FASOR: Rethinking Retransmission Timeout and Congestion Control for Environments with Random Losses and Congestion

draft-jarvinen-core-fasor

Ilpo Järvinen[*], Iivo Raitahila[*], Zhen Cao[†] and Markku Kojo[*]

[*]University of Helsinki [†]Huawei

iccrg @ IETF-102
July 19, 2018

Introduction and Environment

- FASOR (Fast-Slow RTO) is currently primarily designed / specified for Constrained Application Protocol (CoAP)
- Challenges in constrained environments
  - Wireless networks subject to random losses
  - Small exchanges are common
  - Congestion will mainly occur due to large number of devices
    - Tens of billions IoT devices expected to be deployed in the coming decades
- CoAP allows only one message in flight at a time per flow
  - RTO is the only loss recovery mechanism available
- However, FASOR is more general and applicable to RTO mechanisms also in other protocols
  - The biggest benefits expected for small request-reply type of traffic

Problem with Current CoAP RTO Management

- Karn's algorithm: exponential backoff and keep RTO until unambiguous RTT sample acquired
- CoAP CC algorithms: exponential backoff but DO NOT retain the backed off RTO
- Congestion collapse occurs with default CoAP and CoCoA*
  - Unnecessary retransmissions occur persistently if RTT $>$ RTO with the RFC 7252 algorithm
  - CoCoA not safe either but more complicated
    - Weak estimator hacks around the lack of retaining the backed off RTO (but RTO only updated if $<$3 rexmits were made)
    - RTT that triggers 3+ rexmits still causes the collapse
- Lack of retaining RTO good for random losses though

---

* I. Järvinen, I. Raitahila, Z. Cao, and M. Kojo, "Is CoAP Congestion Safe?," in *Proceedings of the Applied Networking Research Workshop 2018 (ANRW'18)*, July 2018

FASOR (Fast-Slow RTO) in Nutshell

- FASOR (Fast-Slow RTO)* tries to find a good middle ground
  - Try to improve random loss
  - . . . but still handles congestion safely, including unnecessary rexmits
- Two ways to calculate RTO
  - FastRTO (normal RTO)
  - New SlowRTO
- New back off logic

