Technical University of Munich
TUM School of Computation, Information and Technology

Coordination, Trust and Orchestration of Cognitive, Open, Multi-Vendor Network Automation Functions

Anubhab Banerjee

Complete reprint of the dissertation approved by the TUM School of Computation, Information and Technology of the Technical University of Munich for the award of the doctoral degree Doktor der Ingenieurwissenschaften (Dr.-Ing.).

Chair:		Prof. Debarghya Ghoshdastidar
Examiners:	1.	Prof. Dr.-Ing. Georg Carle
	2.	Prof. Dr.-Ing. Peter Rost

The dissertation was submitted to the Technical University of Munich on 12.07.2023 and accepted by the TUM School of Computation, Information and Technology on 18.12.2023.

Abstract

Over the past few years, network traffic has grown exponentially with the rise in mobile communication network users and services. This, in turn, increased the operational complexity and hiked the capital and operational expenses of mobile network operations. Present mobile networks are highly complex systems comprising automation mechanisms that leverage numerous technologies. These technologies and automation tools have been developed to deal with the configuration, optimization, and troubleshooting of mobile network operations, facilitating an efficient network management. Self-Organizing Networks (SONs) are an example of one such technology that aims to decrease human intervention by automating network management tasks. Although SON is used widely as an automation tool, it has some shortcomings; particularly, maintenance and upgrade of a rule-based management system like SON are challenging. To address these shortcomings, research is aimed at incorporating machine learning (ML) capabilities in SON and introduced a new management paradigm called cognitive autonomous networks (CANs). Like SON, CAN consists of multiple network automation functions called cognitive functions (CFs). A CF is an intelligent network function that is responsible for managing specific optimization related tasks. Throughout this thesis, we use the following CFs that are relevant for the use cases covered in this thesis: mobility load balancing (MLB), mobility robustness optimization (MRO), coverage and capacity optimization (CCO), and energy savings (ES). These functions have already been standardized by both the network operator community as well as the standardization bodies for SON use cases.

These CFs operate on the same radio network in parallel and learn to optimize the same set of radio network parameters. Since they work independently but simultaneously try to adjust the same set of parameters, conflicts are likely to happen among them. These conflicts need to be resolved to keep the network stable and the network operations uninterrupted. The conflicts can be resolved by introducing some coordination among the CFs. In this thesis, we evaluate different paradigms of coordination for CAN and establish that centralized coordination is the most beneficial for CAN. In the first part of the thesis, we propose solutions for centralized coordination for conflict resolution among the CFs. These solutions are based on the Nash Social Welfare Function (NSWF) and a more advanced version of NSWF, the Fisher Market Model, combined with Eisenberg-Gale optimization. Our proposed solutions are designed to serve the combined interest of all the CFs while resolving a conflict. However, when compared between these two approaches, we find that the second one provides a significant improvement over the first one regarding overall network performance.

Keeping with the current trend of making the network management system more open, flexible, and agile, in this thesis, we consider an open, multi-vendor CAN where different

CFs can be produced and supplied by different vendors. The open and multi-vendor environment raises the concern of trust, since a vendor can produce a CF designed to optimize its objective by manipulating the centralized coordination while degrading the overall network performance. To advertise its product's superiority and establish itself in an open market competition, a vendor has enough motivation to produce such a manipulative CF (MCF). Not only we visualize the concept of an MCF and explain its behavior, but we experimentally also show that an MCF can exist and downgrade the network performance. Furthermore, we investigate the undesired behavior of MCFs in two steps. In the first step, we propose a regression-based solution to detect an MCF with the expectation that when detected, such an MCF could be shut down. However, detecting an MCF is sometimes not enough since an MCF cannot permanently be shut down without affecting certain network services. So, in the second step, we nullify the effect of an MCF using a random noise generation based method.

Besides considering the coordination and the trust, we focus on enhancing the operational capabilities of the CFs. However, it is also crucial to utilize the knowledge of the CFs to maximize the benefit. Customizing cell radio configurations requires complete knowledge of the inter-dependency among the radio parameters and observable metrics. Although it is challenging for the operator to acquire that knowledge, the task becomes easier if the operator can use the knowledge already possessed by the CFs. To utilize the knowledge of the CFs, in the last part of the thesis, we propose an intent-driven interface between the operator and CAN, enabling the operator to easily orchestrate CAN. The interface also provides additional capabilities like detection and removal of conflicts among the intents. After we evaluate the intent-driven interface and show that it is fit for real-life scenarios, we also show that our proposed solution conforms with the ongoing worldwide mobile network standardization activities.

As we see, in this thesis, we cover three important aspects (coordination, trust, and orchestration) regarding cognitive, open, multi-vendor network automation functions for next-generation (5G-Advanced/6G) mobile network management. We see that, although we are moving towards an open, flexible, cognitive network management system, several areas of improvement need to be addressed. On the other hand, as an advanced level of network automation, intent-driven management functionalities also need to be incorporated into the network management system. In this thesis, we address these three important topics which are prevalent and vital for the management and automation of future mobile networks.

Acknowledgments:

Writing this thesis would not have been possible without the support I received from many individuals over the past few years.

First and foremost, I would like to thank Prof. Georg Carle for giving me the opportunity to pursue doctoral research under his supervision and his guidance throughout the journey. I would also like to thank Prof. Peter Rost for being the second examiner and Prof. Debarghya Ghoshdastidar for chairing the examination committee.

I am forever grateful to my parents, Santana Banerjee and Joynarayan Banerjee. Despite being thousands of miles away, their unquestioning and unwavering support helped me tremendously in navigating the difficulties I faced during this time. On this note, I heartily thank my whole family, especially my cousins, whose enthusiasm and support were a constant inspiration during this journey.

As the research work was mostly done during my time at Nokia Solution and Networks in Munich, I am most thankful to Dr. Henning Sanneck for financing my doctoral studies and Dr. Stephen Mwanje for guiding me on this journey. I have received invaluable guidance about everything from Stephen all these years, and I thank him sincerely for being such a wonderful mentor. I want to thank all the colleagues and fellow PhD candidates I had the pleasure of meeting and collaborating with during my time at Nokia. I would also like to thank the researchers I met at various conferences and the anonymous reviewers who provided valuable feedback on my research.

Finally, I sincerely thank Sayantini Majumdar and my other friends, both here and at home, for providing companionship and ease of mind in the hardest of times. Without their endless support, I would not have accomplished this at all.

München, 25.01.2024

Anubhab Banerjee

Contents

List of Figures

List of Tables

List of Algorithms

Part I

Introduction and Background

1. Introduction

1.1 Self-Organization in Mobile Networks

The backbone of cellular networks is the collection of overlapping cells spread over a specific geographical region. A geographical area is covered by multiple cells, where each cell is managed by a base station that includes one or multiple low-power (on the order of 100W or less) transceivers [2]. Each cell allocates a band of radio frequencies used by the transceiver(s) for communication. Adjacent cells are assigned different frequencies to avoid interference, although cells sufficiently distant from one another can use the same frequency band. However, in LTE, adjacent cells can use same frequency band using OFDMA [3].

To support the increasing demand for mobile network services, over the past couple of decades mobile networks evolved with the development of numerous technologies. These developments are further classified into multiple generations, where a new generation symbolizes significant advancement over the previous generation. Although the 5th generation (5G) has been rolled out in some major cities around the world, currently 4G LTE still covers most of the world. LTE stands for Long Term Evolution which started as a project in 2004 by the international telecommunication body 3rd generation partnership project (3GPP)[1] [4]. The high-level network architecture of LTE has three primary components:

- user equipment (UE): it denotes the mobile devices.
- evolved packet core (EPC): it communicates with packet data networks of the outside world, i.e., the internet.
- evolved UMTS Terrestrial Radio Access Network (E-UTRAN): it acts as a connection between the UEs and the EPC.

In this thesis, we focus on E-UTRAN, which is an interconnected network of base stations. In LTE, these base stations are denoted as evolved node B (eNB). An eNB in LTE is responsible for various tasks, like:

[1]3rd generation partnership project or 3GPP is a global initiative that provides specifications for telecommunication technologies.

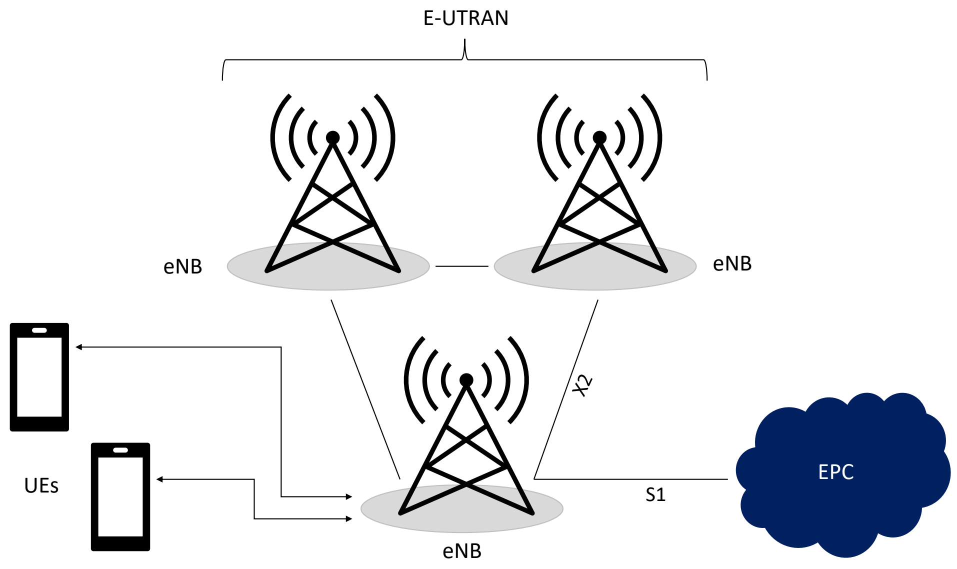


Figure 1.1: Overview of mobile network

(i) radio resource management which includes connection mobility control, radio bearer control, radio admission control, and scheduling.

(ii) the compression of packets and packet encryption.

(iii) the mobility management entity (MME) selection.

(iv) UE to EPC connection by routing the traffic from the user-plane to the service gateway (SGW).

(v) radio measurement analysis and reports.

To manage the exponential increase in the number of mobile devices and data services, network operators are forced to deploy more eNBs to satisfy mobile users' expectations. Such massive deployment increases the capital and operational expenditures of the operator. Also, user demand for a better mobile network experience calls for developing newer access schemes and radio technologies. To overcome the challenge of increasing costs and managing complex technologies, the concept of self-organization in the mobile network was implemented [1]. To make the idea acceptable in the scientific community, 3GPP published a list of use cases where the self-organizing network (SON) can be a part of LTE standards [5].

We already highlighted multiple radio resource management and measurement-related tasks that are carried out by an eNB. In SON, multiple network functions are proposed, among whom those tasks are divided. In the context of mobile network standards, 3GPP defined a SON Function (SF) as a network function that can be made self-organized (c.f. Fig. 1.2). Each SF is a rule-based function that is triggered when a particular network context satisfies the necessary condition(s) that initiates the execution of an associated SON algorithm. The algorithm then re-configures certain network parameters (at a cell level), aiming to optimize certain network metrics or key performance indicators (KPIs). The whole process works in a closed-loop self-organized manner while the mobile network operator (MNO) only needs to adjust the context-based trigger condition(s) as necessary.

3GPP defined three different architectural options for the implementation of SFs related to the general 3GPP network management:

- *Centralized SON*: here SF logic resides in a central entity. In 3GPP network management architecture, such a central entity could be a Network Manager (NM) or

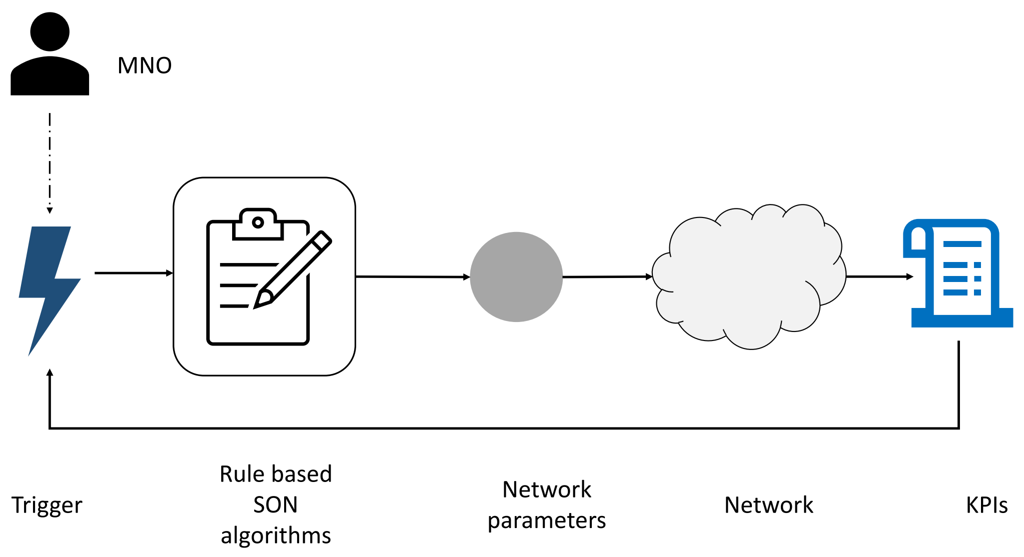


Figure 1.2: Operational structure of SON

Domain Manager (DM) as shown in Fig. 1.3. Information from different eNBs can be accessed at such a management entity directly.

- *Distributed SON*: here SFs are implemented at the eNB level. One eNB can access the necessary information from neighboring eNBs via interfaces like X2, as shown in Fig. 1.3. It is a good choice for SON use cases when the focus is on a single cell or a cluster of neighboring cells.
- *Hybrid SON*: in this case, the benefits of both Centralized and Distributed SON are combined by implementing some SFs at the management level and some SFs at the eNB level.

To serve the interest of the problems investigated in this thesis, throughout the thesis, we consider Distributed SON: we assume that SON is deployed at a cell level by one or multiple SFs running in each eNB. In the context of this thesis, the following SFs are essential:

- *Mobility Robustness Optimization (MRO)* adjusts handover parameters to optimize handover performance.
- *Mobility Load Balancing (MLB)* balances load distribution among neighboring cells and adjusts cell individual offset (CIO) to reduce the load from a high-load cell.
- *Coverage and Capacity Optimization (CCO)* balances the trade-off between cell coverage and the adversarial effects originating from increasing cell coverage.
- *Energy Savings (ES)* reduces energy consumption in a cell.

These SFs are chosen because they are relevant to the use cases described in this thesis.

1.2 Towards Cognitive Autonomy in Mobile Networks

For an SF, this type of manually-defined, context-based trigger limits the flexibility of SON since there can be a very large number of contexts in a real-life mobile network.

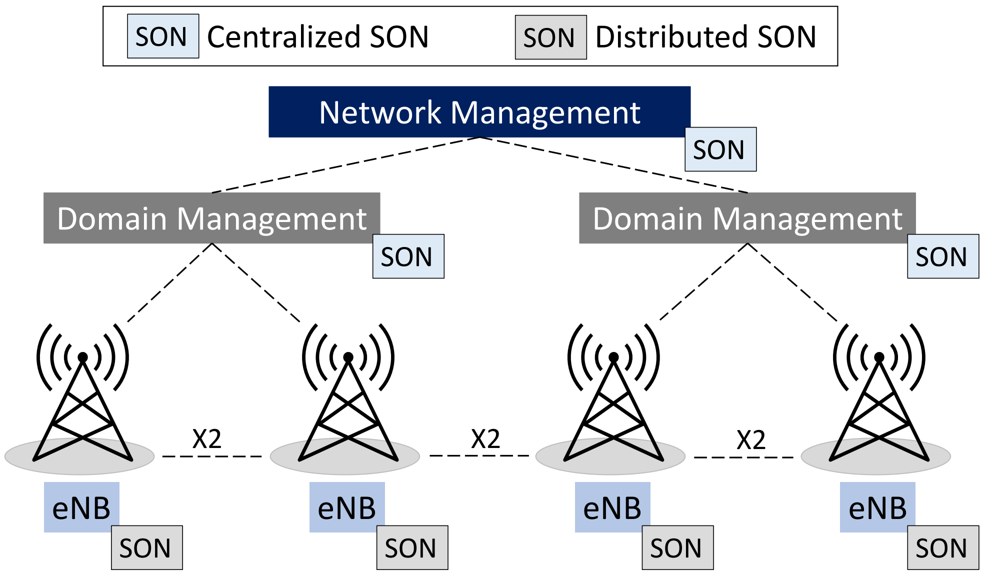


Figure 1.3: Initialization of SON in mobile network architecture

For the same reasons, SFs are unable to adapt to significant environmental or operational changes, e.g., the introduction of new business or service models, network architecture modification, etc. Since such changes will occur even more frequently in B5G networks [6], a more flexible and adaptive network management system is required. On the other hand, the SFs are coordinated hierarchically, following some fixed rules or policies [7]. A modification of a trigger condition of an SF may require rewriting the rules of the entire coordination mechanism. Maintaining such a hierarchical system becomes increasingly complex and costly with the increase in the number of SFs.

Rapid progress in the field of artificial intelligence, particularly machine learning (ML), enabled researchers to introduce cognitive capabilities in SON to overcome the challenges mentioned above. One of the essential prerequisites for using ML-based solutions is to obtain training data. When an ML algorithm is used to solve a problem using the available data, the learning process derives a model by autonomously deducing rules that achieves the problem objective. This removes the necessity of rule-based hierarchical network management system like SON. So, it is desirable to deploy ML based solutions to address the shortcomings of SON. Therefore, when an ML-based solution is applied in this context, it is desired that the solution has the following features:

(i) *Autonomous inference*: In SON, the desired capabilities are expressed by a set of explicit rules or conditions. However, in an ML-based solution, it is done implicitly by supplying the ML algorithm with training data that reflects the notion of desired functionality. This approach removes the requirement of having complex hierarchical rules and focuses on efficiently specifying desired functionalities.

(ii) *Online learning*: Online learning is a type of ML where data becomes available in a sequential manner and is used to update the ML model in each step, as opposed to batch learning techniques where all the data are available at once. As the ML algorithm can already deduct its own rules, this approach can be utilized during the operation phase in a real-life scenario where the environment may change rapidly, and the model is updated accordingly.

Cognitive Autonomous Network (CAN) [7] is conceptualized based on those two aforementioned features in the scope of mobile network management. CAN consists of multiple cognitive functions (CFs), which are independent, closed-loop, learning-based functions.

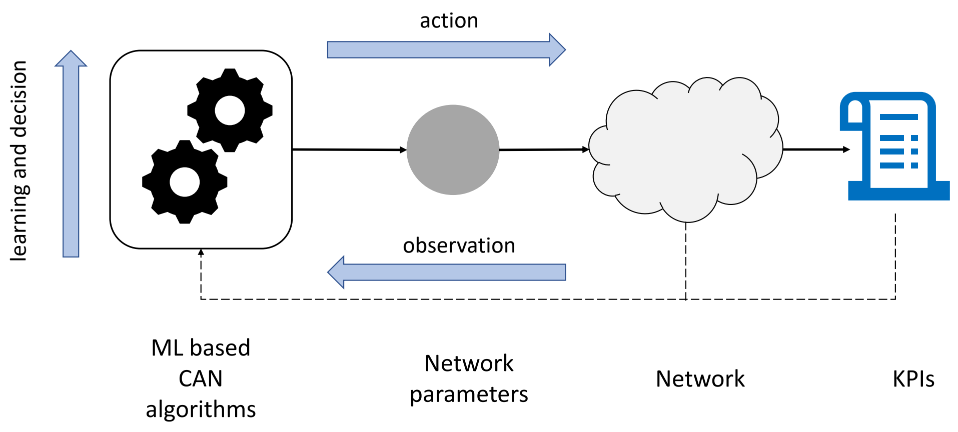


Figure 1.4: Operational structure of CAN

CFs are functionally equivalent to SFs, i.e., the responsibility of a CF is similar to the responsibility of an SF. For example, both the CF MRO and SF MRO perform the same task defined in Section 1.1. However, there is a significant difference between an SF and a CF. SFs are rule-based functions that are triggered when specific predefined rules are satisfied. Unlike SFs, CFs are cognitive entities that act based on their inference and do not require any external trigger. The operational structure of a CF (Fig. 1.4) can be primarily divided into four parts:

(i) *Observation*: the CF observes the network and all related KPI values.

(ii) *Learning*: the CF learns from this observation.

(iii) *Decision*: the CF uses its learning to make a decision.

(iv) *Action*: the CF acts on the decision it made.

So, just like SON, CAN also works in a closed-loop manner. However, in SON, the triggers always need to be defined and adjusted by humans who perform the job of an MNO, whereas no human intervention is needed in CAN. The difference can be observed by comparing Fig. 1.4 with Fig. 1.2.

1.3 Thesis Objective

The objective of this thesis is to study CFs in an open, multi-vendor environment. We see that there are two primary components in the thesis objective: (i) study of CFs, and, (ii) open, multi-vendor environment.

We choose to study the CFs over SFs because CFs provide many advantages and improvements over state-of-the-art SFs. Besides, they are potential candidates for use in the next-generation mobile networks (5G-Advanced/6G). However, since CAN is a relatively new concept (proposed only a few years ago), currently there are many unexplored research questions. Some of these questions are critical in the understanding of CAN which need further investigation.

The environment, in which the CFs are studied, is also equally important. The current trend of mobile network management aims to increase openness and flexibility [8].

Increasing openness drives the mobile industry toward an ecosystem of innovative, multi-vendor, interoperable, and autonomous RAN with reduced cost, improved performance, and greater agility [8]. So, in our thesis objective, we combine these two elements of future mobile network management. Our objective is to investigate three operational categories of CAN: (i) coordination, (ii) trust, and (iii) orchestration.

1.3.1 Coordination

In CAN, the CFs work as independent learning agents, optimizing their objectives while sharing the same resources. For example, mobility load balancing (MLB) and mobility robustness optimization (MRO) both use handover parameters: time to trigger (TTT) and CIO. So, there is a chance of a conflict arising between MLB and MRO over the parameter TTT or CIO. To avoid these conflicts among the CFs regarding resource sharing, coordination among the CFs is necessary. In SON, SFs are coordinated in a hierarchical manner, by a rule-based SON controller [9]. However, since that approach is not applicable for CFs, in the context of coordination, two main questions arise:

Q1. What type of coordination is best for CAN, i.e., centralized coordination or distributed coordination? How do we prove that the chosen type of coordination is better than the other one?

Q2. Can that coordination be done in a real-life scenario?

1.3.2 Trust

In an open, multi-vendor environment, different CFs from different vendors may not be equally trustworthy. To advertise the superiority of the product and establish it in competition with others, a vendor might have the motivation to produce a rogue CF that manipulates the Controller to achieve its objective. This type of CF, denoted as Manipulative Cognitive Function (MCF), does not hesitate to compromise the overall network performance to satisfy its objectives. The existence of an MCF in CAN is unwanted and requires serious attention. Two main questions which arise in this context are:

Q3. Is it possible to detect an MCF if it exists in CAN?

Q4. Even if the MCF can be detected, what is the best way to handle such an MCF in an operational system?

1.3.3 Orchestration

As we already discussed in Section 1.2, CAN works in a closed-loop manner without providing many opportunities for interaction with the MNO: once the MNO defines the objective of a CF, there is no further scope of intervention from the MNO side. So, if the MNO needs to customize specific parameters and KPIs in a cell, she has to gain complete knowledge of the dependence of KPIs on parameters across different network states. However, since the CFs already possess this knowledge, their knowledge can be utilized for this purpose, provided some interface is developed through which CAN and MNO can interact seamlessly. Two fundamental questions prevalent in this context are:

Q5. Is it possible to develop such type of interface through which CAN and MNO can communicate seamlessly?

Q6.Are there shortcomings that can be expected in such a type of interface?

These are the main questions that are investigated and addressed in this thesis.

1.4 Research Approach

After identifying the challenges in CAN in each category, we propose potential solutions and evaluate them in a simulation environment. The research approach followed in this thesis can be summarized as follows:

- We implement CAN in a simulation environment and the CFs as neural network (NN). This is the first and most necessary step since all our research in this thesis is focused on CAN.
- We experimentally demonstrate that our implemented CAN exhibits intelligent behavior and behaves as expected. This proves the worthiness of our implementation to evaluate our proposed solutions in the context of this thesis.
- We keep the simulation scenario the same throughout the thesis to maintain homogeneity.
- While introducing new concepts or terminologies, we always refer to the existing standards to show that our proposed solutions conform with the ongoing worldwide standardization efforts. In the context of this thesis, standards defined by 3GPP SA5 and ETSI ZSM [10] are referred to.
- We provide a mathematical explanation, wherever possible, to establish and validate our hypotheses and other assumptions.

1.5 Thesis Organization

This thesis is divided into five parts: Part I introduces the thesis and provides the necessary background. Part II is focused on coordination. Part III is focused on trust. Part IV is focused on orchestration and Part V concludes the thesis. Each part is then further divided into multiple chapters. The first chapter of each part (except Part I and Part V) describes the problems related to that part, while the rest of the chapters are dedicated to addressing those problems. While discussing each chapter, we also refer to the publications and patent applications, based on which the content of the chapter is produced. The complete list of the publications and patent applications can be found in Section 1.6.

1.5.1 Part I

Part I provides the outline of the thesis and the necessary background to understand CAN, and discusses the simulation environment used in this thesis. This part is further divided into three chapters, as described below.

Introduction is the first chapter that provides a brief overview of the thesis. It narrates the objective of this thesis and furthermore, it describes how the thesis is organized.

Cognitive Autonomous Networks: An Overview is the second chapter that provides a detailed overview of the CAN architecture. While discussing the CAN architecture, we also address Q1 formulated in Section 1.3.1. This background is crucial to understand the rest of the parts of the thesis. The discussion on the architecture is based on the peer-reviewed article P1 and the rest of the content is based on the patent application A1.

Simulator and Evaluation Environment Description is the third chapter that covers three aspects related to the simulation environment: (i) a detailed description of the simulator so the Reader can understand how the simulator works; (ii) configuration of the simulator, i.e., certain values and settings of the simulator used in this thesis. To maintain homogeneity in the simulation environment throughout the thesis, we maintain the same simulator configuration; (iii) CAN extension of the simulator. The simulator is a system-level simulator that only behaves like a mobile network. On top of the simulator, we implemented some Python modules, written by us, to implement CAN. The description of the simulator is necessary since the simulator is a Nokia internal simulator and is not currently available publicly. The discussion on the CAN extension of the simulator is based on the peer-reviewed article P2.

1.5.2 Part II

After providing the necessary background about CAN in Part I, in this part, we focus on the coordination in CAN. This part is primarily focused on addressing the question Q2 formulated in Section 1.3.1. This part is further divided into three chapters, as discussed below.

Coordination: Problem and Related Works is the first chapter in this part that describes the problem statement related to coordination in CAN. It covers different types of conflicts in CAN - how they originate and their effect on the system. Along with that, we also cover relevant existing research works on coordination issues. The discussion about different types of conflicts and existing research works are from the peer-reviewed article P3.

Conflict resolution among CFs is the second chapter in this part that addresses question Q2 formulated in Section 1.3.1. In this chapter, we propose a Nash Social Welfare Function (NSWF) based approach that resolves any type of conflict by choosing a value optimal for the combined interest of all CFs. We evaluate this solution analytically and quantitatively and describe our findings. The NSWF based solution is based on the patent application A2 and the evaluations are published in the peer-reviewed article P1.

Interest based optimal configuration calculation is the third chapter in this part where we highlight the shortcomings of the NSWF based solution and propose an alternative approach based on Eisenberg Gale optimization. One important aspect of that solution is to find the importance of a network parameter on a KPI, which we propose to determine using the Shapley value [11]. We evaluate the proposed solution extensively in the simulation environment introduced in Chapter 3. This content is primarily based on the peer-reviewed article P4 and the patent application A3.

1.5.3 Part III

In this part, we focus on the trust issues in CAN by addressing questions Q3 and Q4. Just like previous parts, this part is also divided into three chapters, as described subsequently.

Trust: Problem and Related Works is the first chapter in this part describes the problem statement related to trust in CAN. We conceptualize the idea of an MCF and experimentally demonstrate that such an MCF is capable of causing severe network performance degradation. We also cover existing research works and standards which are relevant to this problem. All the content of this chapter is based on the peer-reviewed article P5.

Manipulative Cognitive Function Detection is the second chapter in this part, where we address question Q3: we propose an ML-based algorithm called Manipulative CF Detector (MCD) to detect an MCF in an operational system. Our proposed solution can either be used as a standalone component or be integrated with the Controller. We evaluate MCD in different scenarios to prove its effectiveness against MCF. The conceptualization of MCD is based on the patent application A4 and the evaluation of MCD is based on the peer-reviewed article P5.

CoDeRa: Controller Decision Randomizer is the third chapter in this part. Although MCD can be effectively used to catch MCF, sometimes, for various reasons, an MCF cannot be just taken out of an operational system. In such cases, the MCF has to be neutralized, i.e., it is important to make the MCF give up its manipulative behavior. We propose a solution, called Controller Decision Randomizer (CoDeRa), that accomplishes the task and addresses Q4 described in Section 1.3.2. The idea of CoDeRa is based on the patent application A5 and evaluation of CoDeRa is based on the peer-reviewed article P6.

1.5.4 Part IV

We discuss the intent-driven orchestration of CAN in this part. This part is primarily focused on addressing Q5 and Q6, mentioned in Section 1.3.3. Like the previous parts, this part is also divided into three chapters, as described below.

Orchestration: Problem and Related Works is the first chapter in this part that formulates the problem statement related to the orchestration of CAN. Since CAN works in a closed-loop manner, it does not provide any opportunity for MNO to intervene. However, knowledge of CAN is essential for cell parameter customization. In this chapter, we focus on the problem of having no interface between CAN and MNO for knowledge sharing. The content of this chapter is based on the peer-reviewed article P7.

Intent-driven Network Automation Function Orchestration is the second chapter in this part, where we address question Q5: we propose an intent-driven interface between MNO and CAN, called intent-driven network automation function orchestrator (IDNAFO), through which CAN and MNO can communicate with each other. MNO can specify the requirements in the interface; then, the interface generates appropriate commands, which are to be executed by the Controller and the CFs to fulfill the requirements. The content of this chapter is based on the peer-reviewed article P8.

Intent-driven Conflict Detection and Resolution is the third chapter in this part, where we address question Q6: we propose a solution called Intent Contradiction Detector and Remover (ICDR) to address the shortcomings of IDNAFO. ICDR can detect and resolve conflicts (which arise before runtime) and contradictions (which arise during runtime) in IDNAFO. ICDR conceptualization and evaluation are based on the peer-reviewed article P9.

1.5.5 Part V

This is the last part of this thesis that contains a single chapter. We summarize and conclude our work in this chapter. Along with that, we also discuss potential directions in which the research can be carried forward.

1.6 Thesis Contributions

During his doctoral studies, the author contributes to multiple peer-reviewed articles published in different journals and conferences, patent applications and mobile network standards. This section is divided into three subsections to list all the contributions in each category.

1.6.1 Publications

In the context of this thesis, the author has the following publications:

[P1] **A. Banerjee**, S. S. Mwanje, and G. Carle. Game theoretic conflict resolution mechanism in cognitive autonomous networks. In 2020 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS), pages 1–8. IEEE, 2020.

[P2] **A. Banerjee**, S. S. Mwanje, G. Carle (2021). On the implementation of cognitive autonomous networks. Internet Technology Letters, 4(6), e317.

[P3] **A. Banerjee**, S. S. Mwanje, and G. Carle. Optimal configuration determination in cognitive autonomous networks. In 2021 IFIP/IEEE International Symposium on Integrated Network Management (IM), pages 494–500. IEEE, 2021.

[P4] **A. Banerjee**, S. S. Mwanje, and G. Carle. Towards control and coordination in cognitive autonomous networks. IEEE Transactions on Network and Service Management (TNSM), 19(1):49–60, 2022.

[P5] **A. Banerjee**, S. S. Mwanje, and G. Carle. On Detection of Manipulative Cognitive Functions in Cognitive Autonomous Networks. In 2021 17th International Conference on Network and Service Management (CNSM), pages 194-200. IEEE, 2021.

[P6] **A. Banerjee**, S. S. Mwanje, and G. Carle. Trust and performance in future AI-enabled, open, multi-vendor network management. IEEE Transactions on Network and Service Management (TNSM), 20(2):995-1007, 2023.

[P7] S. S. Mwanje, **A. Banerjee**, et al. Intent-Driven Network and Service Management: Definitions, Modeling and Implementation. ITU Journal on Future and Evolving Technologies (ITU J-FET), 3(3): 555-569, 2022.

[P8] **A. Banerjee**, S. S. Mwanje, and G. Carle. An Intent-Driven Orchestration of Cognitive Autonomous Networks for RAN management. In 2021 17th International Conference on Network and Service Management (CNSM), pages 380-384. IEEE, 2021.

[P9] **A. Banerjee**, S. S. Mwanje, and G. Carle. Contradiction management in intent-driven cognitive autonomous ran. In 2022 IFIP Networking Conference (IFIP Networking), pages 1–6. IEEE, 2022.

1.6.2 Patent Applications

The author is also an inventor of multiple patent applications which are mentioned below:

[A1] DESIGN OF AN ENERGY SAVINGS MODE OF OPERATION FOR COGNITIVE AUTONOMOUS NETWORKS
Anubhab Banerjee, Stephen S Mwanje
WO, PCT application no.: PCT/EP2021/054165, filed 19 February 2021

[A2] A COORDINATION AND CONTROL MECHANISM FOR CONFLICT RESOLUTION FOR NETWORK AUTOMATION FUNCTIONS
Anubhab Banerjee, Stephen S Mwanje
WO, PCT application no.: PCT/EP2020/061183, filed 10 April 2020

[A3] METHODS AND APPARATUSES FOR DETERMINING OPTIMAL CONFIGURATION IN COGNITIVE AUTONOMOUS NETWORKS
Anubhab Banerjee, Stephen S Mwanje
US, Provisional application no.: 63/056084, filed 24 July 2020

[A4] DETECTING MANIPULATIVE NETWORK FUNCTIONS
Anubhab Banerjee, Stephen S Mwanje, Abdelrahman Abdelkader
WO, PCT application no.: PCT/IB2021/054489, filed 24 May 2021

[A5] REDUCING SYSTEM DEGRADATION CAUSED BY MANIPULATIVE FUNCTIONS
Anubhab Banerjee, Stephen S Mwanje
WO, PCT application no.: PCT/IB2021/054486, filed 24 May 2021

1.6.3 Other Contributions

Apart from the aforementioned publications and patent applications, the author has also contributed to the ongoing standardization activities in mobile networks. In this context, the author primarily contributed to two standard documents: ETSI ZSM 011 [12] and 3GPP SA5 Technical Report (TR) 28.912 [13]. These contributions are based on the patent applications and publications by the author listed above.

ETSI ZSM 011: "Intent-driven autonomous networks; generic aspects"

Section 5.6.3.3.6. Notification capabilities.
Section 5.6.3.3.7. Testing.
Section 5.6.3.3.8. Verification of intent outcome.
Section 5.7.4. Potential intent conflict resolution approaches.

3GPP SA5 TR 28.912: "Study on enhanced intent driven management services for mobile networks"

Section 4.2. Intent conflicts.
Section 4.3. Enhancement of radio network intent expectation.
Section 4.7. Monitoring intent fulfillment information.
Section 4.8. Enablers for Intent fulfillment.
Section 4.9. Intent fulfillment feasibility check.

1.6.4 Awards

The author of this thesis also won the **Student Travel Grant (STG) Award 2021** at *17th International Conference on Network and Service Management (IEEE/IFIP CNSM 2021)* for the following paper:

A. Banerjee, S. S. Mwanje, and G. Carle. An Intent-Driven Orchestration of Cognitive Autonomous Networks for RAN management. In 2021 17th International Conference on Network and Service Management (CNSM), pages 380-384. IEEE, 2021.

List of Abbreviations Used in This Chapter

3GPP	3rd generation partnership project
CAN	Cognitive Autonomous Network
CoDeRa	Controller Decision Randomizer
CFs	cognitive functions
CIO	cell individual offset
DM	Domain Manager
eNB	evolved node B
EPC	evolved packet core
E-UTRAN	evolved UMTS Terrestrial Radio Access Network
ICDR	Intent Conflict Detection and Removal
IDNAFO	Intent-Driven Network Automation Function Orchestrator
KPIs	key performance indicators
MCD	Manipulative CF Detector
MCF	manipulative cognitive function
ML	machine learning
MLB	mobility load balancing
MME	mobility management entity
MNO	mobile network operator
MRO	mobility robustness optimization
NM	Network Manager
NN	neural networks
NSWF	Nash social welfare function
SF	SON Function
SGW	service gateway
SON	self-organizing network
TTT	time-to-trigger
UE	user equipment

2. Cognitive Autonomous Networks: An Overview

In the previous chapter we gave a brief overview of CAN and its evolution from SON. In this chapter we provide more detailed information about CAN, e.g., we discuss our proposed architecture and end-to-end workflow of CAN. We cover the individual components in detail, and along with that we also highlight how those components interact with one another. The majority of the content in this chapter is from the following publication and patent application by the author:

[14] **A. Banerjee**, S. S. Mwanje, and G. Carle. Game theoretic conflict resolution mechanism in cognitive autonomous networks. In 2020 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS), pages 1–8. IEEE, 2020.

DESIGN OF A ENERGY SAVINGS MODE OF OPERATION FOR COGNITIVE AUTONOMOUS NETWORKS
Anubhab Banerjee, Stephen S Mwanje
WO, PCT application no.: PCT/EP2021/054165, filed 19 February 2021

In addition to the content described in the publication and the patent application, this chapter includes the following contributions by the author:

- Mathematical establishment of the necessity of coordination in CAN.
- Comparison between centralized and distributed coordination.

2.1 CAN Architecture

In Section 1.1 we gave a brief overview of the cellular network. In a cell transceiver there are several adjustable control parameters, e.g., transmit power (TXP), remote electrical tilt (RET), TTT, CIO. In a cell, there are several observable KPIs as well, like, cell throughput, radio link failures (RLF), etc. These KPIs characterize the overall performance of the network and often need to be optimized to meet the service requirements. In a cell, KPI values are affected when one or multiple cell transceiver parameters (hereafter referred as control parameters) are changed. Apart from control parameters, there are some external factors that may influence a KPI. For example, in mobile networks, handover-related KPIs (e.g., RLF) are significantly influenced by the number of users present in the cell and the user mobility. To combine all the factors that influence a KPI into a single entity, the concept of "network state" was introduced [15]. Network states are abstract states built from combinations of quantitative KPIs, abstract (semantic) state labels, and operational contexts (e.g., date, time, number of users), such as the current configurations (e.g., values of TXP, RET, etc.) of the network or its elements.

In Section 1.2 we gave a brief overview of CAN. CAN is an intelligent network management entity that optimizes KPIs across different network states. CAN consists of multiple intelligent functions, called CFs. For each KPI, there is one CF that: (i) identifies the control parameters which influence the KPI, and, (ii) learns how the KPI changes when the network state changes. This learning by a CF is a continuous process that is active as long as the CF is operational. KPI associated with a target is called the objective of the CF. For example, if we use a CF to maximize downlink average user throughput, then, for the CF, KPI is: downlink average user throughput, target is: maximization and objective is: to maximize downlink average user throughput. The control parameters, on which the KPI is dependent, are called input control parameters (ICPs) of the CF. Since downlink average user throughput primarily depends on the cell TXP and RET, TXP and RET are the ICPs of the CF. As soon as the network state changes, the CF determines the values of ICPs, for which its objective is optimized in that network state, based on its learning.

2.2 Necessity of Coordination in CAN

Very often it is found that multiple KPIs are influenced by a single control parameter (for example, both interference and downlink throughput change when TXP changes). In 5G, the environment can change very rapidly [16], leading to frequent changes in the network state, and a CF may detect frequent changes in the optimal ICP values. Since each CF works independently, if each CF starts changing a certain control parameter according to its own will, the stability of the whole system may become compromised. To avoid that fiasco, there needs to be some type of coordination between the CFs. In this section we mathematically establish the necessity of coordination in CAN.

Let us consider a two-CF CAN model and formulate it as a normal-form game where F_1 and F_2 are players of the game. When there is a conflict of interest between the players, each player can take one of the following strategies: (i)the player continues to work on the interest (T), or, (ii) the player gives up the interest (G). There can be four possible different combinations - both of them choose G, both of them choose T, and, one of them chooses T and the other one chooses G. Payoff for each combination is defined as follows:

- when both of them choose G, no one modifies the interest and the interest remains constant. Both the players get an equal payoff, say r_1.
- when both of them choose T, each of them gets a payoff r_2. The benefit of fighting for the interest is worse than keeping it constant because it may change to worse outcomes for any of the players and so, $r_2 < r_1$.
- when one of them selects T and the other one selects G, whoever selects T gets a payoff r_3 and the other one gets a payoff r_4 with $r_3 > r_4$.

Relationship among r_1, r_2, r_3, and r_4: it is obvious that payoff is higher when the CF can work on the interest than when the interest is constant, i.e., $r_3 > r_1$. The payoff for one is also higher when either the interest is controlled by oneself or the interest remains constant than when it is changed according to the other's will, i.e., $r_2 > r_4$ and $r_1 > r_4$.

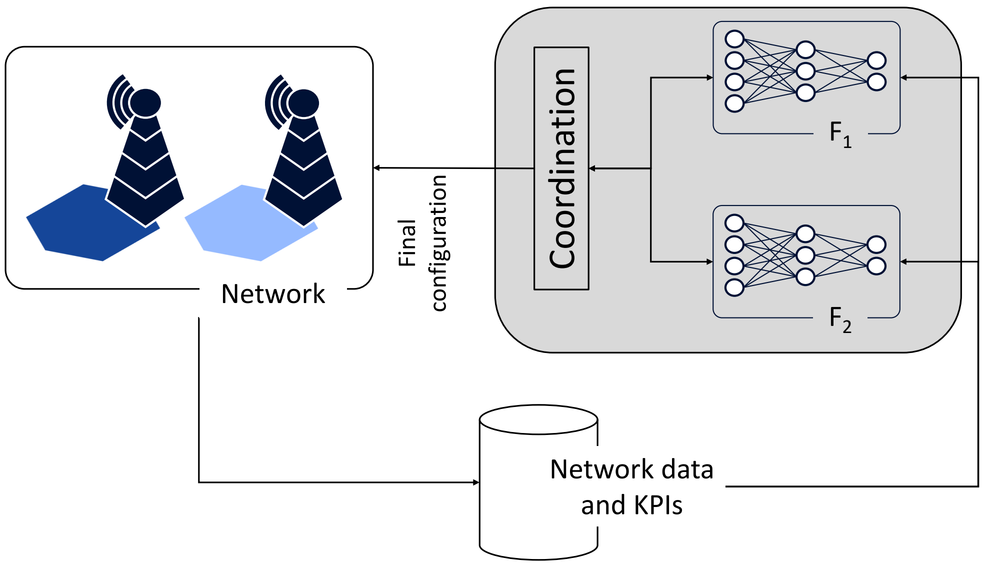


Figure 2.1: Architectural overview of CAN

The payoff for a CF is higher again when only the CF changes it than when both of them change it, i.e., $r_3 > r_1$. Combining all the above observations, we get

$$r_3 > r_1 > r_2 > r_4$$

As mentioned in [17], two criteria for a problem two qualify as a Prisoners' Dilemma are:

- Regardless of what the other players do, each player receives a higher payoff for defecting behavior than for cooperating behavior.
- All agents get a lower payoff if all defect than cooperate.

