Wireless Personal Communications (2023) 131:1339–1363
https://doi.org/10.1007/s11277-023-10483-4

Midhaul Performance Modelling Using Commodity Hardware C-RAN Testbed

Nicolas Malm[1] · Kalle Ruttik[1] · Olav Tirkkonen[1]

Accepted: 24 April 2023 / Published online: 31 May 2023

Abstract
Midhauls introduce new characteristics to wireless networks. We study implementation performance behaviour to make appropriate network and algorithm design decisions soft–real-time midhaul-based radio access network implementations. A model is developed based on data obtained using a testbed, which can be used to estimate midhaul latency as a function of the number of communicating nodes. The testbed comprises one central unit (CU) controlling a variable number of distributed units (DU) synchronized over Precision Time Protocol (PTP). Average reporting latencies of 266 µs were observed with 16 DUs. Typical jitter performance was 99.99 % of values below 471.75 µs but maximum values up to an order of magnitude larger. Variations in performance of up to 16.2 times more deadline misses were observed between best and worst performing DUs. A model fitted to the obtained data estimates the latency and jitter of CU-DU communication as a function of the number DUs. Results indicate suitability for application with moderate synchronization requirements, for example positioning, but insufficient for the most stringent uses case such ultra-reliable communication.

Keywords Midhaul · Cloud radio access networks · Performance modelling · Synchronization

Kalle Ruttik and Olav Tirkkonen equally contributed.

✉ Nicolas Malm
nicolas.malm@aalto.fi

Kalle Ruttik
kalle.ruttik@aalto.fi

Olav Tirkkonen
olav.tirkkonen@aalto.fi

[1] Department of Communications and Networking, Aalto University, Maarintie 8, Espoo 02150, Uusimaa, Finland

1 Introduction

Fifth generation cellular systems (5G) currently being deployed promise to support a 1000-fold increase in traffic volumes along with support for tens of billions of devices [1]. As the commercial deployment of the first 5G networks takes place, research into the evolution of cellular communication system beyond 5G starts to take place [2–4]. Next-generation networks are currently envisioned to improve on 5G by offering, amongst other things, greater computation-communication integration and greater programmability. To cope with expected growth in both traffic volumes and device counts, network capacity must also increase. Recently, network densification [5, 6] and cell-free architectures [7] have been studied to provide the requisite increase in network capacity.

Traditional cellular architectures rely on the concept of individual cells to provide wide-area network coverage. Each cell is responsible for a geographic area defined by the prevailing channel conditions as perceived by user equipment (UE). Downlink cell measurements are reported to the BS on the uplink. A handover is performed when the measurement results indicate a cell other than the serving one has become more suitable. A mobile user will thus be handed over from cell to cell along its trajectory according to the measured channel conditions. During the handover process, the UE experience an interruption in its data connectivity [8]. Measurements gaps also lead to data transmission interruptions while battery consumption increases. 5G networks aim to enable new application classes, some of which impose stringent requirements on the reliability and latency of communication. The network should effectively become invisible to the user by providing constant high-quality service everywhere [2]. Consequently, network architectures must evolve to offer such a capability.

Ultra-dense networks (UDN) increase the re-use factor of spectrum and provide shorter links to UEs by greatly increasing the BS density [5, 9]. Network densification also presents new challenges. UDNs require tighter network-side co-ordination due to the increased potential for inter-cell interference [10] and mobility management signalling overhead [11]. As networks densify, classical network designs become inefficient. In terms of signalling, problems appear since as cell sizes shrink, so do the cell dwelling times. Cells can also be idle for significant periods of time due to the small cell size [5], resulting in unnecessary energy expenditure. A variety of solutions have been proposed in the literature, such as user-centric access point clustering [12], jointly optimizing handover decisions and energy efficiency through power control [13], multi-connectivity scheme to reduce radio link failures [14] and mobility-based user grouping [15]. Many of the presented solutions rely on some form of information sharing and co-ordination to enable making better decision than a single BS could achieve by itself using only its local data.

Software-defined approaches to wireless communication system architecture evolution and implementation have been the subject of extensive study [16–20]. In particular, centralizing and virtualizing radio access network (RAN) processing has attracted significant attention. This concept is known as cloud RAN (C-RAN). In a C-RAN system, baseband unit (BBU) processing is centralized into a shared computing resource pool. Connected to this pool are remote radio head (RRH) units responsible for the transmission and reception of signals. A centralized BBU pool can target resources to those cells currently serving users. This is possible due to the centralized view of the C-RAN controller. Temporarily shutting down unused cells reduces energy expenditure [21, 22]. In addition, the centralized view of network state provides opportunities for load balancing. For instance, more spectrum can be allocated to an overloaded cell [17]. As the BBUs of neighbouring BSs execute in a co-located manner, load-balancing of UEs can be performed with low overhead. Other benefits of software-based implementation include easier integration with computing resources as envisioned for 6G [4] and use of machine learning and artificial intelligence technologies [19, 23].

C-RAN platform software must maintain the time-domain frame structure of the cellular air interface, making it a real-time workload. A software-defined approach thus poses challenges with regards to timing-critical functions. Real-time tasks have processing deadlines they must meet in order to operate properly [24]. The consequences of missed deadlines are used to classify real-time tasks into two groups: hard–real-time and soft–real-time. Hard–real-time task deadline misses engender a system failure. Consequently, such systems are built to prevent any deadline misses. Guaranteeing hard–real-time performance in typically done through design-time analysis of possible execution times and system behaviour [25]. Soft–real-time tasks suffer degraded performance but can continue to operate or try to recover when a failure to meet timing requirements occurs [26].

Soft–real-time approaches present advantages in terms of ease of development and lower cost as not all possible inputs must be analyzed to determine whether the system is schedulable or not. General-purpose commercial off-the-shelf platforms are often too complex to fully predict and analyse in terms of worst-case performance [24, 25]. In particular, flexibility and programmability increase the number of possible execution flow paths through the code. As the number of cases to analyse grows large, it becomes intractable to perform worst-case analysis as in traditional hardware-centric hard-real-time approaches. Moreover, hosting multiple tasks on a single platform (e.g.: multiple BBUs in a C-RAN) introduces the potential for interference among them, thus complicating worst-case analysis even further. Communication over wireless networks is by definition soft–real-time due to the inherent possibility of packet loss due to channel fading. Accordingly, implementing C-RANs on a general purpose soft–real-time platform is a real possibility.

The use of soft–real-time on off-the-shelf provides flexibility and ease of integration into other systems while raising the question of suitability for time-sensitive tasks due to a lack of timing guarantees and domain specific optimizations. Soft–real-time implementation has been considered from different viewpoints. Virtualization [27] and scheduling [28] aim to share the underlying hardware while providing isolation between the various concurrently executing tasks. In addition to computation, networking performance plays an important role [29–31] as data must be obtained in time to be processed by the relevant deadlines. Latency affects the response time of the system while determinism (i.e.: jitter) [32] impacts the degree of co-operation and the time granularity of co-ordination achievable.

Disaggregated network architectures, such as C-RAN, involve splitting RAN functionality into separate nodes sharing information. This exchange of information takes place over a connection called the midhaul. The introduction of the midhaul changes the characteristics of the RAN implementation compared to legacy approaches. These changes must be understood in order to properly design implementation to fully extract the benefits of RAN disaggregation.

This paper studies the feasibility of implementing a software-defined midhaul-based RAN using a general-purpose operating system (GPOS) on commodity hardware. Experiments were conducted using a testbed. This platform was used to accomplish three goals: i) demonstrate the feasibility of software-based midhaul-based RAN implementation, ii) build a model of scalability of soft–real-time RAN in terms of midhaul communication and iii) assess suitability for various potential applications.

This paper is organized as follows. Section 2 presents the system model studied while the methodology used is detailed in Sect. 3. Section 4 discusses the results obtained. Finally, Sect. 5 contains concluding remarks.

2 System Model

We model an ultra-dense urban network with multiple co-operating network nodes serving UEs. Network nodes work together to provide service to UEs over the air. The RAN is responsible for gathering channel state information with no UE-side measurements. In this model, proposed in for example [33] and [34], TDD channel reciprocity is exploited to obtain the downlink channel from uplink measurements. Network nodes are implemented as soft–real-time processes on a GPOS running on commodity hardware (Fig. 1).

2.1 RAN Architecture

RAN functionality is split among network nodes into two tiers: central units (CU) and distributed units (DU) under their control. DUs may have one or more radio units (RU) either integrated into the same physical device or as distinct units from DUs. In this study, the question of how each DU is implemented in terms of RU design is not considered in details. It is assumed that each DU has some suitable and geographically close means of sending and receiving radio frequency (RF) signals to and from UEs. Figure 2 presents the aforementioned node types along with their interconnections. CUs control one or more DUs over a midhaul link.

