Chirping for Congestion Control

ICCRG – IETF80 Prag – March 31st, 2011

Mirja Kühlewind <mirja.kuehlewind@ikr.uni-stuttgart.de>
Bob Briscoe <bob.briscoe@BT.com>

IKR, University of Stuttgart, Germany

This work is partly funded by the German Research Foundation (DFG) through the Center of Excellence (SFB) 627 "Nexus" and by Trilogy, a research project (ICT-216372) supported by the European Community under its 7th Framework Programme.

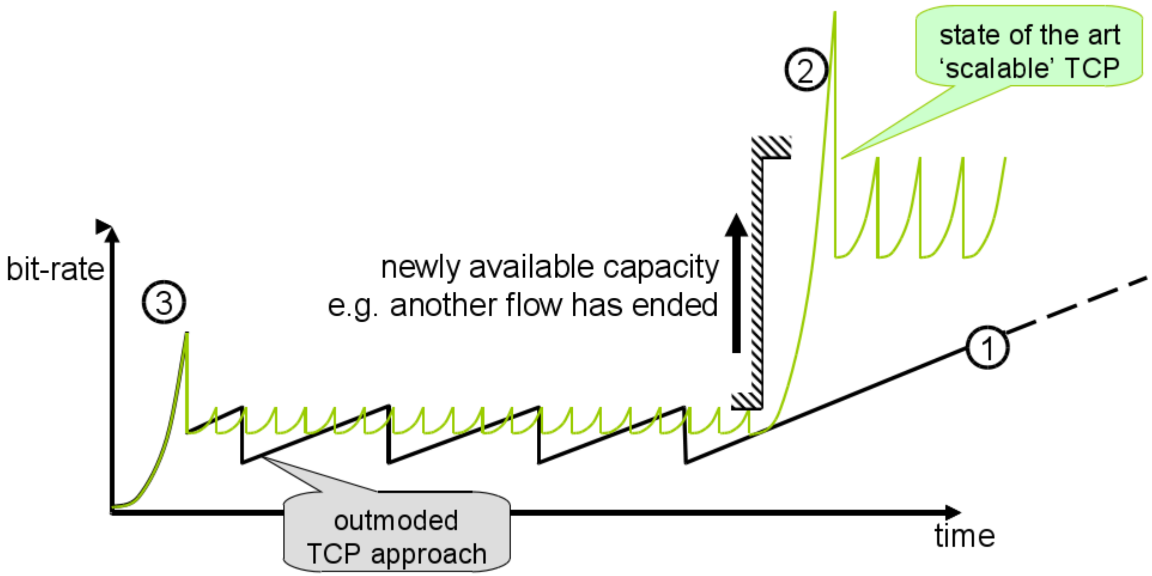
Overview

- Motivation
- Chirping as a Building Block for Congestion Control
- Research Challenges
- Conclusion and Outlook

Motivation

Scaling Problem

1. Original TCP acquires new bandwidth too slowly
2. State-of-the-art approaches overshoot instead
3. Overshoot causes a lot unnecessary congestion

