Towards Scalable and Cost-Effective RAN Emulation Leveraging the Public Cloud

Ujjwal Pawar[†], Andrew E. Ferguson[†], Yuto Takano[†], Jon Larrea[†], Xenofon Foukas[‡]
Mahesh K. Marina[†] and Bozidar Radunovic[‡]
The University of Edinburgh[†] Microsoft Research[‡]

Abstract

The rise of data-driven and AI-powered monitoring in 5G and beyond has created a need for RAN digital twins to test and enhance network performance without disrupting real-world operations. However, large-scale RAN emulation remains challenging due to the complexity of the RAN protocol stack and the high cost of hardware-based solutions. Software alternatives are more affordable but current solutions sacrifice fidelity or scalability. This paper presents Chronos, a cloud-based RAN emulation system that overcomes these challenges by abstracting the compute-intensive PHY layer, virtualizing emulation time, RAN slot level synchronization and scalable switch based traffic forwarding. Experiments using OpenAirInterface based prototype on public cloud infrastructure demonstrate Chronos' ability to emulate 250 base stations and 1000 UEs with 468 CPU cores at high fidelity, reflecting its potential for enabling scalable RAN emulation in an affordable manner.

CCS Concepts

• **Networks** → **Network simulations**; **Mobile networks**; **Cloud computing**.

Keywords

RAN Emulation, 5G Digital Twin, Public Cloud

ACM Reference Format:
Ujjwal Pawar[†], Andrew E. Ferguson[†], Yuto Takano[†], Jon Larrea[†], Xenofon Foukas[‡], Mahesh K. Marina[†] and Bozidar Radunovic[‡]. 2025. Towards Scalable and Cost-Effective RAN Emulation Leveraging the Public Cloud. In *The 26th International Workshop on Mobile Computing Systems and Applications (HOTMOBILE '25), February 26–27, 2025, La Quinta, CA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3708468.3711895

1 Introduction

The emergence of data-driven, AI powered monitoring and control of the mobile Radio Access Network (RAN) in the 5G and beyond era has created a need for RAN digital twins [15–17]. Assisted by the softwarization of the RAN, such emulated environments can be used to evaluate new ideas and test new services that can improve the RAN performance and the user experience, without affecting the real network's operation. This opens up exciting new avenues for the emulation of large scale deployments (e.g., hundreds of base stations or even a whole city), which so far have been impossible to create and experiment with in resource-constrained lab environments.

Despite its promise, the opportunity for RAN emulation at scale still largely remains untapped. To emulate a large-scale production RAN environment with high fidelity, one has to accurately capture the behavior and interactions of its internal components (e.g., radio resource scheduler) and the message exchanges of all the involved protocols (e.g., MAC, RLC, RRC). Considering the high complexity that characterizes the RAN protocol stack, this can only be achieved if the digital twin leverages the same unmodified RAN software that is also used in the production environment. Today, such a capability is only offered by sophisticated emulators that are built on top of custom hardware (e.g., [2, 6]). This hardware based approach makes such solutions very expensive to deploy and scale on demand, limiting their use to the labs of major telco vendors and operators. Similarly, channel emulators [1, 18] and testbeds [8, 9] are expensive to scale.

In contrast, software based approaches (e.g., [12, 13, 19]) offer a cost-effective alternative. Existing solutions in this category sacrifice fidelity to different degrees and vary in the scale they can support. Simulators like ns-3 [13] use a simplified RAN stack that only captures some aspects of the RAN functionality (e.g., a specific RAN protocol or component). Such simulators can scale well and can be useful for early prototyping, but have met limited success in practice from not providing the assurance needed for real-network deployments. On the other hand, software based RAN emulators like EMANE [19] take a different track by abstracting away the computationally heavy physical (PHY) layer and can in principle support an unmodified RAN stack. However, they scale poorly, as we show through our evaluations (§4).

Motivated by these observations, we ask the following question: *Can we push the boundary of the software based approach to cost-effectively achieve real-world, high-fidelity RAN emulation at scale (with at least 100s of base stations and mobile devices) by leveraging the elasticity and scale of the public cloud*? The capability of the cloud for on-demand allocation of compute and network resources at scale means that one could perform large scale emulations with real (unmodified) RAN software without the need for high upfront investments or maintenance costs.

