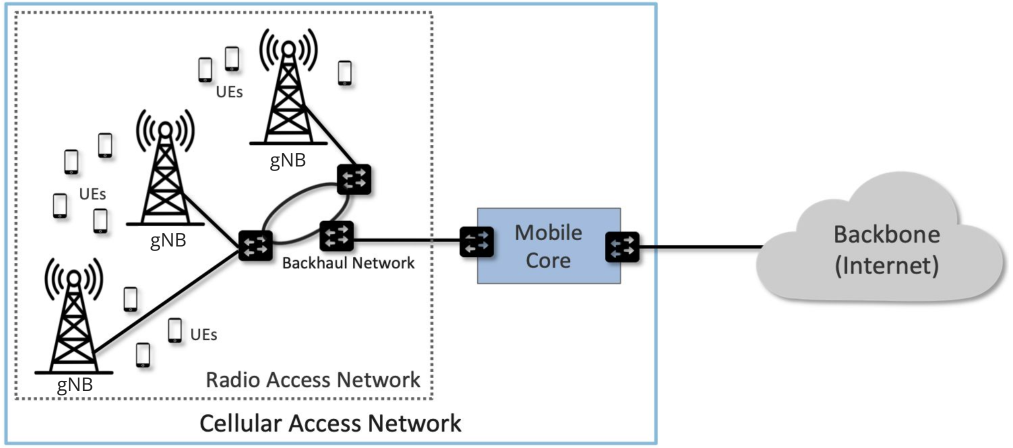
Application-Level Service Assurance with 5G RAN Slicing

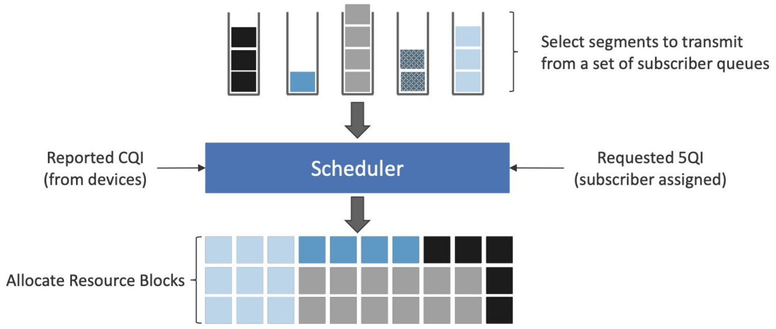
Presenter: Haoran Wan

Background – 5G Radio Access Network (RAN)



5G RAN Architecture [1]

In the gNB, the MAC scheduler maps the packet to the physical resource.

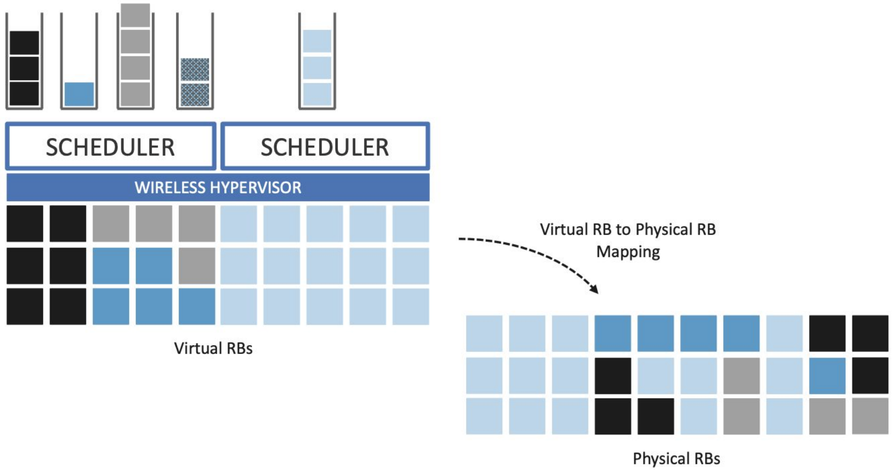

Select segments to transmit from a set of subscriber queues
Reported CQI (from devices)
Scheduler
Requested 5QI (subscriber assigned)
Allocate Resource Blocks

MAC scheduler [1]

[1] Peterson, Larry, Oguz Sunay, and Bruce Davie. Private 5G: A Systems Approach. Systems Approach, LLC, 2023.

Background – RAN Slicing

5G has introduced a mechanism that allows the underlying resources (in this case radio spectrum) to be “sliced” between different uses.

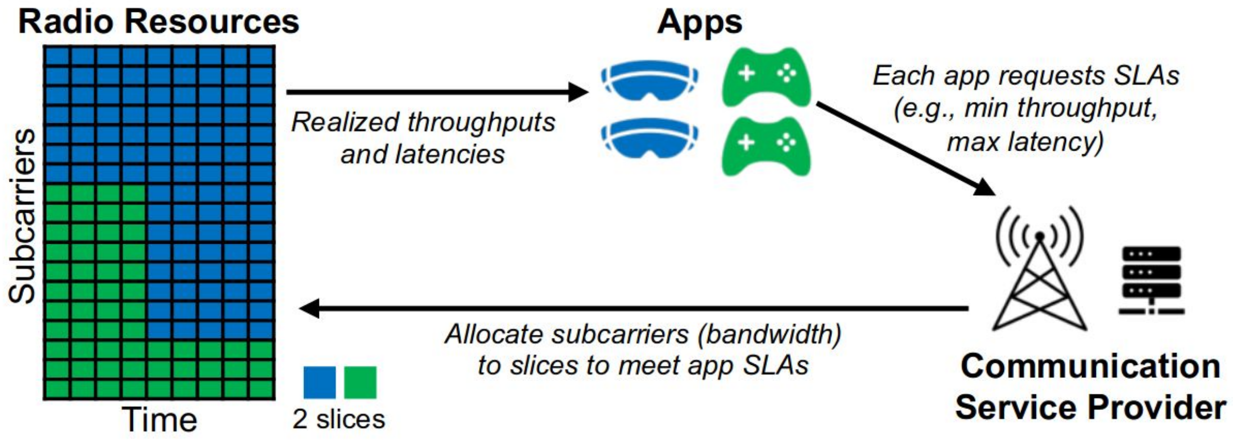


The resource is sliced into 2 equal halves, each scheduled for different purposes. [1]

[1] Peterson, Larry, Oguz Sunay, and Bruce Davie. Private 5G: A Systems Approach. Systems Approach, LLC, 2023.

