Multimedia Networking

#5 Real-Time Transport Protocol

Semester Ganjil 2012

PTIIK Universitas Brawijaya

Schedule of Class Meeting

1. Introduction
2. Applications of MN
3. Requirements of MN
4. Coding and Compression
5. **RTP**
6. IP Multicast
7. IP Multicast (cont'd)
8. Overlay Multicast
9. CDN: Solutions
10. CDN: Case Studies
11. QoS on the Internet: Constraints
12. QoS on the Internet: Solutions
13. Discussion
14. Summary

Today's Outline

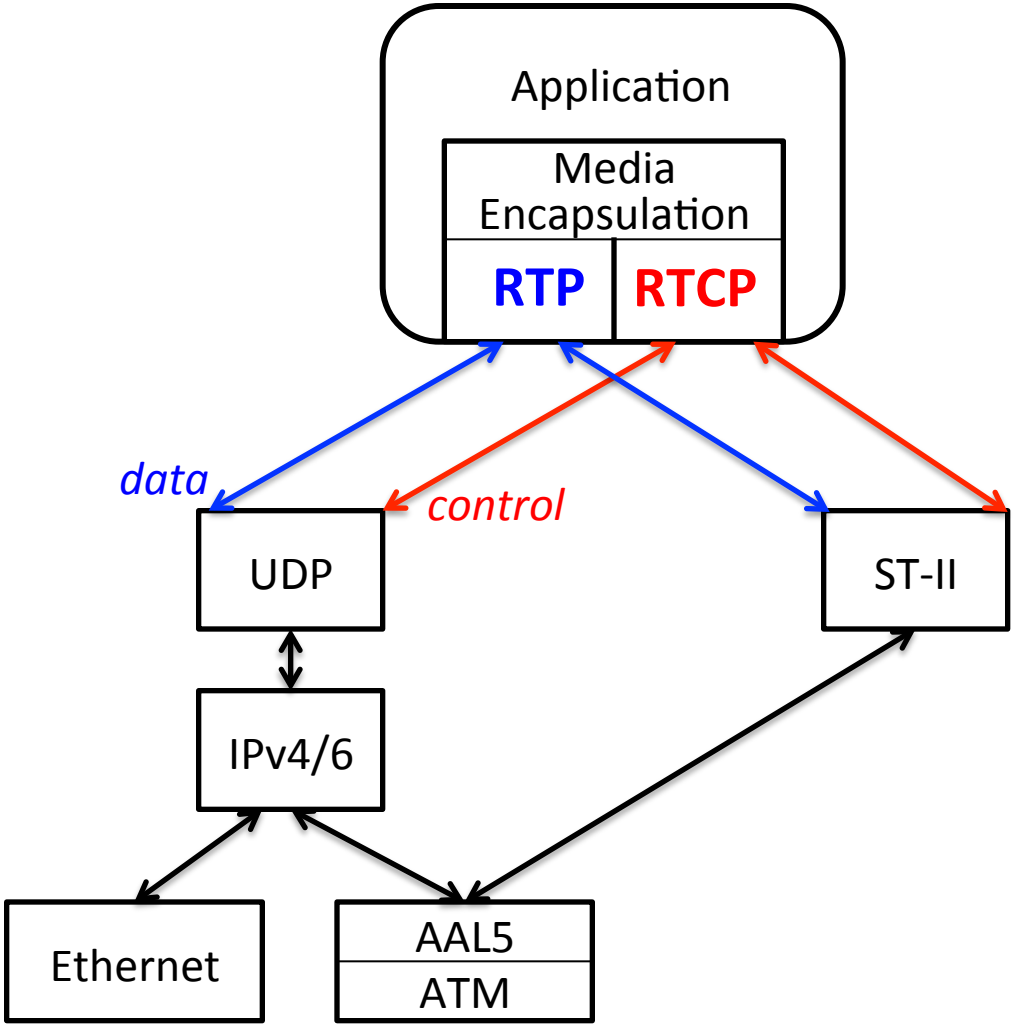
- RTP
  - protocol goals
  - mixers and translators
  - control: awareness, QOS feedback
  - media adaptation

Real-Time Protocol (RTP)

- End-to-end protocol for applications transmitting real-time data, such as audio and video
- RFC 3550
- RTP packets encapsulated in UDP segments

- RTP encapsulation only seen at end systems (*not* by intermediate routers)
  - routers provide best-effort service, no special effort to ensure RTP packets arrive at destination in timely manner

