An Evaluation of Adaptive Multimedia Communication from a QoS Perspective

Michael Welzl
Johannes Kepler University
Linz / Austria

Max Mühlhäuser
TU Darmstadt
Germany

Outline

- Motivation / adaptation overview
- Rate scaling / congestion avoidance & control
- Simulation results
- Real life problems, future work
- Conclusion

When adapt?

- **Scenario:**
  Development of Internet "real-time" multimedia application
  - streaming (video-on-demand, ..)
  - interactive (IP telephony, videoconferencing, ..)

- **Today's best effort service (UDP), no QoS provisioning**

- **Assumption:**
  TCP not feasible - reliable data transfer, rate fluctuations (window based flow / congestion control)
  Well studied, yet not fully solved difficulties with:
  - asymmetric networks
  - "long fat" pipes
  - wireless links

Why adapt?

- Popular recommendation from researchers
  - primary goal: stable network, avoid congestion collapse
  - stability proved for AIMD congestion control
- Less popular among commercial application developers... **why?**
  - QoS gain not clear / not good enough
  - Often: "Lower resolution but constant frame rate (less packet loss)"
- Specific QoS goals: cope with
  - congestion
    - packet loss
    - delay
    - jitter
  - transmission errors

Note:
very critical effects for VoIP!

How adapt? Measure...

throughput ("goodput")
(mean, fluctuations,
packet loss ratio..)

\+ well studied

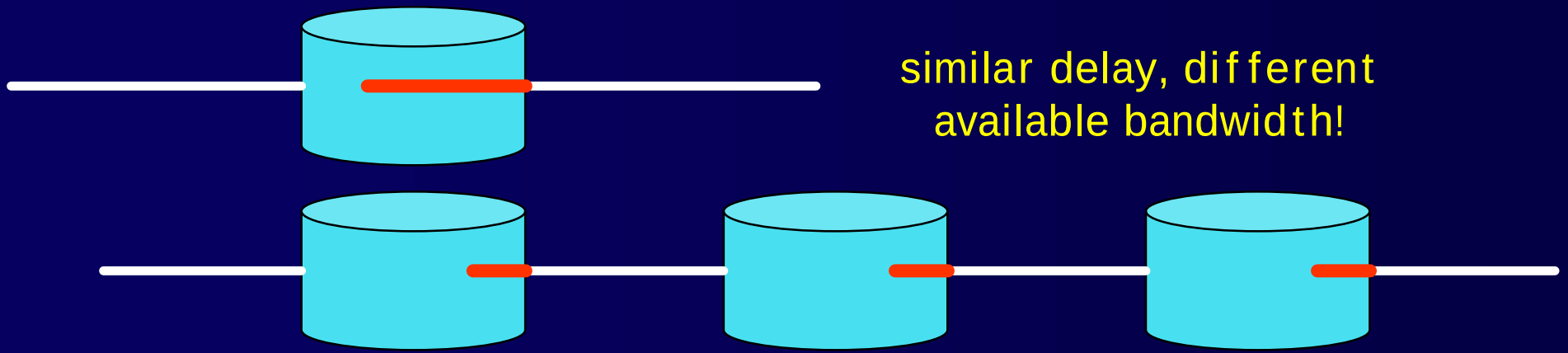
\- leads to misinterpretation of transmission errors

delay
(rtt, one way delay,
jitter..)

\+ easy to measure

\+ independent of transmission errors

\- not practical without throughput



... and change!

- **lower layers**
  - throughput (gap between packets)
    - flow control / congestion avoidance & control
    - well studied, many options - **our main interest!**
  - packet size
    - large: recommended (Path MTU Discovery), less overhead
    - small: less impact of transmission errors!
  - protocol (UDP Lite)

Note: packet size = granularity of throughput measurements

- **content**
  - compression
  - hierarchical encoding
  - FEC

Common difficulties:
bandwidth known (depending on content)?
granularity of rate changes

