SliceGuard: Secure and Dynamic 5G RAN Slicing with WebAssembly

Raphael Cannatà[1], Aoyu Gong[1], Arman Maghsoudnia[1], Dan Mihai Dumitriu[2], Haitham Hassanieh[1]
[1]EPFL, [2]Pavonis LLC

ABSTRACT

5G enables diverse services through network slicing, allowing multiple virtual networks to share physical infrastructure. However, efficiently managing resources across slices is challenging. This demo presents SliceGuard, a two-level scheduling system that leverages WebAssembly (Wasm) to allow slice owners to run customized schedulers in a secure, platform-independent environment, while the network operator manages inter-slice resource allocation. We demonstrate dynamic slicing for cloud gaming over 5G and show how Wasm enables real-time scheduler updates and fault isolation. This approach enhances flexibility, security, and customization for private 5G networks.

KEYWORDS

5G, RAN Slicing, WebAssembly, Service Level Agreement

ACM Reference Format:
Raphael Cannatà[1], Aoyu Gong[1], Arman Maghsoudnia[1], Dan Mihai Dumitriu[2], Haitham Hassanieh[1]. 2024. SliceGuard: Secure and Dynamic 5G RAN Slicing with WebAssembly. In *The 30th Annual International Conference on Mobile Computing and Networking (ACM MobiCom '24), November 18–22, 2024, Washington D.C., DC, USA.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3636534.3698862

1 INTRODUCTION

In the context of cellular communications, 5G introduced the support for diverse services, such as massive machine-type communications (mMTC), ultra-reliable low-latency communications (URLLC), and enhanced mobile broadband (eMBB) [7, 11]. To support these services, 5G introduced the concept of network slicing [1, 5, 10], which allows the creation of multiple virtual networks (slices) on top of a shared physical infrastructure. Each slice is tailored to meet specific service needs, e.g., mMTC slices can prioritize low power consumption [3], while URLLC slices optimize for low latency [9]. Resource isolation and scheduling for each slice is complex due to varying latency, throughput, and reliability demands.

Slice owners may differ from the network operator, e.g., an MVNO may want to adjust resource allocation to differentiate services or perform updates on the fly. Running the slice scheduler on the network operator's stack introduces challenges around trust, recompilation, and hardware support. To address this, we propose using WebAssembly (Wasm).

Wasm is a hardware- and OS-agnostic instruction format designed as a portable compilation target for high-level languages like C, C++, and Rust [6]. It offers portability and efficiency with features like binary encoding, linear memory, AOT compilation, and direct hardware access [2]. Wasm enforces strict memory safety through bounds checking and prevents control flow tampering. Wasm plugins run in sandboxed environments, ensuring that untrusted code does not affect the host system, making it secure [4].

We propose running the slice scheduler as a Wasm module on the network operators' infrastructure. This allows slice owners to customize resource allocation without full trust, and lets operators run the same code across its infrastructure.

In this demo, we present SliceGuard, a two-level Wasm-based scheduler designed to meet the diverse requirements of network slices. The system dynamically ensures each slice's SLAs while optimizing overall throughput. Running the schedulers as Wasm modules allows on-the-fly updates and fault isolation, ensuring seamless network operations even during changes. Additionally, we show how the Wasm-based scheduler can be dynamically loaded and updated without downtime, ensuring safe execution regardless of trust or code integrity.

2 SYSTEM'S OVERVIEW

To support the diverse requirements of 5G services, we designed a Wasm-based scheduler system that isolates network

ACM MobiCom '24, November 18–22, 2024, Washington D.C., DC, USA

ACM ISBN 979-8-4007-0489-5/24/11
https://doi.org/10.1145/3636534.3698862

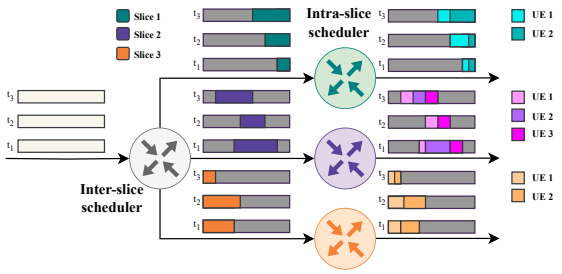

Figure 1: Two-level scheduler system.