---

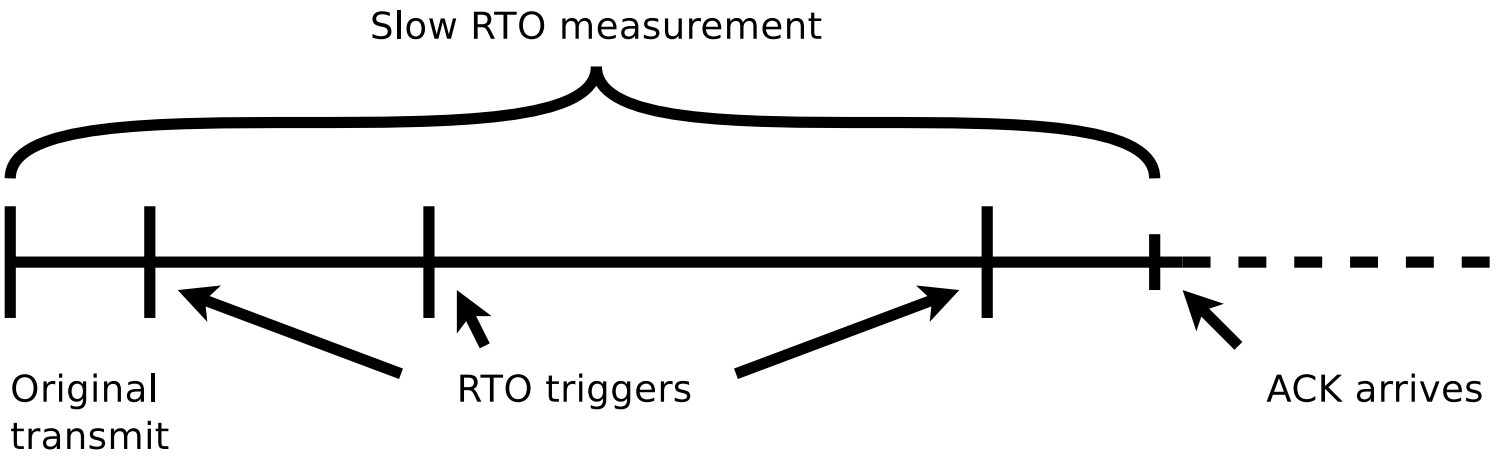
*

I. Järvinen, I. Raitahila, Z. Cao, and M. Kojo, "FASOR Retransmission Timeout and Congestion Control Mechanism for CoAP," in *Proceedings of IEEE Globecom 2018*, Dec. 2018. To appear

I. Järvinen, M. Kojo, I. Raitahila, and Z. Cao, "Fast-Slow Retransmission and Congestion Control Algorithm for CoAP," Internet Draft, June 2018. Work in progress

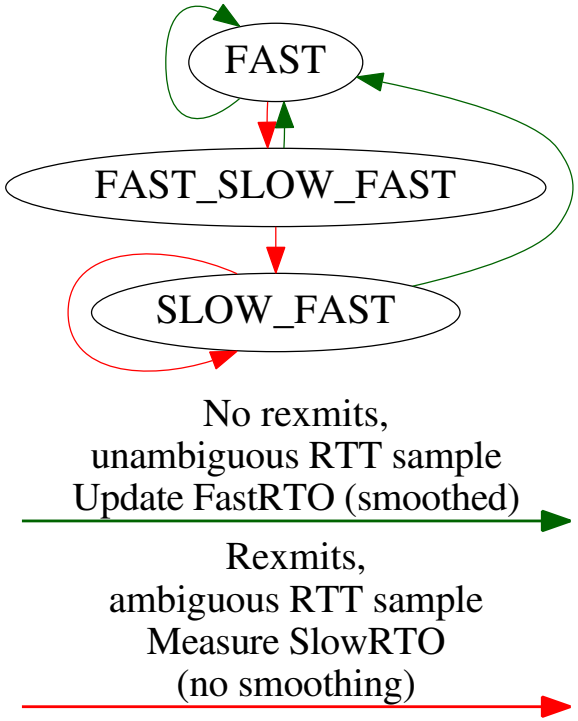
FastRTO and SlowRTO

- FastRTO ≈ RFC 6298 RTT/RTO computation
  - Initialization of RTTVAR changed to R/2K
    - Lowers RTO for short exchanges
- SlowRTO analogous to Karn's algorithm keeping RTO until unambiguous RTT sample
  - Measured when retransmissions were made as the time elapsed from the original copy
  - Multiplied by a factor to allow load growth (1.5 by default)
  - More conservative than Karn's algorithm



FASOR Back Off Logic

- Modify 2-state RTO logic of Karn's algorithm by adding a new state and modify back off series:



FastRTO, FastRTO*2^1, FastRTO*2^2, ...

FastRTO, max(SlowRTO, FastRTO*2), FastRTO*2^1, FastRTO*2^2, ...

SlowRTO, FastRTO, FastRTO*2^1, FastRTO*2^2, ...

FASOR States

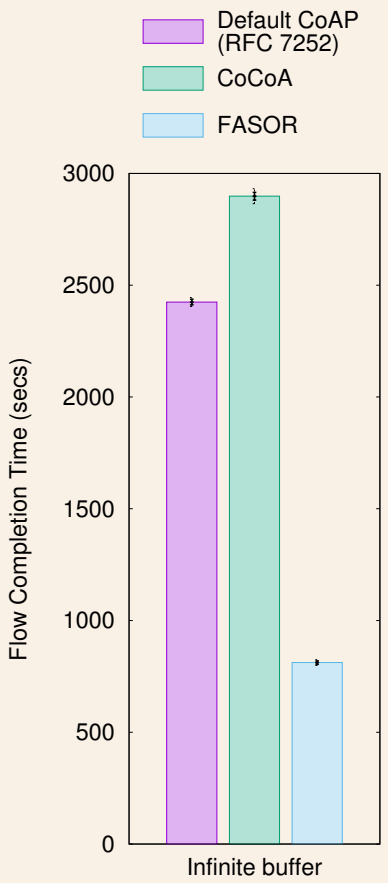
- FAST
  - "Normal" RTO series with exponential back off
  - When network state is not dubious
- FAST_SLOW_FAST
  - Probe first with FastRTO
    - Helps random loss cases to retransmit quickly
  - If no response and RTO expires, drain unnecessary retransmissions from network using SlowRTO
    - Due to lack of response so far, the sender cannot know if unnecessary retransmissions occurred or not
    - Safe and conservative option taken
  - If still more RTOs trigger, continue the Fast RTO based exponential back off
- SLOW_FAST
  - Start with SlowRTO to acquire an unambiguous RTT sample with high probability

