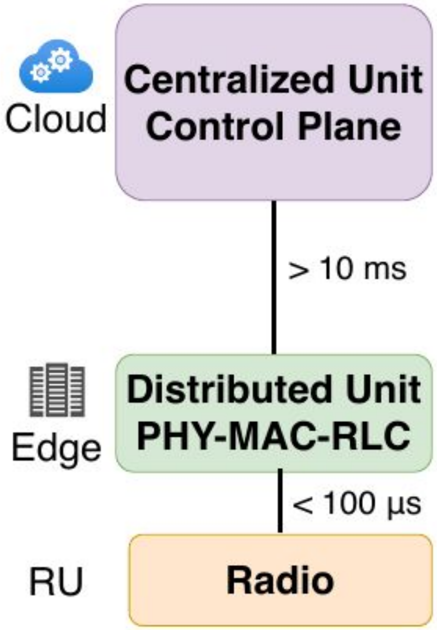
EdgeRIC: Empowering Realtime Intelligent Optimization and Control in NextG Cellular Networks

Presenter: Xuyang Cao • 09.13.2024

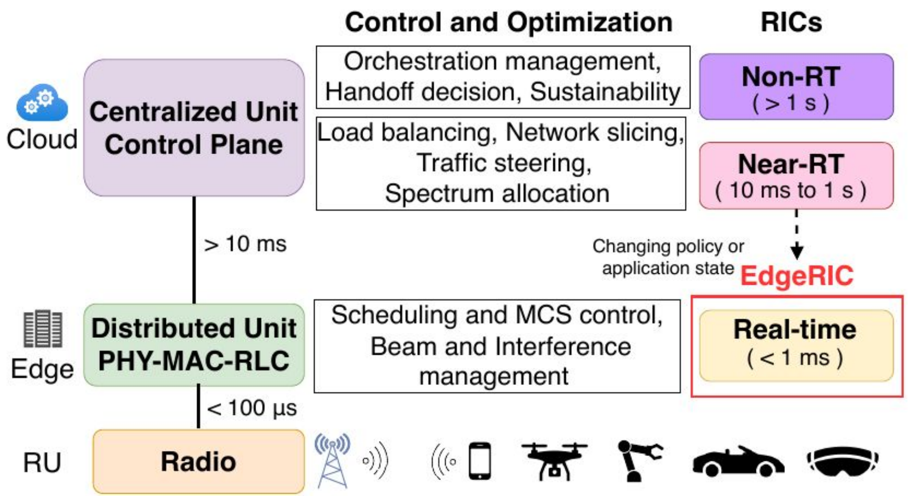
Background



RAN Intelligence Control (RIC) is a promising approach, yet current cloud-based design creates a non-real-time loop, causing delayed and coarse control.

RAN TTI is one slot: 62.5 µsec to 1 msec

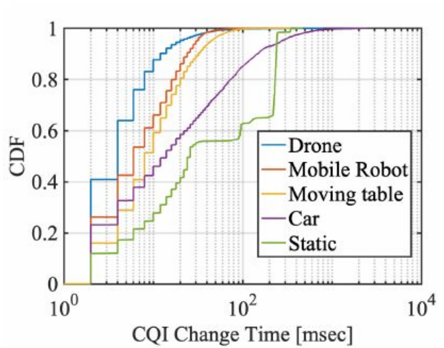
EdgeRIC

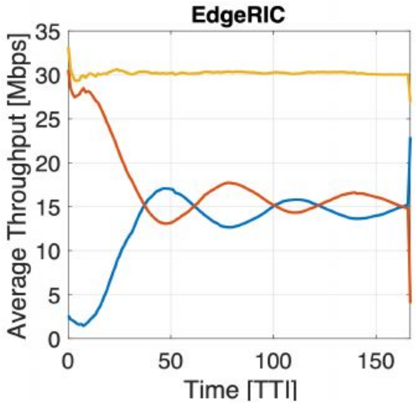


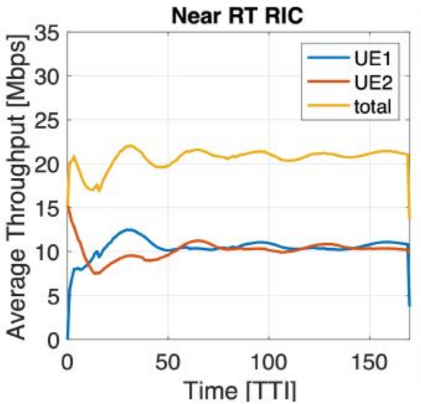
EdgeRIC: a mid layer between RAN and cloud-based central control.

