Poster: Slicing 5G fronthaul networks using programmable switches

Nishant Budhdev, Raj Joshi, Pravein Govindan Kannan, Mun Choon Chan, Tulika Mitra
{nishant,rajjoshi,pravein,chanmc,tulika}@comp.nus.edu.sg
School of Computing, National University of Singapore

CCS CONCEPTS

• **Networks** → **Wireless access points, base stations and infrastructure**; **Programmable networks**.

KEYWORDS

5G Cellular Networks, Slicing, Network Scheduler, Programmable Switches

1 INTRODUCTION

Slicing is a critical technology in 5G, as it allows operators to slice a physical network into multiple virtual networks, each dedicated to a different use case/Mobile Virtual Network Operator (MVNO) [2]. Network slicing enables network operators to deploy a tailored set of resources for specific use cases or MVNO. For example, high performance reliable hardware is required *only* for ultra-reliable low-latency (uRLLC) use cases such as autonomous vehicle networks. Such tailoring of services reduces costs for network operators. Further, 5G systems can now be deployed more quickly due to virtualization provided by slicing, thereby enabling faster time-to-market. To this end, there exists a large body of work that introduces slicing in different parts of the cellular network (see Fig. 1). PRAN [12] and FlexRAN [13] provide slicing in the Radio Access Network (RAN) while Orion [14] provides slicing for the frontend (wireless spectrum). The fronthaul connects the frontend base station to the RAN and carries digitized radio signals between the two parts of the cellular network. However, to the best of our knowledge, there exists no work on slicing in the fronthaul. This severely limits the benefits of slicing in the RAN and the frontend (see §1.1).

In this paper, we first describe the need for slicing in the fronthaul and show why slicing in the fronthaul is challenging. Next, we propose a *high-frequency dynamic forwarding* scheme for uplink traffic which leverages programmable switches for achieving fronthaul slicing.

1.1 Motivation

Problem. State-of-the-art RAN slicing architectures such as IoS-RAN [8] and POSENS [6] provide significant performance improvement while using half the resources as compared to traditional architectures. These gains come from the ability to distribute RAN processing for different use cases/MVNOs on customized hardware. Therefore, the efficacy of these architectures is highly dependent on the ability to do slicing in the fronthaul. As shown in Fig. 1, slicing in the fronthaul creates independent logical networks for packets belonging to different use cases/MVNOs.

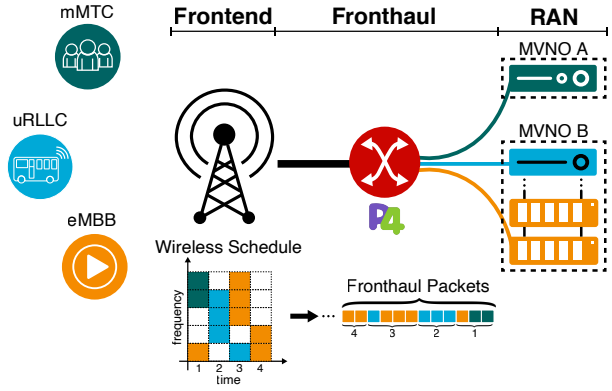

Figure 1: Fronthaul slicing with multiple use cases and network operators

Challenge. The key challenge for slicing in the fronthaul is the lack of user/slice information within the data packets. Since the fronthaul is responsible for carrying digitized radio signals belonging to the Physical Layer, user/slice identifiers available in the Access Layer (MAC) are missing. In essence, it is not possible to look at a packet in the fronthaul network and identify the network slice it belongs to.

Strawman Solution. A straightforward solution would be to implement fronthaul slicing on servers in the RAN as a network function. In such a scenario, the servers route the fronthaul data packets similar to traditional L4 load balancing [5] but with the additional information of the wireless schedule. However, since each base station in 5G is capable of generating over 200 Gbps [4] of fronthaul traffic, multiple servers would be needed to slice it. Furthermore, with each RAN managing a large number of base stations, 100s of servers would be needed for fronthaul slicing. This would dramatically increase both the CAPEX and OPEX of the RAN. Hence we propose using programmable switches for fronthaul slicing, which have been previously shown to replace hundreds of servers for traditional L4 load balancing [10].

2 DESIGN

Our key observation is that all wireless transmissions in the cellular network are scheduled by the network scheduler in advance. Additionally, traffic in the fronthaul is transmitted at a constant bit-rate with no re-transmissions. As a result, all packets in the fronthaul can be predicted in advance using the wireless schedule. Fig. 1 shows

CoNEXT '20, December 1–4, 2020, Barcelona, Spain

ACM ISBN 978-1-4503-7948-9/20/12…$15.00
https://doi.org/10.1145/3386367.3431668

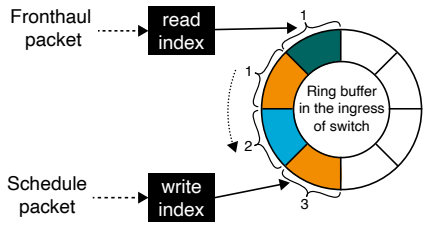

Figure 2: Overview of system design

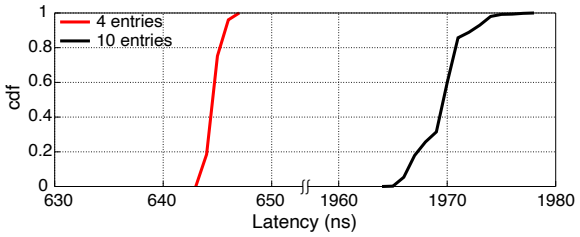

Figure 3: Latency for adding 4 and 10 scheduling entries in the dataplane

a simple representation of the relationship between the wireless schedule and the corresponding sequence of packets in the fronthaul. Thu, we propose a *high-frequency dynamic forwarding* scheme for fronthaul slicing which uses the wireless schedule to identify the network slice and forward the packet accordingly.

