Transport Protocols for Multimedia

Henning Schulzrinne
Dept. of Computer Science
Columbia University
New York, New York
schulzrinne@cs.columbia.edu

March 1998 – Infocom 1998, San Francisco

Overview

- network characteristics
- RTP
- RTP: synchronization, playout delay compensation, aggregation
- scaling RTP to large groups
- congestion control: adaptive applications
- error repair and correction

Transport Protocols

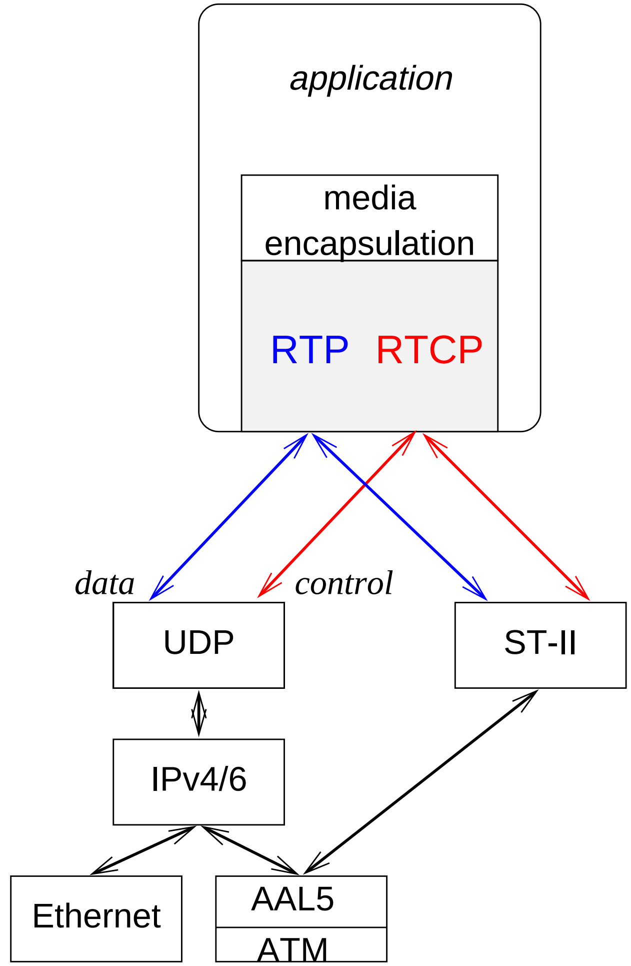
	data	real-time
sequencing	yes	yes, but oo delivery
reliability	full	partial
multicast	rare	common
timing	N/A	yes
congestion c.	blind	media-aware
flow control	yes	inherent
intermediate sys.	firewalls	mixers, translators

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 (to be revised)
- ITU H.323 (conferencing, Internet telephony), 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 $O(10^7)$

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

secure: support for encryption, possibly authentication

Data transport – RTP

Real-Time Transport Protocol (RTP) = data + control

data: timing, loss detection, content labeling, talkspurts, encryption

control: (RTCP) ➠ periodic with $T \sim$ population

- QOS feedback
- membership estimation
- loop detection

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, …

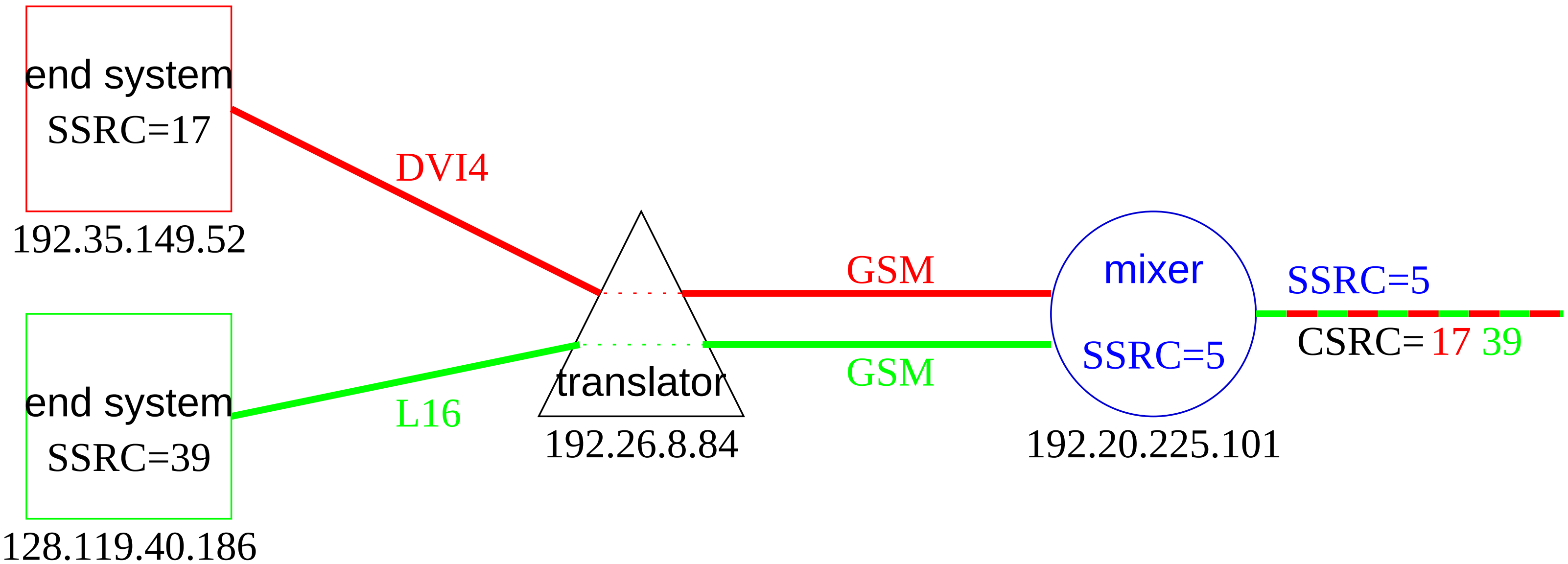
mixer:

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

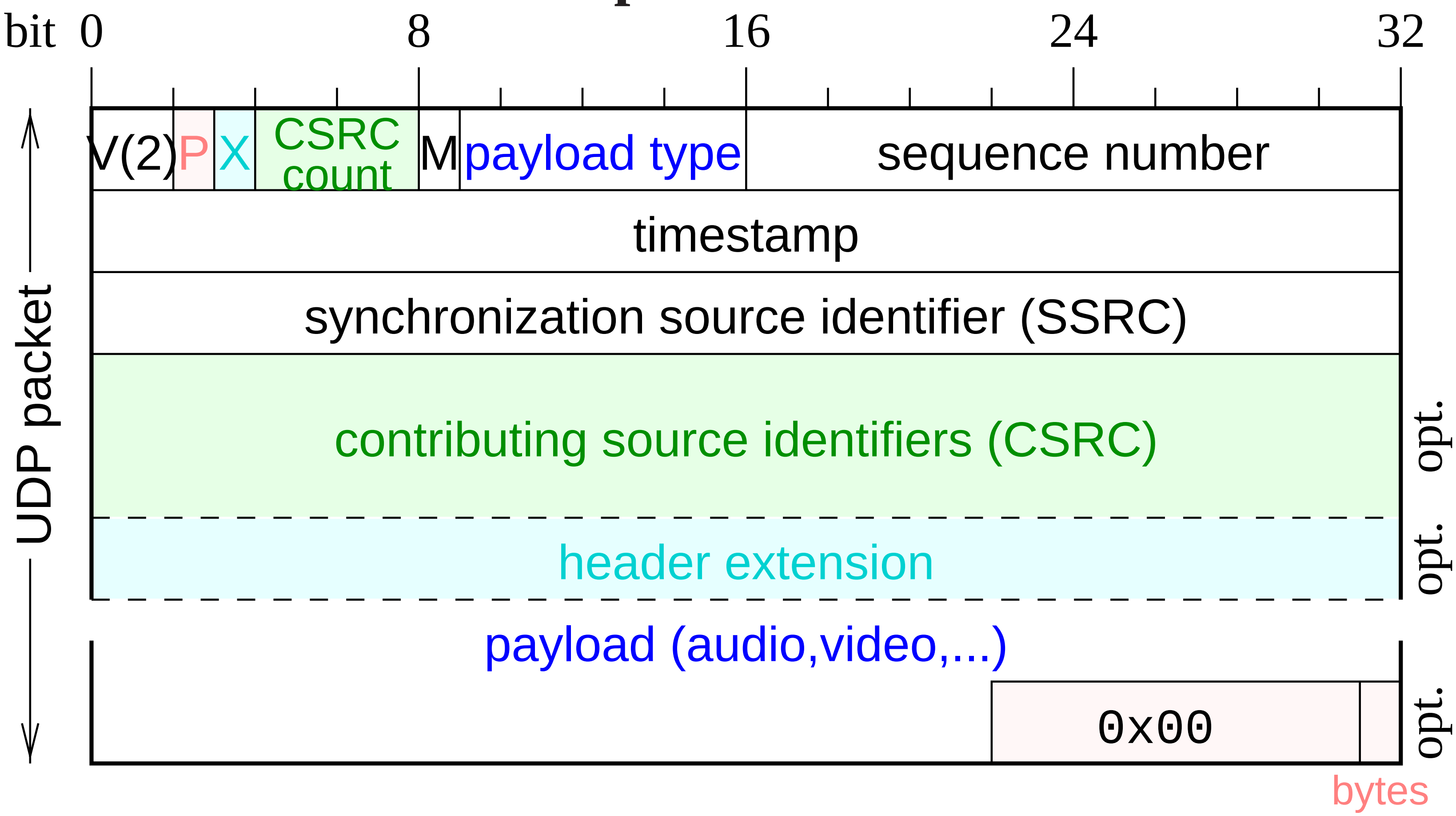
translator:

- 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

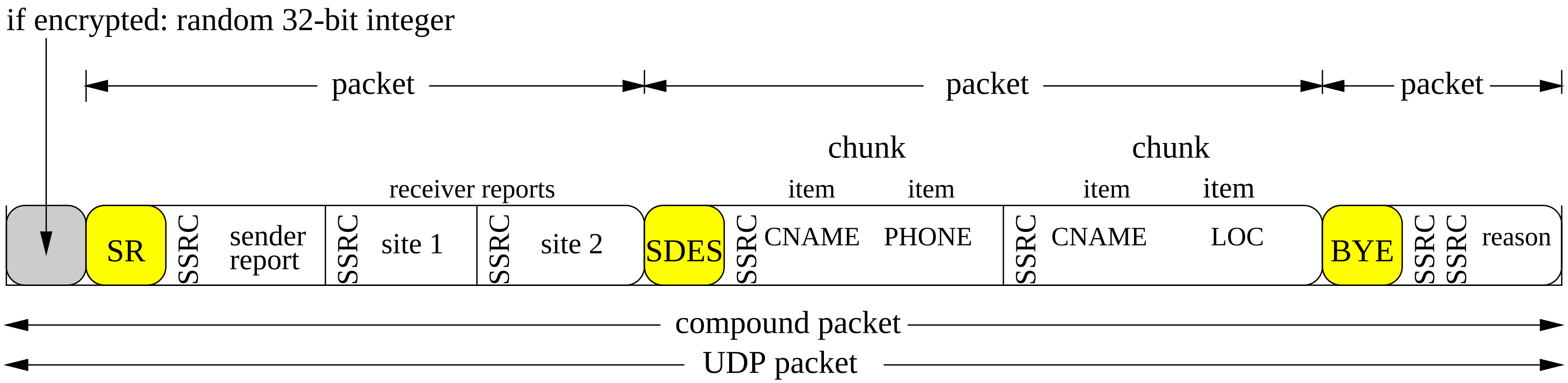
reception reports (RR): number of packets sent and expected ➠ loss,
interarrival jitter, round-trip delay

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

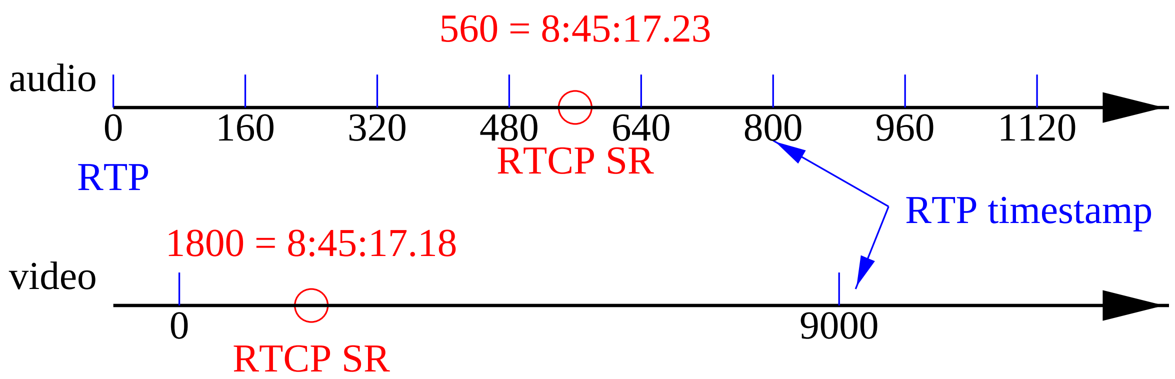
RTCP sender reports (SR)

```
 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|    RC   |   PT=SR=200   |             length            | header
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         SSRC of sender                        |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|              NTP timestamp, most significant word             | sender
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ info
|             NTP timestamp, least significant word             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         RTP timestamp                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     sender’s packet count                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      sender’s octet count                     |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|                 SSRC_1 (SSRC of first source)                 | report
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ block
| fraction lost |       cumulative number of packets lost       |   1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           extended highest sequence number received           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      interarrival jitter                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         last SR (LSR)                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   delay since last SR (DLSR)                  |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|                 SSRC_2 (SSRC of second source)                | report
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ block
:                               ...                             :   2
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|                  profile-specific extensions                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Intermedia synchronization

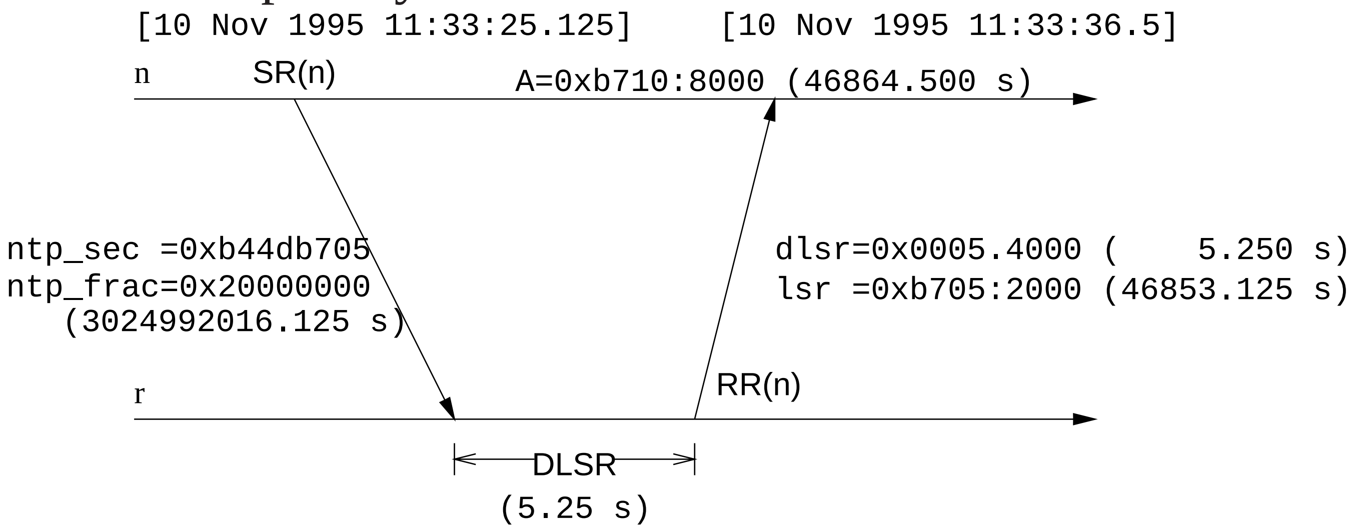
= synchronization between different streams (audio, video, slides, . . .)

- 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)
```

RTP: Large groups

How do manage large groups?

