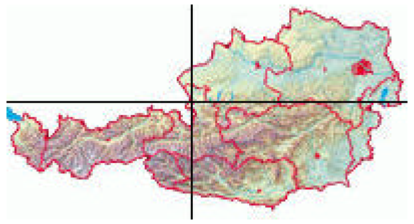
Simulations of a TCP rate controller in inter-domain scenarios

Peter Dorfinger

Christof Brandauer

Ulrich Hofmann



Salzburg Research

Polytechnical University of Salzburg (Fachhochschule)

Goal

- enable rate guarantees for long-lived TCP flows
- DiffServ „compatible“
  - per-flow action only at network ingress
- non-invasive
  - e.g. no packet header modifications

Application

- premium FTP
  - transfer of large files within guaranteed time
  - x-ray images, telemedicine
  - stored video
- streaming media
  - minimum average bandwidth requirement
- AQUILA QoS IP architecture
  - Premium Multi Media (PMM) network service

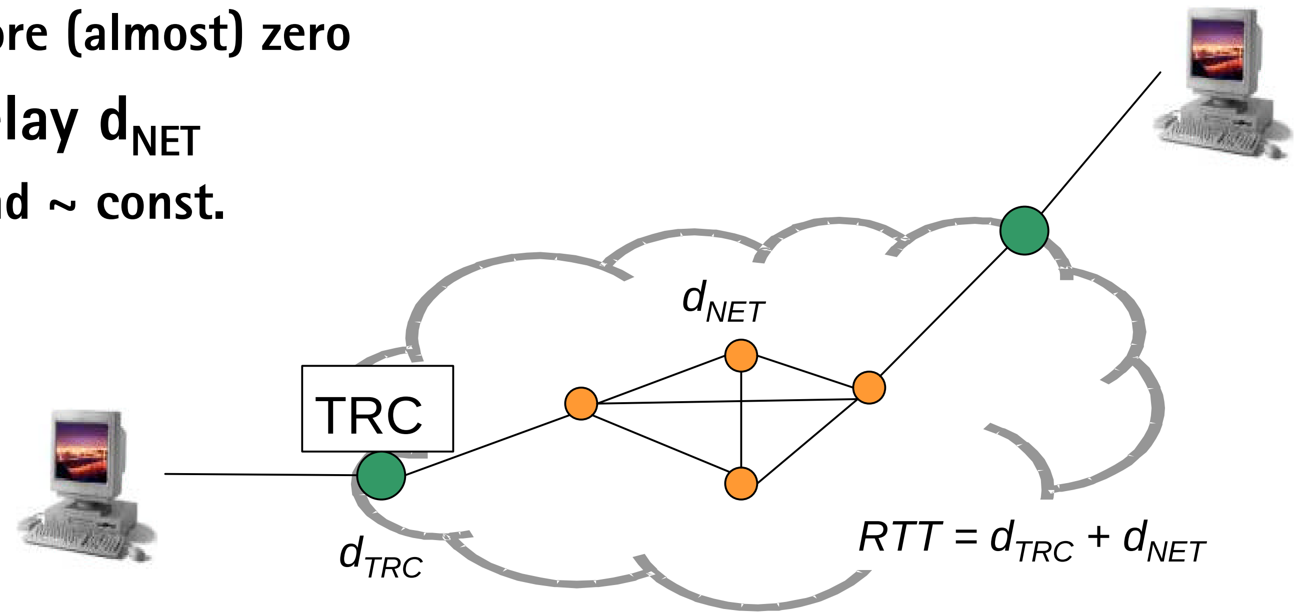
Basic idea

- model of TCP sending behavior
  - sending rate = f (p, RTT)

- exclusively control p and RTT
  - ability to trim TCP flow to a requested rate

Network model

- ingress: $\forall\ TCP\ flow\ \overset{1}{\exists}\ TRC$
- Admission control framework
  - p_{drop} in core (almost) zero
- network delay d_{NET}
  - known and ~ const.



TCP sending behavior

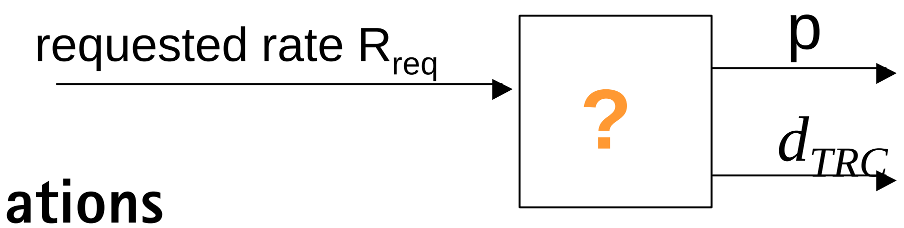
- existing TCP models, e.g.
  - *Padhye, Firoiu, Towsley, Kurose:* „Modeling TCP throughput: A Simple Model and its Empirical Validation", *ACM SIGCOMM 1998*
- simulation study using ns-2.1b6
  - TCP SACK
  - above model didn't fit very well
  - adapted model
    - accurately describes the TCP under investigation

