DOI: 10.1109/TMC.2018.2793859

WizHaul: On the Centralization Degree of Cloud RAN Next Generation Fronthaul

Andres Garcia-Saavedra, Josep Xavier Salvat, Xi Li, Xavier Costa-Perez

Abstract—Cloud Radio Access Network (C-RAN) will become a main building block for 5G. However, the stringent requirements of current fronthaul solutions hinder its large-scale deployment. In order to introduce C-RAN widely in 5G, the next generation fronthaul interface (NGFI) will be based on a cost-efficient packet-based network with higher path diversity. In addition, NGFI shall support a flexible functional split of the RAN to adapt the amount of centralization to the capabilities of the transport network. In this paper we question the ability of standard techniques to route NGFI traffic while maximizing the centralization degree—the goal of C-RAN. We propose two solutions jointly addressing both challenges: (i) a nearly-optimal backtracking scheme, and (ii) a low-complex greedy approach. We first validate the feasibility of our approach in an experimental proof-of-concept, and then evaluate both algorithms via simulations in large-scale (real and synthetic) topologies. Our results show that state-of-the-art techniques fail at maximizing the centralization degree and that the achievable C-RAN centralization highly depends on the underlying topology structure.

Index Terms—5G, Cloud RAN, Next Generation Fronthaul, Crosshaul

1 Introduction

Architectural flatness and decentralization—*pushing intelligence out to the edge*—has traditionally been an axiomatic criterion to design 3G/4G systems with affordable topological flexibility and high capacity; we refer to this architecture as Distributed Radio Access Network (D-RAN). More recently, an opposing paradigm, termed Cloud RAN (C-RAN), has gained momentum and holds itself out as a promising solution for 5G. In its purest form, the functionality of a base station (BS) is fully decoupled from the radio unit (RU) and it is virtualized into a centralized cloud computing platform or central unit (CU) [1]. (Virtual) BSs/CUs are connected to the evolved packet core (EPC), e.g charging, gateways to Internet, etc., via a backhaul (BH) network. On the other side, RUs exchange digitized IQ radio samples[1] with CUs through a high-capacity fronthaul (FH) network, typically using serial line interfaces like CPRI (Common Public Radio Interface) or OBSAI (Open Base Station Architecture Initiative) [1]. This approach has been shown to improve spectrum efficiency and reduce costs (pooling gains) in certain setups [2], [3]. However, its benefits become questionable in many realistic large-scale deployments for 5G. This is due to the stringent requirements on the fronthaul, which can only be met in practice by costly fiber point to point links. Moreover, such costly dedicated infrastructure cannot be amortized due to the following additional limitations [4]:

- Bandwidth usage is constant and independent of user load, i.e. no statistical multiplexing;
- Data rate demand grow linearly with the number of antennas, which disallows massive MIMO;
- Low (or none) path diversity between RUs and CUs (poor resilience, high inefficiency);
- No infrastructure reuse: FH and BH are incompatible in terms of interfaces, data or control planes.

Fig. 1 illustratively describes the aforementioned traditional RAN architectures D-RAN and C-RAN.

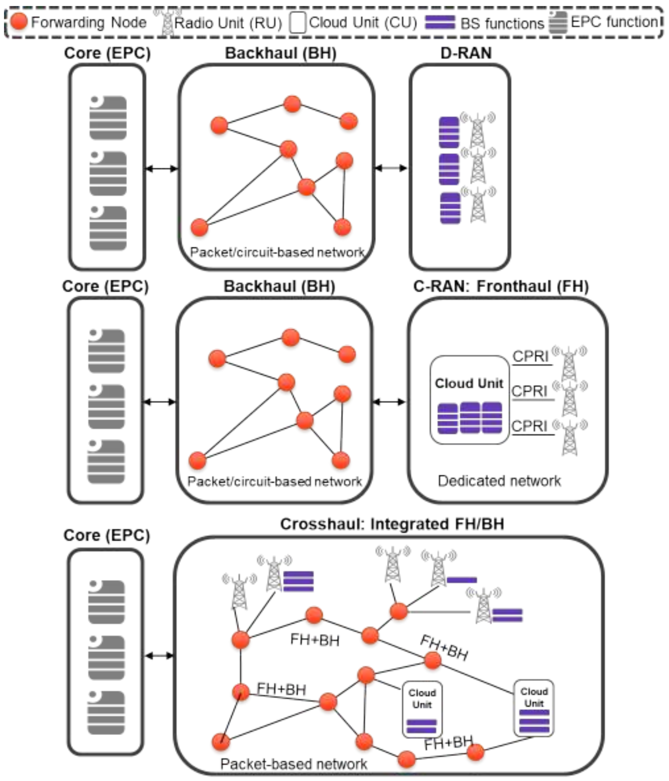


Fig. 1: Traditional Distributed RAN (D-RAN) *vs* Cloud RAN (C-RAN) *vs* Next-Generation Fronthaul (Crosshaul).

- *A. Garcia-Saavedra, J. X. Salvat, X. Li and X. Costa-Perez are with NEC Laboratories Europe, Germany.*
E-mail: {andres.garcia.saavedra, josep.xavier.salvat, xi.li, xavier.costa}@neclab.eu

1. In-Phase and Quadrature (IQ) data is a representation of the changes in amplitude and phase of modulated signals.

TABLE 1: Functional splits analysis in [5]. LTE scenario: 1 user/TTI, 20 MHz bandwidth; Downlink: MCS (modulation and coding scheme) index 28, 2x2 MIMO, 100 Resource Blocks (RBs), 2 transport blocks of 75376 bits/subframe; Uplink: MCS 23, 1x2 SIMO, 96 RBs, 1 transport block of 48936 bits/subframe.

Split #	**LTE BS Functional decomposition**	**DL/UL BW req. (Mb/s)**	**Delay req. (μs)**	**Gains**
A	RRC - PDCP	151/48	30e3	• Enables L3 functionality for multiple small cells to use the same HW; • Enhanced mobility across nodes w/o inter-small cell data forwarding/signaling; • Reduced mobility-related signaling to the mobile core segment; • No X2 endpoints between small cells and macro eNBs; • Control plane and user plane separation.
B	PDCP - RLC	151/48	30e3	• Enables L3 and some L2 functionality to use the same HW.
C	RLC - MAC	151/48	6e3	• Resource sharing benefits for both storage and processor utilization.
D	MAC I - MAC II	151/49	6e3	• Synchronized coordination and control of multiple cells; • Coordination across cells enables CA, CoMP, eICIC or cross carrier scheduling.
E	MAC - PHY	152/49	250	• Enhancements to CoMP with RU frame alignment and centralized HARQ.
F	PHY split I	173/452	250	• More opportunities to disable parts of the CU at quiet times to save power; • Central L1 CU can be scaled based on average utilisation across all cells; • Smaller CU results in less processing resource and power saving; • Enhancements to joint reception CoMP with uplink PHY level combining.
G	PHY split II	933/903	250	
H	PHY split III	1075/922	250	
I	PHY split IIIb	1966/1966	250	
J	PHY split IV	2457.6/2457.6	250	

1.1 Flexible Centralization: Crosshaul

In light of the above, a re-design of the fronthaul is taking place in multiple fora, including industry [4], [6] and major standardization bodies (IEEE [7], 3GPP, [8], Small-Cell forum [5]) that will converge into *the next generation fronthaul interface* (NGFI) tackling the aforementioned limitations (see Fig. 1). Two pivotal paradigms steer its design. First, NGFI shall be based on a simpler packet-based transport protocol which would enable statistical multiplexing, infrastructure reuse and higher degrees of freedom for routing. Note that this blurs the separation between BH and FH. In fact, their convergence towards a common packet-based segment (*Crosshaul* hereafter) is being studied under the umbrella of the 5G public-private partnership (5G-PPP) projects [9].

FH/BH coexistence in a common packet-based network faces an important challenge: the tough requirements of full C-RAN are now subject to more limited—and likely shared—transport resources. *However, retaining as much centralization as possible, when full offloading of BS functionality is unfeasible due to transport constraints, would be desirable.* This leads to NGFI's second driving concept: a flexible split of RAN functionality. The idea is to divide a classic BS into a set of functions that can either be processed at the RU or offloaded into a CU, depending on the transport requirements and centralization needs. In this way we can better balance cost/performance (the more aggressive the offloading, the higher the gains) and requirements (the softer the offloading, the more relaxed the network constraints).

Table 1 illustrates the trade-off between (qualitative) gains and (quantitative) network requirements for different splits in LTE. Split J is equivalent to pure C-RAN, i.e., all functions are centralized enabling maximum gain, namely, interference coordination mechanisms such as CoMP (Coordinated MultiPoint) are enabled [6], computational resources are pooled and can be scaled based on demand, etcetera, at the cost of the toughest network requirements. Note that, as we relax the amount of centralization, the network requirements are also reduced at the loss of centralization gains. For instance, interference management mechanisms are less efficient as channel state information (CSI) for all RUs is not aggregated into a central location when the physical layer (PHY) functions are not centralized. However, centralizing e.g., PDCP (Packet Data Convergence Protocol), still allows to obtain some computational pooling gains and a common radio resource control (RRC) layer to mitigate mobility-related signaling overhead. The conclusion is that there is an inherent trade-off between network requirements and system gains (which can be very heterogeneous and somewhat hard to measure objectively) when deciding the degree of base station function centralization. We refer the reader to [5] for further details into this analysis. Similar conclusions arise in other studies, e.g., [8] by 3GPP.

Note that these heterogeneous network requirements are due to a new degree of freedom: choice of functional splits. Additional requirements from user-end applications may impose further constraints, e.g., an ultra-low latency application may have tighter latency tolerance than, e.g. split B in Table 1. For simplicity, in this paper we focus on the requirements attainable to a diverse set of functional splits only, which is particular for this new type of systems.

1.2 The Problem

Though industry and academia advocate towards this direction for 5G [4]–[6], [9]–[12], to our knowledge, no study has looked into the implications that a flexible RAN centralization poses on path computation with high path diversity. In fact, despite its benefits, a joint implementation of both, flexible RAN centralization and a Crosshaul path computation, is inherently challenging.

On the one hand, a proper choice of BS split points depends on the transport capabilities of the Crosshaul, like available bandwidth, latency or jitter, which in turn are unknown until all paths have been computed (note that in Crosshaul, links can be shared). On the other hand, an adequate routing across the Crosshaul requires knowledge of the BS split points, because such choices set the demands of the flows to route (see Table 1). *Therefore, we face a coupled problem where routing and selection of RAN splits must be optimized jointly, a problem that to the best of our knowledge has not been identified before.*

In order to illustrate this, let us set up a simple scenario, depicted in Fig. 2, with one CU and 5 RUs. We now analyze different strategies to optimize our example.

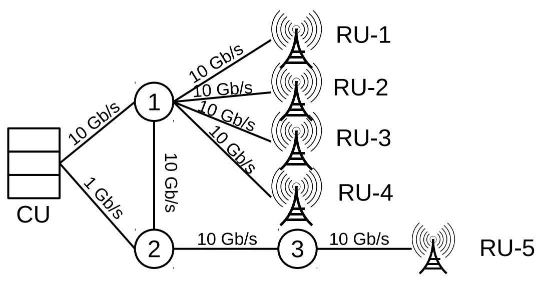


Fig. 2: Simple scenario.

1.2.1 *Best split, then best routing*

The first (rather naïve) strategy is to decide first on the RAN split—without routing knowledge—and then, based on such decision, find a feasible set of paths. Given a split, the problem can be seen as an Unsplittable Flow Problem (UFP), which is NP-Hard [13]. Regarding Fig. 2, we may be tempted to pick split J (Table 1) for all BSs to grasp the advantages of full centralization. This choice, however, implies that five 2.5-Gb/s flows shall be routed from the CU to each RU, which is unfeasible (this is obvious in such a simple example). It thus becomes evident that a (mildly) better approach is to set the paths first and then, based on the transport capabilities given by these routes, find a good feasible RAN split, which leads to our next strategy.