Now, a conflict between CFs is exactly like a Prisoner's Dilemma where the defecting behaviors is choosing T and cooperative behavior is choosing G. So, following the solution of Prisoner's Dilemma, where the best action for each prisoner is to choose defecting behavior, and the best action for each CF is: selecting T. On the contrary, if the CFs work with the existence of a coordination mechanism, they find that the best possible action for each CF is - choosing G, because, when both of them select G, both of them get a higher payoff (r_1) than the payoff they get (r_2) when they select T. This proves the necessity of a coordination mechanism for CFs in CAN.

So, CAN architecture and operations can be depicted as Fig. 2.1 without the intricate details. As we continue with the thesis, in later chapters we introduce more functionalities and add new components to this architecture.

2.3 The Necessity of Centralized Coordination in CAN

In the last section, we emphasized on the necessity of coordination in CAN. Now, the coordination can be done in either of these two ways:

- It can be done in a centralized way, i.e., there is a centralized entity that communicates with the CFs individually. In this case, the CFs do not need to communicate with each other directly.

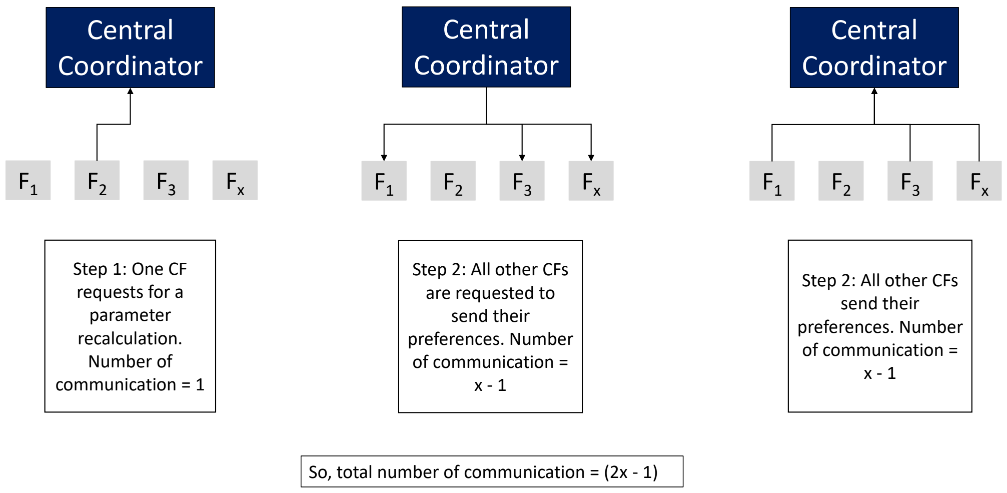

Central Coordinator
Central Coordinator
Central Coordinator
F1 F2 F3 Fx
F1 F2 F3 Fx
F1 F2 F3 Fx
Step 1: One CF requests for a parameter recalculation. Number of communication = 1
Step 2: All other CFs are requested to send their preferences. Number of communication = x - 1
Step 2: All other CFs send their preferences. Number of communication = x - 1
So, total number of communication = (2x - 1)

Figure 2.2: Communication in Centralized Coordination

- It can be done in a distributed way, i.e., the CFs communicate with each other directly and there is not any external centralized entity.

In SON, the coordination is done in a centralized way, i.e., a controller exists for this purpose [9]. However, for CAN, we evaluate the benefits of both types of coordination mathematically to find the most suitable one in this scenario. The suitable one is the one in which the amount of information exchange among the entities is lesser because the less is the amount of internal communication, less is the system overhead and less is the energy consumption. So, for both types of coordination model we calculate the amount of information exchange (which is calculated in terms of number of rounds of communication) within CAN and choose the one with the lesser amount of information exchange. For the mathematical analysis purposes, let us assume that there are x number of CFs: F_1, F_2, F_3, . . . , F_x, which need to be coordinated.

2.3.1 Centralized Coordination

In this type of coordination, a CF cannot change the value of a control parameter, so, at first the CF requests the centralized coordinator to change the value of a control parameter. Along with the request, the CF also sends its preference over the parameter to the coordinator. The coordinator does not change the value instantly, because there can be other CFs in the system using the same parameter and their performances can get affected by the change in the control parameter. The coordinator has to take the interests of all CFs into account before changing the value of such a parameter. So, as soon as the coordinator receives such a request, the coordinator sends messages to all other CFs to send their preferences to the coordinator. After that, the CFs send their preferences back to the coordinator based on which the coordinator decides the final value.

This whole process is shown in Fig. 2.2 with three different steps. So, when there are x CFs, total number of internal communication = $(2x - 1)$.

2.3.2 Distributed Coordination

In this type of coordination, the CFs talk with each other directly and agree on a value before a CF changes the value of a control parameter. These communications are shown

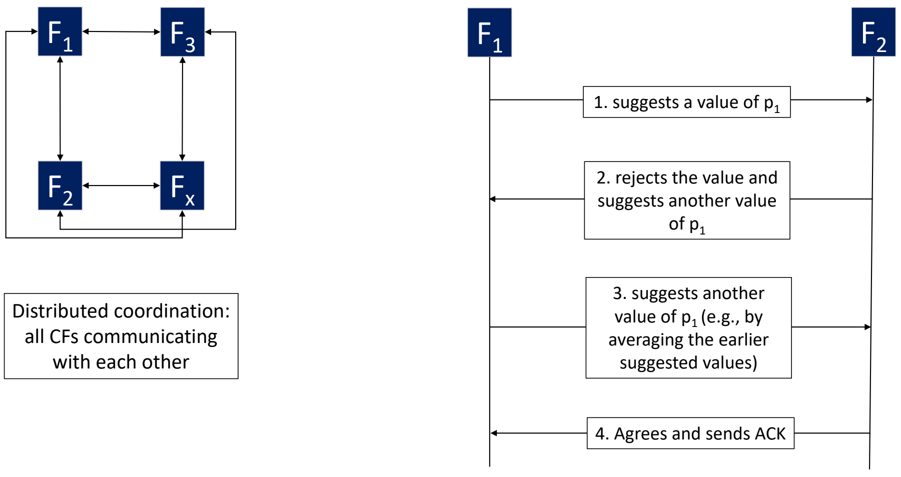

F1
F3
F2
Fx
Distributed coordination: all CFs communicating with each other
F1
F2
1. suggests a value of p1
2. rejects the value and suggests another value of p1
3. suggests another value of p1 (e.g., by averaging the earlier suggested values)
4. Agrees and sends ACK

Figure 2.3: Communication in Distributed Coordination

in the right part of Fig. 2.3. In the best case scenario, there are only 2 rounds of communication between a pair of CFs (Step 1 and Step 4). In all other cases, the number of communication between a pair of CFs is given by 2^m, where $m > 1$ (Step 2 and Step 3 can happen any finite number of times before the CFs finally agree on a value). So, when there are x CFs, total number of internal communication $= {}^xC_2 * 2^m$ where $m = 1$ in the best case scenario (all pairs of CFs have only 2 rounds of communication).

2.3.3 Comparison between Centralized and Distributed Coordination

Let us start the comparison between these two types of coordination by selecting the scenario that is best for the distributed coordination. Even in that scenario, if we can prove that the centralized coordination is more suitable than the distributed coordination, then we can definitely establish that the centralized coordination is always better than the distributed coordination and is the most suitable choice for CAN.

So, the centralized coordination is better when

$$\begin{aligned} {}^xC_2 * 2 &> 2x - 1 \\ \Rightarrow x * (x-1) &> 2x - 1 \\ \Rightarrow x^2 - 3x + 1 &> 0 \end{aligned} \tag{2.1}$$

Eq. 2.1 holds true for $x \geq 3$, i.e., when the number of CFs is 3 or more, centralized coordination is always better than the distributed coordination. Now, let us discuss now what happens when the number of CFs is 2.

When there are 2 CFs, in the centralized coordination there are 3 rounds of information exchange or communication. In the case of distributed coordination, there are total 2^m rounds of communication. Now, only in the best case scenario for distributed coordination, $m = 1$, which makes distributed coordination more preferred than the centralized coordination. However, in reality, the value of m can not be 1 always, since the probability of two CFs always agree on the same value is low. So, to model it realistically, we assume that p is the probability that the two CFs agree on the same value. In that case, total number of communication $= 2p + 2^m(1-p)$, $m > 1$, $m \in \mathbb{Z}^+$. So, under these circumstances, centralized is better than distributed if $2p + 2^m(1-p) > 3$ holds true. Simplifying this condition, we get $2^m(1-p) > 3 - 2p$ has to hold true.

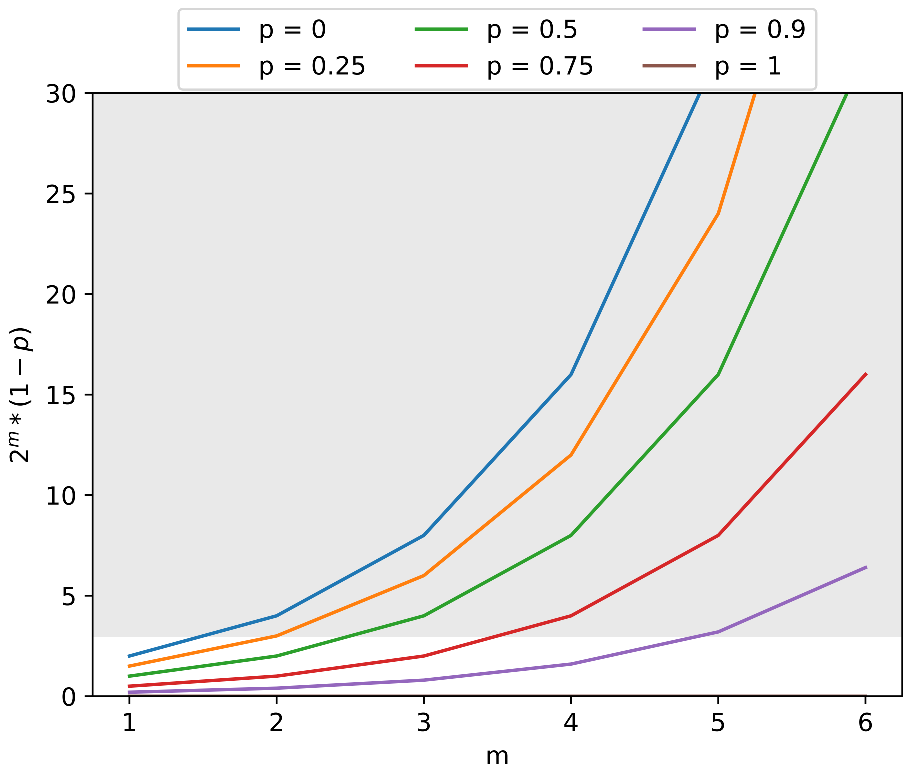


Figure 2.4: Variation of $2^m(1-p)$ against m and p

Now p being a probability value, its value always lies between 0 and 1, i.e.,

$$\begin{aligned} & 0 \leq p \leq 1 \\ \Rightarrow\ & 0 \leq 2p \leq 2 \\ \Rightarrow\ & 0 \geq -2p \geq -2 \\ \Rightarrow\ & 3 \geq 3-2p \geq 1 \end{aligned} \tag{2.2}$$

So, the condition $2^m(1-p) > 3-2p$ reduces to $2^m(1-p) > 3$.

In Fig. 2.4 we plot the variation of $y = 2^m(1-p)$ with respect to m and p and make the regions, where the condition holds true, in solid color. From the plot we see that the value of y does not fall in the colored region when $p = 1$, which is in conjunction with our earlier explanation. For other p values, when $m > 5$, the value of y falls into the colored region.

From these discussions, we conclude that in CAN, decentralized coordination is better than centralized coordination if

- There are only 2 CFs, and,
- Those CFs always agree with each other over their shared control parameter value (or, those CFs do not share any common control parameter).

In all other cases, the centralized coordination is better. That is why, in this thesis, we assume a centralized coordinator in CAN. This coordinator is referred as the Controller hereafter in this thesis.

2.4 Information Exchange between the Controller and CFs

In the centralized coordination, a CF cannot directly change a cell transceiver parameter, it has to request the Controller to do so. So, the CF has to convey its wish to the Controller regarding a control parameter in some way. On the other hand, if the Controller wishes to change a control parameter, it has to take the interests of all the CFs into account, otherwise, it might affect some KPIs (the ones which are managed by that set of CFs to whom the Controller did not pay interest) severely. So, while determining the optimal value of a control parameter, the Controller requires some information from all the CFs to understand their preferred choices. According to the CAN models proposed in this thesis, which is based on [14], a Controller needs two pieces of information from each CF to understand its preference: optimal configuration range set (OCRS) and utility function (UF).

2.4.1 OCRS

Being a learning agent, for a particular network state, a CF can determine the range of values of any ICP for which its objective will be optimum or close to optimum. This set is called the OCRS of that ICP. An OCRS has the following structure: [min_value, max_value] where min_value and max_value denotes the lower and upper bound of the set respectively. If the value of that ICP lies between min_value and max_value, value of the KPI lies within a certain percentage of the optimal value. This percentage is called the KPI Optimality Spread (KOS) and is denoted by β. For example, let us assume that the objective of a CF is to maximize the KPI value, and, for one ICP, the OCRS is [a, b]. If the optimal value of the KPI is v, then for any ICP value c, $a \leq c \geq b$, the KPI value will be $\geq \beta \times v$.

2.4.2 UF

In CAN, objectives of different CFs have different units of measurement. For the Controller to understand and compare among these different objectives, it is beneficial to convert all of them in the same predefined scale. This scale can either be provided by the MNO or by the Controller. Example of one such a scale is [0:10], where 0 means the lowest and 10 means the highest objective value. Utility function (UF) is a function which maps an ICP value to this [0:10] scale. A UF is denoted by $f(p)$, where p means the configuration value and for this p, $f(p)$ provides the utility of the CF in the predefined scale. To do so, $f(p)$ maps the objective obtained from p (which is c) in a certain network state to the predefined ([0:10]) scale using:

$$f(p) = \frac{c - c_{min}}{c_{max} - c_{min}} = \frac{c - 0}{10 - 0} = \frac{c}{10}$$

where c_{max} and c_{min} are respectively the maximum and minimum possible values of attainable objectives. For example, if for a value p_1 objective of the CF is c_1, then the utility value corresponding to p_1 is $\frac{c_1}{10}$.

2.5 CAN Workflow

2.5.1 Workflow of a CF

Each CF is an independent learning agent which makes decisions on its own based on its learning experience. For a certain network state, the CF is able to generate the OCRS and UF for each of its ICPs. As soon as there is a change in the network state, the CF recalculates the OCRSs and UFs, ensuring its preferences are always up-to-date.

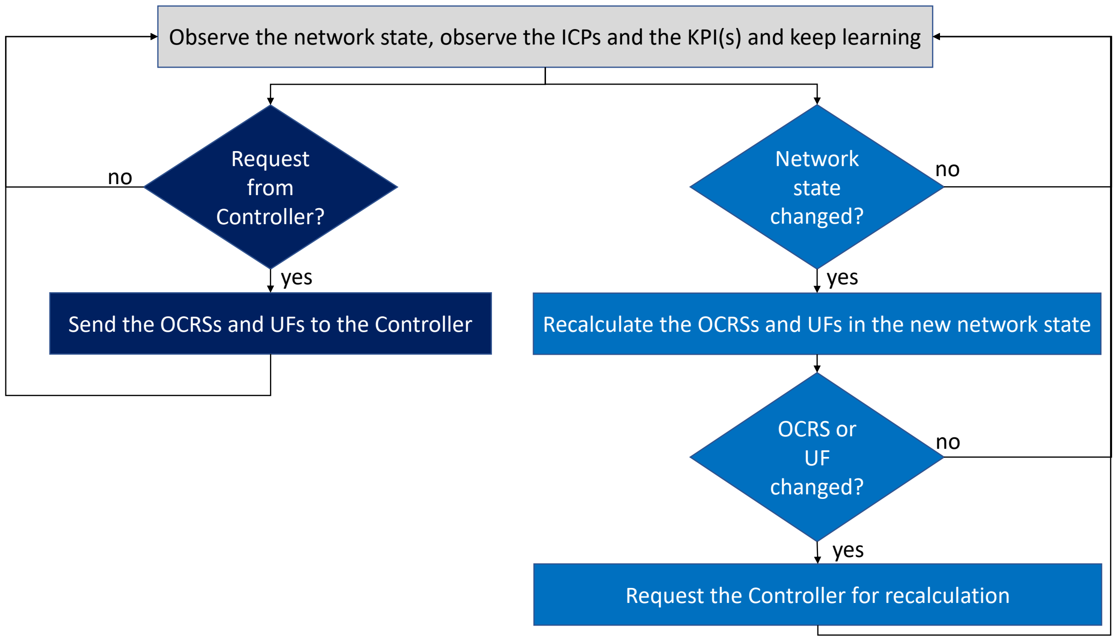

Observe the network state, observe the ICPs and the KPI(s) and keep learning
Request from Controller?
no
yes
Send the OCRSs and UFs to the Controller
Network state changed?
no
yes
Recalculate the OCRSs and UFs in the new network state
OCRS or UF changed?
no
yes
Request the Controller for recalculation

Figure 2.5: Workflow of a CF

The workflow of a CF can be visualized as two parallel processes as shown in Fig. 2.5: one constitutes internal computation and the other constitutes external communication.

internal computation: whenever the CF detects a change in network state, it recalculates the OCRSs and UFs for all its ICPs. If the CF finds that for any ICP the OCRS or UF changed, it requests the Controller to recalculate that parameter.

external communication: whenever the CF gets a request from the Controller, it sends the relevant OCRS and UF to the Controller.

So basically, each CF has the following properties:

- Each agent can learn and decide the best action for itself in a dynamic environment.
- No agent can communicate with each other and no one has a complete knowledge of the system.
- Some or all of these agents share the same resources which give rise to conflicts of interests among them.
- Each agent tries to optimize its own target or goal simultaneously, and the concept of a common or team goal does not exist.

2.5.2 Controller Workflow

The controller takes OCRSs and UFs from CFs as its input and returns a single optimal values as its output. Workflow of the proposed controller consists of four steps as depicted in Fig. 2.6. Whenever a CF wants to change a certain parameter, it sends a request to the controller. Upon receiving the request, the Controller sends requests to all CFs, asking them to send their corresponding individual OCRSs and UFs. After receiving information from all interested CFs, the controller calculates the optimal value for that parameter and makes the necessary change in the network.

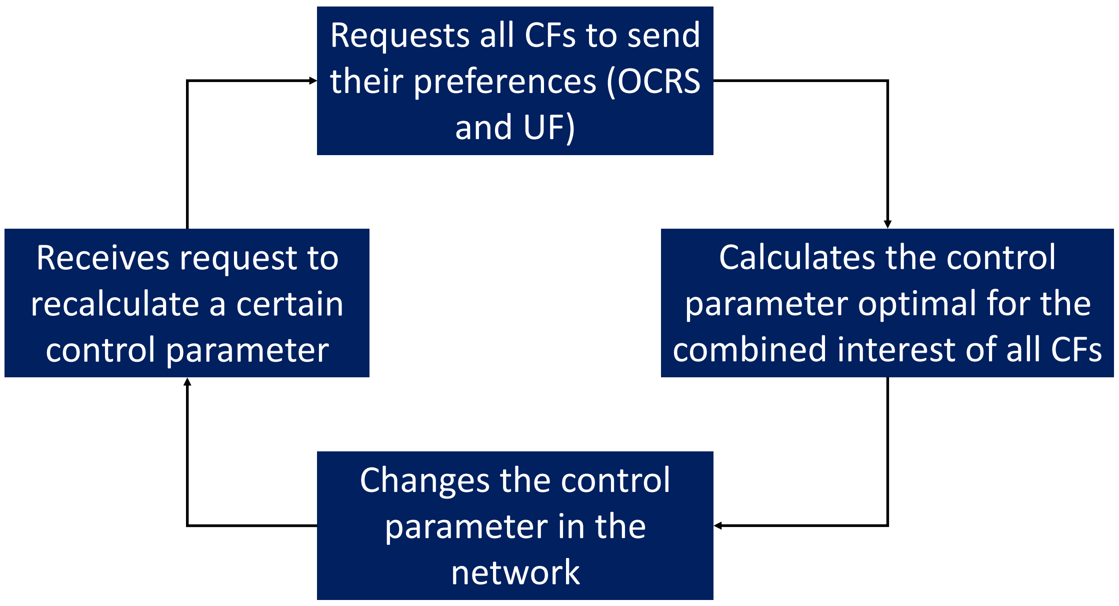

Requests all CFs to send their preferences (OCRS and UF)
Calculates the control parameter optimal for the combined interest of all CFs
Changes the control parameter in the network
Receives request to recalculate a certain control parameter

Figure 2.6: Workflow of the Controller

2.5.3 End-to-end workflow

The end-to-end operation of the CAN is a continuous loop in which a CF evaluates the network state, determines best possible values for its ICPs and sends its requirements to the Controller if necessary. The system starts with some initial values of the control parameters set by the MNO based on her previous experience. In the operational mode, every CF periodically checks if the current configuration is optimal for itself. If the CF finds the current configuration to be optimal, it does not communicate with the Controller and only continues its learning until the next periodic checking. However, if the CF finds that the current configuration is not optimal for itself, it triggers a process of configuration recalculation described below:

- After a CF detects a new optimal configuration other than the current one, it identifies those control parameters whose values need to be recalculated.
- Then the CF sends request(s) to the Controller to recalculate the identified control parameter(s). The CF, which sends such a request, is called the Requesting CF of that particular control parameter.
- After receiving a request from the Requesting CF, the Controller sends requests to all CFs in the system to send their latest OCRS and UF on the control parameter.
- All the CFs send back their OCRSs and UFs to the Controller.
- Based on the received information, the Controller calculates the optimal value of that control parameter.

The end-to-end workflow is shown in Fig. 2.7.

2.6 Discussion on CAN

As we already mentioned in Chapter 1, CAN has several advantages over the existing rule-based network management system. However, from the implementation perspective,

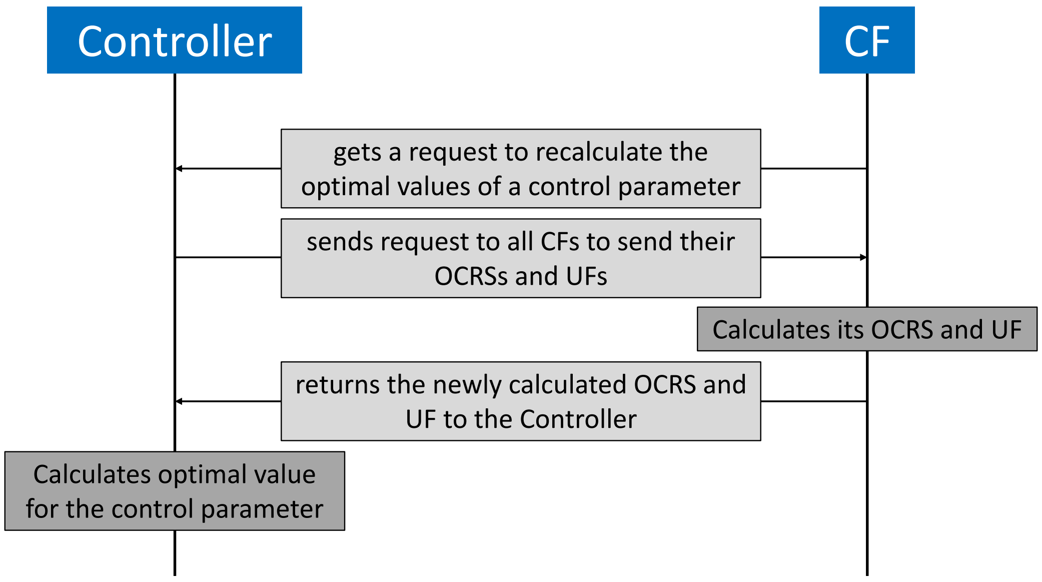

Controller
CF
gets a request to recalculate the optimal values of a control parameter
sends request to all CFs to send their OCRSs and UFs
Calculates its OCRS and UF
returns the newly calculated OCRS and UF to the Controller
Calculates optimal value for the control parameter

Figure 2.7: End-to-end workflow for optimal configuration calculation

it still has some challenges which are discussed in this section. It is to be noted that these are some supplementary discussions that do not bear any impact on the main content of this thesis. The challenges are:

(1) some control parameters are shared among quite several CFs, like TXP which is shared among 4 CFs (coverage and capacity optimization (CCO), MLB, energy savings (ES), Coverage Hole Management). So, the chances of the Controller frequently receiving a request for the TXP recalculation are very high. On the other hand, there is no lower bound specified for the amount of change in OCRS needed to trigger a recalculation. So, even when there is a small, insignificant change in the OCRS, a CF can still trigger the whole parameter recalculation process. Not only the Controller consumes time and energy each time a parameter is calculated, the Controller also changes the value of the parameter in the network. This type of frequent change of a parameter value may compromise the stability of the network.

(2) since we consider a multi-vendor environment, different CFs coming from different vendors may not be equally trustworthy. A CF has enough motivation to consecutively send multiple requests to the Controller about an ICP until its value matches the CF's expectations. A significant amount of time and energy can be wasted in this manner.

To minimize the unnecessary recalculation of a control parameter (and therefore, more time and energy) in a real-life scenario, we propose the following two features which can be implemented alongside CAN:

t-value rule which minimizes the recomputation of a control parameter. Using this rule, the Controller puts a lower bound on the minimum number of requests ($t - value$) necessary to trigger the recalculation of a parameter. For example, if 4 CFs share TXP and $t - value$ is 0.5, then TXP is recalculated by the Controller when at least $4 * 0.5 = 2$ CFs send requests.

m-value rule which limits the number of requests a CF can make within a given time period. For example, if $m - value$ is 3, then, within a fixed time interval, a CF can only send the total m number of recalculation requests to the Controller.

If these two rules are combined, then the variable $t * \frac{1}{m}$ quantizes the energy consumption due to parameter recalculation. The lower the value of this variable, the better the energy savings in parameter recalculation. In this way, the challenges in CAN can be overcome and it can be operated in a more energy-efficient manner.

2.7 Conclusion and Key Takeaways

In this chapter, we gave a conceptual overview of CAN from the perspective of management of mobile network. Using Prisoners' Dilemma, we mathematically established that coordination is needed for CAN. In this context, we evaluated two possible types of coordination: centralized and distributed. We found, both mathematically and experimentally, that centralized coordination is better for CAN. Based on our observation, we proposed the centralized coordination based architecture for CAN which is used throughout the rest of this thesis. We highlighted the workflow of each key component in the proposed framework and their interaction with one another. We concluded the chapter by covering some constraints in our proposed design and different implementation details to overcome those constraints in real-life scenarios.

List of Abbreviations Used in This Chapter

CCO	coverage and capacity optimization	***MLB***	mobility load balancing
CFs	cognitive functions	***OCRS***	optimal configuration range set
CIO	cell individual offset	***RET***	remote electrical tilt
ES	Energy Savings	***RLF***	radio link failure
ICPs	input control parameters	***SON***	self organizing network
KOS	KPI Optimality Spread	***TTT***	Time-To-Trigger
KPIs	key performance indicators	***TXP***	transmission power
MAS	multi agent system	***UF***	utility function

3. Simulator and Evaluation Environment Description

This chapter describes the fundamental components and organization of the simulation environment used throughout the thesis. Along with that, we also detail the novelty required to implement the proposed CAN modules, their functionalities and experimentally demonstrate their feasibilities. It is to be noted that while discussing these two topics, in both cases, we focus on the high-level logic of the components and not on the specific implementation details. The majority of the content in this chapter is from the following journal paper by the author:

[18] **A. Banerjee**, S. S. Mwanje, G. Carle. On the implementation of cognitive autonomous networks. Internet Technology Letters, 4(6), e317. Wiley, 2021.

In addition to the content described in the publication, this chapter includes the following contributions by the author:

- Description of the radio propagation model used in the simulation scenario.
- Description of UE states, UE transceivers and UE mobility model used in the simulation scenario.

3.1 Simulator Description

The simulator mimics the behavior of a real-life mobile network where the operator (here, user of the simulator) can customize certain aspects of the network. To make the simulated mobile network as realistic as possible, the simulator takes multiple inputs from the user which contain: (i) geographical area and size (in sq km), (ii) number of cells, (iii) cell positions, (iv) parameters of cell transceivers, (v) radio propagation model, (vi) number of user equipments (UEs), (vii) UE mobility model, (viii) observable KPIs (at cell or UE level) and (ix) observation period. Description of these user inputs, along with the values used in the simulation, are covered in Section 3.2 in detail. After the user inputs these values, the simulator creates a physical scenario (building and street distribution), populates the location with mobile UEs, creates UE activities, and starts the simulation. Based on the user-specified simulation period, network configurations and KPIs can be observed at the simulator output. While a simulation is going on, similarly to an MNO, the user of the

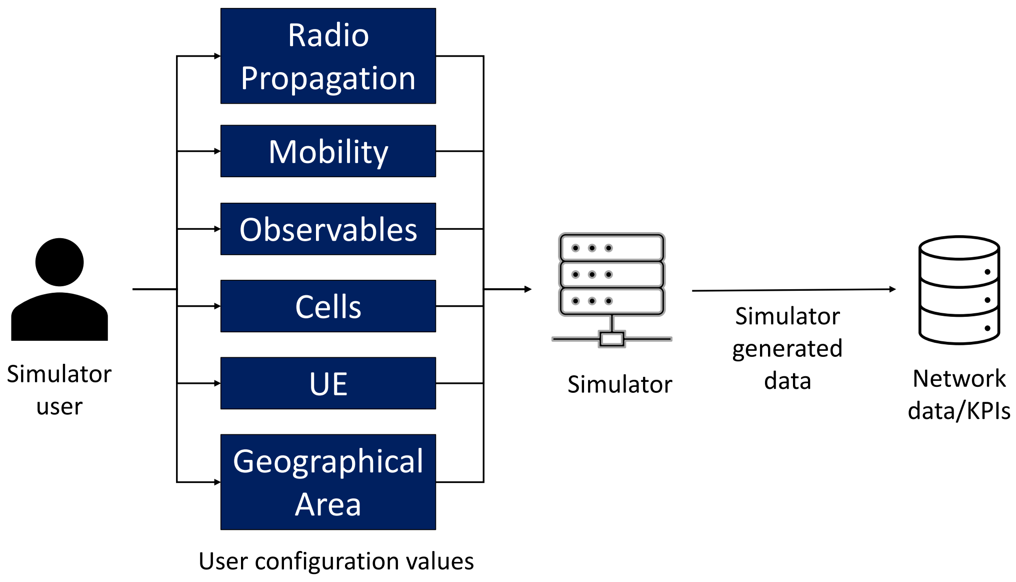


Figure 3.1: Implementation details of the simulator

simulator can make changes in the simulation configurations (described in Section 3.2) and observe the changes in the network via network data collected at the output. This is a brief overview of the simulation environment which is shown in Fig. 3.1.

3.2 Simulator Configuration

Throughout the thesis, we use the same simulation environment and configuration to maintain the homogeneity among the generated results. These configurations are discussed below in detail:

3.2.1 Scenario

For our evaluation scenario, we consider an area of 4 sq. km consisting of long straight streets with strict grid patterns crossed by wide avenues and square blocks (Fig. 3.2). This type of building distribution can be found in big cities in the world (Barcelona, Manhattan, etc.), in most of the European cities and it is widely accepted as a standard city model [19]. The streets generally have a width of 20 m, of which the central 10 m are destined for carriageways and 5 m on each side are designated for the sidewalks for pedestrians. The dimensions of the blocks are given by the distance between the longitudinal axes of the streets and the width of these roads and the length (or breadth) of one block (including roads on all sides) equals 133.3 m. Since the standard width of a road is 20 m, the blocks (without the roads) are formed by quadrilaterals of 113.3 m. The height of the buildings varies between 16 - 20 meters, and in our simulation scenario, those values have been chosen randomly. All these values are taken from the real-life Barcelona city model[1].

3.2.2 Cells

3.2.2.1 Number of cells and their positions

The cellular network consists of 5 cells deployed in the 4 sq. km area, where the cells are deployed as shown in Fig. 3.3 such that the central cell is surrounded by the remaining cells. All the cells are 1-sector macro cells (2GHz), and from here onward the term cell refers to the coverage area of the centrally situated single radio transceiver. Unless specified otherwise, throughout the Thesis all the network measurements data are taken in the central cell.

[1]https://es.wikipedia.org/wiki/Distrito_del_Ensanche

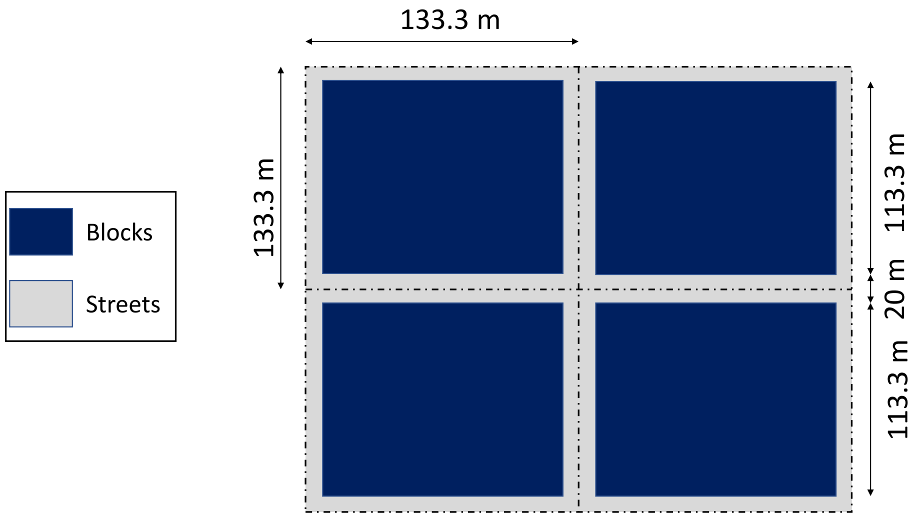


Figure 3.2: Scheme of the blocks and the streets

3.2.2.2 Cell transceivers

The default values used for antenna-specific parameters for a cell transceiver are given in Table 3.1. However, in a later Chapter we'll see that during some KPI measurements, these values need be varied, but, unless specified otherwise, these are the values used for all the cell transceivers in the simulations.

Table 3.1: Default values of cell transceiver parameters

Tx Power (dBm)	Bandwidth (MHz)	Scheduler
46	20	throughput fair
Antenna height (m)	Azimuth compass direction	Antenna Gain (dBi)
18	0	6
Mechanical downtilt (°)	Remote electrical tilt (°)	Azimuth & Elevation beam spread (°)
0	0	0, 0

3.2.3 Radio Propagation Model

In mobile networks, the power of the signal received at one transceiver depends not only on the TXP of the sending transceiver but also on a number of environmental factors like the distance between the transceivers, obstacles on the path, fading, and so on. To make a simulation study as realistic as possible, incorporating an accurate propagation model(s) between the sending and receiving transceiver is a crucial step. We use a realistic radio propagation model, the WINNER+ model [20], in our simulation. Since our simulation environment is an urban macro cell, non line of sight (NLOS) and line of sight (LOS) propagation are common, since the street level is often reached by single diffraction over the rooftop [20]. Knife edge diffraction effects are computed for the leftmost and rightmost building edges as well as the trailing roof edges from the perspective of the transceiver. The computed effects of knife edge diffraction modulate the path loss between the two extremes given by the respective Winner+ LoS and NLoS formulas. For the NLOS scenario, the path loss is calculated using:

$$\mathrm{PL} = (44.9 - 6.55\log_{10}(h_{BS}))\log_{10}(d) + 5.83\log_{10}(h_{BS}) + 14.78 + 34.97\log_{10}(f_c)$$

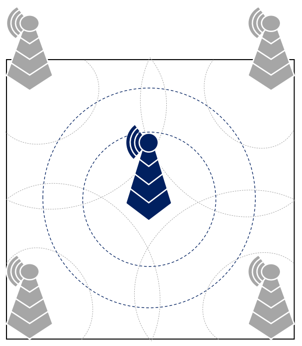


Figure 3.3: Cell placement

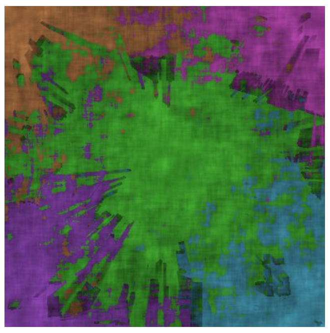


Figure 3.4: Radio coverage map of the cells

For the LOS scenario, the path loss is calculated using:

$$\mathrm{PL} = 22.7\log_{10}(d) + 27.0 + 20.0\log_{10}(f_c)$$

where d is the distance between the antenna and the UE, f_c is the center frequency (2 GHz) and h_{BS} is the actual antenna height.

3.2.4 UEs

The UEs are distributed randomly over the chosen simulation area of 4 sq km and they can move freely within this area without any restriction. A UE is characterized by a UE transceiver that contains all the radio-related parameters. During the simulation, a UE can be in one of the six states as shown in Fig. 3.5. Transition from one state to another state is possible by executing certain state transition events (discussed in Section 3.2.4.2) associated with a corresponding timer (discussed in Section 3.2.4.3).

3.2.4.1 UE states

There are six possible states for a UE which are described below:

RLF: if a UE is in this state, it means that the UE transceiver is not connected to any cell transceiver.

cell selection signalling (CSS): this state corresponds to the initial signaling procedure performed by the network before the UE is connected to any of the cell transceivers.

radio resource control connected (RRCC): it signifies that the UE transceiver is connected to one of the cell transceivers and it is entitled to all possible radio resources provided by the cell.

radio resource control idle (RRCI): A UE transceiver is not scheduled in this state. It indicates that the UE buffer is empty and it has no data to transmit.

handover signalling (HOS): this state models the signaling procedure by the network when certain handover conditions are met for a UE. If the condition, that caused the transition to this state, remains valid for a certain monitoring time period then the UE transceiver transitions to the RRCC state in the new cell.

radio link failure signalling (RLFS): This state models the signaling procedure executed by the network before the RLF state is reached.

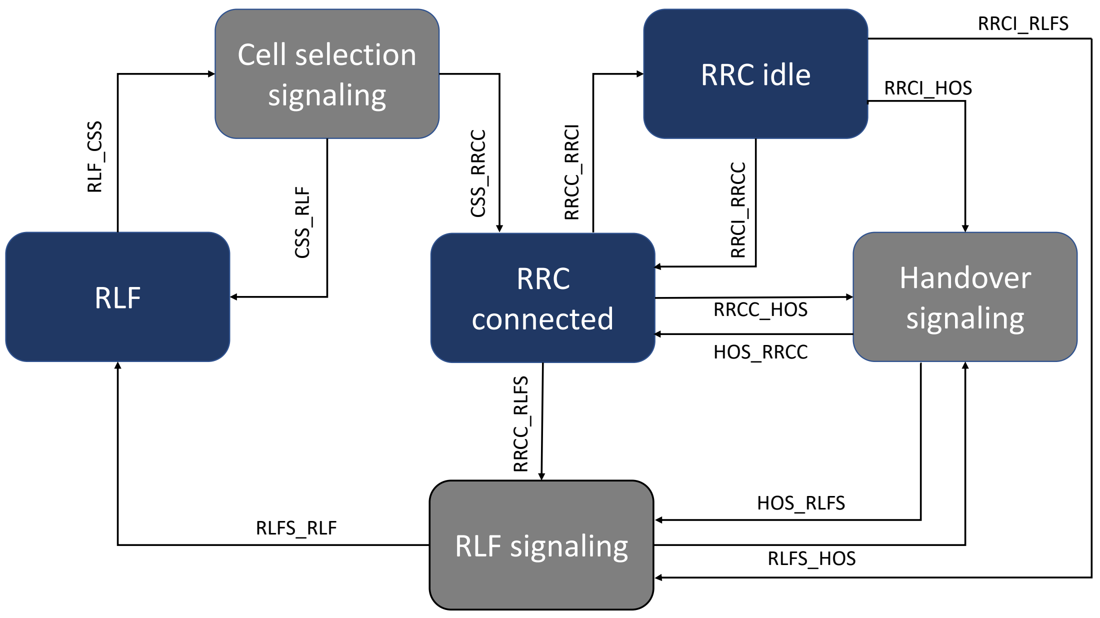


Figure 3.5: UE state machine diagram

3.2.4.2 UE state transition events

The events responsible for the change of UE states are briefly explained below:

events from RLF: from RLF, the only possible transition for a UE transceiver is the CSS. This happens when the event RLF_CSS is triggered.

events from CSS: from CSS, there are two possible states for transition: RLF or RRCC. These transitions take place when the events CSS_RLF and CSS_RRCC are executed respectively.

events from RRCC: from RRCC, three transitions are possible: (i) transition to RRCI when the event RRCC_RRCI is fired, (ii) transition to HOS when the event RRCC_HOS is fired, and, (iii) transition to RLFS when the event RRCC_RLFS is fired.

events from RRCI: from RRCI also, three transitions are possible: (i) transition to RRCC when the event RRCI_RRCC is fired, (ii) transition to HOS when the event RRCI_HOS is fired, and, (iii) transition to RLFS when the event RRCI_RLFS is fired.

events from HOS: from HOS, there are two possible transitions: RRCC and RLFS. These transitions occur when events HOS_RRCC and HOS_RLFS are triggered respectively.

events from RLFS: from RLFS, the UE transceiver can either go to HOS (event RLFS_HOS) or RLF (event RLFS_RLF).

