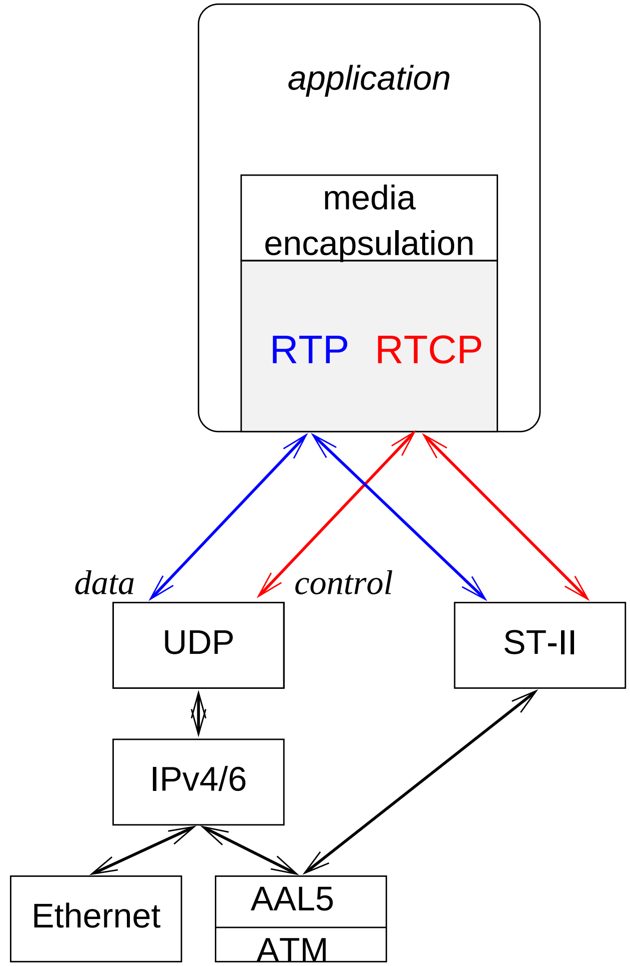
Real-Time Transport Protocol (RTP)

RTP

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

RTP – the big picture



RTP = Real-time transport protocol

- only part of puzzle: reservations, OS, ...
- product of Internet Engineering Task Force, AVT WG
- RFC 1889, 1890
- ITU H.323, RTSP, ...
- support for functions, but does not restrict implementation
- compression for low-bandwidth networks under study

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 $O(1000)$

separate control/data: some functions may be taken over by conference control protocol

secure: support for encryption, possibly authentication

RTP functions

- segmentation/reassembly done by UDP (or similar)
- resequencing (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 mixers, translators, ...

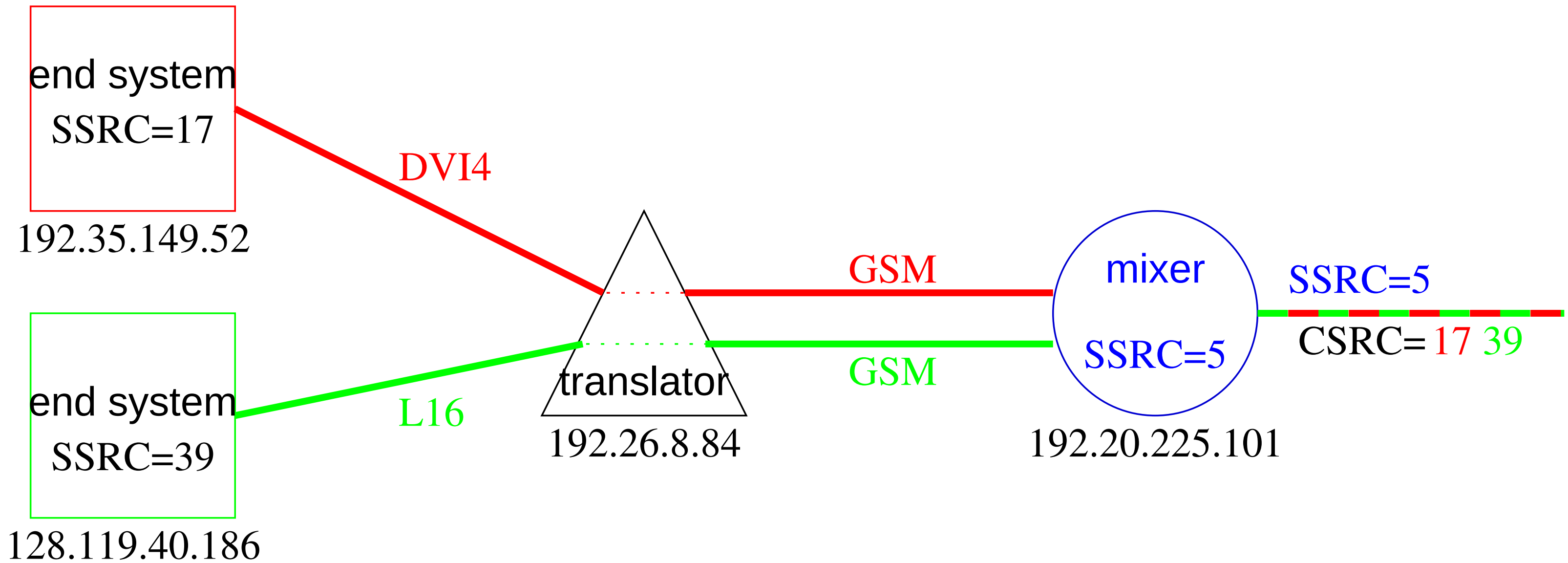
mixer:

- several media stream ➠ one new stream (new encoding)
- mixer: reduced bandwidth networks (dial-up)
- appears as new source, with own identifier