While this is a compelling proposition, running RAN emulation scenarios in a public cloud setting and at large scale presents new challenges, which stem from the resource demanding nature of the RAN, its stringent latency requirements, and the high CPU and network latencies characterizing the public cloud. This is because RAN functions demand substantial compute and network resources, often requiring multiple CPU cores and gigabits of traffic per cell.

HOTMOBILE '25, La Quinta, CA, USA

ACM ISBN 979-8-4007-1403-0/25/02
https://doi.org/10.1145/3708468.3711895

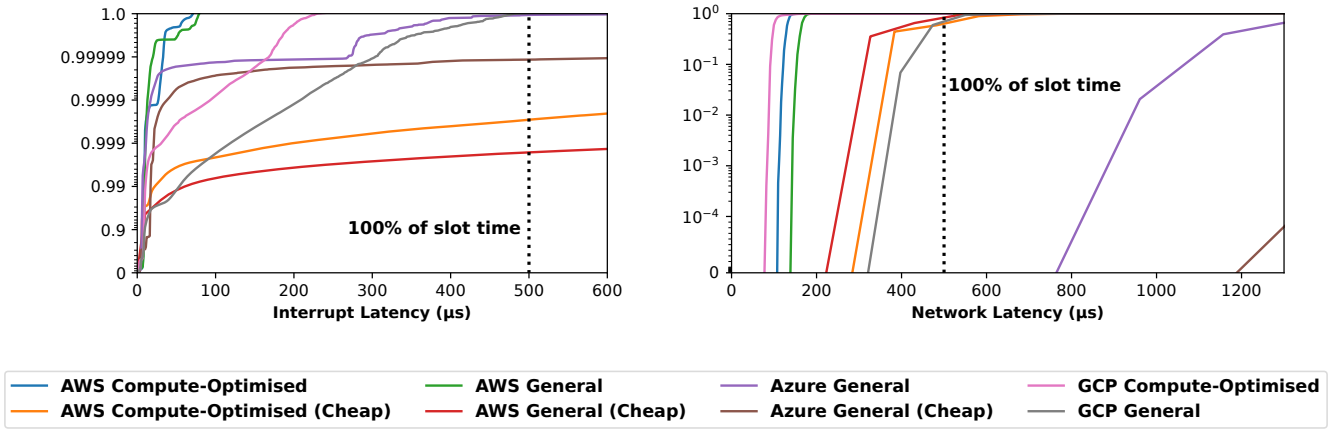

Figure 1: CDFs of VM interruption and network latencies for different cloud providers.

Scaling emulation to hundreds of base stations becomes prohibitively expensive, needing thousands of CPU cores and terabits of capacity. Additionally, RAN software must meet strict sub-millisecond latency requirements, which public clouds struggle to support due to unpredictable hypervisor preemptions and high networking latencies from multi-layered infrastructure. These delays can violate RAN runtime deadlines, compromising emulation fidelity.

To overcome these challenges, we propose a cloud based RAN emulation system design called Chronos. The emulator brings together multiple key ideas to enable RAN emulation at any scale: i) it replaces the compute and network intensive PHY layer and channel of the RAN and associated user devices (UEs) with an emulated link based on the standard FAPI interface; ii) it introduces a custom hypervisor that virtualizes the emulation time, effectively shielding the emulated network functions from external CPU and network latencies; iii) it operates at the granularity of RAN slots and uses the slots as synchronization barriers among emulation components to allow flexible scaling; and iv) it introduces a logically centralized software switch to forward the control and data plane FAPI traffic between the RAN and the emulated mobile devices in a scalable manner.

