On Redesigning the Unequipped Ship: Load Transient Awareness and AQM Algorithms

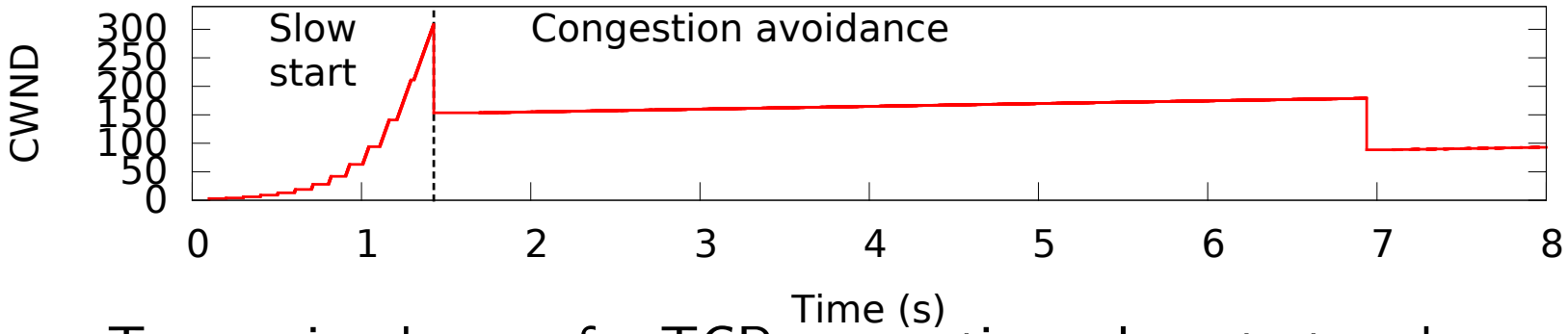
Ilpo Järvinen and Markku Kojo
ilpo.jarvinen@helsinki.fi, markku.kojo@cs.helsinki.fi

Department of Computer Science
University of Helsinki

iccrg @ IETF-97
November 2016

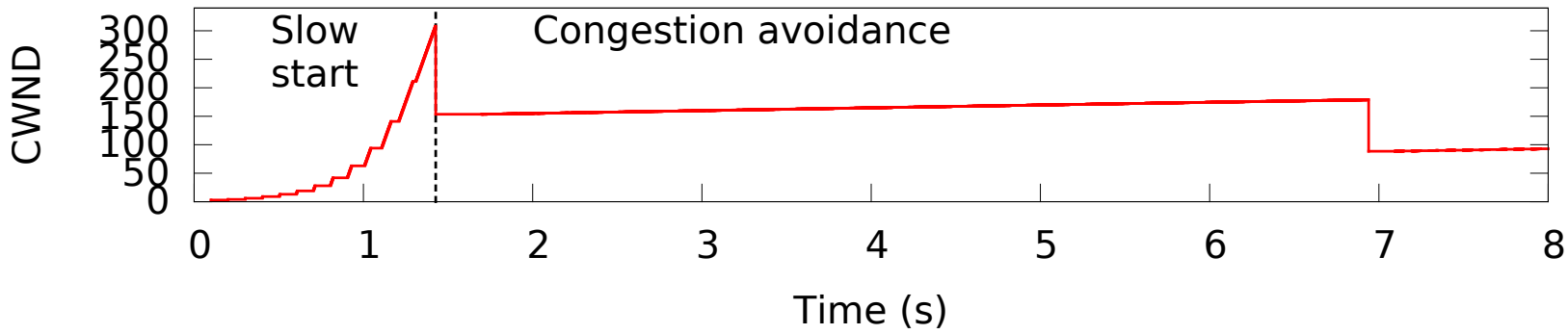
Outline

Motivation: A Storm and Calm Waters



- Two main phases of a TCP connection: slow start and congestion avoidance
- Congestion avoidance resembles calm waters
- Slow start is like a storm
  - The storm gets worse all the time if AQM does not respond
- Slow start is frequent!
  - ON-OFF traffic (e.g., Web traffic), whenever a flow starts
  - Causes exponential load transients at least near the network edge
    - If enough link capacity to the core exists, significant transients possible also deeper in the core
  - Seems natural to design for it
  - While we refer to "TCP slow start" here, this work applies to other exponential self-clocked bandwidth probing too

Motivation: Past Load Transient AQM Work



- Unfortunately, AQM algorithm control loops typically designed congestion avoidance in mind
- We and others have shown that AQM algorithms have issues with load transients*

---

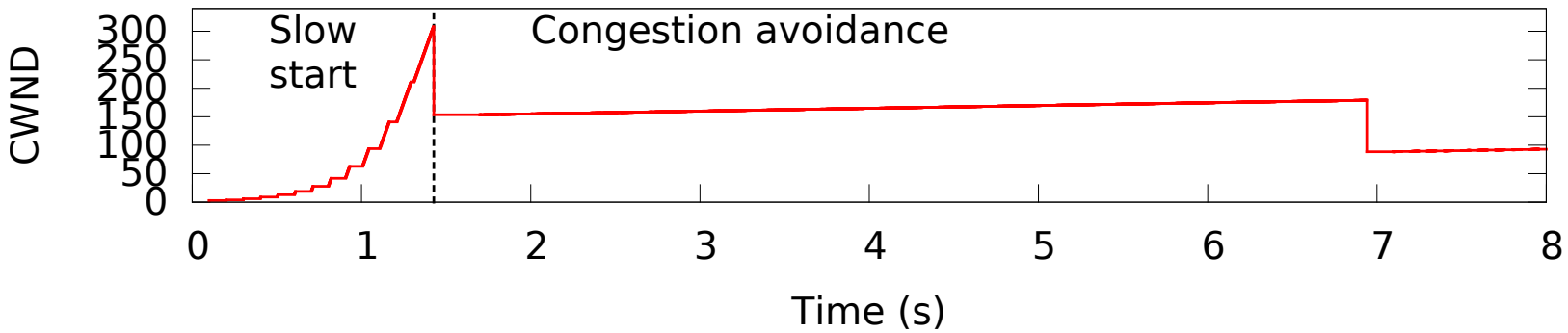
*

I. Järvinen *et al.*, "Harsh RED: Improving RED for Limited Aggregate Traffic," in *Proceedings of the AINA-2012*, Mar. 2012

I. Järvinen and M. Kojo, "Evaluating CoDel, PIE, and HRED AQM Techniques with Load Transients," in *Proceedings of the LCN 2014*, Sep. 2014

T. Høiland-Jørgensen *et al.*, "The Good, the Bad and the WiFi: Modern AQMs in a residential setting," *Computer Networks*, vol. 89, pp. 90–106, Oct. 2015

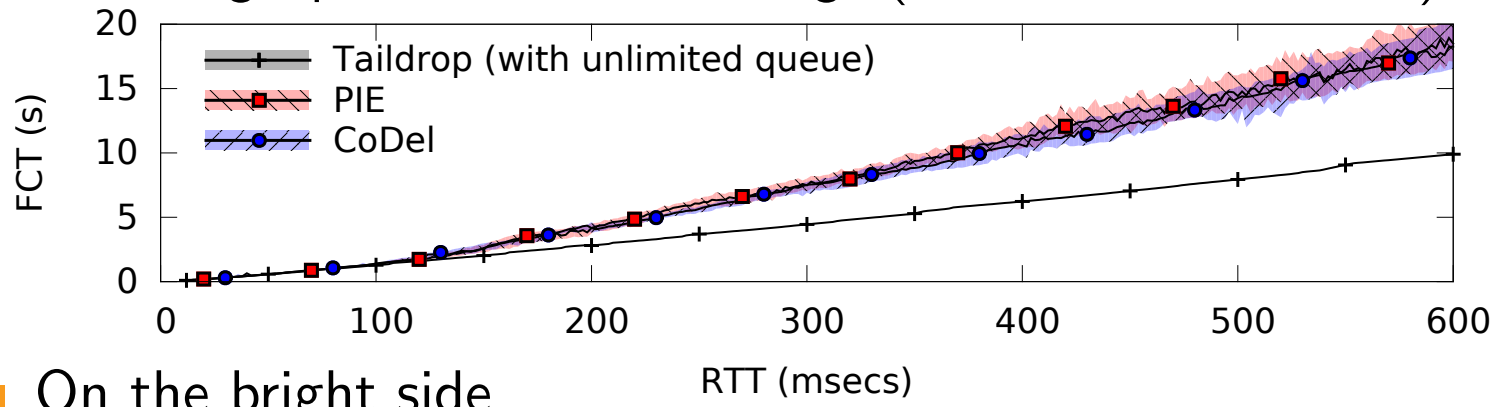
Motivation: Past Load Transient AQM Work (2)