3.2.4.3 UE transceiver timers

All signalling states of the UE transceiver are modeled as a specified time required to execute a particular procedure. These timers are defined to validate whether an event is a random occurrence or it exhibits a consistent behavior. There are seven timers used in the simulator:

cell_selection_signalling_time: this timer specifies the signalling time required by the network when a UE transceiver, which is in the RLF state, tries to connect to the network. In the simulations, the default value of this timer is 0.1 s.

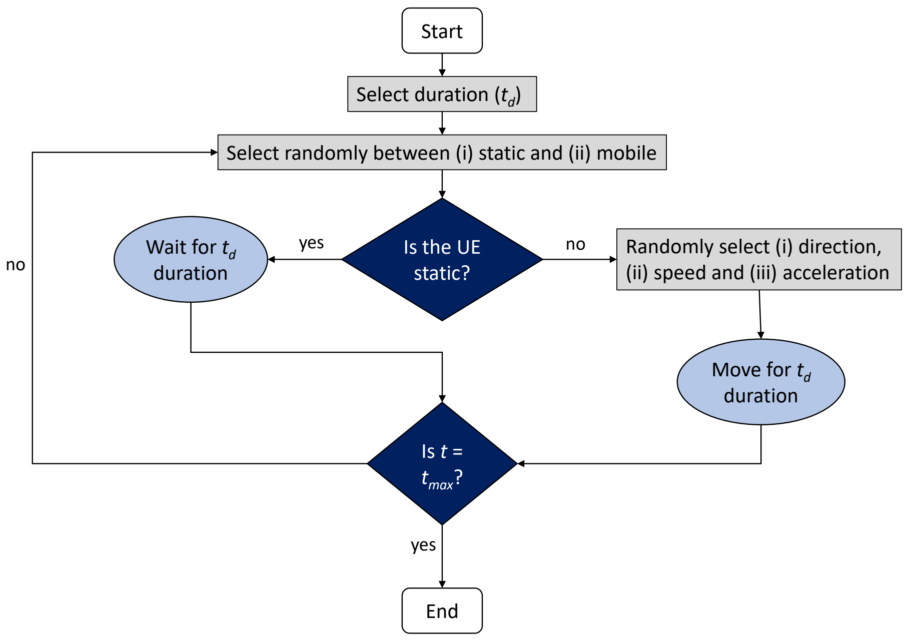


Figure 3.6: Sequence of events for random mobility model

handover_signalling_time: this timer refers to the allowed time when a handover command is being executed by the network. During this time the UE transceiver will be in an HOS state. In the simulations, the default value of this timer is 0.1 s.

RLF_signalling_time: this timer corresponds to the signalling time, required by the network, to perform radio link failure. During this time, the UE transceiver will be in the RLFS state. In the simulations, the default value of this timer is 0.1 s.

late_HO_detection_timer: this timer specifies the time interval threshold to detect a late handover. In the simulations, the default value of this timer is 5 s.

early_HO_detection_timer: this timer specifies the time interval threshold to detect an early handover. In the simulations, the default value of this timer is 5 s.

pingpong_HO_detection_timer: this timer specifies the time interval threshold to detect a ping-pong handover. In the simulations, the default value of this timer is 5 s.

wrongcell_HO_detection_timer: this timer specifies the time interval threshold to detect a handover to a wrong cell. In the simulations, the default value of this timer is 5 s.

3.2.5 Mobility Model

A mobility model characterizes the pattern of mobile UEs, portraying the change of UE location over time. In this thesis, we use a random mobility model as it is the most generic and widely accepted mobility model in the study of mobile networks [21].

3.2.5.1 random mobility model

The general algorithm for the random mobility model is depicted in Fig. 3.6. In this model, a UE decides randomly to either move or be static for a certain time duration (t_d) chosen randomly. After the duration is over, again the UE selects randomly between moving or being static for another randomly chosen time duration. This continues until the simulation is finished.

Each time a UE randomly selects to be static, it remains stagnant in its last recorded position. If a UE chooses to move, in the next step, it also randomly selects: (i) speed, (ii) acceleration and (iii) direction of the movement. For speed, the allowed range to choose from is 4 - 80 km/h (which covers all the possible speeds from pedestrians to cars). For acceleration, the UE can randomly choose between 0 - 3 m/s^2. Values of the speed and acceleration have been chosen to keep the simulation as realistic as possible. Since the simulator supports only 2D movement of a UE, possible directions of movement are +x, -x, +y and -y.

3.2.5.2 wrap around

A wrap-around is implemented in all the mobility models to keep the number of UEs constant in a simulation scenario. When a UE reaches a border of the 4 sq kilometer area, it turns in the opposite direction to ensure that it stays within the designated area. Otherwise, the UEs would continuously move away from the study network and comparison between scenarios with different numbers of UEs would not be possible.

3.2.6 KPI Observation

As already mentioned earlier, all the KPI related measurements are taken in the central cell (green in Fig. 3.4). In the scope of the thesis, the collected KPIs are listed below; however, it is worth mentioning that not all KPIs have been used in all the chapters - based on the necessity, different sets of KPI measurements have been taken.

total_number_of_ues: this number corresponds to the total number of active (RRCC) and idle (RRCI) UEs in the cell.

number_of_idle_ues: this number corresponds to the total number of UEs in the cell which are in the RRCI state.

number_of_connected_ues: this number corresponds to the total number of UEs in the cell which are in the RRCC state.

cell_load: cell load is the ratio between the required radio resources (to satisfy the bit-rate requirement of each RRC connected UE Transceiver) and the number of available radio resources. Theoretically, this can be varied from 0 to ∞. If this value is greater than 1, it means the cell transceiver is overloaded.

cell_throughput: this KPI reports the achieved throughput in bits/sec by a cell transceiver after serving all the RRC connected UE transceivers.

average_user_throughput: this is the achieved average user throughput in bits/sec per UE transceiver. In other words,

average_user_throughput=cell_throughput/number_of_connected_ues

handover_attempts_count: this specifies the total number of handover events originating from a cell transceiver to all other cell transceivers.

successful_handovers_count: it is the total number of successful handovers originating from a cell transceiver to all other cell transceivers.

handover_drops_count: it is the total number of handover drops originating from a cell transceiver to all other cell transceivers.

late_handovers_count: it specifies the total number of late handovers originating from a cell transceiver.

early_handovers_count: it specifies the total number of early handovers originating from a cell transceiver.

wrong_cell_handovers_count: it counts the total number of wrong cell handovers originating from a cell transceiver.

pingpong_handovers_count: it specifies the total number of pingpong handovers originating from a cell transceiver.

radio_link_failures_count: it specifies the total number of RLF events originating from a cell transceiver.

3.2.7 Simulation Data Analysis

The simulator used in this thesis is a Nokia Bell Labs proprietary simulator and it is not publicly available. Although the simulator cannot be used by external users, we published the dataset[2] online so that the results of this research can be reproduced. So, to establish the trustworthiness of the simulator, in this Section, we discuss the precision of the data generated from the simulation using confidence intervals. From that dataset, we plot variations of three KPIs (successful handover percentage, cell load, and average user throughput[3]) against TXP in Fig. 3.8, 3.7, 3.9. From the Figures, we see that the variation in the collected data for each TXP value is very low. Since we collected the data over a long period of time and the variation is still low, it proves that the collected data is quite consistent. Given below are the explanations for the visible KPI variation against the control parameters to prove that the simulator-generated data matches the behavior of a real life network. It is to be noted that to observe the changes in the network better, the TXP in all the five cells is varied simultaneously by the same amount.

3.2.7.1 Load vs TXP

When the TXP of a cell is increased, its coverage area increases, which increases the load of the cell. As shown in Fig. 3.7, the cell load increases with the TXP to 40 dBm. After that, the coverage and capacity of the cell start decreasing with TXP because of (i) the increasing interference from all the neighboring cells, and, (ii) the increasing number of handovers as the overlap between two cells becomes clearer. Beyond 60 Dbm, the cell load becomes almost constant as these effects start canceling each other.

3.2.7.2 Successful handovers vs TXP

When the TXP of a cell is increased, its coverage area increases, which means that more number of handover events originate from the cell. As we see from Fig. 3.8, beyond 50 dBm, the cell coverage does not increase much further, keeping the number of handovers almost constant.

3.2.7.3 User throughput vs TXP

When the TXP of a cell is increased, its coverage area increases, which increases the load of the cell and decreases the average downlink throughput. This is why initially the throughput decreases with the TXP. After a certain point, as load decreases with TXP (as explained earlier), the average throughput increases. Then, as the interference from the neighboring cells increases with increasing TXP, the throughput does not increase further and becomes almost constant.

[2]simulator generated dataset available online: https://tinyurl.com/mnma-dataset

[3]these are the three KPIs which have been used the most in this thesis

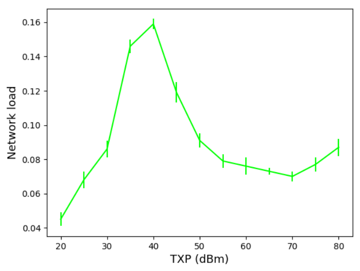


Figure 3.7: Cell load vs TXP

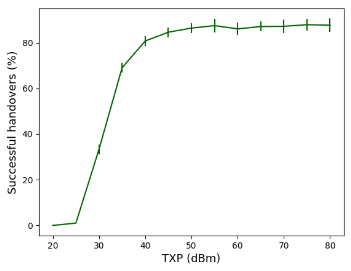


Figure 3.8: Successful handovers vs TXP

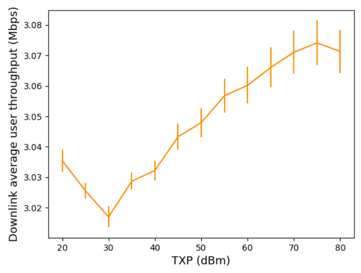


Figure 3.9: Average user throughput vs TXP

3.3 CAN Extension for the Simulator

3.3.1 Objective

The main idea behind the implementation of CAN is to show that CFs can function reactively in a dynamic environment. For example, let us assume that at time t_0 in a cell, number of connected users: n_0, downlink average user throughput: T_0 Mbps, TXP: x dBm, RET: d deg. Let us assume that at a time $t_1 > t_0$, the number of connected users suddenly increases to n_1, $n_1 > n_0$ and because of the sudden increase in the number of connected users, throughput drops to T_1, $T_1 < T_0$. Now, the CF which is responsible for optimizing throughput is expected to propose new values of TXP and RET to the Controller, as soon as throughput drops so that throughput is restored to T_0 or close to T_0. Our main aim is to demonstrate that this expectation can be achieved by implementing CAN.

3.3.2 Implementation Overview

To implement CAN in a simulation environment, the extended simulator shown in Fig. 3.10 is used. Different CFs have been implemented as different methods in a Python module, giving the flexibility to add any number of CFs in the simulation. Within the method, the ICP(s) and KPI(s) of the CFs are defined.

There is a continuous two-way interaction between the CAN module and the simulator:

(i) from the simulator, network-related data and KPIs always come to the CFs via the data processing module (DPM). The DPM has been implemented as an individual Python module that acts as a bridge between the simulator and the CF. It collects all the data from the simulator, processes the data, and sends only the relevant data to the CF. This data is used by the CF to improve its learning and performance.

(ii) as already mentioned previously, the Controller calculates the final optimal values based on the preferences of the CFs. The Controller then sends the value directly to the user input interface of the simulator, so that the necessary configurations are changed without requiring any help from the simulation user.

3.3.3 CFs and Controller Implementation

Our entire thesis discusses different aspects of CAN: coordination issues, trust issues, and orchestration issues. Before discussing those aspects, it is necessary to show that CAN exhibits the expected behavior. To demonstrate the complete workflow of CAN, we implement two CFs - MLB and CCO, and a Controller in the central cell of our simulation environment. To implement a CF, we use several logical blocks and a neural network (NN) which has 5 fully connected layers with 50 nodes in each hidden layer, Adam optimizer, and MSE loss function. While determining the optimal NN architecture for a CF, we started

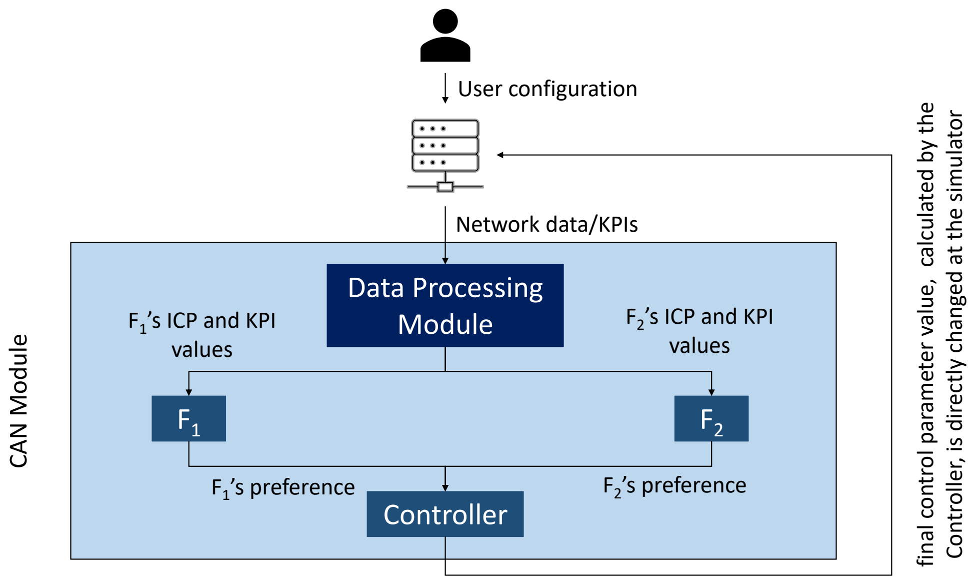

User configuration
Network data/KPIs
CAN Module
Data Processing Module
F$_1$'s ICP and KPI values
F$_2$'s ICP and KPI values
F$_1$
F$_2$
F$_1$'s preference
F$_2$'s preference
Controller
final control parameter value, calculated by the Controller, is directly changed at the simulator

Figure 3.10: Extended simulator for CAN implementation

with a relatively small NN and iteratively looked for an optimal architecture that provides the required performance with the least time and space complexity. While implementing a CF, network state[4] and the ICPs act as the inputs of the NN and the UF acts as the output of the NN. The NNs are trained periodically with the data collected from the simulator. After the NN is trained, it can predict the KPI value given a network state and ICP value. Along with NN, another logical block in Python is implemented to generate the UF. This logical block is called UF generator. In a given network state, UF generator provides different values of the ICP to the NN, collects the NN predicted KPI values, normalize them in a [0:10] scale and generates the UF.

Implementation of the Controller is straightforward. The functionality of the Controller is implemented as a separate method inside the CAN module. Throughout the thesis for all CAN related implementations, we used Python 3.7.8. and PyTorch 1.7. to implement the NNs.

3.3.4 Demonstration

After discussing the details CFs and Controller implementation, here we demonstrate that the CFs can learn and act dynamically and plot them in Fig. 3.11, 3.12, 3.13, 3.14. For each CF, we show two images sequentially - (a) how the CFs react when the network state (number of RRC connected UEs) changes (Fig. 3.11, Fig. 3.13), and, (b) the outcomes of their reactions which can be observed from the KPIs (Fig. 3.12, Fig. 3.14). We select five time instances and in each instance, we deploy a different number of UEs across the 4 sq kilometer area while the simulation is going on (orange bars in Fig. 3.11 and Fig. 3.13). For example, at the time t_0 and t_1, the number of connected UEs are 200 and 500, respectively. Each time the MLB detects a new network state, it calculates the optimal TXP for itself in the new network state (Fig. 3.11). For example, at the time t_0 TXP is 48 dBm. Since at t_1 the network state changes, the MLB calculates a new optimal TXP value: 55 dBm. To display the effectiveness of the learning capability of MLB, in Fig. 3.12 we plot two bars side-by-side: (i) the load value with the existing TXP value (blue bar), and, (ii) the load value if TXP is changed according to MLB's suggestion (green bar). Following the

[4]for this demonstration purpose, we use the number of RRC connected UEs as the sole parameter in the network state; later in the thesis we use a more complex and complete network state implementation

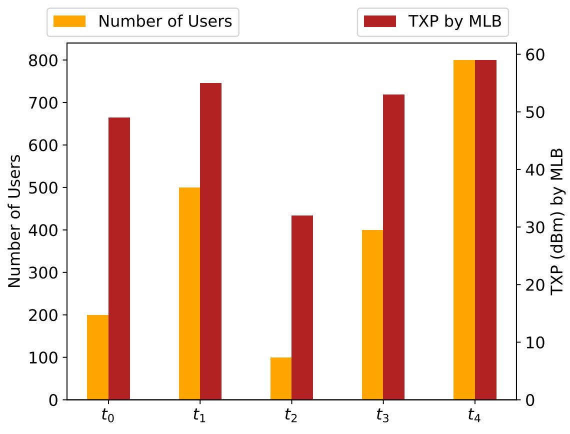


Figure 3.11: TXP propositions by MLB as the number of RRC connected UEs varies

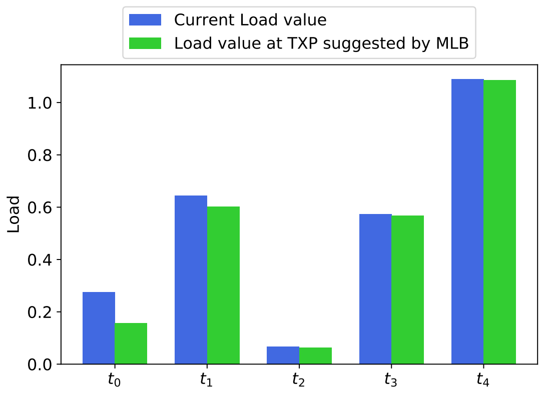


Figure 3.12: Network load variation as the number of RRC connected UEs varies

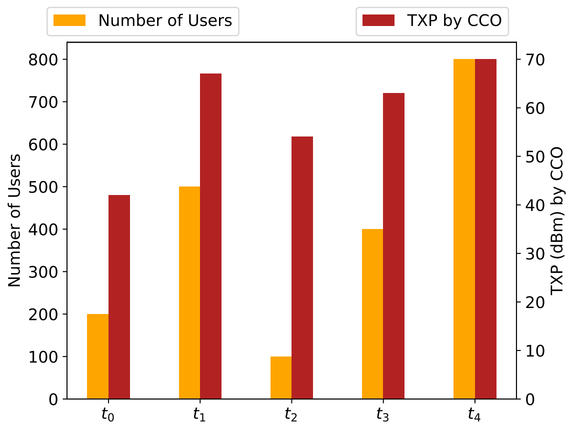


Figure 3.13: TXP propositions by CCO as the number of RRC connected UEs varies

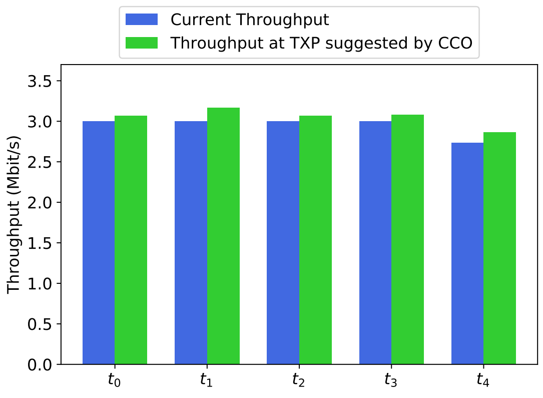


Figure 3.14: User throughput variation as the number of RRC connected UEs varies

same example, at t_1 with the existing TXP, load is 0.64, but, if TXP is set to 55 dBm (as suggested by MLB), load decreases from 0.64 to 0.6. We see from Fig. 3.12 that in all cases, load is reduced if TXP is set at the value suggested by MLB, i.e., it proves that the MLB is able to learn and react continuously in a dynamic environment.

Similarly, from Fig. 3.14, we see that the throughput always increases if the TXP is set as per CCO's recommendation. These plots show that the CFs we implemented can learn and work continuously in a dynamic environment to achieve their targets and both of them improve their assigned KPIs, which proves that our way of implementing the CFs is efficient and reliable.

From Fig. 3.11 and Fig. 3.13, we see that the TXP values suggested by MLB and CCO are always different in a certain time instance. As we already mentioned, in such cases, it is the responsibility of the Controller to select a value of TXP which is optimal for the combined interest of MLB and CCO. In Fig. 3.15 for each time instance, we plot three bars side-by-side: (i) TXP value which is proposed by MLB, (ii) TXP value which is proposed by CCO, and, (iii) the final TXP value which is calculated by the Controller. For example, from Fig. 3.15 we see that at time t_2, MLB suggested 31 dBm and CCO suggested 56 dBm as TXP value, but the Controller finally sets 45 dBm as the final value which is in between the CFs' suggested values.

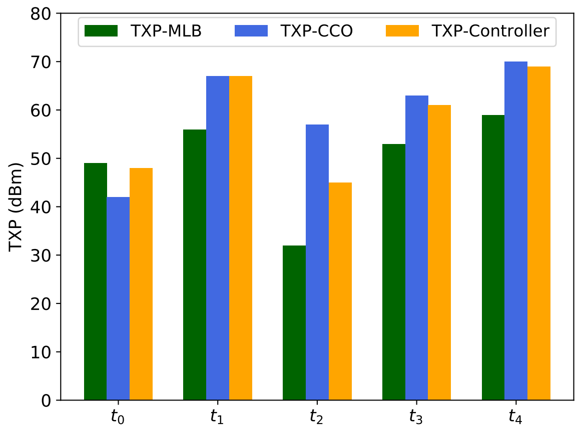


Figure 3.15: TXP values as calculated by the Controller

3.4 Conclusion and Key Takeaways

This Chapter discussed the simulation environment used to evaluate the ideas presented in this thesis. For the sake of uniformity, the same simulation environment has been used throughout the thesis. We presented the generic architecture of the mobile network simulator and its end-to-end workflow. We then discussed each element of the simulator in detail: how it works and its default configuration used in the evaluation environment. After that, we presented the extra extensions that we wrote in Python, to implement CAN in the simulation environment. We also experimentally demonstrated that the CAN module works as expected, making it competent to use in the rest of our thesis.

List of Abbreviations Used in This Chapter

CCO	coverage and capacity optimization
CSS	cell selection signalling
DPM	data processing module
HOS	handover signalling
KPIs	key performance indicators
LOS	line of sight
MLB	mobility load balancing
MNO	mobile network operator
NLOS	non line of sight
RLF	radio link failure
RLFS	radio link failure signalling
RRCC	radio resource control connected
RRCI	radio resource control idle
TXP	transmission power
UEs	user equipments

Part II

Coordination

4. Coordination: Problem and Related Works

In this chapter, we describe the problem related to coordination in CAN. For better understanding, we abstract CAN as a multi-agent system (MAS) and explain different types of conflicts in CAN. Then we cover the reasons why coordination in CAN is a challenging task to address. We also summarize prior research works, highlight the existing gap in state-of-the-art and prove the necessity of our proposed work. Content in this chapter is from the following paper by the author:

[22] **A. Banerjee**, S. S. Mwanje, G. Carle. Optimal configuration determination in cognitive autonomous networks. In *2021 IFIP/IEEE Symposium on Integrated Network and Service Management (IM).*pages 494-500. IEEE, 2021.

4.1 CAN as a Multi-Agent System

The MAS abstraction is an efficient method to simplify the architectures of complex systems. The concept of MAS abstraction can be applied across numerous research domains, which creates a vast state-of-the-art of this field. This motivates us to abstract can as MAS to look for existing solutions to the coordination problem explained in this chapter. The architecture of CAN can be visualized in a simpler way with the help of MAS abstraction. In this abstraction, each CF and the Controller act as an independent agent where all the CFs are in the same layer and the Controller is in a different layer. Following the coordination model discussed in the Section 2.5, all the CFs, in the MAS abstraction, should have the following four properties:

P1 Since each CF is a learning agent, it can learn and decide what is the best action for it by itself in a dynamic environment.

P2 The CFs do not communicate with each other and no one has complete knowledge of the system (i.e., aware of other CFs or any other entity apart from the Controller).

P3 Some or all of these CFs share the same resources (network control parameters) and there may exist conflicts of interests among them regarding the common resource sharing.

P4 The CFs try to optimize their targets or goals simultaneously, and the concept of a common or team goal does not exist.

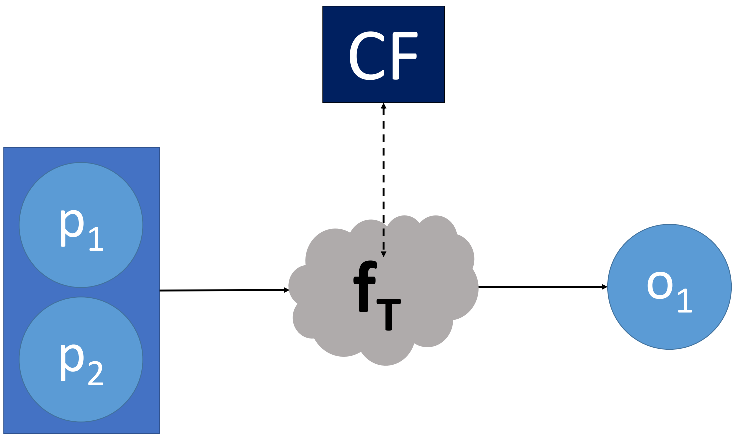


Figure 4.1: CF Modeling

When there is a conflict among the CFs (P3), it is the responsibility of the Controller to resolve the conflict. Under any circumstances, the decision of the Controller is always treated as the final and the Controller is the only one allowed to make any change to the network.

After giving a brief overview of the properties of a CF, let us elaborate on how a CF is modeled. Let us assume a CF F_1 with output o_1 snd ICPs: p_1 and p_2. In a mobile network, the relationship between o_1 and (p_1, p_2) can be visualized as $o_1 = f_T(p_1, p_2)$, where f_T is a transfer function that includes the mobile network and all other external factors into a single function. When the network state changes, o_1 also may change, and so, f is not constant and it changes when the network state changes. Basically we can assume that f_T is a function which is not constant, rather it is dependent on the network state. To state the obvious, f_T is not known to anyone beforehand and it is one of the intrinsic properties of the mobile network.

The functionality of the CF is to imitate and model the f_T. If we can model the CF such that it is the same as the f_T, we can:

- get the value of output o_1 for any (p_1, p_2) without trying (p_1, p_2) directly on the network, and,
- determine in advance the values of (p_1, p_2) for which o_1 is optimum while f_T is fixed.

To model f_T as accurately as possible, the CF always observes o_1 and figures out the dependence of o_1 on (p_1, p_2).

4.2 Conflicts in CAN

In Section 2.3 we established the necessity of the Controller in CAN. In this Section we describe the conflicts, which may arise in CAN and the Controller is supposed to resolve them. To explain these different types of conflicts and how they may appear in CAN, we assume the simplest CAN model (shown in Fig. 4.2) with 2 CFs (F_1 and F_2) and no Controller. The ICPs and outputs of F_1 and F_2 are as shown in Fig. 4.2.

In CAN, primarily three types of conflicts can be observed between F_1 and F_2 (shown in Fig. 4.4):

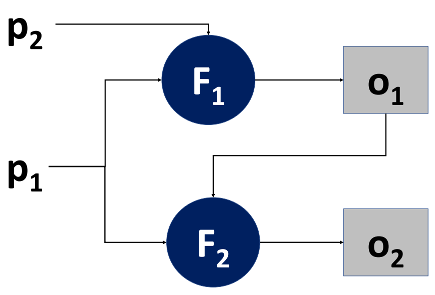


Figure 4.2: Example CAN model

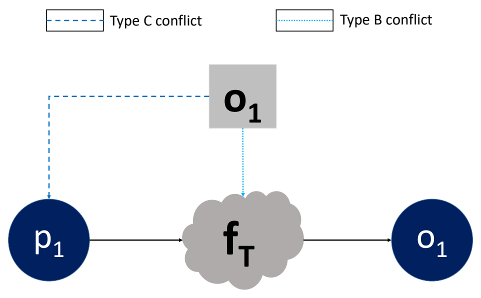


Figure 4.3: Difference between Type B and Type C conflicts

4.2.1 Configuration Conflict

this type of conflict, also known as Type A conflict, is induced by changes to a network control parameter. When two or more CFs share the same set of ICP(s), the chance of this type of conflict arises. The conflict emerges when F_1 requests to increase the value of p_1 and F_2 requests to decrease the value of p_1 either simultaneously or after a short time interval and vice versa. This type of conflict leads to an oscillatory behavior for parameter p_1 which is completely undesirable.

4.2.2 Measurement Conflict

this type of conflict, also known as Type B conflict, is induced by the change to a measurement. When the decision-making of one CF is subject to the influence of another CF, the chance of this type of conflict emerges. The conflict arises when F_1 and F_2 compete over different time scales where the actions of F_2 may interfere with the measurements which are needed by F_1 to make its decisions.

4.2.3 Characteristic Conflict

this type of conflict, also known as Type C conflict, is induced by the change of characteristic of a CF. The chance of this type of conflict arises when there is a logical dependency between the metrics influenced by F_1 and F_2. For example, if F_1 takes some actions which influence o_1, that in turn is an input of F_2, then this type of conflict emerges.

4.2.4 Difference between Type B and Type C Conflict

Although Type B and Type C conflicts may sound similar, there is a significant difference between these two types of conflicts which is depicted in Fig. 4.3. Let us assume that F_2 mimics the behavior of the transfer function f_T shown in the Fig. 4.3. Now, if o_1 influences p_1, then it is a Type C conflict between F_1 and F_2; if o_1 influences f_T, then it is a Type B conflict between F_1 and F_2.

4.3 Problem Description

As already stated in Section 4.1, it is the responsibility of the Controller to resolve these types of conflicts in CAN. As described in Section 2.4, even though the Controller can understand the actual preferences of a CF by the OCRS and UF while determining the optimal value of a control parameter which is shared among multiple CFs, a Controller encounters several challenges:

Unknown behavior of CFs: CFs are always learning, and they are periodically generating OCRS and UF based on their learning. Since their learning is influenced by the network

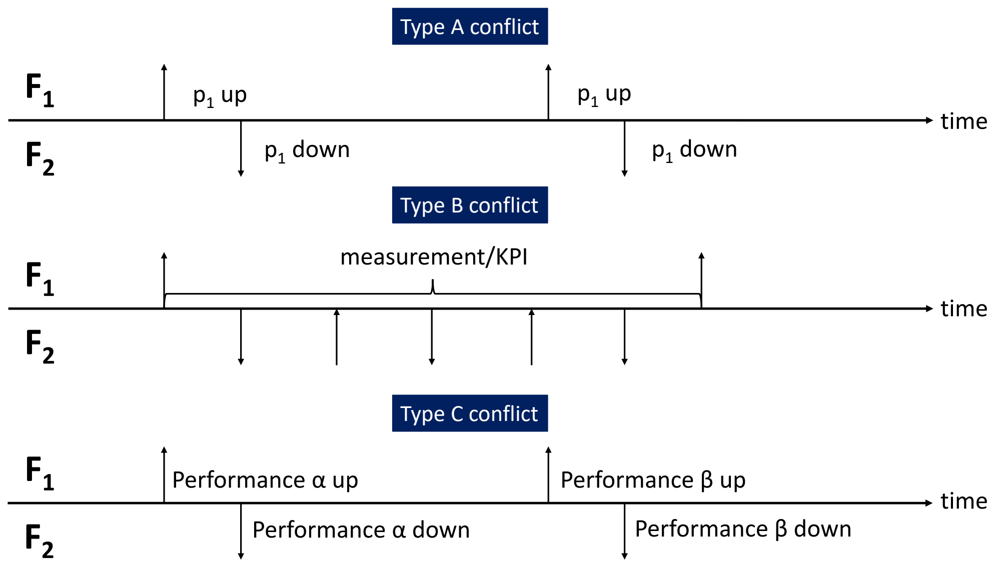


Figure 4.4: Types of conflicts in CAN

state which is not known beforehand, the behavior of a CF in the future can also not be predicted with certainty. Since the Controller cannot make any prediction of any CF behavior, it has to make decisions as soon as it receives information from the CFs.

Dynamic behavior of CFs: CFs are learning based on the external environment and as the external environment changes from time to time, and quite frequently in a 5G scenario [6], the behavior of a CF also changes with the changing environment. So, all the CFs exert a dynamic behavior and thus, the Controller also has to act reactively.

Calculation of combined interest: this is the most significant challenge faced by the Controller in CAN. When a control parameter is shared among multiple CFs, while determining the value of that control parameter, it has to find a compromise among their preferred choices which is a good balance for all.

So, in CAN the Controller should be able to overcome all three aforementioned challenges and resolve all types of conflicts among the CFs. In the next two chapters, we propose the design of Controllers which overcome those challenges and can resolve any type of conflict in CAN.

4.4 Related Works

In this section we cover the existing research works related to the coordination in CAN. At the start of this chapter, we abstracted CAN as a MAS which creates a wide state-of-the-art for this problem. However, not every MAS is relevant in this context; we consider only those cases where agents have similar properties like the CFs.

4.4.1 Related works in MAS

In this section we study already existing research works on MAS ([36, 28]) and the removal of conflicts (or, reaching consensus) in a MAS ([32, 37]).

In the MAS model used in this thesis (Section 4.1), we already highlighted that each CF should have four properties (P1, P2, P3 and P4 as described in Section 4.1). Based on the agent characteristics, we divide existing research works on MAS into several categories so that a combination of these features is covered in each category. These categories are

Table 4.1: Existing works on MAS features

P1	✓				✓	✓	✓				✓	✓	✓		✓
P2		✓			✓			✓	✓		✓	✓		✓	✓
P3			✓			✓		✓		✓	✓		✓	✓	✓
P4				✓			✓		✓	✓		✓	✓	✓	✓
	[23]	[24]	[25]	[26]	[27]	[28]	[29]	[30]	[31]	[32]	[33]	[34]	[35]	[36]	**X**

listed in Table 4.1. From Table 4.1 we see that there are a number of prior research articles which encompass one or some combinations of those four features described above, but there does not exist any paper which covers all the four features. Ours is the first one that considers a MAS with all of these four properties and proposes a solution for conflict removal (or, reaching consensus) in such a MAS.

4.4.2 Related works on controller

As mentioned earlier, the idea of a controller for network automation functions is not new and already exists in SON [38, 39, 40]. In these papers some external controllers have been proposed that work on top of the SON Functions and these controllers also coordinate and remove conflicts among SON Functions. SON coordination has extensively been researched in SOCRATES (2008) [41] and SOCRATES (2011) [42] also. But these coordination mechanisms or controllers cannot be used in CAN because these are rule-based controllers which follow predefined rules and rule-based coordination does not work in CAN.

4.5 Conclusion and Key Takeaways

In this chapter, we modeled CAN as a MAS to widen our search for existing research works related to coordination in CAN. After a thorough search of prior arts, we found an existing gap that justifies the necessity of our research. Along with that, we also summarized why coordination in CAN is a challenging task and worth addressing. In the next two chapters, we propose two new methods for coordination in CAN to address the Q2 described in Section 1.3.3.

List of Abbreviations Used in This Chapter

ICPs input control parameters

MAS multi agent system

OCRS optimal configuration range set

SON self organizing network

UF utility function

5. Conflict resolution among CFs

In this chapter, we propose a solution for coordination in CAN. We propose a Controller that overcomes all the challenges mentioned in Section 4.3 and resolves conflicts in CAN. We also performed a numerical analysis of the controller's performance. The results show that our proposed design fulfills all the requirements and is beneficial for use in real-life scenarios. The majority of the content in this chapter is from the following publication and patent application by the author:

[14] **A. Banerjee**, S. S. Mwanje, and G. Carle. Game theoretic conflict resolution mechanism in cognitive autonomous networks. In *2020 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS)*, pages 1–8. IEEE, 2020.

A COORDINATION AND CONTROL MECHANISM FOR CONFLICT RESOLUTION FOR NETWORK AUTOMATION FUNCTIONS
Anubhab Banerjee, Stephen S Mwanje
WO, PCT application no.: PCT/EP2020/061183, filed 10 April 2020

The discussion in this chapter expands on the content taken from the these with a more in-depth discussion on multi-agent resource allocation and numerical analysis.

5.1 Multi Agent Resource Allocation

Since we abstract CAN as a MAS, the coordination among CFs can be formulated as a multi agent resource allocation (MARA) scenario because:

- MARA is a research area that studies mechanisms for distributing a set of shareable resources among a group of agents [43]; similarly, in CAN also we focus on sharing control parameters among multiple CFs.
- in MARA, each agent has its preference over the resources it may receive, and the perceived quality of a chosen outcome (allocation of resources to agents) depends on these individual preferences [44]; similarly, in CAN, each CF has its preference over the network control parameters and KPIs.

A MARA scenario is defined as a triple $< \mathcal{A}, \mathcal{R}, \mathcal{U} >$. $\mathcal{A}$ is a finite set of n agents $\mathcal{A} = \{1, 2, .., n\}$. $\mathcal{R}$ is a finite set of m resources $\mathcal{R} = \{r_1, r_2, .., r_m\}$. $\mathcal{U}$ is a set of utility functions $\mathcal{U} = \{u_1, u_2, .., u_n\}$, one for each agent. Each $u_i \in \mathcal{U}$ is a mapping from the set of resources to the objective of each agent expressed in a non-negative real number, i.e., $u_i : 2^{\mathcal{R}} \rightarrow \mathbb{R}^+ \cup \{0\}$. Utilities are often expressed within a predetermined scale to make the comparison among them easier. A *utility profile* for a particular resource allocation combination J is a vector that contains the utility of all agents for this particular allocation J and is defined by $u(J) = (u_1(J), u_2(J), .., u_n(J))$. A *collective utility function (CUF)* is a function that operates on all the utility profiles and maps $u(J)$ to some real number.

From the perspective of CAN, we consider two main notions of CUF to be satisfied:

The Utilitarian CUF: in this notion, a higher average utility of the agents is preferred, i.e., $\sum_{i \in \mathcal{A}} u_i$ is maximized.

The Egalitarian CUF: in this notion, it is preferred that the worst-off agent is better off. If $\{u_k | k \in \mathcal{A}\}$ is the lowest utility of all the utilities, that particular allocation is always preferred when u_k is the highest.

One of the shortcomings of the utilitarian CUF is that only average utility is considered without any provision for fairness or equality. For example, in this notion, a utility profile of (100,1) is preferred over the utility profile (50,50), although the latter ensures better equality with a small cost for the average utility. On the other hand, the egalitarian CUF is inconsiderate about the overall utility as long as the worst-off agent performs better, i.e., it prefers a utility profile of (25, 25) better than (24, 76) although the latter provides a much higher average utility with a small cost for the worst-off agent. From the perspective of CAN, we want a CUF that satisfies both the notions simultaneously, i.e., higher average utility is achieved while the worst-off CF also performs the best.

5.2 Conflict resolution using NSWF

The Nash CUF is the one that balances both utilitarian and egalitarian notions, i.e., it balances efficiency and fairness ([45, 44]), and it is sensitive to change in overall welfare. The Nash CUF, or the NSWF, is defined as the product of the individual agent utilities for a particular resource allocation J:

$$NSWF(J) = \prod_{i \in \mathcal{A}} u_i(J) = u_1(J) \cdot u_2(J) \cdot u_3(J) ... u_n(J) \tag{5.1}$$

In NSWF, the fairness in allocation can be observed from the lowest difference in the obtained utilities. For example, NSWF prefers the utility profile (50,50) to (99,1) or (24,76) as (50,50) provides the best and equal fairness to all.

However, NSWF can only be applied as long as the following three conditions are satisfied [46]:

- *Pareto optimality*: this condition states that the preference of an agent between two alternatives should depend only on the individual's preference between these two alternatives and should be independent of the individual's preferences over other alternatives.
- *Independence of irrelevant alternatives with neutral property*: this condition states that the naming of the agents should be irrelevant and the principle of equity should be satisfied.

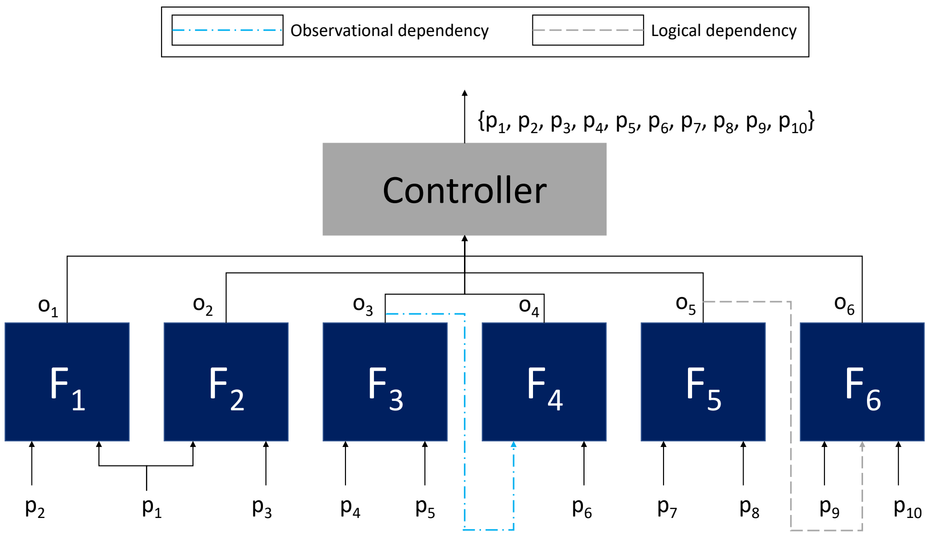


Figure 5.1: CAN model 1

- *anonymity*: this condition states that the ordering of the individual utilities in the CUF should be irrelevant, i.e., for any permutation of the utilities, the outcome should always be the same.

We see that all these three conditions are satisfied from the perspective of CAN:

- since the CFs do not interact with each other, a CF is not influenced by any other CF and its preference is solely based on its learning.
- in our proposed CAN model, all CFs have equal importance.
- ordering or particular permutation of CFs does not have any relevance in our proposed CAN model.

Hence, to calculate an optimal value in CAN, NSWF can be used. In the proposed CAN model, objective of each CF is equally important in the system, that is why the Controller uses NSWF to obtain a solution which provides equal importance to each CF and is optimal for their collective interest. For example, if we consider the simple CAN model of Fig. 4.2 again, there is a possibility of conflict (type A) between F_1 and F_2 over p_1. In such cases, the Controller selects the value of p_1 such that the product of the UFs of F_1 and F_2 is maximum.