1.2.2 *Best routing, then best split*

The second strategy is to decide on routing first, based on some criteria (e.g. shortest-path), with no knowledge of each flow demand (which highly depends on the RAN split). For instance, if we apply a max-min criterion [14], the outcome would be five 2-Gb/s flows. Now, upon such routing knowledge, we could find the highest feasible RAN split. This is also an NP-Hard problem since it could be seen as a BAG-UFP problem in a tree [15] which, to our knowledge, does not have approximation algorithms. However, given the simplicity of our example, we can find, via exploration, that the solution with highest centralization degree is split I for all BSs (Table 1). However, as we show next, we can still do better; in fact our simulation results in §8 show that this strategy leads to poor solutions in general.

1.2.3 *Joint decision*

If, in contrast, we exhaustively looked across all possible combinations of split and routing choices between CU and RUs, we would find out that we could route the traffic of the RU-5 through path CU-2-3 using split G. This releases additional capacity in path CU-1 for the remaining RUs which can now use split J, maximizing in this way the advantages of centralization of the whole system. Obviously, an exhaustive search deems unfeasible in real-systems with hundreds of RUs and switches, and clearly the joint problem is also NP-hard (as it could be particularized to either of the former problems). *Finding an algorithmic solution to this joint problem is precisely the goal of this work.*

1.3 **Our Goal and Contributions**

Despite the enormous interest in the design of NGFI, driven by industry ([4], [16]) and standardization (NGMN [17], 3GPP [8], IEEE [7], Small-Cell forum [5]), to the best of our knowledge, we are the first to formalize a new problem arising of such flexible fronthaul: the coupled problem of optimally taking routing and BS function placement decisions. The above analysis makes evident that functional split selection complicates an already hard problem (QoS routing) and thus traditional solutions do not apply here. *In light of this, we develop algorithmic solutions to this new problem, conveniently wrapped into a decision-making engine named* **WizHaul**, *that serve two purposes*:

- **Network planning.** WizHaul solves a problem that is essential for any Crosshaul planning tool: that of deciding the optimal BS functional split and link provisioning between CUs and RUs. We evaluate the ability of WizHaul to maximize the centralization degree in §8;
- **Fault-tolerance.** WizHaul is a centralized decision engine that can be integrated in Software-Defined Network (SDN)-based platforms to support fault-tolerance in a more dynamic fashion,[2] as we show with a proof-of-concept implementation in §7.

The latter is useful when link changes occur. For instance, an operator may make a configuration choice based on measurements available at a planning phase. However, interfering links may appear with time, possibly rendering the original choice invalid. RAN cloudification, already demonstrated in the past (see e.g. [18]) and realized in some early commercial solutions (see [16]) enables these new use cases.

In summary, the contributions of this paper are:

- **Optimization Framework**. To the best of our knowledge, this is the first work that formalizes a new problem arising from the flexibility that NGFI introduces: optimal routing choices and optimal BS function placement choices are coupled and hard to find in general;
- **Decision-making algorithms**. Based on our analytical framework, we devise two algorithms: an optimal branch-and-bound approach, and a heuristic suitable for large-scale topologies or small-timescale decisions, wrapped up into a common entity named WizHaul;
- **Experimental Proof-of-Concept**. We integrate WizHaul as an application on top of an SDN platform and demonstrate its feasibility in fault-tolerance use cases with a proof-of-concept;
- **Performance Evaluation with Synthetic and Real Topologies**. We evaluate WizHaul both with real-world networks from two major European operators and synthetic topologies to generalize our results.

The rest of the paper is organized as follows. Related works are revised in §2. Our system model and optimization framework are introduced in §3 and §4. §5 presents WizHaul's algorithms: a branch-and-bound approach and a heuristic solution. WizHaul is validated experimentally in a fault-tolerance use case in §7 and as a planning tool with large-scale synthetic and real-world topologies in §8. Concluding remarks are in §9.

2 Related Work

C-RAN has fostered a lot of work lately [1], [3], [18]–[22]. An early study of the feasibility of pure C-RAN in

2. The timescale of these configuration changes is days or even weeks, and thus we do not envision functional split changes adapting to fast events such as, e.g., mobility of single users or wireless fading.

packet-based networks is shown in [10], concluding that frame preemption (802.1Qbu), scheduled traffic (802.1Qbv) and buffers in the receivers would be needed. Very recently, distributed synchronization in 5G systems has been demostrated in [23]. The work of [24] was the first to propose BS functional split points different from pure C-RAN to relax constraints while retaining some centralization degree. Some other work has studied the trade-offs between qualitative benefits and quantitative costs of different splits for simple scenarios [5], [12], [25], [26], the impact of packetization and scheduling in simple setups [27], [28] and, recently, the convergence of Data over Cable Service Interface Specification (DOCSIS) in broadband cable and C-RAN via functional splits [29]. The design of NGFI is driven by different standardization fora: IEEE [7], 3GPP [8], NGMN [17], Small-Cell forum [5] or eCPRI [30] which makes evident the interest of industry in a more flexible fronthaul architecture. For instance, driven by IEEE 1914 work, [31] proposes a two-level crosshaul architecture. Though all this work shows that RAN centralization has many advantages and architectural advantages, we are, to the best of our knowledge, the first to study the implications that a flexible functional split poses on path computation in large-scale packet based networks, and propose solutions to address them.

RAN virtualization has been matter of study in [2], [18], [32], [33]. We rely upon all this work making a flexible fronthaul interface, where BS functions can be virtualized and placed on general-purpose processors, possible.

The most similar problem is the multi-commodity unsplittable flow problem (UFP). In this problem, we have to route multiple flows using single-paths subject to network constraints. There are a few versions depending on the objective [34]. MAX-EDP (maximum edge-disjoint path problem) aims to route a subset of flows across disjoint paths maximizing a profit function [35]. MAX-UFP generalizes it allowing links to be shared [36]. ROUND-UFP (UFP with rounds) partitions the set of flows into the minimum number of subsets with feasible routing instances [37]. BAG-UFP divides all the flows into bags or baskets and finds the subset of flows of maximum profit such that each flow belongs to a different bag [15]. Though most of the algorithms exploit particular graph structures or make the no-bottleneck assumption (no link has less capacity than the highest demand), some works focus on general graphs [13]. ROUND-UFP is NP-Hard as it contains the BIN-PACKING problem. MAX-UFP and Bag-UFP contain the KNAPSACK problem as a particular case and thus they are NP-Hard too. Another related problem is the Virtual Network Embedding (VNE) problem and the problem of placing chains of virtual functions [38]. In our problem, however, (i) we must find paths for *all* RUs (all flows must be admitted), (ii) the flow demands are given by another variable (splits), and (iii) both delay and bandwidth requirements must be met, requirements that render the above solutions unfit.

The problem of routing video flows while optimizing their coding rate (which define the network demands, like BS splits in our case) can also be related. However, given the short timescale in which decisions shall be taken, the related work focus on multipath routing (to exploit coding diversity), single flows and/or make topology simplifications to reduce complexity (e.g. [39]).

3 SYSTEM MODEL

We consider a scenario comprised of M CUs $\mathcal{B} := \{B_1, \cdots, B_M\}$, N RUs $\mathcal{R} := \{R_1, \cdots, R_N\}$, and a packet-based network connecting RUs to CUs by means of packet-switching nodes $\mathcal{V} := \{v_1, \cdots |\mathcal{V}|\}$. Nodes communicate via network links such that $l_{i,j} = 1$ denotes an existing link between nodes i and j and $l_{i,j} = 0$ otherwise. The collection of all nodes is denoted by $\mathcal{N} := \mathcal{B} \cup \mathcal{R} \cup \mathcal{V}$. Note that we do not make any assumptions on topology structure in an attempt to shed some light for arbitrary large-scale scenarios. Also, without loss of generality we will focus on the downlink case.

3.1 Flexible RAN functional split

RUs are in charge of analog processing and digital-to-analog conversion. The remaining functionality of a traditional BS (modulation, HARQ, scheduling, etc.) is split into a set of H atomic functions $\mathcal{F} := \{f_1, \cdots, f_H\}$ that can be executed by either a CU or an RU though, importantly, they must be processed sequentially. This is known as *flexible functional split*. As we explained above, offloading RAN functionality into a cloud platform (CU) has the advantages of lower operational costs (e.g. common refrigeration, single-point maintenance, etc.) and capacity gains to users (joint signal processing, coordinated resource allocation, etc.) [5]. However, the delay and throughput requirements for the transport network between CUs and RUs become more stringent when a larger number of functions are offloaded.

To model this, we let $\theta_{m,n} \subseteq \mathcal{F}$ denote the subset of BS n functions assigned to CU m. In turn, $\bar{\theta}_n := \mathcal{F} \setminus \bigcup_{m \in \mathcal{B}} \theta_{m,n}$ is the subset assigned to RU n. We impose a one-to-one mapping between CUs and RUs and therefore $\theta_{m,n} = \varnothing \, \forall m \in \mathcal{B} \setminus \{A_n\}$, where A_n returns the CU assigned to RU n, i.e., we do not consider chaining functions across different CUs. In this way, we can model traditional scenarios like C-RAN (setting $\theta_{A_n,n} = \mathcal{F}$ and $\bar{\theta}_n = \varnothing \ \forall n \in \mathcal{R}$), traditional D-RAN (with $\theta_{A_n,n} = \varnothing$ and $\bar{\theta}_n = \mathcal{F} \ \forall n \in \mathcal{R}$), and any other configuration between these two.

3.2 Routing and CU assignment

We assume a flexible transport protocol (NGFI) that is able to carry IQ samples from CUs to RUs as well as traffic from different BS splits and BH traffic onto the same substrate (switching-based) Crosshaul infrastructure. Hence, our network must transport N flows (each associated with one RU) with different demands that depend on the functional split of each BS. The path followed by flow n is comprised of a subset of the forwarding nodes $\mathcal{P}_{m,n} \subseteq \mathcal{V}$ from CU $m = A_n$ to RU n such that for any $v_i \in \mathcal{P}_{m,n}$ there is exactly another $v_{j \neq i} \in \mathcal{P}_{m,n}$ with $l_{v_i,v_j} = 1$ (i.e. no loops). If CU $m \neq A_n$ (does not serve RU n), then $\mathcal{P}_{m,n} = \varnothing$. The network between CUs and RUs must satisfy the delay/throughput requirements of a given RAN split $\boldsymbol{\theta} := \{\theta_{m,n} \mid \forall m \in \mathcal{B}, \forall n \in \mathcal{R}\}$. In turn, the transport capacity depends on the routing choices $\boldsymbol{\mathcal{P}} := \{\mathcal{P}_{m,n} \mid \forall m \in \mathcal{B}, \forall n \in \mathcal{R}\}$ between CUs and RUs. Thus, as said earlier, we face a problem where routing $\boldsymbol{\mathcal{P}}$ and BS splits $\boldsymbol{\theta}$ must be optimized *jointly*.

TABLE 2: Computational costs based on the CPU execution times shown in [42] and split mapping from Table 1.

