Scalable High-Speed Congestion Control with Explicit Traffic Signaling

Michael Welzl
michael.welzl@uibk.ac.at
University of Innsbruck

Max Mühlhäuser
max@informatik.tu-darmstadt.de
TU Darmstadt

Outline

- What?
- Why?
- How?
- Conclusion + future work

Out-of-band signaling to support CC ?

- Yes, that's right
  - several reasons against in-band ... I'll explain offline
- Idea similar to ATM ABR Explicit Rate Feedback, but:
  - scalable
  - efficient (lightweight)
  - designed for packet nets
  - a generic signaling framework
- Various endpoint adaptation mechanisms possible
  - I found a good one :)

Outline

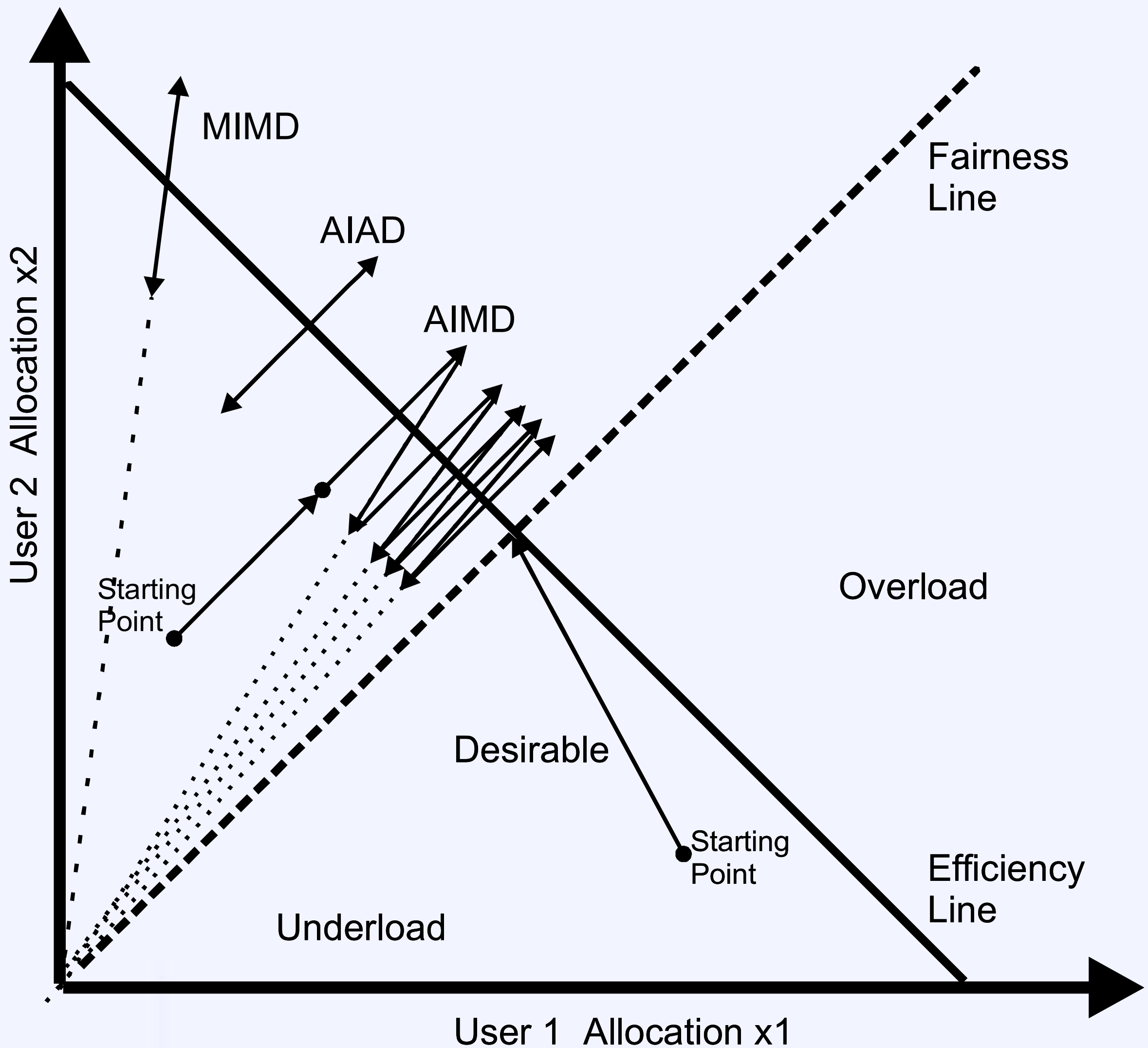
- What?
- Why?
- How?
- Conclusion + future work