5.3 Evaluation

After we establish that NSWF based conflict resolution approach is the most suitable method for CAN, we evaluate the effectiveness of our proposed solution by showing that the Controller can resolve:

- any kind of conflicts which may arise among the CFs
- conflicts among any number of CFs
- any number of simultaneously existing conflicts between the CFs

To that extent, we use three separate CAN models as shown in Fig. 5.1, Fig. 5.2 and Fig. 5.3:

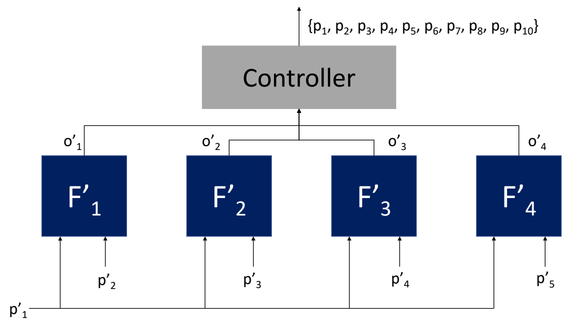


Figure 5.2: CAN model 2

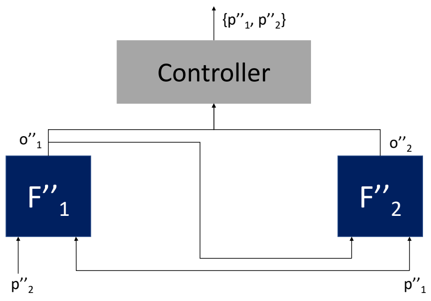


Figure 5.3: CAN model 3

- **CAN Model 1** with six CFs (F_1, F_2, F_3, F_4, F_5, F_6) and a controller (as shown in Fig. 5.1). From this model we see that: both F_1 and F_2 share the same input parameter (p_1), so if they have an input parameter conflict (A). As actions of F_3 affect the measurement of the output of F_4, it is a measurement conflict (B). Also, changing p_7 affects o_5 and o_6 is dependent on o_5, hence it is a logical dependency conflict (C). Thus, this model exhibits all possible types of conflicts.

- **CAN Model 2** with four CFs (F_1', F_2', F_3', F_4') and a controller (Fig. 5.2). We see that four CFs (F_1', F_2', F_3', F_4') have input parameter conflict over input parameter p_1'. It is to be noted that, although we use only 4 CFs in this model, the model can be extended to have any finite number of CFs.

- **CAN Model 3** with two CFs (F_1'', F_2'') and a controller (Fig. 5.3). Here we see that F_1'' and F_2'' have - input parameter conflict over p_1'', measurement conflict (as the action of F_1'' impacts the measurement of o_2''), and, characteristic conflict (logical dependency conflict, $o_2'' \rightarrow o_1'' \rightarrow p_2''$), i.e., all three types of conflicts are existing simultaneously.

If the controller can resolve all the conflicts in CAN Model 1, we can say that it is able to resolve any kind of conflicts that may arise among the CFs. Similarly, if the controller is able to resolve the conflicts in CAN Model 2 and Model 3, we can say that it is able to resolve conflicts among any number of CFs and any number of simultaneously existing conflicts respectively.

During numerical analysis, we assume that - $o_i \forall i \in \{1, 6\}, o_j' \forall j \in \{1, 4\}$ and $o_k'' \forall k \in \{1, 2\}$, are all Gaussian distribution functions. The reason behind this assumption is that in a real-life scenario, mobile network parameters resemble this distribution very often. For example, from [47] we see that both SNR and Latency on a loaded cellular network follow Gaussian distribution. The outputs of the CFs are given by:

$$o_1 = e^{-\frac{(p_1+50)^2}{2p_2^2}} \tag{5.2}$$

$$o_3 = e^{-\frac{(p_4+60)^2}{2p_5^2}} \tag{5.3}$$

$$o_5 = e^{-\frac{(p_7+70)^2}{2p_8^2}} \tag{5.4}$$

$$o'_1 = e^{-\frac{(p'_1+100)^2}{2p'^2_2}} \tag{5.5}$$

$$o'_2 = e^{-\frac{(p'_1+50)^2}{2p'^2_3}} \tag{5.6}$$

$$o''_1 = e^{-\frac{(p''_1+50)^2}{2p''^2_2}} \tag{5.7}$$

$$o_2 = e^{-\frac{(p_1-50)^2}{2p_3^2}} \tag{5.8}$$

$$o_4 = e^{-\frac{(p_6-60)^2}{\frac{2}{o_3^2}}} \tag{5.9}$$

$$o_6 = e^{-\frac{(p_9-o_5)^2}{2p_{10}^2}} \tag{5.10}$$

$$o'_3 = e^{-\frac{(p'_1-50)^2}{2p'^2_4}} \tag{5.11}$$

$$o'_4 = e^{-\frac{(p'_1-100)^2}{2p'^2_5}} \tag{5.12}$$

$$o''_2 = e^{-\frac{(p''_1-50)^2}{\frac{2}{o''^2_1}}} \tag{5.13}$$

These equations have been formulated in such a way that between a pair of CFs, the conflict(s) as mentioned hold true. For example, when p_1 is -50, o_1 is maximum and when p_1 is 50, o_2 is maximum, and thus, F_1 and F_2 have a conflict over p_1. In the second case, o_4 does not have any direct dependency on p_5. However, increasing p_5 increases o_3 which decreases o_4, and thus, F_3 and F_4 have a measurement conflict over p_5. Lastly, o_6 is dependent on o_5 and o_5 is dependent on p_7 (and p_8), thus, F_6 has a logical dependency conflict with F_5 over p_7 (and p_8). Using a similar analysis, input parameter conflict can be observed among (o'_1, o'_2, o'_3, o'_4) and all three types of conflicts can be observed between (o''_1, o''_2). Moreover, for better analysis and graphic visualization, we model all conflicts as parameterized conflicts without any loss of generality.

5.3.1 Analysis Setup

In reality, the underlying relationship between the outputs and network states (or, configurations) is not known beforehand. As mentioned previously, the main functionality of a CF is to learn the variation of its output when the network state or configuration changes, so that it can predict its output and generate the utility function for any particular network state or configuration. To make the numerical analysis as close to reality as possible, we create dummy datasets, using which the CFs can learn the variation of output w.r.t. input parameters. For each CF F_i, we create a dummy dataset D_i following equation o_i. Next, we create a machine learning model (MLM) and train the MLM using the dummy dataset. After the training is complete, the MLM, which now acts as a CF, is able to predict its output corresponding to a particular input configuration.

5.3.1.1 Dummy Dataset Generation

To create a dummy dataset(e.g., D_1 for F_1) we take a random combination of the input parameters (p_1, p_2), calculate the output o_1 (using Eq. 5.2), and store them in a table. We select the value of p_1 and p_2 randomly from their predefined range (which can be found in Table 5.1) and make sure they are unique. The total number of instances of (p_1, p_2), which is used for generating D_1, is denoted by *training-size.*

5.3.1.2 MLM Training

To model the CFs in Python, we use Polynomial Regression block of order 5 using Python package Sklearn. Each F_i is trained using dummy dataset D_i, so that, F_i can predict its output o_i for any combination of its input parameters.

Table 5.1: Parameters and their default values

Parameter	**Value**	**Parameter**	**Value**
p_1	[-150, 150]	p_2	20
p_3	60	p_4	0
p_5	[-50, 100]	p_6	100
p_7	[-150, 80]	p_8	80
p_9	40	p_{10}	60
p'_1	[-200, 200]	p'_2	35
p'_3	60	p'_4	85
p'_5	150	p''_1	[-100, 100]
p''_2	40		

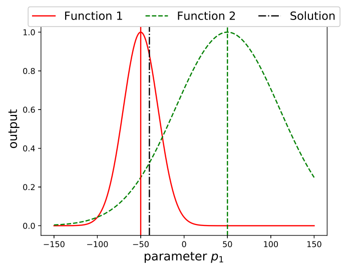


Figure 5.4: Conflict resolution between F$_1$ and F$_2$

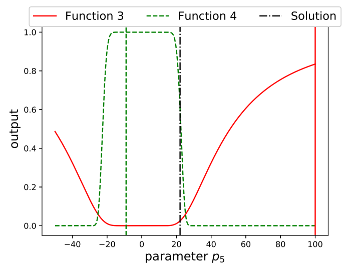


Figure 5.5: Conflict resolution between F$_3$ and F$_4$

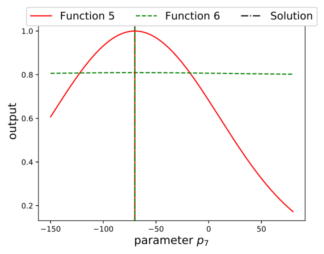


Figure 5.6: Conflict resolution between F$_5$ and F$_6$

5.3.1.3 Parameter Values

Relevant parameters and their values used in the numerical analysis are:

1. *KPI optimality spread (KOS)*: the default value is 25% unless stated otherwise.

2. *controller-sampling-size*: the default value is 2000 unless stated otherwise.

3. *training-size*: Unless stated otherwise, the default value of this variable used in the analysis is 2000.

4. p_i, p'_i, p''_i: The input parameters of the system are chosen in a way such that the outputs are distinct from one another and the impact of the proposed solution is visible. The parameters with their default values or range of values are described in Table 5.1.

5.3.2 Numerical analysis of the controller performance

5.3.2.1 CAN Model 1

Let us start the analysis with the variations of the outputs (o_i) w.r.t. the parameter of conflict. For example, F_1 and F_2 have a conflict over parameter p_1, and in Fig. 5.4 we show the variations of o_1 and o_2 w.r.t. p_1. The black vertical line shows the output values corresponding to the calculated solution and each other vertical line shows the maximum output of the function with the same color. For example, in Fig. 5.4, the red vertical line shows the optimal configuration for F_1, the green vertical line shows the optimal configuration for F_2, and the black vertical line corresponds to the calculated optimal configuration which lies between the other two vertical lines. From Fig. 5.4 we see that when the value of p_1 is as depicted by the black line, neither o_1 nor o_2 has their maximum values, but it is a good balance between these two outputs. In the same way, we plot F_3 and F_4 in Fig. 5.5 and F_5 and F_6 in Fig. 5.6. However, from Fig. 5.6 we see that, although F_5 and F_6 have a logical dependency conflict over p_7, o_6 remains almost constant w.r.t.

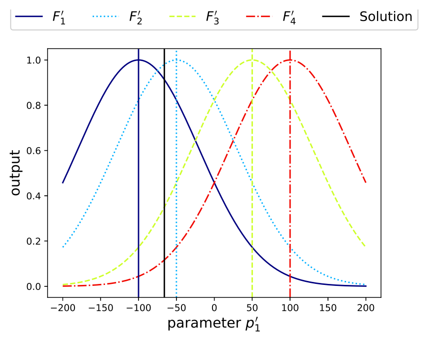


Figure 5.7: Numerical analysis of CAN Model 2

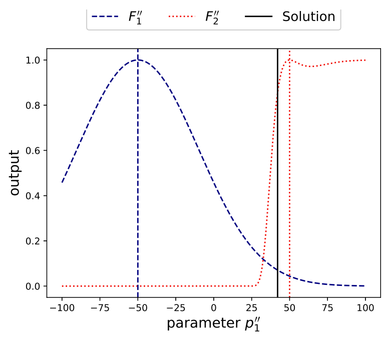


Figure 5.8: Numerical analysis of CAN Model 3

p_7, i.e., the maximum of o_6 is independent of p_7. In such cases, the ideal configuration is the point where the other function has its maximum output. Now from Fig. 5.6 we can see that the proposed configuration in this case coincides with the point where o_5 has its maximum, proving that proposed NSWF based method always provides the best configuration.

5.3.2.2 CAN Model 2

Using this model we show that the proposed controller can resolve conflict among any number of CFs. We consider a CAN model with 4 CFs - F'_1, F'_2, F'_3, F'_4, all of which share the same input parameter p_1 and have a conflict over it. To resolve this conflict, we use the same controller and plot the result in Fig. 5.7 to show the optimal configuration for the combined interest of all four CFs. Although in this model we consider only 4 CFs, the solution idea can be extended and used for any finite number of CFs.

5.3.2.3 CAN Model 3

To show that the proposed controller can resolve any number of simultaneously existing conflicts among CFs, we consider CAN model 3 with 2 CFs - F''_1, F''_2 as shown in Fig. 5.3. All three types of conflicts exist simultaneously between F''_1 and F''_2 - they have input parameter conflict (over p''_1), measurement conflict (as the action of F''_1 influences measurement of o''_2) and logical dependency conflict ($o''_2 \rightarrow o''_1 \rightarrow p''_2$). To resolve all these conflicts simultaneously, we use the same controller and plot the result in Fig. 5.8 to show the optimal configuration. It proves that the proposed controller is able to resolve any number of conflicts that may exist simultaneously.

5.4 Conclusion and Key Takeaways

In this chapter, we provided a controller that coordinates among CFs in CAN and dynamically resolves: any type of conflict among CFs, any number of simultaneously existing conflicts among CFs, and conflict among any number of CFs. To prove the validity of the proposed controller, we implemented three separate scenarios of conflict in CAN and performed a numerical analysis of the controller's performance in each scenario. The results show that the controller works as expected in each case. Our proposed design is generic (i.e., for different scenarios, a single mechanism is used), it has the ability to resolve the conflict reactively (i.e., whenever the conflict arises), and it is scalable (i.e., it works for any number of CFs). Most importantly, the proposed mechanism resolves conflicts using NSWF, so that the calculated value is optimal for the collective interest of the whole system.

List of Abbreviations Used in This Chapter

CUF	collective utility function
MARA	multi agent resource allocation
MLM	machine learning model
NSWF	Nash social welfare function

6. Interest based optimal configuration calculation

In this chapter, we highlight the shortcomings of the NSWF based method and propose a new method that overcomes those shortcomings. We compare the new method with the NSWF based method and highlight the advantages of the new method. We evaluate our proposed solution both numerically, and in the simulation environment used in this thesis. The majority of the content in this chapter is from the following peer-reviewed journal paper and patent application by the author:

[48] **A. Banerjee**, S. S. Mwanje, and G. Carle. Towards Control and Coordination in Cognitive Autonomous Networks. In *IEEE Transactions on Network and Service Management (TNSM)*, 19(1): 49-60, 2022.

METHODS AND APPARATUSES FOR DETERMINING OPTIMAL CONFIGURATION IN COGNITIVE AUTONOMOUS NETWORKS
Anubhab Banerjee, Stephen S Mwanje
US, Provisional application no.: 63/056084, filed 24 July 2020

The discussion in this chapter expands on the content by providing a more in-depth discussion on Fisher Market Model, Eisenberg-Gale optimization and Shapley value based configuration weight calculation method.

6.1 Drawbacks of NSWF solution

In the last chapter, we discussed the process of conflict resolution in CAN and also discussed how the Controller uses NSWF to find an optimal value for the combined interest of the CFs. While using NSWF, we assumed that while sharing a resource (in the perspective of CAN, while setting the value of a control parameter), all CFs have equal priority in that parameter. However, in a real-life scenario, a control parameter may have different levels of influence on different CFs, which leads to the conclusion that different CFs should not necessarily have equal priority over a control parameter. The CF, which is influenced by the parameter more, should have more priority in that parameter and vice versa. Let us explain this with an example.

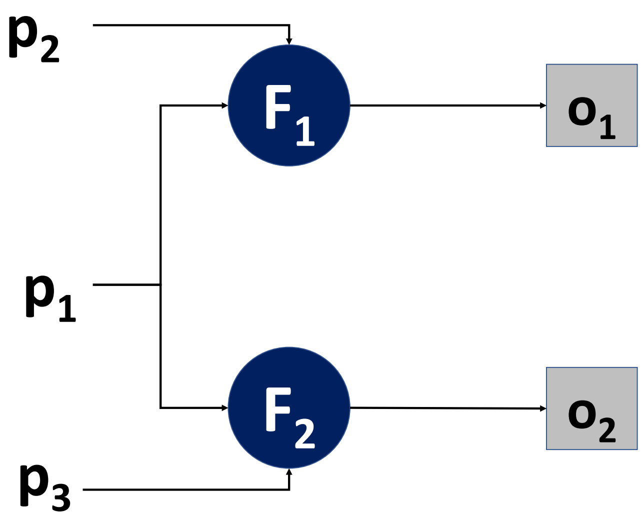


Figure 6.1: Example CAN model

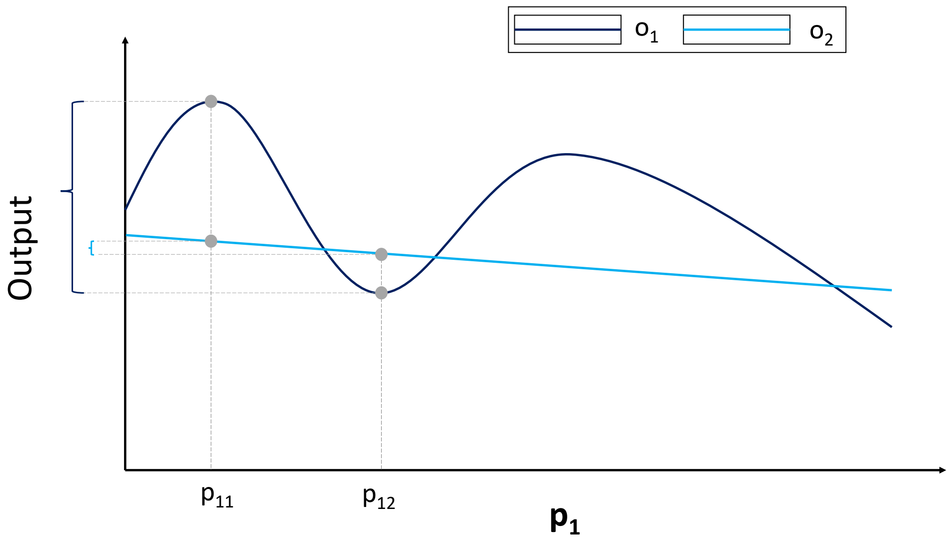


Figure 6.2: KPI variations over shared parameter

Let us consider a CAN (shown in Fig. 6.1) with two KPIs: o_1 and o_2 and both of them have a common ICP: p_1. Two CFs: F_1 and F_2 are tasked to optimize o_1 and o_2 respectively. Let us assume that in a certain network state, we plot the variations of o_1, o_2 while only p_1 is varied and p_2, p_3 are kept constant, and, we obtain the plot as shown in Fig. 6.2. From this image we see that when p_1 is varied by a certain amount, o_1 varies more than o_2 varies. For example, when the value of p_1 is changed from p_{11} to p_{12}, the value of o_2 decreases slightly whereas the value of o_1 decreases from maximum to minimum. So, if the value of p_1 is set at p_{12} instead of p_{11}, it is of little importance for F_2 but of significant importance for F_1. Now it is evident that F_1 should be given more priority than F_2 while determining the value of p_1, else the final value of p_1 might be set at a sub-optimal range.

To express the priority mathematically, while determining p_1, if the Controller gives importance $w_{F_1}^{p_1}$ to F_1 and importance $w_{F_2}^{p_1}$ to F_2, then $w_{F_1}^{p_1}$ should be greater than $w_{F_2}^{p_1}$, i.e., $w_{F_1}^{p_1} > w_{F_2}^{p_1}$. We denote these importance values ($w_{F_1}^{p_1}$ and $w_{F_2}^{p_1}$) as *configuration weight (CW)* values. Later in Section 6.4 we discuss how the CW values can be calculated.

6.2 Fisher Market Model of CAN

Although we deduce that F_1 should be given more priority than F_2 while determining p_1, quantifying and setting the priority values for these two CFs is a challenging task. To overcome this challenge and quantify the importance of a CF towards a control parameter, we model the CAN as a Fisher Market Model (FMM) [49]. The reason we model CAN as FMM in optimal configuration calculation is because FMM provides optimal value for the combined interest of all the CFs while taking individual importance into account.

An FMM $\mathcal{M}$ consists of a set of buyers $\mathcal{D} = \{d_1, d_2, \ldots\ldots, d_d\}$ and a set of items $\mathcal{C} = \{c_1, c_2, \ldots\ldots, c_c\}$, where every buyer d_i has:

- An initial budget (b_i) which can be visualized as money that can only be utilized to purchase the items and has no intrinsic value to the buyer.
- A UF u_i:$[0,1]^c \rightarrow \mathbb{R}$, that maps a quantity vectors of the c items to some real predefined scale.

$u_i(x_i)$ represents the buyer's utility when receiving x_i amount of items. An agent's utility in this game is his utility of the allocated items with respect to his true preference. The set of budgets is denoted by $\mathcal{B} = \{b_1, b_2, ..., b_d\}$. Without any loss of generality, the supply of each good is assumed to be one unit, and the total budget of all buyers is normalized to one, i.e., $\sum_{i=1}^{d} b_i = 1$ [50]. These assumptions are often made for convenience in analyzing the model without affecting the results. When the FMM is used for allocating resources among self-interested agents, it induces an FMM game. In an FMM game, each agent reports its true preferences on the items of $\mathcal{C}$ to some central entity, which, in turn, determines a market equilibrium according to the budgets of the agents. Based on the budget b_i of a buyer d_i, the target of the central entity is to determine the set of items (c_i) the buyer should possess for an optimal allocation of items [50].

Let us use the FMM game to formulate the process of calculating optimal configurations in CAN. We use a single FMM game to determine the optimal value of a single configuration. The Controller can be visualized as the central entity and CFs can be visualized as the buyers. Considering example of Fig. 6.1, while determining optimal p_1, $\mathcal{D} = \{F_1, F_2\}$ and $\mathcal{C} = \{p_1\}$. Now, we visualize the $w_{F_i}^{p_1}$ values as their respective budgets, e.g., $w_{F_1}^{p_1} = w_1$, $w_{F_2}^{p_1} = w_2$ (calculation of w_i values can be found in Section 6.4) and respective UFs are $f_1(p_1)$ and $f_2(p_1)$. It is important to note that w_i values are non-negative, non-zero and $0 < w_i \leq 1$. These $f_i(p_1), i \in \{1, 2\}$ are linear functions which map their objective values to the predefined scale ([0:10]). Now, the game model is complete as: $\mathcal{D} = \{F_1, F_2\}$, $\mathcal{C} = \{p_1\}$ and $\mathcal{B} = \{w_1, w_2\}$ and the target is to find optimal p_1.

6.3 Eisenberg-Gale solution to FMM

To obtain the optimal value of a control parameter in an FMM game (p_1 in our example), it is important to find the equilibrium solution in that game which can be captured by Eisenberg-Gale Solution (EGS) [51]. EGS can be applied if the UFs of the buyers belong to the same class in the constant elasticity of substitution (CES) family [50]. UFs in the CES family take the form of

$$u_i(x_i) \ = \ \Big(\sum_{j=1}^{c} a_{ij} x_{ij}^{\rho}\Big)^{\frac{1}{\rho}} \tag{6.1}$$

where $u_i(x_i)$ represents the buyer's utility when receiving x_i amount of items, a_{ij} is a parameter which quantifies how receiving more item j affects buyer i's utility and ρ parameterizes the family, $-\infty < \rho \leq 1$, $\rho \neq 0$. The Leontief, Cobb-Douglas, and linear UFs are obtained when ρ approaches $-\infty$, 0, and equals 1 respectively [50]:

$$Leontief : u_i(x_i) = \min_{j \in [c]} \left\{ \frac{x_{ij}}{a_{ij}} \right\} \tag{6.2}$$

$$Cobb - Douglas : u_i(x_i) = \prod_{j \in [c]} x_{ij}^{a_{ij}} \tag{6.3}$$

$$Linear : u_i(x_i) = \sum_{j \in [c]} a_{ij} \cdot x_{ij} \tag{6.4}$$

For these three types of UFs, the EGS takes the following form [50]:

$$\begin{aligned} &\max \prod_{i=1}^{d} u_i^{b_i} \\ \text{s.t.} \quad & u_i = \big(\sum_{j=1}^{c} a_{ij} x_{ij}^{\rho}\big)^{\frac{1}{\rho}}, \forall i \in [d] \\ & \sum_{i=1}^{d} x_{ij} \leq 1, \forall j \in [c] \\ & x_{ij} \geq 0, \forall i \in [d], j \in [c] \end{aligned} \tag{6.5}$$

For some values of ρ, e.g., $\rho = 1$, the objective function of EGS is not strictly concave, which means that there can be multiple market equilibria [50].

While using Eq. 6.5 to calculate optimal p_1,we find that $u_i(x_i)$ becomes $f_i(p_1)$ and b_i becomes similar to w_i but not equivalent, because, b_i values are normalized ($\sum_{i=1}^{d} b_i = 1$) but w_i values are not normalized. So, if w_1 and w_2 are normalized to $w_1^{'}$ and $w_2^{'}$ such that $w_1^{'} + w_2^{'} = 1$, then $w_i^{'}$ becomes equivalent to b_i. When the network state is fixed, $f(p_x) = \frac{c_x}{10}$, $f(p_y) = \frac{c_y}{10}$, and so on, thus, the UF of the CF is equivalent to Eq. 6.4 with $i = 1$ (as we calculate only one configuration at a time), $\rho = 1$ and $a_{ij} = \frac{1}{10}$. $\rho = 1$ also signifies the possibility of the existence of multiple optimal values for p_1.

Considering Fig. 6.1 again, let us assume that when the Controller calculates p_1, the values of OCRS, UF and CW, for F$_1$ are - $\{[p_{1min}^{F_1}, p_{1max}^{F_1}], w_{p_1}^{F_1}, f_1(p_1)\}$ and for F$_2$ are - $\{[p_{1min}^{F_2}, p_{1max}^{F_2}], w_{p_1}^{F_2}, f_2(p_1)\}$. The Controller combines two OCRSs into a final optimal config set (FOCS) taking the minimum of ($p_{1min}^{F_1}$, $p_{1min}^{F_2}$) and the maximum of ($p_{1max}^{F_1}$, $p_{1max}^{F_2}$) and selects the p_1^* from the FOCS, for which $f_1(p_1^*)^{w_1^{'}} \cdot f_2(p_1^*)^{w_2^{'}}$ is maximum.

As we already mentioned, the linear nature of the UFs by the CFs gives rise to the possibility of having multiple optimal values. In case there are multiple values of p_1 for which $f_1(p_1)^{w_1^{'}} \cdot f_2(p_1)^{w_2^{'}}$ is maximum, the one for which the weighted utility values are closest to one another is selected. Standard deviation is used to measure the relative closeness of the weighted utility values. The lower the standard deviation value, the closer the values are the values to one another. For example, let us assume p_{11} and p_{12} are two values of p_1, which belong to the FOCS and $f_1(p_{11}) = 10$, $f_2(p_{11}) = 6$, $f_1(p_{12}) = 6$, $f_2(p_{12})$ = 5.83, $w_1^{'} = 0.3$, $w_2^{'} = 0.7$. For both p_{11} and p_{12}, value of $f_1^{w_1^{'}} \cdot f_2^{w_2^{'}}$ is 6.992. As the final value is equal for both p_{11} and p_{12}, we measure the standard deviation of $[f_1^{w_1^{'}}, f_2^{w_2^{'}}]$ for both cases. For p_{11}, the standard deviation value is 0.755 and for p_{12}, the standard deviation value is 1.186 and so, p_{11} is selected as the optimal value of p_1.

6.4 Calculation of CW values

CW values give an estimation and quantification of the importance of each control parameter on each KPI in a cellular network. In a general multiple input single output (MISO) system the impact of each input variable on the output variable is called Sensitivity Index (SI). Although several SIs have been proposed in the literature, the most generic and popular one is called Sobol indices [52]. In [53] the authors showed that when the input variables are independent, many Sobol indices become equal to zero, and a different index is required for such cases. As control parameters in CAN are independent and they do not affect each other, in this paper we formulate the w_i values as Shapley values [11]. Shapley value provides the marginal contribution of each player in a multi-player single-objective cooperative game. In a single-objective cooperative game, a coalition of players cooperates, and obtains a certain overall gain from that cooperation. Since some players may contribute more to the coalition than others, the Shapley value calculates the importance of each player to the overall cooperation. An example use case of the Shapley value is game theoretic centrality [26], which figures out the most important node in a graph towards a single objective completion.

Let us consider a cooperative game with players $N = \{1, 2, .., n\}$. In a cooperative game, the players try to come to an agreement, and they also have a choice to bargain with each other, so that individually they can gain maximum benefit. This benefit is higher than what they could have obtained by playing the game without cooperation. In the game, any number of players can form a coalition S. A payoff function $pf(S)$ is defined to calculate a utility value from each coalition. Then, for each player i, $i \in \{1, 2, .., n\}$, the Shapley value is calculated as

$$\zeta_i(pf) = \sum_{S \subseteq N, i \notin S} \frac{|S|!(|N| - 1 - |S|)!}{|N|!} \cdot [pf(S \cup \{i\}) - pf(S)] \tag{6.6}$$

In a CF, usually there are multiple ICPs and each of them influences the output by a different amount. We formulate the CW value of an ICP same as the marginal contribution of that ICP in achieving the objective of the CF. To formulate the game among the ICPs, we need to define the payoff function for each coalition of the ICPs. The payoff function is defined per coalition basis and it is defined as the normalized difference between the maximum and the minimum obtainable output when the parameters of the coalition are varied. For example, suppose a CF has inputs: $\{r_1, r_2, r_3, r_4\}$ and output: o. o_{max} is the maximum possible output of the CF and corresponding inputs are: $\{r_1^{max}, r_2^{max}, r_3^{max}, r_4^{max}\}$. Similarly, o_{min} is the minimum output of the CF and corresponding inputs are: $\{r_1^{min}, r_2^{min}, r_3^{min}, r_4^{min}\}$. An example of a coalition is R which is formed between r_1 and r_2, i.e., $R = \{r_1, r_2\}$. Then, r_1 and r_2 are varied simultaneously, and, r_3 and r_4 are kept fixed at r_3^{max} and r_4^{max} respectively. If the minimum obtainable output, by varying only r_1 and r_2, is o_R, then,

$$pf(R) = \frac{o_{max} - o_R}{o_{max} - o_{min}} \tag{6.7}$$

The motivation behind defining the payoff is: to ensure the optimal performance of the CF, we measure how much the objective of the CF degrades when one ICP is varied and the ICP, for which the degradation is maximum, has the highest CW value. In Appendix A we validate the effectiveness of Shapley value based CW calculation both mathematically and by simulation.

6.5 Numerical analysis of proposed solution

In this section, we show the significant advantages of using our proposed solution over the NSWF solution using analytical models. Let us again consider the CAN model shown in

Fig. 6.1 and assume that the CFs can generate the UFs using their learning. As both F_1 and F_2 share the same ICP p_1, the calculation of p_1 involves getting OCRS, UF and CW from both CFs. It is important to note that the values of all the parameters have been assumed in accordance with those assumed in Chapter 5.

We assume the UFs generated by F_1 and F_2 to be modeled as Gaussian distributions:

$$f_1(p_1, p_2) = 0.5e^{\frac{-(p_1+50)^2}{2p_2^2}} \tag{6.8}$$

$$f_2(p_1, p_3) = e^{\frac{-(p_1-50)^2}{2p_3^2}} \tag{6.9}$$

Furthermore, while discussing the calculation of optimal p_1, we keep p_2 and p_3 constant at 60 and 20 respectively throughout our analysis. These values have been chosen in a manner to highlight the contrast between the UFs. Both F_1 and F_2 have been trained on p_1: [0, 100] and their UFs are formulated on a [0:1] scale. The reason behind assuming UFs as Gaussian Distributions is: in real life, distribution of KPIs resembles Gaussian Distribution to a great extent [47]. We assume KOS (β) as 50%, so that from Eq. 6.8, F_1 calculates OCRS as [0, 36] and from Eq. 6.9, F_2 calculates OCRS as [27, 73].

As the next step, F_1 has to calculate the CW values of p_1 and p_2 and F_2 has to calculate the CW values of p_1 and p_3. In a two-person game with g_1 and g_2 being the two players, the Shapley value, from Eq. 6.6, becomes

$$\zeta_{g_1/g_2}(pf) = \frac{1}{2}[1 + pf(g_1/g_2) - pf(g_2/g_1)] \tag{6.10}$$

From Eq. 6.8, we see that output of F_1 (o_1) is maximum when $p_1 = 0$ and minimum when $p_1 = 100$, i.e., $p_1^{max} = 0$, $p_1^{min} = 100$. Maximum value of o_1 is 0.441 and minimum value is 0, i.e., $o_1^{max} = 0.441$, $o_1^{min} = 0$. p_1 and p_2 can form four possible coalitions = $\{\{\emptyset\}, \{p_1\}, \{p_2\}, \{p_1, p_2\}\}$. Payoff for each coalition, calculated using Eq. 6.7, are -

$pf(\emptyset)$	0	$pf(p_1)$	0.633
$pf(p_2)$	0.955	$pf(p_1, p_2)$	1

Thus, when we calculate the Shapley values of p_1 and p_2 using Eq. 6.10, we get them as $\zeta_{p_1} = 0.339$, $\zeta_{p_2} = 0.661$. As $\zeta_{p_1} + \zeta_{p_2} = 1$, there is no need for further normalization.

Similarly, while calculating the CW values for F_2, from Eq. 6.9, we get $p_1^{max} = 50$, $p_1^{min} = 0$ and $o_2^{max} = 1$, $o_2^{min} = 0$. p_1 and p_3 can form four possible coalitions = $\{\{\emptyset\}, \{p_1\}, \{p_3\}, \{p_1, p_3\}\}$. Payoff for each coalition, calculated using Eq. 6.7, is -

$pf(\emptyset)$	0	$pf(p_1)$	1
$pf(p_3)$	0	$pf(p_1, p_3)$	1

Thus, when we calculate the Shapley values of p_1 and p_3 using Eq. 6.10, we get them as $\zeta_{p_1} = 1$, $\zeta_{p_3} = 0$. As $\zeta_{p_1} + \zeta_{p_3} = 1$, there is no need for further normalization.

6.5.1 Optimal p_1 Calculation

After a CF generates OCRS, UF, and CW, and sends them to the Controller, the Controller calculates the optimal configuration using EGS. The value of w_1 (CW of p_1 for F_1) is 0.339 and the value of w_2 (CW of p_1 for F_2) is 1. Normalizing the weights we get, $w_1^{'} = 0.25$ and $w_2^{'} = 0.75$. When the Controller combines the OCRSs from F_1 and F_2 into the FOCS, it becomes [min(0, 27), max(36,73)] or, [0, 73]. As both F_1 and F_2 have been trained when p_1 is varied in steps of 1, the Controller also samples values in [0, 73] in steps of 1 and selects the value for which $f_1^{w_1^{'}} \cdot f_2^{w_2^{'}}$ is maximum.

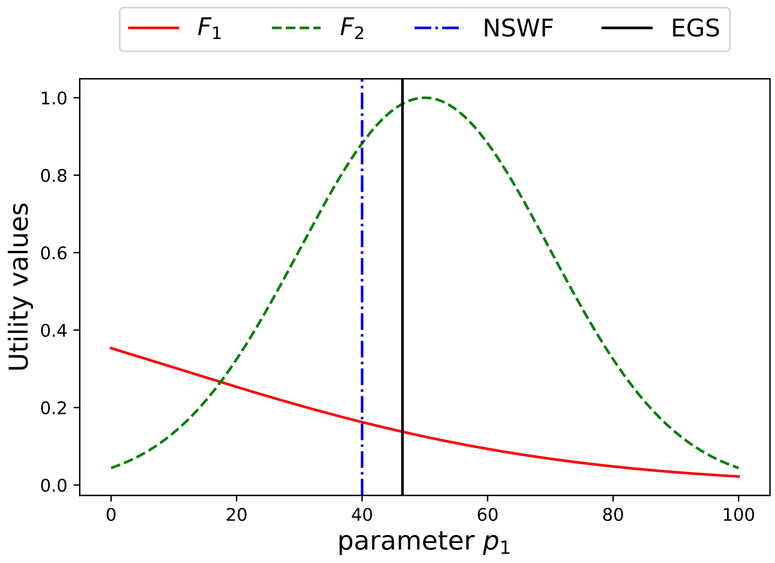


Figure 6.3: Comparison between NSWF and EGS

6.5.2 Comparison between NSWF and EGS

In Fig. 6.3 we plot the UFs of F_1 and F_2 for p_1. The red and green plot shows the UF of F_1 and F_2 respectively. According to NSWF, the optimal p_1 for the system is 40 and according to EGS, the optimal p_1 is 46. However, we see that, when we use EGS, the utility of F_1 decreases by 2.4% but the utility of F_2 increases by 10.1%, so the overall system performance improves by 7.7%.

6.6 Evaluation in the simulation environment

We evaluate our proposed solution in the same simulation environment discussed in Section 3.2. We use four CFs in our simulation: MLB, MRO, CCO and ES. As already shown in [1], TXP has major impact on MLB and MRO, and it has minor impact on CCO and ES. So, we use the calculation of TXP to demonstrate the effectiveness of our proposed solution about finding the best balance among multiple CFs.

6.6.1 UF interpretation

In Fig. 6.4 we plot how the utilities of the four CFs vary when TXP is varied. From Fig. 6.4 we see that when the TXP is increased, the utility value of MLB gradually decreases, and then after a certain value of TXP, it increases again. This happens because when TXP increases, the size of the cell increases, and load on the cell also increases, so the utility of MLB decreases. After a certain TXP, the capacity of the cell starts increasing, which, in turn, reduces the load and increases the utility value of MLB.

From Fig. 6.4 we see that initially, the utility of CCO decreases when TXP is increased because with increasing TXP, cell size and load on the cell increase which in turn reduces the capacity. After TXP = 30 dBm, the utility value increases almost linearly with TXP as the load remains almost constant whereas the capacity increases, which, in turn, increases the utility value. Beyond 70 dBm, interference from neighboring cells becomes so high that further increasing TXP reduces the capacity and utility.

From Fig. 6.4 we see the variation of utility values of ES when TXP varies. ES has the highest utility (1) when TXP is lowest (20 dBm), and the lowest utility (0) when TXP is highest (80 dBm).

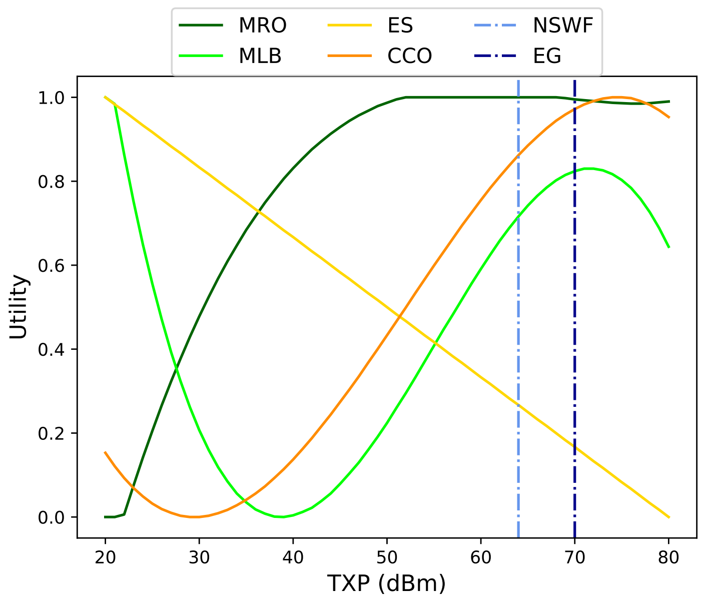


Figure 6.4: UFs of all CFs

From Fig. 6.4 we see that initially, the utility value of MRO increases when the TXP increases because with increasing TXP, the overlap between neighboring cells becomes more prominent and the percentage of failed handovers decreases. After 50 dBm it becomes almost constant.

From these plots we make the following observation about optimal TXP for different CFs: optimal TXP for MLB is either 20 dBm or 70 dBm, for CCO it is between 60 - 70 dBm, for ES it is near 20 dBm and for MRO it is between 50 - 80 dBm. So, apart from ES, for the rest 3 CFs optimal TXP is 70 dBm or some value close to 70 dBm. In the next section, we show that when optimal TXP is calculated using NSWF and EGS, in both cases the value comes near or at 70 dBm. This validates the methods proposed by us for optimal configuration calculation is suitable to be used in a real-life scenario.

6.6.2 Optimal control parameter calculation

We calculate optimal TXP using both NSWF and EGS. According to NSWF, optimal TXP is 64 dBm and according to EGS, optimal TXP is 70 dBm (Fig. 6.4). For each CF, we calculate how much improvement in utility scale we get using EGS instead of NSWF:

Table 6.1: Improvement in utility

CF	Improvement (%)	CF	Improvement (%)	CF	Improvement (%)	CF	Improvement (%)
MLB	6.35	ES	-8.76	CCO	12.99	MRO	-1.4

So, from Fig. 6.4 we see that, although the EGS solution degrades the performance of ES and MRO, the overall system performance improves by 9.18% using EGS. As stated earlier, we give equal importance to all CFs in CAN and our target is to find optimal TXP for the combined interest of all the CFs, EGS serves our purpose.

6.6.3 Time Complexity

Our proposed solutions are beneficial to use in real life if another configuration recalculation request does not arrive before the previous configuration recalculation is complete. We run the simulations to determine the time consumed while calculating optimal values of each of the control parameters (TXP, TTT, CIO, RET) and the results are shown in the following tables -

Table 6.2: Time consumed in NSWF

Param	Time (ms)	Param	Time (ms)	Param	Time (ms)	Param	Time (ms)
TXP	0.21	TTT	0.19	CIO	0.19	RET	0.2

Table 6.3: Time consumed in EGS

Param	Time (ms)	Param	Time (ms)	Param	Time (ms)	Param	Time (ms)
TXP	0.35	TTT	0.22	CIO	0.22	RET	0.32

6.7 Conclusion and Key Takeaways

Table 6.4: Comparison between Controllers

Comparison aspect	NSWF based Controller [14]	EGS based Controller [22]
Easiness to implement	X	
Performance		X
Time complexity	X	
Usefulness in real life	X	X

As we saw, the EGS based solution includes the impact of a network parameter of a CF (denoted as CW) and, thereby, overcomes the drawbacks of the NSWF based solution proposed in the last chapter. In this thesis, we used a Shapley value based method to calculate the CW values. When we performed a numerical analysis between these two methods, we also found an improvement of 7.7% in the EGS based method. However, NSWF based solution works much faster than the EGS based solution since, in the EGS based solution, CW values also need to be calculated. An overview of comparisons on different aspects between these two solutions is given in Table 6.4. In conclusion, the time consumed by any of the two methods is so tiny that we can safely assume that no change in network state can happen within this short time, and both of these solutions are feasible to be used in real life.

