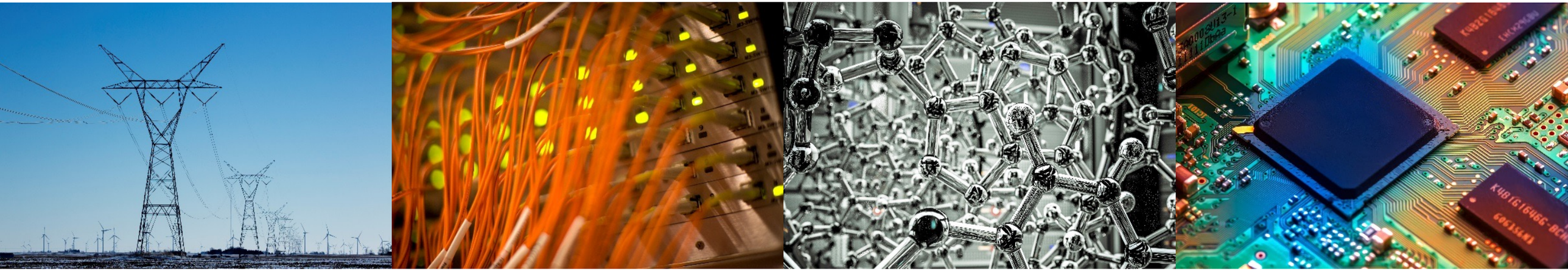
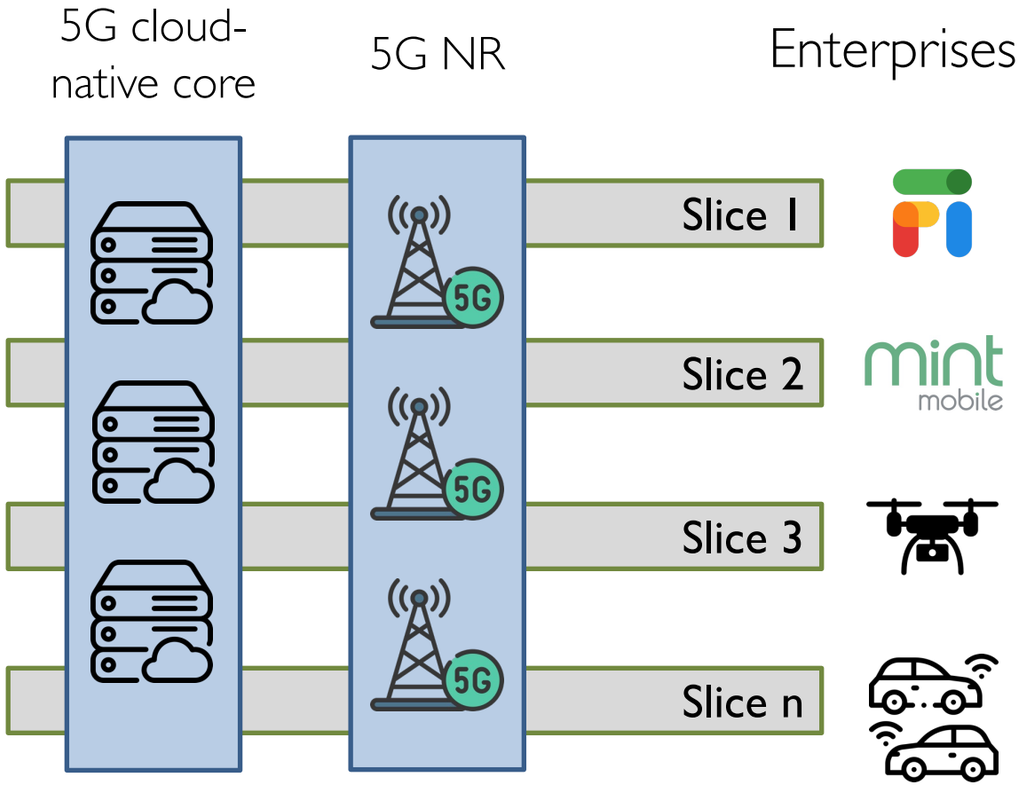
Channel-Aware 5G RAN Slicing with Customizable Schedulers

Yongzhou Chen, Ruihao Yao, Haitham Hassanieh, Radhika Mittal



ILLINOIS
Electrical & Computer Engineering
GRAINGER COLLEGE OF ENGINEERING

5G Slicing

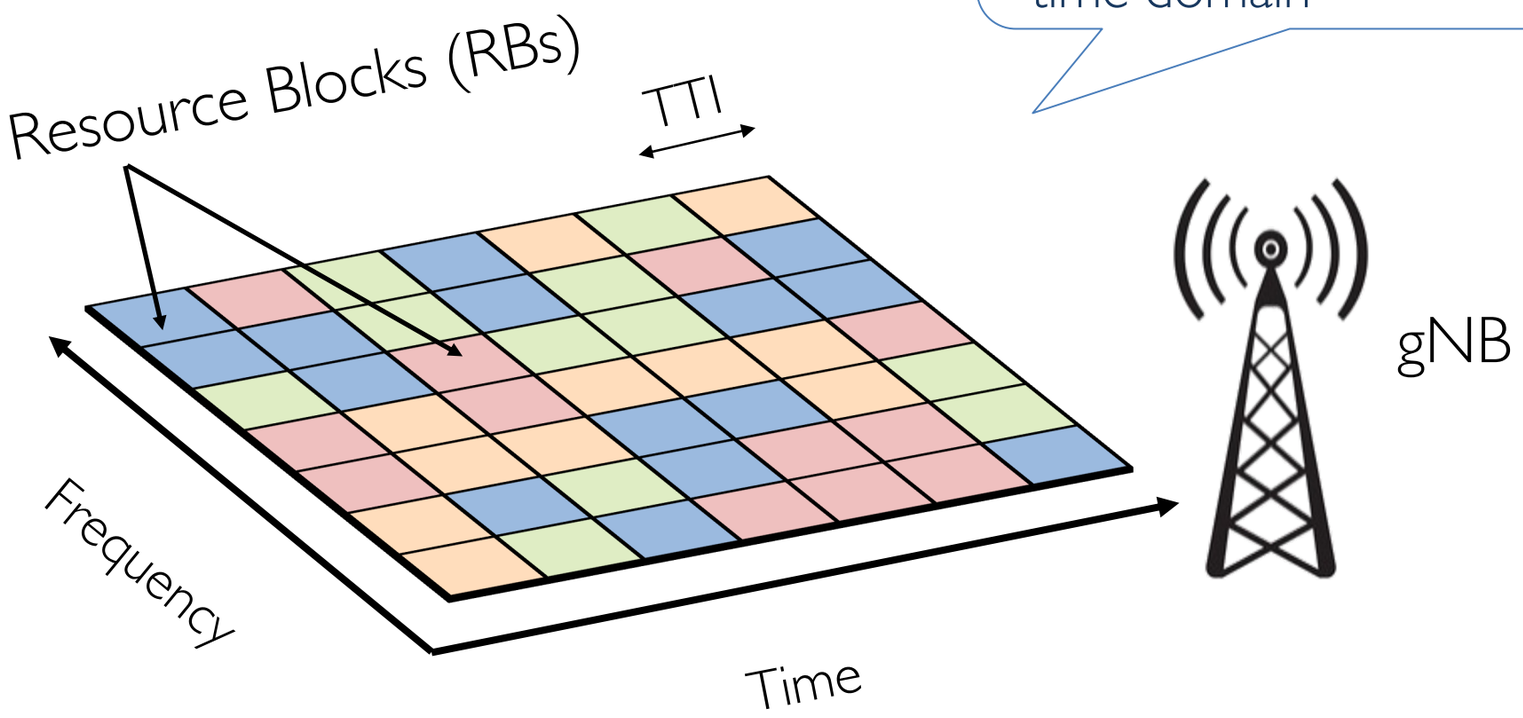


Network slicing is a key feature in 5G standards:

- The operator divides network resources among different groups of users (called slices or enterprises)
- Each slice can customize its virtual network by dividing its allocated resources across its own users