- Exponential load growth during slow start is much more rapid than the current AQM algorithms expect by design
  - Latency spikes
  - Exponential load transients with multiple flows participating overpower control authority of the AQM algorithm even after congestion is detected
  - Load transients usually tested only after the AQM algorithm design phase?
- But this far, we have had no way to show how good the performance could be
  - We need an AQM algorithm that is designed primarily exponential load transients in mind

AQM and Exponential Load Transients

- Load changes very rapidly
  - "Current load" becomes stale very quickly
- Two pitfalls for AQM algorithms
  - Too slow reaction ⇒ Large latency spike
  - Too fast reaction ⇒ Flow completion time (FCT) increases with longer RTTs
    - Makes painfully long wait with intercontinental flows even longer
  - In odds with each other
  - Single parameter is not enough (for RTT, interval, etc.)



- On the bright side
  - Slow start is quite deterministic
  - Signal-to-noise ratio gets better and better because of exponential amplification

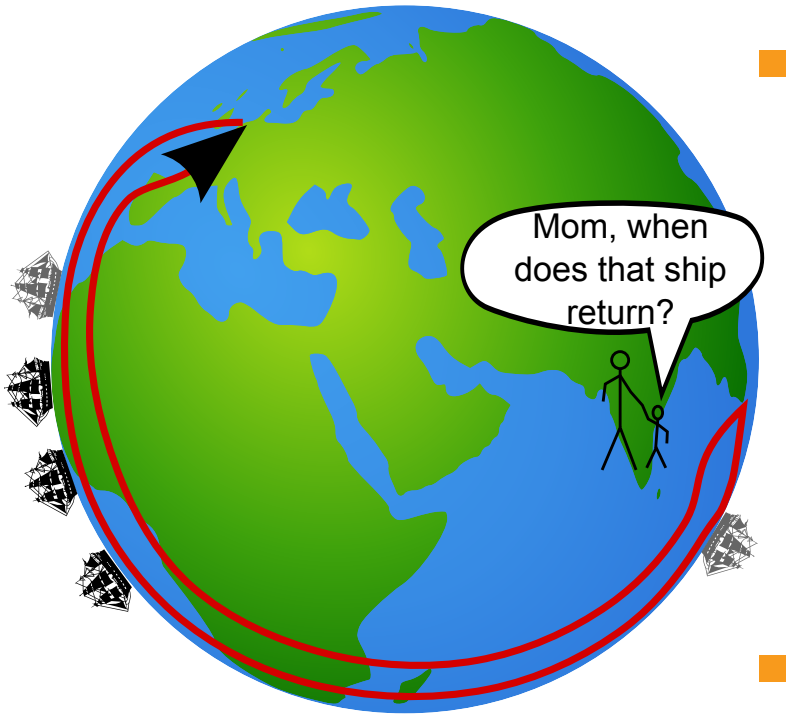
AQM and Exponential Load Transients (2)

- Load estimation is an open question in congestion control research (RFC 6077)
- Three challenges
  - Horizon problem
  - RTT uncertainty
  - Stale load estimates
- Lets look next into these challenges one by one

AQM and Exponential Load Transients: Horizon Problem

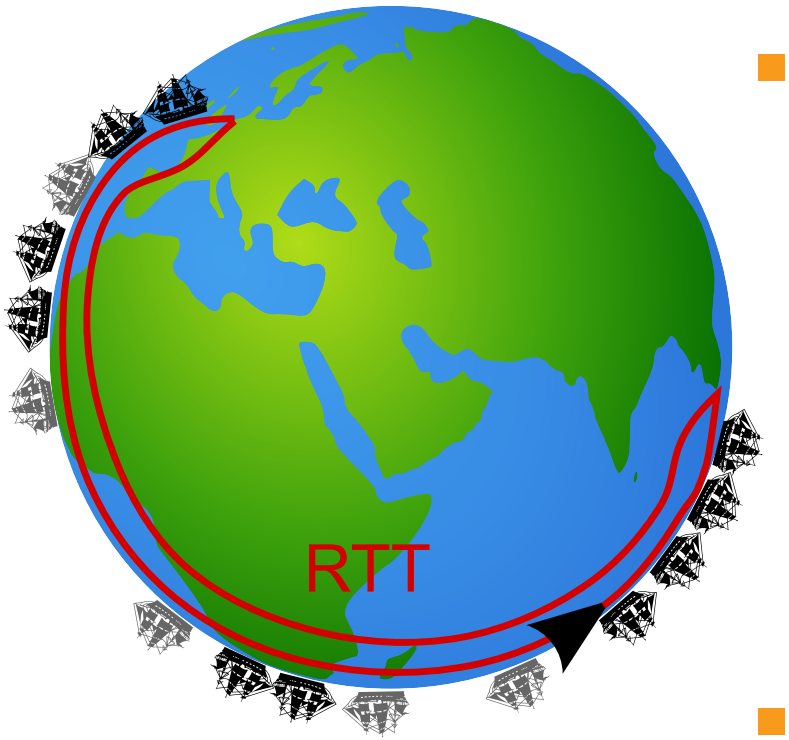
- Horizon limits visibility to a small part of the end-to-end path
  - Only a subset of the outstanding packets in the router queue
  - No packets within horizon does not imply no packets on the end-to-end path
  - And vice-versa, packets within horizon do not imply that end-to-end path is saturated
- Questionable to use queue length or queuing delay for load estimation in AQM algorithms

AQM and Exponential Load Transients: RTT Uncertainty



- RTT is a key parameter determining how quickly the exponential load transient load increases (doubles)
- Routers typically do not know the RTTs of the flows
  - Challenging to use a measurement interval in load estimation (RFC 6077)
  - Typically AQM algorithms select "worst-case" or "average" RTT and tune only to that
    - With other RTTs, the reaction is too slow or too fast
- Would require reconfiguration whenever the peer changes
  - But with very frequent changes, reconfiguration is not an option

AQM and Exponential Load Transients: Stale Load Estimate



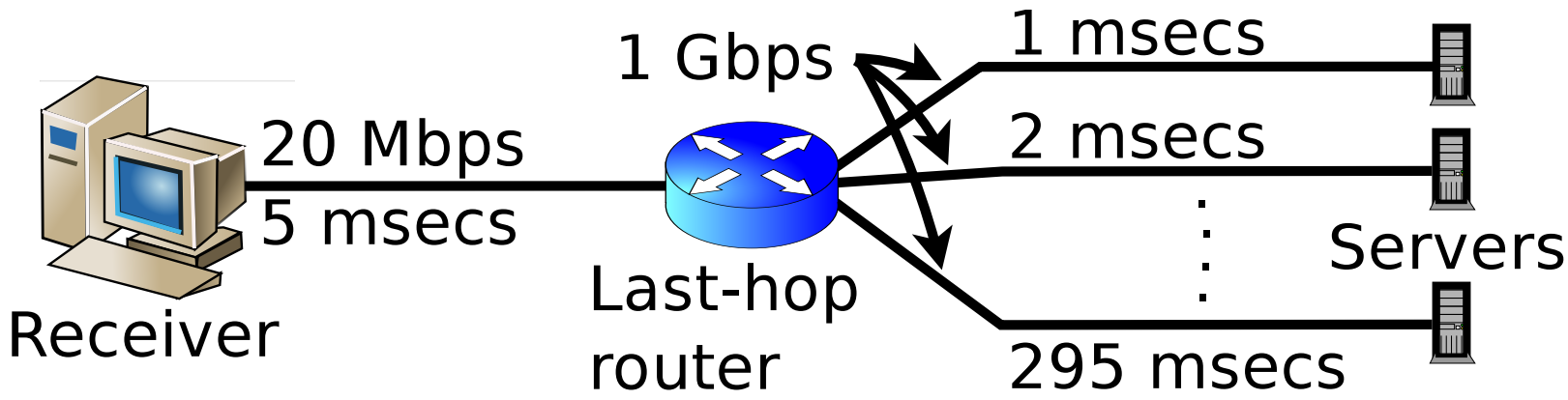
- In exponential load transients, the load grows rapidly
- Congestion signal (its effects) takes one RTT to reach back the signalling router
  - Load grows also during this time
    - If congestion signal is based on "current load", AQM is always behind the traffic by (at least) one RTT
    - Any estimate of "current load" is already stale when measured
    - Must predict load to counter this RTT long delay
- Stale load estimate is solvable for free, once we have a solution to the horizon problem and RTT uncertainty
  - That is, when we know the parameters of the exponential growth