TRC operation

```
for each packet arrival:
    if (packets since drop >= 1/p + E)
        drop packet
    else
        delay packet for d_TRC
```

- E: compensate drop
  - sending rate ® achieved rate
  - E = 1 : “drop“ mode
  - E = 0 : “ECN“mode

Choice of p and d_{TRC}



- >1 possible combinations
- Choice of p
  - accuracy of TCP model ~ 1/p
  - minimum window of 5 ® upper limit for p
    - 1 drop, 3 duplicate ACKs, 1 to trigger further tx in fast recovery
  - ® suggests small p
  - ® high d_{TRC}

Choice of p and d_{TRC}

- Choice of d_{TRC}
  - Error in d_{NET} estimation
    - leads to error in estimated TCP rate
    - impact depends on RTT/(RTT+dev)
    - limit impact by enforcing: RTT > RTTmin
    - ® reduces susceptibility to wrong d_{NET} estimation
  - ® suggests: high d_{TRC}
- Buffer requirements
  - ~ d_{TRC}
  - suggests small d_{TRC}

TRC tuning knobs

- dev_{NET}
  - estimation of the maximum deviation between d_{NET} and the real core network delay
- dev_{gput}
  - what error in the achieved rate should be compensated by the TRC
  - request := request * (1 + dev_{gput})

$$RTT_{\min} = \max\left(\frac{dev_{NET}}{dev_{gput}}, d_{NET}\right)$$

Parameter calculation

```
for each request:
    calculate RTT_req for p_0  (p_0 ≙ W_min)
    if RTT_req > RTT_min
        p ← p_0
    else
        RTT_req ← RTT_min
        calculate p
    endif
    d_TRC ← RTT_req − d_net
```

From intra-domain to inter-domain

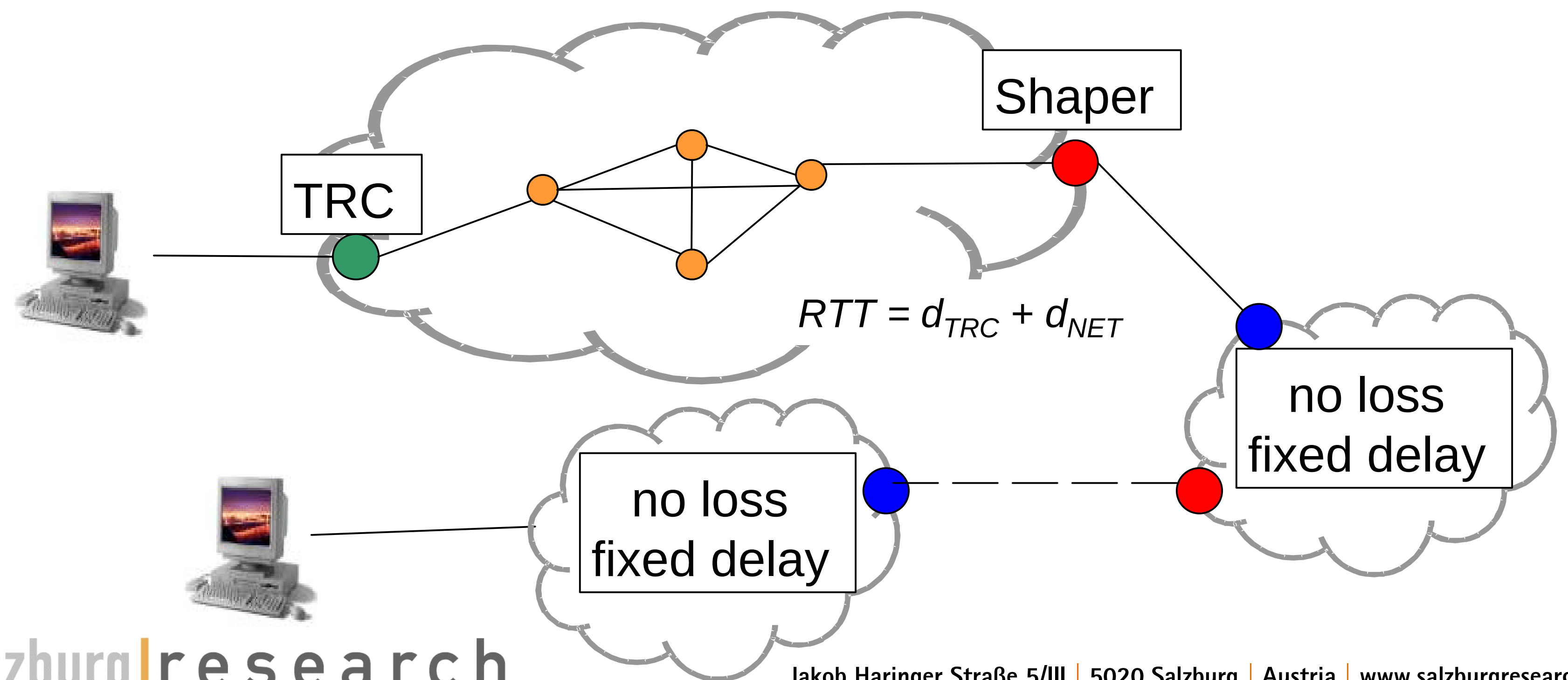
- TRC only in first domain
  - only aggregates in downstream domain
  - TCP flow is already rate controlled
- Requirements on downstream domains
  - no loss
    - TRC must exclusively control losses
  - constant and predefined delay
    - RTT must be known when initialising the TRC

