A Congestion Control Mechanism for Wireless Links

Upperside Wi-Fi Voice 2004

Michael Welzl
http://www.welzl.at, michael.welzl@uibk.ac.at
Distributed and Parallel Systems Group
Institute of Computer Science
University of Innsbruck, Austria

Outline

- Problem identification
- The PTP signaling protocol
- The CADPC congestion control mechanism
- Simulation results
- Conclusion

Wi-Fi + noise: what happens?

- 802.11 link layer ARQ - classify based on persistence:
  - low/high: retransmit for a while, then give up [RFC3366]
  - perfect: not reasonable for WiFi
    (neither unlimited buffer nor "BROKEN LINK" signal feasible)

- Simple test:
  - cheap off-the-shelf equipment, 2 positions at our institute
  - 1800 pings (15 minutes, 1 ping every 0.5 seconds), packet size = 100 byte

Almost perfect environment	Very noisy environment
rtt min/avg/max/mdev = 1.941/**3.139**/6.152/0.424 ms	rtt min/avg/max/mdev = 3.274/**12.410**/1022.437/46.585 ms

Wi-Fi + noise + TCP = poor performance!

- TCP congestion control:
  - introduced in the 80s, necessary to prevent congestion collapse
- TCP interpretation of network feedback:
  - single packet loss = sign of congestion
    or
    Explicit Congestion Notification (ECN) flag = 1 = sign of congestion
    - sender halves rate (Congestion Avoidance)
  - many packets lost in a row = sign of heavy congestion (timeout)
    - sender goes back to Slow Start
  - round-trip time estimate increases / decreases
    - longer RTT = more conservative (e.g., increase by ≈ 1 segment / RTT)

Wireless links: misinterpretation of ***packet loss*** and ***RTT estimate*** ⇒ massive throughput degradation!

Overview of problems with TCP(-like) CC

- Special links (becoming common!):
  - noisy (wireless) links
  - "long fat pipes" (large bandwidth*delay product)
- Stability issues
  - Fluctuations lead to regular packet drops + reduced throughput
    $\Rightarrow$ problematic for streaming media
  - Stability depends on delay, capacity, load, AQM
- Rate hard to control / trace / predict
  - Load based charging difficult
- **Main reason:** binary congestion notification (E)CN
  - when it occurs, it's (almost) too late

The CADPC/PTP Solution

- Totally different CC model
  - *only* rely on rare explicit bandwidth (=traffic) signaling

...Ignore packet loss and delay!

- Assumptions:
  - extra forward signalling for CC = good idea *(≠ common belief)*
  - router support
  - mechanism must clearly outperform TCP to justify (even a little!) additional traffic
  - timeouts necessary for loss of signalling packets
    rate should not depend on round trip time

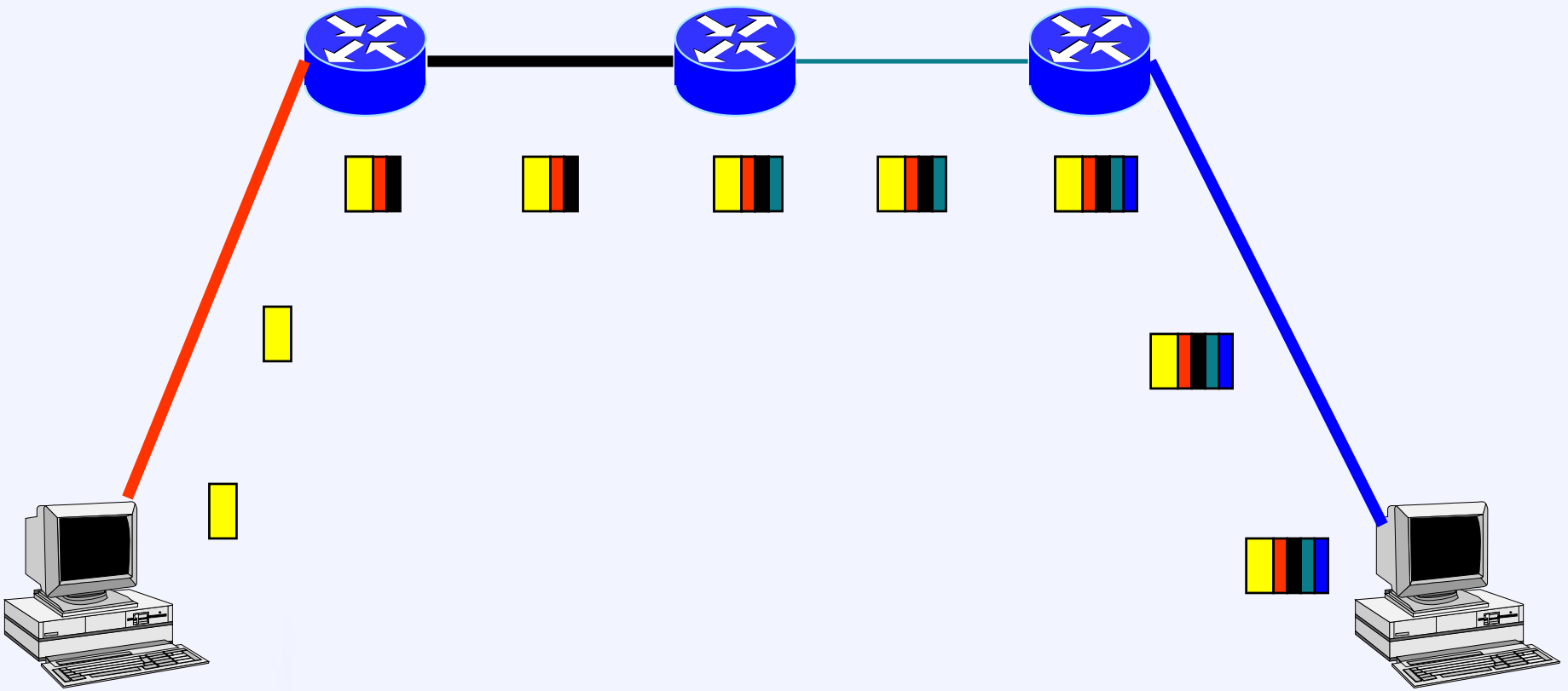
... yes it does :-)

The Performance Transparency Protocol (PTP)