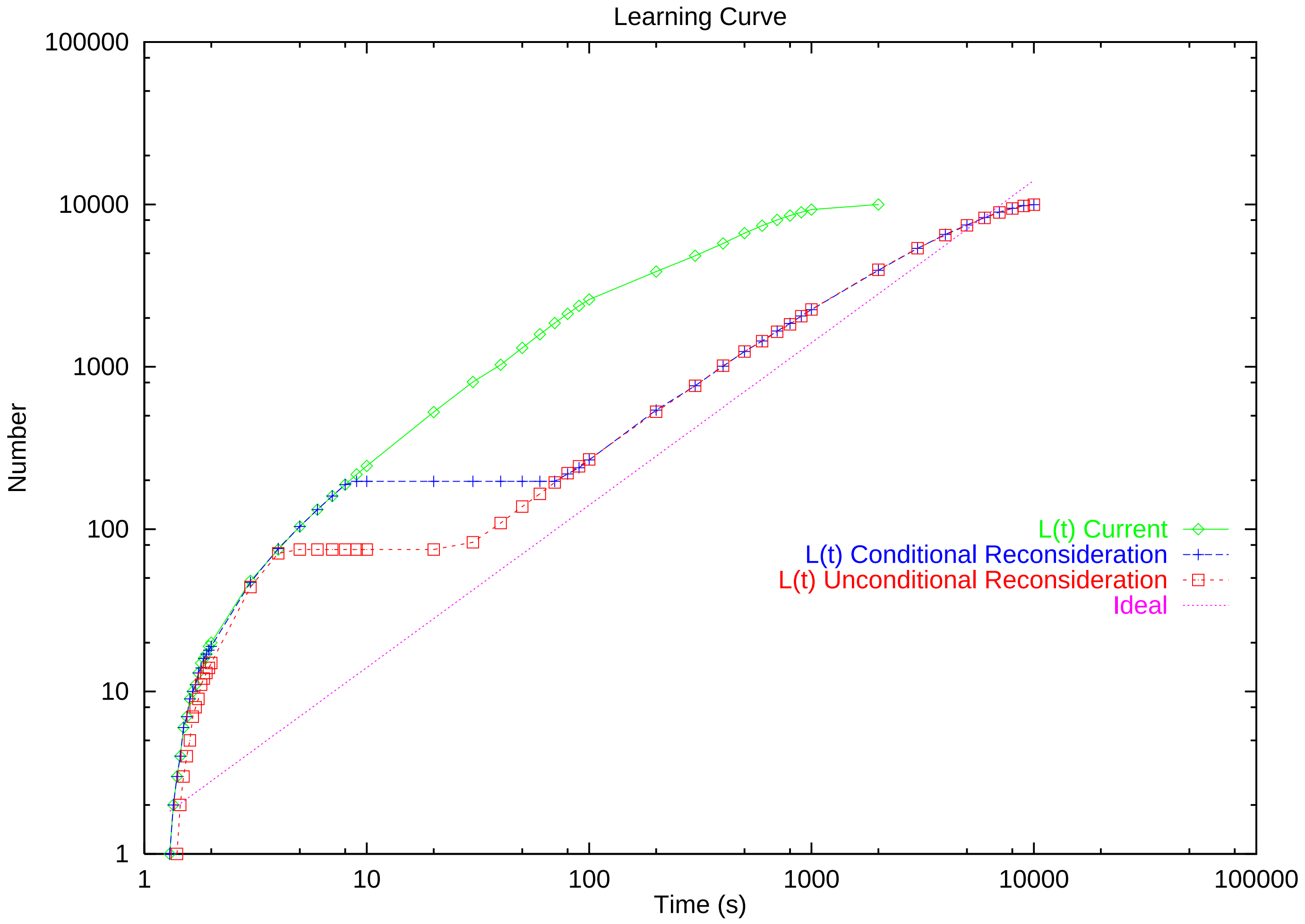
- “movie at ten”
- channel surfing

➠

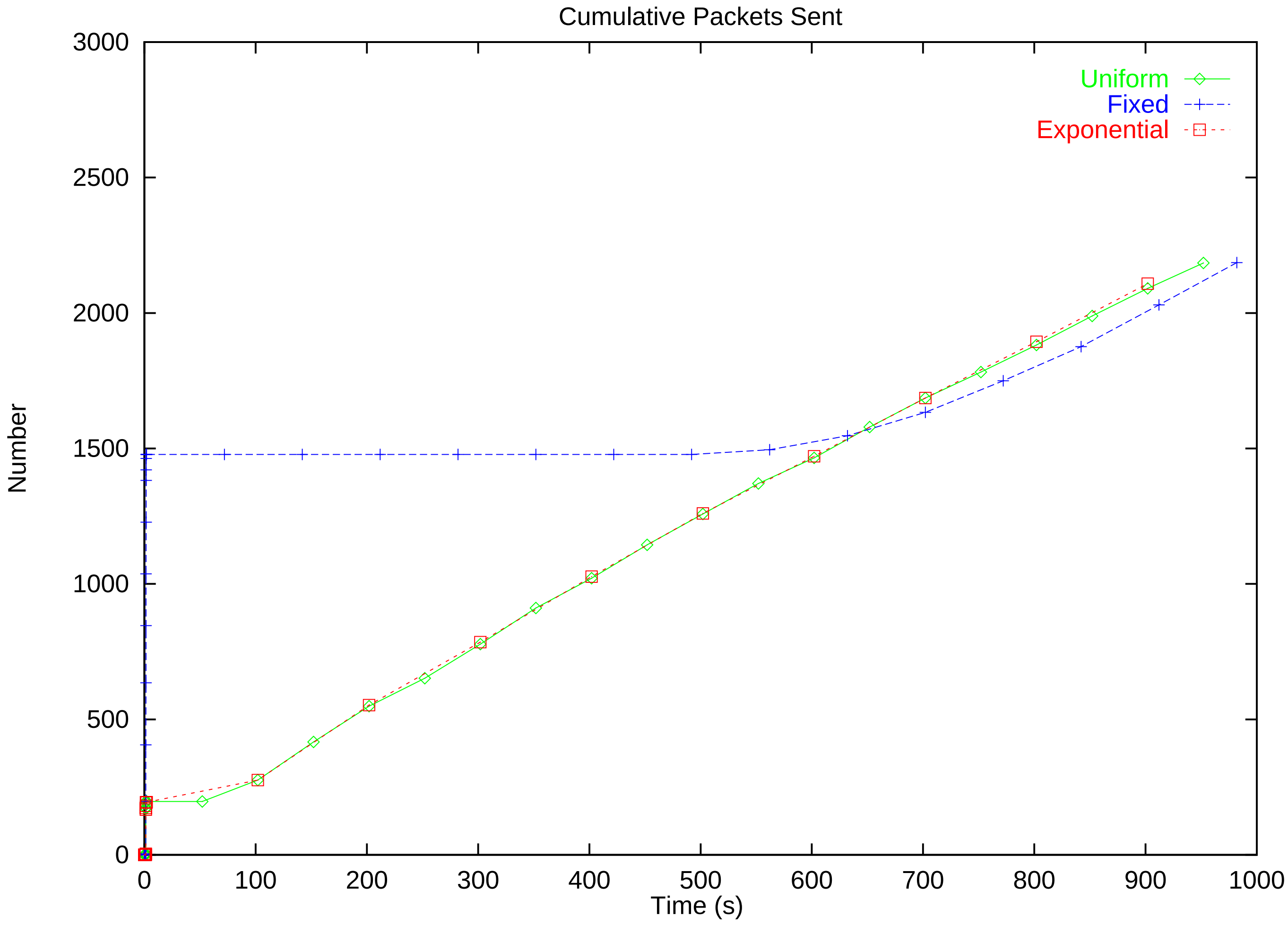
- reconsideration: pause and recompute interval
  - conditional reconsideration: only if group size estimate increases
  - unconditional reconsideration: always
- avoid BYE floods: don’t send BYE if no RTCP, reconsideration
- reverse reconsideration to avoid time-outs
- “squeaky wheel” network management

➠ general bandwidth sharing problem

Reconsideration: learning curve



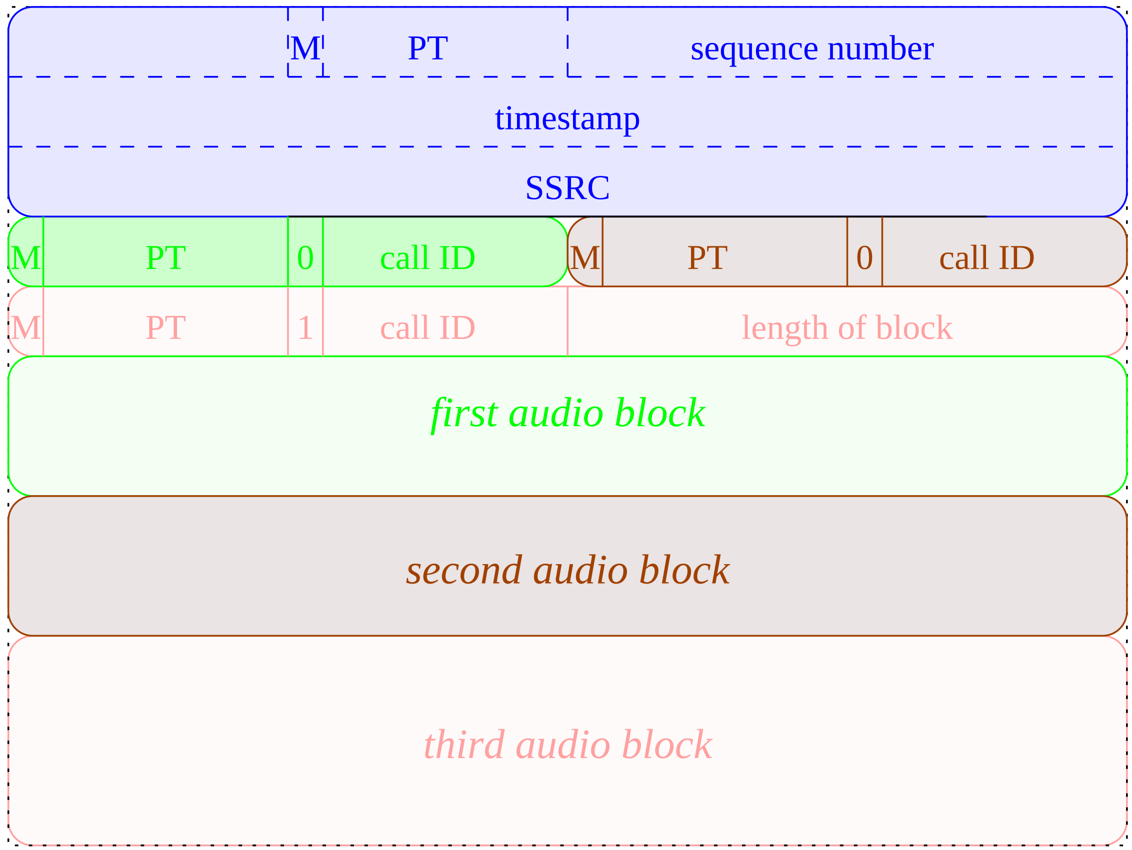
Reconsideration: influence of delay



RTP: Aggregation

- interconnected IPTel gateways ➠ several RTP streams to same destination
- high overhead: G.729, 30 ms packetization ➠ 30 bytes audio, 40 bytes IP + UDP + RTP headers
- with ATM: efficiency = 28%
- solution: bundle several calls into single RTP session

RTP: Aggregation



- for 24 channels ➠ efficiency ↑ 89%
- signal call-ID using SIP

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

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
rat	UCL	audio	no	no
NetMeeting	Microsoft	A/V	no	no
IP/TV	Precept	A/V	no	no

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

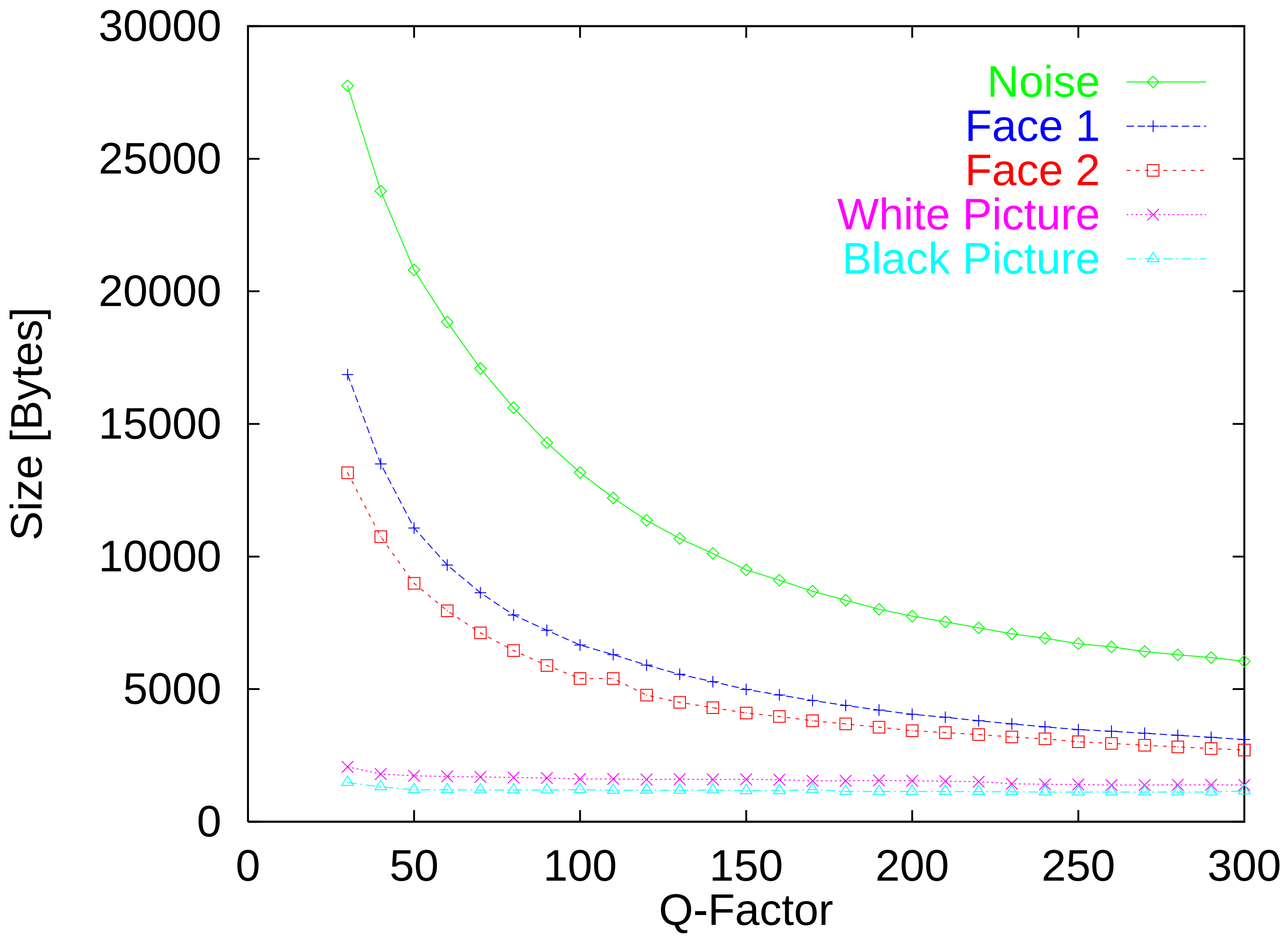
Adaptive applications

Adaptive applications: Audio

encoding parameters (MPEG L3), encoding, sampling rate, mono/stereo

encoding	sampling rate	bit rate
LPC	8,000	2.4
G.723.1	8,000	6.3
GSM	8,000	13.2
DVI4	8,000	32.
μ-law	8,000	64.
DVI4	16,000	64.
a range of DVI4 and MPEG L3		
L16 stereo	44,100	1,411.2

Adaptive Applications: Video



Adaptive Applications: Video

quantization: Q factor = 1…31: 3 very high quality; artifacts for $Q > 20$

frame rate: but increases per-image rate for conditional replenishment! change coding as frame rate ↓

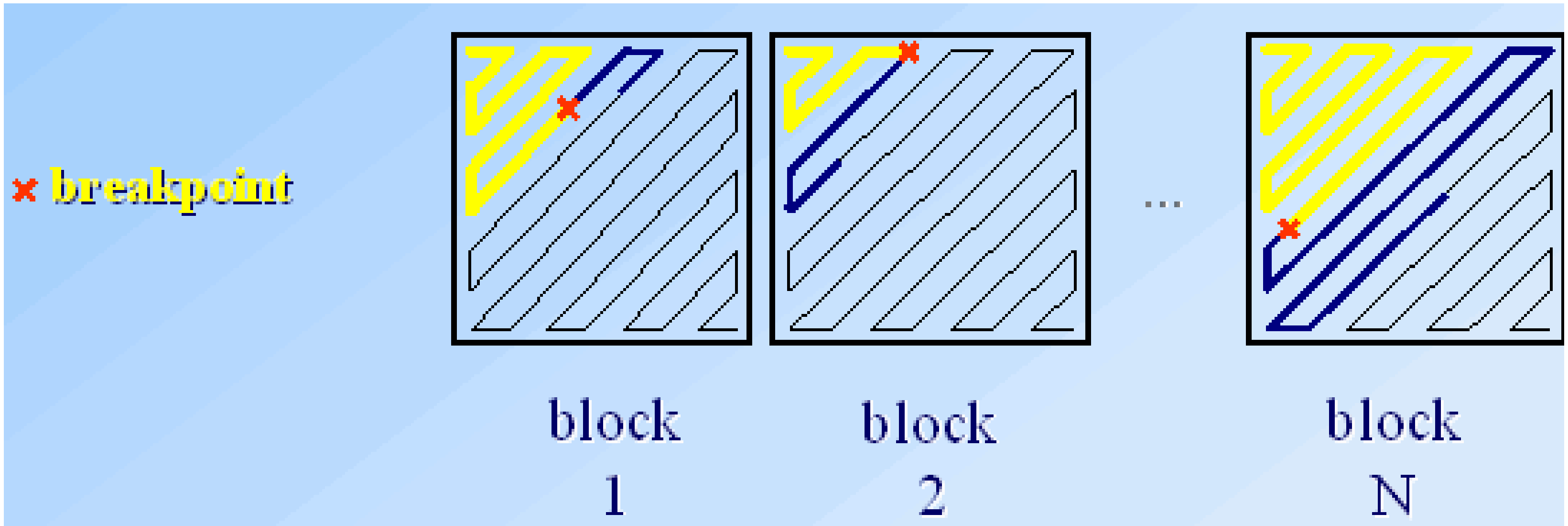
image resolution: CIF ➠ QCIF, with scaling

encoding: different optimal for different qualities

Video-on-demand vs. video conferencing

MPEG Dynamic Rate Shaping (DRS)

- MPEG: picture = > 1 slices = > 1 macroblock = 4 blocks
- DCT run-length encoded in zig-zag fashion
- drop DCT coefficients from MPEG-1 or 2 in compressed domain
- can precisely match desired bandwidth



Application control: Networks with QOS

- QoS negotiation at call set-up time, network guarantees this quality
- QOS guarantees $\equiv$ incumbency protection
- long call durations ➠ network load may change significantly
- "wrong" initial allocation ➠ many rejected calls or low quality
- non-linear utility function

➠ time-limited reservations, changing prices, . . .

Application Control: Networks without Guarantees

- “at-risk” part of differentiated services
- current Internet
- LANs
- variable bandwidth (wireless)
- shared reserved link

➠ **adaptation**

Utility

- non-linear: $\sum$ utility (2 customers, 16 kb/s each) $>$ utility (1 customer, 32 kb/s)
- on-going session more valuable than new session (cp. handover policies)
- utilities differ between network participants
- how to reflect in pricing?