Test Setup

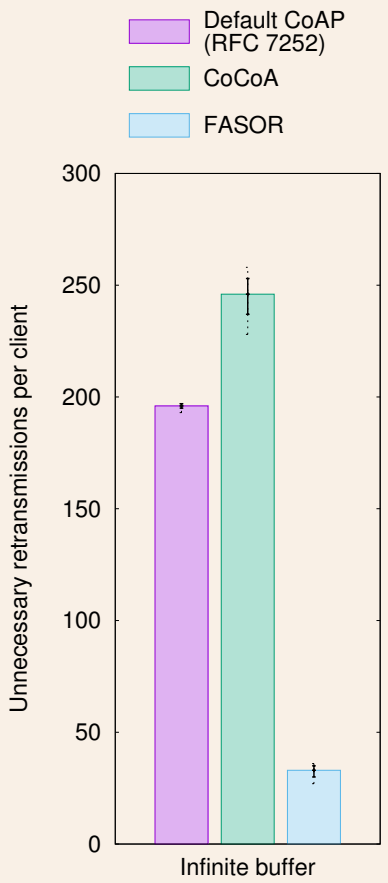
- Bottleneck BW: 30 kbps
- Base RTT $\approx$ 660 msecs
- Workload
  - A flow: a series of short-lived clients perform 50 request-responses exchanges in total
  - CC state reset after 1 to 10 message exchanges (new short-lived client starts)
  - Response payload: 60 bytes
- Test scenarios
  - Heavy congestion and bufferbloat
    - Up to 400 parallel flows
    - Varying buffer size, including infinite buffer (1410000 bytes)
    - RTT $\approx$ 10 secs (for 400 clients + infinite buffer)
    - Error-free link
  - Random losses
    - 10 parallel flows
    - No congestion
    - 2-state error model: 0%/50% (medium) or 2%/80% (high) packet error rate