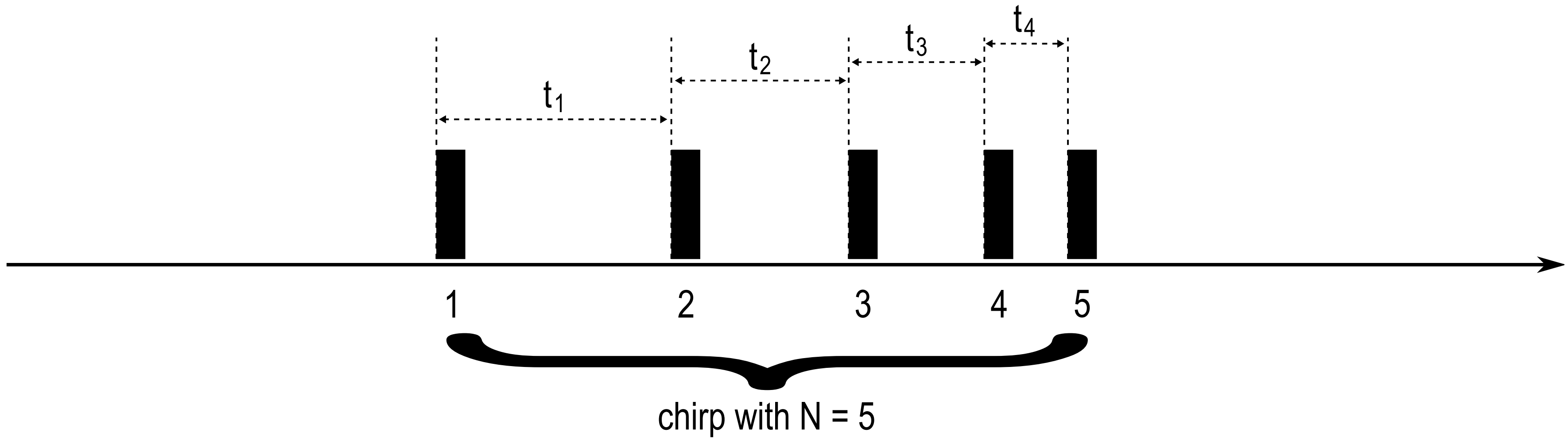


- Do we need to update the interface between host & network?
  → Prior to discovering chirping, we thought we did, but not yet conclusive.
- Chirping provides an estimation about the available bandwidth (fast feedback)
  → Probing for a wide range of bit-rates with minimal harm to others (without overshoot)

Chirping Principle

Chirp: A group of several packets with decreasing inter-packet gaps and increasing rate

– Proposed by pathChirp bandwidth estimation tool [1]



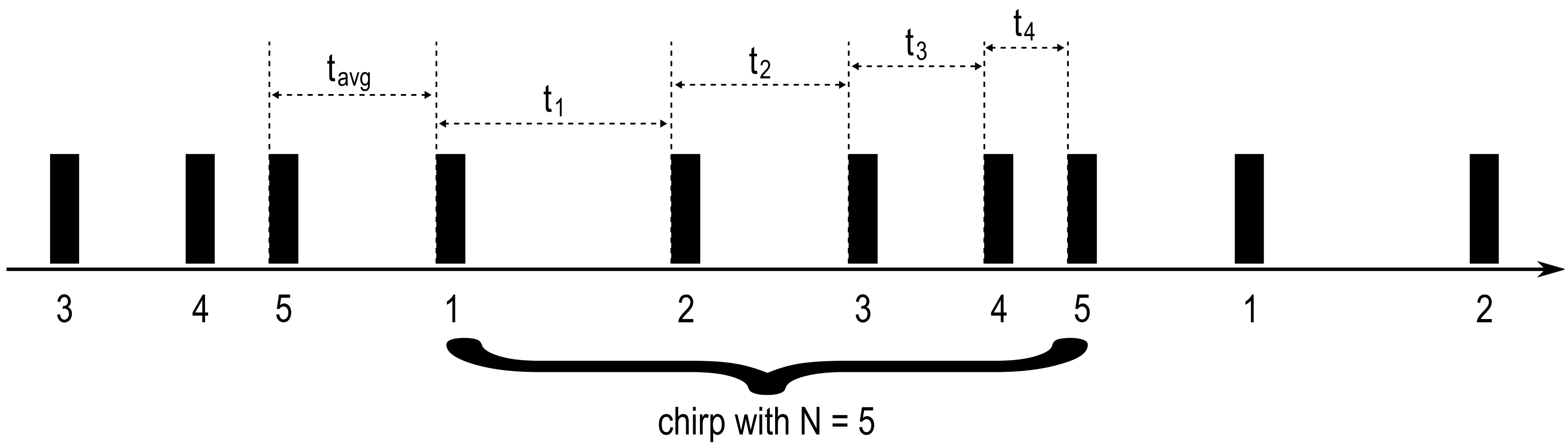
- Bandwidth estimation based on self-induced congestion
- Feedback for monitoring of one-way delay

[1] V. Ribeiro, R. Riedi, R. Baraniuk, J. Navratil and L. Cottrell. "pathChirp: Efficient Available Bandwidth Estimation for Network Paths". Passive and Active Measurement Workshop 2003