Adaptation

sender: sender ↑↓ bandwidth
(Bolot/Wakeman, Busse/Deffner/Schulzrinne, Sisalem/Schulzrinne, Jacobs/Eleftheriadis)

receiver-driven, sender-aware: sender generates multiple multicast groups with different rates (McCanne/Jacobson/Vetterli)

receiver-driven, translators: transcoding at bandwidth and processing power discontinuities (Amir, Campbell; cp. M-HTML)

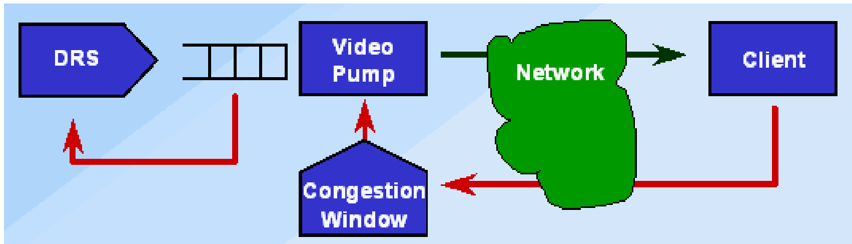
. . . as function of congestion

Adaptation

direct: adjust encoder directly ➠ real-time

buffer occupancy: for media-on-demand (Jacobs/Eleft.):

- modulate buffer output
- buffer occupancy triggers media shaping
- prevent buffer over/underflow



General Goals & Caveats

- sessions still subject to interruption
- what is acceptable congestion?
- adjustment speed $<$ speed of load changes: “in and out of focus”
- convergence with transients (new sources, scene change)
- fairness to other sources
- fairness to other protocols (TCP)
- how to enforce fairness ➠ “penalty box”, one big vendor
- distance/hop unfairness: similar to TCP

Comparison of methods

	sender	receiver	translator
convoy problem	yes	no	no
unicast	yes	no	N/A
encryption	yes	yes	trusted
granularity	fine	3–4	fine
src. coding effort	scales	fixed, high	fixed, low
quality	best	suboptimal	transcoding
bandwidth	small	small (but: DVMRP)	waste until translator
quality	lowest	appropriate	appropriate
media	audio, video	mostly video	all
network state	1	L	1

Measurement Methods

- end-to-end:

  Packet loss: Bolot, BDS, LDA, RLM, Jacobs, Vicisano/Crowcroft, . . .

  Delay jitter: BDS (RTP)

  Packet delay: Sakatani (ICMP echo @ 500 ms)

  CSMA/CD collisions: Sakatani

  Throughput: ThinStreams (TCP Vegas), LDA (packet pair)

- with network help:

  Source quench messages: Sakatani

  congestion bit: frame relay

  rate control: ABR

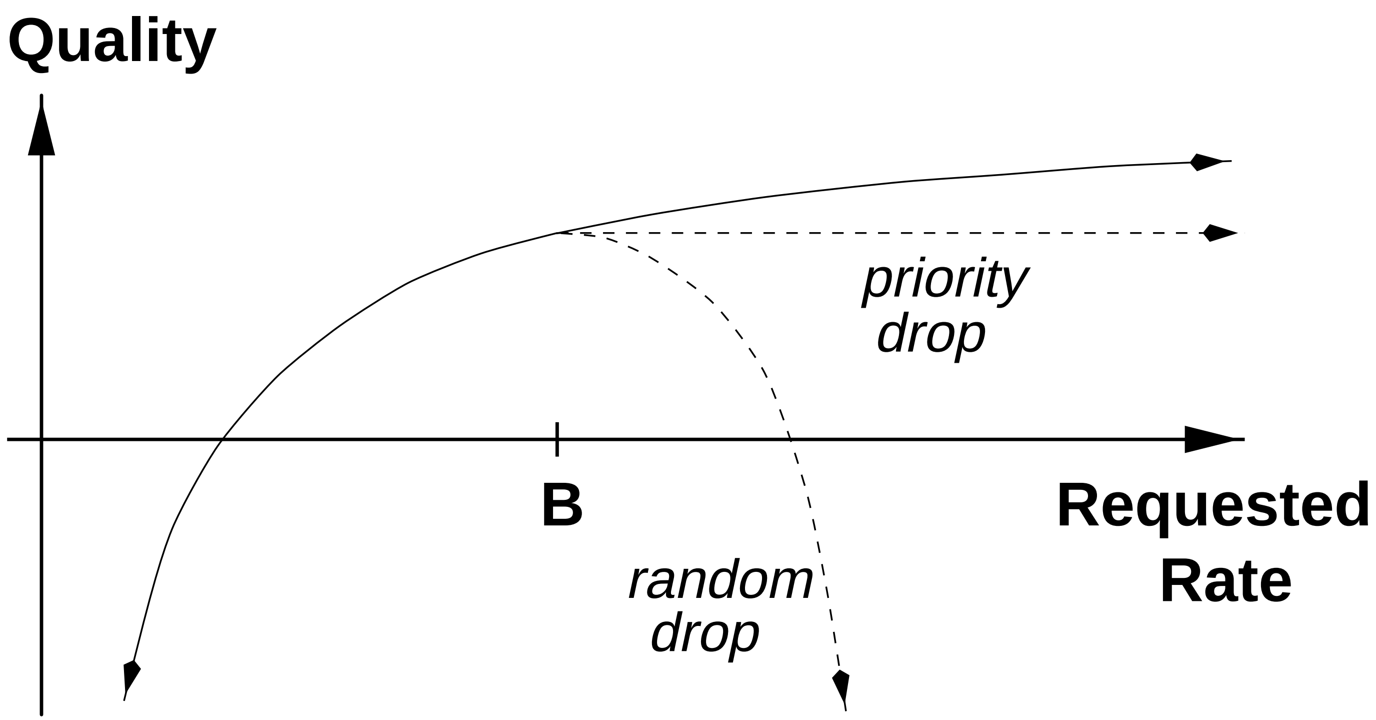
  buffer occupancy: Kanakia/Mishra/Reibman

Measurement Methods

- polling
- periodic unsolicited feedback
  - unicast
  - multicast

all vs. just poorly connected

Interaction of Adaptation with Priority



- beyond bottleneck B, only enhancement dropped ➠ constant quality
- ➠ no unique maximum quality
- ➠ encouragement to go beyond B

Buffer-based TCP-like Adjustment

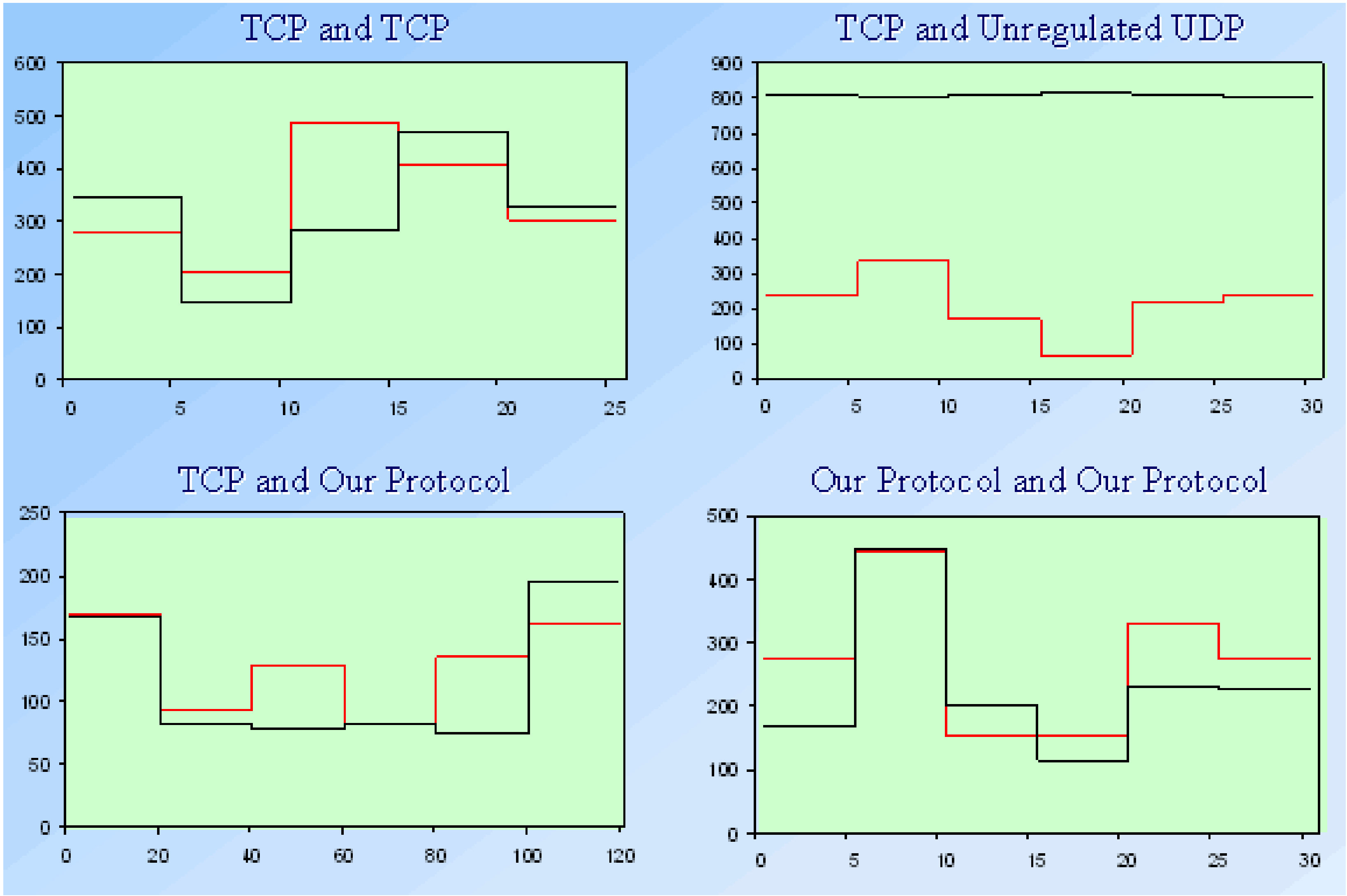
- measure queue occupancy over $t > 1$ second (I, P, B!)
- fill at λ, drain into network at μ
- $B_{i+1} = B_i \lambda_i t - \mu_i t$
- set $\lambda_{i+1} = \mu_i = \lambda_i + \frac{B_i - B_{i+1}}{t}$
- $\Delta = (B_i - B_{i+1})/t$
- modulate around desired occupancy B_d (5 s):

$$\alpha_i = \begin{cases} B_i/B_d & \Delta \leq 0 \\ 2 - B_i/B_d & \text{otherwise} \end{cases}$$

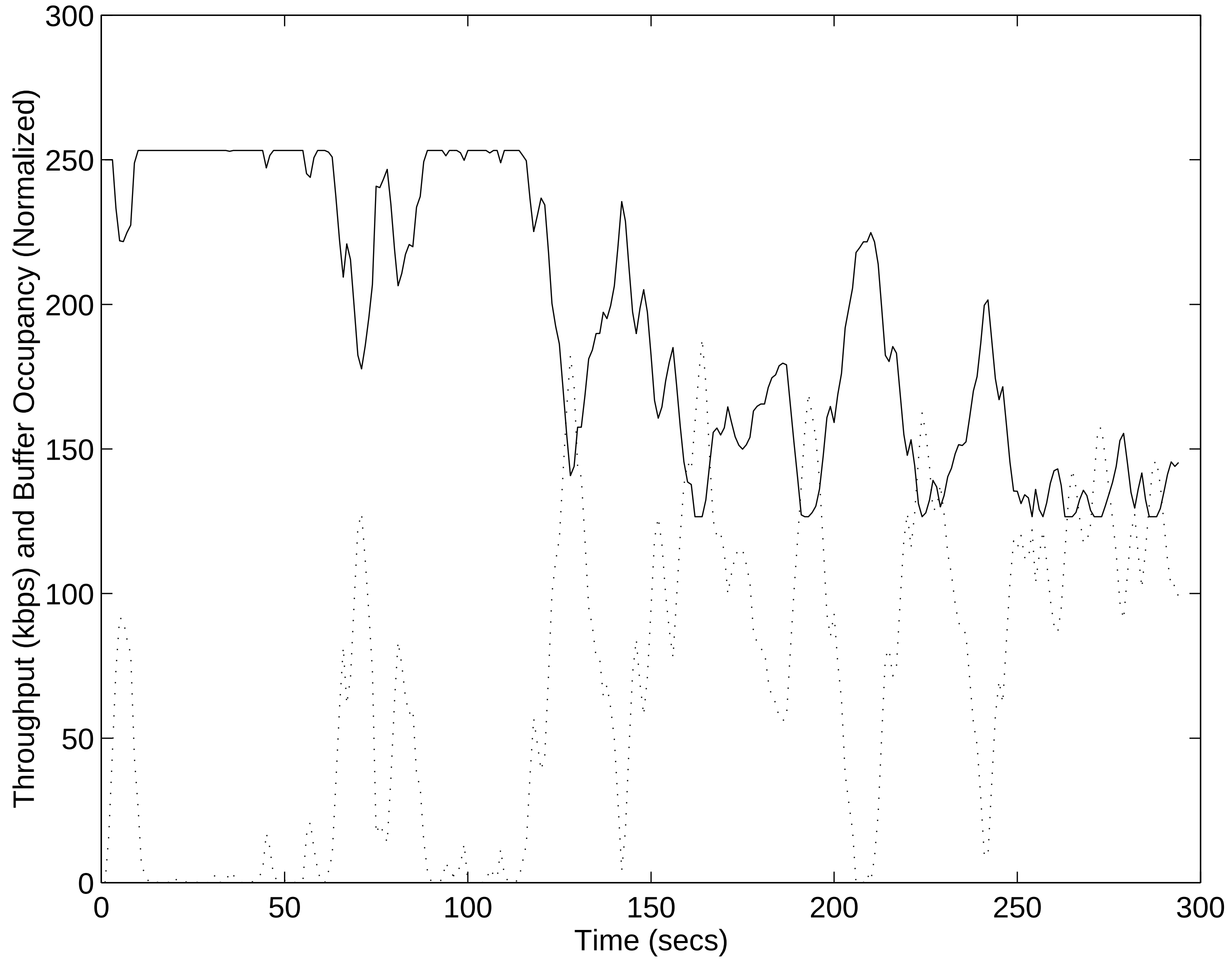
- $\lambda_{i+1} = \lambda_i + \beta_i \alpha_i \Delta$
- smoothing: $0.1 \geq \beta_i \leq 1$: variance of buffer occupancy
- ➠ small rate decrease if empty

TCP-based Congestion Control

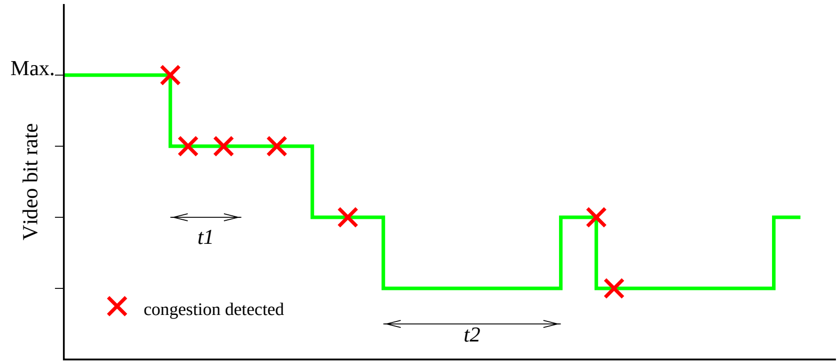
- TCP; retransmit only if client buffer $>$ RTT
- μ measured: send until window exhausted, paced by ACKs
- problems: multicast, buffer delay (typical: 5 seconds)


TCP and TCP
600
500
400
300
200
100
0
0
5
10
15
20
25
TCP and Unregulated UDP
900
800
700
600
500
400
300
200
100
0
0
5
10
15
20
25
30
TCP and Our Protocol
250
200
150
100
50
0
0
20
40
60
80
100
120
Our Protocol and Our Protocol
500
400
300
200
100
0
0
5
10
15
20
25
30

TCP-based Congestion Control



Timer-based Adjustment (Sakatani)



