Load Balancing for a User-Level Virtualized 5G Cloud-RAN

Nishant Budhdev*
National University of Singapore
nishant@comp.nus.edu.sg

Arka Maity*†
Starfive Singapore
arka.maity@starfivetech.com

Mun Choon Chan
National University of Singapore
chanmc@comp.nus.edu.sg

Tulika Mitra
National University of Singapore
tulika@comp.nus.edu.sg

ABSTRACT

5G cellular networks support a wide variety of applications with different Service Level Objectives (SLOs) over a shared infrastructure using virtualization. Virtualization enables network operators to allocate a tailored set of computational resources in the cloud to users from different applications based on their SLOs. Existing virtualization approaches use slices to create logically independent networks for each different application. However, these approaches fail to provide adequate performance isolation among different slices, leading to performance degradation.

In this paper, we present the design and implementation of a load balancer called Dynamic Greedy Spike (DGS), for a cloud Radio Access Network (RAN) architecture with user-level virtualization. With user-level virtualization, network operators can now allocate new users to any host in the cloud irrespective of their source base station. DGS allocates these new users to different hosts to reduce interference between users and improve isolation by modeling the problem similar to weighted improper graph coloring. We implement a prototype of the user-level virtualized RAN architecture called uvRAN using OpenAirInterface. We also perform large-scale evaluations and show that uvRAN along with DGS provides significant improvement in isolation, which improves performance while reducing the compute resources required for baseband processing.

CCS CONCEPTS

• **Networks** → *Mobile networks*; *Cloud computing*; *Network resources allocation.*

KEYWORDS

5G, cloud radio access networks, virtualization, load balancing

ACM Reference Format:
Nishant Budhdev, Arka Maity, Mun Choon Chan, and Tulika Mitra. 2022. Load Balancing for a User-Level Virtualized 5G Cloud-RAN. In *17th ACM Workshop on Mobility in the Evolving Internet Architecture (MobiArch'22), October 21, 2022, Sydney, NSW, Australia.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3556548.3559627

*Co-first authors
†Work completed while at National University of Singapore

MobiArch'22, October 21, 2022, Sydney, NSW, Australia

ACM ISBN 978-1-4503-9518-2/22/10.
https://doi.org/10.1145/3556548.3559627

1 INTRODUCTION

With the introduction of the Fifth Generation (5G) of cellular networks, operators are expected to support applications with a wide variety of SLOs. These applications can be broadly classified into three main categories: (1) enhanced mobile broadband (eMBB) applications such as Augmented Reality/Virtual Reality (AR/VR) and video streaming, requiring high throughput, (2) ultra-reliable and low latency communication (uRLLC) applications such as remote surgery, Vehicle-to-everything (V2X), and (3) massive machine type communication (mMTC) applications such as smart cities that require low power communication with a large number of IoT devices.

In order to support the wide variety of SLOs from different applications on a common physical infrastructure, network operators have introduced virtualization. The earliest works on virtualization introduced Cloud Radio Access Networks (CRAN) [1] that virtualized computational resources in the cellular network by splitting RAN functionality into two components: the frontend component called Radio Unit (RU) and the backend components consisting of the Distributed Unit (DU) and Central Unit (CU). The RU is responsible for radio transceive functions for converting radio signals to/from digital samples. On the other hand, the DU and CU are responsible for baseband processing functions such as de/coding, de/modulation, antenna combiner etc. Network operators could now allocate shared networking and computing resources optimally among multiple DUs and CUs, to ensure all applications meet their SLOs while reducing server sprawl caused by unused processing capacity. Further works on virtualization propose using network slicing [5, 19] to enable dynamic sharing of resources among different users/applications. These works divide the network into independent logical partitions called slices that are designed to support tailored services for distinct application scenarios. Although network slicing can improve flexibility by dynamic resource sharing, existing slicing-based approaches do not provide sufficient inter/intra-slice isolation amongst the different services to provide guaranteed performance [2]. Previous works [11] shown that the improper isolation between slices/users can disproportionately impact uRLLC applications with stringent SLOs.