List of Abbreviations Used in This Chapter

CCO	coverage and capacity optimization
CES	constant elasticity of substitution
CW	configuration weight
EGS	Eisenberg-Gale Solution
ES	energy savings
FOCS	final optimal config set
FMM	Fisher Market Model
MISO	multiple input single output
MLB	mobility load balancing
MRO	mobility robustness optimization
SI	Sensitivity Index
UF	utility function

Part III

Trust

7. Trust: Problem and Related Works

In this section, we discuss how open and multi-vendor-enabled CAN deployment may raise a serious issue concerning trust in the system. When different CFs come from different vendors, all of them may not be equally trustworthy, We propose the idea of an MCF that can degrade the overall network performance to achieve its own objective by manipulating the Controller. In the simulation environment, we experimentally validate the existence of MCF and formulate two essential problems which need to be answered. The content in this chapter is from the following paper by the author:

[54] **A. Banerjee**, S. S. Mwanje, G. Carle. On Detection of Manipulative Cognitive Functions in Cognitive Autonomous Networks. In *2021 17th International Conference on Network and Service Management (CNSM)*.pages 194-200. IEEE, 2021.

7.1 Problem Statement

The vision of open, multi-vendor network automation is to allow anyone (e.g., startups, universities, or any vendor) to develop and deploy CFs in the same network management automation (NMA) architecture, enabling a more competitive and vibrant supplier ecosystem. Similarly, open-source software and hardware reference designs enable faster, more democratic, and permission-less innovation [8]. In multi-vendor environments, CFs may be supplied by different vendors, creating new opportunities for participation to small as well as new vendors. However, to establish itself in competition with others, a new vendor needs to advertise the superiority of its product and such a vendor may be tempted to do so by unfair means. There remains a possibility that a vendor designs a CF that optimizes its own objective, disregarding the interests of all other CFs, to advertise the superiority of its product. Such a CF, denoted as an MCF, deliberately sends misinformation on its preference, e.g., an inaccurate UF (discussed in Section 2.4), intending to manipulate the Controller [54].

Now, when we propose this concept of MCF, two questions arise:

(i) Can MCF exist, i.e., is it possible for a CF to manipulate the Controller?

(ii) Is it beneficial for the MCF to manipulate the Controller, i.e., can the MCF better its achievements by manipulating the Controller?

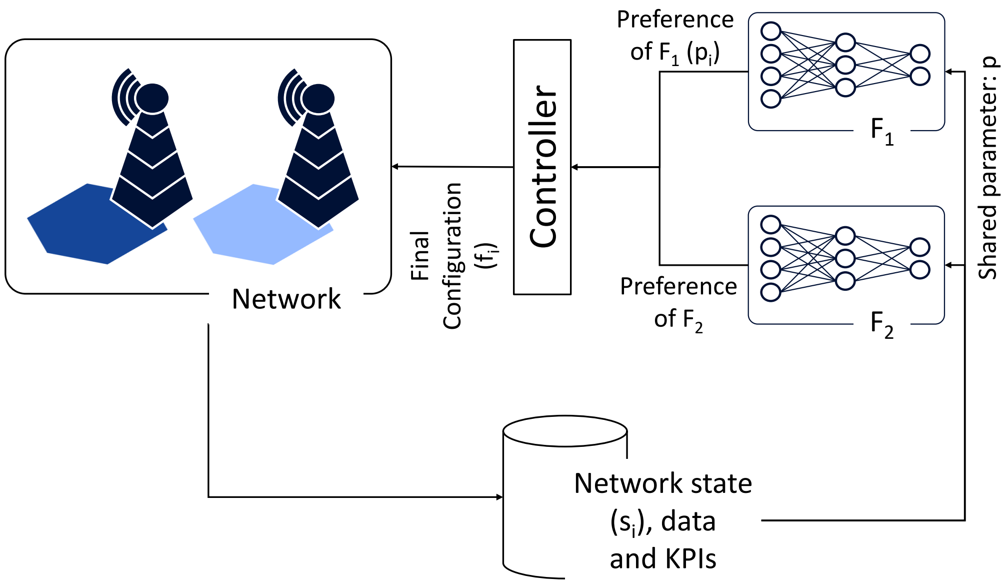

Preference of F_1 (p_i)
F_1
Preference of F_2
F_2
Shared parameter: p
Controller
Final Configuration (f_i)
Network
Network state (s_i), data and KPIs

Figure 7.1: CAN MCF Example

Later in Section 7.1.1 we discuss how an MCF can manipulate the Controller and in Section 7.1.2 we experimentally demonstrate how the MCF can profit from manipulating the Controller.

Since we established that

- an MCF can manipulate the Controller and the MCF has enough leverage to so, and,
- a vendor has enough motivation to produce and sell such MCFs,

in this context, two important questions arise which need to be answered to maintain operational efficiency of CAN:

1. Can such an MCF be detected?
2. More importantly, what is the best way to handle such an MCF?

We address the 1st question in Chapter 8 and the 2nd question in Chapter 9. It is to be noted that while addressing both these questions, we assume that there is only one MCF present. If there are multiple MCFs present, and all of them send manipulated preferences to the Controller, behavior of the Controller becomes erratic and no clear pattern is visible in that behavior. Lack of a learnable pattern in the behavior of the Controller is against the notion of the existence of an MCF. This is validated experimentally as well. That is why we always assume there is only one MCF present.

7.1.1 How manipulation occurs

In this section we experimentally demonstrate that such type of manipulation of Controller is possible. Consider the CAN in Fig. 7.1 with two CFs: F_1, F_2 and a Controller, with p as a shared parameter between F_1 and F_2, and, F_1 as an MCF. In Section 2.4 we mentioned that a CF sends two pieces of information: OCRS and UF, to the Controller to express

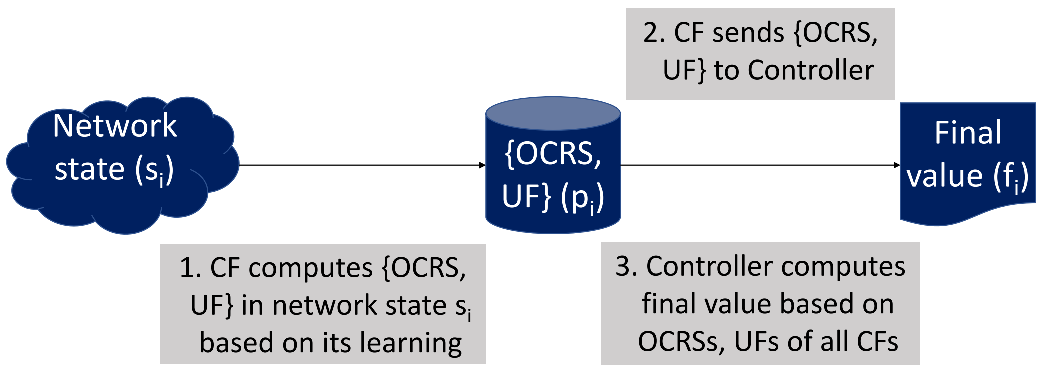

2. CF sends {OCRS, UF} to Controller
Network state (s_i)
{OCRS, UF} (p_i)
Final value (f_i)
1. CF computes {OCRS, UF} in network state s_i based on its learning
3. Controller computes final value based on OCRSs, UFs of all CFs

Figure 7.2: Learning objective of an MCF

its preference. It is important to note that since the UF is more effective than OCRS in influencing the decision making of the Controller [54], hereafter, a CF's preference only refers to the UF. In a network state s_1, F_1 observes that if F_1 proposes p_1 to the Controller (as a preferred value for p), the Controller calculates the final value as f_1. Again, F_1 observes that in another network state s_2, if it sends p_2 to the Controller, the Controller sets f_2 as the final value. So, eventually after n such instances, F_1 observes that for its preferred values $p_1, p_2...p_n$ of the parameter p in network states s_1, s_2 ..s_n, the Controller computes values f_1,f_2...f_n respectively. Given observations over multiple instances, F_1 may become able to learn the relationship among the variables $< s_i,\ p_i,\ f_i >$ as depicted in Fig. 7.2, i.e., in a network state s_i, F_1 can predict what the value of f_i will be if F_1 proposes p_i to the Controller.

Now, in a network state s_{n+1}, F_1 computes its preferred value as p_{n+1}, but F_1 also predicts that if it sends p_{n+1} to the Controller, the final value, set by the Controller, will be f_{n+1}. To manipulate the Controller's outcome, F_1 may propose a false value p'_{n+1}, to ensure that the new final value (f'_{n+1}) will be equal to or as close as possible to p_{n+1}. Thereby, F_1 is able to manipulate the Controller to optimize F_1's own objective, but at the cost of overall network performance and stability.

Algorithm 7.1 shows an example for computing p'_{n+1}. Consider that the Controller allows (from any CF) only the δ-spaced discrete values in $[p_{min},\ p_{max}]$, where $p_{max} = p_{min} + m * \delta$, $m \in \mathbb{Z}^+$. Also consider that F_1 has learned the function $g(\cdot)$, so that $f_{n+1} = g(s_{n+1}, p_{n+1})$, i.e., given the current network state and F_1's preference, F_1 can predict the final value to be set by the Controller. F_1 wishes to compute p'_{n+1} such that f'_{n+1} $(=g(s_{n+1}, p'_{n+1}))$ and p_{n+1} are as close as possible, i.e., $|p_{n+1}$ - $f'_{n+1}|$ is minimum. For that (see Algorithm 7.1), F_1 could iterate over all possible values of p (since they are finite in number) and find the one for which $|p_{n+1}$ - $f'_{n+1}|$ is minimum.

7.1.2 Demonstration of MCF manipulation

To experimentally demonstrate manipulation and its impact on performance, we use the simulation set-up described in Section 3.2 with the two CFs: MLB and MRO. There is also one Controller to coordinate between these two CFs which is flexible to use both NSWF (Chapter 5) and Eisenberg-Gale solution (EG) (Chapter 6) based solutions. We wish to show the extent to which an MCF (MLB in this case) can learn the relationship among $< s_i,\ p_i,\ f_i >$.

Fig. 7.3 shows the prediction success of an MCF's prediction on f_i values relative to the Prediction margin. The Prediction margin is the maximum allowed deviation of the

Algorithm 7.1: MCF manipulation

```
Input: $p_{n+1}$, $s_{n+1}$, $g(\cdot)$, $[p_{min}, \delta, p_{max}]$
Output: $p'_{n+1}$
$f_{n+1} = g(s_{n+1}, p_{n+1})$
 if $|p_{n+1}$ - $f_{n+1}| == 0$ then
    $p'_{n+1} = p_{n+1}$
else
    $d = |p_{n+1}$ - $g(s_{n+1}, p_{min})|$
    $r = p_{min} + \delta$
    while $r <= p_{max}$ do
        if $|p_{n+1} - g(s_{n+1}, r)| < d$ then
            $p'_{n+1} = r$
            $d = |p_{n+1} - g(s_{n+1}, r)|$
        $r = r + \delta$
```

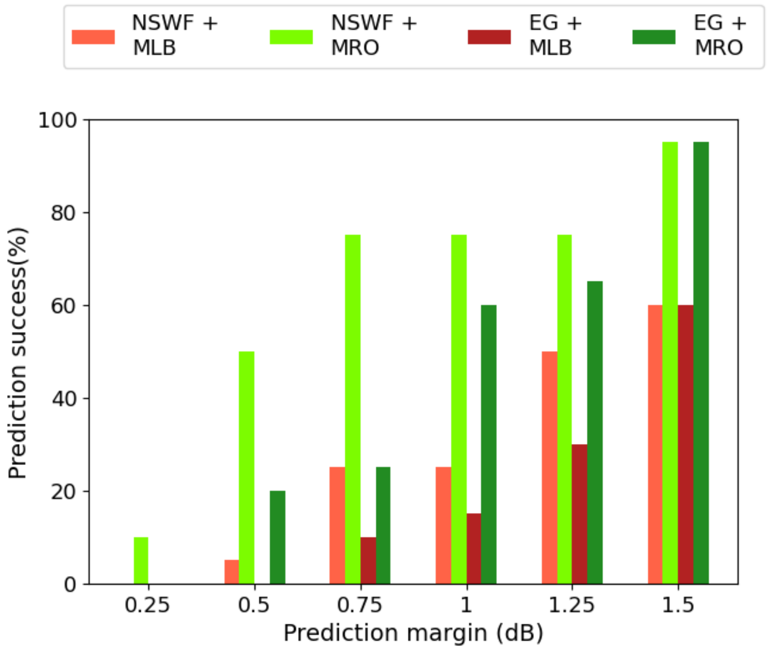


Figure 7.3: MCF prediction accuracy

prediction from the actual value, i.e., if the value of f_i is x dB, and prediction margin is y dB, the prediction is considered a success if the MCF's prediction falls within $[x \pm y]$ dB. Naturally, a higher value of y increases the prediction success range ($[x \pm y]$), and in turn increases the accuracy of the MCF. Fig. 7.3 also shows that an MCF is least successful in predicting f_i values when: (i) the Controller uses EG instead of NSWF, which can be attributed to the use of CW values in computing f_i in EG, and, (ii) the MCF is MLB instead of MRO. So, it is reasonable to assume that in all other cases (MRO + NSWF, MRO + EG, MLB + NSWF), MCF's prediction success, and the degree of manipulation of the Controller, is at least the same or more. So, in the next paragraph, we discuss the degree of manipulation of the Controller and overall performance degradation when the MCF is MLB and the Controller uses EG.

Next, we investigate the impact on the system-wide performance degradation which can be caused by the MCF. System-wide performance degradation is computed as the sum of the degradations of the individual CFs' performance. We consider the scenario which is least favorable for the MCF, i.e. MLB + EG, since the degree of manipulation and consequent

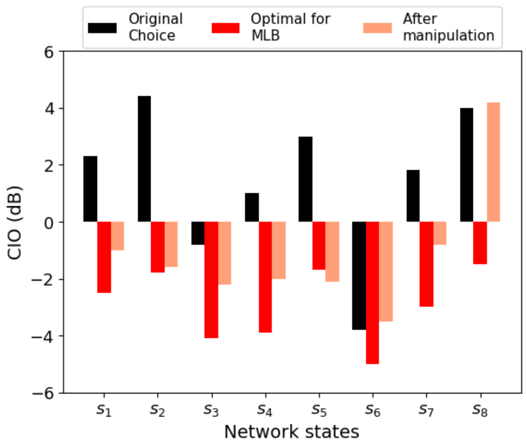


Figure 7.4: self-interested CIO proposal by MLB

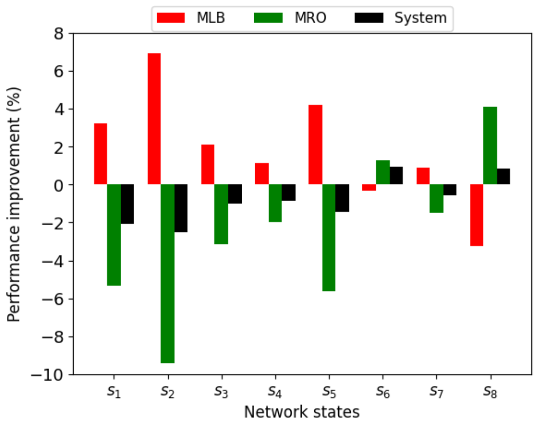


Figure 7.5: Observed overall performance degradation

system performance degradation will be the same or more in all other cases. Thereby, for 8 random network states, Fig. 7.4 plots three bars side by side for: (i) the original CIO value computed by the Controller using EG and without manipulation (f_{n+1}), (ii) the optimal CIO value for MLB in that network state (p_{n+1}), (iii) the CIO value computed by the Controller when manipulated by MLB (f'_{n+1}). We see that in 75% of the cases (6 out of 8), MLB is successful in manipulating the Controller into setting a CIO value that is more favorable for MLB's interest than the overall system. From Fig. 7.5 we see that in these cases, overall performance degrades by up to 10%. Since we show these results considering a scenario that is least favorable for an MCF, we can expect that for any other CF and Controller combination, the manipulation and system performance degradation will be equivalent or even more, and this proves the necessity of a mechanism to tackle the MCF(s) that may be present in a system unknown to the MNO.

7.2 Related Works

We divide this section into two parts: in the 1st part, we give a brief overview of existing research works that are relevant to trust in CAN, and in the 2nd part, we discuss standards related to our work and the impact of our work on ongoing 3GPP standardization efforts.

7.2.1 Related Literature

The concept of a rogue agent or manipulative agent exists in a number of research domains, including social networks [55, 56], distributed systems [57, 58], multi-agent systems [59, 60, 61], robotics [62, 63], and many others. However, there are significant differences between a rogue agent in any of those fields and an MCF in terms of definition and functionality. The concept of an MCF is most similar to the concept of signal jamming in duopoly (SJD) [64, 65], so we briefly review SJD and the prior research works conducted on SJD to understand its similarity to our problem and highlight the novelty of our research compared to the state-of-the-art.

SJD refers to the scenario where one firm intentionally sends manipulative information into a marketplace to force other firms to adjust their market prices of a given product. In an open market, several firms can manufacture the same product and compete in selling the product to the customers. Although a firm A cannot know the internal business strategies of a rival firm B, firm A can always observe the product's market price set by the B. Based on the observations, firm A can roughly estimate the values of different unknown parameters of each rival firm's demand and production curve. Based on that, a firm can potentially signal jam, i.e., strategically vary its production level and market price in order to manipulate the distribution of estimated market prices [64]. It is to be noted that the market prices of the rival firms are subject to random disturbances and may provide noisy information to the firm, which may potentially harm the process of estimation of the firm.

This scenario is important to CAN, since the CFs act like the firms and the characteristics of a shared control parameter match with the common market product. Just like the firms, a CF cannot observe the actions of other CFs, but it can observe the final value of the shared control parameter set by the Controller. Based on it, an MCF can potentially deduce other CFs' preferences and the Controller's final value calculation method and send misleading information to the Controller to manipulate the final value of the shared parameter. The final value of a shared parameter is calculated taking the preferences of all CFs into account, and their preferences may change from time to time, which may provide noisy information to the MCF. So, we see that all the characteristics of an SJD match perfectly with the context provided in this paper, presenting SJD as the most relevant state-of-the-art for this paper.

Over the years different types of problems on SJD, along with different types of solutions, have been discussed in multiple publications. Although SJD has been proposed in earlier research works like [66, 67, 68], these papers assumed information transmission among the firms. This is in contrast to our assumption that no direct information transmission or communication takes place among the CFs. For example, in [69, 70, 71] it was assumed that private information about a firm can be perfectly transmitted through its perfectly observed actions, which is not applicable in CAN since a CF cannot directly observe the action of another CF. Relatedly, the signal jamming models in [72, 73, 74, 75, 76] assume that the signal jammer knows the information that is the subject of signal jamming, which also does not hold true in CAN. The signal jamming models proposed in [77, 64] resemble CAN the most, but even here, the firms try to observe rivals and then develop a strategy to reach an equilibrium. This does not directly apply to our considered scenario, where the focus is on developing a different solution that forces the MCF to stop its manipulative behavior completely and restores the network to its optimal operating point. So, we see that, although CAN resembles SJD in multiple aspects, currently there does not exist any SJD scenario or solution which is applicable to our work.

Currently, with the advancement of AI and ML, there is an ongoing trend of using more AI and ML-based solutions for network management and automation purposes. In [78] the authors introduced the concept and discussed the limitations of AI-enabled zero-touch network and service management (ZSM). In [79] the authors laid out their vision for AI driven closed-loop network automation in 5G and beyond mobile networks. On a similar note, in [80] the authors used NN for small cell coverage optimization, and researchers designed an NN-based adaptive routing scheme for QoS optimization in [81]. Although there are a lot more research works on using AI/ML in the networks, however, none of the existing works discusses trust and performance issues in multi-vendor, open AI-enabled network management automation.

7.2.2 Related Standardizations and relevance to our work

The most real-life application of the Controller-CF based CAN framework is the 3GPP SON [82] and 5G SON [83]. In SON, the network automation functions (NAFs) both in the control and in the management layers monitor the network events, analyze the network data and make decisions on the SON actions based on their analysis. A SON management function provides policies and targets to the NAFs and resolves any types of issues among the NAFs. The NAFs are similar to CFs, but CF decisions are learned and not pre-programmed while the role of SON management is similar to the role of the Controller in CAN. These functionalities remain true for 5G SON as well [83], but 3GPP has started introducing learning-based functionality including the network Data Analytics Function [84] and Management data Analytics [85].

The advancements in ML and deep learning (DL) in recent times are also driving vendors to produce ML based solutions for SON, i.e.slowly pushing towards the CAN framework. For example, if we consider the specifications of MRO in 3GPP [83], we see that the functionalities, like, analyzing reports from UEs and network slice information, and mitigating HO issues by adjusting HO related parameters (TTT, CIO), can be better achieved by using ML and DL capabilities. Relatedly, the specifications present network automation as a multi-vendor environment where 5G SON functions in the management layer can be sourced separately from the base stations or their Distributed SON functions. This is done with the assumption that the SON management layer in the base station will coordinate the actions of these different multi-vendor automation functions. This deployment of multiple learning automation functions (i.e. CFs) in a multi-vendor environment directly runs into the trust and performance issues discussed in this paper. In other words, as a minimum the SON management layer shall need to be adapted to account for the likelihood of manipulation as we have discussed here. For optimal results, however, the entire management automation architecture may need to be overhauled fairly soon before the ML/DL based network automation solutions become prevalent.

7.3 Conclusion and Key Takeaways

Although openness and multi-vendor deployment of mobile network management provide lots of benefits (e.g., cost-efficiency, agility), it can also introduce a significant drawback regarding trust in the management system. We conceptualized the idea of an MCF: a CF that learns how the Controller behaves and manipulates the Controller to achieve its objective sacrificing overall network performance. We experimentally demonstrated that such a type of MCF can exist and can cause severe network performance degradation. After establishing the seriousness of the issue, we framed two critical questions on this topic, which are addressed in the following two chapters. We concluded this chapter by summarizing prior research works and standardization activities on this topic.

List of Abbreviations Used in This Chapter

3GPP	3rd generation partnership project
CIO	cell individual offset
DL	deep learning
EG	Eisenberg-Gale solution
MCF	manipulative cognitive function
ML	machine learning
MLB	mobility load balancing
MNO	mobile network operator
MRO	mobility robustness optimization
NAFs	network automation functions
NMA	network management automation
NSWF	Nash social welfare function
OCRS	optimal configuration range set
SJD	signal jamming in duopoly
SON	self-organizing network
TTT	time to trigger
UF	utility function
ZSM	zero-touch network and service management

8. Manipulative Cognitive Function Detection

In this chapter, we address the Q3 described in Section 1.3.3 by proposing a solution to detect an MCF in CAN. Our solution is based on polynomial regression, which is compatible and easy to implement in CAN. We evaluate our proposed solution extensively and observe key findings regarding the proposed solution. The majority of the content in this chapter is from the following paper and patent application by the author:

[54] **A. Banerjee**, S. S. Mwanje, G. Carle. On Detection of Manipulative Cognitive Functions in Cognitive Autonomous Networks. In *2021 17th International Conference on Network and Service Management (CNSM).*pages 194-200. IEEE, 2021.

DETECTING MANIPULATIVE NETWORK FUNCTIONS
Anubhab Banerjee, Stephen S Mwanje, Abdelrahman Abdelkader
WO, PCT application no.: PCT/IB2021/054489, filed 24 May 2021

The discussion in this chapter takes elements from these, but expands on the description of the solution and evaluation results.

8.1 Brief Overview on Polynomial Regression

In this section we give a brief overview on polynomial regression, based on which our solution MCD is designed. Polynomial Regression is a regression algorithm that models the relationship between a dependent(y) and independent variable(x) as a n-th degree polynomial. The relationship between x and y, i.e., the n-th degree polynomial, is given by:

$$y = b_0 + b_1x + b_2x^2 + ... + b_nx^n$$

Polynomial regression is beneficial to use in cases where the relationship between two variables (x and y in this case) is non linear and difficult to depict. The value of n is not constant and it is adjusted to a suitable value to model the relationship better. The goodness of the model, i.e., how accurately the polynomial represents the true relationship between x and y, is usually measured by root mean squared error (RMSE) or R^2 score.

It is important to keep in mind that in polynomial regression, there is always a bias vs variance trade-off. Bias refers to the error due to the model's simplistic assumptions in

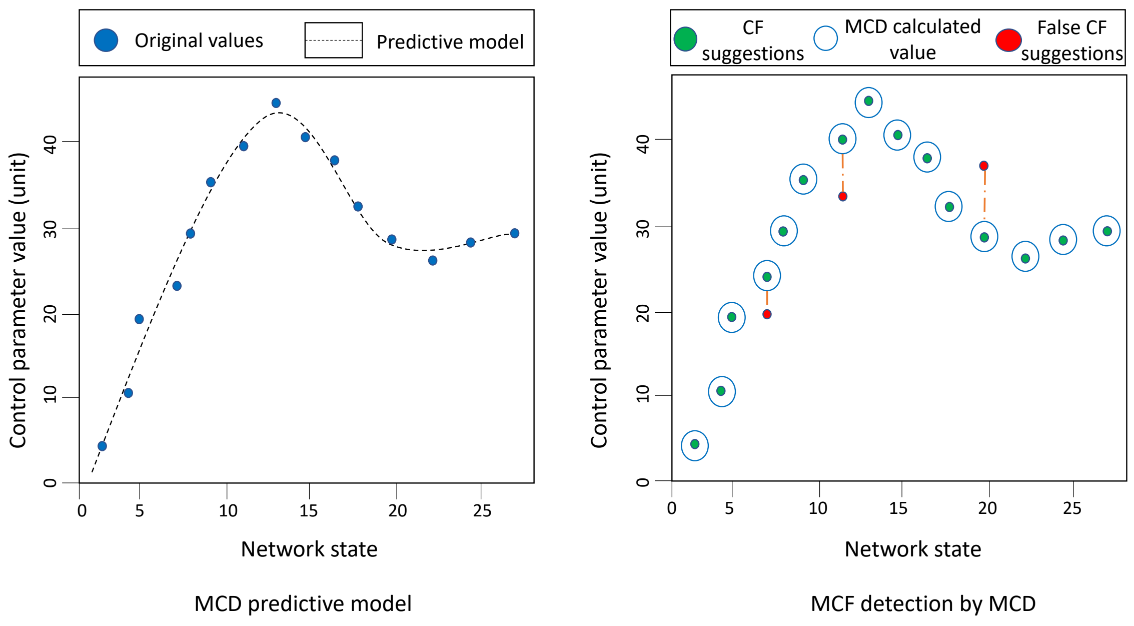


Figure 8.1: MCD and MCF Detection by MCD

fitting the data. A high bias means that the model is unable to capture the patterns in the data and this results in under-fitting. On the other hand, variance refers to the error due to the complex model trying to fit the data. High variance means the model passes through most of the data points and it results in over-fitting the data. So, while using polynomial regression, one has to always find the appropriate balance between bias and trade-off.

8.2 Manipulative CF Detector

In this Section we propose a method, called MCD, to detect an MCF in CAN. The proposed solution is effective and easy to implement, and can be incorporated as an added feature of the Controller. After MCD becomes operational, it keeps tracking $<s_i, p_i>$ values of each CF and learns how p_i varies when s_i varies for each CF. This learning is a continuous process which lasts as long as MCD is active. As soon as MCD receives a p_i value from a CF which is not in par with MCD's learning, MCD raises an alarm. After a certain number of alarms, value of which can be set by network operator or the Controller, MCD marks the CF as an MCF.

Let us consider the CAN model of Fig. 7.1 again, where F_1 is an MCF, to explain how MCD works. After MCD becomes operational, every time F_1 sends Request to the Controller, MCD records the s_i and p_i values and applies polynomial regression to make a predictive model to learn the variation of p_i with s_i (Fig. 8.1). For depiction purposes, we express both s_i and p_i by numerical values with units and they can be adjusted based on how s_i and p_i are defined in simulation environments. Every time the network state changes and the Controller receives a request from F_1 for configuration recalculation, MCD predicts $p_i^{'}$: the value F_1 should send in the current network state based on previous experience. If a p_i is not close to $p_i^{'}$, MCD marks it in red (shown in Fig. 8.1) and raises the alarm.

There are a few points to be noted while using polynomial regression in MCD:

- We use degree of polynomial as 3 when the MCF is MLB or MRO. This value has been chosen such that RMSE of the regression model is always at less than 1 for a

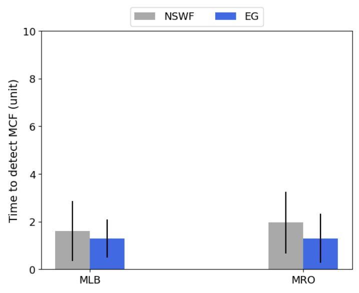


Figure 8.2: MCD starts before MCF

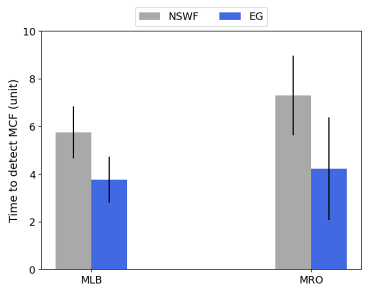


Figure 8.3: MCD and MCF start simultaneously

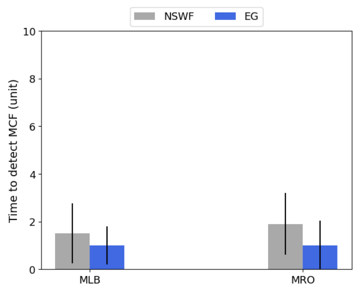


Figure 8.4: MCF starts before MCD

good fit. This value is determined based on trial-and-error. If any other CF is used, the degree of the polynomial may need to be adjusted.

- In our implementation, we allow an error margin ($|p_i - p_i^{'}|$) of 5% on predicted values by MCD. This value can be adjusted based on requirement. If the value is too high, learning of MCD converges faster, but also the learning is more erroneous; if the value is too low, the learning is more accurate, but it takes more time for the learning to converge. This trade-off should be kept in mind while choosing the error margin.

- If enough number of data points are available, use of deep learning or other sophisticated supervised learning algorithms in MCD and further lowering of error margin on MCD predictions may result in a much better performance by MCD. However, since our target was to demonstrate that such an idea would work well, we implemented the regression based solution.

8.3 Evaluation

8.3.1 Set-up

In this Section we evaluate the performance of our proposed solution (MCD) in the same simulation environment, in which the problem statement was described (Section 3.2), with 2 CFs: MLB and MRO. We make one CF to behave as MCF, regarding the CIO values, at one time and measure how quickly MCD can detect the manipulative behavior. We change the network state every certain time interval and perform measurements in terms of number of time intervals, so that is why we express the time in ”unit”. The time unit can be anything: minute, hour or day, as long as network state changes every minute, hour or day respectively. The time unit can be non uniform as well, i.e., interval between two consecutive network states may not always be constant.

8.3.2 Observations

We consider three cases:

- *case 1*: MCD becomes operational t time units before MCF, results of which are shown in Fig. 8.2.

- *case 2*: MCD and MCF start operating simultaneously, results of which are shown in Fig. 8.3.

- *case 3*: MCF becomes operational t time units earlier than MCD, results of which are shown in Fig. 8.4.

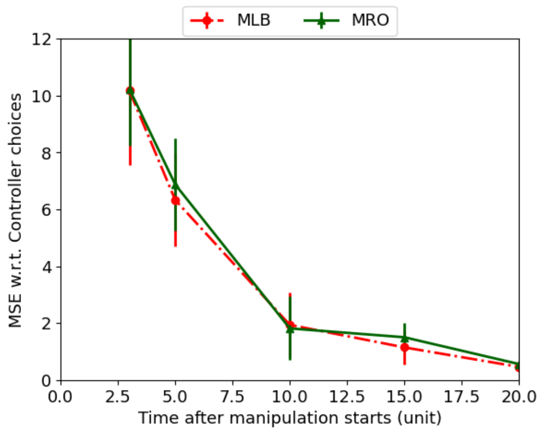


Figure 8.5: MCF learning EGS based Controller behavior

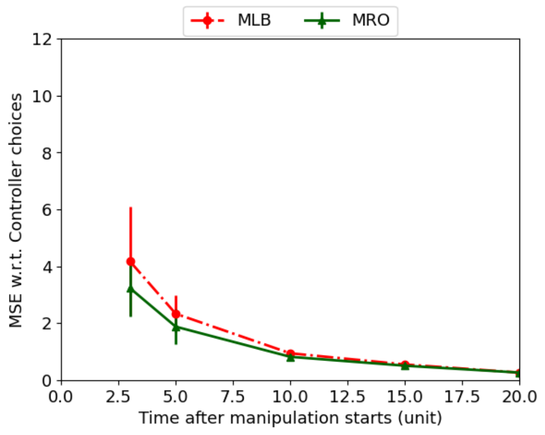


Figure 8.6: MCF learning NSWF based Controller behavior

Both in case 1 and case 3, we consider $t = 5$. Later in this section we discuss the impact of t on MCD performance.

Out of these three cases, time taken to catch an MCF is highest in case 2, because, MCF starts exhibiting manipulative behavior at the same time MCD becomes functional, which makes it difficult for MCD to become certain of the manipulative behavior of MCF. In case 1 MCD starts earlier than MCF, so MCD has enough correct observations from MCF to accurately model the behavior of MCF, and so, it is easier for MCD to detect a manipulative behavioral sign from MCF. In case 3, when MCD starts functioning, MCF already started manipulative behavior, which makes it easier for MCD to see the erratic suggestions proposed by the MCF and, therefore, detect the MCF.

There are two more important things which can be observed from these plots:

- out of those two CFs: MLB and MRO, CIO has more importance for MLB, so that MLB output variation with respect to CIO can be learned very well by MCD, and hence MLB is easier to be detected for manipulative behavior.

- when the Controller uses EGS, as CW values (Section 6.1) are involved in the final value calculation, it is equally hard for MLB and MRO to learn Controller behavior; that is why MCD takes almost the same time for both the CFs.

It is quite intuitive that the more earlier MCD becomes operational than MCF, the better MCD can learn the MCF behavior and the faster MCD can catch MCF. However, in real life, network operator does not have any control over MCF or the start time of its manipulative behavior; so case 2 is the most probable scenario in real life when both MCD and MCF start as soon as the whole system becomes operational. In Fig. 8.5 and Fig. 8.6 we plot how well an MCF can learn Controller behavior when the MCF starts manipulating the Controller t_m time units after the system becomes operational. We see that when t_m is low, mean squared error (MSE) of the MCF predictions about Controller choices is very high, because MCF does not have enough instances to learn the Controller behavior. The higher the value of t_m is, the more instances MCF gets to learn the Controller behavior, the lesser becomes the MSE. As per expectation, MSE is higher when the Controller uses EGS instead of NSWF. From $t_m > 15$, the MSE becomes almost close to zero, which means that the MCF has successfully learned the Controller behavior. So to conclude, if MCD starts 15 time units or later than MCF, it becomes very difficult for MCD to catch the MCF.

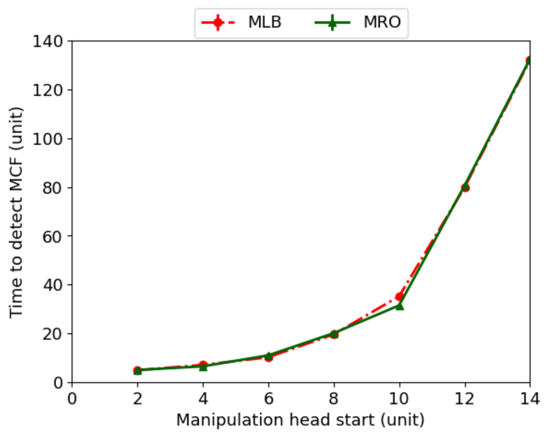


Figure 8.7: Controller using NSWF

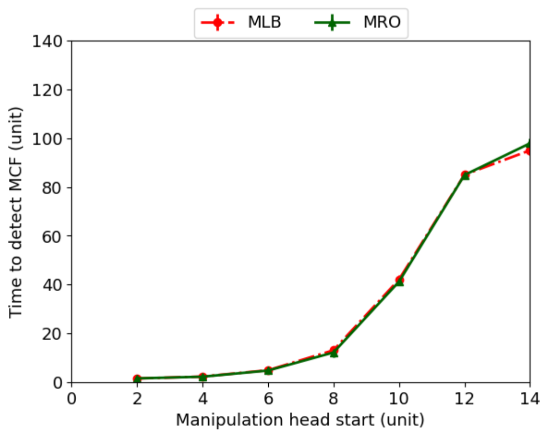


Figure 8.8: Controller using EGS

This conclusion is supported by Fig. 8.7 and Fig. 8.8. We make MCF operational t_c time units before MCD starts, and plot how much time is taken by MCD to catch MCF. When t_c is low, MCF has not learned Controller behavior properly, so MCD catches MCF really fast. However, as t_c goes higher, MCF has more instances to learn Controller behavior, catching MCF becomes more difficult. At $t_c > 15$, it is almost impossible to catch MCF.

8.4 Alternative Approaches

In this thesis, we used polynomial regression based detection mechanism since it is easy to understand and implement. Also, during our simulation, the dataset size was not huge, so regression based detector worked well. However, in real life, when the size of the dataset is huge, a feed-forward NN can be used. The internal features of the NN (e.g., number of hidden layers, number of nodes in each layer) are to be determined experimentally. In Appendix B we discuss one such use case where the optimal NN is determined experimentally. The same approach can be taken here as well.

8.5 Conclusion and Key Takeaways

In this chapter, we proposed and evaluated a solution called MCD to detect an MCF in CAN. From the evaluation results, we made two critical conclusions as follows. First, if there is a significant time difference between the start time of the MCF and MCD, MCD might not be able to catch the MCF at all. So, in a multi-vendor system where the CFs come from different vendors and not all of them are trustworthy, it is always better to use MCD from the start. Second, if the Controller uses EGS instead of NSWF, CW values are also present, which are difficult for an MCF to learn. So, if the Controller uses EGS, an MCF can always be caught much faster.

List of Abbreviations Used in This Chapter

CCO	coverage and capacity optimization
CW	configuration weight
MCD	manipulative CF detector
MCF	manipulative CF
MLB	mobility load balancing
MRO	mobility robustness optimization
MSE	mean squared error
NN	neural network
RMSE	root mean squared error

9. CoDeRa: Controller Decision Randomizer

We proposed a method to detect a manipulative CF (MCF) in the previous chapter. However, sometimes in an operational system, the detection and removal of an MCF are not feasible. In those cases, it is necessary to force the MCF to give up its manipulative behavior. In this chapter, we propose a solution that forces the MCF to abandon its manipulative behavior instead of taking the MCF out of an operational system and address the Q4 described in Section 1.3.3. The majority of the content of this chapter is from the following peer-reviewed journal article and patent application by the author:

[86] **A. Banerjee**, S. S. Mwanje and G. Carle, ”Trust and Performance in Future AI-enabled, Open, Multi-Vendor Network Management Automation,” in IEEE Transactions on Network and Service Management (TNSM), 2022, doi: 10.1109/TNSM.2022.3214296.

REDUCING SYSTEM DEGRADATION CAUSED BY MANIPULATIVE FUNCTIONS
Anubhab Banerjee, Stephen S Mwanje
WO, PCT application no.: PCT/IB2021/054486, filed 24 May 2021

In addition to the content described in the patent application and the publication, this Chapter provides additional information on: (i) deflection computation (Section 9.2) and (ii) discussion on CoDeRa (Section 9.6).

9.1 Basic Concept

Although MCD can detect an MCF, the MCF cannot be just removed or replaced in an operational system, since:

- we cannot shut down an MCF abruptly because then its managed KPI will be heavily affected.
- replacement may require shutting down and we cannot shut down the whole system just to replace an MCF.
- most importantly in case of replacement, there is no guarantee that the new CF won’t be manipulative.

Algorithm 9.1: Deflection calculation

```
Input: $U_{F_1}(\cdot)$, $U_{F_2}(\cdot)$, $f_{n+1}$, $\delta$, $\gamma$, $d_{max}$
Output: $d$
$r = \delta$
while $r < d_{max}$ do
    $u_1 = U_{F_1}(f_{n+1})$, $u_2 = U_{F_2}(f_{n+1})$
    $u'_1 = U_{F_1}(f_{n+1} + d)$, $u'_2 = U_{F_2}(f_{n+1} + d)$
    $x = \frac{u_1 - u'_1}{u_1} + \frac{u_2 - u'_2}{u_2}$
    if $100 \cdot x <= \gamma$ then
        $d = r$
    $r = r + \delta$
```

Since there is currently no available framework to detect for possible manipulative behavior prior to deployment, there is always a chance of one or more MCFs being present in the system. Since an MCF cannot be just shut down or taken out, the only feasible approach is to force the MCF to give up its manipulative behavior. We can validly assume that the MCF will abandon its manipulative behavior when the MCF fails optimize its objective by manipulation. An MCF fails to optimize its objective by manipulation when it cannot learn completely how the system (in this case, the Controller) works. In this case, an MCF will stop manipulative behavior when the MCF is unable to learn the relationship among $< s_i, p_i, f_i >$, since sending unsure preferences to the Controller can cause more harm than benefits to the MCF.

Since out of the three variables $\{s_i, p_i, f_i\}$ (7.1.1), the Controller (or, the MNO) only has control over f_i, we propose to stop MCF from learning the relationship among $< s_i, p_i, f_i >$, by adding unpredictability to the computed f_i values. The unpredictability (a form of deflection or noise) makes the relationship among $< s_i, p_i, f_i >$ difficult (especially when the deflection is high, almost impossible) to learn. We call our proposed solution Controller Decision Randomizer, or in short, CoDeRa [86].

9.2 Deflection Computation

Although adding deliberate noise or deflection is the most reasonable approach, it is also important to minimize the performance degradation caused by the deflection. Using the same example in Section 7.1 with F_1 as the MCF, assume that $U_{F_1}(\cdot)$ and $U_{F_2}(\cdot)$ are the UFs of F_1 and F_2 respectively, while f_{n+1} is the final value computed by the Controller at network state s_{n+1}. CoDeRa seeks to find the maximum possible deflection (d) that maintains the sum of individual performance degradation below the allowed system performance degradation (γ). To compute the deflection d, CoDeRa computes u_1 and u_2, the respective utility values of F_1 and F_2 when $p = f_{n+1}$. Then, CoDeRa computes the sum of the degradation in utility values (or, performance) of the CFs when the final value of p is set at $(f_{n+1} + r)$ instead of f_{n+1} where r is the added deflection. If the total degradation is less than (μ), the deflection is increased and the process repeated until a maximum allowed range (d_{max}).

9.3 CoDeRa Parameters

There are four important parameters in CoDeRa: α, β, γ and *deflection*.

9.3.1 α

It denotes the time gap between the start of the system and CoDeRa. If the system starts at time t_0 and CoDeRa starts at time t_1, then $\alpha = t_1$ - t_0. $\alpha = 0$ means the CoDeRa starts with the system.