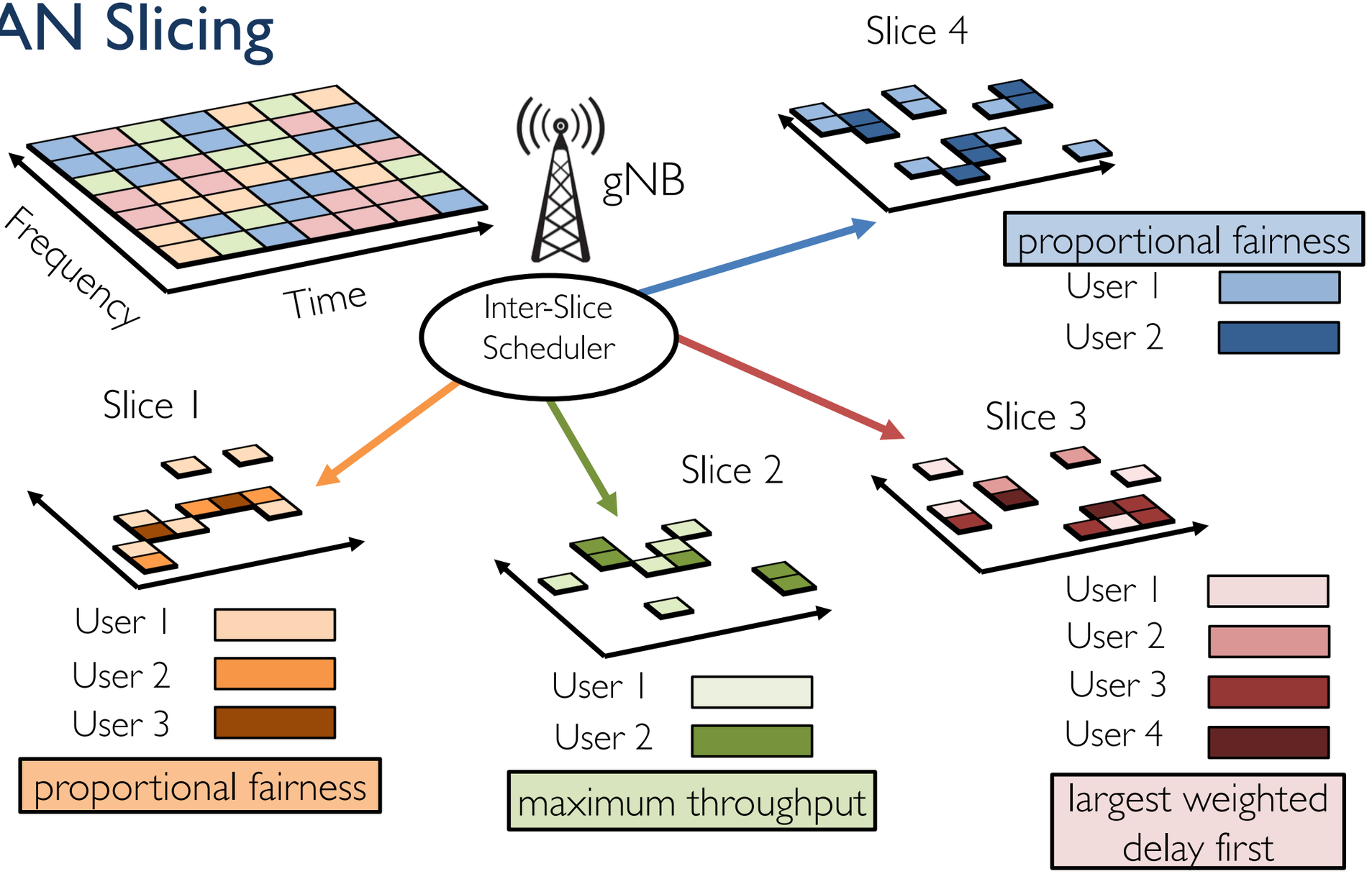
5G RAN Slicing

Downlink radio resources are divided in both frequency and time domain



RB(resource block) is the smallest unit that can be allocated to a user

5G RAN Slicing



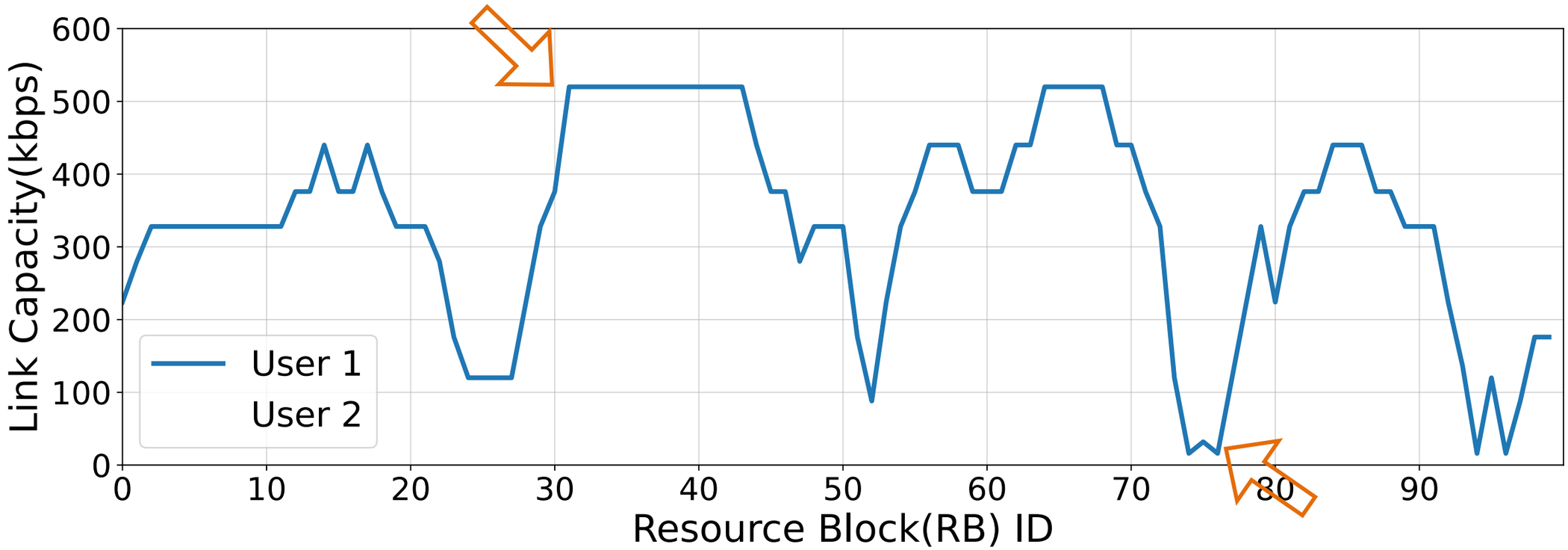
5G RAN Slicing

Objectives for RAN Slicing:

- Weighted fairness across slices
- Support customizable and diverse enterprise schedulers
- High spectrum efficiency (i.e. high throughput)