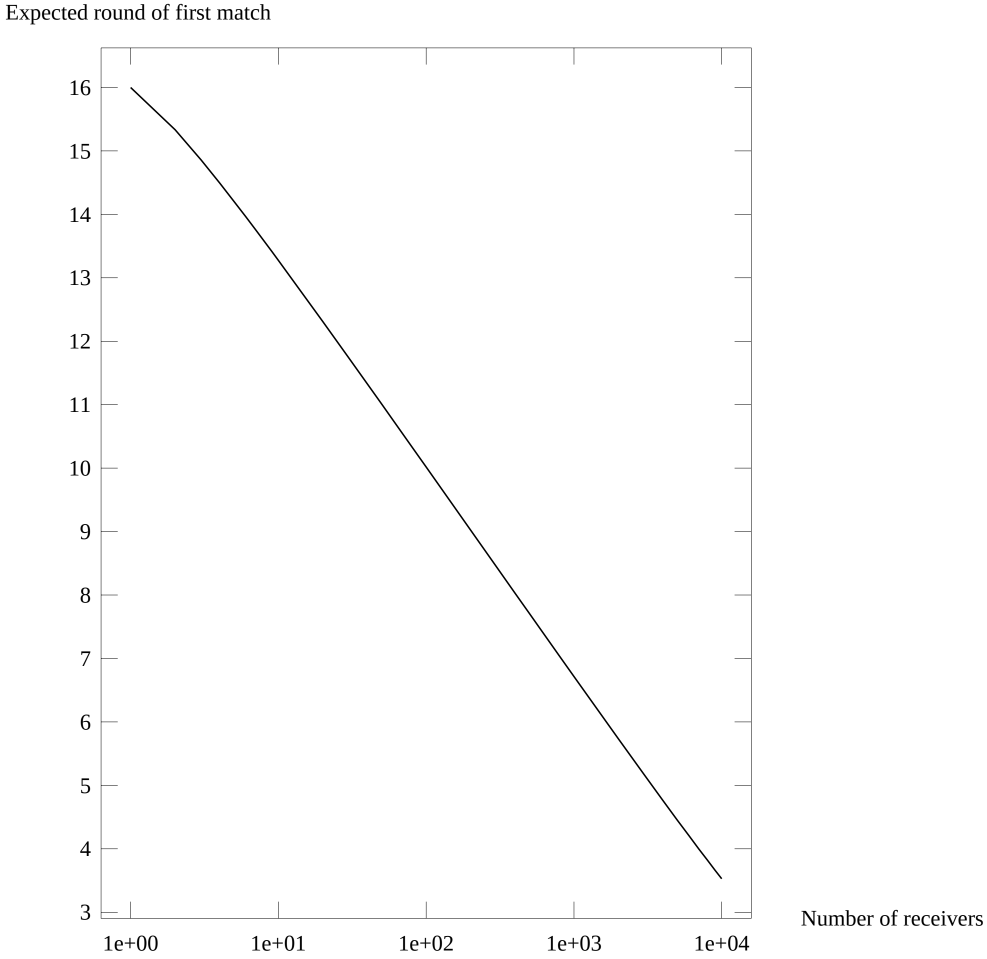
- $t_2 > t_1 > t_0 (t_0 : 0.5s)$
- if ≥ 1 congestion (delay $>$ 100 ms) within t_1 (1 s): $\downarrow$
- if no congestion within t_2 (2 s): $\uparrow$
- if congestion after increase, decrease immediately
- threshold arbitrary (100 ms)

Sender-Based: Estimation via Polling

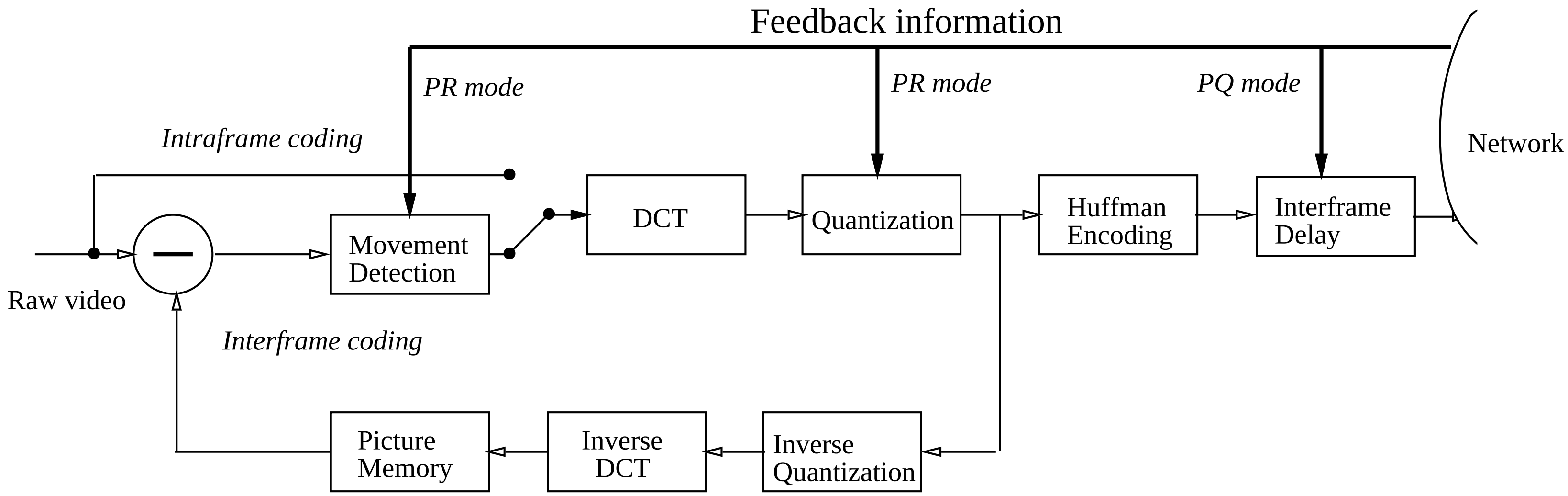
Bolot, Turletti, Wakeman (1994)

- $<$ 10 members ➠ NACK for losses ➠ doesn't scale
- find worst-positioned receiver in one *epoch*
- receiver, source generate random 16-bit key
- source sends key and # of significant bits (16, 15, . . .)
- if match, unicast response to source
- if no response within 2 RTT, reduce significant bits by one
- STATE mode: send current state (unloaded, loaded, congested); respond only if worse
- stop epoch if CONGESTED

Sender-Based: Estimation

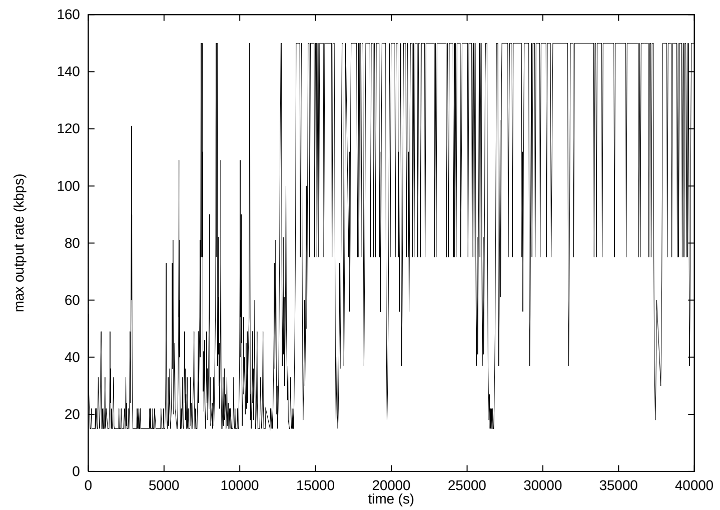


Sender-Based: Video Codec

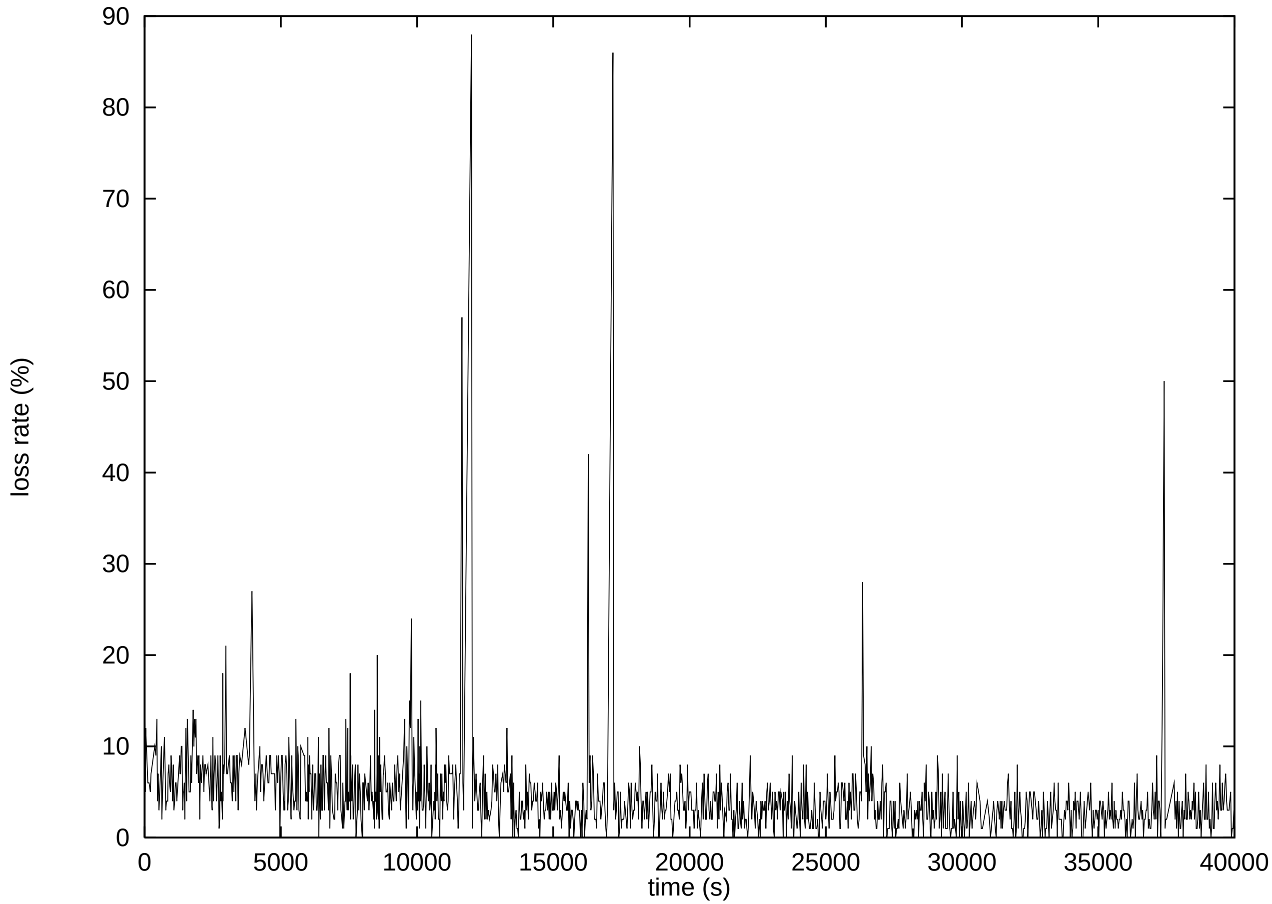


Sender-Based Polling: Adjustment Mechanism

- if congested fraction > threshold ➠ rate *= 1/2 if > min. rate
- if all underloaded ➠ rate += 10 kb/s
- CONGESTED threshold: 5%



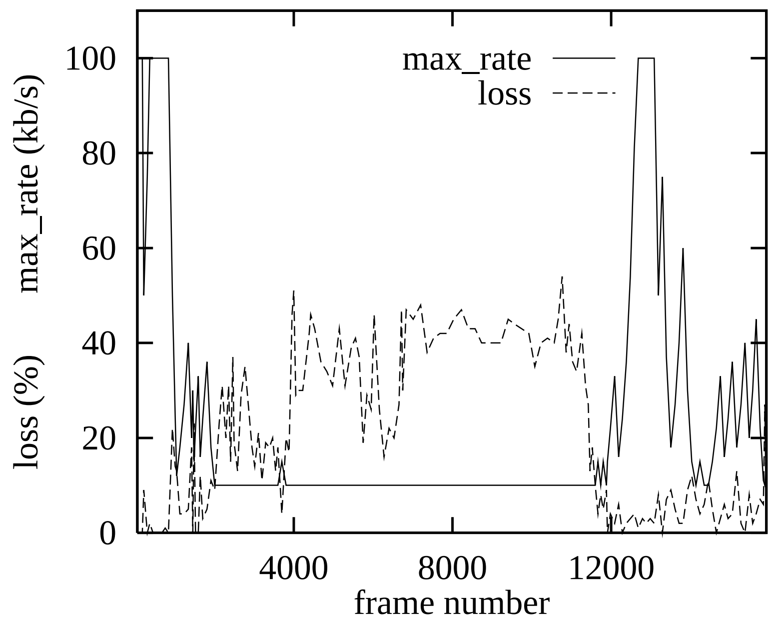
Sender-Based Polling: Adjustment Mechanism



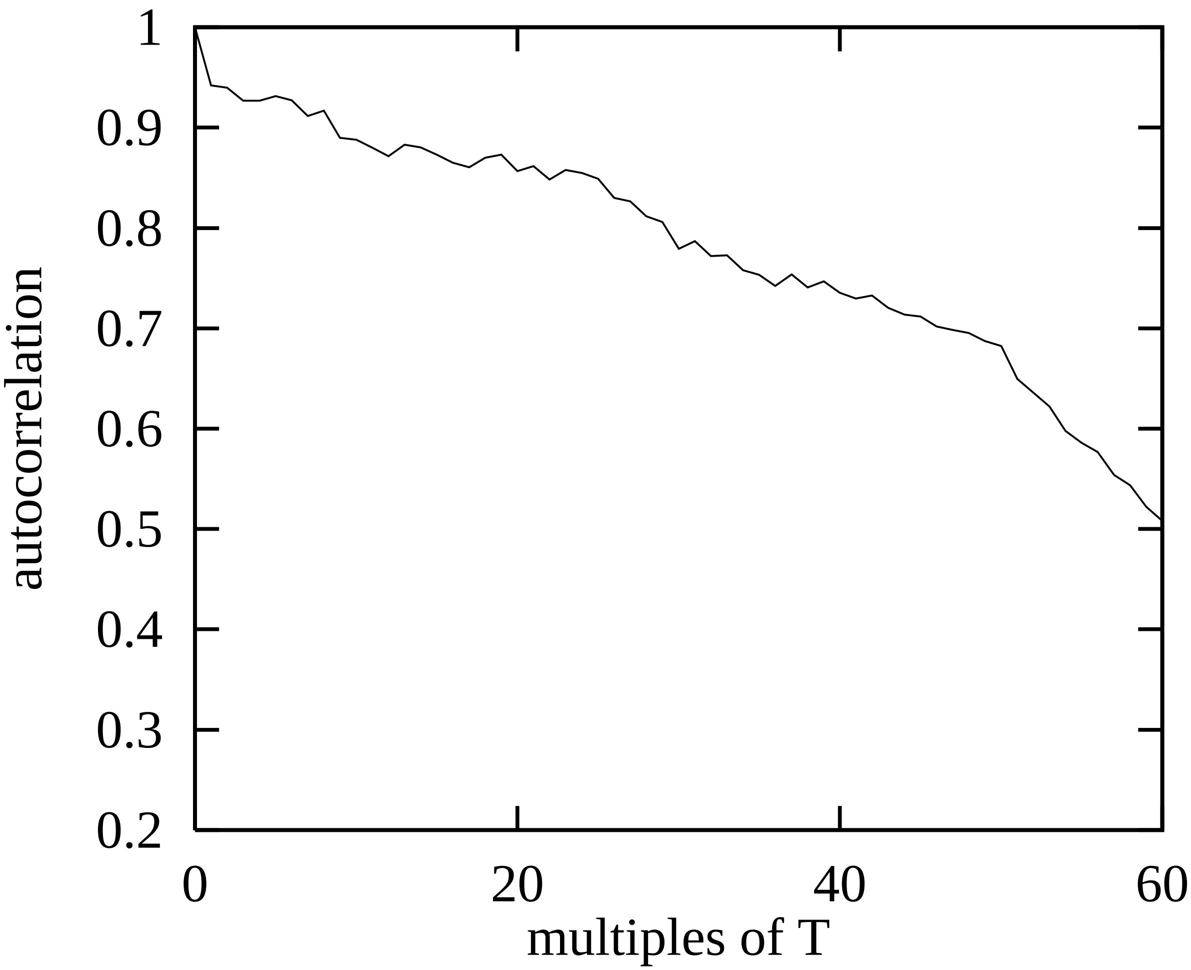
Sender-Based Adaptation: Periodic Feedback

Bolot, Turletti (1994):

- RTP loss over 100 packets or 2 minutes, send feedback
- use median loss rate across multicast group