We address the aforementioned challenge in this work, and present uvRAN which introduces user-level data plane virtualization in the RAN to reduce inter/intra slice interference and improve isolation. In uvRAN, the baseband processing corresponding to each new user session can be executed on any host in the cloud, independent of the user's primary base station. Resource allocation for uvRAN is challenging as the operators need to dynamically allocate each new user to ensure its SLOs.

Traditionally, all users with the same primary base station are processed on the same DU/CU unit that is provisioned with enough computational resources to achieve the SLOs for the worst-case operational workload. However, calculating the worst-case operational workload is non-trivial for a user-level virtualized RAN architecture as users can be allocated to any host in the cloud. At the same time, inefficient allocation of users can cause them to miss their baseband processing deadline. This is further exacerbated in uvRAN as a large number of users can be mapped to a single host leading to resource contentions and causing users to miss their deadlines. These misses are called *computational failures* [9] and are expensive in cellular networks as they cause unnecessary network retransmissions that decrease spectral efficiency and affect other users' performance. We propose the Dynamic Greedy Spike (DGS) load-balancer for uvRAN that allocates new users to hosts while ensuring minimal interference between users. DGS uses weighted graphs where each user is a vertex and the edge weight denotes the interference between a pair of user flows. We design DGS to reduce the graph coloring cost where each color represents a unique physical host in the cloud. Our contributions are as follows:

- We propose the DGS load balancer, an online greedy heuristic for new user placement on the hosts in the cloud. DGS uses improper graph coloring to model the interference among different users.
- We demonstrate the feasibility and effectiveness of user-level slicing in the data plane by modifying OpenAirInterface (OAI) [20], an open-source implementation of the Long Term Evolution (LTE) stack, to support user-level virtualization.

We have performed a large-scale simulation to evaluate our resource allocation approach using real traces captured from network operators that have over 1.2 million user sessions. Our experiments show that compared to the typical CRAN architecture [1], uvRAN with DGS can reduce the drop rate by order of magnitude while requiring 2x fewer resources.

2 BACKGROUND AND RELATED WORK

Traditionally, the cellular network consisted of discrete base stations, each responsible for processing data for all the users connected to the base station. The introduction of virtualization has enabled a disaggregated architecture called the Cloud RAN that consists of two components. The frontend consists of the RU that is responsible for radio functions such cyclic prefix addition/removal, FFT/IFFT etc. The backend consists of DU and CU in the cloud and is responsible for baseband processing functions. In principle, the functions can be distributed in different ways [22] between the RU and DU/CU. The distribution of the functions across the RU and DU/CU is defined by the functional split used by the network operator. We focus mainly on the lower layer functional split in this paper, also called the ORAN 7-2x split, as it is the key for enabling user-level virtualization in CRAN. In 7-2x split, the RU is responsible for low-level radio and L1 functions, and the rest of the functions are processed in the cloud. Each DU/CU performs all the required processing for the corresponding base station and multiple DUs/CUs can be mapped to a single host, to share computing resources among multiple base stations in the cloud.

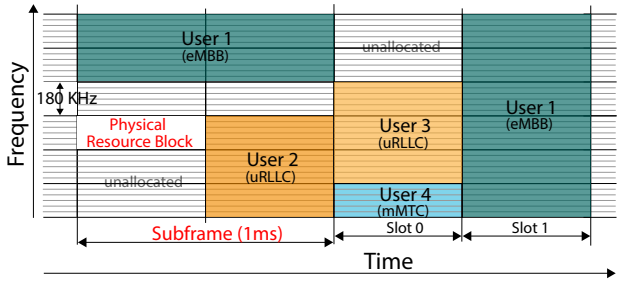


Figure 1: Time-frequency structure for LTE cellular networks. A Physical Resource Block (PRB) is the smallest schedulable resource that can be allocated to a user for transmission on the wireless spectrum.

Cellular network functions can be further separated into control plane and data plane functions using Software Defined Networking (SDN). Multiple recent works [4, 16, 18] use SDN to separate control and data plane functions in the RAN, to enable dynamic control policies. While SDN enables these approaches to dynamically control functions such as network scheduling, packet routing, and authentication, a common and key limitation of the aforementioned works is their inability to provide differentiation in data plane processing for users with diverse Service Level Objectives (SLOs). To resolve this, multiple works [8, 24] introduce data plane programmability by creating multiple slices in the data plane, by decomposing the data plane into a chain of functions. Each function in the chain can be customized for a slice to support different slices running on top of a shared set of hosts in the cloud.