Zipper

This paper presents ***Zipper***, a real-time RAN slicing system that **dynamically allocates PRBs** (i.e., bandwidth) to network slices to **ensure app-level throughput and latency Service Level Assurances (SLAs)** for every app in every slice.


Radio Resources
Subcarriers
Time
2 slices
Realized throughputs and latencies
Apps
Each app requests SLAs (e.g., min throughput, max latency)
Allocate subcarriers (bandwidth) to slices to meet app SLAs
Communication Service Provider

Motivation & Challenges

Existing slicing approach is a bit of coarse-grained, where all the UEs within one slice have the same throughput-latency assurance. But we want to make sure each application gets what it wants.

Challenges:

- **Search space complexity**. One slice for each application – state space grows exponentially.
- **Determining resource availability**. Operators need a way to determine if the RAN has resources to accommodate the SLAs of an incoming app.

Problem Assumption

Assumptions:

i) Each app’s wireless channel quality fluctuates;

ii) Apps join and leave the network asynchronously;

iii) The operator configures its RAN with a set of slices;

iv) The operator configures each slice with a particular MAC scheduler.

Problem Formalization

Each app selects a slice based on the throughput and latency requirements.

$$
\begin{aligned}
\underset{S_s(t),B_s(t)\,\forall s\in S\,\forall t}{\operatorname{argmin}} \quad & \sum_{t}\sum_{s\in S} B_s(t) & \\
\text{s.t.} \quad & \sum_{s\in S} B_s(t) \leq B & \forall t \\
& \overline{x}_a(t) \geq x_a^{SLA} & \forall a\in A_s\ \forall s\in S\ \forall t \\
& \overline{d}_a(t) \leq d_a^{SLA} & \forall \boxed{a} \in \boxed{A_s}\ \forall \boxed{s} \in \boxed{S}\ \forall t,
\end{aligned}
$$

App a and app set

Slice s and slice set

Problem Formalization

Each app selects a slice based on the throughput and latency requirements.

$$\underset{S_s(t),B_s(t)\,\forall s\in S\,\forall t}{\operatorname{argmin}} \quad \sum_t \sum_{s\in S} \boxed{B_s(t)}$$

Slice s' bandwidth allocation at time t

$$\text{s.t.} \quad \sum_{s\in S} B_s(t) \le \boxed{B} \qquad \forall t$$

Total bandwidth of the cell

$$\boxed{\bar{x}_a(t)} \ge \boxed{x_a^{SLA}} \qquad \forall a\in A_s\ \forall s\in S\ \forall t$$

$$\boxed{\bar{d}_a(t)} \le \boxed{d_a^{SLA}} \qquad \forall \boxed{a}\in\boxed{A_s}\ \forall\boxed{s}\in\boxed{S}\ \forall t,$$

App a's average throughput at time t

App a's average latency at time t

App a and app set

Slice s and slice set

App a's latency requirement

App a's throughput requirement

Minimize the bandwidth usage over time.

At any time, sliced bandwidth sum shouldn't exceed the cell's bandwidth.

Accommodate each app's throughput requirement

Accommodate each app's latency requirement

Relaxation

As UE coming in, it’s getting harder to accommodate all the throughput and latency requirement, given the limited RAN resource.

Introduce penalty terms into the optimization goal:

$$f_x^a(t) = \left| \min\left(\bar{x}_a(t) - x_a^{SLA}, 0\right) / x_a^{SLA} \right|$$

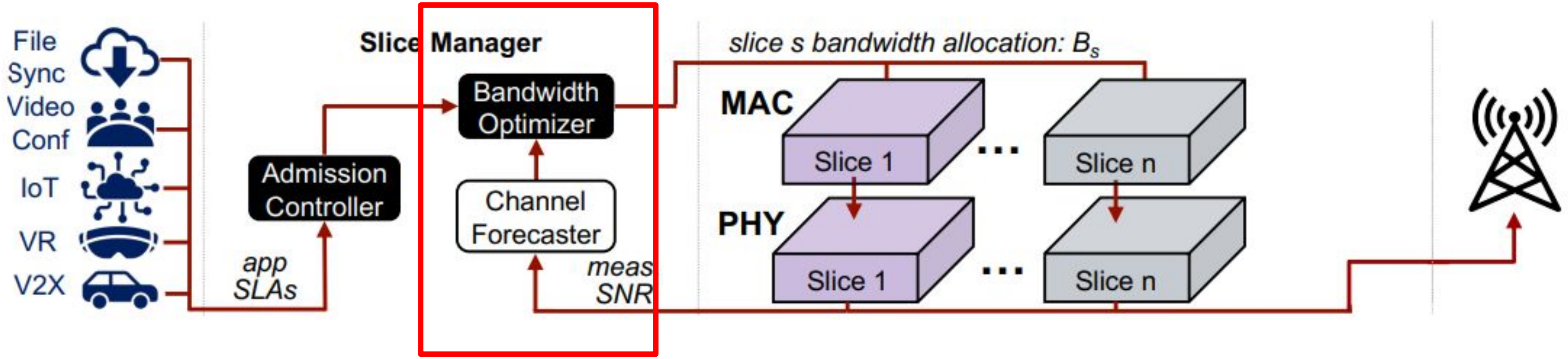
$$f_d^a(t) = \left| \min\left(d_a^{SLA} - \bar{d}_a(t), 0\right) / d_a^{SLA} \right|$$