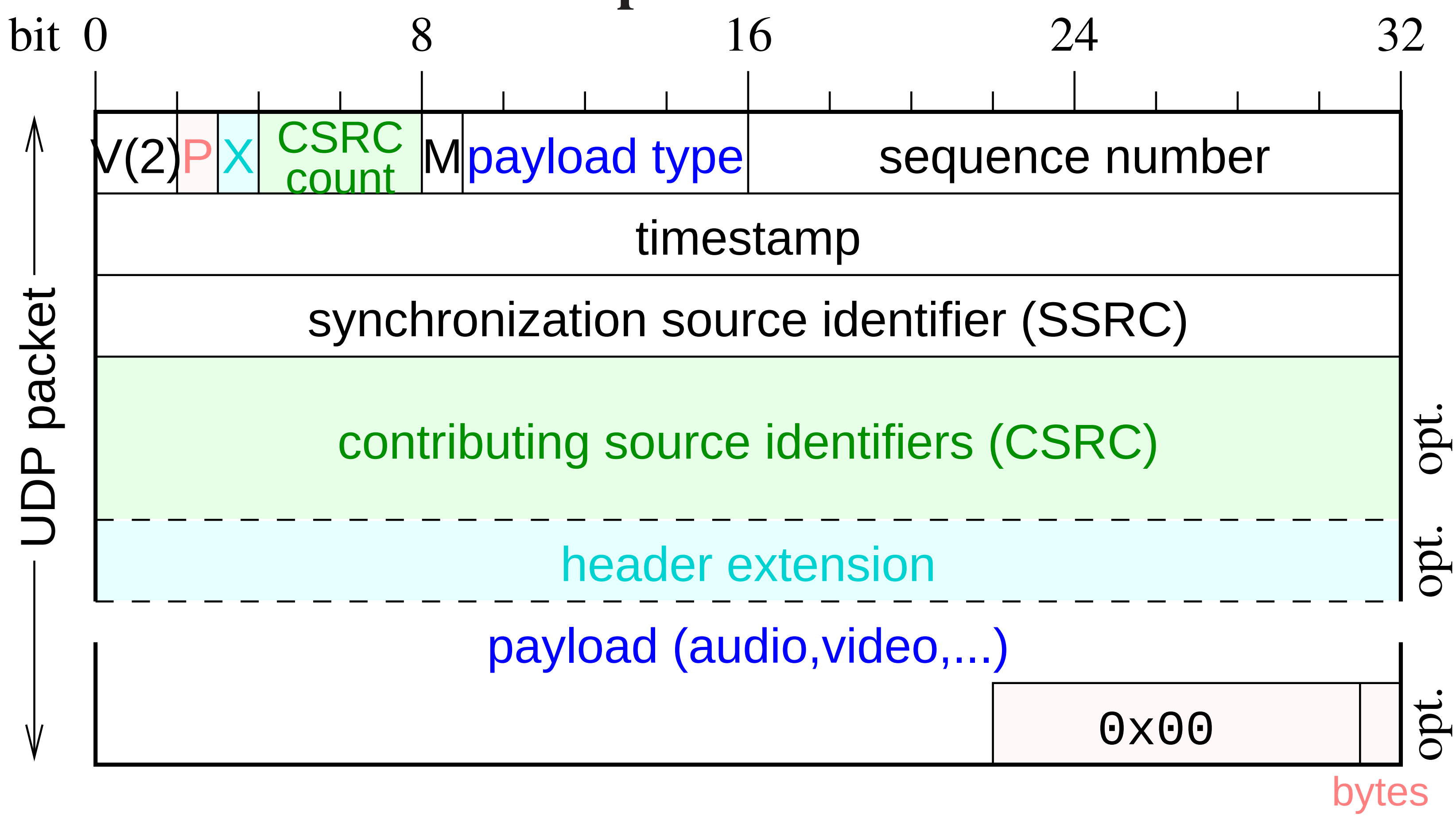
translator:

- single media stream
- *may* convert encoding
- protocol translation (native ATM $\leftrightarrow$ IP), firewall
- all packets: source address = translator address

RTP mixers, translators, . . .



RTP packet header



RTP packet header

Payload type: audio, video encoding method; may change during session

SSRC: sychronization source ➠ each source picks at random ➠ may change after *collision*!

sequence number: incremented by 1 for each packet ➠ gaps ≡ loss

P: padding (for encryption) ➠ last byte contains padding count

M: marker bit; indicates frame, beginning of talkspurt ➠ allow delay adjustment

CC: content source count (for mixers)

CSRC: list of identifiers of those contributing to (mixed into) packet

RTP timestamp

- incremented by 1 for each sample (e.g., 160 for 20 ms packets @ 8000 Hz)
- random starting value
- constant rate for each audio payload type (e.g., 8000 Hz for PCM μ-law, 44100 Hz for linear, 16-bit)
- 90 kHz for video
- several video frames may have same timestamp
- ➠ gaps $\equiv$ silence
- time per packet may vary
- video frame maybe split (carefully…) over several packets
- typical: 20 to 100 ms of audio

RTP in a network

- typical: UDP, no fixed port; RTCP port = RTP port (even) + 1
- typical UDP size limited to few hundred bytes (OS, network, fragmentation)
- native ATM: directly into AAL5 frame
- encapsulation (length field) for others
- typically: one media (audio, video, ...) per port pair
- exception: bundled MPEG

RTP control protocol – types

stackable packets, similar to data packets

sender report (SR): bytes send ➠ estimate rate;
timestamp ➠ synchronization

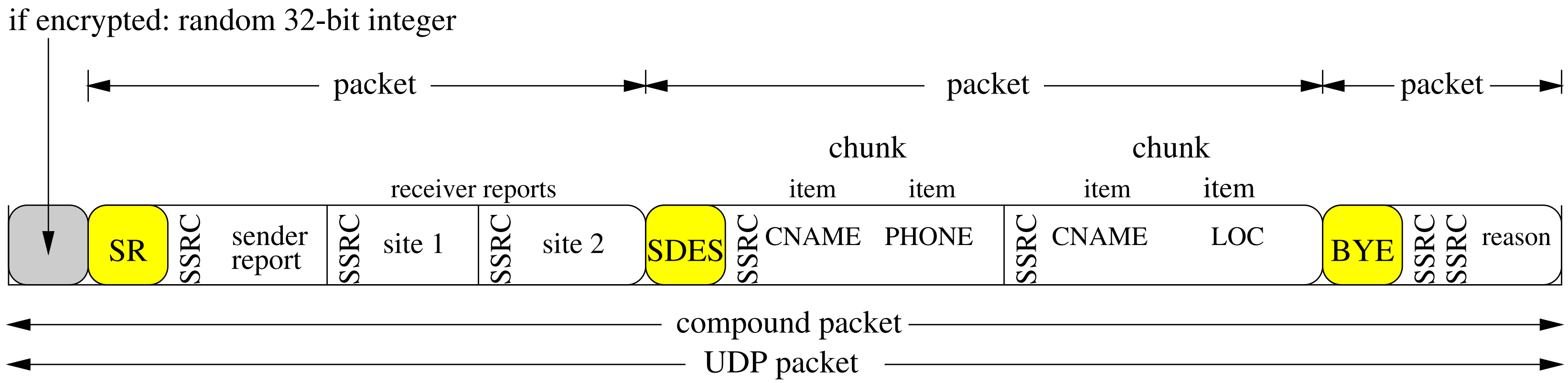
reception reports (RR): number of packets sent and expected ➠ loss,
interarrival jitter, round-trip delay

