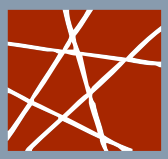


Network Architectures
and Services
NET 2013-04-1

Dissertation

Coordination of autonomic function execution in Self-Organizing Networks

Tobias Bandh

Network Architectures and Services
Department of Computer Science
Technische Universität München

TECHNISCHE UNIVERSITÄT MÜNCHEN
Institut für Informatik
Lehrstuhl für Netzarchitekturen und Netzdienste

Coordination of autonomic function execution in Self-Organizing Networks

Tobias Bandh

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:		Univ.-Prof. Dr. Alexander Pretschner
Prüfer der Dissertation:	1.	Univ.-Prof. Dr.-Ing. Georg Carle
	2.	Prof. Dr. Ir. Aiko Pras (University of Twente, Niederlande)

Die Dissertation wurde am 31.10.2012 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 26.03.2013 angenommen.

Cataloging-in-Publication Data
Tobias Bandh
Coordination of autonomic function execution in Self-Organizing Networks
Dissertation, April 2013
Network Architectures and Services, Department of Computer Science
Technische Universität München

ISBN 3-937201-34-3
ISSN 1868-2634 (print)
ISSN 1868-2642 (electronic)
Network Architectures and Services NET-2013-04-1
Series Editor: Georg Carle, Technische Universität München, Germany

"No one who achieves success does so without the help of others."

(Alfred North Whitehead)

I would like to express my deepest gratitude to all the people who have supported me on this journey in so many different ways.
To my wife, family and friends who had to accept my: "No time - got to work!" To my colleagues for their valuable input, the endless discussions and the good times we had! And to all those I met along the way and who have become my friends.

ABSTRACT

Today, mobile networks are facing an ever increasing number of users and more important industrial applications that strongly employ the wireless and ubiquitously available connectivity. The great demand for voice and especially data services drives their fast development. New technologies are constantly developed and deployed, in addition to the existing infrastructure, to satisfy the increasing demands. Currently LTE as the latest radio access technology is deployed side-by-side with the existing GSM and UMTS systems to provide a seamless and high quality user experience. The already high level of complexity to manage a large-scale, multi-technology network is further increased by the particularities of the mobile network world. Apart from the complexity to align the configurations of a large number of network elements, many of the employed interfaces and technologies are proprietary and only little is standardized. Even network elements from the same vendor but for different radio access technologies often require separate management systems. Hence, no single multivendor management system capable to configure all aspects of a network is available.

The need to reduce the management complexity, and the workload of the human operators in order to lower operational expenditures and at the same time improve network capacity and service quality led to a strong demand highly automated management solutions. This automation should already be available with the introduction of LTE. The majority of the mobile network operators and equipment vendors have created the vision of Self-Organizing Networks (SON). At the moment the focus of SON lies on the automation of individual use cases from all areas of network operation and management: initial network configuration, performance optimization and failure recovery. Use case specific SON-Functions provide the means for the automation. SON-Functions autonomically and independently retrieve and analyze network performance data and - if required - perform configuration changes. Multiple instances of individual SON-Functions will be concurrently active, operating on limited target areas within the network such as a small set of network elements. The danger of this mode of operation is the parallel execution of SON-Function instances that perform conflicting configuration changes on identical target network elements. For example, both, optimization and failure recovery SON-Functions can become active and perform their changes simultaneously. Instead of resolving the detected issues the simultaneous actions will either deteriorate the sit-

uation or lead to a loop of oscillating reconfigurations, which also leads to service degradation.

This thesis presents a concept for SON-Function instance coordination, to enable conflict-free SON-Function execution in accordance with the guidelines of the network operators. SON-Function instance coordination supports the goal that executed SON-Function instances contribute to optimized network performance. Operators anticipate system behavior through a set of policies that allow the SON Coordinator to govern autonomic execution of SON-Function instances. The concept covers not only the coordination but specifies also a set of requirements and guidelines for the SON function design and implementation. If the functions are created in accordance with these guidelines, the specification of the coordination logic is an implicit part of the function development process. Another major benefit is that these requirements enable the coordination of SON-Functions from different sources without a limitation to functions from a specific vendor.

This thesis describes a prototypic implementation to validate and evaluate the concept. For the evaluation the SON Coordinator was used to coordinate SON-Function instance execution requests from a commercial LTE network simulator that was running real-world SON-Function algorithms. In addition a case study of physical cell ID assignment was performed to show the development of a SON-Function and the appropriate coordination logic.

The results prove the benefits of a structured SON-Function development process and the usage of a vendor independent SON Coordinator. It is shown that if all vendors provide SON-Functions based on the proposed scheme and follow the same coordination procedures, management automation based on SON-Functions in a multivendor environment is not only feasible but also provides the possibility to satisfy operational requirements imposed by the network operators.

KURZFASSUNG

Die wachsende Anzahl von Nutzern und, weitaus wichtiger, industrielle Anwendungen, die von einer allgegenwärtigen Netzverfügbarkeit Gebrauch machen, belasten die heutigen Mobilfunknetze bis an die Grenzen ihrer Kapazität. Die starke Nachfrage nach Sprach- und Datendiensten treibt und beschleunigt die Weiterentwicklung dieser Netze. Um die steigenden Ansprüche zu befriedigen, werden ständig neue Technologien entwickelt und zusätzlich zur bestehenden Infrastruktur eingesetzt. Zur Zeit wird LTE als neueste Radiotechnologie parallel zur bestehenden GSM- und UMTS-Infrastruktur aufgebaut um nahtlose und hochqualitative Netzdienste bieten zu können. Der an sich schon sehr hohe Aufwand, ein sehr großes Kommunikationsnetz mit mehreren, parallel aktiven Zugangstechnologien zu betreiben, wird durch die Besonderheiten der Mobilfunkwelt noch weiter erhöht. Abgesehen vom Aufwand die Konfigurationen einer Vielzahl von Netzelementen aufeinander abzustimmen, kommen viele Schnittstellen und Technologien, die nur wenig oder gar nicht standardisiert, sind zum Einsatz. Daher kann es vorkommen, dass selbst für den Betrieb von Netzelementen eines Herstellers, aber für unterschiedliche Funktechnologien, mehrere Managementsysteme benötigt werden. Aus diesen Gründen gibt es kein einzelnes System, das in der Lage ist, über Technologie- und Herstellergrenzen hinweg, alle Aspekte eines Mobilfunknetzes zu konfigurieren.