RTP: Big Picture



RTP: Big Picture

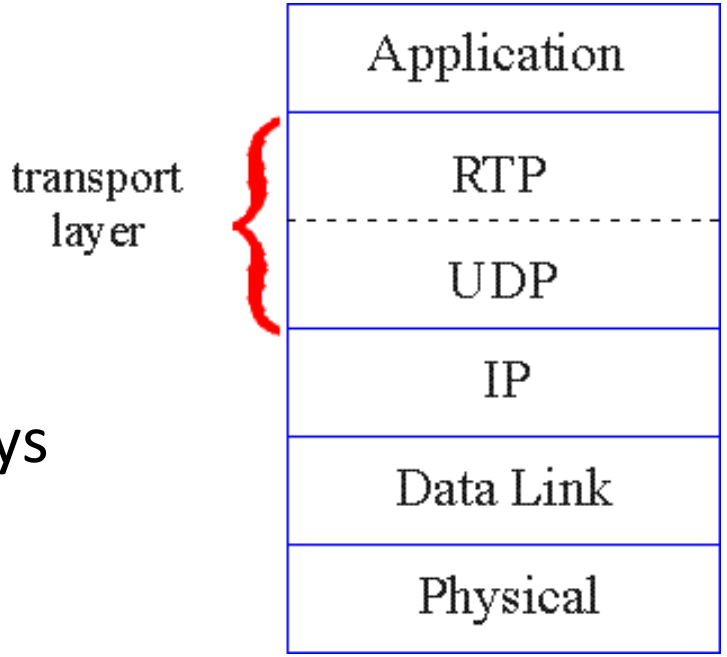
- Real-Time Transport Protocol (RTP) = data + control
- **data:**
  - timing,
  - loss detection,
  - content labeling,
  - talk spurts,
  - encryption
- **control:** (RTCP) → periodic with T ~population
  - QOS feedback
  - membership estimation
  - loop detection

RTP: Goals

- **lightweight:** specification and implementation
- **flexible:** provide mechanism, don't dictate algorithms
- **protocol-neutral:** UDP/IP,ST-II,IPX,ATM-AALx,...
- **scalable:** unicast, multicast from 2 to O(107)
- **separate control/data:** some functions may be taken over by conference control protocol
- **secure:** support for encryption, possibly authentication

RTP runs on top of UDP

- Application layer protocol
- RTP libraries provide transport-layer interface that extends UDP:
  - port numbers, IP addresses
  - payload type identification
  - packet sequence numbering
  - time-stamping