9.3.2 β

The longer an MCF keeps learning while CoDeRa is active, due to the added noise, the MCF is likely to experience deterioration in its learning. So, it is beneficial for the MCF to stop learning and use whatever it has learned after a certain point. β denotes the number of time units after which the MCF stops learning but we also consider the case where the MCF never stops learning (i.e., $\beta=\infty$). Note that the Controller does not influence β, it is entirely up to the MCF and the vendor supplying the MCF.

9.3.3 γ

It denotes the allowed total system performance degradation, to be derived as the sum of the percent performance degradation of the individual CFs.

9.3.4 $deflection$

This is the shift in the value computed by the controller. Since a higher deflection leads to more potential network degradation, d has to be limited by the allowed degree of network degradation. The deflection may be added either as a fixed amount d or as a random value of up to d, i.e.,

$$f_i = \begin{cases} f_i \pm d & ; \quad deflection = simple \\ f_i \pm rand(0,d] & ; \quad deflection = random \end{cases}$$

9.4 CoDeRa with MCD

If we keep CoDeRa operational the whole time, two major problems arise:

- Running CoDeRa is not economical and consumes more energy, so it creates wastage of energy when there is no MCF in the system.
- If there is no MCF in the system, using CoDeRa always causes some amount of overall performance degradation, so the network operates sub-optimally without any reason.

So, it is beneficial to use CoDeRa in the system along with an MCD. If MCD detects an MCF in the system, it can alert CoDeRa about the MCF. CoDeRa then starts adding noise to f_i only when the request comes from the MCF, and in all other cases, CoDeRa does not add any noise. Similarly, when the MCF stops exhibiting manipulative behaviors, MCD can notify the CoDeRa and it can go to the sleep mode, thus conserving energy. Just like MCD, CoDeRa can also be implemented either within the Controller, or, as an added functionality.

9.5 Evaluating CoDeRa

To evaluate CoDeRa, we use the experimental setup described in Section 7.1.1 and assume that at any time, one CFs acts as an MCF. We study the impact of the MCF on the network performance over multiple iterations. We assume that each iteration takes place in every time unit selecting a new randomly-selected network state. This is the same unit introduced in Section 8.3.

We vary the parameters α, β, γ and $deflection$ one at a time and plot the error in the MCF's predicted values. Naturally, we prefer higher errors in the MCF's predicted values which indicates failure to predict controller behavior, which in turn signifies lesser degree of manipulation and overall performance degradation. To demonstrate the efficiency of CoDeRa, we test it in the worst-case scenario, i.e., when the MCF has enough opportunity to learn the Controller behavior before CoDeRa is activated. If not otherwise stated, default values of the parameters are: $\alpha = 20$, $\beta = 50$, $\gamma = 5$, *simple* deflection. The default values have been chosen to create the desired worst-case scenario.

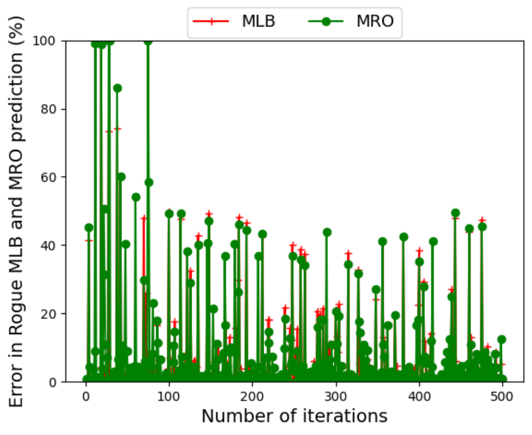


Figure 9.1: Deflection Type 1

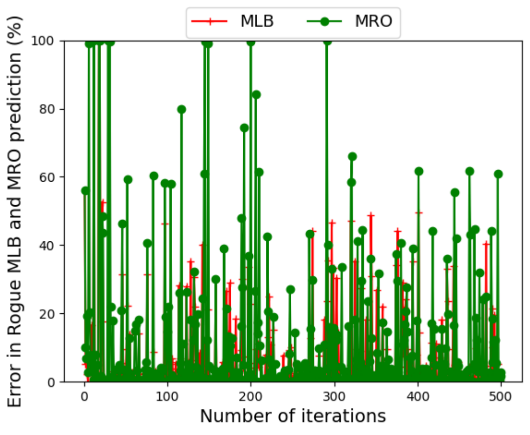


Figure 9.2: Deflection Type 2

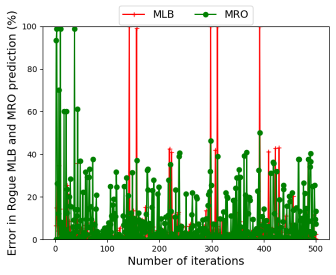


Figure 9.3: Performance degradation 1%

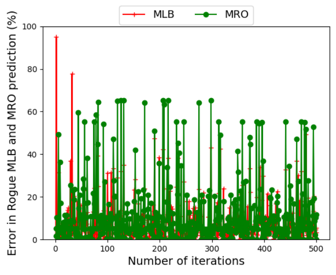


Figure 9.4: Performance degradation 5%

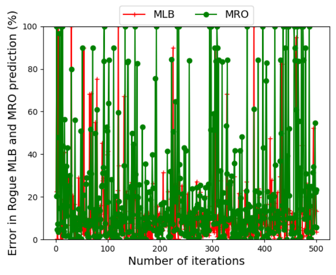


Figure 9.5: Performance degradation 10%

9.5.1 Impact of $deflection$ type

Fig. 9.1 and Fig. 9.2 show that for *simple* deflection, the error in MCF predictions is low. This is because with a constant amount of deflection, MCF can still learn the Controller behavior - both the original relation among $< s_i, p_i, f_i >$ and the amount of added deflection. In ML, sometimes a small constant noise is added to training dataset to prevent over fitting and in this case, *simple* deflection serves the same purpose. When $deflection$ is $random$, i.e., it introduces a random amount of "noise" at each time point, which makes it then difficult for MCF to learn the Controller behavior. This is why we see from Fig. 9.2 that MCF fails to learn, when $deflection$ is $random$, even after 500 iterations. Moreover, $random$ deflection avoids a constant degradation, e.g, for $\gamma = 5$, the system performance degradation is 5% for *simple* deflection but it varies between 0 - 5% for $random$ deflection. For these reasons, $random$ deflection is always the preferred over *simple* deflection.

9.5.2 Impact of γ

Fig. 9.3, Fig. 9.4 and Fig. 9.5 show that a higher performance degradation limit results in more error in MCF predictions, which is because it allows for a higher deflection. From Fig. 9.5 we see that in this case, the deflection value is so high that MCF fails to learn even after 500 iterations. However, although a very high limit (e.g. of γ is 30 or higher) completely inhibits the MCF's learning, it also leads to unacceptably high system performance degradation.

9.5.3 Impact of α

Fig. 9.6, Fig. 9.7 and Fig. 9.8 show that a higher the α value lowers the MCF prediction error, because the MCF gets more noise-free, accurate data to use to learn. So, for the MNO, it is always beneficial to keep $\alpha = 0$, i.e., start CoDeRa as soon as the system starts.

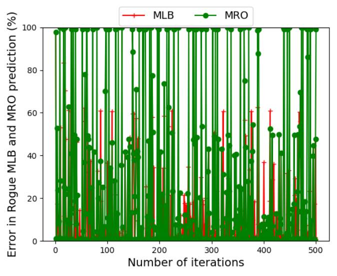


Figure 9.6: $\alpha = 0$

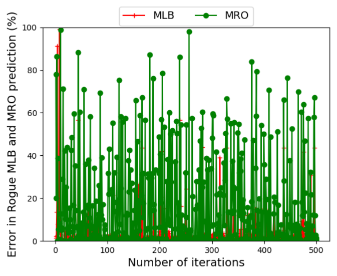


Figure 9.7: $\alpha = 20$

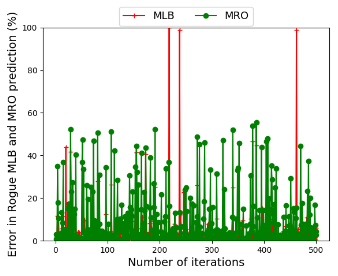


Figure 9.8: $\alpha = 100$

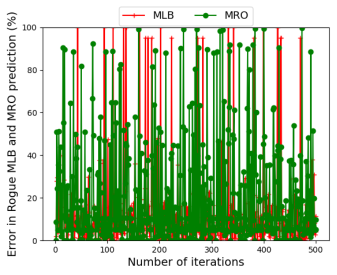


Figure 9.9: $\beta = 20$

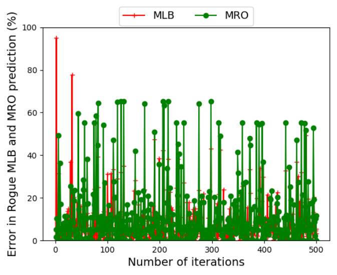


Figure 9.10: $\beta = 100$

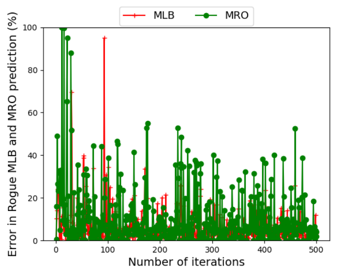


Figure 9.11: $\beta = 500$

9.5.4 Impact of β

Although, β cannot be configured by the MNO and solely depends on the vendor, we still need to understand its impact on CoDeRa performance. Fig. 9.11 shows that since the $deflection$ type is *simple*, despite the added noise, over time MCF is able to learn the Controller behavior and prediction error decreases when β increases.

9.6 Discussion on CoDeRa

Although we argue that MCFs can be neutralized using CoDeRa, there are several drawbacks with the approach:

- From the results we see the best possible configuration for CoDeRa is: $\alpha = 0$, $\gamma =$ as high as possible and *random* deflection. Now, we know $\alpha = 0$ means CoDeRa starts with the system, but, we also mentioned that keeping CoDeRa always active is not economical and causes unnecessary network performance degradation. Conversely, if CoDeRa is activated only after MCD detects an MCF, it might be too late since, by then, MCF could have completely learned how the Controller works, and thus, activating CoDeRa after that point is meaningless.
- CoDeRa is designed assuming an MCF gives up its manipulative behavior once it fails to learn how the Controller works. However, an MCF might be designed in a way to be manipulative irrespective of its success in learning the Controller behavior. CoDeRa is ineffective against such MCF(s).
- Finally, even if CoDeRa is successful in disabling MCF(s), there is unavoidable performance degradation and the network can not achieve its optimal performance.

As we argued and established in Section 9.1, with the current NMA architecture, CoDeRa is the only solution to disable an MCF in an operational system. So, the limitations stated here motivate us to find an alternate architecture to overcome these shortcomings. This alternate architecture is added in Appendix B.

9.7 Conclusion and Key Takeaways

In this chapter, we proposed a method that forces an MCF to give up its manipulative behavior and evaluated our proposed method in the simulation environment. The solution is based on certain assumptions and has some limitations highlighted in Section 9.6. Although the solution has certain drawbacks, we established that CoDeRa is the only solution to disable an MCF in the current mobile network management architecture. These drawbacks also forced us to think of an alternative mobile network management architecture described in Appendix B.

List of Abbreviations Used in This Chapter

CCO	coverage and capacity optimization	***MNO***	mobile network operator
CW	configuration weight	***MRO***	mobility robustness optimization
MCD	manipulative CF detector	***MSE***	mean squared error
MCF	manipulative CF	***NMA***	network management automation
MLB	mobility load balancing	***RMSE***	root mean squared error

Part IV

Orchestration

10. Orchestration: Problem and Related Works

Customizing multiple cell configurations simultaneously is a complicated task for a network operator. However, the task becomes easier if the knowledge of CAN is utilized. To utilize the knowledge of CAN, some type of interface is needed between the operator and CAN. In this chapter, we justify the need of the interface and discuss why an intent-driven interface is the most suitable one. The majority of the content in this chapter are from the following peer-reviewed article:

S. S. Mwanje, **A. Banerjee**, et al. Intent-Driven Network and Service Management: Definitions, Modeling and Implementation. ITU Journal on Future and Evolving Technologies (ITU J-FET), 3(3):555-569, 2022.

Some of the definitions related to intent based networking (IBN) have been taken from the author's contributions to the standard documents, namely ETSI ZSM 011 [12] and 3GPP SA5 Technical Report (TR) 28.912 [13].

10.1 Problem Statement

While providing a background on mobile networks and CAN, we mentioned that in a cell there are multiple adjustable network control parameters (NCPs) (e.g., TXP, RET) and observable KPIs (e.g., downlink throughput, RLF). A KPI value may change if one or multiple NCPs are changed. For example, downlink throughput may change if TXP and/or RET are/is modified. On the other hand, changing a single NCP might affect several KPIs simultaneously. For example, changing CIO might affect both RLF and the load in the cell. In [1] the authors depicted the interconnection among the NCPs and KPIs as shown in Fig. 10.1. We see from Fig. 10.1 that a single NCP has influence over multiple KPIs and simultaneously a KPI is influenced by multiple NCPs as well. For a KPI, an NCP can be one of the two types:

- Primary parameter, or,
- Secondary parameter

A primary parameter means that the NCP has a direct influence over the KPI, i.e., changing the NCP also changes the KPI significantly. On the other hand, a secondary parameter

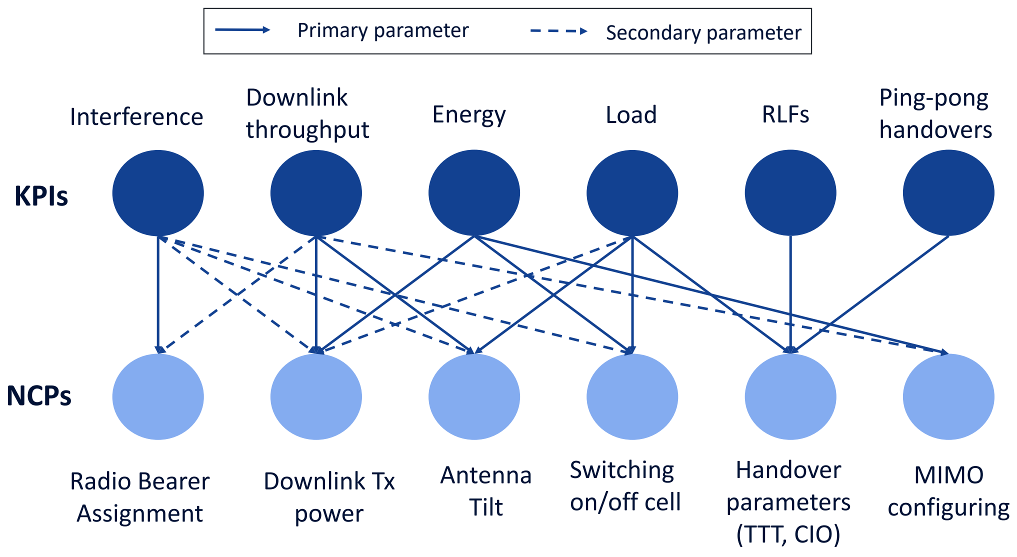


Figure 10.1: Interconnection among the cell NCPs and KPIs [1]

means that there is an indirect connection between the NCP and the KPI, and changing the NCP may not produce significant change in the KPI value. In any case, this type of inter-connected relationships among NCPs and KPIs make the radio access network (RAN) management a challenging task. Since all the KPIs are crucial in determining the quality of service (QoS) of the network, at any time MNO might need to fulfill several KPI related targets simultaneously. To achieve so, the MNO needs to have a complete knowledge of the inter-dependencies among the NCPs and KPIs across different network states, including the degree of change required in a certain NCP to obtain a required target KPI value.

Let us explain the problem with an example. Let us assume that, in cell X, the MNO wants to reduce TXP without affecting the handover related performances. To achieve so, the MNO needs to learn the relationship between TXP and handover related KPIs in the current network state. Now, as we already discussed in Section 1.1, MRO already has the knowledge; however, since there is currently no way of communication between MNO and MRO, MNO cannot utilize the knowledge of MRO to serve her purpose. The job of MNO becomes easier if there is a layer of abstraction between MNO and MRO where MNO specifies her intentions (multiple targets regarding handover related performances) and the layer of abstraction generates the appropriate actions to be executed by MRO to achieve the targets. This type of network management is called intent-driven network management (IDNM) where MNO specifies her targets (or, intentions) in the form of a sentence or multiple sentences and the network fulfills her targets (or, intentions) [12]. IDNM makes the job of network management easier for MNO since the MNO only needs to specify the targets and does not have to explicitly specify how those targets can be achieved. In this part of the thesis, we investigate this hypothetical layer of abstraction between MNO and MRO and propose methods to realize this abstraction in real life.

10.2 An Overview on IDNM

Before discussing our proposed solution on the realization of the abstraction between MNO and MRO, in this section we provide a brief overview on IDNM. An intent is the formal specification of the expectations, including requirements, goals and constraints, given to a technical system without specifying how to achieve them [87]. Intentions of MNO, or the expectations, originate from the existing business strategies or contractual agreements which are then expressed as “intents” by MNO. An intent in an autonomous management framework is expressed declaratively, i.e., as a goal that describes the properties of an outcome rather than prescribing a specific solution - it gives the framework the flexibility to explore various options and find the optimal one [12]. Unlike traditional software systems, where requirements are analyzed offline to detect and resolve conflicts prior to the implementation, intents are added to an autonomous framework during runtime. Adaptation to changed intent as well as conflict detection and resolution are therefore essential capabilities of an intent-driven autonomous framework [12]. In IDNM, intents are used for interactions between the management service consumer (e.g., MNO) and the management service producer (e.g., CAN). In short, the intent establishes a universal mechanism for defining expectations of the network operations by expressing goals, utility, requirements and constraints [12].

10.3 Related Works

Interest in IDNM comes from multiple institutions with concepts published in multiple pieces of work. The subsections below review the most relevant technical surveys, standards activities and open-source projects on or related to IDNM.

10.3.1 Related Literature

Although the earliest work on IDNM is focused on fixed networks, the most recent ones [88, 89, 90] have focused on IDNM for mobile networks as well. Very detailed and comprehensive structural reviews on this topic are covered in [91, 92]. In [91] the authors went beyond the scope of mobile networks to focus on generic intent-driven systems where the proposed system is expected to be utilized in the context of business support systems. The authors intended to find: 1) existing methods/techniques supporting intent-driven systems and 2) proposals for enabling realizations of intent-driven systems. Their observation was that intention 1) has been covered extensively but that there is inadequate work or techniques that can be combined to realize an intent-driven system.

In [92] the authors analyzed the IBN methods proposed in [88, 89, 90], identifying a number of shortcomings in those methods. They observed that the existing work missed at least one of the four core research areas: (i) processing and life cycle, (ii) orchestration and management, (iii) use-cases analysis and (iv) an architecture framework for intent-driven networks. The authors then introduced an architectural framework that relied on closed loop feedback for intent processing and interpretation. However, they still didn’t answer all the questions, so we take a step beyond the architecture to present other framework aspects that are crucial to realizing IDM systems.

The common outcomes from these publications is a set of enablers and architectural features along with a list of open issues. The open issues noted as requiring further work include intent description and translation, the description of services, processes etc. via data modeling languages and the integration of machine learning functionality.

10.3.2 Related Open Source Projects and Standards

Several open-source projects such as Open Network Foundation [93] have specified standardized intent-based northbound interfaces for software-defined networking (SDN) but have rarely published the conceptual ideas and evaluations of these specifications or implementations. On the other hand, IDM has only been recently introduced as a topic for discussion in most standards development organizations, only the Network Management Research Group (NMRG) of Internet Research Task Force (IRTF) has previously released Internet drafts on IDNM in [94] and [95]. The NMRG clarifies the concept of network intents, highlights stakeholder perspectives of intents, describes methods to classify intents, defines relevant intent terminologies and provides an overview of functionalities that serves as the foundation for further intent-based network management research. [96] describes the intent-driven management architecture, its key elements and their interfaces.

Experiential Networked Intelligence (ENI) [97] examined design options for integrating intent operations into its system architecture. This includes accepting, translating and validating intent statements; determining how intents affects the system's goals and operations as well as their use by business users, application developers and network administrators. Another background study is the 3GPP TR 28.812 [98] that describes the concepts of intent-based network management for service-based architecture (e.g., intent expression, intent translation, intent life cycle) covering various scenarios and key stakeholders. The potential use cases, their requirements, needed management services, operations and notifications to support intent-driven management for network and service management are described in 3GPP Technical Specification 28.312 [99].

TM Forum has proposed intent-based operation in IG1253 [100], introducing the intent management function as an architectural building block for intent-based operation which will receive an intent, make decisions about suitable actions towards fulfilling the intent, control the execution of these actions and report on the progress. Then, IG1253A [101] introduces the modeling concepts and artifacts that determine how an intent must be expressed, while IG1253C [102] defines the details of the intent life cycle including the related roles and tasks of the intent management function. Furthermore, [102] defines the interface through which the intent objects are exchanged, negotiated and managed as part of the life cycle management process. It however focuses on only the usage and management of intents within a single organization and so the proposed models and interfaces are not applicable for multi-vendor interactions.

Currently, ETSI ZSM 011 [12] is investigating intent-based network and service management within the ZSM framework evaluating different ways to model intents and intent-based interfaces as well as their use ZSM management domains and ZSM end-to-end service management domains. The research topics and potential future directions related to ZSM 011 are outlined by the original work [103] on Intelligent Intent Based Networks (I2BN). I2BN concept argues that intents are easier fulfilled if they are aligned with the capabilities of the network management micro-services, rather than having a broad intent language with free open-ended formalities that are not supported by the network's implementation. This approach enables the network management to autonomously find and assemble those micro-services that together fulfill an intent in a closed loop manner.

10.4 Conclusion and Key Takeaways

In a cell, there is a complex inter-dependency among the network parameters and KPIs, as shown in Fig. 10.1. So, if the MNO wants to customize multiple parameters and KPIs simultaneously, the MNO has to gain complete knowledge of this inter-dependency. It is a difficult and lengthy process to gain this knowledge. On the other hand, this knowledge is already possessed by CAN. So, to enable MNO to use the knowledge of CAN, some interface between them is needed through which MNO can utilize the knowledge of CAN. In this chapter, we proposed the idea of having an intent-driven interface between the MNO and CAN, and justified its usefulness. For better understanding, we also provided a detailed overview of intent-driven networking and summarized prior arts and standards in this field.

List of Abbreviations Used in This Chapter

CIO	cell individual offset
ENI	Experiential Networked Intelligence
IBN	intent based networking
IDNM	intent-driven network management
I2BN	Intelligent Intent Based Networks
IRTF	Internet Research Task Force
KPIs	key performance indicators
MNO	mobile network operator
NCPs	network control parameters
NMRG	Network Management Research Group
QoS	quality of service
RAN	radio access network
RET	remote electrical tilt
RLF	radio link failures
SDN	software-defined networking
TXP	transmit power

11. Intent-driven Network Automation Function Orchestration

In this chapter, we address the Q5 described in Section 1.3.3, i.e., we propose a method for intent-driven orchestration of CAN. The proposed method utilizes the existing knowledge of CAN for better management of cell parameters and acts as a bridge between the MNO and CAN. We start the chapter by describing a generic intent-driven architecture, which we modify later for the orchestration of CAN. We also evaluate our proposed solution in the simulation environment. The majority of the content of this chapter is based on the following peer-reviewed article:

A. Banerjee, S. S. Mwanje, and G. Carle. An Intent-Driven Orchestration of Cognitive Autonomous Networks for RAN management. In 2021 17th International Conference on Network and Service Management (CNSM), pages 380-384. IEEE, 2021.

The author also won the **Student Travel Grant (STG) Award 2021** in that conference for this paper. Apart from this paper, some content is from the following patent application:

A SYSTEM AND METHOD TO EXECUTE INTENT DRIVEN COGNITIVE AUTONOMOUS NETWORKS
Anubhab Banerjee, Stephen S Mwanje
WO, PCT application no.: PCT/EP2021/065607, filed 10 June 2021

In addition to the content described in the aforementioned entities, this chapter provides a brief overview on intent model and intent-driven system architecture. In particular, this chapter includes intent as sentence of components (Section 11.1.1), declarative intent model (Section 11.1.2) and imperative intent model (Section 11.1.3).

11.1 Intent Model and System Architecture

In Section 10.2 we gave a brief overview on IDNM. In this section we discuss different types of intent modeling and an end-to-end architecture of IDNM for different types of intent models. Intents may be specified in different forms and degrees of complexity. We discuss an intent's information elements and the related simple intent model, which we then evolve into a fully declarative model with multiple different information elements as well as an imperative model supporting the use of verbs.

11.1.1 Intent as sentence of components

From the perspective of the MNO, there can be many kinds of intents, from simple intents achievable via one command on one object to complex intents that include several commands on several network objects and nodes. Example intents may include:

- Restrict Handovers of fast-moving users to small cells.
- Allow load balancing to a cell Y or to only urban cells.
- Rehome a BS from controller A to controller B.
- Create a network slice of type IIoT.

From these examples, we see that an intent may be composed of a set of components, specifically:

- Intent targets, e.g.: Cells shall have a range of 1.5 km.
- Scope of the intent, e.g.: If they are macro-cells.
- Bordering conditions and constraints, e.g.: if cell border interference from neighbors is below -70dB

It is to be noted that the intent may include multiples of each component type, e.g. besides interference the constraint may also include "and if the signal strength at the cell border is above -60 +/-5 dB." The desired outcome, a list of measurable (state) values $s_n[t]$ that describe a managed network at discrete time t, can be collected into a state vector:

$$\mathbf{s}[t] = (s_1[t], s_2[t], \ldots, s_N[t])^T \quad \text{for } t = 0, 1, \ldots,$$

where t represents a generalized discrete time index that is derived from the measurement periods of the respective network functions. The state values include, among others, configuration parameters of the network functions, QoS metrics, performance management (PM) counters, latency values, failure counters and any other observed or derived metric while their combination refers to the state of the network or the fulfillment of the intent. For example, for the intent to "rehome a BS", the state dimensions and values could be scope=BS, homeNB=B. Given this observation, one might consider the intent to be modeled as a simple list of these components, i.e., as:

$$\begin{aligned} intent := \quad [\quad & scope(s), target1, target2, ... \\ & constraint1, constraint2, ...] \end{aligned}$$

Throughout this thesis, we use this structure of intent for implementation and evaluation of our proposed solutions. This, in fact, matches the idea that an intent defines a subspace (a set of points) in a multi-dimensional space, i.e. the intent specifies the desired/acceptable regions of the subspace. In that case, the challenge is that the intent does not only state the boundaries of regions but also includes conditions under which those boundaries will be applied, i.e., the intent must explicitly state the objects under consideration and their context as well as the context to be applied as boundary conditions or constraints for a single target or a group of targets. The intent should be extensively modeled to distinguish these information elements as discussed in the next section.

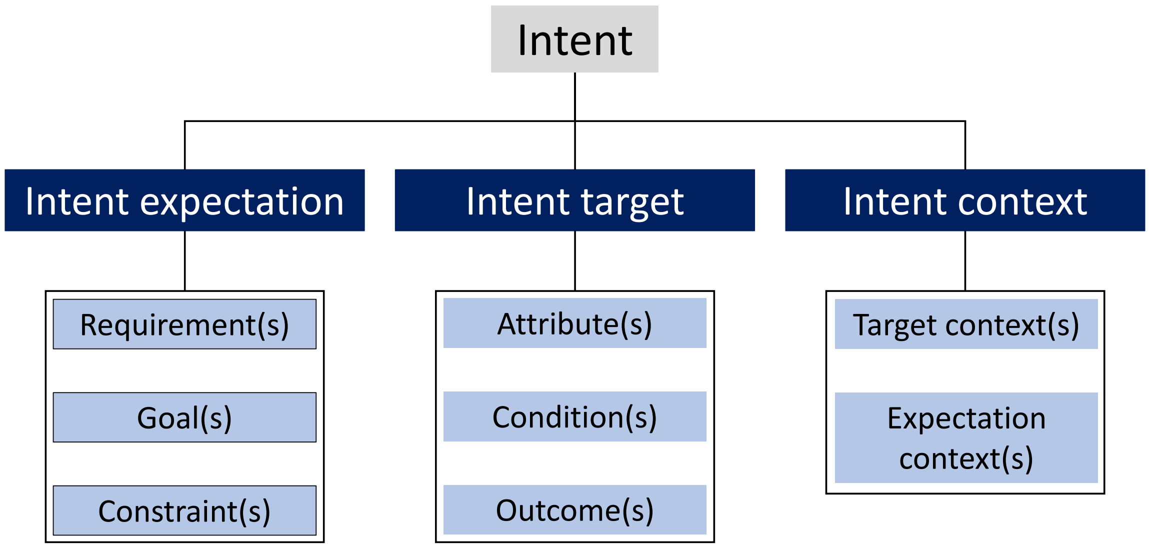


Figure 11.1: Intent Model

11.1.2 Declarative intent model

The intent information model needs to include the information elements that identify its scope, requirements or goals and its constraints. The declarative model, shown in Fig. 11.1, explicitly defines each intent consisting of three components: intent expectations, intent targets and contexts.

The intent expectation is the set of requirements, goals and constraints for one type of object and an intent may have one or more such sets e.g. one for cells and another for network slices. A desired outcome in an intent expectation is the list of one or more intent targets, to be enforced by the IDNM system. Each intent target is a triplet of the attribute on which the target is set, the condition constraining the outcome (e.g. "=", "<",">") and the desired outcome. In this case, the attribute is any aspect of the control or observation state spaces of the system under control.

Each information element may have a context that defines its scope and/or constraints. Context may be set for the intent target and intent expectation as target context and expectation context respectively. On the same note, a global intent context may be added to apply to all intent expectations within an intent.

11.1.3 Imperative intents: using verbs in intents

The declarative model is explicit in defining all the desired components of the intent but it is not concise. For human users and as illustrated by Table 11.1, the declarative statement can be very complex to compile and requires a lot of prior information.

For example, let us consider that an operator wishes to request that no ES should be executed for a given cell or for cells in location X as example 3 in Table 11.1. Stating this intent declaratively is challenging, since:

- Either, the operator has to know the parameters that manage ES in the cell (i.e. ES_On) on which to state the desired values,
- Or, the operator must know the metrics to which ES contributes for the operator to set values for these metrics which are then interpreted by the system to disable ES.

Table 11.1: Example specification of network management intents

	Goal (Imperative Intent)	Declarative Intent (Ensure that ..)	Imperative Intent Challenge
1.	Restrict Handovers of high mobility users to small cells	for Object [cell, Size = small] [HO_Allowed = True] Object [cell, Size$\neq$ small], [HO_Allowed = False] if cellMobilitiy = high	Interpret “Restrict” and ”small cells” as ”True” for small cells; ”False” otherwise
2.	Restrict ES to Rural cells on week days	for Object [cell, Location =Rural] [ES_On=False; Time=Weekdays;]	Interpret “Rural” as filter to ”create cell”; and “week days” as context for ES_On=False
3.	Avoid ES for cell Y / small cells / cells in Location X	for Object [cell, Location=X, ES_On=False]	Interpret “Location X” as a filter on object cell list
4.	Rehome a nodeB from RNC A to RNC B	for Object [cell id= x]; [RNC is B]	Interpret “Rehome” into “cell attribute, RNC“ ”is equal to”
5.	Create an IIoT network slice	for Object [Slice, id=”new”] [SliceProfile=IIoT]	Interpret “Create” into “id=new“

On the other hand, the imperative statement (as stated in the “Imperative” column in Table 11.1) is very concise and can be stated with the same degree of completeness. The statements ”Restrict ES to Rural cells” and ”for Object Rural cells [ES_On=False]” are equivalent, but it is easier for the operator to state this intent using the verb, as opposed to determining the parameter ”ES_On” and its possible values before stating the intent. Such imperative intents can thus be modeled by adding a verb to the declarative model. As a default, an imperative intent expectation implies that by using some terms (e.g., by using a verb), not all the fields of the intent expectation model (as expected for declarative intent expectations) need to be fully stated. However, for intent realization, any such imperative intent needs to be interpreted into a declarative intent, e.g. as illustrated by the examples in Table 11.1.

11.1.4 End-to-end architecture

The intent models as discussed above characterize the interface(s) over which the consumer and the producer of the IDNM service can interact with one another to express and receive the intents. However, for the fulfillment of the intents, it is necessary to insert the interfaces into the end-to-end architecture of the IDNM system. The proposed architecture (shown in the upper half of Fig. 11.2) provides the means to capture intents, format them into realizable outcomes and execute their fulfillment. The intents may be submitted in a form of text e.g. via a text file, legacy commands via a command line interface (CLI), through interaction with a graphical user interface (GUI) or as speech e.g. via an audio interface that captures the operator’s audio commands. Regardless of any such input interface, the operator has access to an intent specification platform (ISP) to which she specifies the intent to be executed. The ISP may take in the operator’s request and parses the input to identify the fields fitting to the defined intent specification syntax, according to the imperative or the declarative intent models.

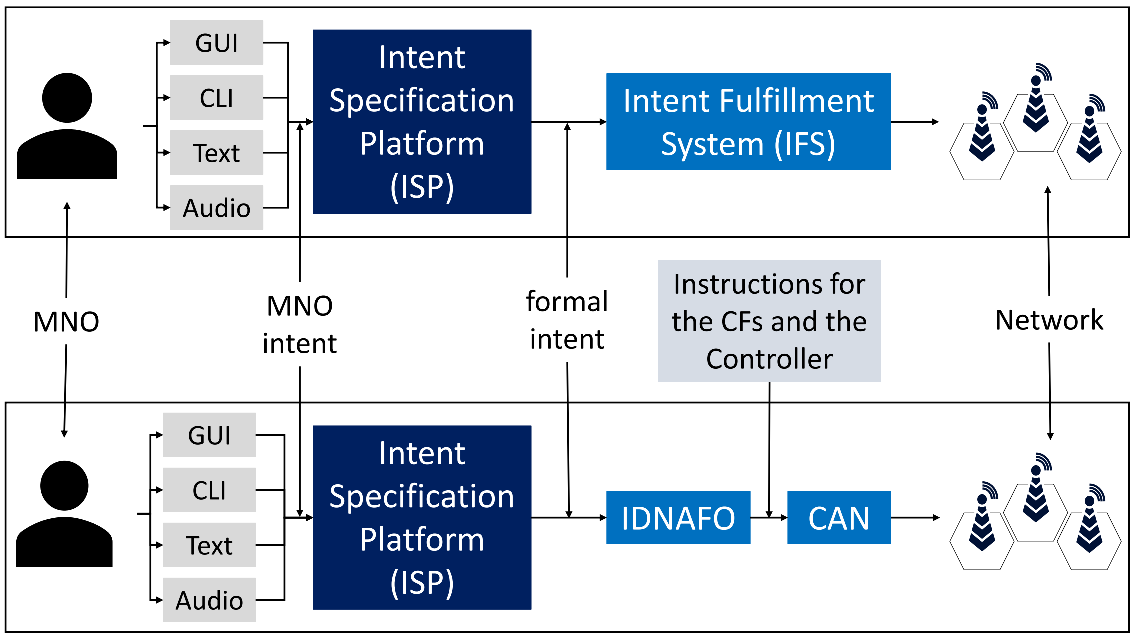

GUI
CLI
Text
Audio
Intent Specification Platform (ISP)
Intent Fulfillment System (IFS)
MNO
MNO intent
formal intent
Instructions for the CFs and the Controller
Network
GUI
CLI
Text
Audio
Intent Specification Platform (ISP)
IDNAFO
CAN

Figure 11.2: Proposed end-to-end IDNM architecture

The outcome of the ISP, which is a hierarchy of hash functions according to the declarative intent model that describe the features of the intent to be fulfilled, is denoted as "formal intent". For example, The intent specifications may be similar to the declarative intent entries in Table 11.1. The declarative intent is the input to the intent fulfillment system (IFS) which then undertake actions to realize the intent. There can be multiple methods for fulfilling intents, which may also depend on the nature and contents of the intents. So, in short, IFS acts as a bridge between ISP and the network.

11.2 Intent-driven CAN orchestration

In this section, we modify the generic end-to-end IDNM architecture to use it for managing CAN. It is to note that, in a similar way, this generic design can always be modified to develop intent-based management solution for any other type of network.

Our proposed IDNAFO takes a formal intent as its input and generates appropriate actions for the Controller and CFs at its output. End-to-end workflow of IDNAFO consists of three sequential steps, based on which we introduce three different functionality blocks within IDNAFO, as depicted in Fig. 11.3. Those are described as follows:

- As soon as IDNAFO receives a formal intent from ISP, its first task is to check if the intent is valid for CAN, i.e., if the intent can be executed by the Controller or CFs. This task is done by the block Intent Identifier (II).
- After the II identifies that an intent is executable by CAN, next task is to classify the intent based on its content. This task is done by Intent Classifier (IC).
- After the classification is done, based on the type of intent, IDNAFO takes subsequent actions like sending specific commands to the Controller or CFs or both. This is done by Intent Decision Maker (IDM).

11.2.1 Intent Identifier (II)

As we already know, only the intents which deal with key performance indicator (KPI)(s) and/or network control parameter (NCP)(s) are relevant for CAN. An intent like "switch

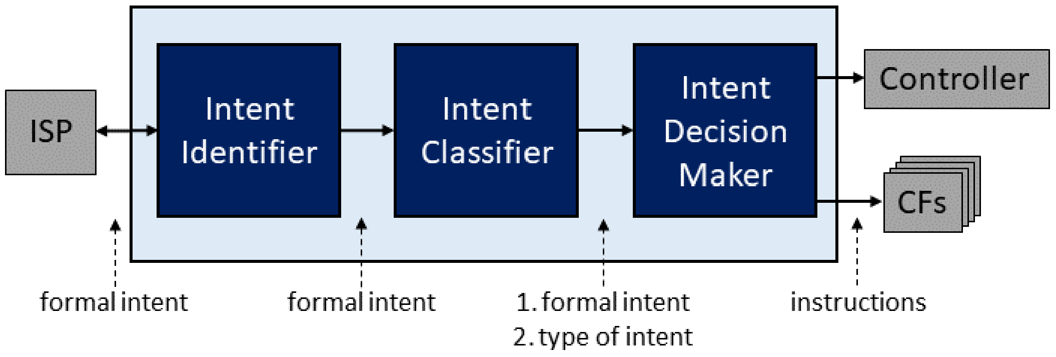


Figure 11.3: Proposed architecture of IDNAFO

Algorithm 11.1: Algorithm of Intent Identifier

```
Input: target, constraint
Output: intent_accept, intent_reject
for each target do
    if (target related to NCP) or (target related to KPI) then
        for each constraint do
            if (constraint related to NCP) or (constraint related to KPI) or
              (constraint is ∅) then
                intent_accept
            else
                intent_reject
    else
        if target is ∅ then
            for each constraint do
                if (constraint related to NCP) or (constraint related to KPI) then
                    intent_accept
                else
                    intent_reject
        else
            intent_reject
```

cell X off/on" is beyond the operational capability of CAN. So, as soon as IDNAFO receives a formal intent from ISP, its first task is to check if the intent is relevant for CAN which is done by II. To accomplish this task, we propose Algorithm 11.1 for II to check the validity of the intent. If II finds the intent to be valid for CAN, it passes the intent to IC for further processing. Otherwise, II may send the intent back to the MNO or to other components in the IDNM system based on the implementation.

11.2.2 Intent Classifier (IC)

Based on the components of CAN, by which an intent can be executed, IC classifies a formal intent into three distinct categories. Classification is done to find out by whom the intent is to be executed. Classification is done by evaluating an intent following the three rules:

Rule 1: If either the target(s) and constraint(s) are defined on a NCP while the other is not defined or both are defined on a NCP, then the intent is Type 1 intent.

Table 11.2: Categorization of intents

Type	Description	Example intents
Type 1	Intent fulfills one of: 1.1 target(s) defined on NCP, no constraint(s) defined 1.2 constraint(s) defined on NCP, no target(s) defined 1.3 both constraint(s) and target(s) defined on NCP	1.1 increase cell X TXP 1.2 make cell X TXP remain constant 1.3 make cell X TXP remain constant and change RET by 3 degree
Type 2	Intent fulfills one of: 2.1 target(s) defined on KPI, no constraint(s) defined 2.2 constraint(s) defined on KPI, no target(s) defined 2.3 both constraint(s) and target(s) defined on KPI	2.1 reduce cell X interference 2.2 avoid increase in cell X interference 2.3 increase cell X throughput without changing interference
Type 3	Intent fulfills one of: 3.1 target(s) defined on NCP, constraint(s) defined on KPI 3.2 target(s) defined on KPIs, constraint(s) defined on NCP	3.1 make cell X TXP constant and reduce interference 3.2 change cell X TXP to 45 dBm without changing interference

Rule 2: If either the target(s) and constraint(s) are defined on a network metric like a KPI while the other is not defined or both are defined on a network metric like a KPI, then the intent is a Type 2 intent.

Rule 3: Rest all are Type 3 intents. In these intents one of either the target(s) or the constraint(s) is defined on a NCP while the other is defined on a network metric like a KPI.

The corresponding categories may be grouped as described in Table 11.2.

11.2.3 Intent Decision Maker (IDM)

Based on the category of the intent, IDM decides what instructions are to be sent to the Controller and/or CFs.

Type 1: As Type 1 intent contains only NCP, it is executed only by the Controller.

Type 2: To execute a Type 2 intent, IDM first identifies the CFs which are responsible for managing the KPIs. After that, instructions are sent to those CFs to take specific actions (increase, keeping constant or decrease) on their managed KPIs. At the same time, instructions are also sent to the Controller to make all the changes in the network as proposed by those particular CFs.

Type 3: Type 3 intent contains both KPIs and NCPs, so instructions are given to both the CFs and the Controller. Instructions are sent to the specific set of CFs to take specific actions (increase, keeping constant or decrease) on their managed KPIs. At the same time, two types of instructions are sent to the Controller: (i) take instructed actions on control parameters, and, (ii) make all the changes in the network as proposed by those particular CFs.