Another limitation of existing works, which serves as one of the motivations for our work, is the lack of performance isolation among the slices processed in the cloud to guarantee SLOs for a diverse set of applications running on the shared infrastructure. POSENS [17] proposes a new RAN architecture to improve performance isolation by distributing data plane processing, above the Medium Access Control (MAC) layer, on different hosts. While this is a step in the right direction, POSENS still requires common Physical Layer (PHY) processing for all users associated with the same base station, thus limiting the effectiveness of this approach. Thus, we propose a new RAN architecture to further improve performance isolation, by virtualizing a user's data plane processing to enable operators to distribute users to any host in the cloud.

With virtualization, network operators also need to allocate the virtualized computational resources in the cloud. Multiple works [13, 26] introduce load balancers to allocate computational resources across multiple slices. These algorithms distribute new DUs/CUs to different hosts in the cloud while improving throughput or reducing power consumption. However, these algorithms are not suitable for load balancing in uvRAN, as we require a fine grained allocation of computational resources to users in the network. The load balancer also needs to account for the dynamic nature of communication in cellular networks as new users with different SLOs can be scheduled each millisecond (subframe duration) in LTE, as shown in Fig. 1. Additionally, most user sessions are only 1 subframe long and require less than 4 Physical Resource Blocks (PRBs) [25], the smallest allocatable resource for a user. Thus, we need a new load balancing

algorithm for uvRAN that can dynamically allocate new users in the network with low-overhead while ensuring performance isolation.

3 SYSTEM OVERVIEW

uvRAN is built to improve scaling and performance isolation for 5G cellular networks which are critical for supporting a wide variety of SLOs such as high throughput, low latency, high reliability, low-power communication etc. uvRAN consists of three major components as shown in Fig. 2:

- **User level virtualization** to enable the cellular network to scale efficiently as users' baseband processing can be spread across different hosts in the cloud irrespective of their primary base station. Network operators can increase the number of hosts to increase the available computational resources during peak traffic to ensure SLOs for all applications.
- **Load Balancer** allocates new users to hosts in the cloud. The load balancer acts as the global controller that collects and aggregates network-wide metrics, such as CPU utilization, network bandwidth, etc., from all hosts to determine the optimal host for new users. The load balancer is triggered when a new user session is created. Details of the Load Balancer are given in Section 4.
- **Task Allocator** runs as a local controller on each host to schedule the baseband processing tasks for each User Equipment (UE) to different cores on the processor. The task allocator ensures that each user's deadline is satisfied. The Load Balancer and the Task Allocator operate independently, and hence the proposed Load Balancer can be used with any existing Task Allocator. The details of the Task Allocator are beyond the scope of this paper.

Design & Implementation of uvRAN. The objective of user-level virtualization is to distribute the monolithic baseband processing of a base station to multiple users that can then be allocated to different hosts in the cloud RAN. uvRAN achieves this by separating the data plane RAN functions into two parts: user-level and base station level functions. Base station functions include baseband processing of data transmitted and received on common channels responsible for critical requests such as random access, network scheduling, ACK/NACK, broadcast, etc. These data are transmitted/received periodically irrespective of the network load, i.e., the workload corresponding to these channels is mostly constant. These critical functions are executed by the base station thread and hence, we use one-to-one static mapping between these threads and each RU to ensure consistent performance. On the other hand, the processing load for user-specific threads varies with user traffic in each subframe [10]. The key challenge in the design and implementation of user-level virtualization is the separation of the base station and user-specific data functions while meeting real-time constraints. This is solved by having multiple daemon threads for user-level functions that are shared across all the users allocated to the host. On the other hand, each RU has a dedicated daemon thread for its base station level functions to provide maximum performance isolation.

User-level virtualization is achievable due to two fundamental principles of data plane processing. First, the processing of one user is independent of other users in the subframe/slot in 4G/5G.