The functional split studied in this work is Option 5 as defined in TR38.801 [35]. MAC layer processing is divided between the CU and DUs. Figure 1 presents the functional split. MAC processing in the CU and DUs communicate over the midhaul to exchange information. CUs manage the overall state of the cell but do not operate the air interface used to

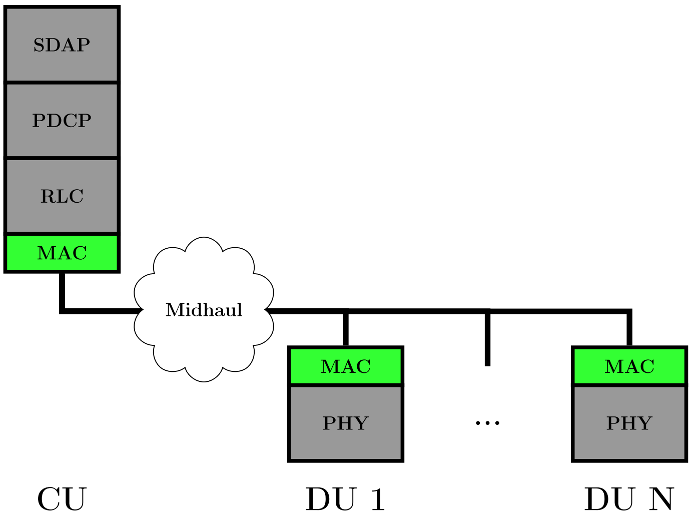


Fig. 1 Functional split considered in this work

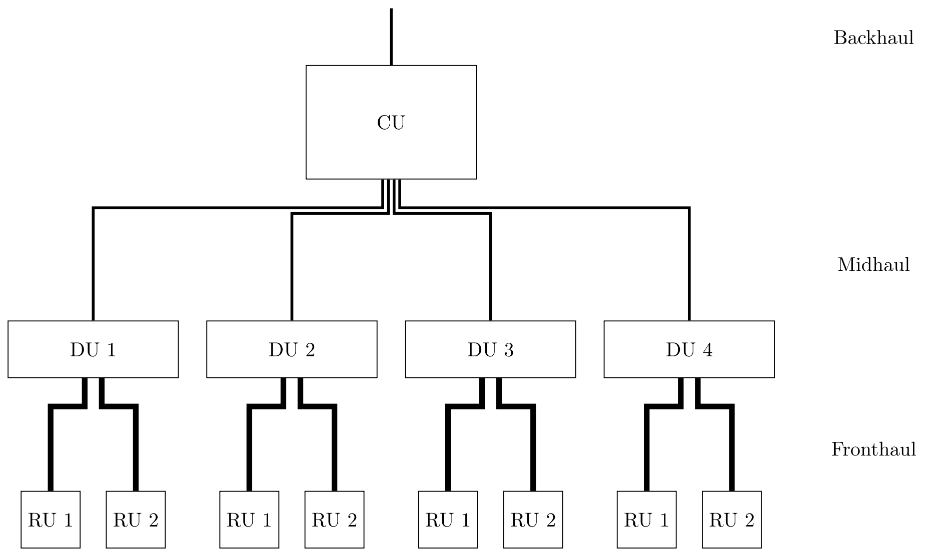


Fig. 2 RAN node types and their interconnections. The backhaul connects to the mobile core network

communicate with UEs. Instead, they delegate this task to the DUs. The latter are grouped by geographical area under a CU. Functionality handled by CUs operates on a slower time-scale than that placed in DUs. An Option 5 functional split enables the use of coordinated multipoint (CoMP) at the level of coordinated scheduling but not joint transmission. This is due to the CU not having channel state information (CSI) available. The sharing of CSI would increase the overhead and complexity of the implementation. The choice of Option 5 is motivated by a balance between centralization gains and midhaul load [36]. It provides opportunities for co-operation through the CU without imposing stringent requirements on the transport link between CU and DU [37]. The main challenges in Option 5 is the realization of the CU-DU interface. The ability of the CU to co-ordinate the operation of DUs depends on a properly designed midhaul.

The CU serves as a co-ordinator by managing spectrum use between DUs. Co-operation takes place at the cell-level by means such as inter-DU inteference control, centralized precoding decisions and beam direction management. No joint RF processing takes place as this would require sharing of CSI. Centralizing such functions requiring short reaction times and transfer of large volumens of CSI would impose an excessive load on the midhaul. Instead, instructions sent by CUs indicate what type of service each UE should be provided with by each DU in a given period of time. Within this period, DUs have flexibility in how to carry out their instructions in terms of fast scheduling and link adaptation. At the end of each period, DUs send a report to the CU containing information such as the number of retransmissions and UEs detected. The CU uses DU reports to make decision for the next period. The studied model reduces signalling requirements and interface complexity between the CU and DUs compared to joint RF processing requiring sharing of CSI.

The CU uses all information it possesses to determine which DU should handle which UE. The aim is to enable the CU to optimize overall network performance through co-ordination using the information it collects and the instructions it sends to DUs. The CU distributes its clock to the DUs to enable distributed scheduled functions. A clock synchronization protocol is necessary as the CU and DU execute on physically separate equipment with therefore no common hardware clock.

Figure 3 illustrates network deployment in an urban environment with DUs placed along streets. DUs and UEs divide between the two CUs along the vertical dashed line through the middle of the figure. All DUs and UEs on either side (left or right) are managed by the CU on the same side. Each CU can use any of the DUs under its control to serve any of the UEs the CU has responsibility for. In other words, each of the UEs in Figure 3 maintains long-term state with the CU but can receive its physical signal from any of the DUs of that CU. Each DU operates one logical cell. One such cell comprises, for example, the RUs and antennas mounted on one or more adjacent lampposts. As the UE moves, the CU changes the serving DU. In order to minimize connection interruptions, the CU handles high-level management of dataflows. Doing so avoids the need to transfer connection information, for example radio bearer state, during intra-cell mobility. Hence, signalling overhead is reduced. Aggregating network state information in the CU also offers it a better network view for optimizing network resource use. The CU does not need to exchange messages with DUs or other nodes when making radio resource management decisions within its service area. From the UE's perspective state information does not change as it remains connected to the same CU. Instead, the CU is responsible for forwarding the necessary information in a timely manner to the DUs.

DUs are colocated as virtual instances in the same physical server. Each DU can provide a different type of service. For example, DUs instances providing eMBB and mMTC can by operated using the same physical hardware. The service mix can thus be adapted easily by starting and stopping DU instances on demand. Sharing physical infrastructure reduces costs as fewer power supply, cooling and other auxilliary systems must be procured. DUs are connected to antennas using optical connections. Reducing costs by sharing

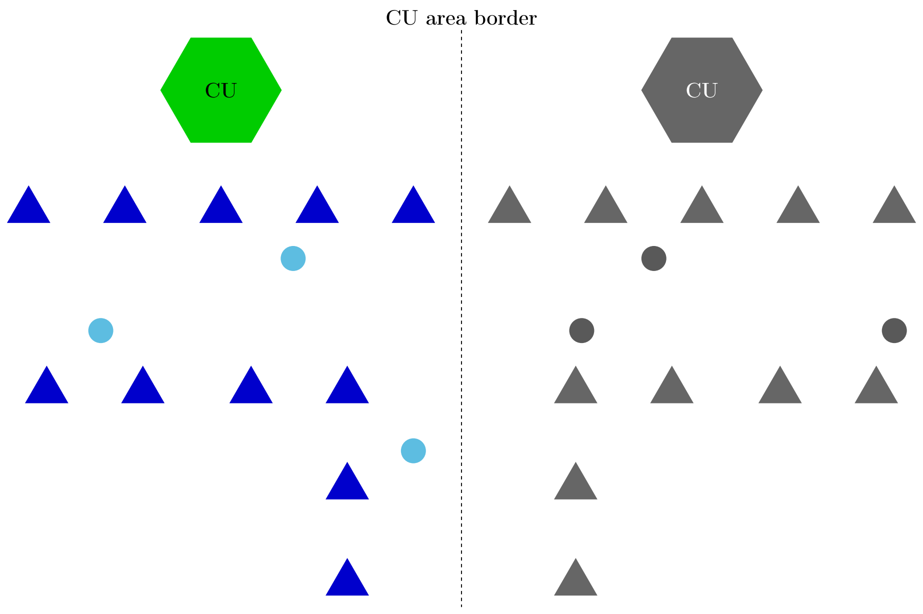


Fig. 3 Ultra-dense network layout. CUs are indicated by hexagons, DUs are indicated by triangles and UEs are represented by circles. DUs and UEs divide between the two CUs along the dashed line

infrastructure is important in UDN scenarios due to the high number of sites. Furthermore, placing a physically larger, fully-fledged base station on street furniture is more difficult than a smaller frontend and antenna unit. Sharing infrastructure can also enable a neutral-host model, where the infrastructure owner rents facilities to mobile network operators.

2.2 C-RAN Delay Model

The introduction of the midhaul changes the structure of the RAN and hence its operational characteristics. In particular, the split into the CU and DU introduces a new source of latency and jitter. A midhaul link increases system delays in two ways. First, it forms an extra component that all information shared between CU and DU must traverse, as compared to a monolithic design. Second, splitting the CU and DU into separate physical devices leads to separate clocks, which need to be synchronized. Any difference in the notion of local time can result in variation in the timing of scheduled events. Similarly to the need for a channel model to properly design a wireless network, the latency performance characteristics of a C-RAN platform influence the design of a communication system. This is especially true in real-world implementations of system with centralized decision making but distributed data collection. Such systems will only achieve their performance targets if information can be transferred in a timely manner. Latency profile characterization also constitutes a prerequisite for certain new use-cases that are more latency sensitive than traditional applications [38]. Quantifying the scaling behaviour helps network planning and configuration.