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 announcement interval computation

Goals:

- estimate current number & identities of participants – dynamic
- source description ("SDES") ➠ who's talking?
- quality-of-service feedback ➠ adjust sender rate
- scale to $O(1000)$ participants, small fraction of data bandwidth

➠ randomized response with rate $\downarrow$ as members $\uparrow$

- group size limited by tolerable age of status
- gives active senders more bandwidth
- soft state: if not heard from for multiple of announcement interval, delete

RTCP bandwidth scaling

- every participant: periodically multicast RTCP packet to same group as data
- ➠ everybody knows (eventually) who's out there
- session bandwidth:
  - single audio stream
  - $\sum$ of concurrently active video streams
- sender period T:

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

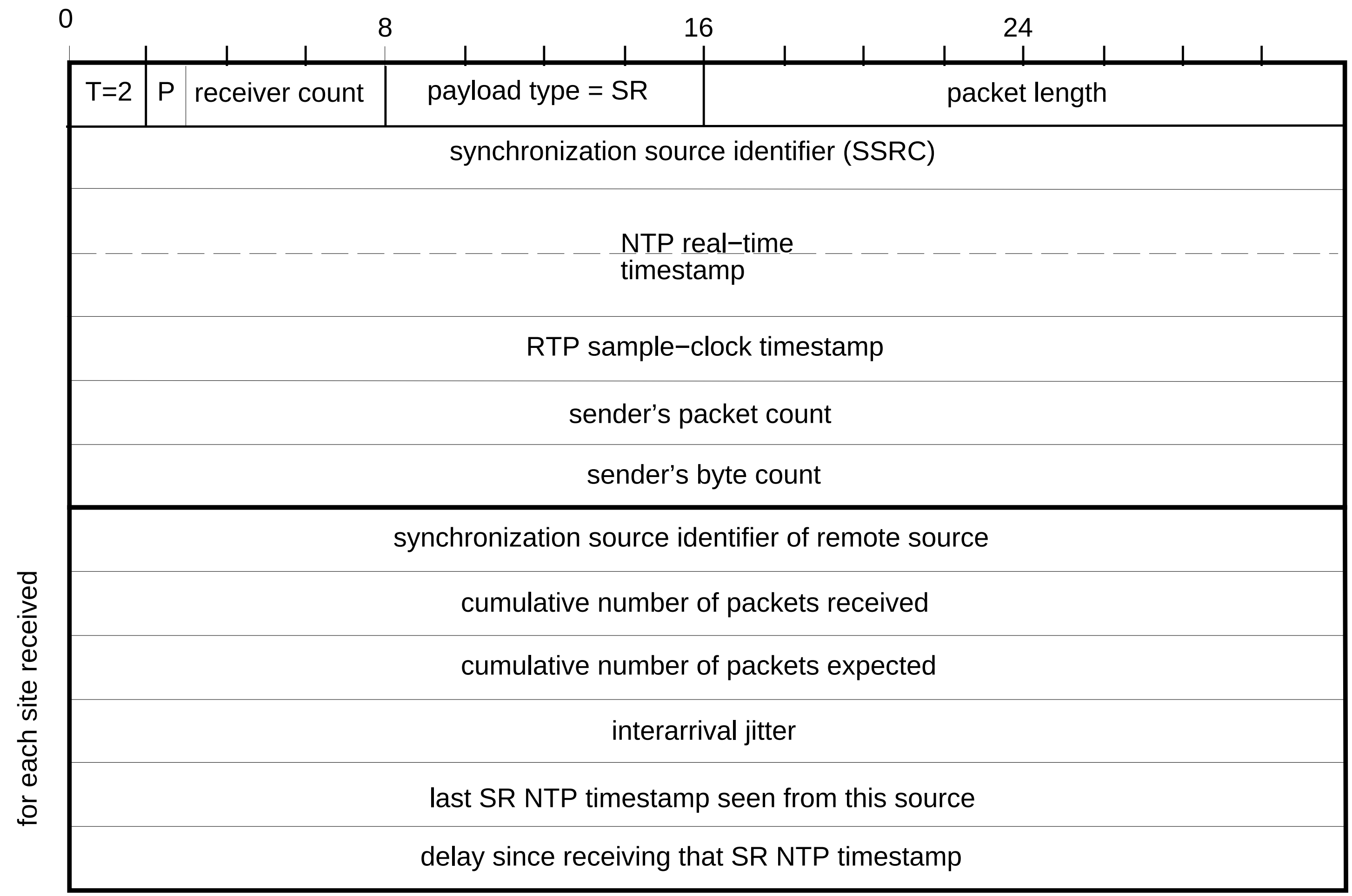
- receivers:

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

RTCP bandwidth scaling

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

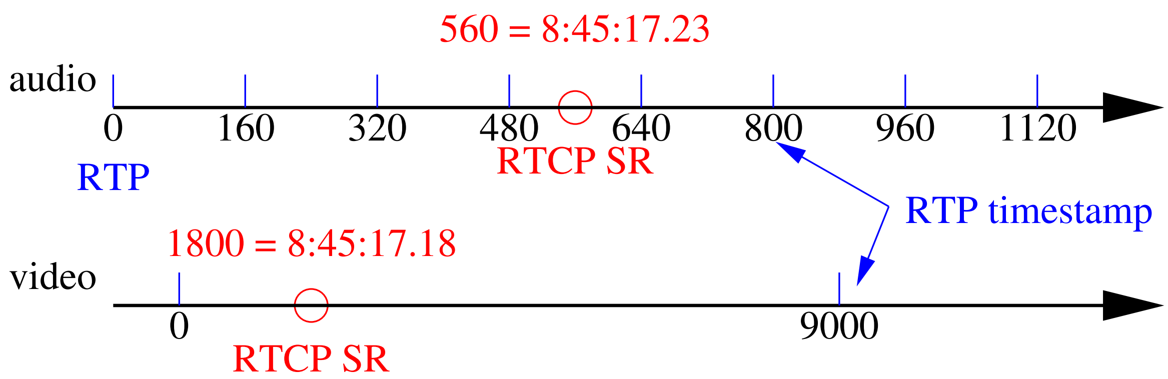
RTCP sender reports (SR)