Predict AQM Algorithm* Overview

- **Measures the parameters of an exponential load transient from the traffic**
  - Truly auto-tunes for different RTTs
    - Able to use "right measurement interval" (RFC 6077)
  - Capability to predict the load growth to future
- **Tracks load below saturation accurately, also when queue length or queuing delay $\approx$ 0**
  - Should be fully compatible with pacing
  - In contrast to most AQM algorithms that rely on non-paced TCP artifacts to work as good as they do
    - Pacing during slow start delays response as no queue forms
- **Timely full load detection allows eliminating/relaxing multiplicative decrease**
  - TCP is not ready to timely signal and backs off
  - ECN-based hack for testing purposes to terminate TCP slow start without window reduction (TCP continues in CA)
- Only in proof of concept state currently
  - Fairness / congestion avoidance aspects not solved yet

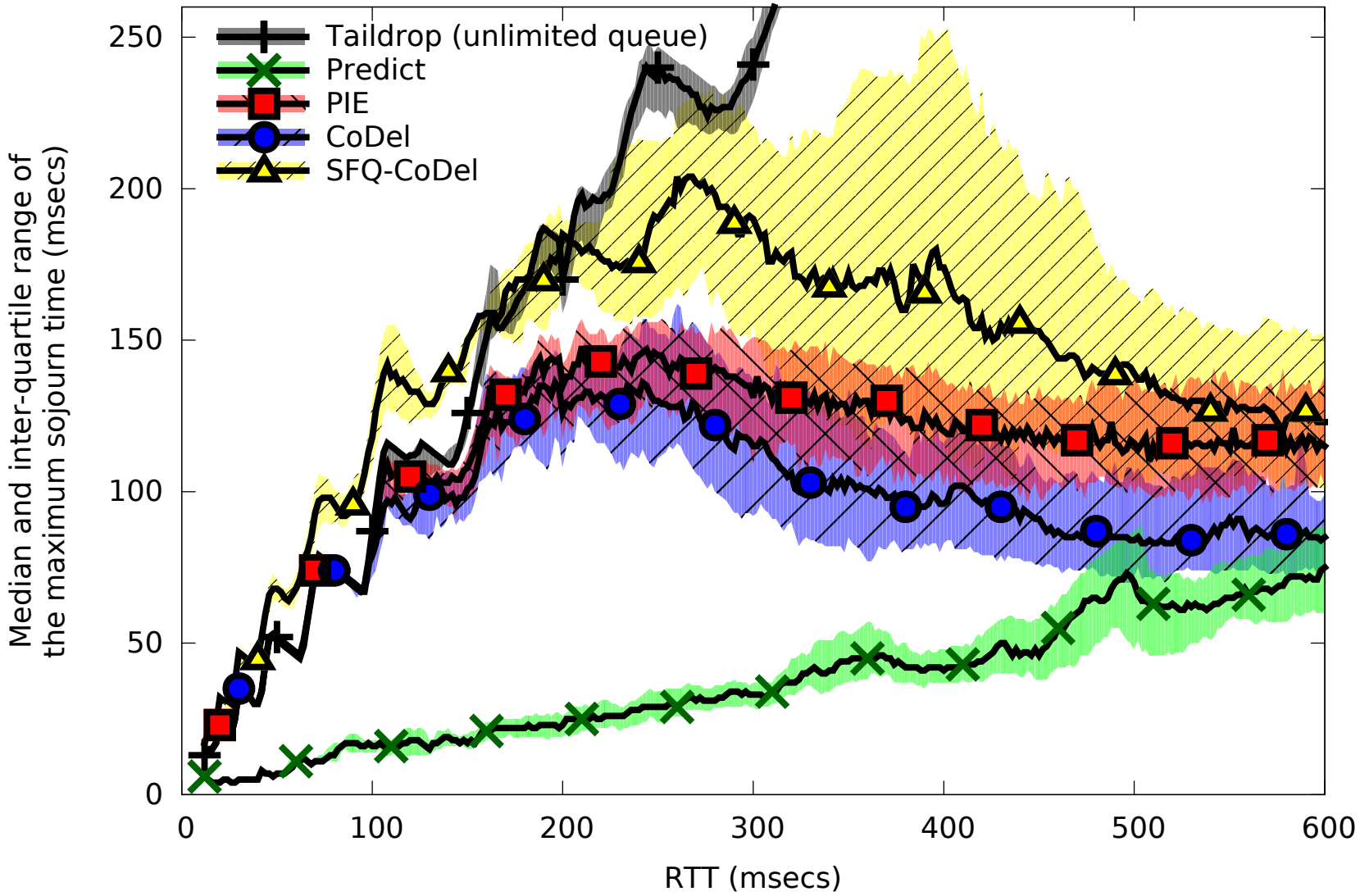
*Paper under submission

Test Setup

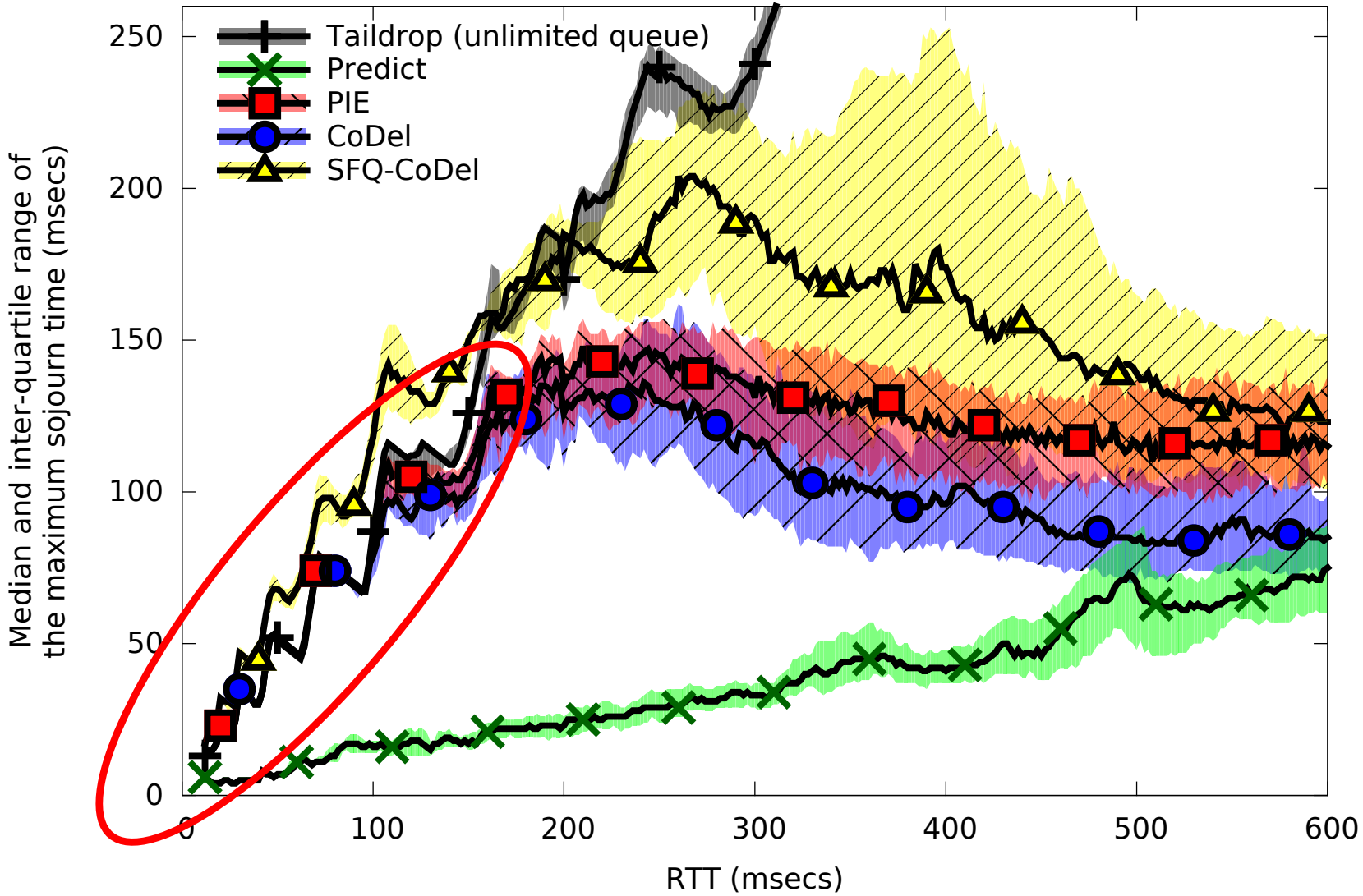


- End-to-end RTT varied: 12 msecs - 600 msecs
- "Infinite" buffer
  - Only AQM algorithm drops
  - No drops with Taildrop (for "optimal" flow completion time)
- Workload:
  - 1 to 4 TCP flows (only 4 shown here)
    - Flow start times distributed over the first RTT
  - Scaled payload ($12 \cdot$ end-to-end BDP)
    - To avoid sampling too much from congestion avoidance
    - Divided equally to the TCP flows
- Ns2 simulations, 100 replications
- In addition, some real network measurements

Results: Maximum Sojourn Time

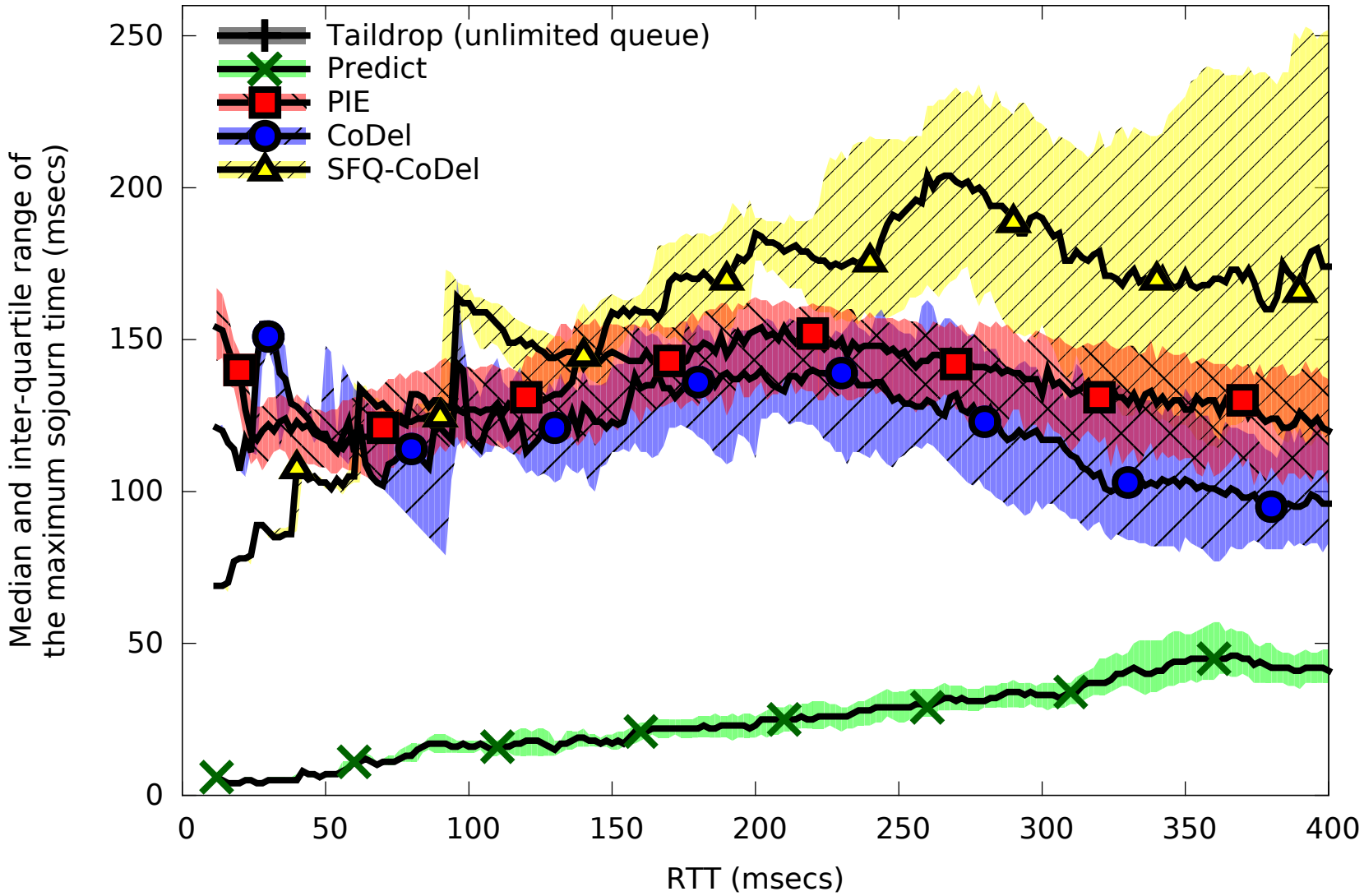


Results: Maximum Sojourn Time



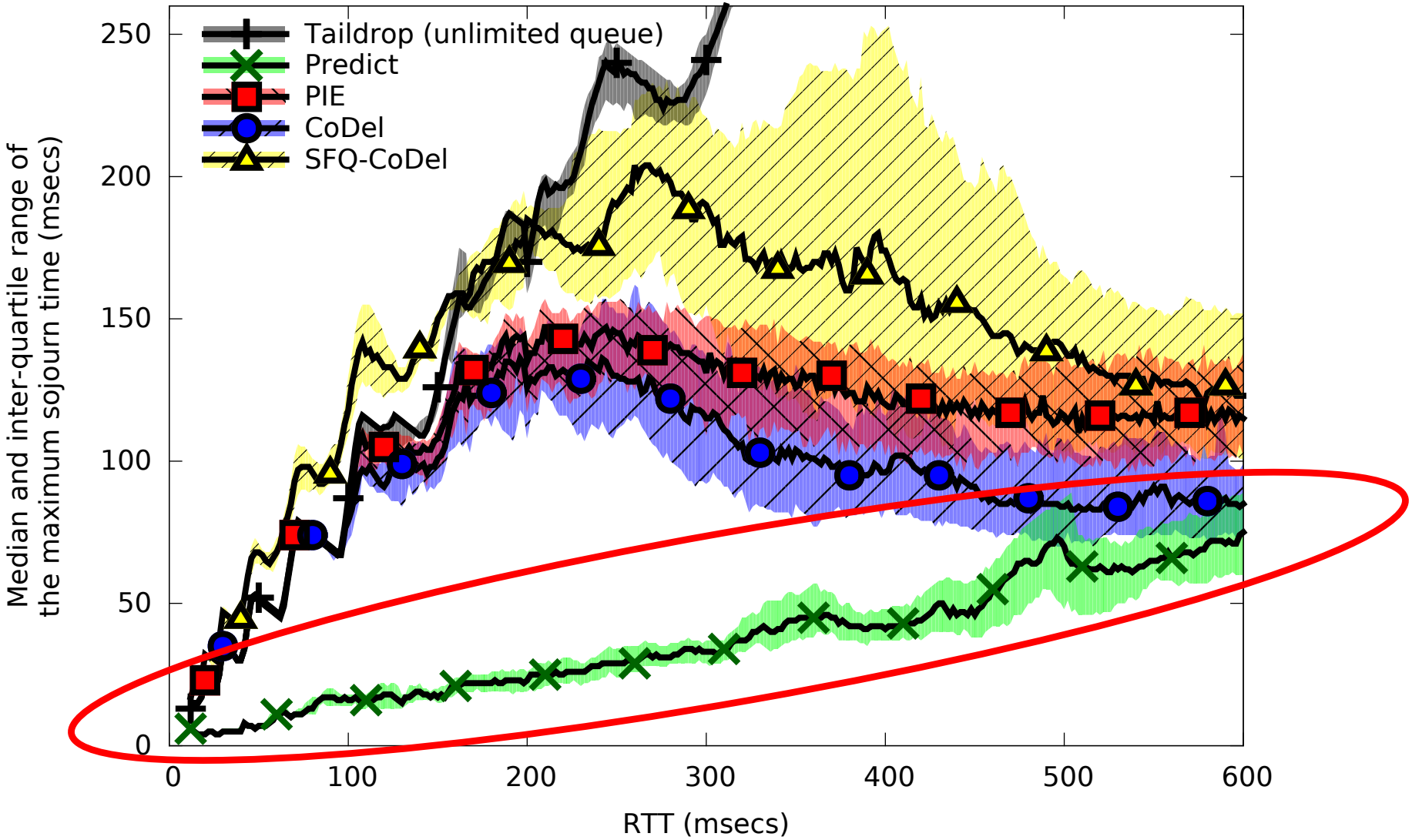
- PIE and CoDel do not control congestion with short flows and small RTTs (too slow reaction)