- Physically close to RAN
- Disaggregated
- Ensure the control loop < 1ms
- Decoupled from RAN
- (Optional) ML-driven control

Motivation and contribution







Motivations

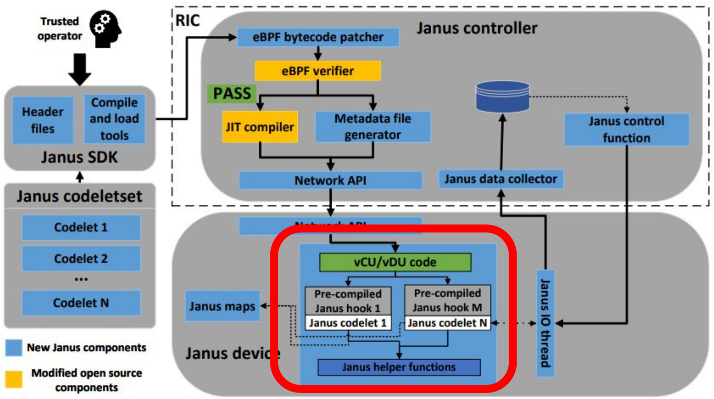
- Timely and fine-grained control
- ms-level channel dynamics
- Control loop can be nicely fit into RL

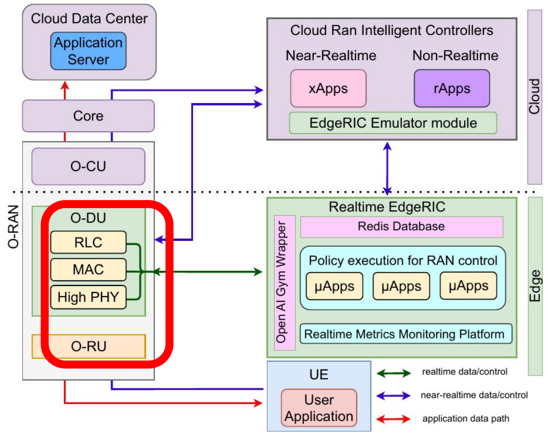
Contributions

- TTI-level sync between EdgeRIC and RAN
- RL on the control + an emulator for offline training
- Open-sourced