Intermedia synchronization

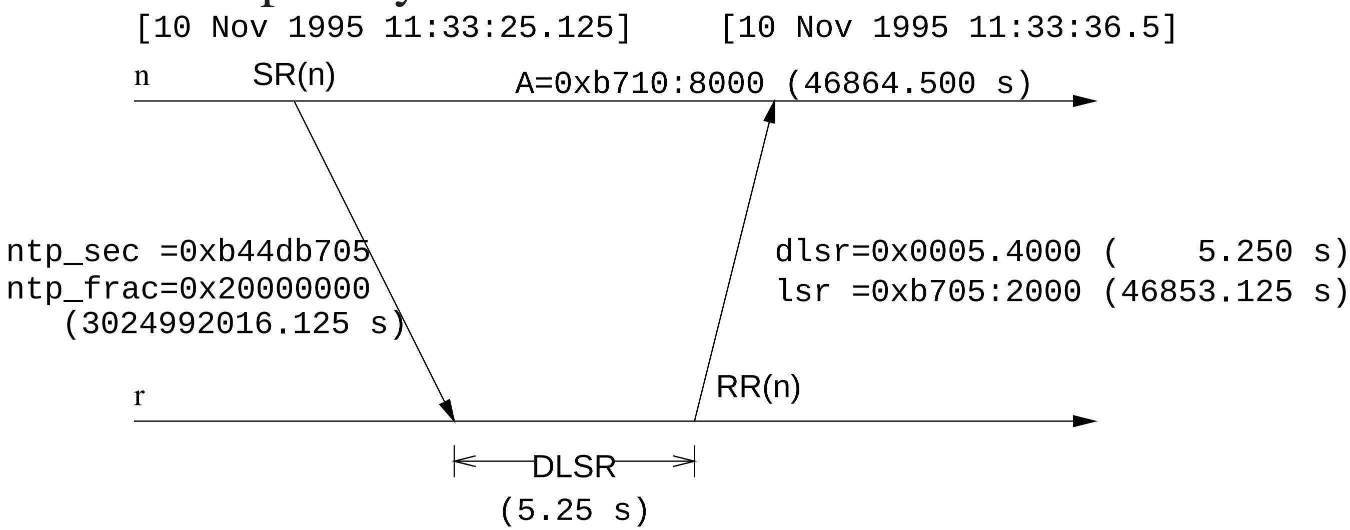
= synchronization between different streams (audio, video, slides, . . .)

- timestamps are offset with random intervals
- may not tick at nominal rate
- every SR correlates "real" time (wallclock time) with RTP timestamp
- ➠ compute when sample was generated