Understanding the delay model of the midhaul in a split C-RAN design is important as its latency and jitter performance affects all network co-ordination functions as well as the feasible function splits. Lower level splits (more centralization in the CU) required tighter bounds [36] as their function operate on shorter timescales. Algorithms designed for a midhaul-based C-RAN must therefore account for timing differences compared to monolithic independent BS implementations. The requirements for these RAN function algorithms vary in timing requirements widely, from tens of nanoseconds to tens of milliseconds [32, 35, 39, 40].

This work aims to characterize the scalability of CU-DU midhaul communication in a GPOS-based platform on commodity hardware. The number of DUs placed under the control of a single CU impacts the latency and jitter of communication. This occurs because messages to and from DUs must queue in the CU. Thus, the first message will experience a shorter delay than the last. Message generation times can vary due to the soft–real-time nature of the platform. This variation causes the DU messages' ordering to not be static, thus creating jitter. Furthermore, the soft–real-time nature of CU processing can result in additional variance in the time required to execute tasks distributed between CU and DU. Of particular note is the fact that clock synchronization occurs using a software-based PTP implementation. Like other software, the PTP task is subject to variance in execution time leading, in turn, to variance in the synchronization between CU and DU.

3 Experimental Assessment

Distributed network architectures rely on co-ordination between network nodes to provide service to UEs. The more information the network can gather and share the better decisions it can make. However, greater information volume also increases the capacity required to transmit it. Measurements were carried out to quantify the communication latency between RAN nodes with GPOS and commodity hardware. The testbed used implements a distributed architecture as discussed in Sect. 2.1 with one CU controlling multiple DUs. Performance is assessed in terms of the latency observed for a given number of DUs under the control of one CU. In this work, two signalling performance metrics are used: command delay and reporting time accuracy.

3.1 Midhaul Metrics

Command latency from CU to UE represents the time elapsed from the moment the CU issues a command through a DU to the moment the UE's frontend receives the signal. It includes delays incurred by the radio frontend and its driver. No processing of data takes place in this testbed in either CU or DU and therefore the processing time is assumed to be zero. This is done as the delay would depend on the algorithm and the focus of this work is on the midhaul. Command latency impacts the CU's response time for changing the signal a UE receives. It thus also represents a lower bound for the time taken by a message from the CU to reach an application running on the UE.

Reporting latency measures the time elapsed between the expected report arrival time at the CU and when the DU reports actually arrive. If each DU sends a report at nominally the same time, these reports would all ideally arrive at the CU at the same time. In a practical system however, the notion of current time in each node likely differs. A mismatch of the current time between CU and DU will result in DUs making measurements at a different time than the CU intended. Combining the measurements will then suffer from increased error due to the data not pertaining to the same time instant. In addition to current time differences, there will likely be congestion and queuing effects as well. Consequently, messages are transmitted at slightly different times from each DU. The non-zero difference results in delay variation in reports originating from different DUs. Consequently, the CU faces uncertainty in the arrival time of measurement data. Estimating the delay is done in the CU by taking the difference between its local time at reception and the DU's timestamp in the message.

As an illustrative example, consider location-based mobility management, based on wireless positioning performed by the RAN. Figure 4 illustrates the impact of midhaul delay as pertains to UE positioning. DUs measure the AoA of the UE using its positioning reference signal at some point in time (dashed lines). They then send their measurement report to the CU. Upon arrival, the DU measurements are used by the CU to compute a

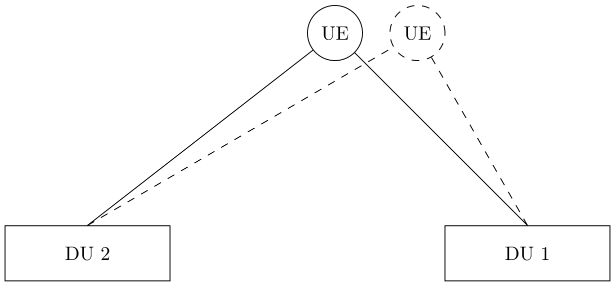


Fig. 4 Midhaul delay induced positional error. The solid lines represent the UE's true location while the dashed lines represents the information available to the CU

position for the UE, update mobility management decisions and send these decisions to the DUs. The UE continues to move and has changed position by the time the CU has computed its position (continuous lines). Synchronization imperfections impact the performance of a location-based mobility management scheme in two aspects.

Firstly, in order for a handover to be successful, both the target and source DUs must have the same notion of when to execute the transfer. Otherwise, one of the two may execute its part of the handover too early or too late, resulting in either interference due to overlap or an interruption in service for the UE.

Secondly, clock offsets cause the position estimate of a UE to suffer degraded accuracy. This results from the CU having to wait for all required AoA reports to arrive before it can compute the UE's position. During this wait, the UE can potentially continue to move. An alternative to waiting would be to compute the UE's position using whatever reports are available. However, this approach also entails a loss of accuracy as the CU uses less information for its computations. Delayed or less accurate information can impair the CU's ability to perform interference management between DUs based on the UE's position.

3.2 Measurement Setup

Experiments were conducted using two directly connected servers. Server specifications are given in Table 1. One server operated as the CU and the other executed the DUs either directly on the host OS or inside containers. In an actual deployment, the CU would be located in a datacenter while the server hosting DUs would be closer to the served physical locations. The servers were connected directly using two connections: one for the midhaul link and one for PTP clock synchronization. PTP was chosen as it is suitable for standard Ethernet networks with no built-in clock transfer and synchronization method.

Testbed software was run with scheduling priority set to the SCHED_FIFO class of the Linux scheduler. DUs were pinned to specific CPU cores in an effort to reduce timing variance. Futher isolation was provided by the use of containers to virtualize the environment of each DU. PTP software daemons were ran on dedicated cores as well to reduce interference from the testbed software and vice-versa. Messages are exchanged as Ethernet frames with no higher-layer protocol. Design of suitable midhaul protocols is left to future work.

Measurements for the command latency metric (see Sect. 3.1) were carried out as depicted in Fig. 5. The CU sends a command to the DU to transmit a pulse. The latter then instructs its frontend to send a square wave pulse. The CU continuously monitors its frontend. Upon detection of the RF pulse, the CU computes the latency from its issuing of the

Table 1 Main test server hardware and software components

Component	Model
CPU	AMD Epyc 7401P
NIC	Intel X710-DA4
OS	Arch Linux kernel 5.5.9
Container system	LXD/LXC version 4.0.0
PTP software	linuxptp version 2.0
PTP profile	ITU G.8275.1
Frontend	National instruments USRP-2932 (UHD 3.15)

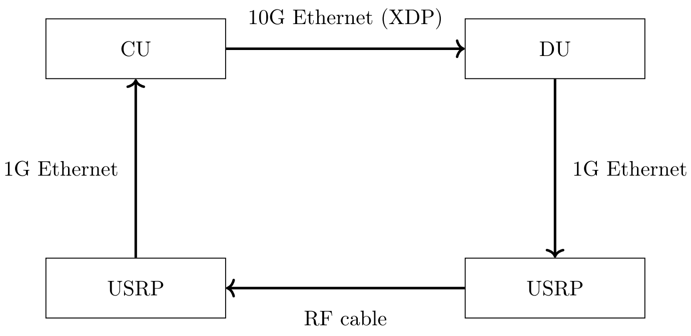


Fig. 5 Measurement setup for CU-to-UE command latency

transmission command to the reception of the pulse. The CU plays the role of UE which, in an actual system, would be the recipient of the CU's transmission. Since the CU and UE are the same in this setup, the transmit time recorded for the command can be directly compared to the reception time of the RF pulse.

Transmitting and retrieving samples from the frontends incurs some delay. The measured arrival time of commands is lengthened by the time it takes for the radio frontend's driver to retrieve the samples containing the pulse. This delay is expected to be larger in this setup than with commercial hardware used in a real-world deployment. The latency seen by the CU when recording arrival time will likely be greater than that seen by an application running on a commercial UE with lower processing time. The results therefore constitute a pessimistic assessment of midhaul command latency.

Command latency was tested with a sampling rate of 1 Msps and 250 samples per buffer. Both values impact the command latency by changing how long it takes for a command buffer to get filled and how fast it is transmitted. However, the focus of this work is on the midhaul and the design of a low-latency air interface is thus out of scope.

Measurements for the reporting metric (see Sect. 3.1) were carried out as depicted in Figure 6. The CU configures DUs to transmit a 100-byte long report at 100 ms intervals. The inter-message interval is kept long to test the accuracy of synchronization for scheduled events as opposed to steady-state flows. When the time to transmit arrives, each DU sends a message to the DU. These messages go through the virtual network bridge in the DU host before reaching its 10G network interface card (NIC) for transmission to the CU. Upon receipt, message arrival times are timestamped. Since the CU knows the transmission interval it configured, it is able to compute the offset of the timestamps from the expected time. In order for the CU and DUs to share the same notion of time, the clock of the DU server is synchronized to that of the DU server using PTP.

