REINDEER

D2.2

Evaluation of the distribution of processing across infrastructure and associated requirements on back-haul and synchronization

Project number:	101013425
Project acronym:	**REINDEER**
Project title:	REsilient INteractive applications through hyper Diversity in Energy Efficient RadioWeaves technology
Project Start Date:	1st January, 2021
Duration:	42 months
Programme:	H2020-ICT-52-2020
Deliverable Type:	Report
Reference Number:	ICT-52-2020 / D2.2 / 0.01
Reference Number:	ICT-52-2020 / D2.2 / 0.01
Workpackage:	WP 02
Due Date:	March 31, 2023 (M27)
Actual Submission Date:	April 5, 2023
Responsible Organisation:	Linköping University
Editor:	Sai Subramanyam Thoota (LiU)
Dissemination Level:	PU
Revision:	0.01
Abstract:	This deliverable provides estimates of distributed hardware resources, and their synchronization, needed to implement targeted algorithms for selected services and the associated requirements on back-haul data transfer between processing/hardware units.
Keywords:	Synchronization, Distributed processing, Over-the-air, back-haul

The REINDEER project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101013425.

Editor

Sai Subramanyam Thoota

Contributors

Liang Liu, Ove Edfors (ULUND)
Pål Frenger, Ke Wang Helmersson (EAB)
Liesbet Van der Perre, Gilles Callebaut, Vida Ranjbar (KU Leuven)

Disclaimer

Executive Summary

This REINDEER deliverable reports on the several studies conducted for the task T2.2 which deals with a detailed evaluation of the distributed processing across infrastructure and the associated requirements on back-haul and synchronization. It also focuses on identifying the processing elements to execute the various tasks to provide the required services using the Radioweaves infrastructure. It evaluates several distributed processing algorithms for both uplink and downlink in a radioweave system, and studies on the trade-offs between computational complexity and performance. A scalable algorithm is proposed to easily decide on addressing the trade-off issues. Furthermore, the performance is assessed when the fronthaul capacity is limited, and proposes algorithms for uplink processing. Memory distribution and the associated data exchange between clusters of service points is also studied in detail.
The overall outcome of this deliverable is a comprehensive study and evaluation of the distributed processing algorithms in a RadioWeaves infrastructure and the associated requirements on the fronthaul. It also addresses several architectures to synchronize the multiple service points in a detailed manner.

Contents

List of Figures

List of Tables

List of Abbreviations

ADC analog-to-digital converter. 9

CDF cumulative distribution function. 19, 29

CLK clock. 6, 9

CPU central processing unit. 42

CSP contact service point. 3–17, 19–21, 24–29, 31, 32, 34–49, III, V, VI

DRAM dynamic random-access memory. 45

ECSP edge computing service point. 4, 6, 12–16, 20, 24, 26–28, 42–45, 48, V, VI

EN energy neutral. 7

FD Front-haul distance. 40–42, 47, VI

ISI intersymbol interference. 7

LO local oscillator. 8–11, 31

LS least squares. 43

MIMO multiple-input multiple-output. 49

MMSE minimum mean square error. 3, 6, 12, 14, 15, 18–20, 24, 25, 27, 29–31, V

MRC maximum ratio combining. 18, 19, V

NTP Network Time Protocol. 11

OFDM orthogonal frequency-division multiplexing. 11, 44–46

OTA over-the-air. 11, 31

PLL phase-locked loop. 8–10, 31, V

PPS pulse per second. 8–11, 31, V

PTP Precision Time Protocol. 11

QR-RLS QR decomposition based recursive least squares. 34–38, 49, III

RF radio frequency. 8–11

RLS recursive least squares. 36, 43, 45–47, VI

RW RadioWeaves. 3–6, 27, 31, 40–49, III, V

SE spectral efficiency. 6, 12, 15, 17, 19, 24, 26, 29

SINR signal-to-interference-plus-noise ratio. 6, 15, 20, 21, 26

SRAM static random-access memory. 45

SRLS standard recursive least squares. 34–39, 49, III, V

TDOA time-difference-of-arrival. 7

TOA time-of-arrival. 7

UE user equipment. 4, 5, 7, 10, 13–17, 19–21, 24–26, 29, 31–34, 36–38, 48, 49, V

WD Wireless distance. 41, 42, 47, VI

WPT wireless power transfer. 7

WR White Rabbit. 11

ZF zero forcing. 42, 44, 45

Chapter 1

Introduction

Algorithms for RadioWeaves (RW) data processing must always consider the associated hardware impact. The key to success is to find a good balance between implementation complexity and performance. It is very easy to over-engineer an algorithm resulting in a highly expensive product. Alternatively, focusing only on what is easy to implement at low cost can result in lackluster performance. By jointly designing algorithms and hardware where close attention is paid to performance versus algorithmic complexity trade-offs as well as consequences on the hardware components, we can design solutions that meet the highest expectations.

This document reports on the results of the research evaluating the distribution of processing across infrastructure and associated requirements on back-haul and synchronization. Estimates of distributed hardware resources, and their synchronization, needed to implement targeted algorithms are provided for selected services and the associated requirements on back-haul data transfer between processing/hardware units.

In Chapter 2, high-performing and cost-efficient distributed processing algorithms suitable for uplink processing in a RadioWeaves (RW) system are evaluated. It also discusses the synchronization of distributed hardware resources. We show that the proposed decentralized subset method is a scalable solution providing an easily controlled trade-off between computational complexity and performance. By describing the distributed processing as a Kalman filter operation, we show that the proposed algorithm provides an optimum solution in the minimum mean square error (MMSE) sense. The decentralized Kalman filter solution makes it simple to decide the most cost efficient trade-off between performance and implementation cost.

In Chapter 3, the impact of limited front-haul links between service points is further examined. A low-precision implementation of two recursive algorithms to solve the regularized least-squares (LS) problem for uplink signal estimation in a RW network is considered. The performance impact and the front-haul requirement relaxation when communication on the front-haul links between contact service points (contact service points (CSPs)) are quantized, is studied. It is shown that the proposed algorithm is numerically stable under all simulated scenarios.

Memory distribution and exchange of data between processing clusters are studied in Chapter 4. The required memory for storing channel matrices and buffering radio frames is discussed, as well as the required bandwidth for data exchange between processing clusters. Requirements depend on, e.g., the selected processing algorithms, the mapping of the algorithms to the hardware architecture, the number of users served, the structure and format of the radio frames, etc. The data exchange requirements for federation orchestration, user association, and resource

allocation are also discussed.

1.1 Key terminology for RW architectures

We here include the overall architectural terminology as adapted in the REINDEER project to enhance readability of this deliverable as a stand-alone document. Fig. 1.1 shows an illustrative architecture demonstrating the distributed CSPs that host radio equipment with an antenna or antenna array establishing the actual link(s) to the user equipments (UEs), as well as the edge computing service point (ECSP) that provides substantial compute power.

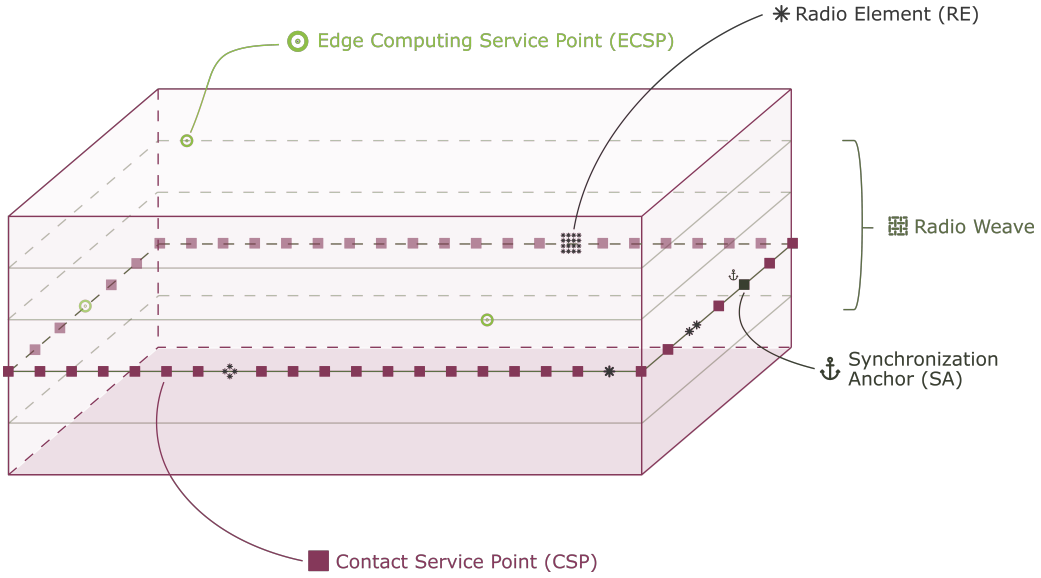


Figure 1.1: Overview of a RW architecture with its main components. The RW system consists of at least one, but preferably many ECSPs. This component is responsible for data aggregation and coordination of CSPs. A CSP is the first contact point from the UE perspective and provides the necessary services to support user applications. A CSP can be equipped with one or several radio elements. To allow for a synchronized/coherent system, ECSP can act as a synchronization anchor to synchronize its CSPs. More details can be found in Table 1.1.

Table 1.1: Short description of the components.

Term	Abbreviation	Description
RadioWeave	RW	Wireless access infrastructure consisting of a fabric of distributed radio, computing, and storage resources.
Radio Element	RE	Transmit/receive units, most often including an antenna, that can serve to exchange data or charge devices using electromagnetic waves.
Contact Service Point	CSP	Integrates local computation and storage resources, and provides at least communication, sensing or charging functionality. It is the first contact point as seen from the UE and takes the role of an anchor in the context of position related applications.
Edge Computing Service Point	ECSP	Shared compute resources integrated in the RW that can support applications in need of substantial compute power and/or connection to the back-haul or other RW infrastructures.
(Dynamic Service) Federation	DSF	(Temporary) set of cooperating resources in the RW, working in unison, that could be more or less synchronized, and including at least CSPs and typically a synchronization anchor, and potentially edge processing unit(s), established to serve a cluster of devices and/or application(s).
Synchronization Anchor	SA	Logical function flexibly located attributed to a certain CSP to serve as a synchronization reference for a set of cooperating CSPs for some period.

Chapter 2

Synchronization of distributed hardware resources and distributed processing algorithms

In this chapter, we present performance evaluations of some of the cost efficient distributed algorithms for processing uplink and downlink data in a RadioWeaves (RW) system. This chapter is organised as follows:

In Section 2.1, we discuss different kinds of clock (CLK) sources and architectures which can be used to synchronize the contact service points (CSPs) in a distributed wireless communication system. In Section 2.2, we analyse a decentralised processing and combining method for uplink for different federation sizes. A subset of CSPs that are connected to the same edge computing service point (ECSP) is defined for each UE. A subset that consists of only one CSP turns the RW network into "small cell" network; a subset that consists of all CSPs connected to the ECSP give the same performance as a fully centralised method. This shows that the subset method provides a very scalable trade-off between complexity and performance. It is simpler to implement as the processing can be distributed and parallelised. In the studied simulation scenarios, we show that it is possible to reach 85% of the performance upper bound by including only 20% of total CSPs in the subset, and to reach 95% of the performance upper bound by including 40% of CSPs in the subset.

In Section 2.3, we extend the subset combining method by Kalman filtering to estimate the received uplink signals. We analyse the uplink performance as well as the computational complexity with different combining methods. We show that the Kalman filter implementation provides the same result as the minimum mean square error (MMSE) method in terms of the spectral efficiency (SE) and equivalent signal-to-interference-plus-noise ratio (SINR). However, the Kalman filter implementation is shown to be very efficient as it provides the possibility to fully utilise parallel computing of distributed hardware processors. Moreover, the processing can be decentralised and the estimates can be aggregated from local estimates to as many CSPs as needed to reach the desired performance target. A Kalman filter implementation has the flexibility to aggregate signals in different ways, allowing the front-haul architecture to support connectivity of individual CSPs in any combination of parallel or serial manners.

In the Reindeer project, synchronization between the CSPs is a key requirement to support the functionalities of multiple applications. The physically distributed systems make it challenging to

provide coherent operation. There is a need for frequency and phase synchronization to avoid interference at the user equipment (UE) (or energy neutral (EN) device for wireless power transfer (WPT)) locations and time synchronization to avoid intersymbol interference (ISI). In distributed systems, a frequency synchronization mismatch will always occur between two CSPs called the **frequency offset** or carrier frequency offset and a **phase offset**. The RF front-end of each transceiver causes the phase offset. An offset within certain margin is permissible. Therefore, it is important to understand the offset differences between multiple CSPs.

A summary of mismatches between two signals is described by six parameters. Here the phase drift and time drift are related to the frequency offset.

- Frequency offset (over time, partly depending on frequency drift) and frequency drift (independent)
- Phase offset (independent) and phase drift (depending on frequency offset)
- Time offset (independent) and time drift (depending on frequency offset)

Both the architecture, software and selected hardware are crucial to grasp the occurring mismatches.

Previous deliverable D2.1 already discussed the functionalities that require synchronized CSPs.

- Localizing UEs via time-of-arrival (TOA) and time-difference-of-arrival (TDOA) require accurate time synchronization (discussed in D2.1 section 6.2.1.2).
- Wireless powering EN devices require frequency and time synchronization (discussed in D2.1 section 6.3.3 Option 2).

This chapter mainly looks at the different synchronization architectures between multiple CSPs. Hardware blocks for enabling WPT are discussed in D2.3.

2.1 Clock Sources

2.1.1 One physically shared clock

2.1.1.1 Shared carrier frequency

One shared clock is distributed among all the CSPs presented in Figure 2.1. The clock foresees the carrier frequency for every radio frequency (RF)-chain of each CSP. A phase shifter can adjust the carrier frequency phase and steer the signal in the desired direction. This shared carrier frequency is directly connected to the mixer resulting in an equal carrier frequency of all transmitted signals. A central controller is connected to all RF chains/front-ends and is additionally capable to change phase-locked loop (PLL) output frequency. Table 2.1 describes this approach more in detail.

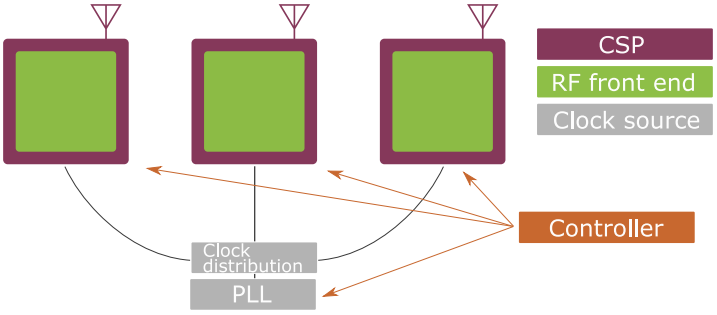


Figure 2.1: One shared oscillator connected to all CSPs.

Synchronisation description	
Mechanism	One carrier frequency, generated by the central PLL, is distributed to each CSP using, e.g., coaxial cables.
Synchronisation mismatches	
Frequency offset	Not present, there is only one PLL generating the shared signal.
Frequency drift	Cannot occur between CSPs.
Phase offset	Possible. Will not occur if all cable lengths are equal.
Time offset	Potentially, if communication between controller and RF front-end cannot occur synchronous. Proposed solution: An additional cable (e.g. PPS-cable or a pulse generated by the controller) could be added to synchronize RF front-end commands. For power transfer with single tone signals, time offsets are tolerable, especially as long as the signals are transmitted with desired phase offset relative to the reference LO.
Architectural strengths and weaknesses	
Scalability	Challenging and expensive. On the one hand, a highly accurate clock distribution device should have as many output ports as there are CSPs. Cable lengths must be perfectly equal or must exactly match a number of times the wavelength. The latter is only possible if the operating frequency is constant and thus the wavelength is fixed. The bandwidth of the cables should be high enough to prevent attenuation and distortion of the signals.
CSP complexity	Despite its low scalability, the CSP architecture is limited to a mixer, power amplifier and antenna. A baseband or phase offset signal is sent to each CSP through the controller.

Table 2.1: Advantages and shortcomings of the shared clock architecture.

2.1.1.2 Shared low frequency clock

Two cables from a central distribution rack are connected to every CSP to ensure both a low frequency clock and PPS signal. The low frequency clock serves as a stable reference clock for the PLLs in each CSP. The PPS can be engaged for phase offset synchronisation to ensure each PLL achieve relative phase coherence. PLL phase synchronisation is discussed in D2.3 [1]. The RF-front can be similar to Section 2.1.1.1 consisting of a mixer, a low pass filter and a power amplifier. Again, the controller can send the corresponding analog baseband signals to all RF chains. Contrary, digital data can be forwarded to the CSPs and converted internally to the analog baseband signal, therefore an additional analog-to-digital converter (ADC) is required. Figure 2.2 represents the architecture and Table 2.2 describes this approach more in detail.

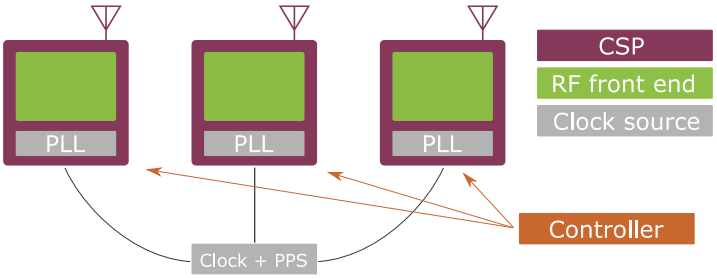


Figure 2.2: Architecture with distributed low frequency clock as reference signal for the integrated PLL and PPS signal for phase synchronisation.

Synchronisation description	
Mechanism	Every CSP is equipped with a PLL to generate the LO signals. Each individual PLL is supplied with one shared stable reference signal. Additionally, a distributed PPS signal can synchronize the generated PLL outputs to achieve relative coherence between all LOs.
Synchronisation mismatches	
Frequency offset	Not present, if all PLL registers are set equally.
Frequency drift	Cannot occur due to shared reference clock.
Phase offset	Possible. There will be phase coherent LOs outputs, if some accurate synchronisation signal is available. This could be a PPS signal. Therefore, equal PPS cable lengths are required and the PLLs architecture should be equipped with an additional synchronisation input. See also [1].
Time offset	Can be solved also with the PPS input. The commands for the RF front-end, generated by the controller, can be triggered synchronously on the PPS signals.
Architectural strengths and weaknesses	
Scalability	Two cables must be distributed towards each CSP. Again, a highly accurate distribution device for both the CLK and PPS signal should have as many output ports as there are CSPs. The cable bandwidth can be rather low, since the reference clock will be only several MHz. Moreover, the frequency of the PPS signal is only 1 Hz. Additionally, the controller should be connected to each CSP to control all devices of the RF front-end.
CSP complexity	The CSP hardware is slightly more complex since here the PLL is also built-in compared to Section 2.1.1.1. A PPS, reference clock controller input is needed, although the cabling can be more low cost due to the lower required bandwidth.

Table 2.2: Advantages and shortcomings of the distributed low frequency clock architecture.

2.1.1.3 Shared low frequency clock controlled over network

Similarly as discussed in Section 2.1.1.2, a shared reference clock and a PPS signal are connected to all CSPs. The main difference is that each CSP is equipped with an edge controller. Communication between multiple CSPs can occur through a network. The architectures introduced above need more control lines between central controller and all CSPs. This somewhat more complex CSP architecture, represented in Figure 2.3, comes with some advantages, explained in Table 2.3.

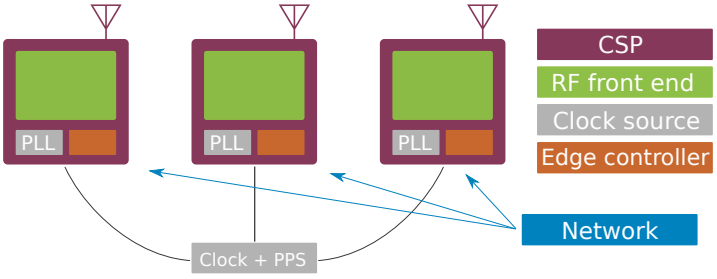


Figure 2.3: Representation of distributed CSPs, each having their own edge computing circuitry and network connection.

