Enabling Resilience in Virtualized RANs with Atlas

Jiarong Xing[‡*], Junzhi Gong[§*], Xenofon Foukas[†], Anuj Kalia[†], Daehyeok Kim[†¶], Manikanta Kotaru[†]
[†]Microsoft, [‡]Rice University, [§]Harvard University, [¶]UT Austin

ABSTRACT

Virtualized radio access networks (vRANs), which allow running RAN processing on commodity servers instead of proprietary hardware, are gaining adoption in cellular networks. Two properties of the vRAN's "Distributed Unit (DU)" that implements the lower RAN layers—its real-time deadlines and its black-box nature—make it challenging to provide resilience features such as upgrades and failover without long service disruptions. These properties preclude the use of existing resilience techniques like virtual machine migration or state replication that are used for typical workloads. This paper presents Atlas, the first system that provides resilience for the DU. The central insight in Atlas is to repurpose existing cellular mechanisms for *wireless* resilience, namely handovers and cell reselection, to provide *software* resilience for the DU. For planned resilience events like upgrades, we design a novel technique that simultaneously serves cells from both the old and new DUs via the same radio, and uses handovers between these cells to migrate user devices. For unplanned failures, we identify deficiencies in existing RAN protocols that disrupt cell reselection after DU failure, and show how we can eliminate these disruptions using a middlebox between the DU and higher layers. Our evaluation with a state-of-the-art 5G vRAN testbed shows that Atlas achieves minimal disruption to cellular connectivity during resilience events, while incurring low overhead.

CCS CONCEPTS

• **Networks** → **Mobile networks**; **Wireless access points, base stations and infrastructure**; • **Computer systems organization** → *Real-time systems*; **Reliability**; **Availability**.

[*] The first two authors contributed equally to this work.

ACM MobiCom '23, October 2–6, 2023, Madrid, Spain

ACM ISBN 978-1-4503-9990-6/23/10…$15.00
https://doi.org/10.1145/3570361.3613276

KEYWORDS

5G, vRAN, resilience, handover, RU sharing, RAN failover, RAN migration

1 INTRODUCTION

The emergence of vRANs [3, 13, 49] brings new promises and challenges for cellular network operators. On the one hand, softwarization of the RAN promises benefits like increased feature velocity, a larger vendor ecosystem, security, and reduced CapEx/OpEx. On the other hand, it opens up a variety of unexplored problems that naturally arise from running the vRAN's real-time and often black-box software on commodity Linux servers.

This paper focuses on one such problem: the lack of resilience in today's vRANs. Cellular networks provide a critical infrastructure that is relied upon for emergency services and mission-critical applications. However, there are no methods for vRAN maintenance or upgrades, or to recover from crashes in vRAN software or hardware, without causing significant service disruption lasting many seconds to minutes (§7). Operators therefore rely on planned late-night downtime windows for maintenance, and accept outages during crashes. This limits the benefits of vRANs, e.g., since version upgrades or security patches are difficult to apply.

This paper presents Atlas, the first system to provide resilience for the vRAN's real-time component, called the "Distributed Unit" (DU). The DU is deployed in far edge datacenters close to the radios, and consists of the lower layers of the vRAN stack, including the Physical (PHY), Media Access Control (MAC), and Radio Link Control (RLC) layers. Atlas provides a new primitive called "DU migration", which allows moving the PHY–RLC processing for a cell from one DU to another DU on a different server. With Atlas' *proactive* migration, operators can upgrade or service a DU without downtime, seamlessly moving each user device (also called user equipment, or UE) attached to the source DU to another DU. Atlas' *reactive* migration handles abrupt hardware and software failures, with a sub-second disruption.

There are two main challenges in making the DU resilient. First, the DU has strict sub-millisecond real-time latency deadlines, which precludes the use of general-purpose resilience approaches like virtual machine or container migration that require pausing the running software for hundreds of milliseconds. Second, since the DU requires highly complex and optimized software written by domain experts, it is often a black box where the source code is proprietary

or unavailable. For example, Radisys' minimal open-source DU that implements only a small subset of DU functionality contains 180K source lines of code [17]. This precludes resilience techniques used for higher layers of the cellular stack that significantly modify the source code to externalize all computation states into a reliable state store [29, 40].

Our observation behind Atlas is we can use cellular networks' existing mechanisms for wireless-level resilience to provide software-level resilience. For example, during proactive DU migration, we need to move UE processing from the source DU to the destination DU; this happens naturally during UE handovers, where UEs disconnect from one DU and reconnect to another. Similarly, during reactive DU migration, we need to re-create UE sessions at the destination DU; this happens naturally during a process called "cell reselection", where UEs lose coverage from their original DU (e.g., after turning a sharp corner while walking), and then quickly discover and reconnect to a different DU.

Using the existing resilience mechanisms in Atlas presents two challenges. First, for proactive migration, we need a way to serve two cells—one each for the source and destination DUs—from the same radio unit (RU) during a transient period. This is challenging because the entire vRAN stack is designed assuming a one-to-one RU–DU association. We investigate intuitive RU-sharing approaches, e.g., via splitting only along the time and frequency dimensions, and discuss their infeasibility. Instead, we design a novel lightweight RU-sharing mechanism that also splits along the spatial dimension by using the RU's multiple antenna ports, and exploits special properties of cellular control channel signals. We implement the RU sharing by introducing a vendor-agnostic fronthaul network function for multiplexing the fronthaul traffic of the DUs, and by introducing radio resource scheduling logic for mitigating interference at the MAC layer.

Second, for reactive migration, we identify 5G RAN protocol limitations, caused by their design being agnostic to DU failures. We show how these limitations can delay UEs' reconnection to a backup DU after the primary DU fails, and cause the UEs to subsequently disconnect. In lieu of the standards' protocol fixes, we present a clean and lightweight midhaul network function that brings DU failure awareness to the "Centralized Unit" (CU) above the DU, enabling fast failover.

To demonstrate the generality and portability of our design, we implement Atlas' fronthaul NF for three different targets (an eBPF hook in Intel FlexRAN PHY, DPDK-based software middlebox, and Tofino switching ASICs), and we realize the interference mitigation logic at the MAC layer by leveraging parameters exposed by the open O-RAN E2 interface. We also implement the midhaul NF in C++ as a lightweight process running on the CU server. Our implementation requires minimal modifications to the vRAN software, which is well aligned with 3GPP and O-RAN specifications.

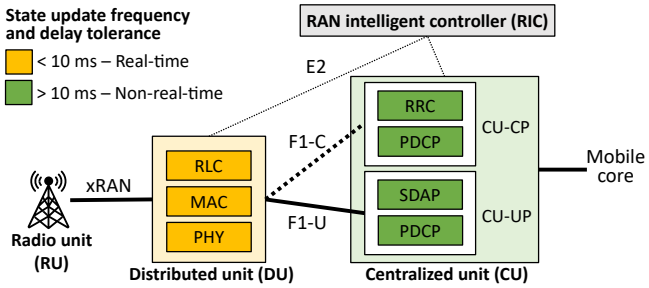


Figure 1: A typical vRAN deployment model. The RU and DU are connected via a fronthaul network, and the DU and CU are connected via a midhaul network. They communicate through open interfaces.

We evaluate Atlas in a production-grade 5G vRAN testbed with commercial UEs. Our experiments show that Atlas supports proactive migration with zero UE downtime, and reactive migration with as little as 700 ms of downtime. In both cases, UEs regain their full performance after migration finishes. We show how Atlas can be implemented with near-zero compute and latency overhead.

2 MOTIVATION AND BACKGROUND

2.1 A primer on virtualized RANs

RAN components. Figure 1 illustrates a typical 5G vRAN deployment, with its three main components: the Radio Unit (RU), the Distributed Unit (DU), and the Centralized Unit (CU). The RU is typically implemented in fixed-function hardware (e.g., ASICs or FPGAs), while the DU and CU are software applications running on commodity servers. Each RU constitutes a "cell", and connects to the DU via a fronthaul network. For this work, and without loss of generality, we use the terms RU and cell interchangeably to indicate a one-to-one mapping of a cell to the radio hardware.

The "virtualized" DU and CU serve the various RAN protocol layers (see below). The DU has strict real-time latency requirements and therefore runs close to the RUs. The CU is delay-tolerant and can therefore run farther away from the RU. A DU server typically hosts several (e.g., ten) cells, and a CU server hosts hundreds of DUs. A RAN Intelligent controller (RIC) facilitates the programmability of the RAN.

RAN protocol layers. RAN functionality is divided into several layers, each responsible for a distinct set of control and/or data plane operations (Figure 1). The DU implements the real-time layers, e.g., the PHY layer for wireless signal processing, and the MAC layer for scheduling radio resources among UEs. The CU implements the more delay-tolerant layers, such as the Radio Resource Control (RRC) layer that manages radio-related UE operations, including handovers that move the UE from one cell to another.

Open RAN interfaces. The RAN components communicate with each other via a set of standardized interfaces, designed for interoperability by consortiums like 3GPP and the O-RAN Alliance [9]. Three DU interfaces are relevant to this work: the xRAN fronthaul interface with the RU, the F1-C interface with the CU's control plane, and the E2 interface with the RIC. The xRAN interface that carries digitized radio signals (IQ samples) has real-time latency requirements.

