RANBooster: Democratizing advanced cellular connectivity through fronthaul middleboxes

Xenofon Foukas
xefouk@microsoft.com
Microsoft

Tenzin Samten Ukyab
ukyab@berkeley.edu
UC Berkeley

Bozidar Radunovic
bozidar@microsoft.com
Microsoft

Sylvia Ratnasamy
sylvia@eecs.berkeley.edu
UC Berkeley

Scott Shenker
shenker@icsi.berkeley.edu
ICSI & UC Berkeley

Abstract

The 5G Radio Access Network has shifted towards virtualization and disaggregation. This change aims to reduce costs and foster innovation by promoting vendor interoperability and by expanding the ecosystem. In this environment, smaller RAN vendors and open-source projects have emerged, focusing on low-cost, modular stacks. However, challenges such as achieving state-of-the-art performance and accessing data and control knobs hinder their widespread adoption. To address these issues, we propose a middlebox architecture, called RANBooster, that enhances the RAN capabilities without modifying existing network functions, by leveraging the open fronthaul interface. To demonstrate the benefits of the RANBooster framework, we build four reference applications (distributed antenna system, distributed MIMO, RU sharing, real-time physical resource block monitoring), and evaluate them on an enterprise-scale, commercial-grade 5G testbed.

CCS Concepts

• **Networks → Middle boxes / network appliances**; **Wireless access points, base stations and infrastructure**; **Programmable networks**; **In-network processing**; **Mobile networks**.

Keywords

Open RAN, fronthaul, RAN middleboxes, RAN programmability, distributed MIMO, DAS, RU sharing, RAN telemetry

ACM Reference Format:
Xenofon Foukas, Tenzin Samten Ukyab, Bozidar Radunovic, Sylvia Ratnasamy, and Scott Shenker. 2025. RANBooster: Democratizing advanced cellular connectivity through fronthaul middleboxes. In *ACM SIGCOMM 2025 Conference (SIGCOMM '25), September 8–11, 2025, Coimbra, Portugal.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3718958.3754349

1 Introduction

The 5G Radio Access Network (RAN) has undergone a paradigm shift in the recent past, characterized by the virtualization and disaggregation of the RAN network functions and their communication via open interfaces, leading to an Open RAN design [50]. This led to the emergence of several smaller RAN vendors and open source projects, which focus on developing low-cost "commoditized" network components, driven by the vision of becoming part of the emerging modular and interoperable Open RAN ecosystem both in the telco and enterprise space. Some examples include vendors of Radio Units – or RUs – (e.g., Fujitsu, Mavenir, Foxconn, Benetel), Distributed and Centralized Units – or DUs and CUs – (e.g., Radisys, CapGemini, srsRAN, OpenAirInterface), as well as RAN Intelligent controllers (RICs) and optimization applications – or xApps and rApps – (e.g., HPE-Juniper and VMware).

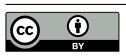
Despite the initial optimism, innovating in this space has turned out to be challenging for small players for two reasons:

1. Difficult to achieve state-of-art performance – Enterprises and telco networks require advanced features that improve network performance (e.g., interference mitigation, RU sharing, coverage/capacity expansion, massive MIMO). Such features are typically integrated in the solutions of major telco vendors (e.g., as part of their DU/RU product offering or as standalone sophisticated dedicated hardware boxes). On the other hand, smaller vendors typically lack the resources for implementing such features, leading to suboptimal performance and thus reducing their competitiveness.

2. Lack of access to data and control knobs – The top tier vendors, who dominate the market, only partially adhere to open standards and interfaces (e.g., see [44]). This introduces obstacles to third-party developers, who require full access to implement advanced features. Thus, top tier vendor products remain the most innovative through access to proprietary parts of the interfaces, effectively leading to vendor lock-in.

Motivated by the above, our goal is to accelerate innovation in the Open RAN space, while maintaining backward compatibility with existing implementations of RAN network functions from a broad range of vendors. Guided by this philosophy, we propose RANBooster, a software-based middlebox architecture for the RAN fronthaul network (the network that connects the RU to the DU), that allows third parties to develop and deploy standalone RAN features or applications in the form of middleboxes, on-the-fly and without requiring any RAN network function modifications.

Our approach mirrors the evolution in the Internet ecosystem, where innovation has involved changing core protocols (e.g., IPv6, BGPsec), but more often has been through middleboxes (e.g., firewalls), which can be transparently deployed and hence have seen rapid adoption. Similarly, RANBooster middleboxes can be used to enhance the network's capabilities using commodity hardware and the network functions of any RAN vendor. This increases the competitiveness and levels the playing field for smaller players,



SIGCOMM '25, Coimbra, Portugal

ACM ISBN 979-8-4007-1524-2/2025/09
https://doi.org/10.1145/3718958.3754349

allowing them to focus on their own niche field of expertise. Ultimately, it can help commoditize the closed world of today's RAN deployments.

The architectural choice of leveraging fronthaul middleboxes stems from the following observations:

(1) The fronthaul carries control and data plane information about the connected mobile devices (UEs) and the scheduling of radio resources. It can therefore be a great vantage point for introducing new monitoring and control capabilities.

(2) The fronthaul network of virtually all modern RAN RUs and DUs follows the Open RAN specifications, meaning that a middlebox solution can be inter-operable across the network functions of all major RAN vendors [14]. The semantics of the fronthaul interface are much simpler than those of other cellular interfaces, making it much easier to verify interoperability.

(3) The Open RAN fronthaul is Ethernet-based, meaning that the introduction of new features can be reduced into tasks of forwarding and manipulating Ethernet packets. This makes their deployment and management by non-wireless experts simpler and more accessible.

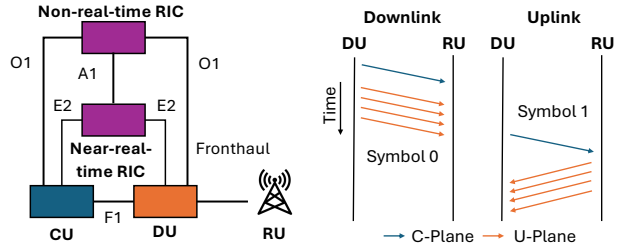
(4) The RAN softwarization means that middleboxes can be deployed and chained on-the-fly by leveraging widely used networking technologies (e.g., DPDK, XDP, SR-IOV), removing the requirement for costly hardware-based solutions.

To illustrate the power of the RANBooster design, we develop four middlebox applications: i) a distributed antenna system (DAS) for coverage expansion, ii) a distributed MIMO (dMIMO) system for capacity improvement, interference mitigation and handover-free mobility, iii) a RAN sharing system for sharing RU radio resources and enabling neutral host deployments, and iv) a monitoring system to estimate the radio resource utilization of 5G cells in real-time (sub-millisecond granularity). We demonstrate the performance and interoperability of the developed solutions by deploying them on an enterprise-scale testbed spanning several floors and RUs, and by using three different RAN stacks (srsRAN, CapGemini, Radisys), *without any source code modification*. Finally, we discuss several other applications that one could develop using RANBooster.
Disclaimer: This work does not raise any ethical issues.

2 Background

2.1 RAN architectural overview

The 5G RAN architecture is defined by standardization bodies, like the 3GPP [1] and the O-RAN Alliance [48], that group the different layers of functionality of the RAN into network functions with open interfaces between them, based on standard protocols (e.g., fronthaul, F1, E2). This is in contrast to the closed and proprietary interfaces of the past. As seen in Figure 1a, the functions include the RU, the DU, the CU, and the RIC. The RU is implemented on fixed-function hardware, whereas the DU and CU are virtualized and running on commodity servers. The DU is responsible for real-time signal processing and scheduling of radio resources, while the CU is responsible for delay tolerant functionalities, such as mobility management. The optional RIC separates the control from the data plane, allowing third-party developers to implement applications that optimize the RAN's end-to-end network performance.



(a) RAN functions and open interfaces. **(b) RAN fronthaul protocol message exchanges.**

Figure 1: 5G RAN architecture and fronthaul protocol.

2.2 Fronthaul protocol

This work focuses on the standardized O-RAN fronthaul interface [48]. The fronthaul is Ethernet-based and transports the modulated higher layer data (e.g., user packets, control signals) between the DU and the RU. It defines an S-plane for synchronization messages, an M-plane for management, a C-plane for control instructions and a U-plane for transferring the radio signals. In this work, we focus on the C-plane and the U-plane.

RU resource scheduling – The DU uses the C-plane messages to instruct the RU on how to schedule its radio resources (U-plane). The radio resources span the time, the frequency and the spatial dimension. In the time dimension, the scheduling takes place in time increments called *symbols*, with a duration of a few tens of microseconds (e.g., 33.3μs for a typical cell configuration). In the frequency dimension, the minimum resource that can be scheduled is a physical resource block (PRB), which includes 12 orthogonal equally spaced sub-carriers (e.g., 30kHz). Finally, in the spatial dimension, resource allocations take place in the granularity of logical RU antenna ports, which correspond to distinct data streams sent in parallel to and from the RU (e.g., for MIMO transmissions).

Fronthaul message sequences – A typical exchange of fronthaul messages between the DU and the RU is illustrated in Figure 1b. The DU sends a C-plane message to the RU containing scheduling instructions for one or more upcoming symbols, followed by a series of U-plane messages carrying the actual data. The U-plane messages originate from the DU in the case of downlink, and from the RU in the case of uplink traffic. All the messages must be sent within a strict time window, otherwise the packets will be dropped and ignored by the DU and the RU. The typical way to guarantee such deadlines is to use strict nanosecond-level synchronization protocols, like PTP and SyncE, across the DU and the RU.

Message structure – Both the C-plane and U-plane messages contain Ethernet headers indicating their source and destination (i.e.,

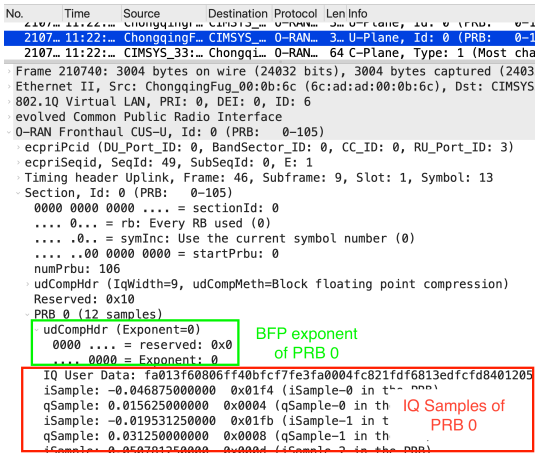


Figure 2: Wireshark capture showing compressed IQ samples in U-plane packet for one PRB.

DU and RU). They also encapsulate O-RAN headers, with fields describing the resources that are being scheduled (i.e., symbols, PRBs, antenna port ids). U-plane packets also carry a payload, which contains the waveforms with the modulated higher layer data (user packets, UE signaling messages). The payload is encoded in the form of complex numbers called *IQ samples* (*I* being the real part and *Q* the imaginary), with each IQ sample corresponding to one subcarrier of the frequency grid. As such, 12 consecutive IQ samples constitute a single PRB, as illustrated in the Wireshark capture of a real U-plane packet in Figure 2.

Payload compression – Each IQ sample is 32-bits in size, resulting in jumbo Ethernet U-plane frames, when employing high bandwidth cell configurations (e.g., 100Mhz with 273 PRBs). To reduce the high fornthaul bandwidth overhead introduced by the jumbo frames, the U-plane payload is typically compressed and is accompanied by compression headers that carry the applied compression parameters. A typical compression scheme employed by many implementations (including the ones we studied) is called Block Floating Point (BFP). It compresses the IQ samples at the PRB granularity and appends a header with a compression exponent to each PRB, as illustrated in Figure 2.

3 RANBooster middlebox

Motivated by the challenges outlined in the outset, we argue that the fronthaul interface is an ideal point to introduce innovations through a middlebox-based architecture.