Reasons against TCP

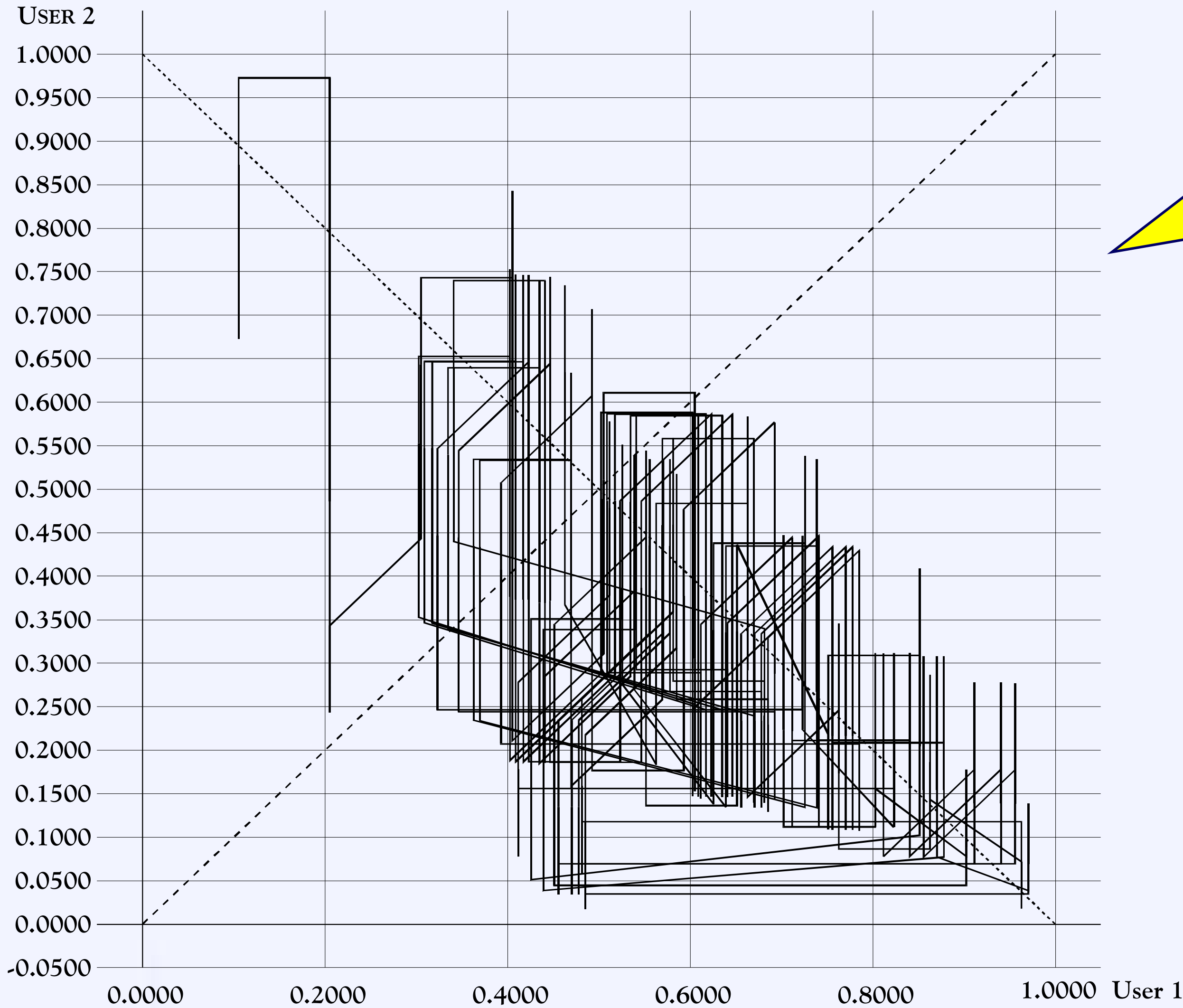
- TCP over wireless:
  - checksum error -> packet drop misinterpretation
- TCP over "long fat pipes": (large bandwidth*delay product)
  - long time to reach equilibrium, MD = dramatic!
- TCP stability issues:
  - equilibrium, not a stable point - fluctuations lead to regular packet drops & reduced throughput
  - not feasible for streaming multimedia apps
  - Stability depends on delay, capacity, load and AQM [Steven Low]
- ...wild claims:
  - AIMD is definitely not necessary for stability
  - TCP-friendly congestion control is like building a slow Porsche
  - we can do better than TCP!