Approaches

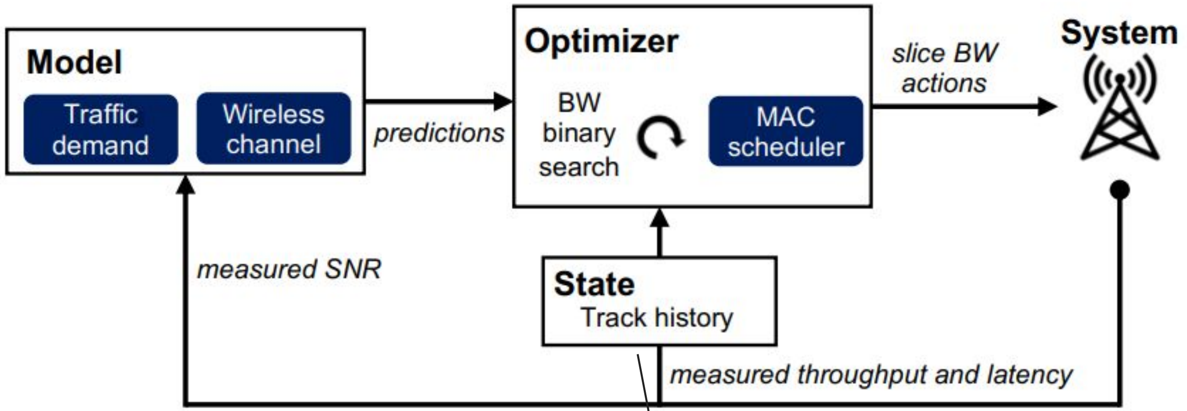
To achieve the goal, they

- Formulate SLA-compliant PRB allocation as a **model predictive control** (MPC) problem.
- Propose an **efficient control algorithm** to allocate PRBs (i.e., bandwidth) amongst the slices.
- Design a family of deep neural networks (DNNs) to **forecast RAN resource availability** and predict the distribution of required PRBs.

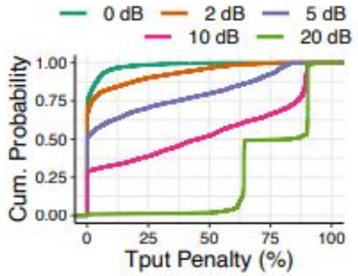
Overall Architecture



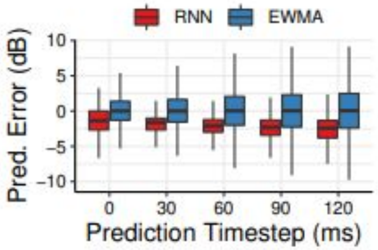
Model Predictive Control



Use an RNN to forecasting the wireless channel



(a) Resilience to SNR error.



(b) SNR prediction error.

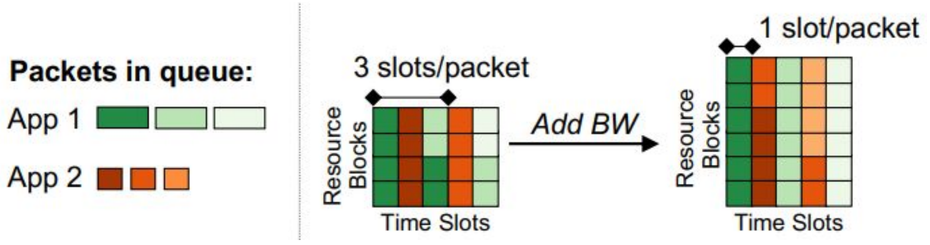
State space:
(i) the average throughput and average latency experienced by each app over the past Tw slots;
(ii) the average signal-to-noise ratio (SNR) of each user over Tw as a measure of the channel quality;
(iii) the incoming data traffic.

Tuning Slice Bandwidths Efficiently (Optimizer)

How to find the most spectrally-efficient slice bw allocation? (optimization goal)

Design is based on one observations:

Monotonicity – both app throughput and app latency are monotonic functions of slice bandwidth.



Because app throughput and latency vary monotonically with slice bandwidth, there exists a **minimum bandwidth Bs** for s ∈ S that satisfies all SLAs.

Tuning Slice Bandwidths Efficiently (Optimizer)

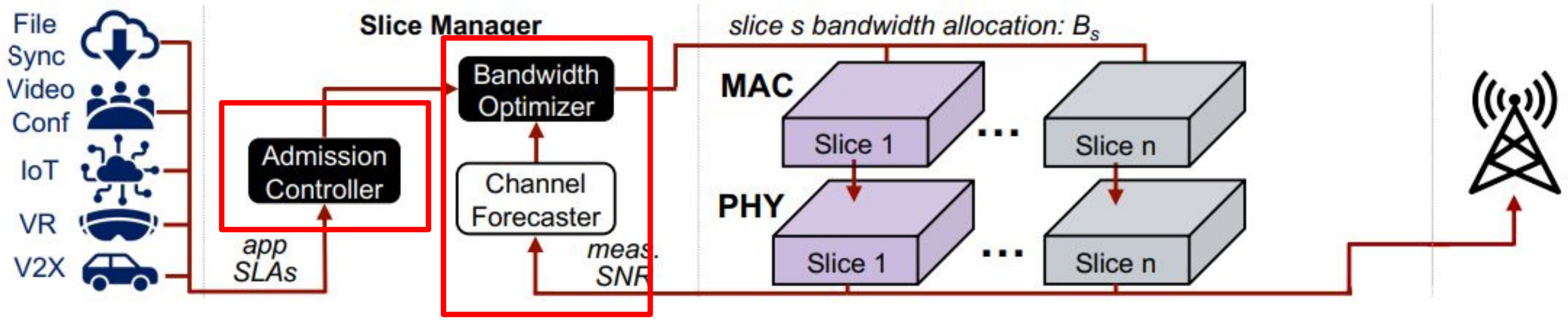
Computing bandwidths – Zipper treats slice MAC schedulers as a black box; the search algorithm does not need to know the scheduling logic:

Query MAC scheduler → Evaluate the results' penalty → Find the most spectral efficient option through binary search of allocated bandwidth.

Resolving conflicts – If the sum of bandwidth allocation is bigger than the cell's bandwidth:

Deduct the **excess bandwidth in proportion** to the sliced bandwidth.

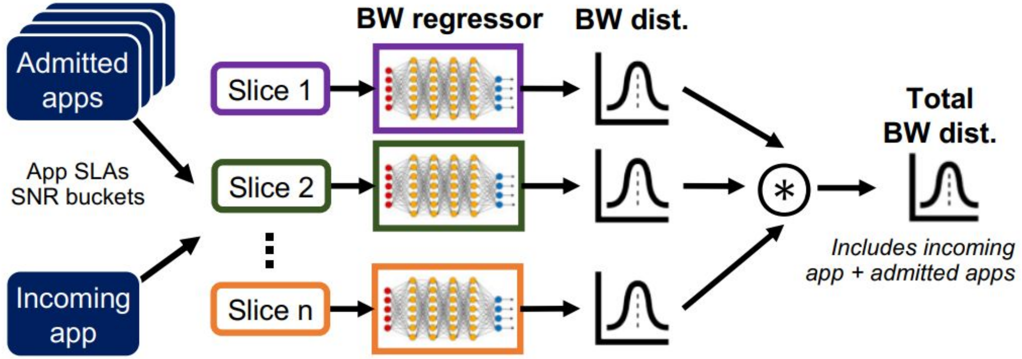
Overall Architecture



Forecasting RAN Resource Availability (Admission Ctrler)

When there is new app coming, Zipper needs to determine if the RAN has the resource to accommodate its requirements.