3.1 The case for a middlebox approach

We start with the first key observation that the protocol state complexity in the RAN increases as we move up the stack. As outlined in Section 2.2, the fronthaul protocol has very little state, making it easier to achieve interoperability across vendors. It is also the primary focus of the interoperability efforts today, as it would allow for the independent evolution of the software stack (CU/DU) without requiring matching hardware updates at the RU side. Therefore,

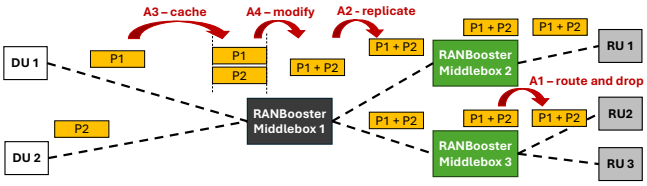


Figure 3: Overview of a RANBooster deployment. The middlebox colors represent different applications. P1-P2 are packets and A1-A4 are actions applied on them.

due to its limited state, clean and standardized abstractions, and Ethernet-based nature, the fronthaul allows the insertion of middleboxes and service chaining without requiring any modifications across the rest of the stack.

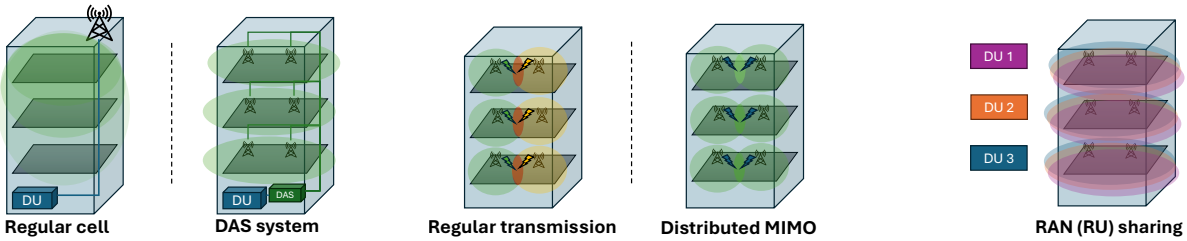
The second key observation is that one can innovate by gaining access to the fronthaul. A common perception is that modifying the fronthaul interface provides limited value because (i) it provides low level data abstractions (modulated IQ samples), and (ii) it is characterized by high bandwidth and low latency traffic, which is computationally expensive to process. We show that both of these are untrue. Firstly, we show that a number of existing high-value RAN features (such as DAS and distributed MIMO), as well as new and useful applications, can be implemented as fronthaul middleboxes. Secondly, we show that these can be implemented efficiently, using existing networking software and hardware acceleration stacks.

3.2 RANBooster middlebox architecture

We begin with a high-level overview of the architecture and capabilities of a typical RANBooster middlebox. As illustrated in Figure 3, a RANBooster middlebox intercepts the C- and U-plane traffic between one or more DUs and RUs (N to M mapping). Each fronthaul packet can be passively inspected before forwarding (e.g., to extract information for monitoring applications), or can be modified inline or at a later point (e.g., for enabling control applications). RANBooster middleboxes can be chained together, as illustrated in Figure 3, to create more advanced connectivity services on-the-fly. Finally, RANBooster middleboxes expose monitoring and management interfaces to modify their behavior on-the-fly (e.g., apply forwarding rules) and to send telemetry data to applications.

3.2.1 RANBooster actions. RANBooster supports the following processing actions, which are illustrated in Figure 3:

- **Packet redirection and drop (A1) –** This refers to the dropping and steering of packets to a different DU or RU by modifying Ethernet source and destination addresses, VLAN ids, etc.
- **Packet replication (A2) –** This refers to the cloning of an incoming packet, to send it to several destinations (e.g., RUs).
- **Packet caching (A3) –** This refers to the storing of a packet for later use (e.g., to combine its data with that of other incoming packets that arrive later or from different sources).



(a) Signal distribution from a regular cell (left) vs a DAS system (right).

(b) Regular multi-cell transmission (left) vs distributed MIMO (right).

(c) RAN sharing, with several DUs sharing the RUs to transmit their signals.

Figure 4: Overview of advanced cellular connectivity configurations that can be enabled using RANBooster middleboxes.

- **Payload inspection and modification (A4) –** This refers to the access and modification of the fronthaul payload, including fields of the O-RAN headers (e.g., number of PRBs, antenna port ids), as well as raw IQ samples of the U-plane packets.

We argue that the aforementioned actions are expressive enough to create middleboxes that can realize a plethora of innovative applications when appropriately combined, as we later demonstrate through the concrete examples of Section 4 and the additional use cases outlined in the discussion of Section 8.1.

3.2.2 RANBooster middlebox template. Instead of requiring developers to write middleboxes from scratch, RANBooster introduces a templated middlebox design. Developers can use a library provided by RANBooster to initialize a new middlebox and to process C- and U-plane packets through a handler function with user-provided code, by leveraging the actions of Section 3.2.1. Each of the actions has been distilled into a set of API calls that simplify the development of new middleboxes. To demonstrate the extendibility of this approach, we used the same template of the RANBooster library to create all the middleboxes discussed in Section 4. We believe that the same templated design can be used for any other application leveraging the fronthaul interface (e.g., the use cases discussed in Section 8.1).

4 RANBooster applications

Here we provide a detailed description of several concrete applications that we have built using RANBooster. The choice of the applications was driven by i) their importance for many practical scenarios of interest, like in the enterprise/private 5G context (e.g., see case study in Section 7), and ii) the high cost of existing proprietary solutions and stacks (e.g., see cost analysis in Appendix A.2).

4.1 Distributed Antenna System

Motivation: A DAS is a network of antennas placed across large areas, such as buildings, stadiums and airports. As illustrated in Figure 4a, DAS allows the distribution of radio signals (e.g., cellular or WiFi) in a uniform manner, compared to a single more powerful antenna. This leads to high-quality coverage in areas where natural signal strength is weak due to physical obstructions or high user density. DAS solutions are a standard practice across the industry.

DAS systems are composed of a signal source (e.g., RAN macro or small cell) and a signal distribution system (distribution network and antennas). The distribution system configuration can greatly vary in terms of complexity and cost. On one end are passive DAS solutions that utilize passive RF components like coaxial cables, splitters and couplers to distribute the signal in a simpler and less costly manner. On the other end are more advanced digital DAS solutions, where RF signals are digitized before being combined and transmitted over fiber optic or Ethernet cables, making them less susceptible to interference, at the expense of a higher cost. Most importantly, existing DAS solutions are proprietary end-to-end, which significantly contributes to their very high cost, that can be in the order of several hundred thousand dollars, even for relatively small deployments [31, 57].

Implementation: We leverage the observation that the fronthaul carries digitized radio signals, allowing us to build a cost-efficient digital DAS using a RANBooster middlebox and commodity RUs. The functionality of the middlebox is illustrated in Figure 5a, where the signal from a single cell (DU) is distributed across two RUs. In the downlink, the middlebox forwards the C- and U-plane packets from the DU, to all the DAS RUs, using the packet replication and forwarding actions A1 and A2. In the uplink, it combines the signals of all the RUs into a single signal that is fed back to the DU. It first leverages action A3, to cache the incoming U-plane packets of all the RUs for a given symbol and antenna port. Once all the RU packets are received, the middlebox creates a single U-plane packet, by summing up their IQ samples in an element-wise manner (i.e., on a sub-carrier basis), leveraging the payload modification action A4. If the IQ samples are compressed, the A4 action is also used prior and after summing them, to decompress and re-compress them. In the last step, the middlebox leverages action A1, to send the packet combining the merged IQ samples to the destination DU, while dropping the rest.

Summing the signals of different RUs suggests that interference may be introduced in the aggregated uplink signal fed to the DU. However, the DAS system leverages a single radio resource scheduler to allocate non-overlapping frequency domain resources (PRBs) to all the UEs across the DAS coverage area. Therefore, the sum of the IQ samples of two RUs for a PRB is guaranteed to carry the data of at most one UE per MIMO layer. This creates a clean signal with interference free coverage. Conversely, in multi-cell deployments neighboring cells make independent radio resource scheduling decisions, leading to interference and thus degraded performance, as we show in Section 6.3.

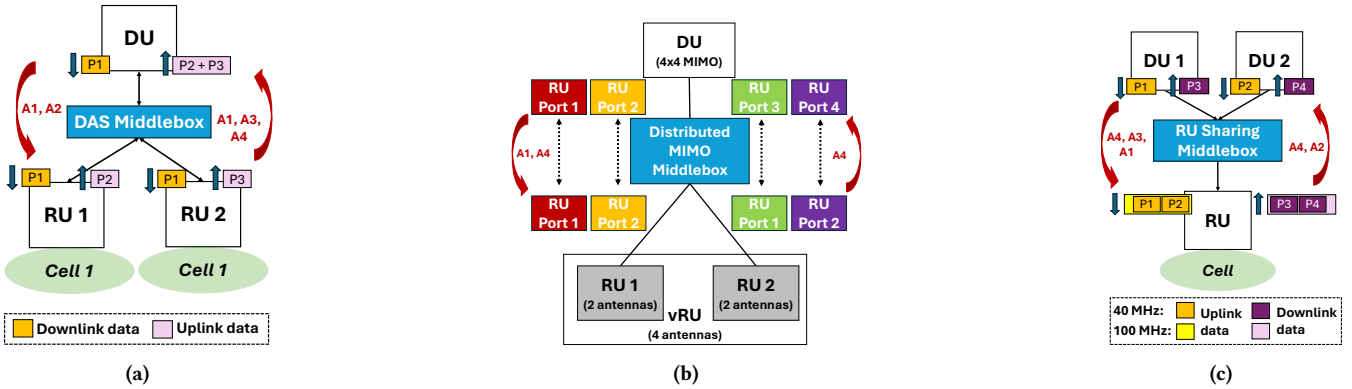


Figure 5: High level overview of (a) DAS, (b) dMIMO and (c) RAN sharing RANBooster middleboxes. The applications have been explicitly mapped to the actions of Section 3.2 that each middlebox has to perform (marked with red fonts).

4.2 Distributed MIMO

Motivation: MIMO radios use an array of antennas to transmit multiple data streams simultaneously over the same frequency. They exploit multipath propagation (different paths taken by the signal of each antenna), to introduce constructive interference that amplifies the transmitted signal, leading to increased spectral efficiency. dMIMO takes this concept to the next level, by allowing the antennas of a single cell to be distributed. As illustrated in Figure 4b, in contrast to DAS, where the same signal is replicated across all the antennas, and to multi-cell deployments, where each antenna belongs to a different cell, leading to cross-talk, dMIMO expands both the range and the capacity of a single cell. This leads to higher throughput compared to DAS, without the need to develop sophisticated mobility or interference management techniques. This is particularly beneficial in places where both range and increased capacity are important (e.g., factories, stadiums, etc.).

Like DAS, dMIMO requires a high capacity distribution network, but also presents a requirement for very strict time and phase synchronization of the radios (in the order of a few nanoseconds [12, 66]). To deal with this, several dMIMO solutions have been proposed in the recent past, both in industry and academia (e.g., [13, 26, 27]). While promising, such solutions require specialized software and/or hardware (e.g., custom RAN software stacks [26, 27] or proprietary RU designs and interfaces [13]), which limit their applicability in practical deployment scenarios or lead to vendor lock-ins.

Implementation: We argue that, while the problem of achieving dMIMO at city-scale is still open, one can already build practical dMIMO systems at smaller scales (e.g., enterprise-scale), and in a cost-efficient manner, by leveraging RANBooster and commodity RAN stacks and RUs. To this end, we leverage RANBooster, to build a dMIMO middlebox. As illustrated in Figure 5b, the middlebox combines several commodity RUs, each equipped with a small number of antennas, to create a larger virtual RU. A DU can be interfaced with the virtual RU via the RANBooster middlebox, to achieve higher layer MIMO than each individual physical RU, while also expanding the cell range. For example, one could leverage two O-RAN RUs with two antennas each, to create a single four antenna virtual RU that can be interfaced with a DU to achieve 4×4 MIMO.