2.2 Need for resilience in vRANs

The cellular network is a crucial piece of infrastructure that needs to be highly available to support critical applications, such as public safety and factory automation. Resilience has been studied extensively for the cellular network's non-real-time components, i.e., the mobile core and CU (e.g., [29, 40, 44, 46, 47]). Supporting resilience for the DU, however, poses unique challenges due to its strict real-time latency requirements and black-box nature, with no solution existing at this time. We divide DU resilience events into two types:

Planned events. A key promise of vRANs is the ease of (1) rolling out updates, such as new RAN features and OS-/security patches; and (2) hardware servicing via planned maintenance. These happen frequently, e.g., according to AT&T, some subsets of their RANs are upgraded daily, but with pre-planned downtime [45, 51]. However, due to the lack of resilience mechanisms for vRAN DUs, such updates today create significant downtime for users, lasting several seconds to minutes (see §7).

Unplanned events. The vRAN must quickly recover from unplanned failures like software crashes or hardware malfunctions. Mean time of server hardware failures can range between 10 and 60 days [7, 15], with repairs taking several hours [7]. If a vRAN DU server fails, UEs attached to the DU should quickly be migrated to another DU server.

2.3 Requirements for resilient vRANs

Ideally, a system that supports resilience for vRANs should provide the following three properties:

Minimal downtime during resilience events. To allow operators to perform vRAN upgrades and security patches without concern for their impact on user connectivity, the system must incur zero downtime during planned resilience events. For failures, it must keep downtime comparable to wireless-related service interruptions that users naturally experience, e.g., due to handover failures.

Minimize overhead and cost. Under normal operations, it must add near-zero overhead. vRANs operate with tight CPU and latency budgets, so any overhead must be carefully considered. Cost should also be kept to a minimum, considering the large scale of DU deployments (e.g., Dish and Rakuten report more than 5K DU sites in their networks [38, 48]).

Vendor agnostic. It must be compatible with any standards-compliant vRAN implementation without modifications, for two reasons. First, commercial-grade vRAN software is typically proprietary, written by domain experts, and has extreme complexity. Recreating it from scratch is not feasible. For example, even the simplified open-source reference implementation of OpenAirInterface [4] has more than 600K lines of code. Second, there are numerous vRAN implementations for different hardware architectures (e.g., GPUs and SoCs), so a vendor-specific approach has limited applicability.

2.4 Limitations of existing RAN resilience approaches

We now discuss how existing cellular resilience approaches are insufficient for DUs. While no existing techniques target the DU directly, we discuss their natural DU adaptations.

Offloading UEs to neighbor cells. In traditional RANs, each cell site has a separate DU appliance. This contains the impact of DU upgrades or failures to one cell site, since these events can often be handled by offloading affected UEs to neighboring cells [45, 51, 52]. For planned events, the network can do this gracefully using handovers. For unplanned events, UEs can re-attach to a neighboring cell.

Adapting neighbor cell offload for DU resilience has several limitations. First, it is feasible only with dense cell coverage, which is not always available, e.g., in sparse rural and enterprise deployments. Second, for unplanned failures, UEs fail to reconnect reliably to the neighboring cell due to the failure-agnostic nature of existing 5G protocols (§5.2). Third, ensuring good overlapping coverage requires maintenance windows, as well as complex cell planning and antenna management to adjust the power and tilt of RU antennas [51] during maintenance. Fourth, UEs experience reduced throughput due to worse signal quality from the neighbor, e.g., up to 25% worse TCP throughput in our indoor testbed (§7).

In addition to the above issues, due to the small size of vRAN datacenters, it is not always possible to even place neighboring cells' DUs on different servers. Unlike traditional RANs, one DU server may host multiple cell sites, and a server failure may bring down all sites providing coverage overlap. Imposing physical world constraints on DU placement also goes against the principles of virtualization, which treats every server as a homogeneous resource. Such constraints will limit the benefits of virtualization, such as load-aware dynamic RU-to-server bin packing (called BBU pooling [42]), an important vRAN feature for efficiency.

Fault-tolerant state store. One technique to build resilient network functions is to externalize their state in a replicated, fault-tolerant state store. For example, ECHO [40] uses a key-value store to replicate the state of an LTE Evolved Packet Core (EPC), which is delay-tolerant. This approach does not

apply to the DU for two reasons. First, these techniques require extensive modifications to the entire DU source code, making it vendor-specific. Second, the latency of fault-tolerant key-value stores—roughly 100 µs tail latency—takes away a significant portion of the DU network functions' TTI-sized (500 µs) processing budgets. For example, the MAC layer alone may generate tens of state updates in every TTI.

Creating a new DU instance. One can consider using a general-purpose resilience mechanism provided by a cluster orchestration system such as Kubernetes [14] for DU resilience. In this approach, the orchestration system handles updates and failures by creating a new DU instance (e.g., Kubernetes brings up a new DU pod that runs the DU software stack) and starts routing traffic to the new instance. Such approaches are not designed for real-time systems, e.g., in our experiments the long startup time of Kubernetes results in UE disconnection lasting over 50 seconds (§7).

Stateless migration to a hot-standby DU. Slingshot [32] shows how the DU's PHY processing can be seamlessly migrated to another PHY without transferring PHY state between the two. This works because the PHY has no long-lived state that affects UEs. In contrast, the DU layers above the PHY (i.e., MAC and RLC) maintain long-lived UE state (e.g., RLC acknowledgement tracking). Our experiments in §7 show that simply bringing up a hot-standby DU disconnects attached UEs for over three seconds.

3 ATLAS OVERVIEW

Atlas provides a new primitive that we call "DU migration". A DU typically serves multiple cells, each connected to a separate RU. Since Atlas migrates all cells of a DU between servers during resilience events, we use the terms cell and DU interchangeably.

To provide resilience, Atlas creates an illusion of two co-located cells providing the same coverage and signal quality by exposing a single RU to two DUs—the source and destination DUs for migration. Building on this illusion, Atlas uses standard 3GPP mechanisms to move UEs between cells:

1. **Proactive migration ~ handovers**: We repurpose the handover mechanism [5] to seamlessly migrate UEs from the source cell to the destination cell.
2. **Reactive migration ~ recovery from handover failures**: We repurpose the cell reselection mechanism [6], originally designed to recover from handover failures, to quickly reattach UEs to the destination cell.

3.1 Key ideas

3.1.1 Efficient and interference-free RU sharing.

For proactive migration, the main challenge is moving UEs to the destination cell without disrupting their connections. Given the limited radio resources and the inherent 3GPP handover protocol delays, not all UEs can be migrated simultaneously. This limitation of handovers introduces a transient period during which some UEs may have moved to the destination DU, while others still remain connected to the source DU. The duration of this transient period depends on the vendor-specific handover procedure implementations (e.g., the time gap between two consecutive handovers, the number of simultaneous handovers that can be triggered, etc.). As a result, to guarantee zero downtime, the source and destination DUs must co-exist during the transient period, which presents two challenges.

(1) **Spectrum sharing.** Allowing both DUs to transmit simultaneously causes radio interference, which significantly degrades performance.
(2) **RU hardware sharing.** Today's commodity RUs support only one DU. Provisioning a spare RU for redundancy has a prohibitive cost.

We solve these problems with the following two contributions. **First**, we develop a method for sharing the wireless spectrum between the two DUs during migration with minimal interference. For user data transfers, we observe that we can configure the MAC schedulers to multiplex user data for the two DUs in the time domain by scheduling them in different TTIs. For the time-sensitive downlink control channel signals that cannot be similarly re-scheduled, we exploit their inherent robustness to interference and multiplex them in the spatial antenna domain.

Second, we develop a method for two DUs to share an RU's fronthaul link, giving the illusion that each DU communicates with a separate physical RU. We implement Atlas' RU sharing logic in a "logical" fronthaul network function (NF) that manipulates and forwards fronthaul packets to/from the RU. We provide three alternative fronthaul NF implementations, tailored to deployments with different hardware, compute, and source-code modification requirements.

3.1.2 Fast UE reconnection after failure.

How quickly can a UE reconnect to a backup DU after the primary DU's failure? We make two contributions to answer this question. **First**, we perform a detailed measurement of the timing of UE actions during cell reselection. We show that when adapted well for DU failovers, this can provide a downtime comparable to normal recovery from failed handovers, which UEs experience occasionally in normal operation.

Second, we show how existing 5G protocols agnostic to DU failures. Although UEs can quickly reattach to a second DU after losing connectivity with a first DU that remains operational, UEs cannot regain connectivity when the first DU is no longer operational. As we show in Section 5, long protocol timeouts delay the UE's initial re-attachment attempt, and subsequent timeouts completely disconnect the UE. We identify these protocol limitations along with possible fixes

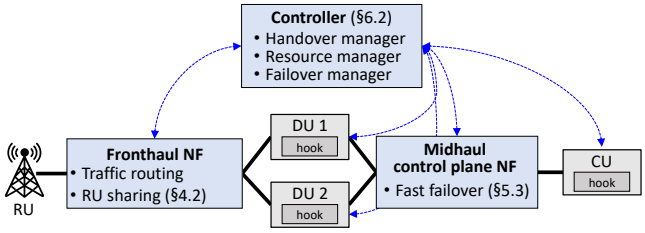


Figure 2: The overview of Atlas. Blue boxes represent the key logical components, and blue lines signify the control channels.