Behavior of downstream domain

- request bandwidth
  - no loss
  - constant delay
  - aggregate of TRC controlled TCP flows
- ingress policer

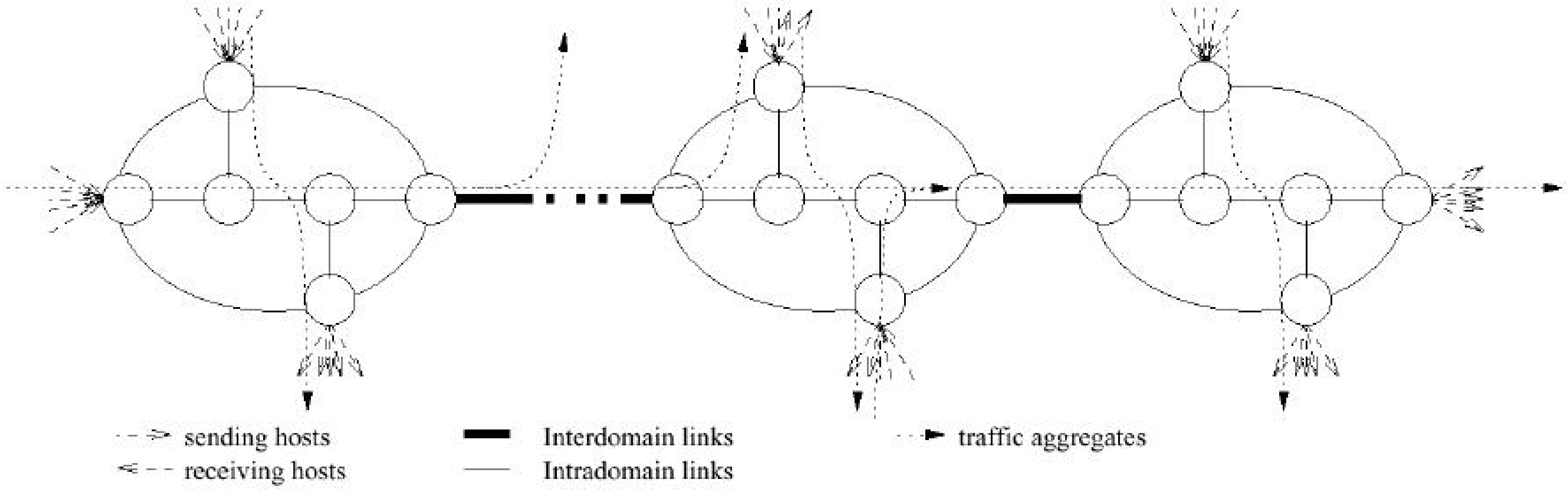
Improvements on TRC

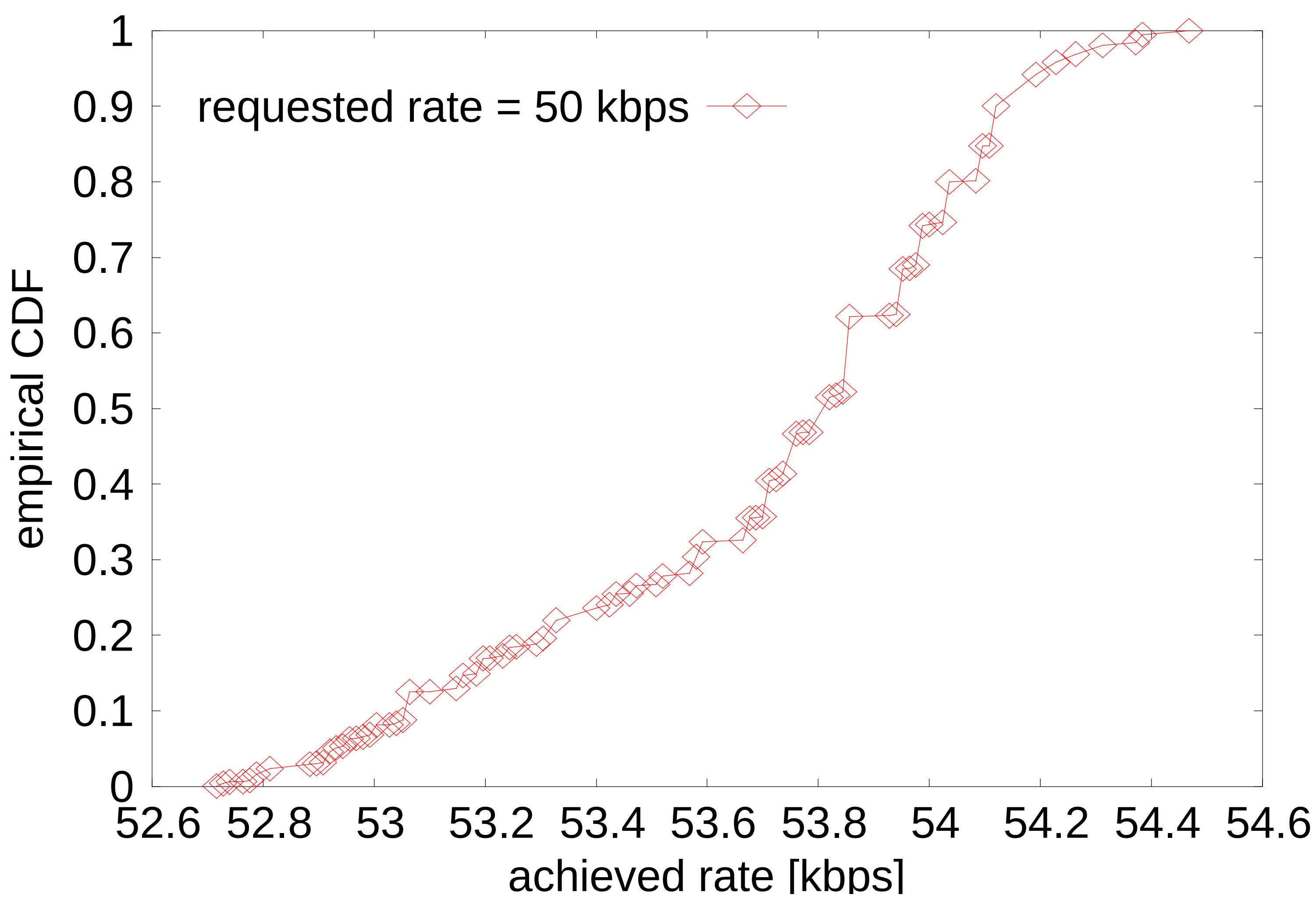
- **Traffic shaper at egress of domain**