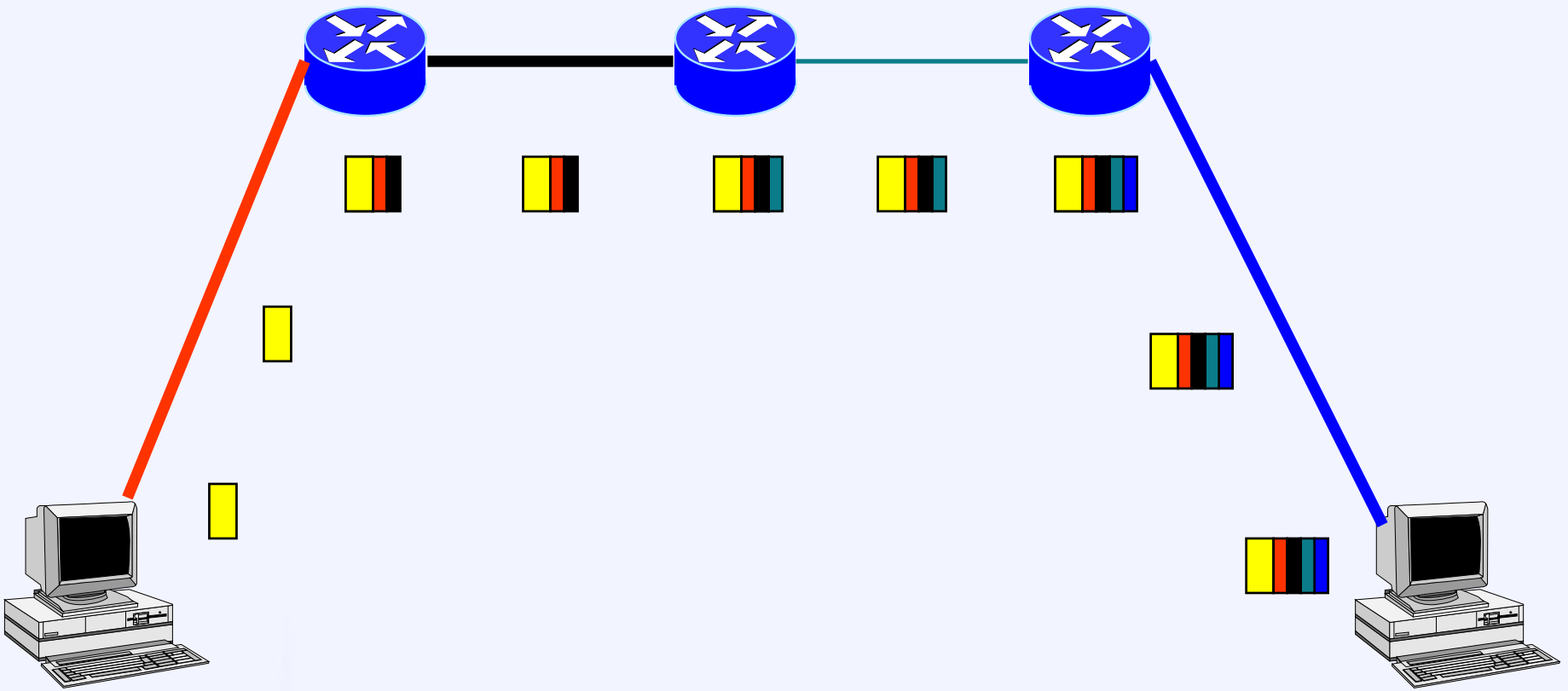
- Basic idea: query routers for performance related information
  - designed to be as lightweight as possible
- **Stateless + simple ⇒ scalable!**
  - all calculations @ end nodes
- Only every 2nd router needed for *full* functionality
- PTP Available Bandwidth Determination:
  - packet queries for:
    - nominal bandwidth ("ifSpeed") + address + traffic counter ("if(In/Out)Octets") + timestamp)
    - 2 consecutive packets: table of traffic + interval from all routers at receiver
    - receiver calculates available bandwidth at bottleneck, informs sender
- **Designed for flexibility - two modes:**
  - Forward Packet Stamping, Direct Reply (not for available bandwidth (byte counters))

No problems w/ wireless links!

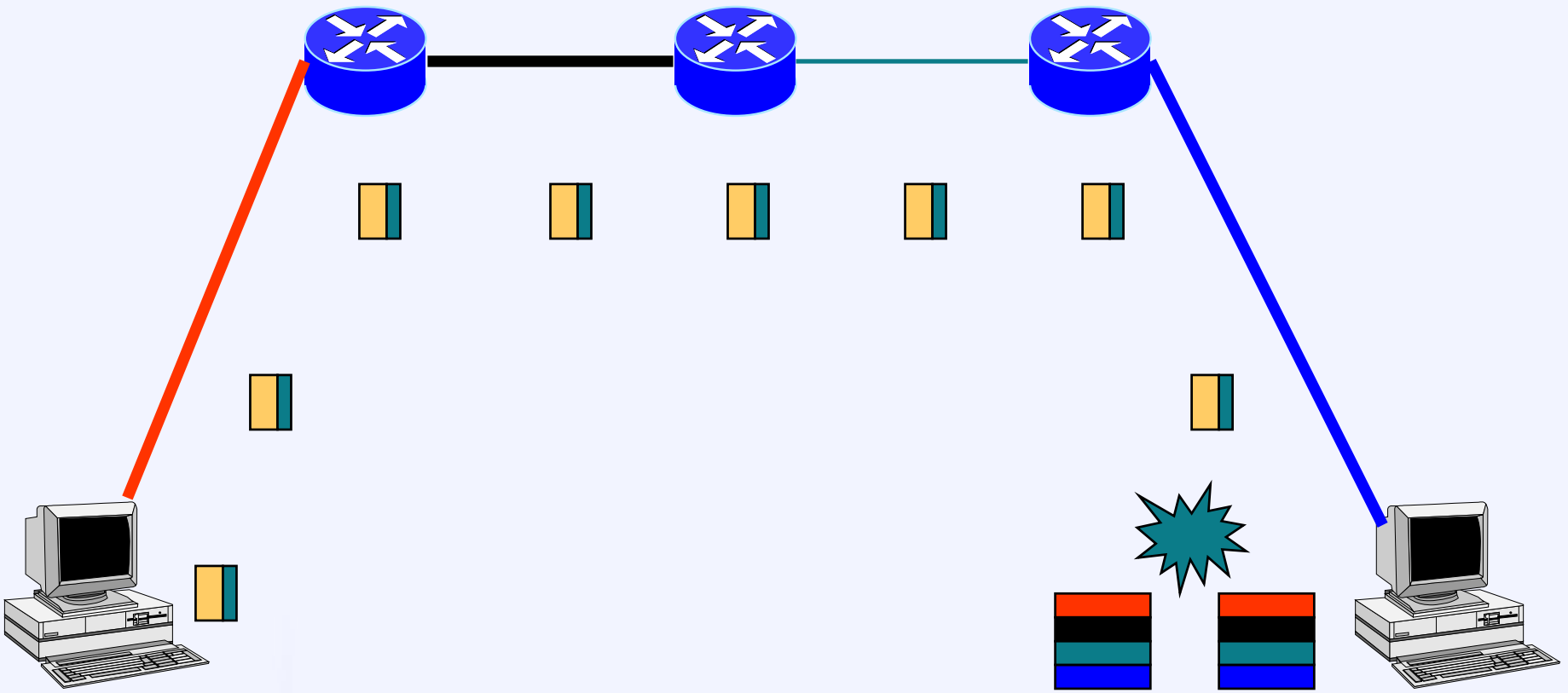
Forward Packet Stamping: how it works



Forward Packet Stamping: how it works



Forward Packet Stamping: how it works



CADPC: a new CC mechanism

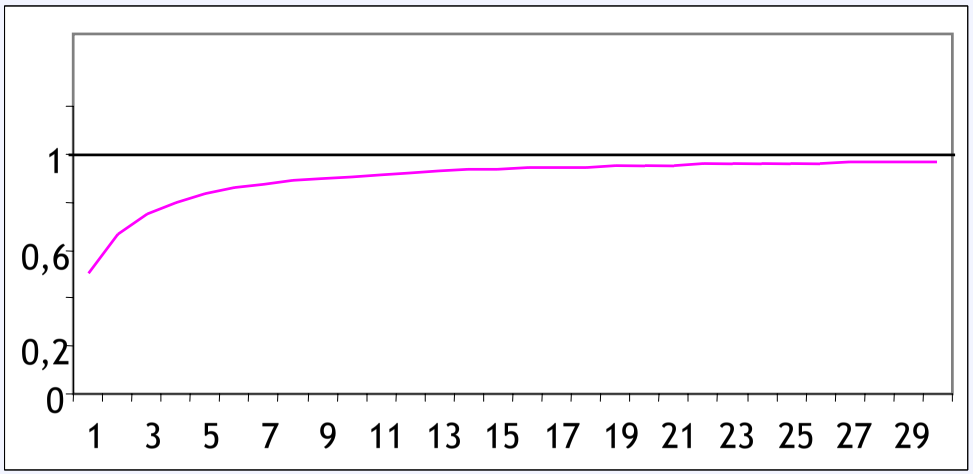
- "Congestion Avoidance with Distributed Proportional Control"
  fully distributed convergence to max-min fairness
  - each source increases/decreases the rate depending on its capacity share
- Only depends on old rate, smoothness factor and traffic
  - does not depend on packet loss or RTT
    - Feedback packets can be delayed $\Rightarrow$ scalable
    - reasonable choice: 4 x RTT
- Rate based, simple control
- Smooth convergence to a steady rate