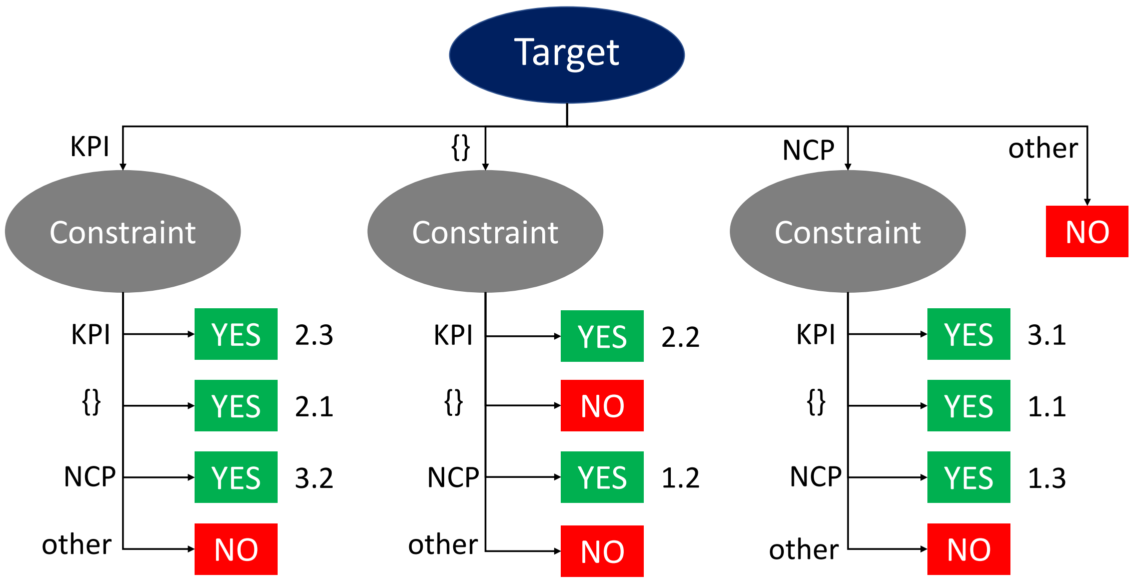


Figure 11.4: Intent classification by IC

11.2.4 Alternate Design

In an alternate design, functionality of II and IC can be combined into a single block which implements the decision tree shown in Fig. 11.4.

11.3 Implementation and Evaluation

To show that our proposed architecture can be used in a real-life scenario, we implement the end-to-end design in the same simulation environment described in Chapter 3 and provide the implementation of each component in detail.

11.3.1 ISP implementation

ISP can be implemented in one of these two ways:

- In the first way, an intent from MNO does not necessarily follow a predefined structure, it can be just an intention or wish of MNO expressed in valid English [89]. In this case natural language processing (NLP) can be used for parsing the language, for example, the algorithm proposed in [104] is a suitable candidate for use in this purpose. However, our focus in this paper is to provide an end-to-end design for intent driven CAN, so we do not concentrate on NLP based ISP or propose any new algorithm for implementation, rather we focus on how a formal intent can be executed by CAN.

- In the second way, MNO has to express the intent in a particular language structure. In this case a rule based parsing algorithm can be used in ISP to convert the intent into machine understandable format.

Both these ways of ISP implementation have some advantages and disadvantages. First way of implementation gives more flexibility to MNO, but it is difficult to implement whereas the second way is easy to implement but provides less flexibility to MNO. Since the way of implementation does not have any impact on the final result, in this thesis we implement ISP using the second way.

11.3.2 IDNAFO implementation

We use Python 3.7 to implement IDNAFO in a simulation environment. We use two separate lists to store names of all the predefined KPIs and NCPs. These lists are used by II to check if the targets and constraints in the intent are defined on KPIs and/or NCPs. In a real life scenario, these lists can be defined by MNO before the system starts and can be modified at any time without affecting rest of the system. Next we implement the decision tree shown in Fig. 11.4 which is equivalent to the implementation of both II and IC. Using the decision tree II can determine if an intent is relevant for CAN. If an intent is found to be relevant for CAN, corresponding to the tree traversal we assign the intent type. Under the green boxes in the tree, we also mention the type of intent, detecting which is the functionality of IC. For example, for the leftmost tree traversal, i.e., when target is defined on KPI and constraint is defined on KPI, the intent is defined as Type 2.3. Corresponding to each type of intent, we initialize a set of commands which are to be sent to the Controller and/or CFs which have predefined structures where the names of the KPIs and/or the constraints can be directly filled from the formal intent. This standalone module does not have any simulator specific dependency and can be integrated on top of suitable networking simulators for demonstration purposes as well.

11.3.3 CAN implementation

Implementation of CAN is the same as described in Section 3.3.

11.3.4 Observation

The simulator automatically updates the changes done by the CAN module and reconfigures the radio processes, and thus, in this way the whole system remains up and running all the time just like a real life network. Since we implement our proposed design on top of this simulator, it proves that our proposed design is fit for use in real life. Since no intent database for RAN management is currently available, we create a database with 20 intents by ourselves, test and observe that all of them can be executed using our proposed design. Average execution time is 0.12 ms, which proves that our proposed design works fast and worthy for real life use. We also observe the necessity of a feedback mechanism for the intents which cannot be fulfilled by IDNAFO.

11.4 Conclusion and Key Takeaways

In the previous chapter, we discussed the problem of cell parameter customization in the mobile network. We highlighted that the problem could be addressed by having an intent-driven interface between the MNO and CAN. In this chapter, we proposed the end-to-end design of such an intent-driven interface. We implemented the interface in the simulation environment and showed that our interface works as per expectation. From the simulation results, we also can conclude that the interface is fit for use in real-life scenarios.

List of Abbreviations Used in This Chapter

CLI command line interface

ES energy savings

GUI graphical user interface

IC Intent Classifier

IDNAFO intent-driven network automation function orchestrator

IDM Intent Decision Maker

IDNM intent-driven network management

IFS intent fulfillment system	***NCP*** network control parameter
II Intent Identifier	***NLP*** natural language processing
ISP intent specification platform	***PM*** performance management
KPI key performance indicator	***QoS*** quality of service
MNO mobile network operator	***RNC*** radio network controller

12. Intent-driven Conflict Detection and Resolution

In this chapter, we address the Q6 described in Section 1.3.3, i.e., we discuss possible shortcomings of IDNAFO and how they can be overcome. Although in the last chapter, we saw that IDNAFO could successfully coordinate with CAN to serve the purpose of the MNO, it has one primary shortcoming: it cannot detect conflicts within an intent. The conflicts, that arise from the same intent, are denoted as contradictions, and can only be detected in runtime. In this chapter, we propose a method to detect such contradictions and evaluate our proposed solution in the simulation environment. The majority of the content in this chapter is from the following peer-reviewed article:

A. Banerjee, S. S. Mwanje, and G. Carle. Contradiction management in intent-driven cognitive autonomous ran. In 2022 IFIP Networking Conference (IFIP Networking), pages 1–6. IEEE, 2022.

This chapter expands the content of the aforementioned article on different aspects of the proposed solution.

12.1 Contradiction in intent-driven CAN

Although in Chapter 11 we found that IDNAFO with CAN provide an end-to-end design to execute MNO intents, it does not detect the contradictions arising from the same intent. To elaborate the concept of contradiction, let us consider the following intent: ”Increase handover success to x_1% and reduce network load by y_1%”. MRO and MLB are the two CFs responsible for managing the KPIs “successful handovers” and “network load”. Both MLB and MRO share two ICPs: TTT and CIO. To achieve x_1% handover success and y_1% load reduction, let us assume that MRO and MLB propose t_1 and t_2 respectively as required TTT values. If $t_1 \neq t_2$, it gives rise to a contradiction in TTT value. Similarly, if MLB and MRO propose different CIO values, that gives rise to a contradiction in CIO value as well.

Now, it is not possible for MNO or IDNAFO to know beforehand what TTT or CIO values MLB and MRO will suggest and if those values will be equal. As already mentioned in Chapter 2, depending on the network state, these TTT or CIO values are generated by the CFs based on their learning and unknown to any other entity. Since IDNAFO is

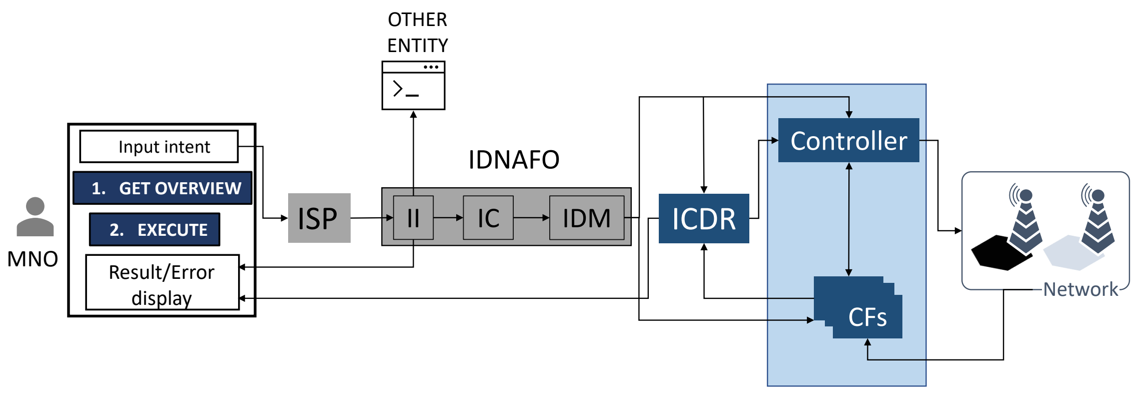


Figure 12.1: End-to-end architectural design of our proposed solution

not capable of detecting or removing this kind of contradiction within an intent, any intent with multiple KPIs can potentially give rise to contradictions which are not visible until execution. To circumvent the contradictions, it is recommended to have a system or solution that enables MNO to visualize potential contradictions before the intent is executed.

12.2 Proposed solution

To overcome the aforementioned problem, we propose a new functionality, called ICDR, which can be implemented separately or within the Controller. The functionality of ICDR is two-fold:

- Contradiction detection, i.e., it notifies MNO if an intent contains any type of potential contradiction.
- Contradiction removal, i.e., ICDR also notifies MNO about possible alternatives to circumvent the contradiction.

Our proposed design, depicted in Fig. 12.1, consists of two parts: (i) a GUI or similar interface that interacts with MNO, and, (ii) ICDR that interacts with CAN. The interface provides two options:

- Option 1 (”GET OVERVIEW”): if MNO is unsure if the intent is contradiction free, it is always recommended to use this option. If the intent is contradiction free, via the GUI display ICDR informs MNO that the intent can be executed as it is. If ICDR detects contradiction(s) in the intent, it proposes a few possible alternatives to MNO to circumvent the contradiction.
- Option 2 (”EXECUTE”): if MNO is sure that the intent is contradiction free, it is recommended to select this option. Usually intent that deals with a single NCP or KPI can be safely selected for this.

12.2.1 Contradiction detection

As soon as ICDR is triggered, it separates the NCPs from the KPIs and requests the CFs, who manage those KPIs, to propose configurations so that each individual KPI target can be achieved. Following the previous example, ”increase handover success to x_1% and reduce load by y_1%", ICDR identifies the KPIs (handover success, load) and requests the CFs managing those KPIs (MRO, MLB) to propose configurations (values of TTT and CIO) so that the targets can be achieved. Now it might be the case that both targets cannot be achieved simultaneously, for example, MLB finds that in the current network state, load can be reduced by y_1'% at maximum, $y_1' < y_1$. In that case, MLB always sends the TTT and CIO values for which load reduction is y_1'%.

Now, for the sake of explanation, without any loss of generality, let us assume that MLB proposes (t_l, c_l) and MRO proposes (t_r, c_r) as required (TTT, CIO) values to achieve the targets specified in the intent. There are four possibilities:

- Possibility 1 (P1): $t_l = t_r$ and $c_l = c_r$.
- Possibility 2 (P2): $t_l = t_r$ and $c_l \neq c_r$.
- Possibility 3 (P3): $t_l \neq t_r$ and $c_l = c_r$.
- Possibility 4 (P4): $t_l \neq t_r$ and $c_l \neq c_r$.

Out of these four possibilities, only in P1 there is no contradiction, in P2 and P3 there is one contradiction each and in P4 there are two contradictions.

12.2.2 Contradiction removal

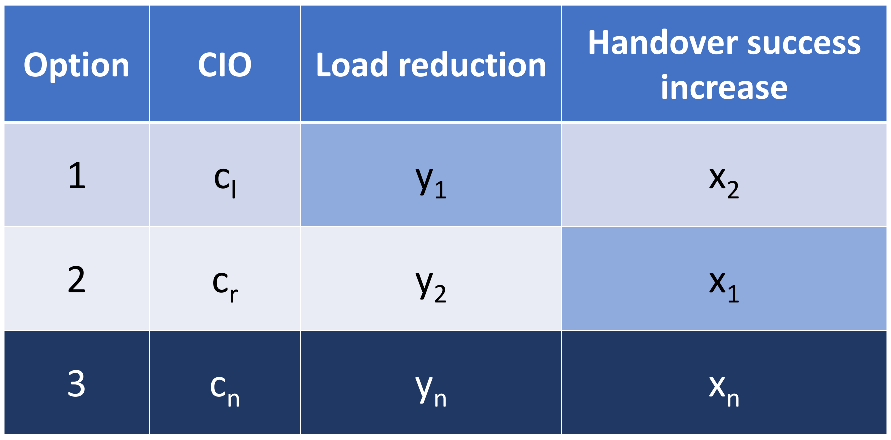
For contradiction removal, ICDR uses a NSWF (discussed in Section 5.2) based approach. The reason behind using NSWF based solution is: in [45], it has been mathematically proven that in a traditional resource sharing scenario, NSWF provides the 'optimal' solution, where 'optimal' signifies that the solution is optimal for all the parties involved in the sharing. Since in CAN all the CFs are given equal priorities, NSWF is therefore used to remove contradictions and find values which are optimal for the combined interests of all the CFs.

In case of a contradiction, ICDR requests all the CFs involved (following the same example, MLB and MRO) to send their utility functions (UFs) (discussed in Section 2.4) over the conflicting NCP(s) (in P2, conflicting NCP is CIO, in P3 it is TTT and in P4, both TTT and CIO). After obtaining the UFs, ICDR finds the conflicting NCP value for which the product of the UFs is maximum since that is how NSWF optimal is calculated. To find the NSWF solution, for each configuration value, the Controller calculates the product of the UFs and selects the one for which the product is maximum. To put it mathematically, if p is a shared configuration among CFs: $\mathrm{F_1}$, $\mathrm{F_2}$, . . . , $\mathrm{F_n}$, UFs of the CFs are: $f_1(p)$, $f_2(p)$, . . . , $f_n(p)$ and $\{p_1, p_2, ..., p_m\}$ are acceptable values of p, then p_i is an NSWF solution provided

$$\prod_{r=1}^{n} f_r(p_i) > \prod_{r=1}^{n} f_r(p_j)$$

$\forall j \in [1, m]$, $j \neq i$.

Let us elaborate further with an example. Let us assume that ICDR finds a contradiction in the intent and it is the case P2, i.e., there is a contradiction over CIO. ICDR asks both MLB and MRO to send their UFs, calculates NSWF and finds c_n to be the NSWF value of CIO. Then ICDR asks MLB to send the load values corresponding to c_n, c_r and asks MRO to send the handover success values corresponding to c_n, c_l. Then ICDR puts everything in a table as shown in Fig. 12.2 and send them to MNO to choose from, highlighting the recommended option. Targets specified in the intent are highlighted in the first two rows, and ICDR recommended option is listed in the last row. MNO can choose the one that suits her interests the best from these alternatives.



Option	CIO	Load reduction	Handover success increase
1	c_l	y_1	x_2
2	c_r	y_2	x_1
3	c_n	y_n	x_n

Figure 12.2: Suggestions to MNO from ICDR as alternatives

12.2.3 End-to-end workflow

After MNO inputs an intent, the intent is sent to ISP and is converted into a formal intent. The formal intent is then forwarded to II by ISP to check if the intent can be executed by CAN. If II finds the intent to be invalid, it sends the intent back to MNO or to other networking entities, based on the implementation. Otherwise, II forwards the intent to IC, which classifies the intent based on its content and separates the control parameters and KPIs. These are then forwarded to IDM, which identifies the CFs managing the KPIs and forwards the control parameter and KPI related instructions to the Controller and those CFs respectively.

If MNO chooses ”EXECUTE” while inputting the intent, IDM asks the Controller to execute the parameter related actions. IDM also asks the CFs to send the necessary configurations to the Controller which are required to execute the KPI related actions and IDM asks the Controller to change the configurations proposed by the CFs. In case of a conflict, the Controller resolves it using NSWF based approach (Chapter 5) or Eisenberg-Gale based solution (Chapter 6). If MNO chooses ”GET OVERVIEW”, IDM sends the parameter related instructions to ICDR. IDM also asks the CFs to send the necessary configurations to the ICDR which are required to execute the KPI related actions. After ICDR receives the configurations from the CFs, it checks for potential contradictions (overlaps) in those suggested configurations. If no contradiction is found, ICDR informs MNO that the intent is contradiction-free and ready for execution. If ICDR finds contradiction, it calculates the possible alternatives as discussed in the previous section and sends these alternatives to MNO.

End-to-end workflow of our proposed solution (ICDR combined with IDNAFO) is shown in Fig. 12.3.

12.3 Evaluation

12.3.1 Experimental setup

To evaluate ICDR, we implement the end-to-end system in a simulation environment. The implementation of IDNAFO and CAN have already been covered in the previous chapter and they are not reproduced here. Implementation of ICDR is pretty straightforward and is done using Python 3.7. Since the performance of ICDR is independent of how IDNAFO takes the intent as its input, we did not implement ISP and used formal intent as input.

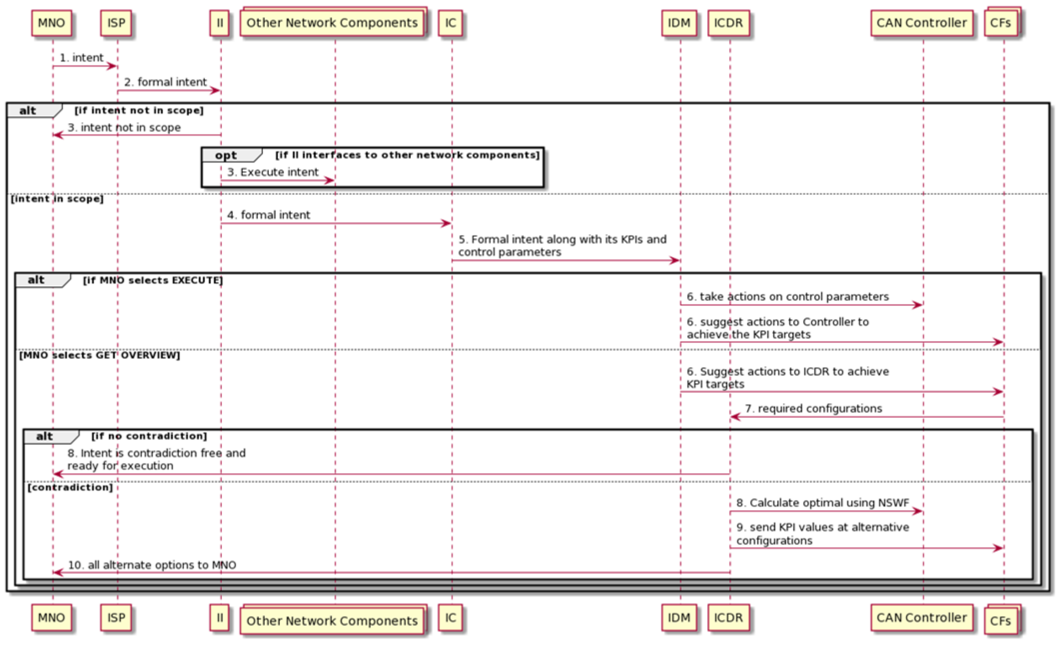

MNO
ISP
II
Other Network Components
IC
IDM
ICDR
CAN Controller
CFs
1. intent
2. formal intent
alt [if intent not in scope]
3. intent not in scope
opt [if II interfaces to other network components]
3. Execute intent
[intent in scope]
4. formal intent
5. Formal intent along with its KPIs and control parameters
alt [if MNO selects EXECUTE]
6. take actions on control parameters
6. suggest actions to Controller to achieve the KPI targets
[MNO selects GET OVERVIEW]
6. Suggest actions to ICDR to achieve KPI targets
7. required configurations
alt [if no contradiction]
8. Intent is contradiction free and ready for execution
[contradiction]
8. Calculate optimal using NSWF
9. send KPI values at alternative configurations
10. all alternate options to MNO

Figure 12.3: Flowchart of our proposed solution

Most challenging part of the implementation is creating an intent database for RAN (IDB-R) since there is currently no such database is publicly available. We created an IDB-R consisting of 30 formal intents manually by ourselves, while making sure that each intent contains at least 1 contradiction. Theoretically, any finite number of contradictions can exist in an intent, however, in IDB-R, maximum number of contradictions in a single intent is currently 4. However, since the contradiction removal process of ICDR is independent of the number of contradictions, this value does not affect our evaluation framework. On an average, an intent in IDB-R contains 3 contradictions.

12.3.2 Observation

We already mathematically established that when NSWF is used, contradiction removal process is optimal. So, instead of evaluating the optimality of our proposed solution, in this Section use the following three metrics for performance evaluation of our proposed framework:

(i) *Success in contradiction detection*: as the name implies, this measures the success rate of ICDR in detecting a potential contradiction in an intent. For our proposed solution, the success rate 100%, i.e., ICDR is able to detect all the 90 contradictions.

(ii) *Success in contradiction removal*: this measures the success rate of ICDR in removing a potential contradiction using NSWF. For our proposed solution, this success rate is also 100%, i.e., ICDR is able to remove all the 90 contradictions using NSWF.

(iii) *Time*: this accounts for the total time taken to detect and resolve a contradiction and is measured in ms. For our proposed solution, we used a quadcore Intel Core i5-10310U@1.7 GHz CPU, where this value varies between 0.012 - 0.019 ms.

From these three metrics, we see that our proposed solution is efficient for use in real time network management.

12.4 Standardization impact

After discussing the existing standards on intent-driven RAN management in Section 10.3.2, in this Section we discuss the impact of our work on ongoing standardization efforts. It is advisable that any real-life deployment of ICDR supports multi-vendor integration. To achieve so, inputs and outputs of ICDR need to be specified in network management specifications, for example, in 3GPP SA5 or ETSI ZSM. Although 3GPP SA5 already provides descriptions on controlling the behavior of the CFs by configuring their goals, it does not provide any mean for configuring the Controller. So, in that case 3GPP resource model needs to be extended with models for CAN Controller functionality and the methods to configure such Controller.

On the other hand, the ICDR takes a formal intent based on which it determines if the intent can be fulfilled without any contradiction. Such a formal intent may be generated and compiled by an ISP coming from any non-telco vendor like a ML audio processing startup. To allow for integration between these non-telco-centric ISPs and the telco-centric ICDR, the intent specification interface and specifically the structure of formal intent need to be standardized. This is actually an in-extensive extension to existing standards since the formal intent's attributes and their values are already speechified in the existing specifications. For example, managed objects, control parameters and metrics are already specified in the 3GPP network resource models. Additionally, the intent specification interface needs to be extended with messages through which

- The ICDR indicates to the operator that a given intent contains contradictions.
- The ICDR informs the MNO of possible alternatives.
- The MNO specifies a given preference among the candidate options.

12.5 Conclusion and Key Takeaways

IDNM plays a crucial part in network and service management in the next-generation networks. Although quite a number of research papers exist on IDNM based RAN management, the majority of them only provide an abstract overview without any implementation design. In this chapter, we discussed contradiction detection and removal in intent-driven CAN orchestration for cell configuration management and implemented our proposed solution in a simulation environment for evaluation purposes. Apart from that, we also discussed the impact of our work on worldwide network standardization activities. From these discussions, we can conclude that our proposed solution can address the problem above and is fit for use in real-life scenarios.

List of Abbreviations Used in This Chapter

CIO	cell individual offset
GUI	graphical user interface
ICDR	Intent Contradiction Detector and Remover
ICPs	input control parameters
IDB-R	intent database for RAN
IDNAFO	intent-driven network automation function orchestrator
IDNM	intent-driven network management
ISP	intent specification platform
KPIs	key performance indicators
MLB	mobility load balancing
MNO	mobile network operator
MRO	mobility robustness optimization
NCP	network control parameter
NSWF	Nash Social Welfare Function
TTT	time to trigger
UFs	utility functions

Part V

Conclusion

13. Conclusion and Future Directions

This chapter summarizes all the key contributions, spread throughout the thesis in multiple chapters, into a single container. Along with our key findings, in the latter half of this chapter, we also talk about certain aspects of our work that are considered worthy of further investigation in the future.

13.1 Thesis Contributions

One of the critical components of any research activity is the environment in which the research has been conducted. The environment is used to study and evaluate the solutions proposed in the research activity. In our case, we used a simulation framework for this purpose. We used a Nokia internal system-level mobile network simulator in our thesis, implemented some independent Python modules on top of it, and kept the simulation environment homogeneous throughout the thesis.

The premise of this thesis is the study of cognitive, open, multi-vendor network management automation. Based on this premise, we studied three essential aspects: (i) coordination, (ii) trust, and (iii) orchestration. For each aspect, we identified relevant critical research problems, justified the importance of those problems, proposed solutions to those problems, and provided a description of the state-of-the-art to show the novelty of our proposed solutions. The thesis is divided into three parts, where each part covers the problem statements and proposed solutions pertaining to each aspect. So, while summarizing the crucial observations related to each aspect, we divide this section also into three subsections, and each subsection is devoted to one aspect. In each subsection, we briefly introduce the problem we studied and summarize our proposed solutions, evaluations, and key findings.

13.1.1 Coordination

Earlier, we mathematically established that: (i) coordination is necessary among the CFs, and (ii) centralized coordination is the most suitable one. Coordinating multiple intelligent Cognitive Functions (CF) using a centralized coordinator (referred to as the Controller in this thesis) is a challenging task since

- the behavior of the CFs is dynamic and cannot be predicted beforehand, and,
- while resolving a conflict, the chosen value by the Controller has to be optimal for the combined interest of all the CFs.

To solve the coordination problem, in Chapter 5, we proposed a Controller which uses an Nash Social Welfare Function (NSWF) based method to resolve conflicts among the CFs. We used NSWF since it has already been established in [45] that NSWF provides a value that is optimal for the combined interest of all. We used multiple Gaussian functions, where each function represented the behavior of one CF, to perform a numerical analysis of the proposed NSWF-based solution and proved that our proposed solution works can resolve

- any type of conflict,
- any number of simultaneously existing conflicts and
- conflicts among any number of CFs.

Although it makes the solution fit for use in real-life scenarios, this solution has a potential drawback. It assumes that all the CFs have equal priority while determining the value of a network control parameter. In reality, all network parameters are not equally important to all the CFs simultaneously. To overcome this drawback, we proposed an EGS-based solution in Chapter 6. In this solution, the importance of each network parameter on each CF is quantized, and the CFs are prioritized accordingly while determining the final value of a network parameter. To quantify the importance of a network control parameter to the functionality of a CF, we use a Shapley value based method. We performed numerical analysis between these two solutions and found an improvement of up to 7.7% using EGS over NSWF based solution. For further evaluation, we implemented the EGS based solution in the simulation environment, determined the optimal value of the shared parameter, and measured the time elapsed to resolve conflicts in different scenarios.

We made the following observations: in both the proposed approaches, a conflict is resolved in a way optimal for the combined interests of all the CFs. We found that although both the proposed solutions resolve all types of conflicts, EGS is preferable to NSWF since it provides an improvement in overall network performance. Also, an EGS based solution can be implemented such that it takes a few milliseconds to resolve any type of conflict in any possible scenario, so that the solution can be used in real-life environments.

13.1.2 Trust

In a multi-vendor scenario, different CFs can be supplied by different vendors, creating new opportunities for participation to small as well as new vendors. However, to establish itself in competition with others, a new vendor needs to advertise the superiority of its product, and such a vendor may be tempted to do so by unfair means. There remains a possibility that a vendor designs a CF that optimizes its own objective, disregarding the interests of other CFs, thereby allowing to advertise the superiority of its product. Such a CF, denoted as an MCF, deliberately sends misinformation on its preference, intending to manipulate the Controller.

Although this was a theoretical concept visualized by us, to validate this concept, in this thesis

- we experimentally demonstrated that it is possible for an MCF to learn how the Controller works and manipulate the Controller.
- the manipulation can cause severe performance degradation of the network.

In this thesis, we proposed two ways to address the problem of MCFs:

1. One option we proposed is to detect the MCF and to handle it after that (e.g., removed or replaced). We proposed a solution called MCD in Chapter 8.
2. On the other hand, for the cases in which the MCF cannot be eliminated, we proposed to neutralize the MCF without detecting it explicitly at first. We proposed a solution called CoDeRa to neutralize an MCF in Chapter 9.

In Chapter 8, we proposed MCD, a polynomial regression-based MCF detection mechanism. We also evaluated MCD to analyze how fast it can detect an MCF in an operational system. During the evaluation, we considered three different cases:

- *case 1*: MCD becomes operational before MCF.
- *case 2*: MCD and MCF become operational simultaneously.
- *case 3*: MCD becomes operational after MCF.

Out of these three cases, we found that the time taken to detect the MCF by MCD is highest for case 2. We also found that when the Controller uses EGS instead of NSWF, it is easier for MCD to catch the MCF. Apart from that, we also observed that if the MCF starts way ahead of MCD, and the difference in their start time exceeds a certain value, MCD can no longer catch the MCF. This proves the necessity of having an MCD, i.e., MCD should be used before it is too late.

Although MCD can successfully detect an MCF, taking a CF out of an operational system may affect the managed KPI and overall network performance. This is why the MCF needs to be kept active until an alternative is found. Also, taking an MCF out of a complex operational system may not always become feasible in real life. Under these circumstances, it is better to neutralize the manipulative behavior of the MCF than to take it out entirely. In Chapter 9, we proposed a solution named CoDeRa to neutralize an MCF without taking it explicitly. Earlier, we established that an MCF could manipulate the Controller as long as the MCF can successfully predict the Controller's behavior. By adding a small amount of random noises in the decisions made by the Controller, CoDeRa makes it very difficult for the MCF to learn and predict Controller outputs. Although the deliberately added noise causes some degradation of overall network performance, we experimentally determined the amount of noise that can be added without affecting the network performance severely.

From our work in this part, we concluded that even though MCD can detect an MCF, CoDeRa is more effective than MCD since CoDeRa can neutralize the manipulative behavior. However, CoDeRa also causes a certain amount of, no matter how minimal, unavoidable network performance degradation. This is why an alternative architecture, that is proposed in Appendix B, should be considered in the future. In the alternative architecture, we propose to combine the functionalities of CFs and the Controller into a single entity, thereby removing the problems discussed above.

13.1.3 Orchestration

A network operator might need to customize cell configurations during certain times for better quality of service. To customize several cell parameters and KPIs simultaneously, the operator needs complete knowledge about the dependence among the parameters and KPIs across different network states. Gaining this knowledge is a difficult and lengthy task for the operator. Since the CFs already possess this knowledge, it will be beneficial for the operator to use it. The current architecture, however, does not allow the operator to access this knowledge. Therefore, we proposed an intent-driven interface between the operator and CAN in Chapter 11. An intent is a human readable sentence. The operator can express the coveted customization to achieve in the form of a single or multiple intents, and CAN is expected to execute those intents. The proposed intent-driven interface contains three primary functional elements:

- Intent Identifier, which identifies if the intent can be executed by CAN,
- Intent Classifier, which classifies the intent based on its content, and,
- Intent Decision Maker, which generates appropriate commands to be executed by CAN.

We also implemented this intent-driven interface in the simulation environment to evaluate different metrics.

While implementing this interface, we found that the most challenging issue is the conflicts and contradictions which may reside in an intent. Although conflicts are easier to detect and resolve, contradictions, which arise during runtime, are harder to detect and resolve. In Chapter 12, we proposed an intent contradiction detector and remover (ICDR) to address that problem. We also evaluated ICDR in the simulation environment and showed that it could detect and resolve contradictions with almost 100% accuracy.

From our work in this part, we learned that with the help of an intent-driven interface, knowledge of CFs can be used to enable the MNO to customize the cell configurations. Since our proposed intent-driven interface can operate conflict-free, it is suitable for use in real-life.

13.2 Future Directions of Work

Although this thesis covers essential aspects related to open, multi-vendor network management automation and addresses the important problems, there are open issues that can be further studied. Network management automation will experience changes induced by future technologies (5G Advanced and 6G), and the concept of CAN, which evolved from SON, will be further developed. For example, current handover optimization functions like MRO may not remain relevant since future mobile networks will be highly dense and heterogeneous. Thus, most of the traffic load will be carried by small cells which, in turn, will lead to frequent handovers. Hence, conceptualization of new network functions will be necessary. Additionally, CAN also needs to operate with legacy SON functions until a complete transition to cognitive networking takes place. So, CAN, as we formulated in this thesis, will take at least a few years to be fully implemented in mobile networks. Keeping these points in mind, we highlight the scopes that can be further investigated in the future.

13.2.1 Coordination for Combined Optimization

Throughout the thesis, we implemented and studied the CFs in a single cell (the central green cell in Fig. 3.4). However, CFs like MLB or MRO also have an impact on the neighboring cells. When the network load in a cell is high, MLB tries to offload some UEs to the neighboring cells by reducing the CIO value. Let us explain with an example how it works. In the left side of Fig. 13.1, we showed two cells: the source cell and the target cell, in two different colors. When a UE moves from the source cell to the target cell (e.g., from point A to point B), somewhere near point C, the handover takes place. If the handover margin and the CIO of the target cell are not changed, the source cell can offload the network load quicker by lowering its CIO value. That is precisely what

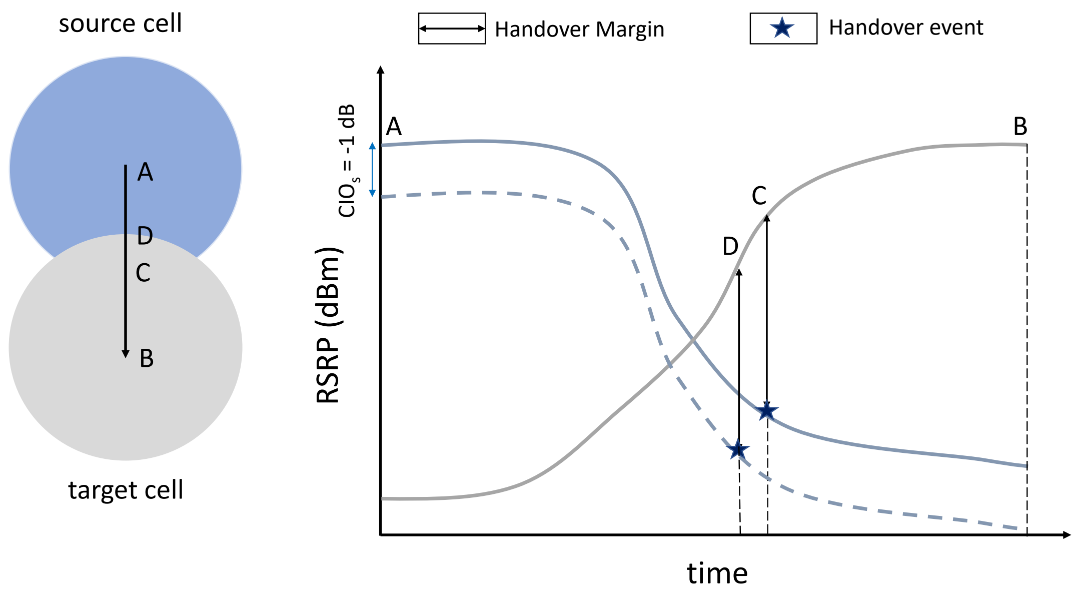


Figure 13.1: Effect of CIO on Handover

MLB of the source cell (in our case, the central green cell in Fig. 3.4) does. In Fig. 13.1, we illustrate how the handover can be done faster by lowering the CIO value. Now, if in both neighboring cells, the MLBs start lowering their respective CIO values, the handover margin will be challenging to achieve, and the handover events will be disturbed. Not only by MLB, but handover events can also be affected by CCO. To increase the average user downlink throughput, CCO can reduce the cell transmit power so that cell coverage area decreases and fewer UEs are served. If the coverage area of both neighboring cells keeps decreasing, there will be no overlap between two neighboring cells, and many UEs will not be served at all.

This is why we see that, along with coordinating different CFs in the same cell, it is also important to have coordination among different cells simultaneously. The coordination among cells can possibly be done by coordinating not only the CFs in the source cell, but also taking the CFs of the neighboring cells into account. The advantage of our proposed solutions (NSWF or EGS) is that both are scalable, so any number of CFs from the neighboring cells can be included in the coordination process. The challenge, however, in that case, is to calculate the utility function of a CF from a neighboring cell.

13.2.2 Other Application Areas

Although the thesis has been written keeping the scenario of network management automation in mind, the proposed ideas can be applied to other fields as well. As already discussed earlier in Section 4.1, CAN can be visualized as a MAS with specific properties. The problem statements and solutions, covered in this thesis, are valid for any MAS with the properties mentioned in Section 4.1. So, it would be interesting to find other research areas that, when abstracted, resemble CAN and see if the problem statements and our proposed solutions are applicable or how much they need to be adjusted.

13.2.3 Machine Readable Intent Translation

While evaluating the intent-driven solutions proposed in this thesis, we always assumed that an MNO provides an intent in a particular format (denoted by *formal intent*). This enables our proposed intent-driven solution to parse the intent quickly and act accordingly. However, to make the intent input from the MNO side much more accessible, the

MNO should have the flexibility to input the intent in any format the MNO wants. The intent-driven system should have additional capabilities to translate any intent and act accordingly. This is an advanced feature that should be incorporated into the future intent-driven management solutions. Advanced deep learning based parsers (e.g., LSTM, Transformer) may be used for this purpose, although it needs further detailed study and evaluation.

List of Abbreviations Used in This Chapter

CF	Cognitive Functions
CoDeRa	Controller Decision Randomizer
EGS	Eisenberg-Gale Solution
MAS	multi-agent system
MCD	Manipulative CF Detector
MCF	Manipulative Cognitive Function
NSWF	Nash Social Welfare Function

Appendices

Appendix A. Effectiveness of Shapley value based of CW calculation

Here we establish the effectiveness of the Shapley value based CW value calculation proposed in Section 6.4. We prove the effectiveness both mathematically and by simulation.

Cross validation via Mathematics

In this section, we prove mathematically that our proposed definition of calculating CW values, described in Section 6.4, is the best one. To prove so, we take the UFs defined in Eq. 6.8 and Eq. 6.9, and maximize

$$v_1^{w_1^{'}} v_2^{w_2^{'}} - v_1 v_2 \tag{13.1}$$

$$\text{where} \quad v_1 = e^{\frac{-(p_1+50)^2}{2p_2^2}}, \tag{13.2}$$

$$v_2 = e^{\frac{-(p_1-50)^2}{2p_3^2}}, \tag{13.3}$$

p_1, $w_1^{'}$, $w_2^{'}$ vary simultaneously, $0 < w_1^{'}, w_2^{'} < 1$, $w_1^{'} + w_2^{'}$=1, $p_2 = 60$ and $p_3 = 20$.

Putting $p_2^2/p_3^2 = 9$ further simplifies Eq. 13.1 to

$$e^{-\frac{1}{9p_3}}\left[e^{g_1} - e^{g_2}\right] \tag{13.4}$$

where

$$g_1 = w_1^{'}(p_1+50)^2 + 9w_2^{'}(p_1-50)^2 \tag{13.5}$$

$$g_2 = (p_1+50)^2 + 9(p_1-50)^2 \tag{13.6}$$

If we consider g_1, we know that, by definition of CW, $w_1^{'} > 0$ and $w_2^{'} > 0$. If p_1 = -50, $w_1^{'}(p_1+50)^2$ becomes 0 but $9w_2^{'}(p_1-50)^2$ becomes > 0. Similarly, if p_1 = 50, $9w_2^{'}(p_1-50)^2$ becomes 0 but $w_1^{'}(p_1+50)^2$ becomes > 0. So, we see that, for all possible values of p_1, $w_1^{'}$ and $w_2^{'}$, $g_1 > 0$. Following the same logic, we can also conclude that $g_2 > 0$.

As, both $g_1, g_2 > 0$, Eq. 13.4 is maximized when

$$F = g_1 - g_2 \tag{13.7}$$

is maximized.

After putting the values of g_1, g_2 in Eq. 13.7 and simplifying, we get the final expression

$$F = w_1^{'}(p_1+50)^2 - (p_1+50)^2 - 9w_1^{'}(p_1-50)^2 \tag{13.8}$$

which needs to be maximized.

Theorem. (**Second Partials Test**) If $z = f(x, y)$ has a critical point at (x_0, y_0), so $\frac{\partial f}{\partial x}|_{(x_0,y_0)}$ = 0 and $\frac{\partial f}{\partial y}|_{(x_0,y_0)} = 0$. Let

$$\mathrm{D} = \frac{\partial^2 f}{\partial x^2}|_{(x_0,y_0)} \cdot \frac{\partial^2 f}{\partial y^2}|_{(x_0,y_0)} - \big(\frac{\partial^2 f}{\partial x \partial y}|_{(x_0,y_0)}\big)^2 \tag{13.9}$$

(1) If D > 0 and $\frac{\partial^2 f}{\partial x^2}|_{(x_0,y_0)} > 0$, then ($x_0$, y_0) is a local minimum.
(2) If D > 0 and $\frac{\partial^2 f}{\partial x^2}|_{(x_0,y_0)} < 0$, then ($x_0$, y_0) is a local maximum.
(3) If D < 0, then (x_0, y_0) is a saddle point (neither maximum nor minimum).
(4) In all other cases, no conclusion can be drawn.

From Eq. 13.8 we see that F has two variables: $w_1^{'}$ and p_1. So, F is maximized when both $\frac{\partial F}{\partial p_1} = 0$ and $\frac{\partial F}{\partial w_1^{'}} = 0$ hold simultaneously. From Eq. 13.8, we get

$$\frac{\partial F}{\partial p_1} = 0 \implies p_1^2 - 125p_1 + 2500 = 0$$

Solving which, we get $p_1 = 25$ or 100.

Again, from Eq. 13.8, we get

$$\frac{\partial f}{\partial p_1} = 0 \implies w_1^{'} = \frac{2p_1 + 100}{1000 - 16p_1} \tag{13.10}$$

In Eq. 13.10, putting the values of $p_1 = 25$ and 100, we get $w_1^{'} = 0.25$ and -0.5 respectively. As $w_1^{'} > 0$, only critical point for F is (p_1, $w_1^{'}$) = (25, 0.25).

However, (p_1, $w_1^{'}$) = (25, 0.25) is only a critical point for F, i.e., F can be either maximum or minimum at this point. For F to be maximum at this point, condition (2) from the Theorem has to hold true. As $\frac{\partial^2 F}{\partial w_1^{'2}} = 0$, $\frac{\partial^2 F}{\partial p_1^2}$ = -(16 w_1 + 2), $\frac{\partial^2 F}{\partial p_1 \partial w_1^{'}}$ = 1000 - 16 p_1, from Eq. 13.9 we see that D = 16 p_1 - 1000 = 600 > 0, thus F is maximum for $w_1^{'} = 0.25$.