Synchronisation description	
Mechanism	Each PLL generates a stable LO and is connected to the shared clock that provides a stable frequency. A PPS signal can again ensure relative phase coherent LOs between CSPs. A third cable, such as an Ethernet cable, could provide communication between the different CSPs.
Synchronisation mismatches	
Frequency offset	Equivalent as Section 2.1.1.2. If all PLL registers are set equal, frequency offset will not occur.
Frequency drift	Cannot occur due to shared reference clock.
Phase offset	Possible. Similar solution possible as suggested in Section 2.1.1.2. Also network synchronization protocols can be enabled making the PPS obsolete. Still, the PPS input will most likely provide the best accuracy.
Time offset	Can be solved with the PPS input or by network synchronisation protocols. The commands and instructions, generated by the controller, to configure the RF front-end and start transmission to the UEs, can be triggered synchronously by the PPS signal.
Architectural strengths and weaknesses	
Scalability	Remains similar to Section 2.1.1.2 with the advantage that only one cable for communication is required. The reference clock and PPS signals are still required, assuming that the network could not ensure sufficient synchronisation accuracy.
CSP complexity	This is not limited anymore to just RF front, antenna and PLL, but also a peripheral computeris required, resulting in a higher cost and more complex hardware implementation of every CSP.

Table 2.3: Advantages and shortcomings of the shared low frequency clock controlled over network architecture.

2.1.2 CSPs with their own clock source

Previous architectures were based on a shared clock. As a result, they did not suffer from frequency drift, nor the associated time or phase drift. Here we consider architectures where CSPs operate based on their own clock. Dedicated solutions are then required to achieve synchro-

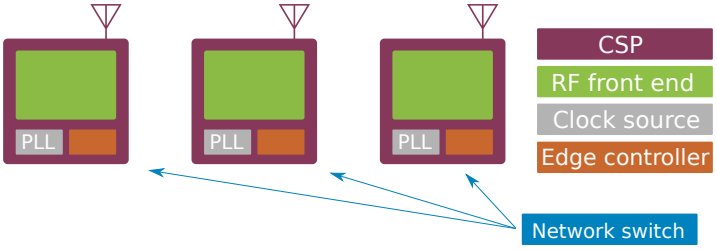


Figure 2.4: Highly flexible architecture with customized hardware for Ethernet synchronization.

nization over the CSPs, for which we discuss the options via Ethernet connectivity or via an over-the-air (OTA) approach.

2.1.2.1 Synchronisation over the network: Ethernet synchronization

Several Ethernet protocols such as Network Time Protocol (NTP), Precision Time Protocol (PTP) and White Rabbit (WR) could provide the required synchronisation between all edge devices. The latter is the most complicated and most complex solution to implement on a large scale, yet sub-nanosecond accuracy and picoseconds precision of synchronization is achievable [2]. Consequently, the PPS and reference clock can be removed, making the architecture more scalable, contrary also making the network switches more expensive, because PTP and WR require specific hardware. In addition, the CSP hardware should also be modified to be compatible with the corresponding protocols and achieve the high synchronisation accuracy. Accordingly, the very flexible architecture comes with more complex CSP hardware. Moreover, the Ethernet protocols should be supported by all network devices and switches. Figure 2.4 represents the architecture with only network cables between the network switch and CSPs.

2.1.2.2 Synchronisation over the network: OTA synchronization

In a wireless transmission system, there are two important clock sources namely the sampling clock and the RF carrier clock. Due to mismatches in the LOs at the distributed nodes, the generated frequency and phase of the clocks will slightly differ at different nodes. The drift and jitter in the sampling clock are quite insignificant. The timing mismatches in the sampling clock can be addressed by aligning the signaling within the length of the cyclic prefix of an orthogonal frequency-division multiplexing (OFDM) symbol. However, the carrier clock is affected by the drift and jitter and needs to be managed in the distributed networks so as the system does not perform poorly. To clarify this, let us consider two distributed nodes transmitting a real signal $\bar{x}(t)$ after modulating it to the required carrier frequency. The narrowband signals transmitted from two nodes can be written as

$$x_1(t) = \bar{x}(t)\cos\left(2\pi f_1 t + \alpha_1 + w_1(t)\right), \tag{2.1}$$
$$x_2(t) = \bar{x}(t)\cos\left(2\pi f_2 t + \alpha_2 + w_2(t)\right), \tag{2.2}$$

where f_1 and f_2 are the carrier frequencies generated from the corresponding LOs at the two nodes, α_1 and α_2 are the unknown constant phase shifts at the two nodes, and $w_1(t)$ and $w_2(t)$ are zero mean stationary phase noise process. The statistics of the phase noise process depends on the LO hardware implementation. The superposition of the signals transmitted from both the

nodes is given by

$$\begin{aligned} y(t) &= x_1(t) + x_2(t) \\ &= \bar{x}(t)\cos\left(2\pi f_1 t + \alpha_1 + w_1(t)\right) + \bar{x}(t)\cos\left(2\pi f_2 t + \alpha_2 + w_2(t)\right) \\ &= 2\bar{x}(t)\cos\left(2\pi\frac{f_1 - f_2}{2}t + \frac{\alpha_1 - \alpha_2}{2} + \frac{w_1(t) - w_2(t)}{2}\right) \\ &\quad \times \cos\left(2\pi\left(f_1 - \frac{f_1 - f_2}{2}\right)t + \frac{\alpha_1 + \alpha_2}{2} + \frac{w_1(t) + w_2(t)}{2}\right). \end{aligned} \tag{2.3}$$

From (2.3), it is clear that the composed signal can undergo destructive interference when the carrier frequencies f_1 and f_2 as well as the the phases α_1 and α_2 are not synchronized. Thus, in a distributed transmitting system it is very critical to estimate and synchronize the frequency and phase among the different transmitting nodes, otherwise the communication system will perform poorly.

In the presence of unknown multiplicative gains from the RF hardware, the CPU back-haul network is unable to provide accurate timing and phase information to the CSPs. Thus, it becomes harder to synchronize the distributed CSPs through a wired back-haul network. Over-the-air (OTA) synchronization methods are advocated to overcome this issue and were studied in [3–11]. Coherent beamforming from distributed transmitters is studied in [4–7, 12] and they are based on master-slave protocol. In the AirSync method proposed in [8], a master CSP transmits out-of-band pilots continuously to all the slave CSPs. The slave CSPs track the incoming signals phase offset and compensates it during the data transmission. The AirShare technique proposed in [10] uses a dedicated emitter circuit to transmit two low-frequency tones over-the-air and the distributed CSPs generate their reference signal with the frequency equal to the difference of the two tones. Airshare is robust to external variations like temperature and supply voltage at the emitter. However, it does not compensate for phase impairments from the hardware. REINDEER partners in [13] propose a beam-sweep approach where the calibration data between all pairs of the participating CSPs are collected and are sent to the CSP.

An OTA frequency synchronization protocol named ”Beamsync” which is based on beamforming the frequency synchronization signal among the CSPs is developed in REINDEER and included in the second chapter of the deliverable D3.2 [12, 14].

2.2 Uplink D-MIMO with Decentralised Subset Combining

In this section we evaluate different combining methods in the distributed MIMO (D-MIMO) in the uplink. In [15] and [16], four different uplink combining methods were proposed, from a fully centralised method to locally distributed methods. The fully centralised method, which is called level-4 implementation, uses global MMSE combining, hence providing the best performance in terms of SE, or equivalently SINR. Fully centralised level-4 combining gives an upper bound, however it is difficult to realise in practise, both the front-haul capacity requirement and the high computational complexity at the ECSP is staggeringly high. To reduce the computational complexity at the ECSP, another three methods, called level 1-3: process the signals at each CSP based on local information first, and then pass the processed signals to the ECSP for final combining and decoding. Level 1-3 still require centralised combining at ECSP by collecting locally estimated signals from all CSPs that are connected to a ECSP. Level 1-3 reduce the computational complexity and traffic load at front-haul but performance is far from level 4.

To find the possibility to approach the upper bound performance provided by level 4, we propose to use a decentralised method by defining a subset of serving CSPs for each UE. The subset can be defined from one serving CSP to all CSPs that are connected to a ECSP. When the subset consists of only one serving CSP, it turns D-MIMO network into a "small cell" network; subset can be aggregated by adding more CSPs, from smaller to larger, they are denoted as sub-1, sub-2 depending on the number of CSPs in the subset. If there are L CSPs that connected to a ECSP, sub-L reaches the same performance as the level 4. However, the performance upper bound reached by sub-L uses decentralised processing and aggregating. From the signal processing view in this exploration we are using the term subsets. In further consideration of management complexity in multi-user scenarios, these may be mapped to the approach of dynamic federations.

The evaluation shows that the proposed subset method provides a scalable trade-off between complexity and performance. The subset method is simpler to realise as the processing can be decentralised and parallelised. Moreover, the subset method is very flexible to aggregate. We can aggregate gradually from local CSPs to reach the desired performance target.

2.2.1 System model

The uplink data processing and combining in a D-MIMO network is shown in Fig. 2.5. The D-MIMO network consists of L geographically distributed CSPs, each equipped with N antenna elements. The total number of antennas in the network is $N \times L$. The CSPs are connected via front-haul links to ECSPs, which facilitate the coordination among CSPs. Note that front-haul links connecting the CSPs with the ECSP can be serial such as "radio stripes" [17] or other topologies (as in Fig. 2.7), as well as wired or wireless. The CSPs are cooperating to serve K UEs in the coverage area jointly by phase coherent transmission in the downlink and phase coherent reception in the uplink.

In Fig. 2.5 we introduce some of the notation that will be used throughout this section. We start with UE$_k$, as shown in Fig. 2.5 (a), the uplink channel between SP$_l$ and UE$_k$ is denoted as $\mathbf{h}_{lk}$. If SP$_l$ has N antenna elements, $\mathbf{h}_{lk}$ is a vector of size N. The data symbol s_k transmitted by UE$_k$ is a complex variable. The received signal at SP$_l$ denoted as $\mathbf{y}_l$, a vector of length N, is a superposition of the signals sent from all UEs and can be written as

$$\mathbf{y}_l = \sum_{k=1}^{K} \mathbf{h}_{lk} s_k + \mathbf{n}_l, \tag{2.4}$$

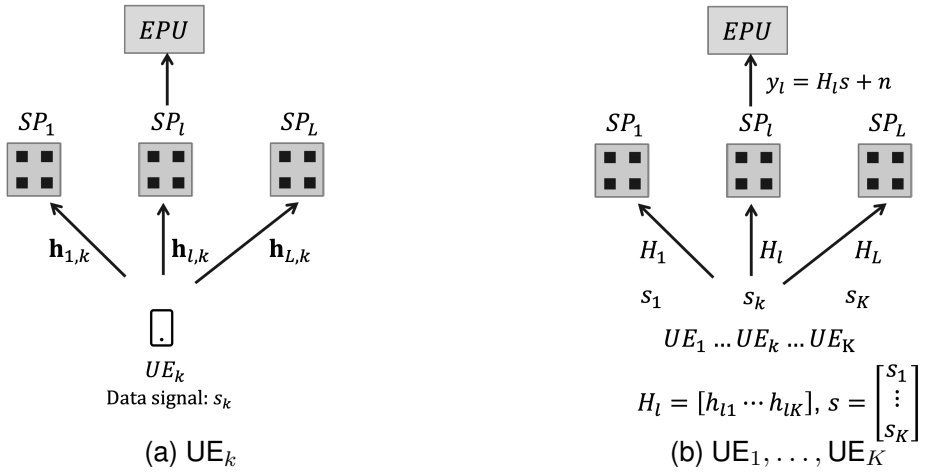
where $\mathbf{n}_l$ is the receiver noise at SP$_l$.

In the D-MIMO network, each UE$_k$ can be served by several serving CSPs, say L CSPs: SP$_1, \ldots,$ SP$_L$. The received signals can be expressed as a vector of length NL, $\mathbf{y} = [\mathbf{y}_1 \ldots \mathbf{y}_L]^T$. Then (2.4) becomes

$$\begin{bmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_L \end{bmatrix} = \sum_{k=1}^{K} \begin{bmatrix} \mathbf{h}_{1k} \\ \vdots \\ \mathbf{h}_{Lk} \end{bmatrix} s_k + \begin{bmatrix} \mathbf{n}_1 \\ \vdots \\ \mathbf{n}_L \end{bmatrix}. \tag{2.5}$$

We have K UEs in the D-MIMO network, as shown in Fig. 2.5 (b), we express the uplink signal from all UEs as a vector of a length K: $\mathbf{s} = [s_1, \ldots, s_K]^T$. We put the channel vectors between all UEs and SP$_l$ in a matrix form $H_l = [\mathbf{h}_{l1} \cdots \mathbf{h}_{lK}]$, which has a size $N \times K$. The received signal $\mathbf{y}_l$ can now be written as

$$\mathbf{y}_l = H_l \mathbf{s} + \mathbf{n}_l. \tag{2.6}$$



(a) UE_k (b) $\text{UE}_1, \dots, \text{UE}_K$

Figure 2.5: Uplink data transmission in a D-MIMO network

The uplink data processing is to use the received signals **y** to find what data signal **s** that UE transmitted. It can be processed centralised at a ECSP or locally at CSPs. Note that the received signals are distributed in different CSPs. The uplink data combining is to put together either received signals or processed signals to get an estimation of the data signal **s**.

The more CSPs that are selected for processing uplink signals from a UE, the better performance and the larger complexity in processing. The trade-off between the performance and complexity needs not to be identical for each UE active in an uplink transmission time interval (TTI).

2.2.2 Centralised Combining

There are different approaches to process and combine the uplink data. In [15], the four levels combining method using MMSE was proposed: from fully centralised method, which is called level 4, to local processing and centralised combing level 1-3. We summarise the four levels method briefly in this section, more details are found in [15].

2.2.2.1 Level 4: Fully centralised Processing

Level 4 is a method with fully centralised processing and combining. It requires that all L CSPs that are connected to an ECSP of a D-MIMO network send all received pilot signals and received data signals $\{\mathbf{y}_l : l = 1, \dots, L\}$ to the ECSP. For each UE, the ECSP estimates the channel $\{\hat{\mathbf{h}}_{lk} : l = 1, \dots, L, k = 1, \dots, K\}$ using received pilot signals and channel statistics obtained from CSPs. Then the ECSP selects combining weights $\mathbf{v}_k \in \mathbb{C}^{NL}$ for UE_k based on the collective channel estimate $\hat{\mathbf{h}}_k = [\hat{\mathbf{h}}_{1k}, \dots, \hat{\mathbf{h}}_{Lk}]$.

The MMSE combining vector for UE_k that maximises the instantaneous SINR minimises the mean-squared error $\text{MSE}_k = \mathbb{E}\{|s_k - \mathbf{v}_k^H \mathbf{y}|^2 \mid \hat{\mathbf{h}}_k\}$, see [18] for details, is given by

$$\mathbf{v}_k = p_k \left(\sum_{i=1}^{K} p_i \left(\hat{\mathbf{h}}_i \hat{\mathbf{h}}_i^H + C_i \right) + \sigma^2 I_{NL} \right)^{-1} \hat{\mathbf{h}}_k, \tag{2.7}$$

where p_k is the transmission power for UE_k, σ^2 is the variance of the thermal noise, and I_{NL} is the identity matrix of size $N \times L$. The maximum SINR is given by

$$\text{SINR}_k^{(4)} = p_k \hat{\mathbf{h}}_k^H \left(\sum_{i=1, i \neq k}^{K} p_i \hat{\mathbf{h}}_i \hat{\mathbf{h}}_i^H + \sum_{i=1}^{K} p_i C_i + \sigma^2 I_{NL} \right)^{-1} \hat{\mathbf{h}}_k.$$

Using level 4 combining, an achievable SE of UE k is shown to be

$$\mathrm{SE}_k^{(4)} = \left(1 - \frac{\tau_P}{\tau_c}\right) \mathbb{E}\left\{\log_2\left(1 + \mathrm{SINR}_k^{(4)}\right)\right\}, \tag{2.8}$$

where τ_c it the length of the channel coherence interval and τ_P is the pilot sequence length. The level 4 provides the best performance, an upper bound, in terms of SE, equivalently SINR. However the computational complexity is very high since it requires first an inverse of $NL \times NL$ matrix and then a matrix-vector multiplication. The other challenge is the requirement on front-haul connection capacity, i.e., the number of signals and measurements that are required to send from all CSPss to ECSP in order to calculate the combining weights.

2.2.2.2 Level 1-3: Local Processing and Centralised Combining

Level 1-3 are the methods based on local processing and centralised combining. Instead of sending the N-dimensional vector $\{\mathbf{y}_l : l = 1, \ldots, L\}$ and channel estimates to the ECSP, each CSP pre-processes its signals by computing the local estimates of the data signals that are then passed to the ECSP for further combining. The local estimate for UE_k at SP_l is $\breve{s}_{kl} = \mathbf{v}_{lk}^H \mathbf{y}_l$ where the local MMSE combining vector is

$$\mathbf{v}_{lk} = p_k \left(\sum_{i=1}^{K} p_i \left(\hat{\mathbf{h}}_{li}\hat{\mathbf{h}}_{li}^H + C_{li}\right) + \sigma^2 I_N\right)^{-1} \hat{\mathbf{h}}_{lk}. \tag{2.9}$$

The maximum value of SINR with the local MMSE combining (2.9) is given by

$$\mathrm{SINR}_{kl}^{(1)} = p_k \hat{\mathbf{h}}_{lk}^H \left(\sum_{i=1, i\neq k}^{K} p_i \hat{\mathbf{h}}_{li}\hat{\mathbf{h}}_{li}^H + \sum_{i=1}^{K} p_i C_{li} + \sigma^2 I_N\right)^{-1} \hat{\mathbf{h}}_{lk}.$$

Different from level 4, SP_l uses only its own local channel estimates $\{\hat{\mathbf{h}}_{lk} : k = 1, \ldots, K\}$ for calculating $\mathbf{v}_{lk}$. The local estimates $\{\breve{s}_{kl} : l = 1, \ldots, L\}$ are then sent to the ECSP where they are combined in three different methods, which are defined in [15]:

Level 1: $\hat{s}_k = \underset{\breve{s}_{kl}}{\operatorname{argmax}}\ \mathrm{SINR}_{kl}^{(1)},\ l = 1, \ldots, L$

Level 2: $\hat{s}_k = \frac{1}{L}\sum_{l=1}^{L} \breve{s}_{kl}$

Level 3: $\hat{s}_k = \sum_{l=1}^{L} a_{lk}^* \breve{s}_{kl}$

The weighting coefficients a_{lk} at level 3 can be obtained based on the channel statistics, see [15] for more details.

2.2.3 Decentralised Processing and Combining

In [19] a decentralised processing and combining method that utilises a sub-set of CSPs for processing of uplink signals was proposed. Instead of sending all signals or estimated signals to the ECSP to determine the combining vector, a sub-set of CSPs, $\mathcal{L} \subset \{1, \ldots, L\}$, is selected as serving CSPs for each UE and one of the CSPs in the sub-set is assigned to be the aggregating CSP for this UE. We call this the "subset" or "subset method". The selection of the serving CSPs can be based on the path-gain between CSPs and UE. The number of serving CSPs to form

the subset is a trade-off between the complexity and performance. In general, we denote the subset by sub-ℓ when ℓ out of L CSPs are selected. Sub-1 has a single serving CSP, which turns D-MIMO into "small cell". We can aggregate the number of serving CSPs depending on the front-haul and performance requirement, from two serving CSPs, sub-2, and all the way up to include all CSPs in the D-MIMO network, sub-L.

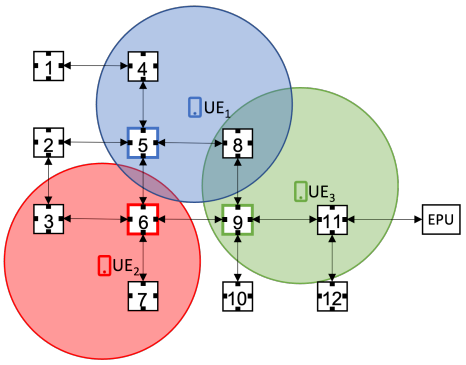


Figure 2.6: Subset that consists of three CSPs for each UE.