A simple solution would be to have a forwarding table that matches on the packet sequence number [4]. We can then add entries to the forwarding table according to the sequence of packet arrival derived from the wireless schedule (Fig. 1). In 5G, a wireless schedule is typically generated every 1 ms and each schedule can contain up to 10 users [1], leading to 10 forwarding table updates per ms. However, prior studies [7] have shown that hardware switches can only update up to ∼1.2 forwarding rules per ms via the control plane. Therefore, even with the network schedule being generated 4 ms [3] in advance, it is not possible to finish updating 10 forwarding rules per base station within this duration. Thus, we need a system design that can support high-frequency update of forwarding table rules.

We propose to leverage SRAM-based transactional stateful memory (called register arrays) in the dataplane of programmable switches to store the forwarding information corresponding to each schedule. For each base station, we create a ring buffer using register arrays as shown in Fig. 2. Since the ring buffer is implemented using register arrays, its entries can be updated in the dataplane at a high frequency. The ring buffer entries contain the destination address of the servers in order of the expected packet arrivals as determined by the wireless schedule. To update the entries, the scheduler generates a "Schedule Packet" and sends it to the switch every 1 ms by calling an API function. The API function creates a "Schedule Packet" which contains the destination address of the RAN server for each user in the wireless schedule. On receiving the "Schedule Packet", the switch writes new entries to the ring buffer and increments the write index. On the other hand, when a fronthaul packet arrives, the switch reads the destination server address from the ring buffer at the read index and forwards the fronthaul packet accordingly. Note that this scheme is applicable to the fronthaul traffic in the uplink direction only. In the downlink direction, such a scheme is not required since all the packets go to the destined base station.

3 EVALUATION & CONCLUSION

We have implemented our high-frequency dynamic routing scheme on a Barefoot Tofino programmable switch using ∼1000 lines of P4 code. For evaluation, we consider a 7-2x functional split [11] in the cellular network, where the common RF processing is executed at the base station while the user-specific processing happens in the RAN. We generate the corresponding fronthaul and schedule packets using Scapy. We have validated the correctness of our implementation for both reading and writing of the ring buffer using PTF tests.

The average and maximum number of users allocated in a wireless schedule are 4 and 10 respectively [3]. Therefore, to evaluate the latency for updating schedule in the ring buffer, we generate schedule packets with 4 and 10 entries. Fig. 3 shows the latency for updating the ring buffer after the schedule packet is received by the switch dataplane. We can see that updating 10 schedule entries requires less than 2 μs, which is 4000x faster than updating the forwarding table via the control plane (§2). Additionally, using techniques similar to DPTP [9], we measure the latency for sending a schedule packet from the RAN to the switch, and find it to be at most 1.5 μs. Overall, the total latency from the generation of the schedule packet to updating of the switch ring buffer takes at most 3.5μs. Thus the time to update the ring buffer is negligible when compared to the arrival rate of "Schedule Packets" which is 1 ms.

In future, we plan to perform evaluations using real network traces to simulate fronthaul traffic patterns for a large number of base stations.

Acknowledgement. We thank the anonymous reviewers for their valuable feedback. We would also like to thank Changhoon Kim and Jeongkeun Lee from Intel (Barefoot Switch Division) for their helpful suggestions. This research was supported by the Singapore Ministry of Education Academic Research Fund Tier 1 (Grant Number: T1 251RES1910).

REFERENCES

[1] 3GPP TR 38.802. 2017. Study on New Radio Access Technology Physical Layer Aspects. (2017).
[2] 3GPP. 2016. TS 28.500: Management Concept, Architecture and Requirements for Mobile Network that include Virtualized Network Functions. (2016).
[3] 3GPP. 2017. TS 36.213 v14.2.0 : E-UTRA Physical Layer Procedures. (2017).
[4] CPRI Consortium et al. 2019. eCPRI Specification V2.0.
[5] D. Eisenbud et al. 2016. Maglev: A fast and reliable software network load balancer. In *NSDI*.
[6] G. Garcia-Aviles et al. 2018. POSENS: A practical open source solution for end-to-end network slicing. *IEEE Wireless Communications* (2018).
[7] M. Kuźniar et al. 2015. What you need to know about SDN flow tables. In *PAM*.
[8] N. Budhdev et al. 2020. Poster: IsoRAN: Isolation and Scaling for 5G RAN via User-Level Data Plane Virtualization. In *IFIP Networking*.
[9] P. Kannan et al. 2019. Precise time-synchronization in the data-plane using programmable switching ASICs. In *SOSR*.
[10] R. Miao et al. 2017. Silkroad: Making Stateful Layer-4 Load Balancing Fast and Cheap Using Switching ASICs. In *SIGCOMM*.
[11] Umesh A. et al. 2019. Overview of O-RAN Fronthaul Specification. *NTT Docomo Technical Journal* (2019).
[12] W. Wu et al. 2014. PRAN: Programmable Radio Access Networks. In *HotNets*.
[13] X. Foukas et al. 2016. FlexRAN: A Flexible and Programmable Platform for Software- Defined Radio Access Networks. In *CoNEXT*.
[14] X. Foukas et al. 2017. Orion: RAN slicing for a flexible and cost-effective multi-service mobile network architecture. In *MOBICOM*.