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)
```

Collision detection and resolution

Collision:

- two sources may pick the same SSRC (“birthday problem”)
- probability: about 10^{-4} if 1000 session members join more or less simultaneously
- but: don’t pick one you know about already ➠ probability much lower unless everyone joins at the same time
- send BYE for old, pick a new identifier

Loops:

- forward packet to same multicast group (directly or through translators)
- looks similar to collision, but changing SSRC doesn’t help

RTP for the masses

Problems using RTP for radio, TV:

- for 14.4 RealAudio: 90 bytes/second $\approx$ one new site per second
- takes $\approx$ 3 hours to get to know 10,000 people ➠
  - who cares? (Nielsen!)
  - useless for QOS feedback
  - control rate too high
- ➠ faster convergence: everybody reports estimate, compute max()
- ➠ statistical sample (sender determines rate): send value $[0, 1]$; pick random value; if $<$, lucky winner ➠ needs to be adaptive
- ➠ report just to sender, instead of multicast

Adaptive applications

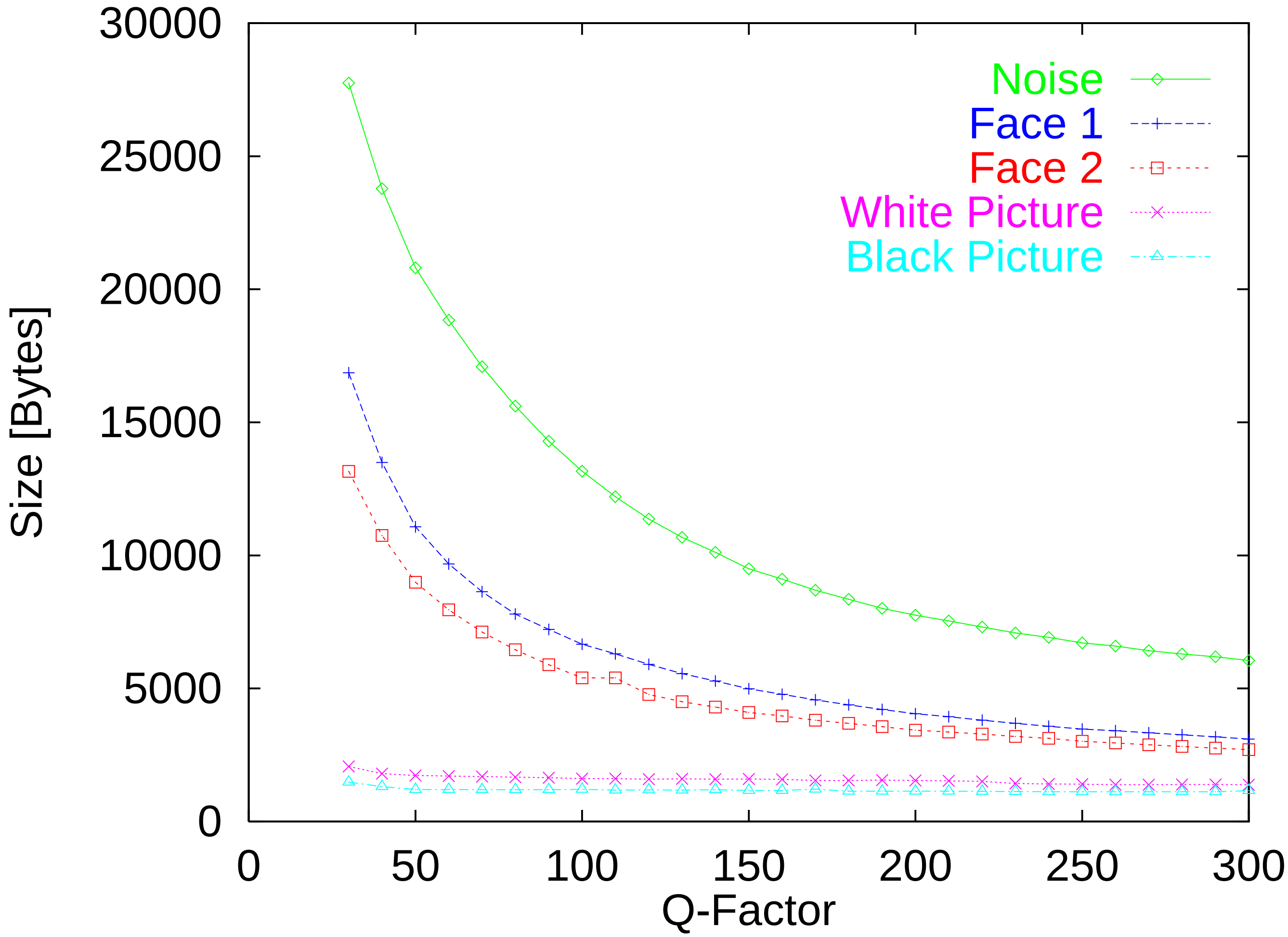
Adaptive applications

Multimedia applications can adjust their data rates:
Audio: encoding parameters (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

Adaptive applications

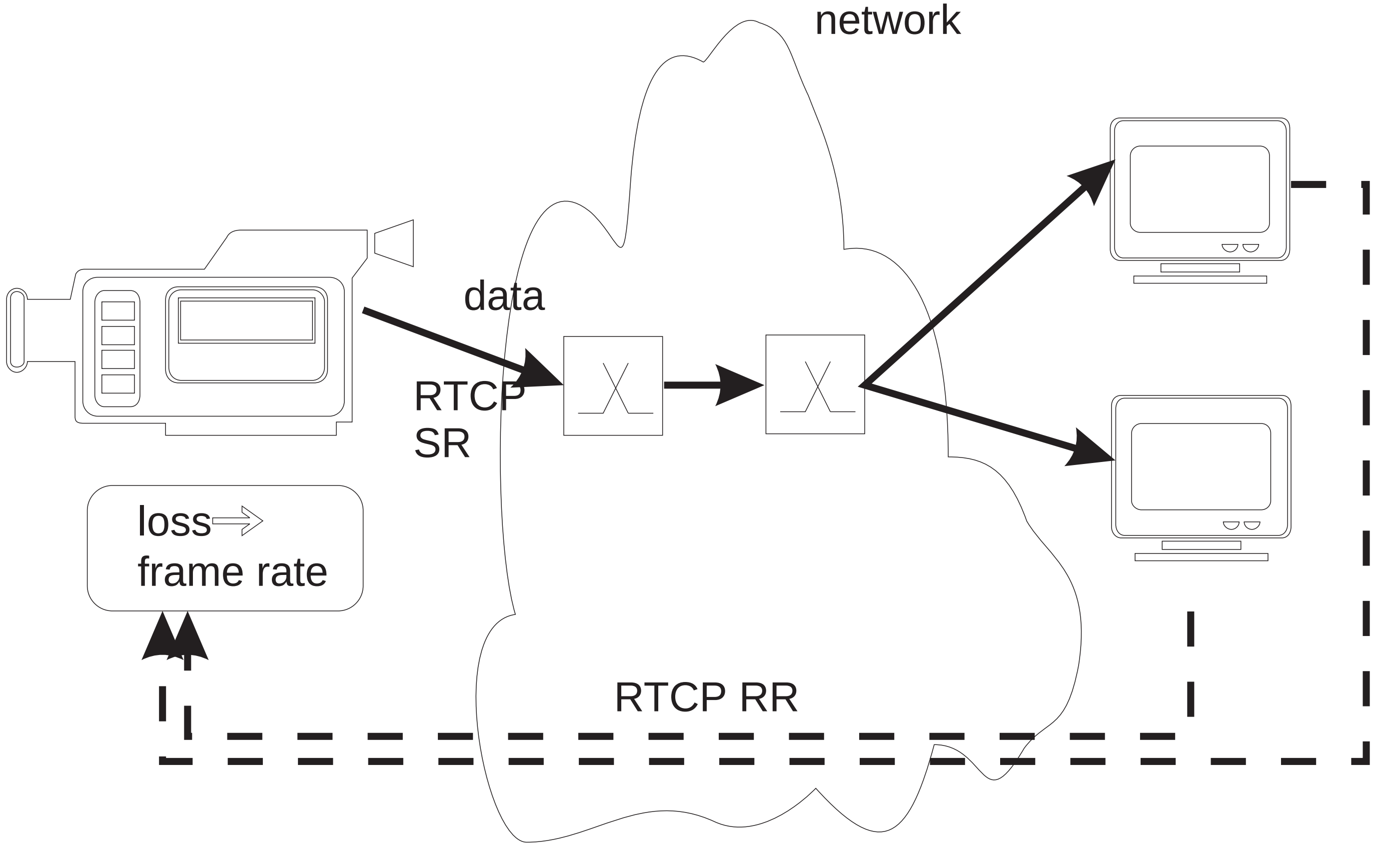
Video: frame rate, quantization, image resolution, encoding



Application control

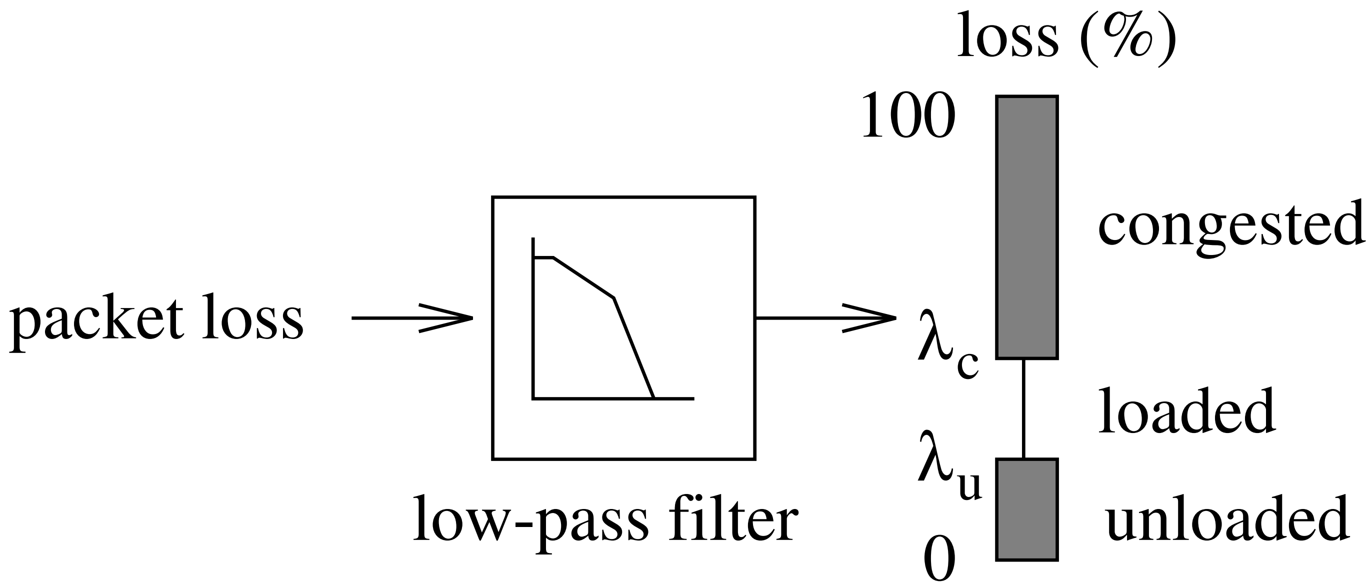
- networks with QoS guarantees:
  - QoS negotiation at call set-up time, network guarantees this quality
  - long call durations ➠ network load may change significantly
  - “wrong” initial allocation ➠ many rejected calls or low quality
- networks without QoS guarantees (= current Internet) or shared reserved link:
  - adapt application to available bandwidth
  - share bandwidth fairly with TCP?