More generally, for N MIMO layers, the RANBooster middlebox must give the DU the illusion that only a single RU with N antennas exists, while each physical M-antennas RU must be given the illusion that it communicates with a DU supporting M antennas. To achieve this, for each fronthaul packet, the middlebox remaps their antenna port ids, by modifying their O-RAN header (action A4). In the downlink, it also redirects the packet to the correct RU (action A1). As a concrete example, we consider the setup of Figure 5b. The middlebox sends all the packets of the DU with antenna ports 1 and 2 to RU 1 unmodified, since the RU expects packets for these ports. However, in the case of packets for antenna ports 3 and 4, the middlebox re-maps them to ports 1 and 2, before sending them to RU 2, since the RU does not expect any packets for port 3 and 4. The opposite is done in the uplink.

One problem of this approach is related to the transmission of a periodic broadcast message, called the synchronization signal block (SSB). The SSB is a critical message that contains the physical cell id and its reference transmission power, which are required by the UEs for cell synchronization and monitoring of the radio link quality. Typically, the SSB is only transmitted by the first DU antenna, which can negatively affect our distributed design. This is because, once a UE moves far from the primary antenna, and even though it might still be within the range of other antennas, it stops receiving the SSB message, leading to issues like detachments or to sub-optimal handovers. To overcome this issue, the middlebox uses the payload modification action A4, to copy the SSB received by the U-plane packets of the primary antenna, to those of other antennas. This is easily done, since the SSB is transmitted periodically in well known symbols and PRBs of the cell.

We note that for the proposed middlebox to work, we make the following assumptions about the RAN deployment:

- *Cat-A O-RAN RUs* – We assume that the employed O-RAN RUs are of type Cat-A, meaning that all the MIMO-related operations (e.g., channel estimation, precoding, etc.) take place at the DU. While this increases the computational complexity at the DU side and the fronthaul throughput requirements, it also leads to a simpler and cheaper RU design.

- *Time and phase synchronization* – Distributed MIMO requires tight time and phase synchronization across the distributed radios and the DU. Such a requirement can be guaranteed in any RANBooster deployment, given that all the open fronthaul compliant RUs and DUs have to be synchronized using some high accuracy synchronization method, like GNSS and PTP [51].

4.3 RU sharing

Motivation: RAN sharing is the practice of sharing the infrastructure and spectrum for cost reductions or due to space constraints [25, 34]. Several sharing models exist, including models where an operator owns the spectrum or the infrastructure (or both) and shares it with competitors, as well as models where a third-party (neutral host), is the owner [16]. Given the importance of sharing, 3GPP and O-RAN provide specifications on how it can be enabled at several levels, including the DU [49] and the RU [51].

Here, we focus on the capability of RU sharing, i.e., allowing a single RU to be shared by many DUs, as illustrated in Figure 4c. Despite the standardization of RU sharing, it is typically a feature that is implemented as a premium in the RUs of major telco vendors, and is missing from commodity RUs. Therefore, if a RAN vendor wanted to build a simple solution with sharing capabilities (e.g., for a neutral host deployment), they would be unable to do so without relying on telco-grade RUs, which would be difficult to obtain and would greatly increase the overall cost of the solution.

Implementation: We argue that RU sharing can be implemented in a cost-efficient way using a RANBooster middlebox and commodity RUs. Figure 5c, illustrates the high level idea of how this is achieved. In the downlink, RANBooster multiplexes the packets from two DUs of different operators into one to be sent to the RU, while for the uplink, it demultiplexes the packets. This is done so that the DUs operate as though they are in control of their own RU, while the RU believes it is being controlled by one DU.

Sharing the RU's spectrum – The shared RU is configured for a specific center frequency and bandwidth. The challenge is dividing the RU's spectrum between the DUs. When U-plane packets carrying PRBs of each DU are multiplexed in the downlink, the PRBs must be copied to the correct location in the final U-plane packet, for the RU to interpret the correct IQ sample at the correct frequency. For uplink U-plane packets, the reverse operation is performed. If the PRBs of the DU align with those of the RU (left part of Figure 6), copying them to the correct location is sufficient. However, if they are misaligned (right part of Figure 6), the middlebox must de-compress, copy, and re-compress them. This is because IQ samples are typically compressed as a PRB using a common compression parameter (see Section 2.2), which may differ between PRBs. Misalignment can cause decompression issues due to mismatched compression parameters. Therefore, as an optimization, one could pick the center frequency of the DUs, so that the PRBs of the DUs and the RUs are aligned. We provide a formula on how to ensure this alignment in Appendix A.1.1.

Ensuring consistent resource requests – The middlebox must ensure that for each symbol, the RU will receive a C-plane message that contains instructions about processing the PRBs sent or received by all active DUs. Ideally, the middlebox would achieve this

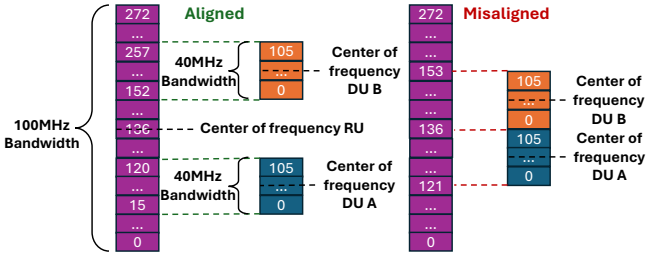


Figure 6: Example of sharing a 100MHz RU between two 40MHz DUs, with PRBs aligned (left) or misaligned (right).

by first caching all the C-plane packets from all the DUs for a particular symbol and antenna port and then combining their requests to a single C-plane message towards the RU. In practice, this is challenging to do, since the middlebox does not know a priori whether all the DUs will send a C-plane message (e.g., due to lack of traffic).

To overcome this issue, the middlebox takes the following approach in both the downlink and uplink: When the first C-plane message arrives from any DU for a given symbol and antenna port, we use the packet modification action A4, to modify the O-RAN header field that encodes the number of requested PRBs, to designate the max number of PRBs supported by the RU. This makes the RU process the whole spectrum, ensuring that any follow-up C-plane request from any DU for the same symbol and antenna port will already be satisfied (at the expense of extra bandwidth for transporting all the PRBs). Furthermore, the middlebox caches all the incoming C-plane packets (from any DU) for the given symbol and antenna port. This allows the middlebox to remember which DUs have made a request, so that it can do the mapping of the resources, once the corresponding U-plane packets arrive.

Dealing with the U-plane packets – Using the above approach, the processing of the U-plane traffic becomes straightforward. In the downlink, the middlebox uses action A3 to cache any incoming U-plane packet, until all the packets from all the DUs that have made a C-plane request on a given symbol and antenna port have arrived. The middlebox then uses the payload modification action A4 on one of the U-plane packets, to copy to it all the PRBs (from all the DUs) that should be sent to the RU. It also uses the packet drop action A1, to discard the remaining U-plane packets and the cached C-plane packets. In the uplink, once the middlebox receives a U-plane packet from the RU, it uses the packet replication action A2, to create one copy for each DU that made a request (i.e., for each cached C-plane message). It then uses the payload modification action A4, to copy to each replica the correct PRBs expected by the corresponding DU.

4.4 Real-time PRB monitoring

Motivation: Real-time monitoring of RAN KPIs (e.g., PRB utilization, signal qualities, etc.) is becoming increasingly important for a plethora of services, including congestion control [23, 68], video bitrate adaptation [17, 71], energy savings [42], interference detection [18], etc. To this end, O-RAN has introduced the E2 interface that defines and exposes such KPIs to applications via the RIC. Despite its potential, the E2 interface has not yet been met with

Algorithm 1 PRB Utilization Estimation Algorithm

Require: Fronthaul packets containing PRB data (Fig 2)
Ensure: Bitvector `PRB_Utilized` populated with utilization status

```
Initialize PRB_Utilized as an empty bitvector
for each PRB in Fronthaul_samples do
    Extract Exponent from PRB's BFP header (Fig 2)
    if Packet is Downlink then
        PRB_Utilized[PRB] := (Exponent > thr_dl)
    else if Packet is Uplink then
        PRB_Utilized[PRB] := (Exponent > thr_ul)
    end if
end for
Expose PRB_Utilized to external applications through telemetry interface
```

widespread success in commercial deployments, with major vendors being reluctant to adopt it (e.g., see [44]), opting to expose few and coarse-grained KPIs instead (in the order of several minutes or hours) [18], which can be very limiting for applications like the ones mentioned above.

A typical approach to overcome this problem is the use of sniffers, which leverage software defined radios, to capture and analyze the IQ samples broadcasted by the base stations in order to extract useful real-time information, like the PRB utilization, the radio resource scheduling of users, etc. While this approach has been used for network optimization and performance analysis [5, 15, 28, 43, 67]), it has a major limitation, in that it can be costly and does not scale to larger networks, as you need to deploy one hardware receiver within the area of each cell that you wish to monitor.

In contrast to the over-the-air sniffer approach, the fronthaul network becomes an ideal vantage point for deploying software-based real-time monitoring services. This is because the fronthaul exposes the same data as that captured by over-the-air sniffers, but does not require sniffing hardware. Motivated by this, we develop a RANBooster middlebox, that acts as a simple real-time monitoring service. The service monitors the PRB utilization of a cell, which is an important KPI for several applications, like for example in the context of energy savings and load balancing [30, 42, 46], and exposes it to the applications via the telemetry interface.

Implementation: One could use several approaches to calculate the PRB utilization from the fronthaul. One approach could be to feed the captured IQ samples to a 5G sniffer [43, 67], to obtain the higher layer signaling messages (radio resource scheduling decisions) that indicate the utilized PRBs. Another approach could be to identify all the PRBs carrying IQ samples that are above some energy level threshold (e.g., like in [18]), indicating the presence of user data. However, such approaches incur additional overheads, e.g., for de-compressing and decoding the IQ samples carrying the scheduling decisions.

To avoid such overheads, we opted for a lightweight approach, in which we estimate the PRB utilization by leveraging the compression information carried in the uplink and downlink U-plane packets. The intuition is that a high level of compression indicates that the PRB carries little useful information, and as such, it can be considered as unutilized. Based on this observation, we develop an estimation algorithm (Algorithm 1) that leverages the BFP exponent (the compression method used by all the RAN implementations that we studied) to estimate the utilization without decompressing the samples. A PRB is simply marked as utilized if its BFP exponent is above some threshold (0 in the downlink and 2 in the uplink in the setups we tried), otherwise it is marked as idle.

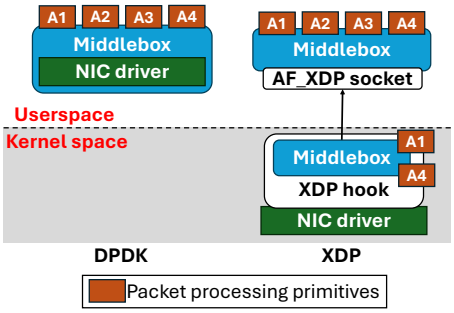


Figure 7: Implementation outline of RANBooster with DPDK (left) and XDP (right).

5 RANBooster implementation

All middleboxes of Section 4 were implemented in approximately 10K lines of C/C++ code. and can be used as a reference for realizing other middleboxes in the future.

Network technology options: There are two popular network packet processing technologies we considered; DPDK [20] and XDP, with different pros and cons. DPDK bypasses the kernel and provides optimal throughput and latency performance. Its downsides include unstable APIs, separate and harder to manage userspace NIC drivers and tools, and the need for dedicated CPU cores for its polling-based operation.