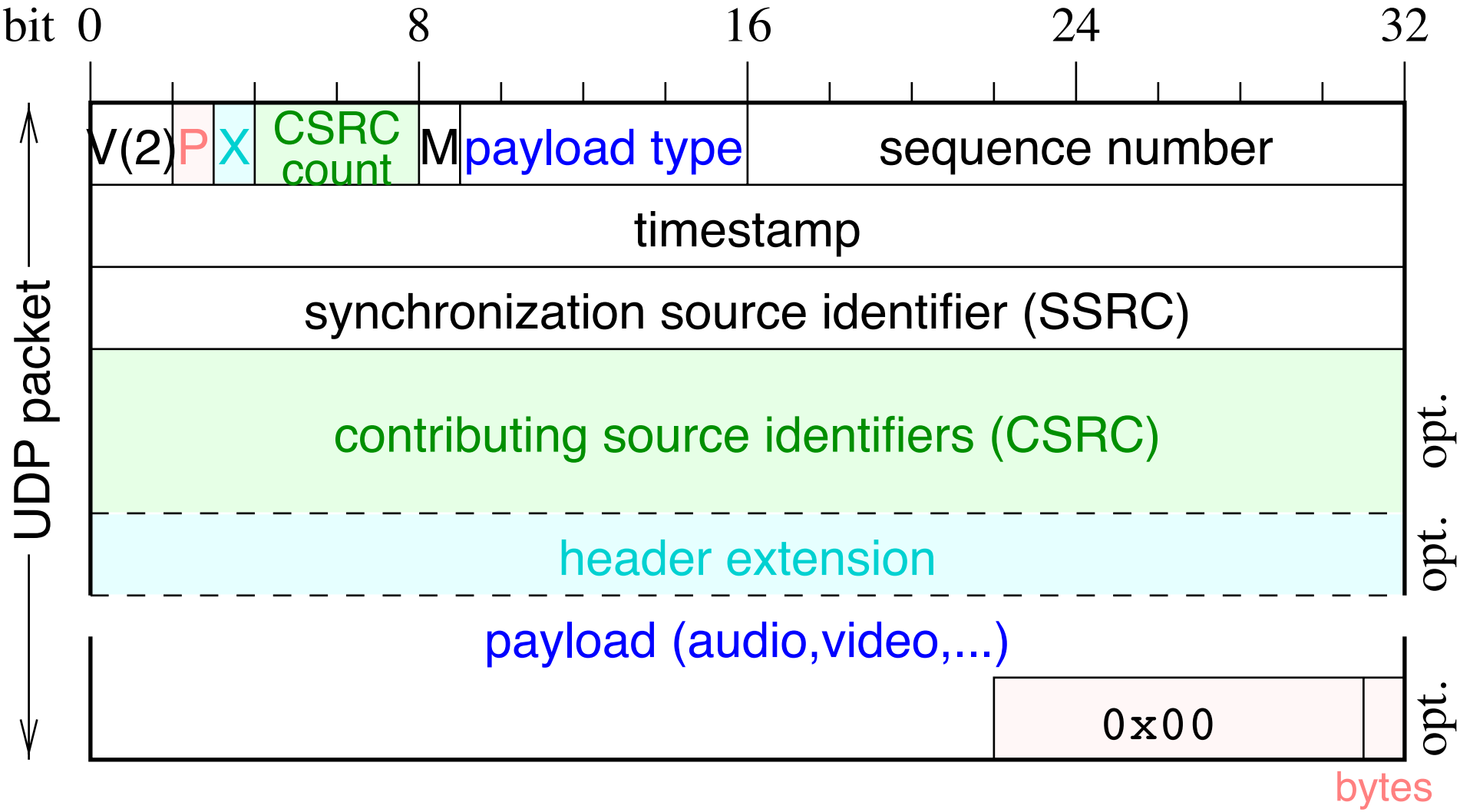
- Applications that use RTP are:
  - Less sensitive to packet loss
  - Very sensitive to packet delays

- UDP provides key services:
  - Multiplexing
  - Checksum

RTP Functions

- segmentation/reassembly done by UDP (or similar)
- re-sequencing (if needed)
- loss detection for quality estimation, recovery
- intra-media synchronization: remove delay jitter through playout buffer
- intra-media synchronization: drifting sampling clocks
- inter-media synchronization (lip sync between audio and video)
- quality-of-service feedback and rate adaptation
- source identification

RTP header


bit 0
8
16
24
32
V(2)
P
X
CSRC count
M
payload type
sequence number
timestamp
synchronization source identifier (SSRC)
contributing source identifiers (CSRC)
header extension
payload (audio,video,...)
0x00
opt.
opt.
opt.
UDP packet
bytes

RTP header

payload type (7 bits): audio/video encoding method; may change during session. If sender changes encoding during call, sender informs receiver via payload type field

- Payload type 0: PCM mu-law, 64 kbps
- Payload type 3: GSM, 13 kbps
- Payload type 7: LPC, 2.4 kbps
- Payload type 26: Motion JPEG
- Payload type 31: H.261
- Payload type 33: MPEG2 video

sequence # (16 bits): +1 for each RTP packet sent

- detect packet loss, restore packet sequence

RTP header

- *timestamp field (32 bits long):* sampling instant of first byte in this RTP data packet
  - for audio, timestamp clock increments by one for each sampling period (e.g., each 125 usecs for 8 KHz sampling clock)
  - if application generates chunks of 160 encoded samples, timestamp increases by 160 for each RTP packet when source is active. Timestamp clock continues to increase at constant rate when source is inactive.
- *SSRC field (32 bits long):* synchronization source
  - sources pick at random ➠ may change after *collision!*

RTP header

- *P:* padding (for encryption) ➠ last byte has padding count
- *M:* marker bit; frame, start of talk spurt ➠ delay adjustment
- *CC:* content source count (for mixers)
- *CSRC:* identifiers of those contributing to (mixed into) packet

RTP example

example: sending 64 kbps PCM-encoded voice over RTP

- application collects encoded data in chunks, e.g., every 20 msec = 160 bytes in a chunk
- audio chunk + RTP header form RTP packet, which is encapsulated in UDP segment

- RTP header indicates type of audio encoding in each packet
  - sender can change encoding during conference
- RTP header also contains sequence numbers, timestamps