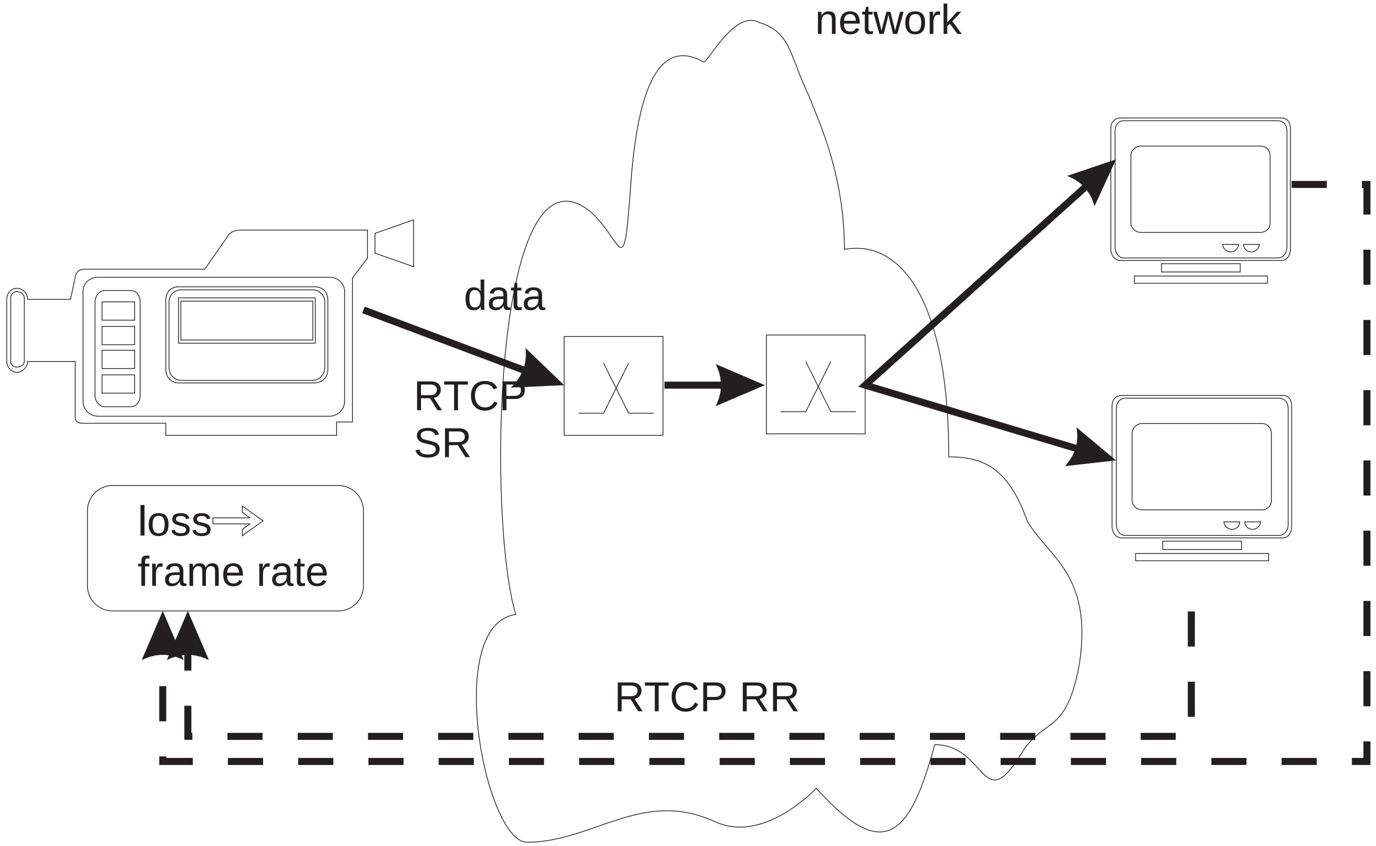


Sender-Based Adaptation: Loss Rate Correlation



$T = 15$ s

End-to-End RTP Feedback Control Mechanism

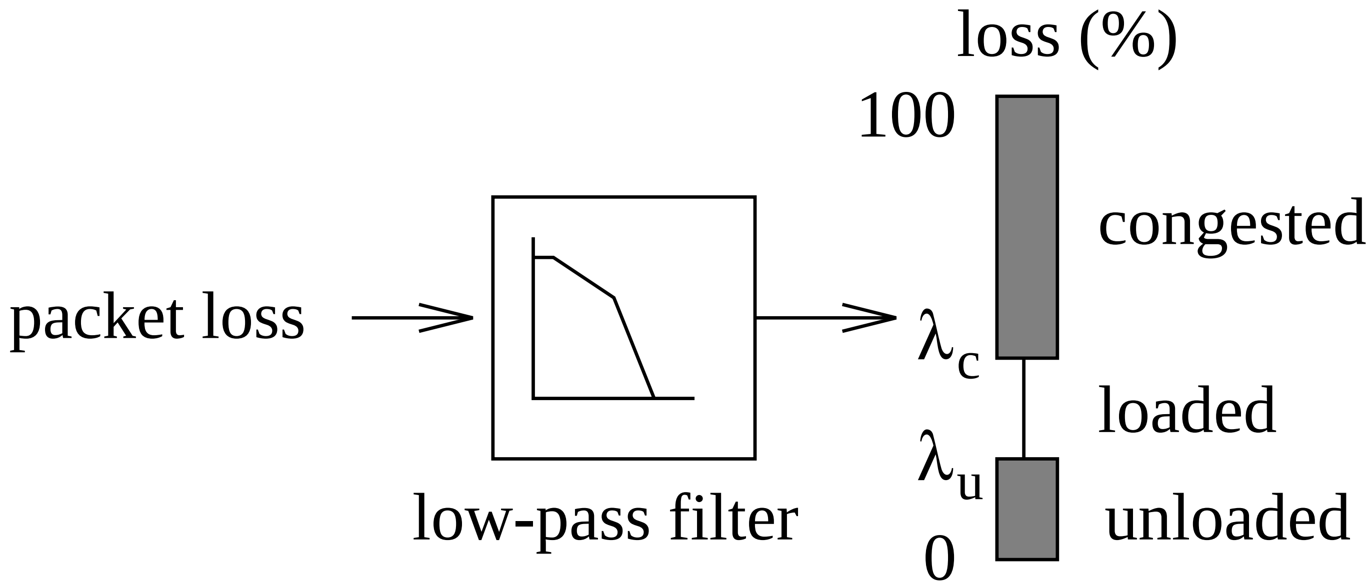


video application bandwidth is based on network feedback:

low losses ➠ slow bandwidth increase ➠ higher framerate

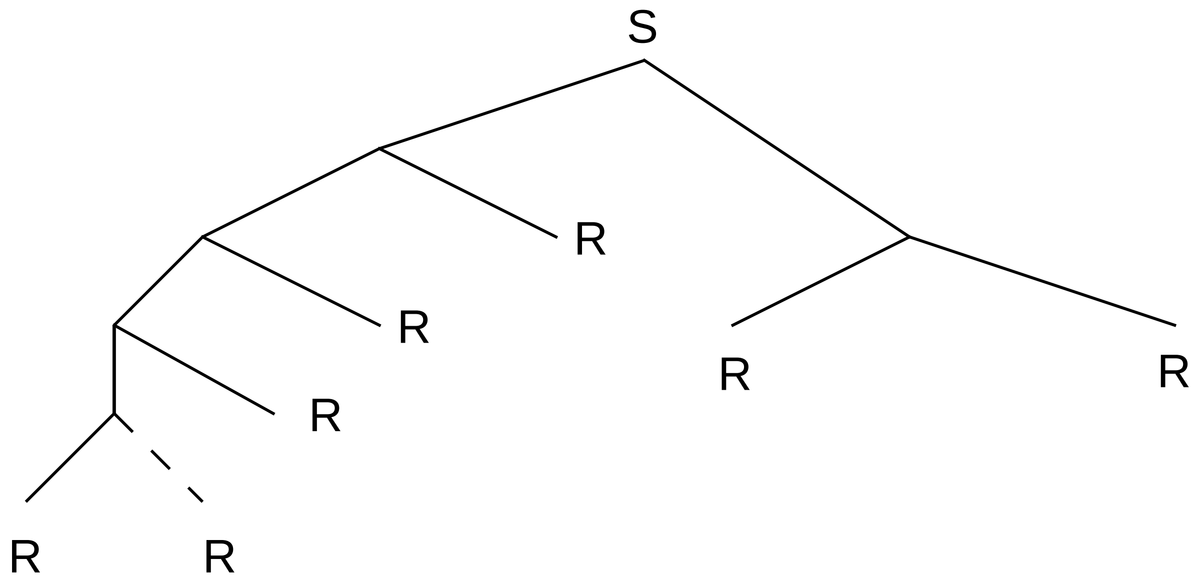
high losses ➠ bandwidth decrease ➠ lower framerate

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



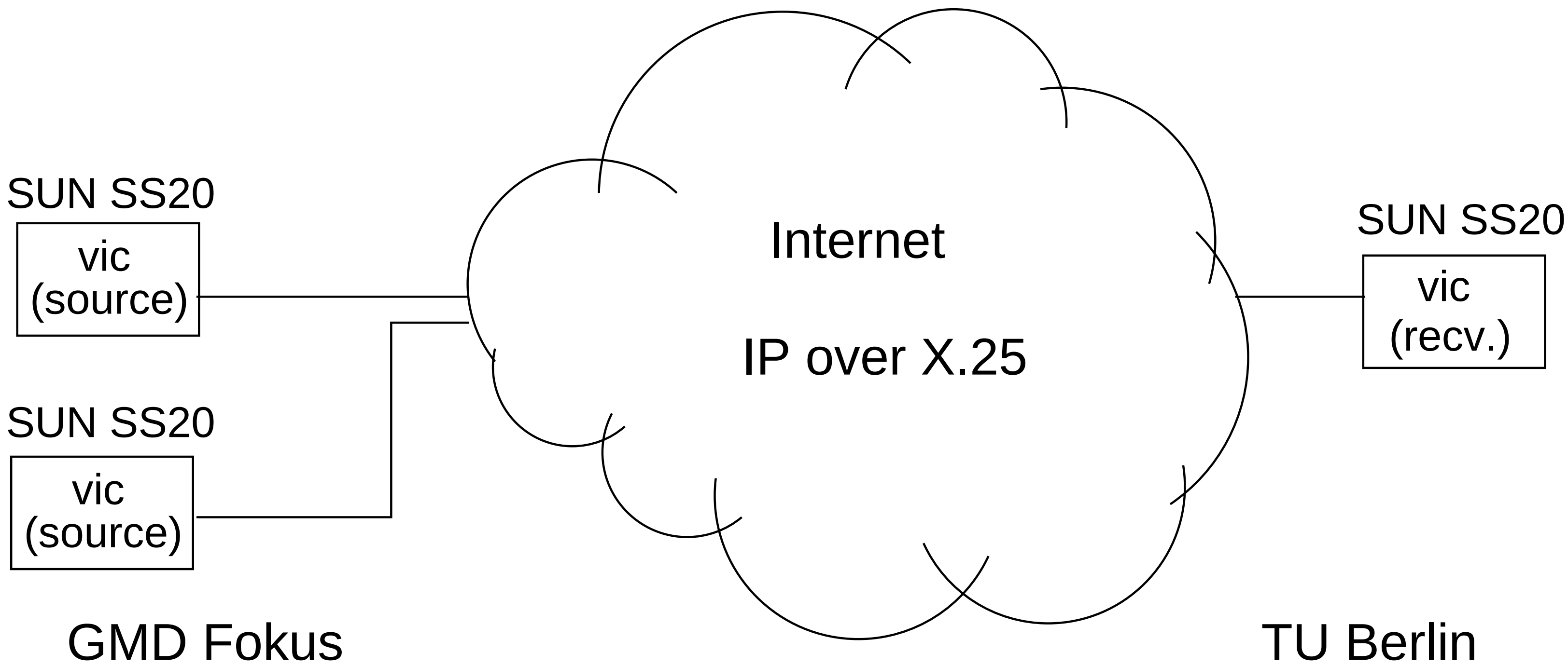
algorithm 1: adjust according to the worst-positioned receiver

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

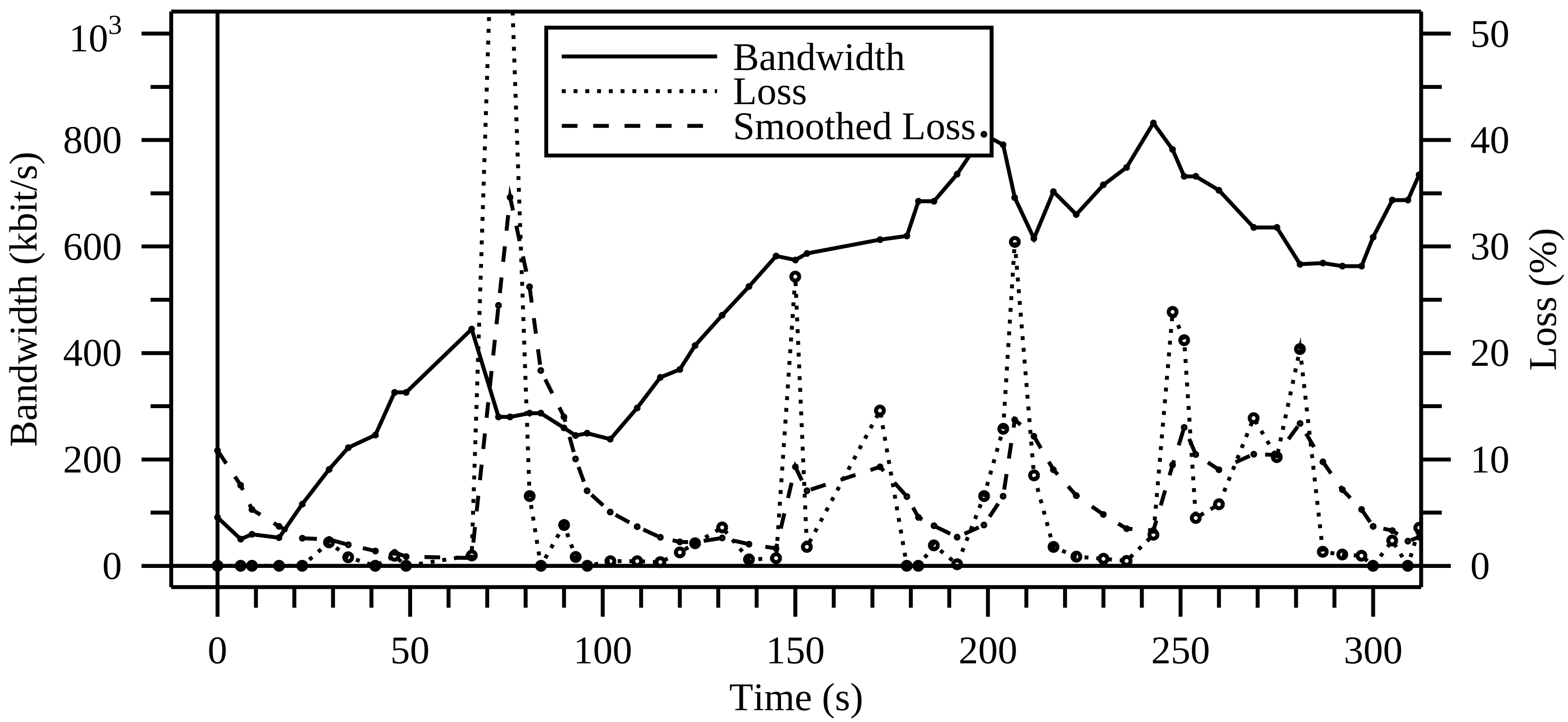
other solutions: video gateways, layered encodings

Internet scenario

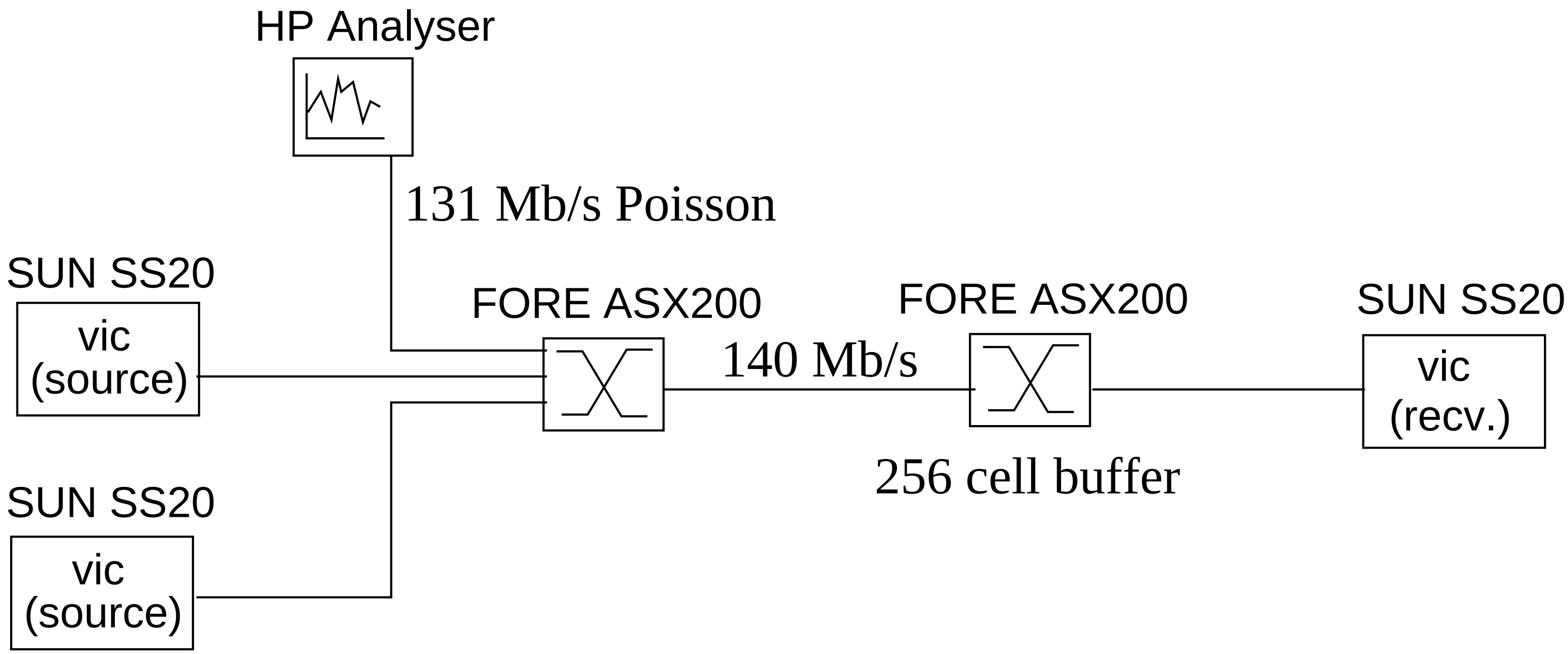
- Measurements made on the 2 Mbit/s X.25 link between GMD Fokus and TU Berlin (5 hops distance)
- deadzone between 5% and 10%