Für die Betreiber von Mobilfunknetzen ist es eine unumgängliche Notwendigkeit, die Komplexität des Netzmanagements zu veringern, um die Arbeitslast der Operatoren und damit die betrieblichen Gesamtaufwendungen zu reduzieren. Zur gleichen Zeit muss aber die Qualität der angebotenen Dienste erhöht und die Fehlerbehebung deutlich beschleunigt werden. Dies führte zu einer starken Nachfrage automatisierter Betriebskonzepte, welche schon zur Einführung von LTE verfügbar sein sollten.

Die Mehrheit der Netzbetreiber, zusammen mit den Herstellern von Netzkomponenten, entwickelten aus diesem Grund die Vision der Self-Organizing Networks (SON). Zur Zeit liegt der Fokus der Arbeiten zu SON auf der Automatisierung einzelner Anwendungsfälle, welche aus allen Gebieten des Netzbetriebs stammen: Initiale Netzkonfiguration, Leistungsoptimierung und Fehlerbehebung. Die benötigte Automatisierung wird durch anwendungsfallspezifische SON-Funktionen erbracht. SON-Funktionen beziehen und analysieren, vollautomatisch und unabhängig von einander, Betriebsdaten aus dem Netzwerk und, veranlassen, falls nötig, Konfigurationsänderungen.

Mehrere Instanzen einzelner SON-Funktionen werden zukünfigt zur selben Zeit aktiv sein. Jede dieser Funktionen wird innerhalb eines beschränkten Gebiets, beispielsweise einer kleinen Anzahl von Zellen, operieren. Die daraus entstehende Gefahr ist die parallele Ausführung konfliktbehafteter SON-Funktionen auf denselben Netzelementen. Beispielsweise könnten verschiedene Optimierungsfunktionen aktiv werden und zur selben Zeit ihre Aktionen ausführen. Anstatt eine Verbesserung herbeizuführen, könnten die ausgeführten Aktionen entweder die Situation verschlimmern oder eine endlose Schleife sich wiederholender Konfigurationsänderungen auslösen, welche zu einer Verschlechterung der Qualität oder dem Ausfall der angebotenen Dienste führt.

In dieser Arbeit wird ein Konzept zur Koordination der Ausführung von Instanzen einzelner SON-Funktionen, mit dem Ziel eines konfliktfreien Betriebs in Übereinstimmung mit den Betriebskonzepten des Netzbetreibers, vorgestellt.

Die Netzbetreiber beeinflussen den Betrieb ihres Netzes durch eine Menge vordefinierter Policies, welche vom SON Coordinator genutzt werden um das Verhalten der SON-Funktionen zu steuern.

Das vorgestellte Konzept deckt nicht nur die Koordination an sich ab, sondern spezifiziert auch Anforderungen und Leitlinien für den Entwurf und die Implementierung von SON-Funktionen.

Wenn SON-Funktionen entsprechend dieser Leitlinien entwickelt werden, so ist die Spezifikation der benötigten Koordinierungslogik ein impliziter Teil des Entwicklungsprozesses. Ein weiterer Vorteil ist, dass diese Anforderungen die Koordinierung von SON-Funktionen aus unterschiedlichen Quellen ermöglichen, ohne dass sich ein Netzbetreiber auf die Funktionen eines Herstellers festlegen muss.

Eine prototypische Implementierung des SON Coordinators wird beschrieben, welche für die Validierung und Bewertung des vorgestellten Konzepts verwendet wurde. Für die Bewertung des Ansatzes wurde der implementierte SON Coordinator zusammen mit einem kommerziellen LTE Netzsimulator betrieben. Innerhalb des Simulators wurden Instanzen von SON-Funktionen ausgeführt, die mit dem SON Coordinator interagierten. Zusätzlich zu dieser Auswertung wurde eine Fallstudie durchgeführt, um die Entwicklung einer SON-Funktion und der zugehörigen Koordinierungslogik zu zeigen.

Die Ergebnisse verdeutlichen die Vorteile eines strukturierten Entwicklungsprozesses für SON-Funktionen und der Nutzung eines herstellerunabhängigen SON Coordinators. Es wird gezeigt, dass, wenn nur SON-Funktionen, die den vorgestellten Spezifikationen folgen zum Einsatz kommen und mit den vorgeschlagenen Koordinierungsverfahren betrieben werden, eine Automatisierung des Netzbetriebs nicht nur möglich ist, sondern auch die Möglichkeit bietet, die Anforderungen der Netzbetreiber voll zu erfüllen.

PUBLISHED WORK

This list provides an overview over the author's publications that are relevant for this thesis.

Chapters 4 - 7

- Tobias Bandh and Christoph Schmelz. Impact-time Concept for SON-Function Coordination. In *International Workshop on Self-Organizing Networks (IWSON 2012)*, Paris,France, August 2012.

- Tobias Bandh, Raphael Romeikat, and Henning Sanneck. An Integrated SON Experimental System for Self-Optimization and SON Coordination. In *International Workshop on Self-Organizing Networks (IWSON 2012)*, Paris,France, August 2012.

- Tobias Bandh, Raphael Romeikat, and Henning Sanneck. Policy-based Coordination and Management of SON Functions. In *IM 2011*, 2011.