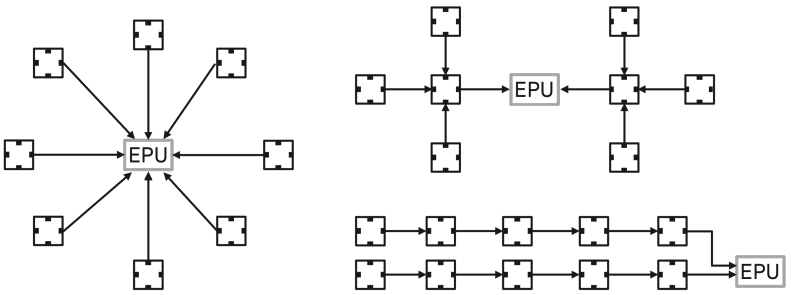


Figure 2.7: Different topologies for CSPs connection to ECSP.

In Fig. 2.6, we illustrate one example of a subset with $\ell = 3$ serving CSPs, sub-3. In this example, each UE picks three best CSPs as serving CSPs based on the path-gain to CSPs. As shown in this case the serving CSPs for UE_1 is $\mathcal{L}_{\text{UE}_1} = \{4, 5, 8\}$ where SP_5 is assigned to be the aggregating CSP where antenna signals from other two SP_4 and SP_8 are collected. The serving CSPs for UE_2 is $\mathcal{L}_{\text{UE}_2} = \{3, 6, 7\}$ where SP_6 is assigned to be the aggregating CSP and the serving CSPs for UE_3 is $\mathcal{L}_{\text{UE}_3} = \{8, 9, 11\}$ where SP_9 is assigned to be the aggregating CSP.

This enables the front-haul communication to send information to different aggregation CSPs, using different sections of the front-haul, all at the same time. It further enables parallel processing of uplink signals from multiple UEs in different aggregating CSPs. The processing for each UE is also significantly reduced when a sub-set of CSPs are used for reception compared to the full set of CSPs.

2.2.4 Deployment Model and Simulation Parameters

In this section, we evaluate the subset method by deploying a D-MIMO network in a 100×100 m^2 area as shown in Fig. 2.8. A number of service points (CSPs), denoted by L, are deployed on a square grid. The propagation of a reference case is show in Fig. 2.8 when $L = 36$. Each CSP is equipped with a number of antenna elements, from single antenna element, i.e. $N = 1$, to multiple antenna elements e.g. $N = 16$. To be able to compare the level 1-4 methods given

by [15] and our decentralised subset method we use the same channel propagation model and estimation as described in [15], which was based on 3GPP Urban Microcell model in [20]. Except for the size of simulation area, we keep the same parameters used by [15]. The system simulation parameters are summarised in Table 2.4.

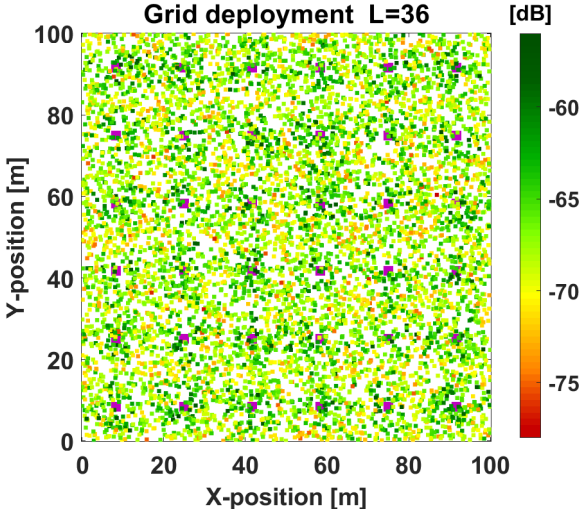


Figure 2.8: Path-gain in a D-MIMO network with 36 CSPs.

Table 2.4: System parameters used for downlink performance evaluations

Carrier frequency	2 GHz
Bandwidth	20 MHz
Duplex	Symmetric TDD Number of data samples for UL and DL $\tau_D = \tau_C - \tau_P$
Simulation area	100×100 m^2
Wrap-around	8 twin areas
Channel Assumptions	Perfect reciprocity, Block fading, Independent Rayleigh fading Correlated log-normal shadow fading, $\sigma_{sh} = 4$dB, 9 m correlation distance 3GPP Urban Microcell path-loss model
Coherence time	$\tau_C = 200$ samples
Channel estimation	τ_P uplink pilots (no downlink pilots), random pilot allocation
UE, CSP height	1.5 m and 10 m respectively
UE and CSP power	100 mW
Antenna power budget	50 mW (4 antenna elements per CSP)
Noise power	-94 dBm

2.2.5 Numerical Comparisons

The performance metrics used is the achievable SE [bit/s/Hz] based on the Shannon formula, which is the same as it was used in [15] [16]. The evaluation of the performance with four-level centralised combining method was done by the code provided by [15]. The evaluation of the performance with decentralised combining method proposed in this section was done by the square-root implementation of the Kalman Filter. Each simulation result contains 1000 random

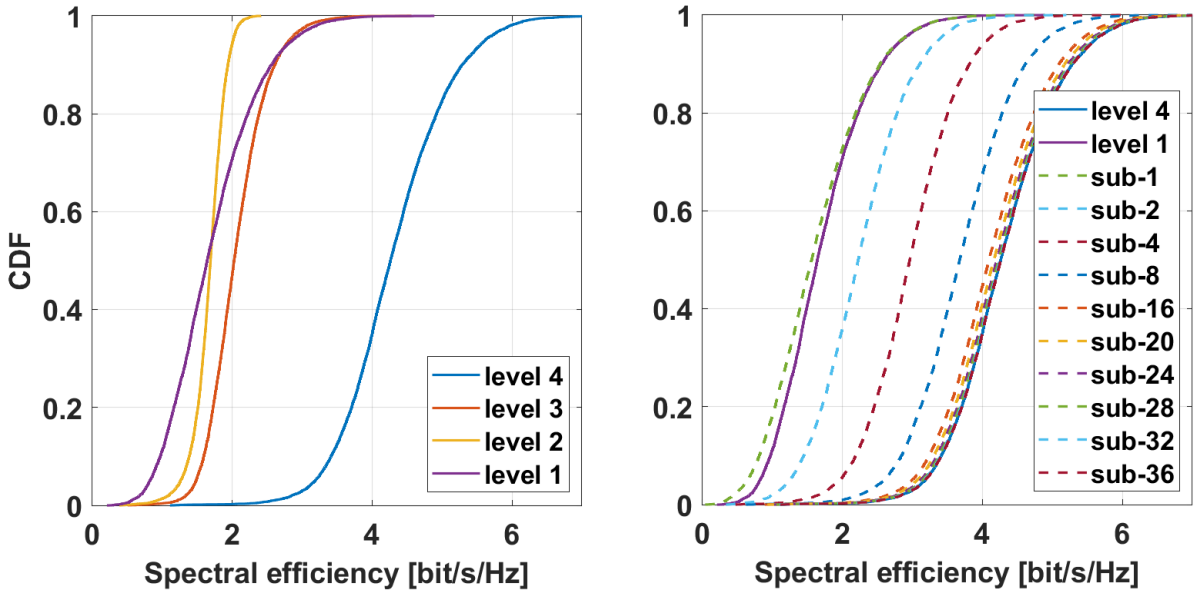


Figure 2.9: Centralised and decentralised combining methods using MMSE.

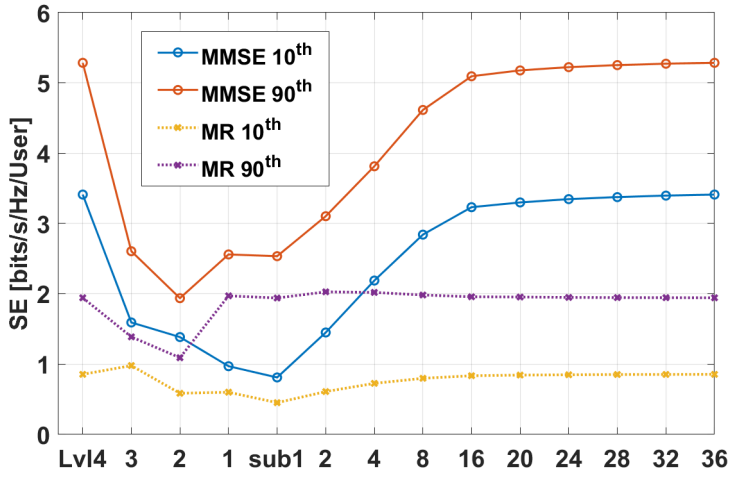


Figure 2.10: SE@10^{th} and 90^{th} percentile with different combining methods using MMSE and maximum ratio combining (MRC).

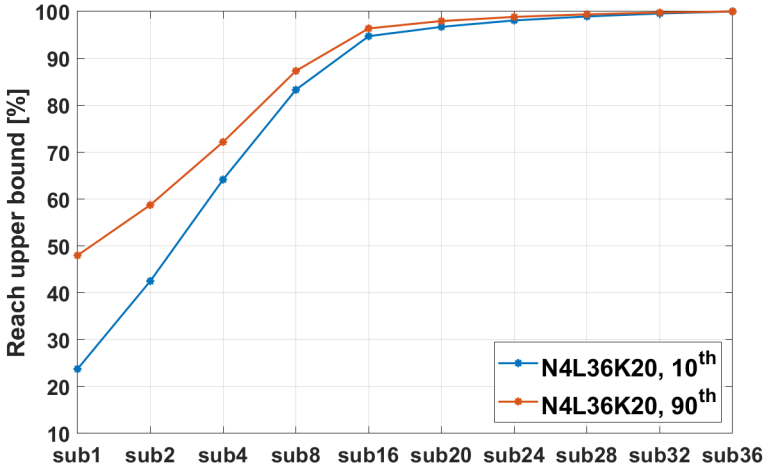


Figure 2.11: Comparing subsets with level 4.

channel realisations and 8000 random UEs samples. When comparing the results in the sub-sections we use solid curves for four levels centralised combining and dashed curves for the subsets method whenever they are presented in the same figure.

2.2.5.1 Centralised vs. Decentralised Combining

To compare the simulation results for centralised and decentralised combining methods, a reference case is created where $L = 36$ CSPs were deployed as a grid in an area of $100 \times 100\,\mathrm{m}^2$, each CSP is equipped with $N = 4$ antennas and $K = 20$ UEs are randomly generated in the area.

In Fig. 2.9 we show the cumulative distribution function (CDF) of the SE when different combining methods are applied: the four-level centralised combing methods are the solid curves, from level 1 to level 4, and decentralised subset method are the dashed curves, from sub-1 to sub-36. We can see that the performance upper bound of fully centralised level 4 is reached by decentralised sub-36 when all $L = 36$ CSPs are added into the subset, which numerically verifies the *Proposition 2*.

Fig. 2.10 shows the 10^{th} and 90^{th} percentile of the CDFs in Fig. 2.9 as a function of combining methods: four levels on the left, decreasing from level 4 to level 1, and moving towards the right the decentralised combining method with increasing number of serving CSPs in the subset from sub-1 to sub-36 on x-axis. Both MMSE (solid) and MRC (doted) were used. The first conclusion that we get is that the MMSE combing outperforms the MRC. In the rest of sections we show only the results with MMSE combining. The second observation is that the performance upper bound of level 4 can be reached adding more serving CSPs in the subset, the more serving CSPs the better performance, however diminish return. We observed that a subset with 16 CSPs, sub-16, is almost as good as level 4, sub-4 outperforms level 1-3 and sub-2 is almost as good as level 3.

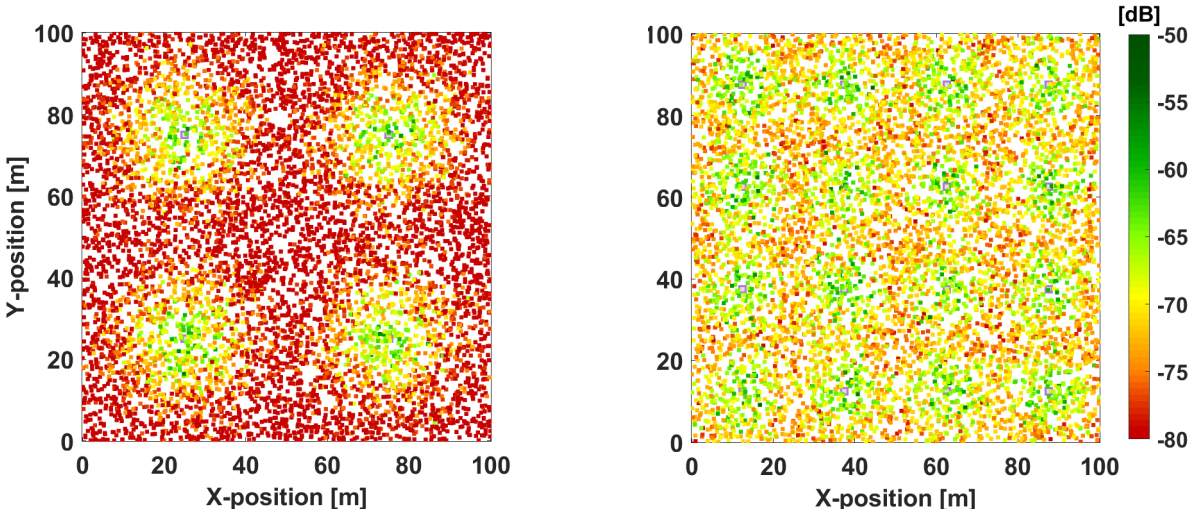
In [21] we proofed theoretically that applying the Kalman filter, the performance upper bound can be approached by incrementing the subset stepwise. Fig. 2.11 shows numerically how to reach the performance upper bound by using different number of serving CSPs in the subset. The upper bound performance of level 4, shown as 100% of the performance, is reached by sub-36. Considering adding more CSPs into the subset gives little gain at high costs not only the traffic load at front-haul but also the processing delay due to the computation complexity, see Section 2.3.2, we can reduce performance requirement by setting a performance target, for example to reach 95% of the performance upper bound by using 16 serving CSPs, sub-16, which consists of 40% of CSPs; If sub-16 is considered to cause too much traffic load at front-haul we can reduce the performance target to reach for example 85% of the performance upper bound using sub-8 only 20% of CSPs; Or even lower to reach 65% of the performance upper bound by using only 4 serving CSPs, sub-4, that is 10% of CSPs; So decentralised subset combing provides another advantage: it is very flexible and easy to select the trade-off between the complexity and performance.

2.2.5.2 Sweep the Number of Deployed CSPs

The number of CSPs deployed in the area can be varied. Besides the reference case shown in Fig. 2.8 with $L = 36$, we show the propagation of grid deployment with $L = 4$ and $L = 16$ are shown in Fig. 2.12(a) and (b).

In this sub-section we show the performance when the number of CSPs deployed in the area is varied from $L = 1$ to $L = 100$ while keeping the other parameters as the reference case, $N = 4$,

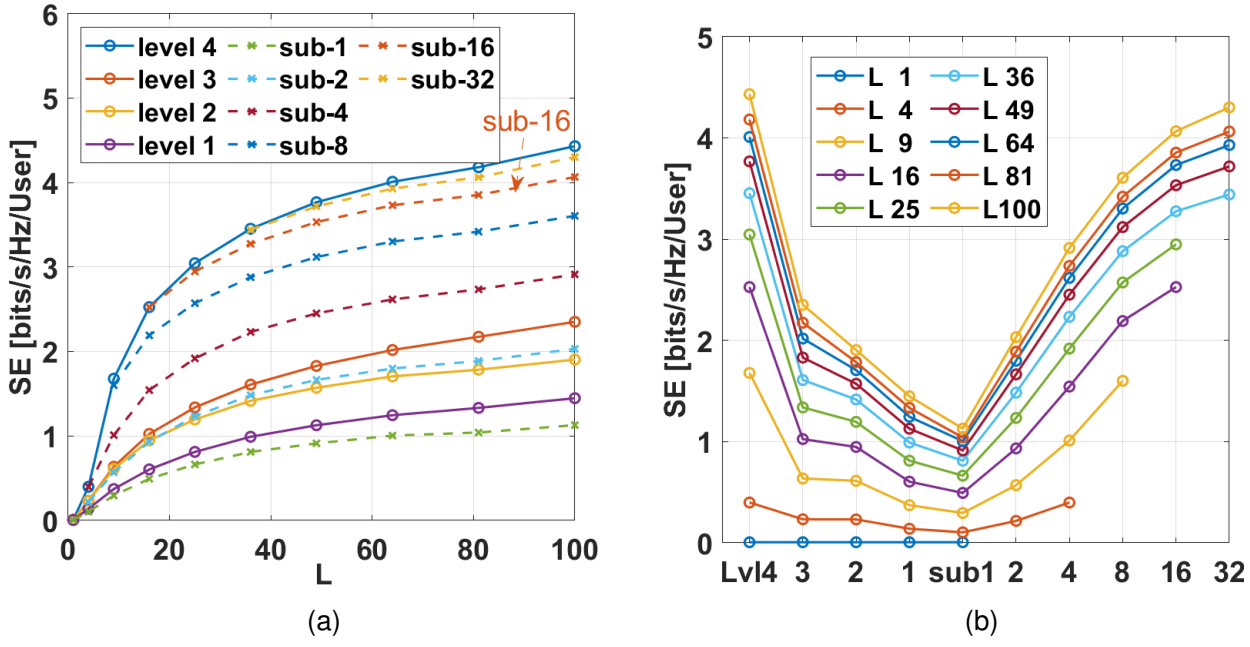
$K = 20$ and $\tau_P = 15$. From radio propagation point of view, the more CSPs the higher the path-gain to the best CSP the UEs are likely to have.



(a) A deployment with four CSPs.

(b) A deployment with 16 CSPs.

Figure 2.12: Propagation path-gain in a D-MIMO network.



(a)

(b)

Figure 2.13: SE@10^{th} percentile by sweeping L, MMSE combining.

In Fig. 2.13(a) we show the 10^{th} percentile of SE as a function of the number of deployed CSPs, L, when we double the number of CSPs. We can see that sub-1 is the lower bound which is completely distributed using local processing without any information from other CSPs. Level 1 is slightly better than sub-1 as level 1 requires each CSP to forward local estimates of the signal to the ECSP for centralised combining. It relies on the ECSP to find the maximum value of SINR or SE among all CSPs. Hence the difference is that sub-1 captures the slow fading while level 1 captures the fast fading and interference. Level 4 provides the upper bound, which is the most complexity in terms of the implementation. Sub-16 gets very close to level 4 in the performance and the implementation is much simpler with Kalman Filter. The sub-2 requires only aggregating

two CSPs and gets quite close to the performance of level 3. Fig. 2.13(b) shows the 10^{th} percentile of SE as a function of all levels and subsets. Each curve is a deployment with the number of L CSPs. Note that the number of CSPs in the subset cannot exceed the total number of CSP L. We can see that the more deployed CSPs the better the performance, however with diminishing return.

2.2.5.3 Sweep the Number of Antenna Elements at Each CSP

We study the performance when the number of antenna elements at each CSP is varied from $N = 1$ to $N = 16$ while keeping the other parameters as the reference case, $L = 36$, $K = 20$ and $\tau_P = 15$. Fig. 2.14 shows the 10^{th} percentile of SE as a function of the number of antenna elements at each CSP, N. The sub-4 outperforms all level 1-3 and sub-2 outperforms level 1 and is almost as good as level 3 when $N \geq 8$. In Fig. 2.15 we group the results for each number of antenna elements in the bar charts and make it easier to compare four levels and 6 selected subsets for each setting of antenna elements. As a general rule we can see that the more antenna elements we have the better performance we get. It is also interesting to note that for $N = 16$ the performance of Level 1 combining is better than for Level 2 combining (which is not the case for $N = 8$, $N = 4$, $N = 2$, $N = 1$). Level 1 combining implies calculating the SINR at each CSP and then selecting the maximum value out of all N such estimates. Level 2 combining implies taking a linear average of local symbol estimates calculated at each CSP, see [15]. In these experiments, the performance for Level 1 combining (purple bars in Fig. 2.15) increase more rapidly as the number of CSP in the subset increase and for N=16 Level 1 combining becomes better than Level 2 combining. The results represent the 10^{th} of the worst UE. In this setup, the poor performance is due to the reuse of pilots, increase the number of antennas doesn't help to improve their performance.

