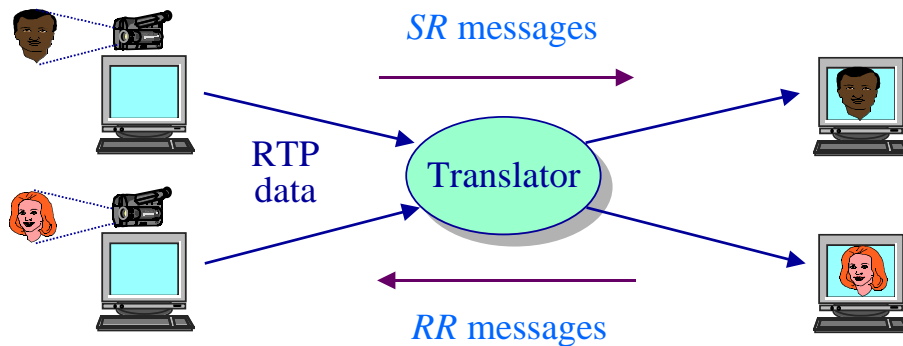
- ◆ For BYE message we “splurge” and allow bursts of RTCP messages
  - » Set *session size* to 1, *average packet size* to BYE message size and compute *T* as before
  - » Only increment the session size on receipt of BYE messages
- ◆ In the worst case BYE messages consume 5% of session bandwidth
  - » Other RTCP traffic consumes 5% of session bandwidth...
  - » Hence worst case is an additional 5% of session bandwidth consumed

26

# 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 so that the reported statistics reflect the performance of the modified stream



27

# The Real-Time Control Protocol RTCP

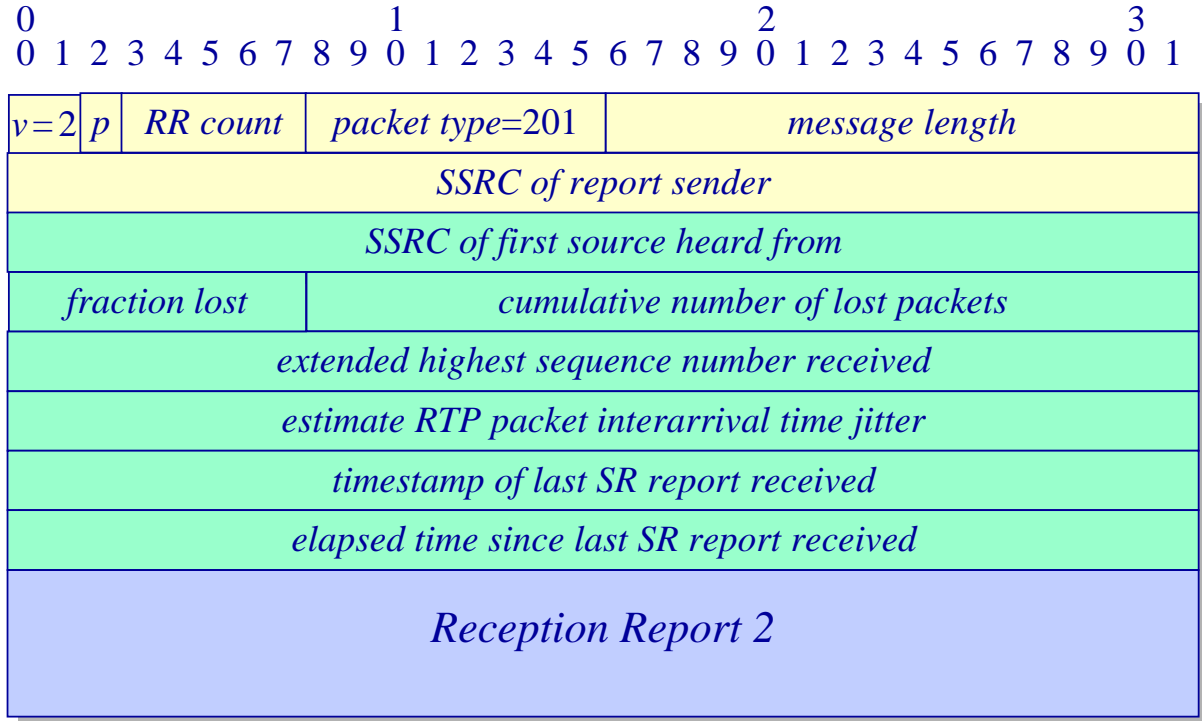
## Sender reports

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

28

# The Real-Time Control Protocol RTCP

## Receiver reports

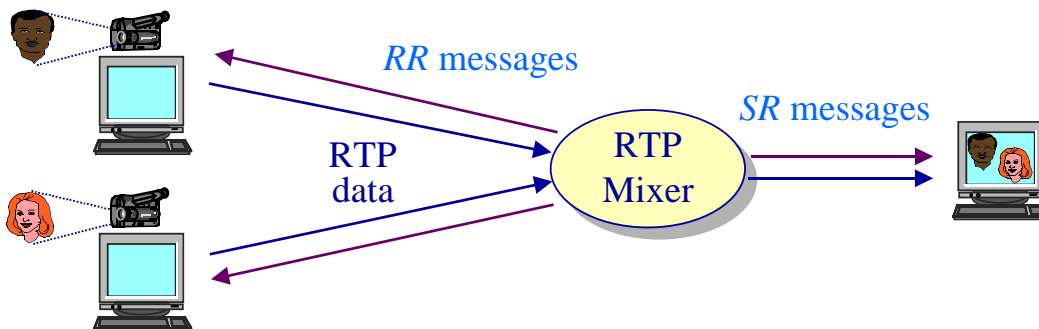


29

# 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”



30