LTE subfunction (f)	Others	RLC	MAC	PHY 2	PHY 1
Relative cost (ς_f)	0.2	0.01	0.14	0.17	0.48
Split 1 (B in Table 1)	CU	RU			
Split 2 (C in Table 1)	CU		RU		
Split 3 (E in Table 1)	CU			RU	
Split 4 (G in Table 1)	CU				RU
Split 5 (J in Table 1)	CU				

3.3 Clustering

The main challenge of our problem is the large space of candidate solutions, i.e. a network has $(k \cdot M)^N \cdot (|\mathcal{F}|+1)^N$ possible settings, where k is the number of possible paths between any CU/RU. In order to reduce such huge space, we leverage the fact that joint processing of different BSs mostly makes sense if it is done within the same CU. Moreover, there is little gain to do joint processing of a set of BSs with different functional splits [40]. We thus assume that the set of RUs is partitioned into $\mathcal{Q} := \{q_1, \dots |\mathcal{Q}|\}$ clusters where q_i contains a subset of RUs constrained to the same split choice and CU assignment. Note that RUs can still follow a different route to their CU even if they belong to the same cluster. For instance $|\mathcal{Q}| = 2$ in our toy example in §1. This brings down the space of candidate solutions to $k^N \cdot M^{|\mathcal{Q}|} \cdot (|\mathcal{F}|+1)^{|\mathcal{Q}|}$. Since our focus is on joint routing and RAN centralization, we will rely on state-of-the-art clustering mechanisms (e.g., [40], [41]).

4 Optimization Framework

Our goal is threefold: (i) pair CUs to RUs (or clusters of RUs), (ii) set the paths between CUs and RUs, and (iii) choose the functional decomposition of BSs in an optimal way. To this aim, we rely upon estimates (e.g. in peak hour) of user demands at each radio site and capacity/latencies in each network link. We expect configuration changes across the Crosshaul (if possible) to happen in large timescale (hours, days or even weeks) and so we do not target adaptation to very quick events, e.g. fading events in the wireless channel. Although the capacity gains *vs.* cost trade-offs due to function centralization could be modeled to some extent [3], [21], [43], the actual benefits are much broader than such quantitative measures (see [5], descriptively condensed in Table 1) and are hard to model in general. For instance, maintenance is simplified by centralizing functions which should then reduce costs; however, the extent of these gains depends on several factors which differ across operators. In the literature today we can find some models on the trade-offs between full centralization (C-RAN) and distributed BSs (D-RAN), e.g. [3], [21]. However, these are only two split options out of the many we consider here. A recent paper [42] has experimentally studied the computational costs of different functions of Open Air Interface (OAI)'s LTE protocol stack (a well-known software-defined radio implementation); however, they do not model the gains of centralization, which remain an open issue.

In this paper, given such research gap, we use a simple model that serves our purpose, to study the implications of flexible RAN centralization in a packet-based FH, and leave a more accurate modeling of different splits gains/costs for future work. Thus, we aim at maximizing the ***degree of centralization*** γ of our system subject to network constraints:

$$\max_{\boldsymbol{\theta},\boldsymbol{\mathcal{P}}} \gamma := \left[\frac{1}{N}\sum_{n\in\mathcal{R}}\left(\alpha\sum_{f\in\theta_n}\varsigma_f(s_n) + \sum_{f\in\bar{\theta}_n}\varsigma_f(s_n)\right)\right]^{-1} \quad (1)$$

s.t.

$$\sum_{i\neq j\in\mathcal{P}_{m,n}} l_{i,j}\cdot\delta_{l_{i,j}} \le d(\theta_{m,n}, s_n), \quad \forall m\in\mathcal{B}, \forall n\in\mathcal{R} \quad (2)$$

$$\sum_{m=1}^{M}\sum_{n=1}^{N} I(i,j,\mathcal{P}_{m,n})\cdot b(\theta_{m,n}, s_n) \le \beta_{l_{i,j}}, \forall i\neq j\in\mathcal{N} \quad (3)$$

where $\varsigma_f(s_n)$ models the computational burden of function f given configuration s_n (MIMO setting, bandwidth, etc.) of BS n, and $0 \le \alpha \le 1$ models the relative cost savings when a function is centralized, i.e., if $\alpha = 1$, centralizing a function is as costly as maintaining the function co-located with the RU and therefore our problem becomes a simple feasibility problems where D-RAN and C-RAN configurations provide the same cost to the system. Conversely, if $\alpha < 1$ there is a cost saving when a function is centralized (e.g. coming from pooling gains) and our optimization problem will lean towards C-RAN as much as possible. The first set of constraints (2) handle delay requirements, where $\delta_{l_{i,j}}$ is the latency of link $l_{i,j}$, which we assume is an additive constant for simplicity (i.e. we do not consider queueing effects)[3] and $d(\theta_{m,n}, s_n)$ is the delay requirement of split $\theta_{m,n}$ and setting s_n. We skip jitter constraints because they can be mitigated with buffers at the receivers at the cost of latency [10]. Eq. (3) handles capacity violations, where $\beta_{l_{i,j}}$ is the total bit-rate capacity of link $l_{i,j}$, $I(i,j,\mathcal{P}_{m,n})$ is an indicator function which is 1 if nodes i and j are contained in $\mathcal{P}_{m,n}$ and 0 otherwise, and $b(\theta_{m,n}, s_n)$ gives the throughput demanded by an RU with split $\theta_{m,n}$ and setting s_n (including all protocol overheads). For instance, Table 1 shows values of $d(\cdot)$ and $b(\cdot)$ for different splits and a certain setting s_n (with 1 user and 150 Mb/s downlink load). Without loss of generality, we consider the functions, splits and costs given in Table 2 (the shaded and white area highlight the functions running in a CU and an RU, respectively), and assume $s_n \, \forall n$ given in Table 1 to simplify our evaluation. In practice, different traffic profiles could be used to minimize resource wastage. The above can be easily extended to, e.g., consider computational constraints, but here we only focus on networking constraints for simplicity.

5 WizHaul Algorithms

We let the triple $X_n := \{A_n, \theta_{A_n,n}, \mathcal{P}_{A_n,n}\}$ describe a configuration of BS n, and $\boldsymbol{X} := \{X_n \mid \forall n \in \mathcal{R}\}$ be a candidate solution to our problem. Our goal is to find the optimal $\boldsymbol{X}^*$ that maximizes the degree of centralization γ. Since this is an integer non-linear optimization problem ($b(\cdot)$, $d(\cdot)$ or $\varsigma_f(\cdot)$ can be discontinuous non-linear functions), we focus on combinatorial algorithms. In order to reduce complexity, we constraint our combinatorial search to settings where all BSs within the same cluster have the same split and CU assignment. We propose two algorithms: a nearly-optimal branch-and-bound backtracking algorithm (BBB), and a low-complex greedy approach (GA).

3. Frame preemption (802.1Qbu) and scheduled traffic (802.1Qbv) can be used mitigate queueing effects [10].

5.1 Backtracking Branch-and-Bound (BBB)

We start with an optimal branch-and-bound approach that explores a discrete space of candidate solutions. This type of algorithms has the advantage of being highly parallelizable [44], i.e. suitable for cloud computing platforms. First, we store in $\Pi_n := \{\mathcal{P}_{m,n}^{(1)} \cdots \mathcal{P}_{m,n}^{(k)} \mid \forall m \in \mathcal{B}\}$ k candidate paths between each CU m and RU n, for all $n \in \mathcal{R}$. This can be readily obtained with k-shortest path routing versions of Dijkstra, for example, with complexity $\mathcal{O}(MNk(L + |\mathcal{N}| \log |\mathcal{N}|))$ with $L := \sum_{i,j \in \mathcal{N}} l_{i,j}$ [45].

0. State notation. The space of candidate solutions can be represented as a tree where a node i in level τ represents one possible setting $X_{\psi(\tau)}^{(i)}$ for RU $\psi(\tau)$, where $\psi(\tau)$ is a function that maps a level τ to RU n ($\psi(0) = \varnothing$ is the root level) and i is a point in the configuration space $\mathcal{C}_n := \{(m, \theta_{m,n}, \mathcal{P}_{m,n}) \mid m \in \mathcal{B}, \theta_{m,n} \in 2^{\mathcal{F}}, \mathcal{P}_{m,n} \in \Pi_n\}$. A full branch represents thus a candidate $\boldsymbol{X}$ (see Fig. 3).

1. Initialization phase. We make use of a heuristic *score function* that pre-evaluates how "good" configuration X_n is. To this aim, inspired by [13], we define F_{X_n} as

$$F_{X_n} := \frac{W(\theta_{m,n})}{\displaystyle\sum_{p \in \Phi(\mathcal{P}_{m,n})} \frac{b(\theta_{m,n}, s_n)}{\displaystyle\min_{\{l_{i,j} \mid \forall i,j \in p\}} \beta_{l_{i,j}}}}, \tag{4}$$

where $W(\theta_{m,n}) := \left(\alpha \sum_{f \in \theta_i} \varsigma_f(r_i) + \sum_{f \in \bar{\theta}_i} \varsigma_f(r_i)\right)^{1}$ is a *reward* function: the individual degree of centralization of BS n when using split $\theta_{m,n}$ and CU m, with α and $\varsigma_f(\cdot)$ being the same parameters used in Eq. (1). In the denominator, function $\min_{\{l_{i,j} \mid \forall i,j \in p\}}$ represents the maximum capacity of a path p (i.e. the bottleneck link on that path) and $b(\theta_{m,n}, s_n)$ is the throughput requirement of BS n when using configuration X_n (i.e. functional split $\theta_{m,n}$). Additionally, $\Phi(\mathcal{P}_{m,n})$ is a set that collects all paths $p \in \bigcup_{j \in \mathcal{R} \setminus n} \Pi_j$ (all potential paths for all CU/RU pairs) that *share some link with the path considered in configuration X_n, $\mathcal{P}_{m,n}$*. In this way, the denominator of the above equation sums up across the load required by flow n relative to the maximum capacity of path p *for all paths that share some link with the path under consideration by configuration X_n*. This serves us as a rough estimation of the penalty incurred by any split choice and path combination. *Therefore, F_{X_n} represents the reward of using path $\mathcal{P}_{m,n}$ and split $\theta_{m,n}$ (i.e. configuration X_n) relative to an estimation of the burden such choice would add to the whole system.* If, e.g., a path p shares many links with other paths that could potentially be used by other flows, the denominator would contribute to lower the score of X_n, which is maximum for disjoint paths (with no links in common) and high-centralization functional splits. In this initialization phase, we pre-compute $F_{X_n^{(i)}}$ $\forall i \in \mathcal{C}_n, \forall n \in \mathcal{R}$. We will use this information in our branching phase.

We also keep track of an upper bound on the minimum cost γ^{1} achievable in the system, which is tightened as it advances. As we will see in §8, a good initial bound helps enormously to lower the searching time. In our case, we will use the greedy approach we introduce later to this aim.

2. Branching. Our branching method is based on a Depth-First-Search (DFS) tree-exploring scheme that starts at the root $\tau = 0$ and visits one unvisited node per level until it reaches the depth of a branch, when it backtracks

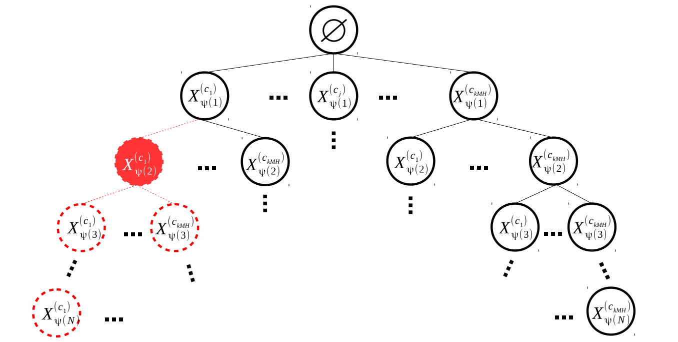


Fig. 3: BBB algorithm. Partial candidate $(\{X_{\psi(1)}^{(c_1)}, X_{\psi(2)}^{(c_1)}, \cdots\})$ violates constraints and all hanging branches are thus pruned.

one level, until all nodes in the tree are visited. This can be implemented recursively or iteratively.