The PTP implementation performs synchronization in two parts. First, the CU uses its OS time of day to set the time of its NIC, which then redistributes it to the DU's NIC. Second, the now synchronized NIC clock is used to set the DU server's OS clock. The aim of time of day synchronization is to provide the same notion of current time for the testbed software running on the CU and DU. DUs transmit a message to the CU at a pre-determined periodicity as determined by the current OS time. All DUs send their reports synchronously at the same nominal time. Upon reception of the DU's report messages, the CU computes the difference between the timestamp contained in the message and its own local time. The difference between the scheduled time of measurement

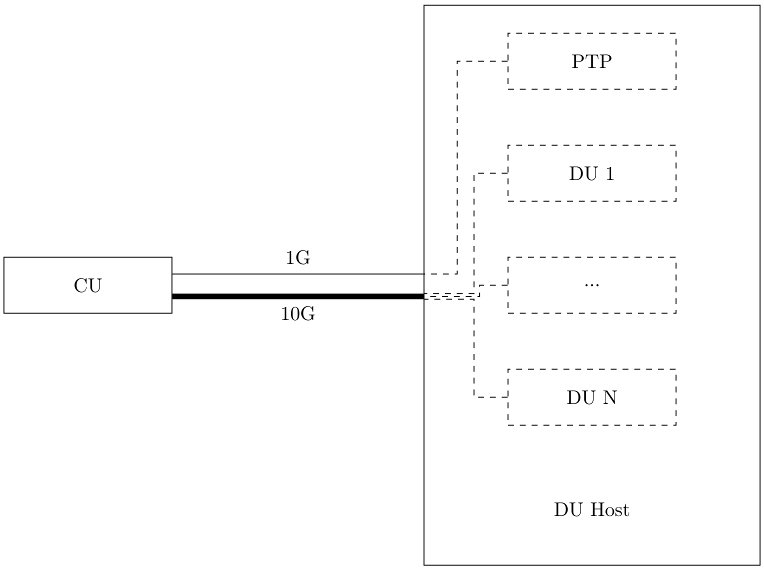


Fig. 6 Measurement setup for DU-to-CU timing synchronization

and the time of reception of these messages at the CU then indicates the combined latency and clock synchronization error of the system. Both servers must therefore have the notion of current time to schedule distributed tasks correctly.

Ideally, the difference between the actual and expected arrival time of DU report messages would be as close to zero as possible. In practice there will be an offset from imperfect clock synchronization, processing delay, queueing and transfer delay for the messages. Additionally, software incurs variable delays depending on load and the behaviour of other software executing concurrently. This impacts DUs' ability to begin transmission at precisely the intended time. The aforementioned sources of variance are part of the nature of soft–real-time GPOS systems impacting performance.

The measurement software runs in userspace making use of the Express Data Path (XDP) feature of the Linux kernel. XDP is a kernel networking stack bypass technology. It aims at providing the lowest possible latency for network frame processing by passing frames directly from the NIC driver to the userspace application to which they are destined. Doing so avoids the overhead of going through the kernel's processing for those applications that do not need it. Designated frames or packets are handled by XDP while all other traffic transmits through the kernel's networking stack. The ability to utilise both on a single network interface enables retaining the features of kernel processing where required while enabling low latencies for select traffic. The control plane can therefore reside in userspace, where programming is easier due to many libraries and frameworks being available. Once the control plane has computed a course of action, the rules in the XDP are updated. One of the advantages of XDP lies in providing a high performance network frame processing capability without requiring hardware-specific solutions. Consequently, portability and migration of BBUs becomes easier. This, in turns, helps in building inter-operable and maintainable implementations conforming to standards such as O-RAN [41].

4 Results and Analysis

Measurements were carried out as described in Sect. 3.2 with 1, 2, 4, 8, 16 synchronized DUs sharing a midhaul link. Workload and DU process mappings to CPU cores were kept constant. Data recording aimed to asses the impact of the commodity hardware GPOS platform on midhaul performance.

First, results are presented for the command and reporting latencies. Second, homogeneity of the performance is analyzed across DUs and over time. Third, a model is developed based on the data collected. The model aims to generalize the results in order to enable estimation of performance in other network configurations. Finally, the impact of midhaul performance on localization performance is studied as an example of the impact of midhaul latency.

4.1 Measured Latencies

As described in Sect. 3.1, midhaul latency is assessed using the two metrics of command and reporting latency. Measured command latencies between CU and DU are presented in Figure 7. While the maximum is more than an order larger than the mean, it can be observed that the majority of values are clustered together. Such clustering is indicative of low jitter for the majority of data points. The distribution has a low typical value but a long tail with most latencies being of similar magnitude.

The second metric analyzed is reporting latency. Figure 8 presents its distributions for the case of 2, 4, 8 and 16 DUs, from left to right and top to bottom. Similarly to the command latency, the recorded values cluster around a low typical value with a long tail. Table 3a–e presents the minimum, maximum and standard deviations of the latency across

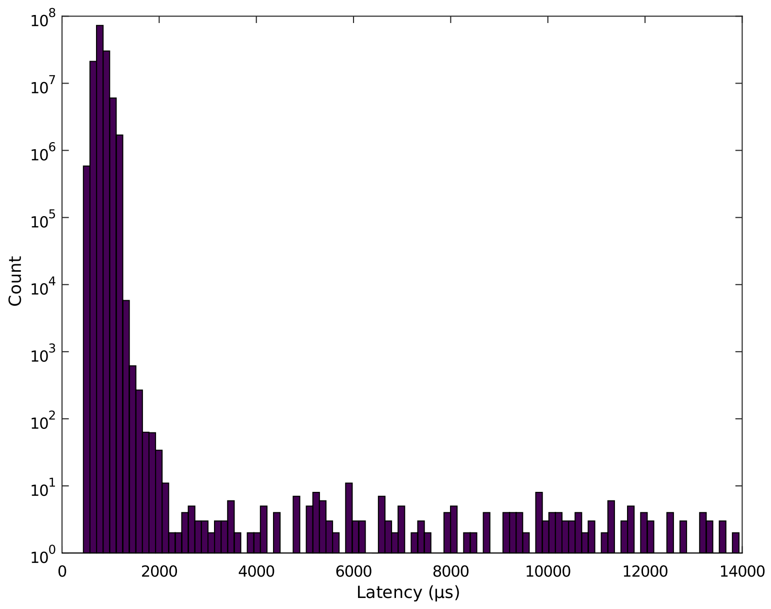


Fig. 7 Histogram of CU command latency on a semi-log scale. N = 132 692 265

all DUs. Certain standard deviations are zero due to the rounding of measured latencies resulting in the same value for all DUs.

Figure 8 has large values in the dataset omitted from the figures for clarity. It can be observed that the distribution exhibits a dependency on the number of DUs reporting. This dependency can be explained by the centralization of the network. Since the CU must handle all the reports from the DUs, it constitutes a bottleneck in terms of scalability. As all DUs report synchronously, there will be contention on the shared midhaul. The extra latency caused by concentrating decision making into a single controller constitutes a trade-off with regards to the benefits of centralization. The more DUs report to a single CU, the more information is available to make optimal decisions on the network level. On the other hand, a high degree of centralization creates congestion for transferring the information required.

Results for both the command and reporting latency exhibit a low typical value and long tail. This behaviour results from the soft–real-time nature of the experimental platform. Since there are no upper limits on the duration of operations, very large latencies may occur when tasks interfere with each other. While relatively rare, these events can produce delays much larger than the typical case as shown by the difference between the 99.99 % and 99.999 % quantiles, and maximum value in Tables 2, 3a–e.

4.2 Performance Homogeneity

Scaling up the RAN to higher N_{DU}-per-CU ratios increases not only the average latency of messages but also the differences between DUs. In Table 3e, it can be seen that the latency ratio of worst performing to best performing DU is higher for $N_{DU} = 16$ than the cases with fewer DUs. For instance, in terms of worst-to-best mean ratio, the value grows from 1.3 at $N_{DU} = 4$ to 1.8 at $N_{DU} = 16$.

Figure 9 depicts the late rate performance for each DU as function of TTI duration in the $N_{DU} = 4$ (top of the figure) and $N_{DU} = 16$ (bottom of the figure) cases. The number of DUs affects the late rate with curves being steeper at lower N_{DU}. Adding DUs results in more queuing on the shared midhaul. This increases the range of possible delays experienced and thus makes the late rate curve less steep. The stair-like pattern seen in Figure 9 is cause by the separation of the peaks in Figure 8.

At $N_{DU} = 16$, two DUs diverge in behaviour. DU-6 and DU-12 exhibit a higher rate of deadline misses than the other DUs. They benefit less from an increase in TTI length in terms of their deadline miss performance. Platform overhead affects DU-6 and DU-12 more than other DUs. This overhead is visible in Figure 8 as the values between the peaks. Interference from other processes cause latencies to grow beyond the typical value represented

Table 2 Descriptive statistics of CU command latency

	μs
min	440
mean	799
max	13 934
std	107
99.999 % quantile	1 378

Values are rounded to the nearest microsecond.