Chirping as a Building Block for Congestion Control

Chirping for Congestion Control: Continuous transmission of data packets as chirps

– proposed by RAPID congestion control [2]



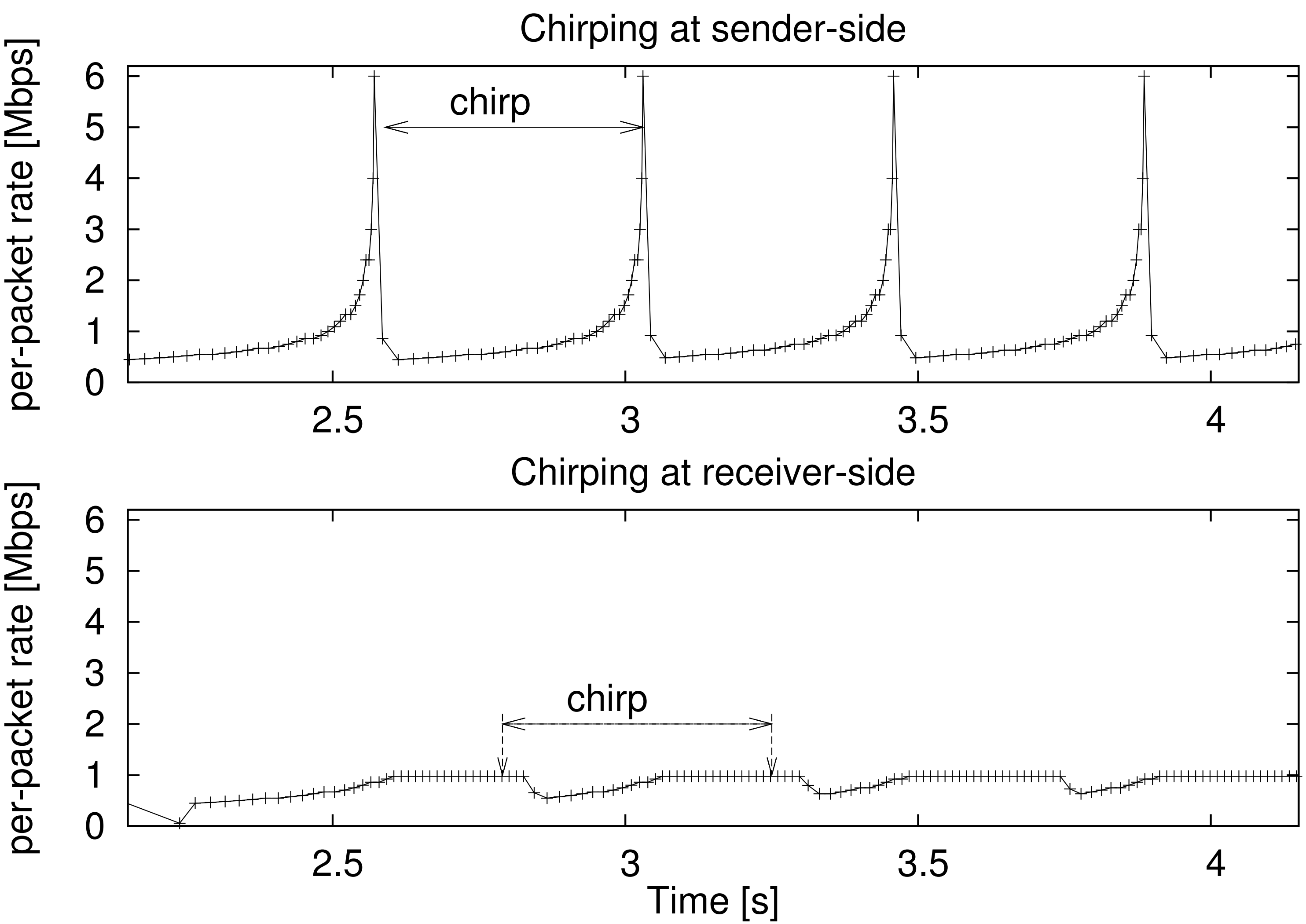
- Average rate r_{avg} should equal intended sending rate of congestion control
- Actual per-packet rates are lower and higher than r_{avg}

