The Impact of Baseband Functional Splits on Resource Allocation in 5G Radio Access Networks

Iordanis Koutsopoulos
Department of Informatics
Athens University of Economics and Business
Athens, Greece
jordan@aueb.gr

***Abstract*—We study physical-layer (PHY) baseband functional split policies in 5G Centralized Radio-Access-Network (C-RAN) architectures that include a central location, the baseband unit (BBU) with some BBU servers, and a set of Base Stations (BSs), the remote radio heads (RRHs), each with a RRH server. Each RRH is connected to the BBU location through a fronthaul link. We consider a scenario with many frame streams at the BBU location, where each stream needs to be processed by a BBU server before being sent to a remote radio-head (RRH). For each stream, a functional split needs to be selected, which provides a way of partitioning the computational load of the baseband processing chain for stream frames between the BBU and RRH servers. For streams that are served by the same BBU server, a scheduling policy is also needed. We formulate and solve the joint resource allocation problem of functional split selection, BBU server allocation and server scheduling, with the goal to minimize total average end-to-end delay or to minimize maximum average delay over RRH streams. The total average end-to-end delay is the sum of (*i*) scheduling (queueing) and processing delay at the BBU servers, (*ii*) data transport delay at the fronthaul link, and (*iii*) processing delay at the RRH server. Numerical results show the resulting delay improvements, if we incorporate functional split selection in resource allocation.**

***Index Terms*—Functional split selection, resource allocation, end-to-end delay, optimization.**

I. Introduction

Recent developments in 5G wireless architectures have brought forth the promising architectural paradigm of Centralized Radio-Access-Network (C-RAN), also referred to as *Cloud-RAN*. In traditional architectures, the whole chain of data-link control (DLC) and physical (PHY) layer functionalities is carried out at Base Stations (BSs). In C-RAN architectures, BS functionalities become *disaggregated*, and most of them take place at a central location, the Baseband Unit (BBU) one, while each BS/Remote radio head (RRH) performs only time-domain RF processing and A/D conversion.

There exist several advantages in a centralized architecture. First, remote BSs have easier maintenance and lower deployment costs for mobile operators. Second, with virtualization technologies, computational and other resources are consolidated at the BBU location, and this leads to resource utilization efficiency. Furthermore, many small cells can be jointly coordinated with advanced techniques such as coordinated multi-point (CoMP) transmission and inter-cell interference management, and through global system optimization. On the other hand, the centralized approach places significant throughput strain and stringent delay constraints on the *fronthaul* transport network from the BBU to RRHs.

The concept of *flexible functional splits* is a tradeoff between these two extremes, by splitting the computational load of the chain of functions between the BBU and RRH locations. A functional split is a *partition* of the chain of MAC-layer and PHY-layer baseband processing functions into two sub-chains, one executed at the BBU, and the other at the RRH. For a chain of r functions, there exist $(r-1)$ possible functional splits. A functional split determines the amount of *computational load* at the BBU and RRH servers and the amount of *transported data traffic* from the BBU to RRHs. Such data are needed by the RRH to run its sub-chain of computations.

Consider for example modulation that may be executed at the BBU or at the RRH in the downlink. Assume m-QAM modulation, a q-bit representation per complex baseband sample, and one radio resource block (RRB) in LTE-Advanced, with a user symbols. If modulation is the first executed function of the RRH sub-chain, $a \log_2 m$ bits need to be sent from the BBU to the RRH, while if modulation is the last executed function of the BBU sub-chain, $2aq$ bits that represent the I/Q complex baseband samples need to be sent to the RRH.

Several options exist for functional splits, spanning DLC and PHY-layer functionalities [1]. PHY-layer functionalities are channel coding and decoding, modulation and demodulation, resource mapping and demapping, Fast Fourier transform (FFT) and inverse FFT, channel estimation and equalization, analog-to-digital (A-to-D) and D-to-A conversion, and antenna transmission and reception. DLC functionalities include the Packet Data Convergence Protocol (PDCP), Radio Link Control (RLC) and Media Access Control (MAC), with tasks such as data multiplexing, packet scheduling, CoMP, automatic repeat-request (ARQ) protocols, packet reordering, handover management, and security functions such as ciphering.

In this paper, we study PHY-layer functional splits of the chain of baseband processing with the goal to understand their impact on higher-layer resource allocation decisions. We use *end-to-end frame delay* as a performance metric. For downlink, this is the time elapsed from the moment when a frame arrives

This work was supported by the CHIST-ERA LeadingEdge project, from call "Smart Distribution of Computing in Dynamic Networks (SDCDN)".

at the BBU location, until the moment when the frame is transmitted to the user by the RRH antenna.

A. Our contribution

We consider a scenario with multiple traffic streams at the BBU location, where each stream needs to be allocated to a BBU server before being sent to a RRH through the fronthaul link. Since the latter is usually an optical fiber with abundant bandwidth, it affects delay only through data transport, namely there is no queueing delay at the fronthaul link. On the other hand, BBU servers are realized through virtual machines (VM) at the BBU location, and this implies finite computing capacity. Each frame of the stream needs to undergo baseband processing at a BBU server prior to transmission. For each stream, a functional split needs to be selected which is a way of splitting the required computational load of the chain of baseband processing between the BBU and RRH. The functional split also determines the volume of fronthaul data, and thus the data transport delay. The assignment of frame streams to BBU servers determines the computational load of each server. If more than one streams are served by the same server, a scheduling policy is also needed.

The total average end-to-end delay per frame is the sum of (*i*) scheduling (queueing) and processing delay at the BBU server, (*ii*) data transport delay at the fronthaul link, and (*iii*) processing delay at the RRH server. Scheduling delay depends on the functional split selection, server allocation and scheduling policies, while the other two delays depend only on the functional split selection policy. This work contributes to the literature is as follows:

- We model frame processing at each BBU server with a $M/D/1$ queue, with deterministic computational job size per frame that depends on the selected functional split. The job size and the amount of transported data on the fronthaul are increasing functions of the functional split.
- We formulate the joint optimization problem of functional split selection, BBU server allocation and server scheduling with the aim to minimize the total average end-to-end delay over RRH streams, and to minimize the maximum average delay over RRH streams.
- We characterize the solution and complexity for the cases of one BBU server ($K = 1$), and for the cases when the number of BBU servers, K is equal to or smaller than the number of streams, N ($K = N$, $K < N$). Each of these cases leads to an optimization problem with a different set of controls. For $K = 1$ and $K < N$, the problems are continuous-valued and convex, hence they can be solved optimally in an efficient manner. For $K = N$, the problem is equivalent to a minimum-cost bipartite matching one, and hence solvable in polynomial time.

Current works consider a fixed functional split, same for all RRHs, which is selected with back-of-envelope calculations on throughput and delay. Our work fills the gap of deciding on the optimal functional split selection, with a view towards delay minimization. Further, this work is the first to introduce a queueing model to characterize the impact of limited BBU computing resources on average delay. It also makes plausible assumptions to capture fronthaul transport and RRH server processing delay. Our model and joint consideration of functional splits, stream-to-BBU-server assignment and scheduling are novel, and they showcase in a systematic way the interaction and coexistence of functional split selection with critical resource allocation decisions at the BBU location. In section II, we present the model and assumptions. In sections III and IV we formulate the problem and different extensions of it. Numerical results are presented in section V, and related work is discussed in section VI. We conclude in section VII.

II. Model

A. Frame streams and fronthaul

We consider a C-RAN architecture with a central BBU location with K co-located servers. Server k has computing capacity C_k operations (ops)/sec. There are also N RRHs. The server of RRH n has computing capacity C_n^R ops/sec, $n = 1, \ldots, N$. We do not consider coordinated multi-point (CoMP) transmission, thus each user is served by one RRH.