Table 3 Report arrival time latency distribution for (a)1, (b)2, (c)4, (d)8 and (e)16 DUs

(a)1 DU

	μs
min	36
mean	55
max	858
std	2
99.9 %	40
99.99 %	42
99.999 %	248

(b)2 DUs

	Min–Max (μs)	Std (μs)
min	48–48	0
mean	66–67	0.71
max	758–903	102.53
std	13–13	0
99.9 %	69–69	0
99.99 %	74–74	0
99.999 %	237–239	1.41

(c)4 DUs

	Min–Max (μs)	Std (μs)
min	49–49	0
mean	79–103	11.21
max	1093–1161	31.22
std	27–29	0.96
99.9 %	134–137	1.26
99.99 %	143–146	1.26
99.999 %	313–481	74.76

(d)8 DUs

	Min–Max (μs)	Std (μs)
min	42–43	0.46
mean	139–164	9.44
max	2013–2118	33.40
std	58–64	2.20
99.9 %	264–265	0.52
99.99 %	292–296	1.28
99.999 %	744–897	58.53

(e)16 DUs

	Min–Max (μs)	Std (μs)
min	38–40	0.72
mean	214–377	49.04
max	3192–7295	1389.50

Table 3 (continued)

(e)16 DUs

	Min–Max (μs)	Std (μs)
std	102–124	6.13
99.9 %	449–472	7.33
99.99 %	466–494	9.01
99.999 %	865–1055	49.52

Percentage values indicate quantiles. Minimums, maximums and standard deviation are calculated using the values for each individual DU. Values are rounded to the nearest microsecond. N = 50000000.

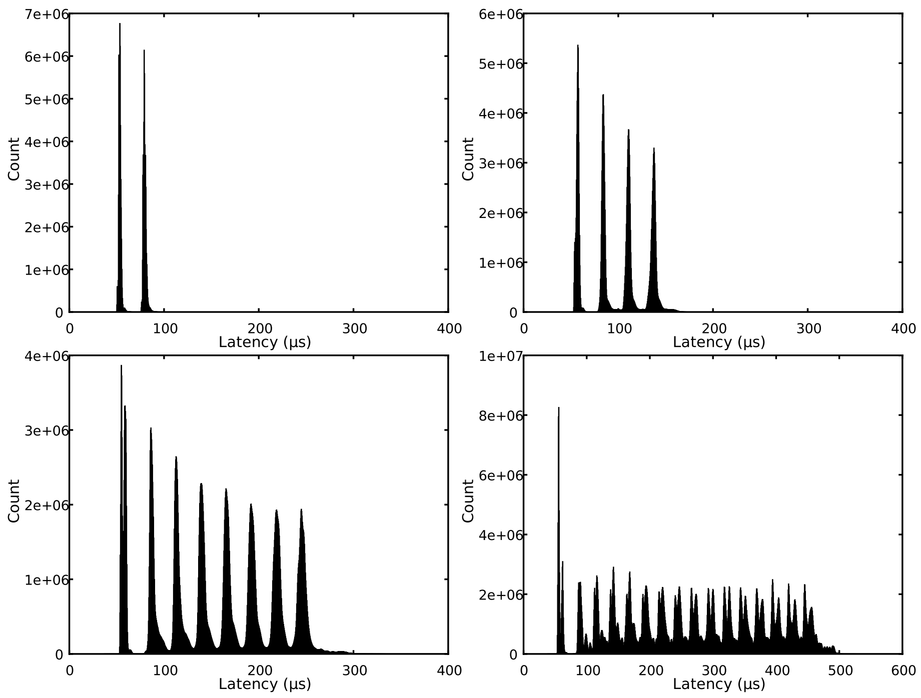
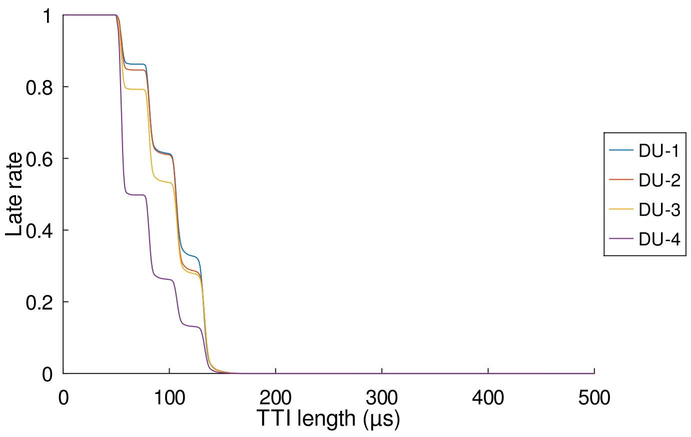


Fig. 8 Histogram of difference between DU nominal timestamp and CU local time at reception with 2, 4, 8 and 16 DUs

by the peaks. At $N_{DU} = 4$, one DU also performs better than the others but the shape of its curve remains broadly similar. Greater variation at $N_{DU} = 16$ suggests that higher DU-per-server densities generate less homogenous latency performance across DUs.

In addition to inter-DU differences, behaviour also varies over time. Variation occurs due to the soft–real-time nature of the platform on which the CU and DUs execute. External factors such as background processes, OS tasks and hardware characteristics can increase or decrease the processing time of operations. These sources of interferences do not operate with TTI granularity. Patterns may therefore appear at other time scales. It can be seen in Figure 8 that latencies are less quantized for higher N_{DU}.



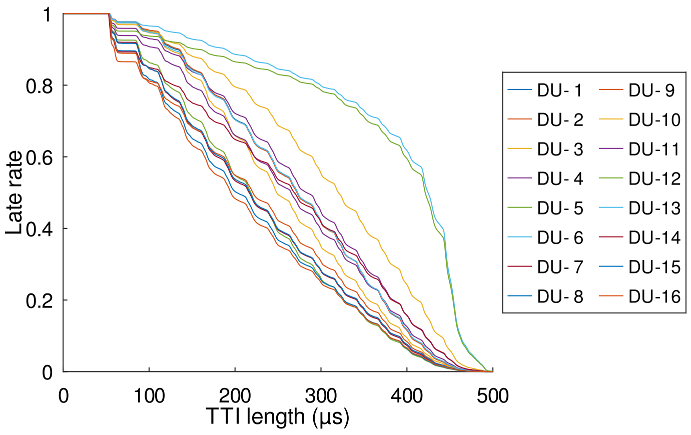


Fig. 9 Deadline miss rate for individual DUs as a function of TTI duration for $N_{DU} = 4$ (top) and $N_{DU} = 16$ (bottom)

The coefficient of variation is computed as the standard deviation divided by the mean from the per-time slice high latency rate for each DU. A coefficient of variation close to zero indicates that the DU has a similar number of large latencies in each period and its behaviour is thus predictable. Conversely, a high coefficient of variation indicates that the DU undergoes periods of either particularly good or poor performance throughout the recorded data. Table 4 presents the coefficient of variation of the number of large latency TTIs per time periods for each DU in the $N_{DU} = 16$ case. The threshold for a large latency was taken as the 90 % quantile over all values for each DU. Table 4 presents variability of the distribution of the 10 % highest latencies at aggregation periods of one second, one minute and one hour. The ratio of worst-to-best coefficient of variation for the three periods considered are: 8.6 at one second, 16.2 at one minute and 6 at one hour. Performance monitoring should therefore be performed at multiple granularities and not only consider aggregates over long periods.

Figure 10 shows the evolution of performance in terms of large latency for each DU in the $N_{DU} = 16$ over time in one minute slices. For each time slice, the number of large values is computed for each DU and then divided by the total number to obtain their

Table 4 Inter-DU comparison of the number of large latencies for 16 DUs at three different time aggregation lengths

CV	DU							
	1	2	3	4	5	6	7	8
1 s	0.13	0.12	0.13	0.87	0.12	0.23	0.12	0.14
1 min	0.06	0.06	0.07	0.63	0.05	0.15	0.05	0.06
1 h	0.03	0.03	0.05	0.12	0.03	0.04	0.02	0.03
CV	DU							
	9	10	11	12	13	14	15	16
1 s	0.54	0.85	0.82	0.21	0.13	1.03	0.12	0.12
1 min	0.38	0.61	0.63	0.13	0.06	0.65	0.04	0.06
1 h	0.07	0.10	0.12	0.03	0.03	0.12	0.02	0.04

Coefficient of variation (CV) values indicate variation across time slices. The threshold for a large latency was set to the 90 % quantile of the recorded latencies.