  → Probing for a wide range of possible sending rates but still limited impact of probing on other flows

[2] V. Konda and J. Kaur. "RAPID: Shrinking the Congestion-Control Timescale". In IEEE INFOCOM 2009

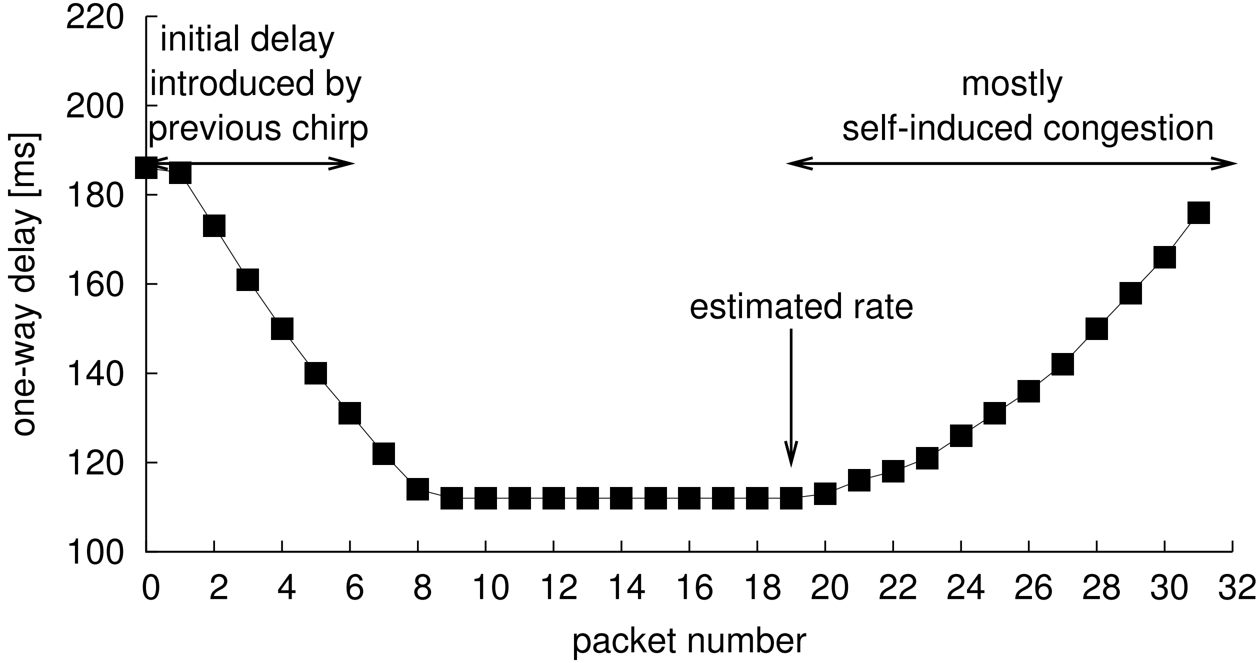
Chirping Implementation

Per-Packet rate of one chirping connection with **N=32** on **1Mbit/s** bottleneck link



Bandwidth Estimation based on relative OWD

Bandwidth estimation: Monitoring of the relative queuing delays of one chirp



- Growth in queuing delay between packets: $\triangle q_n = q_n - q_{n-1}$
  → Increasing values at the end of reveals available capacity (*self-induced congestion*)

OWD with cross-traffic implications

Excursion: Temporary increase in delay due to cross traffic

- Bandwidth estimation heuristics used (provided by pathChirp)

Chirping Implementation in the Linux Kernel

- Implementation in the Linux kernel version 2.6.26 (current version 2.6.38)
  - Rate-based approach and timer-based sent-out to realize inter-packet gaps
  - Usage of the kernel code in a simulation environment
- Framework separates
  - Rate estimation: Estimation of the available bandwidth r_{est} (pathChirp)
  - Rate adaption: Decision on new r_{avg} (RAPID: $r_{avg} = r_{est} \rightarrow$ scavenger)
  - Inter-packet gap calculation: Harmonic progression of rates
- Feedback based on TCP Timestamp Option (by default enabled in most OSs)
  - Every packet gets a time-stamp TSval assigned at sent-out
  - Receiver will echo this TSval and provide an own time-stamp TSecr on sent-out of the acknowledgement
  - One-Way-Delay: OWD = TSval - TSecr
  - Currently no one-ended deployment (because of delayed ACKs)