CADPC synchronous case fluid analysis

- **Final formula per user:**

  $x(t+1) = x(t)a(1-x(t)-traffic)+x(t)$

  x(t) ... normalised rate at time t
  a... smoothness factor
  (should be 0 < a <= 1)
  traffic (normalised) ... from PTP



- Converges to: n/(n+1)

- Continuous-time version of synchronous case *(traffic=nx)*:
  logistic growth $x'(t) = x(t)a(1-x(t)/c)$ *[c = 1/(1+n)]*
  asymptotically stable equilibrium point: c

Some simulation results

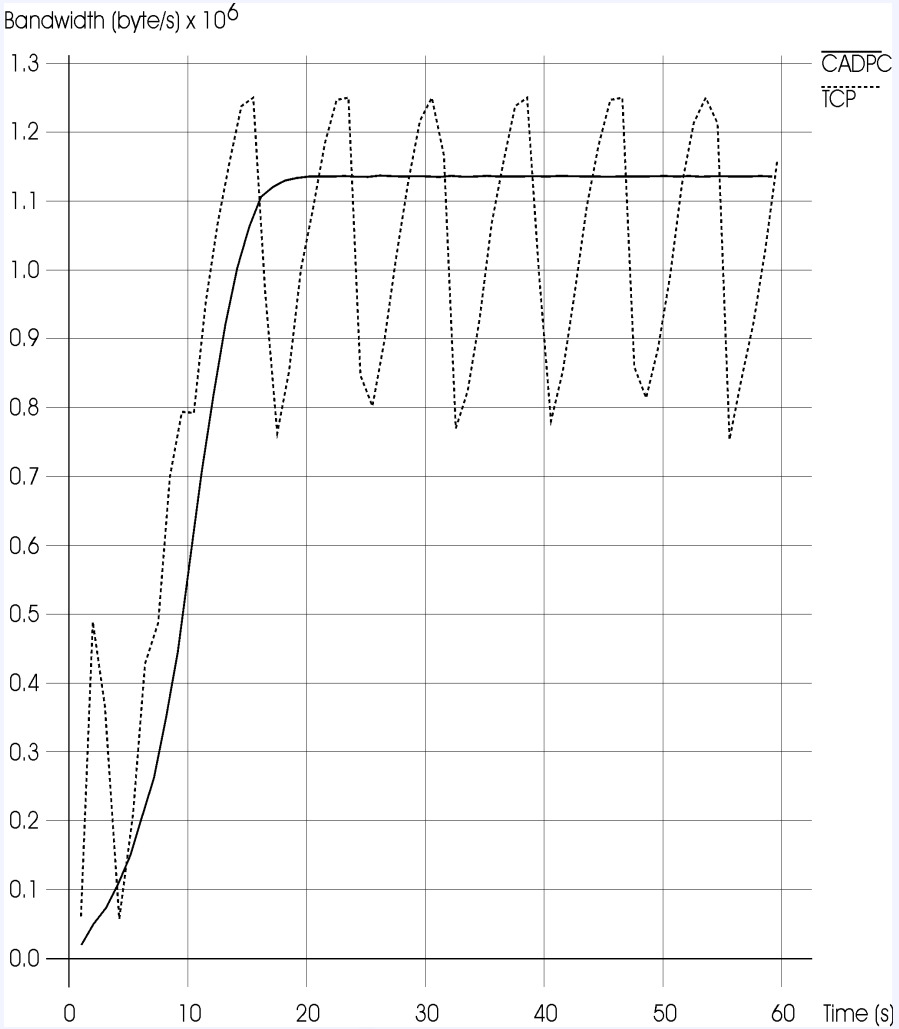
Many more can be found in:

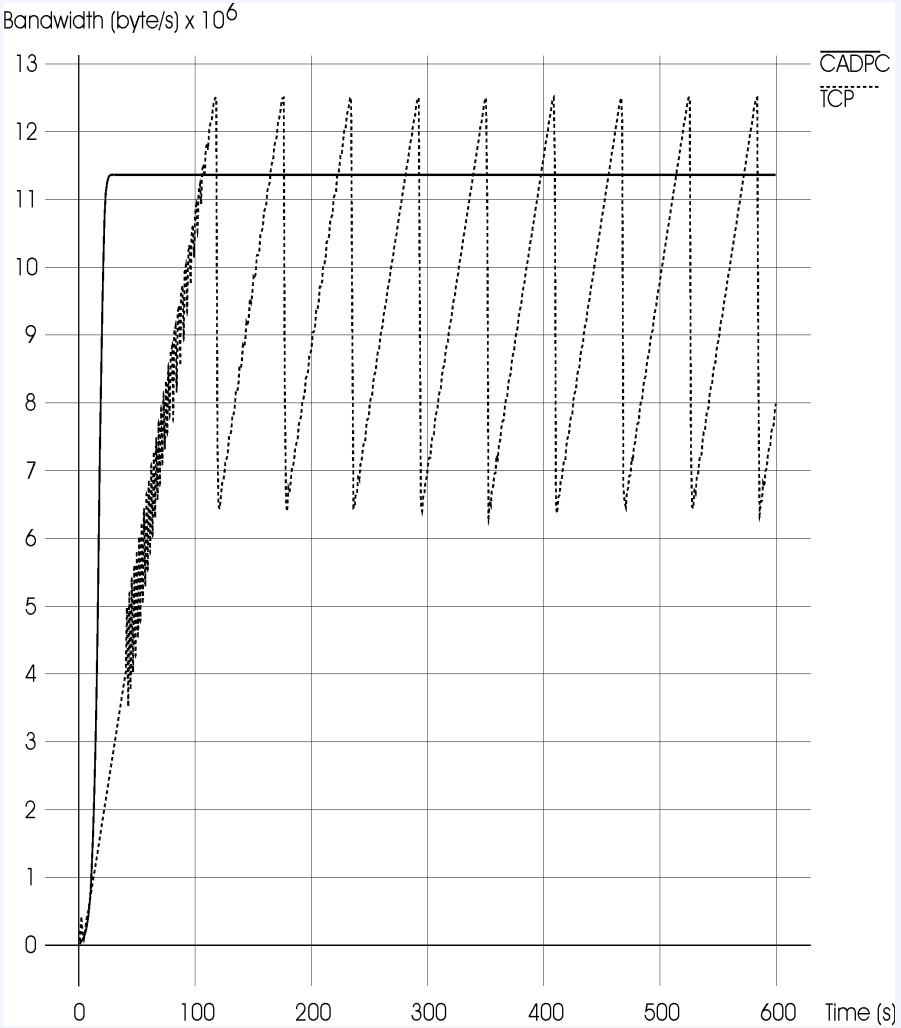
Michael Welzl, "Scalable Performance Signalling and Congestion Avoidance", Kluwer Academic Publishers 2003.