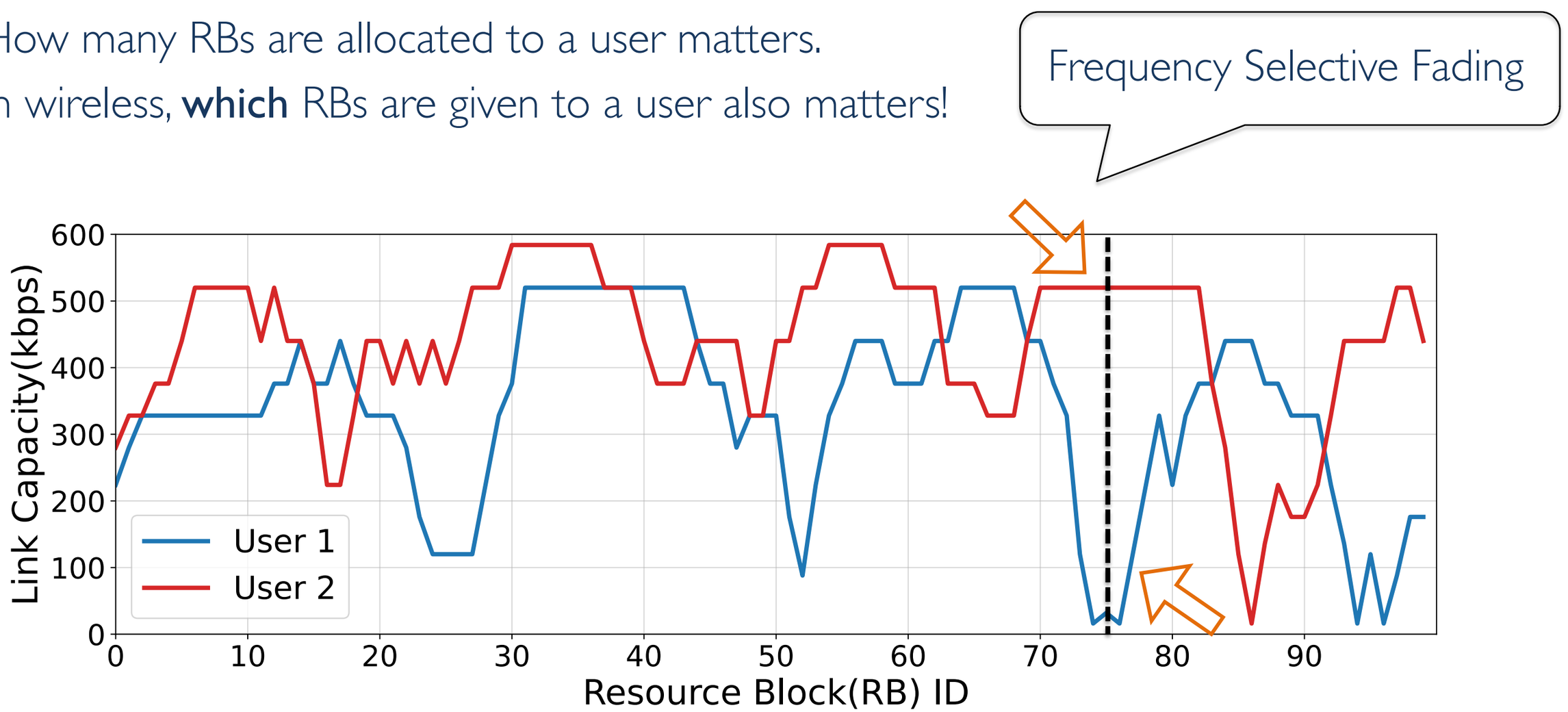
Need for Channel-Awareness

- How many RBs are allocated to a user matters.
- In wireless, **which** RBs are given to a user also matters!



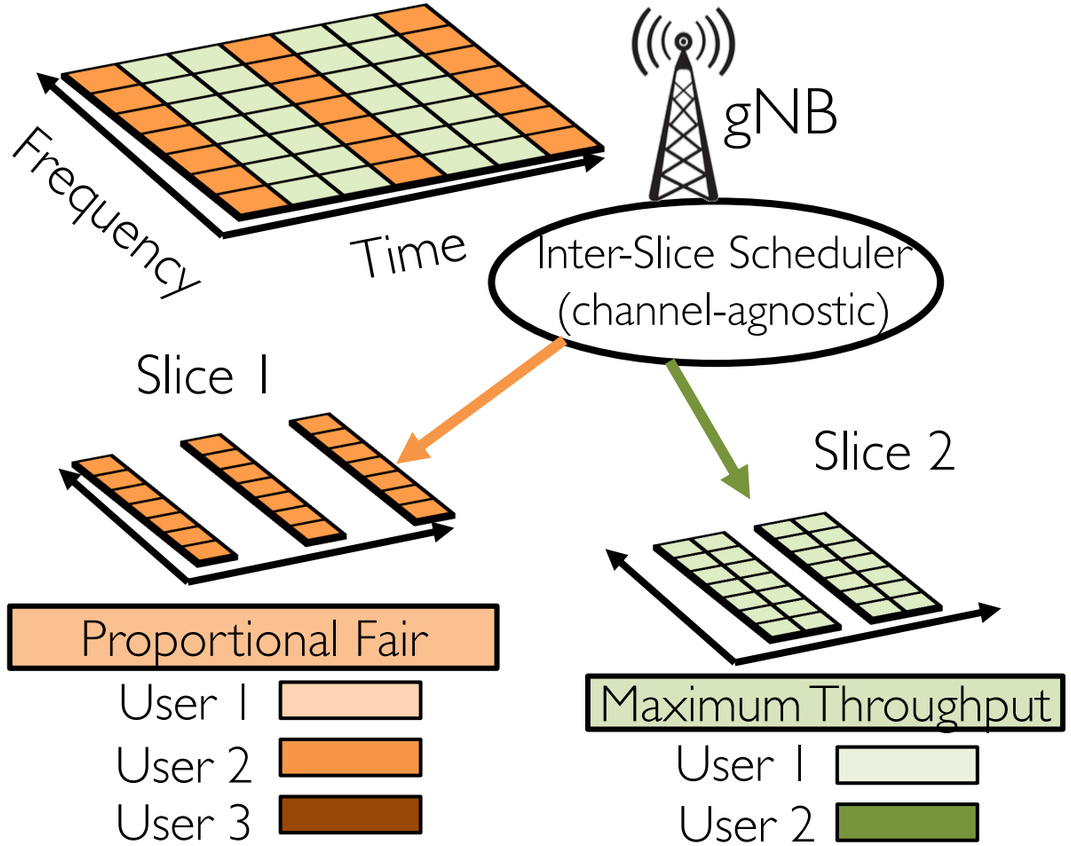
Need for Channel-Awareness

- How many RBs are allocated to a user matters.
- In wireless, **which** RBs are given to a user also matters!



Existing Techniques for RAN Virtualization

- Inter-slice scheduler divides RBs across slices, agnostic of channel quality.
- Each slice can then allocate its RBs to its users in a channel-aware manner.

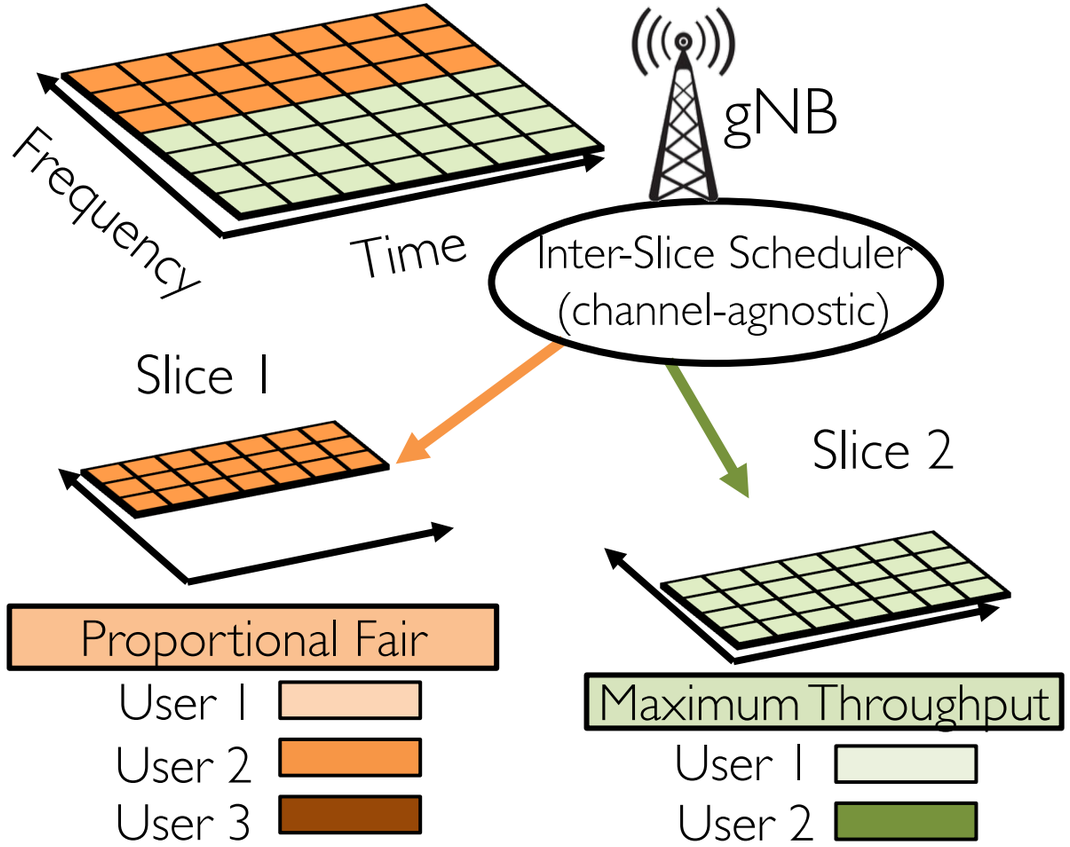


Example one:
NVS[1] allocates RBs to slices in time domain

[1]: NVS: a virtualization substrate for WiMAX networks, Mobicom10

Existing Techniques for RAN Virtualization

- Inter-slice scheduler divides RBs across slices, agnostic of channel quality.
- Each slice can then allocate its RBs to its users in a channel-aware manner.