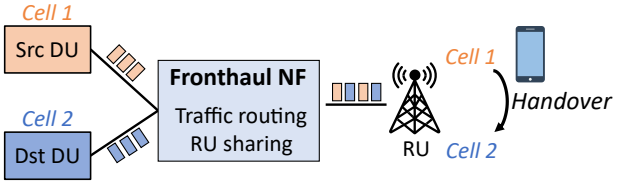


Figure 3: The high level idea of proactive migration. The source and destination DUs share the RU with Atlas' fronthaul NF. UEs are migrated via handover.

for the standards in the future. In the meantime, we design a lightweight midhaul control plane NF that interposes on the CU-DU control plane traffic to bypass these limitations.

3.2 Atlas architecture

Figure 2 illustrates a logical view of Atlas' architecture, consisting of the Atlas controller, a fronthaul NF, and a midhaul control plane NF. Several agents run on the Atlas controller; these are responsible for initiating DU migration by notifying the fronthaul/midhaul NFs, for interacting with the DU and CU through hooks that expose parameters for the scheduling of radio resources and handovers, as well as for receiving telemetry feedback about the state of the DUs and the CU. The controller communicates with the NFs and the DU through an RPC channel.

The Atlas NFs are logical entities that can run in different physical locations. The fronthaul NF modifies/forwards/-drops fronthaul packets for RU hardware sharing, and redirects the fronthaul traffic from the primary DU to the backup DU for failover. We present three implementation options for the fronthaul NF: i) inline in the DU process using userspace eBPF hooks, ii) on an existing programmable top-of-rack switch, and iii) on a software switch. The midhaul NF intercepts the SCTP connection between DUs and the CU to provide the CU with early DU failure notifications. The midhaul NF can reside in the CU, or a separate process; we currently implement the latter approach.

DU configuration. The source and destination DUs for migration connect to the same CU, and they both have the same configuration in terms of the physical layer grid (e.g., bandwidth, TDD/FDD config, and MIMO capabilities). A key difference is that the source and destination cells have distinct Physical Cell IDs (PCIs); we reserve a few PCIs for destination cells from the set of reserved PCIs that operators maintain for future cell deployment [41].

Additionally, we configure the broadcast channels carrying the Master Information Block (MIB) and System Information Block (SIB) to use non-overlapping resources in the time domain (MIB and SIB are crucial information for UEs for cell detection, synchronization, and attachment). These configurations are required for successful handovers during proactive migration. They have no negative effect on the cell's performance.

RIC integration. Our current prototype uses a custom-built controller, NFs, RPC channels and CU/DU hooks. However, we have taken care to align Atlas' design with the O-RAN RAN Intelligent Controller (RIC) specifications [11]. The Atlas controller can be implemented as a RIC xApp. The NFs and CU/DU hooks can be implemented as service models running inside E2 Nodes [10], communicating with the RIC via the E2 interface, with only minor enhancements to the existing O-RAN specs (see §6).

4 PROACTIVE MIGRATION

Proactive DU migration in Atlas uses handovers (Figure 3), achieved by serving two 5G cells—one for the source DU and one for the destination DU—via the same RU. The challenge here is simultaneously serving two 5G cells that UEs can successfully communicate with, via one RU shared between the two DUs. As already discussed in §3, triggering and executing handovers takes time, depending on the RAN implementation, leading to a transient period where some UEs have moved to the destination DU while others still remain on the source DU. By sharing the RU between the DUs, we can prevent UE connection disruptions during this period. However, the current 5G vRAN stack is designed assuming a 1:1 DU–RU relationship. For example, the RU is configured with one destination fronthaul address (e.g., IP or Ethernet) of its peer DU, and it expects to receive an xRAN protocol-compliant stream of packets from the DU. Similarly, the DU's MAC scheduler and PHY layer expect signals to be sent/received via the entire RU, assuming exclusive use.

We begin by describing two intuitive RU sharing approaches—time sharing and frequency sharing—and discuss their limitations. We then present our insight of sharing RU *antenna ports* in combination with time sharing. Finally, we discuss how our RU sharing approach fits within the xRAN fronthaul protocol constraints, and by our requirement to be vendor-agnostic. We focus our discussion on the downlink, and briefly summarize the uplink direction in §4.3.

4.1 RU resources available for sharing

There are two intuitive RU resources that the source and destination DUs can share: time and frequency. We discuss below why sharing these resources does not work or is inefficient. As a piece of background, note that the DU's MAC scheduler makes scheduling decisions once every TTI (lasting for 1ms or less in 5G) [53], and over-the-air signal exchanges happen once every symbol. Each TTI duration contains multiple symbol duration (e.g., 14 in a typical 5G configuration).

Time sharing. Is it possible to simply time-share the RU? For example, our fronthaul NF could allow source and destination DUs to communicate with the RU in alternating TTIs. This can be done, e.g., by dropping the source DU's downlink fronthaul packets in even-numbered TTIs, and the destination's in odd-numbered TTIs. However, this approach is infeasible because both DUs have to send crucial and time-sensitive control channel information for their Physical Downlink Control Channel (PDCCH) in every TTI. In the symbol that one DU sends PDCCH, we must drop either the other DU's PDCCH signals, or its user data signals; the latter could contain important RRC signaling. We implement this approach with multiple optimizations (e.g., using non-overlapping PDCCH symbols for the two DUs) and find that UEs either cannot connect or get near-zero throughput.

Frequency sharing. Can we split the frequency spectrum available to the RU between the source and destination DUs? For example, during migration, the source and destination DUs could use the lower and upper 50 MHz halves of a 100 MHz RU, respectively. Unlike time sharing, we find that frequency sharing is feasible through the 3GPP mechanism of Bandwidth Parts (BWP) [37]. BWP allows dividing a single 5G carrier into multiple segments, each of which can be assigned to a different service or, in our case, to a different DU. Unfortunately, BWP is an optional 5G feature that some UEs and vRAN software may not support to minimize their complexity and cost, limiting the generality of this approach.

Furthermore, even in scenarios where all participants support BWPs, statically splitting the bandwidth between source and destination DUs could degrade user performance. During the start of the migration, more UEs are connected to the source DU, which thus has higher radio resource requirements compared to the destination DU. As UEs are gradually moved to the destination DU, the traffic load and the radio resource requirements also shift. Considering that the migration period can be non-negligible when a cell serves numerous UEs, such a capacity crunch is not acceptable.

4.2 RU sharing in Atlas

Given that the obvious resource-sharing approaches are not applicable, we investigate whether one of them can be enabled once we remove its limitations. We observe that time sharing is feasible if we solve the problem of control channels collocated over the same TTI. In Atlas, we solve this problem by observing that a third, less obvious resource can be shared between the two DUs in addition to time and frequency: the spatial resource, arising from the RU's multiple antenna ports. To simplify the discussion below, we use the terms "antenna ports", "antennas", and "spatial streams" interchangeably, though, in practice, their mapping is a complex process determined by the DU configuration [36]. For each antenna port, the RU expects to receive (at most) one packet from the DU for every symbol duration. In the following, we explain how, by using the spatial domain, we can bypass the limitation of RU sharing for control data. With this problem solved, we present our solution for time-sharing the RU between the source and destination DUs.

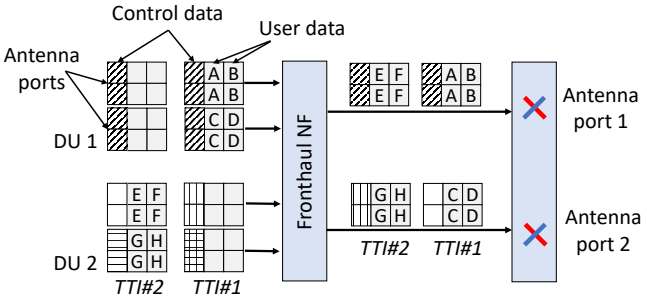


Figure 4: Atlas' RU sharing. Each 2×3 grid signifies two frequency resource blocks and three symbols. Checkered boxes contain control signals (spatially shared); lettered boxes contain data signals (time-shared).

4.2.1 *Antenna port sharing for control channel data.*

We use the following RU sharing approach for control channel data in Atlas: When both DUs send fronthaul packets with control signals in the same symbol, Atlas splits the antenna ports and maps the signals to non-overlapping subsets of ports as illustrated in Figure 4. The symbols carrying control signals are known a priori to Atlas, since they are statically configured for all cells at deployment time and source and destination cells have the same configuration.

Naturally, transmitting the control signals of both DUs at the same time (possibly with some spatial streams dropped) leads to destructive interference. Our choice to adopt this approach, despite the interference, is driven by the following observations: The downlink control channel carries two broad types of messages: i) data-related (e.g., UE scheduling decisions, HARQ feedback), and ii) non-data-related (e.g., paging, group UL power control). We observe that data-related control messages from the two DUs do not interfere because they implicitly fall under Atlas' user data time sharing, i.e., for downlink, they are sent at the same TTI as the corresponding data; for uplink, they are sent in a previous Atlas-controlled TTI. Non-data-related control messages do not fall under the

same time-division scheme and can interfere, but they are i) relatively rare, and ii) highly robust (QPSK with high code rate, 24 CRC bits), because they target UEs with unknown signal quality. Given this, we expect interference for edge users to be minimal and not worse than the interference experienced by the existence of neighboring cells.