Overview of rate scaling congestion avoidance & control

Fairness

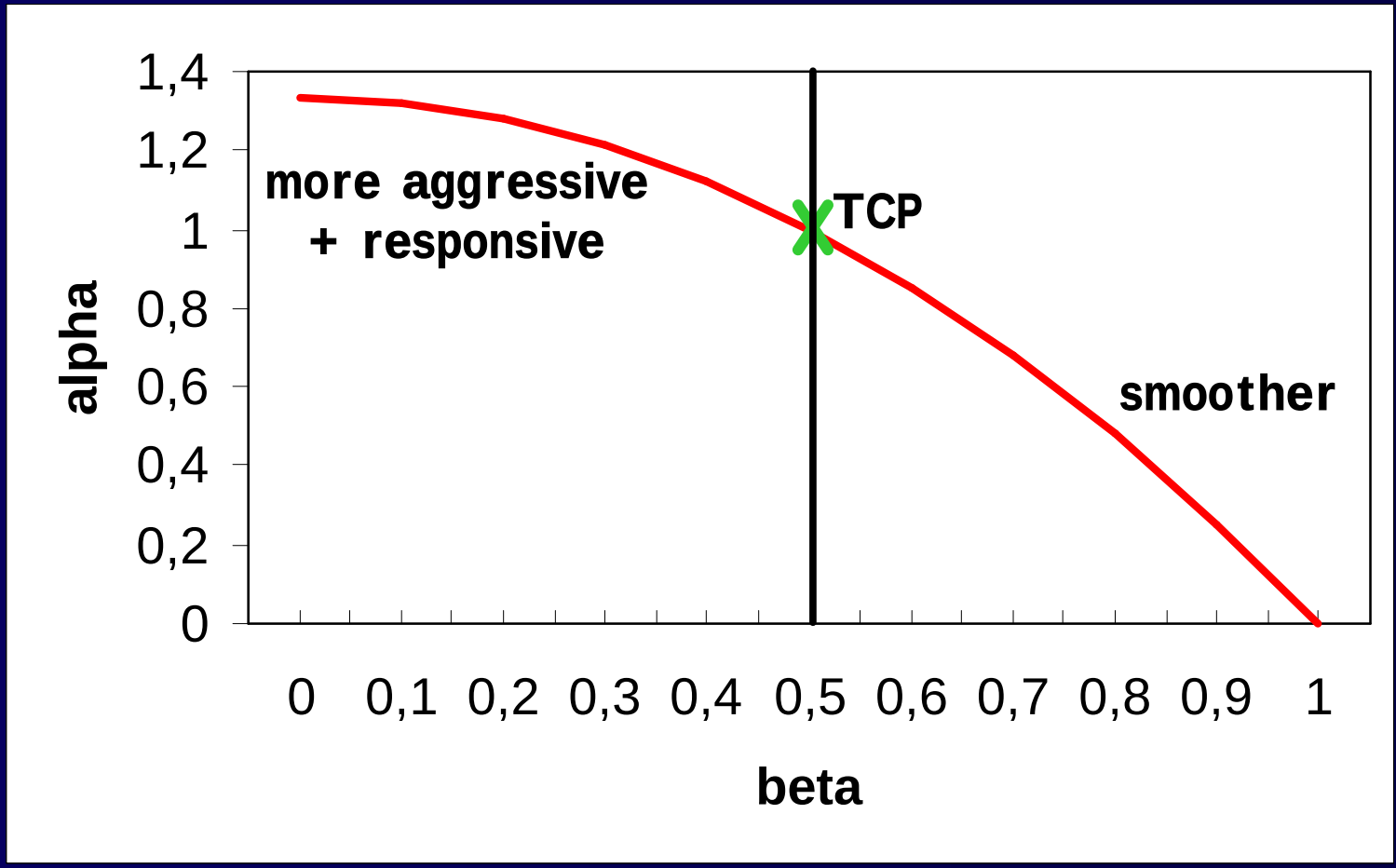
- **ATM:**
  - Max-Min-fairness
  - Mathematical definition satisfies "general" understanding of fairness - resources are divided equally among competitors
  - Usually requires knowledge of flows in routers (switches)
- **Internet:**
  - TCP dominant, but does not satisfy Max-Min-fairness criterion!
  - Ack-clocked - flows with shorter RTT react sooner (slow start, ..) and achieve better throughput
- Therefore, Internet definition of fairness: TCP-friendliness
  "A flow is TCP-compatible (TCP-friendly) if, in steady state, it uses no more bandwidth than a conformant TCP running under comparable conditions."