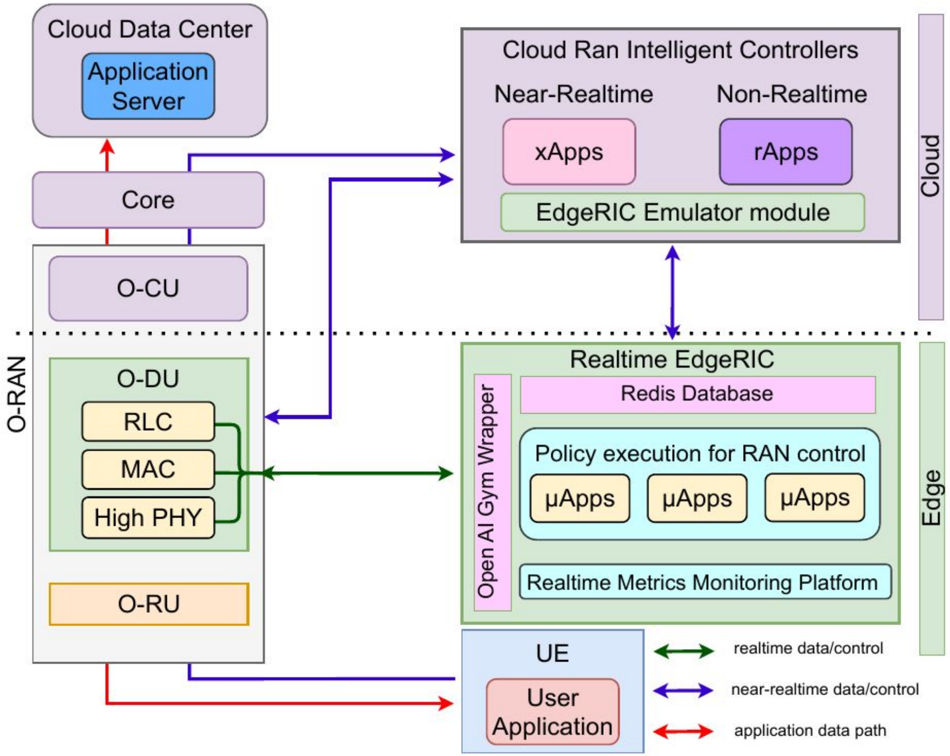
Relevant work

Framework	Connectivity to RAN stack	Monitoring and control	Application awareness	Adaptability to channel fluctuations	Full stack AI training support with real traces	Real World OTA evaluations
FlexRIC [35]	Disaggregated	10ms-1s	✓	×	×	×
ColO-RAN [32]	Disaggregated	10ms-1s	✓	×	×	×
dApps [7]	Disaggregated	6-10ms	×	✓	×	×
Janus [10]	Integrated	<1ms	×	✓	×	✓
EdgeRIC	Disaggregated	<1ms	✓	✓	✓	✓





EdgeNIC architecture

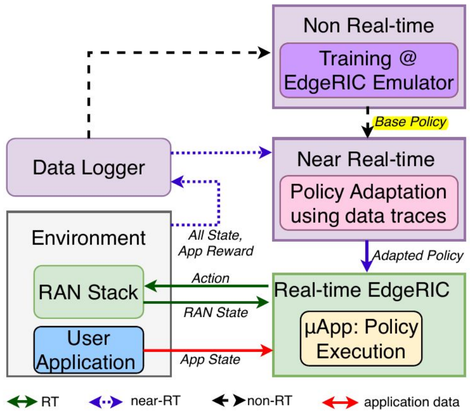


Highlights:

- Redis DB
- RT-E2 messaging
- µApp
- Gym interface
- Async Logging
- EdgeRIC emulator and RL

EdgeNIC architecture (continued)

Redis: in-memory cache for adapted policy from cloud



RT-E2 Messaging between RAN and EdgeRIC

- An *application-level* messaging technique/choice?
    - Use ZMQ library
    - Still over TCP/UDP/SCTP
- TTI-sync
    - Ground-truth RAN maintains *RANtime*
    - Ideally, EdgeRIC sends command with *RANtime* + 1 label
    - RAN retains only most recent messages and disregard any not matching current *RANtime*

Let's just go through a quick example…

RT-E2 example

RAN's tracked current *RANtime*: 10

If it receives:

- Command with *RANtime* 9
  - Discard
- Command with *RANtime* 10
  - Execute the command
- Command with *RANtime 11*
  - Retain but disregard at this moment

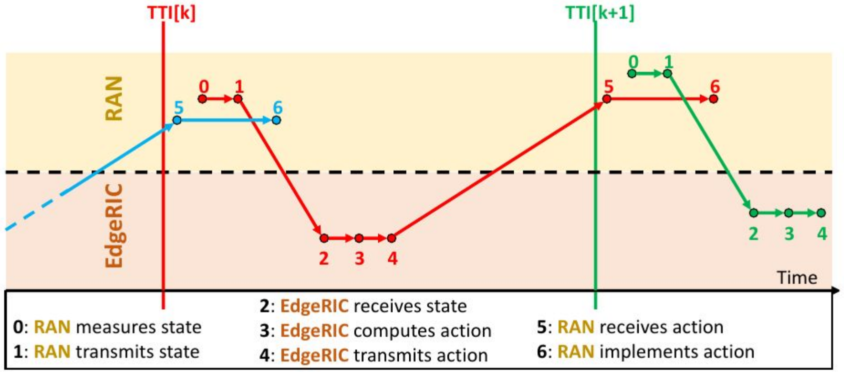


Figure 5: TTI-level events for EdgeRIC to RAN loop.