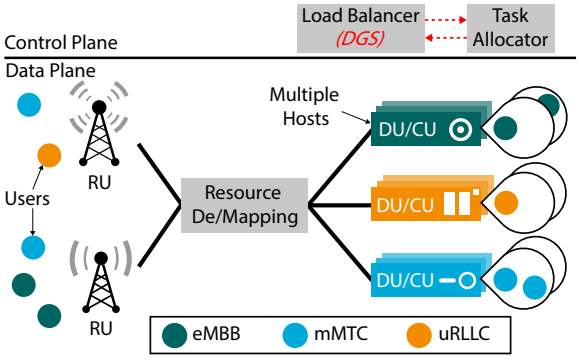


Figure 2: The 3 major components of the uvRAN architecture: (1) user-level virtualization denoted by the distribution of users from the same RU on different hosts, (2) DGS Load Balancer in the control plane for allocating new users to different hosts, and (3) local Task Allocator on each host for managing tasks from different users allocated to a host.

Intuitively, this is true for any communication system; however, different users can transmit/receive data at the same time on different frequencies. Combining/separating user data transmitted/received on different frequencies, LTE needs to perform operations such as IFFT/FFT, de/serialization, prefix removal/insertion, etc., on all user data in the subframe. Since uvRAN follows ORAN 7-2 [22] functional split which separates common cell processing and user processing, all the aforementioned common functions are processed on the RU. Split 7-2 provides the ideal trade-off between fronthaul network bandwidth and savings due to centralized processing by transmitting the clean analog signals while still keeping the majority of the baseband processing in the cloud [14]. Additionally, network operators can use works such as Fronthaul Slicing Architecture [12] that support slicing in the fronthaul networks for sending/receiving individual user data to/from respective hosts.

The second principle is the minimal information required for a user's data plane processing. Data plane processing for a user requires information such as bandwidth allocated, modulation scheme, and the number of layers, which are determined by the network scheduler usually a few milliseconds in advance. Conversely, the output of each user's processing for a subframe is required within 2.5 milliseconds in LTE as the base station needs to send an ACK/NACK back to the user [3]. While all these dependencies come with tight timing constraints, the amount of control data transmitted/received between the user-level and base station level threads is insignificant. Thus there is minimal network overhead in uvRAN with user-level virtualization as Round Trip Time (RTT) between hosts in the cloud is in the order of tens of microseconds [15]. Section 5 provides more details about the prototype implementation of uvRAN.

4 LOAD BALANCER

The load-balancer algorithm allocates new users to hosts when they arrive in the network. These allocation decisions are irrevocable, as migration between hosts can cause significant service disruption for the user. First, we take a closer look at the different causes of

computational failures that cause the users' data within a subframe to be dropped. The deadline for baseband processing exists as the network needs to transmit an ACK/NACK to every active user in each subframe [3].

4.1 Understanding subframe drops

To further understand the cause of these drops we use an oracle B:H load-balancer, wherein H hosts are optimally allocated to B base stations. The allocations are done after observing the entire workload from all the base stations a-priori, to ensure a near-optimal allocation decision. Such an oracle load balancer cannot be implemented in practice, but it can highlight the distribution of subframe drops in real-world traces. We categorize the dropped subframes in the above setup into three main categories:

- An **interfering drop** is caused when a subframe with multiple large user transmissions is not processed before its deadline. These subframes are disproportionately vulnerable to queuing delays due to the large amount of computational resources required to successfully process them.
- A **variable latency drop** occurs when the processing time for the subframe is randomly inflated due to the unpredictable timing behavior of the host platform.
- A **backlog drop** occurs when the user's data plane processing is increased due to the wait for computational resources. These drops can easily be avoided by either increasing the amount of available computation resources or by improving the efficiency of the load balancing algorithm.

We observe all these types of subframe drops even in an ideal scenario, where each base station is allocated to a unique host. However, as we reduce the number of computational resources by decreasing the number of total hosts, the share of interfering and backlog drops increase exponentially. This is caused by the increased resource competition among users from different base stations allocated to the same host. Additionally, users' data processing not only takes slightly longer to complete but users with large data transmissions within a subframe are also disproportionately dropped due to the long waiting time in the system. Overall, we observe up to a 100x increase in the number of interfering and backlog drops when the total number of hosts is halved as compared to the ideal scenario.