Results with Heavy Congestion and Bufferbloat

FCT



Unnec. Rexmits

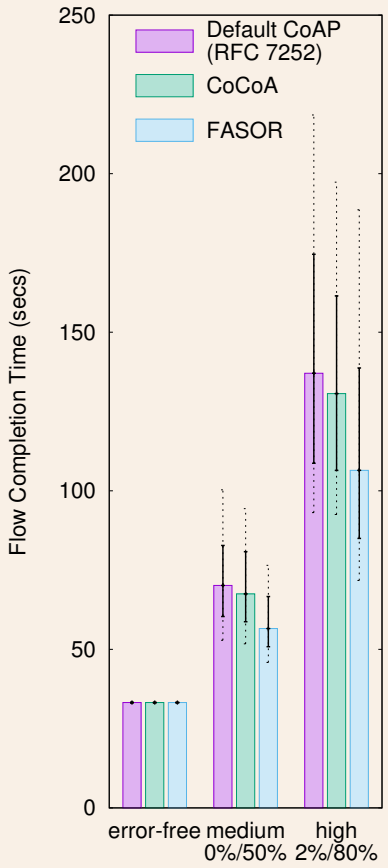


Observations

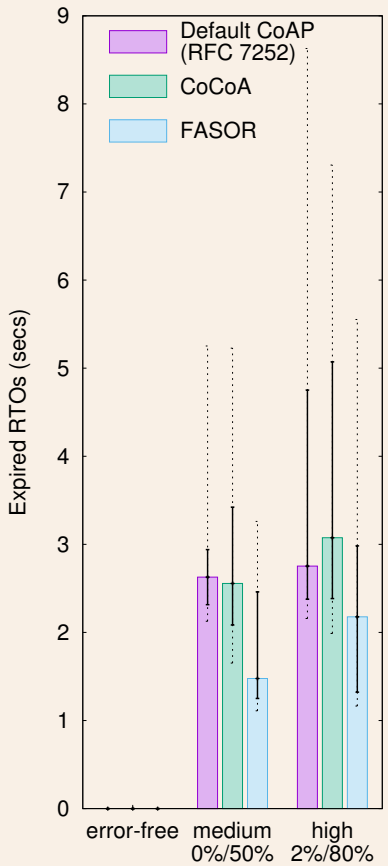
- FCT for Default CoAP and CoCoA long due to unnecessary retransmissions
- Reduction in median with FASOR
  - FCT: 67%-72%
  - Unnecessary rexmits: 83%-87%
- Similar pattern visible also in RTT

Results with Random Loss

FCT



Expired RTOs



Observations

- Median of the FCT shorter with FASOR:
  - medium: 16%-19%
  - high: 19%-22%
- FASOR is able to lower RTO value despite the challenging short-lived clients
- CoCoA's weak estimator measures random loss noise on ambiguous RTT samples
  - Its RTO values increase instead of converging towards the real RTT ($\approx$ 660 msecs)

Conclusions

- We believe FASOR achieves good balance between handling link errors efficiently and responding to congestion adequately
  - Slightly more aggressive than the traditional RTO algorithm – but safely so
- FASOR handles congestion and unnecessary retransmissions robustly
- But does not compromise performance for cases with random loss

Questions? Comments? Thoughts?