XDP relies on the Linux kernel and it offers stable APIs, making it easier to use. It is also interrupt-driven, meaning that it does not need to utilize a full CPU core when the traffic is low. A downside of XDP is that it introduces additional performance and latency overheads due to the involvement of the kernel stack in the processing of the packets [65]. It also introduces memory-related overheads when handling jumbo Ethernet frames [45], frequently used for fronthaul communication (e.g., 100MHz cells generate packets > 7KB).

Since both technologies can be relevant, and since the performance gap between DPDK and XDP is closing (e.g. [65, 73]), we implemented and evaluated RANBooster with both.

Implementation of middlebox actions: A typical implementation of a RANBooster middlebox is illustrated in Figure 7. In the case of DPDK, middleboxes are monolithic blocks that leverage the packet processing actions described in Section 3.2 to implement all their logic in userspace. In contrast, XDP middleboxes are composed of a kernel XDP program loaded at the NIC driver hook and (optionally) a userspace component. C- and U-plane packets go through the kernel XDP program, which can be used to efficiently implement redirection and drop operations (action A1), as well as simple payload inspection and modification operations (action A4). Other actions, that require more advanced packet processing (i.e., caching, replication and modifications of IQ samples) are inefficient when using eBPF [65]. Therefore, logic requiring such actions can leverage a simple forwarding XDP program that sends them to the

	DAS	dMIMO	RU sharing	PRB monitoring
Kernel space	-	✓	-	✓
Userspace	✓	-	✓	-

Table 1: Location of packet processing for the applications of Section 4 in the XDP implementation.

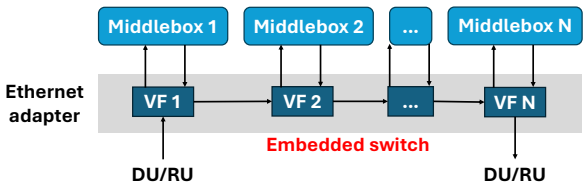


Figure 8: Chaining of RANBooster middleboxes.

userspace component of the middlebox for further processing, using a type of socket called AF_XDP [39] (at the expense of context switching overhead). It is left to the developer to choose the right location for packet processing, depending on the middlebox logic. As a reference, Table 1 lists the split of packet processing between userspace and kernel space for the middleboxes of Section 4.

Middlebox chaining: RANBooster leverages SR-IOV to chain middleboxes, as illustrated in Figure 8. SR-IOV is a technology that allows a physical NIC to be split into multiple virtual functions (VFs), each of which can be assigned directly to a virtual machine or container with minimal overhead. Each middlebox is configured to use one VF of the NIC for receiving and forwarding fronthaul packets, through an embedded NIC switch. Typical SR-IOV enabled NICs (e.g., Mellanox and Intel) allow the creation of several tens of VFs per NIC port, meaning that several RANBooster middleboxes can be deployed on the same NIC port. The total number of middleboxes that can be chained using this approach is constrained by two factors; i) the PCIe throughput for forwarding traffic between the chained middleboxes, and ii) the total latency added to the DU packet processing time, which needs to be below some threshold (typically in the order of a few tens of microseconds). It should be noted that while SR-IOV is a low-latency and easy method for chaining middleboxes on the same host, it is not the most scalable one, with the PCIe bus becoming the bottleneck [35]. As already shown in the literature, one could leverage alternative approaches, like software-based accelerated switches (e.g., OVS [54]), to increase the scale at the expense of more compute. We leave the investigation of such approaches and their trade-offs as future work.

6 Evaluation

6.1 Testbed setup

For our evaluation, we leverage an enterprise-scale 5G testbed that we have deployed in our premises, in Microsoft Research Cambridge, UK [3]. The testbed spans five 50.9m × 20.9m floors (see Figure 9a), each equipped with four O-RAN compliant Foxconn RPQN-7800 RUs (Figure 9b), supporting up to 100MHz 4×4 MIMO in 5G band 78 (3.3-3.8GHz). In addition, the testbed also offers another four RUs that are not ceiling-mounted, which we use for experiments where we need flexible RU placement.

The RUs are connected to a rack of HPE Telco DL110 servers through a 100GbE Arista 7050 switch (Figure 9c). The servers are Linux-based (real-time kernel v6.6.44) and they feature Intel Xeon 6338N CPUs and a Mellanox ConnectX-6 Dx NIC. In terms of the RAN, we use three O-RAN compliant stacks; two closed source commercial-grade RAN stacks (Intel FlexRAN L1 with CapGemini and Radisys L2/L3), and the open source srsRAN. All stacks support up to 4 layers downlink MIMO. For the mobile core, we use the open source Open5GS. To achieve time synchronization between the RUs and the DU, we use a Qulsar QG2 PTP grandmaster clock (GM). For the UEs, we leverage five smartphones (OnePlus Nord N10 5G, Samsung Galaxy S22 and A52s) and fifteen Raspberry Pis equipped with Quectel RM502Q-AE modems (Figure 9d), spread across the five floors of our testbed. Unless stated otherwise, we use the DPDK implementations of the middleboxes.

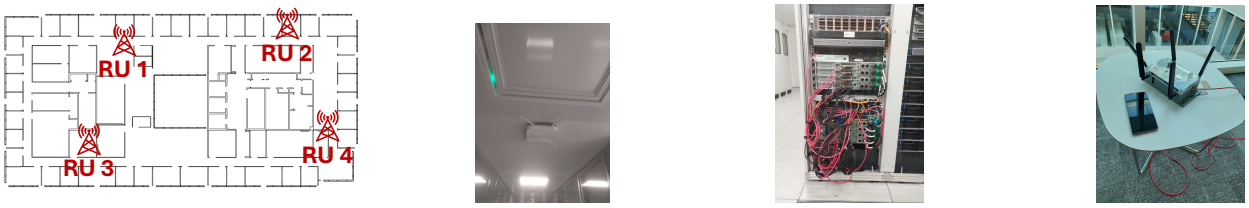
6.2 Correctness of middleboxes

Here, we evaluate the correctness of the middleboxes described in Section 4. To demonstrate the portability and inter-operability of our approach, we have verified our results with all three RAN stacks described in Section 6.1 (i.e., srsRAN, CapGemini, Radisys). In all cases, we reused the same RANBooster middlebox implementations, without any source code modifications, and with only small configuration parameter changes (e.g., TDD pattern). Due to the lack of space, the rest of this section presents only the srsRAN results with the DPDK implementation. However, the findings of our experiments apply to all stacks, with only differences in terms of the obtained throughput, caused by the variations in the implementation quality and cell configurations provided by each vendor.

6.2.1 DAS. We verify that the cell signal of the RANBooster middlebox increases the coverage of our network, without affecting performance. As a baseline, we deploy a 100MHz 4×4 MIMO cell on the ground floor of our building and attach two UEs that are in close proximity to the RU (less than 10 meters). We perform an iperf UDP test and measure the aggregate throughput achieved by the UEs, which is illustrated in Figure 10a. Furthermore, we try to attach other UEs that are located on the upper floors of our building and observe that they are unable to do so, due to weak signal.

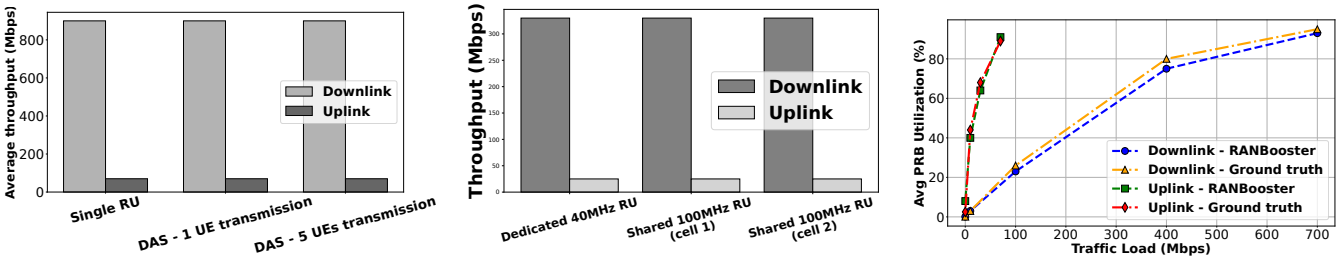
Next, we deploy the RANBooster DAS middlebox, which replicates the cell's signal across all five floors (i.e., on one RU per floor). We place one UE on each floor, in close proximity to the corresponding RU. With this configuration we observe that all the UEs (across all floors) are able to attach, which demonstrates that the signal of the cell is indeed extended, due to the RANBooster middlebox. Furthermore, we use the attached UEs to perform iperf UDP tests, similar to the single RU case. Specifically, we perform two types of tests; i) all the UEs across all floors run iperf simultaneously, and ii) each UE (one per floor) runs iperf individually, while the other UEs are attached, but idle. As shown in Figure 10a, the achieved aggregate downlink and uplink throughput is the same in all cases, and also matches the baseline, indicating that the DAS middlebox does not affect the network's ideal performance.

6.2.2 dMIMO. To evaluate the correctness of the dMIMO middlebox, and given the limitations of our RAN stacks, we focus on up to 4 layers downlink MIMO. As a baseline, we use a 100MHz cell



(a) Floorplan with RU placement. (b) Ceiling-mounted RU. (c) 5G server rack. (d) UEs.

Figure 9: Enterprise-scale testbed overview with five floors and twenty RUs.



(a) Downlink and uplink throughput of single cell with 1 RU (single floor) vs RANBooster DAS with 5 RUs (five floors). (b) Downlink and uplink throughput of 40MHz cells with dedicated 40MHz RU vs shared 100MHz RU. (c) Average PRB utilization per second estimated by the RANBooster middlebox vs ground truth for various levels of traffic.

Figure 10: Correctness results for DAS, RU sharing and the real-time PRB monitoring middleboxes.

associated with a single RU and measure the average downlink iperf UDP throughput of a UE in close range (~5 meters) for 2 and 4 layers MIMO. The results are listed in rows one and three of Table 2. We also verify that the UE rank indicator, which is a KPI indicating the degree of parallelism in MIMO transmissions, is 2 and 4 correspondingly.

Next, we place two RUs approximately 5 meters apart and we use one or two antennas of each, to enable 2 and 4 layers distributed MIMO correspondingly. As we can see from the second and fourth rows of Table 2, the throughput in the case of both the distributed MIMO configurations is the same as that of the baselines. The same applies for the UE rank indicators, demonstrating the correctness of the distributed MIMO middlebox. We also note that in all cases the achieved uplink (SISO) throughput is the expected (70Mbps).

6.2.3 RU sharing. For the correctness of RU sharing, we use as a baseline a 40MHz cell associated with a 40MHz RU and we measure the iperf UDP throughput of an attached UE. As illustrated in Figure 10b, the UE can achieve approximately 330Mbps and 25Mbps on the downlink and the uplink correspondingly. Next, we use a 100MHz RU configured to a central frequency of 3.46GHz and we deploy two 40MHz cells that share the RU, with central frequencies of 3.43GHz and 4.48GHz. We then force the association of one UE to each cell based on the physical cell id and we repeat the iperf UDP throughput experiments. As illustrated in Figure 10b, the throughput of the cells sharing the RU is identical to the throughput of the baseline in both the uplink and the downlink.

Configuration		Downlink throughput	UE rank indicator
2 layers MIMO	Single RU - 2 antennas	653.4Mbps	2
	Two RUs - 1 antenna each (RANBooster)	654.1Mbps	2
4 layers MIMO	Single RU - 4 antennas	898.2Mbps	4
	Two RUs - 2 antennas each (RANBooster)	896.9Mbps	4

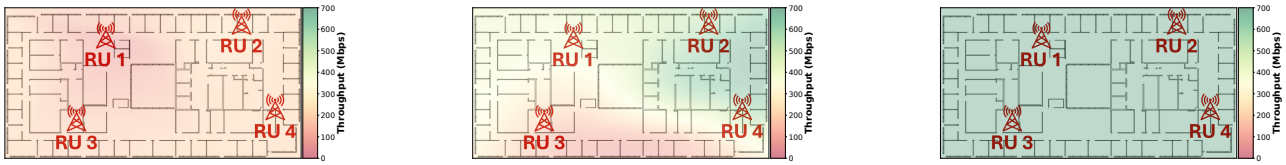
Table 2: Average downlink throughput of dMIMO vs single RU MIMO ground truth for two and four antennas.