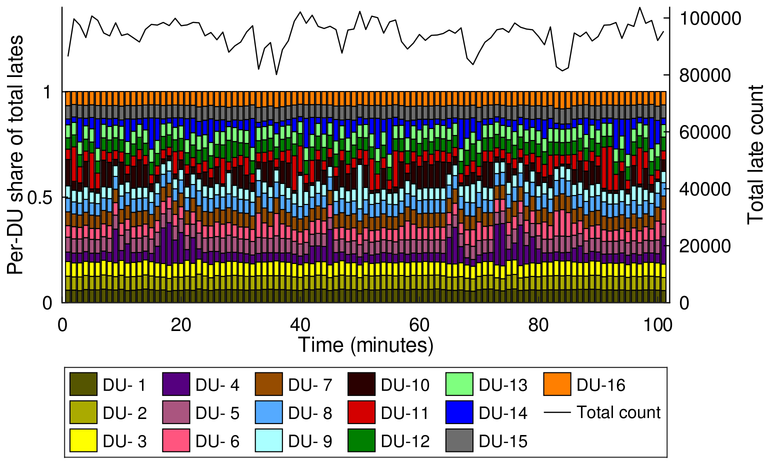


Fig. 10 Total large-latency TTIs and per-DU relative share for the $N_{DU} = 16$ at one minute aggregation period

relative share. In addition, the total number of lates is plotted as a line graph. Figure 11 presents the same information as Figure 10 for a time slice length of one second. The time range of Figure 11 corresponds to the second and third bars of Figure 10. It can be observed that some DUs, such as 1-3, have consistent performance throughout the timespan of Figure 10. Others, such as DU-10 and DU-11, exhibit much more instability. Unstable DUs do not have the highest means or maximums. For example, DU-6 experiences higher values than the other DUs as can be seen in Figure 9 but its performance is consistent over time. Its high mean only reduces the computation time budget. Assessing performance should therefore be done using multiple metrics suitable for the

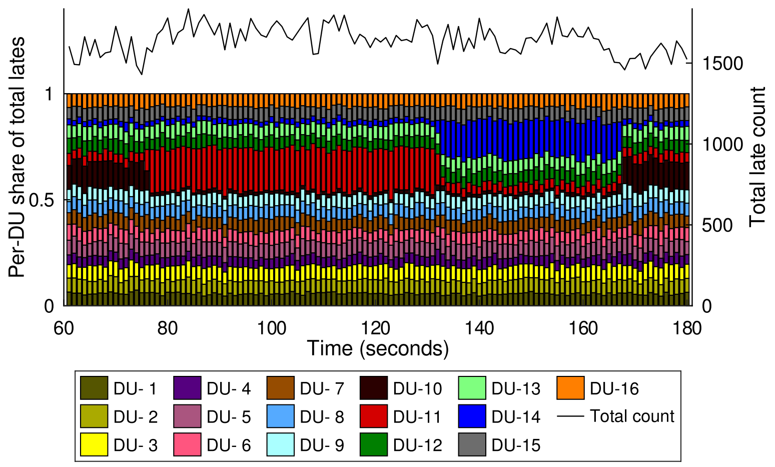


Fig. 11 Total large-latency TTIs and per-DU relative share for the $N_{DU} = 16$ at one second aggregation period

studies application. Certain use-cases benefit more from a high-but-stable mean while other benefit from lower averages even at the cost of lower predictability.

The variability of DU performance impacts algorithm and system design. Algorithms should be able to cope with changes in the midhaul latency reducing the available processing time budget. Correlations between DUs also affect the reliability by impacting the probability of two DUs serving the same UE being late at the same time. If this probability is low, use of multiple DUs to transmit the same data can be effective in reducing the probability of outage. Another aspect impact by inter-DU variability is large-scale scaling of applications. Different MEC providers will very likely use different equipment and software platforms. Assumptions about fungibility may thus not hold. An edge computing workload migrating from one server to another might thus experience different latency performance even when allocated nominally the same amount of resources. One reason for migrating workloads is to load balance DU instances. Since load balancing aims at achieving more even performance, it is important to account for possible inter-DU differences to achieve the desired end result.

4.3 Latency Model

In order to generalize the results, a simplified model of midhaul latency can be built. Such a model for estimating the performance of the RAN for various DU counts can be useful in dimensioning a network. The average delay and its standard deviation can be estimated from the number of DUs. To build the model, the mean and standard deviation of all DUs at a given DU count are averaged. Model fitting was done using Matlab [42]and its Curve Fitting Toolbox [43]. The model predicts the latency in microseconds. The resulting functions are:

$$f_{\text{mean}}(N_{DU}) = 14.16 N_{DU} + 39.39 \quad (1)$$

$$f_{\mathrm{std}}(N_{DU}) = 7.14 N_{DU} - 1.503 \tag{2}$$

where N_{DU} is the number of DUs, $f_{\mathrm{mean}}(N_{DU})$ is the mean delay of reporting and $f_{\mathrm{std}}(N_{DU})$ is the standard deviation of reporting delay.

Since the latency distributions for command and reporting delays possess long tails, a log-normal distribution was selected to model the behaviour. The obtained mean and standard deviation in (1) and (2) are then used to compute the parameters of the log-normal distribution estimating the midhaul reporting latency. The mean and the variance of the log-normal distribution are:

$$m_{\mathrm{Log-normal}} = e^{\mu + \frac{\sigma^2}{2}} \tag{3}$$

$$v_{\mathrm{Log-normal}} = (e^{\sigma^2} - 1)e^{2\mu + \sigma^2} \tag{4}$$

Relating the experimental mean to the closed form ones, we find the parameter μ of the distribution as

$$\mu = \frac{2 \log f_{\mathrm{mean}} - \sigma^2}{2}$$

By substituting the value obtained for μ into the formula for the variance f_{std}^2,

$$\begin{aligned} f_{\mathrm{std}}^2 &= v_{\mathrm{Log-normal}} \\ &= (e^{\sigma^2} - 1)e^{2\mu + \sigma^2} \\ &= (e^{\sigma^2} - 1)e^{2\frac{2\log f_{\mathrm{mean}} - \sigma^2}{2} + \sigma^2} \end{aligned}$$

σ can be obtained:

$$\sigma = \sqrt{\log\left(\frac{f_{\mathrm{std}}^2}{e^{2\log(f_{\mathrm{mean}})}} + 1\right)}$$

The obtained log-normal distribution will provide a first-order approximation of the reporting delay seen by a CU. The system latency model can be used during system and algorithm design to assess the impact of the midhaul on the considered solution. A trade-off exists between the amount of information and co-ordination (DU-to-CU ratio) and the midhaul latency generated by the control signalling. Values generated for the case of 16 DUs agree well with the empirically recorded 16 DU data with regards to the mean averaged over all DUs: 266.15 µs compared to 265.95 µs. In terms of the .999, .9999 and .99999 quantiles, however, the approximation is more conservative: 868.70 µs compared to 458.31 µs, 1124.10 µs compared to 471.75 µs and 1406.10 µs compared to 956.50 µs. The difference can be used as a safety margin during system design.

The structure of the CU-DU latency can be further modelled into peaks and a tail as seen in Figure 8. The peaks are comprised of N_{DU} log-normals. These contain the bulk of the probability mass and represent the typical case in terms of communication latency. Each peak contains roughly the same number of values indicating that all cases are equally likely to occur. This reflects the queueing experienced by DU messages over the shared midhaul. Larger latencies form a long tail with a shape dependent on N_{DU}. Fitting the log-normals to the empirical data yields:

$$p_{peak} = 53.625 + \frac{N_{DU}}{2} + 26.1047n$$

$$\mu = \begin{cases} \log\left(\frac{p_{peak}^2}{\sqrt{4.25+p_{peak}^2}}\right), & \text{if n} = 0 \\ \log\left(\frac{p_{peak}^2}{\sqrt{10+p_{peak}^2}}\right), & \text{otherwise} \end{cases}$$

$$\sigma = \begin{cases} 0.03, & \text{if n} = 0 \\ \sqrt{\log\left(1 + \frac{10+n}{p_{peak}^2}\right)}, & \text{otherwise} \end{cases}$$

where n is in $[0, N_{DU}]$. Total variation distance (TVD) quantifies the difference between two probability distributions. TVD is calculated as:

$$\mathrm{d}_{\mathrm{tv}} = \frac{\sum_{x \in X} |p(x) - q(x)|}{2}$$

where X are the latency values being compared, $p(x)$ are the empirical latency bin counts and $q(x)$ are the predicted latency bin counts. Comparing the predicted values to the empirical ones using a bin size of 10 µs for $N_{DU} = 1, 2, 4, 8, 16$ yields TVD values average over all DUs of: 0.013, 0.144, 0.160, 0.142, 0.234. The reason for the larger value in the $N_{DU} = 16$ likely lies in the dissimilar behaviour of DU-6 and DU-12. Their different latency distribution results in probability mass not aligned with the peaks generated by other DUs, which the model fails to capture.

A transition exists between the typical and high latency regime. Estimation of this point can be done by setting a probability threshold for latencies considered rare. The transition point can be approximated by $\mu + 2\sigma$, which for $N_{DU} = 1, 2, 4, 8, 16$ yields 65 µs, 93 µs, 150 µs, 264 µs and 491 µs. In each case, the late rate will be less than 0.01. Such an approximation is useful to determine whether the log-normals or the tail sets the late rate for a given TTI duration. Since the majority of probability mass is in the typical area and there are N_{DU} equal peaks in the model, the late rate depends on whether the TTI length considered is above or below the transition point. If the target TTI duration is below, there will a late rate roughly proportional to the number of peaks on both sides of the target.

4.4 Midhaul Impact Assessment

As an illustrative example, the created midhaul latency profile was used in a simulation assessing the UE position estimate error induced by CU-DU communication. One use-case is for network-provided location data to complement global navigation satellite systems (GNSS) [44]. When UEs enter deep urban canyons, GNSS performance degrades. The RAN can then provide a replacement or augmentation system to enhance accuracy and reliability.