suits the user + is fair!

AIMD in Theory (equal RTTs)

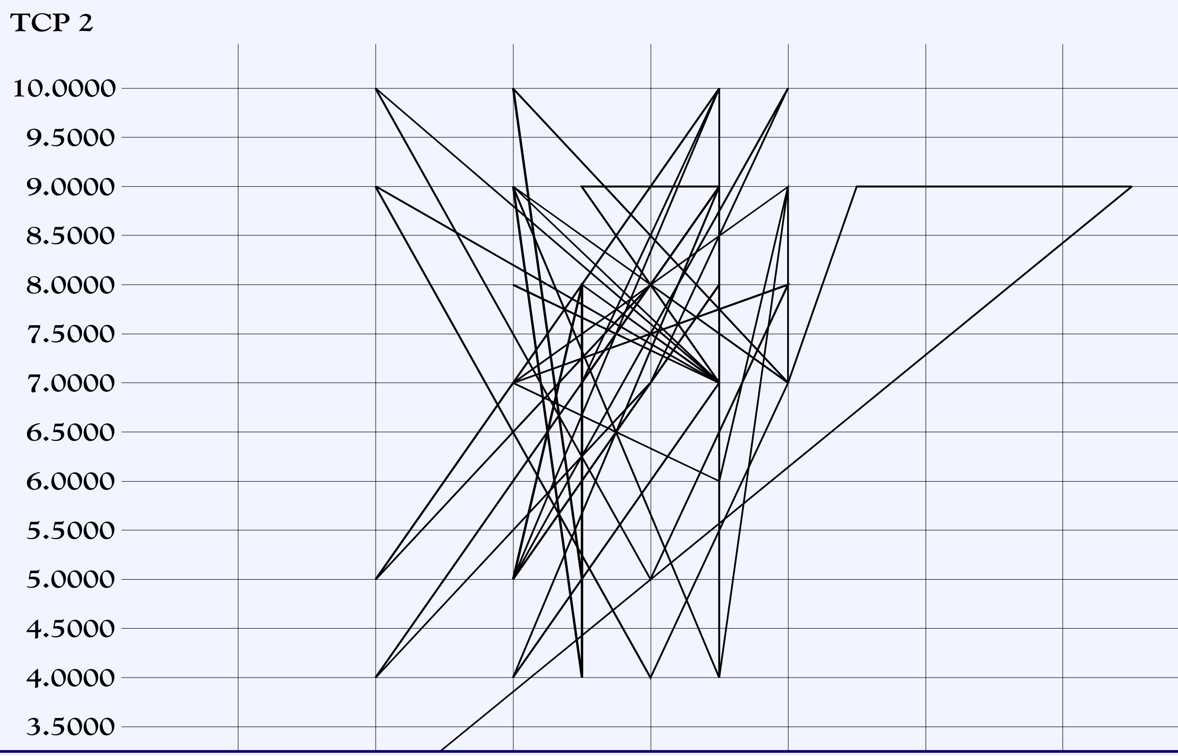


AIMD / asynchronous RTTs



- fluid model
- RTT: 7 vs. 2
- AI=0.1, MD=0.5
- Simul. time=175

AIMD in practice (TCP)