6.2.4 Real-time PRB monitoring. To verify the correctness of the PRB monitoring middlebox we deploy a 100MHz cell and generate various levels of traffic. For each level, we record the average PRB utilization per second, as reported by the middlebox. As a baseline, we also record the MAC scheduling logs emitted by the RAN stack, which we use to calculate the actual PRB utilization. As illustrated in the results of Figure 10c the middlebox PRB utilization estimates closely match the ground truth for all levels of traffic.

6.3 RANBooster benefits

Here we provide examples of the benefits of RANBoosterin terms of performance/ease of use and flexible upgrades. We have also performed an analysis of cost-related benefits, which is available in Appendix A.2, due to lack of space.

6.3.1 Ease of use. Here we show how one could achieve high levels of performance by leveraging RANBooster, without employing sophisticated deployment configurations. We consider a use case, where we need to cover a floor, like the one presented in Figure 9a, having 100MHz of spectrum at our disposal for cell planning. We have verified that in order to provide full coverage to the floor and



(a) 25MHz multi-cell (O1 – cells on non-overlapping frequencies).

(b) 100MHz multi-cell (O2 – frequency reuse across all cells).

(c) RANBooster DAS (O3 – single 100MHz cell signal across the floor).

Figure 11: Performance of simple 4 RU RANBooster DAS deployment vs multi-cell deployment configurations.

to avoid dead spots, we need 4 RUs, in a placement like the one shown in the figure. We consider the following cell deployment options:

- **Four cells/25MHz (O1) –** We assign a 25MHz 4×4 MIMO cell to each RU, to avoid inter-cell interference across the floor.
- **Four cells/100MHz (O2) –** We assign a 100MHz 4×4 MIMO cell to each RU, re-using the same spectrum.
- **RANBooster DAS (O3) –** We deploy a single 100MHz 4×4 MIMO cell, and replicate its signal across all four RUs.

O1 and O2 are intuitive options an operator might consider in the absence of a commercial DAS solution. Using O1-O3, we evaluate the throughput that one can achieve across the floor. We deploy one UE at a fixed location near RU 1 and generate 100Mbps of downlink iperf traffic. Next, we walk around the floor with one of the UEs running a downlink throughput test of 700Mbps, and measure the achieved throughput.

The results are shown in Figure 11. For O1, the throughput of the mobile UE is limited to a max of 200Mbps, due to the limited spectrum of 25MHz. For O2 the mobile UE suffers from low throughput in several locations, due to inter-cell interference, caused by the transmissions of the static UE, when the two UEs are attached to different cells. On the other hand, we observe that the RANBooster DAS middlebox of O3, allows the UE to achieve the best performance, with approximately 700Mbps across the whole floor.

We conclude that a middlebox is a simple way to get wide coverage with high performance. The DAS middlebox is easily deployable by an IT person with basic knowledge of orchestration frameworks, like Kubernetes. For better results with a multi-cell deployment (e.g., in the spirit of O1 and O2), a more sophisticated solution would be required (i.e., interference mitigation [7, 8, 11] and mobility management [36]). This requires deep wireless knowledge and access to RAN functions (e.g., MAC scheduler) not exposed by the RAN vendors, but are add-on features in costly integrated solutions.

6.3.2 Flexible upgrades. Here, we show how RANBooster can help introduce new features, boost performance and save energy.

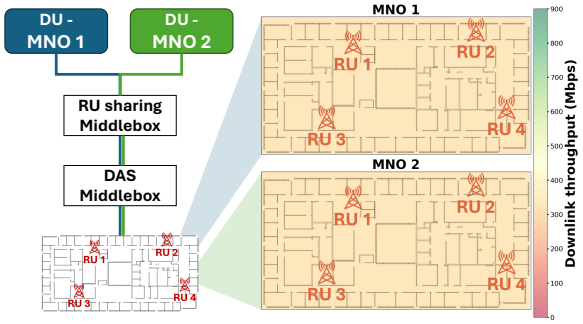


Figure 12: Chaining of RU sharing and DAS middlebox to enable multi-tenancy with seamless coverage.

Enhancing the network's capabilities – We consider the deployment scenario of Section 6.3.1, but we now introduce a second Mobile Network Operator (MNO). We deploy the RANBooster RU sharing middlebox, which we chain with the DAS middlebox. As illustrated in Figure 12, this allows us to deploy two networks over the same infrastructure, by allocating 40MHz of spectrum per MNO, while sharing the same 100MHz RUs. We attach one UE to each MNO's network and run a downlink iperf test. Each UE can achieve ~350Mbps across the floor, thanks to the seamless coverage provided by the DAS middlebox. Most notably, the RU sharing feature upgrade is achieved *without any modification of the infrastructure*, and only requires software updates (deployment of new MNO's DU/CU/core, middlebox and reconfiguration of network functions). This is in stark contrast to existing solutions, which would require new infrastructure to support the new scenario, which significantly increases the upgrade complexity and cost.

Boosting the network's performance – We consider a scenario, where one has deployed cheap 1-antenna RUs across the floor of Figure 9a. Based on the findings of Section 6.3.1, we observe that one could deploy a DAS middlebox, provided by vendor A, to guarantee uniform coverage across the whole floor, with a single SISO cell. We measure that this would allow UEs to achieve approximately 250Mbps of downlink throughput, as illustrated at the top part of Figure 13. We now assume that the capacity requirements of this deployment have increased. In such a case, one could replace the

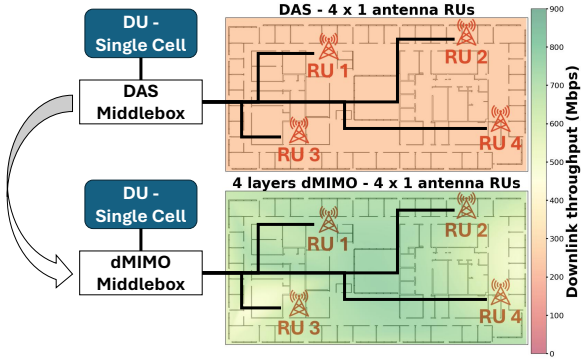
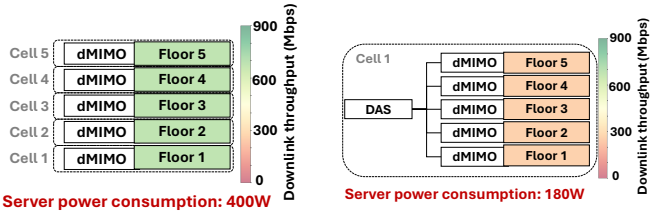


Figure 13: Downlink throughput across single floor when using DAS vs swapping to distributed MIMO middlebox (4×1-antenna RUs), without any infrastructural changes.



(a) One distributed MIMO middlebox per cell (floor).

(b) Sigle cell with DAS and dMIMO middleboxes.

Figure 14: Power consumption and UE downlink throughput with different cell deployment configurations using RANBooster middleboxes, for covering five floors.

DAS middlebox of vendor A, with a distributed MIMO middlebox from vendor B, which could leverage the four single-antenna RUs of the floor, to create a more powerful dMIMO cell with 4 layers. Using our reference implementation of a distributed MIMO middlebox, and as illustrated at the bottom part of Figure 13, we measure that the downlink throughput across the floor increases by a factor of 2 or 3, depending on the location. Similarly to the previous example, this was possible without any infrastructural changes and just with some software upgrades and network re-configurations.

Energy savings – We consider an enterprise scenario, where we want to provide coverage across all five floors of our building. One could deploy one cell per floor with frequency reuse (interference across floors is minimal). For each floor, one could use the dMIMO middlebox that was presented in Section 6.3.2, to combine 4×1-antenna RUs into a single virtual MIMO RU. In this case, a total of two servers (described in Section 6.1) is required. This configuration is illustrated in Figure 14a, along with the total power consumption of the servers (measured using the servers' out-of-band management interface). The figure also illustrates the downlink throughput achieved across the floors, when all 20 UEs of our setup are active and receiving iperf traffic. The total throughput is on average 650Mbps per floor, for a total server power consumption of approximately 400 Watts.

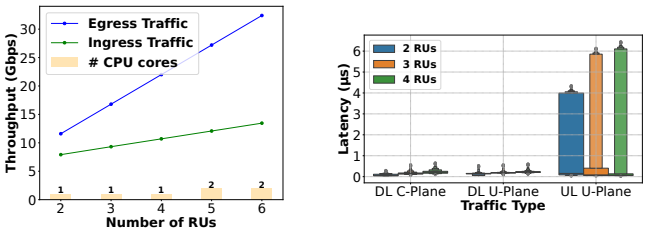
Alternatively, when only few UEs are active across the building, one could leverage middlebox chaining, to deploy a single cell across all five floors, using a combination of DAS and dMIMO, as illustrated in Figure 14b. This drives the total server power consumption down to approximately 180 Watts, since the single cell deployment frees up the CPU cores where the rest of the cells were running, allowing us to shut one server down, and set half of the CPU cores of the second server to low frequency. This translates to a 16% reduction in the overall network power consumption, and thus to significant power (and therefore cost) savings. It should also be noted that, while the downlink throughput per floor in the second case is approximately 150Mbps when all the UEs are active, the instantaneous floor traffic can still reach up to 650Mbps, if the UEs of other floors are idle.

6.4 Microbenchmarks

6.4.1 Scalability. Here we evaluate the RANBooster scalability in terms of compute and network resources. Due to space constraints, we focus on the DAS middlebox, as the most compute and network demanding from Section 4. We consider a 100MHz cell configuration and we use the DPDK implementation of the middlebox, since, the XDP version can currently only handle smaller bandwidths. As illustrated in Figure 15a, a single CPU core can support up to four RUs, without packet loss. By adding one extra CPU core, the solution can scale beyond five RUs. In terms of the networking, we see that the egress and ingress traffic of the middlebox increases linearly with the RUs, but is well below the capacity of the NIC. Given the above, we conclude that most smaller scale deployments (e.g., enterprises and industrial) could be accommodated with a modest number of CPU cores and a single high-end NIC.

To understand why more CPU cores are required as we increase the number of RUs, we measure the packet processing time for a varying number of RUs and for different types of packets processed by the middlebox (control/data plane, uplink/downlink). As we can observe in the boxen plot of Figure 15b, the processing of the downlink control and data plane packets requires less than 300ns in all cases, since the middlebox actions involved (forwarding – action A1, and replication – action A2) are lightweight. For uplink packets, the processing time is split in two segments. The majority of the packets (~ 75%) require processing of less than 300ns, since the middlebox simply caches them (action A3). For the remaining packets, the middlebox performs more heavyweight payload modification actions (action A4), including the (de-)compression and merging of the IQ samples received by all RUs (including the cached ones). This introduces higher packet processing latency overheads (4-6μs) that increase with the number of RUs.

Based on the above, we observe that for each uplink slot of a DAS configuration with four 4×4 100MHz RUs, the middlebox would have to cache 12 packets (3 per RU antenna) and perform 4 IQ sample merge operations (1 per RU antenna) for a total packet processing overhead of ~ 26μs. This overhead maintains the total slot processing time of the vRAN below the deadline threshold, resulting in no negative effect in the end-to-end latency and throughput of the cell. Once we add an extra RU, the total processing overhead of the middlebox exceeds the 30μs, leading to deadline violations and



(a) Network/CPU requirements. (b) Latency overhead.