Antenna sharing can be implemented with near-zero overhead using simple forwarding rules, making it cost-efficient and easy to implement. It should be noted that to handle the DUs' inability to share spectrum for control channels, we also considered an alternative approach of combining the two DU's downlink fronthaul packets in our fronthaul NF, by summing up their IQ samples. However, we opted for our current solution, because, while the latter works, it adds latency overhead and cost, since it cannot be implemented inline in the hardware switch or the DU. Instead, it requires a software switch that, as we show in §7.4, has an order of magnitude higher latency compared to the other two implementation approaches.

4.2.2 Time sharing for user data and broadcast channels. Unlike control channel data that is low-rate and designed to be robust, user data may use MIMO for high-performance transmissions, which is significantly more prone to interference. For example, a typical transmission of user data could use 4 layers MIMO with 256QAM, while control signal transmissions typically use diversity gain SISO with QPSK and high code rates. If we use the same spatial multiplexing technique as in the case of control channels, and we drop spatial streams of user data from the other DU, this can result in significant performance degradation. Furthermore, user data may be multiplexed with broadcast messages (MIB/SIB). If we opt to drop the broadcast data, UEs will lose synchronization and will be disconnected (or will fail to attach).

We overcome this problem by using time-sharing for user data. Using dynamic RIC policies, we configure the MAC layer of the source and destination DUs to schedule user data in symbols of non-overlapping TTIs. Time sharing allows us to dynamically adjust the resource allocation on the fly to avoid capacity crunches during the migration period. Atlas collects telemetry data from the DUs (utilization of radio resources) and changes the TTI allocation ratio accordingly (see more details in §4.4). Our fronthaul NFs work synchronously, ensuring that only one DU is allowed to send downlink data in all the symbols of each TTI. The scheduling policy is configured to guarantee that DUs will send downlink data whenever they have broadcast messages (configured in non-overlapping TTIs, as explained in §3).

4.3 xRAN protocol-compatible RU sharing

We have described a high-level view of Atlas' RU sharing. We now show how to make our approach compatible with the standard xRAN fronthaul protocol. For simplicity, we omit some details such as xRAN "control plane" packets (distinct from the control channel signals discussed above).

Algorithm 1: Atlas fronthaul NF

```
Input: An xRAN packet pkt.
Output: Traffic steering decision.
if pkt is an uplink packet then
    return duplicate;
else if pkt is a downlink control plane packet then
    if pkt is from source DU then
        if pkt is for antenna port 0 then
            return forward
        else return drop ;
    else // pkt is from destination DU
        if pkt is for antenna port 0 then
            return remap_to_1_and_forward
        else return drop ;
    end
else // downlink user plane packets
    if pkt is received in the allocated time then
        return forward
    else return drop ;
end
```

In every symbol duration, the DU and RU exchange xRAN packets as follows. For every antenna port, the RU receives one downlink packet from the DU, and generates one uplink packet for the DU. Violating this causes undefined RU behavior, affecting user connectivity. To share the RU among two DUs while following the xRAN protocol, we need to carefully handle xRAN packets. We realize it as follows (Algorithm 1):

Time alignment. The xRAN packet sequences of the two DUs must be time-aligned . For this, we time-synchronize the DUs with the RU using the Precision Time Protocol (PTP).

Uplink packets. For any given symbol, each DU must receive an uplink packet from the RU. To enable it, Atlas duplicates and sends uplink packets to both DUs. This works because although a DU's PHY layer may receive signals meant for the other DU, this is no different from normal interference. The vRAN layers naturally filter out only the desired information via the MAC scheduling decisions and standard error checking such as forward error correction.

Downlink packets. For any given symbol and antenna, the RU must receive only one downlink packet, either from the source or destination DU. As discussed in §4.2, our fronthaul NF is aware of the type of signals (i.e., control or data) in each packet. The NF parses the downlink packet headers to extract the information, including the TTI and symbol number, and antenna ID (called extended Antenna Carrier (eAxC) [9] in xRAN parlance). In our configuration, the source DU always uses antenna port 0 for its critical downlink signals (i.e., downlink control channel). During the migration process, the NF remaps the destination DU to use antenna 1 for its critical downlink signals by overwriting the antenna ID field

in the packets from the destination DUs carrying such signals. This can be trivially extended for cell configurations with downlink signals using more than one antenna port.

4.4 Adaptive resource allocation

We further optimize our DU migration to improve resource utilization and minimize the impact on UEs. First, Atlas dynamically adjusts the time slots allocated to each DU, using telemetry collected from the DUs about the resource block (RB) utilization. The TTIs are allocated to the DUs proportionally based on their average RB utilization over a small window (e.g., 10ms). Moreover, when choosing UEs to migrate, Atlas prioritizes inactive UEs with low data rates and waits active UEs to complete their transmission. This ensures that the impact on UEs' activities will be minimized, e.g., due to interference on the downlink control channel or packets dropped during the handover.

5 REACTIVE MIGRATION

Atlas handles unplanned resilience events (i.e., DU software or hardware failures) differently from proactive migration, for two reasons. First, proactive migration requires the original DU to participate in handovers, which is not possible when it has crashed. Second, failures are much less frequent than planned events, so the RAN can afford a short downtime, e.g., comparable to handover failures that UEs experience occasionally during normal operation.

Reactive migration ~ handover failure recovery. Recall that our approach in Atlas is to use existing mechanisms for wireless resilience. For reactive migration (i.e., failover), we use the mechanism that UEs use to recover from handover failures. When a primary DU fails, Atlas quickly pairs the RU with a backup DU on a different server. Since today's O-RAN RUs lack the functionality to quickly re-route the fronthaul traffic to a different DU, we use the fronthaul NF for this purpose.

To the UE, this appears as if it has lost the signal from one cell and entered the coverage area of a different cell. This is identical to a phenomenon called "too-late handover" that happens, for example, when a mobile UE turns a sharp corner in a city. Since the change in coverage happens abruptly, the network has no time to smoothly handover the UE between cells, requiring the UE to reconnect to the new cell.

5.1 High-level approach

To tolerate the failure of a DU server in an edge datacenter with N primary DUs (one per server), Atlas maintains a shared backup DU on a separate server that can take over if any of the N DUs fail. Since it takes a long time (over 40 s in our case, see §7) to bring up a new DU from scratch, the backup DU is active and ready to take over. During failure-free operation, the backup DU performs no work (e.g., PHY signal processing or MAC scheduling), so it can run in a low power mode.

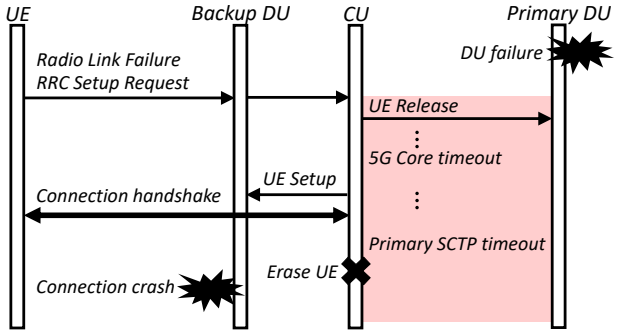


Figure 5: High-level message flows during primary DU failover without Atlas.

When a primary DU fails, Atlas' fronthaul NF starts forwarding fronthaul traffic for the RUs associated with the failed DU to the backup DU. This brings up the cell served by the failed DU back online before affected UEs begin a process called "cell reselection" to regain connectivity. The UE notices that its wireless signal to the primary DU is not working when it repeatedly fails to decode downlink signals it expects from the primary DU. After the UE's Radio Link Failure (RLF) timer expires, the UE searches for the best available cell and tries to reconnect to it. Since Atlas' backup DU is online at this time with the same signal strength as UE's previous cell choice, the UE chooses the backup DU's cell.

Comparison with handover failure recovery. Our key finding is that this approach provides a downtime comparable to the UE's time to recover from handover failures. We argue that Atlas' reactive migration downtime is therefore acceptable since handover failures are far more likely than DU failures. We expect DU failures to happen at most a few times per year. In comparison, Li et al. [34] report that walking UEs in a commercial LTE network experience handovers once every 70 s; approximately 1 % of handovers fail, resulting in handover failures every ten minutes. Li et al. [33] report handovers every 50 s while driving in a 5G network, off which 1.1 % fail as too-late handovers.

The Radio Link Failure timeout is network-configurable. In our testbed, we set it to 5G's lowest possible value (250 ms) to emulate reliable 5G networks without coverage holes.