2.2.5.4 Sweep the Number of UEs in the Simulation Area

We further investigate the performance when the number of UEs is varied from $K = 1$ to $K = 40$ while keeping the other parameters as the reference case, $N = 4$, $L = 36$ and $\tau_P = 15$. Fig. 2.16 shows the 10^{th} percentile of SE as a function of the number of UEs, K. In the case of a single UE, $K = 1$, there is no interference and no pilot contamination hence the highest performance for all levels and subsets is achieved. When the number of UEs increases to $\tau_P = 15$, the interference increases without pilot contamination, sub-2 outperforms level 1-3. When the number of UEs becomes large than the number of pilots τ_P, sub-4 outperforms level 1-3.

2.2.5.5 Sweep the Number of Pilot Symbols

We show the performance when the number of pilot symbols is varied from $\tau_P = 1$ to $\tau_P = 50$ while keeping the other parameters as the reference case, $N = 4$, $L = 36$ and $K = 20$. Fig. 2.17 shows the 10^{th} percentile of SE as a function of the number of UEs, τ_P. Since we have 20 UEs there are pilot contaminations for all $\tau_P < 20$. Once again, we see that sub-4 outperforms level 1-3.

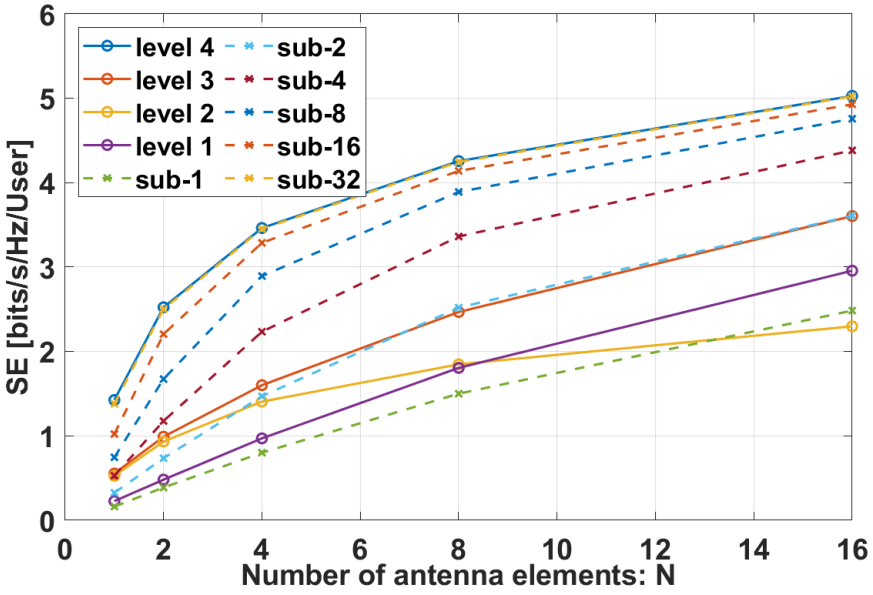


Figure 2.14: SE@10^{th} percentile by sweeping N.

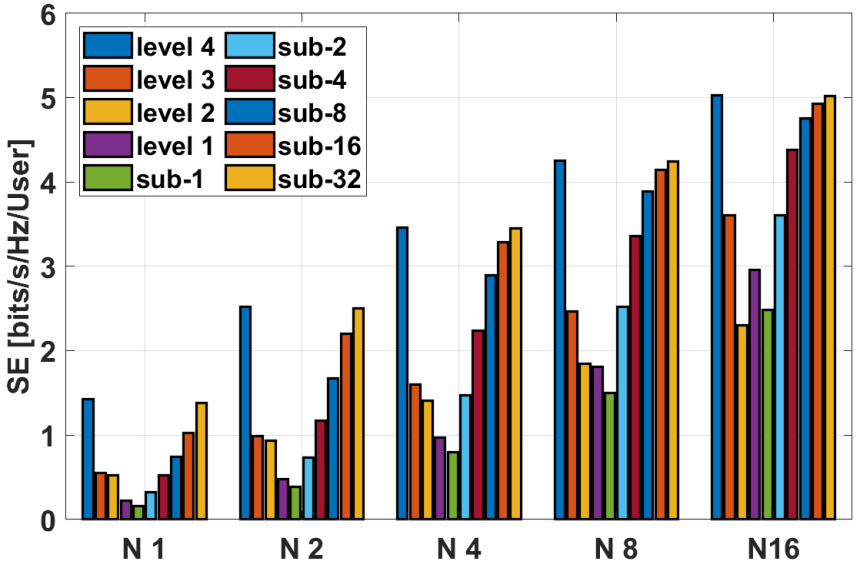


Figure 2.15: SE@10^{th} percentile by sweeping N.

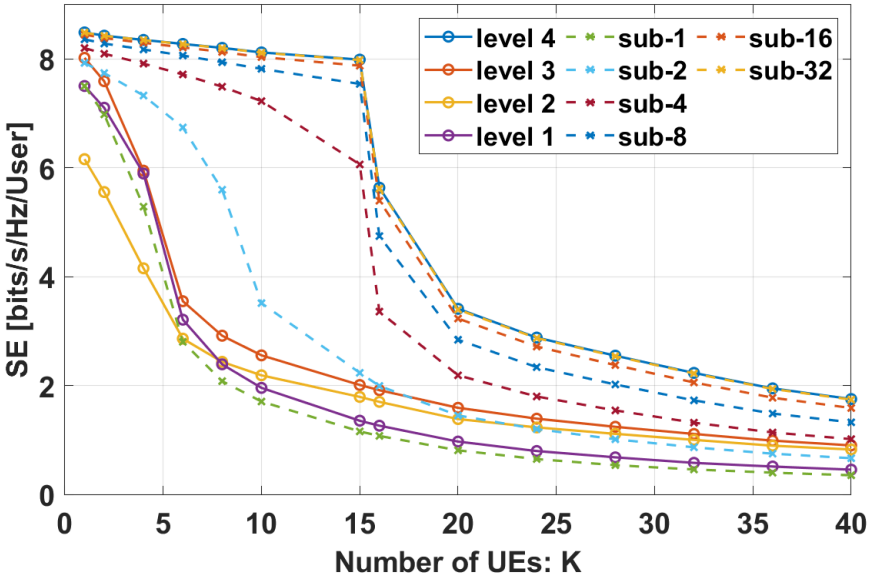


Figure 2.16: SE@10^{th} percentile by sweeping K, where $\tau_P = 15$.

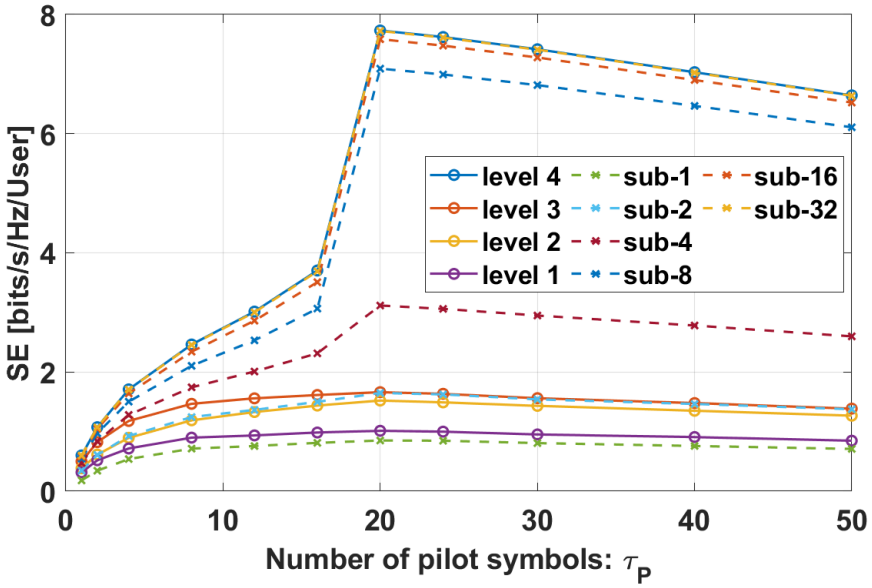


Figure 2.17: SE@10^{th} percentile by sweeping N, where $K = 20$.

2.3 Uplink D-MIMO Processing using Kalman Filter Combining

In Section 2.2 different uplink D-MIMO processing and combing methods were introduced and evaluated, from the fully centralised method to distributed subset methods. The fully centralised approach, which is denoted level 4 implementation, uses global MMSE combining, and hence it provides the best performance in terms of the SE and equivalently SINR. However, the level 4 implementation is difficult to realise in practice, due to the very high front-haul connection requirement and computational complexity at the D-MIMO ECSP. To reduce the computational complexity at the ECSP, three alternative methods, denoted level 1-3, first process the signals at each CSP based on local information, and then pass them to the ECSP for final combining and decoding. Level 1-3 do reduce the front-haul requirement, however, these methods still require centralised combining, which has an impact on the processing latency. The performance of level 1-3 reaches below 40% of the upper bound performance provided by level 4. To find the possibility to reach the upper bound performance, we propose to use a decentralised partial level-4 method by defining a subset of CSPs for each UE. The subset can be defined from one CSP to all CSPs that are connected to a ECSP. When the subset consists of only one CSP, which we call sub-1, it turns D-MIMO network into a "small cell" network; when the subset consists of all CSPs connected to the ECSP, it is equivalent to the level 4. The subset combing method introduced Section 2.2 can be implemented by using MMSE, it can also be implemented by using the Kalman filter [22]. The advantage of using the Kalman filter is that it provides the combining vector that minimises the estimation error (MSE) at each step of iteration, from a subset of one CSP to a subset of all CSPs. In this section we first describe the Kalman filter in Section 2.3.1. We analyse and compare the implementation requirements when different combing methods are applied in Section 2.3.2. Finally we evaluate and make numerical comparisons of different implementations in in Section 2.3.3.

2.3.1 Kalman Filtering for Decentralised Combining

In [19], a decentralised processing and combining method was proposed. Instead of sending all signals and statistical parameters to the ECSP for centralised combining they are sent to predefined aggregating CSPs, or local process unit (LPU) for decentralised processing and combining first and then forwarded to the ECSP for final decoding. A closely related method on decentralised MMSE combining for signal detection was developed earlier in [23]. The predefined aggregating CSP consists of a subset of CSPs, which is selected for each UE, $\mathcal{L} \subset \{1, \ldots, L\}$.

The implementation complexity of the UL combining methods is dominated by the calculation of the inverse of the covariance matrix. We propose to use the square-root implementation of the Kalman filter [24] for decentralised processing and combining at aggregating CSPs. The square-root implementation is known to be more numerically sound when inverting the covariance matrix, as it always assures the covariance matrix to be symmetric and positive semi-definite.

The Kalman filter is used to estimate the data signal vector $\mathbf{s}$ given $\mathbf{y}_l$ in (2.6) at any SP$_l$ as

$$\hat{\mathbf{s}} = \hat{\mathbf{s}}_0 + \mathcal{K}\left(\mathbf{y}_l - H_l \hat{\mathbf{s}}_0\right), \tag{2.10}$$

where $\hat{\mathbf{s}}_0 = 0$ and $P_0 = \text{diag}(p_1, \ldots, p_K)$ are initial state vector and covariance matrix. The Kalman filter gain $\mathcal{K}$, a matrix of size $K \times N$, provides the optimal solution in terms of the minimum mean-square error MSE $= \mathbb{E}\{|\mathbf{s} - \mathcal{K}\mathbf{y}_l|^2\}$.

Proposition 1. *The combining vectors for all UEs can be calculated by the Kalman filter gain $\mathcal{K}$, which is equivalent to the MMSE combining vector given by* (2.7)*.*

The MMSE combining vector $\mathbf{v}_k$ given by (2.7) assumes that the channel H_l is unknown, but the estimate of the channel can be done locally at each CSP based on known pilot signals transmitted from UEs. Let $\widehat{H}_l = [\hat{\mathbf{h}}_{l1}, \ldots, \hat{\mathbf{h}}_{lK}]$ be the estimate of channel H_l, see [18] for a detailed description of channel estimation. Applying the estimation of the channel, the estimate of the state vector (2.10) can then be written as

$$\hat{\mathbf{s}} = \hat{\mathbf{s}}_0 + \mathcal{K}(\mathbf{y}_l - \widehat{H}_l \hat{\mathbf{s}}_0), \quad l = 1, \ldots, L \tag{2.11}$$

and the covariance matrix becomes

$$P = P_0 - P_0 \widehat{H}_l^H (\widehat{H}_l P_0 \widehat{H}_l^H + R_l)^{-1} \widehat{H}_l P_0. \tag{2.12}$$

The Kalman filter gain is given by

$$\mathcal{K} = P_0 \widehat{H}_l^H (\widehat{H}_l P_0 \widehat{H}_l^H + R_l)^{-1}, \tag{2.13}$$

where R_l is the covariance matrix including both the correlation matrix of the channel estimation error and receiver noise, $R_l = \sum_{k=1}^{K} p_k C_{lk} + \sigma^2 I$. The equation (2.13) provides the combining vectors that the minimises the MSE for all UEs, $\mathcal{K}^T = [\mathbf{v}_1 \cdots \mathbf{v}_K]$. Thus, the combining vector for UE$_k$, $\mathbf{v}_k$, is the same as it is given in (2.7).

Remark 1. *In the square-root implementation of the Kalman filter, the covariance matrix is replaced by its square-root, S, such that $P = SS^T$.*

Corollary 1. *The performance measure SINR for each UE$_k$ is equivalent to the inverse of the covariance matrix given by the Kalman filter*

$$\mathit{SINR}_k = \frac{p_k}{P_{kk}} - 1, \tag{2.14}$$

where p_k is the power of UE$_k$ use for transmitting data signals and P_{kk} is the k^{th} diagonal element of the covariance matrix P.

The SINR represents the ratio between desired signal power and undesired total signal power, which is often called the interference and noise. For UE$_k$, it can be rewritten as

$$\text{SINR}_k = \frac{s_k}{A_k - s_k} = \frac{\frac{s_k}{A_k}}{I - \frac{s_k}{A_k}},$$

where s_k is desired signal power and A_k is the total sum of received signal power. The ratio of desired signal to the total sum is a part of the covariance matrix as

$$\frac{s_k}{A_k} = p_k \hat{\mathbf{h}}_{lk}^{\mathsf{H}} (\widehat{H}_l P_0 \widehat{H}_l^{\mathsf{H}} + R_l)^{-1} \hat{\mathbf{h}}_{lk} = \mathbf{v}_k \hat{\mathbf{h}}_{lk}.$$

The covariance matrix P in (2.12) has two parts, the initial value P_0 and the updating part, which can be written as $\mathcal{K}\widehat{H}_l P_0$ by using $\mathcal{K}$ in (2.13). Hence (2.12) can be written as

$$P = P_0 - \mathcal{K}\widehat{H}_l P_0 \quad \text{or} \quad \mathcal{K}\widehat{H}_l = (P_0 - P)P_0^{-1}.$$

Hence

$$\mathbf{v}_k \hat{\mathbf{h}}_{lk} = \mathbf{u}_k^{\mathsf{T}} \mathcal{K} \widehat{H}_l \mathbf{u}_k = \mathbf{u}_k^{\mathsf{T}} (P_0 - P) P_0^{-1} \mathbf{u}_k$$

where $\mathbf{u}_k^{\mathsf{T}} = [0 \dots 1 \dots 0]$ is a vector with 1 at k^{th} element.

$$\text{SINR}_k = \frac{\frac{s_k}{A_k}}{1 - \frac{s_k}{A_k}} = \frac{\mathbf{u}_k^{\mathsf{T}}(P_0 - P)P_0^{-1}\mathbf{u}_k}{1 - \mathbf{u}_k^{\mathsf{T}}(P_0 - P)P_0^{-1}\mathbf{u}_k} = \frac{p_k}{P_{kk}} - 1.$$

We have introduced the Kalman filter to estimate the data signals by (2.11)-(2.13). They can be applied to any CSPs locally. They can also be applied to any aggregating CSPs or local processing unit (LPU), such as the subset, sub-l, or at ECSP where signals from other CSPs are collected as shown in Fig. 2.7. Furthermore, when CSPs are connected sequentially where one CSP forwards the estimates to another CSP, the update of the state vector (2.11) can be generalised with the proposition below.

Proposition 2. *The estimate of data signal at any aggregating CSP can be obtained by the estimation of the state vector from the Kalman filter whenever new measurement from additional SP$_l$ becomes available.*

Let $\mathcal{L}$ be the subset consisting of ℓ CSPs, and the new measurement from additional SP$_l$ be $\mathbf{y}_l$, $l \notin \mathcal{L}$. The subset of ℓ CSPs is denoted by sub-ℓ. The new subset $\mathcal{L} \cup l$ has $\ell + 1$ CSPs, sub-$(\ell + 1)$. Applying the Kalman filter, the estimate of the state vector is given by

$$\hat{\mathbf{s}}(\mathcal{L} \cup l) = \hat{\mathbf{s}}(\mathcal{L}) + \mathcal{K}(\mathbf{y}_l - \widehat{H}_l\, \hat{\mathbf{s}}(\mathcal{L})) \tag{2.15}$$

and the covariance matrix is given by

$$P(\mathcal{L} \cup l) = P(\mathcal{L}) - P(\mathcal{L})\widehat{H}_l^H(\widehat{H}_l P(\mathcal{L})\widehat{H}_l^H + R_l)^{-1}\widehat{H}_l P(\mathcal{L}). \tag{2.16}$$

The Kalman filter gain that minimises the mean-square errors between the state and estimated state is given by

$$\mathcal{K} = P(\mathcal{L})\widehat{H}_l^H \left(\widehat{H}_l P(\mathcal{L})\widehat{H}_l^H + R_l\right)^{-1}. \tag{2.17}$$

The equation (2.15) updates the state vector from $\hat{\mathbf{s}}(\mathcal{L})$ to $\hat{\mathbf{s}}(\mathcal{L} \cup l)$ with the new measurement $\mathbf{y}_l$ at the aggregating CSP and the equation (2.16) updates the covariance matrix from $P(\mathcal{L})$ to $P(\mathcal{L} \cup l)$ when the SP$_l$ is added to the subset. The Kalman filter gain (2.17) updates the estimate of the state vector which is the new combining weights that minimises the MSE for all UEs.

Corollary 2. *The best performance in terms of SINR or equivalently SE is achieved by sub-L when all CSPs in the same cluster are added to the subset $\mathcal{L}$.*

By using the Kalman filter, the estimate of the state vector is updated by (2.10) when a new CSP is added into the subset. Each new CSP brings the new measurement into the Kalman filter and reduces the estimation error hence the corresponding covariance matrix is decreased. Each update increases SINR as SINR is the inverse of the covariance matrix as shown in *Corollay 1*. Hence the best performance in terms of SINR or equivalently SE is achieved by sub-L when all CSP in the same cluster are added to the subset $\mathcal{L}$.

Remark 2. *If the new measurements in Proposition 2 are collected from several CSPs, $l_1, \dots, l_r \notin \mathcal{L}$. The proposition is still valid by defining $\mathcal{L}_{new} = \{l_1, \dots, l_r\}$ and replacing l by $\mathcal{L}_{new}$, the new subset $\mathcal{L} \cup \mathcal{L}_{new}$ has thus $\ell + r$ CSPs and is denoted as sub-$(\ell + r)$.*

The advantage of using the Kalman filter in this context is that the upper bound performance by the fully centralised level 4 can be reached by sub-L. However, the processing of sub-L can be done either centralised or decentralised at any aggregating CSPs by using (2.11)-(2.13); it can also be done by distributing the processing at aggregating CSPs and push the estimate to another CSP. The aggregating can also be done when CSPs are connected in parallel.

Proposition 3. *The Kalman filter provides equivalent results regardless if the signals are collected and processed centralised at ECSP; or decentralised at an aggregating CSP; or distributed at each of the serving CSPs.*

Since the Kalman filter minimises the estimation error between the signal and estimate of the signal based on the available inputs and measurements at current stage, the estimate of the signal contains the same amount of information as what is available to the Kalman filter at any subset. They can be forwarded to another CSP or aggregating CSP without loss of the information of that subset.

Remark 3. *With Proposition 3, the Kalman filter can be applied in any topology of CSP, e.g. star, serial, parallel, grid connections. The estimate of data signal at any aggregating CSP is the same whether new measurements are collected from several CSPs or sequentially aggregated from the first CSP to the aggregating CSP.*

The Kalman filter processing performed in a service point within a RW thus consists of two slightly different Kalman filter operations, as shown in Fig. 2.18. The first type of Kalman filter (type 1) operation combines symbol estimates from different branches of the RW, and the second type of Kalman filter (type 2) operation updates the symbol estimate using the antenna signal observations.