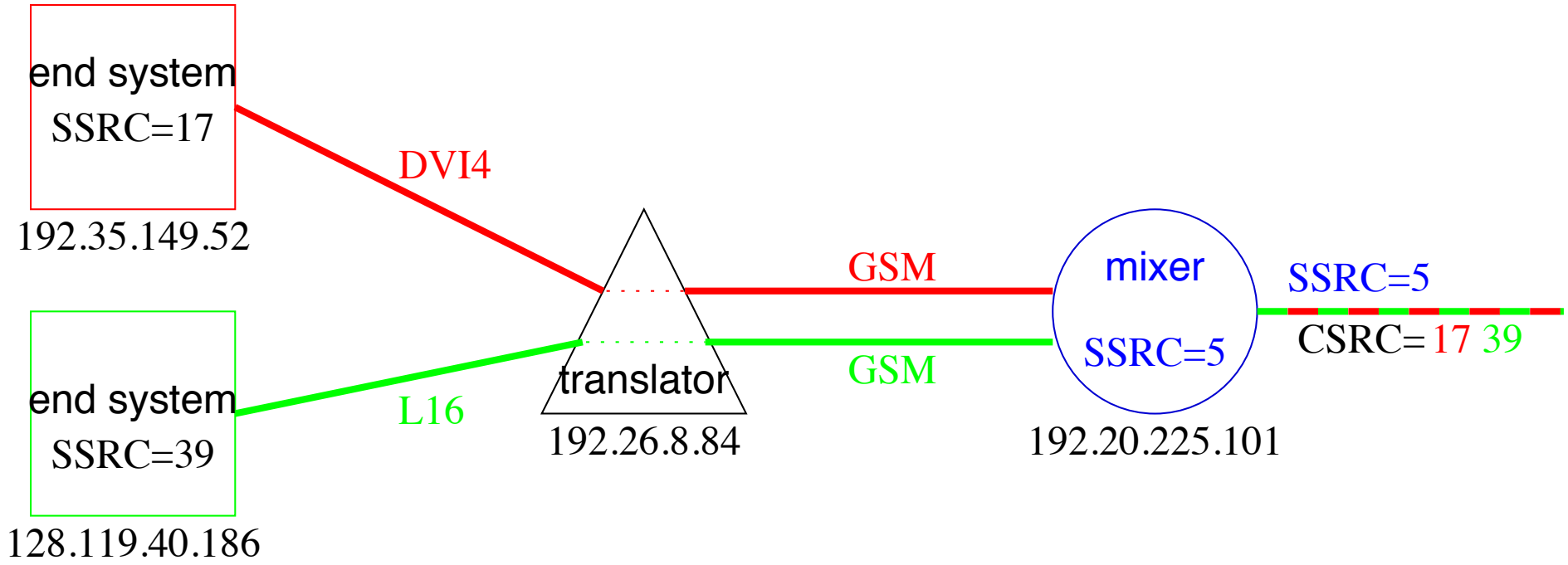
Today's Outline

- RTP
  - protocol goals
  - mixers and translators
  - control: awareness, QOS feedback
  - media adaptation

RTP: Mixers and Translators

- *mixer:*
  - several media stream ➠ one new stream (new encoding)
  - reduced bandwidth networks (dial-up)
  - appears as new source, with own identifier
- *translator:*
  - single media stream
  - *may* convert encoding
  - protocol translation (native ATM ↔ IP), firewall
  - all packets: source address = translator address
- Goals: Accommodate participant network resources

RTP: Mixers and Translators



Today's Outline

- RTP
  - protocol goals
  - mixers and translators
  - control: awareness, QOS feedback
  - media adaptation

Control: awareness, QOS feedback

- Packet loss, congestion, jitter, delivery times
  - Directly useful for control of adaptive encodings
  - Identify if problems are local or global
  - Short-term and long-term statistical analysis

- Self-adjusting network
  - Each participant eventually knows about the other members
  - Source description dynamically identifies who is sending
  - Active senders get more bandwidth
  - Session bandwidth kept constant by adjusting transmission rate based on the number of participants