In the simulation scenario, two DUs are used by a CU to track a single moving UE's position, see Figure 4. The CU is assumed to also control other DUs that are not involved in the tracking of the considered UE. For each report sent to the CU, the latency is drawn randomly in a two-step process from a mixture distribution.

First, the mixture component for the DUs are selected using a uniform distribution. Doing so simulates the multi-modal distribution seen in Figure 8. This represents the uncertainty as to the ordering of DU reports. Each mixture component represents a particular position in the queue.

Second, a log-normally distributed sample is generated for both the command latency and the report latency. Latencies are drawn from the statistics of the component picked in the first step.

Table 5 presents the positioning error induced for the case of one CU and two DUs. Values for the two DUs are drawn from the 16 DU case above even though only two DUs are used to serve the UE being studied. In an actual deployment, the CU would also handle other UEs using other DUs and thus have to process their reports as well. Latency values were obtained by using the command and reporting latencies (Tables 2, 3a and e) to compute the position error resulting from delays in the midhaul for various UE speeds. The values in the table represent the whole area served by the two DUs in question. The average is taken over 10^6 locations and the realisation of the log-normally distributed command and reporting random variables.

Positioning targets for 5G aim to support [45] cars with speed up to 200 $\mathrm{km\,h^{-1}}$ and less than one meter accuracy in 95 % of the service area. Results in Table 5 show that midhaul communication latency will not prevent meeting the positioning accuracy targets set for 5G.

The delay model can also be taken into account during algorithm design. Algorithms can be then be made robust against the expected variation in midhaul communication latency. For instance, a scheduler can account for the latency between CU and DU when making its decisions. Decisions could be made sufficiently in advance to account for expected jitter to reduce the likelihood of the scheduling instructions reaching the DU late. Missing the deadline for sending scheduling information from CU to DU might result in no transmissions, thus weakening spectral efficiency and jeopardizing application level latency targets. In general, the latency for a desired reliability level can be obtained by computing its corresponding quantile using the delay estimation function:

$$q(\alpha) = e^{\mu + \sigma q_N(\alpha)} \tag{5}$$

where α is the desired quantile, μ and σ are parameters of the log-normal distribution and q_N is the corresponding quantile of the normal distribution.

Table 5 UE positioning error obtained in midhaul latency simulation results

User speed ($\mathrm{km\,h^{-1}}$)	Min (cm)	Median (cm)	99.9 % (cm)	99.99 % (cm)	99.999 % (cm)
6	0.09	0.17	0.35	0.45	0.59
30	0.50	0.87	1.74	2.26	2.93
60	0.93	1.75	3.48	4.40	5.53
90	1.48	2.62	5.22	6.80	8.37
120	1.79	3.49	6.98	9.04	11.92
200	3.28	5.82	11.60	14.73	19.15

One CU controlling two DUs with the latency distribution of the 16 DU case above. DUs are 25 m from each other and the UE travels in a line parallel to DU 1-DU 2. Values rounded to two decimal place.

5 Conclusion

Next-generation RAN architectures will require increased amounts of inter-node co-operation due to designs splitting functions into separate hardware. The latency and jitter of the midhaul joining these separate nodes thus impacts overall system performance. The midhaul performance of the RAN must be taken into account when networks, just as the behaviour of the channel is considered. Failure to account for the different characteristics of distributed architectures may jeopardize the network ability to meet its targets.

A testbed was used to obtained data on the performance of GPOS-based C-RAN platforms operated on commodity hardware. Functionality was distributed between one CU and multiple virtualized DUs and scheduled using a shared notion of current time. Both the CU-to-DU command latency and DU-to-CU reporting latency were studied. Results show that the presented commodity hardware implementation offers adequate performance. The collected data also indicates that while the typical command or reporting latency is close to the average, when deviation do occur, they can be orders of magnitude larger and occur too frequently to support ultra-reliable communication (99.999 % reliability requirement). This is expected considering the testbed's soft–real-time nature but warrants further work to improve bounds in order to support more applications.

Midhaul performance variability was observed to occur between DUs and across time. This impacts RAN and MEC design. Creating or moving DU instances on servers can result in a change in latency behaviour for existing instance. Algorithms must therefore be able to cope with variable time budgets.

Measurement results were used to create a model of scaling performance as a function of the number of DUs controlled by a single CU. Modelling the RAN latency enables improved development and design of new networks. A model of latency enables determining the supported number of DUs per CU given a particular set of requirements, such as a given functional split. Changing the number of DUs assigned to each CU enables modifying the trade-off between the quantity of information available for co-ordination and the extra latency and jitter created by collecting said information. Network and algorithm design could benefit from such a model for use in optimization.

Additional work is required to better understand the both the causes and implications of the observed timing indeterminism. Tighter bounds could enable support for features with tighter timing and reliability requirements. Improved understanding of the causes of jitter might also offer insights into suitable scheduling intervals, functional splits and the optimal ratio of DUs to CU as well as the distribution of tasks between them.

Funding Open Access funding provided by Aalto University. This work has been supported in part by the Business Finland project Clodify 6G (grant number 8822/31/2022).

Data Availability The datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request.

Declarations

Competing Interests The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

1. Olwal, T. O., Djouani, K., & Kurien, A. M. (2016). A survey of resource management toward 5G radio access networks. *IEEE Communications Surveys Tutorials, 18*(3), 1656–1686.
2. Katz, M., Matinmikko-Blue, M., Latva-Aho, M. (2018). 6Genesis flagship program: Building the bridges towards 6G-enabled wireless smart society and ecosystem. In IEEE Latin-American conference on communications (LATINCOM), pp. 1–9 . https://doi.org/10.1109/LATINCOM.2018.8613209.
3. Zhang, L., Liang, Y., & Niyato, D. (2019). 6G visions: Mobile ultra-broadband, super internet-of-things, and artificial intelligence. *China Communications, 16*(8), 1–14. https://doi.org/10.23919/JCC.2019.08.001
4. Saad, W., Bennis, M., & Chen, M. (2019). A vision of 6G wireless systems: Applications, trends, technologies, and open research problems. IEEE Network, pp. 1–9 . https://doi.org/10.1109/MNET.001.1900287.
5. Kamel, M., Hamouda, W., & Youssef, A. (2016). Ultra-dense networks: A survey. *IEEE Communications Surveys Tutorials, 18*(4), 2522–2545. https://doi.org/10.1109/COMST.2016.2571730
6. Yu, W., Xu, H., Zhang, H., Griffith, D., & Golmie, N. (2016). Ultra-dense networks: Survey of state of the art and future directions. In International conference on computer communication and networks (ICCCN), pp. 1–10.
7. Zhang, J., Chen, S., Lin, Y., Zheng, J., Ai, B., & Hanzo, L. (2019). Cell-free massive MIMO: A new next-generation paradigm. *IEEE Access, 7*, 99878–99888.
8. Singhal, D., Kunapareddy, M., Chetlapalli, V., & James, V. B., Akhtar, N. (2011). LTE-advanced: Handover interruption time analysis for IMT-A evaluation. In International conference on signal processing, communication, computing and networking technologies, pp. 81–85.
9. Adedoyin, M. A., & Falowo, O. E. (2020). Combination of ultra-dense networks and other 5G enabling technologies: A survey. *IEEE Access, 8*, 22893–22932. https://doi.org/10.1109/ACCESS.2020.2969980
10. Kim, E., Lee, J., Kim, Y., & Hong, E. (2019). Analysis of the optimal number of clusters in UDN environment. In IEEE Asia Pacific wireless communications symposium (APWCS), pp. 1–4 . https://doi.org/10.1109/VTS-APWCS.2019.8851643.
11. Bilen, T., Canberk, B., & Chowdhury, K. R. (2017). Handover management in software-defined ultra-dense 5G networks. *IEEE Network, 31*(4), 49–55. https://doi.org/10.1109/MNET.2017.1600301
12. Hu, B., Wang, Y., Wang, C., & Wang, L. (2017). A user-centric clustering method for mobility management in ultra-dense networks. In International conference on wireless communications and signal processing (WCSP), pp. 1–5 . https://doi.org/10.1109/WCSP.2017.8170946.
13. Park, J., Jung, S.Y., Kim, S., Bennis, M., & Debbah, M. (2016). User-centric mobility management in ultra-dense cellular networks under spatio-temporal dynamics. In IEEE global communications conference (GLOBECOM), pp. 1–6 . https://doi.org/10.1109/GLOCOM.2016.7842367.
14. Tesema, F.B., Awada, A., Viering, I., Simsek, M., & Fettweis, G.P. (2016). Fast cell select for mobility robustness in intra-frequency 5G ultra dense networks. In IEEE international symposium on personal, indoor, and mobile radio communications (PIMRC), pp. 1–7 . https://doi.org/10.1109/PIMRC.2016.7794931.
15. Zhu, J., Zhao, M., & Zhou, S. (2018). An optimization design of ultra dense networks balancing mobility and densification. *IEEE Access, 6*, 32339–32348. https://doi.org/10.1109/ACCESS.2018.2845690
16. Sexton, C., Kaminski, N. J., Marquez-Barja, J. M., Marchetti, N., & DaSilva, L. A. (2017). 5G: Adaptable networks enabled by versatile radio access technologies. *IEEE Communications Surveys Tutorials, 19*(2), 688–720.
17. Kitindi, E. J., Fu, S., Jia, Y., Kabir, A., & Wang, Y. (2017). Wireless network virtualization with SDN and C-RAN for 5G networks: Requirements, opportunities, and challenges. *IEEE Access, 5*, 19099–19115. https://doi.org/10.1109/ACCESS.2017.2744672