At the BBU location, there exist N downlink traffic streams, where stream $n \in \{1, \ldots, N\}$ carries traffic frames of users of RRH n and needs to be transported to RRH n. The frame arrival process for stream n is Poisson with mean arrival rate λ_n frames/sec. The Poisson assumption is justified by the fact that each stream n consists of many independent small substreams corresponding to individual users of each RRH, and the fact that the aggregate of a large number of (not necessarily Poisson) independent arrival processes, can be approximated by a Poisson process with arrival rate equal to the sum of individual arrival rates [2, Chap.5]. Let $\lambda = \sum_{n=1}^{N} \lambda_n$ be the total frame arrival rate for all RRHs at the BBU location.

A frame of stream n is viewed at the BBU servers as a computation job with *fixed*, known computing requirements w_n, also refered to as load or job size. The computational load refers to the arithmetic operations that are needed to execute the entire chain of PHY-layer baseband processing of the frame. In practice, w_n varies according to random factors such as the used modulation scheme, coding rate and SNR [3], [4]. For instance, higher modulation schemes and coding rates lead to larger computational load w_n, due to the complex processes of coding/decoding. Here, we assume that we know the joint probability distribution of using different modulation and coding schemes, and thus w_n is the expected value of job size for a frame of stream n with respect to that distribution. Without loss of generality, we normalize it in $[0, 1]$, where $\max_n w_n = 1$.

We consider a simple fronthaul topology in which the BBU location is connected to each RRH n through a link of communication capacity u_n bits/sec. The model and fronthaul topology are depicted in Fig. 1.

B. Functional splits

A functional split is a *partition* of the baseband processing computation chain into two sub-chains, one executed at the BBU location and the other at the RRH. It results in a portion

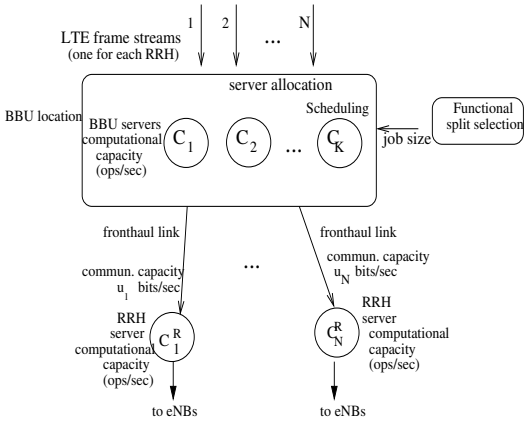

Fig. 1. Pictorial view of basic components of the model.

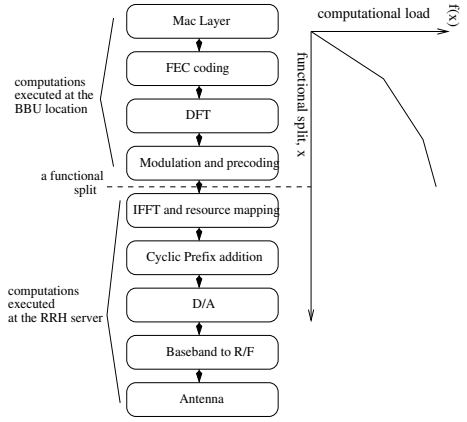

Fig. 2. An example chain of baseband processing computation, a possible functional split, and a possible form of increasing function $f(\cdot)$. The upper and lower parts of the chain with respect to the functional split denote computations performed at the BBU and RRH respectively.

of computational load of each frame executed at the BBU servers, and the rest at the RRH. A set of possibilities exist for functional splits. Fig. 2 depicts a computation chain and a given functional split.

Let $x \in [0,1]$ denote the chain partition, where $x = 1$ and $x = 0$ respectively mean that all chain functions are executed at the BBU or RRH. Let $f(x)$ denote the amount of computing at BBU that corresponds to chain partition x, where $f(\cdot) : [0,1] \to [0,1]$ is a continuous, increasing function. We assume that $f(x)$ is increasing and linear function, whose form can be obtained through historical data (e.g. a dataset that involves chain partitions and computational load measurements), and a machine-learning curve-fitting model e.g., linear regression.

For each frame stream n, $n = 1, \ldots, N$, a different functional split x_n may be selected. The amount of computation (size of computational job) per frame at the BBU location and at the RRH n is $f(x_n) = w_n x_n$ and $w_n(1 - x_n)$ respectively.

A functional split x_n results in certain amount of data at the BBU that needs to be transported to RRH n over the fronthaul link. Measurements show that this amount of data increases with the number of functions executed at the BBU [5], [6]. We assume that the amount of data per frame for functional split x is given by an increasing function $r(x)$, whose form can be found with curve-fitting methods on data.

Remark: Other forms of $f(\cdot)$ are possible. For instance, if measurements suggest that early-stage functions in the chain (e.g. coding, DFT) are more computation-intense than subsequent functions, and that these in turn are more computation-intensive than next ones, $f(\cdot)$ could be modeled as piece-wise linear concave function, as in Fig. 2.

C. Frame computation scheduling at the BBU servers

We consider the class of non-preemptive priority scheduling policies $\mathcal{P}$. Namely, new arriving higher-priority jobs do not interrupt the service of a lower-priority job, but they wait until the service time of that job is completed. A pure priority scheduling policy $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_N) \in \mathcal{P}$ is a vector whose n-th component $\pi_n \in \{1, \ldots, N\}$ denotes the priority with which frames of the n-th stream are served, and $(\pi_1, \ldots, \pi_N)$ is a permutation of $\{1, \ldots, N\}$. For example for $N = 2$, it is $\mathcal{P} = \{(1,2), (2,1)\}$. For policy $(2,1)$ it is $\pi_1 = 2$ and $\pi_2 = 1$, i.e., stream 2 is served with highest (first) priority.

Let $\mathbf{W}(\boldsymbol{\pi}) = (W_1(\boldsymbol{\pi}), \ldots, W_N(\boldsymbol{\pi}))$ be the stream queueing delay vector achieved by pure priority scheduling policy $\boldsymbol{\pi}$. Let $\mathcal{W}$ be the set of queueing delay vectors achievable by any scheduling policy. We have [7, Lec. 2], [8] that

$$\mathcal{W} = \text{Conv.Hull}\{\mathbf{W}(\boldsymbol{\pi}) : \boldsymbol{\pi} \in \mathcal{P}\}, \tag{1}$$

where the convex hull of discrete set $\mathcal{X} = \{x_1, \ldots, x_m\}$ is

$$\text{Conv.Hull}(\mathcal{X}) = \{y \colon y = \sum_{i=1}^{m} a_i x_i : \sum_{i=1}^{m} a_i = 1 \text{ and } a_i \geq 0 \,\forall\, i\}. \tag{2}$$

Thus, any achievable delay vector by *any* scheduling policy can be obtained through an appropriate convex combination of pure priority scheduling policies.

Given the Poisson frame arrival processes and the deterministic (yet, functional split-dependent) processing times of computation jobs, a BBU server may be modeled as a $M/D/1$ queue. We consider the following three cases for BBU servers.

1) Case 1: One BBU server ($K = 1$): If BBU computing resources are aggregated to a single server of computing capacity C ops/sec, the controller needs to choose a functional split policy $\mathbf{x} = (x_1, \ldots, x_N)$ and a server scheduling policy $\boldsymbol{\pi}$.