- Tobias Bandh, Haitao Tang, Christoph Schmelz, and Henning Sanneck. SON Operation. In *Chapter in Book: LTE Self-Organising Networks (SON): Network Management Automation for Operational Efficiency, Seppo Hämäläinen and Henning Sanneck and Cinzia Sartori (Eds.), Wiley.* December 2011.

Chapter 9

- Tobias Bandh, Henning Sanneck, and Raphael Romeikat. An Experimental System for SON Function Coordination. In *Vehicular Technology Conference (VTC Spring), 2011 IEEE 73rd*, pages 1 –2, May 2011.

- Tobias Bandh, Raphael Romeikat, Bernhard Bauer, Georg Carle, Henning Sanneck, and Lars Christoph Schmelz. Policy-driven workflows for mobile network management automation. In *Proceedings of the 6th International Wireless Communications and Mobile Computing Conference*, IWCMC '10, pages 1111–1115, New York, NY, USA, 2010. ACM.

- Tobias Bandh, Raphael Romeikat, Henning Sanneck, and Haitao Tang. Policy-based coordination and management of SON-functions. In *VDE/ITG Fachgruppengespräche, Consistent System Optimization in 3G/4G Wireless Networks*, Stuttgart, Germany, October 2010.

Chapter 10

- Tobias Bandh and Christoph Schmelz. Impact-time Concept for SON-Function Coordination. In *International Workshop on Self-Organizing Networks (IWSON 2012)*, Paris,France, August 2012.

Chapter 11

- Péter Szilágyi, Tobias Bandh, and Henning Sanneck. Physical Cell ID Allocation in Multi-layer, Multi-vendor LTE Networks. In *Proceedings of 4th International Conference on Mobile Networks and Management*, Hamburg, Germany, September 2012. **Best Paper Award**.

- Tobias Bandh, Cinzia Sartori, Péter Szilágyi, Yves Bouwen, Eddy Troch, Jürgen Goerge, Simone Redana, Raimund Kausl, Christoph Schmelz, and Henning Sanneck. Self-Configuration (Plug-and-Play). In *Chapter in Book: LTE Self-Organising Networks (SON): Network Management Automation for Operational Efficiency, Seppo Hämäläinen and Henning Sanneck and Cinzia Sartori (Eds.), Wiley*. December 2011.

- Tobias Bandh, Raphael Romeikat, Henning Sanneck, Lars Christoph Schmelz, Bernhard Bauer, and Georg Carle. Optimized Network Configuration Parameter Assignment Based on Graph Coloring. In *NOMS 2010*, 2010.

- Tobias Bandh, Henning Sanneck, and Georg Carle. Graph Coloring Based Physical Cell ID Assignment for LTE Networks. In *IWCMC2009*, 2009.

CONTENTS

Part I

INTRODUCTION AND BACKGROUND

1. INTRODUCTION

1.1 Motivation for Network Management Automation in 3GPP Networks

Since the introduction of the first commercial Global System for Mobile Communications (GSM) networks in 1991, cellular networks experienced a continuous growth. Today mobile communication has become a ubiquitous and pervasive part of the daily life - not only in developed, but also in developing countries where wired communication systems have never been deployed to a large extend. The introduction of mobile communication systems is cheaper and therefore provides communication capabilities to parts of the population which were either out of reach for wired communication systems or could not afford the usage. There are several drivers for this fast development and deployment of new technologies. On the one hand the increasing usage of smart phones and tablet PCs which are more than personal communication devices but keep the user connected to a multitude of services and allow him to access information independently from place and time. In addition to the traditional services, location-based services will become more and more common. These require reliable connectivity to allow communication with their cloud based back ends.

On the other hand, mobile communication technologies are increasingly often used within industrial applications. Vehicles are equipped with data communication capabilities to provide latest traffic information, offer dedicated emergency assistance services [Mer09] and enable general fleet management. Other applications are goods tracking, mobile data acquisition terminals, remote metering for smart networks, surveillance devices, medical applications and many more.

With an increased availability of high speed wireless connections, the variety and number of applications and services that make use of mobile communication technologies will further increase.

The most prominent effect of the comprehensive usage of mobile communication technologies is the increasing amount of data transmission in mobile networks and the demand for spotless coverage. Cisco [Cis11] reported a global mobile data traffic usage of 236.7 Terabytes (TB) per month for 2010 and forecasts about 1163 TB per month for 2012.

In order to satisfy the demand for higher data rates and reliable ubiquitous connectivity new technologies are constantly developed and rolled-out

in addition to the existing deployments. In a first step, the existing technologies are further developed until they reach their limits, in a next step additional new technologies are deployed. One example is the deployment of Universal Mobile Telecommunications System (UMTS) that was introduced in addition to existing GSM networks. The deployment of new radio technologies always follows the same scheme. In the beginning so-called island coverage is provided in areas with very high traffic volumes. These islands are then subsequently extended to a full coverage. Vodafone announced in 2011 [Plc11] that their existing European 3G networks with about 66000 base stations will be extended with about 24000 additional base stations until 2013. It can be assumed that not all of these new base station are used to extend the coverage of their 3G network but will also be used to serve pico and micro cells to satisfy additional capacity demands in areas with existing 3G coverage.

The deployment of LTE as the latest radio technology follows the same scheme, it is deployed in addition to the existing GSM and UMTS infrastructure. Initial island deployments will stepwise be extended until full coverage is reached. It is noteworthy, that many operators introduce LTE networks, while they are still upgrading their deployed 3G networks to support HSPA or HSPA+ connections. This shows the unprecedented demand for network coverage and capacity.