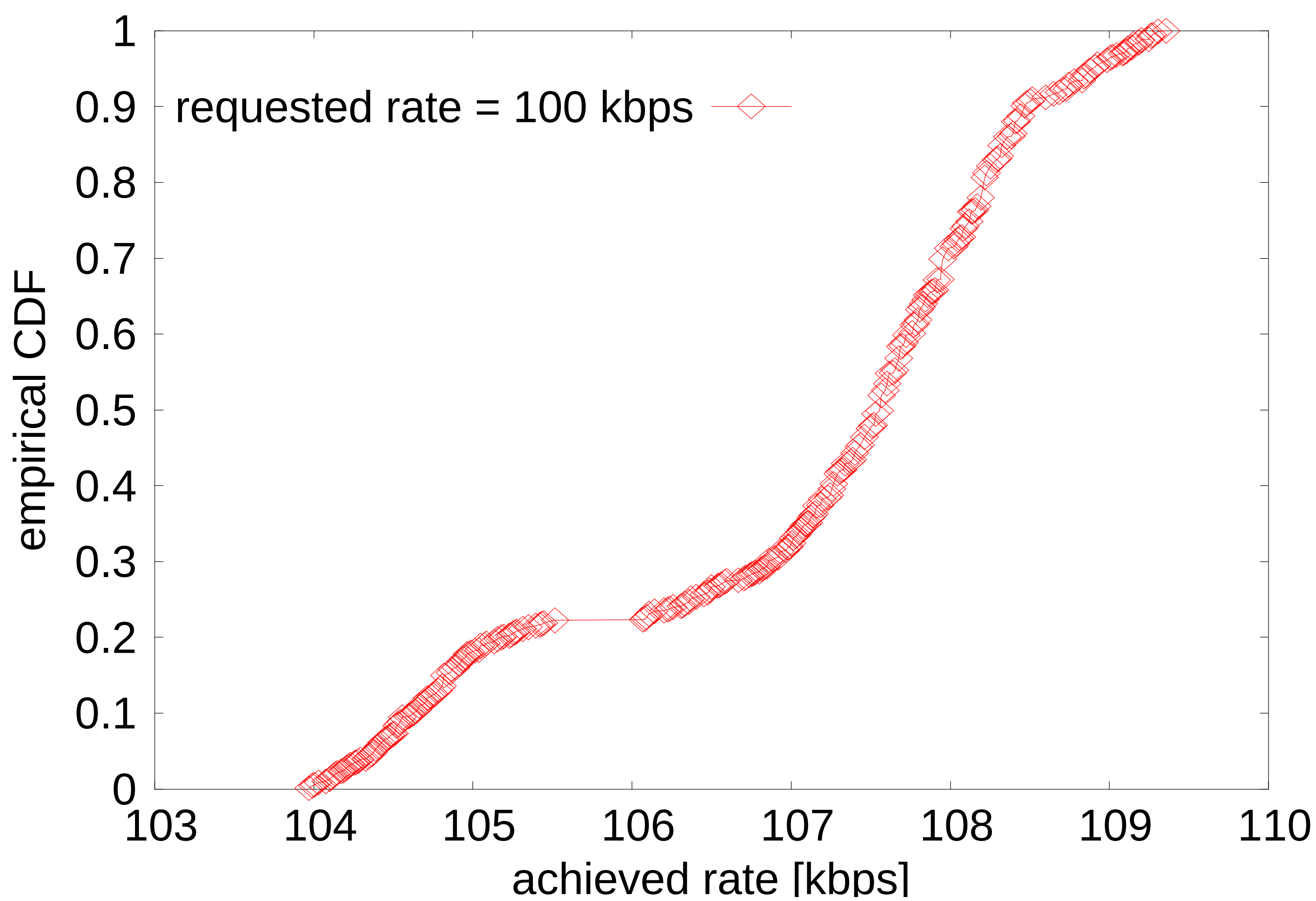
Simulation setup

- Aggregates of FTP flows with different requested rates
- DropTail queueing
- sim.time 1000 s
- 100 repetitions