4.2 Dynamic Greedy Spike (DGS) Algorithm

The idea behind the DGS algorithm is to minimize the aggregate interference amongst different user flows. The aggregate user interference can be understood in terms of a Susceptible Interference Graph (SIG). The SIG is an undirected edge and vertex weighted graph, as shown in Fig. 3. The vertices of the graph represent active user flows, while the edges capture, the interference between a pair of user flows. The vertex-weight is a positive integer that counts the number of load spikes seen by a user flow up to and including the current subframe. The edge-weight denotes the number of subframes where the connecting vertices spike simultaneously. A user load spike occurs whenever the user's estimated execution time, when running in isolation, exceeds a given threshold. The execution time can be approximated using a linear model dependent on the number of PRBs and Modulation and Coding Scheme (MCS).

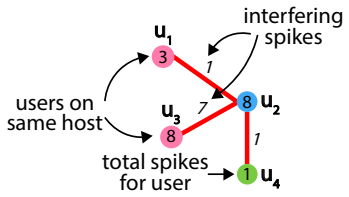


Figure 3: SIG showing the users as vertices, user spikes as the corresponding vertex weight, and interfering spikes as the edge weight. The vertex color denotes the specific host that the user is allocated to.

The SIG evolves in response to new user arrivals (vertex addition), load-spike transmission (vertex weight increment), interfering load-spike transmission (vertex weight increment and edge weight increment), and user timeouts (vertex and incident edge deletion). Furthermore, the assigned host for a given user flow can be visualized by its vertex color. The total coloring cost is defined as the sum of edge weights whenever the adjacent vertex colors are the same. This estimates the degree of interference suffered by the user flows in the same host. This problem is similar to weighted improper graph coloring [6], except that the color needs to be determined online, with incomplete information. We should emphasize that, while SIG topology is dependent only on the workload trace, the color cost also depends on the user-to-host assignment.

DGS attempts to minimize the *coloring cost change* associated with each allocation decision. We design a cost change estimator to exploit the following two key properties of uvRAN. First, load spikes from two different user flows can only interfere if they are assigned the same host while being connected to *different* base stations, as a single base station cannot schedule more than its total bandwidth. Note that each host has to be capable of handling the baseband processing of a single base station within the deadline. Second, the interfering load spikes between any two pair of user flows is upper-bounded by individual load spikes experienced by any one of them. For every new user arrival, DGS computes the interfering aggregate load spike for each host by summing up the load spikes across all the base stations, except for the one that the new user belongs to. The user is allocated to the host minimum value of interfering aggregate load-spike.

5 EVALUATION

The evaluation section is organized into two parts. First, we verify user virtualization on a realistic setup using OpenAirInterface, an open-source LTE stack implementation. The data plane threads for baseband processing are daemon threads with each core containing exactly one thread. We evaluate the efficiency of the Host and Task Allocator by running a large-scale simulation using PHY benchmark [21] with real LTE traffic traces.

5.1 Prototype using OpenAirInterface (OAI)

We use a setup consisting of a USRP B210 for the RU and OpenAir-CN for the core network. We implement uvRAN on OpenAirInterface (OAI) and verify the system's functionality by establishing two-way communication, using ping packets, between the mobile

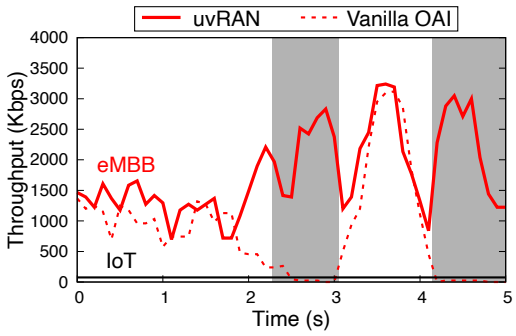


Figure 4: Comparison of throughput for eMBB user between Vanilla OAI and uvRAN. The shaded area shows intervals with reduced throughput for vanilla OAI.