To minimize running time, it is of paramount importance that reasonably good solutions are explored early, and also that configurations that violate constraints are detected early. This would allow us to maximize the amount of pruning that we do over the tree, i.e., sections that do not have to be explored via *early backtracking* and *bounding*.

In order to do that, we carefully choose our $\psi(\tau)$ function to map RUs in ascending order according to $\sum_i F_{X_n^{(i)}}$, where $F_{X_n^{(i)}}$ is the score function defined earlier. Then, each node i in level τ corresponds to a configuration from the ordered sequence $\langle X_{\psi(\tau)}^{(i)} \rangle$, where candidate settings are sorted in descending order according to $F_{X_{\psi(\tau)}^{(i)}}$ $\forall i \in \mathcal{C}_{\psi(\tau)}$ first, and in ascending order according to $|\theta_{m,\psi(\tau)}^{(i)}|$, second. This tests high-order centralization levels with a high score first (potentially good solutions) and "worse" RUs higher up in the tree (configurations with potential constraint violations).

Additionally, it is important to note that inferior levels may have a lower space of candidate configurations if an ancestor node belongs to the same cluster. This is because, in such case, a split choice and a CU assignment have already been made for this RU (that of the ancestor); thus the space of possible settings for this RU drops to k, that is, simply the set of candidate paths.

3. Early backtracking. Every time we visit a node of the tree we have a larger partial candidate solution, until we reach the depth of the tree, when we have a complete candidate solution. To speed up the process, when a node is visited we check that such partial candidate does not violate constraints (2)-(3). If it does, the hanging nodes are pruned, we backtrack to the previous level $\tau \quad 1$ and visit a new unvisited node (see Fig. 3).

4. Lower bounding. As we explained above, given a branch, every level of the tree τ corresponds to a candidate configuration for one BS or flow. When visiting a node of the current branch, we compute a lower bound by assuming the inferior levels of the branch (unexplored) has the lowest possible contribution to γ^{1} (our cost function). This is straightforward to compute as we can just assume these unexplored nodes use the highest possible split, which obviously represents a lower bound (unexplored nodes cannot contribute less to the overall cost than what we assume to compute the lower bound). If the computed lower bound is higher than the current upper bound, we prune the hanging branches, backtrack to the previous level $\tau \quad 1$ and visit a new unvisited node.

Algorithm 1 Greedy algorithm.

```
1: function GREEDY_OPT($\mathcal{B}, \mathcal{R}, \mathcal{V}, \mathcal{Q}$)
2:     /*Initialization*/
3:     for $\forall q \in \mathcal{Q}$ do
4:         $\theta'_q = \varnothing$
5:     end for
6:     ($\boldsymbol{\mathcal{P}'}, \gamma'$) = find_routes($\mathcal{B}, \mathcal{R}, \mathcal{V}, \mathcal{Q}, \boldsymbol{\theta'}$)
7:     while $|\boldsymbol{\mathcal{P}'}| = |\mathcal{R}|$ do
8:         /*All flows could be routed*/
9:         ($\boldsymbol{\theta^*}, \boldsymbol{\mathcal{P}^*}, \gamma^*$) ← ($\boldsymbol{\theta'}, \boldsymbol{\mathcal{P}'}, \gamma'$)
10:        for $\forall q \in \mathcal{Q}, |\theta_q| < |\mathcal{F}|$ do
11:            $\psi_q = \sum_{n \in q} \max_{\forall i \in \mathcal{C}_n} \left\{ F_{X_n^{(i)}} \mid \theta_q = \{\theta_q^*, f_{|\theta_q^*|+1}\} \right\}$
12:        end for
13:        $q \leftarrow \arg\max(\boldsymbol{\psi})$
14:        $\theta'_q \leftarrow \{\theta_q^*, f_{|\theta_q^*|+1}\}$
15:        ($\boldsymbol{\mathcal{P}'}, \gamma'$) = find_routes($\mathcal{B}, \mathcal{R}, \mathcal{V}, \mathcal{Q}, \boldsymbol{\theta'}$)
16:    end while
17:    return ($\boldsymbol{\theta^*}, \boldsymbol{\mathcal{P}^*}, \gamma^*$)
18: end function
```

5. Upper bounding. We use the output of the heuristic approach in §5.2 as an initial upper bound. Then, every time depth of a branch is reached, this branch becomes the new best solution, $\gamma\ ^1$ the new upper bound, we backtrack to the previous level and visit a new unvisited node.

5.2 Greedy algorithms (GA-GR and GA-RR)

Unfortunately, the complexity of branch-and-bound approaches does not scale well in general, particularly if an initial upper bound is not properly chosen. For this reason, we now propose a low-complex combinatorial algorithm described in Algorithm 1. The algorithm assumes that $F_{X_n^{(i)}}$ has been computed for all RU $n \in \mathcal{R}$ and all $i \in \mathcal{C}_n$. (Although $\mathcal{C}_n$ is built using a set of pre-computed routes, this algorithm is not constrained to choose them). Note that we abuse notation and we let θ_q denote the functional split $\theta_{A_n,n}$ for any RU n in cluster $q \in \mathcal{Q}$ (since all of them shall have the same setting).

Algorithm 1 greedily increases the functional split setting of the cluster $q \in \mathcal{Q}$ with largest ψ_q such that

$$\psi_q := \sum_{n \in q} \max_{\forall i \in \mathcal{C}_n} \left\{ F_{X_n^{(i)}} \mid \theta_q = \{\theta_q^*, f_{|\theta_q^*|+1}\} \right\} \tag{5}$$

ψ_q is another score function that we use as a heuristic approach to favor higher degrees of centralization of RUs in larger clusters with a roughly higher likelihood to satisfy constraints. The intuition is that larger clusters contribute to increasing ψ_q (because we are summing up across all RUs in the cluster) and, given that a higher F_{X_n} has (roughly) higher chances of meeting constraints (because RU n overlaps with less and higher-capacity links used by others), clusters with RUs that have good "best potential configurations" will also contribute to increasing ψ_q. Thus, clusters with higher ψ_q are in better position to cause less *damage* if their split is increased but also have higher improvement over the degree of centralization (because there are more RUs in the cluster).

This score is computed in step (11) of Algorithm 1. The algorithm tests BS configurations greedily in an orderly fashion based on the above score. Step (13) selects the cluster with highest aggregated score and increases the degree of centralization of all the BSs within the cluster (step (14)). Then, for every split change, we invoke `find_routes(·)` (step (15)) to find a CU assignment first and a feasible routing instance second. Algorithm 1 has a worst-case complexity of $\mathcal{O}(|\mathcal{Q}||\mathcal{F}| + 1)$ times that of the CU assignment and routing algorithms which we present next. Our CU assignment algorithm is a simple heuristic that pairs each cluster to the CU which is closest (in terms of lowest latency) to the centroid of the cluster. This can be readily done with complexity $\mathcal{O}\left(M(L + |\mathcal{N}| \log |\mathcal{N}|)\right)$ applying Dijkstra. We now propose two alternative routing schemes.

5.2.1 *Greedy Routing (GA-GR)*

This is a quick greedy algorithm that leverages the $F_{X_n^{(i)}}$ scores for the set of pre-computed routes mentioned above. We note that, although we use a set of pre-defined paths, the output of this approach may not return those routes—we are just interested on the scores. The algorithm is a simple greedy approach, inspired by the combinatorial UFP solver of [13]. We first sort all flows $n \in \mathcal{R}$ in descending order according to $\sum_{i \in \mathcal{C}_n} \{F_{X_n^{(i)}} \mid \theta_q = \theta_q^*\}$. The intuition is that high-score flows are (roughly) less prone to penalize the degree of centralization of other flows (note that F scores favor disjoint paths). Then, for each flow of the sorted sequence, we iteratively apply shortest-path routing (or any QoS routing approach) to greedily find a feasible route for each path in the sorted set. The topology is updated in each iteration with the new residual link capacities. The algorithm returns a set of feasible routes for all flows or an empty set if we could not find a feasible route for some flow. Using Dijkstra, this has complexity $\mathcal{O}\left(N(L + |\mathcal{N}| \log |\mathcal{N}|)\right)$.

5.2.2 *Randomized Rounding routing (GA-RR)*

Alternatively to that greedy approach, we also propose a randomized rounding routing algorithm, inspired by [46], which works as follows. We first solve the following linear relaxation that give us fractional flows (multi-path routing) from CUs to RUs:

$$\min_{\boldsymbol{Y}} \quad U := \sum_{\forall n \in \mathcal{R}} \sum_{\forall i \neq j \in \mathcal{N}} \delta_{l_{i,j}} \cdot y_{n,l_{i,j}} \tag{6}$$

$$\text{s.t.} \sum_{\forall e \in \epsilon^+(v)} y_{n,e} \qquad \sum_{\forall e \in \epsilon\ \ (v)} y_{n,e} = 0, \quad \forall n \in \mathcal{R}, \forall v \in \mathcal{V} \tag{7}$$

$$\sum_{\forall e \in \epsilon^+(A_n)} y_{n,e} \qquad \sum_{\forall e \in \epsilon\ \ (A_n)} y_{n,e} = 1, \qquad \forall n \in \mathcal{R} \tag{8}$$

$$\sum_{\forall e \in \epsilon\ \ (R_n)} y_{n,e} \qquad \sum_{\forall e \in \epsilon^+(R_n)} y_{n,e} = 1, \qquad \forall n \in \mathcal{R} \tag{9}$$

$$\sum_{\substack{\forall e_v \in \epsilon^+(v) \\ \forall n \in \mathcal{R}}} y_{n,e_i} b(\theta_{A_n,s_n}) + \sum_{\substack{\forall e_o \in \epsilon\ \ (v) \\ \forall n \in \mathcal{R}}} y_{n,e_o} b(\theta_{A_n,s_n}) \le \beta_{e_o}, \forall v \in \mathcal{N} \tag{10}$$

where $\boldsymbol{Y} := \{y_{n,l_{i,j}} \mid \forall n \in \mathcal{R}, \forall i \neq j \in \mathcal{N}\}$. $y_{n,l_{i,j}}$ is a real value that represents the fraction of the flow of RU n that is routed through link $l_{i,j}$. $\epsilon^+(i)$ ($\epsilon\ \ (i)$) are sets containing all outgoing (incoming) links from (to) node i. The objective function aims to find paths with low latency. Eq. (7) is the flow conservation constraint, eq. (8)-(9) guarantee that flows go from CUs to RUs, and eq. (10) is the capacity constraint. Since this formulation is quite standard, we refer the reader to [46] for further details. Secondly, we perform path decomposition on the fractional solution. To this means, we use DFS to map the N fractional flows found during the previous step to N sets of paths where each path is given a weight which corresponds to the minimum fraction of flow assigned to the path. Finally, we apply randomized rounding to select one path of each set and then we assign the whole flow to that path. To do so, we use the weight proposed before for each candidate path as the probability of selecting this path. We proceed until all flows have one path, which is one round. The idea is to try out several rounds and select the best one. After each round, we check if the resulting set of paths violates constraints; if so, we discard this round. Otherwise, we compute the objective function of the linear relaxation (using the integer solution). We repeat the whole process until the standard error over the average objective function is small enough (e.g. 10%) or we arrive to a pre-defined number of iterations. The algorithm returns the routes with lowest cost U or null if some flow could not be routed. The worst-case performance of this approach is $\mathcal{O}(\log(|\mathcal{N}|)/\log(\log(|\mathcal{N}|)))$ relative to the optimal multipath solution [46].