The great benefit of cellular networks is the seamless user experience they provide. Services can be continuously used while roaming through the network. Connections are maintained by handing them over to the next cell, as soon as a user reaches the border of the cell that is currently serving the connection. Such a handover between cells of the same technology are called horizontal handovers. To enable those, the network has to be configured properly, which is a very complex task that requires a lot of operational experience. There is a large number of parameters that influence the success of handovers between two cells which have to be aligned. When a new radio access technology is deployed, vertical handovers between cells operating on different technologies have to be supported. In the same seamless way as with horizontal handovers, services should not be interrupted when the connection is transferred vertically, for example, from LTE to UMTS or UMTS to GSM. This requirement increases the configuration complexity for the mobile networks, as the configurations of base stations using different technologies also have to be closely aligned. Apart from the introduction of LTE additional concurrent network evolutions will increase the configuration complexity. In order to satisfy the user demands for coverage and available data rates the existing networks are enhanced at different levels at the same time. The overall number of cells is increased on the one hand by replacing three sector cells with six sector cells, and on the other hand through the introduction of additional smaller cells at high traffic hotspots. That means large macro cells are complemented with smaller micro, pico and even femto

cells in hierarchical cell layers. Large cells serve moving users and provide large scale coverage while small cells serve stationary users at particular places and guarantee high data rates. A major increase in cell numbers will come from the introduction of femto cells at the customer premises. These cells serve individual users or families in their homes by providing very small cells. First estimates predict [Sma12] that the deployments of small cells, including femto, pico, and micro cell, will outnumber macro cells by the end of 2012. These so called Heterogeneous Network (HetNet) [ea11] deployments, with a large number of cells that need to be configured properly, will have a large impact on the increasing configuration complexity.

The configuration of a mobile network is not static. As the network evolves over time, configurations have to be adapted whenever equipment is newly installed or replaced. It is impossible to provide an optimized configuration, based on theoretic knowledge and simulations, when a new base station is deployed. At least subsequent optimizations will always be required to fine tune the configuration in accordance with the local circumstances. But also due to the characteristics of wireless connections, the configuration often needs to be adapted over time. Radio propagation can be strongly influenced by different environmental factors, for example, changes in the building density, interference through other radio sources, or through changing seasons and the involved changes in the foliation. To compensate the radio propagation variations, additional configuration changes are required over time.

The overall high complexity of a mobile wireless network, makes the management of such a network very labor-intensive and therefore very expensive. Even for day-to-day tasks a lot of operational experience is required which keeps the operational staff occupied. Due to the large number of interdependencies and not fully obvious cause and effects manual mobile network management and operation is very error prone. Another source of errors lies in the large number of human operators that is required to run a network. This involves the risk that performed actions are not communicated in time to other operators before they perform conflicting actions. It increases also the risk for misinterpretation of an observed situation, as it can be strongly influenced or already be resolved by another task that has just been performed. Management automation has been introduced in todays Operation Administration Maintenance (OAM) systems to a smaller extent, mostly in form of scripting capabilities. This kind of automation allows a human operator to combine task sequences and letting them run automatically. A higher degree of automation is necessary to free human operators from recurring tasks and let them concentrate on more important failure recovery and optimization tasks, but automation is also necessary to reduce the number of errors. It is obvious that the available management automation points into the right direction, but a bare scripting capability is far from sufficient. New management concepts are required to allow the

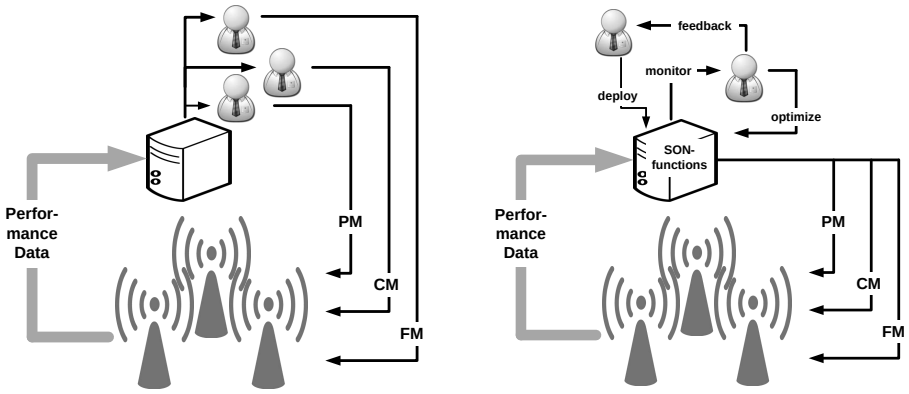
network operators to reduce their Operational Expenditures (OPEX) and at the same time increase the operational efficiency and the service levels.

1.2 Self-Organizing Networks and SON-Functions

Already with the introduction of UMTS it became obvious that the prevailing management concepts have reached a limit in terms of scalability. The management of the new base stations could often not be performed through available management systems, but Radio Access Technology (RAT) specific systems had to be used. If equipment from different vendors was used, multiple vendor specific management systems had to be employed. There were no central multi vendor management systems available to configure all aspects of the deployed base stations. This multitude of management systems used in parallel strongly increased the operational overhead.

The pressure to provide high quality services with minimal interruptions on the one hand and to reduce the OPEX on the other hand led the Mobile Network Operators (MNOs) to demand improved management concepts. As a result and in order to be able to better express their demands, the Next Generation Mobile Networks (NGMN) Alliance was founded by a number of MNOs in 2006 [All11] with the clear goal to formulate and agree on industry-wide requirements for future mobile network generations and influence the development and standardization processes beginning with the specification and development of LTE and the Evolved Packet Core (EPC). Today the alliance consists of 57 partners including equipment vendors, universities, and other, non-industrial research institutions.