Unless otherwise mentioned...

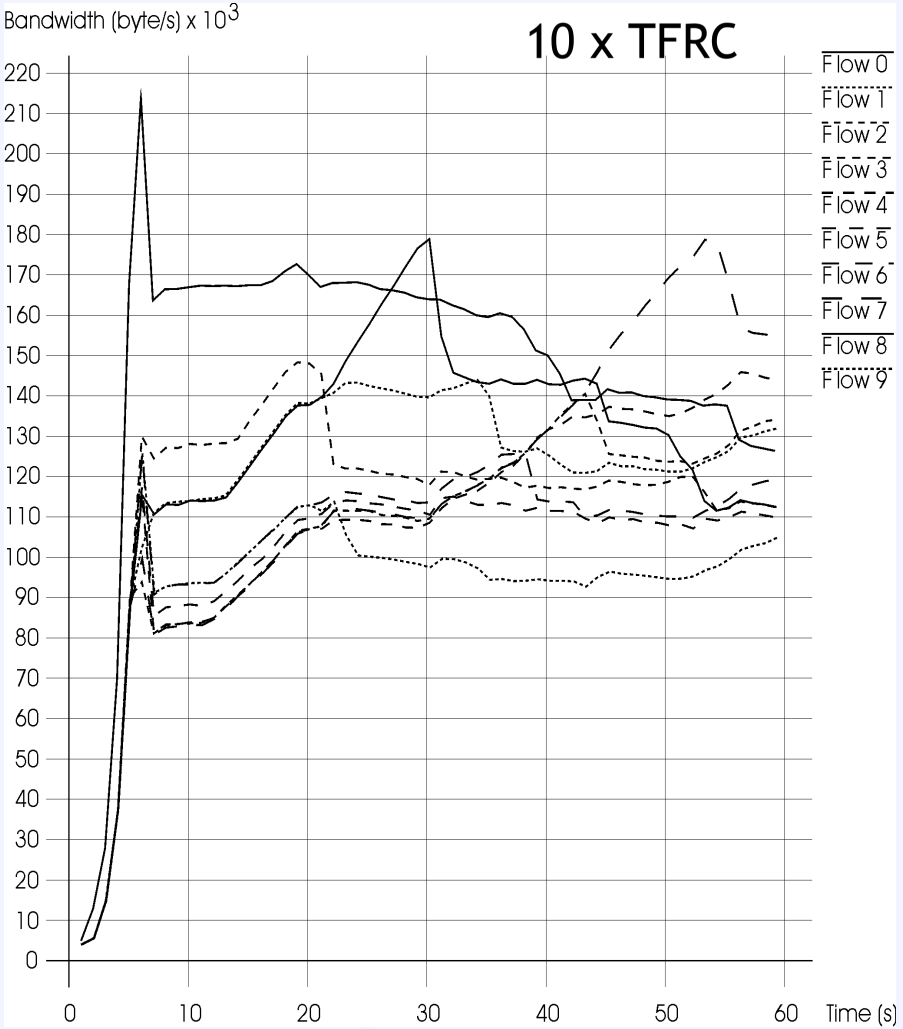
Topology	**Dumbbell**
CADPC update frequency	**4 RTTs**
CADPC smoothness factor a	**0.5**
Packet Sizes	1000 bytes
Bottleneck Link Bandwidth	10 Mbit/s
Bandwidth of all other links	1000 Mbit/s
Link delay	50 ms each
Queueing discipline	Drop-tail
Duration	Long-term: 160 seconds
All flows start after	0 seconds
Flow type	Greedy, long-lived
TCP flavour	TCP Reno
Bottleneck Queue Length	50 packets

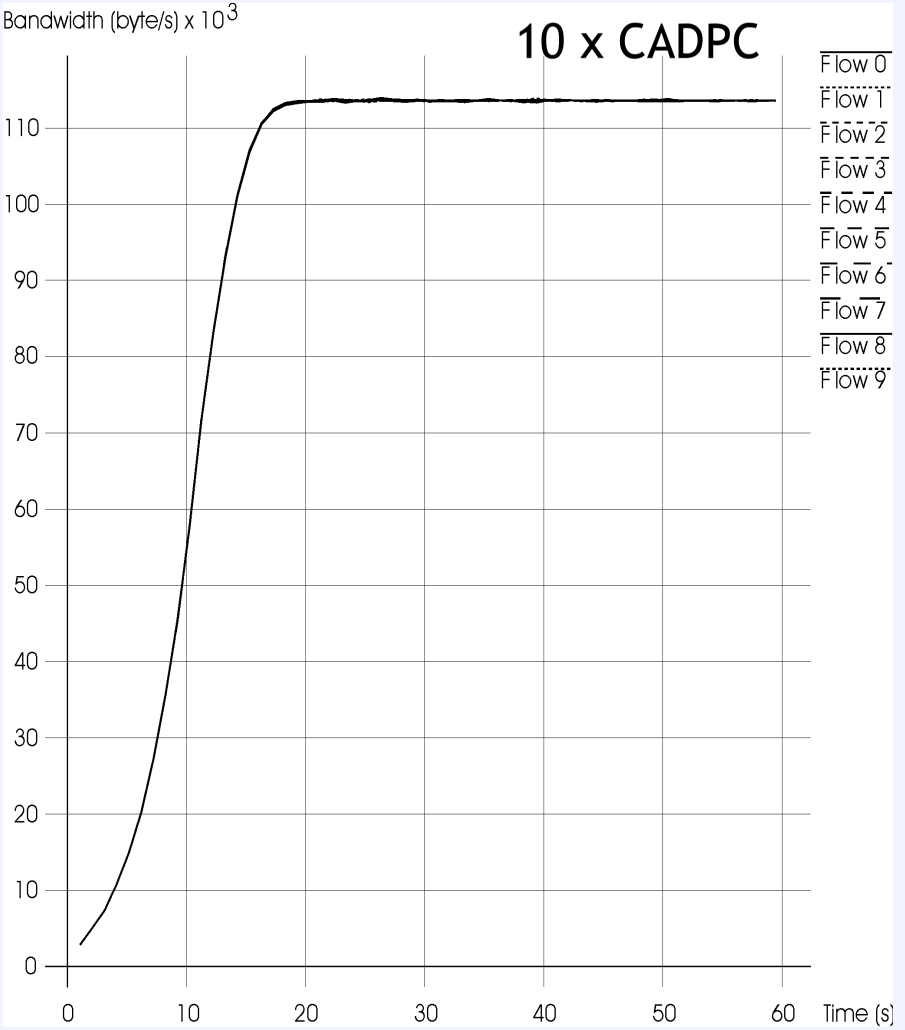
CADPC vs. TCP


Bandwidth (byte/s) x 10^6
1.3
1.2
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
10
20
30
40
50
60
Time (s)
CADPC
TCP