In the next section, we revise the key requirements and limitations of such flexible fronthaul and WizHaul.

6 Practical Considerations

An NGFI-compliant packet-based network capable of transporting flows from a variety of functional splits needs to be deployed [7]. Fortunately, such function placement flexibility saves operators from having to deploy dedicated fiber between RUs and CUs like a traditional C-RAN does [12]. WizHaul adapts the choice of function splits and routing to the forwarding limitations in terms of capacity or latency in the data plane, rather than imposing specific requirements. Even though this new system and thus WizHaul relaxes constraints on the data plane, it does require some degree of *performance determinism* that can be obtained in cost-effective Ethernet networks via e.g. Clock synchronization (IEEE 802.1AS), Stream Reservation Protocol (IEEE 802.1Qat), Frame Pre-emption (IEEE 802.1Qbu) and scheduled traffic (IEEE 802.1Qbv)—see IEEE's Time-Sensitive Networking Task Group and [10].[4] In the scope of optical infrastructure, a packet-based fronthaul over TDM PON is discussed in [47] and demonstrated in [48].

Note that these requirements are mild: even though these extensions mitigate uncertainty in the underlaying data plane, their absence (i.e. uncertainty in the data plane) simply makes WizHaul lean towards less centralized (but safe) configurations. This is a substantial cost-efficiency improvement over the requirement of dedicated fiber in traditional C-RAN, which explains the interest by the industry [4], [6].

In order to support fault-tolerance, i.e., system reconfiguration in a more dynamic fashion, in addition to the above requirements, WizHaul requires explicit forwarding path control (IEEE 802.1Qca or OpenFlow) and a system architecture that follows the Software Defined Networking (SDN) principles, namely, (i) decoupled data and control planes, (ii) logically centralized control, and (iii) exposure of abstract resources and state to the network controller. In this case, the control plane is not only in control of the forwarding behavior of the data plane but also manages the placement of Base Station functions (into RUs/CUs' compute processor units) as shown in [9].

4. IETF DetNets is another group working on deterministic WANs.

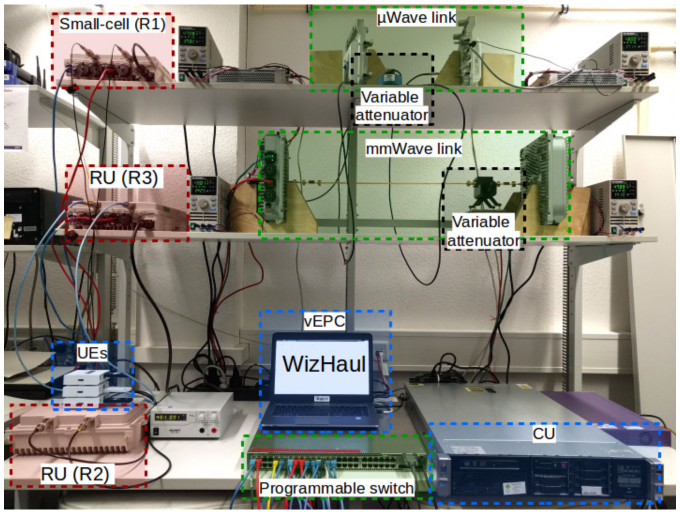


Fig. 4: Testbed.

Although in this paper we focus on the algorithmic solution to a new fundamental problem arising in a flexible fronthaul, rather than the design of the system itself (e.g. design choices such as whether BS functions are implemented on virtual machines or linux containers, or whether migrating virtual functions or simply switching from one configuration to another upon a functional split change) we do refer the reader to relevant literature on how SDN can be exploited in this manner. For instance, in [9], the authors presented the architecture of a system that integrates BH/FH operating multiple functional splits with a common forwarding plane; in addition, in the scope of optical network technology, a comprehensive survey of SDN is shown in [49], and an SDN-controlled digital signal processor enabled spectral converter for converging optical and FH/BH systems is presented in [50].

In the next two sections we evaluate the ability of the above algorithms, wrapped up into a common decision engine named WizHaul, to (i) operate in fault-tolerance use cases when integrated into an SDN platform (§7), and (ii) maximize the degree of centralization in large-scale synthetic and real topologies, suitable when integrated in a planning tool for NGFI systems (§8).

7 Experimental Proof of Concept

In this section, we implement our algorithms in a centralized decision-making entity based on SDN and present a fault-tolerance use case. Our goal is to validate the ability of WizHaul to maintain maximum centralization upon topology changes.

7.1 Experimental Setup

We deploy the testbed shown in Fig. 4, illustratively described as a baseline scenario in Fig. 5. We use two commercial wireless products in the BH/FH segment, namely a μWave link (connecting nodes 3-5 in Fig. 5) and a mmWave

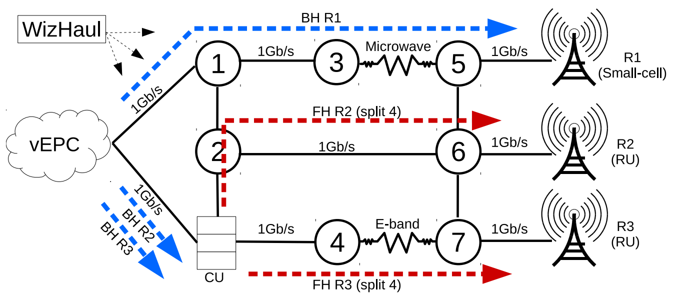


Fig. 5: Baseline PoC scenario.

link (connecting nodes 4-7). Both support adaptive modulation schemes so the actual capacity varies with the average signal-to-noise ratio (a common measure of wireless channel conditions or quality). For demonstration purposes, the radio links are wired with an SMA cable (μWave RF over coaxial) and a rigid waveguide (E-band RF). Moreover, variable attenuators let us emulate different average channel conditions. The remaining topology of the transport segment is virtualized using a programmable switch via gigabit Ethernet interfaces. In addition, a softwarized virtual EPC is deployed on a commodity laptop. The RAN equipment is comprised of R1, a fully-fledged LTE small-cell (i.e., its base station functions are not centralized), and two RUs (R2 and R3) connected to a CU. Given that we do not have yet the ability to configure all functional splits from Table 2 in our CU/RU hardware, we derive traffic flow patterns based on a commercial product [16], namely a flexible C-RAN solution with available splits 1 and 3 from Table 2, and generate UDP flows accordingly. Table 3 provides more details of the HW components of our testbed. All elements are connected across the same L2 Ethernet-based domain and are synchronized with PTP, which allows us to measure one-way delay.

7.2 Software Deployment

Fig. 6 depicts an illustration of our software architecture. We implement GA-RR as an application on top of `Floodlight`, a Java-based Apache-licensed SDN controller[5]. In the initialization phase, the application retrieves (through a REST-based interface) a graph abstraction of the physical topology using `Floodlight`'s Topology Manager which in turn uses LLDP discovery protocol. Then, our application uses GA-RR to jointly compute the optimal paths between radio access points, vEPC and CU (when needed) and the functional split of capable BSs. Upon topology

5. http://www.projectfloodlight.org/floodlight/

TABLE 3: Detailed HW components in our testbed.

Device type	Description	Ref.
vEPC	OpenEPC Rel. 6	[51]
μWave	56 MHz bandwidth @ 7GHz band Adaptive rate $\leq$ 1 Gb/s	[52]
mmWave	500 MHz bandwidth @ E-band Adaptive rate $\leq$ 3.2 Gb/s	[53]
Switch	OpenFlow switch 48 one-gigabit, 4 ten-gigabit ports	[54]
Small-cell	20 MHz channel @ band 3	[55]
RU	20 MHz BW @ band 3 Split 1 (PHY, MAC, RLC) and 3 (PHY)	[16]
CU	Virtual MAC, RLC, PDCP, RRM, RRC	[56]

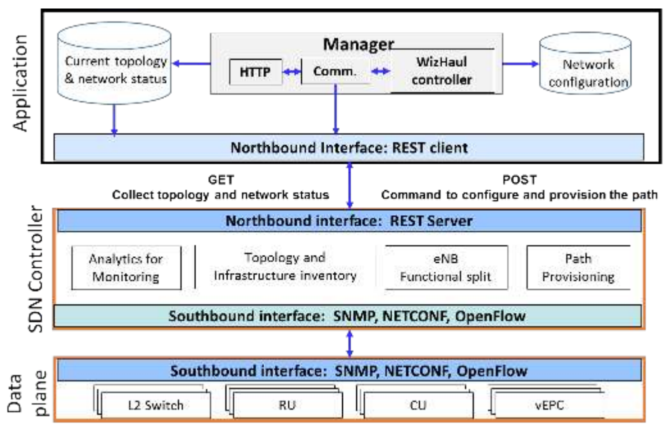

Application
Current topology & network status
Manager
HTTP
Comm.
WizHaul controller
Network configuration
Northbound Interface: REST client
GET
Collect topology and network status
POST
Command to configure and provision the path
SDN Controller
Northbound interface: REST Server
Analytics for Monitoring
Topology and Infrastructure inventory
eNB Functional split
Path Provisioning
Southbound interface: SNMP, NETCONF, OpenFlow
Data plane
Southbound interface: SNMP, NETCONF, OpenFlow
L2 Switch
RU
CU
vEPC

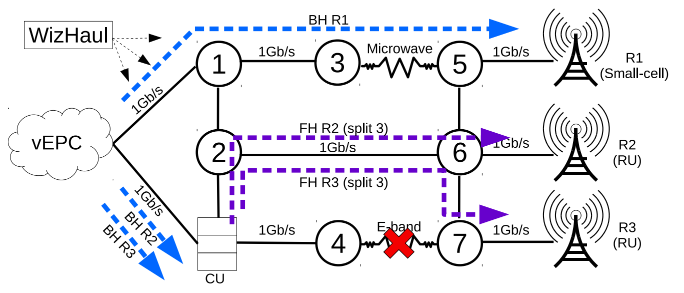
Fig. 6: WizHaul as an application on top of an SDN platform.

changes, e.g. a link failure or a change on the modulation of a wireless link, our application receives a notification from the SDN controller (which in turn receives a notification from the hardware components via SNMP) and a new configuration is computed (and enforced by the SDN controller via SNMP and OpenFlow for path setup). As depicted in Fig. 6, the different key architectural components of our software application are the following. The `HTTP management` class is used to create different HTTP request objects that will be used to communicate with the controller's REST API. The bodies of the HTTP methods are filled out with JSON data objects. The `Communications management` class enqueues all the HTTP objects in a FIFO queue and are processed by several threads associated to different callback functions in charge of managing replies of requested objects. The `Crosshaul controller` class provides the different data structures to maintain the network state and implement the logic of Algorithm 1. The `Manager` class manages the callbacks upon notifications from the SDN controller, and coordinates the above classes for the recovery process (i.e. computation of a new configuration) while it maintains updated data structures and network state information. We rely on existing SDN services in `Floodlight` (shown in Fig. 6) for path provisioning, topology and monitoring information, as well as an extended service to let us (re)configure the CU/RU split in our equipment.