Chirping Implementation in the Linux Kernel

Sender-side Delay Measurement based on TCP Timestamp Option

One-way delay measurement based on TCP Timestamp Option

```
+--------+--------+---------------------+---------------------+
|Kind=8  |   10   |    TS Value (TSval)  |TS Echo Reply (TSecr)|
+--------+--------+---------------------+---------------------+
     1        1              4                     4
```

→ Option header includes echoed timestamp of data packet and ACK timestamp

→ One-way delay estimate: q = TSecr - TSval

→ Monitoring of relative increase in OWD within one chirp: $\triangle q_n = q_n - q_{n-1}$

Challenges

- TCP Timestamp Option does not ensure certain resolution (add. negotiation needed)
- Feedback needs to be assigned to one specific packet in a chirp (delayed ACKs?)
- Accuracy of time-stamping at send-out of data packet and ACK
  - Additional delay on network device (hardware timestamping)
  - Improved accuracy by use of the actual sending time gaps (reconstructed from the TCP TS Option) as long as the inter-packet gaps are getting smaller within one chirp

Research Challenges (1)

1. Processing overhead because of interrupt handling for sent-out timers
   - Threaded interrupts
   - Possibility of hardware support for timing and time-stamping

2. Additional delays on the network device/in the OS of a real system (e.g. delayed ACKs, TCP Segmentation Offload)

   Real-world testbed with current kernel version

3. Limitations in timestamp resolution and computational restrictions for algorithms
   - hrtimers in the Linux kernel provide currently nanosecond resolution
   - that’s enough to serve high-speed links

4. Additional negotiation for TCP Timestamp Option (draft-scheffenegger-tcpm-timestamp-negotiation)
   - about timesamp resolution
   - to reassign right timestamp to the right chirp

Research Challenges (2)

5. Interdependencies with a large number of chirping senders
   - Accuracy of measurement with a large aggregation of probing chirps
   - Impact of short term probing delays on the queue burstiness
   - Influence of a large aggregation of probing chirps on the base queue length

     → Reduced overshoot and respectively reduced maximum queue length

6. Adaption of chirping parameters to prevailing conditions (inter-packet gap calculation)
   - smaller number of packet per chirp for low mean sending rate
   - variation of probing range
   - arrangement of probing rates depending on previous estimation

Conclusion and Outlook

Design of a robust congestion control based on chirping

- (If it works) bandwidth estimation is a valuable information; more than just ’there is congestion’ or ’there is no congestion’ as today loss/delay measurements do
- Fast feedback chirping information only in addition to other network state information
- Converenge in capacity sharing also when competing with other protocols
  - RAPID is scavenger protocol: Not designed to take capacity share from loss-based protocols
- The transport layer needs to have mechanism to adapt to the different networks/ network conditions and not the other way around!
- Chirping information can be used to avoid large overshoots

Conclusion

- Use faster feedback to enable **more scalable rate adaption with minimal overshoot**!
- Do we need to update the interface between host & network?

  → Prior to discovering chirping, we thought we did, but not yet conclusive.