TCP-friendly congestion control: AIMD

- TCP-friendliness can be achieved by emulating TCP-behaviour (but without retransmission)
- Simplified TCP: **AIMD** (additive incr. **a**, multiplicative decr. **b**)
  - **0 < a, 0 < b < 1** -> stable and fair congestion control
  - **a = 4 x (1 - b^2) / 3** -> **GAIMD** TCP-friendly congestion control
  - **a = 1, b = 1/2** -> TCP RENO
- Rate-based AIMD mechanisms for multimedia applications:
  - **RAP**
    regular AIMD, ignore loss bursts
  - **LDA+**
    adjust **a** and **b** based on bottleneck bandwidth (packet pair approach)

GAIMD congestion control

Relationship between a and b for TCP-friendliness:



Equation based congestion control

- Based on TCP steady-state response function
  - gives upper bound for transmission rate T (bytes/sec):

$$T = \frac{s}{R\sqrt{\frac{2p}{3}} + t_{RTO}\left(3\sqrt{\frac{3p}{8}}\right)p\left(1+32p^2\right)}$$

s: packet size

R: rtt

t_{RTO}: TCP retransmit timeout

p: steady-state loss event rate (the difficult part!)

- well known example: **TFRC** - TCP-friendly rate control protocol
  - smooth sending rate

TCP Emulation At Receivers (TEAR)

- Assumption: probability of packet loss within a window rate independent
- TCP calculations (cwnd calculation, fast recovery, ...) moved to receiver
- Emulate TCP behaviour, calculate rate at receiver
- Less feedback (TCP acks every packet!)
- Even smoother throughput than TFRC

Evaluation:

RAP, TFRC, TEAR, GAIMD, "Naive Adaptation"

Evaluation

- Infocom 2001 paper by Yang, Kim, Lam examines simulations of **TCP** (Reno) vs. **GAIMD** (a=7/8, b=0.31) vs. **TFRC** vs. **TEAR**: (bigger is better)

- Network simulator "ns-2"
  - open source, original mechanism source codes used

- Well known scenario: compete on a single bottleneck link ("dumbbell")

Fairness	**TEAR ~ TFRC > GAIMD > TCP**
Smoothness	**TEAR > TFRC >= GAIMD > TCP**
Responsiveness	**TCP > GAIMD > TFRC >= TEAR**
Aggressiveness	**TCP > GAIMD > TFRC > TEAR**

...and QoS?

- Parameters not very QoS oriented
- Smoothness not very interesting if large buffers are available (one-way streaming)
- Despite efforts to identify non-adaptive flows:
  The more you send, the more you get.
- Probably most interesting QoS question beside smoothness:
  How close does the mechanism follow the available bandwidth?
  - measurable parameter: **packet loss ratio**
- Performed QoS-centric studies
  Not enough for proof - just to get an idea

Simulation scenario

- ns-2, original source codes for **RAP** (instead of GAIMD), **TFRC**, **TEAR**, default parameters; TCP behaviour not studied

- **"Naive adaptation":**
  - not TCP-friendly
  - Assumption: real application with **lower**/**upper** bound, **discrete rate steps**, **start bandwidth**
  - No loss -> linear increase by 1 bandwidth step
  - Loss -> assume measured rate = available bandwidth, use measured bandwidth
  - RTT clocking, timeout based on initial ping

- Scenario: Compete individually with TCP Reno flows on a single bottleneck link, 10 minutes, packet size=500 byte, bottleneck=0.5Mbit/s

Simulation results (best.....worst)

10 vs. 10	RAP	TFRC	TEAR	Naive
Throughput	77044000	56083000	56194000	92910000
Packet Loss Ratio	0.04180088 30296623	0.01869592 22417609	0.02621865 6315525	0.03079932 19454949
avg. Jitter	0.00389460 475532101	0.00535009 936957743	0.005361715 11295396	0.00334239 88669218

100 vs. 100	RAP	TFRC	TEAR	Naive
Throughput	125344000	18625000	132290000	306876500
Packet Loss Ratio	0.166149435 036705	0.107827169 95593	0.20472272 9035225	0.39179203 2043856
avg. Jitter	0.00241562 348469903	0.016587441 3293519	0.00264160 998181682	0.00264768 346476151

Interpretation

- Note: high throughput only a good sign if packet loss is low
- Although TEAR shows smoother throughput, TFRC always showed the smallest packet loss ratio
- Naive adaptation generally sent the most, but still smallest (or close to smallest) jitter
  - RAP has low jitter too
- Neither RAP nor Naive adaptation show a very good packet loss ratio
- Benefits / drawbacks of simple adaptation not clear - but: Naive good enough to make QoS of other mechanisms questionable!