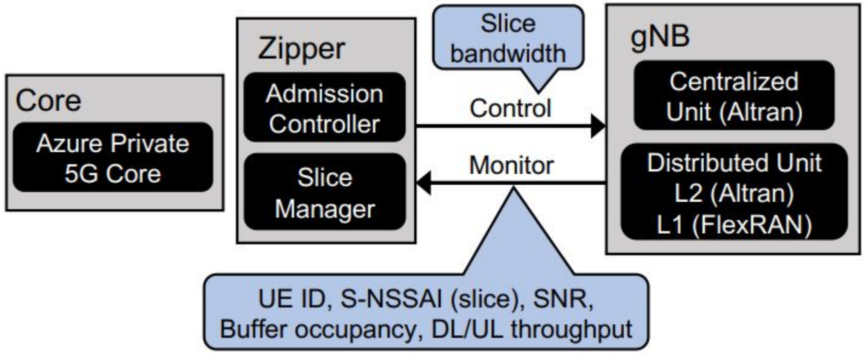
Predicting bandwidth statistics. Zipper estimates the distribution of bandwidths, i.e., number of PRBs, that each slice will require over a predetermined contract duration—including the incoming app.



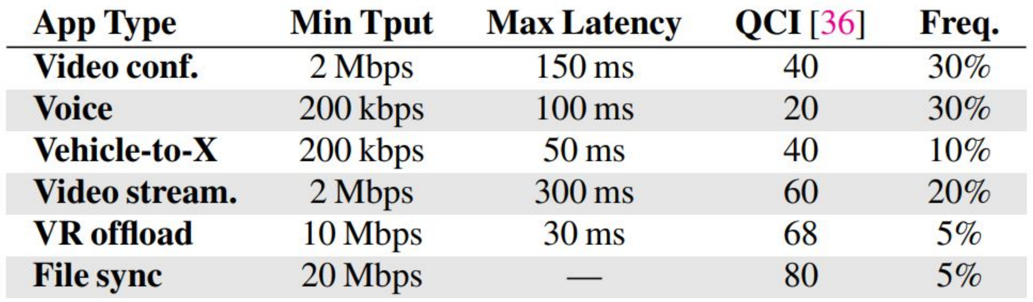
Implementation

Some major components:

- Foxconn RPQN-7800 5G Open Radio Units (RUs). Support O-RAN split 7.2x.
- Altran 5G vRAN software.
- Azure Private 5G Core (AP5GC).
- Zipper slice manager.


Core
Azure Private 5G Core
Zipper
Admission Controller
Slice Manager
Slice bandwidth
Control
Monitor
gNB
Centralized Unit (Altran)
Distributed Unit L2 (Altran) L1 (FlexRAN)
UE ID, S-NSSAI (slice), SNR, Buffer occupancy, DL/UL throughput

Evaluation – Setup



App Type	Min Tput	Max Latency	QCI [36]	Freq.
Video conf.	2 Mbps	150 ms	40	30%
Voice	200 kbps	100 ms	20	30%
Vehicle-to-X	200 kbps	50 ms	40	10%
Video stream.	2 Mbps	300 ms	60	20%
VR offload	10 Mbps	30 ms	68	5%
File sync	20 Mbps	—	80	5%

Emulation. App data is emulated.

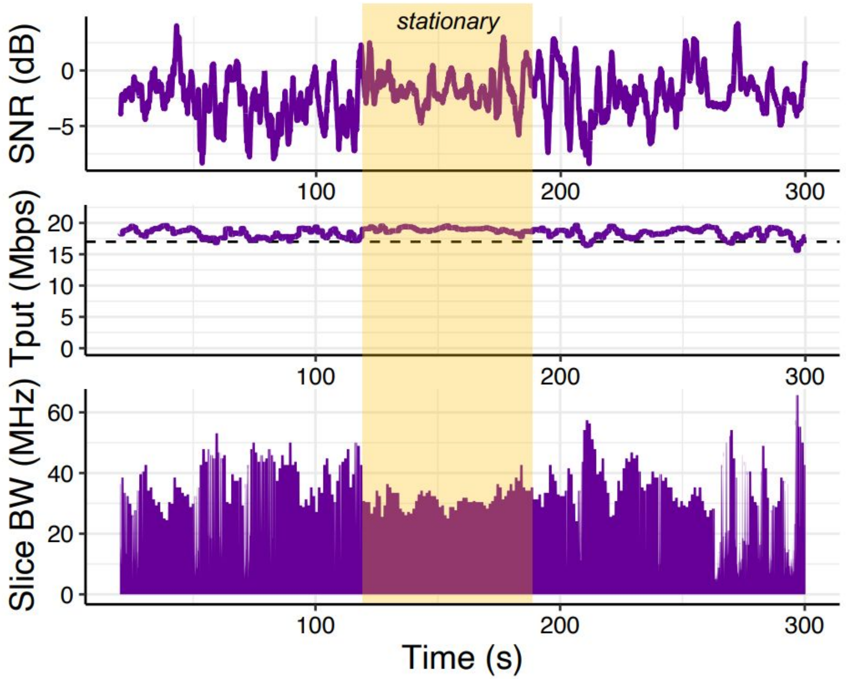
Apps. 6 representative apps.

SNR Traces. Publicly available dataset.

Base station configuration. 100MHz, 4x4 MIMO, 30kHz subcarrier spacing.

Evaluation – End-to-end Evaluation

Iperf flow, walk around and stand in a fixed location.