5.2 Lack of DU failure awareness in vRAN

The main challenge in making failovers quick is that existing vRAN protocols, specifications, and software implementations were not designed with DU failures in mind. We believe that this is because historically, RANs have used specialized

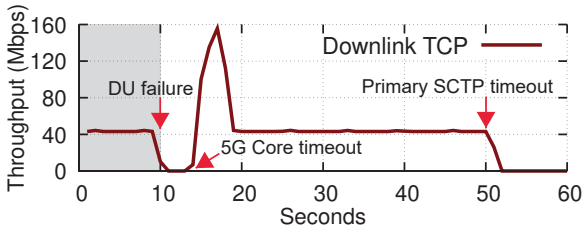


Figure 6: Downlink TCP throughput changes during DU failover implemented by simply re-routing fronthaul traffic to a backup DU.

hardened appliances that are less prone to failures. While one would expect that simply re-routing the RU's fronthaul link to the backup DU would suffice, we find that this approach results in (1) an initial downtime lasting over 3 s, and (2) complete disconnection of UEs after some time. Figure 6 shows these disruptions for a downlink TCP transfer with one UE in our testbed (§7). We discuss the causes below.

For background, the control plane protocol between the CU and DU is called the "F1 Application Protocol" (F1AP) [2], which uses the Stream Control Transmission Protocol (SCTP) as its underlying transport. Figure 5 shows a UE trying to reattach via the backup DU, using an RRC message sequence. The CU treats this a normal cell reselection attempt.

Problem 1. Since the CU is unaware of the primary DU's failure, on receiving the UE's re-attachment request via the backup DU, it sends an F1AP request to the primary DU to release the UE's connection. The primary DU cannot respond to this request since it has crashed, but the CU keeps waiting for a response. Progress can be made only after the 5G core (Figure 1) triggers a timeout (3 s in our case), and allows the CU to establish the UE's connection on the backup DU without releasing it on the primary DU.

Problem 2. The CU later learns about the primary DU's failure when their F1AP session's underlying SCTP connection times out, which takes 30 s in our case. The F1AP protocol [2] does not clarify how the CU should react in this case. In the vRAN implementation that we use, the CU handles the primary DU's SCTP timeout by deleting all UEs that were attached to the primary DU. This disconnects any of these UEs that had successfully reconnected via the backup DU.

Before describing Atlas' failover handling in detail, we discuss the limitations of two straightforward approaches to tackle the above problems.

F1AP reset messages. Although F1AP supports "reset" messages allowing the DU to notify the CU of failures, these seem designed for only partial failures. For example, management threads in a DU with irrecoverable errors limited to its MAC layer could send such messages. A full DU failure (e.g., due to a server crash) does not allow the DU to send such messages.

Timeout tuning. While it may be possible to tune the 5G core or SCTP timeouts to reduce the downtime, doing so would affect failure-free operations which we wish to avoid. Lowering timeouts to hundreds of milliseconds is dangerous since such low timeouts can be triggered even during normal operation, such as when the CU server or a backhaul network is overloaded. A single CU or 5G core may serve thousands of remote DUs at various cell sites with different latency and bandwidth congestion, and we do not want to tune the timeout for each DU. (Even intra-datacenter transports employ retransmission timeouts of hundreds of milliseconds, and connections time-out only after tens of retransmissions).

5.3 Adding DU failure awareness to vRAN

To overcome the above limitations, we design a midhaul network function and a failover manager running in our controller (Figure 2), which provide low-latency notification of DU failures to the CU. The NF interposes on the SCTP protocol between the CU and the DUs. For each DU, the NF creates separate SCTP connections with the CU and the DU, and transparently relays messages between the two during failure-free operation. When the primary DU fails, the failover manager detects the failure by noticing the lack of heartbeats from the DU, and notifies the midhaul NF.

How can our midhaul NF quickly indicate the DU failure to the CU? One could consider sending an F1AP reset message to the CU. However, since F1AP reset messages must contain the list of UEs affected by the failure, our NF would need to decode all F1AP messages, which is infeasible in cases where F1AP messages are encrypted. Instead, our NF simply closes its SCTP connection to the CU that it created for the primary DU, which quickly signals the CU that the DU has failed. The CU then releases all UEs associated with the primary DU. This happens *before* the UEs try to reattach, avoiding the two problems described above.

Since the midhaul NF interposes on only the delay-tolerant low-bandwidth CU-DU control plane (roughly 10 Kbps per UE), it can scale to a large number of DUs and UEs. In addition, since this NF does not handle realtime data, existing NF resilience techniques (e.g., ECHO [40]) can make it resilient.

5.4 Failover steps

To complete the picture, we break down Atlas' failover process. For simplicity, we use average numbers for the various phases, measured with one UE in our testbed (§7).

(1) **T = 0 ms**: The primary DU crashes.
(2) **T = 50 ms**: Our failure manager notices the lack of fronthaul packets from the primary DU, and notifies the fronthaul and midhaul NFs. The former re-routes the affected RU's fronthaul to the backup DU, and notifies the midhaul NF of the primary DU's failure.
(3) **T = 150 ms**: The midhaul NF closes the primary DU's SCTP connection to the CU.

(4) **T = 600 ms**: The UE declares Radio Link Failure, and tries to reconnect to the backup DU. The CU installs the UE's context at the backup DU, restoring the UE's connectivity.
(5) **T = 700 ms**: The UE can exchange data via the backup DU, completing the failover process.

Importantly, it takes only around 700 − 600 = 100 ms for the vRAN to re-establish the UE's connection after the RLF timeout. This is because much of the UE's state elements, including its connection contexts and authentication keys, do not need to be reconstructed since they are maintained in the higher layers of the protocol stack (i.e., the CU and core network). During reactive migration, these get reused to quickly restart the UE's connection at the backup DU. For comparison, it takes approximately 950 ms instead of 100 ms for the UE to connect from scratch after the vRAN receives the UE's RRC setup message.

6 IMPLEMENTATION

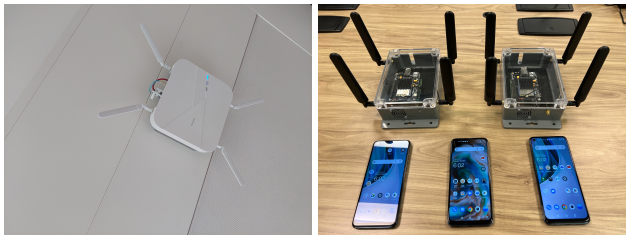
6.1 Atlas network functions

Atlas fronthaul NF. To demonstrate the generality and portability of our design, we implement the Atlas fronthaul NF on three different targets. First, we implement it as a software middlebox for x86 CPUs in ≈ 4500 lines of C++, using DPDK [25] for low latency.

Second, we implement it in a distributed manner on each DU using the Janus RAN programmability framework [22], which allows the introduction of hooks in the RAN CU/DU functions, for injecting sandboxed userspace eBPF code [18, 27]. Using Janus, we introduce a hook at the xRAN layer of our PHY software (Intel FlexRAN [26]), and load an eBPF code (≈ 500 lines of C), which intercepts the downlink xRAN packets and implements the steering decisions described in §4.3. In addition, our top-of-rack (ToR) switch mirrors uplink packets received from the RU to both DUs using the traffic mirroring feature supported by commodity Ethernet switches (e.g., port mirroring in Arista EOS [23]).

Third, we implement it for an Intel Tofino switching ASIC in ≈ 600 lines of P4-16 [43], and deploy it on a programmable ToR switch. We find that the fronthaul NF logic can be implemented using only a single match-action table by effectively leveraging the range-matching feature. This implementation can process the fronthaul traffic at line rate (e.g., 6.5 Tbps using a 32-port switch).

Midhaul control plane NF. The Atlas midhaul NF is a lightweight application running on the same server as the CU. It communicates with the CU-CP and DU control planes using Linux SCTP sockets. During normal operation, it simply forwards CU-DU F1AP messages. Upon receiving a DU failure notification from the Atlas controller, it notifies the CU of this failure by closing the corresponding SCTP socket.



(a) RU installed on ceiling **(b) Five UEs**

Figure 7: RU and five UEs used in our evaluation.

6.2 Atlas controller

The Atlas controller orchestrates migration by managing the fronthaul and midhaul NFs via their corresponding control APIs, e.g., Barefoot Runtime RPC APIs for a P4-based fronthaul NF, and via gRPC with our midhaul NF. It also interacts with the CU and DU using Janus control and telemetry hooks that have been integrated in the commercial-grade CU and DU software of Capgemini [19].

Handover and resource manager. First, the handover manager uses a non-realtime hook in the CU to trigger handovers. Second, our resource manager uses a hook in the DU's MAC scheduler to implement RU time sharing for user data. We also use telemetry information from the CU's RRC layer to prioritize UEs during migration.

Failover manager. The failover manager uses DU telemetry probes in the PHY layer to detect hardware and software crashes. The probe sends heartbeats to our controller in every 500 µs TTI duration while the DU is alive. If heartbeats are not received for a timeout duration (set conservatively to 20 ms), our controller notifies the fronthaul and midhaul NFs to trigger DU failover.

It should be noted that our current implementation relies on Janus for realizing the CU/DU hooks, due to the flexibility that it offers for changing the control and telemetry logic on-the-fly. However, the same functionality could also be implemented using existing O-RAN E2 service models and 3GPP specifications, with minor enhancements. Specifically, the handover functionality could be implemented using the traffic steering E2 service model [31] and the required telemetry information could be obtained using the KPM [12] and Network Information [8] service models. Finally, the per-TTI scheduling of the radio resources between the two DUs could be achieved using the 3GPP slicing parameters exposed in [1], using a service model similar to the one described in [28].