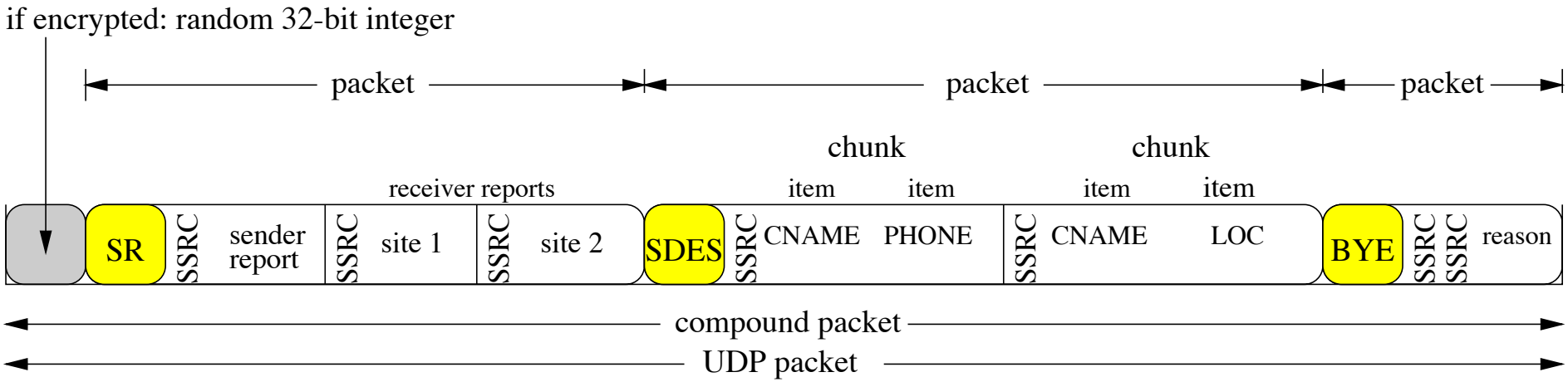
Real-Time Control Protocol (RTCP)

- works in conjunction with RTP
- each participant in RTP session periodically sends RTCP control packets to all other participants
- each RTCP packet contains sender and/or receiver reports
  - report statistics useful to application: # packets sent, # packets lost, interarrival jitter
- feedback used to control performance
  - sender may modify its transmissions based on feedback

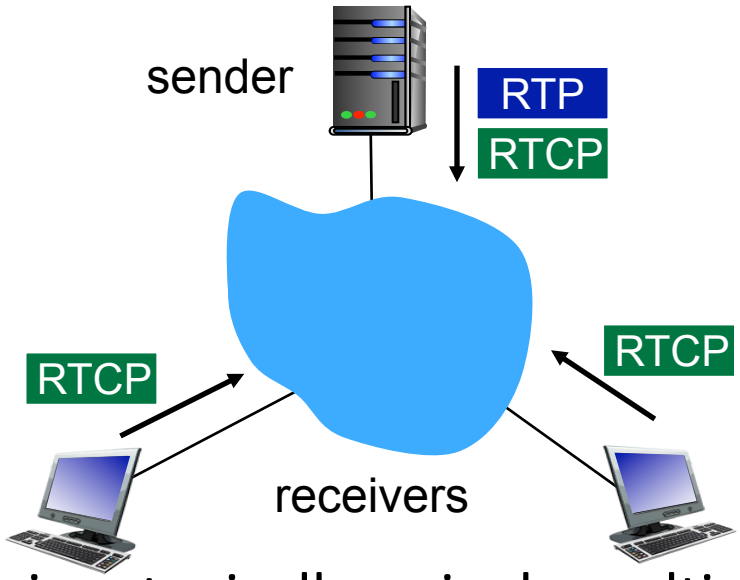
RTCP: packet types

- stackable packets, similar to data packets
- ***sender report (SR):***
  - bytes send ➠ estimate rate;
  - timestamp ➠ synchronization
- ***reception reports (RR):***
  - number of packets sent and expected ➠ loss, avg. inter-arrival jitter, round-trip delay
- ***source description(SDES):***
  - name, email, location,...
  - CNAME (canonical name = user@host) identifies user across media
- ***explicit leave (BYE):*** in addition to time-out
- ***extensions (APP):*** application-specific (none yet)

RTCP: packet structure



RTCP: multiple multicast senders



- each RTP session: typically a single multicast address;
- every participant: periodically multicast RTCP packet to same group as data
- RTP, RTCP packets distinguished from each other via distinct port numbers
- to limit traffic, each participant reduces RTCP traffic as number of conference participants increases

RTCP: announcement interval

- Goals:
  - estimate current number of participants & identities of participants – dynamic
  - source description (“SDES”) ➠ who’s talking?
  - quality-of-service feedback ➠ adjust sender rate
  - to O(1000) participants, few % of data
    - ➠ randomized response with rate ↓ as members ↑
  - group size limited by tolerable age of status
  - gives active senders more bandwidth
  - soft state: delete if silent