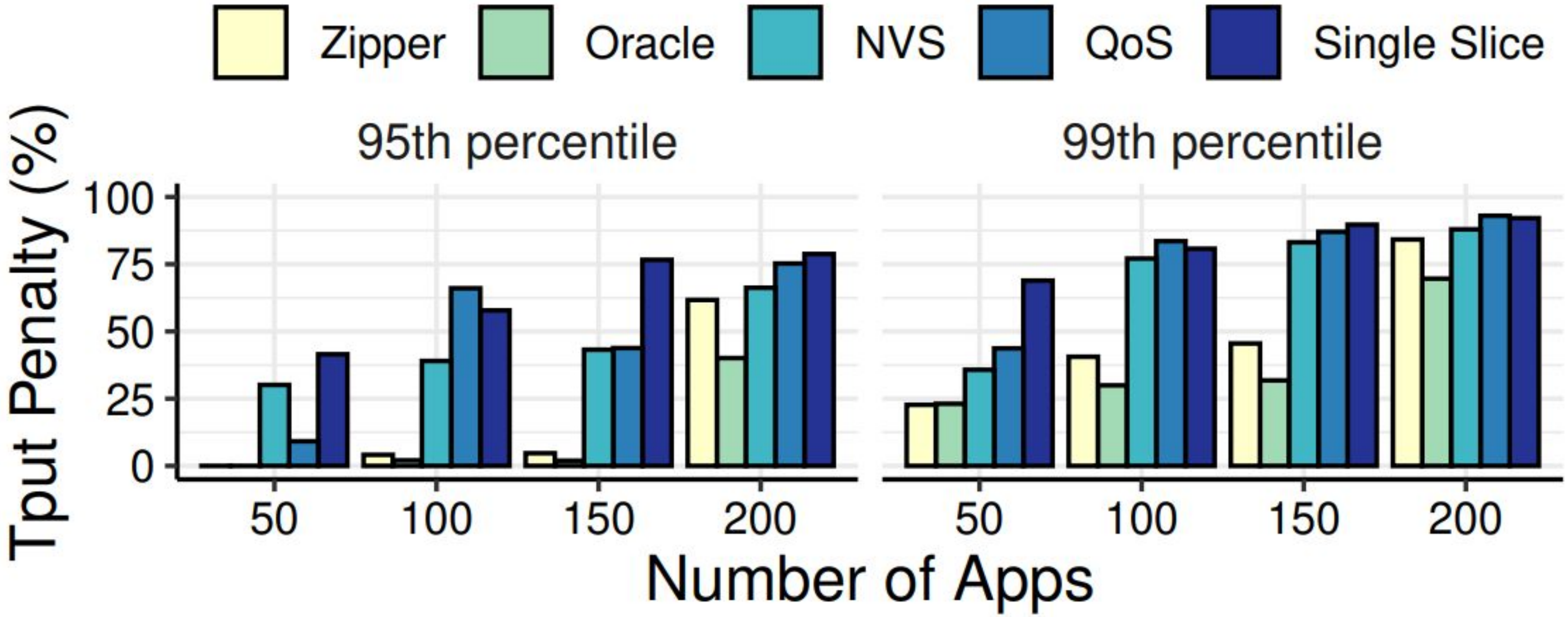
Evaluation – SLA Compliance

Baselines:

- Single Slice policy
- QoS policy
- NVS policy
- Oracle policy

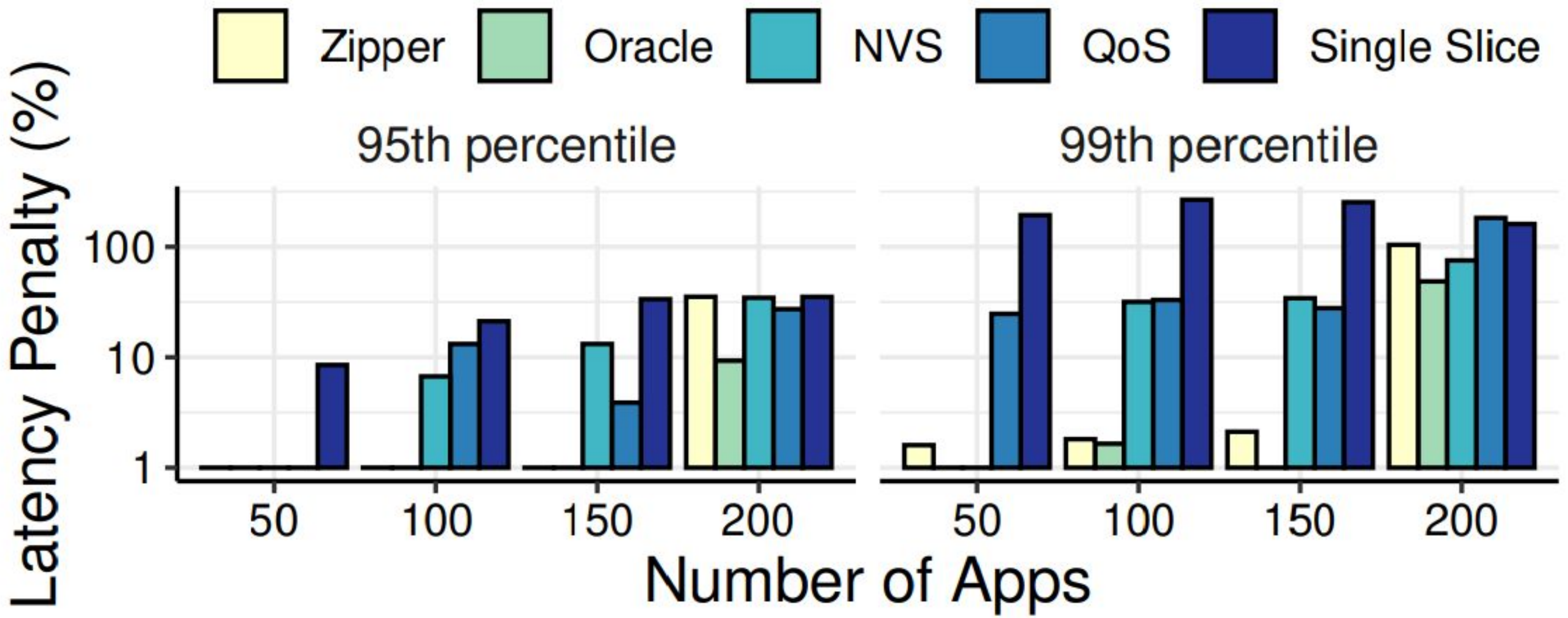
Evaluation – SLA Compliance (Throughput Penalty)

Throughput penalty.