- ns-2 simulator
- TCP Tahoe
- “equal“ RTT
- 1 bottleneck link

Quote from a colleague:

„That‘s what my (9 months old) daugther does when I give her a pen“

Outline

- What?
- Why?
- How?
- Conclusion + future work

The signaling protocol: PTP

- Framework: "generic" ECN - to carry traffic information (standardized Content Types, e.g. queue length, ..)
- **Stateless & simple -> scalable!**
  - all calculations: end nodes
- Only every 2nd router needed for full functionality
- Available Bandwidth Determination:
  - nominal bandwidth ("ifSpeed") + 2* (address + traffic counter ("if(In/Out)Octets") + timestamp) = available bandwidth
- two modes:
  - Forward Packet Stamping
  - Direct Reply (not for available bandwidth (byte counters))

No problems w/ wireless links unless combined with packet loss!

Endpoint Mechanism Design Algorithm(tm)

- find useful (closely related) ATM ABR mechanism
- start with simplifications, then expand the model
- A new mechanism must work for 2 users, equal RTT
  - simple analysis similar to Chiu/Jain (diagram + math)
- it must also work with heterogeneous RTTs
  - simulate using a simple Diagram Based Simulator(tm)
- it must also work with more users and in more realistic scenarios
  - simulate with ns

The ATM ABR best match: CAPC

- "**Congestion Avoidance with Proportional Control**" (Barnhart, 1994)

- Uses load factor LF: Input Rate IR / Target Rate R0
  - R0 e.g. 95% of nominal bandwidth, d = 1 - LF (available bandwidth)

- *"As long as the incoming rate is greater than R0, the desired rate, ERS will diminish at a rate that is proportional to the amount by which R0 is exceeded. Conversely, whenever the incoming rate is less than R0, ERS will increase."*

- for each new cell entering the queue:
  LF<=1: ERX = min(ERU, 1 + d*Rup) ... else ERX = max(ERF, 1 + d*Rdn)
  ERS = ERS*ERX
    - constants: Rup, Rdn define the speed of rate increase / decrease, ERU, ERF = upper / lower bound
    - different default values for LAN and WAN!

hint for RTT dependance!