The BBU server is a multi-class $M/D/1$ queue with priorities, where each class corresponds to a RRH stream. Starting from the $M/G/1$ queue with priorities [9, Section 3.5.3], we get that the average waiting time in the queue (queueing delay) for a frame of stream n for priority scheduling policies is

$$W_n(\mathbf{x}, \boldsymbol{\pi}) = \frac{\frac{1}{2C^2} \sum_{n=1}^{N} \lambda_n f^2(x_n)}{\Big(1 - \frac{1}{C} \sum_{i:\pi_i < \pi_n} \lambda_i f(x_i)\Big)\Big(1 - \frac{1}{C} \sum_{i:\pi_i \leq \pi_n} \lambda_i f(x_i)\Big)}. \tag{3}$$

Queueing delay depends on the functional split policy $\mathbf{x}$ and the scheduling policy $\boldsymbol{\pi}$. The queueing delay for a frame of stream n depends on the arrival rates and service times of jobs of higher priority than those of n. If $f(\cdot)$ is *linear*, then $W_n(\mathbf{x}, \boldsymbol{\pi})$ is convex in $\mathbf{x}$. Indeed, it can be verified through standard algebra that e.g., for $N = 2$ the Hessian matrix of $W_n(\mathbf{x}, \boldsymbol{\pi})$ is positive semi-definite for all $\mathbf{x}$ and any $\boldsymbol{\pi}$.

The average service time per frame of stream n is $S_n(x_n) = f(x_n)/C$, and the total average delay per frame of stream n at the BBU server is

$$d_n^B(\mathbf{x}, \boldsymbol{\pi}) = W_n(\mathbf{x}, \boldsymbol{\pi}) + S_n(x_n)\,. \tag{4}$$

If we allow schedules to be convex combinations of pure priority policies, the set of feasible scheduling policies $\mathcal{P}^*$ is the set of $(1 \times N!)$ probability vectors $\mathbf{p} = (p_1, \ldots, p_{N!})$, where p_i denotes the *portion* of time when a pure priority policy $\boldsymbol{\pi}_i$, $i = 1, \ldots, N!$ is used, and $\sum_i p_i = 1$. Then, the average queueing delay is

$$W_n(\mathbf{x}, \mathbf{p}) = \sum_{i=1}^{N!} p_i W_n(\mathbf{x}, \boldsymbol{\pi}_i)\,. \tag{5}$$

D. Transport over the fronthaul and RRH processing

After completion of the computation for a frame at the BBU location, the generated data for this frame is transported over the fronthaul to RRH n. Since the fronthaul link is usually an optical fiber, fronthaul link capacity is abundant and link queueing does not occur. If functional split x_n is chosen for stream n, the data transport delay from the BBU to RRH n is

$$d_n^F(x_n) = \frac{r(x_n)}{u_n}\,. \tag{6}$$

When data arrives at RRH n, the remaining $w_n(1-x_n)$ amount of the computation per frame takes place at the RRH server. Since frames arrive at RRH server n at a given, fixed rate $1/d_n^F(x_n)$ and computational jobs have fixed size, equal to $w_n(1 - x_n)$, the RRH server may be modeled as a $D/D/1$ queue. If functional split x_n is chosen so that the arrival rate is less than the service rate at that queue, i.e.

$$\frac{u_n}{r(x_n)} < \frac{C_n^R}{w_n(1 - x_n)}\,, \forall\ n \tag{7}$$

there will be no queueing delay at the RRH server, and frame processing takes time

$$d_n^R(x_n) = \frac{w_n(1 - x_n)}{C_n^R}\,. \tag{8}$$

The *end-to-end delay* for a frame destined to RRH n is

$$D_n(\mathbf{x}, \boldsymbol{\pi}) = d_n^B(\mathbf{x}, \boldsymbol{\pi}) + d_n^F(x_n) + d_n^R(x_n)\,. \tag{9}$$

If $f(\cdot)$ is linear, then $D_n(\mathbf{x}, \boldsymbol{\pi})$ is a convex function.

III. Problem Formulation and solution

A first objective, which we refer to as *objective I*, is to minimize total average end-to-end delay (9), which depends on the scheduling policy at the BBU server, and on the functional split policy. A given functional split x_n for stream n affects the execution time of sub-chains of functions at the BBU server and the server of RRH n. Together with the scheduling policy at the BBU server, it also affects the queueing delays of other streams that are served with lower priority than stream n at the BBU server. The functional split also influences fronthaul data transport delay to RRH n through the volume of generated data per frame.

In order to impose a sense of fairness in treating different frame streams, a second objective which we call *objective II* is to minimize the maximum end-to-end delay over all streams. For the objective functions, we use notation $\bar{D}_\omega^z(\cdot)$ where $\omega \in \{1, 2, 3\}$ denotes one of the three cases: Case 1: $K = 1$; case 2: $K = N$; case 3: $K < N$; the latter two cases will be studied in section IV); and $z \in \{I, II\}$ denotes objective I or II.

A. Objective I: Minimize total average end-to-end delay for $K = 1$ BBU server

We need to solve:

$$\min_{\mathbf{x}, \boldsymbol{\pi}} \bar{D}_1^I(\mathbf{x}, \boldsymbol{\pi}) = \frac{1}{\lambda} \sum_{n=1}^{N} \lambda_n D_n(\mathbf{x}) = \\ = \frac{1}{\lambda} \sum_{n=1}^{N} \lambda_n \Big(W_n(\mathbf{x}, \boldsymbol{\pi}) + S_n(x_n) + d_n^F(x_n) + d_n^R(x_n) \Big)\,. \tag{10}$$

with $W_n(\mathbf{x}, \boldsymbol{\pi})$ given by (3), subject to $0 \le x_n \le 1$ and (7) for each stream n, and subject to the necessary and sufficient condition for queue stability, $\sum_{n=1}^{N} \lambda_n f(x_n) < C$.

a. Fixed functional split selection policy. If the functional split selection policy $\mathbf{x}^0$ is given, the scheduling policy that minimizes $\bar{D}_1^I(\mathbf{x}^0, \boldsymbol{\pi})$ is *Shortest-Processing-Time-First (SPTF)*, which minimizes the total average queueing delay, $\sum_{n=1}^{N} \lambda_n W_n(\cdot)$. SPTF serves streams n in increasing order of $f(x_n)$.

b. Fixed scheduling policy. For fixed scheduling policy $\boldsymbol{\pi}^0$, we need to minimize $\bar{D}_1^I(\mathbf{x}, \boldsymbol{\pi}^0)$ with respect to $\mathbf{x}$ subject to constraints $0 \le x_n \le 1$, constraints (7), and the queue stability constraint. This is a *convex* optimization problem in $\mathbf{x}$ that can be solved through first-order optimality conditions [10, Ch.3] or numerical methods such as gradient descent.

c.Joint problem. If we allow scheduling policies to include convex combinations of pure priority schedules i.e., schedules $\mathbf{p} \in \mathcal{P}^*$, the joint problem of finding the functional split $\mathbf{x}$ and scheduling policy $\mathbf{p}$ so as to minimize $\bar{D}_1^I(\mathbf{x}, \mathbf{p})$ is a nonlinear program (NLP) that can solved with numerical methods.

If we consider only pure priority scheduling policies in discrete set $\mathcal{P}$, the joint problem of finding the functional split and scheduling policy $(\mathbf{x}, \boldsymbol{\pi})$ so as to minimize $\bar{D}_1^I(\mathbf{x}, \boldsymbol{\pi})$ is a mixed-integer convex programming (MICP) problem that can be solved with numerical methods and solvers [11].

If N is small and the $N!$ scheduling policies can be enumerated, we may proceed as follows. For each pure priority scheduling policy $\boldsymbol{\pi}_i$, $i = 1, \ldots, N!$, we find $\mathbf{x}_i = \arg\min_{\mathbf{x}\in A} \bar{D}_1^I(\mathbf{x}, \boldsymbol{\pi}_i)$, where A is the set of feasible solutions determined by the constraints above, and we pick solution $\mathbf{x}^* = \arg\min_i \bar{D}_1^I(\mathbf{x}_i, \boldsymbol{\pi}_i)$ as the optimal solution.