Figure 15: Network and CPU requirements and per-packet latency overhead for DPDK-based DAS middlebox.

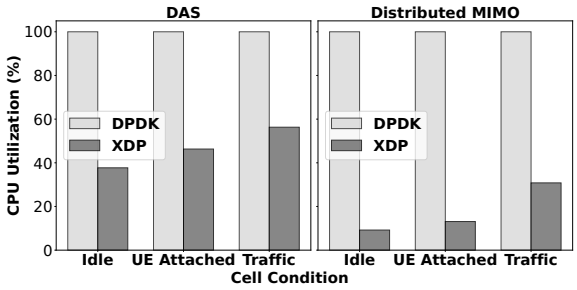


Figure 16: CPU utilization of DPDK vs XDP middleboxes.

packet drops. However, by adding one more CPU core, as illustrated in Figure 15a, we can parallelize the uplink processing operations (i.e., each CPU core handles only a subset of the RU antennas) and can bound the processing latency below the RAN slot processing deadline threshold, leading, once more, to no packet losses.

6.4.2 Trade-offs of DPDK and XDP. Here, we investigate the CPU utilization of the DPDK and XDP implementations of RANBooster, using the DAS and dMIMO middleboxes with a 40MHz cell configuration (due to the XDP limitations). We deploy the middleboxes in a single CPU core and we measure their CPU utilization i) when no UE is attached to the cell, ii) when a UE is attached but idle, and iii) when a UE is attached and receiving downlink traffic at full capacity.

As shown in Figure 16, the DPDK middleboxes always have 100% CPU utilization, due to the use of a poll mode driver. On the other hand, the XDP middleboxes' utilization changes based on the level of fronthaul traffic, due to the interrupt-driven nature of XDP. We also observe that the CPU utilization DAS is ~25-30% higher compared to dMIMO. This is because dMIMO relies on lightweight actions (e.g., header modifications), which can be performed efficiently in the kernel. On the other hand, DAS requires IQ samples decompression and addition, which are performed in the userspace. This introduces additional CPU overhead for both the signal processing and for context switches. We conclude that XDP can be a more suitable option for CPU constrained deployments (e.g. enterprise edge), or middleboxes with lightweight operations that can run in-kernel. DPDK is a more suitable for deployments where more scalability is required.

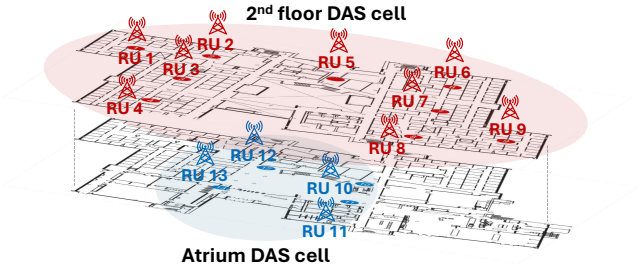


Figure 17: RANBooster private 5G Redmond deployment.

7 Case study: Private 5G with DAS

In addition to the controlled experiments of Sections 6.2-6.4, we have used RANBooster to enable interference-free private 5G connectivity across the Microsoft Research buildings in both Cambridge and Redmond. We have made this 5G network available to all researchers of Microsoft Research, to use for research projects requiring 5G connectivity, by bringing their own 5G devices (smartphones or IoT devices with 5G modems). The network has been operational 24/7 for over five months across both locations.

In the Cambridge case, the network includes the first four floors of the setup presented in Section 6.1 and Figure 9 (four RUs per floor for a total of sixteen RUs). In each floor, we provide coverage through a single cell with a RANBooster DAS middlebox, as shown in Figure 11c, i.e., four cells across all floors with frequency reuse. In the case of Redmond, the network includes a first floor atrium area and the whole second floor of the building (a total of thirteen RUs), as illustrated in Figure 17. For this location, we have deployed two RANBooster DAS middleboxes; one for a cell that includes the four RUs of the atrium area and one for a cell that includes all nine RUs of the second floor. The two cells are configured on non-overlapping frequencies in the 3.3-3.5GHz range, ensuring no inter-cell interference. Using this setup, we have fully covered the floors of interest in both buildings without having to perform sophisticated cell planning or mobility management, despite the challenging layout of the buildings, and with a cost, which we speculate is lower compared to commercial options (see Appendix A.2).

8 Discussion

8.1 Other RANBooster use cases

Sensing – The access of RANBooster middleboxes to raw IQ samples in both the downlink and the uplink direction (action A4 of Section 3.2) facilitates the extraction of unencrypted, physical layer data, such as channel state information, reference signals and radio resource scheduling decisions. Such information can be leveraged to enable a plethora of sensing applications, such as localization [58], interference detection [18], spectrum sensing [56], human activity recognition [9], etc., which are becoming increasingly important in the context of beyond 5G mobile networks [40, 53].

Security – Several RAN fronthaul works have noted security issues from the lack of mandatory integrity and confidentiality protection [2, 24, 41, 64, 70]. While adding such protection mechanisms is an obvious solution, it introduces latency overheads [24, 70], that are often prohibitive in scenarios where hardware cost-efficiency

is important. The RANBooster architecture could be leveraged as a more lightweight alternative, to develop solutions that monitor and mitigate such attacks in real-time through a combination of inspection and dropping of the fronthaul packets (actions A1 and A4 of Section 3.2) , e.g., as proposed in [70]. In addition to mitigating fronthaul attacks, RANBooster, could also be leveraged to mitigate other types of over-the-air cellular attacks, like signal jamming, by manipulating the uplink IQ samples (action A4 of Section 3.2), e.g., using techniques like MIMO null-steering, as described in [74].

RAN resilience – RANBooster middleboxes could be leveraged to introduce resilience to vRAN deployments. For example, one could detect RAN failures by monitoring inter-packet delays (action A4 of Section 3.2) and re-routing the RU traffic to a new DU within a few milliseconds (e.g., using action A1 of Section 3.2, to implement mechanisms along the lines of [38, 69]). The same techniques could be used for enabling RAN software updates.

8.2 Hardware offloading

This work focused on software-based middleboxes. Hardware offloading techniques can further enhance the efficiency of the compute infrastructure in terms of CPU resource utilization and power consumption and can reduce the middlebox processing latency, which can be important in scenarios targeting ultra low latency communications (URLLC). To achieve this, one could consider offloading options such as Data Processing Units (DPUs) and programmable switches with P4 [4]. To align with the RANBooster design, the processing actions described in Section 3.2 need to be mapped to operations of the programmable hardware. While this is straightforward for actions A1 and A2 (packet redirection and replication), it becomes more difficult for actions A3 and A4 (packet caching and payload modification), due to their inherent complexity and memory requirements. To deal with this, several techniques could be explored, such as storing the payload in the switch or some external memory [22, 32, 59, 72] or distributing the processing between both the programmable hardware and the server's CPUs.

8.3 Using Open RAN RUs of other vendors

For our interoperability tests in Section 6.2 we leveraged Foxconn RUs for all three RAN stacks we tested. The same RAN stacks have been shown to work with other Open RAN RUs (e.g., Benetel, VVDN, LiteOn) without any source code modifications, by leveraging the same open fronthaul interface [10, 47, 62]. As such, we expect RANBooster to also work with these radios out of the box, and we are already working on integrating them to our testbed.

8.4 Security risks of fronthaul middleboxes

The use of RANBooster middleboxes in the RAN fronthaul can introduce security risks and vulnerabilities. To protect against such issues, we expect that the RANBooster middleboxes should only be provided by trusted sources and should follow the same security standards and principles as any other network function that is part of the Open RAN architecture (e.g., following the O-RAN Security Work Group specifications [52]).

9 Related Work

Applications leveraging the RAN fronthaul – Many recent works leverage the Open RAN fronthaul interface in several contexts, including resilience [38, 69], security [70], monitoring [63] and network slicing [37]. They leverage fronthaul characteristics (e.g., certain fields of the fronthaul packets) to achieve their optimization goals or to introduce new capabilities. However, they do not consider how such capabilities can be introduced in a principled and non-intrusive way. As such, we view RANBooster as complementary to them, since it provides a concrete framework and a principled way of realizing their logic (as already discussed in Section 8.1), without requiring the involvement of the RAN vendors.

RAN programmability frameworks – Several programmable RAN frameworks have been recently proposed (e.g., [6, 17–19, 29, 33, 55, 60, 61]). Such frameworks typically introduce a controller that can be real-time (e.g., EdgeRIC [33], CloudRIC [60], Janus and Decima [19]) or near real-time (e.g., FlexRIC [61], FlexRAN [17]). The controller hosts the users' applications and facilitates their operation by exchanging control and monitoring data with the RAN. Despite the programmability benefits of such frameworks, they require RAN stack modifications, in the form of "hooks" that are used to expose the data and control knobs. This makes their adoption challenging, as RAN vendors are often reluctant to integrate and modify the required interfaces based on the applications' needs [18, 44]. In contrast RANBooster middleboxes leverage the fronthaul interface, which is standardized and adopted by virtually all RAN vendors, making solutions developed on top of it portable, without the support of the underlying RAN vendor.

Advanced cellular connectivity solutions – Several works focus on ways of introducing advanced cellular connectivity features, like Massive MIMO [21], dMIMO [26, 27], and interference mitigation techniques [29, 74]. While such works demonstrate performance benefits, they rely on custom hardware and software stacks (e.g., modified RAN, custom radios). In contrast, the RANBooster applications rely on commodity hardware and unmodified RAN stacks, reducing their cost and making widespread adoption easier.

10 Conclusions

This work presents RANBooster, which is a middlebox-based architecture for the cellular RAN fronthaul, focused on enabling advanced connectivity features. Through the design and implementation of several reference middleboxes (DAS, dMIMO, RU sharing, real-time PRB monitoring), and by leveraging an enterprise-scale 5G testbed, we demonstrated the benefits of RANBooster in terms of flexibility, performance and ease of use. We hope that the ideas brought out by this work will help lower the barrier of entry for new players in the cellular space and will accelerate the research and development of innovative advanced connectivity features.

Acknowledgments

We would like to thank our shepherd and the anonymous reviewers for their invaluable feedback that helped us improve our work. This research was funded in part by the National Science Foundation, Grant No. 2148230.

References

[1] 3GPP. 2023. About 3GPP. https://www.3gpp.org/about-3gpp.

[2] Aly Sabri Abdalla and Vuk Marojevic. 2024. End-to-end O-RAN security architecture, threat surface, coverage, and the case of the open fronthaul. *IEEE Communications Standards Magazine* 8, 1 (2024), 36–43.

[3] Paramvir Bahl, Matthew Balkwill, Xenofon Foukas, Anuj Kalia, Daehyeok Kim, Manikanta Kotaru, Zhihua Lai, Sanjeev Mehrotra, Bozidar Radunovic, Stefan Saroiu, et al. 2023. Accelerating Open RAN Research Through an Enterprise-scale 5G Testbed. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*. 1–3.

[4] Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese, and David Walker. 2014. P4: programming protocol-independent packet processors. *SIGCOMM Comput. Commun. Rev.* 44, 3 (July 2014), 87–95. https://doi.org/10.1145/2656877.2656890

[5] Nicola Bui and Joerg Widmer. 2016. OWL: A reliable online watcher for LTE control channel measurements. In *Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges*. 25–30.

[6] Raphael Cannatà, Haoxin Sun, Dan Mihai Dumitriu, and Haitham Hassanieh. 2024. Towards Seamless 5G Open-RAN Integration with WebAssembly. In *Proceedings of the 23rd ACM Workshop on Hot Topics in Networks*. 121–131.

[7] Yongce Chen, Y Thomas Hou, Wenjing Lou, Jeffrey H Reed, and Sastry Kompella. 2023. OM 3: Real-Time Multi-Cell MIMO Scheduling in 5G O-RAN. *IEEE Journal on Selected Areas in Communications* (2023).