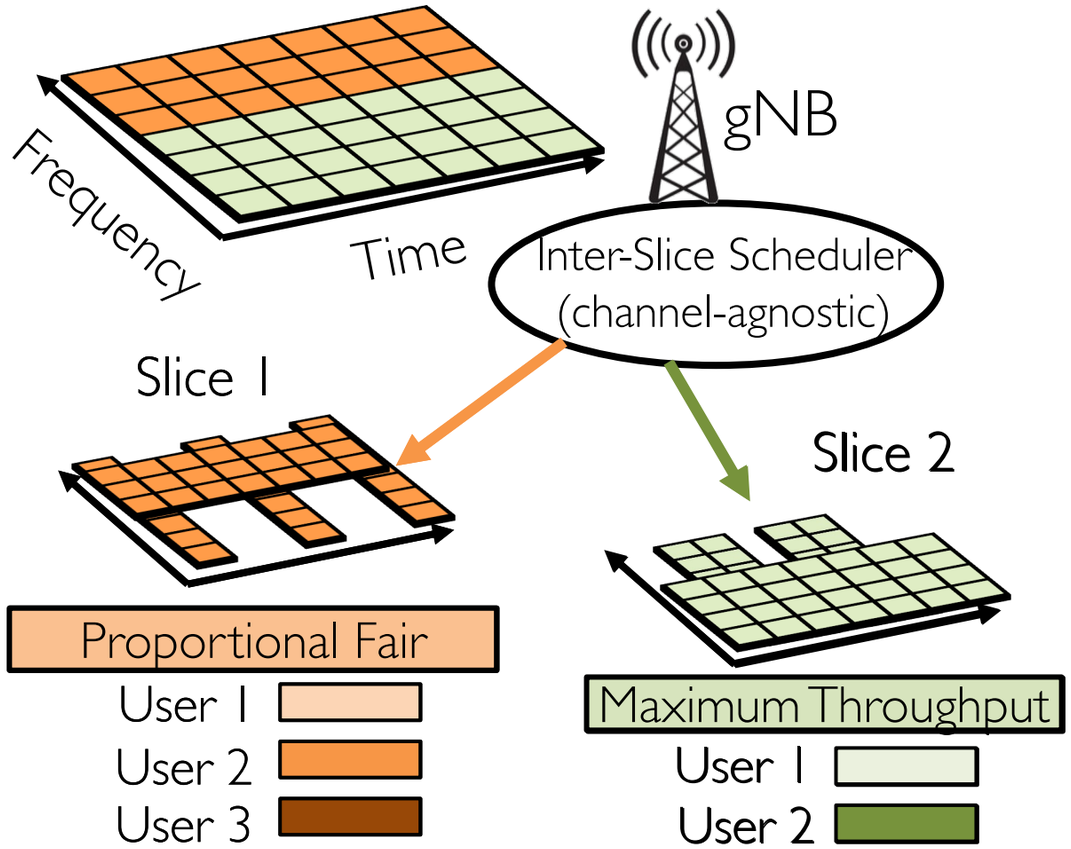


Example two:
Flare[2] statically assigns a dedicated chunk of spectrum to a slice

[2]: End-to-end Network Slicing for 5G Mobile Networks, Journal of Information Processing

Existing Techniques for RAN Virtualization

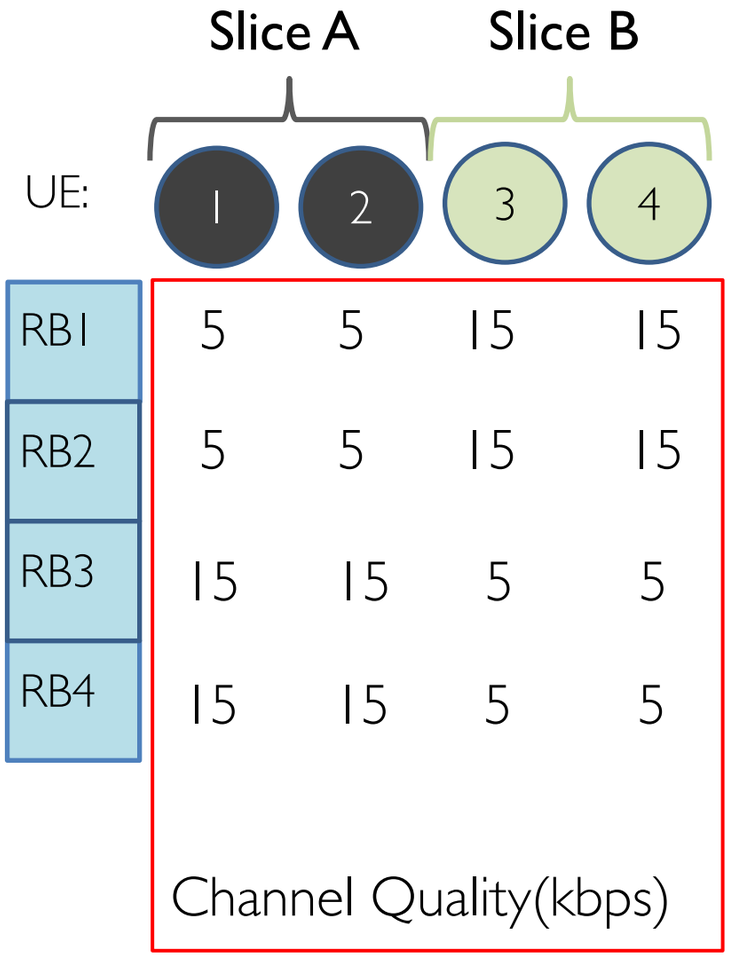
- Inter-slice scheduler divides RBs across slices, agnostic of channel quality.
- Each slice can then allocate its RBs to its users in a channel-aware manner.



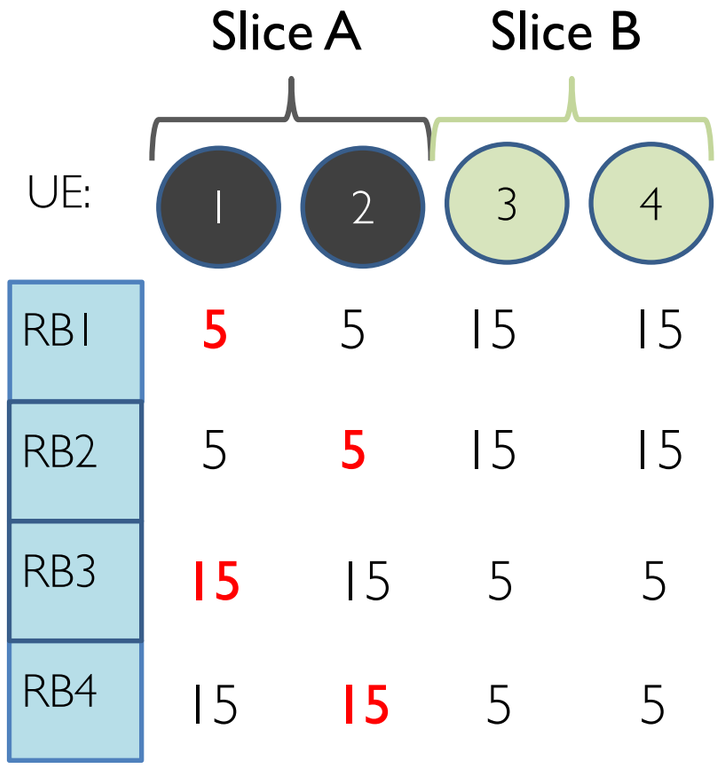
Pros: Cleanly decouple inter-slice scheduling and enterprise scheduling