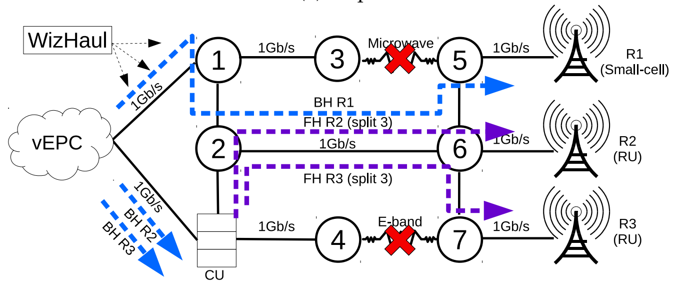
7.3 Experiment

We create 3 UDP downlink flows with the fixed user load assumed in Table 1 with `mgen`[6] from the vEPC towards each of the radio access points, as shown in Fig. 5. Note that the flows of R2 and R3 must go through the CU which connects with the RUs via fronthaul paths with different requirements depending on the functional split. Fig. 8 depicts the mean throughput and packet delay performance of all flows at the access point side (i.e., fronthaul flow for R2 and R3, and backhaul flow for R1). We start with the baseline scenario shown in Fig. 5. Due to the limitation of our testbed to 1-Gb/s interfaces, the algorithm settles with split 4 (from Table 2) for R2 and R3, i.e. no full C-RAN which would require transporting ¿2-Gb/s flows. After 20 sec, we attenuate the signal of the E-band link, as illustrated in Fig. 7a. This causes two reactions: (i) the fronthaul flow of R3

6. https://www.nrl.navy.mil/itd/ncs/products/mgen



(a) Step 1



(b) Step 2

Fig. 7: Use case illustration

is re-routed through switches 2 and 6, and (ii) subsequently the functional split of both R2 and R3 must be softened to split 3 in order to keep a high degree of centralization while guaranteeing that network constraints are satisfied. Finally, 20 sec later, we attenuate the power of the μWave link. This causes the backhaul flow of R1 to be re-routed through link 2-6, which is shared with 2 fronthaul flows. As we can see from Fig. 8, the requirements of both fronthaul flows are still satisfied in this step and thus no functional split change is required. In summary, the results shown in Fig. 8 validate that our algorithm (i) maximizes the degree of centralization, and (ii) ensures the flow requirements of each split are met at all times.

In Fig. 9 we provide insights regarding the nature of the time taken to re-deploy a new configuration when topology changes occur in the use case presented before. The figure shows the amount of time taken on each of the basic operations of our software, namely (i) receiving a notification of the affected hardware elements (labelled "HW reaction"), (ii) running our algorithm (labelled "Algorithm), (iii) install new OpenFlow rules ("Path installation"), and other functions like building/processing messages through the REST interface, etc. (labelled as "Others"). From the figure we see that the overall reaction time is dominated by the delay of the HW components in noticing physical changes. Remarkably, re-computing a new solution barely has a toll given by the simplicity of the topology. We also note that the reaction takes longer in Step 1 (Fig. 7a) due to path installation process because the flow of R3 has to be managed by a different physical network card at the CU (as opposed to virtual interfaces in our virtual topology) which results in longer processing.

8 Large Scale Simulations

We now assess via simulations the performance of our algorithms with both synthetic large-scale networks and two topologies from two major operators in Europe to assess the performance of WizHaul as a planning tool.

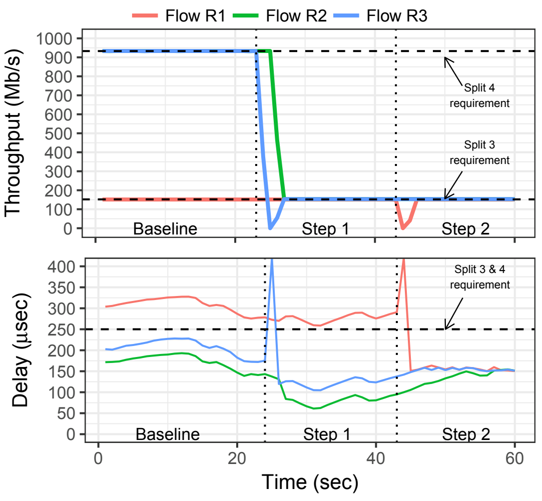


Fig. 8: Experimental validation.

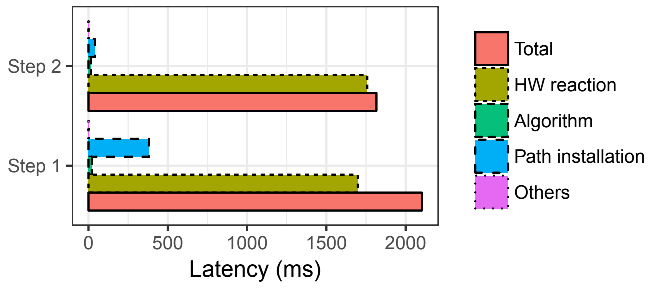
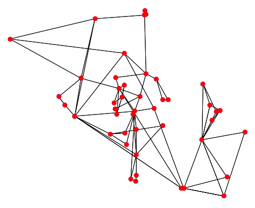


Fig. 9: Software reaction time

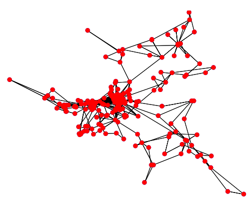
In order to get statistically meaningful insights, we run each of the algorithms described above over a large set of simulated topologies and extract three parameters for each simulation: degree of centralization γ, whether it is a feasible solution (D-RAN otherwise) and its elapsed time. Each simulation runs on a single Intel i7 core operating at 2.4 GHz. Based on [3], we set $\alpha = 0.5$ to compute γ in eq. (1), i.e. processing functions in an RU is twice as expensive as doing it in a CU. The first and second set of topologies are based on two backhaul networks of existing operators in Romania and Switzerland (up to 900 topologies). The third and forth sets correspond to random topologies based on tree structures (up to 1800 topologies) and Waxman random graphs (up to 1500 topologies).

8.1 Real Topologies

We create semi-random scenarios based on the backhaul topology of two operators in Romania and Switzerland, illustrated in Fig. 10. We know the distance between switching nodes, links connecting them and their capacities. Based



(a) Romania topology.



(b) Switzerland topology.

Fig. 10: Snapshot of real topologies.

TABLE 4: Ethernet-based link profiles (proc. delay = 5 μs, packet size = 1518 Bytes)

Technology	**Bandwidth (Gb/s)**	**Prop. delay (μs)**	**distance (km)**
mmWave (60-80 GHz)	0.9, 1.25, 1.5, 2, 3, 4, 8	1-20	0.3-6
μWave (6-60GHz)	0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1, 1.25, 1.5, 2	1-100	0.3-30
Copper (1000/10G/40GBASE-T)	1, 10, 40	0.05-0.5, 0.275, 0.15	0.001-0.1, 0.055, 0.03
SMF fiber @ 1310 nm (1000, 10G, 40, 100GBASE-EX, LR, LR-4)	1, 10, 40, 100	1-200, 50, 50, 50	0.2-40, 10, 10, 10
SMF fiber @ 1550 nm (1000, 10G, 40, 100GBASE-ZX, ER, ER-4)	1, 10, 40, 100	1-350, 200, 200, 200	0.2-70, 40, 40, 40
TbE (*under development)	200, 400	1-50	0.2-10

on this, we choose the profiles from Table 4 that match better the known capacities. Note that we only consider Ethernet over different PHYs. We assume classic store-and-forward switching which induces a per-hop latency equal to propagation delay—distance(m)/200 μs for copper, distance(m)/300 μs the rest—plus serialization delay (bits/rate). Finally, we randomly pick nodes to be RUs and CUs according to Table 5 and use simple k-means to create as many RU clusters as CUs. We note that, though we use simple k-means for simplicity (and no generality loss), clustering is important in Cloud RAN and has been object of much study over the years [20], [40], [41].

8.2 Tree-based Topologies

To generate topologies with a tree structure, first we generate a set of CUs connected in a ring. These will be located at the root of the trees. Second, hanging from each CU, we create a pure random tree using independent Poisson processes with parameters λ_{levels} and $\lambda_{\text{siblings}}$ to model the number of levels of the tree and nodes per level. The leafs of the tree correspond to RUs. For each level of each tree, we randomly add a backup link with the upper layer of a neighboring tree, to add additional degrees of freedom for routing, following another Poisson process with parameter λ_{backup}. Table 5 shows the parameters we used in our evaluation. Once we have the graph structure, we randomize the profile of all the links and their length, using the same assumptions as in the previous scenarios (links in Table 4). For each link and topology, we randomly pick one profile depending on its proximity to a CU or an RU. To this aim, we group links based on its proximity to a CU and assign them the same link profile. Additionally, we assign high-capacity profiles to links closer to CUs with more probability, to capture the fact that aggregation segments typically provide higher capacities. Finally, RU clustering is done as in the previous scenarios.

8.3 Waxman-based Topologies

The last set of topologies consists of Waxman random graphs [57], based on the Erdös-Renyi random graph model, which is popular to evaluate realistic backhaul topologies [58]. To this aim, we used the parameters displayed in Table 5, λ being the intensity of the Poisson process, α the maximal link probability and β a parameter to control the length of the edges. Link profiles, capacities, latencies and clusters are assigned in the same way we did before.

8.4 State-of-the-art Techniques

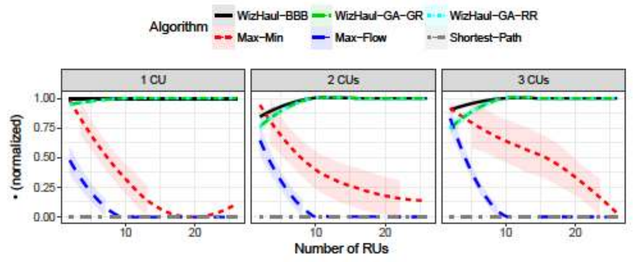
The related literature presented in §2 is not well-suited to solve our problem. Instead, we compare our algorithms against a few "best routing, then best split" strategies (see §1) to demonstrate that simple heuristics based on state-of-the-art tools are also unfitted for large-scale setups. We first use the same simple heuristic we adopted in our GA algorithm to assign RU clusters to CUs. Then, we use three different baseline routing techniques to compute paths and choose BS splits. The first algorithm (SHORTEST-PATH) greedily chooses, for each RU, the shortest path to its CU. Then, based on the paths set, we exhaustively search for the best RAN split. The second algorithm (MAX-FLOW) solves an LP relaxation of the routing problem that maximizes the aggregate flow load; then we use randomized rounding (explained above) to find single paths and select feasible splits. The third approach (MAX-MIN) uses the algorithm proposed in [14] to obtain a max-min fair single-path routing solution; then we pick the best functional splits given the resource allocation of such routing instance. We note however that our BBB algorithm renders an optimal configuration and therefore we can use it as a benchmark for our heuristic and this state-of-the-art techniques.

8.5 Results

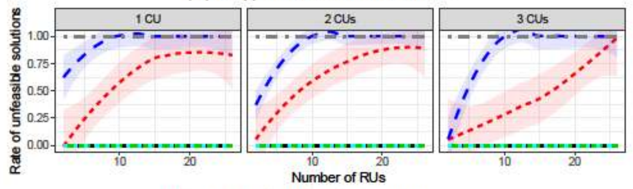
Fig. 11-14 (top) depict the normalized degree of centralization γ (y axis), i.e., a real value between 0 (all functions co-located with the RUs, that is, D-RAN) and 1 (all functions centralized in a CU, that is, C-RAN) for all algorithms and topologies presented earlier, as a function of the number of RUs in the topology (x axis). At the bottom, the figures represent the ratio of unfeasible solutions. We compare our BBB, our GA with greedy routing approach (GA-GR), and our GA with randomized rounding approach (GA-RR) against the benchmarks presented earlier. Importantly, we use GA-RR as initial upper bound for BBB. Their elapsed time is shown in Fig. 15. If one simulation reaches $10^5 s$, we stop it and declare the result as unfeasible (which only happens occasionally for MAX-MIN). Given the large amount of scenarios and heterogeneity of the algorithms, we show curves from a non-parametric local regression fit model (*loess* [59]) in all figures. Shaded areas show the standard error of the fitted model.