[8] Supratim Deb, Pantelis Monogioudis, Jerzy Miernik, and James P Seymour. 2013. Algorithms for enhanced inter-cell interference coordination (eICIC) in LTE HetNets. *IEEE/ACM transactions on networking* 22, 1 (2013), 137–150.

[9] Manu Dwivedi, Ian Ellis L Hulede, Oscar Venegas, Jonathan Ashdown, and Amitav Mukherjee. 2024. 5G-based passive radar sensing for human activity recognition using deep learning. In *2024 IEEE Radar Conference (RadarConf24)*. IEEE, 1–6.

[10] Keith Dyer. 2021. Benetel confirms Radisys tie-up for new Open RAN Radio Unit venture. . https://the-mobile-network.com/2021/01/benetel-unites-with-radisys-for-open-ran-ru/. Accessed on 06.13.2025.

[11] Salvatore D'Oro, Leonardo Bonati, Francesco Restuccia, and Tommaso Melodia. 2021. Coordinated 5G network slicing: How constructive interference can boost network throughput. *IEEE/ACM Transactions on Networking* 29, 4 (2021), 1881–1894.

[12] Ericsson. 2019. 5G is all in the timing. https://www.ericsson.com/en/blog/2019/8/what-you-need-to-know-about-timing-and-sync-in-5g-transport-networks. Accessed on 01.31.2025.

[13] Ericsson. 2019. Radio Stripes: re-thinking mobile networks. https://www.ericsson.com/en/blog/2019/2/radio-stripes. Accessed on 01.08.2025.

[14] Ericsson. 2023. Bringing performance at scale to Open RAN. https://www.ericsson.com/en/reports-and-papers/further-insights/driving-open-ran-forward. Accessed on 01.31.2025.

[15] Robert Falkenberg and Christian Wietfeld. 2019. FALCON: An Accurate Real-time Monitor for Client-based Mobile Network Data Analytics. In *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE, Waikoloa, Hawaii, USA. https://doi.org/10.1109/GLOBECOM38437.2019.9014096 arXiv:1907.10110

[16] Xenofon Foukas, Mahesh K Marina, and Kimon Kontovasilis. 2019. Iris: Deep reinforcement learning driven shared spectrum access architecture for indoor neutral-host small cells. *IEEE Journal on Selected Areas in Communications* 37, 8 (2019), 1820–1837.

[17] Xenofon Foukas, Navid Nikaein, Mohamed M Kassem, Mahesh K Marina, and Kimon Kontovasilis. 2016. FlexRAN: A flexible and programmable platform for software-defined radio access networks. In *Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies*. 427–441.

[18] Xenofon Foukas, Bozidar Radunovic, Matthew Balkwill, and Zhihua Lai. 2023. Taking 5G RAN analytics and control to a new level. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*. 1–16.

[19] Xenofon Foukas, Bozidar Radunovic, Matthew Balkwill, Zhihua Lai, and Connor Settle. 2023. Programmable RAN Platform for Flexible Real-Time Control and Telemetry. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*. 1–3.

[20] Linux Foundation. 2015. Data Plane Development Kit (DPDK). http://www.dpdk.org

[21] Junzhi Gong, Anuj Kalia, and Minlan Yu. 2023. Scalable distributed massive MIMO baseband processing. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. 405–417.

[22] Swati Goswami, Nodir Kodirov, Craig Mustard, Ivan Beschastnikh, and Margo Seltzer. 2020. Parking packet payload with P4. In *Proceedings of the 16th International Conference on Emerging Networking EXperiments and Technologies*. 274–281.

[23] Prateesh Goyal, Anup Agarwal, Ravi Netravali, Mohammad Alizadeh, and Hari Balakrishnan. 2020. ABC: A simple explicit congestion controller for wireless networks. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. 353–372.

[24] Joshua Groen, Salvatore D'Oro, Utku Demir, Leonardo Bonati, Davide Villa, Michele Polese, Tommaso Melodia, and Kaushik Chowdhury. 2024. Securing O-RAN Open Interfaces. *IEEE Transactions on Mobile Computing* (2024).

[25] GSMA. 2012. Mobile Infrastructure Sharing. https://www.gsma.com/solutions-and-impact/connectivity-for-good/public-policy/wp-content/uploads/2012/09/Mobile-Infrastructure-sharing.pdf.

[26] Ezzeldin Hamed, Hariharan Rahul, Mohammed A Abdelghany, and Dina Katabi. 2016. Real-time distributed MIMO systems. In *Proceedings of the 2016 ACM SIGCOMM Conference*. 412–425.

[27] Ezzeldin Hamed, Hariharan Rahul, and Bahar Partov. 2018. Chorus: Truly distributed distributed-MIMO. In *Proceedings of the 2018 conference of the ACM special interest group on data communication*. 461–475.

[28] Tuan Dinh Hoang, CheolJun Park, Mincheol Son, Taekkyung Oh, Sangwook Bae, Junho Ahn, Beomseok Oh, and Yongdae Kim. 2023. LTESniffer: An open-source LTE downlink/uplink eavesdropper. In *Proceedings of the 16th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. 43–48.

[29] Sesha Sai Rakesh Jonnavithula, Ish Kumar Jain, and Dinesh Bharadia. 2024. MIMO-RIC: RAN Intelligent Controller for MIMO xApps. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*. 2315–2322.

[30] Vaishnavi Kasuluru, Luis Blanco, Cristian J Vaca-Rubio, and Engin Zeydan. 2024. On the Impact of PRB Load Uncertainty Forecasting for Sustainable Open RAN. In *2024 IEEE 35th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*. IEEE, 1–7.

[31] Sina Khanifar. [n. d.]. How Much Does a Distributed Antenna System Cost? https://www.waveform.com/pages/how-much-does-a-distributed-antenna-system-cost. Accessed on 01.08.2025.

[32] Daehyeok Kim, Zaoxing Liu, Yibo Zhu, Changhoon Kim, Jeongkeun Lee, Vyas Sekar, and Srinivasan Seshan. 2020. Tea: Enabling state-intensive network functions on programmable switches. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*. 90–106.

[33] Woo-Hyun Ko, Ushasi Ghosh, Ujwal Dinesha, Raini Wu, Srinivas Shakkottai, and Dinesh Bharadia. 2024. EdgeRIC: Empowering Real-time Intelligent Optimization and Control in NextG Cellular Networks. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*. 1315–1330.

[34] Tereza Krásová. 2024. Neutral host network sharing in cities can slash costs by up to 47% – study. *Light Reading* (25 Nov. 2024).

[35] Patrick Kutch and Brian Johnson. 2017. SR-IOV for nfv solutions. *Practical Considerations and Thoughts (Intel, Networking Division)* (2017).

[36] Andrea Lacava, Michele Polese, Rajarajan Sivaraj, Rahul Soundrarajan, Bhawani Shanker Bhati, Tarunjeet Singh, Tommaso Zugno, Francesca Cuomo, and Tommaso Melodia. 2023. Programmable and customized intelligence for traffic steering in 5G networks using open RAN architectures. *IEEE Transactions on Mobile Computing* 23, 4 (2023), 2882–2897.

[37] Line MP Larsen, Michael S Berger, and Henrik L Christiansen. 2018. Fronthaul for cloud-RAN enabling network slicing in 5G mobile networks. *Wireless Communications and Mobile Computing* 2018, 1 (2018), 4860212.

[38] Nikita Lazarev, Tao Ji, Anuj Kalia, Daehyeok Kim, Ilias Marinos, Francis Y Yan, Christina Delimitrou, Zhiru Zhang, and Aditya Akella. 2023. Resilient baseband processing in virtualized rans with slingshot. In *Proceedings of the ACM SIGCOMM 2023 Conference*. 654–667.

[39] Linux kernel. [n. d.]. AF_XDP. https://www.kernel.org/doc/html/latest/networking/af_xdp.html. Accessed on 01.31.2025.

[40] Fan Liu, Yuanhao Cui, Christos Masouros, Jie Xu, Tony Xiao Han, Yonina C Eldar, and Stefano Buzzi. 2022. Integrated sensing and communications: Toward dual-functional wireless networks for 6G and beyond. *IEEE journal on selected areas in communications* 40, 6 (2022), 1728–1767.

[41] Madhusanka Liyanage, An Braeken, Shahriar Shahabuddin, and Pasika Ranaweera. 2023. Open RAN security: Challenges and opportunities. *Journal of Network and Computer Applications* 214 (2023), 103621.

[42] David López-Pérez, Antonio De Domenico, Nicola Piovesan, and Mérouane Debbah. 2024. Data-driven Energy Efficiency Modelling in Large-scale Networks: An Expert Knowledge and ML-based Approach. *IEEE Transactions on Machine Learning in Communications and Networking* (2024).

[43] N. Ludant, P. Robyns, and G. Noubir. 2023. From 5G Sniffing to Harvesting Leakages of Privacy-Preserving Messengers. In *2023 IEEE Symposium on Security and Privacy (SP)*.

[44] Iain Morris. 2024. AT&T, all in with Ericsson, seems to have shut the door to xApps . https://www.lightreading.com/open-ran/at-t-all-in-with-ericsson-seems-to-have-shut-the-door-to-xapps. Accessed on 01.08.2025.

[45] Jalal Mostafa, Suren Chilingaryan, and Andreas Kopmann. 2023. Are Kernel Drivers Ready For Accelerated Packet Processing Using AF_XDP?. In *2023 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. IEEE, 117–122.

[46] Ahmad M Nagib, Hatem Abou-Zeid, Hossam S Hassanein, Akram Bin Sediq, and Gary Boudreau. 2021. Deep learning-based forecasting of cellular network utilization at millisecond resolutions. In *ICC 2021-IEEE International Conference*