user and public websites. We compare the memory and processing overhead in uvRAN by using OAI's in-built emulator and use a similar workload by forcing each user to send UDP packets at a constant bitrate of 400Kbps to a public server. uvRAN consumes only ~80 MB higher memory than Vanilla OAI because of the need to communicate between different daemon threads for satisfying dependencies in the LTE stack. This overhead remains constant, as it is independent of the number of users in the system. Instead, it depends only on the base station configuration parameters such as channel bandwidth, sampling frequency, bits per sample, etc. Additionally, average core utilization is relatively low for uvRAN, as we spread the processing across multiple cores. This is important as uvRAN scales with an increasing number of cores/hosts as compared to contemporary works that are limited to a single core/host. Thus Vanilla OAI experiences severe performance degradation when three users are active, often leading to user data being dropped due to its inability to complete processing before the deadline. For a single user, uvRAN's core utilization is only 1.5% higher than Vanilla OAI due to the overhead of thread communication. While the result is not surprising as uvRAN can utilize more resources (cores), it shows that the user-level virtualization in uvRAN is a viable solution with minimal overhead.

To evaluate the impact of users with different SLOs running on the host, we consider two users from different use cases: eMBB and IoT (mMTC communication). We simulate the eMBB user with a TCP connection to saturate the wireless bandwidth. The IoT user sends a 400-byte ping packet every 100 ms to simulate transmissions that are small and bursty in nature. The additional processing load imposed by the ping (IoT) traffic is negligible as compared to the TCP (eMBB) traffic. Fig. 4 shows the throughput of an eMBB user for uvRAN and Vanilla OAI alongside an IoT user. From Fig. 4, we can observe that the throughput of the eMBB user in Vanilla OAI is significantly affected by the presence of background traffic in the form of an IoT user. Interestingly, such a small increase in CPU utilization can cause severe interference. On the other hand, by assigning these users to different cores in uvRAN, the performance of eMBB users remains unaffected enabling it to transfer 2x more data. To summarize, the ability of uvRAN to distribute RAN processing across multiple hosts leads to significant performance gains, as compared to the traditional Vanilla OAI.

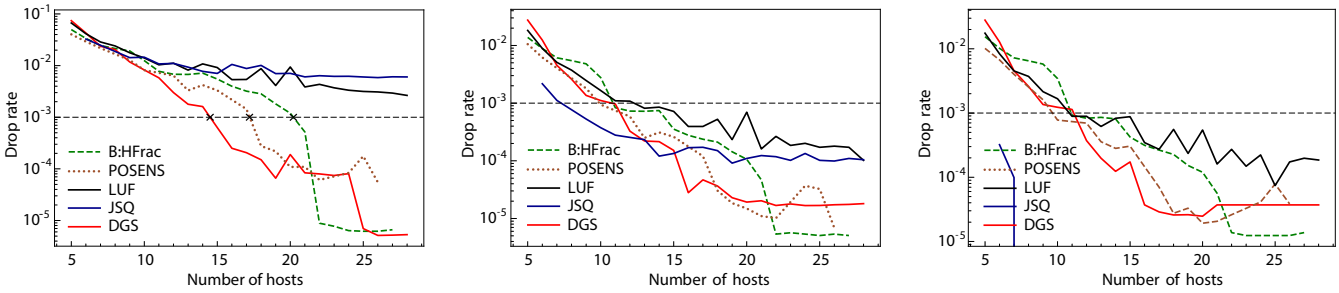
5.2 Large Scale Evaluation

We perform large-scale simulations using real traces [11] captured from network operators to evaluate the impact of user-level virtualization combined with DGS load balancer. Additionally, to simulate multiple use cases, we classify users into three different user types based on their traffic patterns using parameters such as Transport Block Size [11]. The trace captures 1.2 million user sessions, of which nearly 91% were classified as eMBB users. Additionally, the trace consists of a small number of heavy-hitter eMBB users (0.06%) whereas a large number of the users (~60%) are short flow eMBB users, which is in line with previous works [25].