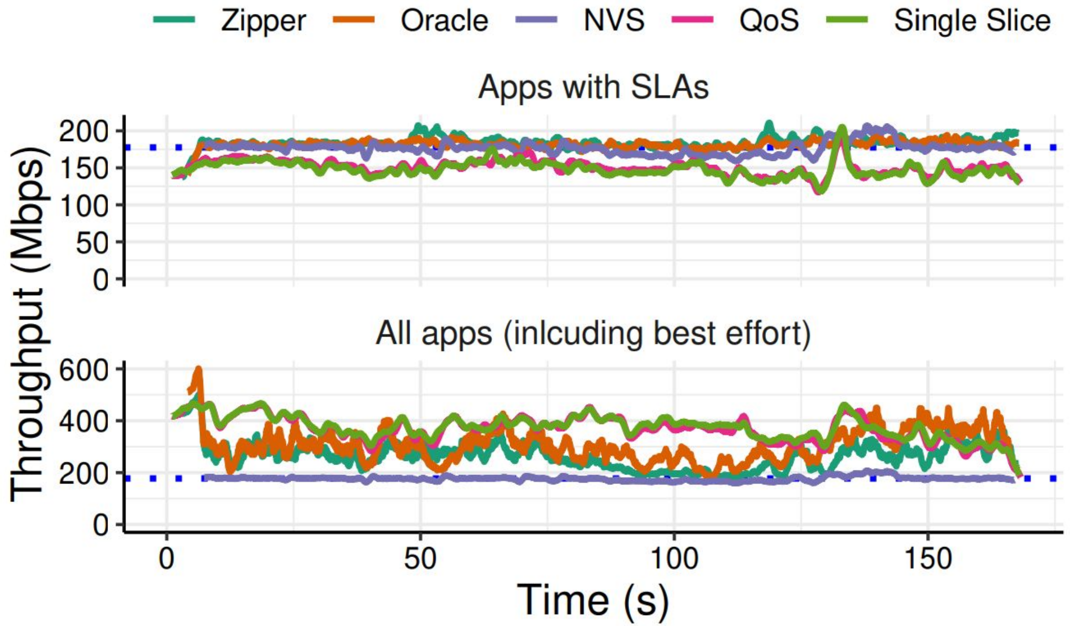
Evaluation – SLA Compliance (Latency Penalty)

Latency penalty.



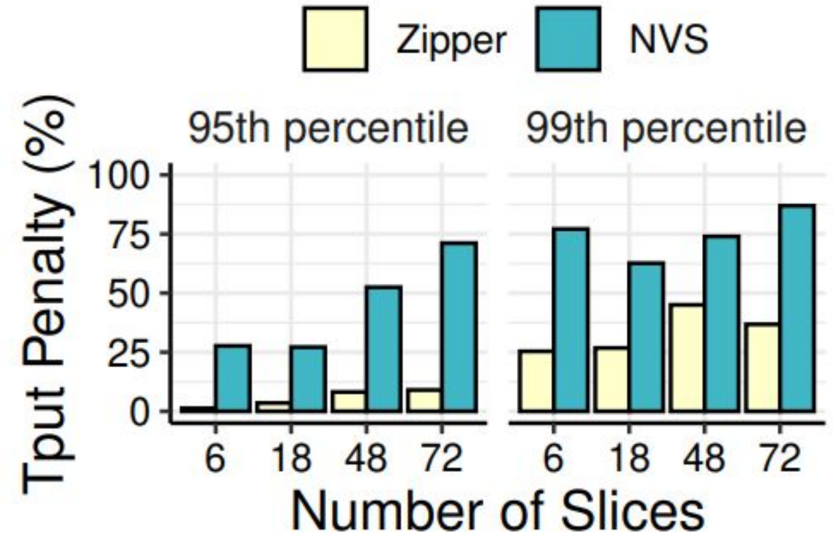
Evaluation – SLA Compliance (RAN utilization)

Zipper accommodates the SLA of apps and utilizes spectrum 50% better than NVS and 30% worse than Single Slice or QoS.

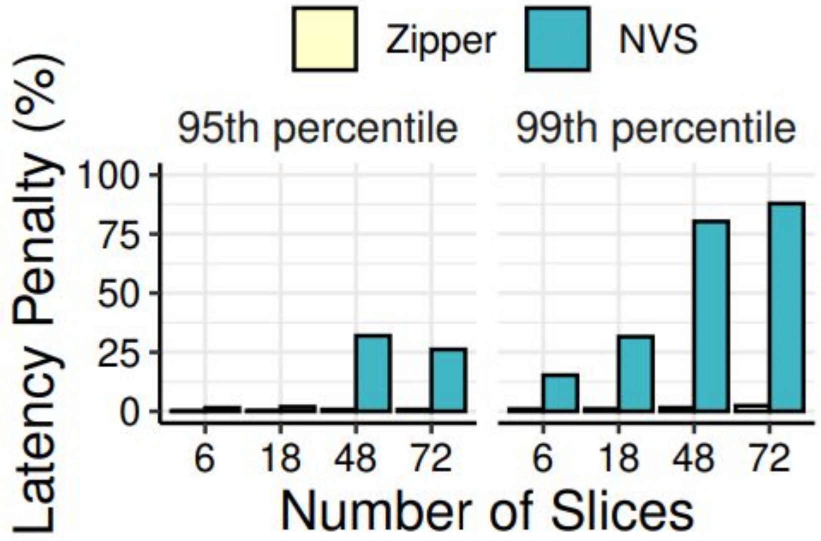


Evaluation – SLA Compliance (# of Slices)

Zipper’s performance is invariant to the number of slices.