RTCP: bandwidth scaling

RTCP attempts to limit its traffic to 5% of session bandwidth

example : one sender, sending video at 2 Mbps

- RTCP attempts to limit RTCP traffic to 100 Kbps
- RTCP gives 75% of rate to receivers; remaining 25% to sender

- 75 kbps is equally shared among receivers:
  - with R receivers, each receiver gets to send RTCP traffic at 75/R kbps.
- sender gets to send RTCP traffic at 25 kbps.
- participant determines RTCP packet transmission period by calculating avg RTCP packet size (across entire session) and dividing by allocated rate

RTCP: bandwidth scaling

- sender period T:

$$T = \frac{\text{\# of senders}}{0.25 \cdot 0.05 \cdot \text{session bw}} \cdot \text{avg. RTCP packet size}$$

- receivers:

$$T = \frac{\text{\# of receivers}}{0.75 \cdot 0.05 \cdot \text{session bw}} \cdot \text{avg. RTCP packet size}$$

- next packet = last packet + $\max(5\text{ s}, T) \cdot \text{random}(0.5 \ldots 1.5)$
- randomization prevents “bunching”
- to reduce RTCP bandwidth, alternate between SDES components

RTCP sender reports (SR)

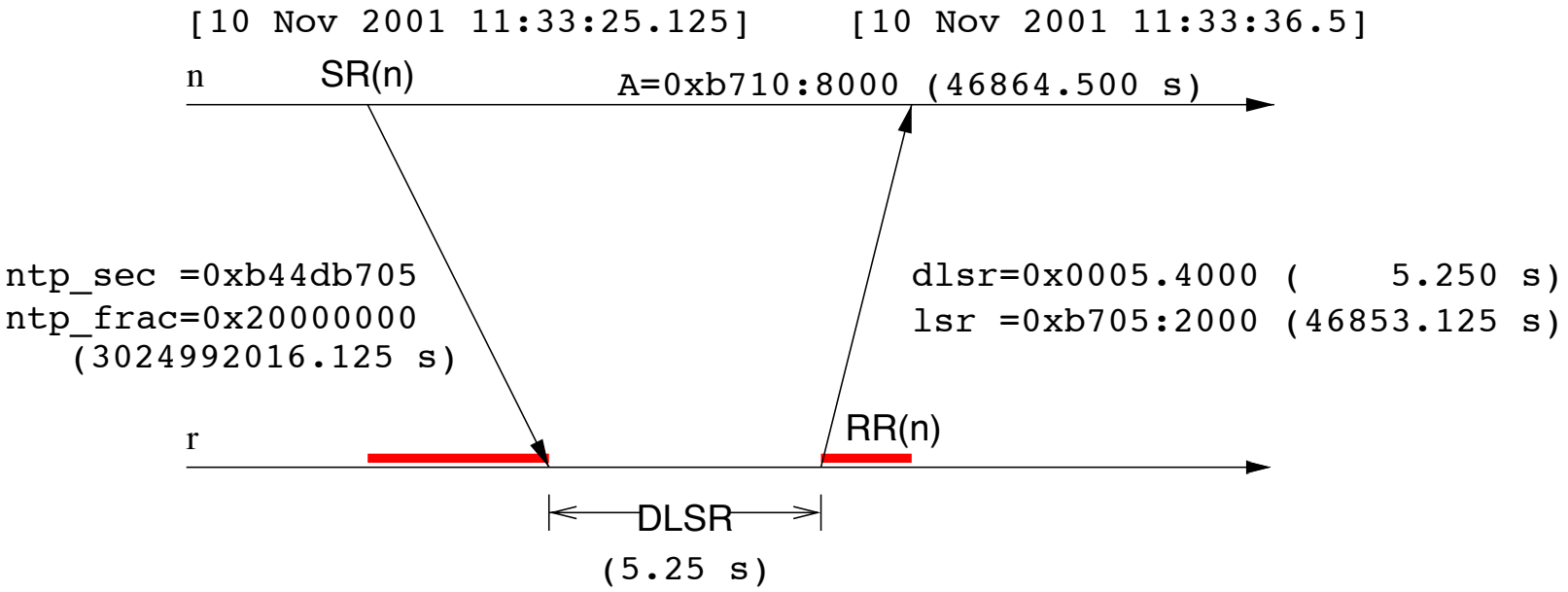
- *SSRC of sender:* identifies source of data
- *NTP timestamp:* when report was sent
- *RTP timestamp:* corresponding "RTP time" ➠ lip sync
- *sender's packet count:* total number sent
- *sender's octet count:* total number sent
- followed by zero or more receiver report