To demonstrate the benefits of Chronos' design, we develop a Proof of Concept (POC) of the system on Ubuntu Linux, using a custom Kernel-based Virtual Machine (KVM) Hypervisor and a RAN based on the OpenAirInterface (OAI) software stack. Our experiments, performed over a public cloud, highlight the fidelity of our emulator in terms of both control and data plane operations when compared with a standard over-the-air RAN deployment with real UEs, using the same OpenAirInterface stack as in the emulation setting. Furthermore, they demonstrate the scalability of Chronos' design, which allows us to achieve emulation scenarios with 250 base stations and 1000 UEs using 3 cloud VMs with an aggregate of 468 CPU cores. To our knowledge, this is the first work that has demonstrated high-fidelity RAN emulation at such scales cost-effectively, bringing us one step closer to realizing the vision of a scalable and affordable RAN digital-twin.

2 Challenges of large scale RAN emulation in the public cloud

Here we discuss the challenges of performing large scale RAN emulation in the public cloud.

RAN resource requirements – The 5G RAN network functions can be very demanding in terms of compute and network resources, requiring several CPU cores and high network bandwidth for their operation. As an example, we deploy a single 100MHz 4×4 MIMO 5G cell based on the open source srsRAN software stack [5] and measure its CPU and network utilization when fully saturated with traffic. We observe that a minimum of 8 CPU cores and a 10G NIC interface is required to achieve full downlink throughput (~1Gbps DL). Upon further inspection, we observe that more than 60% of the CPU resources are required for the PHY signal processing. Similarly, more than 3.5Gbps of the cell's network traffic correspond to the fronthaul traffic (radio signals) going out of the base station, towards the radio unit. Similar observations can also be made for other RAN stacks [3, 11]. As such, emulating a large scale network with hundreds of base stations would require thousands of CPU cores and terabits of network capacity, which would make the emulation unviable, due to the cost or the availability of resources.

RAN latency requirements – The RAN operates in a time slotted fashion, with each slot having a duration in the order of hundreds of microseconds up to a millisecond. During each slot, the MAC scheduler of the RAN needs to perform the radio resource allocation in the control plane for both the uplink and the downlink direction. This involves deciding which users to schedule, their traffic flow priorities, how many resource blocks to allocate, etc. At the same time, the RAN also has to deal with the processing of the actual packets of users in the dataplane (e.g., segmentation, adding headers, retransmissions). Both these control and the data plane operations must take place within the duration of the slot. Otherwise, emulation fidelity will suffer due to packet drops, user detaches or even worse RAN crashes.

Cloud VM interruption and network latencies – Public clouds provide an almost infinite amount of compute and network resources on demand, making them promising environments for

realizing digital twins. However, they have been designed with cost and scale in mind considering general purpose workloads, but are not optimized for latency sensitive workloads like the RAN. For example, a single server may be shared between multiple different VMs causing frequent VM preemptions. Similarly, VMs of a single virtual network can often be placed across different racks, with multiple layers of switches between them. While this design is appropriate for the vast majority of cloud workloads, it can lead to serious problems when used for RAN emulation, where the tail latency has to be strictly bounded.

The magnitude and frequency of such VM interruptions and network latencies can vary across cloud providers, or even within the same provider, but they are always present and therefore problematic. To quantify the nature of these latencies, we performed measurements, using cloud VMs of different providers (Microsoft Azure, Amazon Web Services (AWS) and Google Cloud Platform (GCP)), including different flavors of VMs from the same provider tailored for reduced cost or higher performance. We used `osnoise` [4], provided as part of the Linux kernel, for measuring CPU interference and thus VM interruption latencies. For network latencies, we measured the `ping` RTT time for two VMs of the same type, that are co-located within the same region.

From the measurement results shown in Figure 1, Right, we observe that network related latency is relatively higher. Considering a typical 5G slot time of 0.5ms, the best median network latency is around 20% of the slot time and the tail latency is close to 40%. Given the scale of a public cloud, this is a fundamental challenge, since the likelihood that all VMs taking part in the emulation will be located close to each other (e.g., the same TOR switch) becomes increasingly smaller as we scale out the emulated network size.

The tail VM interruption latency of most cloud VMs is also high compared to what would be seen on a local VM (Figure 1, Left). Several of them have tail latencies that exceed 20% of the time allocated to one slot. The differences between the latency distribution of different variants stem from the technology used and the hypervisor configuration, with cheaper machines generally having higher tail latencies.