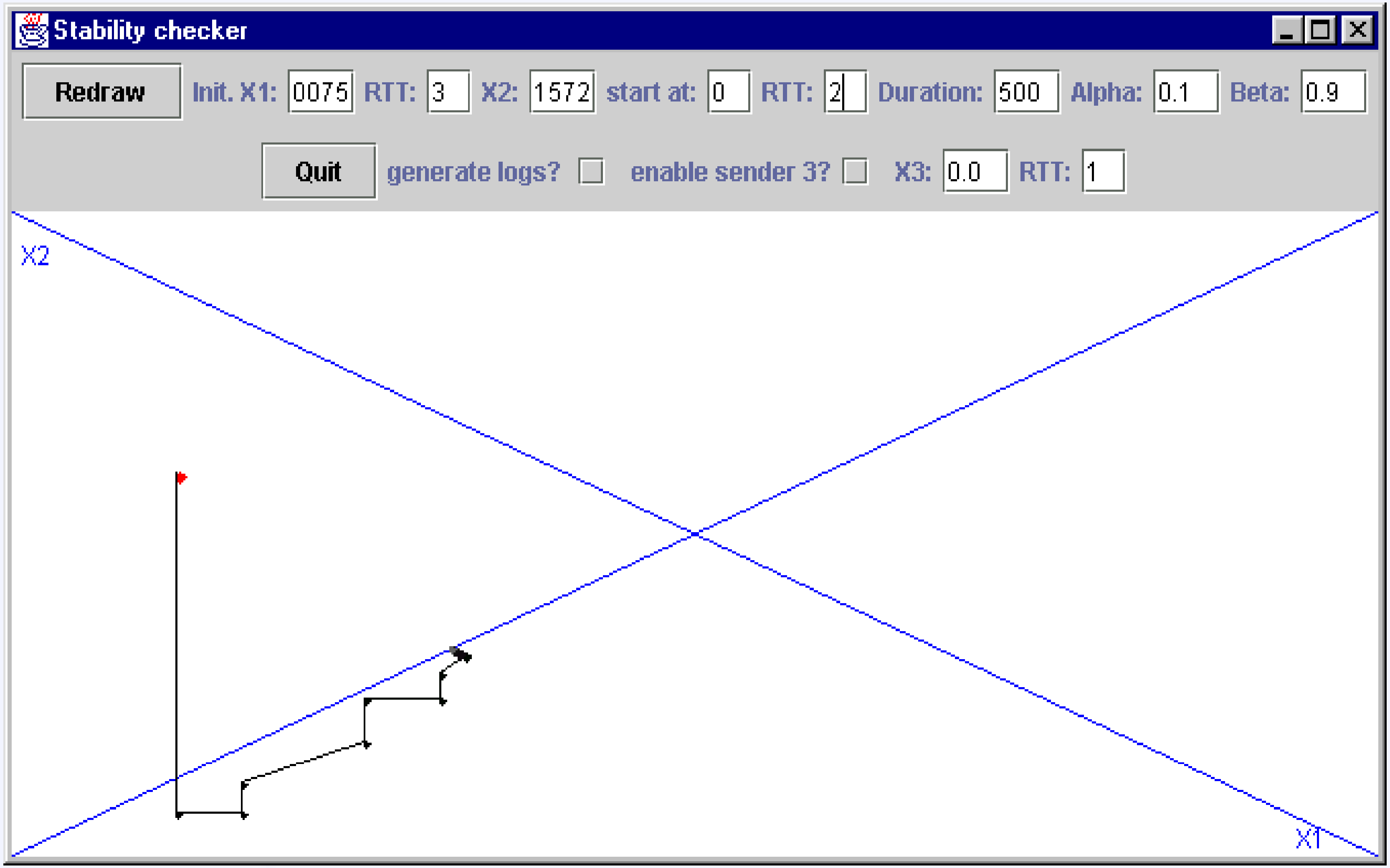
Conversion for packet nets: CADPC

- "Congestion Avoidance with Distributed Proportional Control"
- Only ask for current load, do calculations at sender
  - implementation in diagram based simulator trivial
  - rates leave fairness line if RTT's are not equal :(
- Idea:
  - relate user's current rate to the state of the system! (also in LDA+)
    Thought: in the Chiu-Jain-diagram, if the rate increase factor is indirectly proportional to the user's current rate, the rates will equalize.
- From:
  - erx = 1 + d* rup = 1 + rup * (1 - traffic/r0)
- To:
  - erx = 1 + rup * (1 - myRate/d)
- dependence on rup not desirable
  - rate changes should be proportional to the current load -> use d instead of rup!

relate
traffic : target rate

relate
user's rate : available bandwidth

CADPC vector diagram analysis



CADPC synchronous case analysis

- <u>Final formula per user:</u>
  d = 1 - traffic / r0;
  erx = 1 + d * (1 - myRate/d);
  ers = ers*erx;

- Combined:
  $x_i(t)$ = rate of user i, n users

$$x_i(t+1) = x_i(t)\left(1 + d\left[1 - \frac{x_i(t)}{1 - \frac{\sum_{j=1}^{n} x_j(t)}{r_0}}\right]\right)$$