on Communications. IEEE, 1–6.
[47] Office of Public Affairs NTIA. 2023. NTIA, Department of Defense Announce Final Winners of the 2023 5G Challenge . https://www.ntia.gov/press-release/2023/ntia-department-defense-announce-final-winners-2023-5g-challenge. Accessed on 06.13.2025.
[48] O-RAN Alliance. 2023. O-RAN Alliance Who We Are. https://www.o-ran.org/who-we-are.
[49] O-RAN Alliance 2023. *O-RAN Work Group 1 (Use Cases and Overall Architecture): Use Cases Detailed Specification.* O-RAN Alliance.
[50] O-RAN Alliance 2024. *O-RAN Architecture Overview.* O-RAN Alliance.
[51] O-RAN Alliance 2024. *O-RAN Working Group 4 (Open Fronthaul Interfaces WG): Control, User and Synchronization Plane Specification.* O-RAN Alliance.
[52] O-RAN Alliance 2025. *O-RAN Work Group 11 (Security Work Group) Security Requirements and Controls Specifications.* O-RAN Alliance.
[53] Qingrui Pan, Mahesh K Marina, and Thanos Triantafyllou. 2025. On NextG Open RAN as a Sensing Infrastructure. In *Proceedings of the 26th International Workshop on Mobile Computing Systems and Applications*. 55–60.
[54] Ben Pfaff, Justin Pettit, Teemu Koponen, Ethan Jackson, Andy Zhou, Jarno Rajahalme, Jesse Gross, Alex Wang, Joe Stringer, Pravin Shelar, et al. 2015. The design and implementation of open {vSwitch}. In *12th USENIX symposium on networked systems design and implementation (NSDI 15)*. 117–130.
[55] Van-Quan Pham, Huu-Trung Thieu, Ahan Kak, and Nakjung Choi. 2023. HexRIC: Building a better near-real time network controller for the Open RAN ecosystem. In *Proceedings of the 24th International Workshop on Mobile Computing Systems and Applications*. 15–21.
[56] Clifton Paul Robinson, Daniel Uvaydov, Salvatore D'Oro, and Tommaso Melodia. 2024. Deepsweep: Parallel and scalable spectrum sensing via convolutional neural networks. In *2024 IEEE International Conference on Machine Learning for Communication and Networking (ICMLCN)*. IEEE, 505–510.
[57] Ivan Romanov. 2023. How Much Does a Distributed Antenna System (DAS) in the UK Cost? https://www.uctel.co.uk/blog/how-much-does-a-distributed-antenna-system-das-in-the-uk-cost. Accessed on 01.08.2025.
[58] Yanlin Ruan, Liang Chen, Xin Zhou, Guangyi Guo, and Ruizhi Chen. 2022. HiLoc: Hybrid indoor localization via enhanced 5G NR CSI. *IEEE Transactions on Instrumentation and Measurement* 71 (2022), 1–15.
[59] Mariano Scazzariello, Tommaso Caiazzi, Hamid Ghasemirahni, Tom Barbette, Dejan Kostić, and Marco Chiesa. 2023. A High-Speed Stateful Packet Processing Approach for Tbps Programmable Switches. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. 1237–1255.
[60] Leonardo Lo Schiavo, Gines Garcia-Aviles, Andres Garcia-Saavedra, Marco Gramaglia, Marco Fiore, Albert Banchs, and Xavier Costa-Perez. 2024. CloudRIC: Open radio access network (o-ran) virtualization with shared heterogeneous computing. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*. 558–572.
[61] Robert Schmidt, Mikel Irazabal, and Navid Nikaein. 2021. FlexRIC: An SDK for next-generation SD-RANs. In *Proceedings of the 17th International Conference on emerging Networking EXperiments and Technologies*. 411–425.
[62] SRS. 2024. Software Radio Systems Announces Interoperability with Nine Open RAN Radio Units, Expanding Private 5G Deployment Options . https://srs.io/software-radio-systems-oru-integrations/. Accessed on 06.13.2025.
[63] Chuanhao Sun, Ujjwal Pawar, Molham Khoja, Xenofon Foukas, Mahesh K Marina, and Bozidar Radunovic. 2024. SpotLight: Accurate, explainable and efficient anomaly detection for Open RAN. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*. 923–937.
[64] Azadeh Tabiban, Hyame Assem Alameddine, Mohammad A Salahuddin, and Raouf Boutaba. 2023. Signaling Storm in O-RAN: Challenges and Research Opportunities. *IEEE Communications Magazine* (2023).
[65] William Tu, Yi-Hung Wei, Gianni Antichi, and Ben Pfaff. 2021. Revisiting the open vswitch dataplane ten years later. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*. 245–257.
[66] Viavi. [n. d.]. Considerations for Edge Master Deployment. https://www.viavisolutions.com/en-us/literature/considerations-edge-master-deployment-application-notes-en.pdf. Accessed on 01.31.2025.
[67] Haoran Wan, Xuyang Cao, Alexander Marder, and Kyle Jamieson. 2024. NR-Scope: A Practical 5G Standalone Telemetry Tool. In *Proceedings of the 20th International Conference on emerging Networking EXperiments and Technologies*. 73–80.
[68] Yaxiong Xie, Fan Yi, and Kyle Jamieson. 2020. PBE-CC: Congestion control via endpoint-centric, physical-layer bandwidth measurements. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*. 451–464.
[69] Jiarong Xing, Junzhi Gong, Xenofon Foukas, Anuj Kalia, Daehyeok Kim, and Manikanta Kotaru. 2023. Enabling Resilience in Virtualized RANs with Atlas. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*. 1–15.
[70] Jiarong Xing, Sophia Yoo, Xenofon Foukas, Daehyeok Kim, and Michael K Reiter. 2024. On the criticality of integrity protection in 5G fronthaul networks. In *33rd USENIX Security Symposium (USENIX Security 24)*. 4463–4479.
[71] Dongzhu Xu, Anfu Zhou, Guixian Wang, Huanhuan Zhang, Xiangyu Li, Jialiang Pei, and Huadong Ma. 2022. Tutti: coupling 5g ran and mobile edge computing for latency-critical video analytics. In *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking*. 729–742.
[72] Chaoliang Zeng, Layong Luo, Teng Zhang, Zilong Wang, Luyang Li, Wenchen Han, Nan Chen, Lebing Wan, Lichao Liu, Zhipeng Ding, et al. 2022. Tiara: A scalable and efficient hardware acceleration architecture for stateful layer-4 load balancing. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*. 1345–1358.
[73] Yang Zhou, Xingyu Xiang, Matthew Kiley, Sowmya Dharanipragada, and Minlan Yu. 2024. DINT: Fast In-Kernel Distributed Transactions with eBPF. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*. 401–417.
[74] Frederik Jonathan Zumegen, Ish Kumar Jain, and Dinesh Bharadia. 2024. BeamArmor: Seamless Anti-Jamming in 5G Cellular Networks with MIMO Null-steering. In *Proceedings of the 25th International Workshop on Mobile Computing Systems and Applications*. 121–126.

A Appendices

Appendices are supporting material that has not been peer-reviewed.

A.1 RU sharing details

Here we describe more details about the RU sharing design.

A.1.1 Selecting DU center of frequency with aligned PRBs. As described in Section 4.3, PRB copying can be optimized by skipping PRB (de)compression. This is achieved by aligning the DU PRBs with the RU PRBs, as seen in Figure 6, by shifting the DU center of frequency. To pick a suitable DU center of frequency, one can pick a PRB from the RU's spectrum to align it to. We call this PRB the prb_offset, since it is the offset that the DU's center of frequency is from PRB 0 of the RU's spectrum. The derivation of the DU's center of frequency, $DU\_center\_of\_frequency$, is as follows:

$$PRB\_0\_frequency = RU\_center\_of\_frequency \quad (1)$$

$$- 12 \times SCS \times \frac{RU\_num\_prb}{2} \quad (2)$$

$$DU\_center\_of\_frequency = PRB\_0\_frequency \quad (3)$$

$$+12 \times SCS \times \left(prb\_offset + \frac{DU\_num\_prb}{2} \right) \quad (4)$$

where *SCS* is the subcarrier spacing and *num_prb* is the number of PRBs of the DU or RU's full spectrum.

A.1.2 RU Sharing Algorithm. Here we describe the packet manipulation algorithm of the RU sharing middlebox of Section 4.3 in more detail. First, we describe the handling of C-Plane and U-Plane messages pertaining to data. Then, the handling of the unique Physical Random Access Channel (PRACH) control messages is discussed.
C-Plane: When a C-Plane message arrives, the field *numPRBS*, that encodes the number of PRBs expected by the DU, is set to the number of PRBs that the RU supports. This is so the RU sends back all the PRBs of its full spectrum that can then be demultiplexed. Only the first C-Plane message is sent to the RU per symbol, while the other C-Plane messages for that symbol that arrive are not. However, all C-Plane messages for each symbol are cached for later U-Plane message processing, as seen on line 2 in Algorithm 2.
U-Plane: All downlink U-Plane messages arriving for a particular symbol are cached. When each of the C-Plane messages of a symbol have a corresponding cached downlink U-Plane packet, then U-Plane messages for that symbol can be multiplexed. Similarly, when

Algorithm 2 RU Sharing Algorithm for data messages

```
if Packet is a data message then
    Cache packet
    if packet is C-Plane then
        if first packet in cache for symbol and antenna port then
            Change numPRB and MAC addresses to forward to RU in packet
            Send packet
        end if
    else if packet is U-Plane then
        if packet is a downlink and all cached U-Plane packets match all cached C-Plane packets then
            Create new packet new_packet to send to RU
            for each packet in cached U-Plane packet do
                copy PRBs of packet into new_packet in correct location
            end for
            Send new_packet
            Uncache all packet in cached U-Plane packet and cached C-Plane packet for that symbol and antenna port
        else if packet is a uplink then
            for each DU do
                if packet is cached in C-Plane for the DU then
                    Create new packet new_packet to send to RU
                    Copy PRBs in correct location in packet into new_packet
                    Send new_packet
                end if
            end for
            Uncache all packet in cached U-Plane packet and cached C-Plane packet for that symbol and antenna port
        end if
    end if
end if
```

an uplink U-Plane message arrives for a symbol, it is demultiplexed according to the C-Plane packets cached for that symbol.

PRACH handling: UEs attempt to attach to a cell by sending a request through a special uplink channel called PRACH. The PRACH signals are sent periodically over a different set of antenna ports than the data messages and only span a subsection of the entire frequency range assigned to a DU. This subsection is denoted by a number of PRBs, and by a *frequency offset* field, that indicates the location of the first resource element of the first PRB of the PRACH frequency range in the full frequency range assigned to the DU. In order to map the subsection from the DU to the full bandwidth of the RU, the *frequency offset* field in the U-Plane message needs to be translated to the RU spectrum. This is so that the RU sends back signals of the correct frequency to the DU as PRACH and the DU can decode the UE attach attempts.

The following is a derivation of the frequency offset given to the RU based on the C-Plane packet from the DU. *freqOffset* is the field found in the DU C-Plane packet that needs to be converted to the frequency offset using the subscarrier spacing, denoted as *SCS*. The frequency of the first resource element of the first PRB, $frequency\_re0rb0$, is calculated, then converted to the *freqOffset* needed for the C-Plane packet destined for the RU.

$$frequency\_offset\_DU = freqOffset\_DU \times 0.5 \times SCS \quad (5)$$

$$frequency\_re0rb0 = DU\_center\_of\_frequency \quad (6)$$

$$- frequency\_offset\_DU \quad (7)$$

$$frequency\_offset\_RU = RU\_center\_of\_frequency \quad (8)$$

$$- frequency\_re0rb0 \quad (9)$$

$$freqOffset\_RU = \frac{frequency\_offset\_RU}{0.5 \times SCS} \quad (10)$$

Algorithm 3 RU Sharing Algorithm for PRACH messages

```
if packet is a PRACH message then
    Cache packet
    if packet is C-Plane then
        if all packets are cached then
            Append all sections into one packet
            Change the freqOffset field of each section as described in text
            Change the section ID of each section to match the DU iD
            Send packet
        end if
    else if packet is U-Plane then
        for each DU do
            Create new packet new_packet to send to DU
            Copy the section with matching ID as the DU in packet into new_packet
            Send new_packet
        end for
        Uncache all packet in cached U-Plane packet and cached C-Plane packet for that symbol and antenna port
    end if
end if
```

Or more simplified as:

$$freqOffset\_RU = freqOffset\_DU + \frac{RU\_center\_of\_frequency - DU\_center\_of\_frequency}{0.5 \times SCS} \quad (11)$$

For PRACH messages, in contrast to data messages, only the PRBs that are requested by the C-Plane are sent by the RU in the U-Plane. We use this to our advantage by appending the PRACH C-Plane sections to the same packet with the section ID as the ID of the DU that sent the section. In the uplink, the U-Plane sections for each DU are demultiplexed according to the cached C-Plane messages and DU ID. The algorithm is depicted in Algorithm 3.

A.2 RANBooster cost benefits

Here, we use our enterprise-scale testbed deployment of Section 7 to make a best effort estimate about the CapEx reduction benefits that a speculative RANBooster offering could bring, compared to conventional solutions used for enabling advanced cellular connectivity features. For this analysis, we focus on the Cambridge location. We calculate that the cost of deploying our commodity RAN infrastructure was approximately $60,000, including the cost of commodity RUs, cabling equipment and building work, switches, grandmaster clock, NICs, as well as 8 CPU cores, for running the RANBooster middleboxes. In comparison, we estimate the cost of deploying a conventional DAS solution leveraging the approximate reference prices found in several online sources [31, 57]. Considering that our deployment is 77,015 square feet (15,403 square feet per floor × 5 floors), a conventional DAS solution would cost approximately $154,000, assuming a conservative $2 cost per square foot. We can observe that, even by assuming a profit margin of 50%, a deployment based on RANBooster is 41% cheaper than the conventional solution, without even taking into account that it can also enable other additional features beyond DAS, like the RU sharing scenario demonstrated in Section 6.3.2, which could increase the price of conventional solutions by 3 times or more [57].