Among other topics [ABG+06] a strong focus of the alliance lies on the development of use cases and guidelines for network operation and management, and in particular the introduction of network management automation by design and not as a somehow alien add-on. Automated functions perform data aggregation and analysis and, if required, perform respective management tasks. In a first step, operational staff should be fully freed from standard, time-consuming, day-to-day management and operation tasks to allow them to focus on emergency intervention and large-scale optimization and network planning. Subsequently the automation is strongly extended into all areas of network management, changing the operators duties from manual interventions to design and supervision of functionalities. The expected effects of the automation are, apart from the OPEX reduction, a minimized reaction time and a more reliable and faster resolution of detected problems in the network with a reduced number of erroneous configuration decisions. The NMGN Alliance emphasized their strong demand for management automation through the publication of several whitepapers, which first introduced the requirements on a general level [Leh07b] and then provided a list of use cases related to Self-Organizing Networks (SONs) [Leh07a].



(a) Traditional Network Management (b) SON based Network Management

Fig. 1.1: Network Management Differences

From these initial thoughts and requirements the concept of SONs emerged. The vision and overall requirements of the MNOs on SON were clarified in a subsequent whitepaper [Leh08] stating both, additional recommendations on SON and specific Operation and Maintenance (OM) requirements. The work of the NGMN Alliance has affected the work of the Third Generation Partnership Project (3GPP) and SON use cases have been included into the standardization process and are now part of the 3GPP standards [3GP10b, 3GP08]. SON as management paradigm is seen as a holistic approach that comprises all aspects of network management as a continuous loop of the self-* phases [KC03a], including initial self-configuration, self-optimization and self-healing. The differences to traditional mobile network management are shown in Figure 1.1. In traditional network management different management paths exist for Fault Management (FM), Configuration Management (CM) and Performance Management (PM). Operators analyze the performance data provided from the network and perform their respective tasks. Whereas, in a SON enabled network the operational staff performs no manual interventions but defines and deploys, and monitors and optimizes automated management functionalities.

The SON concept evolved over time, the first publications focused more on the use cases and general scenarios, the subsequent publication in 2008 [Leh08] already refers to specific SON-Functions as the means to provide the functionality described in the use cases. In addition to the mainly operator controlled NGMN Alliance also equipment vendors and academia, for example in the EU funded FP7 SOCRATES project [Sch08], contributed to the overall understanding of SON. The full concept will not be introduced at once, but SON functionalities will be introduced step-by-step with each new 3GPP release [3GP11a] into the mobile networks.

Table 1.1 gives an overview on the releases and the SON aspects they

Tab. 1.1: 3GPP Releases and introduced SON-Functions

3GPP Release	Management Area	SON-Function Example
Release 8	Self-configuration	PCI, ANR
Release 9	Self-optimization	MRO, CCO
Release 10	Self-healing	Enhanced ICIC

focus on. In Release 8 [3GP12d], self-configuration functionality is targeted, as for example the Automated Configuration of Physical Cell IDs or the Automatic Neighbor Relation function. SON in Release 9 [3GP12e] has a focus on self-optimization in particular the optimization of handover and radio parameters. 3GPP Release 10 [3GP12b], apart from continuing the work of the previous releases introduces first solutions in the area of self-healing. Functions that target coverage restoration after a cell failure are part of this release. Depending on the success of the standardization the introduction of individual SON-Functions can also be delayed and shifted from one into the next release.

The standardized features are only a first step for the realization of SON. The concept has to follow the evolution of the networks. More SON use cases have to be defined to keep up with the new requirements of the evolved networks, existing solutions have to be adapted to be usable in future HetNet deployments. Features that have first been introduced into LTE networks will have to be introduced for all deployed technologies as GSM and UMTS to align and simplify the management of all parts of the deployed networks, especially with the focus on HetNets.

In a final deployment state, the mobile networks will be managed to almost full extent through a large number of use case specific SON-Functions, provided by different sources. Equipment vendors will provide an initial set of functions, which then can be extended by the network operators or through specialized function suppliers. It is likely that different functions for single SON use cases will be deployed in parallel within a single network. Those are specific to particular vendor equipment or for different network deployment scenarios.

The deployed SON-Functions will be concurrently active. Depending on their mode of operation they will either continuously, at certain points in time or triggered by particular events analyze data to detect the use case they are dedicated to. If the data indicates the existence of the triggering situation for a particular use case the SON-Function will take actions to resolve the detected situation. SON-Functions will not always operate on the complete network, often their spatial activity will be restricted to limited areas, for example, to optimize the handover parameter settings between two cells. A SON-Function instance describes a particular instance that operates on a single target or a set of targets. In general it is possible that

multiple instances of the same SON-Function are active in the network, for example, optimizing the handover parameter settings for multiple cell pairs in the network. The limited spatial scope of individual SON-Functions allows the concurrent execution of instances from different SON-Functions in non-overlapping parts of the network.

The possibility to concurrently perform different management tasks in different parts of the network make SON based network management much more efficient. SON-Functions instances will automatically be triggered whenever required and will provide a reduced delay between detection and resolution of SON use cases.

1.2.1 Risks of uncoordinated SON-Function Execution

Independent of management tasks being performed manually by the human operator or automatically through SON-Functions, there is always a risk of conflicts between the performed tasks. Conflict-free management tasks execution in a large scale network can only be reached through a full serialization of all performed tasks. This approach is not feasible as it is very time-consuming, especially if the execution of individual tasks is very long-running, therefore parallel execution is an absolute requirement. In general a concurrent task execution is not a problem, since many tasks affect only a limited fraction of the network. As long as two tasks affect only disjoint parts of the network they can be executed concurrently. But, in a worst case, conflicting management tasks with overlapping target areas will have severe effects on the network and can lead to service quality degradation or even major network outages. In traditional network management the human operator has to be aware of potential conflicts and act accordingly. Tasks are either not performed, delayed, or put into a particular sequence based on priorities and operational guidelines. For example, usually failure recovery will always be given precedence over network optimization. But a network operator will not take his decisions only based on absolute priorities but will consider more criteria and include context information. Examples for context information that often has a strong influence on management decisions are the region where a particular effect has been detected, but also date and time. Cell failure is treated differently in a high-traffic urban area with multiple cell layers than in a low-traffic rural area with coverage provided only through macro cells. Sometimes a network operator has to decide whether it actually makes sense to perform the tasks that are usually performed when a particular situation has been detected.