Results: Maximum Sojourn Time (6MB payload)



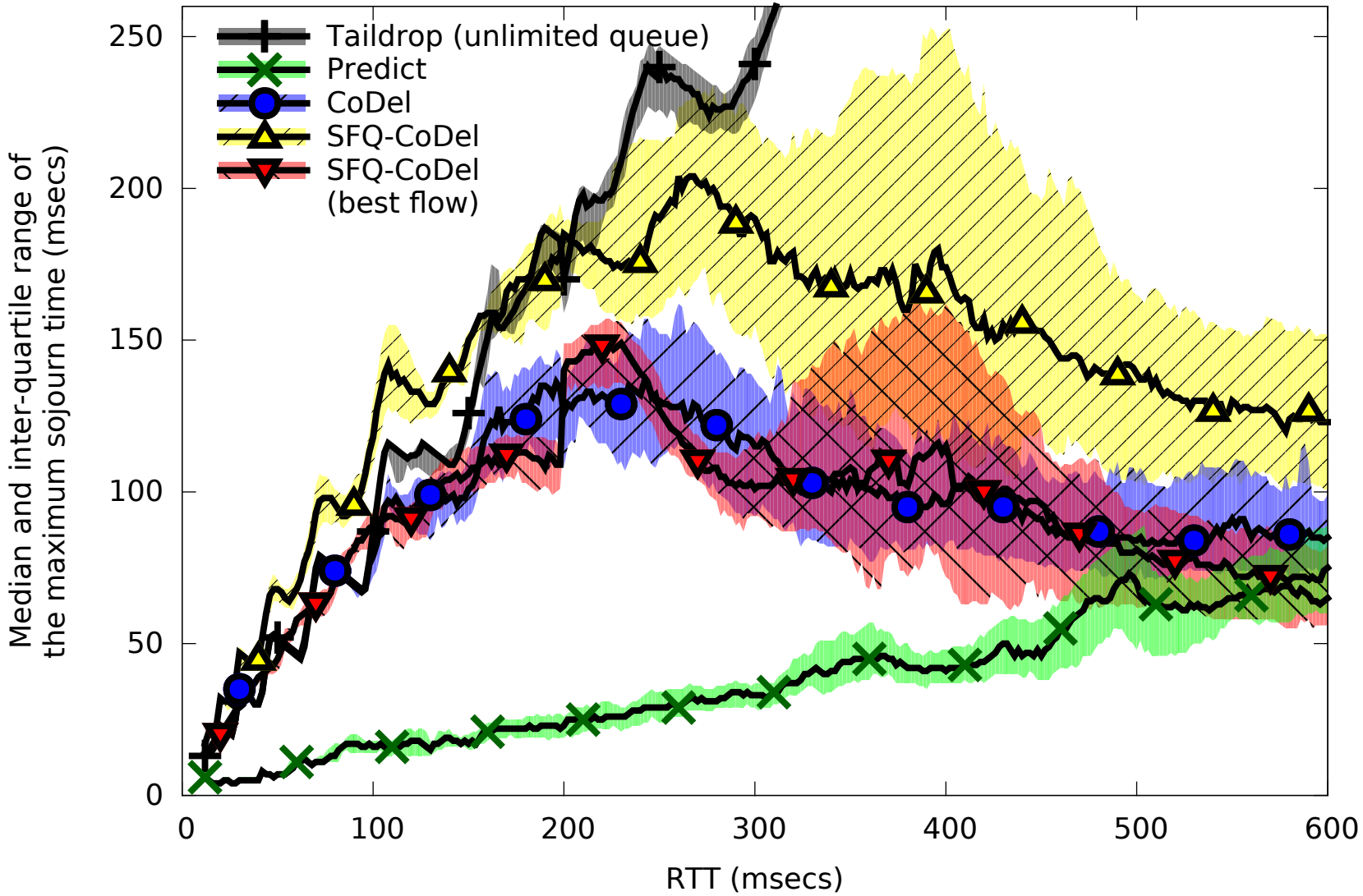
- This is what happens when the flows are long enough for PIE and CoDel to react also with small RTTs

Results: Maximum Sojourn Time



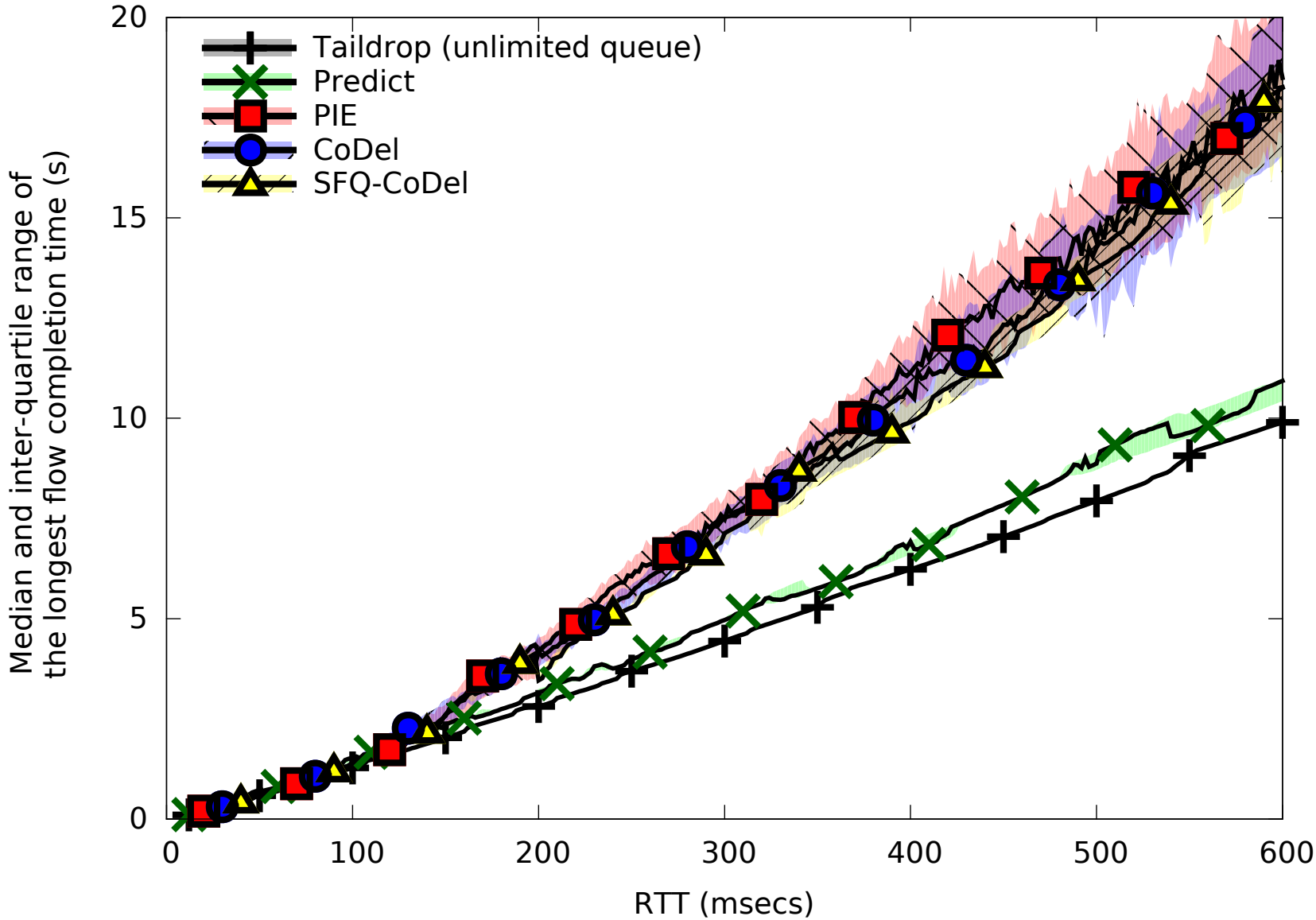
- Shallow increasing slope with Predict due to larger transient-only queue spikes with larger RTTs