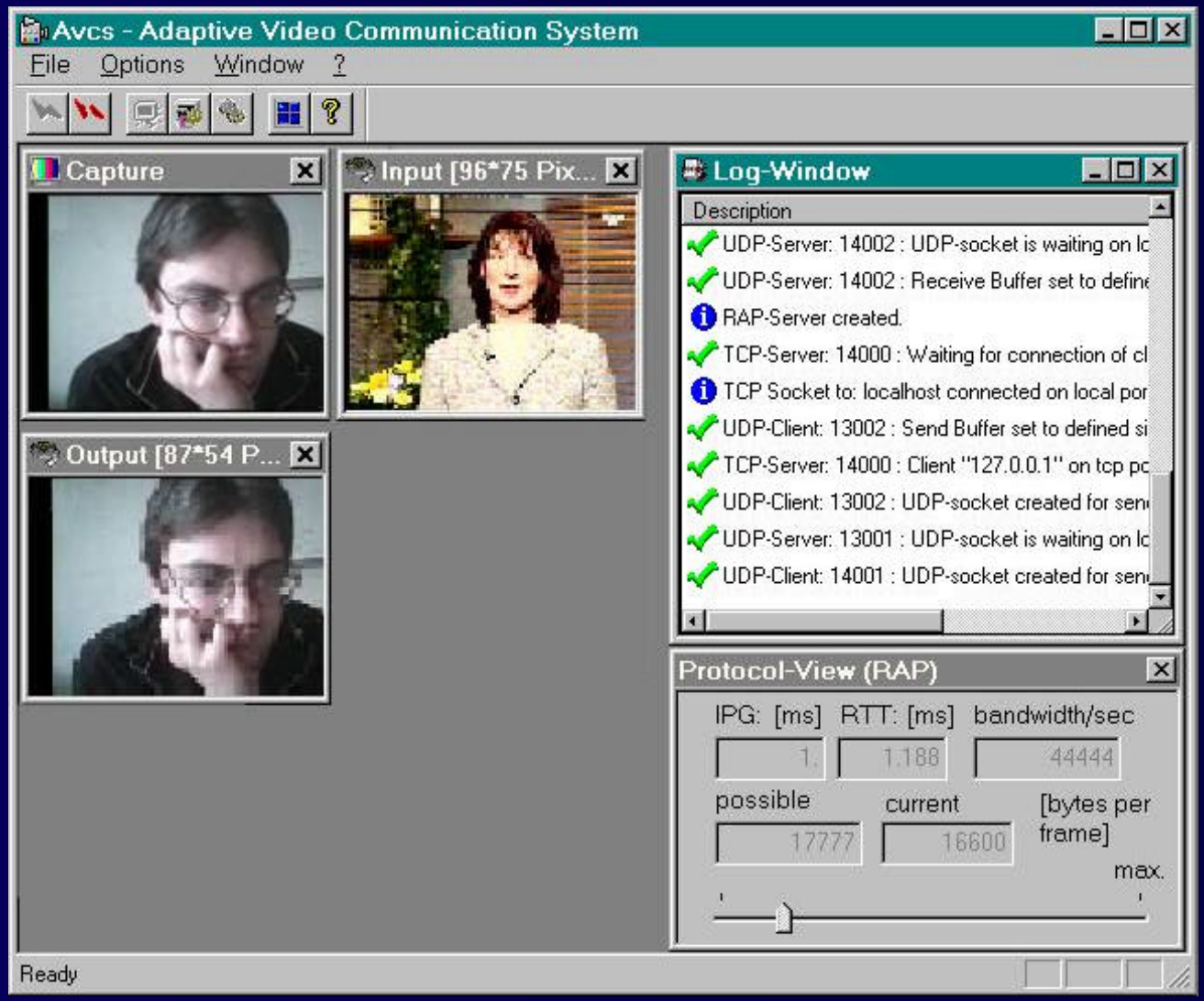
Real life evaluation: AVCS

Adaptive Video Communication System:

Testbed for adaptation mechanisms

RAP, TFRC implemented

Not yet finished, but already illustrates unsolved difficulties



Problems with AVCS

- **Operating System timing**
  - sleep(3ms): MS Windows NT: **>= 10ms**, MS Win 9x: **>= 55ms,**
    MS Win 98 multimedia timers: **>= 2ms**
    Example: 56 kbit/s = 7000 byte/s.
    packetsize: 1000 byte -> send a packet every 1.43 ms
    max. bw: ok (do not wait, fill buffer). Decrease: wait 1.43*2 ms !
  - undeterministic timing: threading, protocol stack, ..
  - but: TFRC successfully tested under Unix by authors!