Cons: Inter-slice scheduler is agnostic of channel quality

Need for Channel-Aware Inter-Slice Scheduler



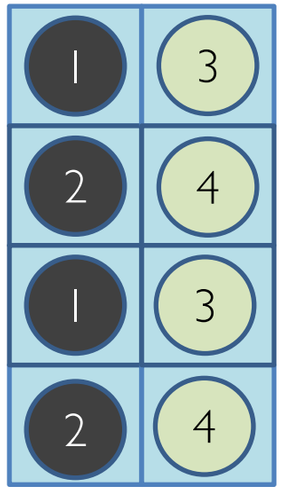
Need for Channel-Aware Inter-Slice Scheduler



Channel Quality(kbps)

Channel-Agnostic

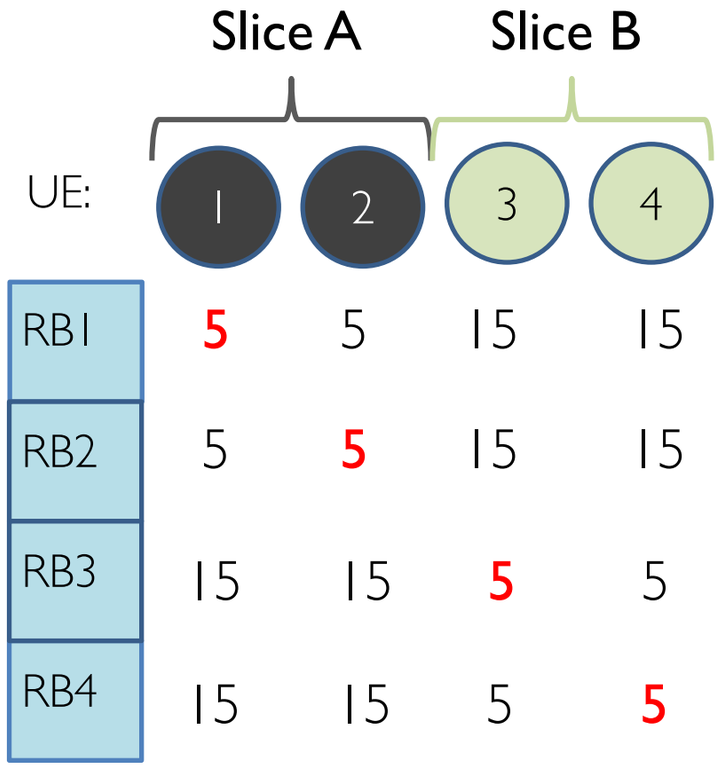
Round Robin(NVS)



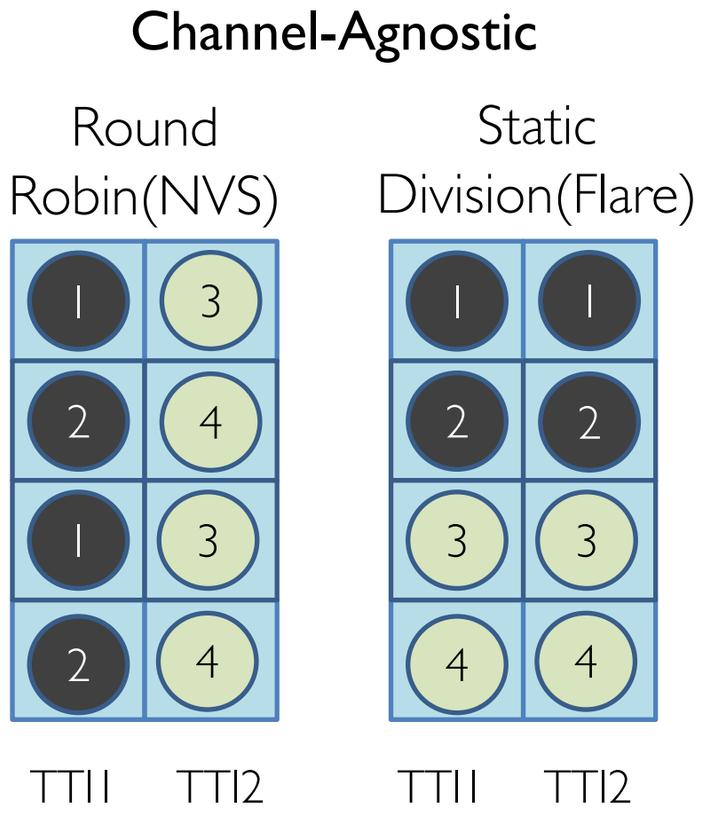
TTI1 TTI2

40 kbps

Need for Channel-Aware Inter-Slice Scheduler



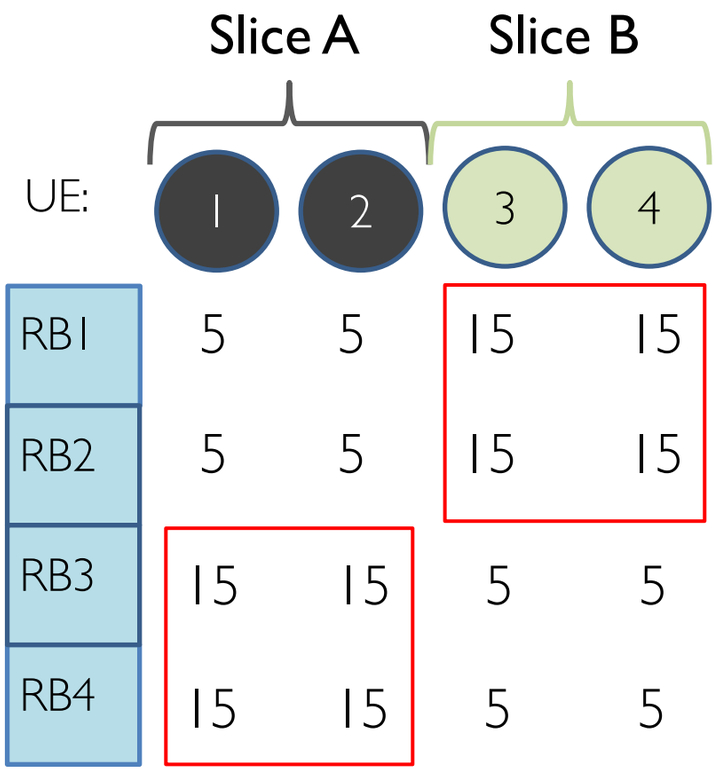
Channel Quality(kbps)



40 kbps

20 kbps

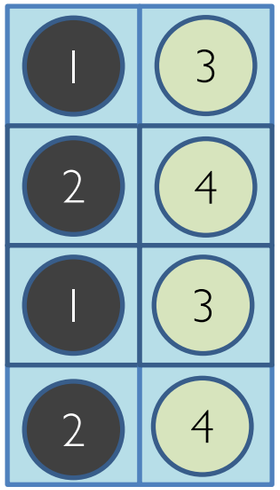
Need for Channel-Aware Inter-Slice Scheduler



Channel Quality(kbps)

Channel-Agnostic

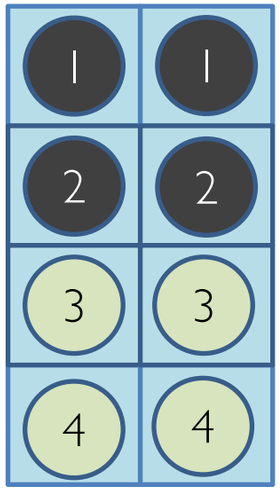
Round Robin(NVS)



TTI1 TTI2

40 kbps

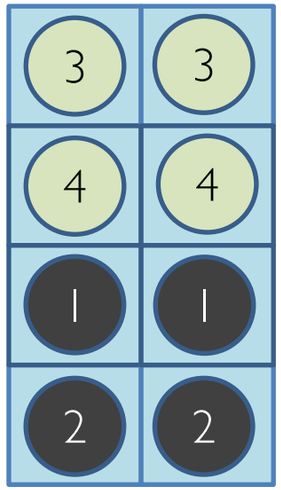
Static Division(Flare)



TTI1 TTI2

20 kbps

Channel-Aware



TTI1 TTI2

60 kbps

Need for Channel-Awareness

We need channel awareness at both levels

- Channel-aware enterprise scheduler to ensure each slice can squeeze out high spectrum efficiency from its allocated RBs

- Channel-aware inter-slice scheduler to ensure each slice gets a "**good**" set of RBs to begin with.