Overall, given these observed latencies, we can see that the RAN software can spend a large fraction of a slot time (> 20%) waiting. Since each missed deadline reflects in packet loss, this directly affects the emulation fidelity – we observe packet losses of up to 20% if the RAN software runs *as-is* in the public cloud.

3 Chronos System Overview

We aim to achieve high fidelity and cost-effective large scale RAN emulation leveraging the public cloud by addressing the challenges highlighted in the previous section. To this end, we introduce Chronos, our cloud based RAN emulation system design. We start by first outlining our approach and system architecture (illustrated in Figure 2), and then describe its proof-of-concept implementation.

3.1 Approach and System Architecture

To alleviate the high RAN resource requirements, and thereby allow scalable and cost-effective RAN emulation in the cloud, we choose to replace the real PHY and channel between a base station and

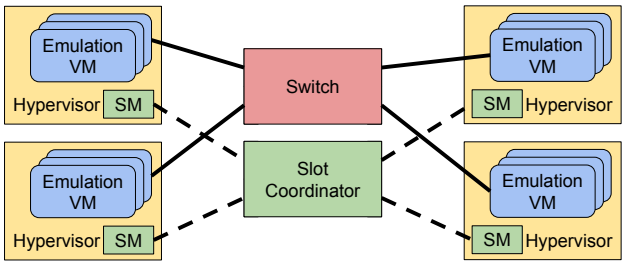

Figure 2: Chronos Architecture Schematic. SM: Slot Monitor.

associated UEs with an "emulated link", leveraging the standard FAPI interface [10]. While the FAPI was originally intended for standardizing the interface between the MAC and PHY layers on the base station side, we extend this to the UE side. This not only permits using real unmodified RAN stacks on the base station side (DU and CU – MAC layer and above) but also allows communicating directly between 3GPP compliant base stations and softwarized UEs through their MAC layers using FAPI message exchanges. While this approach has been adopted in prior software based RAN emulators (notably in EMANE [19]), we use this as one piece of a comprehensive approach towards scalable RAN emulation over the public cloud infrastructure.

To meet the stringent timing requirements for high-fidelity RAN emulation in the presence of vagaries characteristic of the public cloud environment (VM interruption and network latencies), our approach involves "virtualizing" the time. To realize this idea, our design features a custom VM `Hypervisor` that runs inside each cloud VM instance used for realizing a RAN emulation scenario (with a given number of base stations, UEs and their traffic and mobility behaviors). This `Hypervisor` shields any network function in the emulation scenario (e.g., RAN, core, UEs, user applications) deployed in nested "emulation VMs" from external sources of latency that can affect the real-time behavior and deadlines of the RAN. Emulation VMs are the VMs deployed over the custom `Hypervisor` to realize the various components of an emulation scenario (CU/DU vRAN functions of a base station, emulated RUs, emulated UEs, mobile core). The `Hypervisor` achieves this goal by creating a sandboxed environment, where the time is virtualized and advanced based on the progress of the emulated components (similar in spirit to [7]), rather than based on the real time that passes in the physical world. This gives the network functions the illusion that no deadline violations ever occur (due to VM preemptions, high network latency, limited compute or network resources etc.), meaning that the correct behavior of the RAN is always ensured.

Building on the above, we enable RAN emulation of *any* scale by allowing the use of multiple emulation VMs within and across cloud VM instances, while not compromising fidelity. To make this possible, we advance the emulation time at the granularity of the RAN slot duration (1ms in 4G and as low as 0.125ms in 5G) and then use the slots as natural synchronization barriers for emulation VMs. We use a `Slot Monitor` (Figure 2) within each `Hypervisor` to monitor and ensure that each of its hosted emulation VMs finish their processing within the current slot. The `Slot Coordinator` plays the

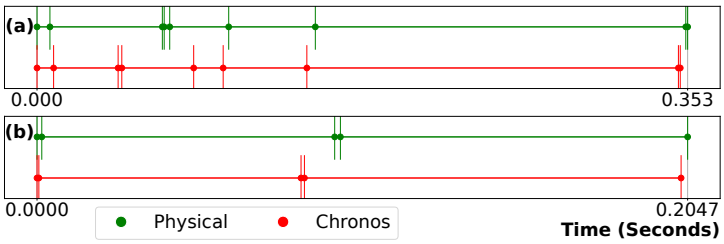

Figure 3: Control plane fidelity: Physical (testbed) vs. Chronos.