Also,

- EdgeRIC to RAN subscription: blocking
- RAN to EdgeRIC subscription: non-blocking

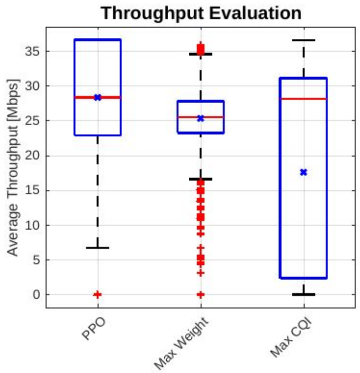
EdgeNIC architecture (continued)

µApp: essentially execute a specific policy

- RL PPO policy (talks later)
- CQI-Fair Allocation
  - $w_i[t] = CQI_i[t]$
- Proportionally-Fair Allocation
  - $w_i[t] = CQI_i[t]/AvgCQI_i[t]$
- Max-weight Allocation
  - $w_i[t] = CQI_i[t]B_i[t]$

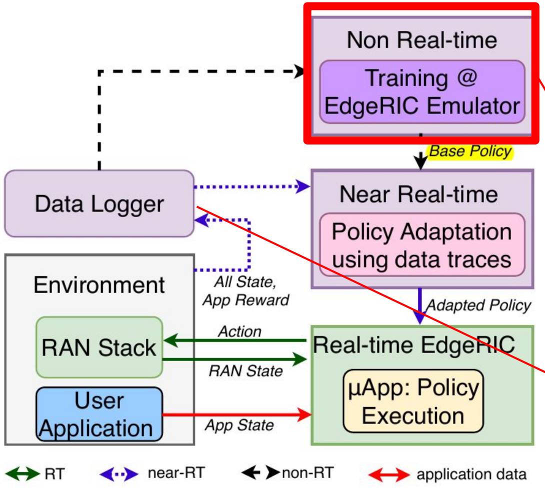
Table 3: Load: 35Mbps, Channel: 2 UE synthetic channel

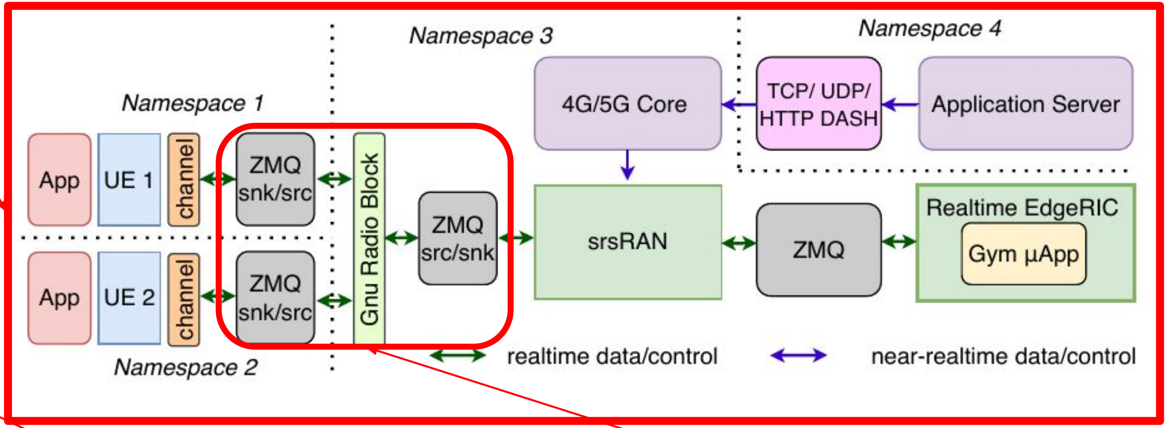
		EdgeRIC	15ms	30ms
Max CQI	Avg. Thrpt.	**32.6**	24.2	18.0
	BL[MB]	0.61	0.64	**0.57**
Prop. Fair.	Avg. Thrpt.	**30.7**	25.7	21.9
	BL[MB]	**0.65**	0.67	0.68
Max Weight	Avg. Thrpt.	**30.0**	23.3	20.9
	BL[MB]	**0.60**	0.62	0.65



PPO RL is not absolutely winning

Emulator and RL training loop





Simulate channel condition with samples from data logger

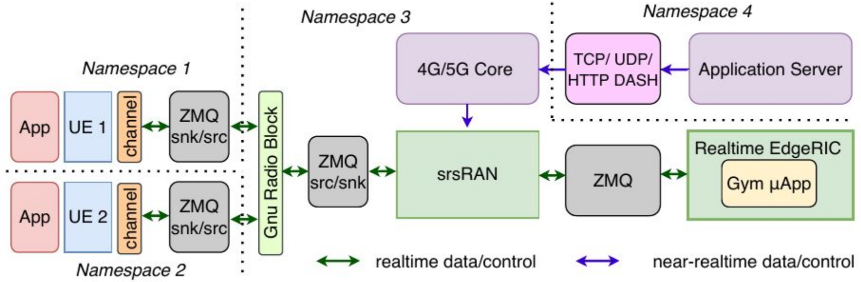
Table 2: RL Specifications: Throughput Maximization