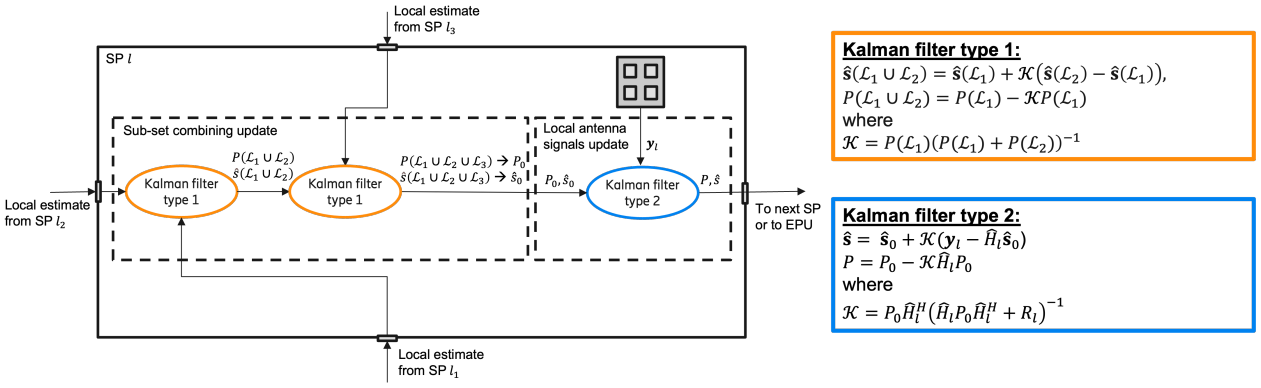


Figure 2.18: Example of a CSP within a RW performing two types of Kalman filter operations.

2.3.2 Implementation requirements

We analyse two important aspects that nees to be considered when it comes to the implementation of a D-MIMO system.

Firstly, we analyse the front-haul connection capacity requirement, i.e. the number of signals and measurements that are required to send from CSPs to ECSP in order to calculate the combining weights. The four levels combining method using MMSE requires to collect signals $\mathbf{y}_l$ or the processed signals $\breve{s}_{kl}$ at ECSP to be able to calculate the combining vector as described in Section 2.2.2. In contrast, the decentralised subset method based on the Kalman filter has the flexibility to estimate the signals and aggregating the estimated signal, sequentially or partially at aggregating CSPs or ECSP, as shown in Fig. 2.7.

The second aspect to consider is the time delay, i.e. how much time that is required for processing and estimating the signals. The time delay depends on computational processing time and the time for collecting the signals. Here, the highest computational demanding operation is

Table 2.5: The front-haul capacity requirement.

	signals	total signals	covariance	total elements
level 4	$\mathbf{y}_l$	$\tau_C LN$	R_{lk}	$LKN^2/2$
level 3	$\breve{s}_{kl}$	$(\tau_C - \tau_P)LK$	a_{lk}	$LK + \frac{L^2K^2+LK}{2}$
level 1-2	$\breve{s}_{kl}$	$(\tau_C - \tau_P)LK$	-	-
subset	$\hat{\mathbf{s}}(\mathcal{L})$	$(\tau_C - \tau_P)LK$	$P(\mathcal{L})$	$LK^2/2$

to calculate the inverse of the covariance matrix and the sizes or the number of elements in the covariance matrix determines the computational complexity. The fully centralised level 4 method requires collecting all signals and estimating a full-sized covariance matrix at ECSP before the processing can start, the number of operations is in the order of $\sim (NL)^3$, as shown in Fig. 2.19, bars in red. Level 4 complexity increases rapidly with the number of CSPs hence has very high requirement on the hardware processing capabilities. However it has very low requirement on the CSPs since CSPs simply push everything to the ECSP. Level 4 is difficult to realise in practise partly because the time delay for collecting all measurements and partly high computational complexity to invert the very large covariance matrices when number of CSP is large. Level 1-3 process the signals at locally at each CSP, which put some requirement on the processor at CSPs. The number of operations is in the order of $\sim NL^3$, as show in Fig. 2.19, bars in yellow. However, the combining weights using level 1-3 are calculated at the ECSP which still requires to collect processed signals from all CSPs. By using the Kalman filter in the decentralised subset combining method, the signals can be processed locally at each CSP or collected at aggregating CSPs. The aggregating CSPs that are not overlapping can process the signals in parallel, which reduces the delay for processing and collecting the processed signals. However, the aggregating CSPs have certain requirements on processing capabilities. With the square-root implementation of the Kalman filter, see *Remark* 1, the number of operations is in the order of $\sim NLK(N+K)$, as shown in Fig. 2.19, bars purple and green for $K = 5$ and $K = 10$ respectively.

In Table 2.5, we summarise the front-haul capacity requirements. The column *signals* can be the received signals or the estimated signals at each CSP depending on the implementation methods. The column *total signals* contains the total sum of signals sent from of all CSPs to the ECSP for final decoding.

Corollary 3. *Applying the Kalman filter implementation, the required hardware capacity can be gathered at an ECSP or distributed among CSPs.*

In *Proposition* 3 it is shown that the Kalman filter provides equivalent results whether the signals are collected at an ECSP or at CSPs. This implies that the required hardware capacity to process the signals has different requirements whether the signals are processed centralised at ECSP or distributed at CSPs. The Kalman filter implementation enables parallelisation of the processing to many less capable processors at the distributed CSPs.

We know that the highest computational demanding task is to calculate the inverse of the covariance matrix. The size and number of elements of the covariance matrix determine the computational complexity. We show in Fig. 2.19 the computational complexity in terms of the number of operations required for calculating the inverse of the covariance matrix. The complexity increases with the number of CSPs, L, and antennas elements, N, For each number of combining CSPs, we show 4 bars: the two bars from the left are for centralised combining and two bars on the right are for decentralised Kalman filter combining. Comparing red and green bars, for example level 4 and sub-36, they have the same performance as we shown earlier, but Kalman filter combining

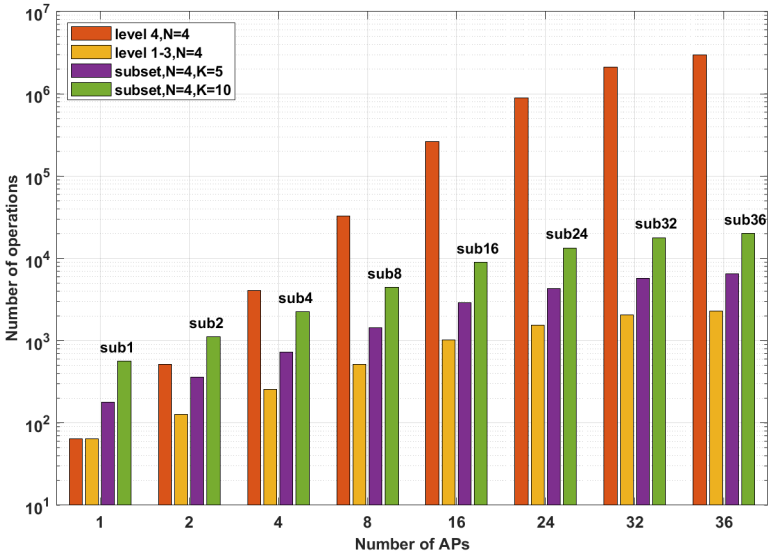


Figure 2.19: Calculation complexity

has a lower complexity when combining more than 4 CSPs. Compared to level 1-3, bars in yellow, Kalman filter combining has a superior performance with some increase in number of operations, but the Kalman filter can process the data in parallel and therefore reduce the time delay.

2.3.3 Numerical Comparisons

We evaluate the Kalman filter implementation by using the same deployment of a D-MIMO network in a 100×100 m^2 area as shown in Fig. 2.8, where a number of CSPs, denoted by L, are deployed on a square grid. A reference case is created where $L = 36$ CSPs, each CSP is equipped with $N = 4$ antenna elements and $K = 20$ UEs are randomly generated in the area. To be able to compare the work done in [15] and our implementations with Kalman Filter, we use the same channel propagation model and estimation as described in [15], which was based on 3GPP Urban Microcell model in [20], the pathgain is shown in Fig. 2.8. Except for the size of simulation area, we kept the most of parameters used by [15]. The system simulation parameters are summarised in Table 2.4.

The performance metrics used for comparing different uplink combining methods is the achievable SE [bit/s/Hz]. For the *centralised combining methods*, the simulations are based on the Matlab code provided by [15]. For the *decentralised combining methods*, two implementations were used in the simulations, one implementation based on MMSE using QR factorisation; the other one is the square-root implementation of the Kalman filter as described in section 2.3.1. Each simulation contains 1000 random channel realisations and 8000 random UEs samples. More results for varying the parameters can be found in [19].

In Fig. 2.20 the CDF of the four-levels centralised combing methods (level 4 to level 1), solid curves, are compared with decentralised subset method, dashed curves, sub-1 up to sub-36. We can see that level 4 is equivalent to sub-36 when all L CSPs are added into the subset, which numerically verifies the *Corollary 2*.

Fig. 2.21 shows the 10^{th} and 90^{th} percentile of SE as a function of decreasing in levels and increasing of CSPs in the subset on x-axis. The results from two implementations are shown, the

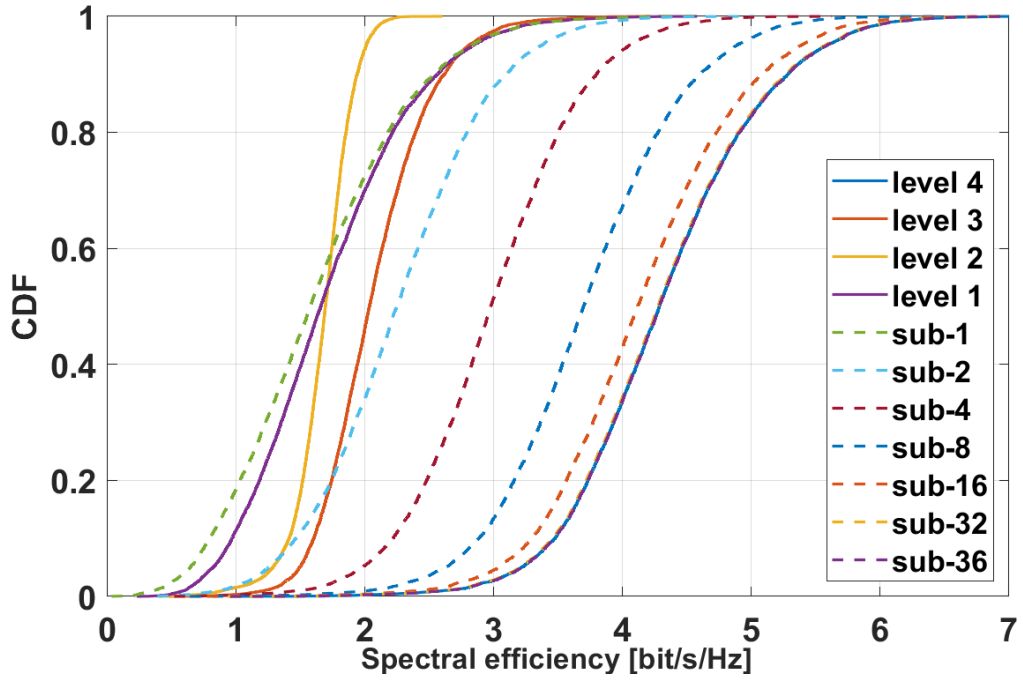


Figure 2.20: Centralised and decentralised combining methods.

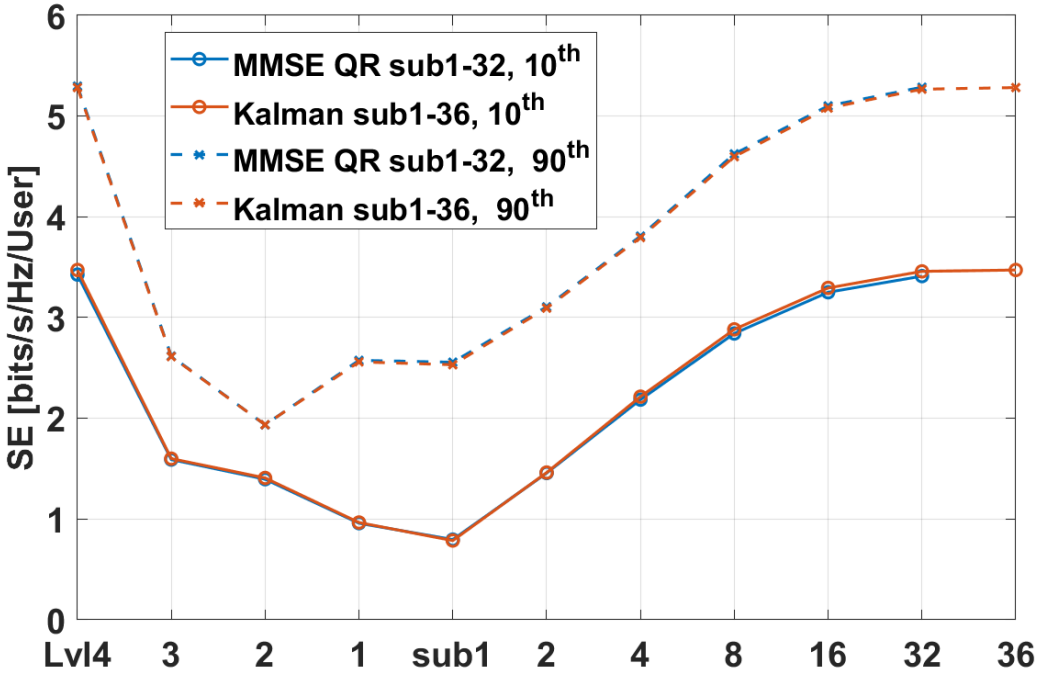


Figure 2.21: MMSE and Kalman Filter implementations.

blue curves based on MMSE with QR factorisation and the red curves based on the Kalman Filter. They are slightly different due to the different random seeds. Two conclusions can be drawn from the results: Firstly, the two implementations (MMSE with QR factorisation) give the same results; Secondly, sub-16 is almost as good as level 4 and sub-4 performs on par with level 1-3.

2.4 Conclusions

In Section 2.1, we discussed the different types of clock sources to synchronize the multiple CSPs deployed to serve the UEs in a RW architecture. One central clock equipped with a PLL located at a physical location is shared by different CSPs using coaxial cable connections, or each CSP can be equipped with a local PLL to generate the LO signals. For the latter case, a distributed PPS signal is shared to synchronize the PLL outputs at each CSP to achieve coherence between all LOs. In addition, we also explained another architecture which shares a low frequency clock that can be controlled over an Ethernet connection. We also discussed in detail about the scalability and complexities associated with each clock architecture. In addition, we briefly elucidated on a different type of architecture where each CSP can have its own clock source which can be used to synchronize multiple CSPs, either by an Ethernet connection, or OTA using an algorithm such as Beamsync.

In Section 2.2, we evaluate a decentralised subset method for the receiver combining in a D-MIMO network. The sub-set combining method is scalable in the sense that the more CSPs you deploy the more distributed processing resources and the more front-haul segments you get. This enables simultaneous and parallel forwarding of antenna data over different front-haul segments. It also enables simultaneous and parallel processing of UE uplink signals in different aggregation CSPs. The proposed method provides both good performance and low complexity compared to single CSP decoding (Level 1) or sequential CSP aggregation (Level 2 or Level 3). The method approaches the performance of fully centralised processing (Level 4) as the size of the cooperating set expands. Diminishing returns are noticed once the CSPs closest to the UE are included in the sub-set, which implies that for most UEs a small sub-set will provide sufficient performance. This ensures that only the most relevant antenna elements in the D-MIMO system are processed for each UE. This leads to less demanding processing (e.g., smaller matrix inversions) and less forwarding of information over the front-haul, resulting in reduced power consumption and reduced hardware cost.

Section 2.3 introduces the Kalman filter to estimate the uplink signals in a D-MIMO network. The Kalman filter provides an optimal estimate by minimising the estimation error. It can be applied to both centralised and decentralised processing and combining. When applied to the decentralised combining methods, it provides the flexibility to aggregate the estimates in different topologies, star, serial, parallel or grid. Applying the Kalman filter, the performance upper bound can be approached by incrementing the subset size stepwise. In the studied simulation scenario, where 36 CSPs are deployed in the D-MIMO network, we can reach 85% of the upper bound by including only 20% of total CSPs in the subset, and reach 95% of the upper bound by a subset with 16 CSPs, sub-16. With a Kalman filter implementation, it is simple to decide the most cost efficient trade-off between performance and implementation cost.

Chapter 3

Processing elements for efficient execution of distributed processing

In the previous chapter, a square root implementation of a fixed state Kalman filter for uplink signal estimation in a cell-free network has been proposed. As mentioned, the contact service points (CSPs) exchange information in the form of a matrix per coherence block and a vector for each received sample with each other to assist in the user equipments (UEs)' data estimation. However, it is assumed that the front-haul link between the CSPs is of infinite capacity. In a real-life network, this assumption is not the case. In this chapter, the complexity and hardware implementation of the algorithm in a cell-free network with daisy-chain topology with limited capacity front-haul links is considered where each transmitted scalar on the front-haul link between two CSPs should be quantized to a finite bit precision. In the following sections, we first overview the sequential signal estimation algorithms between the CSPs and then evaluate their computational complexity and numerical stability under the realistic assumption of limited-bandwidth front-haul links.

It is worth mentioning that in this chapter, the two recursive algorithms implemented sequentially in the CSPs are the same as two different implementations of fixed-state Kalman filtering introduced in the previous chapter.

3.1 Problem statement

We consider uplink transmission in a cell-free network consisting of K single antenna UEs, and L CSPs, each having N antennas. Assuming a Massive MIMO scenario, $NL \gg K$. Using OFDM, the received signal vector at CSP l can be modeled as follows:

$$\mathbf{y}_l = \mathbf{H}_l \mathbf{s} + \mathbf{n}_l. \tag{3.1}$$

In (3.1), $\mathbf{y}_l$ is the $N \times 1$ received vector at CSP l, $\mathbf{H}_l$ is the $N \times K$ channel matrix from all the UEs to the antennas of CSP l, and $\mathbf{n}_l$ is the additive noise vector $\mathbf{n}_l \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$. The columns of the matrix $\mathbf{H}_l$ are complex normal random vectors with correlation matrices based on the local scattering spatial correlation model in [18, Chapter 2]. The network-wide received vector, noise vector and channel matrix can be represented as $\mathbf{y} = [\mathbf{y}_1^T, \dots \mathbf{y}_L^T]^T$, $\mathbf{n} = [\mathbf{n}_1^T, \dots \mathbf{n}_L^T]^T$ and $\mathbf{H} = [\mathbf{H}_1^T, \dots \mathbf{H}_L^T]^T$ respectively. The network-wide received signal vector can be formulated as follows:

$$\mathbf{y} = \mathbf{H}\mathbf{s} + \mathbf{n}. \tag{3.2}$$

Knowing that covariance matrix of $\mathbf{n}$ is $\mathbf{E}\{\mathbf{n}\mathbf{n}^H\} = \sigma^2\mathbf{I}$, we normalize (3.2) by σ:

$$\frac{\mathbf{y}}{\sigma} = \frac{\mathbf{H}}{\sigma}\mathbf{s} + \tilde{\mathbf{n}}, \tag{3.3}$$

where $\tilde{\mathbf{n}} = \frac{\mathbf{n}}{\sigma}$.

If we assume that the transmitted signals are zero mean and mutually uncorrelated, i.e. $\mathbf{E}\{\mathbf{s}\mathbf{s}^H\} = p\mathbf{I}_K$, then we can map this prior knowledge to an additional equation as follows:

$$\mathbf{0} = \frac{1}{\sqrt{p}}\mathbf{s} + \mathbf{w}, \tag{3.4}$$

where $\mathbf{w}$ is a random vector with i.i.d. unit variance elements, i.e. $\mathbf{E}\{\mathbf{w}\mathbf{w}^H\} = \mathbf{I}_K$.