Now, when calculated using our proposed Shapley value based method in Section 6.5.1, we got the value of $w_1^{'}$ as 0.25. In this Section, we mathematically prove that optimal value for $w_1^{'}$ is 0.25. This shows that our proposed Shapley value based CW calculation method is optimal.

Cross-validation via Simulation

To prove the effectiveness of our proposed method of calculating CW values (Eq. 6.7), we also cross-validate the results using simulation. We vary $w_1^{'}$ and $w_2^{'}$ in between 0 and 1 simultaneously in steps of 0.01 while ensuring that $w_1^{'} + w_2^{'}$ is always 1. For each combination of ($w_1^{'}$, $w_2^{'}$), we calculate the optimal p_1 value and correspondingly find the overall improvement (sum of the improvement in individual utility) of using EGS over NSWF. From Fig. 13.2 we find that the overall system utility becomes maximum when ($w_1^{'}$, $w_2^{'}$) = (0.25, 0.75), which is in par with our earlier result and proves again that our CW calculation method is the best.

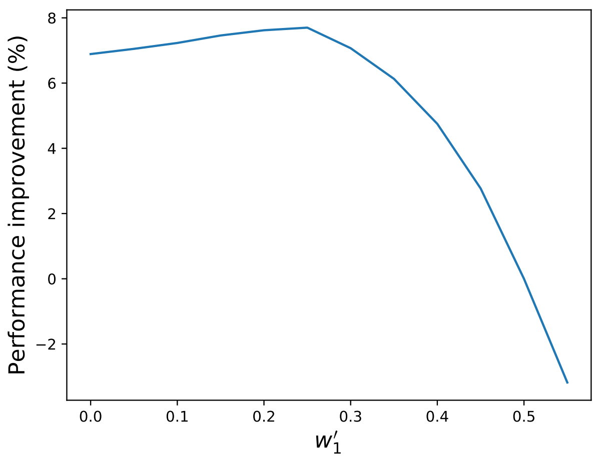


Figure 13.2: Overall performance improvement of the system for varied $w_1^{'}$

Appendix B. Alternate Mobile Network Management Architecture

Motivation

The mobile network is a complex system bound to produce erroneous measurements or sometimes completely fail to report. CFs are usually trained on complete and accurate datasets, so they may not be robust against such data corruption. In [105] researchers showed that a single corrupted input to an ML model, which is part of an ML model chain, can generate multiple corrupted outputs with each step in the chain multiplying the effect. In networks, multiple CFs often share the same input control parameter (ICP), implying that missing or corrupted data for the ICP will affects the learning of all the CFs. With the UFs of CFs used by the Controller, the final value computed by the Controller may be sub-optimal due to the inaccurate UFs, and thus, there is likely to be major degradation in all the KPIs influenced by the ICP. Naturally, the more the CFs that share an ICP with corrupted or missing data, the more is the likelihood of large deviations of final ICP value from the optimal value and the more likely is the network performance degradation. Fig. 13.3 illustrates the error propagation: a single error (say, corrupted value) in an ICP, goes to all the CFs affecting their learning and resulting in erroneous UFs produced by the CFs (shown by a different color in Fig. 13.3). When the Controller uses multiple of these erroneous UFs, its decision is distorted, so the changes the Controller makes in the network lead to network degradation.

Since real-life systems do encounter corrupted or missing data, this degradation is more or less guaranteed, caused by the multiplication of effects of the erroneous measurement across multiple CFs and the Controller in current network management automation (NMA) architecture. The multiplication effect can be reduced if the number of entities propagating the error is low, so this motivates the need for an alternative architecture with a lower number of entities.

Alternative architecture

A single ML Model

We hypothesize that it is better to replace all the CFs and the Controller with a single ML Model, specifically a neural network (NN) as shown in Fig. 13.4. The NN takes the combination of network state, network control parameters and KPIs as input, and, it is expected to change the network control parameters appropriately such that all the KPIs in the input network state are optimized. This architecture has a scalability limitation since the CFs, which are to be concurrently operated, must be tested and deployed together. It however, addresses the identified shortcomings, such as:

1. *Trust issue*: a single NN completely destroys the idea of an MCF, since all CFs are merged into one. This, in effect, removes all the trust-related problems (of Chapter 7)

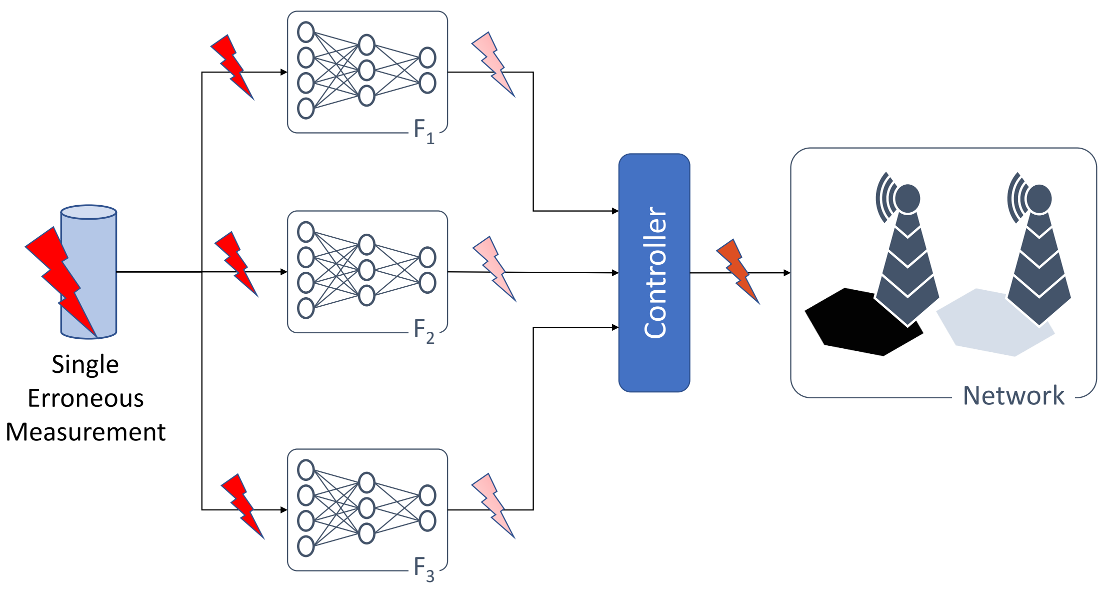


Figure 13.3: Error propagation in current NMA architecture

and the shortcomings arising from having MCF, MCD and CoDeRa in the system since all these functions are not needed anymore.

2. *Performance issue*: Data corruption in one control parameter is captured only once in the computation of the final value, so multiplication effect does not occur anymore.

3. *simplicity*: The reduction to only one ML model makes results onto a simpler implementation, avoiding the information exchange among modules.

As an example implementation of our proposed architecture, we use a fully connected NN with 5 hidden layers and 50 nodes in each hidden layer using ReLU as activation function in each node, Adam as optimizer and MSE as loss function. Inputs and outputs of the NN are as shown in Fig. 13.4. Network state consists of the combination of TXP, RET, timestamp, connected UEs, antenna height and antenna gain. For a first demonstration, we use the default model but we later also vary the number of hidden layers and nodes in each layer to find the optimal NN architecture. The NN is always trained with a batch size of 64 over 50 epochs.

Demonstration of expected benefit

To experimentally demonstrate the problem depicted in Fig. 13.3 and the advantages of our proposed solution, we wish to compare our proposed architecture of a single NN (hereafter referred as Model 1) with a current architecture with 2 CFs: MLB, MRO (hereafter referred as Model 2), so we only take the control parameters and KPIs into consideration which are relevant for MLB and MRO, i.e., control parameters: TTT, CIO and KPIs: network load, late handovers (LHO), early handovers (EHO) and ping-pong handovers (PPHO). We consider the same simulation setup but different data scenarios for training the models before deploying them in the simulator to observe changes in KPI values. specifically, we consider the following scenarios:

- *scenario 1*: the dataset is complete and error free.
- *scenario 2*: the dataset is incomplete (has some missing datapoints), but all the datapoints are accurate.

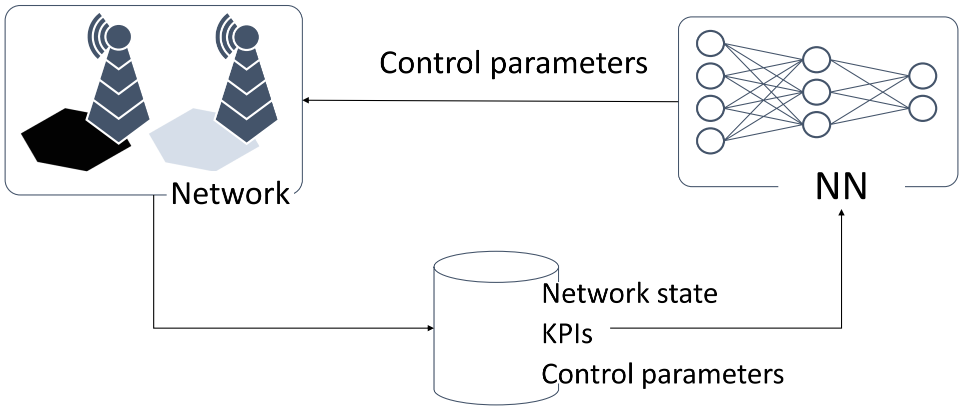


Figure 13.4: Proposed NMA architecture

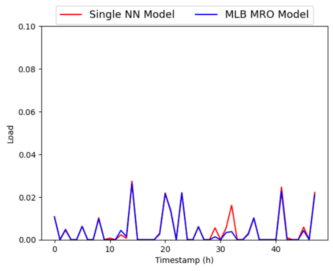


Figure 13.5: Load in Scenario 1

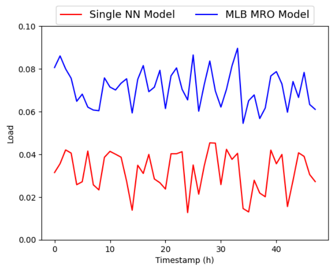


Figure 13.6: Load in Scenario 2

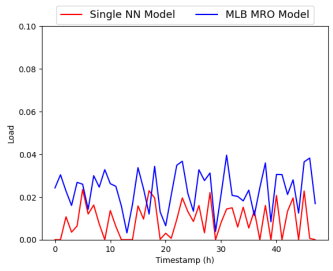


Figure 13.7: Load in Scenario 3

- *scenario 3*: the dataset is complete but it has some erroneous datapoints.
- *scenario 4*: the dataset is incomplete (missing data points) and some existing datapoints are erroneous.

Since scenario 4 is a combination of scenarios 2 and 3, we argue that if Model 1 outperforms Model 2 in both scenarios 2 and 3, the same will hold scenario 4 as well.

When the dataset is complete and error free (Scenario 1), we see from Fig. 13.5 and Fig. 13.8 that both Model 1 and Model 2 perform equivalently. However, both in scenario 2 and scenario 3, using Model 1 we get lesser load values and better handover performance, i.e., Model 1 outperforms Model 2 in both KPIs. This degradation in the performance of Model 2 happens because of the error propagation through current architecture, that we explained earlier, and it proves the necessity of an alternate architecture.

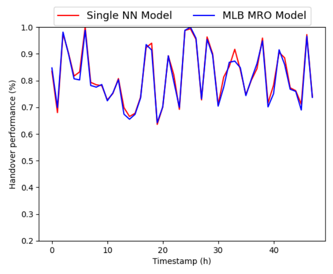


Figure 13.8: Scenario 1 Handover performance

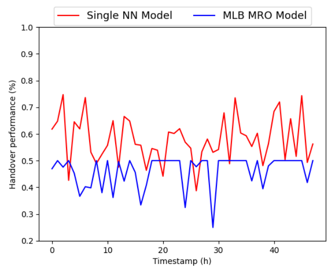


Figure 13.9: Scenario 2 Handover performance

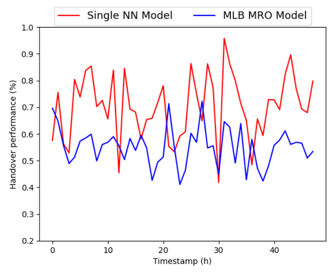


Figure 13.10: Scenario 3 Handover performance

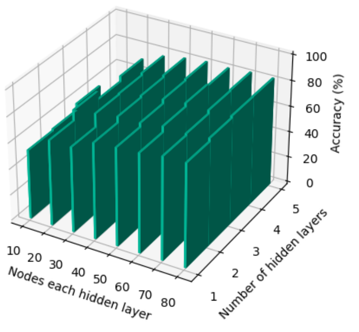


Figure 13.11: error margin 0.01

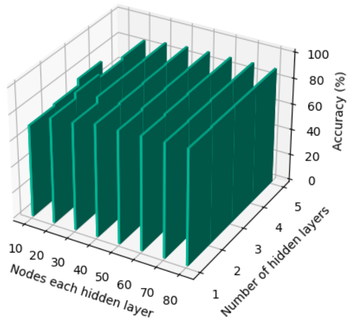


Figure 13.12: error margin 0.02

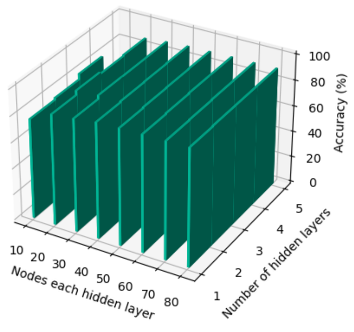


Figure 13.13: error margin 0.025

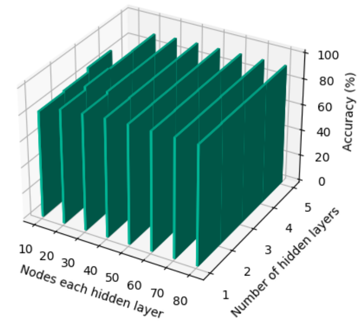


Figure 13.14: error margin 0.03

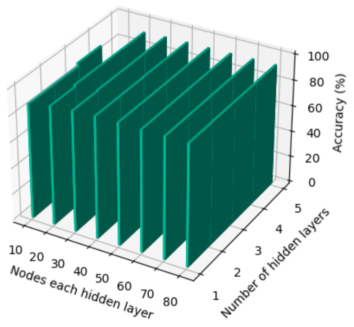


Figure 13.15: error margin 0.04

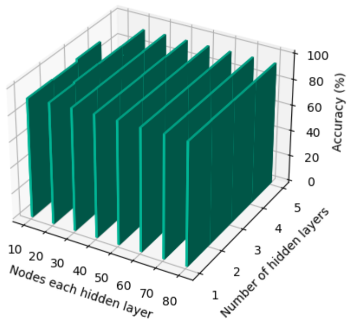


Figure 13.16: error margin 0.05

Evaluation of the proposed architecture

Since we already showed that when the dataset is complete and error free both the current architecture and our proposed architecture perform equivalently, we evaluate the accuracy of our proposed architecture (a single NN) against the Controller choices over different NN architectures and hyperparameters. We found experimentally that for NN predictions, a deviation of up to 5% does not affect the overall network performance. Correspondingly, unless otherwise stated, we allow for an error margin of up to 5% and consider the NN prediction to be a success, if, for a controller value x, the NN prediction is within 5% of x ($(1 \pm 0.05)x$).

NN accuracy for different error margins

Although we found that an error margin up to 5% can be allowed, we wish to consider the effect of the error margin while determining an optimal architecture. For each margin, we vary the number of hidden layers and nodes in each layer with the accuracy result plotted in Fig. 13.11 - Fig. 13.16. Specifically, for an error margin of 5%, we see that the accuracy is highest with 2 or more hidden layers and 20 or more nodes in each layer. However, as the error margin is lowered the accuracy of each specific NN model instance reduces. In general however, a NN with 5 hidden layers and 40 nodes in each layer performs well in all cases and increasing beyond this does not provide any better accuracy.

NN accuracy for different network states

The ML model uses a combination of the network control parameters, listed in Table 13.1, as the network state and we see that the more parameters we include in the network state, the better is the NN accuracy. This is because every extra parameter provides a better insight on the state of the network allowing for better learning of the NN.

Table 13.1: Efficiency of the NN depending on the parameters in network state

TXP	RET	timestamp	connected UEs	antenna height	antenna gain	NN efficiency (%)
✓	✓					91.02
✓	✓	✓				92.59
✓	✓		✓			92.67
✓	✓	✓	✓			92.77
✓	✓	✓	✓	✓		93.72
✓	✓	✓	✓		✓	93.83
✓	✓	✓	✓	✓	✓	95.21

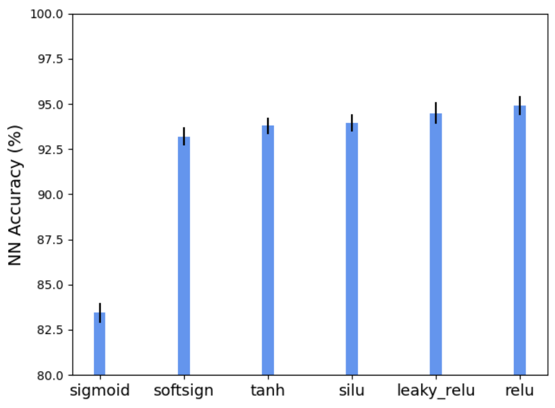


Figure 13.17: Accuracy of the NN for different activation functions

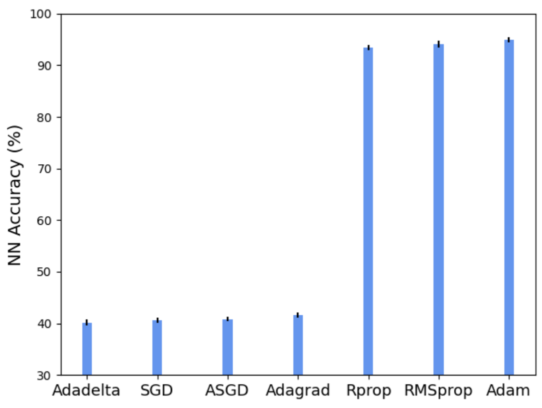


Figure 13.18: Accuracy of the NN for different optimizers

NN accuracy for different hyperparameters

We also change other hyperparameters of the NN, like optimizer, activation function and weight decay, and evaluate the performance of the NN. We see in Fig. 13.17 that using the activation function ReLU gives the highest accuracy but other activation functions like Leaky ReL and Tanh also provide almost similar accuracy. Fig. 13.18 shows that Adam is the most suitable optimizer here but Rprop and RMSprop also perform quite well. Finally, Fig. 13.19 investigates the impact of weight decay shows that it is better to not use any weight decay.

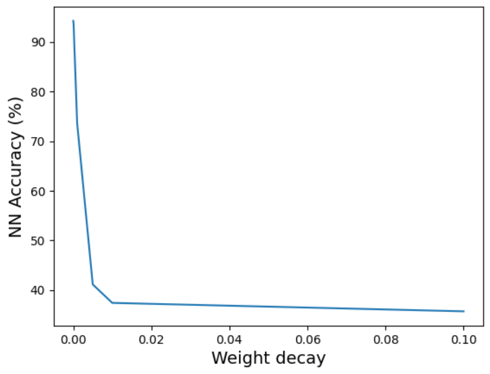


Figure 13.19: Accuracy of the NN for different weight decays

Table 13.2: Time (s) required to train different NNs

		Number of Hidden Layers				
		1	2	3	4	5
Number of nodes in each hidden layer	10	11.8028	13.3601	15.3602	17.7529	19.5792
	20	11.8785	13.7531	15.8903	18.1351	19.9858
	30	11.9136	14.3287	16.1308	19.8758	21.2116
	40	11.9672	16.1902	17.2016	20.4782	27.2301
	50	12.0296	17.0586	18.0273	22.1835	30.2202
	60	12.0857	18.0924	20.8091	24.2016	32.1022
	70	12.1205	19.1934	22.2585	26.3187	33.6997
	80	12.1801	20.7132	24.2271	28.8041	35.8526

Complexity analysis of the proposed architecture

We also perform the complexity analysis of our proposed architecture to find the trade-off between the performance and computational efficiency. We run the simulations and train the neural networks in an Quadcore Intel Core i5-10310U@1.7 GHz CPU. Time taken (in seconds) to train each NN is given in Table 13.2. From this Table we see that on an average it takes less than a half a minute to train the NN and the maximum time difference in training any two types of NNs is 24 seconds, which makes it fit to use in real life system. We also see that the amount of RAM needed to train the NN varies between 4.01 MB and 4.14 MB, i.e., our proposed solution can be implemented using any modern computer.

Standardization impact

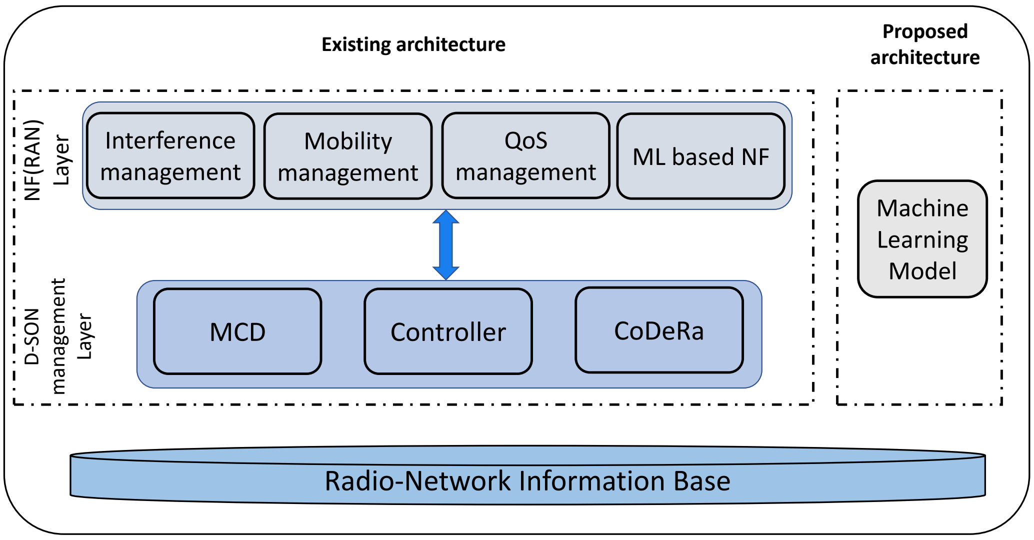


Figure 13.20: 3GPP D-SON and proposed architecture

3GPP defines distributed SON (D-SON) when the SON algorithm is located in the network function (NF) layer [82]. For D-SON, the NFs (like, QoS management, mobility management) monitors the network events, analyses the network data, makes decision on SON actions and executes those actions (shown in Fig. 13.20). The advancements in ML and DL in recent times are driving vendors to produce ML based NFs. For example, if we consider the specifications of MRO in 3GPP [83], we see that the functionalities, like, analyzing reports from UEs and network slice information, mitigating HO issues by adjusting HO related parameters (TTT,CIO), can be better achieved by using ML and DL capabilities. The deployment of multiple learning-based NFs (i.e. CFs) in a multi-vendor environment directly runs into the trust and performance issues discussed earlier. For that reason, as a minimum, the SON management layer shall need to be adapted to account for the likelihood of manipulation (MCD, CoDeRa) as shown in Fig. 13.20. However, if we follow the alternate architecture proposed here, we do not need separate D-SON management and NF layers. The functionalities of both these layers can be combined into a single model in a single layer as shown in Fig. 13.20. This makes the architecture less complex and easy to implement.

List of Abbreviations Used in This Chapter

EHO early handovers	***NN*** neural network
ICP input control parameter	***PPHO*** ping-pong handovers
LHO late handovers	***RET*** remote electrical tilt
MSE mean squared error	***TXP*** transmit power
NMA network management automation	***UEs*** user equipments

Literature

[1] S. Hämäläinen, H. Sanneck, and C. Sartori. *LTE self-organising networks (SON): network management automation for operational efficiency.* John Wiley & Sons, 2012.

[2] William Stallings. *Wireless communications & networks.* Pearson Education India, 2009.

[3] T. Ali-Yahiya. *Fractional Frequency Reuse in LTE Networks*, pages 199–210. Springer New York, 2011.

[4] 3gpp. [Online]. Available: https://www.3gpp.org/.

[5] 3GPP. Evolved universal terrestrial radio access network (e-utran); self-con

guring and self-optimizing network (son) use cases and solutions. Technical Report 36.902 version 9.3.1 Release 9, May, 2011.

[6] A. Alnoman and A. Anpalagan. Towards the fulfillment of 5g network requirements: technologies and challenges. *Telecommunication Systems*, 65(1):101–116, 2017.

[7] Stephen S Mwanje and Christian Mannweiler. Towards cognitive autonomous networks in 5g. In *2018 ITU Kaleidoscope: Machine Learning for a 5G Future (ITU K)*, pages 1–8. IEEE, 2018.

[8] O-RAN ALLIANCE. O-ran whitepaper - building the next generation ran. [Online]. Available: https://www.o-ran.org/resources, Oct, 2018.

[9] K. M. Addali, S. Y. B. Melhem, Y. Khamayseh, Z. Zhang, and M. Kadoch. Dynamic mobility load balancing for 5g small-cell networks based on utility functions. *IEEE Access*, 7:126998–127011, 2019.

[10] ETSI. Zero touch network and service management (zsm). [Online]. Available: https://www.etsi.org/technologies/zero-touch-network-service-management.

[11] L. S. Shapley. A value for n-person games. *Contributions to the Theory of Games*, 2(28):307–317, 1953.

[12] ETSI GR ZSM-011. Intent-driven autonomous networks; generic aspects. [Online]. Available: https://portal.etsi.org/webapp/WorkProgram/Report_WorkItem.asp?WKI_ID=61992, 2022.

[13] 3GPP. Study on enhanced intent driven management services for mobile networks. [Online]. Available: https://www.3gpp.org/ftp/Specs/archive/28_series/28.912/, 2022.

[14] A. Banerjee, S. S. Mwanje, and G. Carle. Game theoretic conflict resolution mechanism in cognitive autonomous networks. In *2020 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS)*, pages 1–8. IEEE, 2020.

[15] S. S. Mwanje, M. Kajó, S. Majumdar, and G. Carle. Environment modeling and abstraction of network states for cognitive functions. In *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*, pages 1–8. IEEE, 2020.

[16] A. Alnoman and A. Anpalagan. Towards the fulfillment of 5g network requirements: technologies and challenges. *Telecommunication Systems*, 65(1):101–116, 2017.

[17] R. M. Dawes. Social dilemmas. *Annual review of psychology*, 31(1):169–193, 1980.

[18] A. Banerjee, S. S. Mwanje, and G. Carle. On the implementation of cognitive autonomous networks. *Wiley Internet Technology Letters*, 4(6):e317, 2021.

[19] P. Agyapong, V. Braun, et al. Deliverable d6. 1-simulation guidelines. 2013.

[20] J. Meinilä, P. Kyösti, L. Hentilä, T. Jämsä, E. E. Suikkanen, and M. Narandzic. Wireless world initiative new radio winner+, d5. 3: Winner+ final channel models. *CELTIC Telecommunication Soultions, Tech. Rep*, 2010.

[21] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *Wireless communications and mobile computing*, 2(5):483–502, 2002.

[22] A. Banerjee, S. S. Mwanje, and G. Carle. Optimal configuration determination in cognitive autonomous networks. In *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 494–500. IEEE, 2021.

[23] S Sarika and V. Paul. Agenttab: An agent based approach to detect tabnabbing attack. *Procedia Computer Science*, 46:574–581, 2015.

[24] M. Khayyat and A. Awasthi. An intelligent multi-agent based model for collaborative logistics systems. *Transp. Res. Procedia*, 12:325–338, 2016.

[25] F. Rahimzadeh, L. Khanli, and F. Mahan. High reliable and efficient task allocation in networked multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 29(6):1023–1040, 2015.

[26] A. Banerjee, N. Sastry, and C. M. Machuca. Sharing content at the edge of the network using game theoretic centrality. In *2019 21st International Conference on Transparent Optical Networks (ICTON)*, pages 1–4. IEEE, 2019.

[27] JP Mano and P Glize. Self-adaptive network of cooperative neuro-agents. In *AISB'04 Symposium on Adaptive Multi-Agent Systems*, 2004.

[28] M. Gatti, P. Cavalin, S. Neto, C. Pinhanez, C. dos Santos, D. Gribel, and A. Appel. Large-scale multi-agent-based modeling and simulation of microblogging-based online social network. In *International Workshop on Multi-Agent Systems and Agent-Based Simulation*, pages 17–33. Springer, 2013.

[29] SS Manvi and MS Kakkasageri. Multicast routing in mobile ad hoc networks by using a multiagent system. *Information Sciences*, 178(6):1611–1628, 2008.

[30] T. Morstyn, B. Hredzak, and V. Agelidis. Cooperative multi-agent control of heterogeneous storage devices distributed in a dc microgrid. *IEEE Transactions on Power Systems*, 31(4):2974–2986, 2015.

[31] R. Bianchi, M. Martins, C. Ribeiro, and A. Costa. Heuristically-accelerated multiagent reinforcement learning. *IEEE transactions on cybernetics*, 44(2), 2013.

[32] S. Resmerita and M. Heymann. Conflict resolution in multi-agent systems. In *42nd IEEE International Conference on Decision and Control (IEEE Cat. No. 03CH37475)*, volume 3, pages 2537–2542. IEEE, 2003.

[33] E. Semsar-Kazerooni and K. Khorasani. A game theory approach to multi-agent team cooperation. In *2009 American Control Conference*, pages 4512–4518. IEEE, 2009.

[34] J.K. Gupta, M. Egorov, and M. Kochenderfer. Cooperative multi-agent control using deep reinforcement learning. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 66–83. Springer, 2017.

[35] H. Liu, P. Zhang, B. Hu, and P. Moore. A novel approach to task assignment in a cooperative multi-agent design system. *Applied Intelligence*, 43(1):162–175, 2015.

[36] Q. Liu, X. Cui, and X. Hu. Conflict resolution within multi-agent system in collaborative design. In *International Conference on Computer Science and Software Engineering.* IEEE, 2008.

[37] M.R. Genesereth, M.L. Ginsberg, and J.S. Rosenschein. Cooperation without communication. In *Readings in distributed artificial Intelligence*, pages 220–226. Elsevier, 1988.

[38] D. Kominami, M. Sugano, M. Murata, and T. Hatauchi. Controlled and self-organized routingfor large-scale wireless sensor networks. *ACM Transactions on Sensor Netw. 10*, 1(13):1–27, 2013.

[39] M. Lamba and M. S. Dagar. Review on self organizing networks (son) in lte-advanced hetnets.

[40] T. Bandh, H. Sanneck, and R. Romeikat. An experimental system for son function coordination. In *2011 IEEE 73rd Vehicular Technology Conference (VTC Spring)*, pages 1–2. IEEE.

[41] SOCRATES Deliverable. D2. 1: Use cases for self-organising networks. *EU STREP SOCRATES (INFSO-ICT-216284), Version*, 1, 2008.

[42] FP7 SOCRATES INFSO-ICT216284. Final report on self-organisation and its implications in wireless access networks. *D5*, 9:v1, 2010.

[43] Y. Chevaleyre, P. E. Dunne, U. Endriss, J. Lang, M. Lemaitre, N. Maudet, J. Padget, S. Phelps, J. Rodrígues-Aguilar, and P. Sousa. Issues in multiagent resource allocation. *Issues in multiagent resource allocation.*

[44] Sara Ramezani and Ulle Endriss. Nash social welfare in multiagent resource allocation. In *Agent-mediated electronic commerce. Designing trading strategies and mechanisms for electronic markets*, pages 117–131. Springer, 2009.

[45] Eric Van Damme. The nash bargaining solutions is optimal. *Journal of Economic Theory*, 38(78):100, 1986.

[46] Mamoru Kaneko and Kenjiro Nakamura. The nash social welfare function. *Econometrica: Journal of the Econometric Society*, pages 423–435, 1979.

[47] I. Marsh, B. Grönvall, and F. Hammer. The design and implementation of a quality-based handover trigger. In *International Conference on Research in Networking*. Springer, 2006.

[48] A. Banerjee, S. S. Mwanje, and G. Carle. Towards control and coordination in cognitive autonomous networks. *IEEE Transactions on Network and Service Management (TNSM)*, 19(1):49–60, 2022.

[49] W. C. Brainard, H. E. Scarf, et al. *How to compute equilibrium prices in 1891*. Citeseer, 2000.

[50] S. Brânzei, Y. Chen, X. Deng, A. Filos-Ratsikas, S. K. S. Frederiksen, and J. Zhang. The fisher market game: equilibrium and welfare. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[51] E. Eisenberg and D. Gale. Consensus of subjective probabilities: The pari-mutuel method. *The Annals of Mathematical Statistics*, 30(1):165–168, 1959.

[52] I. M. Sobol. Sensitivity analysis for non-linear mathematical models. *Mathematical modelling and computational experiment*, 1:407–414, 1993.

[53] B. Broto, F. Bachoc, M. Depecker, and J. Martinez. Sensitivity indices for independent groups of variables. *Mathematics and Computers in Simulation*, 163:19–31, 2019.

[54] A. Banerjee, S. S. Mwanje, and G. Carle. On detection of manipulative cognitive functions in cognitive autonomous networks. In *2021 17th International Conference on Network and Service Management (CNSM)*, pages 194–200. IEEE, 2021.

[55] B. Ross, L. Pilz, B. Cabrera, F. Brachten, G. Neubaum, and S. Stieglitz. Are social bots a real threat? an agent-based model of the spiral of silence to analyse the impact of manipulative actors in social networks. *European Journal of Information Systems*, 28(4):394–412, 2019.

[56] X. Song, W. Jiang, X. Liu, H. Lu, Z. Tian, and X. Du. A survey of game theory as applied to social networks. *Tsinghua Science and Technology*, 25(6):734–742, 2020.

[57] K. Koorehdavoudi, S. Roy, and M. Xue. Distributed decision-making algorithms with multiple manipulative actors: A feedback-control perspective. In *2018 IEEE conference on decision and control (CDC)*, pages 4439–4444. IEEE, 2018.

[58] W. A. Wulf, C. Wang, and D. Kienzle. A new model of security for distributed systems. In *Proceedings of the 1996 workshop on New security paradigms*, pages 34–43, 1996.

[59] T. A. Wettergren. The destabilizing impact of non-performers in multi-agent groups. In *Recent Trends in Naval Engineering Research*, pages 257–276. Springer, 2021.

[60] R. K. Jyoti, M. K. Malhotra, and D. Ghose. Rogue agent identification and collision avoidance in formation flights using potential fields. In *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1080–1088. IEEE, 2021.

[61] P. Valianti, S. Papaioannou, P. Kolios, and G. Ellinas. Multi-agent coordinated close-in jamming for disabling a rogue drone. *IEEE Transactions on Mobile Computing*, 2021.

[62] P. Valianti, S. Papaioannou, P. Kolios, and G. Ellinas. Multi-agent coordinated interception of multiple rogue drones. In *GLOBECOM 2020-2020 IEEE Global Communications Conference*, pages 1–6. IEEE, 2020.

[63] E. A. M. Ghalamzan, F. Abi-Farraj, P. R. Giordano, and R. Stolkin. Human-in-the-loop optimisation: mixed initiative grasping for optimally facilitating post-grasp manipulative actions. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3386–3393. IEEE, 2017.

[64] L. J. Mirman, L. Samuelson, and A. Urbano. Duopoly signal jamming. *Economic Theory*, 3(1):129–149, 1993.

[65] T. D. Jeitschko, T. Liu, and T. Wang. Information acquisition, signaling and learning in duopoly. *International Journal of Industrial Organization*, 61:155–191, 2018.

[66] R. N. Clarke. Collusion and the incentives for information sharing. *The Bell Journal of Economics*, pages 383–394, 1983.

[67] D. Fried. Incentives for information production and disclosure in a duopolistic environment. *The Quarterly Journal of Economics*, 99(2):367–381, 1984.

[68] E. Gal-Or. Information sharing in oligopoly. *Econometrica: Journal of the Econometric Society*, pages 329–343, 1985.

[69] G. J. Mailath. An abstract two-period game with simultaneous signaling—existence of separating equilibria. *Journal of Economic Theory*, 46(2):373–394, 1988.

[70] E. Gal-Or. Multiprincipal agency relationships as implied by product market competition. *Journal of Economics & Management Strategy*, 6(1):235–256, 1997.

[71] M. Katz. A welfare analysis of cost information sharing among oligopolists: signaling versus direct exchange. Discussion paper 98, Princeton University, 1985.

[72] D. Fudenberg and J. Tirole. A" signal-jamming" theory of predation. *The RAND Journal of Economics*, pages 366–376, 1986.

[73] R. Gibbons. Incentives and careers in organizations. 1996.

[74] B. Holmström. Managerial incentive problems: A dynamic perspective. *The review of Economic studies*, 66(1):169–182, 1999.

[75] A. S. Kyle. Continuous auctions and insider trading. *Econometrica: Journal of the Econometric Society*, pages 1315–1335, 1985.

[76] S. A. Matthews and L. J. Mirman. Equilibrium limit pricing: The effects of private information and stochastic demand. *Econometrica: Journal of the Econometric Society*, pages 981–996, 1983.

[77] M. H. Riordan. Imperfect information and dynamic conjectural variations. *The RAND Journal of Economics*, pages 41–50, 1985.

[78] C. Benzaid and T. Taleb. Ai-driven zero touch network and service management in 5g and beyond: Challenges and research directions. *IEEE Network*, 34(2):186–194, 2020.

[79] R. Boutaba, N. Shahriar, M. A. Salahuddin, S. R. Chowdhury, N. Saha, and A. James. Ai-driven closed-loop automation in 5g and beyond mobile networks. In *Proceedings of the 4th FlexNets Workshop on Flexible Networks Artificial Intelligence Supported Network Flexibility and Agility*, pages 1–6, 2021.

[80] T. Maksymyuk, J. Gazda, M. Ružička, E. Slapak, G. Bugar, and L. Han. Deep learning based mobile network management for 5g and beyond. In *2020 IEEE 15th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET)*, pages 890–893. IEEE, 2020.

[81] X. Yuan, H. Yao, J. Wang, T. Mai, and M. Guizani. Artificial intelligence empowered qos-oriented network association for next-generation mobile networks. *IEEE Transactions on Cognitive Communications and Networking*, 7(3):856–870, 2021.

[82] 3GPP. Telecommunication management; self-organizing networks (son); concepts and requirements. Technical Specification (TS) 32.500, 3rd Generation Partnership Project (3GPP), 2020.

[83] 3GPP. Management and orchestration; self-organizing networks (son) for 5g networks. Technical Specification (TS) 28.313, 3rd Generation Partnership Project (3GPP), 2021.

[84] 3GPP. 5g security assurance specification (scas);network data analytics function (nwdaf). Technical Specification (TS) 33.521, 3rd Generation Partnership Project (3GPP), 2021.

[85] 3GPP. Management and orchestration; management data analytics. Technical Specification (TS) 28.104, 3rd Generation Partnership Project (3GPP), 2022.

[86] A. Banerjee, S. S. Mwanje, and G. Carle. Trust and performance in future ai-enabled, open, multi-vendor network management automation. *IEEE Transactions on Network and Service Management (TNSM)*, 20(2):995–1007, 2023.

[87] 3GPP. Management and orchestration; intent driven management services for mobile networks. Technical Specification (TS) 28.312, 3rd Generation Partnership Project (3GPP), 2022.

[88] E. Zeydan and Y. Turk. Recent advances in intent-based networking: A survey. In *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, pages 1–5. IEEE, 2020.

[89] L. Pang, C. Yang, D. Chen, Y. Song, and M. Guizani. A survey on intent-driven networks. *IEEE Access*, 8:22862–22873, 2020.

[90] Y. Wei, M. Peng, and Y. Liu. Intent-based networks for 6g: Insights and challenges. *Digital Communications and Networks*, 6(3):270–280, August 2020.

[91] J. Silvander, K. Wnuk, and M. Svahnberg. Systematic literature review on intent-driven systems. *IET Software*, 14(4):345–357, 2020.

[92] K. Mehmood, K. Kralevska, and D. Palma. Intent-driven autonomous network and service management in future networks: A structured literature review. arXiv:2108.04560, 2021.

[93] C. Janz, N. Davis, D. Hood, M. Lemay, D. Lenrow, L. Fenkai, F. Schneider, J. Strassner, and A. Veitch. Intent nbi – definition and principles. *ONF TR-523*, 2016.

[94] A. Clemm, L. Ciavaglia, L. Granville, and J. Tantsura. Intent-based networking-concepts and definitions. *IRTF draft*, 2020.

[95] C. Li, O. Havel, W. Liu, A. Olariu, P. Martinez-Julia, J. C. Nobre, and L. DR. Intent classification. *IETF draft*, 2019.

[96] B. Claise, J. Quilbeuf, Y. El Fathi, D. Lopez, and D. Voyer. Service assurance for intent-based networking architecture. *IRTF draft*, 2021.

[97] ETSI GR ENI. Intent aware network autonomicity (itana). Technical report, 2021.

[98] 3GPP. Study on scenarios for intent driven management services for mobile networks. [Online]. Available: https://www.3gpp.org/ftp/Specs/archive/28_series/28.812/, 2020.

[99] 3GPP. Intent driven management services for mobile networks. [Online]. Available: https://www.3gpp.org/ftp/Specs/archive/28_series/28.312/, 2021.

[100] TM Forum. Intent in autonomous networks. [Online]. Available: https://www.tmforum.org/resources/how-to-guide/ig1253-intent-in-autonomous-networks-v1-2-0/, 2022.

[101] TM Forum. Intent modeling. [Online]. Available: https://www.tmforum.org/resources/standard/ig1253a-intent-common-model-v1-1-0/, 2022.

[102] TM Forum. Intent life cycle management and interface. [Online]. Available: https://www.tmforum.org/resources/standard/ig1253c-intent-life-cycle-management-and-interface-v1-1-0/, 2022.

[103] P. Szilágyi. I2BN: Intelligent Intent Based Networks. *Journal of ICT Standardization*, 9(2):159–200, 2021.

[104] J. Bogojeska, D. Lanyi, M. Botezatu, and D. Wiesmann. eznl2sql: A system for network devices management with a natural language interface for databases. 2021.

[105] M. Kajo, J. Schnellbach, S. S. Mwanje, and G. Carle. Robust deep learning against corrupted data incognitive autonomous networks. In *2022 IEEE/IFIP Network Operations and Management Symposium (NOMS)*. IEEE, 2022.