If N is not small, a heuristic algorithm may be as follows. We start with an arbitrary scheduling policy $\boldsymbol{\pi}(0)$ such as a given priority policy or First-Come-First-Serve. At iteration 1, we find $\mathbf{x}(1) = \arg\min_{\mathbf{x}\in A} \bar{D}_1^I(\mathbf{x}, \boldsymbol{\pi}(0))$. For solution $\mathbf{x}(1)$, let $\boldsymbol{\pi}(1)$ be the corresponding SPTF policy. Then, we fix $\boldsymbol{\pi}(1)$ and find $\mathbf{x}(2) = \arg\min_{\mathbf{x}\in A} \bar{D}_1^I(\mathbf{x}, \boldsymbol{\pi}(1))$, and so on until convergence to a local minimum is achieved.

B. Objective II: Minimize maximum average end-to-end delay for $K = 1$ BBU server

In objective II, the goal is to balance average end-to-end delays across RRH streams as much as possible. We solve:

$$\min_{\mathbf{x},\boldsymbol{\pi}} \bar{D}_1^{II}(\mathbf{x}, \boldsymbol{\pi}) = \min_{\mathbf{x},\boldsymbol{\pi}} \max_{n=1,\ldots,N} D_n(\mathbf{x}, \boldsymbol{\pi}) \tag{11}$$
$$= \min_{\boldsymbol{\pi},\mathbf{x}} \max_n \Big(W_n(\boldsymbol{\pi}, \mathbf{x}) + S_n(x_n) + d_n^F(x_n) + d_n^R(x_n) \Big)$$

subject to the same constraints as objective I above.

a. Fixed scheduling policy. If scheduling policy $\boldsymbol{\pi}^0$ is fixed, the problem $\min_{\mathbf{x}} \bar{D}_1^{II}(\mathbf{x}, \boldsymbol{\pi}^0)$ is solved by first defining an auxiliary variable, $t = \max_n D_n(\mathbf{x}, \boldsymbol{\pi}^0)$. Then, the problem becomes a nonlinear program (NLP):

$$\min_{\mathbf{x}, t>0} \; t \tag{12}$$

subject to:

$$D_n(\mathbf{x}, \boldsymbol{\pi}^0) \le t \,, \forall\; n \,, \tag{13}$$

and the same constraints on $\mathbf{x}$ as in objective I above.

b. Fixed functional split policy. If the functional split $\mathbf{x}^0$ is fixed, then, in order to solve $\min_{\boldsymbol{\pi}} \max_n D_i(\mathbf{x}^0, \boldsymbol{\pi})$, we define variable $t = \max_n D_n(\mathbf{x}^0, \boldsymbol{\pi}) > 0$. We need to find t and the schedule that solve

$$\min_{\boldsymbol{\pi}, t>0} \; t \tag{14}$$

subject to:

$$W_n(\boldsymbol{\pi}) + \gamma_n \le t \text{ for all } n \,, \tag{15}$$

where $\gamma_n = S_n(x_n^0) + d_i^F(x_n^0) + d_n^R(x_n^0)$. We can express $W_n(\boldsymbol{\pi})$ as in (5), and then derive t and the mixtures p_i of pure priority policies $\boldsymbol{\pi}_i$, for $i = 1, \ldots, N!$ from a linear programming (LP) problem.

Alternatively, we can use the conservation law at the queue,

$$\sum_{n=1}^{N} \rho_n W_n(\boldsymbol{\pi}) = \frac{\rho}{1-\rho} W_0 = B \,, \tag{16}$$

where $\rho_n = \lambda_n f(x_n^0)/C$ is the server utilization factor due to stream n, $\rho = \sum_n \rho_n$ is the total utilization, and $W_0 = \sum_n \lambda_n (\frac{f(x_n)}{C})^2$. Then, problem (14) subject to (15), (16) is a linear programming (LP) problem with unknowns $\{W_n(\boldsymbol{\pi})\}$, $n = 1, \ldots, N$ and t, and its solution gives the queueing delay vector $\mathbf{W} = (W_n(\boldsymbol{\pi}) : n = 1, \ldots, N)$.

Next, recall that the achievable delay region i.e. the set of achievable queueing delay vectors under any scheduling policy is the convex hull of the set of queueing delay vectors, each of which is achieved by a pure priority scheduling policy. Thus, vector $\mathbf{W}$ is attained through a unique convex combination of pure priority policies. Therefore, we can solve a linear system of equations to derive the time portions p_i when each pure scheduling policy $\boldsymbol{\pi}_i$, $i = 1, \ldots, N!$ will be employed so as to achieve the vector $\mathbf{W}$ of queueing delays.

c. Joint problem. The joint problem is a NLP with unknowns t, the mixtures $\{p_i\}$ for $i = 1, \ldots, N!$, and vector $\mathbf{x}$.

IV. Model extensions

A. Case 2: Number of BBU servers $K = N$

When virtualization allows the creation of a number of BBU servers $K = N$, each stream is assigned to one BBU server, and each server serves one stream. The computing capacity of BBU server k is C_k ops/sec. Besides the functional split policy $\mathbf{x}$, we need to find a stream-to-server assignment policy $\boldsymbol{\alpha} = (\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_N)$, where $\boldsymbol{\alpha}_n = (\alpha_{n1}, \ldots, \alpha_{nK})$ is a $0-1$ vector with one entry equal to 1 and all other entries 0, with $\alpha_{nk} = 1$ if stream n is assigned to BBU server k, and 0 otherwise.

Assume that stream n is assigned to server $k(n)$, i.e., $\alpha_{n\,k(n)} = 1$. Since each BBU server is a $M/D/1$ queue, the total average BBU (queueing and processing) delay for stream n is

$$d_n^B(\boldsymbol{\alpha}_n, x_n) = d_n^B(k(n), x_n) = \frac{\lambda_n \Big(\frac{f(x_n)}{C_{k(n)}}\Big)^2}{2\Big(1 - \frac{\lambda_n f(x_n)}{C_{k(n)}}\Big)} + \frac{f(x_n)}{C_{k(n)}} \,. \tag{17}$$

and the total average end-to-end delay is

$$D_n(\boldsymbol{\alpha}_n, \mathbf{x}) = d_n^B(\boldsymbol{\alpha}_n, x_n) + d_n^F(x_n) + d_n^R(x_n) \,. \tag{18}$$

1) Objective I: Minimize total average delay: When $K = N$, we need to find a stream-to-server assignment policy $\boldsymbol{\alpha}$ and a functional split policy $\mathbf{x}$ to solve

$$\min_{\boldsymbol{\alpha},\mathbf{x}} \bar{D}_2^I(\boldsymbol{\alpha}, \mathbf{x}) = \frac{1}{\lambda} \sum_{n=1}^{N} \lambda_n \Big(d_n^B(\boldsymbol{\alpha}_n, x_n) + d_n^F(x_n) + d_n^R(x_n) \Big) \,, \tag{19}$$

subject to: (*i*) assignment constraints $\sum_{k=1}^{K} \alpha_{nk} = 1$ for each stream n, and $\sum_{n=1}^{N} \alpha_{nk} = 1$ for each server k, (*ii*) $0 \le x_n \le 1$ for each stream n, (*iii*) constraint (7) for each n, and (*iv*) stability constraints $\lambda_n f(x_n) < C_{k(n)}$ for each n.

We construct a bipartite graph with a node set $\mathcal{U}_1$ of N nodes with one node for each RRH stream, and a node set $\mathcal{U}_2$ of $K = N$ nodes, with one node for each BBU server. For each link (n, k) between $n \in \mathcal{U}_1$ and $k \in \mathcal{U}_2$, we define a cost

$$\beta_{nk} = \min_{x_n \in A_{nk}} \lambda_n \Big(d_n^B(k, x_n) + d_n^F(x_n) + d_n^R(x_n) \Big) \,, \tag{20}$$

where A_{nk} is the feasible region for x_n when stream n is assigned to server k, i.e. it is the set of x_n's that satisfy

constraints (*ii*)-(*iv*) above. The optimal solution is found through *minimum-cost bipartite matching* in the graph above in polynomial time through the Hungarian algorithm [12].