7 EVALUATION

Testbed hardware. We evaluate Atlas on a commercial-grade 5G vRAN testbed (Figure 7). Our testbed uses ceiling-mounted O-RAN RUs from FoxConn, with 100 MHz 4x4 MIMO operating at 3.5 GHz (Figure 7a). We use three HPE Telco DL110 servers, each with one Intel Xeon 6338N CPU,

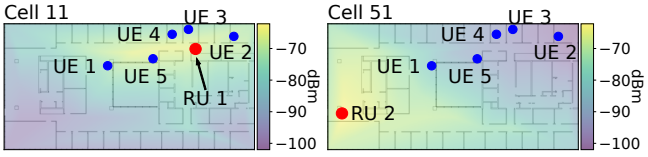


Figure 8: Radio signal coverage and UE placement in our testbed with two 5G cells. Red and blue circles indicate the location of radio units and UEs, respectively.

an Intel E810 100 GbE NIC, and an Intel ACC100 accelerator for PHY forward error correction. The RU and servers are connected via a 100 GbE Arista 7170 Tofino-based P4 switch, and synchronized with a Qulsar QG2 PTP grandmaster clock. There are five UEs, which are a mix of Raspberry Pis with Quectel RM502Q-AE modems, and OnePlus Nord N10 5G phones (Figure 7b).

Software. We use Intel FlexRAN v22.03 for the PHY, CapGemini's 5G stack for the higher DU and CU layers, and Metaswitch's Fusion 5G Core. The servers run real-time Linux kernel v5.15. We run the primary and secondary DUs and the CU on different servers. For proactive migration, we start the destination DU only when needed. For reactive failovers, we keep a backup DU always running as a hot standby.

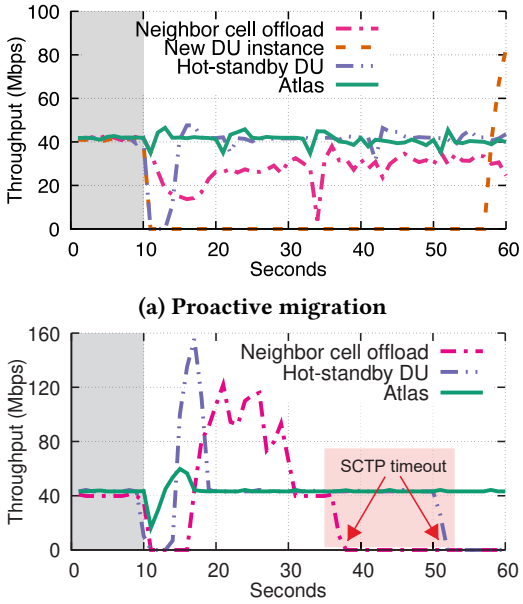
Cell configuration. We use the two RUs and five UEs in our building to emulate a realistic deployment. We deploy two cells with PCIs 11 and 51, named "cell 11" and "cell 51", respectively. Figure 8 illustrates the radio signal coverage and UE placement in our testbed.

7.1 Comparison with baselines

We use one UE in this section for clarity, and show results with multiple UEs in later sections. In the experiment, we use iperf to send downlink TCP traffic with a target rate of 40 Mbps and measure the throughput once every second. We compare Atlas with three baselines for vRAN resilience:

(1) **Neighbor cell offload.** Although offloading UEs to a neighboring cell may not always be possible due to DU centralization (§2.4), we evaluate it for completeness. We offload the UE from cell 11 (served by DU 1) to cell 51 (served by DU 2) as follows. For proactive migration, we use a CU-triggered handover. For failover, we kill DU 1 and let the UE re-attach to DU 2.

(2) **Stateless migration to a hot-standby DU.** We use only one RU (cell 11) in this case. Both DUs connect to Atlas' fronthaul NF, which initially forwards traffic between RU 1 and DU 1. For proactive migration, we steer the fronthaul traffic from the RU to DU 2. For failover, we kill DU 1 and then immediately steer the fronthaul traffic to DU 2.



(a) Proactive migration

(b) Failover

Figure 9: Performance of Atlas' proactive migration and failover, compared to the three baselines. The resilience event happens at t = 10 seconds.

(3) **Creating a new DU instance.** Only cell 11 is used. For both proactive and reactive migration, we kill DU 1, and immediately launch a new DU on the same server.

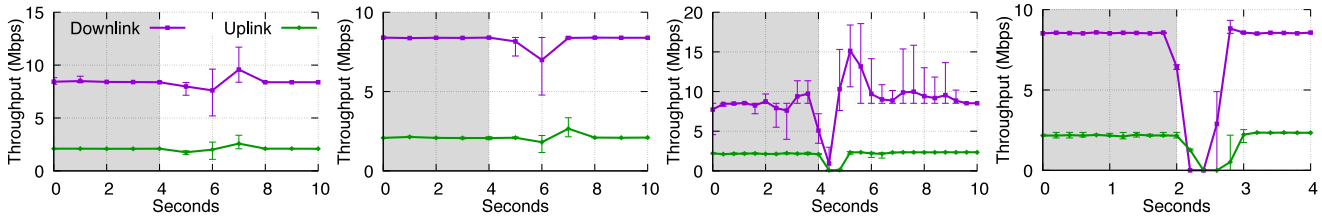
Figure 9 shows the results. For proactive migration, Atlas seamlessly moves the UE to the new DU with no visible connectivity disruption, other than minor TCP rate variations that also happen in normal operation. In comparison, offloading to the neighbor cell results in a persistent throughput drop of around 25%, because RU 2 is farther and has worse signal quality than RU 1 for the UE (UE #5). The other approaches—stateless migration to a hot standby DU, and creating a new DU instance—result in complete disconnection for around 3 s and 45 s, respectively.

For reactive migration during failover, Atlas maintains non-zero TCP throughput for every second interval. With the neighbor cell offload and stateless migration approaches, we observe UE disconnections, shown by the red arrows in Figure 9b. As discussed in §5.2, the UE completely disconnects from the network at around 30–40 s after failure, caused by the CU deleting the primary DU's UEs when their SCTP connection times out. Results for the "new DU instance" approach are identical to those for proactive migration, and are omitted for brevity.

7.2 Multi-UE performance

We next evaluate Atlas' performance using five UEs.

TCP and UDP throughput. We measure the five UEs' UL (uplink) and DL (downlink) throughput changes during DU



(a) Proactive migration (TCP) (b) Proactive migration (UDP) (c) Failover (TCP) (d) Failover (UDP)

Figure 10: Average, min, and max TCP/UDP throughput on downlink/uplink with five UEs. Migration happens at t = 4 seconds in (a)-(c) and t = 2 seconds in (d).

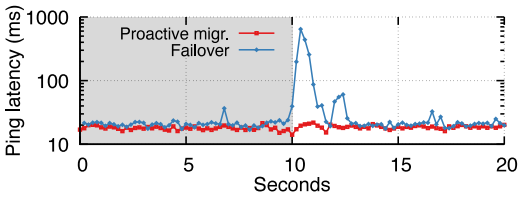


Figure 11: Average RTT of five UEs during proactive migration and failover.

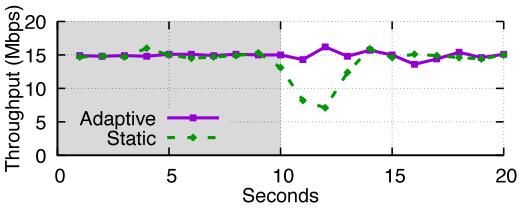
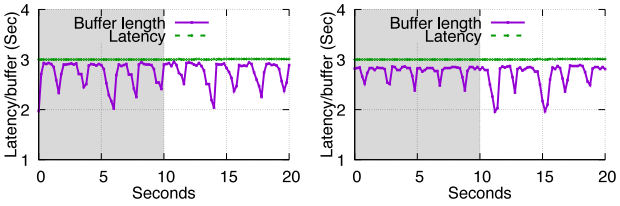


Figure 12: Average throughput of five UEs with adaptive and static resource allocation.



(a) Proactive migration (b) Failover

Figure 13: Performance of low-latency live streaming with 3s target latency.

migration when they all concurrently transmit TCP or UDP traffic. All UEs initially connect to cell 11, and run iperf with a target rate of 8 Mbps for DL and 2 Mbps for UL. Figure 10 shows the average throughput of all UEs during the migration, with the minimum/maximum throughput across UEs.

We make the following observations. Overall, Atlas' migration techniques have little impact on the UEs' connectivity and throughput. For proactive migration, both the TCP and UDP throughput remain largely unaffected. There are small throughput variations caused by the migration event, but these are comparable to natural throughput variations caused by the noisy nature of wireless channels.

For failover, the TCP throughput drops to zero, but for only 600 ms and 850 ms for DL and UL, respectively. As discussed in §5.1, this is comparable to the recovery timeout from handover failures (e.g., 670 ms in an LTE study by Li et al. [35]). After failover, the throughput returns to normal, with high variation caused by TCP buffering lasting approximately 5 s. The UDP throughput drops to zero for only 750 ms.