Bringing the two equations (3.3) and (3.4) together, we have:

$$\begin{bmatrix} \frac{\mathbf{y}}{\sigma} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \frac{\mathbf{H}}{\sigma} \\ \frac{1}{\sqrt{p}}\mathbf{I} \end{bmatrix} \mathbf{s} + \begin{bmatrix} \tilde{\mathbf{n}} \\ \mathbf{w} \end{bmatrix}. \tag{3.5}$$

3.2 Uplink signal estimation

To estimate UEs' signals, we apply LS estimation to (3.5), i.e,

$$\begin{aligned} \hat{\mathbf{s}} &= \arg\min_{\mathbf{s}} \|\frac{\mathbf{y}}{\sigma} - \frac{\mathbf{H}}{\sigma}\mathbf{s}\|^2 + \frac{1}{p}\|\mathbf{s}\|^2 \\ &\overset{a}{=} \arg\min_{\mathbf{s}} \|\mathbf{y} - \mathbf{H}\mathbf{s}\|^2 + \frac{\sigma^2}{p}\|\mathbf{s}\|^2, \end{aligned} \tag{3.6}$$

where a comes from the fact that multiplying the objective function of an optimization problem with a positive scalar does not change the optimal point. We do not consider the channel estimation problem as it is out of the scope of this chapter. Hence, we assume that the channel matrix is known exactly, which is possible in case of having a unique pilot per UE and high enough transmission power during pilot transmission. The solution to the problem (3.6) is:

$$\hat{\mathbf{s}} = (\mathbf{H}^H\mathbf{H} + \frac{\sigma^2}{p}\mathbf{I}_K)^{-1}\mathbf{H}^H\mathbf{y}. \tag{3.7}$$

The matrix $\mathbf{\Gamma} = (\mathbf{H}^H\mathbf{H} + \frac{\sigma^2}{p}\mathbf{I}_K)^{-1}$ is the inverse of the $K \times K$ regularized Gram matrix. The solution defined in (3.7) is also referred to as the regularized zero-forcing (RZF) solution where $\delta = \frac{\sigma^2}{p}$ acts as the regularization parameter to avoid singular matrix inversion. However, here this regularization results from the inclusion of the prior knowledge of the UEs' transmit power in the system model in section 3.1.

Another way of solving the problem defined in (3.6) is based on QR decomposition of the regularized channel matrix:

$$\begin{bmatrix} \mathbf{H} \\ \frac{\sigma}{\sqrt{p}}\mathbf{I}_K \end{bmatrix} = \mathbf{Q}\begin{bmatrix} \mathbf{R} \\ \mathbf{0}_{NL\times K} \end{bmatrix} = \tilde{\mathbf{Q}}\mathbf{R}, \tag{3.8}$$

where $\tilde{\mathbf{Q}} = \mathbf{Q}_{[:,1:K]}$ (Subscript $[:, 1:K]$ means all the rows and the first K columns). Matrix $\mathbf{Q}$ is an $(NL+K) \times (NL+K)$ orthogonal matrix, i.e. $\mathbf{Q}^H\mathbf{Q} = \mathbf{I}_{(NL+K)}$ and the $K \times K$ upper

triangular matrix $\mathbf{R}$ is the square root of the regularized channel matrix i.e.

$$\begin{bmatrix} \mathbf{H}^H & \frac{\sigma}{\sqrt{p}}\mathbf{I}_K \end{bmatrix} \begin{bmatrix} \mathbf{H} \\ \frac{\sigma}{\sqrt{p}}\mathbf{I}_K \end{bmatrix} = \mathbf{R}^H\mathbf{R}. \tag{3.9}$$

substituting (3.8) and (3.9) in (3.7), (3.7) leads to

$$\hat{\mathbf{s}} = (\mathbf{R})^{-1}\tilde{\mathbf{Q}}^H_{[1:NL,:]}\mathbf{y}. \tag{3.10}$$

In this chapter, QR decomposition is implemented using givens rotations.

Computing (3.7) and (3.10) in the CPU requires high local computational complexity. The $K \times K$ matrix inversion in (3.7) is of order $\mathcal{O}(K^3)$ and (3.10) requires the computation of the QR decomposition of the $(NL + K) \times K$ regularized channel matrix which is of order $\mathcal{O}(NLK^2)$. In order to avoid the high local computational complexity and high inter-connection data rate to the CPU, we adhere to recursive algorithms and distributed implementations. Two possible recursive algorithms that can be distributed easily are discussed below.

3.2.1 Standard recursive least squares (SRLS) algorithm

The SRLS is the recursive procedure to compute the solution defined in (3.7) based on the recursive computation of the inverse regularized Gram matrix. The steps of the algorithm are as follows:

- Once per coherence block, CSP l updates $\mathbf{\Gamma}_l$ based on its local channel matrix and $\mathbf{\Gamma}_{l-1}$ that it receives from the previous CSP:

$$\mathbf{\Gamma}_l = \mathbf{\Gamma}_{l-1} - \mathbf{\Gamma}_{l-1}\mathbf{H}_l^H(\mathbf{I}_N + \mathbf{H}_l\mathbf{\Gamma}_{l-1}\mathbf{H}_l^H)^{-1}\mathbf{H}_l\mathbf{\Gamma}_{l-1}^H. \tag{3.11}$$

- For each received data vector $\mathbf{y}_l$ in one coherence block, CSP l updates the estimate of the UEs' signals based on the signal estimate vector $\hat{\mathbf{s}}_{l-1}$ that it receives from the previous CSP:

$$\hat{\mathbf{s}}_l = \hat{\mathbf{s}}_{l-1} + \underbrace{\mathbf{\Gamma}_l\mathbf{H}_l^H}_{\mathcal{K}}(\mathbf{y}_l - \mathbf{H}_l\hat{\mathbf{s}}_{l-1}). \tag{3.12}$$

  Note that $\mathcal{K}$ is the so-called Kalman gain introduced in the previous chapter.

- Then it sends $\mathbf{\Gamma}_l$ and $\hat{\mathbf{s}}_l$ to CSP $l + 1$.

It is worth mentioning that in any CSP in the sequence, the updated estimates of the UEs' signals are the solution to a smaller LS problem (i.e. LS problem with the observations from the first antenna in the first CSP until the last antenna of the corresponding CSP).

3.2.2 QR decomposition based recursive least squares (QR-RLS) algorithm

In QR-RLS, the information that is propagated from CSP $l-1$ to CSP l is a vector of size K as well as the square root matrix of size $K \times K$. The steps of the recursive algorithm are summarized as follows:

- Once per coherence block, CSP l computes the QR decomposition of matrix $\mathbf{U}_l = \begin{bmatrix} \mathbf{R}_{l-1} \\ \mathbf{H}_l \end{bmatrix}$ $= \tilde{\mathbf{Q}}_l\mathbf{R}_l$, where $\mathbf{R}_{l-1}$ is the square root matrix that it receives from the previous CSP.

- For each received signal vector $\mathbf{y}_l$, CSP l computes the vector $\mathbf{z}_l = \tilde{\mathbf{Q}}_l^H \begin{bmatrix} \mathbf{z}_{l-1} \\ \mathbf{y}_l \end{bmatrix}$ where $\mathbf{z}_{l-1}$ is the $K \times 1$ vector that it receives from previous CSP.
- Then it sends $\mathbf{R}_l$ and $\mathbf{z}_l$ to CSP $l+1$.

In the last CSP, $\hat{\mathbf{s}}_L = \mathbf{R}_L^{-1}\mathbf{z}_L$ is computed. As matrix $\mathbf{R}_L$ is an upper triangular matrix, $\hat{\mathbf{s}}_L$ can be computed (efficiently without matrix inverse) using back substitution.

Note that, similar to the SRLS, also for the QR-RLS, we can have the solution to the smaller LS problem at any CSP.

3.2.3 Low precision implementation of the recursive algorithms

It is well known that the SRLS algorithm is susceptible to numerical instability in the case of a low-precision implementation, quantizing the exchanged matrix and vector between two subsequent CSPs with a low number of bits. On the other hand, QR-RLS is a square root variant of the SRLS, which is well-known for its numerical stability in the case of a low precision implementation [25]. Unlike [23], in our simulation, we consider the link connecting two CSPs to have limited bandwidth. Therefore, the data that needs to be exchanged between the CSPs in the two recursive algorithms have to be quantized with a finite number of bits.

A uniform quantizer is considered using fixed-point arithmetic in MATLAB at each CSP. The number of bits (word length) used to represent the quantization levels is W. As the range of the elements of the matrices and vectors to be exchanged is unknown, we consider a scaled version of the quantizer. Suppose that we want to quantize the elements of one particular matrix $\mathbf{A}$. The smallest and largest scalar in $\mathbf{A}$ (can be real or imaginary part of an element of $\mathbf{A}$) are a_{min} and a_{max} respectively. The range of the elements is defined as $r_d = a_{max} - a_{min}$. So we have a range r_d, and we can have 2^W quantization level to represent this range. Therefore, we build a scaled quantizer with limits $[a_{min}, a_{max}]$, the first level representing a_{min} and the last level representing a_{max} and the remaining $2^W - 2$ levels partition the r_d into $2^W - 1$ equal segments. The precision or the distance between two subsequent quantization levels becomes

$$\Delta = S = \frac{r_d}{2^W - 1}. \tag{3.13}$$

For illustration purposes, a schematic figure of the scaled quantizer is shown in Fig. 3.1. Note that the values for a_{min} and a_{max} can be positive or negative.

By scaling the quantizer based on the range of the matrix/vector that we want to quantize, we make sure that overflow/underflow never appears.

It is also worth mentioning that CSP l also needs to send the parameters S and a_{min} (or a_{max}) to CSP $l+1$ so CSP $l+1$ is able to reconstruct the data. We assume that these two parameters are sent with high enough precision.

3.2.4 Initialization

The recursive algorithms defined in subsections 3.2.1 and 3.2.2 need initialization in the first CSP. The choice of initialization affects the error propagation, especially in the low-precision implementation of the SRLS. Let's first consider the infinite precision implementation of the SRLS algorithm. Assume that the matrix $\mathbf{\Gamma}_0$ in (3.11) is initialized as a scaled identity matrix, i.e. $\mathbf{\Gamma}_0 =$

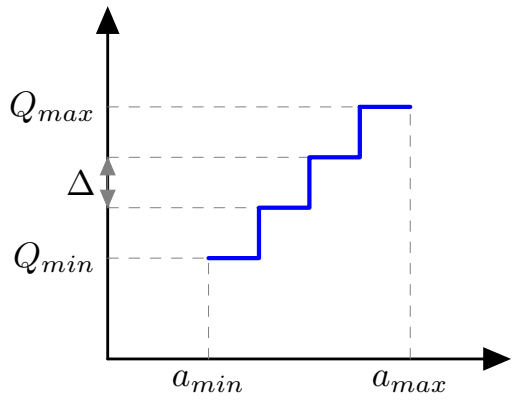


Figure 3.1: Scaled fixed-point quantizer. $a_{min}, a_{max}, Q_{min}, Q_{max} \in \mathbb{R}$.

$\frac{1}{\delta}\mathbf{I}_K$. After iteration l in CSP l, the local UEs' signals estimate, $\hat{\mathbf{s}}_l$ is the solution to the optimization problem defined as:

$$\hat{\mathbf{s}}_l = \arg\min_{\mathbf{s}} ||\mathbf{y}_{[1:Nl]} - \mathbf{H}_{([1:Nl],:)}\mathbf{s}||^2 + \delta||\mathbf{s}||^2. \tag{3.14}$$

Based on (3.6) and (3.14), we would have chosen $\delta = \frac{\sigma^2}{p}$ in the infinite precision implementation which can be a small number. However, in a finite precision implementation, selecting an extremely small δ may make the exchanged matrices among CSPs ill-conditioned, which may result in early divergence of the SRLS algorithm. On the other hand, selecting an extremely large δ biases the sequential refinement of the UEs' signals estimate.

Another way of looking at the different choices of δ is through its possible effect on the range of the exchanged matrices between CSPs. Selecting a small δ makes the range of the exchanged data large and, as a consequence, lowers the precision of the quantization. This, in turn, results in the early divergence of the SRLS algorithm.

Based on the discussion in the previous paragraph, we may not be able to initialize the SRLS algorithm with $\mathbf{\Gamma}_0 = \frac{p}{\sigma^2}\mathbf{I}$. In this case, we initialize the $\mathbf{\Gamma}_0 = \frac{1}{\delta}\mathbf{I}$ with a large enough value of δ and then in the last CSP, after updating the inverse of the regularized Gram matrix to the $\mathbf{\Gamma}_L$ based on the local channel matrix, we add two extra processing steps which are a so-called down-dating operation and updating based on prior knowledge to convert the regularization parameter from δ to $\frac{\sigma^2}{p}$. We refer to [26] for more information on the down-dating operation in the RLS algorithm. As it is also seen in the simulation section, the down-dating operation can be numerically unstable, especially for extremely large δ.

As for the QR-RLS algorithm, it seems that QR-RLS is not sensitive to the choice of initialization owing to its inherent numerical stability, hence for QR-RLS, we can initialize $\mathbf{R}_0 = \sqrt{\frac{\sigma^2}{p}}\mathbf{I}_K$ in the first CSP.

3.3 Simulation results

This section provides the simulation results for a cell-free network with a daisy-chain topology. Unless otherwise stated, we assume that there are $L = 25$ CSPs, each having $N = 4$ antennas serving $K = 5$ UEs. We consider the path loss (in dB) model of an urban microcell with 2GHz carrier frequency [20]:

$$PL = C - \alpha 10 \log_{10} \frac{d_{kl}}{1m}. \tag{3.15}$$

The constant value C is set to -30.5dB and it is the path loss between CSP l and UE k at the distance $d_{kl} = 1$ meter. The transmit power of the UEs is $p = 20$dBm and receiver noise power is $\sigma^2 = -85$dBm. The performance metric considered is the squared error between the transmitted signal by a UE and its estimate, averaged over UEs, samples in one coherence block, different coherence blocks and different locations of the UEs.

The finite precision recursive algorithms are compared against the infinite-precision central RZF solution in (3.7).

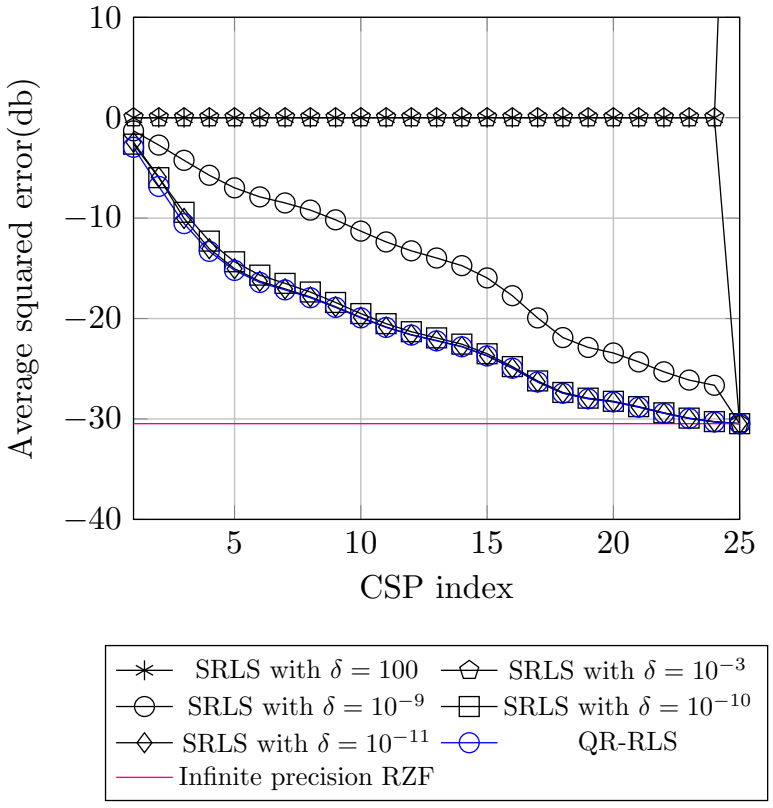


Figure 3.2: Average squared error in CSPs, $L = 25$, $N = 4$, $K = 5$, $W = 32$.

Fig. 3.2 shows the performance of the two recursive algorithms using a 32 bit scaled quantizer. As observed, SRLS (with a small enough δ) has the same performance as QR-RLS.

However, Fig. 3.3 shows that by decreasing the precision to 16 bits, SRLS diverges in the case of small δ. The smaller the δ, the earlier the divergence happens.

In Fig. 3.4, the number of UEs is increased to $K = 15$, but the quantization word length remains the same as in Fig. 3.3. We observe that assuming different values for δ, the SRLS starts diverging earlier compared to the case of $K = 5$. By increasing the number of UEs, the condition number of the inverse regularized Gram matrix at each CSP increases, which will affect the error propagation of the quantized exchanged matrix adversely. Thus we need to consider more regularization for the problem at hand to have a well-conditioned exchanged matrix and less error propagation due to a finite precision implementation.

It is worth mentioning that in Fig. 3.2, the divergence behavior at the last CSP for the two large regularization parameters is mainly due to the down-dating operation. In other words, with a large regularization parameter, the inverse of the regularized Gram matrix does not get updated through the sequence of CSPs, and in the last CSP, it is a scaled identity matrix (almost) the same

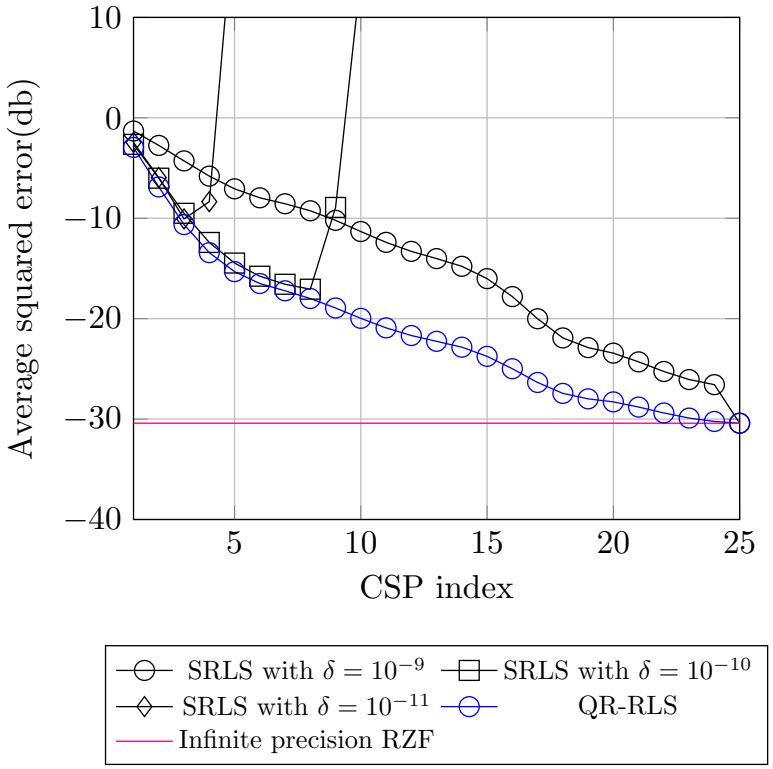


Figure 3.3: Average squared error in CSPs, $L = 25$, $N = 4$, $K = 5$, $W = 16$.

as the initial matrix in the first CSP, and then the matrix inversion in the down-dating operation will be the inversion of an almost zero matrix.

In Fig 3.5, the divergence behavior of the SRLS algorithm under the assumption of $\delta = 10^{-10}$ and $\delta = 10^{-11}$ and $K = 5$ and $K = 25$ is depicted via a histogram plot. The plot shows the probability of divergence in each CSP in case divergence happens, measured over different realizations of the UEs' locations. The main message of the plot is that, by increasing the number of UEs, the mass concentration of the histogram will be shifted toward the first CSP. This observation validates the claim that by increasing the number of UEs, SRLS divergence happens earlier compared to the case with a smaller number of UEs.

In summary, to implement SRLS with low precision, initialization gets a delicate matter, and it gets more crucial when the number of UEs increases. These considerations make the use of SRLS quite impractical, as the number of CSPs to consider should vary as a function of the number of active UEs. The QR-RLS algorithm is not sensitive to the choice of regularization parameter and has a numerically stable behavior through the chain of the CSPs and, as a consequence, lower error on the UEs' signals estimate.