Need for Channel-Awareness

We need channel awareness at both levels

- Channel-aware enterprise scheduler to ensure each slice can squeeze out high spectrum efficiency from its allocated RBs
  - Allowed by existing RAN virtualization techniques

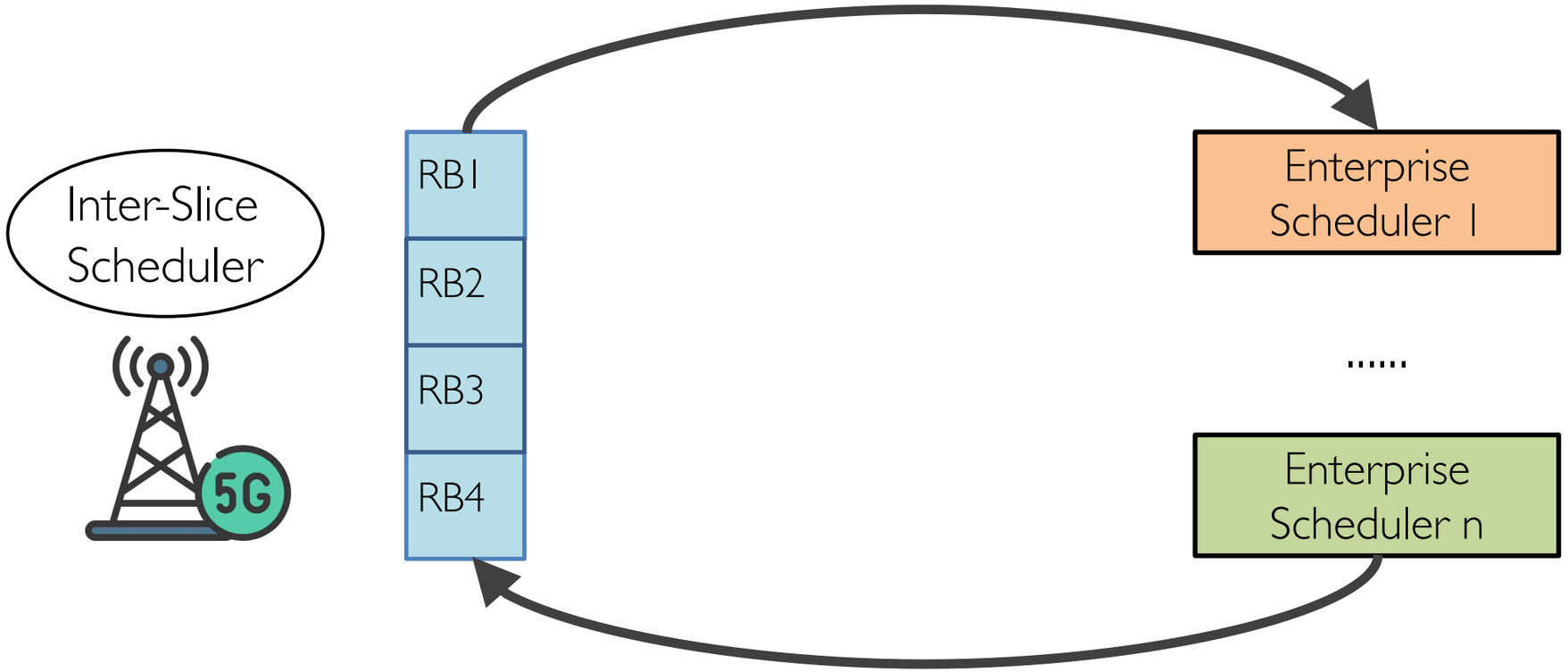
- Channel-aware inter-slice scheduler to ensure each slice gets a "**good**" set of RBs to begin with.
  - Not supported by existing RAN virtualization techniques.

Enabling channel-awareness at both levels is difficult

The Cyclic Dependency

The inter-slice scheduler needs to know which UE will be scheduled on each RB

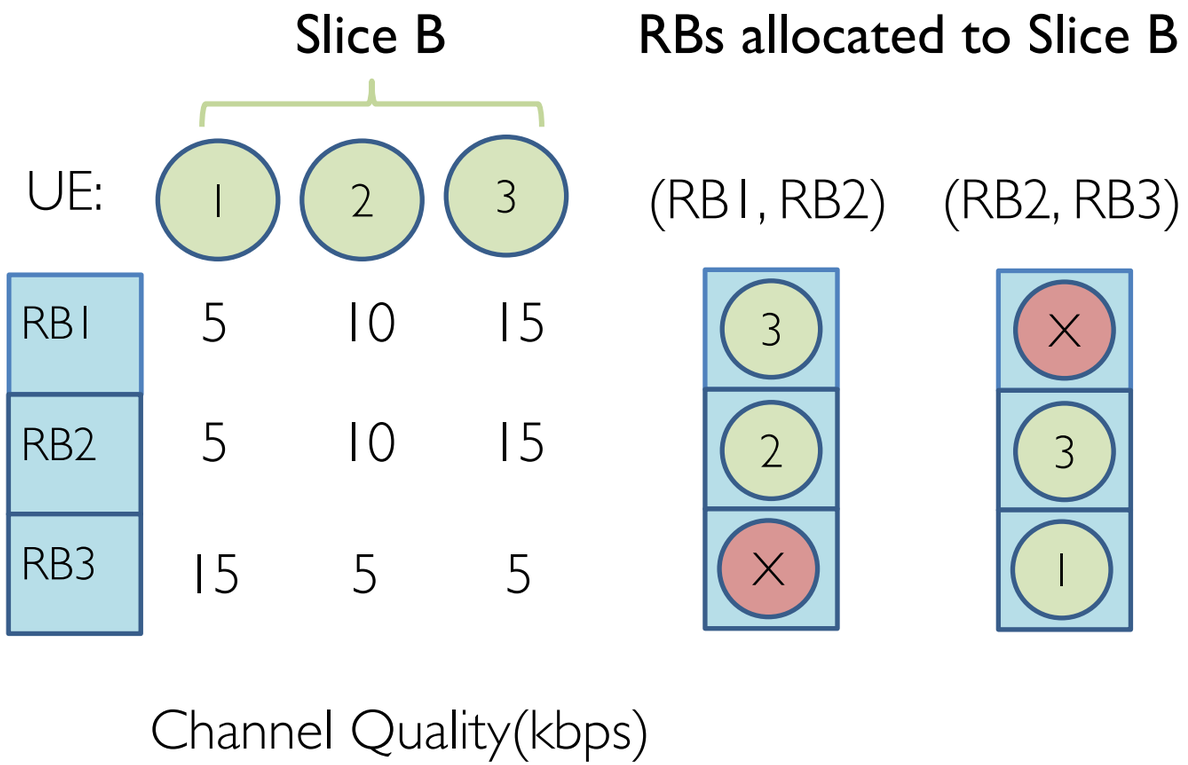
- Determined by the ***enterprise scheduler***