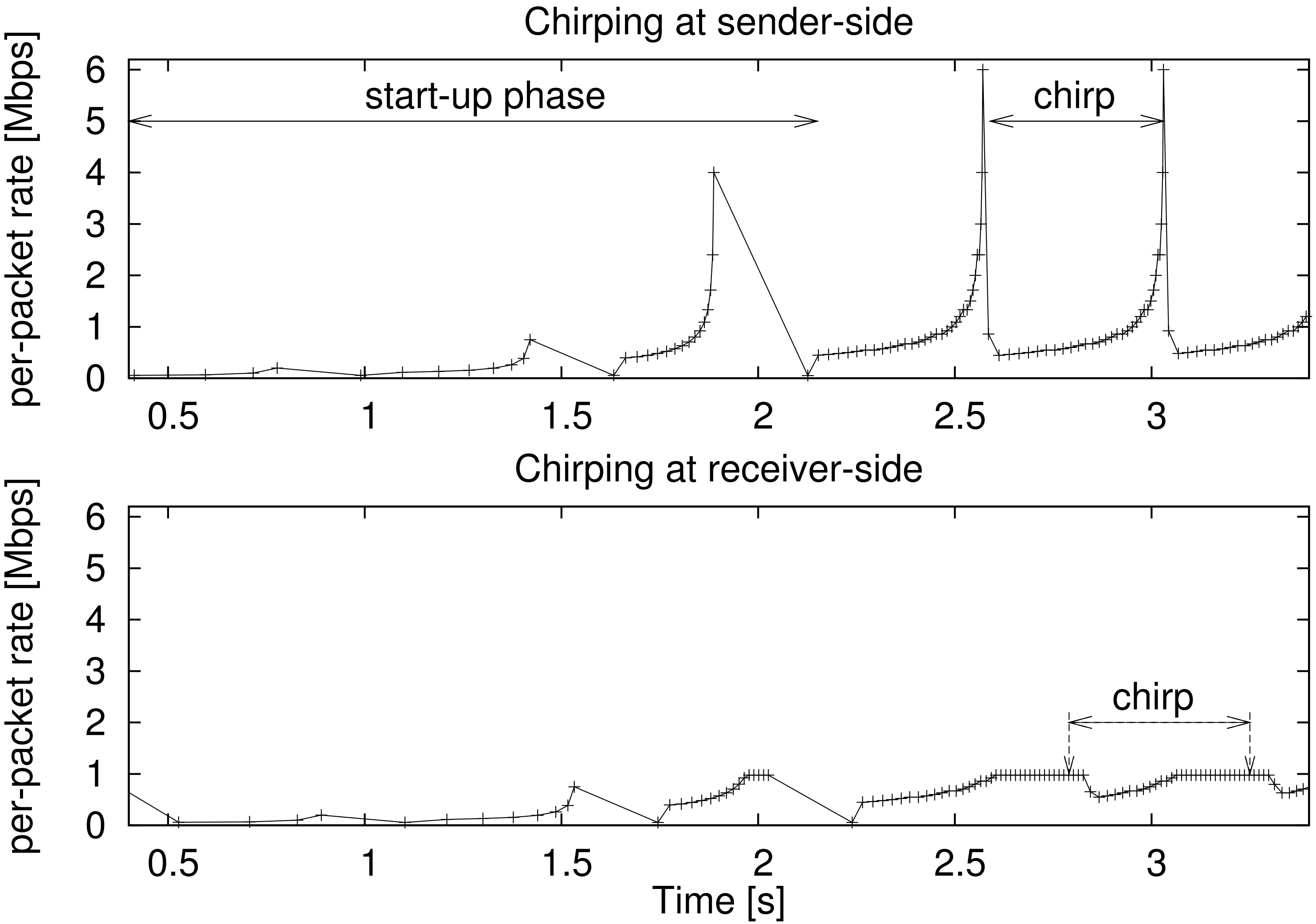
Thank you for your attention!

Questions?