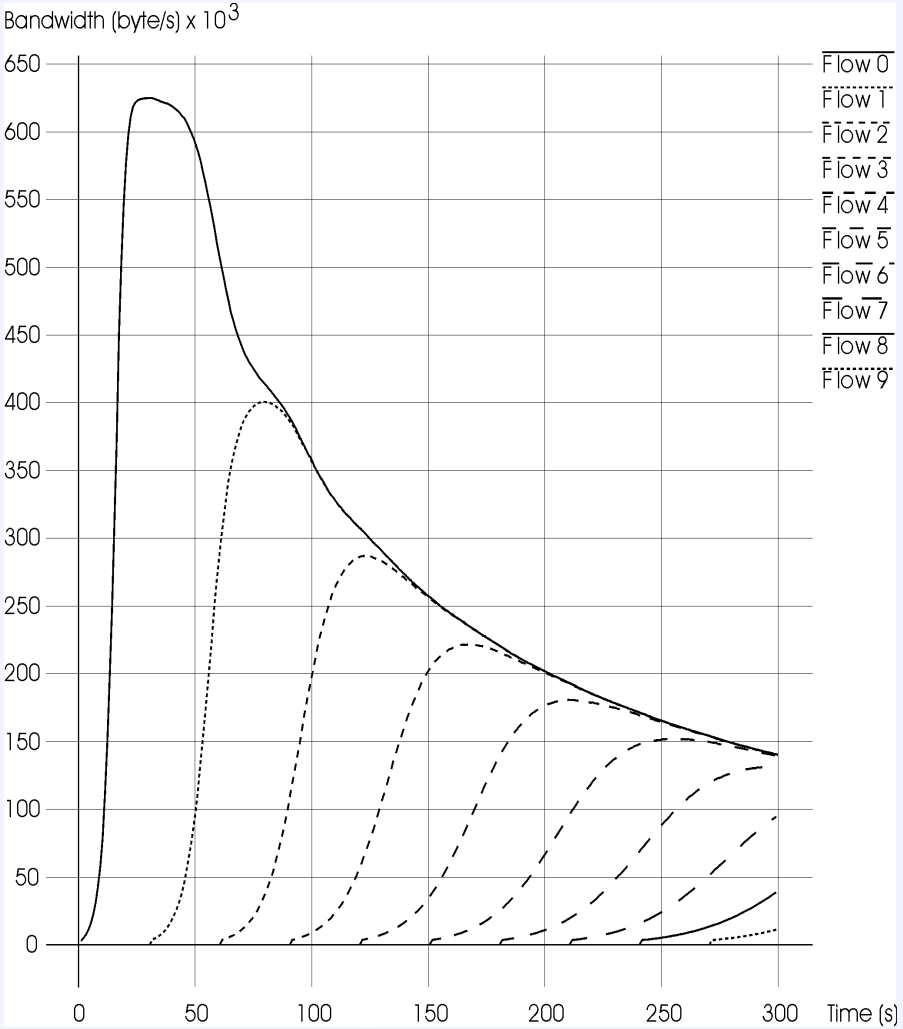

Bandwidth (byte/s) x 10^6
13
12
11
10
9
8
7
6
5
4
3
2
1
0
0
100
200
300
400
500
600
Time (s)
CADPC
TCP

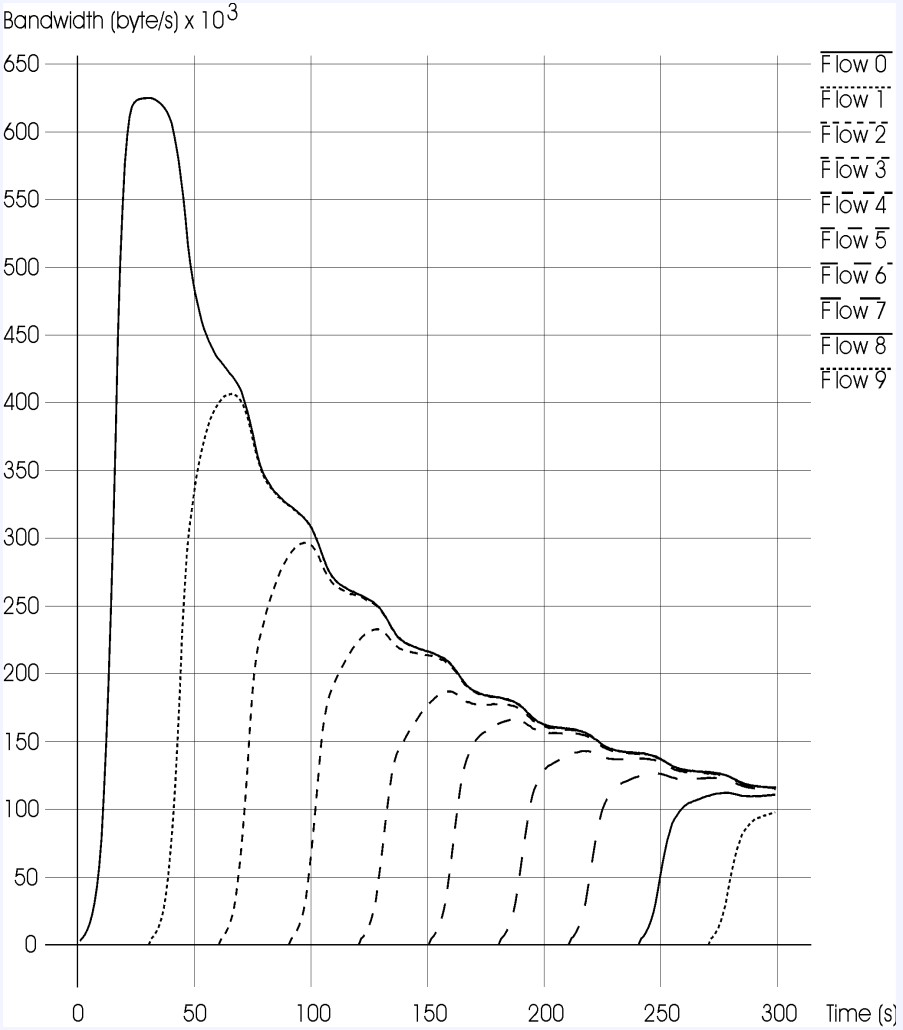
Smoothness


Bandwidth (byte/s) x 10^3
10 x TFRC
Flow 0
Flow 1
Flow 2
Flow 3
Flow 4
Flow 5
Flow 6
Flow 7
Flow 8
Flow 9
0
10
20
30
40
50
60
70
80
90
100
110
120
130
140
150
160
170
180
190
200
210
220
0
10
20
30
40
50
60
Time (s)