Ping latency. Figure 11 shows the average RTT across the five UEs to a remote server during migration, measured every 200 ms. Proactive migration has no visible impact on ping latency. For failover, the RTT increases to at most 650 ms for less than one second and then returns to normal values.

Adaptive time slot allocation. We evaluate the effectiveness of Atlas' adaptive resource allocation (§4.4) by comparing it with a static allocation scheme that statically partitions TTIs evenly between the source and destination DU. In this experiment, we send 3 Mbps UL TCP traffic from each of the 5 UEs (15 Mbps overall), while the total cell capacity is 20 Mbps. Figure 12 shows that with static resource allocation, the total UE throughput drops below 10 Mbps during migration, since each cell statically gets half of the radio resources. With the adaptive allocation, the throughput remains unaffected since Atlas gradually shifts capacity from cell 11 to cell 51 as more UEs are migrated.

7.3 Real-world applications

We use live video streaming as an example to evaluate how well Atlas preserves connectivity for low-latency applications. Our experiments show that Atlas migrates the DU with negligible impact on video streaming. Our experiment uses the DASH.js framework, which plays low-latency streams and reports live latency, video bitrates, and video buffer sizes [21]. We set the target latency to 3 s, with which the DASH player tries to stream the video while staying at most 3 s behind the live event. We use framework-recommended values for other settings, resulting in a 6 Mbps video bitrate.

Figure 13 presents the lag behind the live event, and the server's buffer size during proactive migration and failover. We find no significant change in either metric. The video

quality and fluency are stable during the migration, as the video buffer is never fully drained.

7.4 Microbenchmarks

Migration time. We measure the time that Atlas needs to complete DU migration with different numbers of UEs. Overall, Atlas can complete the migration within a few seconds. For proactive migration, Atlas takes 0.5 s for one UE, which increases linearly with the number of UEs to 1.3 s for five UEs. Much of this time comes from a limitation of the DU implementation used in our experiments, which allows triggering only one handover every 0.5 seconds. This time will decrease by an order of magnitude with more optimized handover triggers (e.g., multiple simultaneous handovers) since handovers take <100 ms to complete. A typical number of users that are RRC-connected to a cell is 1–25 [20], so all UEs will migrate in a few seconds. In addition, each UE retains connectivity for most of this duration since the over-the-air actual handover signaling lasts for only around 20–50 ms. During failovers, the time to re-attach one and five UEs is 0.7 s and 0.9 s, respectively.

CPU and latency overhead. Atlas' components add little CPU overhead and latency. Our eBPF hooks in the PHY, MAC, and RRC layer add at most 2 µs per TTI. This is negligible for the non-realtime RRC layer, and only 0.4 % of the PHY and MAC's 500 µs TTI budget.

The overhead of our fronthaul NF depends on the implementation used. The P4 switch-based implementation adds no additional latency beyond the switch's 800 ns port-to-port forwarding latency, which is negligible compared to the xRAN fronthaul's 100 µs latency budget. If implemented in a software switch, the NF adds up to 10 µs. A distributed implementation based on eBPF hooks in the PHY also adds negligible latency (0.8 µs).

We have not optimize our midhaul NF for performance yet, because it interposes on the delay-tolerant control plane between the CU and DU, which carries little traffic (e.g., compared to the data plane). Our Linux sockets-based implementation adds under 100 µs per message.

8 RELATED WORK

Supporting resilience in cellular networks. Existing literature studies offloading UEs to neighboring cells to make cellular networks more resilient [45, 51, 52]. Yang et al. [52] design a way to estimate the service impact on UEs with hypothetical cellular-tower outages when multiple cells exist around the UEs. Concord [45] and Magus [51] propose solutions to minimize the disruption due to network upgrades by carefully coordinating neighboring cells. These approaches are constrained by traditional RAN deployments having only one DU per cell site, with no backup DUs for the cell site's radio. In contrast, Atlas develops the necessary techniques to use the multiple servers available to handle a cell site in vRAN deployments. The result is a design that works without needing inter-cell coordination or coverage from neighboring cells, and has less impact on UE performance since it does not affect signal quality.

Another line of work replicates the computation state of cellular network functions for high availability [29, 40]. For example, ECHO [40] replicates the state of an LTE core network to an external key-value store. We believe that it is infeasible to apply such approaches to DUs due to their stricter real-time requirements. Slingshot [32] migrates the DU's stateless PHY processing to another server, but it does not handle the DU's higher stateful layers.

RU sharing/virtualization. Picasso [24] proposes a new RF front-end design that enables simultaneous transmission on arbitrary spectrum fragments with a single RF front end and antenna. Although such features would be useful for vRAN resilience, they are not available in current commercial RU designs. HyDRA [30], SVL [50], and Mendes et al. [39] multiplex the RF front-end in the frequency domain, based on FFT or filterbank, creating virtual RF front-ends using isolated spectrum bands. However, more work is needed to realize these techniques in real vRANs setups. NRflex [16] uses Bandwidth Parts (BWP) for slicing radio resources, which is an optional 5G feature. In contrast, Atlas can be implemented in the existing vRANs stack without requiring such features.

9 CONCLUSIONS

We have shown how Atlas solves a key problem in the journey toward resilient vRANs: making the real-time black-box DU upgradeable and fault-tolerant. The key insight is to repurpose existing cellular resilience mechanisms for software resilience. To accomplish this, we develop a novel method for sharing an RU between two DUs by exploiting the spatial antenna dimension; and we address limitations in existing 5G protocols arising from their failure-agnostic nature. Experiments in a commercial-grade 5G vRAN testbed show that Atlas handles DU upgrades and failures with little connectivity disruption, and requires few vRAN modifications. We believe that these techniques form a practical basis for future resilient vRANs.

ACKNOWLEDGMENTS

We sincerely appreciate the anonymous shepherd and reviewers for their insightful comments and suggestions. We thank Bozidar Radunovic and Victor Bahl for their help and feedback on this project. This work was supported in part by the Open Networks Programme within the UK Department for Science, Innovation and Technology, CNS-1955422, CNS-2214272, and a Google Ph.D. Fellowship.

REFERENCES

[1] 2020. 5G Network Resource Model (NRM) (3GPP TS 28.541 version 16.6.0 Release 16). https://www.etsi.org/deliver/etsi_ts/128500_128599/128541/16.06.00_60/ts_128541v160600p.pdf.

[2] 2020. F1 Application Protocol (F1AP) (3GPP TS 38.473 version 15.8.0 Release 15). https://www.etsi.org/deliver/etsi_ts/138400_138499/138473/15.08.00_60/ts_138473v150800p.pdf.

[3] 2022. The Journey to a Cloud-Native, Fully Software-Defined vRAN Architecture. https://www.vodafone.com/sites/default/files/2022-12/journey-to-cloud-native-fully-software-defined-vran-architecture.pdf.

[4] 2023. OpenAirInterface. https://gitlab.eurecom.fr/oai/openairinterface5g.

[5] 3GPP. 2018. Procedures for the 5G System (3GPP TS 23.502 version 15.2.0 Release 15). https://www.etsi.org/deliver/etsi_ts/123500_123599/123502/15.02.00_60/ts_123502v150200p.pdf.

[6] 3GPP. 2020. User Equipment (UE) Procedures in Idle Mode and in RRC Inactive State (3GPP TS 38.304 version 15.6.0 Release 15). https://www.etsi.org/deliver/etsi_ts/138300_138399/138304/15.06.00_60/ts_138304v150600p.pdf.

[7] Kazi Main Uddin Ahmed, Manuel Alvarez, and Math H. J. Bollen. 2020. Characterizing Failure and Repair Time of Servers in a Hyper-Scale Data Center. In *IEEE PES Innovative Smart Grid Technologies Europe, ISGT Europe 2020, Delft, The Netherlands, October 26-28, 2020*. IEEE, 660–664. https://doi.org/10.1109/ISGT-Europe47291.2020.9248891

[8] ORAN Alliance. 2020. O-RAN Near-Real-time RAN Intelligent Controller E2 Service Model (E2SM), RAN Function Network Interface (NI) 1.0. *ORAN-WG3.E2SM-NI-v01.00* (2020).

[9] ORAN Alliance. 2022. Control, User and Synchronization Plane Specification. *O-RAN Fronthaul Working Group, ORAN-WG4.CUS.0-v10.00* (2022).

[10] ORAN Alliance. 2023. E2 Service Model (E2SM). *O-RAN Fronthaul Working Group, O-RAN.WG3.E2SM-R003-v03.00* (2023).

[11] ORAN Alliance. 2023. Near-RT RIC Architecture. *O-RAN Fronthaul Working Group, O-RAN.WG3.RICARCH-R003-v04.00* (2023).

[12] ORAN Alliance. 2023. O-RAN E2 Service Model (E2SM) KPM 3.0. *O-RAN.WG3.E2SM-KPM-R003-v03.00* (2023).

[13] Altiostar. 2021. Altiostar and Rakuten Mobile Demonstrate Success Across Performance and Scalability for Open RAN Network. https://www.prnewswire.com/news-releases/altiostar-and-rakuten-mobile-demonstrate-success-across-performance-and-scalability-for-open-ran-network-301254947.html.