Chirping

Preliminary Results

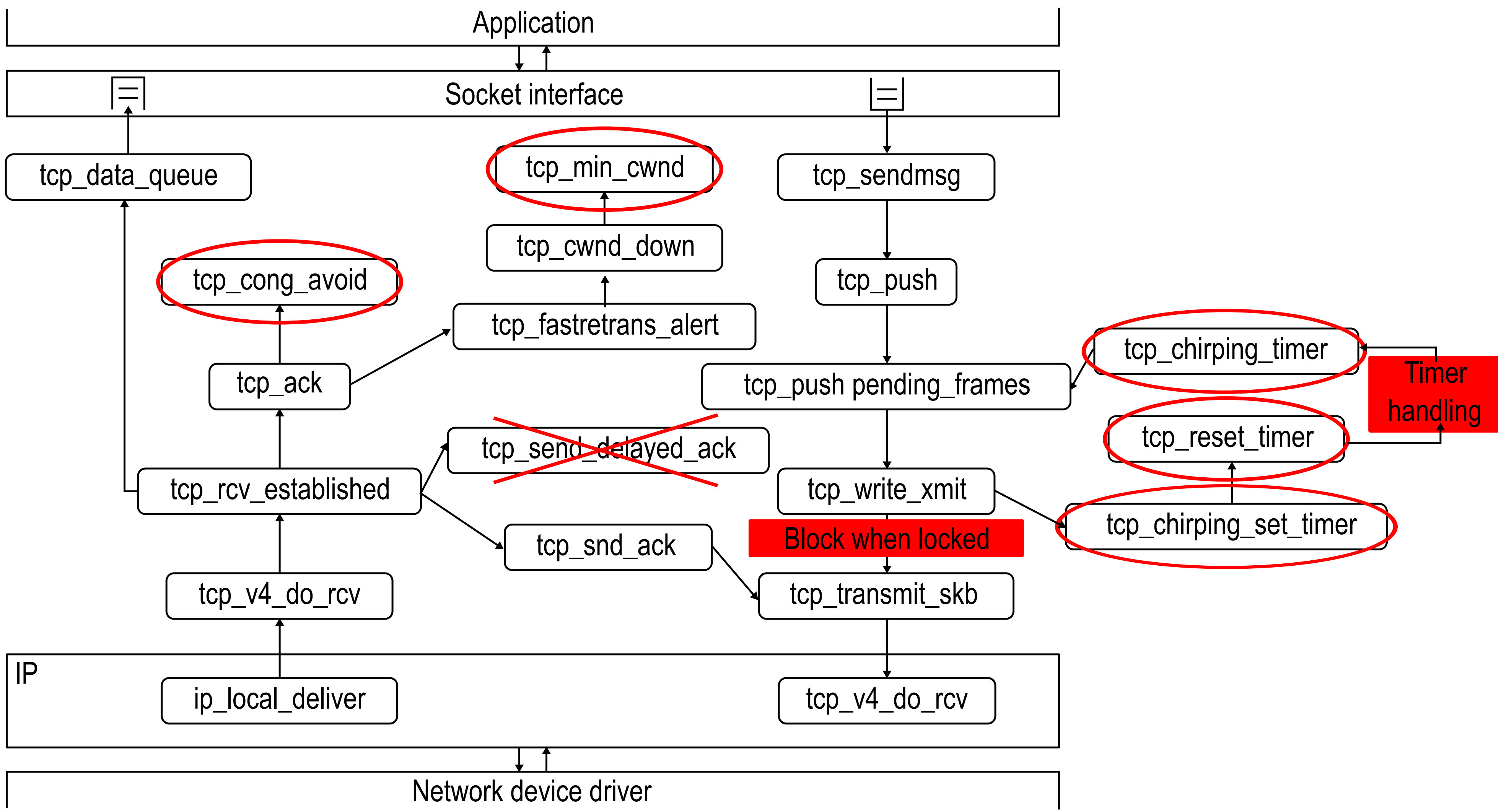
Per-Packet rate of one chirping connection on 1Mbit/s bottleneck link



Chirping Implementation in the Linux Kernel

Implementation Details

→ Extended congestion control kernel module interface and TCP timer for send-out timing



Chirping Implementation in the Linux Kernel

Algorithm for Inter-packet gap Calculation

- Fully based on inter-packet time gaps instead of rate
- N should be an the integer power of 2

  $\rightarrow$ Initiallly hard-coded to N = 32 (=2^5)
- Harmonic progression of rates by linear decrease of inter-packet gaps

  $\rightarrow$ Linear decrease of inter-packet gaps: $gap_i = gap_{i-1} - gap_{step}$ with $gap_{step} = (2 * gap_{avg}) / N$
- Implementation with integer arithmetic

$$gap_i = gap_{step} * (N - i + 1) = (2 * gap_{avg}) / N * (N - i + 1) \qquad \text{with } i = 1...N-1;\ gap_0 = gap_{avg}$$

  – Probing range:1/2 r_{avg} to N/4 r_{avg}
  – Maximum rates of harmonic progression not used

$\rightarrow$ Slightly lower average rate than the estimated one