RTCP receiver reports (RR)

- *SSRC of source:* identifies who's being reported on
- *fraction lost:* binary fraction
- *cumulative number of packets lost:* long-term loss
- *highest sequence number received:* compare losses, disconnect
- *inter-arrival jitter:* smoothed inter-packet distortion
- *LSR:* time last SR heard
- *DLSR:* delay since last SR

RTCP: round trip delay estimation

- compute round-trip delay between data sender and receiver




```
A     0xb710:8000 (46864.500 s)
DLSR -0x0005:4000 (    5.250 s)
LSR  -0xb705:2000 (46853.125 s)
-------------------------------------
delay 0x   6:2000 (    6.125 s)
```

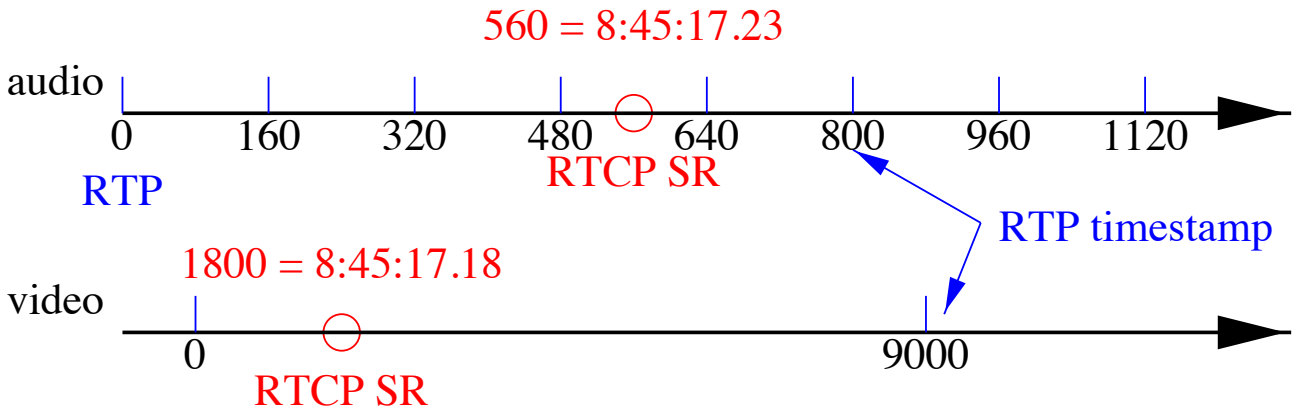
RTCP: stream synchronization

- RTCP can synchronize different media streams within a RTP session
- e.g., videoconferencing app: each sender generates one RTP stream for video, one for audio.
- timestamps in RTP packets tied to the video, audio sampling clocks
  - *not* tied to wall-clock time
- each RTCP sender-report packet contains (for most recently generated packet in associated RTP stream):
  - timestamp of RTP packet
  - wall-clock time for when packet was created
- receivers uses association to synchronize playout of audio, video

RTCP: stream synchronization

= sync different streams (audio, video, slides, . . .)

- timestamps are offset with random intervals
- may not tick at nominal rate
- SRs correlate "real" time (wall clock time) with RTP ts



Today's Outline

- RTP
  - protocol goals
  - mixers and translators
  - control: awareness, QOS feedback
  - media adaptation