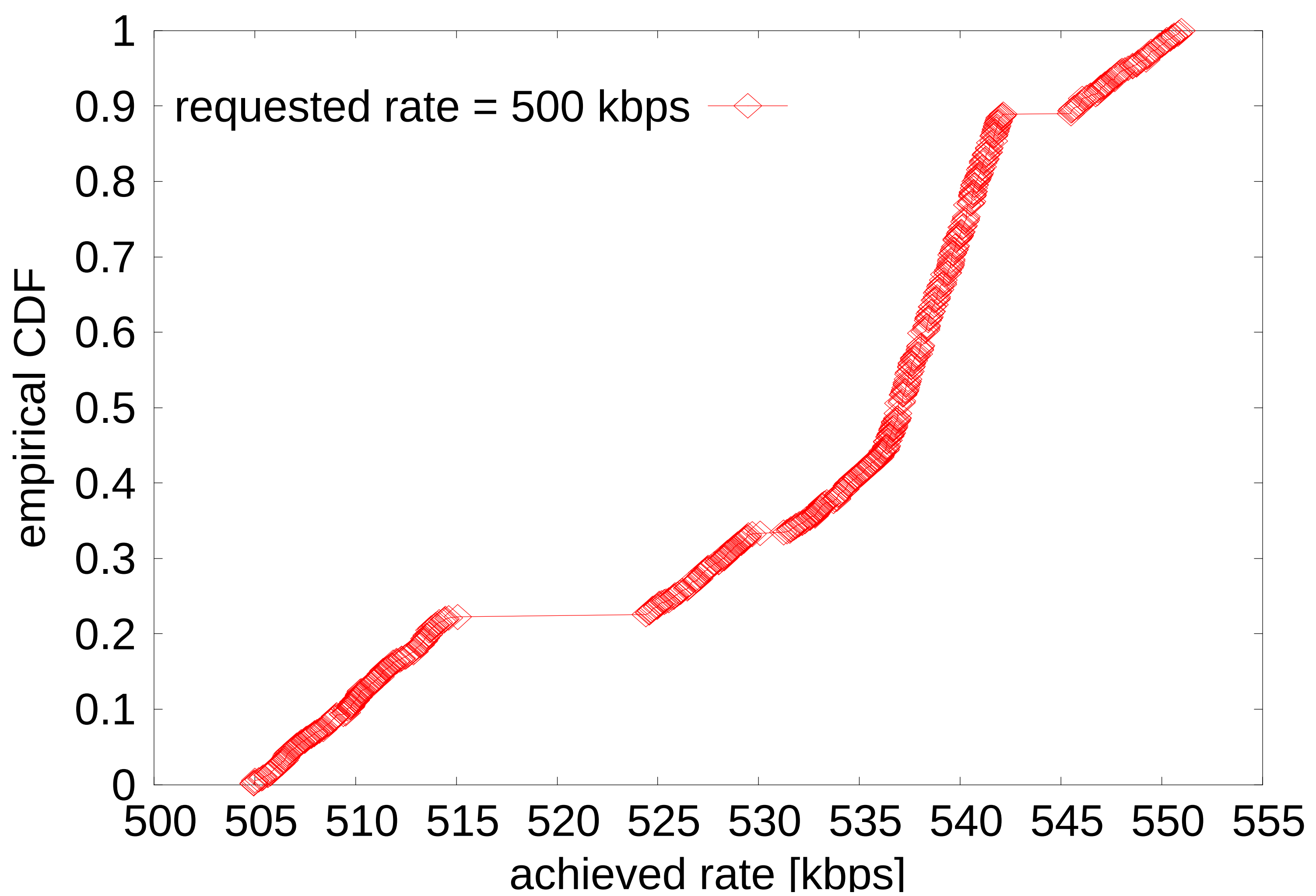




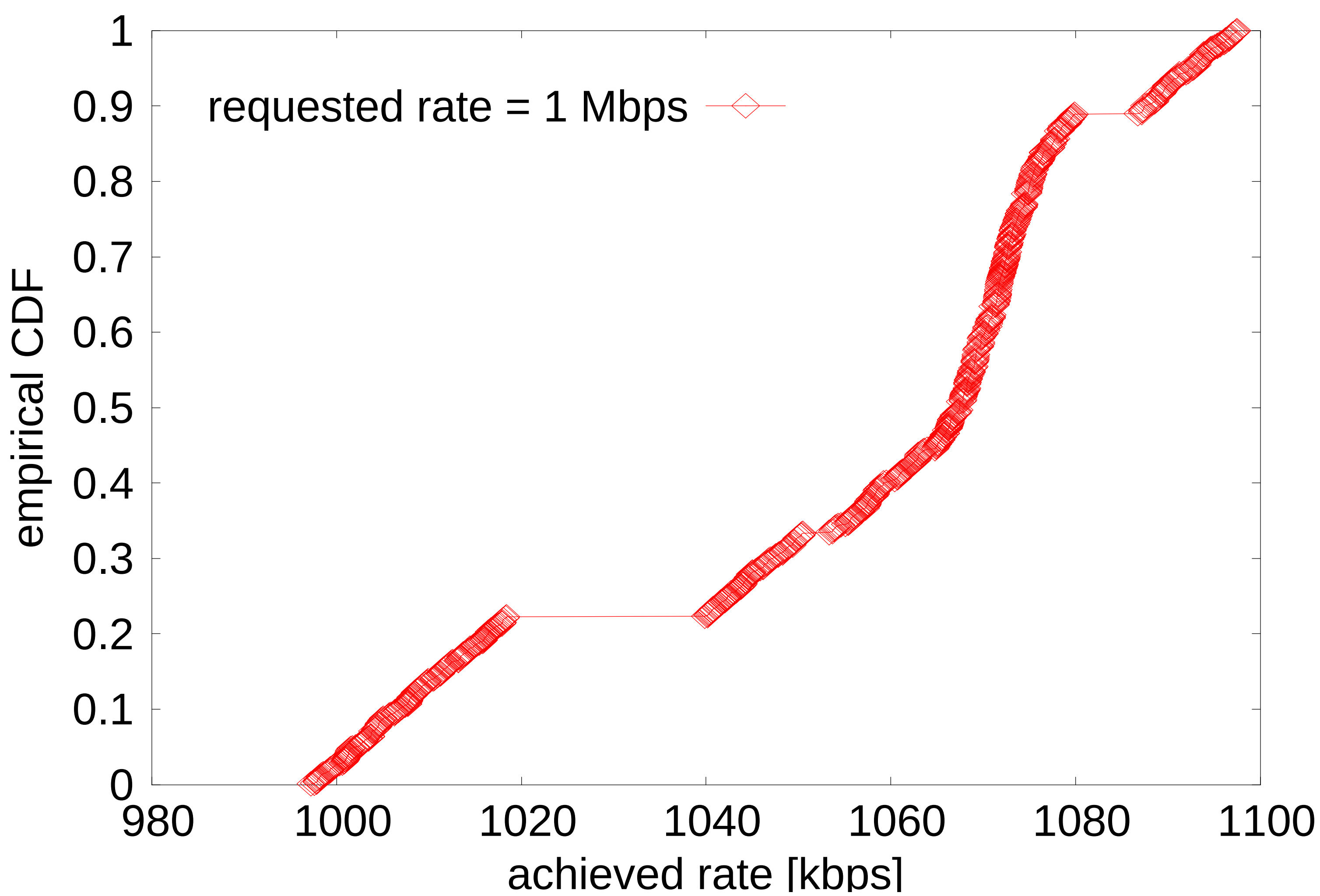
requested rate = 50 kbps
empirical CDF
achieved rate [kbps]
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
52.6
52.8
53
53.2
53.4
53.6
53.8
54
54.2
54.4
54.6