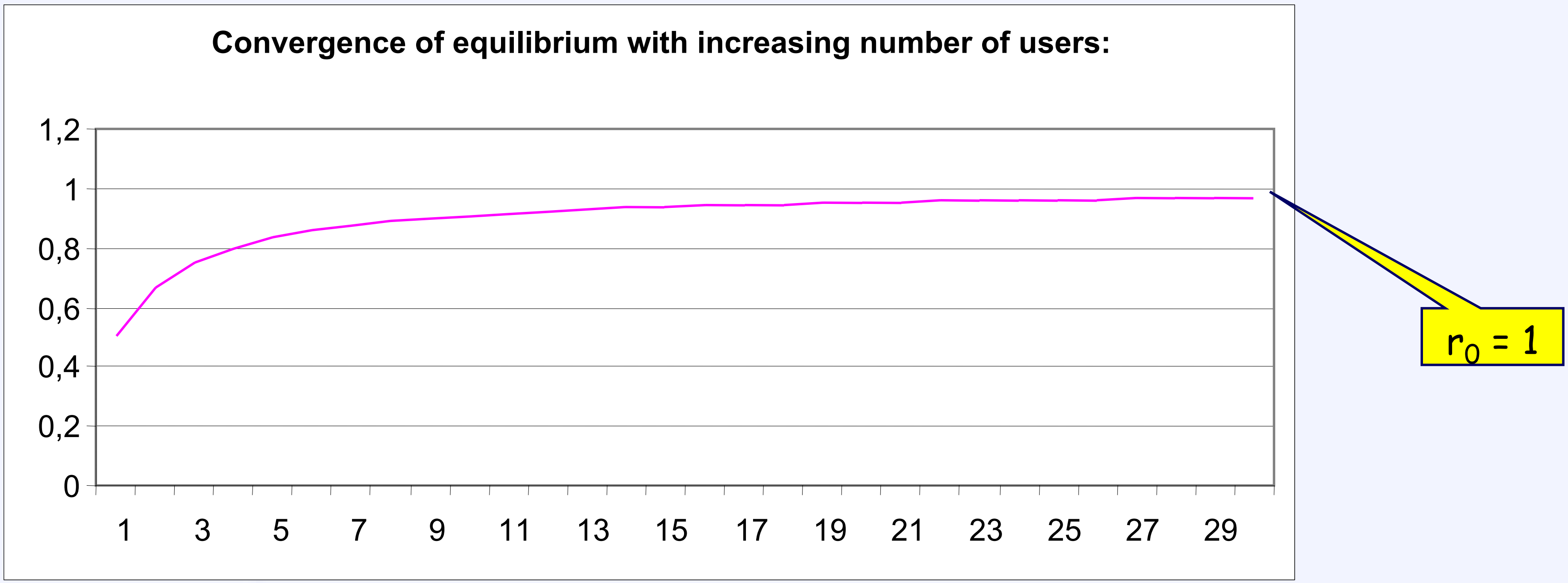
1 user, **r0=1**:
logistic equation
=> **stable!**

- after some straight-forward derivations:

$$x_i(t+1) = x_i(t)\left(r_0 + 1 - r_0 x_i(t) - \sum_{j=1}^{n} x_j(t)\right)$$

CADPC synchronous case analysis /2

- Equilibrium: assume $x(t+1) = x(t)$
- leads to: $x(t) = r_0/(n+r_0)$
- traffic (n users): $n*x(t) = n*r_0/(n+r_0)$



... the simplest ns code ever :)

- Upon timeout (>= 2 RTTs), send a PTP packet
- Upon PTP packet arrival do:
  - UpdateRTT
  - *// normalize*
    traffic = traffic / bottleneckBW;
    currentRate = currentRate / bottleneckBW;