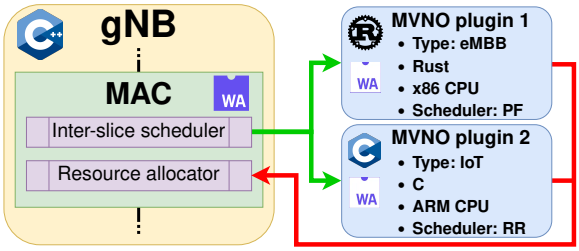

Figure 2: Outsourcing of intra-slice resource allocation to an external Wasm plugin

slices and guarantees the SLA of each slice. The system consists of two-level of schedulers: a slice scheduler and a user scheduler, see fig. 1. The first level, the inter-slice scheduler, is responsible for dividing the resources among slices, while the second level, the intra-slice scheduler, is responsible for dividing the resources between users subscribed to the same slice.

To perform the inter-slice resource allocation, we designed a slicing algorithm that is able to dynamically assign resources to three types of slices: Low Latency (LL) slices, Guaranteed Bit Rate (GBR) slices, and Best Effort (BE) slices. For LL slices, it timestamps packets when they are placed in the queues, and reserves the Resource Blocks (RBs) needed to avoid breaking the latency requirements. For GBR slices, it keeps track of the slice's average throughput via a moving average window, and assigns it a priority level using a logistic curve based on the distance from its target throughput. For BE slices, it assigns them the remaining resources after the LL and GBR slices have been allocated their resources. In doing so, it is able to guarantee the SLA of each slice, while maximizing the overall system throughput.

Moreover, to guarantee security between the network operator and the slice tenants, to allow on-the-fly updates, and to avoid recompilation of the code, our system runs the intra-slice scheduler as a Wasm module, see fig. 2. In this architecture, the network operator runs the inter-slice scheduler, which decides the resources allocated to each slice. This information is then passed to the Wasm modules, which run the intra-slice scheduler. This way, the slice tenants can customize their resource allocation simply by loading a new bytecode in the Wasm sandbox. The network operator can run the same code on all its infrastructure, without the need to trust the slice tenants.

Figure 3: Hardware setup

3 DEMONSTRATION

3.1 Goals of the demo

This demo aims to highlight the need for dynamic slicing in 5G networks and demonstrate the capabilities of Wasm in 5G setups. Specifically, we show how slicing enhances application performances, enables real-time updates, and isolates faults.

Slicing for cloud gaming over 5G. We demonstrate the performance of two smartphones connected to our software-based private 5G network. A local cloud gaming server renders frames and transmits them over the network to the users. The demo compares a single-slice network with a multi-slice setup and show how different slice types impact gaming quality.

Wasm capabilities in 5G RAN. First, we show on-the-fly change of the intra-slice scheduler's code. Moreover, we demonstrate how Wasm isolates issues, such as segmentation faults, preventing them from affecting other slices. For this, we load a buggy scheduler for one of the slices and show that only the phone in the buggy slice stops working, while the other two phones continue functioning normally.

Another result which we demonstrated is the change of user's slices on-the-fly using the Wasm plugins.

3.2 Demo's Setup

Our setup is depicted in fig. 3. We use open-source 5G RAN stack [12] and core [8]. We modified the RAN code to support slicing. The RAN and Core code are executed on a commodity Intel NUC. For the radio transceiver, we use a USRP B210, which is connected to the RAN machine. The cloud gaming server runs on a laptop, which renders and sends the frames to the RAN machine over a LAN to be further transmitted over the air to the two tablets that we use for remote gaming.