2) Objective II: Minimize maximum average delay: We need to solve problem:

$$\min_{\boldsymbol{\alpha},\mathbf{x}} D_2^{II}(\boldsymbol{\alpha},\mathbf{x}) = \min_{\boldsymbol{\alpha},\mathbf{x}} \max_n \Big(d_n^B(\boldsymbol{\alpha}_n, x_n) + d_n^B(x_n) + d_n^R(x_n) \Big), \tag{21}$$

subject to the constraints above for case 2, objective I. We again use auxiliary variable t.

a. Fixed server assignment. Assume a fixed server assignment $\boldsymbol{\alpha}^0$, such that $\alpha^0_{n\,k(n)} = 1$ for each n, i.e. stream n is assigned to server $k(n)$. The problem becomes:

$$\min_{t,\mathbf{x}} t \tag{22}$$

subject to:

$$d_n^B(k(n), x_n) + d_n^B(x_n) + d_n^R(x_n) \leq t \;\forall\; n\,, \tag{23}$$

and subject to constraints (*ii*)-(*iv*) above on $\mathbf{x}$. This is a NLP.

b. Fixed functional split policy. If functional split $\mathbf{x}^0$ is given, we define the bipartite graph as above, with cost for link (n,k),

$$\beta^0_{nk} = \lambda_n \Big(d_n^B(k, x_n^0) + d_n^F(x_n^0) + d_n^R(x_n^0) \Big)\,. \tag{24}$$

The problem becomes:

$$\min_{t,\boldsymbol{\alpha}} t \tag{25}$$

subject to:

$$\sum_{k=1}^{K} \beta^0_{nk} \alpha_{nk} \leq t \;\forall\; n \tag{26}$$

and assignment constraints above (*i*). This is a mixed-integer-linear program (MILP) and can be solved either with relaxation methods as a LP, or with Lagrangian duality [13, Ch.10] or other numerical methods.

c. Joint problem. Define auxiliary variable t, and the bipartite graph with costs as in (20). Constraints (*i*)-(*iv*) of case 2 of objective I also need to be satisfied. This is a mixed-integer nonlinear program (MINLP).

B. Case 3: Number of BBU servers $K < N$

When virtualization results in a number of BBU servers $K < N$, we have the more general form of the problem, where (portions of) several streams can be served by the same server. BBU server capacity constraints may dictate that different portions of a stream are routed to different BBU servers. Let $\boldsymbol{\lambda}_n = (\lambda_{n1}, \ldots, \lambda_{nK})$ be the splitting (routing) policy for stream n, where λ_{nk} denotes the amount of stream n traffic routed to server k, for $k = 1, \ldots, K$. Let $\boldsymbol{\Lambda} = (\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_N)$ be the global stream-to-server routing policy. A functional split policy $\mathbf{x}$ needs also to be selected. Finally, a scheduling policy should be found for each server k to serve the portions of streams routed to that server. Pure priority schedules or convex combinations thereof can be applied. In the former case, let the priority scheduling policy for server k be $\boldsymbol{\pi}^k = (\pi_1^k, \ldots, \pi_N^k)$ where $\pi_n^k \in \{1, \ldots, N\}$ is the priority with which frames of the n-th stream are served at BBU server k. Let $\boldsymbol{\Pi} = (\boldsymbol{\pi}^1, \ldots, \boldsymbol{\pi}^K)$ be the overall scheduling policy at different servers. The total average delay for stream n at the BBU servers is

$$d_n^B(\boldsymbol{\Lambda}, \mathbf{x}, \boldsymbol{\Pi}) = \sum_{k=1}^{K} \frac{\lambda_{nk}}{\lambda_n} \Big(W_n^k(\boldsymbol{\Lambda}, \mathbf{x}, \boldsymbol{\pi}^k) + \frac{f(x_n)}{C_k} \Big)\,, \tag{27}$$

where $W_n^k(\boldsymbol{\Lambda}, \mathbf{x}, \boldsymbol{\pi}^k)$ is the average delay for stream n at server k. This depends on the priority with which stream n is served at server k, on the portions of other streams routed to server k that are served with higher priority than stream n in k, and on the functional splits of these streams.

For convex combinations of pure priority scheduling policies, i.e., policies in $\mathcal{P}^*$, we have

$$d_n^B(\boldsymbol{\Lambda}, \mathbf{x}, \mathbf{P}) = \sum_{k=1}^{K} \frac{\lambda_{nk}}{\lambda_n} \Big(W_n^k(\boldsymbol{\Lambda}, \mathbf{x}, \mathbf{p}^k) + \frac{f(x_n)}{C_k} \Big)\,, \tag{28}$$

with

$$W_n^k(\boldsymbol{\Lambda}, \mathbf{x}, \mathbf{p}^k) = \sum_{i=1}^{N!} p_i^k \, W_n^k(\boldsymbol{\Lambda}, \mathbf{x}, \boldsymbol{\pi}_i^k)\,, \tag{29}$$

where $\mathbf{p}^k = (p_1^k, \ldots, p_{N!}^k)$ is the convex combination of pure priority scheduling policies $\{\boldsymbol{\pi}_i^k\}$ at server k, $i = 1, \ldots, N!$, and $\mathbf{P} = (\mathbf{p}^1, \ldots, \mathbf{p}^K)$ is the ensemble of scheduling policies at servers. The *total average delay* for a frame of stream n is

$$D_n(\boldsymbol{\Lambda}, \mathbf{x}, \boldsymbol{\Pi}) = d_n^B(\boldsymbol{\Lambda}, \mathbf{x}, \boldsymbol{\Pi}) + d_n^F(x_n) + d_n^R(x_n)\,. \tag{30}$$

1) Objective I: Minimize total average delay: We solve:

$$\min_{\boldsymbol{\Lambda},\mathbf{x},\boldsymbol{\Pi}} \bar{D}_3^I(\boldsymbol{\Lambda}, \mathbf{x}, \boldsymbol{\Pi}) = \frac{1}{\lambda} \sum_{n=1}^{N} \lambda_n \Big(d_n^B(\boldsymbol{\Lambda}, \mathbf{x}, \boldsymbol{\Pi}) + d_n^F(x_n) + d_n^R(x_n) \Big) \tag{31}$$

where $d_n^B(\cdot)$ is given by (27), subject to constraints on: (*i*) traffic splits, i.e. $\sum_{k=1}^{K} \lambda_{nk} = \lambda_n$, for each stream n, (*ii*) queue stability: $\sum_{n=1}^{N} \lambda_{nk} f(x_n) < C_k$ for each server k, $\lambda_n f(x_n) < \sum_{k=1}^{K} C_k$ for each n and $\sum_{n=1}^{N} \lambda_n f(x_n) < \sum_{k=1}^{K} C_k$, (*iii*) constraint (7), and (*iv*) $0 \leq x_n \leq 1$ for each stream n.

a. Fixed functional split and traffic splitting policy. If functional split $\mathbf{x}^0$ and traffic splitting $\boldsymbol{\Lambda}^0$ are given, the scheduling policy that minimizes $\bar{D}_3^I(\boldsymbol{\Lambda}^0, \mathbf{x}^0, \boldsymbol{\Pi})$ is Shortest-Processing-Time-First (SPTF), applied to each server.

b. Fixed scheduling policy. When the scheduling policy $\boldsymbol{\Pi}^0$ is given, the problem $\min_{\boldsymbol{\Lambda},\mathbf{x}} \bar{D}_3^I(\boldsymbol{\Lambda}, \mathbf{x}, \boldsymbol{\Pi}^0)$ is a NLP.

c. Joint problem. If we allow schedules to be convex combinations of pure priority scheduling policies, the problem is a NLP. If we consider the class of pure priority scheduling policies $\mathcal{P}$, the problem is a MICP. We can have an iterative heuristic along the lines of that presented for $K = 1$, iterating among finding the best $\boldsymbol{\Lambda}, \mathbf{x}$ for given schedules at each server, and then fixing these policies $\boldsymbol{\Lambda}, \mathbf{x}$ and applying SPTF at each server.