  - lowest common demoninator ➠ combine with mixers, translators

End-to-End feedback control mechanism



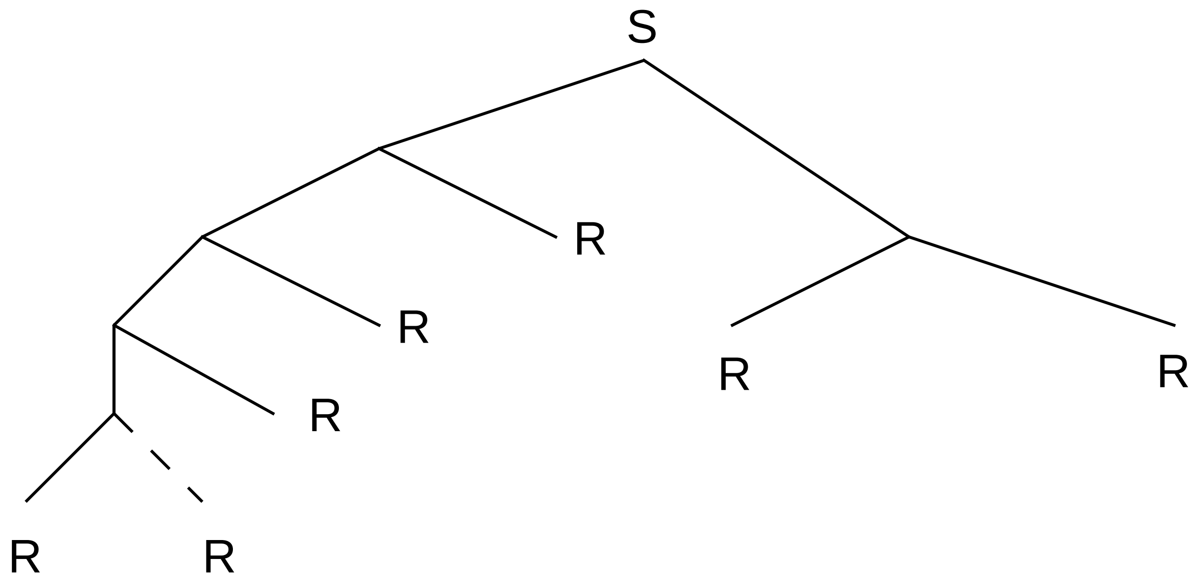
video application bandwidth is based on network feedback:
low losses ➠ slow bandwidth increase ➠ higher framerate
high losses ➠ bandwidth decrease ➠ lower framerate

Network state estimation and bandwidth adjustment



- loss information is filtered: $\lambda \leftarrow (1-\alpha)\lambda + \alpha b,\ \alpha \leq 1$
- linear regulator with deadzone
- multiplicative decrease if network is congested:
  $b_a \leftarrow \max\{b_a * \mu, b_{min}\},\ \mu < 1$
- additive increase if network is unloaded: $b_a \leftarrow \min\{b_a + \nu, b_{max}\}$

Multicast scalability



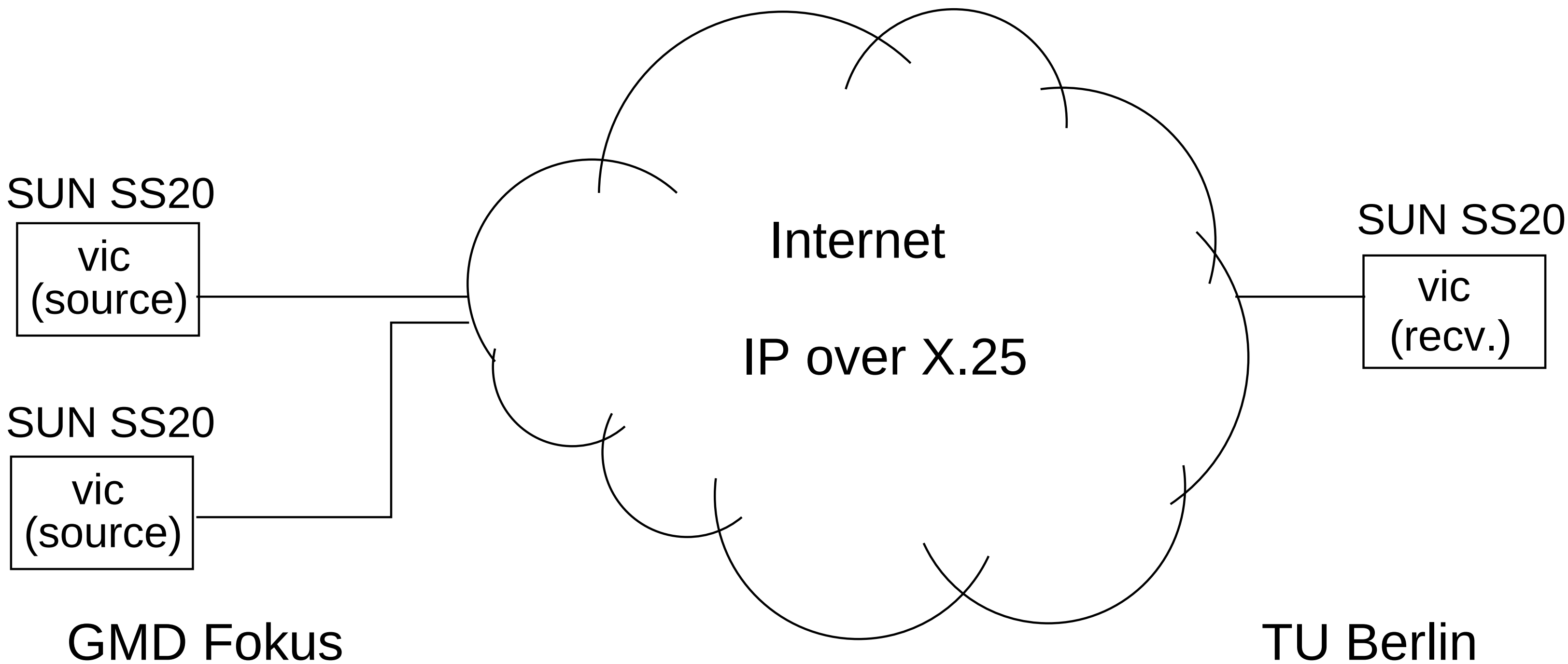
algorithm 1: adjust according to the worst-positioned receiver

algorithm 2: allow a fraction of the receivers to be congested

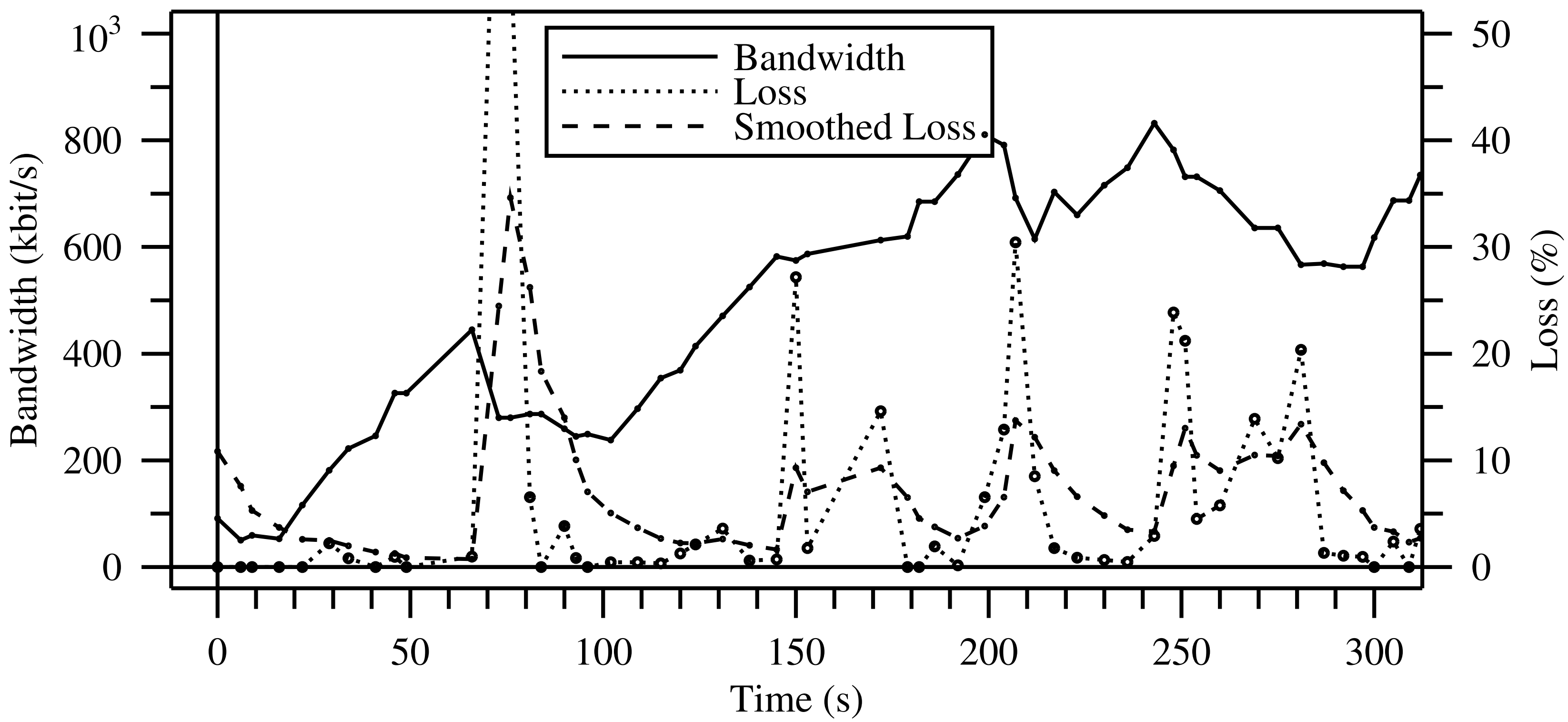
other solutions: video gateways, layered encodings