Each host in the cloud uses a Xeon Gold-6126, a dual-socket Skylake scalable platform each with 12 cores running at 2.6 GHz. To simulate the user-level processing on each host, we use the PHY benchmark [21]. For eMBB and mMTC, 99.9% of the subframes should be processed within a deadline of 2.5 ms as the ACK/NACK is sent to the user within 4 ms [3]. uRLLC users have a much more stringent requirement where 99.999% of the subframes have to be processed within 0.5 ms as the ACK/NACK is expected within 1 ms. If a user misses the processing deadline, the data transmitted in the subframe for the user is dropped.

Baseline Algorithms. We compare DGS against two offline algorithms and two online algorithms. The offline algorithms construct the SIG for a large segment of the workload trace (usually ~ 90%) and thereafter improperly color it with the number of hosts. In the static B is to H Frac (B:HFrac) algorithm, the nodes in the SIG represent base stations, whereas, in POSENS [17], the nodes represent a base-station slice pair. In the latter case, the SIG is larger as the allocation happens at a much finer granularity. The improper coloring problem is then formulated as an integer quadratic programming problem to find the optimal allocation. The trace segment is also slid across the workload trace to obtain different SIG and coloring assignments and the final result obtained is an average over all such assignments. We also consider two online algorithms: Least Users First (LUF) and Join Shortest Queue with Processing Sharing (JSQ). The LUF algorithm assigns new users to the host with the least number of active users. JSQ allocates incoming user to the host with the shortest instantaneous queue size [7].

Fig. 5 compares the subframe drop rate of DGS against both offline and online algorithms for users with different throughput and SLOs. DGS outperforms not only online algorithms like LUF and JSQ, but also offline algorithms like B:HFrac and POSENs. DGS performs especially well with heavy hitter eMBB users, as shown in Fig. 5(a), because it allocates possible pairs of interfering users to different hosts. Both static algorithms also perform significantly better than the dynamic algorithms as they can exploit information from the entire trace akin to an oracle. Moreover, POSENS outperforms B:HFrac as it allocates the base-station slice pairs that enables the network operator to perform load balancing at a finer granularity. On the other hand, JSQ underperforms as it relies on instantaneous host congestion for user placement. The instantaneous values can often cause the network operator to miss the presence of heavy hitter eMBB users that are responsible for 40% of the spectral utilization while accounting for less than 0.1% of the users. Similarly, LUF also underestimates the impact of heavy hitter eMBB users as it relies on only the number of users on a host



(a) Heavy-hitter eMBB users (b) Small eMBB users (c) mMTC users

Figure 5: Subframe drop rate for different load balancing algorithms as the number of available hosts increases. DGS outperforms others by improving performance isolation by reducing interference between users.

but does not take into account the spectral allocation of these users determining their baseband processing needs.

Fig. 5(b) and (c) show that the advantage of DGS is significantly reduced for light eMBB and mMTC users. These users are short-lived flows that do not require significant processing and hence can finish within their deadline. Additionally, these users introduce minimal interference due to their short-lived nature. Overall, DGS reduces the number of hosts required to 14 down from 17 as compared to POSENS, representing a net savings of more than 20%. Moreover, DGS is an online algorithm that executes at the granularity of a single user similar to LUF, while still maintaining a tractable amount of state.

uRLLC Allocations. All uRLLC users are allocated to a dedicated set of hosts because the subframe processing deadline for uRLLC is much lower at 0.5 ms. The vast majority of the drops are variable latency drops, and re-distributing the workload over more hosts will not reduce these drops. Next, we explore the use of replication on the reliability of transmission. With replication, we only require at least one of the replicated hosts to complete processing before the deadline. In our evaluation, replication is performed using 3 hosts. However, replication can only achieve the latency requirements for 99.95% of the transmissions, which still falls short of the 99.999% reliability requirement for uRLLC users. Thus RAN processing for uRLLC users often requires special purpose accelerators that can provide predictable performance [23] to ensure both high reliability and low latency.

6 CONCLUSION

In this paper, we show how existing CRAN proposals do not provide sufficient scaling. To enable fine-grained scaling, we introduce uvRAN that virtualizes user plane RAN processing. This enables per-user orchestration and improves isolation between users in the CRAN. We also propose a user-aware load balancer for allocating new users at runtime. We have built a working prototype of uvRAN on OpenAirInterface and show that uvRAN insulates users from interfering with each other. Additionally, we show that using uvRAN with the proposed load balancing algorithm DGS can improve the user QoS while requiring fewer hosts.