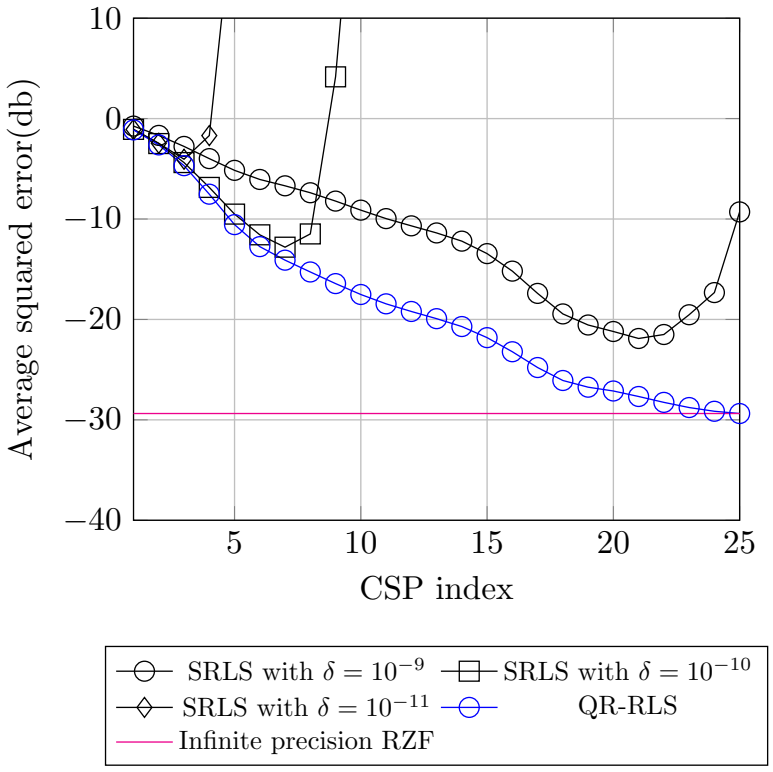


Figure 3.4: Average squared error in CSPs, $L = 25$, $N = 4$, $K = 15$, $W = 16$.

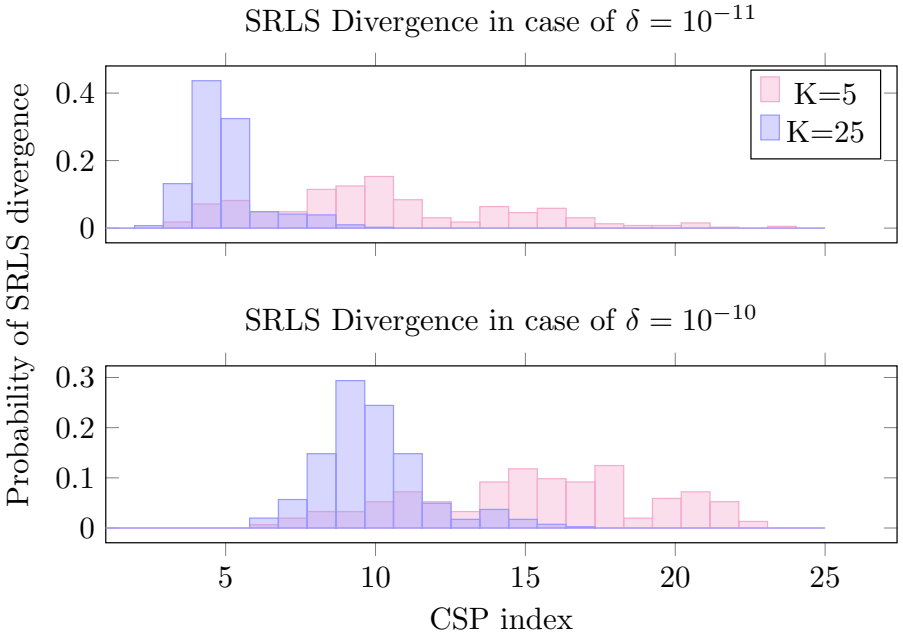


Figure 3.5: Probability mass function of SRLS divergence in each CSP, $L = 25$, $N = 4$, $W = 16$.

Chapter 4

RadioWeaves (RW) topology and government

When designing a RadioWeaves (RW) infrastructure, there are many different aspects to take into account. In this chapter we discuss how the overall architecture/topology influences the performance, cost, and flexibility, related to distributed processing across the infrastructure. We start by discussing considerations on a high level, followed by a quantitative analysis of requirements on front-haul capacity between, and memory requirements in contact service points (CSPs). Finally, we discuss the processing required to compute and optimize the allocation of distributed resources for federations to support different services.

4.1 Infrastructure architecture/topology considerations

Distributed processing is affected in several ways by the architecture of a RW, since it puts limitations on the possible routes in the back-haul for message passing between distributed processing units. As a general statement, the more connections we have between processing units, the shorter the routes for message passing and the lower the latency such message passing incurs on the total processing latency. However, a fully-connected front-haul network is neither practically, nor economically, feasible. On top of this come considerations regarding, total energy efficiency, reliability, etc. A good RW infrastructure design therefore has to strike a good balance between pros and cons – a task that is hugely complex and depends on many parameters, such as use case requirements. For tractability, we present a simplified high-level view of some of the main considerations. An initial discussion was presented in REINDEER D2.1, Chapter 5, and is here expanded with more details.

Before we start the discussion, let us introduce two distance measures that will show up frequently in the following, loosely defined as:

Front-haul distance (FD): The front-haul distance between any two processing nodes describes how far from each other the nodes are, either in terms of the number of hops in the route between them or in terms of the latency introduced by message passing from one to the other. The further two nodes are in terms of Front-haul distance (FD), the higher the latency in message passing between them.

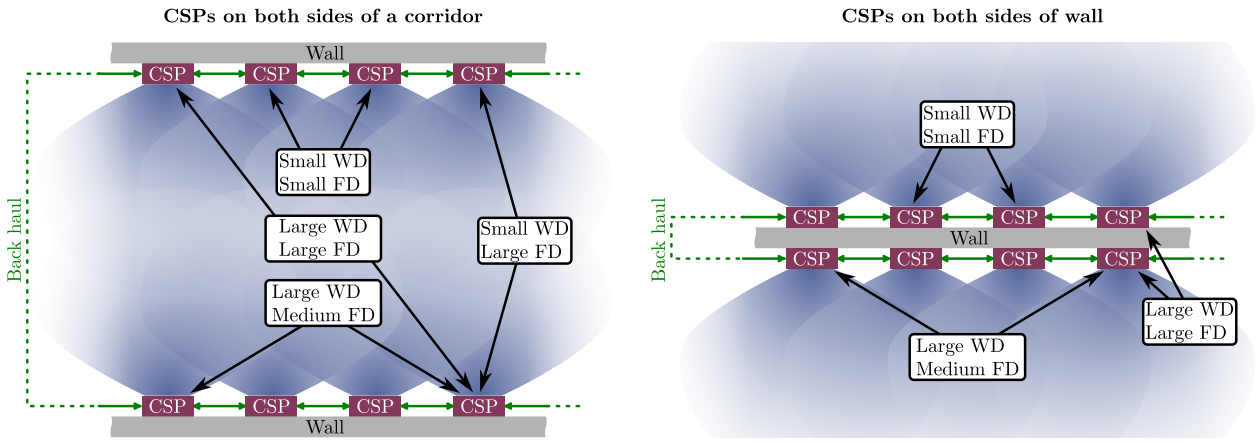


Figure 4.1: Two deployment scenarios with CSPs on a Daisy-chain front-haul. Coverage area of each CSP indicated in blue.

Wireless distance (WD): The wireless distance between any two nodes describes the probability that a certain distributed algorithm, in a certain use case, would benefit from co-processing wireless data/signals from the two nodes. The further two nodes are in Wireless distance (WD), the less likely that co-processing of data will be beneficial to the overall performance of said distributed algorithm.

Let us illustrate these two distance measures in Fig. 4.1, where we can see two different deployment scenarios with CSPs on a Daisy-chain front-haul. In both scenarios the FD depends on how far two CSPs are on the Daisy chain, while the WD depends on if the indicated coverage areas of the CSPs overlap or not in the respective scenarios. Wireless data/signals from CSPs with a larger overlap of their respective coverage areas can benefit from coordinated processing and therefore have a lower WD than CSPs with less or no overlap of coverage areas.

From a performance point of view, an infrastructure deployment should be done so that CSPs with a low WD, between which co-processing of wireless data/signals are likely to benefit performance, should also have a low FD, making message passing more efficient, in terms of less data in the front-haul, lower incurred latency and better overall energy efficiency. Going back to the illustration in Fig. 4.1, we can see that in the case of CSPs on both sides of a wall (right), the Daisy-chained front-haul provides a small FD between all CSPs with small WD. In the case of CSPs deployed on both sides of a corridor (left), the situation is different. CSPs facing each other on opposite sides of the corridor have a small WD, while the corresponding FD is large. To remedy this situation we can, e.g., introduce additional front-haul links across the corridor, as shown in Fig. 4.2, to lower FD between CSPs with low WD on opposite sides. This will reduce the data rate in front-haul links and provide additional opportunities for co-processing data/signals without excessive latency in the front-haul, leading to, e.g., improved wireless data rates and/or higher precision positioning. However, as mentioned above, there will be restrictions on, e.g., deployment cost, putting practical limits in a real environment both on the number of links in the front-haul and where such links can be deployed.

While more formal definitions of FD and WD, as well as their practical use in optimizing RW deployment in any given scenario, are formidable tasks beyond the scope of this deliverable, the loose definitions given above provide valuable insight into some of the main considerations when designing a RW infrastructure. Taking the first steps in the direction of infrastructure design guidelines, we continue with more focused and detailed analysis of distributed processing requirements and elaborate on dynamic allocation of processing resources in an existing deployment.

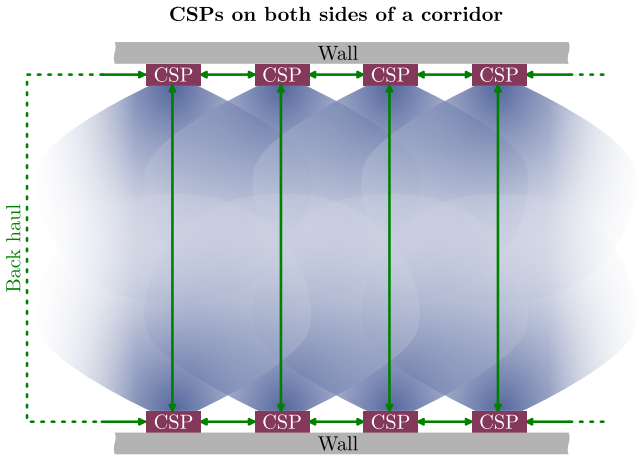


Figure 4.2: Introducing additional front-haul links (compared to Fig. 4.1) across the corridor to lower FD between CSPs with low WD on opposite sides.

4.2 Distributed processing and federations

In this section we mainly discuss the required memory capacity for storing channel matrices and buffering radio frames as well as the required bandwidth for data exchange between processing clusters. The specific requirements will depend on many different aspects, including the processing algorithms selected (e.g., detection and precoding algorithms), the topology of the RW infrastructure, the mapping of the algorithms to the hardware architecture, the number of users being served, the structure and format of the radio frames, etc.

As a starting point, we will select a group of distributed processing algorithms for uplink detection and the corresponding topology that have been discussed in REINDEER Deliverable 2.1 and Deliverable 2.2. Then we will abstract the general expression of the required memory capacity and data exchange bandwidth for the selected algorithms and topologies. We will start with more straightforward topologies, e.g., Daisy-chain and tree, which will serve as the basis to the discussion in the end of this section on how the FD and WD would affect the memory capacity and data exchange bandwidth requirements. There are more classes of algorithms that are being developed in REINDEER (e.g., real-time federation resource scheduling) whose memory and data exchange bandwidth requirement will be addressed when they are at a more mature development stage.

4.2.1 Selected algorithms and topology

For completeness, here we briefly summarize the distributed algorithms and topologies that have been selected for analysis. The traditional centralized solution is also included here as the baseline for the analysis.

4.2.1.1 Zero forcing (ZF) detection in a centralized architecture (star topology)

In the traditional centralized architecture, CSPs send the channel estimates ($\mathbf{H}_i$) and the received signals ($\mathbf{y}_i$) to a central processing unit (CPU), e.g., edge computing service point (ECSP) in the RW infrastructure, [27] where ZF detection is performed as

$$\widehat{\mathbf{x}} = (\mathbf{H}^H\mathbf{H})^{-1}\mathbf{H}^H\mathbf{y}. \tag{4.1}$$

Fig. 4.3 shows the high-level architecture of the centralized solution.

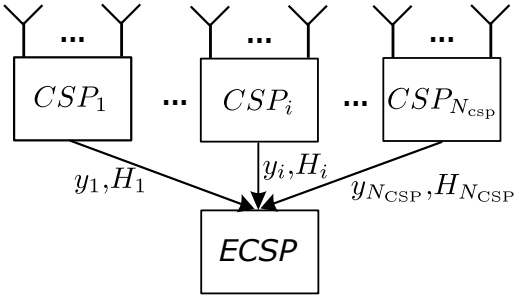


Figure 4.3: Centralized architecture with all the CSPs connecting to a centralized processing unit (CPU), e.g., ECSPs, in a "star" topology.

4.2.1.2 Recursive least squares (RLS) algorithm mapped to a Daisy-chain topology

The RLS algorithm finds the least squares (LS) solution to the problem $\min_{\mathbf{x}} \|\mathbf{y}-\mathbf{H}\mathbf{x}\|^2$ to perform zero-forcing multi-user detection in a distributed and serial manner [28]. More specifically, the ith CSP updates the estimation of the transmitted vector $\widehat{\mathbf{x}}_i$ with the information of local received signal vector $\mathbf{y}_i$, local channel matrix $\mathbf{H}_i$, and the residual matrix $\mathbf{R}_{i-1}$ from the $(i-1)$th CSP:

$$\widehat{\mathbf{x}}_i = \widehat{\mathbf{x}}_{i-1} + \mathbf{K}_i(\mathbf{y}_i - \mathbf{H}_i\widehat{\mathbf{x}}_{i-1}), \tag{4.2}$$

where

$$\mathbf{K}_i = \mathbf{R}_{i-1}\mathbf{H}_i^H(\sigma_n\mathbf{I} + \mathbf{H}_i\mathbf{R}_{i-1}\mathbf{H}_i^H)^{-1}. \tag{4.3}$$

and updates the residual matrix according to:

$$\mathbf{R}_i = (\mathbf{I} - \mathbf{K}_i\mathbf{H}_i)\mathbf{R}_{i-1}, \tag{4.4}$$

The estimated vector and the residual matrix are initialized to $\widehat{\mathbf{x}}_0 = \mathbf{0}$ and $\mathbf{R}_0 = \mathbf{I}$. The RLS algorithm can be mapped to a RW architecture in which CSPs are cascaded in a Daisy-chain as illustrated in Fig. 4.4

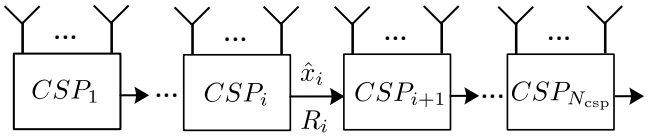


Figure 4.4: RLS algorithm mapped to a Daisy-chain topology.

4.2.1.3 Kalman Filter Combining mapped to a tree topology

The decentralized Kalman filter combing algorithm presented in Section 2.2 shares a similar concept with the RLS algorithm. These algorithms can be applied to different topologies beyond Daisy-chain. As depicted in Fig. 2.18 and Fig. 4.5, by aggregating measurements from N CSPs in parallel, a multi-level N-way tree topology can be formed. One of the benefits of the tree topology, comparing to the Daisy-chain topology, is a reduced processing latency by processing several CSPs at the same time.

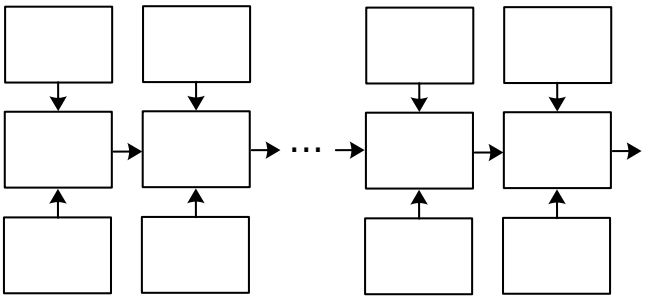


Figure 4.5: A 2-D CSP array is connected in a multi-level tree topology.

4.2.1.4 System and design parameters for analysis

In Table 4.1, we list the system and design parameters (and the corresponding definitions) that will be used in the memory and data exchange analysis. More specifically, we assume a RW infrastructure consisting of N_{csp} CSPs and each CSP is equipped with M_{csp} antenna elements and RF chains. The infrastructure is serving N_{ue} signal-antenna users at the same time and frequency resource. The system operates in a TDD manner with orthogonal frequency-division multiplexing (OFDM) modulation. We assume the channel matrix is estimated per resource block, which has in total $N_{\text{rb}}^{\text{f}} N_{\text{rb}}^{\text{t}}$ time-frequency resource grids. For simplicity, we assume all the complex-valued data has the same word-length of w_{b}.

Table 4.1: System and design parameters used for memory and data exchange analysis

Parameter	Definition
M_{csp}	number of antenna elments and RF chains per CSP
N_{csp}	number of CSPs in the RW
N_{sub}	number of sub-carriers per OFDM symbol
T_{ofdm}	time duration of an OFDM symbol
w_{b}	bit-width per sampled data (real + imag)
N_{ue}	number of users served at the same time and frequency resource
N_{rb}^{f}	number of sub-carriers per resource block
N_{rb}^{t}	number of OFDM symbols per resource block

4.2.2 Required memory size and data exchange bandwidth

4.2.2.1 Data exchange bandwidth analysis

Centralized ZF detection: During the channel estimation phase, all the CSPs send the local channel estimation ($\mathbf{H}_i$) to the ECSP, resulting an aggregated data exchange bandwidth at the ECSP being

$$\text{BW}_{\text{zf}}^{\text{form}} = \frac{w_{\text{b}} N_{\text{ue}} N_{\text{csp}} M_{\text{csp}} N_{\text{sub}}}{T_{\text{ofdm}} N_{\text{rb}}^{\text{f}} N_{\text{rb}}^{\text{t}}}. \tag{4.5}$$

The ECSPs send the received signal vector $\mathbf{y}_i$ to the ECSP during the detection phase and the corresponding aggregated data exchange rate at the ECSP is

$$\text{BW}_{\text{zf}}^{\text{fil}} = \frac{w_{\text{b}} N_{\text{csp}} M_{\text{csp}} N_{\text{sub}}}{T_{\text{ofdm}}}. \tag{4.6}$$

The RLS algorithm with a Daisy-chain topology: The execution of the RLS algorithm can be divided into two phases, the formulation phase and the filtering phase. During the formulation phase, the matrices $\mathbf{K}_i$ and $\mathbf{R}_i$ are calculated, according to (4.3) and (4.4), once we have the updated channel matrix $\mathbf{H}_i$, e.g., per resource block. The information needs to be exchanged between CSPs during the formulation phase is the residual matrix $\mathbf{R}_i$ of size $N_{\text{ue}} \times N_{\text{ue}}$, with the corresponding (average) bandwidth requirement per Daisy-chain link being

$$\text{BW}_{\text{rls}}^{\text{form}} = \frac{w_{\text{b}} N_{\text{ue}}^2 N_{\text{sub}}}{T_{\text{ofdm}} N_{\text{rb}}^{\text{f}} N_{\text{rb}}^{\text{t}}}. \tag{4.7}$$

During the filtering phase (here consider only uplink detection), the estimated $\widehat{\mathbf{x}}_i$ needs to be passed serially through CSPs, with the corresponding (average) bandwidth requirement per Daisy-chain link as following:

$$\text{BW}_{\text{rls}}^{\text{fil}} = \frac{w_{\text{b}} N_{\text{ue}} N_{\text{sub}}}{T_{\text{ofdm}}}. \tag{4.8}$$

The entire RW infrastructure contains $(N_{\text{csp}} - 1)$ such links.

Kalman filter combining with a tree topology: The amount of information that needs to be exchanged between the nodes in a tree topology is the same as that in the Daisy-chain topology, thereby the per link data exchange bandwidth requirement in the tree topology can also be formulated as (4.7) and (4.8).