2) Objective II: Minimize maximum average delay: The rationale is similar to that in case 1-objective II, by extending to multiple servers the methodology for $K = 1$.

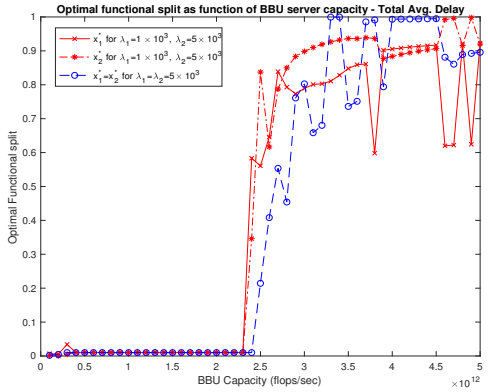

Fig. 3. Minimizing total average delay (Objective I): Optimal functional splits as function of BBU server capacity for different frame arrival rates λ_1, λ_2.

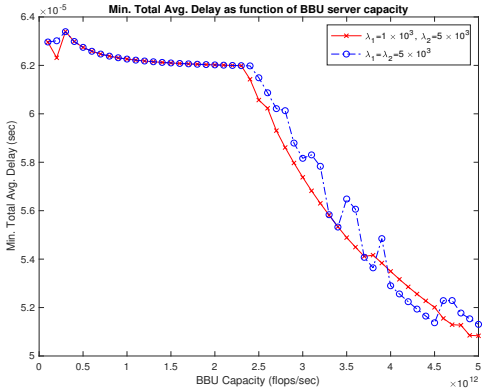

Fig. 5. Minimum total average delay as function of BBU server capacity for different frame arrival rates λ_1, λ_2.

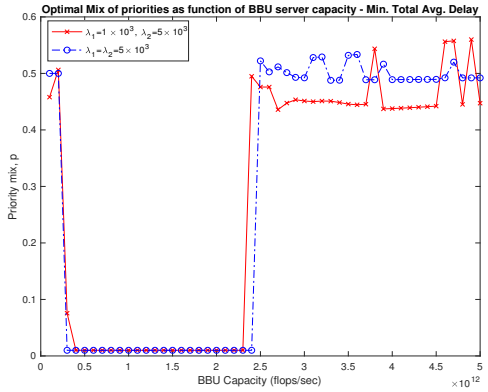

Fig. 4. Minimizing total average delay (Objective I): Optimal mix of priority schedules as function of BBU server capacity for different frame arrival rates λ_1, λ_2. A value p means that stream 1 is served with first priority for $p\%$ of the time, and with second priority for $(1-p)\%$ of the time.

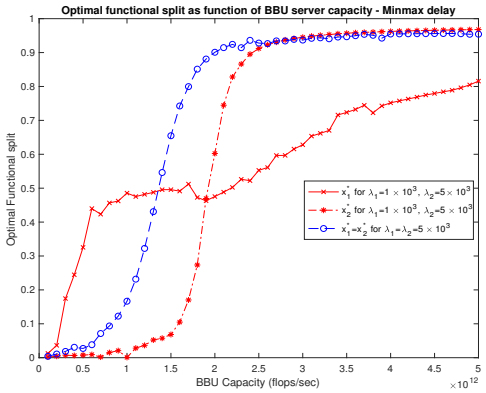

Fig. 6. Minimizing max average delay (Objective II): Optimal functional splits as function of BBU server capacity for different frame arrival rates λ_1, λ_2.

V. Numerical evaluation

In order to demonstrate the benefits of our optimization-based approach, we consider the following cases.

A. Case 1: K=1 BBU server, N=2 RRH streams

We consider a system with $K = 1$ BBU server of capacity C flops/sec and $N = 2$ RRHs with fronthaul links of capacity $u_i = 50$Gbps, $i = 1, 2$. Each RRH has a server of capacity $C^R = 8 \times 10^{11}$ flops/sec. This corresponds to a 2GHz 4-core Intel Xeon processor with 100 flops/cycle. Frame streams have arrival rates λ_i, in frames/sec. The workload is equal for RRHs, i.e.$w_1 = w_2 = 4.8 \times 10^7$ flops/frame. Function $f(x)$ that maps split x to the percentage of computation load at the BBU is linear i.e. $f(x) = wx$, with $x \in [0, 1]$. The function that maps split x to the amount of fronthaul data (bits/frame) is also linear and is equal to

$$r_i(x) = 10^6 \times \frac{0.1 + 1.9x}{u_i}, \; i = 1, 2\,. \tag{32}$$

For the order of magnitude, we have used Table 1 in reference [6]. The linear functions above are only indicative, and their exact forms can be determined through machine-learning techniques on measurement datasets.

1) Objective I: Minimum average delay: In Figures 3-5 we show the optimal functional split x_1^*, x_2^*, optimal schedule mixture p_1^*, p_2^*, and optimal value of the objective function in objective I, as functions of BBU server capacity C, for different arrival rates i.e., $\lambda_1 = \lambda_2 = 5 \times 10^3$ frames/sec, and $\lambda_1 = 10^3$ frames/sec, $\lambda_2 = 5\lambda_1$.

For BBU server capacity below a threshold $C^0 = 2.4 \times 10^{12}$, no operations are executed at the BBU server. For $C > C^0$, optimal functional splits mostly increase with C i.e. more computations from both streams are executed at BBU. Furthermore, the differential mostly decreases, i.e. $x_i^*(C)$ appears to be a concave function of C. This is more evident for higher values of C. Most of the times, the functional split is larger for the stream with larger arrival rate. For $\lambda_1 = \lambda_2$, the optimal functional splits for the two streams are the same. In terms of schedules, mixes of $0.45 - 0.55$ are seen most of the times, which implies that the two streams are served with almost the same priority in the long run, especially when they have equal arrival rates. The minimum average delay is at the range

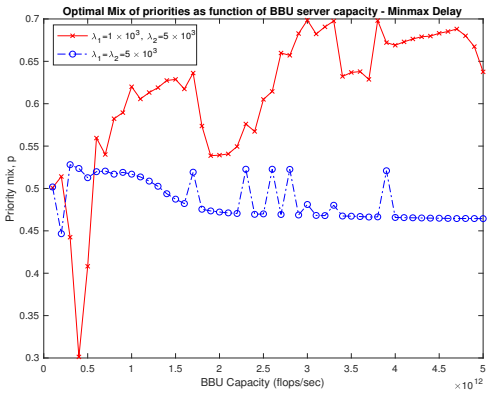

Fig. 7. Minimizing maximum average delay (Objective II): Optimal mixture of priority scheduling policies as function of BBU server capacity for different frame arrival rates λ_1, λ_2.

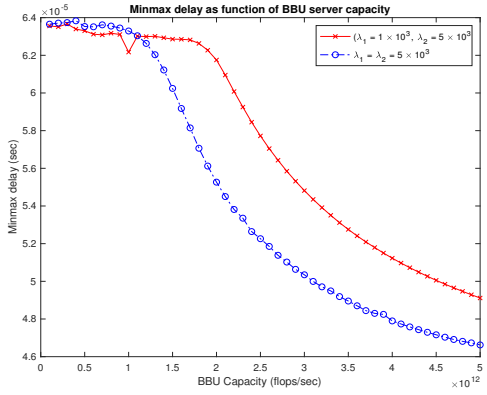

Fig. 8. Minmax average delay as function of BBU server capacity for different frame arrival rates λ_1, λ_2.

$54 - 65\mu$sec for the depicted range of values of C.

In Table I, we compare average delay for: (*i*) optimization over functional split and scheduling policies; (*ii*) all chain computations are performed at the BBU server, hence $x_1 = x_2 = 1$ as in typical C-RANs, and we optimize only over schedules. The proposed optimized functional split policy achieves $2 - 15$ times lower delays that those in C-RAN, and the benefit is larger for smaller BBU server capacity values.