Acknowledgements. This work was supported by the Singapore Ministry of Education Academic Research Funds T1 251RES1910 and T1 251RES1905.

REFERENCES

[1] 2010. C-RAN - Road Towards Green Radio Access Network.
[2] 3GPP TR 38.801. 2017. Study on new radio access technology: Radio access architecture and interfaces.
[3] 3GPP TS 36.321. 2017. LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Medium Access Control (MAC) protocol specification.
[4] I. F. Akyildiz et al. 2015. SoftAir: A software defined networking architecture for 5G wireless systems. *Elsevier Computer Networks* (2015).
[5] X. An et al. 2017. On end to end network slicing for 5G communication systems. *Transactions on Emerging Telecommunications Technologies* (2017).
[6] J. Araújo et al. 2012. Weighted improper colouring. *Discrete Algorithms* (2012).
[7] V. Gupta at al. 2007. Analysis of join-the-shortest-queue routing for web server farms. *Performance Evaluation* (2007).
[8] M. Bansal et al. 2012. Openradio: A Programmable Wireless Dataplane. *ACM HotSDN* (2012).
[9] D. Bega at al. 2018. CARES: Computation-Aware Scheduling in Virtualized Radio Access Networks. *IEEE Transactions on Wireless Communication* (2018).
[10] N. Budhdev et al. 2018. PR 3: Power Efficient and Low Latency Baseband Processing for LTE Femtocells. *IEEE INFOCOM* (2018).
[11] N. Budhdev et al. 2020. Poster: IsoRAN: Isolation and Scaling for 5G RAN via User-Level Data Plane Virtualization. *IEEE IFIP Networking Conference* (2020).
[12] N. Budhdev et al. 2021. FSA: fronthaul slicing architecture for 5G using dataplane programmable switches. *ACM MOBICOM* (2021).
[13] Y. Chen et al. 2017. A dynamic BBU–RRH mapping scheme using borrow-and-lend approach in cloud radio access networks. *IEEE Systems Journal* (2017).
[14] U. Dötsch et al. 2013. Quantitative analysis of split base station processing and determination of advantageous architectures for LTE. *Bell Labs Technical Journal* (2013).
[15] D. Firestone et al. 2018. Azure Accelerated Networking:{SmartNICs} in the Public Cloud. *USENIX NSDI* (2018).
[16] X. Foukas et al. 2016. FlexRAN: A Flexible and Programmable Platform for Software- Defined Radio Access Networks. *ACM CoNEXT* (2016).
[17] G. Garcia-Aviles et al. 2018. POSENS: A Practical Open Source Solution for End-to-End Network Slicing. *IEEE Wireless Communications* (2018).
[18] A. Gudipati et al. 2013. SoftRAN: Software defined radio access network. *ACM SIGCOMM* (2013).
[19] NGMN Alliance. 2015. 5G: Next generation mobile networks. *White paper* (2015).
[20] N. Nikaein et al. 2014. OpenAirInterface: A flexible platform for 5G research. *ACM SIGCOMM Computer Communication Review* (2014).
[21] M. Själander et al. 2012. An LTE uplink receiver PHY benchmark and subframe-based power management. *IEEE ISPASS* (2012).
[22] Umesh et. al. 2019. Overview of O-RAN Fronthaul Specifications. (2019).
[23] V. Venkataramani et al. 2020. SPECTRUM: A software-defined predictable many-core architecture for LTE/5G baseband processing. *ACM Transactions on Embedded Computing Systems (TECS)* (2020).
[24] W. Wu et al. 2014. PRAN: Programmable radio access networks. *ACM HotNets* (2014).
[25] Y. Xie et al. 2020. PBE-CC: Congestion control via endpoint-centric, physical-layer bandwidth measurements. *ACM SIGCOMM* (2020).
[26] D. Zhu et al. 2013. Traffic and interference-aware dynamic BBU-RRU mapping in C-RAN TDD with cross-subframe coordinated scheduling/beamforming. *IEEE ICC* (2013).