Maximum Sojourn Time (with SFQ-CoDel best flow)



- Previous SFQ-CoDel results were from the worst flow only
  - Even the best flow with SFQ-CoDel is no match for Predict

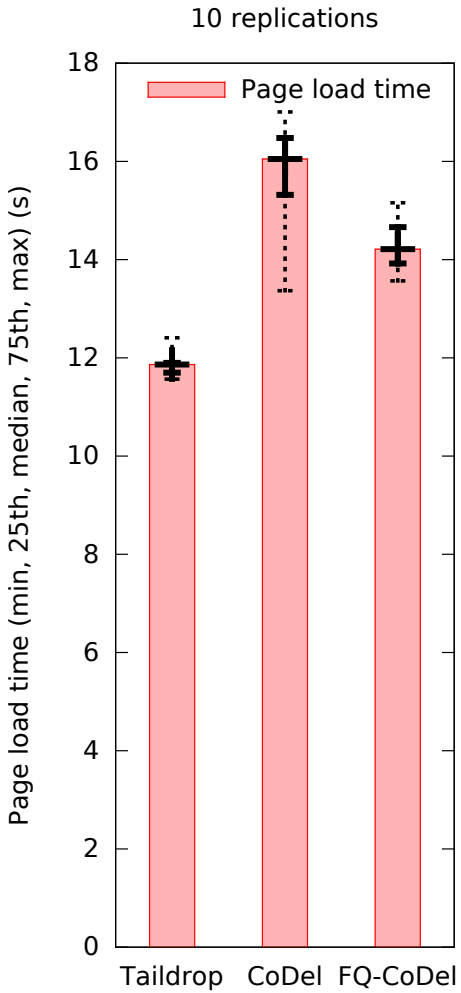
Results: Flow Completion Time



- PIE and CoDel react too fast with large RTTs

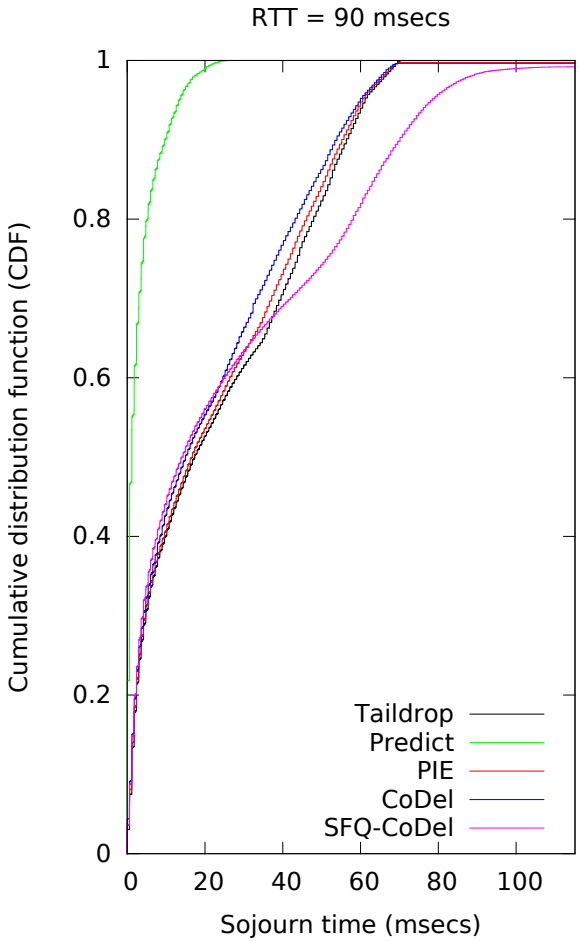
Flow Completion Time (FCT) of a real Web Page

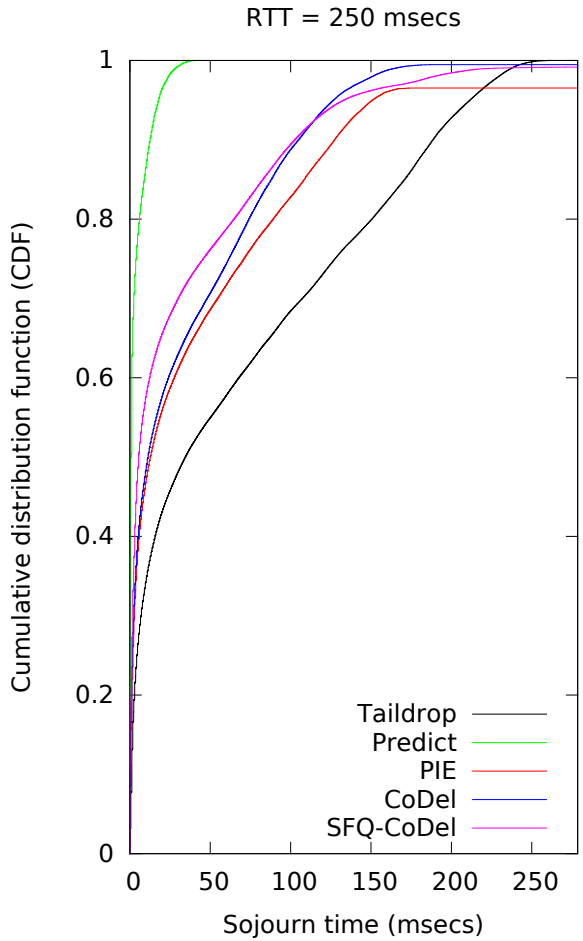
- Simulated results with FCT increase made us curious if we could also see the effect for a real Web site too as increase in Page load time
  - Since we're at Seoul now, we used a relevant Website (AREX*mainpage http://www.arex.or.kr/)
  - Web transaction was initiated from Finland by a Firefox browser (SYN RTT $\approx$ 310 msecs)
  - Turns out the answer is YES
- Due to time constraints only a 5Mbps bottleneck link test was possible (vs 20Mbps in simulations)
  - With larger BW and BDP, the difference might be even larger