State ($s[t]$)	$B_i[t], CQI_i[t]\ \forall i$
Action ($a[t]$)	$w_i[t]\ \forall i$
Reward ($r[t]$)	*total throughput*

Emulator and RL training loop example

Table 2: RL Specifications: Throughput Maximization

State ($s[t]$)	$B_i[t], CQI_i[t]\ \forall i$
Action ($a[t]$)	$w_i[t]\ \forall i$
Reward ($r[t]$)	*total throughput*

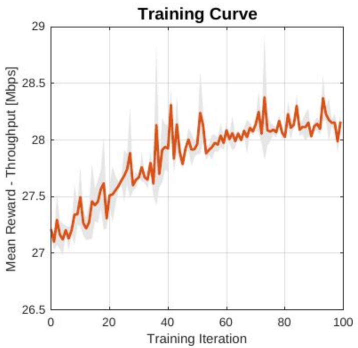


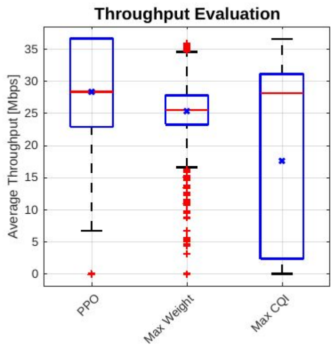
Collected CQI traces: 14,13,12,13,14,15,15,11,10,9,10,12,15,15,10,9,12,15,15...

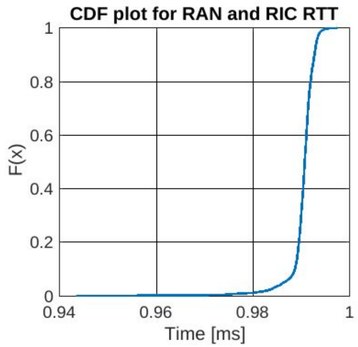
Run the emulator and PPO training...
Slot 1: {CQI(UE1) 14, Backlog(UE1) 0.1MB, reward 12Mbps} sends to Gym interface; Gym gives action of weight(UE1) being 60%.
Slot 2: UE1 is given 60% PRBs. {CQI(UE1) 13, Backlog(UE1) 0.3MB, reward 14Mbps} sends to Gym; Gym gives action of weight(UE1) being 50%.
Slot 3: ...

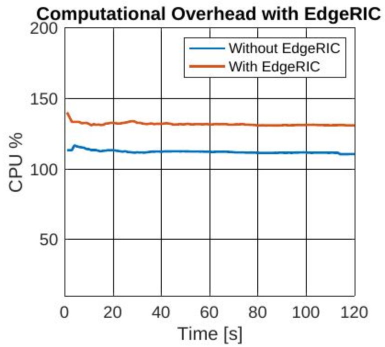
RL training and performance

“100 iterations, equivalent to 500,000 TTI samples.”
“Total training completes in about ten minutes”