18. Thaalbi, K., Missaoui, M.T., & Tabbane, N. (2018). Short survey on clustering techniques for RRH in 5G networks. In International conference on communications and networking (ComNet), pp. 1–5 . https://doi.org/10.1109/COMNET.2018.8622300.
19. Ren, H., Liu, N., Pan, C., Elkashlan, M., Nallanathan, A., You, X., & Hanzo, L. (2018). Low-latency C-RAN: An next-generation wireless approach. *IEEE Vehicular Technology Magazine, 13*(2), 48–56.
20. Rost, P., Berberana, I., Maeder, A., Paul, H., Suryaprakash, V., Valenti, M., Wübben, D., Dekorsy, A., & Fettweis, G. (2015). Benefits and challenges of virtualization in 5G radio access networks. *IEEE Communications Magazine, 53*(12), 75–82. https://doi.org/10.1109/MCOM.2015.7355588
21. Luo, J., Chen, Q., & Tang, L. (2018). Reducing power consumption by joint sleeping strategy and power control in delay-aware C-RAN. *IEEE Access, 6*, 14655–14667.
22. Pei, L., Shen, G., Yi, S., Zhiwen, P., Xiaohu, Y., & Fei, D. (2018). Optimal BS sleeping ratio for energy-delay tradeoff in wireless-backhauling UDN. In IEEE vehicular technology conference (VTC-Fall), pp. 1–5.
23. Morocho Cayamcela, M.E., & Lim, W. (2018). Artificial intelligence in 5G technology: A survey. In International conference on information and communication technology convergence (ICTC), pp. 860–865.
24. Reghenzani, F., Massari, G., & Fornaciari, W. (2019). The real-time Linux kernel: A survey on PREEMPT_RT. *ACM Computing Surveys*. https://doi.org/10.1145/3297714.
25. Maiza, C., Rihani, H., Rivas, J. M., Goossens, J., Altmeyer, S., & Davis, R. I. (2019). A survey of timing verification techniques for multi-core real-time systems. *ACM Computing Surveys*. https://doi.org/10.1145/3323212.
26. Brandt, S.A., Banachowski, S., Caixue Lin, & Bisson, T. (2003). Dynamic integrated scheduling of hard real-time, soft real-time, and non-real-time processes. In IEEE real-time systems symposium (RTSS), pp. 396–407.
27. Hong, D., Shin, J., Woo, S., & Moon, S. (2017). Considerations on deploying high-performance container-based NFV. In Workshop on cloud-assisted networking, pp. 1–6 . https://doi.org/10.1145/3155921.3155925.
28. Abeni, L., Balsini, A., & Cucinotta, T. (2019). Balsini A Cucinotta T. *Container-Based Real-Time Scheduling in the Linux Kernel SIGBED Rev, 16*(3), 33–38. https://doi.org/10.1145/3373400.3373405.
29. Nayak, N.G., Dürr, F., & Rothermel, K. (2016). Time-sensitive software-defined network (TSSDN) for real-time applications. In International conference on real-time networks and systems, pp. 193–202 . https://doi.org/10.1145/2997465.2997487.
30. Alimi, I. A., Teixeira, A. L., & Monteiro, P. P. (2018). Toward an efficient C-RAN optical fronthaul for the future networks: A tutorial on technologies, requirements, challenges, and solutions. *IEEE Communications Surveys Tutorials, 20*(1), 708–769.
31. Mathew, A., Srinivasan, M., & Murthy, C.S.R. (2019). Packet generation schemes and network latency implications in SDN-enabled 5G C-RANs: Queuing model based analysis. In IEEE international symposium on personal, indoor and mobile radio communications (PIMRC), pp. 1–7.
32. Giannone, F., Kondepu, K., Gupta, H., Civerchia, F., Castoldi, P., Antony Franklin, A., & Valcarenghi, L. (2019). Impact of virtualization technologies on virtualized RAN midhaul latency budget: A quantitative experimental evaluation. *IEEE Communications Letters, 23*(4), 604–607. https://doi.org/10.1109/LCOMM.2019.2899308
33. Buzzi, S., & D'Andrea, C. (2017). Cell-free massive MIMO: User-centric approach. *IEEE Wireless Communications Letters, 6*(6), 706–709. https://doi.org/10.1109/LWC.2017.2734893
34. Gelabert, X., Qvarfordt, C., Costa, M., Kela, P., & Leppänen, K. (2016). Uplink reference signals enabling user-transparent mobility in ultra dense networks. In IEEE international symposium on personal, indoor, and mobile radio communications (PIMRC), pp. 1–6. https://doi.org/10.1109/PIMRC.2016.7794816.
35. 3GPP: Study on new radio access technology: Radio access architecture and interfaces (Release 14) (2017)
36. Larsen, L. M. P., Checko, A., & Christiansen, H. L. (2019). A survey of the functional splits proposed for 5G mobile crosshaul networks. *IEEE Communications Surveys & Tutorials, 21*(1), 146–172. https://doi.org/10.1109/COMST.2018.2868805
37. Saha, R.K., Nanba, S., & Nishimura, K. (2018). Clustering and centralized resource scheduling of 3D in-building small cells for intra MAC functional split control-/user-plane decoupled CRAN. In 2018 IEEE international conference on communications (ICC), pp. 1–7 . https://doi.org/10.1109/ICC.2018.8422357.
38. Das, R. M., Sree, L. S., Ponnekanti, S., & Paunovic, M. (2019). Key enablers to deliver latency-as-a-service in 5G networks. *Telecommunications forum (TELFOR)* (pp. 1–4). Manhattan, New York: IEEE.

39. Anritsu: 5G Networks require quality of service white paper. Accessed Nov 19 (2020). https://dl.cdn-anritsu.com/en-en/test-measurement/files/Technical-Notes/White-Paper/mt1000a-ms2090a-5g-er1100.pdf.
40. IEEE: IEEE standard for packet-based fronthaul transport networks. IEEE Std 1914.1-2019, 1–94 (2020). https://doi.org/10.1109/IEEESTD.2020.9079731.
41. Alliance, O.-R. (2020). O-RAN. Accessed Apr 24. https://www.o-ran.org/.
42. Mathworks: Matlab. Accessed July 24 (2020). https://mathworks.com/products/matlab.html?s_tid=hp_products_matlab.
43. Mathworks: Matlab Curve Fitting Toolbox. Accessed March 17 (2021). https://se.mathworks.com/products/curvefitting.html.
44. Abu-Shaban, Z., Seco-Granados, G., Benson, C.R., & Wymeersch, H. (2020). Performance analysis for autonomous vehicle 5g-assisted positioning in GNSS-Challenged environments. In IEEE/ION position, location and navigation symposium (PLANS), pp. 996–1003 . https://doi.org/10.1109/PLANS46316.2020.9109885.
45. Keating, R., Säily, M., Hulkkonen, J., & Karjalainen, J. (2019). Overview of positioning in 5G new radio. In International symposium on wireless communication systems (ISWCS), pp. 320–324 . https://doi.org/10.1109/ISWCS.2019.8877160.

Nicolas Malm received the Bachelor's degree in communications engineering from the Helsinki University of Technology, Finland, in 2015, and the Master's degree in communications engineering from Aalto University, Finland, in 2016. He is currently pursuing a doctoral degree with the Department of Communications and Networking, Aalto University, focusing on software-defined radio research.

Kalle Ruttik received the Diploma degree in engineering from Tallinn Technical University, in 1993, and the Lic.Tech. degree from the Helsinki University of Technology, in 1999. He is currently a Teaching Researcher with the Department of Communications and Networking (ComNet), School of Electrical Engineering, Aalto University. His research interests include radio physical layer algorithms, software radio, base station implementation, and cloud RAN.

Olav Tirkkonen is associate professor in communication theory at the Department of Communications and Networking in Aalto University, Finland, where he has held a faculty position since 2006. He received his M.Sc. and Ph.D. degrees in theoretical physics from Helsinki University of Technology in 1990 and 1994, respectively. Between 1994 and 1999 he held post-doctoral positions at the University of British Columbia, Vancouver, Canada, and the Nordic Institute for Theoretical Physics, Copenhagen, Denmark. From 1999 to 2010 he was with Nokia Research Center (NRC), Helsinki, Finland. In 2016-2017 he was Visiting Associate Professor at Cornell University, Ithaca, NY, USA. He has published some 300 papers, is the inventor of some 85 families of patents and patent applications and is coauthor of the book "Multiantenna transceiver techniques for 3G and beyond". His current research interests are in coding for random access and quantization, quantum computation, and machine learning for cellular networks. He is an associate editor of IEEE Transactions on Wireless Communication.