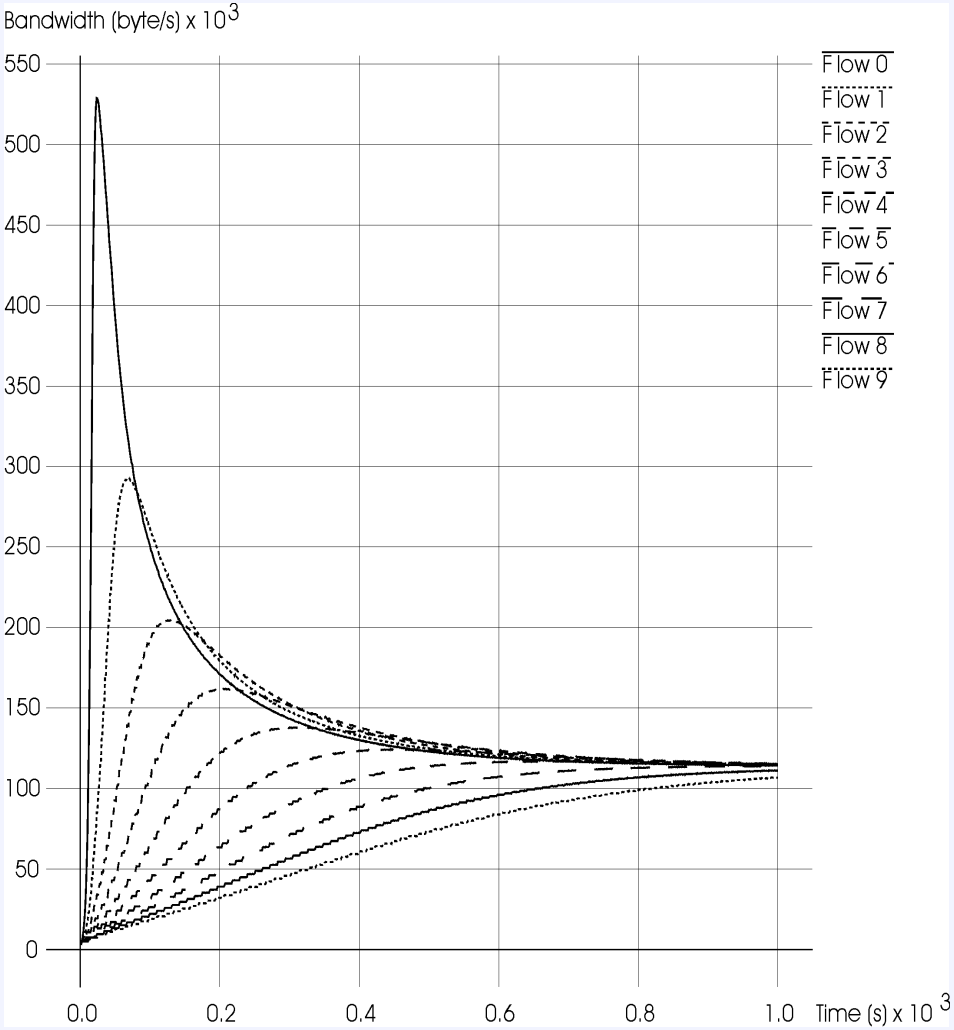

Bandwidth (byte/s) x 10^3
10 x CADPC
Flow 0
Flow 1
Flow 2
Flow 3
Flow 4
Flow 5
Flow 6
Flow 7
Flow 8
Flow 9
0
10
20
30
40
50
60
70
80
90
100
110
0
10
20
30
40
50
60
Time (s)