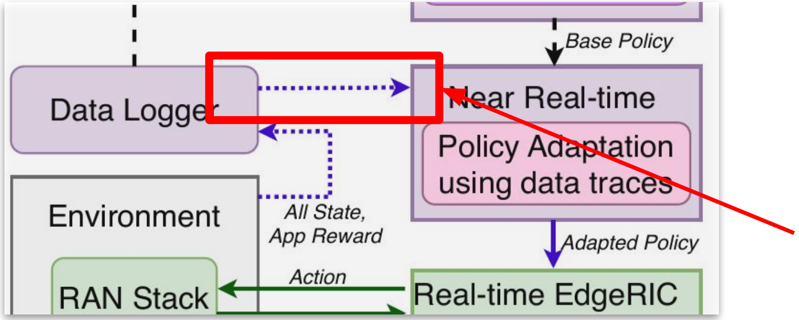








Might not work for SCS30KHz where slot = 0.5ms for their current configuration



To bridge the sim-to-real gap

Evaluation

Trace-driven emulation and Over-the-Air real tests

Table 4: Summary of all scenarios

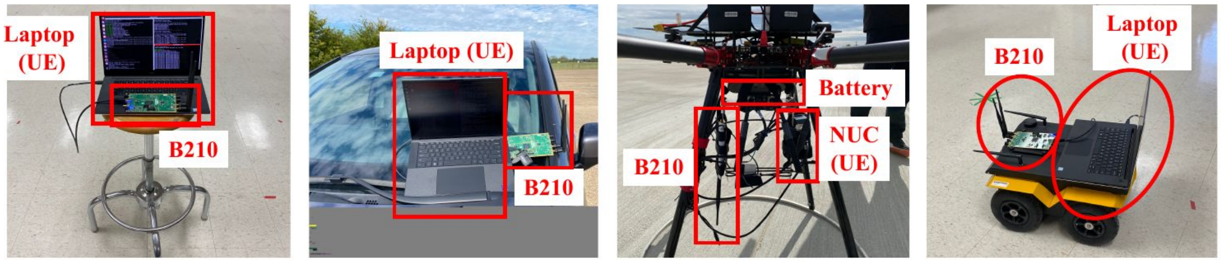
Scenario	Channel Description
Channel Traces from Experiments	
Scenario 1	2 Drone UEs
Scenario 2	2 Turntable UEs
Scenario 3	2 Car UEs and 2 Drone UEs
Scenario 4	2 Car UEs and 2 Indoor Robotic UEs
Scenario 5	2 Random Walk UEs and 2 Turntable UEs
Complete Over-the-Air Experiments	
Scenario 6	2 UEs on indoor mobile robots
Scenario 7	2 UEs on indoor stationary robots

Setup:

- Intel Xeon Gold 5218R CPU @ 2.10GHz, 20 cores
- Intel i9 CPU @ 2.4GHz, 12 cores
- both without using GPUs

For OTA:

- One X310 as the base station
- Two B210s as the UEs



(a) Turntable (b) Car (c) Drone (d) Mobile Robot

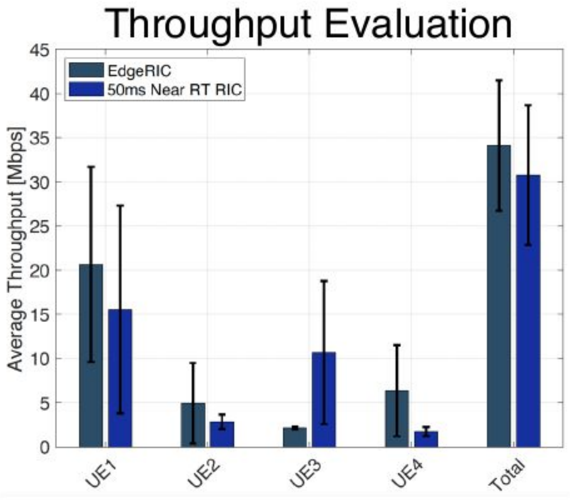
← Where they collected traces

Evaluation (continued)

Evaluate RT latency is important rather than cloud-based approach (emulation)

Table 5: Load: 35Mbps, Channel Trace: 4 Turntable UEs

		EdgeRIC	50ms	100ms
Max CQI	Avg. Thrpt.	**33.4**	21.2	29.5
	BL[MB]	1.34	**0.84**	1.12
Prop. Fair.	Avg. Thrpt.	**28.6**	26.6	23.5
	BL[MB]	1.20	1.29	**0.93**
Max Weight	Avg. Thrpt.	**33.2**	28.8	31.0
	BL[MB]	1.14	1.30	**1.12**



a) Is real-time needed?

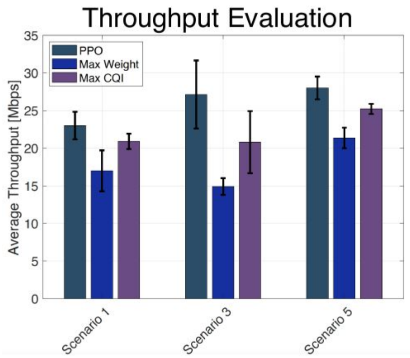
iPerf setup might introduce noise?