Scenario	Single UE (Mbps)			Two UEs (Mbps)		
	Physical	Chronos	EMANE	Physical	Chronos	EMANE
TCP UL	18	18	14.4	10.1, 8.09	9.98, 8.30	8.33, 6.28
TCP DL	32	32	29.2	17.3, 15.3	17.9, 15.7	16.1, 15.6
UDP DL	35	35	32.9	17, 17	17, 17	17.4, 16
UDP UL	18.5	18.5	15	8.97, 8.94	8.75, 8.95	9.94, 7.06

Figure 4: Data plane fidelity.

same role at the level of cloud VM instances, each with its own custom `Hypervisor`. The `Slot Coordinator` additionally advances the emulation to the next slot, once every emulation VM finishes with the current slot. This ensures that emulation progresses once all VMs complete processing in the slot.

To enable the communication of emulation VMs across cloud VM instances, our design employs a logically centralized software `Switch`, which is responsible to forward the control and data plane traffic between a base station and its emulated UEs. Mimicking the broadcast nature of the communication medium between base stations and UEs as in the real world can lead to a very high network bandwidth requirement, especially in the downlink from base station to UE as every message regardless of its intended destination UE is received by every UE. To avoid such redundant network bandwidth usage, the `Switch` in Chronos filters the message and logically creates point-to-point communication between base stations and emulated UEs in both the uplink and downlink directions. Our design, as outlined above allows realizing any given RAN emulation scenario at high fidelity by suitably *dilating* the emulation runtime to cope with the underlying public cloud characteristics and the provided (compute and network) resources.

3.2 POC Implementation

We developed a POC implementation of Chronos on Ubuntu 20.04 as per the above outlined design, consisting of all the system components shown in Figure 2.

The `Hypervisor` is implemented based on the KVM Hypervisor (in kernel version 5.15.160) and setup to use our `clock source`. The `clock source` is implemented as a kernel module to generate a new Time Stamp Counter (TSC) value every 10 microseconds using a high-resolution timer (HRtimer). This module also provides an interface, via shared memory, to dynamically adjust the flow of time, enabling on-demand time dilation. `Slot Monitor` running in userspace can stop time through this shared memory.

The `Slot Monitor` and the `Slot Coordinator` communicate via UDP sockets. The `Slot Coordinator` generates nFAPI slot indication messages and interacts with the `Slot Monitor` to signal the start of a slot. Upon receiving a slot start message from the coordinator, the `Slot Monitor` starts a timer for the slot duration. Once the timer expires, it uses the shared memory interface provided by the local `clock source` to pause the clock, keeping it paused until the next slot start is received.

4 Preliminary Evaluation

Here we present our preliminary evaluation of Chronos using its POC implementation. All our experiments were conducted on the public cloud using the GCP. We have also tested the functionalities of our custom kernel and hypervisor on AWS and Azure. To simplify the deployment, we set up a Kubernetes cluster with emulation VMs running on our `Hypervisor`. The `Switch` runs on a separate machine with an unmodified Linux kernel provided by GCP.

For the RAN, we used the OpenAirInterface 4G codebase [3]. To detect when the RAN has completed processing a slot, we evaluated two approaches: modifying the RAN to send an explicit indication message, or using eBPF probes to monitor the RAN's state. Both strategies were tested, and for this evaluation, we used the former. For the UE, we used the OpenAirInterface emulator UE, which is compliant with 3GPP standards, and modified it to interface with the `Switch`.