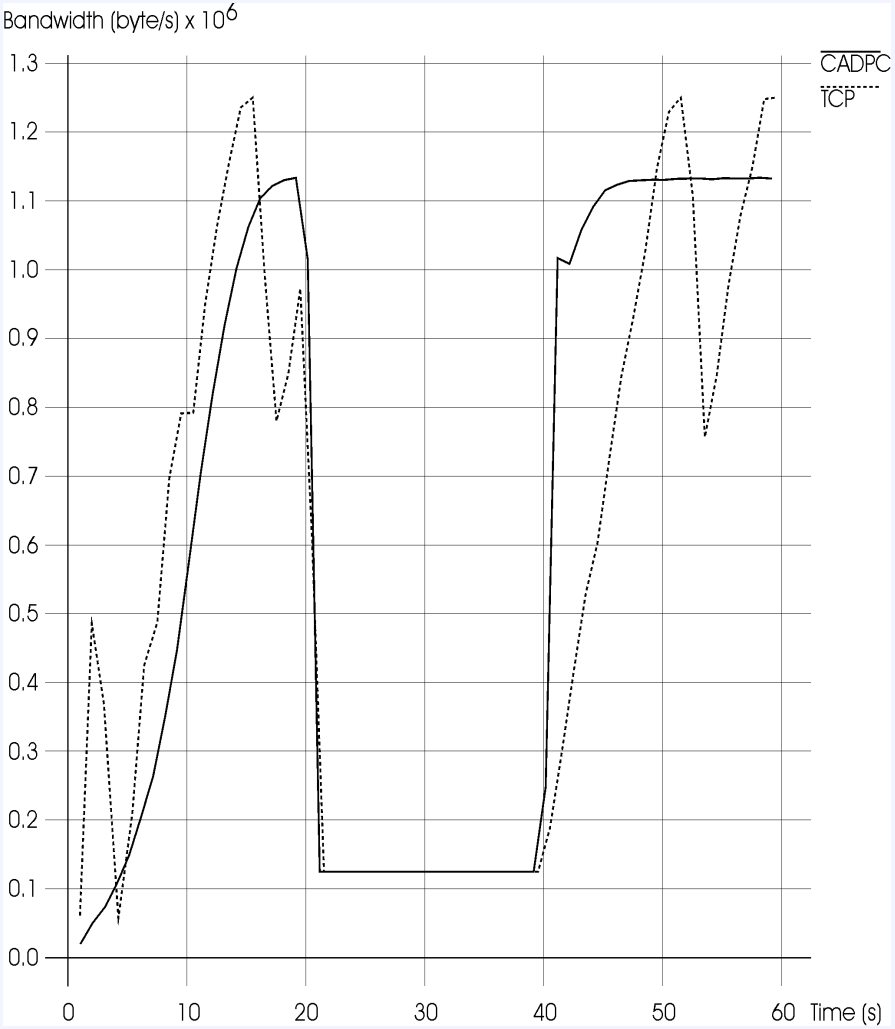
Startup enhancement



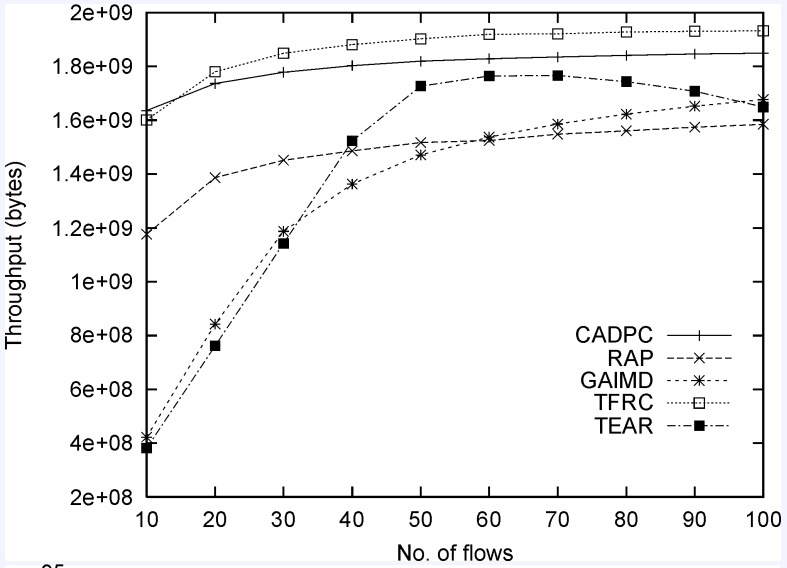


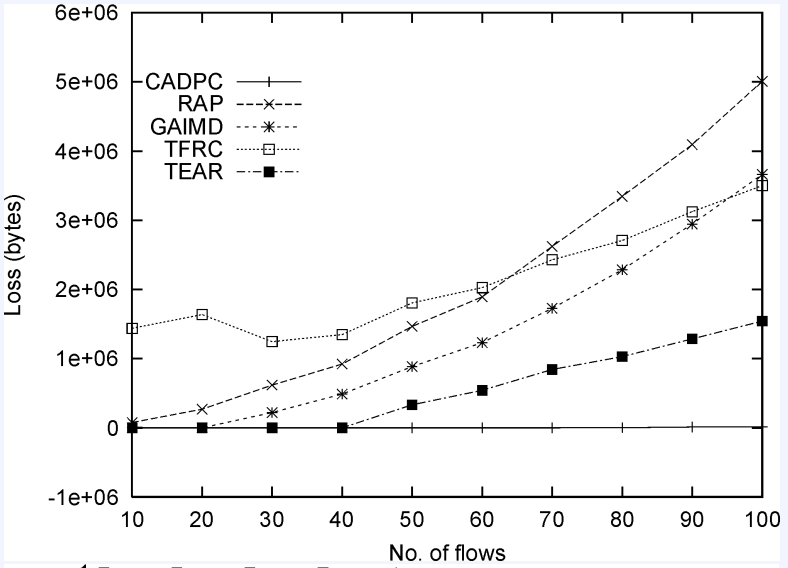
Heterogeneous RTTs + Robustness

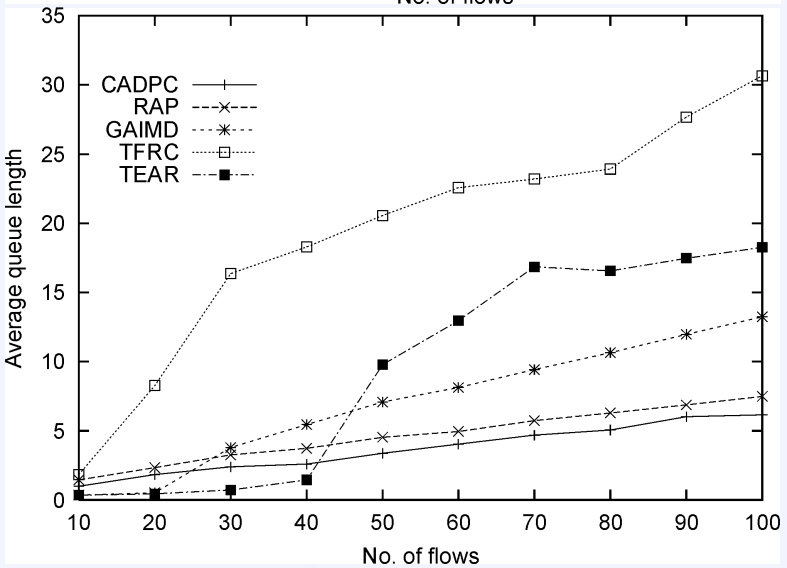


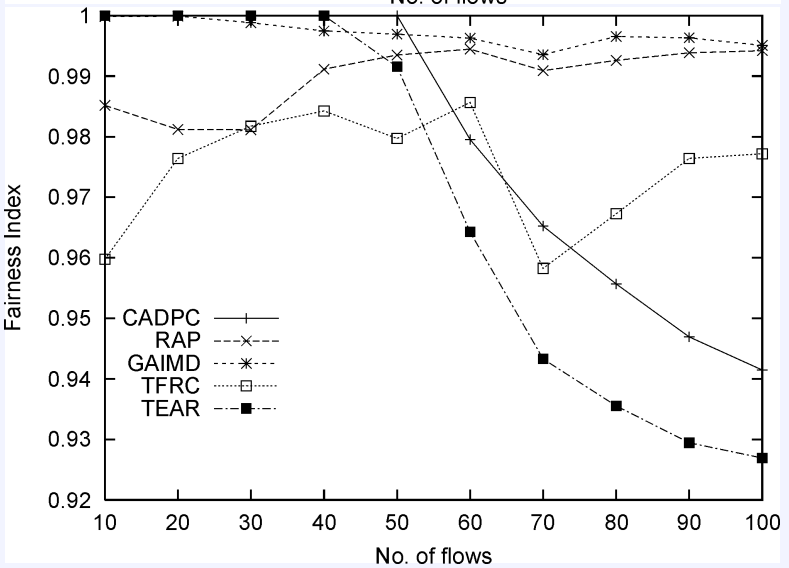


CADPC vs. TCP-friendly CC. mechanisms

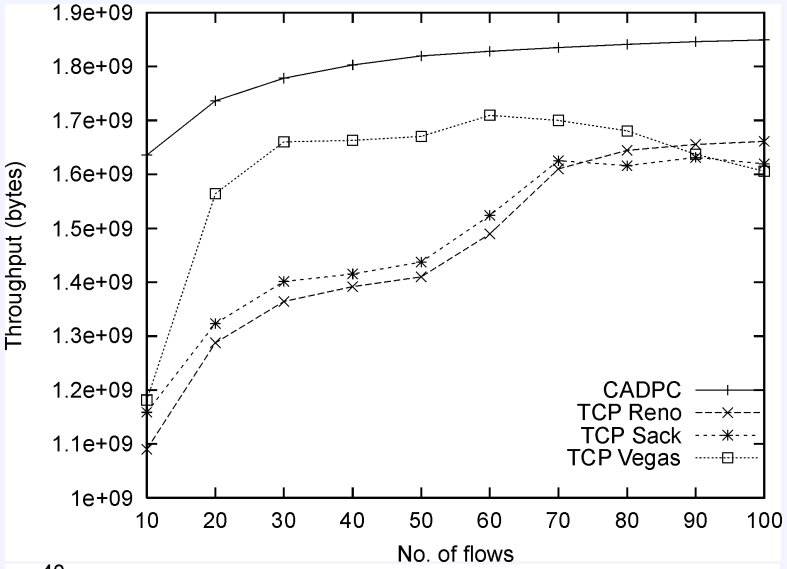


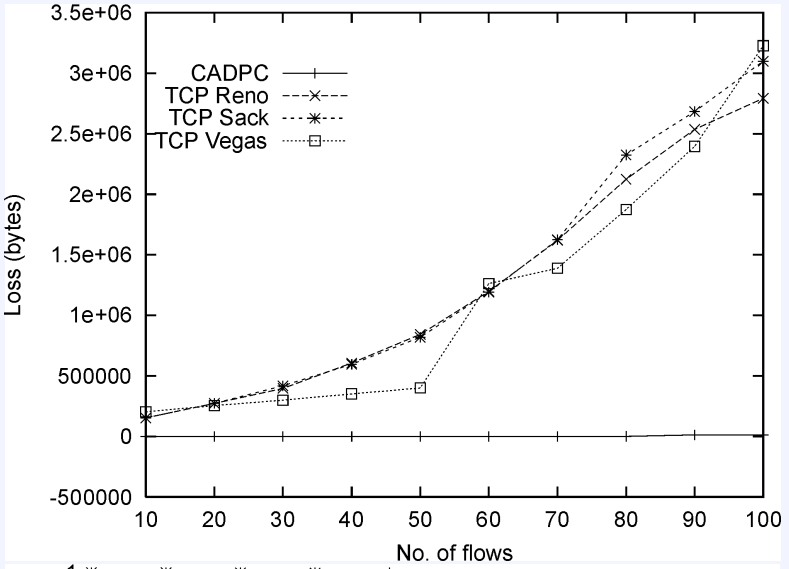


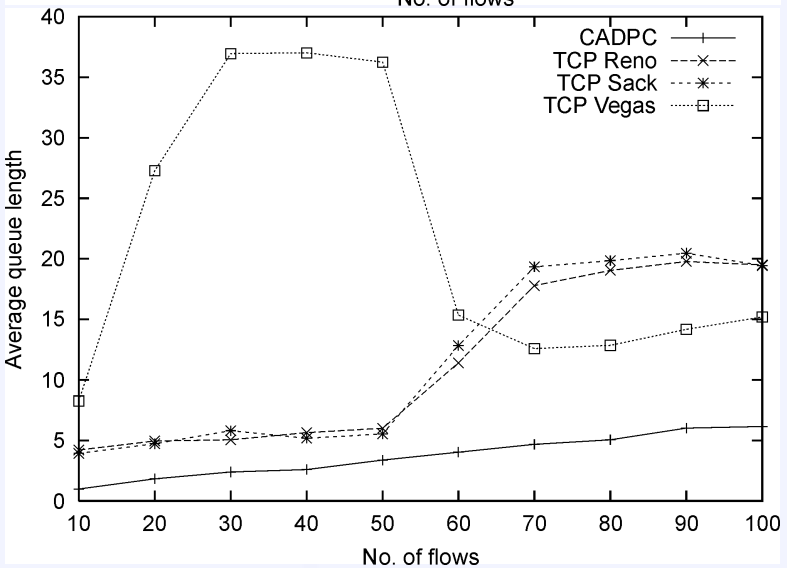


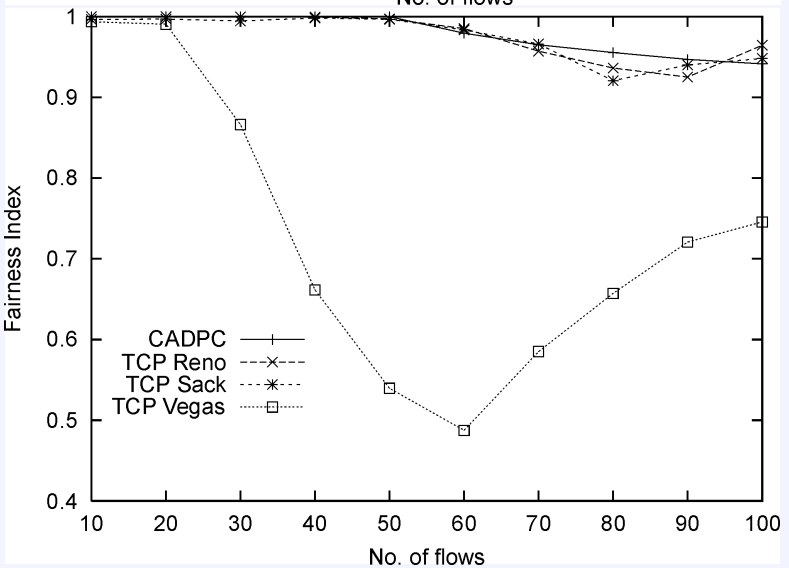


CADPC vs. 3 TCP(+ECN) flavors









CADPC Advantages

- high utilisation
- close to zero loss
- small bottleneck queue length
- very smooth rate
- fully distributed precise max-min-fairness

some say it's impossible :)

- rapid response to bandwidth changes (e.g. from routing)
- provable asymptotic stability (synchronous RTTs, fluid model)

CADPC Advantages /2

- Useful for asymmetric links
- Useful for noisy (wireless) links + "long fat pipes"
- Useful for QoS and load-based charging

Disadvantages

- Requires router support