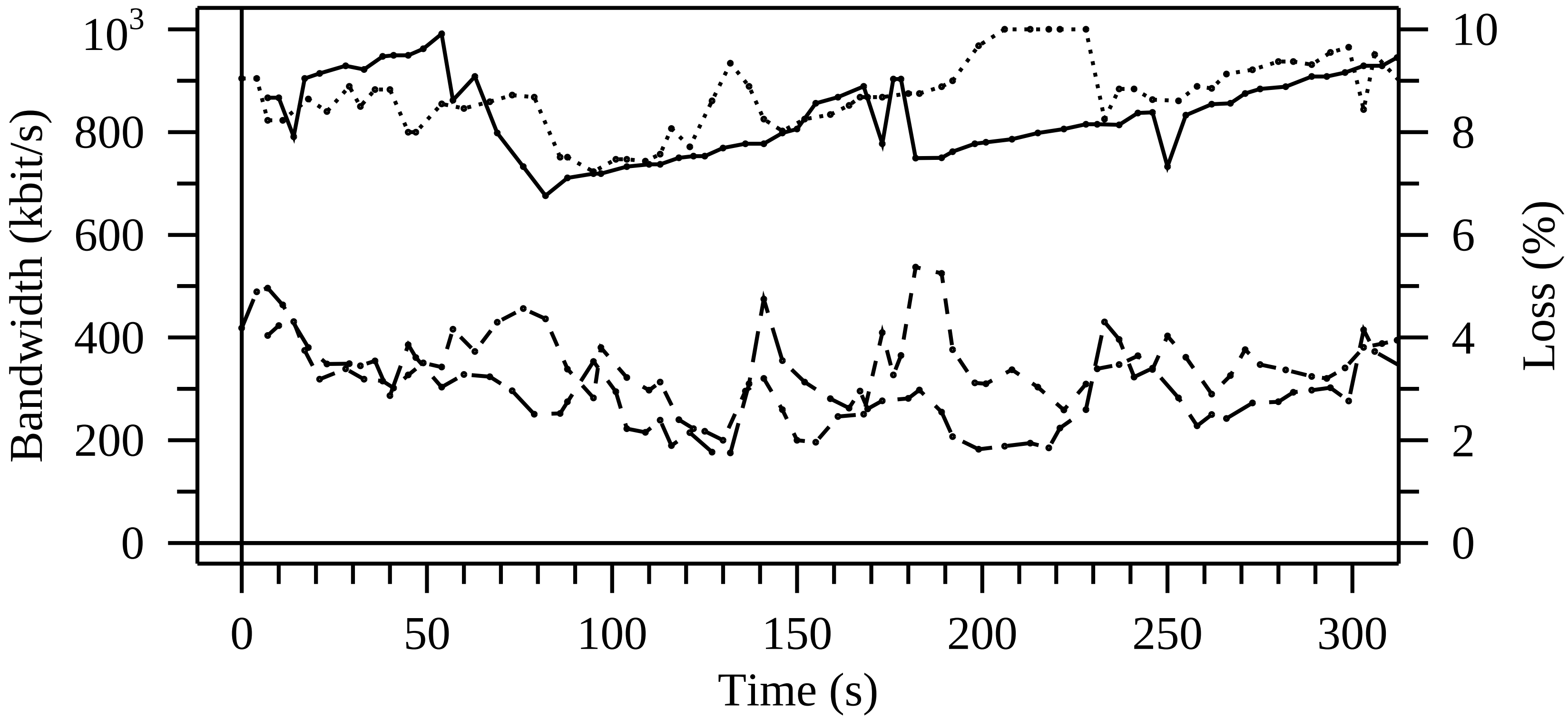
Internet measurement



ATM scenario



ATM measurements



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

TCP-Friendly Adaptation

Sisalem/Schulzrinne, 1998

- …but be usable with multicast
- attempt to use (Floyd, Ott) with RTT τ, loss ℓ ($< 16\%$):

$$r_{\text{TCP}} = \frac{1.22M}{\tau\sqrt{\ell}}$$

- based on averages, rather than measurements
- attempt: use RTCP loss, delay reports for ℓ ➠ oscillates, low throughput

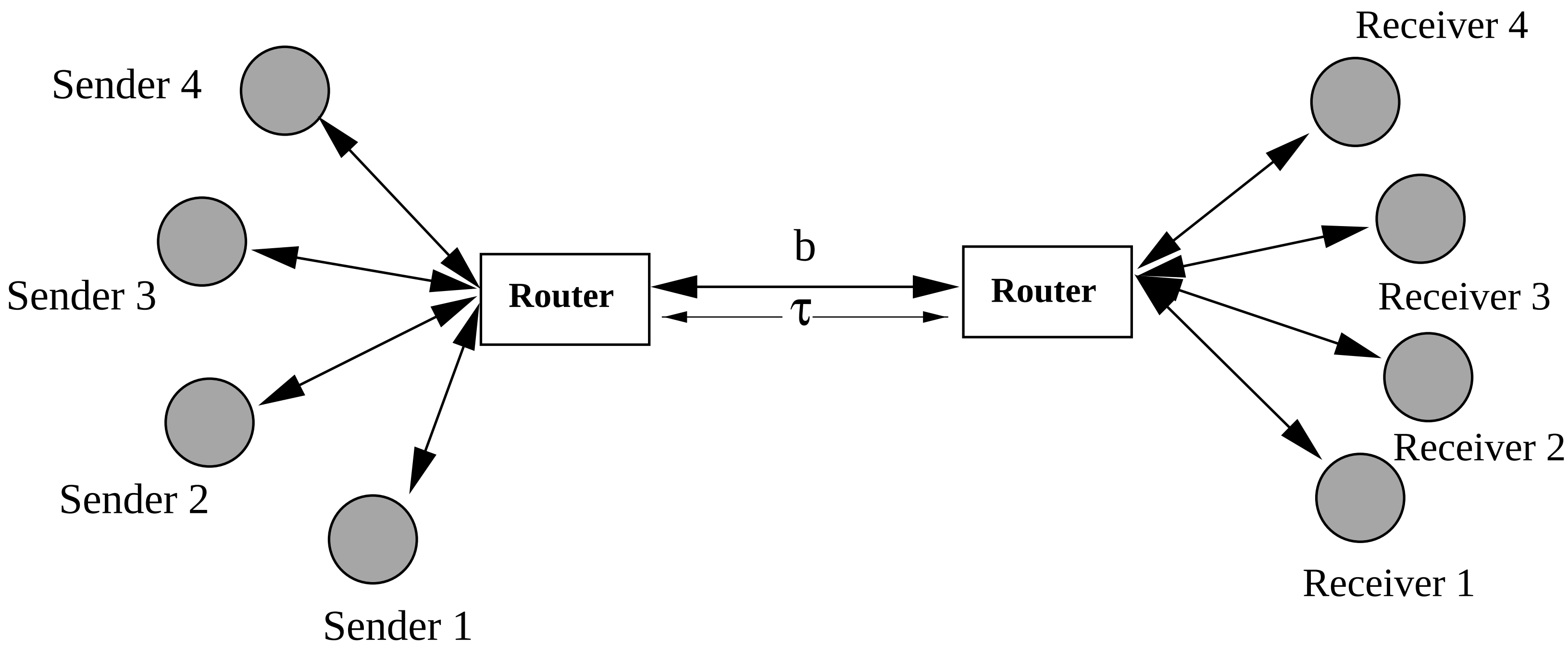
TCP-Friendly Adaptation: LDA

- packet pair bandwidth estimation: $b =$ size/gap
- use video packets within frame
- BPROBE: cluster estimates, average biggest cluster (cp. LBNL pathchar)
- additive increase (AIR), multiplicative decrease of rate r
- no loss: AIR$* = B_f$, $B_f = 1 - r/b$ ➠ favor small
- limit to rate increase of TCP connection per RTCP interval T
- adjustment based on receiver i, with empirical reduction factor R_f (3):

$$r_i = \begin{cases} r + \mathrm{AIR}_i & \text{no loss} \\ r(1 - (\ell R_f) & \text{loss} \end{cases}$$

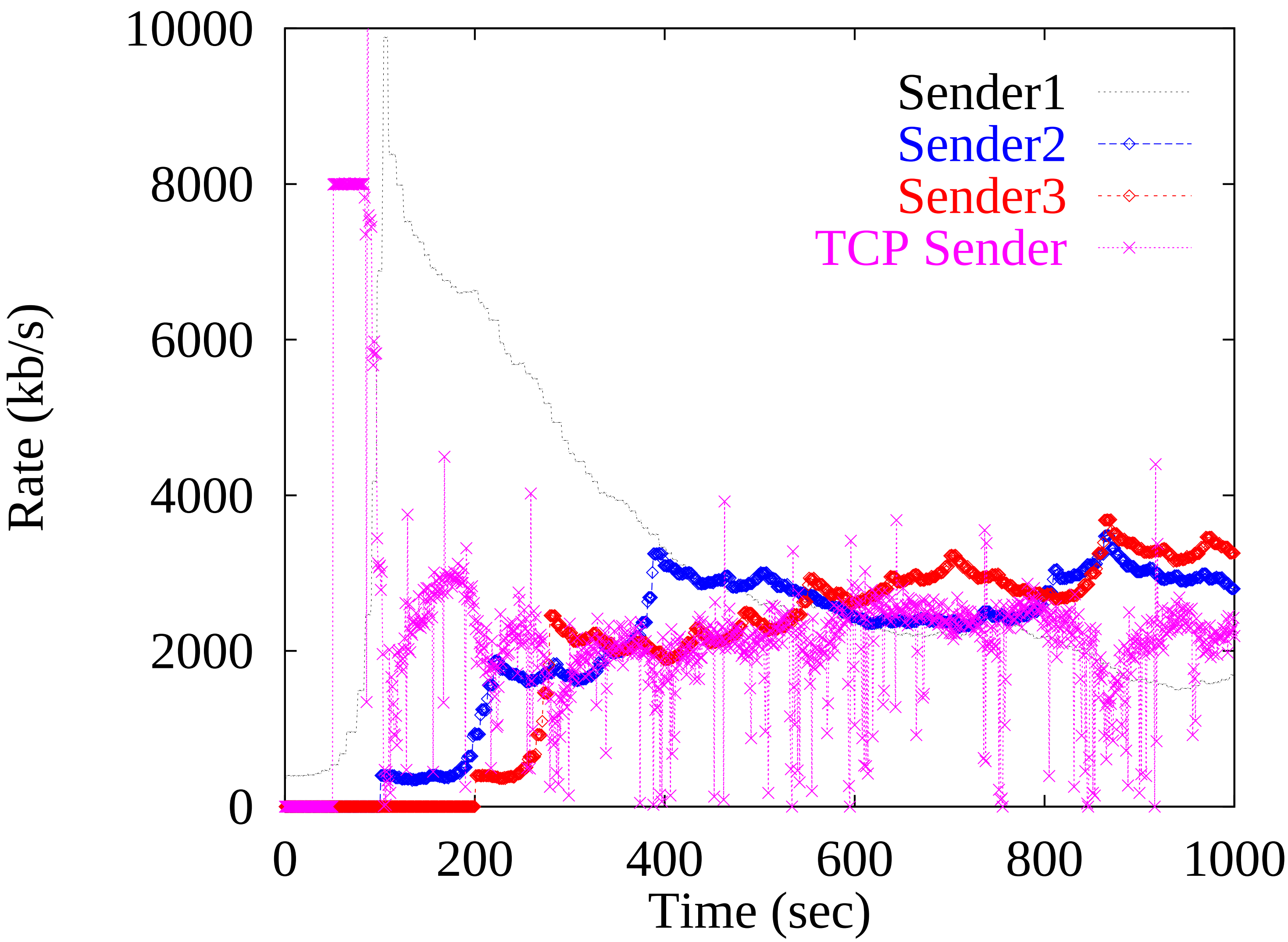
- at fixed-interval adaptation (5 s) points, compute r_{min}; if loss, AIR $\leftarrow$ 10 kb/s

LDA: Measurements



RED routers

LDA: Measurements



average utilization: 95%

Adaptation with Network Support

Kanakia *et al.*

Regulate buffer occupancy at *bottleneck* to x^*:

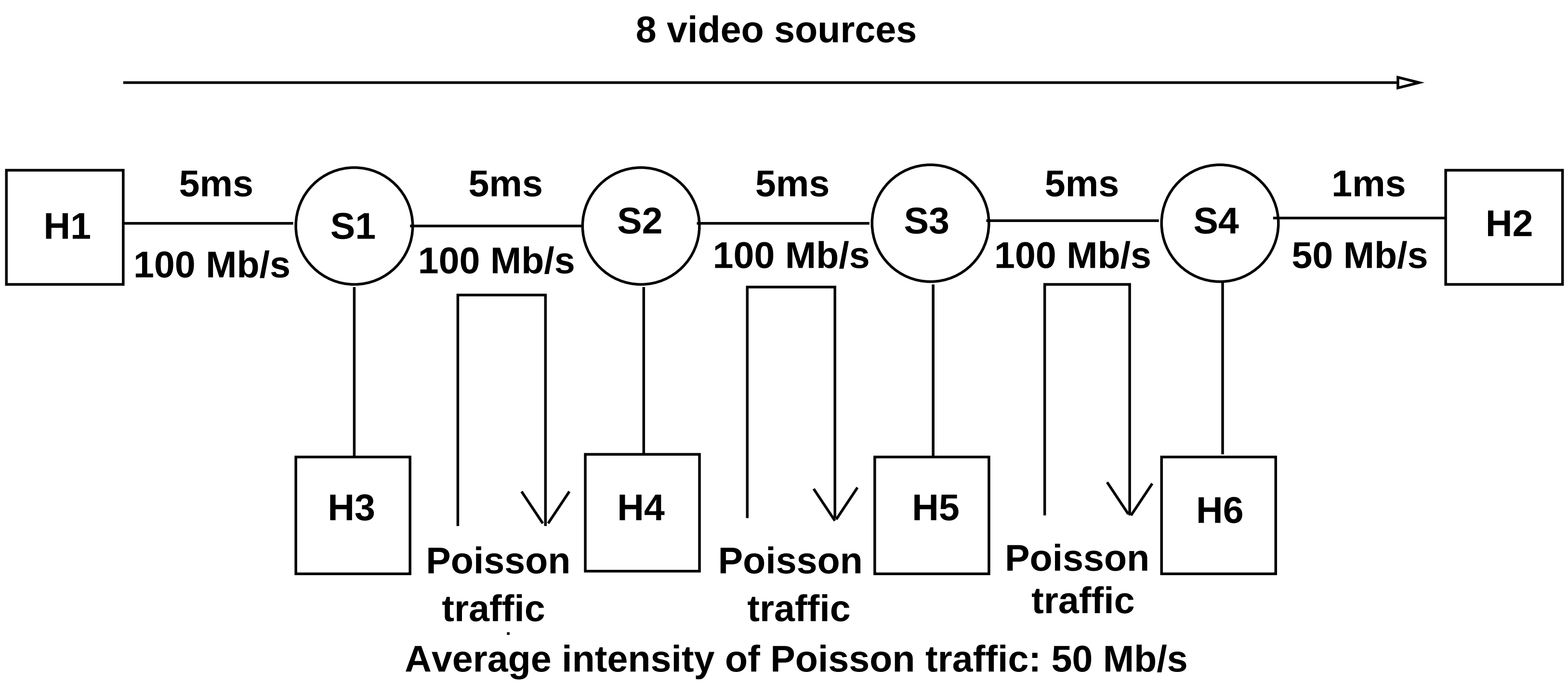
$$\lambda_n = \begin{cases} \hat{\mu}_n + \frac{x^* - \hat{x}_n}{\alpha F} & x_{n-k} > 0 \\ \lambda_{n-1} + \delta & \text{otherwise} \end{cases}$$

with frame rate F, service rate μ

- service rate through *adaptive* first-order filter
- feedback every 4 ms (!)
- separate I, P, B rates for MPEG
- adjust Q factor between 3 and 20

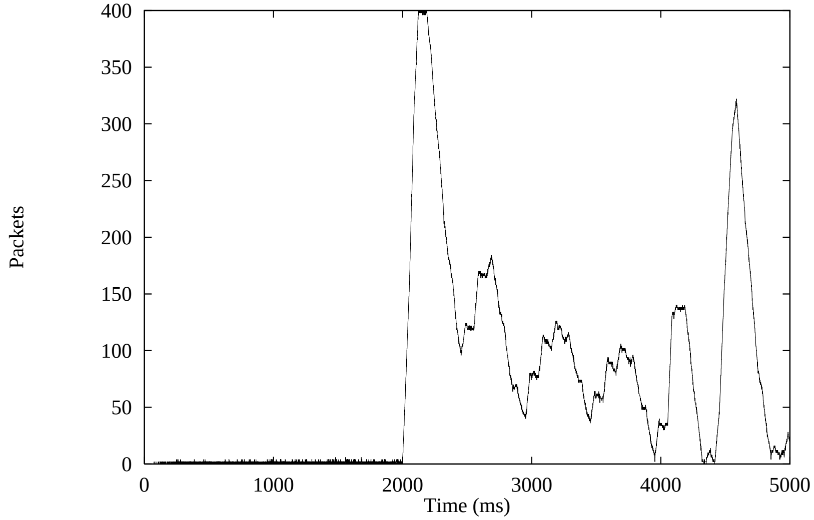
Adaptation with Network Support

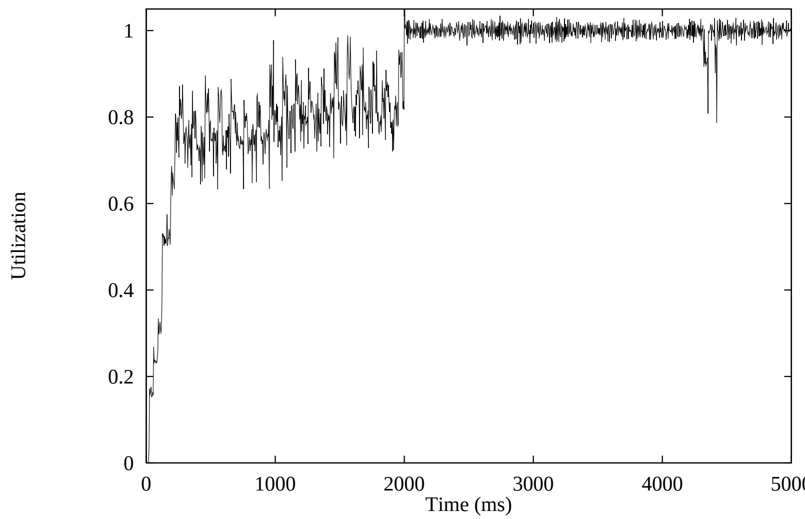
S4-H2 Bandwidth reduced to 30 Mb/s at $t = 2000$ ms