4.1 Fidelity

For fidelity evaluation, we deploy a testbed consisting of two OAI 4G RAN base stations, an Open5GS core, and a Commercial Off-The-Shelf (COTS) UE, with the RAN operating in FDD mode at 10 MHz for all experiments unless otherwise specified. We then create a Digital Twin (DT) of this setup on GCP using Chronos, where the RAN and core versions are identical to the real testbed, but instead of using the COTS UE, we utilize our emulated UE. To build this digital twin, we use three GCP machines, each with 30 cores. Two of the machines run Ubuntu 20.04 with a modified kernel with `Hypervisor`. Each machine hosts two RAN emulation VMs each. A third machine, running Ubuntu 20.04 with the default GCP kernel, is used for the `Switch` and `Slot Coordinator`.

To evaluate the fidelity of our DT, we focus on three common mobile network operations: Attach, Handover, and Data Transfer. Figure 3 (a) illustrates the timing of events during the Attach procedure, showing that both the DT and the testbed follow a similar timeline. Likewise, Figure 3 (b) shows the events during the Handover procedure, where handover events are occurring at the same intervals in both cases. We also measure the average time required to complete these procedures using a COTS UE and emulated UE. Both the Attach and Handover procedures took nearly the same average time in both environments. Specifically, the handover was completed in 0.2 seconds, while the attach procedure took 0.35 seconds. These two experiments demonstrate the fidelity of our DT in replicating control plane procedures, as both the timing of

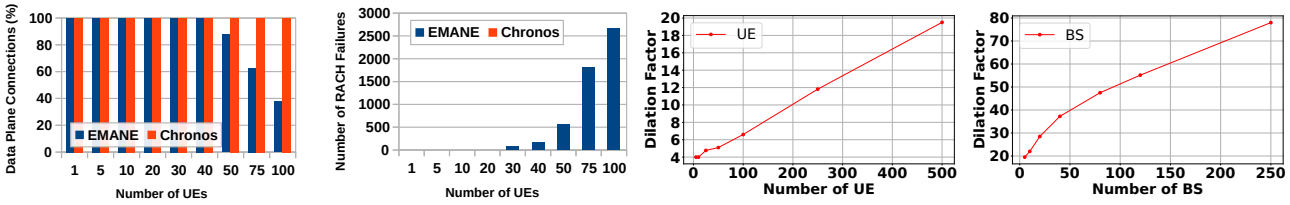

Figure 5: (a) Scalability of Chronos vs. EMANE, expressed as the percentage of UEs able to attach and form a data plane connection; (b) Number of reported RACH failures at varying scales of UEs for EMANE. Chronos not shown as there were zero RACH failures. Scaling of Chronos given a fixed compute resource with adaptive dilation of emulation runtime with (c) varying UEs and (d) varying BSs.

events and the total duration closely match those observed in the testbed.

To evaluate data plane fidelity, we generate various types of traffic in both the DT and the real testbed using iperf3. The results, shown in Table 4, demonstrate that the DT performs similarly to the testbed, achieving comparable throughput across all traffic types. Most notably, the DT replicates the scheduler behavior seen in the testbed. This is evident in the two-UE experiments with TCP traffic, where one UE consistently achieves higher throughput in both uplink (UL) and downlink (DL). This behavior occurs because the scheduler implementation in the OAI version used for the experiment assigns a minimum of 3 Resource Blocks (RB) for downlink and 5 RB for uplink. In the DL, 50 RB were available for assignment, while 45 were available for UL. As a result, one UE consistently received 24 RB in the DL and 20 RB in the UL, while the other received 26 RB in the DL and 25 RB in the UL. We verified this behavior across different bandwidths – 20 MHz (100 RB) and 5 MHz (25 RB) – and observed the same pattern. One UE consistently received more RBs due to the scheduler's minimum RB allocation. Since both the DT and the testbed run the same scheduler algorithm and, more importantly, the exact same code, the behavior was identical in both setups. This is important because it allows large-scale scheduler testing using the same codebase, ensuring fidelity. Reimplementing in simulators like ns-3 could introduce inaccuracies, but our DT avoids this risk.