- Requires traffic isolation because…
  - not TCP-friendly
  - slowly responsive: bad results with web traffic

CADPC/PTP + Wi-Fi Voice

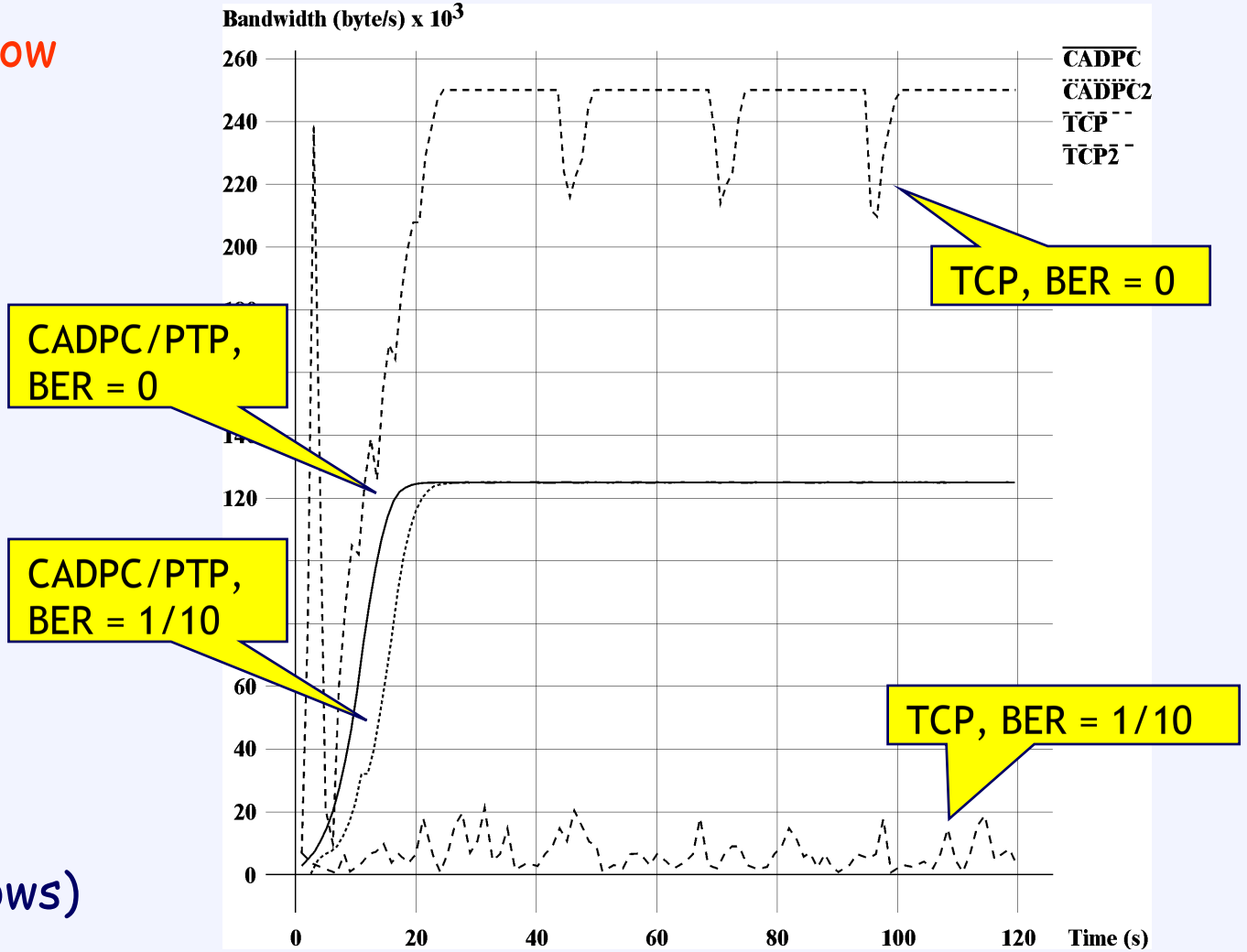
- Wi-Fi
  - Packet loss from link noise
    - misinterpreted as congestion by TCP (halve rate)
    - ignored by CADPC/PTP (only relies on byte counters)
  - Increased RTT estimate from link layer ARQ
    - more conservative behavior in TCP (less throughput)
    - ignored by CADPC/PTP (convergence RTT independent)
- VoIP
  - long-lived flows: a requirement for CADPC
  - QoS requirements: inherently supported by CADPC
  - small packets: make CADPC work even better (more precisely)

Link noise: simulation example

1 TCP vs. 1 CADPC flow

Remember:
CADPC converges to n/(n+1) !

(easy to change for known max. no of flows)



The End ...

- Publications
- CADPC+PTP ns code
- PTP Linux code (router kernel patch + end system implementation)
- Future updates

http://www.welzl.at/ptp

Note: I am eager for projects;
⇒ contact me!