Evaluation (continued)

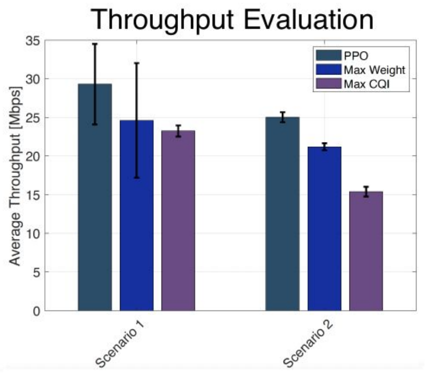
Then a holistic evaluation

Table 6: Throughput and Backlog Buffer Evaluation

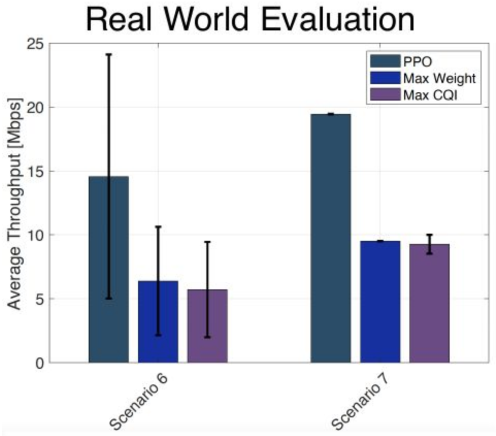
	PPO	Max Weight	Max CQI
Realistic Channel Traces			
Scenario 1	**29.1/0.38**	26.1/0.53	14.9/0.39
Scenario 2	30.5/**0.38**	**31.9**/0.43	14.42/0.39
Scenario 3	**25.3**/1.5	22.9/1.3	18.67/**0.97**
Scenario 4	**25.9**/1.5	23.9/1.21	20.3/**1.05**
Scenario 5	**28.5/0.96**	26.3/1.46	23.3/1.01
Over the Air Experiments			
Scenario 6	**14.6/0.19**	6.4/0.45	5.7/0.44
Scenario 7	**19.33/0.05**	10.71/0.34	9.06/0.35



b) Is RL useful?



c) Is RL generalizable?



d) Does EdgeRIC work in real world?

Max Weight is close to PPO perf. How many rounds they repeated for each case for each technique?

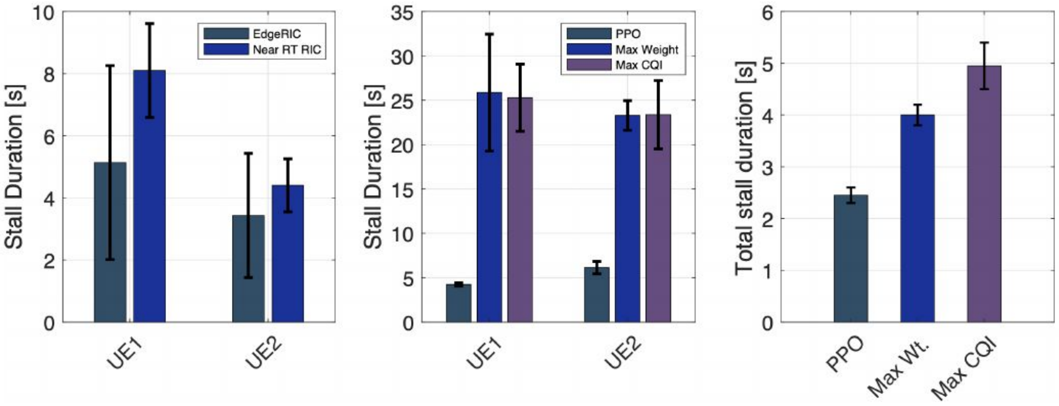
App-aware RAN/cross-layer optimization

Table 7: RL specifications: Video Streaming

State ($s[t]$)	$B_i[t], CQI_i[t], MB_i[t]\ \forall i$
Action ($a[t]$)	$w_i[t]\ \forall i$
Reward ($\sum_i r_i[t]$)	$r_i[t] = \begin{cases} -20, & \text{if } MB_i[t] < 2\ sec \\ +2, & \text{otherwise} \end{cases} \quad \forall i$

Media buffer data length

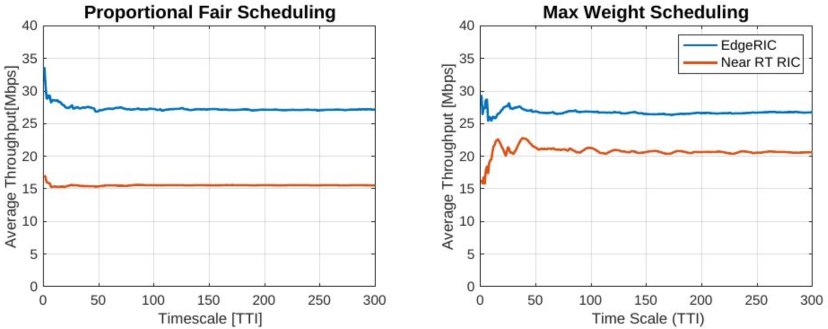
Test setup: 2 video watchers and 2 iPerf users