Furthermore, we compare the fidelity of our design with the EMANE system [19], which also uses PHY bypass and nFAPI for RAN-UE communication through a middlebox called proxy. EMANE was deployed on an identical setup (three GCP machines, each with 30 cores), albeit with an unmodified OS rather than `Hypervisor`. Both emulators used the same version of the OAI RAN and OAI UE, with their respective middlebox designs. We observe (Table 4) noticable differences between the performance of EMANE and that of COTS UE and Chronos. Specifically, the throughput is in all cases lower for EMANE, due to packet losses caused by the cloud hypervisor preemptions and networking latencies. Indeed, we observed packet losses of up to 20% for EMANE. For the UDP uplink two UE case, although EMANE has higher throughput on one UE, the aggregated throughput on both is lower.

4.2 Scalability

We evaluate the scalability of our design through comparison with EMANE. The first test used the same deployment as in the fidelity experiment, and involved attaching UEs to a single base station (BS) and measuring the percentage of UEs that were able to successfully establish a data plane connection to the core, as the number of UEs increased. Figure 5a shows the results. Both systems successfully attached 40 UEs without any data plane failures. However, starting from 50 UEs, EMANE began experiencing data plane failures, with not all of the UEs being able to establish a data plane connection. In contrast, Chronos achieved 100% data plane connectivity for all scales of UEs tested.

In Figure 5b, we dig deeper into the underlying issues in EMANE that prevent the data plane in some UEs from working. Specifically, we observe an increasing number of PRACH failures, as reported by the BS, when the number of UEs increases above 30. These failures were due to timer expirations, not contention (a normal occurrence in 4G/5G initial attach). Timer expiry failures happened because messages were not received in the correct slot. In contrast, Chronos experienced *zero* PRACH failures for all the scales tested, and as such is not shown in Figure 5b.

To further test the scalability of our design, we deployed Chronos on machines with higher core counts. We used three 96-core machines running our modified kernel and `Hypervisor`, each hosting one VM with 95 cores. As in previous experiments, we set up a three-node Kubernetes cluster using these VMs to simplify deployment. Of the three nodes, the RAN was restricted to two, while the UEs and Core could run on any machine. For the `Switch`, we used a 180-core machine.

In this experiment, our goal was to explore the maximum achievable scale and observe how the time dilation factor changes with increasing scale. The dilation factor was not manually set – it is a feature of our design, allowing time dilation to occur in response to slot processing and network delays. Figure 5c illustrates how the dilation factor changes as we increase the number of UEs with a fixed set of base stations (BSs). We began with 10 UEs across 5 BSs and gradually increased the number to 500. As shown, the dilation factor rises with the increasing number of UEs, as the `Switch` must manage a greater load. In summary, the dilation factor increases from 2 for 10 UEs to 19 for 500 UEs.

Similarly, Figure 5d shows how the dilation factor changes when the number of BSs is increased while keeping the UEs fixed at

500. In this case, we varied the number of BSs from 5 to 250. As expected, the dilation factor also increases with more BSs; however, the rate of increase is significantly higher compared to the previous figure. This is due to the greater computational requirements of BSs, which run a more complex processing pipeline than the UEs. In this case the dilation factor went from 19 for 10 BSs to 78 for 250 BSs. In all above tests, the Attach success rate was 100%. During our testing, we also deployed scenarios with up to 1000 UEs and 250 BSs. Thanks to the scalability achievable with Chronos, we can deploy a large number of UEs and BS to test complex real-world scenarios cost-effectively, which is currently not feasible with alternatives (hardware or software).

5 Limitations and Future Work

In the POC, the focus was on validating the Chronos design, so we make some simplifications. For example, we do not incorporate any channel modeling. However, some effects of the channel can be simulated using channel models to generate Channel Quality Indicator (CQI) values for the UEs [14]. Additionally, testing at larger scales revealed inefficiencies in the implementation of certain components (particularly the `Switch` and PHY layer bypass library), resulting in additional overhead and slower emulation. Addressing this is planned for future work. Crucially, we intend to comprehensively evaluate the ability of Chronos to emulate large city-scale scenarios (thousands of BSs and UEs).

While we argue that the Chronos design is general and applicable to 4G and 5G, the POC presented was 4G-only, due to the maturity of existing open source RAN software stacks and missing features like X2 Handover in OAI 5G Stack. Extending the implementation to 5G is essential to enable Chronos to be used in the modeling and research of 5G networks, an area (along with other use cases) that we have significant interest in.