For example, a network will experience overload situations with a lot of dropped calls and lost handovers at certain days of the year. New years eve is a typical day for effects that normally would indicate a major malfunctioning. In such a situation, which is not caused by any error, there is no need to perform large scale optimizations or even failure recovery. Those actions

might even be counterproductive as soon as the load is back to normal. The same applies to special events like trade-fairs, large sports or political events. Operational knowledge gathered over long time is required to provide operational guidelines, specific to a particular network, to correctly handle such exceptional circumstances.

In order to maintain a proper and reliable network operation in a SON enabled network two important aspects have to be treated. First, it is immanent to either pre-exclude conflicting behavior of SON-Functions or detect and resolve it at run-time. Second, the operation of the SON-Functions has to be in accordance with the operational guidelines.

In a network that is almost fully managed through SON-Functions it is impossible to deploy only non-conflicting SON-Functions, particularly because often failure recovery and optimization SON-Functions modify identical configuration parameters, thus are in a potential conflict. Other means as, for example, self-organization between individual SON-Functions is not scalable. With each introduction of a new function already deployed functions need to be adapted, which can be time-consuming and increases the risk of conflicting behavior.

Whenever the execution of a SON-Function instance is triggered it has to be assured that the execution of this particular instance does not violate operator guidelines, which requires a detailed analysis of the network context. It is difficult to implement this analysis into the SON-Functions, as the guidelines are operator and network specific and evolve over time. With a function internal implementation of context analysis for operational guideline compliance all SON-Functions need to be tailored to a particular network. In addition the functions would need to be updated each time the operational principles are changed.

In case of conflicts caused by SON-Function execution or the violation of operational guideline there is a major risk of a reduced network reliability, service level degradation or even network outages.

1.3 Research Questions

In order to avoid the risks of SON-Function execution the operation of the SON-Functions has to be controlled. This control has to happen in an fully automated way to maintain the benefits of the autonomously acting SON-Functions. Therefore a **reliable**, **robust**, **efficient**, and also **flexible** approach for the run-time control of SON-Function execution in accordance with operator guidelines is required. From these basic requirements, different research questions are derived and treated within Part II of this thesis. The following sections describe the questions in detail and group them according to the category they fall into.

1.3.1 Reliability and Robustness

Reliability and robustness are the most important requirements for the management of a mobile network. Network operators require a system behavior that will not lead to service degradation due to conflicting actions being taken.

R1 How can the danger of negative effects through the execution of conflicting SON-Function instances be minimized?

R2 How can the network operator, despite the automation, still be in full control over the ongoing processes in the network?

R3 How can the enforcement of and the compliance with all operational guidelines be assured?

R4 Which parts of the SON-Functions are affected through conflicts?

R5 How can the structure of a SON-Function support the detection and resolution of conflicts?

R6 Are there characteristics of SON-Functions that can support the detection of run-time conflicts?

R1 is the central question within this thesis, the solution is presented in Chapter 7, where the proposed approach is compared to related approaches. R2 and R3 target very similar areas, the operator wants to be in full control and at the same time enforce guidelines which exceed bare conflict resolution. The answers to questions R4 to R6 will influence the preparational design-time processes and have an affect on the success of the run-time conflict detection and resolution. Their individual aspects are studied in Chapters 4-6.

1.3.2 Efficiency

Multi-technology and Multi-vendor networks consist of thousands of Network Elements (NEs) distributed over a large geographic area. The introduction of management automation through SON-Functions will lead to a large number of concurrently operating SON-Function instances. This requires a highly efficient system to control the SON-Function operation, otherwise it will reduce the benefits of the automation.

E1 How can new SON-Functions be designed and developed such that they can be easily integrated into the SON enabled network?

E2 How can conflicting behavior be detected and prevented or resolved?

E3 Is there a single solution for all conflict types, or are different solutions required?

E4 Which is the most efficient way to take the required coordination decisions?

E5 How can the required context information be provided efficiently to the system controlling the SON-Function instance execution?

E6 How can the number of concurrently, conflict free executed SON-Functions be maximized?

The questions that concern the efficiency of the conflict detection and resolution and also the execution of a large number of SON-Functions in a very network consisting of a large number of NEs are related to each other. A general design scheme for SON-Functions (E1) is studied in Chapter 4. Based on these results the first aspect of E2, the possible conflicts, is examined in Chapter 5. The detection and resolution (E2 - E4) is further developed in Chapter 7. The amount of information that needs to be stored and processed (Question E5) is an important factor for the efficiency and scalability and is therefore examined separately in Chapter 7.2.3. Equally important is the possibility to execute as many SON-Functions in parallel as possible. The required SON-Functions characteristics for E6 are examined in Chapter 6.

1.3.3 Flexibility

Mobile networks are constantly evolving, new network elements are added, and new network technologies are introduced. In the future, new SON-Functions will be added or existing SON-Functions will be adapted to fit to the performed network changes.

F1 Is there a uniform, future proof design scheme for SON-Functions?

F2 How can the decision logic be provided in a easily adaptable way?

F3 How can the system be designed such that it will be able, without problems, to cope with newly introduced SON-Functions?

F4 Is the concept sufficiently flexible to be used in networks with different operational modes?

Chapter 4 introduces a uniform design scheme (F1) for the development of SON-Functions. A flexible approach to design and represent the coordination logic (Question F2) for the SON Coordinator is provided in Chapter 7.2.1. The way how this decision logic is represented has also strong effects on F3. If the logic is easily adaptable, new SON-Functions can be added to the system without further problems.