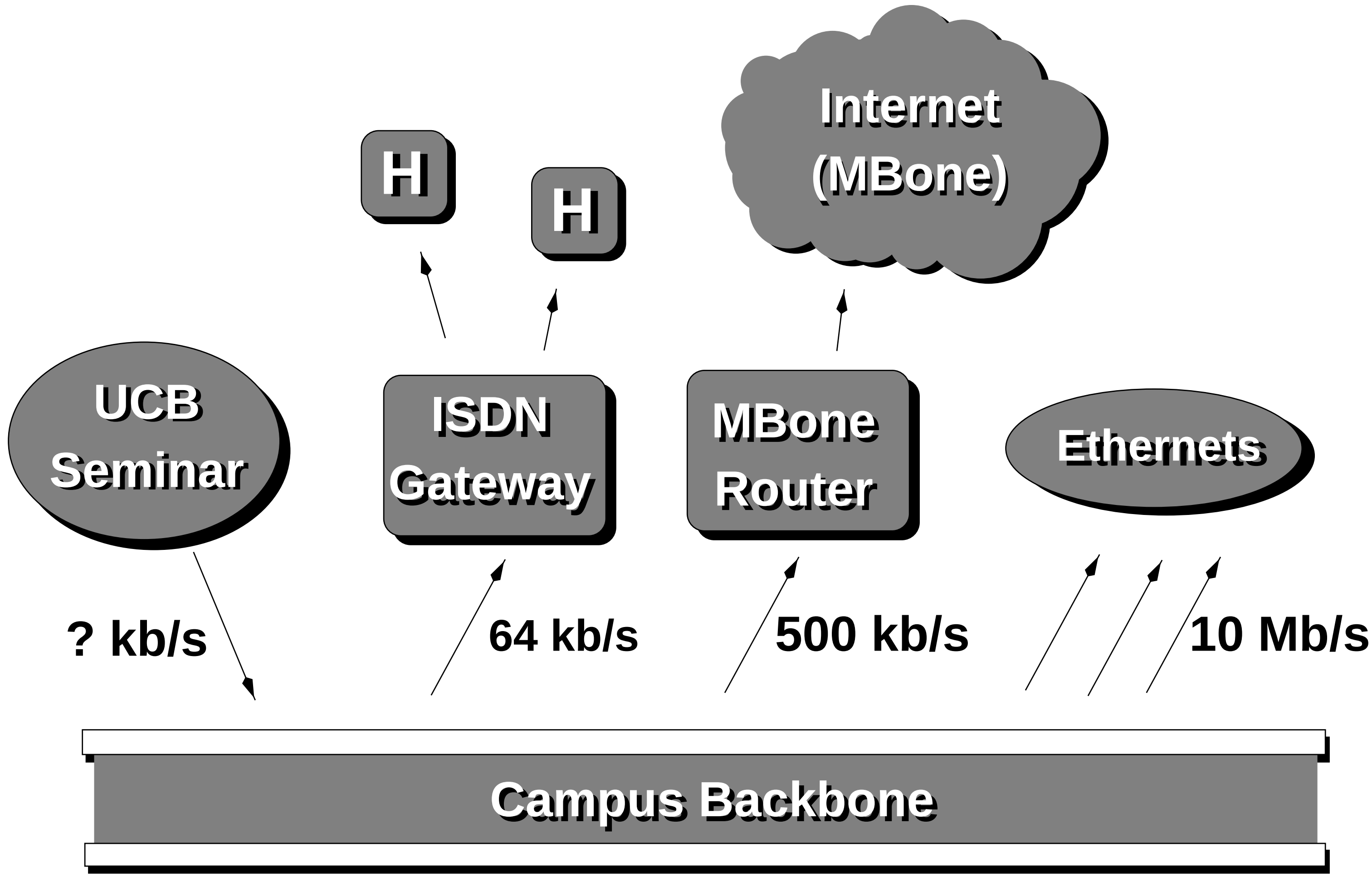
Adaptation with Network Support

Buffer occupancy and link utilization:





Receiver-Based Adaptation



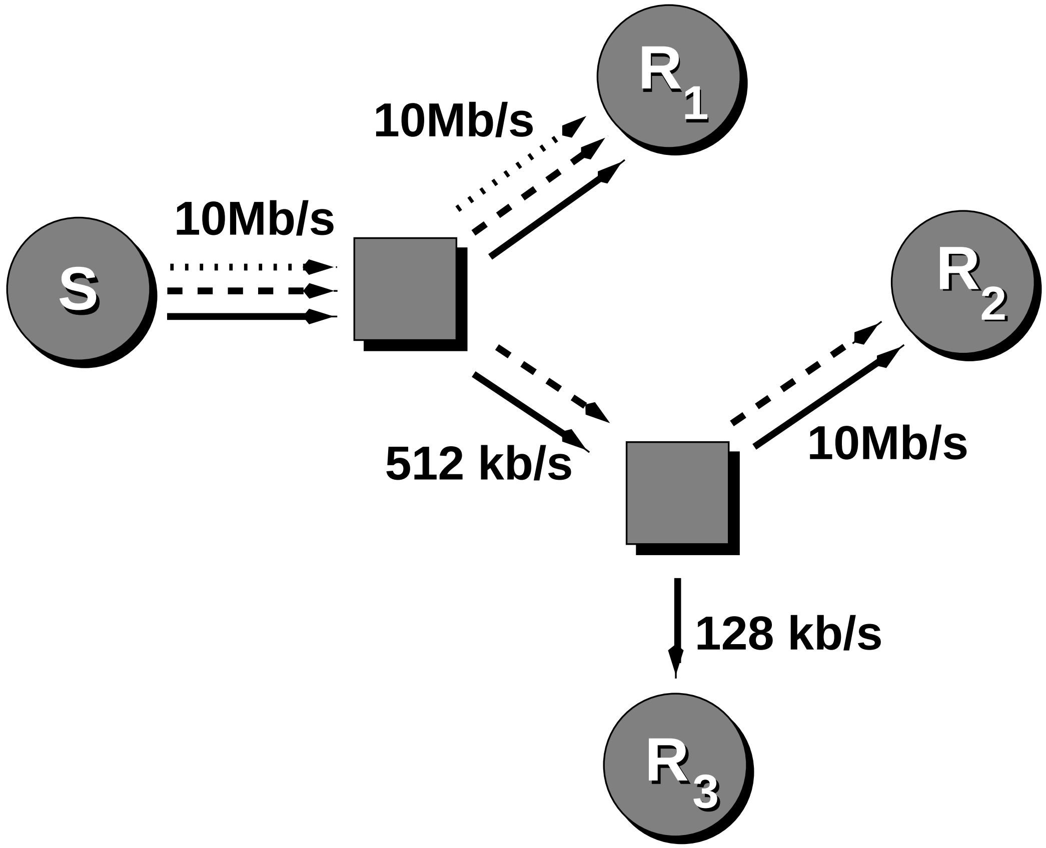
Recap: IP Multicast

- groups identified by class-D IP address
- receivers can subscribe without knowledge of/knowing sender
- host can send to group without being a member
- IGMP at local network signals leaves and joins
- DVMRP, PIM, CBT, . . . for routing in Internet
- Mbone as experimental overlay network of *tunnels*

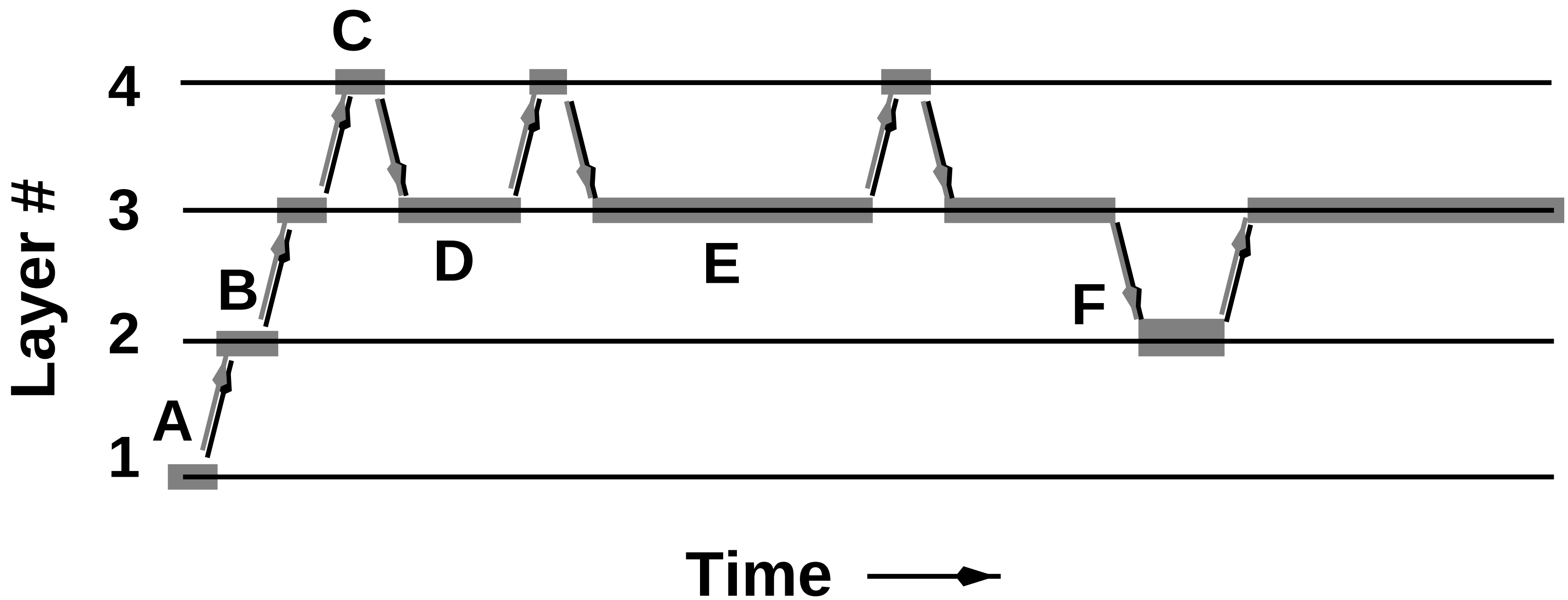
Layered Multicast

- layered media in L groups (*session*):
  - MPEG frame types – but: I frame $\gg$ P, B
  - JPEG parameters
- **cumulative**: always subscribe to groups $1 \ldots n < L$
- simulcast: subscribe to *one* of $1 \ldots L$ (audio!)
- drop top layers on congestion, add when capacity
- *join experiments*: join next-higher group and observe loss over decision time
- shared learning: announce intent to do join experiment
- rely on source-based pruning

Receiver-Driven Layered Multicast (RLM)



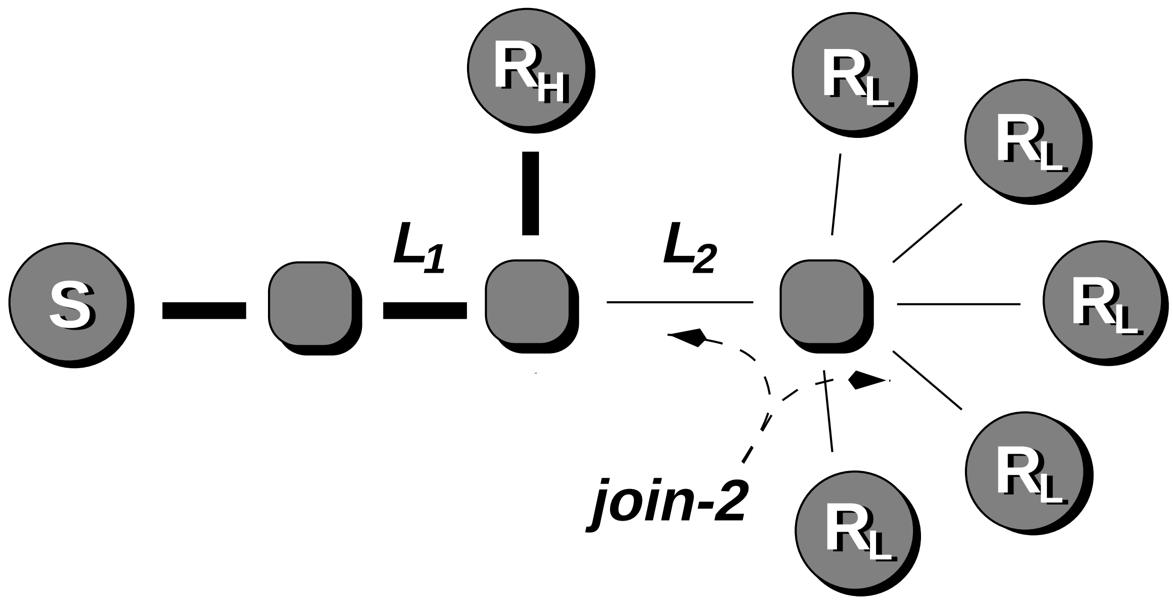
RLM



On congestion, increase join timer multiplicatively

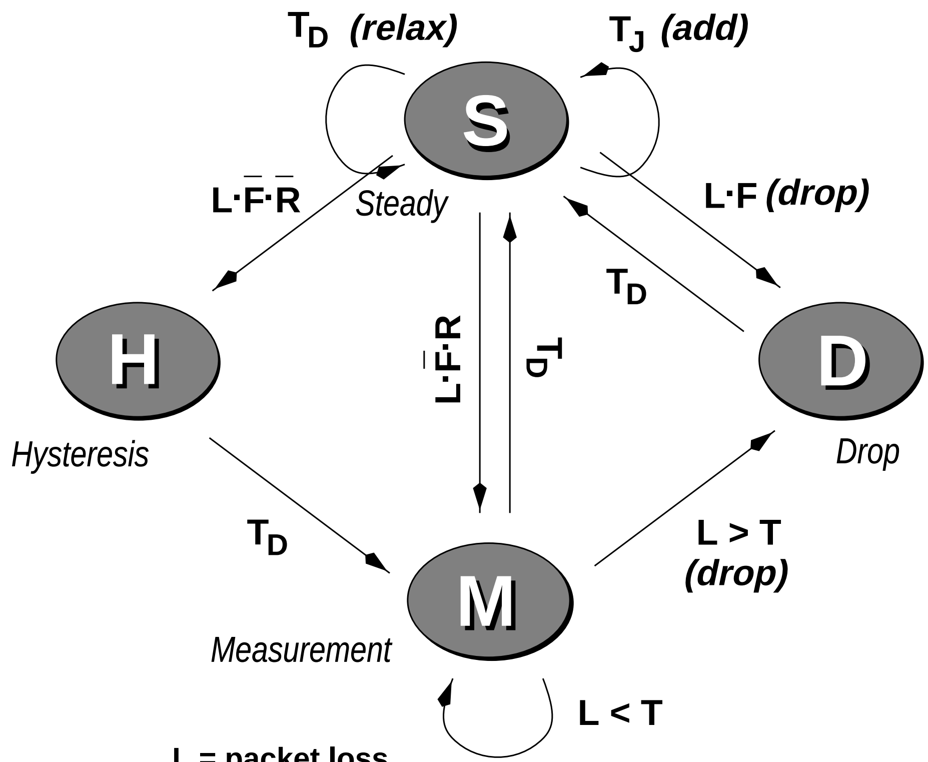
if no congestion within *detection time* ➠ success

RLM: Shared Learning



- membership ↑ ➠ congestion due to join experiment ↑
- join experiment interfere ➠ measurement noise
- multicast a join-experiment notification to *all*
- if others detect congestion, scale back join timer
- suppress new experiments with higher level during on-going ones

RLM: State Machine



L = packet loss

F = our layer is highest of recently added layers

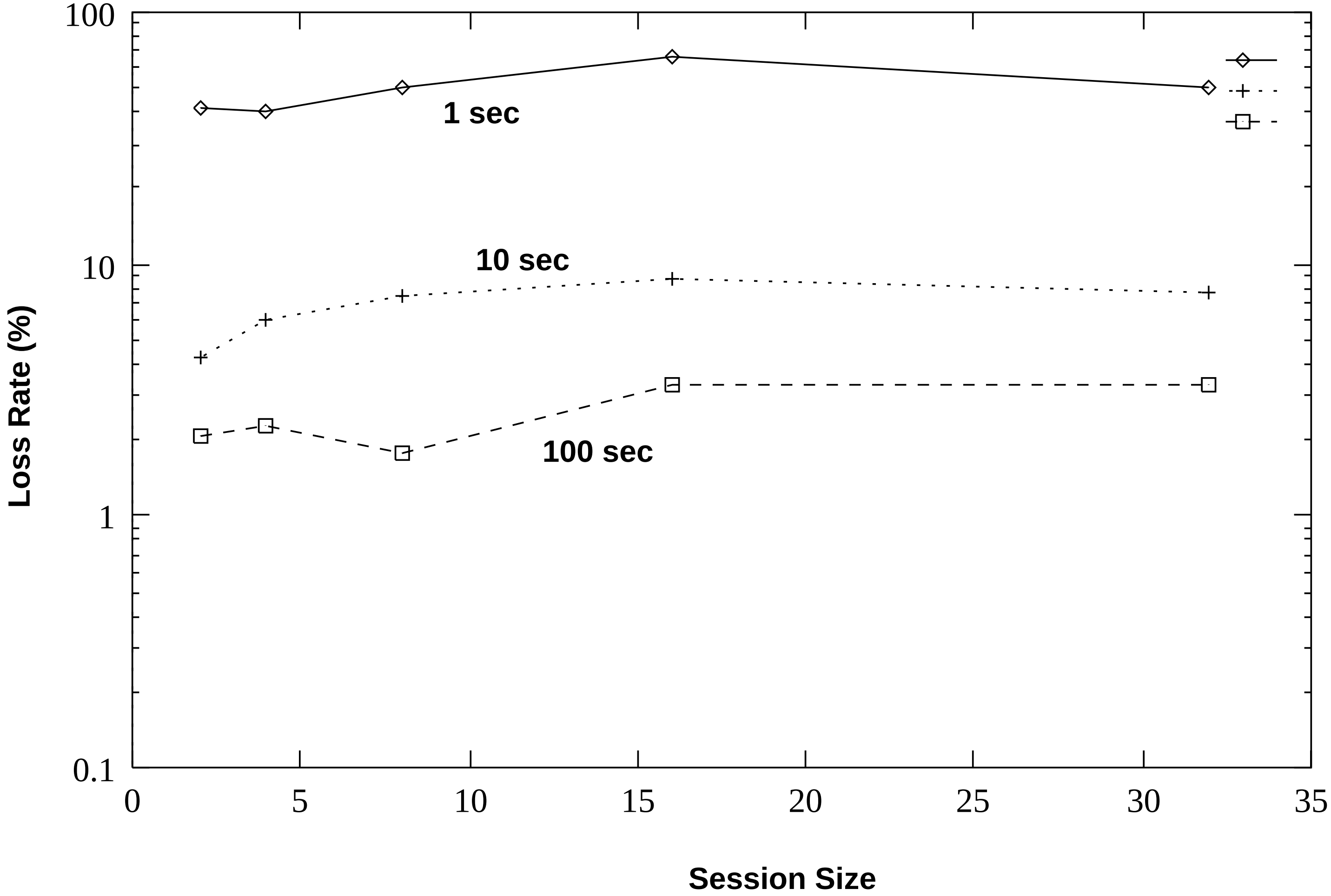
R = our layer was recently added

L > T = loss rate exceeds theshold

join timer T_J, detection timer T_D
relax: decrease join-timer

RLM: Performance

Worst-case loss over varying windows for heterogeneous environment:



Thin Streams

Wu/Sharma/Smith

- split video layer (*thick* layer) into several fixed-bandwidth *thin streams*
- for join experiment that buffers can absorb: T and layer rate R: $B \leq R \cdot T$
- assume $B = 4kB$ ➠ $R = 16$ kb/s
- expected $-$ measured throughput

Thin Streams

- G groups joined, N bytes received in interval I
- actual bw: $A = \alpha A + N(1 - \alpha)/I$
- expected: $E = \alpha E + GR(1 - \alpha)$
- leave threshold = $GRe^{(1-G)/8}$ ➠ more groups, leave earlier
- join threshold = $GR\beta$
- hold-off time $\propto G$
- if $E - A >$ leave threshold ➠ leave
- if time since last join $>$ hold-off $\wedge$ $E - A <$ join threshold ➠ join

Thin Streams

- independent experiments ➠ overloading ↑ with group size
- ➠ synchronize join experiments *within session* with clock in base layer
- *different* sessions must not synchronize (loss!) ➠ random start times
- unclear: reduction in packet loss
- enhancements: wait longer if several failed join experiments

Bandwidth Scaling = f(Receiver Interest)

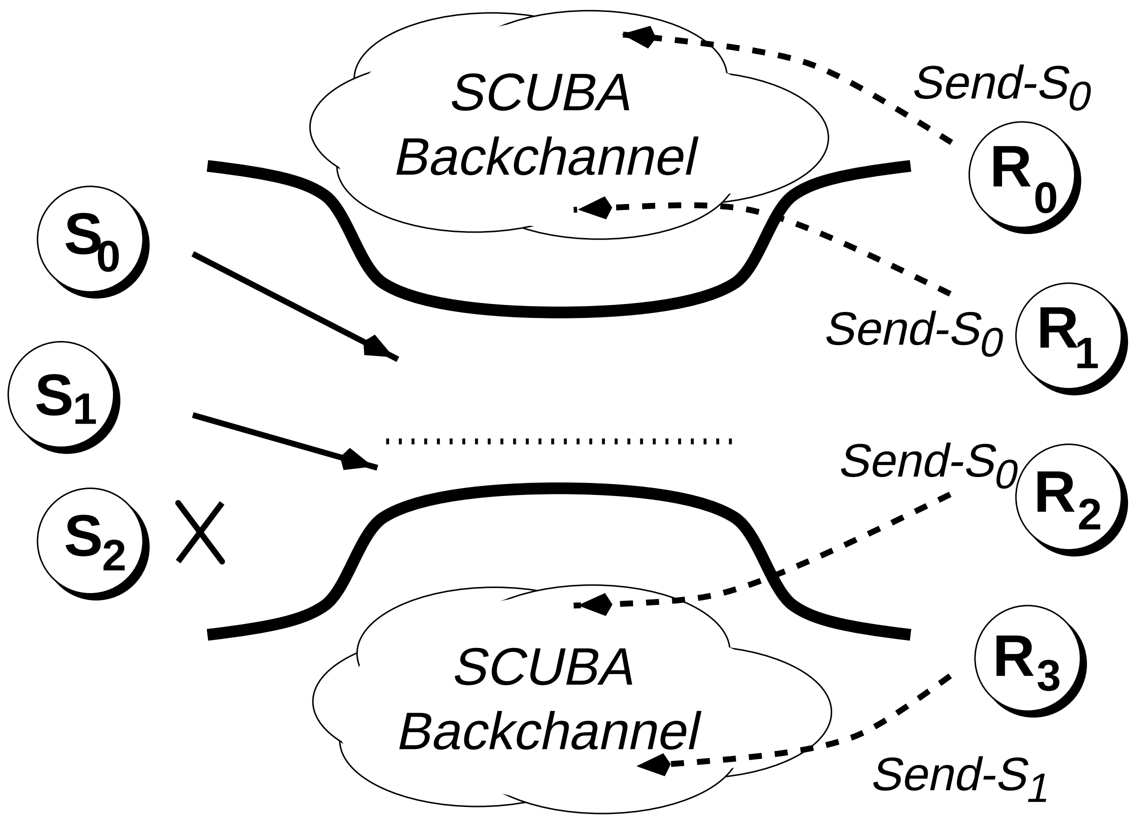
- so far: single sources, compete against same + other groups
- SCUBA (Amir et al.): weigh traffic allocation across senders according to receiver interest
- "exit poll"
- M receivers i sends *source weight report* $w_{j,i}$ for N sources j: de-iconize ➠ weight $\uparrow$
- source computes *average source weight*:

$$w_k = \frac{1}{M} \sum_{i=0}^{M-1} \frac{w_{k,i}}{\sum_{j=0}^{N-1}}$$

Weights across sources sum to 1:

$$w_k = \frac{1}{M} \sum_{i=0}^{M-1} w_{k,i}$$

Dynamic Bandwidth Allocation

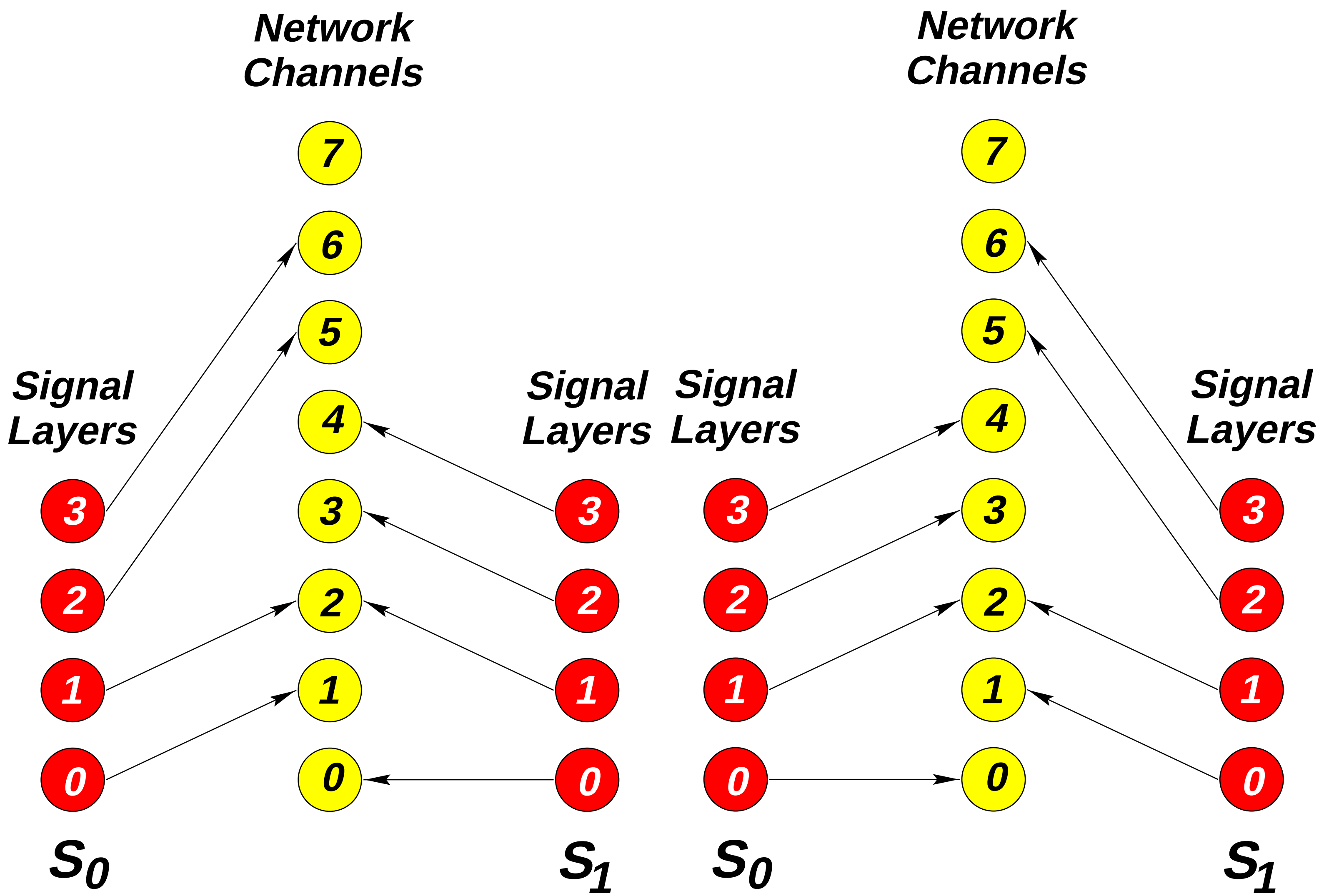


soft-state, idempotent
separate protocol, 25 kb/s for 10 s switch time ➠ event-driven

SCUBA: Flat Delivery

- uniform aggregate *session bandwidth* B (static, dynamic)
- $B_k = w_k B$
- heuristic: 95% for sources with non-zero weight, rest share 5

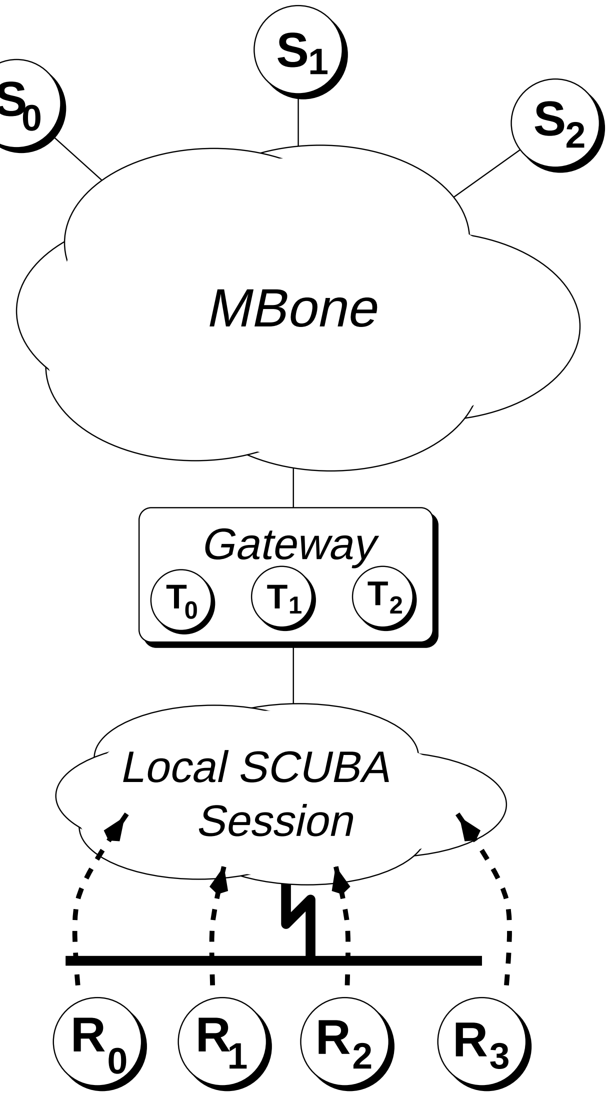
Prioritized Layered Delivery



SCUBA: Layered Delivery

- RLM: signal layer $k \rightarrow$ network channel k
- here: map ≥ 1 signal layers into C network channel with bw L_i
- source limit: $w_k B$, $\sum C_i = B$
- $\mu_{k,n}$: network channel assigned to signal n at source j
- $w_k \geq w_j \Rightarrow \mu_{k,n} \leq \mu_{j,n}$

SCUBA: Media Gateway Control



Error Correction for Real-Time Media

Motivation

- lossy (1% to 20%+), typical 2-5%
- most losses single packet:
- delay requirement: one-way $<$ 400 ms, goal: 150 ms
- common *average* one-way delay (*IEEE Network*, Jan. 1998): 200 ms (Chicago – San Diego)
- ➠ can't use retransmission for *interactive* multimedia
- each data unit: 20 to 80 ms ➠ audible
- reliable multicast (cp. Kurose tutorial)

Internet characteristics

Losses seen by a single voice flow (10.6 kb/s with 30 ms frames . . . 32 kb/s with 20 ms frames): (J. Rosenberg)

- Columbia U. to Germany

Fri	2/28.97	afternoon	2.3%
Fri	2/28/97	morning	7.2%
Thurs	2/27/97	afternoon	2.2%
Thurs	2/27/97	evening	2.0%
Tues	2/25/97	afternoon	8.5%
Tues	2/25/97	morning	20.8%

- Columbia U. to Bell Labs, Murray Hill

Fri	2/27/97	afternoon	1.1%

- losses bursty on all time scales

Internet loss correlation

Tues, 2/25/97, morning, Columbia to Germany:

loss block	count
1	12216
2	3646
3	1307
7	58
8	33
9	28
10	12
13	14
⋮	⋮
44	1

Mean: 1.687 (varies from 1.62 to 2.21)

Handling packet loss

- discover via gap in packet sequence (account for reordering)
- retransmit
- forward error correction
- redundancy
- danger: increase send rate under congestion-induced loss
- cover up: fill in waveform at receiver, e.g., based on prior and next block or interleaving
- avoid loss propagation ➠ make each packet individually usable

see Carle and Biersack, *IEEE Network*, Nov./Dec. 1997

Retransmission

- if enough time, ask for retransmission ➠
- multicast dangerous: most traffic lost by at least one receiver
- control traffic overhead (one control for each data)
- ➠ combine with FEC
- piggyback onto regular packets ➠ lower packet-header overhead

Forward Error Correction: Media-Independnet

k data packets, with $n - k$ parity packets ➠ can loose any k of n

- low complexity (for XOR, 1-of-n)
- can make n very large, increasing
- higher delay?
- recover seq. no. and timestamp
- sent as separate stream (port, RTP PT, $\ldots$)
- only for most significant bits?
- overlapping:
  - $a, f(a,b), b, f(b,c), c, f(c,d), \ldots$ ➠ single loss only
  - $f(a,b), f(a,c), f(a,b,c), f(c,d), f(c,e), f(c,d,e), \ldots$
  - designate by bit mask as offset from base SN

Forward Error Correction: Media-Specific

send lower-rate codec packets with delay offset:

- packet contains audio for t and low-bandwidth version of $t - n$ (e.g., LPC at 2.4 kb/s)
- H.263+: directly include key portions in each packet
- loose codec state – bad for low bit-rate codecs
- duplicate coding effort: low bitrate ➠ expensive
- with G.723.1 (6.3 kb/s), overhead high, but still only single loss recovered

Interleaving

data unit $<$ packet size ➠ packet contains unit $i, i+k, i+2k, \ldots$

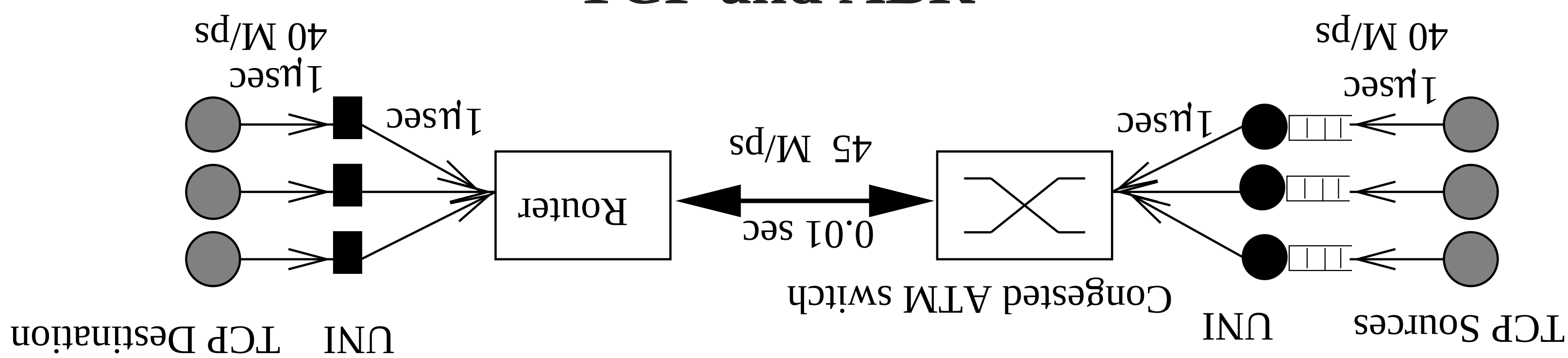
- delay = k
- possible with aggregation
- ➠ multiple small gaps ➠ similar to bit errors
- mostly useful for sample-based codecs (μ-law, DVI)
- no additional bandwidth needed

TCP and ABR

ATM ABR

- resource management cells every 32 cells
- switches compute “fair” bandwidth allocation
- explicit rate indication modified by switches
- sink reflects RM back to source
- can be zero loss, $\approx$ 100% throughput and distance-independent

TCP and ABR



- ABR reduces rate, but TCP side only notices when packets dropped
- use BCN in TCP to better match ABR

buffer	Tahoe	Reno	BCN-TCP
25 kB	98%	90%	99%
5 kB	94.6%	86%	99%