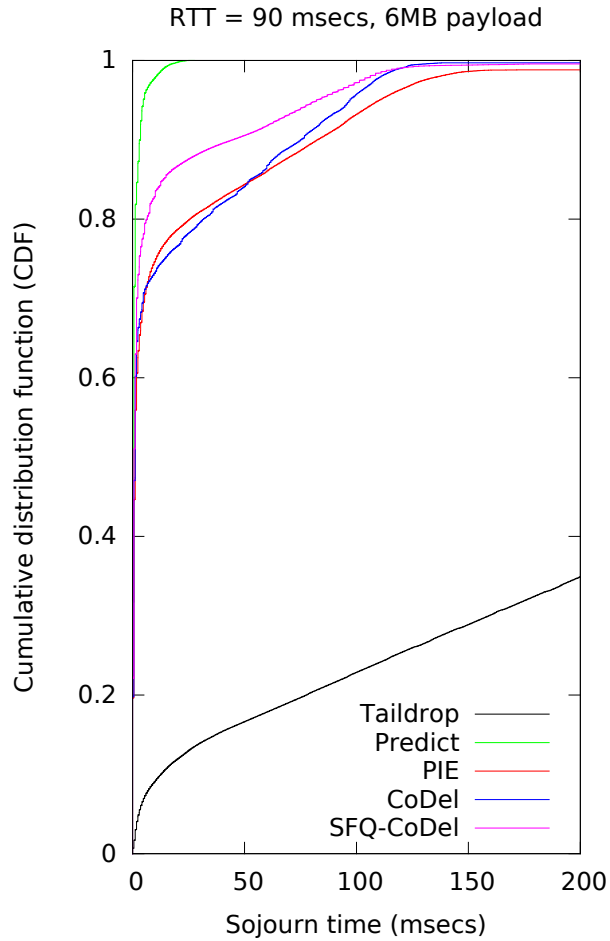


*Not affiliated to us in any way

Results: CDF of Sojourn Time

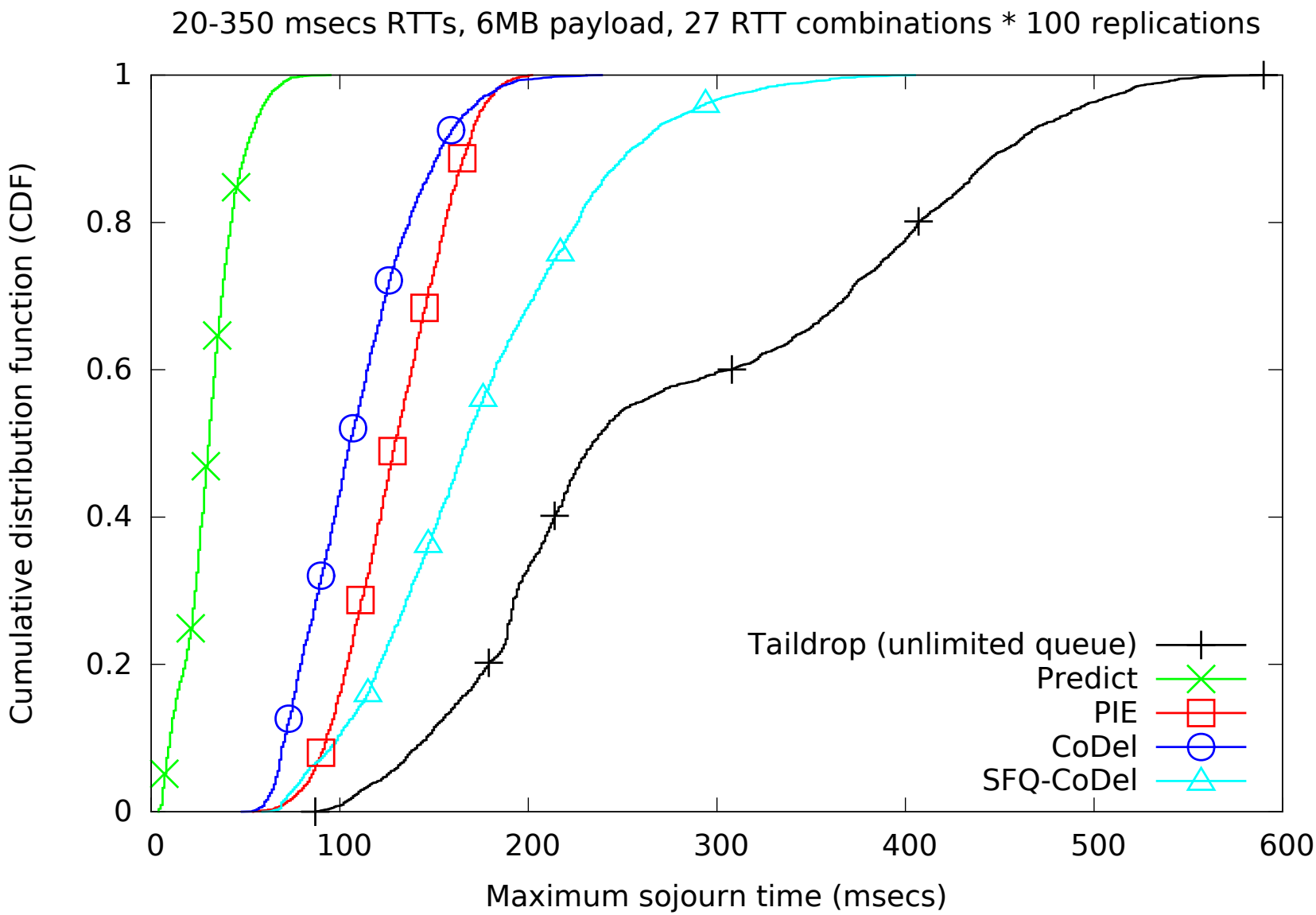




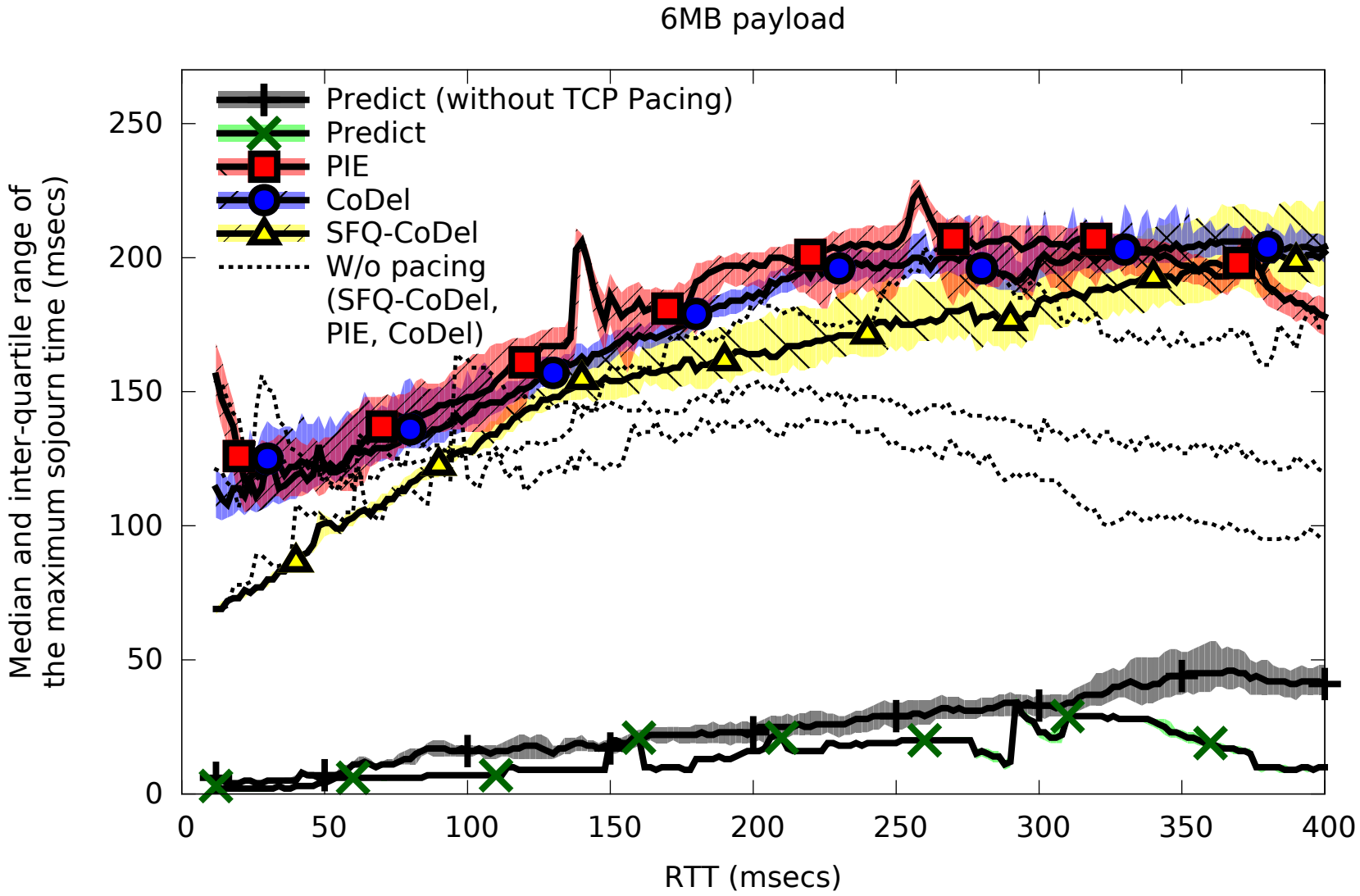


- Beware: different x-axis scales
- Again, Predict leads with a very clear margin
- Note: workload/metrics selection can diminish slow start effects in results (leftmost vs rightmost figure)