Different networks are operated differently. Especially in the transition phase between traditional network management and SON based network management, there are still many things that have to be done the traditional way. Therefore F4 is important. The used approach has to be flexible enough to allow the usage in networks with different operational modes. Examples how the presented approach can be adapted to be usable in different environments are given in Chapter 6.2.3.

1.4 Approach and Contributions

The previous Sections have shown how SON realizes the vision of management automation for next generation mobile networks. The thesis shows, that it is not sufficient to just deploy SON-Functions in the networks to benefit from their features. Uncoordinated execution of SON-Function instances introduces the strong risk of service quality degradation or network outages. Not only inter- and intra SON-Function coordination to avoid conflicting configuration changes has to be performed, but also the compliance with operational guidelines has to be assured.

The listing of SON features in the coming 3GPP releases, shows that the main focus of the current developments is on the stepwise realization of individual SON use cases through the introduction of particular SON-Functions. Which is not only reflected in the 3GPP standardization specifications, but also in the available scientific publications. For example publications on particular network optimization functions [JBS+11, HL11, EKG11, RdlBMB11] or failure recovery functions [AJLS11].

There is only very little work done on the detection and resolution of conflicting SON-Function behavior. Apart from the generic coordination framework introduced by the SOCRATES project [SAE+11] most work focuses on the coordination of pairs or small numbers of SON-Functions for example handover and load balancing optimization which affect each other strongly [LSJB11].

The main contribution of this thesis focuses on the introduction of a generic SON Coordinator concept that allows vendor and technology agnostic coordination of SON-Function instance executions. As an enabler to the coordination process the requirements on the development of SON-Functions are specified. Section 1.4.1 gives an overview on the SON Coordinator concept and the general requirements that influenced the concept development.

The thesis is based on a thorough analysis of the proposed SON features. Descriptions of the state of the art, available SON use cases, and SON-Function description serve as an initial input to provide formal concepts for SON-Functions and the way how SON-Functions are executed. In an iterative process, prototypic implementations together with more fine grain analyses and simulations of the network behavior are used to refine the

proposed approach for the conflict-free execution of SON-Function instances in a SON enabled network.

The final concepts are evaluated with help of commercial network simulation tools, first generation SON-Functions and an implementation of the proposed SON Coordinator concept.

1.4.1 SON Coordinator and SON-Function Concept

The goal of the presented concept is to provide means for efficient SON-Function execution with run-time conflict detection and resolution. The design of the concept is influenced by the following criteria:

- Maximize the number of parallel SON-Function instance executions
- Provide run-time capabilities to detect and resolve conflicts between SON-Function instances
- Allow the enforcement of context-sensitive operational guidelines
- Allow full operator control to modify the system behavior at run-time
- Facilitate the introduction of new SON-Functions and adaptation of the coordination behavior

The proposed SON Coordinator concept introduces a coordination layer which operates between the SON-Function instances and the targeted network elements. Before a SON-Function instance is allowed to perform any of its tasks it has to request the permission. The SON coordinator evaluates the request and the current network context to decide whether the request can be granted or has to be rejected. For successful coordination it is important to include among others information about concurrently active SON-Function instances into the decision process. This information is required to enable the predefined coordination logic to detect and resolve SON-Function conflicts or violations of the operational guidelines. The policy based coordination logic directly reflects the operational guidelines for a particular network. It is maintained within the SON Coordinator and can therefore, and due to the policy foundation, be easily adapted if required. This allows to limit the SON-Function functionality to the detection and analysis of a particular SON use case trigger situation and the capability to perform the required actions to resolve the situation. That way network independent, but SON use case specific, SON-Functions can be provided and deployed.

1.4.2 SON-Function Structure Requirements

In order to provide a flexible approach that is not limited to the coordination of currently known SON-Functions, or SON-Functions from a particular vendor, this thesis defines a schema with a set of minimal requirements for the

design and implementation of SON-Functions. The schema gives a rough specification on how functionality has to be assigned to the Monitoring-, Algorithm-, and Action-part of a SON-Function. The transition points between the SON-Function parts define control points where a communication with the SON coordinator can be required, to determine whether the SON-Function instance execution should be continued or interrupted. All SON-Functions that follow the introduced schema can be coordinated through the SON coordinator, which allows a vendor independent SON-Function instance coordination at network level.

1.4.3 Coordination Schemes

The SON Coordinator concept together with the SON-Function design scheme allows the usage of different coordination schemes. Depending on the level of information that is required an operator can assign different coordination schemes to a SON-Function, which define at which control points of the SON-Function execution the function instance has to interact with the SON coordinator. The selected coordination schemes are highly dependent on the potential conflicts of a given SON-Function to other deployed SON-Functions.

1.4.4 Conflict Analysis and Categorization

The input data of SON-Functions and their impacts on the network was analyzed to provide a categorization of potential conflicts between deployed SON-Functions. Both, the SON-Function design scheme and the coordination schemes are results of this analysis. The performance of a conflict analysis based on the introduced conflict categorization is proposed as part of the SON-Function design-process. The conflict analysis provides the input that is required to define the function specific coordination logic and supports the selection of an appropriate coordination scheme.

1.4.5 Evaluation of SON Coordinator concept

The performed evaluation of the overall SON Coordinator concept is based on two approaches. In a first approach artificial event traces were used which simulated the requests from different multiple instances of different SON-Functions in the network. This approach served mainly the goal to get a better understanding of the behavior and performance of the SON coordinator and the deployed coordination logic. Multiple instances of up to eight different SON-Function types were concurrently active. After this evaluation the SON Coordinator was integrated into an existing demonstrator setup. This demonstrator consists of a LTE Network simulator and a SON-Function execution entity which receives Key Performance Indicator (KPI) values from the simulator and has the ability to enforce the configuration

changes within the simulator. In order to analyse all aspects of the concept and prove the benefits of SON-Function instance execution coordination, different experiments were performed.