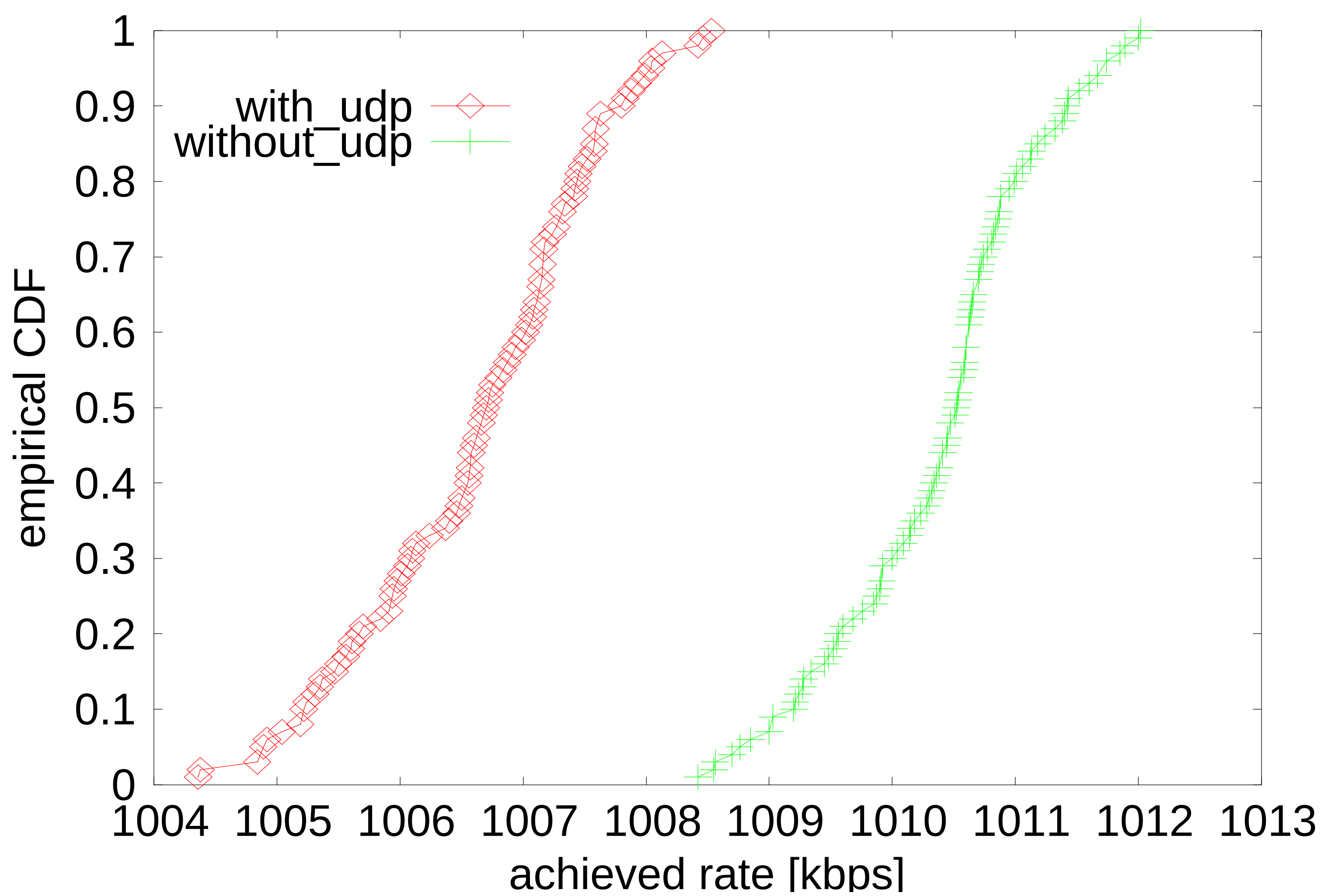
requested rate = 100 kbps
empirical CDF
achieved rate [kbps]
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
103
104
105
106
107
108
109
110



empirical CDF
requested rate = 500 kbps
achieved rate [kbps]
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
500
505
510
515
520
525
530
535
540
545
550
555



requested rate = 1 Mbps
empirical CDF
achieved rate [kbps]
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
980
1000
1020
1040
1060
1080
1100