[14] The Kubernetes Authors. 2023. Kubernetes. https://kubernetes.io/.

[15] Robert Birke, Ioana Giurgiu, Lydia Y. Chen, Dorothea Wiesmann, and Ton Engbersen. 2014. Failure Analysis of Virtual and Physical Machines: Patterns, Causes and Characteristics. In *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. 1–12. https://doi.org/10.1109/DSN.2014.18

[16] Karim Boutiba, Adlen Ksentini, Bouziane Brik, Yacine Challal, and Amar Balla. 2022. NRflex: Enforcing Network Slicing in 5G New Radio. *Computer Communications* 181 (2022), 284–292.

[17] O-RAN Software Community. 2023. O-RAN Software Community DU. https://github.com/o-ran-sc/o-du-l2.

[18] eBPF.io. 2023. eBPF. https://ebpf.io/.

[19] CapGemini Engineering. 2023. CapGemini 5G gNodeB. https://capgemini-engineering.com/nl/en/services/next-core/wireless-frameworks/.

[20] 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)*. 1–7. https://doi.org/10.1109/GLOBECOM38437.2019.9014096

[21] Dash Industry Forum. 2023. Low Latency Streaming Powered by DASH.js. https://reference.dashif.org/dash.js/latest/samples/low-latency/testplayer/testplayer.html.

[22] 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.

[23] Scott Geba. 2023. Introduction to Port Mirroring. https://arista.my.site.com/AristaCommunity/s/article/introduction-to-port-mirroring.

[24] Steven S. Hong, Jeffrey Mehlman, and Sachin Katti. 2012. Picasso: Flexible RF and Spectrum Slicing. *SIGCOMM Comput. Commun. Rev.* 42, 4 (Aug 2012), 37–48. https://doi.org/10.1145/2377677.2377683

[25] Intel. 2023. Data Plane Development Kit (DPDK). http://dpdk.org/.

[26] Intel. 2023. FlexRA Reference Architecture for Wireless Access. https://www.intel.com/content/www/us/en/developer/topic-technology/edge-5g/tools/flexran.html.

[27] iovisor. 2023. Userspace eBPF VM. https://github.com/iovisor/ubpf.

[28] David Johnson, Dustin Maas, and Jacobus Van Der Merwe. 2022. NexRAN: Closed-loop RAN slicing in POWDER-A top-to-bottom open-source open-RAN use case. In *Proceedings of the 15th ACM Workshop on Wireless Network Testbeds, Experimental evaluation & CHaracterization*. 17–23.

[29] Antonios Katsarakis, Zhaowei Tan, Matthew Balkwill, Bozidar Radunovic, Andrew Bainbridge, Aleksandar Dragojevic, Boris Grot, and Yongguang Zhang. 2021. *rVNF: Reliable, Scalable and Performant Cellular VNFs in the Cloud*. Technical Report. Technical Report MSR-TR-2021-7, Microsoft.

[30] Maicon Kist, Juergen Rochol, Luiz A DaSilva, and Cristiano Bonato Both. 2018. SDR Virtualization in Future Mobile Networks: Enabling Multi-Programmable Air-Interfaces. In *2018 IEEE International Conference on Communications (ICC)*. IEEE, 1–6.

[31] 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 Ssing Open RAN Architectures. *IEEE Transactions on Mobile Computing* (2023).

[32] 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 *ACM SIGCOMM*.

[33] Yuanjie Li, Qianru Li, Zhehui Zhang, Ghufran Baig, Lili Qiu, and Songwu Lu. 2020. PBeyond 5G: Reliable Extreme Mobility Management. 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 (SIGCOMM)*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3387514.3405873

[34] Yuanjie Li, Zengwen Yuan, and Chunyi Peng. 2017. A Control-Plane Perspective on Reducing Data Access Latency in LTE Networks. In *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking (MobiCom '17)*. Association for Computing Machinery, New York, NY, USA, 56–69. https://doi.org/10.1145/3117811.3117838

[35] Yuanjie Li, Zengwen Yuan, and Chunyi Peng. 2017. A Control-Plane Perspective on Reducing Data Access Latency in LTE Networks. In *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking (MobiCom '17)*. Association for Computing Machinery, New York, NY, USA, 56–69. https://doi.org/10.1145/3117811.3117838

[36] Xingqin Lin, Jingya Li, Robert Baldemair, Jung-Fu Thomas Cheng, Stefan Parkvall, Daniel Chen Larsson, Havish Koorapaty, Mattias Frenne,

Sorour Falahati, Asbjorn Grovlen, et al. 2019. 5G New Radio: Unveiling the Essentials of the Next Generation Wireless Access Technology. *IEEE Communications Standards Magazine* 3, 3 (2019), 30–37.

[37] Xingqin Lin, Dongsheng Yu, and Henning Wiemann. 2021. A Primer on Bandwidth Parts in 5G New Radio. *5G and Beyond: Fundamentals and Standards* (2021), 357–370.

[38] Mavenir. 2023. World's First 5G SA Network Using Open vRAN on a Public Cloud. https://www.mavenir.com/case-studies/mavenir-and-dish/.

[39] José Mendes, Xianjun Jiao, Andres Garcia-Saavedra, Felipe Huici, and Ingrid Moerman. 2017. Cellular Access Multi-Tenancy through Small Cell Virtualization and Common RF Front-End Sharing. 35–42. https://doi.org/10.1145/3131473.3131474

[40] Binh Nguyen, Tian Zhang, Bozidar Radunovic, Ryan Stutsman, Thomas Karagiannis, Jakub Kocur, and Jacobus Van der Merwe. 2018. ECHO: A Reliable Distributed Cellular Core Network for Hyper-Scale Public Clouds. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking (MobiCom '18)*. Association for Computing Machinery, New York, NY, USA, 163–178. https://doi.org/10.1145/3241539.3241564

[41] Sofia Nyberg. 2016. Physical Cell ID Allocation in Cellular Networks.

[42] O-RAN Alliance. 2023. Cloud Architecture and Deployment Scenarios for O-RAN Virtualized RAN. https://www.o-ran.org/specifications.

[43] p4.org. 2023. P4_16 Language Specification. https://p4.org/p4-spec/docs/P4-16-v1.2.0.html.

[44] Zafar Ayyub Qazi, Melvin Walls, Aurojit Panda, Vyas Sekar, Sylvia Ratnasamy, and Scott Shenker. 2017. A High Performance Packet Core for Next Generation Cellular Networks. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. 348–361.

[45] Mubashir Adnan Qureshi, Ajay Mahimkar, Lili Qiu, Zihui Ge, Max Zhang, and Ioannis Broustis. 2017. Coordinating Rolling Software Upgrades for Cellular Networks. In *25th IEEE International Conference on Network Protocols, ICNP 2017, Toronto, ON, Canada, October 10-13, 2017*. IEEE Computer Society. https://doi.org/10.1109/ICNP.2017.8117537

[46] Shunmugapriya Ramanathan, Koteswararao Kondepu, and Andrea Fumagalli. 2022. Resiliency in Open-Source Solutions for Disaggregated 5G Cloud Radio Access and Transport Networks. In *2022 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. IEEE, 124–129.

[47] Muhammad Taqi Raza, Zhowei Tan, Ali Tufail, and Fatima Muhammad Anwar. 2022. LTE NFV Rollback Recovery. *IEEE Transactions on Network and Service Management* 19, 3 (2022), 2468–2477.

[48] Rethink Technology Research. 2023. Rakuten Claims Huge Edge Cloud, as Other Operators Follow Suit. https://rethinkresearch.biz/articles/rakuten-claims-huge-edge-cloud-as-other-operators-follow-suit/.

[49] Karen Schulz. 2022. Verizon Deploys More Than 8,000 VRAN Cell Sites, Rapidly Marches Towards Goal of 20,000. https://www.verizon.com/about/news/verizon-deploys-more-8000-vran-cell-sites.

[50] Kun Tan, Haichen Shen, Jiansong Zhang, and Yongguang Zhang. 2012. Enable Flexible Spectrum Access With Spectrum Virtualization. In *2012 IEEE International Symposium on Dynamic Spectrum Access Networks*. 47–58. https://doi.org/10.1109/DYSPAN.2012.6478115

[51] Xing Xu, Ioannis Broustis, Zihui Ge, Ramesh Govindan, Ajay Mahimkar, N. K. Shankaranarayanan, and Jia Wang. 2015. Magus: Minimizing Cellular Service Disruption During Network Upgrades. In *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies, CoNEXT 2015, Heidelberg, Germany, December 1-4, 2015*. ACM. https://doi.org/10.1145/2716281.2836106

[52] Sen Yang, Yan He, Zihui Ge, Dongmei Wang, and Jun Xu. 2017. Predictive Impact Analysis for Designing a Resilient Cellular Backhaul Network. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 1, 2 (2017), 1–33.

[53] Ali A Zaidi, Robert Baldemair, Vicent Molés-Cases, Ning He, Karl Werner, and Andreas Cedergren. 2018. OFDM Numerology Design for 5G New Radio to Support IoT, eMBB, and MBSFN. *IEEE Communications Standards Magazine* 2, 2 (2018), 78–83.