TABLE 5: Parametrization of topology generators.

Type of topology	**Parameters**
Romania BH, Switzerland BH	• $\lvert\mathcal{N}_{\text{Romania}}\rvert = 46, \lvert\mathcal{N}_{\text{Swiss}}\rvert = 272$ nodes; • $\lvert\mathcal{R}\rvert=\{\lvert\mathcal{N}\rvert \cdot 0.025i \mid i \in \mathbb{N}, 0 < i \leq 25\}$ RUs; • $\lvert\mathcal{B}\rvert = \lvert\mathcal{Q}\rvert = \{1, 2, 3\}$ CUs/clusters.
Tree-based	• $\lambda_{\text{levels}} = \lambda_{\text{siblings}} = \lambda_{\text{backup}} = 1$; • $\lvert\mathcal{B}\rvert = \lvert\mathcal{Q}\rvert = \{2, 3, 4\}$ CUs/clusters.
Waxman-based	• $\lambda = \{20i \mid i \in \mathbb{N}, 0 < i \leq 10\}, \alpha = 0.4, \beta = 0.1$; • $\lvert\mathcal{R}\rvert = \{1, 2, \cdots 100\}$ RUs; • $\lvert\mathcal{B}\rvert = \lvert\mathcal{Q}\rvert = \{1, 2, 3\}$ CUs/clusters.

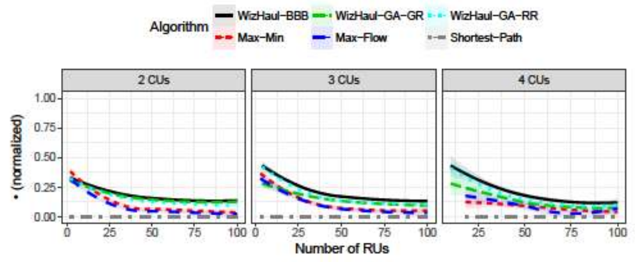


(a) Degree of centralization.

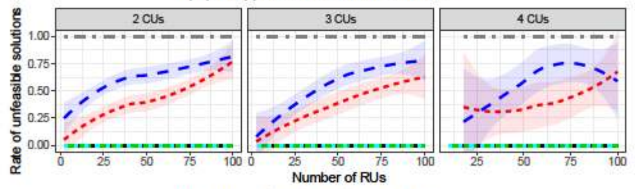


(b) Rate of unfeasible solutions.

Fig. 11: Romanian topology.

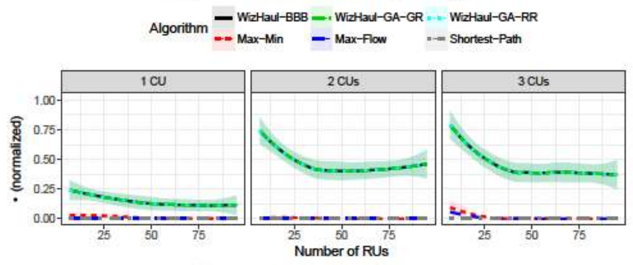


(a) Degree of centralization.

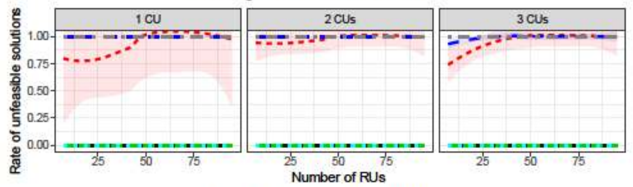


(b) Rate of unfeasible solutions.

Fig. 13: Tree topologies.

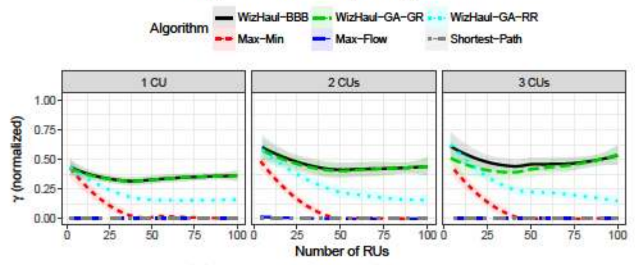


(a) Degree of centralization.

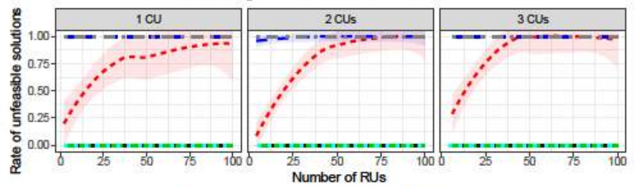


(b) Rate of unfeasible solutions.

Fig. 12: Swiss topology.



(a) Degree of centralization.



(b) Rate of unfeasible solutions.

Fig. 14: Waxman topologies.

The first observation is that both greedy approaches (GA-RR and GA-GR) behave very close to the benchmark BBB in most cases. On the contrary, SHORTEST-PATH, MAX-FLOW and MAX-MIN render low degree of centralization and a high rate of unfeasible solutions (i.e. no centralization whatsoever) across all types of topologies. The reason is that none of the latter three algorithms have the ability to trade off the centralization degree of some flows to benefit larger clusters, like we illustrated in the toy example of §1. This is because they all have simpler goals: SHORTEST-PATH aims to find low-latency paths without favouring disjoint paths; MAX-FLOW targets high-capacity paths irrespective of their distance (latency) towards a CU; MAX-MIN focuses on both capacity and fairness across RUs, without balancing centralization towards larger clusters.

A second observation is that full centralization can only be achieved in low-scale deployments like our Romanian-based scenarios. This is because CUs and RUs are generally too far apart. This implies not only that the tight latency demands of full C-RAN can be rarely met but, also, that more links are compromised, limiting the chances of finding disjoint paths. *Our third observation* is rather intuitive, that is, we can achieve higher degree of centralization in scenarios with lower number of RUs (less contention) and larger number of CUs (shorter paths between RUs and CUs). In addition, the tree-based topologies render lower centralization degree than the other structures under evaluation. This occurs because these scenarios have less diversity of routes (e.g. no disjoint paths) with many links aggregating multiple flows and therefore with fewer opportunities of transporting high-load flows (high-centralization splits).

The last observation is that, whilst GA-GR follows closely the performance of BBB in Waxman-based scenarios, GA-RR deviates as the number of RUs grow. The reason lays in the fact that the linear program used in GA-RR, eq. (6)-(10), tends to find paths of minimum latency concentrating many flows in a few low-latency paths, failing in this way to trade off latency of some flows (that could still meet latency requirements) for path diversification that would release capacity for high-centralization flows. Note that these are the most complex topologies in terms of path diversity.

Regarding the running time of these algorithms, shown in Fig. 15, we can observe that MAX-MIN is the most complex algorithm of them all (though it could be accelerated by using dual variables [14]). Perhaps surprisingly, BBB behaves similarly to GA-RR. This is simply because we use GA-RR as an initial upper bound configuration for BBB and, because GA-RR usually finds a good solution, BBB quickly prunes worse candidate solutions out of the whole space. MAX-FLOW is the fastest because it is based on a simple LP relaxation, faster than SHORTEST-PATH which exhaustively search for a RAN split after path computation.

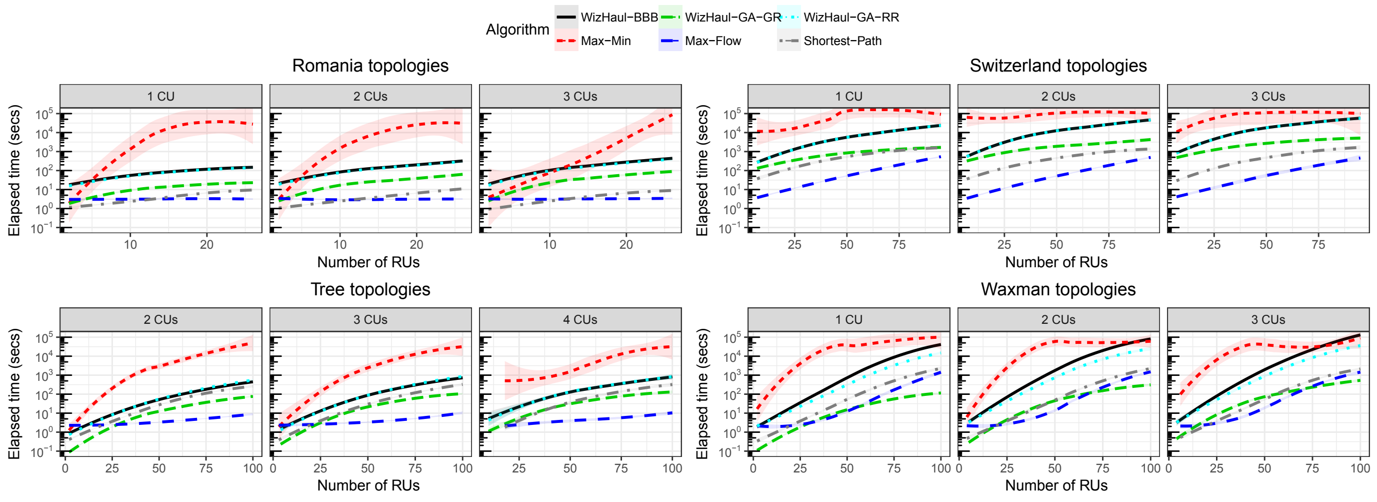


Fig. 15: Elapsed time (s) vs. number of RUs and CUs.

9 Conclusions

A main requirement in the path towards 5G systems is to increase the centralization degree of RAN functionality. Classical C-RAN technology is deemed unfeasible in many realistic scenarios for 5G and new RAN functional splits have been defined to enable a smoother migration. In this paper we showed that next generation C-RAN faces a joint problem to route traffic across fronthaul/backhaul (crosshaul) transport networks while maximizing the amount of centralization. To address this issue, we defined a metric to measure the degree of centralization and proposed three algorithms to maximize it while meeting transport constraints: WizHaul-BBB, WizHaul-GA-GR and WizHaul-GA-RR. WizHaul-BBB provides a near-optimal solution which yields a performance upper bound. WizHaul-GA-GR and WizHaul-GA-RR are greedy heuristics where WizHaul-GA-GR achieves the best trade-off between computational load reduction and distance to the optimal solution. The mechanisms designed can be used both at network planning and operational runtime phases to achieve the maximum centralization degree. WizHaul-GA-GR has been implemented in a proof-of-concept with commercial hardware and its properties at operational runtime have been analyzed.

Acknowledgements

This work has received funding from the European Unions Horizon 2020 research and innovation programme under grant agreement No 671598 (5G-Crosshaul project) and 761536 (5G-Transformer project).

References

[1] Y. Lin *et al.*, "Wireless network cloud: architecture and system requirements," *IBM Journal of Research and Dev.*, vol. 54, Jan. 2010.

[2] K. C. Garikipati *et al.*, "RT-OPEX: Flexible Scheduling for Cloud-RAN Processing," in *Proceedings of the 12th International on Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '16. New York, NY, USA: ACM, 2016, pp. 267–280. [Online]. Available: http://doi.acm.org/10.1145/2999572.2999591

[3] V. Suryaprakash *et al.*, "Are heterogeneous cloud-based radio access networks cost effective?" *IEEE J. Sel. Areas Commun.*, vol. 33, no. 10, pp. 2239–2251, Oct 2015.

[4] "Next Generation Fronthaul Interface," White paper, China Mobile, Alcatel-Lucent, Nokia, ZTE, Broadcom, Intel., June 2015.

[5] "Small Cell Forum, R6.0. Small cell virtualization functional splits and use cases," Jan. 2016.