Another limitation of Chronos is its reliance on a specific RAN software stack. We are in the early stages of exploring how to separate the different components of Chronos to enable third-party RANs to be used, including those with different architectures such as a lack of PHY layer bypass.

Finally, during the evaluation of the POC it became clear that there is a non-trivial relationship between the complexity of the scenario being emulated, the amount of time required to emulate the scenario, and the compute resources required. We intend to explore this relationship, as we believe a deeper understanding would reduce the complexity of Chronos, and enable its wider use in research.

Acknowledgments

We thank our shepherd, Shubham Jain, and the anonymous reviewers for their helpful comments and suggestions that greatly improved this paper. This work was supported by a project funded by the UK Department for Science, Innovation and Technology (DSIT).

References

[1] Keysight Propsim. https://www.keysight.com/us/en/products/channel-emulators/propsim-platforms.html.
[2] Keysight UESIM. https://www.keysight.com/us/en/product/P8800S/uesim-ue-emulation-ran-solutions.html.
[3] OpenAirInterface. https://openairinterface.org/.
[4] OSNOISE. https://docs.kernel.org/trace/osnoise-tracer.html.
[5] srsRAN Project. https://www.srslte.com/.
[6] Viavi TM500. https://www.viavisolutions.com/en-us/products/tm500-network-tester.
[7] V. Babu and D. Nicol. Precise virtual time advancement for network emulation. In *Proc. ACM SIGSIM-PADS '20*, pages 175–186, 2020.
[8] P. Bahl, M. Balkwill, X. Foukas, A. Kalia, D. Kim, M. Kotaru, Z. Lai, S. Mehrotra, B. Radunovic, S. Saroiu, et al. Accelerating Open RAN Research Through an Enterprise-scale 5G Testbed. In *Proc. ACM MobiCom '23*, pages 1–3, 2023.
[9] J. Breen et al. Powder: Platform for Open Wireless Data-driven Experimental Research. *Computer Networks*, 197:108281, 2021.
[10] S. C. Forum. FAPI specifications. https://www.smallcellforum.org/work-items/fapi.
[11] X. Foukas and B. Radunovic. Concordia: Teaching the 5G vRAN to share compute. In *Proc. ACM SIGCOMM '21*, pages 580–596, 2021.
[12] U. Ghosh, I. K. Jain, D. Bharadia, and S. Shakkottai. Poster: Tiny-twin: A Lightweight and Verifiable Digital Twin for NextG Cellular Networks. In *Proc. ACM HOTMOBILE '24*, pages 145–145, 2024.
[13] T. Henderson, M. Lacage, G. F. Riley, C. Bèrnier, and S. Floyd. Network Simulations with the ns-3 Simulator. In *Proc. ACM SIGCOMM '08*, 2008.
[14] W.-H. Ko, U. Ghosh, U. Dinesha, R. Wu, S. Shakkottai, and D. Bharadia. EdgeRIC: Empowering real-time intelligent optimization and control in NextG cellular networks. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, pages 1315–1330, Santa Clara, CA, Apr. 2024. USENIX Association.
[15] A. Masaracchia, V. Sharma, M. Fahim, O. A. Dobre, and T. Q. Duong. Digital Twin for Open RAN: Toward Intelligent and Resilient 6G Radio Access Networks. *IEEE Communications Magazine*, 61(11):112–118, 2023.
[16] H. X. Nguyen, R. Trestian, D. To, and M. Tatipamula. Digital Twin for 5G and Beyond. *IEEE Communications Magazine*, 59(2):10–15, 2021.
[17] O-RAN next Generation Research Group. Research Report on Digital Twin RAN Use Cases. Technical report, O-RAN, May 2024.
[18] M. Polese et al. Colosseum: The Open RAN Digital Twin. *arXiv preprint arXiv:2404.17317*, 2024.
[19] B. Ryu, R. Knopp, M. Elkadi, D. Kim, and A. Le. 5G-EMANE: Scalable Open-Source Real-Time 5G New Radio Network Emulator with EMANE. In *Proc. IEEE MILCOM 2022*, pages 553–558. IEEE, 2022.