Backup Slides

Backup Slides

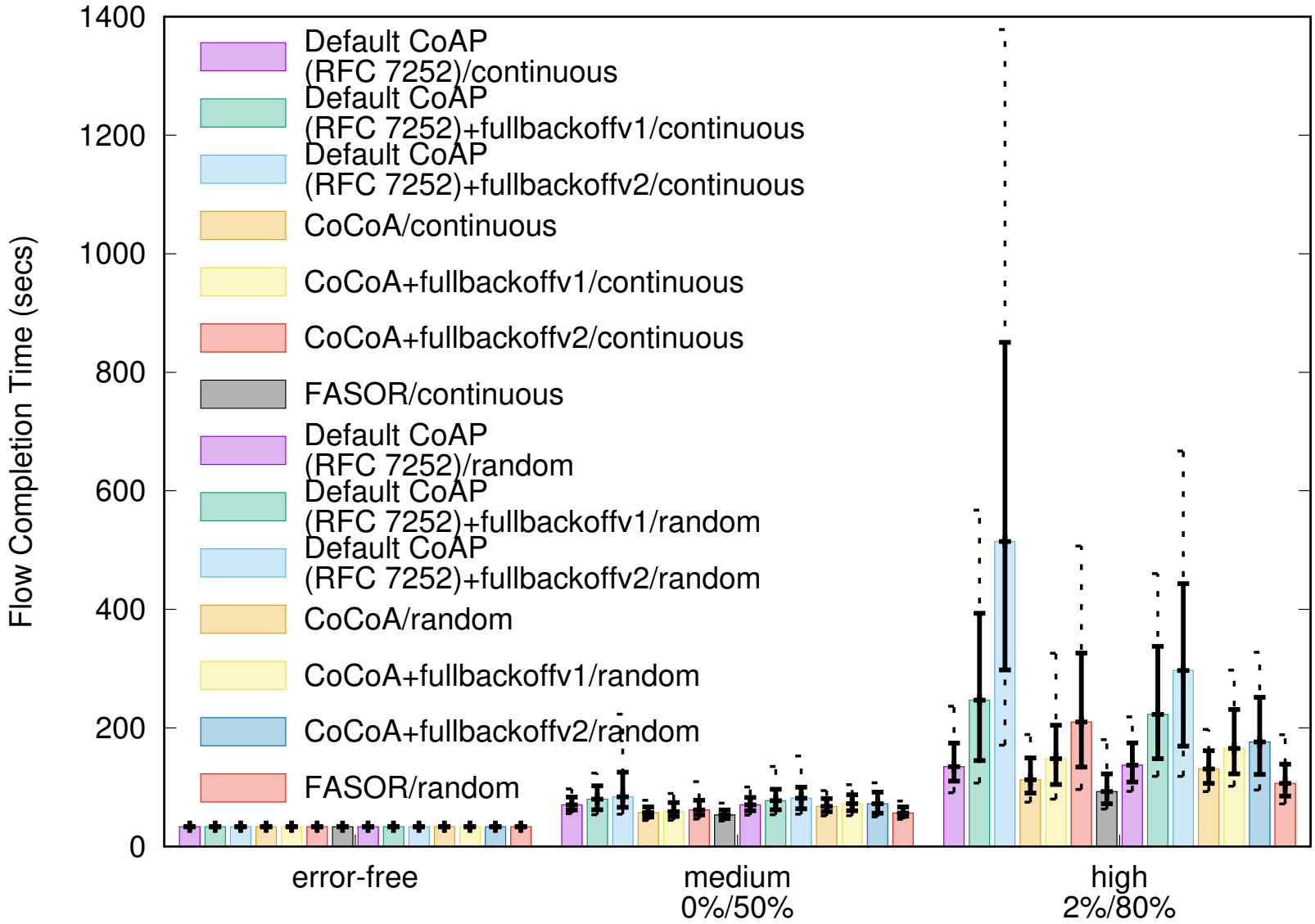
- "Continuous" workload: 50 request-replies; does not reset CC state after 1 to 10 exchanges
- "Random" workload: 50 request-replies; CC state reset after 1 to 10 exchanges
- "Fullbackoff" variants* are congestion safe versions of default CoAP and CoCoA adding retaining RTO similar to Karn's algorithm