2) Objective II: Minmax average delay: In Figures 6-8 we show the optimal functional splits, optimal schedules, and minmax delay for objective II, as functions of C. In Fig. 6, we observe increasing, and in most cases sigmoid-like forms for optimal functional splits $x_i^*(C)$. For $\lambda_1 = \lambda_2$, the optimal functional splits for the two streams are the same. In Fig. 7, we see less balanced priority mixtures than those in Fig. 4.

B. Case 2: K=20 BBU servers, N=20 RRH streams

In Table II we consider the case of $K = N = 20$ and objective I. RRH frame arrival rates are uniformly distributed in $(0, 10^4)$ frames/sec, and results are averaged over 100 experiments. We compare the delay of the following policies:

CAPACITY C	TOT. AVG. DELAY OPT. FUNCT. SPLITS	TOT. AVG. DELAY CLOUD-RAN
(in flops/sec)	(in μsec)	(in μsec)
1×10^{11}	62.95	953.38
5×10^{11}	62.74	363.94
1×10^{12}	62.26	227.3
2×10^{12}	62.02	136.1
3×10^{12}	57.37	115.18
4×10^{12}	53.49	105.69
5×10^{12}	50.83	100.25

TABLE I
MIN TOTAL AVERAGE DELAY VS. BBU SERVER CAPACITY FOR OPTIMAL FUNCTIONAL SPLITS AND FOR CLOUD-RAN ($\lambda_1 = 1 \times 10^3$, $\lambda_2 = 5 \times 10^3$).

CASE	$C \in [10^{10}, 10^{11}]$	$[10^{11}, 10^{12}]$	$[10^{12}, 10^{13}]$
CASE A	601.88	238.37	48.61
CASE B	610.85	261.0	53.12
CASE C	2,810	355.39	49.57
CASE D	9,300	5,400	53.39
CASE E	602	602	602

TABLE II
MIN TOTAL AVERAGE DELAY (IN μSEC) FOR DIFFERENT BBU SERVER CAPACITY RANGES FOR DIFFERENT POLICIES ($\lambda_i \in [0, 10^4]$ FRAMES/SEC).

- A: Optimization of functional splits and BBU server assignment, as in subsection IV.A.
- B: Optimization of functional splits, with random BBU server assignment.
- C: C-RAN, i.e., all operations executed at BBU servers ($x_i = 1, \forall i$), and optimal server assignment.
- D: C-RAN, with random BBU server assignment.
- E: all operations executed at RRHs ($x_i = 0, \forall i$).

The columns show different ranges of the K BBU server capacities: uniformly distributed in $[10^{10}, 10^{11}]$, $[10^{11}, 10^{12}]$ and $[10^{12}, 10^{13}]$ flops/sec. Significant delay reductions are seen for our optimized policy (policy A) over policy B which does random server assignment, and over the C-RAN policies C,D, but these diminish with increasing BBU server capacity.

VI. RELATED WORK

A. Fronthaul data transport protocols

The first chronologically proposed fronthaul transport protocols were the Common Public Radio Interface (CPRI) and the Open Base Station Architecture Initiative (OBSAI) [14], [15]. The former sends Constant-Bit-Rate (CBR) raw I/Q sample data over a dedicated channel, while the latter uses a packet-based mode to send I/Q samples, and both use Radio over Fiber (RoF). Transmission of raw samples leads to high fronthaul bandwidth. Next Generation Fronthaul Interface (NGFI) [16] was created with the mission to rethink fronthaul data transfer. NGFI supports Ethernet for fronthaul data transport and exploits statistical multiplexing and packet routing. Current standardization efforts with different possibilities for functional splits at DLC and PHY layers include the 3GPP [17] and the IEEE 1914 [18], where the latter supports NGFI.

B. Functional split selection and C-RAN resource allocation

C-RAN fronthaul requirements and converged fronthaul/backhaul architectures are discussed in [19]. Flexibility in functional splits reduces the required data rate between the BBU and RRHs [20]. With respect to PHY-layer splits, the work in [21] considers delay minimization for a static problem of functional split selection (out of a discrete set of choices) and packet sequencing, in which a given set of frames need to be transported to RRHs. In [6], the authors study the joint problem of functional split selection and routing to RRHs over different fronthaul topologies. The objective is to maximize the degree of centralization i.e. the number of functions executed at the BBU, subject to delay requirements and fronthaul link capacity constraints. In a similar thread, the work [22] studies the joint problem of selecting functional splits and BBU-RRH routing paths out of a set of three splits and a finite set of paths so as to minimize the network operational (i.e. computing and routing) cost.

In [23], the authors do a numerical study on the impact of functional split and traffic packetization on fronthaul delay. They consider RRH multiplexing and select the functional split and packetization mode so that a large number of RRHs are supported, subject to deadline constraints due to packet retransmission and fronthaul link capacity constraints. An adaptive functional split for the DLC layer is proposed in [24] based on optimizing the tradeoff between reaping coordination benefits of centralization and maintaining low fronthaul traffic.

The work [25] views the problem of functional split and function placement in the network as a graph clustering one. The baseband function chain is modeled as a directed acyclic graph. A node is a baseband function, and a link between two nodes denotes precedence. Node weights show computing costs for the function, while link weights model communication traffic between functions. The work [26] considers clustering of BSs and scheduling of computational tasks on different cores so as to maximize the number of BSs for which scheduling complies to deadline constraints. Functional split selection and placement are also studied in [27] through integer-linear programs so as to minimize inter-cell interference, power and fronthaul bandwidth consumption.

A broader perspective of computation chain placement on a physical network graph is *graph embedding*. The chain of functional operations is a directed acyclic graph in which each node is an operation, and links denote operation precedence. An embedding of the computation graph on the network graph is a mapping from operation nodes to network computation nodes. The embedding that minimizes delay and cost is NP-complete [28]. A dynamic-decision version of graph embedding through a queue backpressure-based policy is presented in [29] with the goal to maximize query computation rate.

C. Operations research and queueing theory

An array of works from operations research on job scheduling are related to this work, e.g., [30] and references therein. Choosing among functional splits is reminiscent of variable, controllable processing times of jobs [31]- [33]. A relevant thread from queueing theory is the multi-class $M/G/1$ queue, where each class is served according to a priority discipline [9, Chap.3.5.3]. The $c\mu$-rule specifies the priority assignment that minimizes total average delay $\sum_k c_k D_k$. Classes k are served in decreasing order of $c_k\mu_k$, where c_k is a cost per unit of delay and μ_k is the processing rate for class k. Namely, the highest-priority class is the one with highest $c_k\mu_k$. For $c_k = c$, the $c\mu$-rule reduces to Shortest-Processing-Time-First (SPTF).

VII. Conclusion

End-to-end latency minimization will play a central role in future wireless architectures, towards enabling advanced low-latency services. This work is a first step towards a principled approach to the problem of minimizing end-to-end delay, through functional split selection and BBU server resource allocation. The novel contributions are: (*i*) a model that abstracts functional split as a partition of the computation chain between the BBU and RRH locations, (*ii*) a queueing-theoretic approach to compute end-to-end delay, in which the computation job size and the amount of transported data depend on the functional split, and (*iii*) the formulation of the delay optimization problem that obtains different twists, depending on the number of BBU servers and RRH streams.

Several important extensions could be considered. The main objective of the work is to showcase the formulation of optimization problems in this setting and provide insights on the problem and associated policies. The implementation and performance benefits of such policies in real conditions warrant further investigation. The creation of datasets with employed functional split and values of attributes such as the amounts of computation and transported data, and possibly others would be an important next step towards the precise form of functions $f(\cdot)$ and $r(\cdot)$. Other performance metrics can be considered and included in the formulation, such as the amount of consumed energy due to BBU server computation and fronthaul data transmission. Functional splits could be extended to the MAC layer, by capturing in the model the benefits of joint coordination and optimization of multiple cells at the BBU location. CoMP transmission could also be incorporated. In that case, the average delay for each stream would need to be reconsidered.