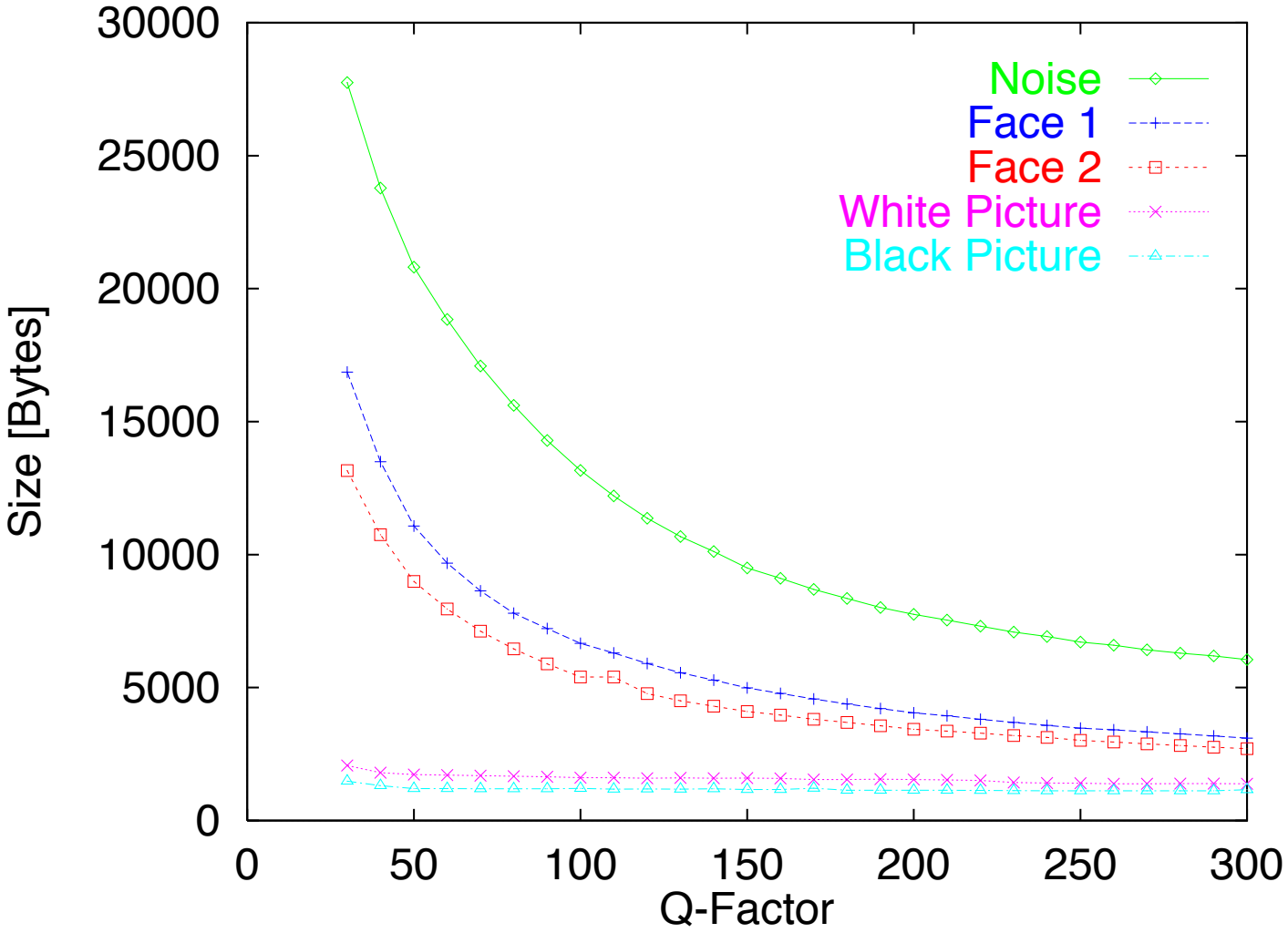
Media Adaptation

- Multimedia applications can adjust their data rates:
- Audio: (MPEG L3), encoding, sampling rate, mono/ stereo

encoding	sampling rate	bit rate
LPC	8,000	5,600
GSM	8,000	13,200
DVI4	8,000	32,000
μ-law	8,000	64,000
DVI4	16,000	64,000
a range of DVI4 and MPEG L3		
L16 stereo	44,100	1,411,200

Media Adaptation

- Video: frame rate, quantization, image resolution, encoding



Application Control

- networks without QoS or shared reserved link:
  - adapt application to available bandwidth
  - share bandwidth fairly with TCP?
  - lowest common denominator
    - mixers, translators

RTP/RTCP *Does NOT*

- Define media data formats or encodings
  - Need media specific profiles
- Handle connection setups
  - Need other protocols like SIP or H.323
- Handle resource reservation
  - Need other protocols like RSVP
- Guarantee timely data delivery or Quality of Service
  - However, it does provide necessary data to application to order packets and adjust signal quality

References

- RFC 3550 - http://tools.ietf.org/html/rfc1889
- RFC 3551 - http://tools.ietf.org/html/rfc3551

- Wikipedia:
  - RTP - http://en.wikipedia.org/wiki/Real-time_Transport_Protocol
  - RTCP - http://en.wikipedia.org/wiki/RTCP