---

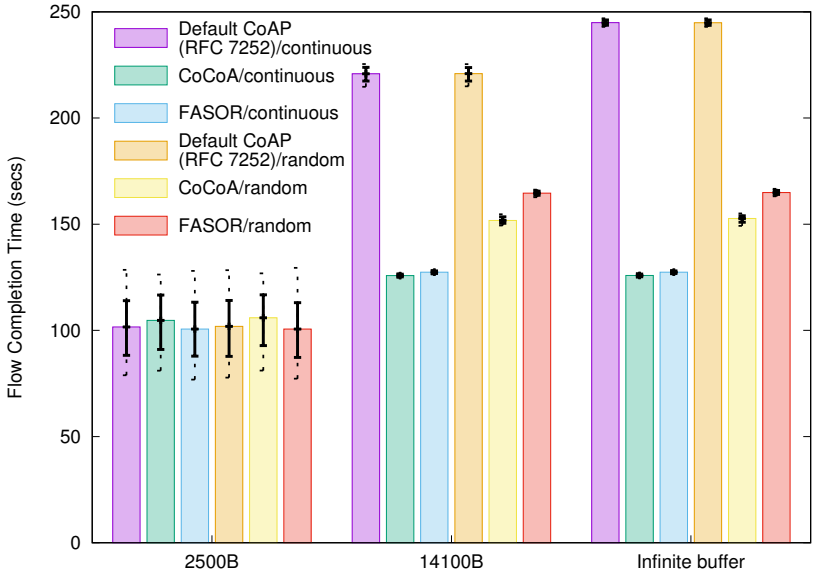
*

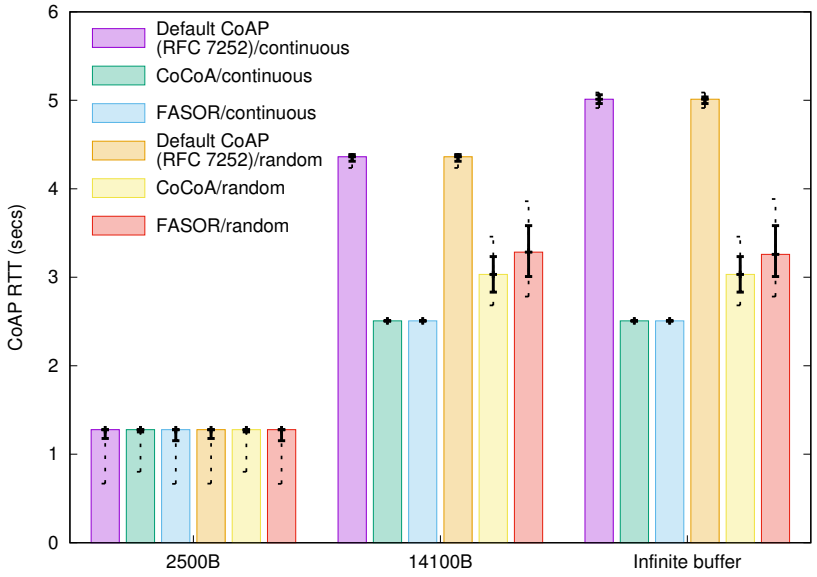
I. Järvinen, I. Raitahila, Z. Cao, and M. Kojo, "Is CoAP Congestion Safe?," in *Proceedings of the Applied Networking Research Workshop 2018 (ANRW'18)*, July 2018