(a) Throughput penalty



(b) Latency penalty

Evaluation – Forecasting RAN Resource Availability

Compare Zipper against a **“conditional”** resource availability primitive:

(i) admits the incoming app into a best effort slice,

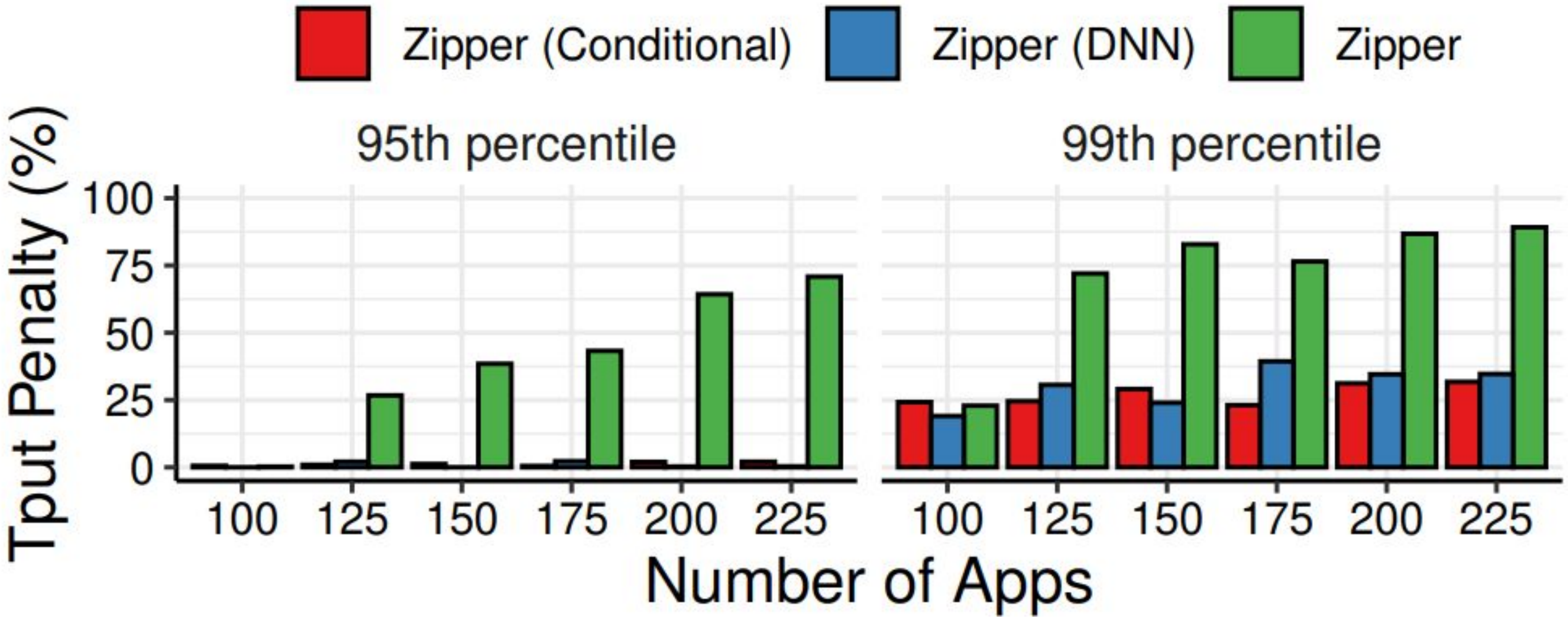
(ii) measures its throughput and latency penalties for 10 seconds, and

(iii) then admits in FCFS order if it incurs zero penalty or leaves it in the best effort slice otherwise.

And Zipper with **“no admission controller”**.

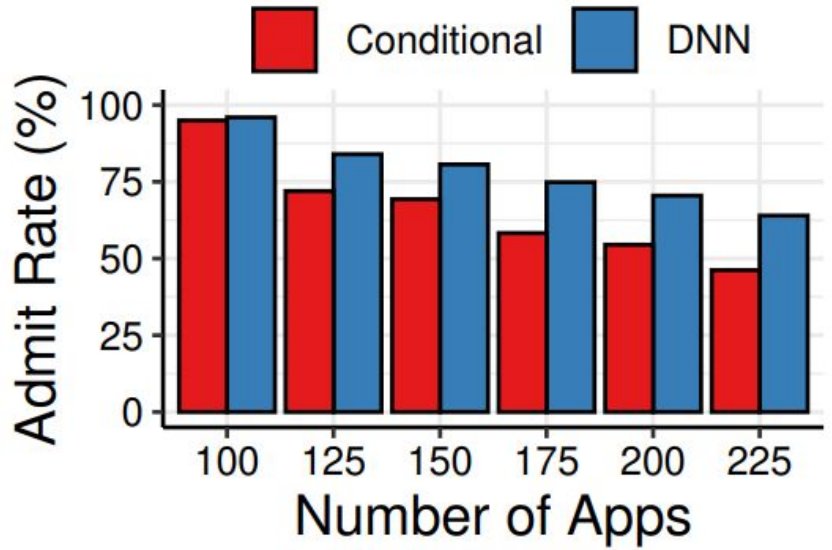
Evaluation – Forecasting RAN Resource Availability

Throughput penalty of 3 different methods.

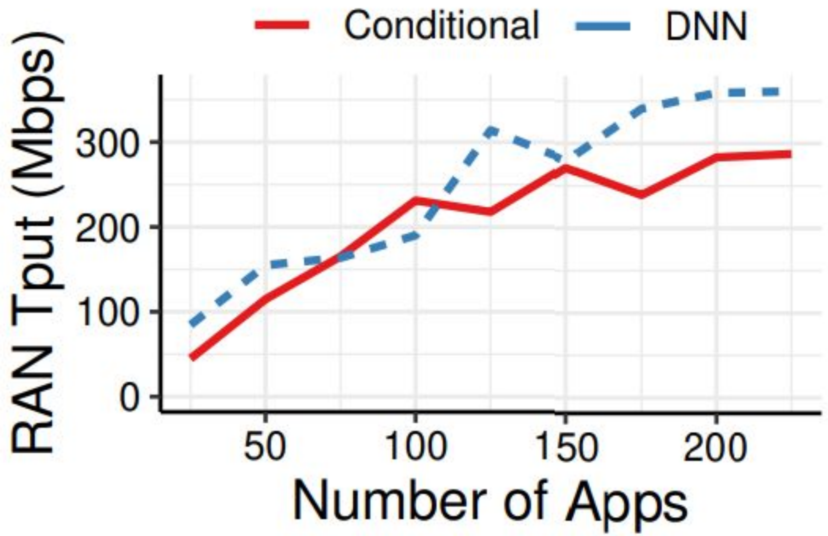


Evaluation – Forecasting RAN Resource Availability

Admission rate and RAN utilization.