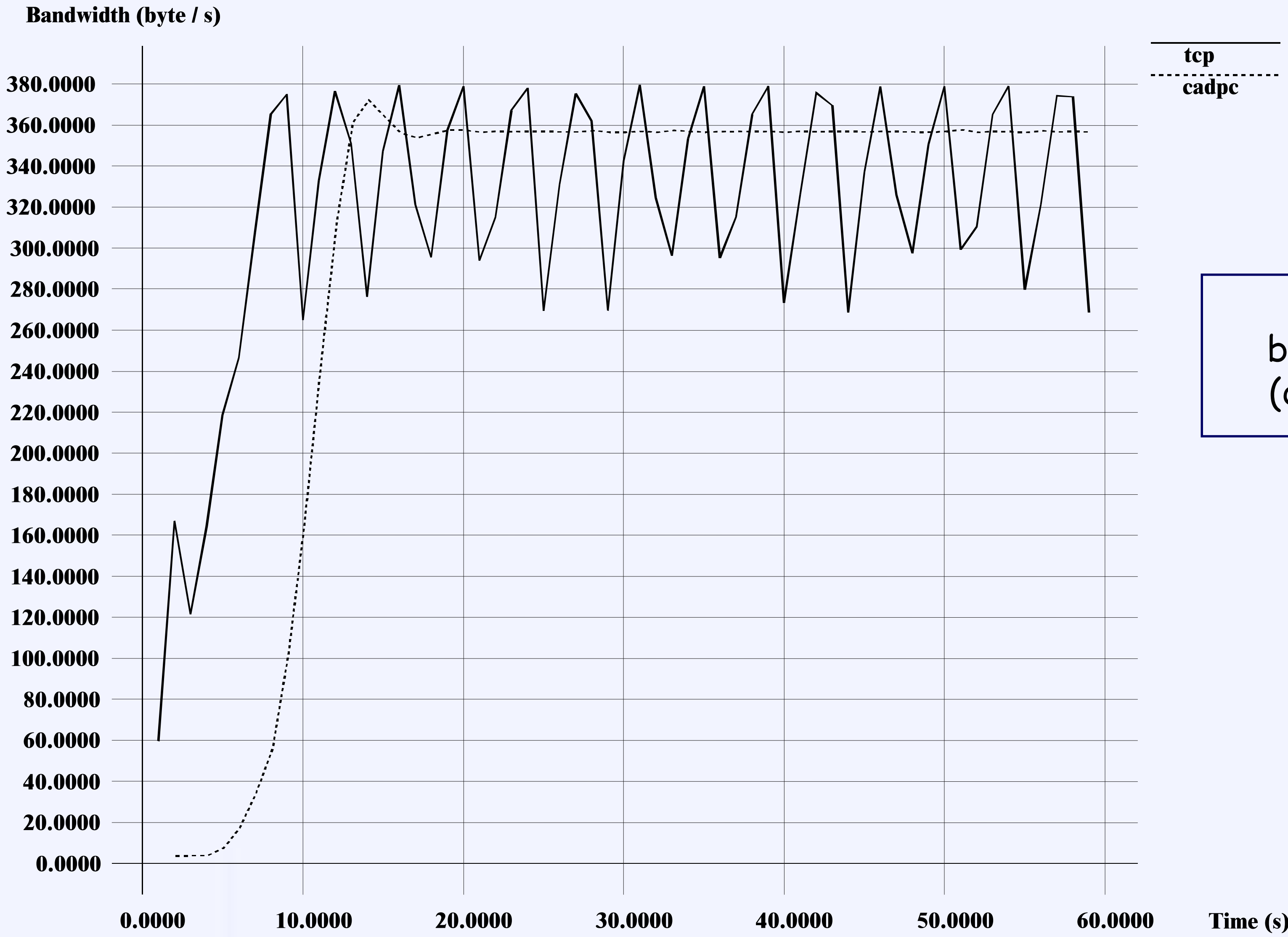
    newRate = currentRate*(2.0-currentRate-traffic);

    // de-normalize
    newRate = newRate * bottleneckBW;

from PTP

not simultaneously!

ns simulation: 25 TCP / 25 CADPC



single bottleneck (dumbbell)

Results

- Implementation: r_0 normalized to 1 -> calc -> de-normalize
- 1 PTP packet every 4 RTTs, no other acks!
  - rate indeed converges to n/n+1
- **No** packet loss
- Very smooth rate, rapid convergence
  - the higher the link bandwidth, the better!
- Not in the picture:
  - rapid convergence to almost perfect fairness
  - bg traffic: rapid backoff and recovery