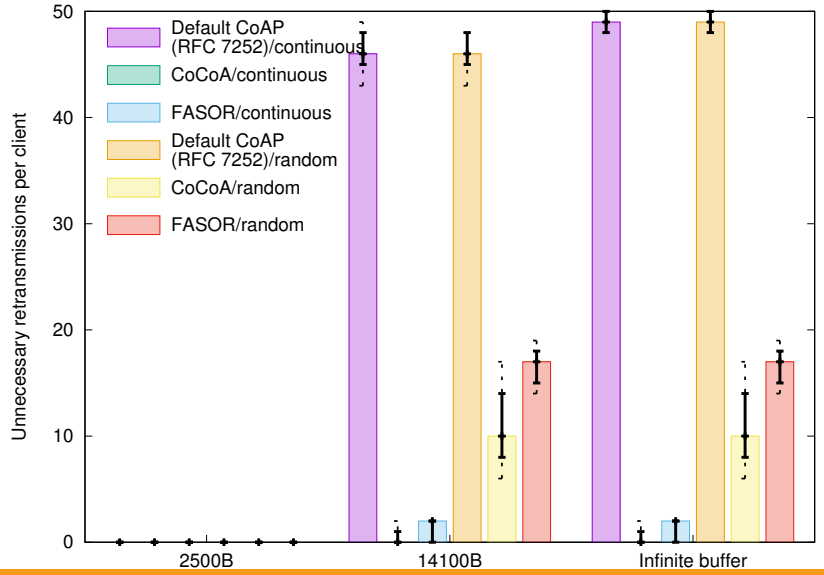
Backup Slides: Fullbackoff Variants



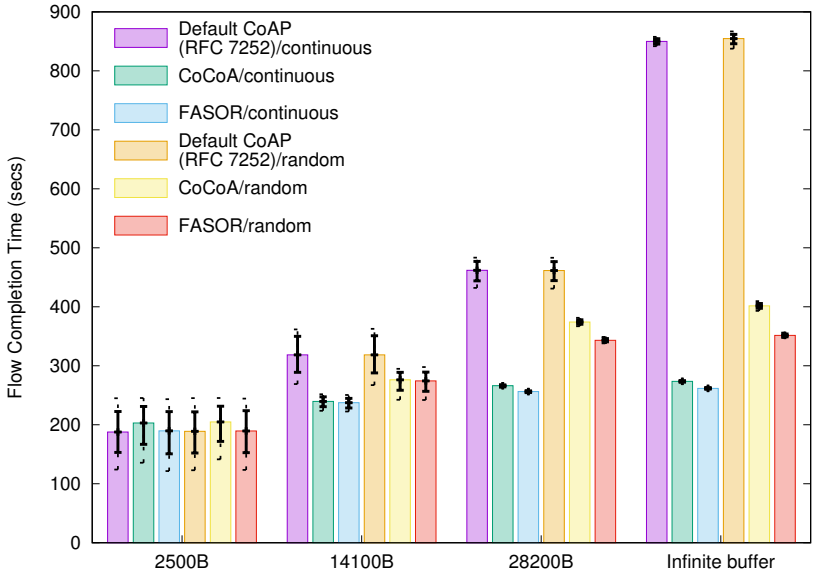
Backup Slides: 100 Parallel Flows

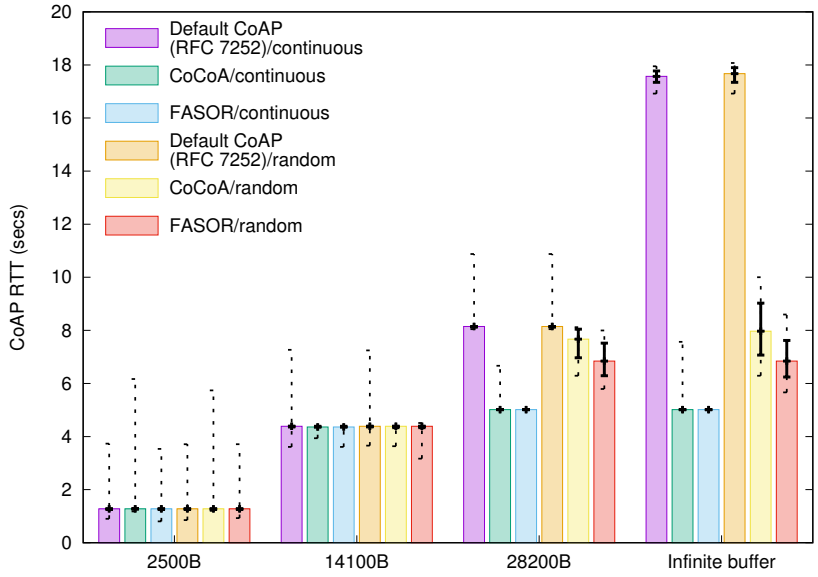


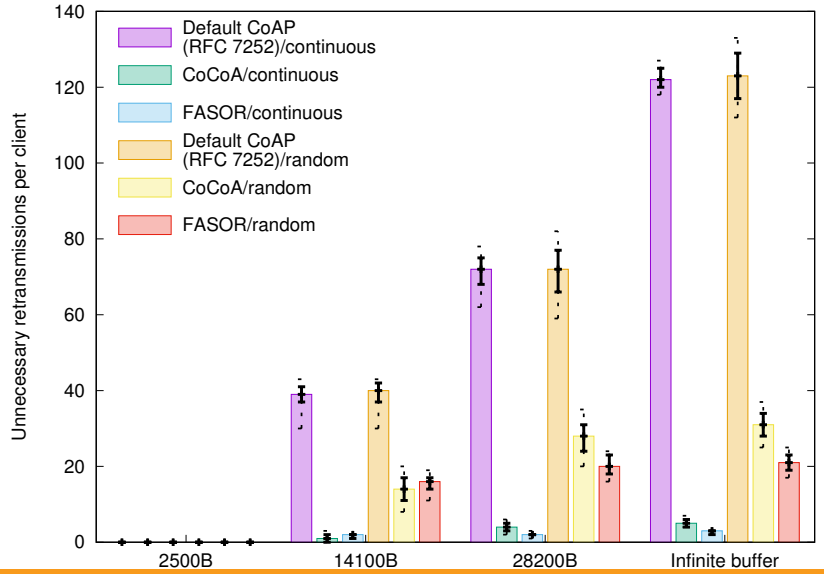




Backup Slides: 200 Parallel Flows







Backup Slides: 400 Parallel Flows