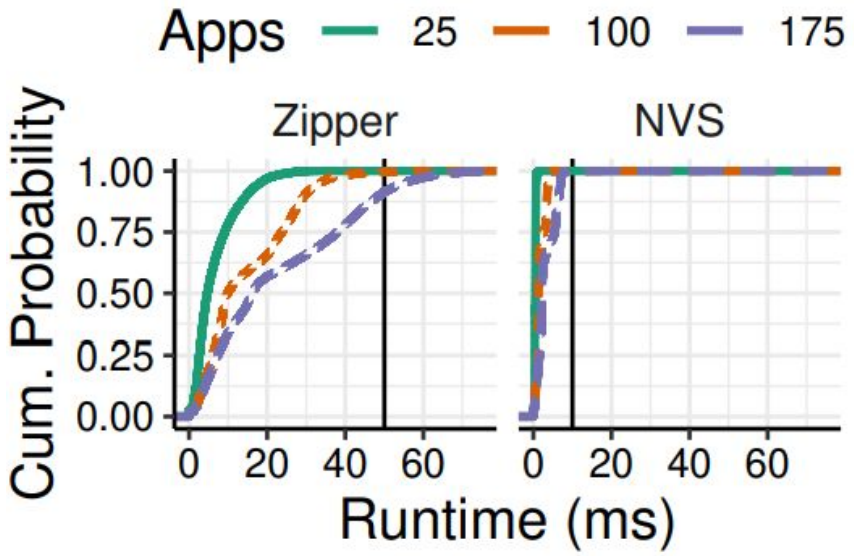
(a) Admit rate



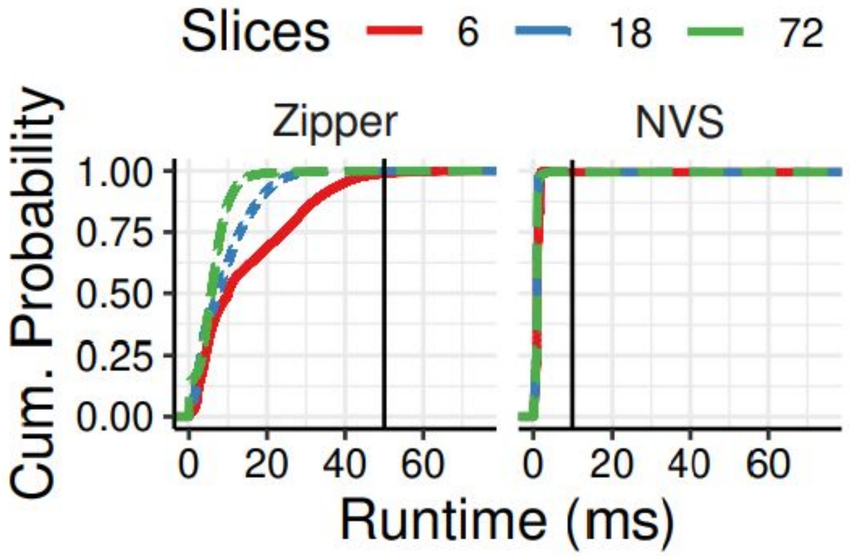
(b) RAN utilization

Evaluation – Microbenchmark

Computation time – longer than NVS but still practical.



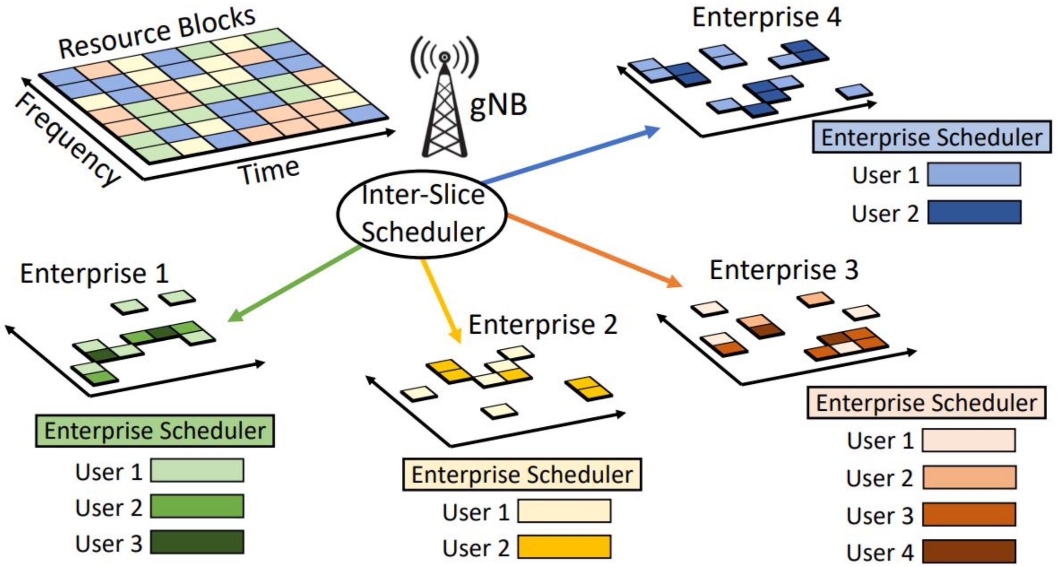
(a) Number of apps



(b) Number of slices

Related Work

- RadioSaber: directly use UE’s channel condition for VRB to PRB scheduling to maximize the UE’s throughput.



Opinion

The paper is well-designed but suffers from the lack knowledge of MAC scheduler.

- For slicing optimizer: They have to query MAC scheduler for tentative results for searching.
- For admission control: They have to train RNN to mimic the behavior of MAC to check availability.

Predicting the wireless channel within the DDL is a hard task and their result looks good.

Conclusion

- Zipper achieves application-level service assurance through noval PRB allocation.
- Zipper introduces a primitive to forecast RAN resource availability.
- Zipper is implemented and evaluated on a production-grade 5G vRAN.

Discussion Questions

- How do you feel about the evaluation of this work? What additional evaluation(s) can be added to make the paper better?

https://app.perusall.com/courses/cos597s_f2024-advanced-topics-in-computer-science-recent-advances-in-wireless-networks/zipper?assignmentId=LmAiCmgx43j8S4xWC&part=1