Internet scenario

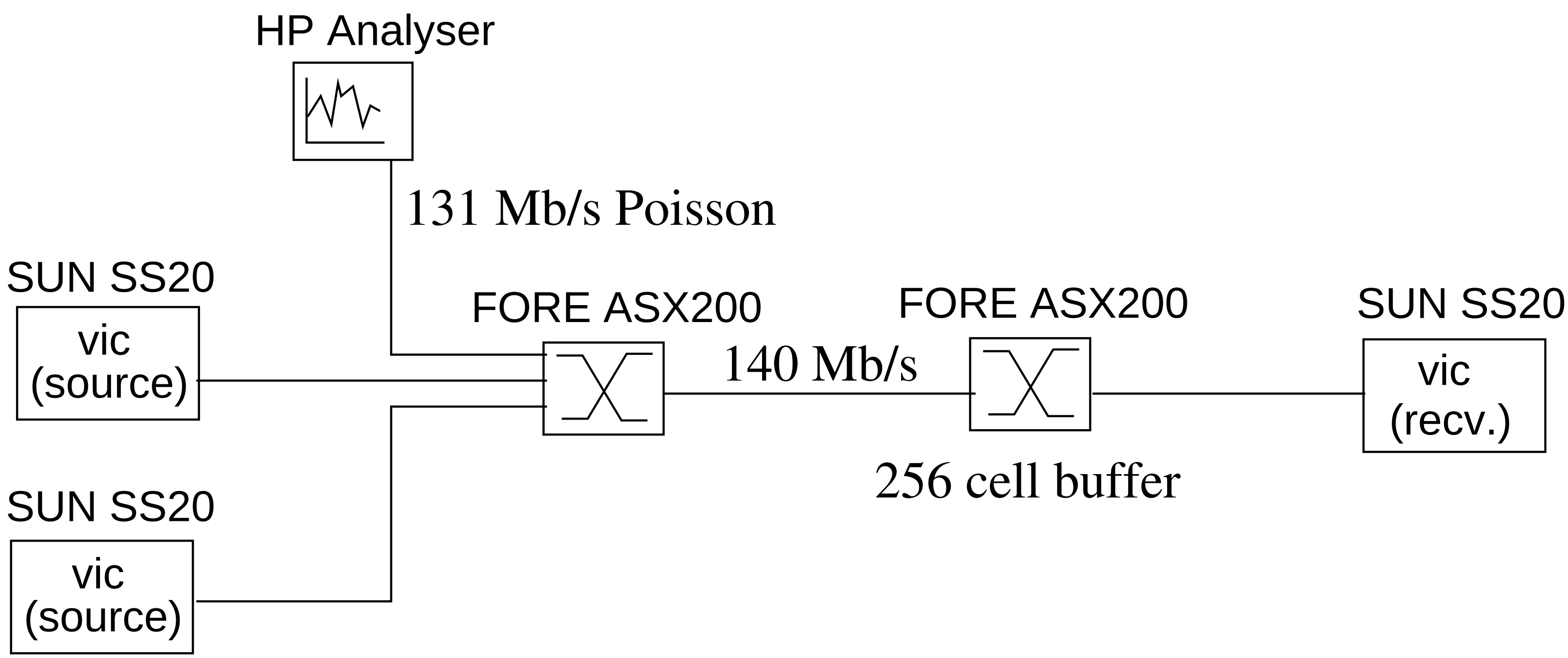
- Measurements made on the 2 Mbit/s X.25 link between GMD Fokus and TU Berlin (5 hops distance)
- deadzone between 5% and 10%

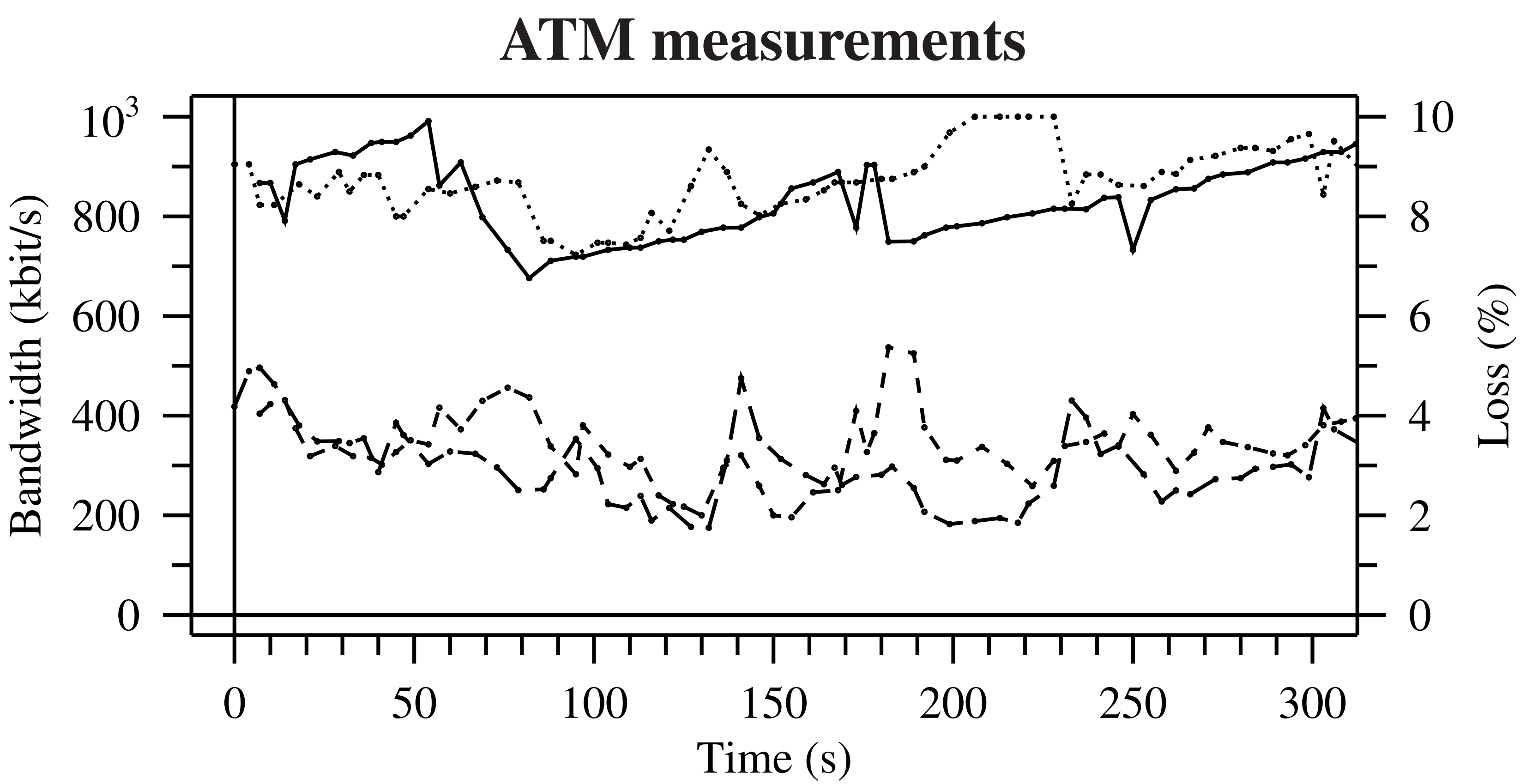


Internet measurement



ATM scenario




ATM measurements
10^3
800
600
400
200
0
Bandwidth (kbit/s)
10
8
6
4
2
0
Loss (%)
0
50
100
150
200
250
300
Time (s)

Observations

- jitter as loss predictor does not work well
- coexistence with controlled data applications
- use in ABR-like services?
- *no impact policy:* keep loss to within range observed without video application (monitor in transmission pauses)
- loss compensation can be dangerous ➠ ever higher loss fractions

RTP Implementations

tool	who	media	RSVP	adaptive
NeVoT	GMD Fokus	audio	yes	not yet
NeViT	GMD Fokus	video	yes	yes
Fphone	INRIA	audio	no	yes
vic	LBNL	video	no	no
vat	LBNL	audio	no	no

http://www.cs.columbia.edu/~hgs/rtp/