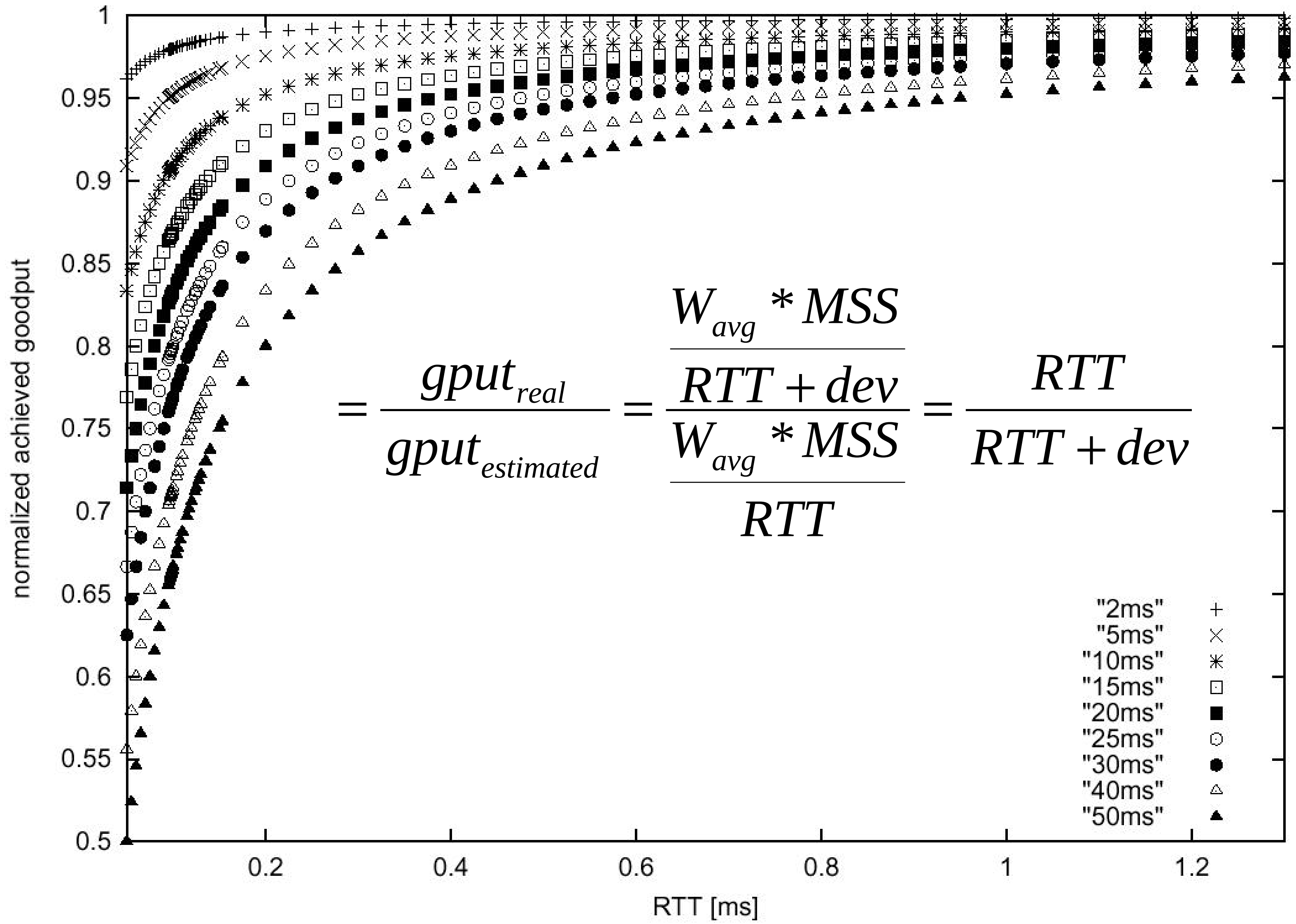


empirical CDF
achieved rate [kbps]
with_udp
without_udp
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013

Conclusion

- success
  - gput assurance for inter-domain connections
  - small deviation between requested and achieved rate
    - good service differentiation
  - TCP traffic can be mixed with udp traffic
- open issues
  - need measurements
  - accuracy of the TCP model ?
    - problem for all approaches based on TCP model
    - delayed ACKs

Questions?



$$= \frac{gput_{real}}{gput_{estimated}} = \frac{\dfrac{W_{avg} * MSS}{RTT + dev}}{\dfrac{W_{avg} * MSS}{RTT}} = \frac{RTT}{RTT + dev}$$