ACKNOWLEDGMENTS

The authors would like to thank Haoxin Sun for his contributions to building this system.

REFERENCES

[1] 3GPP. 2022. *5G; System architecture for the 5G System (5GS).* Technical Report TS 28.530 version 17.2.0 Release 17. 3GPP. 7–17 pages. https://www.etsi.org/deliver/etsi_ts/123500_123599/123501/17.05.00_60/ts_123501v170500p.pdf

[2] Rick Battagline. 2021. *The Art of WebAssembly: Build Secure, Portable, High-Performance Applications.* No Starch Press, San Francisco, CA.

[3] Carsten Bockelmann, Nuno K. Pratas, Gerhard Wunder, Stephan Saur, MòNica Navarro, David Gregoratti, Guillaume Vivier, Elisabeth De Carvalho, Yalei Ji, Čedomir Stefanović, Petar Popovski, Qi Wang, Malte Schellmann, Evangelos Kosmatos, Panagiotis Demestichas, Miruna Raceala-Motoc, Peter Jung, Slawomir Stanczak, and Armin Dekorsy. 2018. Towards Massive Connectivity Support for Scalable mMTC Communications in 5G Networks. *IEEE Access* 6 (2018), 28969–28992. https://doi.org/10.1109/ACCESS.2018.2837382

[4] Jules Dejaeghere, Bolaji Gbadamosi, Tobias Pulls, and Florentin Rochet. 2023. Comparing Security in eBPF and WebAssembly. In *Proceedings of the 1st Workshop on EBPF and Kernel Extensions* (New York, NY, USA) *(eBPF '23).* Association for Computing Machinery, New York, NY, USA, 35–41. https://doi.org/10.1145/3609021.3609306

[5] Xenofon Foukas, Georgios Patounas, Ahmed Elmokashfi, and Mahesh K. Marina. 2017. Network Slicing in 5G: Survey and Challenges. *IEEE Communications Magazine* 55, 5 (2017), 94–100. https://doi.org/10.1109/MCOM.2017.1600951

[6] Andreas Haas, Andreas Rossberg, Derek L Schuff, Ben L Titzer, Michael Holman, Dan Gohman, Luke Wagner, Alon Zakai, and JF Bastien. 2017. Bringing the web up to speed with WebAssembly. In *Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation.* Association for Computing Machinery, New York, NY, USA, 185–200. https://doi.org/10.1145/3062341.3062363

[7] Oğuz Sunay Larry Peterson. 2020. *5G Mobile Networks.* Springer Cham. https://doi.org/10.1007/978-3-031-79733-0

[8] Sukchan Lee. 2017. Open5GS. https://open5gs.org/. [Online; Last accessed: 5-Sept-2024].

[9] Zexian Li, Mikko A. Uusitalo, Hamidreza Shariatmadari, and Bikramjit Singh. 2018. 5G URLLC: Design Challenges and System Concepts. In *2018 15th International Symposium on Wireless Communication Systems (ISWCS).* Institute of Electrical and Electronics Engineers, Piscataway, NJ, USA, 1–6. https://doi.org/10.1109/ISWCS.2018.8491078

[10] Petar Popovski, Kasper Fløe Trillingsgaard, Osvaldo Simeone, and Giuseppe Durisi. 2018. 5G Wireless Network Slicing for eMBB, URLLC, and mMTC: A Communication-Theoretic View. *IEEE Access* 6 (2018), 55765–55779. https://doi.org/10.1109/ACCESS.2018.2872781

[11] M Series. 2015. IMT Vision: Framework and overall objectives of the future development of IMT for 2020 and beyond. *Recommendation ITU* 2083, 0 (2015).

[12] Software Radio Systems. 2022. srsRAN Project. https://github.com/srsran/srsRAN_Project. [Online; Last accessed: 5-Sept-2024].