[6] I. Chih-Lin *et al.*, "Rethink fronthaul for soft RAN," *IEEE Comm. Magazine*, vol. 53, no. 9, pp. 82–88, September 2015.

[7] IEEE, *IEEE 1914 WG "IEEE WG, Next Generation Fronthaul Interface"*.

[8] 3GPP, *TR 38.801: Study on New Radio Access Technology: Radio Access Architecture and Interfaces*, Dec. 2016.

[9] X. Costa-Perez *et al.*, "5G-Crosshaul: An SDN/NFV Integrated Fronthaul/Backhaul Transport Network Architecture," *IEEE Wireless Communications*, vol. 24, no. 1, pp. 38–45, February 2017.

[10] T. Wan and P. Ashwood-Smith, "A performance study of CPRI over Ethernet with IEEE 802.1Qbu and 802.1Qbv enhancements," in *Proc. IEEE GLOBECOM*, Dec 2015, pp. 1–6.

[11] "NEC and Intel collaborate in mobile base station virtualization," Press Release, December 2015. [Online]. Available: http://www.nec.com/en/press/201512/global_20151218_01.html

[12] D. Wubben *et al.*, "Benefits and impact of cloud computing on 5G signal processing: flexible centralization through cloud-RAN," *IEEE Signal Processing Magazine*, vol. 31, no. 6, pp. 35–44, Nov. 2014.

[13] Y. Azar and O. Regev, "Combinatorial algorithms for the unsplittable flow problem," *Algorithmica*, vol. 44, no. 1, pp. 49–66, 2005.

[14] D. Nace and M. Pioro, "Max-min fairness and its applications to routing and load-balancing in communication networks: a tutorial," *IEEE Communications Surveys Tutorials*, vol. 10, no. 4, pp. 5–17, April 2008.

[15] V. T. Chakaravarthy *et al.*, "Varying bandwidth resource allocation problem with bag constraints," in *Proc. of IEEE IPDPS 2010*, Apr. 2010.

[16] NEC NFV C-RAN. http://www.nec.com/en/global/solutions/nsp/sc2/prod/c-ran.html.

[17] N. Alliance, *NGMN 5G white paper*, Feb. 2015.

[18] I. Chih-Lin *et al.*, "Recent Progress on C-RAN Centralization and Cloudification," *IEEE Access*, vol. 2, pp. 1030–1039, 2014.

[19] ——, "Toward green and soft: a 5G perspective," *IEEE Comm. Magazine*, vol. 52, no. 2, pp. 66–73, February 2014.

[20] A. Checko *et al.*, "Cloud RAN for mobile networks – a technology overview," *IEEE Comm. Surv. Tut.*, vol. 17, no. 1, pp. 405–426, 2015.

[21] P. Rost *et al.*, "The complexity-rate tradeoff of centralized radio access networks," *IEEE Trans. Wireless Comm.*, vol. 14, no. 11, pp. 6164–6176, Nov 2015.

[22] M. Peng *et al.*, "Recent advances in cloud radio access networks: System architectures, key techniques, and open issues," *IEEE Communications Surveys Tutorials*, vol. 18, no. 3, pp. 2282–2308, thirdquarter 2016.

[23] L. Han *et al.*, "First demonstration of distributed time synchronization system over transport network towards 5g requirements," in *2017 Optical Fiber Communications Conference and Exhibition (OFC)*, March 2017, pp. 1–3.

[24] U. Dötsch *et al.*, "Quantitative analysis of split base station processing and determination of advantageous architectures for LTE," *Bell Labs Tech. Journal*, vol. 18, no. 1, pp. 105–128, 2013.
[25] P. Rost *et al.*, "Cloud technologies for flexible 5G radio access networks," *IEEE Comm. Magazine*, vol. 52, no. 5, pp. 68–76, May 2014.
[26] A. Checko *et al.*, "Evaluating C-RAN fronthaul functional splits in terms of network level energy and cost savings," *Journal of Communications and Networks*, vol. 18, no. 2, pp. 162–172, April 2016.
[27] C. Y. Chang *et al.*, "Impact of packetization and functional split on C-RAN fronthaul performance," in *Proc. IEEE ICC*, May 2016, pp. 1–7.
[28] ——, "Impact of Packetization and Scheduling on C-RAN Fronthaul Performance," in *2016 IEEE Global Communications Conference (GLOBECOM)*, Dec 2016, pp. 1–7.
[29] A. Thyagaturu *et al.*, "R-FFT: Function Split at IFFT/FFT in Unified LTE CRAN and Cable Access Network," Tech. Rep., 2017.
[30] "Common Public Radio Interface (CPRI): eCPRI 1.0 specification," Sept. 2017.
[31] Chih-Lin *et al.*, "RAN Revolution with NGFI (xhaul) for 5G," *Journal of Lightwave Technology*, vol. PP, no. 99, pp. 1–1, 2017.
[32] X. Foukas *et al.*, "FlexRAN: A Flexible and Programmable Platform for Software-Defined Radio Access Networks," in *Proceedings of the 12th International on Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '16. New York, NY, USA: ACM, 2016, pp. 427–441. [Online]. Available: http://doi.acm.org/10.1145/2999572.2999599
[33] Y. Zaki *et al.*, "Lte wireless virtualization and spectrum management," in *WMNC2010*, Oct 2010, pp. 1–6.
[34] A. Pal, "Approximation algorithms for covering and packing problems on paths," Ph.D. dissertation, 2014.
[35] J. M. Kleinberg, "Approximation algorithms for disjoint paths problems," Ph.D. dissertation, Cambridge, MA, USA, 1996.
[36] K. Elbassioni *et al.*, "Approximation algorithms for the unsplittable flow problem on paths and trees," in *Proc. of LIPIcs*, vol. 18, 2012.
[37] L. Epstein *et al.*, "Online capacitated interval coloring," *SIAM Journal on Discrete Mathematics*, vol. 23, no. 2, pp. 822–841, 2009.
[38] T. W. Kuo *et al.*, "Deploying chains of virtual network functions: On the relation between link and server usage," in *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, April 2016, pp. 1–9.
[39] J. Kim *et al.*, "Joint scalable coding and routing for 60 GHz real-time live HD video streaming applications," *IEEE Trans. on Broadcasting*, vol. 59, no. 3, pp. 500–512, Sept 2013.
[40] M. M. U. Rahman *et al.*, "RRH clustering and transmit precoding for interference-limited 5G CRAN downlink," in *IEEE Globecom Workshops*, Dec 2015, pp. 1–7.
[41] Y. Du and G. de Veciana, "Wireless networks without edge: dynamic radio resource clustering and user scheduling," in *Proc. IEEE INFOCOM*, April 2014, pp. 1321–1329.
[42] C. Y. Yeoh *et al.*, "Performance study of LTE experimental testbed using OpenAirInterface," in *Proc. of ICACT 2016*, Jan 2016, pp. 617–622.
[43] N. Nikaein, "Processing radio access network functions in the cloud: Critical issues and modeling," in *Proceedings of the 6th International Workshop on Mobile Cloud Computing and Services*, ser. MCS '15. New York, NY, USA: ACM, 2015, pp. 36–43. [Online]. Available: http://doi.acm.org/10.1145/2802130.2802136
[44] M. E. Lalami and D. El-Baz, "GPU implementation of the branch and bound method for knapsack problems," in *Proc. of IEEE IPDPSW 2012*, May 2012, pp. 1769–1777.
[45] J. Y. Yen, "An algorithm for finding shortest routes from all source nodes to a given destination in general networks," *Quarterly of Applied Mathematics*, vol. 27, no. 4, pp. 526–530, 1970.
[46] A. Kashyap *et al.*, "Single-path routing of time-varying traffic," in *Proc. IEEE GLOBECOM*, Nov 2006, pp. 1–6.
[47] J. i. Kani, "Solutions for future mobile fronthaul and access-network convergence," in *2016 Optical Fiber Communications Conference and Exhibition (OFC)*, March 2016, pp. 1–43.
[48] Z. Tayq *et al.*, "Real time demonstration of the transport of ethernet fronthaul based on vran in optical access networks," in *2017 Optical Fiber Communications Conference and Exhibition (OFC)*, March 2017, pp. 1–3.
[49] A. S. Thyagaturu *et al.*, "Software Defined Optical Networks (SDONs): A Comprehensive Survey," *IEEE Communications Surveys Tutorials*, vol. 18, no. 4, pp. 2738–2786, Fourthquarter 2016.
[50] M. Z. Mao *et al.*, "Dsp-enabled reconfigurable and transparent spectral converters for converging optical and mobile fronthaul/backhaul networks," *Opt. Express*, vol. 25, no. 12, pp. 13 836–13 856, Jun 2017. [Online]. Available: http://www.opticsexpress.org/abstract.cfm?URI=oe-25-12-13836
[51] OpenEPC. http://www.openepc.com/.
[52] NEC iPASOLINK VR4. http://www.nec.com/en/global/prod/nw/pasolink/products/ipasolink_VR4.html.
[53] NEC iPASOLINK EX. http://www.nec.com/en/global/prod/nw/pasolink/products/ipasolinkEX.html.
[54] NEC ProgrammableFlow. http://www.nec.com/en/global/prod/pflow/pf5240.html.
[55] NEC MB4420 small-cell. http://www.nec.com/en/global/solutions/nsp/sc2/prod/e-nodeb.html.
[56] HP DL380p Gen8 server. https://www.hpe.com/h20195/v2/default.aspx?cc=za&lc=en&oid=5177957.
[57] B. M. Waxman, "Routing of multipoint connections," *IEEE J. Sel. Areas Commun.*, vol. 6, no. 9, pp. 1617–1622, Dec 1988.
[58] J. Lessmann, "Resource optimization in realistic mobile backhaul networks," in *Proc. IEEE ICC*, June 2015, pp. 3861–3866.
[59] W. S. Cleveland *et al.*, "Locally weighted regression: an approach to regression analysis by local fitting," *Journal of the American Statistical Assoc.*, vol. 83, no. 403, pp. 596–610, 1988.

Andres Garcia-Saavedra received his M.Sc and Ph.D. from University Carlos III of Madrid (UC3M) in 2010 and 2013, respectively. He then joined the Hamilton Institute, Ireland, as a Research Fellow till the end of 2014 when he moved to Trinity College Dublin (TCD). Since July 2015 he is a Senior Researcher at NEC Laboratories Europe. His research interests lie in the application of fundamental mathematics to real-life communications systems and the design and prototype of wireless systems and protocols.

Josep Xavier Salvat received his M.Sc from Polytechnic University of Catalonia (UPC-BarcelonaTech) in 2016. He currently works as a Research Associate in NEC Laboratories Europe. Since 2017, he is pursuing his Ph.D. degree at the Kaiserslautern University (TU KL) in Germany. His research interests are computer networks, 5G transport networks and the design and prototyping of SDN/NFV solutions.

Xi Li Xi Li received her M.Sc. in 2002 from TU Dresden and her Ph.D. in 2009 from the University of Bremen, Germany, where she worked as a research fellow and lecturer. In 2014 she worked as a solution designer at Telefonica, Germany. Since 2015 she has been with NEC Laboratories Europe, working as a senior researcher on 5G networks. Her research interest are mobile backhaul, fronthaul and backhaul integration, SDN/NFV, multihomed and multiconnectivity management.

Xavier Costa-Perez is head of 5G R&D at NEC Laboratories Europe, where he manages several projects focused on 5G mobile core, backhaul/fronthaul, and access networks. He is a 5GPPP Technology Board member and Technical Manager of the 5G-Transformer project. His team contributes to projects for NEC products roadmap evolution, European Commission research projects and related SDOs. He received his M.Sc. and Ph.D. in telecommunications from the Polytechnic University of Catalonia.