Outline

- What?
- Why?
- How?
- <u>Conclusion + future work</u>

CADPC advantages

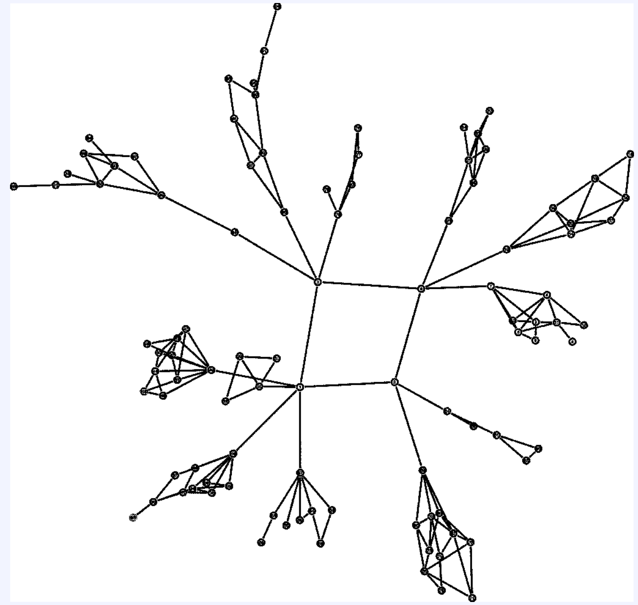
- Better stability than TCP
  - smooth rate advantageous for streaming media apps
- No problems with wireless links (no packet loss interpretation)
- Rare feedback - good in environments with long delay
  - rapid convergence & reaction - good in environments with a high bw*delay product
- Rate calculation independent of RTT => independent of position
  - **scalable!** if PTP = x% of generated traffic *n*, PTP scales O(*n*)
- Only (rare) PTP packets necessary to calculate rate
  - Satellite environments:
do receiver's calculations at sat. base station and give earlier feedback
  - easier to differentiate pricing
  - easier to implement metering => traffic shaping, policing, admission control, ..

Deployment plans

- Problem: PTP needs router support
  - CADPC needs complete path information (every 2nd router)

- Possibilities:
  - CADPC / PTP within a DiffServ class (QoS "in the small"):
    "we offer QoS & provide router support,
    you use CADPC and get a good result
    [and we can calculate your rate, too]"
  - If CADPC works with non-greedy senders:
    edge2edge PTP signaling (TCP over CADPC)
    PTP supported traffic engineering
  - CADPC <=> TCP translation at edge routers?

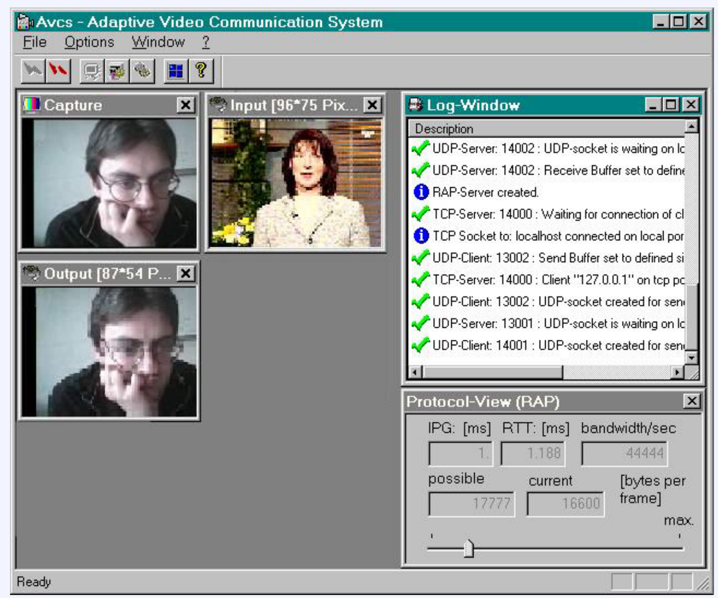
Future work



- More ns simulations
  - CADPC vs. AIMD in vector diagram simulator: CADPC is much less aggressive
  - compare with TCP-friendly binary mechanisms
  - compare with other ER mechanisms (PCP, ALS)

- Extension to proportional fairness?

- CADPC implementation
  - PTP already available for Linux
  - compare with TCP, TFRC, RAP, ...
  - evaluate QoS



The End ...

- Further documentation
- PTP ns code
- PTP Linux code (router kernel patch + end system implementation)
- Future updates: Ph.D. thesis, CADPC code, ..

http://fullspeed.to/ptp