Results: Max Sojourn Time with Heterogeneous RTTs

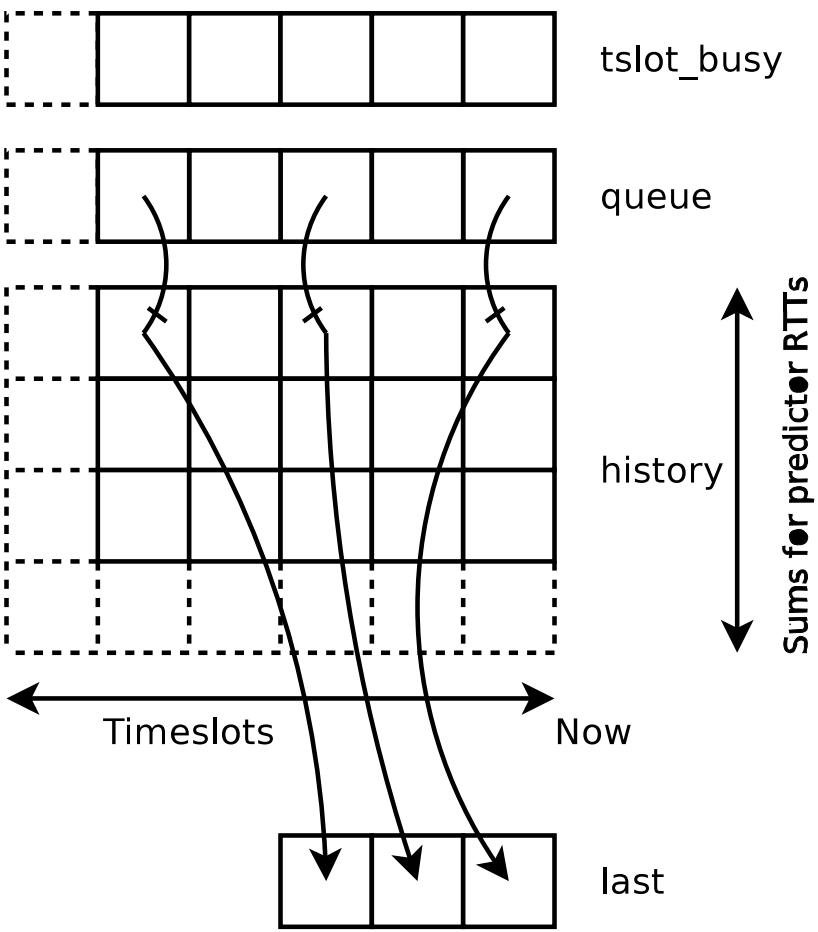


Results: Max Sojourn Time with Pacing



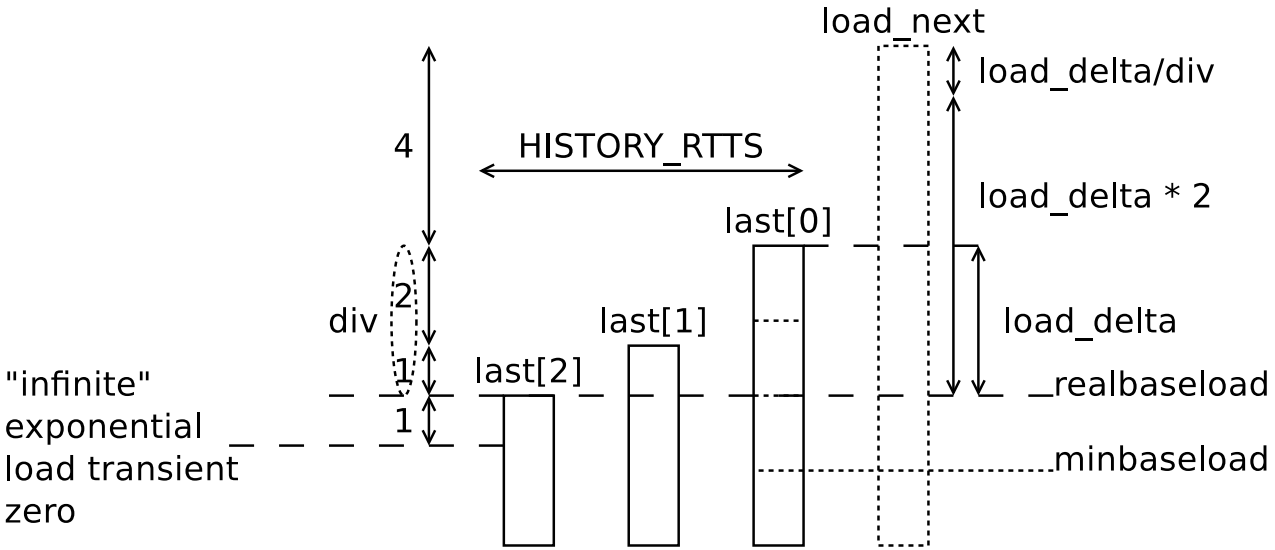
- Predict yields equal or better sojourn time with pacing
- Higher maximum sojourn times with PIE and CoDel

Predict AQM Internals



- Timeslots
- At the end of each timeslot, store load (link busytime) into a history array (big, 2-d)
  - Rolling sums, the tslot_busy array used for subtracting at the tail
- Queue history in another array
- At end of each timeslot, collect the loads (history+queue) from prior RTTs into a last array
  - Collect using a stretched RTT that takes into account self-induced RTT variations due to transient queuing

Predict AQM Internals (2)



- Multiple predictors for candidate RTTs, heuristics to select the right candidate RTT
  - The last array is collected for each predictor
  - Look for exponential load growth using the last array
- Predict the load growth using exponential trend one RTT ahead in order to signal the sender in time
  - Good enough match between the candidate and real RTT is enough for in the ballpark prediction
- Well ahead of traffic, no need to use head drop hack to deliver feedback "sooner"

Predict AQM Computational Complexity

- More detailed analysis of traffic than in other AQM algorithms
  - Inherently higher computational cost
- However,
  - Reasonable enqueue/dequeue complexity
  - Most work done at the end of a timeslot
  - Uses precalculated values, at the cost of memory
    - Less memory needed than the memory required to buffer the worst-case non-saturating slow-start transients

Fair Queuing as a Solution?

- In many cases, FQ works just fine (for hiding AQM shortcomings with load transients)
- However,
  - Latency sensitive flow must have low enough rate
    - In general, latency sensitive flows cannot control how many flows they compete with
    - "Equal-rate" (or "fair-share") is unknown, could be quite small
  - If a flow participating in the exponential load transient is latency sensitive, FQ does not help
- Better to control the exponential load transient in timely manner with AQM algorithm
  - Good latency for all

Conclusions and Research Insights

- Handling exponential load transients should be considered already during AQM algorithm design phase
  - Not done in the past $\Rightarrow$ AQMs have hard time to correctly manage such transients, results in flow completion time (FCT) and delay issues
    - It is like designing a ship only for calm waters
    - ...and releasing it to the rough waters on the Internet
- Predict AQM algorithm
  - First(?) AQM algorithm addressing exponential load transients as the primary design goal
  - Nearly optimal control w.r.t. FCT vs delay tradeoff
  - Scales to a large RTT range
  - Allows relaxing the requirement for multiplicative decrease
  - By no means, we claim that Predict AQM approach is the best or only way to handle the exponential load transients
    - A curve fitting approach would likely be more robust, but needs to be cheap computationally
- This work gives additional insights on solving the open challenge about "measuring a current link load" (RFC 6077)

Backup Slides

Related Work Comparison

	Modifies	Flow completion time (FCT)	Overshoot with pacing during SS	Relaxed MD at the end of SS
Router only AQMs (PIE, CoDel)	Router	Harmed with large RTTs	Yes	Not allowed
Predict	Router+end host	Ok	No	Yes
DualQ/DCTCP	Router+end host	Harmed with large RTTs	Yes	Yes, but limited
HyStart	End host	Ok	Incompatible with pacing	Yes, to CA without congestion signal