In this work, we considered priority scheduling at the level of RRH flows so as to control delays of different RRH flows. An addition would be to also include priorities among user flows. Dynamic adaptation of the functional split can be considered in response to dynamics such as background processes that alter the effective computing capacity of BBU and RRH servers. Various models can be defined to address this problem that also account for possible switching costs between functional splits. Finally, we have assumed fixed job sizes, which led to a tandem of a multi-class $M/D/1$ queue at the BBU, and a $D/D/1$ queue at the RRH, for which the total delay was the sum of delays in the two queues. If real data suggest a random job size distribution, then the first queue in the tandem would be a $M/G/1$ one, and the queues in tandem would be interesting to analyze in terms of total delay.

REFERENCES

[1] L. M. P. Larsen, A. Checko, and H. L. Christiansen, "A Survey of the Functional Splits Proposed for 5G Mobile Crosshaul Networks", *IEEE Comm. Surveys and Tut.,* vol.21, no.1, pp.146-172, First Quarter 2019.

[2] D.P. Bertsekas and J.N. Tsitsiklis, Introduction to probability. Belmont, Athena Scientific, 2008.

[3] P. Rost, S. Talarico and M. C. Valenti, "The Complexity–Rate Tradeoff of Centralized Radio Access Networks", *IEEE Trans. Wireless Commun.,* vol. 14, no. 11, pp. 6164-6176, Nov. 2015.

[4] N. Nikaein, "Processing Radio Access Network Functions in the Cloud: Critical Issues and Modeling", *in Proc. Int. Workshop on Mobile Cloud Comput. and Services,* 2015.

[5] J. Bartelt, N. Vucic, D. Camps-Mur, E. Garcia-Villegas, I. Demirkol, A. Fehske, M. Grieger, A. Tzanakaki, J. Gutierrez, E. Grass, G. Lyberopoulos, and G. Fettweis, "5G transport network requirements for the next generation fronthaul interface", *EURASIP J. Wireless Commun. and Networking,* 2017:89, May 2017.

[6] A. Garcia-Saavedra, J. X. Salvat, X. Li and X. Costa-Perez, "WizHaul: On the Centralization Degree of Cloud RAN Next Generation Fronthaul," *IEEE Trans. on Mobile Comput.,* vol. 17, no. 10, pp. 2452-2466, Oct. 2018.

[7] S.C. Borst, L.C.M. Kallenberg, and G.M. Koole, "Stochastic Dynamic Programming and the control of Queues". Lecture Notes http://www.math.leidenuniv.nl/∼kallenberg/SDP.html.

[8] A. Federgruen, and H. Groenevelt, "Characterization and optimization of achievable performance in queueing systems", *Oper. Research,* vol.36, pp.733-741, 1988.

[9] D.P. Bertsekas and R.G. Gallager, Data Networks, Prentice Hall, 1992.

[10] D. P. Bertsekas, Nonlinear Programming, Athena Scientific, 2nd Ed., 1999.

[11] M. Lubin, "Mixed-integer convex optimization: outer approximation algorithms and modeling power", Ph.D. Thesis, MIT, 2017.

[12] H.W. Kuhn, "The Hungarian Method for the Assignment Problem", *Naval Research Logistics Quarterly 2,* pp.83-97, 1955.

[13] L.A. Wolsey, Integer Programming, *Wiley*, 1998.

[14] A. de la Oliva, J.A. Hernandez, D. Larrabeite and A. Azcorra, "An overview of the CPRI specification and its application to C-RAN based LTE scenarios", *IEEE Comm. Mag.* vol.54, no.2, pp.152-159, Feb. 2016.

[15] OBSAI, http://www.obsai.com/.

[16] China Mobile Research, Alcatel-Lucent, Nokia Networks, ZTE Corp., Broadcom Corp., Intel China, "White paper of Next Generation Fronthaul Interface", v1.0, June 2015.

[17] "Study on new radio access technology: Radio access architectureand interfaces, V14.0.0 (2017–03)," 3GPP, Sophia Antipolis, France,Rep. TR 38.801, 2017.

[18] IEEE 1914 Website. [Online]. http://sites.ieee.org/sagroups-1914.

[19] J. Bartelt, P. Rost, D. Wuebben, J. Lessmann, B. Melis and G. Fettweis, "Fronthaul and Backhaul requiremements of flexibly centralized radio access networks", *IEEE Wireless Comm. Mag.,* vol.22, no.5, pp.105-111, Oct. 2015.

[20] J. Duan, X. Lagrange and F. Guilloud, "Performance Analysis of Several Functional Splits in C-RAN", *in Proc. Veh. Tech. Conf.,* 2016, Spring.

[21] I. Koutsopoulos, "Optimal functional split selection and scheduling policies in 5G Radio Access Networks", *in Proc. Workshop on 5G Archit. (5GArch), in IEEE ICC,* 2017.

[22] A. Garcia-Saavedra, X. Costa-Perez, D.J. Leith and G. Iosifidis, "FluidRAN: Optimized vRAN/MEC Orchestration", *in Proc. IEEE INFOCOM,* 2018.

[23] C.-Y. Chang, R. Schiavi, N. Nikaein, T. Spyropoulos, and C. Bonnet, "Impact of Packetization and Functional Split on C-RAN Fronthaul Performance", *in Proc. IEEE ICC,* 2016.

[24] A.M. Alba, J.H.G. Velasquez and W. Kellerer, "An adaptive functional split in 5G networks", *in Proc. Int. Workshop on Flexible and Agile Networks: 5G and Beyond (FlexNets), in IEEE INFOCOM,* 2019.

[25] J. Liu, S. Zhou, J. Gong, Z. Niu and S. Xu, "Graph-based Framework for Flexible Baseband Function Splitting and Placement in C-RAN", *in Proc. ICC*, 2015.

[26] S. Bhaumik, S. P. Chandrabose, M. K. Jataprolu, G. Kumar, A. Muralidhar, P. Polakos, V. Srinivasan, and T. Woo, "CloudIQ: A Framework for Processing Base Stations in a Data Center", *in Proc. ACM Mobicom,* 2012.

[27] A. Alabbasi , X. Wang, and C. Cavdar, "Optimal Processing Allocation to Minimize Energy and Bandwidth Consumption in Hybrid CRAN", *IEEE Trans. on Green Commun. Networking,* vol.2, no.2, pp.545-555, June 2018.

[28] P. Vyavahare, N. Limaye, and D. Manjunath, "Optimal embedding of functions for in-network computation: Complexity analysis and algorithms," *IEEE/ACM Trans. on Networking,* vol.24, no.4, pp.2019 - 203, Aug. 2016.

[29] A. Destounis, G. S. Paschos and I. Koutsopoulos, "Streaming Big Data meets Backpressure in Distributed Network Computation", *in Proc. IEEE INFOCOM,* 2016.

[30] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnoy Kan and D.B. Shmoys, "Ch 9. Sequencing and scheduling: Algorithms and complexity", *in Handbook of Oper. Res. and Manag. Sci., Elsevier* vol.4, pp.445-522, 1993.

[31] D. Shabtay and G. Steiner, "A survey of scheduling with controllable processing times", *Discrete Appl. Math.,* vol. 155, pp.1643-1666, 2007.

[32] Z.-L. Chen, Q. Lu, G. Tang, "Single machine scheduling with discretely controllable processing times", *Oper. Res. Let.,* vol.21, pp.69-76, 1997.

[33] E. Nowicki and S. Zdrzalka, "A survey of results for sequencing problems with controllable processing times", *Disc. Appl. Math.,* v.26, pp.271-287, 1990.