- **Goal: visible adaptation, not best quality:**
  - **BMP:** change resolution. Specific number of fixed rates;
    "Lower resolution but constant frame rate (less packet loss)"
  - **M-JPEG:** uses IJG open JPEG library; "linear compression"
    size curve approximated by **size = x-y*ln(compression)**
    encode each frame 3 times (2 times to determine x, y)!

General problems

- Adaptation schemes assume arbitrary data stream scalability
- Data stream may fluctuate (e.g. MPEG: I-, B-, P-frames)
- Bandwidth may depend on content
- Real life distance learning example:
  40kbps enough for streaming video (Smartboard) + audio (speech),
  but speech suffers dramatically if teacher visible
- Distributed multimedia apps often need multicast; quality adaptation for multicast under research, but difficult

Not-so-TCP-friendly solutions

- Network stability: some adaptation still better than none!
- Overcome rate fluctuations:
  limit encodings (e.g. 2 or 3 qualities), let user decide
- Cross-media-adaptation:
  choose from video, audio, single pictures, text
- Limit by bottleneck bandwidth
  - often: "last mile" - e.g. RealMedia
  - better: determine actual bottleneck via packet pair approach
- If wireless link involved: small packets, UDP Lite

Another solution: Explicit Rate

- Explicit Rate works great for ATM ABR: very small cell loss ratio
- TCP over ABR: control loop on top of control loop
- **"ABR to the Internet" project:**
  - **goal**: motivate developers to use a TCP-friendly service
    QoS gain: very small packet loss ratio, also works over wireless links
  - small complexity for routers, move calculations to the edges
  - TCP-friendliness instead of Max-Min-Fairness
    (ack clocking instead of per flow state)
  - Special session at SCI'2001
  - **BOF request** for 51st IETF meeting, London
  - please join! **http://www.tk.uni-linz.ac.at/~michael/abr-internet/**

Conclusion

- Basic QoS gain of adaptation:
  - reduced packet loss ratio -> choose TFRC
  - reduced jitter -> choose RAP
  - smoother rate -> choose TEAR
- These factors are crucial for audio transmission (VoIP)!
- Unconsidered possibility: send more (do FEC) in response to packet loss
  - (very network-unfriendly behaviour, but may yield a smaller data loss ratio)
- Real-life result: TCP-friendly adaptation expensive and difficult
  - consider non-TCP-friendly alternatives
- Consider joining the "ABR to the Internet" project ;)

References

- Sally Floyd, Mark Handley, Jitendra Padhye, and Jörg Widmer, "Equation-Based Congestion Control for Unicast Applications", ACM SIGCOMM 2000
  http://www.aciri.org/tfrc

- Injong Rhee, Volkan Ozdemir, Yung Yi, "TEAR: TCP emulation at receivers - flow control for multimedia streaming", Technical Report, Department of Computer Science, NCSU
  http://www.csc.ncsu.edu/faculty/rhee/export/tear_page/

- R. Rejaie, M. Handley, D. Estrin, "RAP: An End-to-end Rate-based Congestion Control Mechanism for Realtime Streams in the Internet", IEEE INFOCOM 1999
  http://www.research.att.com/~reza/RAP/

References /2

- Y. Richard Yang, Simon S. Lam, "General AIMD Congestion Control", ICNP 2000, November 2000
- Sally Floyd and Kevin Fall, "Promoting the Use of End-to-End Congestion Control in the Internet", IEEE/ACM Transactions on Networking, May 1999
- Yang Richard Yang, Min Sik Kim, Simon S. Lam, "Transient Behaviors of TCP-friendly Congestion Control Protocols", IEEE INFOCOM 2001
- AVCS written by Gerhard Stummer / supervised by Michael Welzl