4.2.2.2 Memory size requirement analysis

A storage of data is needed in case that there is a time interval between the data generation and data consumption. The specific requirement on the storage size will depend on many aspects including the radio frame structure, the algorithms, the scheduling of the operations, the processing hardware used, etc. Here we provide a high level analysis of storage requirements, ignoring implementation details, e.g., required pipeline registers, cache memory and register files for intermediate results. We also don't go to the details of different storage hardware, e.g., resisters, on-chip static random-access memory (SRAM), off-chip dynamic random-access memory (DRAM), yet instead use memory as a general term.

The structure of the used radio frame has a significant impact on the required memory size. As an illustration, we discuss two types of frame structures shown in Fig. 4.6. For the type (a) frame structure, uplink pilots are transmitted after the uplink data, meaning that a data buffer is required to store the first two uplink data OFDM symbols. Depending on the processing latency (e.g., channel matrix pre-processing can be completed before the uplink data arrives), a data buffer may not be needed for the type (b) frame structure. The following analysis will be based on the type (b) frame structure to focus on the required memory due to different algorithms and topologies selected for the RW infrastructure.

Centralized ZF detection: The ECSP stores the channel matrix $\mathbf{H}$ of size $M_{\text{csp}} N_{\text{csp}} \times N_{\text{ue}}$ and the inversion of the Gram matrix of size $N_{\text{ue}} \times N_{\text{ue}}$. Thereby the total required memory size at ECSP is

$$\text{MS}_{\text{zf}}^{\text{tot}} = \frac{w_{\text{b}} (N_{\text{ue}} M_{\text{csp}} N_{\text{csp}} + N_{\text{ue}}^2) N_{\text{sub}}}{N_{\text{rb}}^{\text{f}}}. \tag{4.9}$$

The RLS algorithm with Daisy-chain topology: Each CSP needs to store the local channel matrix

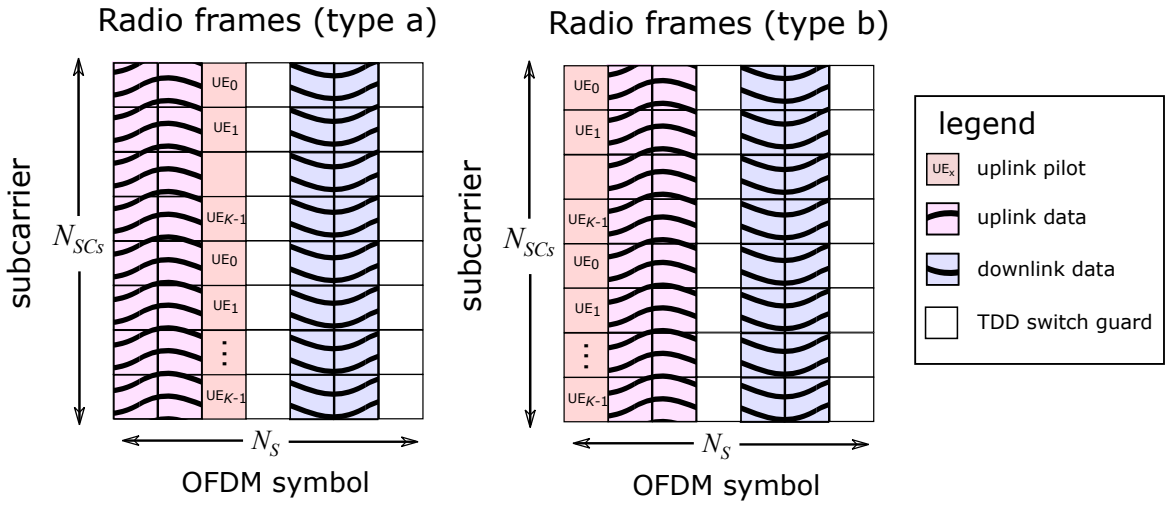


Figure 4.6: Illustration of different frame structures.

$\mathbf{H}_i$ of size $M_{\text{csp}} \times N_{\text{ue}}$ and $\mathbf{K}_i$ of size $N_{\text{ue}} \times M_{\text{csp}}$. The required memory size for one CSP is then given by

$$\text{MS}_{\text{rls}}^{\text{form}} = \frac{2w_{\text{b}}N_{\text{ue}}M_{\text{csp}}N_{\text{sub}}}{N_{\text{rb}}^{\text{f}}}. \tag{4.10}$$

One of the drawbacks of the RLS algorithm is the processing latency, i.e., during the filtering phase, the last CSP in the Daisy-chain needs to wait for the calculation and propagation of $\widehat{\mathbf{x}}_i$ through the entire chain. Thereby, data buffers are needed to store the received uplink data $\mathbf{y}_i$. The precise estimation of the buffer size will need the knowledge of the processing and data exchange latency. Here we assume that the latency is a fraction of the OFDM symbol duration with a factor of F_{latency}. The required memory size to buffer the received vector $\mathbf{y}_i$ per CSP is then:

$$\text{MS}_{\text{rls}}^{\text{filt}} = w_{\text{b}}M_{\text{csp}}N_{\text{sub}}F_{\text{latency}}. \tag{4.11}$$

For simplicity, (4.11) assumes that all the CSPs buffer the same amount of received vectors $\mathbf{y}_i$. In practical implementations, CSPs at the earlier stages of the Daisy-chain can have smaller size buffers for $\mathbf{y}_i$ comparing to CSPs at the later stages in the chain.

The total memory size needed for the RW is

$$\text{MS}_{\text{rls}}^{\text{tot}} = (\text{MS}_{\text{rls}}^{\text{form}} + \text{MS}_{\text{rls}}^{\text{filt}})N_{\text{csp}} \tag{4.12}$$

Kalman filter combining with a tree topology: The required memory size to store the local channel matrix $\mathbf{H}_i$ per CSP is the same as in the Daisy-chain case, i.e.,

$$\text{MS}_{\text{kf}}^{\text{form}} = \frac{2w_{\text{b}}N_{\text{ue}}M_{\text{csp}}N_{\text{sub}}}{N_{\text{rb}}^{\text{f}}}. \tag{4.13}$$

On the other hand, the processing latency in a tree topology can be significantly reduced comparing to the Daisy-chain topology, $F_{\text{latency}}^{\text{tree}} < F_{\text{latency}}^{\text{chain}}$. As a result, the required memory size to buffer the received vector $\mathbf{y}_i$ is expected to be much smaller in a tree topology.

4.2.3 Discussion

4.2.3.1 The impact of WD and FD on memory capacity and data exchange bandwidth

As discussed at the beginning of this section, there is a design trade-off between system performance and deployment cost, given different WD and FD scenarios. Once the CSP cooperation strategy and the corresponding decentralized processing algorithms have been decided, the data exchange bandwidth requirement is then fixed independent of the FD of the interconnections. However, the cost to support the same data exchange bandwidth for large FD can be much higher. For instance, according to Table 4.1 of Reindeer Deliverable 3.1, the energy per bit transfer for larger distance (e.g., 100G Ethernet with router/repeaters) is 100$\times$ higher comparing to that for smaller distance (e.g., 100G Ethernet without router/repeaters). In terms of memory capacity, large FD will incur much longer latency due to multi-hop data exchange. This implicates that large size memory would be needed to buffer the received data vectors, according to the analysis in (4.11).

4.2.3.2 Generalize to other topologies and connections

In section 4.2.1, we used the Daisy-chain and tree topologies with uni-directional links between CSPs as examples to perform the initial analysis of the hardware requirements for distributed processing algorithms. Depending on the real-life RW deployment, quality of service requirements, and service scenarios, more complicated topologies with bi-directional links can be implemented, as shown in Fig. 4.7. In these cases, more than one message passes per uplink detection may occur, depending on the deployed algorithms, for further interference cancellation [28]. Here we can introduce I_{ave}, the average number of message passes (per link per detection) when executing decentralized uplink detection algorithms on a selected topology. For instance, the RLS algorithm can run I_{rls} iterations in the Daisy-chain [28] and $I_{\text{ave}} = I_{\text{rls}}$. The corresponding data exchange bandwidth requirement will then be I_{ave} times of the uni-directional cases shown in Section 4.2.2.1. The latency of the decentralized processing will also increase accordingly with increased I_{ave}, resulting higher memory capacity requirement according to (4.11).

One of the follow-up steps in the project is to analyse the hardware cost and compare different distributed processing algorithms/architectures with numerical examples using more realistic deployment scenarios, use cases, and specific hardware characteristics.

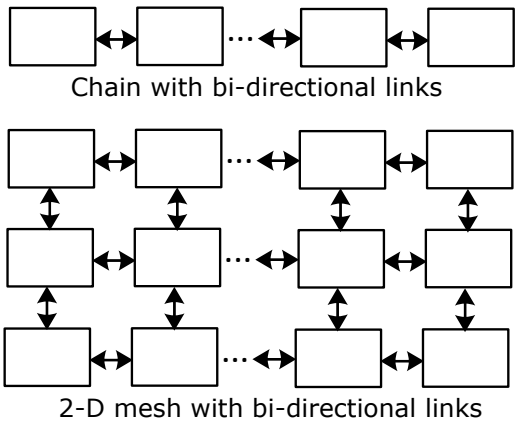


Figure 4.7: CSPs connected with bidirectional links in a Daisy-chain and a 2-D mesh topology, respectively.

4.3 Allocation of resources for dynamic management of federations

The concept of federations has been introduced in REINDEER Deliverable 2.1 [29] as means to serve diverse applications and many user equipments (UEs) in a RW environment. Representative cases illustrating the need and value of a federations-based approach were worked out in [30]. REINDEER Deliverable 3.2 clarified the overall problem of federation orchestration to include:

- Creation and removal of federations as necessary
- Association of UEs and applications to federations
- Allocation of resources to federations
- Dynamic update of resource allocations and association of UEs to federations

The above tasks clearly fall in the category collective, and are of a different nature than the typical data processing: they both need to operate on a more RW central level, and temporally require relatively sporadic updates only. However, as elaborated in the above-mentioned REINDEER D3.2, the allocations to be performed relate to the Santa Claus problem [31], which was shown to be NP-hard.

1. **The input** to the federation orchestration should comprise information on the resources in and the topology of the RWs. This includes the position and capabilities of CSPs and the interconnect between them. Moreover, information is needed on the UEs that want to get connected and their requirements. Hence, this information will need to be gathered and updated on a regular basis. Regular basis in this context definitely is not at the level of real-time data processing. It should also be noted that this does not require a lot of data volume, nor bandwidth. It will hence not result in any significant load on the front-haul.
2. **The problem solving** for the allocation problems is quite, potentially very, demanding in terms of computations. It is therefore envisioned that this orchestration will require an ECSP to be executed on.
3. **The outcome** of the allocation problem again will need to be communicated in the RW, to all CSPs involved in the orchestration. This will require sporadic updates to perform dynamic re-allocations. The communication regarding the (novel) allocation itself will also require a relatively low bandwidth. If a broadcasting/multicasting mode is available in the RW, this could be an appropriate way of sharing this information. It should also be noticed that this is not a time-critical process, yet changes in resource allocation should happen at the CSPs in an orchestrated mode.

Chapter 5

Summary and Conclusions

This deliverable reports on the studies and assessments regarding architectural requirements, bottlenecks, and algorithm-architecture co-design opportunities in distributed infrastructures.
In Chapter 2, we undertook a comprehensive study on the advantages and disadvantages of different architectures to provide the clock sources in a RadioWeaves (RW) network. We also provided several methods to overcome the shortcomings of the shared clock architectures using PPS signals or network synchronization signals. For the architecture with individual clock sources in each contact service point (CSP), we discussed Ethernet connectivity and over the air methods as two alternative schemes to synchronize the CSPs. We then evaluated the different combining methods for receive signal processing in the uplink of a distributed multiple-input multiple-output (MIMO) architecture. We developed a scalable decentralized method by only using a subset of CSPs for each user equipment (UE) in order to achieve the performance of a fully centralized architecture. We also provided a Kalman filtering combining method for the uplink signal processing and did a detailed numerical study for all the methods.

In Chapter 3, we conducted a detailed numerical study on the uplink performance of standard recursive least squares (SRLS) algorithm in a daisy-chain topology RW architecture with finite capacity back-haul communication between CSPs. We concluded that the QR decomposition based recursive least squares (QR-RLS) algorithm is not sensitive to the choice of regularization parameter and hence better estimation performance than the SRLS method.

In Chapter 4, we provided several front-haul link architectures which can be deployed in a RW network, to co-process the wireless signals, each coming with its own cost and benefits/challenges. We then discussed the memory and bandwidth requirements as a function of various system parameters to transfer the data between different processing clusters for several network topologies and distributed processing algorithms. We also briefly commented on the memory and data exchange requirements for general network topologies. We also elucidated the different kinds of data to be exchanged between CSPs for federation orchestration, UE association, and resource allocation and updation.

Bibliography

[1] REINDEER project, “Hardware requirements to support energy transfer to energy-neutral nodes,” Deliverable ICT-52-2020 / D2.3, unpublished (March 2023).

[2] CERN. (2022) The White Rabbit Project. Visited on 2023-03-23. [Online]. Available: http://white-rabbit.web.cern.ch/

[3] Y.-S. Tu and G. J. Pottie, “Coherent cooperative transmission from multiple adjacent antennas to a distant stationary antenna through AWGN channels,” in *IEEE 55th Vehicular Technology Conference*, vol. 1, 2002, pp. 130–134.

[4] R. Mudumbai, D. R. B. Iii, U. Madhow, and H. V. Poor, “Distributed transmit beamforming: challenges and recent progress,” *IEEE Communications Magazine*, vol. 47, no. 2, pp. 102–110, 2009.

[5] M. M. Rahman, H. E. Baidoo-Williams, R. Mudumbai, and S. Dasgupta, “Fully wireless implementation of distributed beamforming on a software-defined radio platform,” in *Proceedings of the 11th international conference on Information Processing in Sensor Networks*, 2012, pp. 305–316.

[6] F. Quitin, U. Madhow, M. M. U. Rahman, and R. Mudumbai, “Demonstrating distributed transmit beamforming with software-defined radios,” in *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2012, pp. 1–3.

[7] F. Quitin, M. M. U. Rahman, R. Mudumbai, and U. Madhow, “Distributed beamforming with software-defined radios: frequency synchronization and digital feedback,” in *IEEE Global Communications Conference (GLOBECOM)*, 2012, pp. 4787–4792.

[8] H. V. Balan, R. Rogalin, A. Michaloliakos, K. Psounis, and G. Caire, “AirSync: Enabling distributed multiuser MIMO with full spatial multiplexing,” *IEEE/ACM Transactions on Networking*, vol. 21, no. 6, pp. 1681–1695, Dec. 2013.

[9] M. Rahman, S. Dasgupta, and R. Mudumbai, “A distributed consensus approach to synchronization of RF signals,” in *IEEE Statistical Signal Processing Workshop (SSP)*, 2012, pp. 281–284.

[10] O. Abari, H. Rahul, D. Katabi, and M. Pant, “Airshare: Distributed coherent transmission made seamless,” in *IEEE Conference on Computer Communications (INFOCOM)*, 2015, pp. 1742–1750.

[11] R. Rogalin, O. Y. Bursalioglu, H. Papadopoulos, G. Caire, A. F. Molisch, A. Michaloliakos, V. Balan, and K. Psounis, “Scalable synchronization and reciprocity calibration for distributed multiuser MIMO,” *IEEE transactions on wireless communications*, vol. 13, no. 4, pp. 1815–1831, 2014.

[12] U. K. Ganesan, R. Sarvendranath, and E. G. Larsson, "Beamsync: Over-the-air carrier synchronization in distributed radioweaves," in *WSA 2021; 25th International ITG Workshop on Smart Antennas*, 2021, pp. 1–6.

[13] J. Vieira and E. G. Larsson, "Reciprocity calibration of Distributed Massive MIMO Access Points for Coherent Operation," in *IEEE 32nd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, 2021, pp. 783–787.

[14] REINDEER-Horizon-2020, "Deliverable D3.2: Methods for Communication and Initial Access with Radioweaves," 5th October, 2022.

[15] E. Björnson and L. Sanguinetti, "Making Cell-Free Massive MIMO Competitive With MMSE Processing and Centralized Implementation," *IEEE Trans, Wireless Commun.*, vol. 19, pp. 182–186, Jan. 2020.

[16] Ö. T. Demir, E. Björnson, and L. Sanguinetti, *Foundations of User-Centric Cell-Free Massive MIMO*. PO Box 179, Delft, The Netherlands: now Publishers Inc., 2021.

[17] G. Interdonato, E. Björnson, H. Q. Ngo, P. Frenger, and E. G. Larsson, "Ubiquitous Cell-Free Massive MIMO," *EURASIP Journal on Wireless Communications and Networking*, vol. 2019:197, 2019.

[18] E. Björnson, J. Hoydis, and L. Sanguinetti, "Massive MIMO network: Spectral, energy and hardware efficiency," *Foundations and Trends ®in Signal Processing*, vol. 11, pp. 154–655, 2017.

[19] K. W. Helmersson, P. Frenger, and A. Helmersson, "Uplink D-MIMO Processing Using Kalman filter Combining," in *Proc. IEEE Global Communications Conference*, Rio de Janeiro, Brazil, December 2022.

[20] *3GPP, Further advancement for E-UTRA physical layer aspects (Release 9)*, 3GPP Std. TS 36.814, Mar. 2017.

[21] K. W. Helmersson, P. Frenger, and A. Helmersson, "Uplink D-MIMO with Decentralized Subset Combining," in *Proc. IEEE International Conference on Communications*, Seoul, South Korea, May 2022.

[22] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Trans, ASME, J. Basic Eng.,*, vol. 82, pp. 35–45, March 1960.

[23] Z. H. Shaik, E. Björnson, and E. G. Larsson, "MMSE-Optimal Sequential Processing for Cell-Free Massive MIMO With Radio Stripes," *IEEE Transactions on Communications*, vol. 69, no. 11, pp. 7775–7789, 2021.

[24] J. E. Potter and R. G. Stern, "Statistical filtering of space navigation measurements," in *Proc. of the 1963 AIAA Guidance and Control Conference, AIAA*, New York, 1963, pp. 333–1–333–13.

[25] S. Haykin, *Adaptive filter theory*, 4th ed. Upper Saddle River, NJ: Prentice Hall, 2002.

[26] Q. Zhang, "Some Implementation Aspects of Sliding Window Least Squares Algorithms," *IFAC Proceedings Volumes*, vol. 33, no. 15, pp. 763–768, 2000, 12th IFAC Symposium on System Identification (SYSID 2000), Santa Barbara, CA, USA, 21-23 June 2000. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1474667017398440

[27] S. Malkowsky, J. Vieira, L. Liu, P. Harris, K. Nieman, N. Kundargi, I. C. Wong, F. Tufvesson, V. Öwall, and O. Edfors, "The World's First Real-Time Testbed for Massive MIMO: Design, Implementation, and Validation," *IEEE Access*, vol. 5, pp. 9073–9088, 2017.

[28] J. Rodríguez Sánchez, F. Rusek, O. Edfors, M. Sarajlić, and L. Liu, "Decentralized Massive MIMO Processing Exploring Daisy-Chain Architecture and Recursive Algorithms," *IEEE Transactions on Signal Processing*, vol. 68, pp. 687–700, 2020.

[29] O. Edfors, R. Brazil, H. Petautschnig, G. Callebaut, T. Feys, L. V. der Perre, E. G. Larsson, O. Edfors, E. Fitzgerald, L. Liu, J. R. Sanchez, W. Tärneberg, P. Frenger, B. Deutschmann, T. Wilding, and K. Witrisal, "Radioweaves high-level architecture and processing distribution strategies," REINDEER project, Deliverable ICT-52-2020 / D2.1, Jan. 2022. [Online]. Available: https://doi.org/10.5281/zenodo.5938909

[30] G. Callebaut, W. Tärneberg, L. Van der Perre, and E. Fitzgerald, "Dynamic federations for 6g cell-free networking: Concepts and terminology," in *2022 IEEE 23rd International Workshop on Signal Processing Advances in Wireless Communication (SPAWC)*, 2022, pp. 1–5.

[31] N. Bansal and M. Sviridenko, "The santa claus problem," in *Proceedings of the Thirty-Eighth Annual ACM Symposium on Theory of Computing*, ser. STOC '06. New York, NY, USA: Association for Computing Machinery, 2006, p. 31–40. [Online]. Available: https://doi.org/10.1145/1132516.1132522