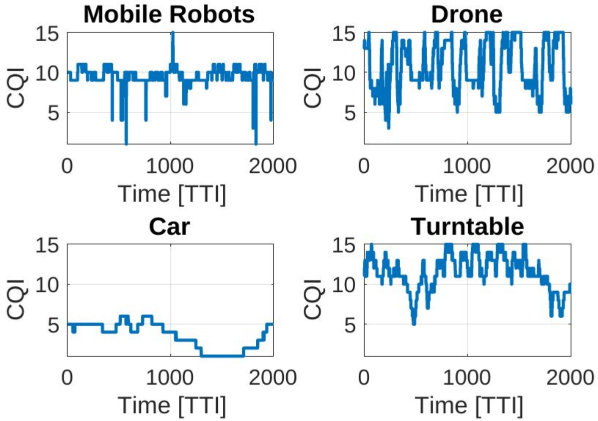
a) Is real-time needed? b) Is RL useful? c) Does EdgeRIC work in real world?

Q: while the RL focus on optimizing for video streaming, how about the iPerf users? They are legit users as well...

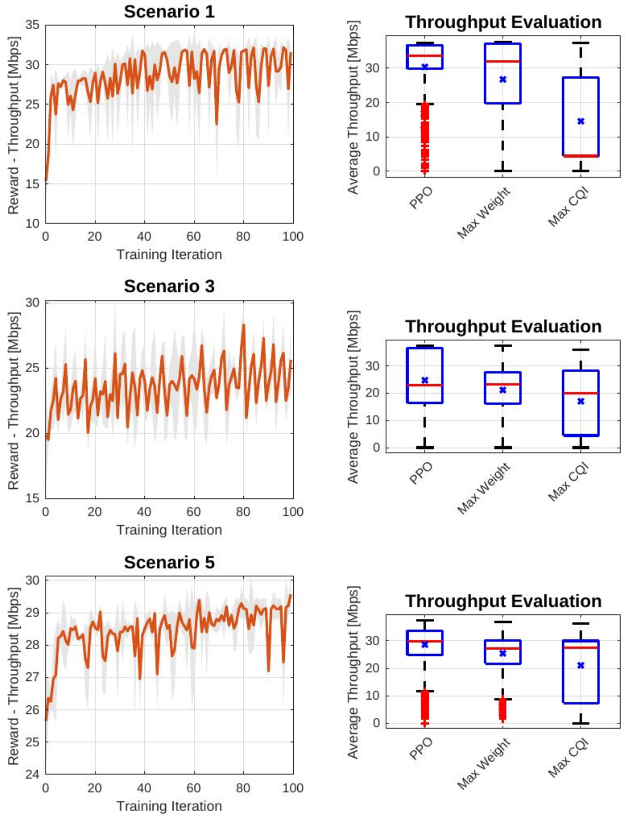
Appendix (good insights)



Throughput trace



CQI trace of different scenarios



RL training on DigitalTwin

Opinion

- The paper's motivation and architecture design is robust and good to me
  - E.g., RL loop; flow of each component; low-latency messaging

- The use of ZMQ and advertisement is a little over-marketing
  - Not something original; a messaging service choice at the application-level

- Had questions about their evaluations
  - Seems each result is from a single run; how many they repeated?
  - Also the gain is not decisive

- They claimed RT-E2 latency is sub-millisecond. They evaluated EdgeRIC and RAN on the same host. Every computer can achieve localhost ping <1ms latency…

Discussion (deep dive)

Let's look at Perusall.

How do you think of the overall architecture to achieve low latency?

How do you think of the RL design?

How do you think of the evaluation?

Conclusion

- RIC is a promising technique controlling RAN. Now the cloud-based approaches impose unbearable high latency.
- They introduce EdgeRIC, which puts the RIC on the edge, preferably co-located with RAN.
- The control loop can be nicely represented as a RL feedback loop. They tried PPO-based RL with EdgeRIC.
- Evaluations show EdgeRIC’s low latency is crucial; the RL policy is also beneficial.