The enterprise scheduler needs to know which RBs are allocated to it

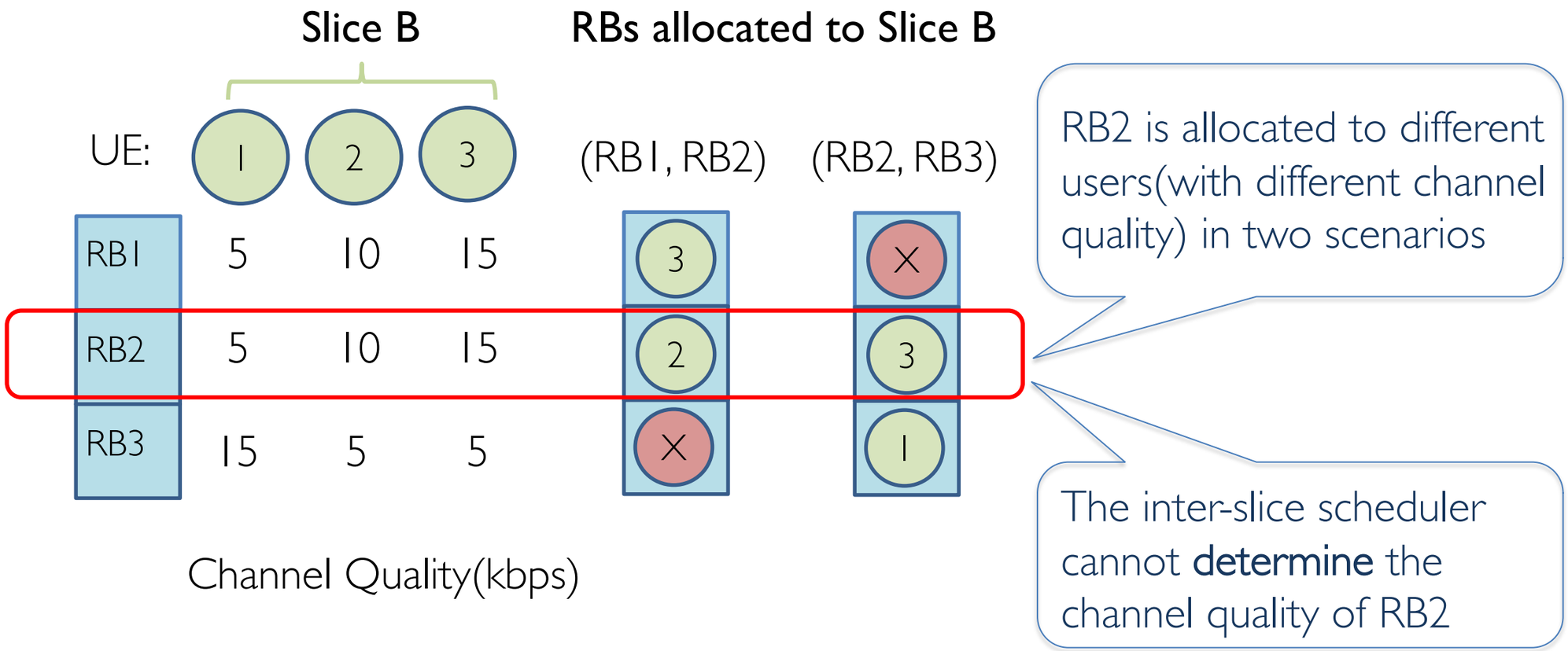
- This is determined by the ***inter-slice scheduler***

Challenge: the Cyclic Dependency



- The slice is to be allocated two RBs.
- The enterprise scheduler assigns RBs to the users with highest channel quality, under the constraint that each user gets at most one RB.

Challenge: the Cyclic Dependency



- The slice is to be allocated two RBs.
- The enterprise scheduler assigns RBs to the users with highest channel quality, under the constraint that each user gets at most one RB.

Our Insights

- The inter-slice scheduler issues a query to each enterprise scheduler:
  - "If I give resource ***R*** to slice ***S***, which UE in slice ***S*** will get resource ***R***?"

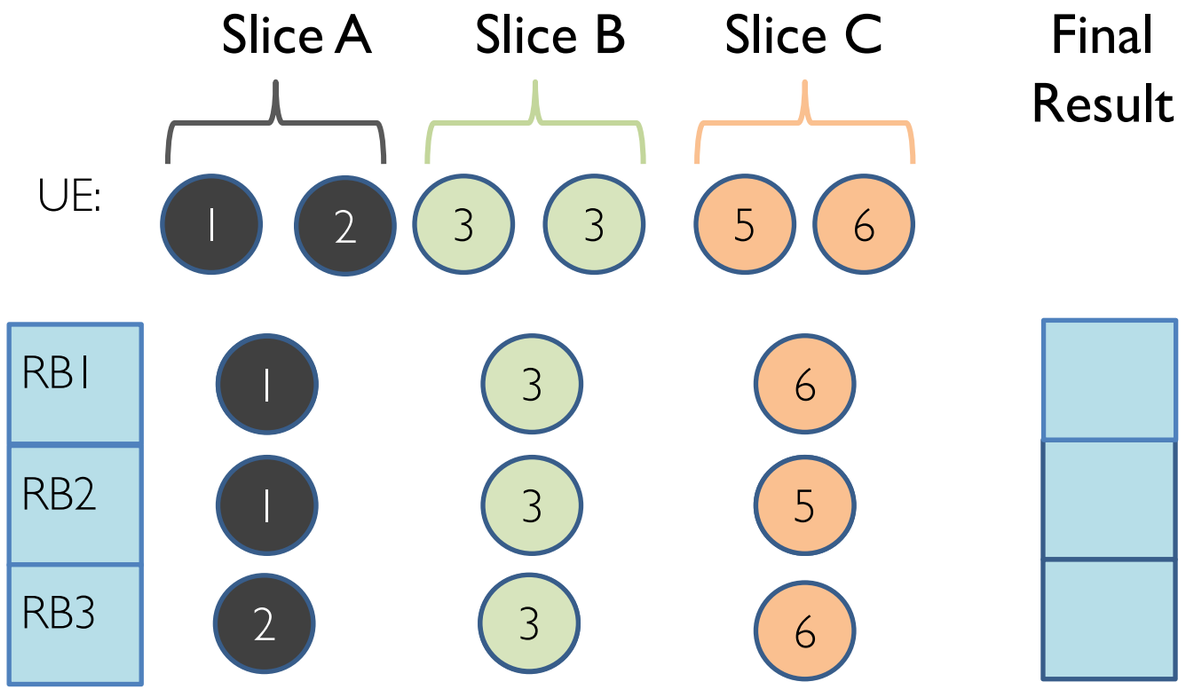
- Being able to answer the query restricts the enterprise scheduler:
  - It must allocate RBs to its users independent of the future given RBs
  - It must be greedy
  - *Many cellular scheduling policies are already greedy*

- The inter-slice scheduler must also be greedy
  - This allows the enterprise scheduler to account for **historical allocations** and **user demands**

RadioSaber

- Channel-aware Inter-Slice Scheduler
- Customizable Enterprise Scheduler

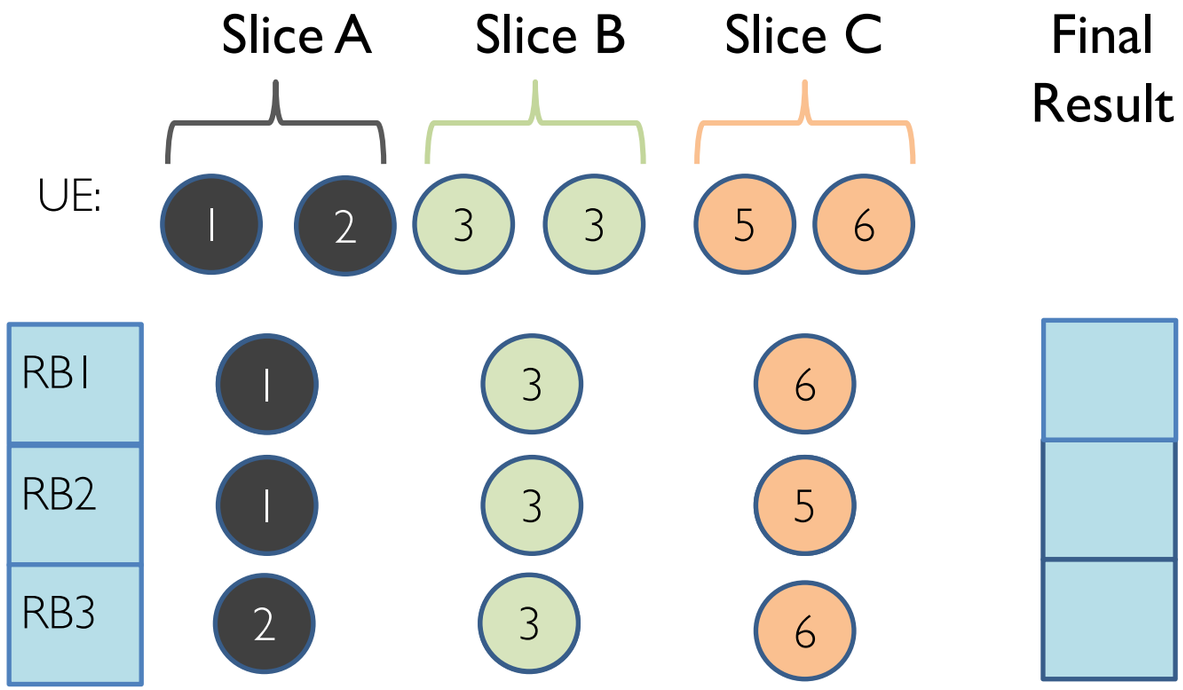
RadioSaber: Channel-aware Inter-Slice Scheduler



- Computing quota on RBs per TTI for each slice based on their weights

Other details: picking the order in which resource blocks get assigned

RadioSaber: Channel-aware Inter-Slice Scheduler



- Pick a RB
- Query every enterprise scheduler
- Assign it to the slice which achieves **maximum channel quality** for it (and hasn't satisfied its quota)

Other details: picking the order in which resource blocks get assigned

RadioSaber

- Inter-slice scheduling design
- Customizable enterprise scheduler

RadioSaber: Customizable Enterprise Scheduler

- Factors to consider:
  - channel quality, fairness, flow priority, queuing delay

- Paradigm one: Pick a user and schedule the highest priority flow for that user
  - Parameters to tune relative weightage of channel quality and historical allocation when selecting a user
    - *Inspired from generalized proportional fairness scheduler*

- Paradigm two: First pick the highest priority level, and then pick the specific flow in that priority level
  - Parameters to tune relative weightage of channel quality, historical allocation, and head packet delay when selecting a flow
    - *Inspired from M-LWDF scheduler*

Implementation of Core Workflow

- 5G Core Support:
  - Extend Open5GS to add support for RadioSaber control workflow
    - Allow slices to configure the scheduling parameters in the core
    - Propagate them from the core to the base station for scheduling

Implementation of Scheduling Algorithm

- Large-scale RAN simulation:
  - Extend the open-source RAN simulator to 5G
  - Trace-driven simulation using traces from LTEScope
  - Up to 800 users

Evaluation

- RadioSaber can correctly enforce isolation and weighted fairness across slices

- RadioSaber achieves higher overall throughput than the state-of-the-art inter-slice scheduler

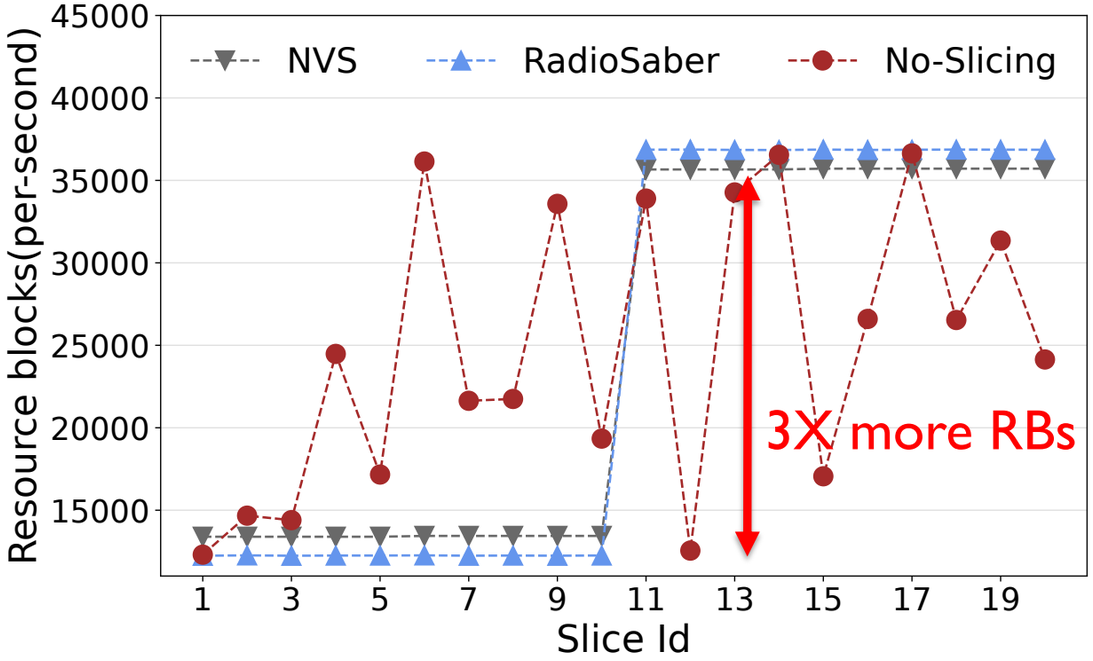
Evaluation

20 slices, each with 5-15 users

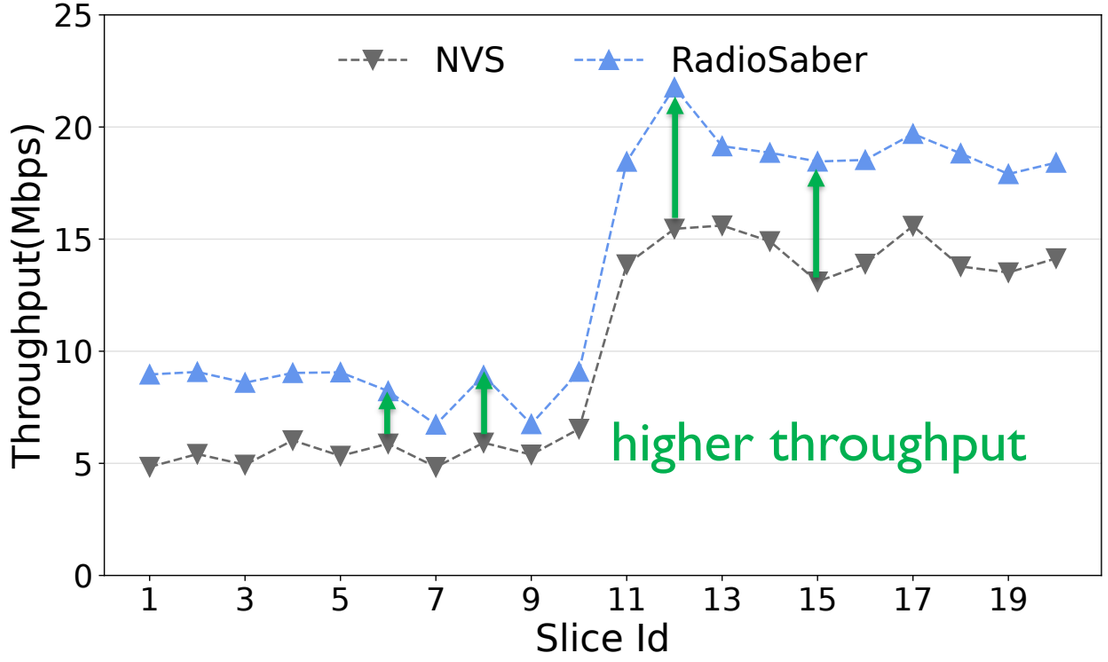
- First 10 slices: weight = 1 (*max throughput* enterprise scheduling)
- Last 10 slices: weight = 3 (*proportional fair* enterprise scheduling)

Baseline:

- NVS(state-of-the-art inter-slice scheduler)
- No Slicing(*proportional fair* scheduler across individual users)



(a) per-slice RBs per second



(b) per-slice throughput

Other Evaluations

Support diverse enterprise scheduling policies

Benefits hold as we change the number of slices or users per slice

Low runtime overhead and scheduling latency

And more …

Summary

- We identify a fundamental challenge for channel-aware RAN slicing:
  - **cyclic dependency** between the inter-slice scheduler and the enterprise scheduler.
- We break the cyclic dependency by restricting both schedulers to be **greedy**.
- We build RadioSaber, the first RAN slicing system that is **channel-aware** both at the inter-slice scheduler and enterprise scheduler.
- RadioSaber provides **parameterized interface** to customize enterprise scheduling policies.
- Achieves **17%-72% higher link capacity** than state-of the-art (channel-agnostic slicing).

Website: https://radiosaber.web.illinois.edu

https://github.com/elvinlife/RadioSaber

Thank you!