The results showed that the coordination of SON-Function instances has a positive effect on the network performance. The SON Coordinator detects and prevents respectively resolves conflicting SON-Function instance behavior by giving precedence to individual function instances based on the provided coordination logic. It also makes use of the ability to influence the operation of individual SON-Function instances, for example, to prevent oscillating reconfigurations or enforce other operational guidelines.

The comparison of the results from experiments without and with coordination shows that the coordination of the SON-Function instance execution has a consistent, significant positive impact on the relevant KPIs.

1.4.6 Case Study

For a case study, the design process was applied to the Physical Cell ID (PCI) use case. Each cell in an LTE Network has to be assigned a Physical Cell ID. The difficulty is that there are only 504 valid PCIs available, and certain requirements on the re-use of IDs exist. The use case was analyzed, and based on the resulting requirements the respective SON-Function was developed. After the conflict analysis an appropriate coordination scheme was selected and the coordination logic specified.

For the evaluation of the PCI SON-Function several different network scenarios were used to analyze the behavior and the performance of the SON-Function. The results show that even for complex networks scenarios, with a large average number of neighborships per cell, the developed SON-Function is able to provide a valid PCI assignment, and the operator requirements for the minimal re-use distance are obeyed.

1.5 Outline

This Thesis consists of five parts. The first part provides an introduction and background knowledge on 3GPP based mobile networks, network management and related work. In Part 2 the details for the SON-Function coordination approach are presented. Details on Implementation and Evaluation are given in Part 3. A case study based on a SON use-case is performed in Part 4. The final conclusions and the outlook are presented in Part 5.

Chapter 1 provided the motivation for management automation in 3GPP based mobile networks and introduced the concept of SON and SON-Functions. Based on the operational risks that have been identified for SON enabled networks the central research questions and contributions of this thesis have been highlighted.

Chapter 2 provides the background information about Mobile Networks and Mobile Network Management which are required for the further understanding of the thesis.

Relevant related work is discussed in Chapter 3. The most important related work has been provided by the SOCRATES FP7 Project, which is presented in Chapter 3.1.

The conceptual part contains multiple chapters. A basic, formal definition of the SON-Function structure is introduced in Chapter 4. Conflicts that are possible between different SON-Functions are described in Chapter 5. The concept of SON-Function instances and the properties of SON-Function instances, like the Impact-area and Impact-time, are presented in Chapter 6.

The detailed description of the SON Coordinator and the coordination approach is given in Chapter 7. After an initial presentation of different conflict resolution and prevention methods, requirements on the input that is required to perform the coordination are discussed and the coordination process is introduced.

Based on the findings of the previous chapters, a dedicated SON-Function design process is derived in Chapter 8. The process covers all steps from the use-case description to the final SON-Function which can be deployed to the network including the conflict analysis and the coordination logic.

The SON Coordinator concept has been realized in a prototypical implementation which served for verification and evaluation purposes of the concept. Details about the implementation are given in Chapter 9.

Chapter 10 presents the evaluation of the coordination concept. The evaluation is based on the implementation of the SON Coordinator concept, performance measurement data from a commercial LTE network simulator, and a set of first-generation SON-Functions.

The case study in Chapter 11 presents the complete process for the development of a SON-Function starting from a SON use case description. It focuses on the development of a reliable SON algorithm that provides deterministic results.

Finally, Chapter 12 summarizes the contributions, provides a conclusion and an outlook on open questions and future work.

2. BACKGROUND

This chapter provides background information which is required for the better understanding of the thesis. It gives an introduction to the most important details and concepts of 3GPP based mobile networks and also an overview of related work.

2.1 Mobile Networks

This thesis focuses on SON as a feature of the 3GPP based evolved mobile networks, in particular Long Term Evolution (LTE). This section gives an overview of the 3GPP as the responsible standardization body and its work. It will also give a high level introduction of the structure of 3GPP based wireless mobile networks and their management.

2.1.1 Standardization in the 3GPP

The development of the GSM communication networks was a pan-European effort to create a public wireless communication system with standardized technical and operational characteristics [Tem10]. The GSM working group was initiated by the CEPT in 1982 to coordinate the work in the different European countries. After the introduction of the first GSM Networks in 1991, they became a global success. Today there is almost no country in the world without an active GSM based mobile network.

The 3GPP was founded in 1998 to drive the evolution of the GSM networks and introduce a globally usable 3rd Generation (3G) mobile communication system based on the already deployed GSM technology. Today the 3GPP has over 300 institutional members and is also responsible for the evolved 3rd Generation and beyond Mobile Systems [3GP].

The 3GPP has the responsibility to standardize all aspects of a mobile communication network that are required to guarantee interoperability between equipment provided by different vendors. The topics that are covered are also reflected in the organizational structure of the 3GPP. Four Technical Specification Groups (TSGs) cover the basic topics:

- **TSG GERAN:** Radio Access Network (RAN) for GSM and EDGE
- **TSG RAN:** RAN for UMTS Terrestrial Radio Access (UTRA) and Evolved Universal Terrestrial Radio Access (E-UTRA)
- **TSG SA:** Services and Systems Aspects

- **TSG CT:** Core Network and Terminals

Each of the TSGs has number of Working Group (WG) which work on specific subtopics. For example the SA 5 (TSG SA, WG 5) is responsible for the standardization of telecom management topics.

2.1.1.1 3GPP Releases

The work of all 3GPP WGs is combined into a set of releases. A release comprises a set of standardized features and serves as a stable platform for the developers, but also to define the new features being introduced to the market.

During an initial phase, the WGs contribute new features and functionalities to a release until a point where the release is frozen. No new functions can be added to the release from that point on, but have to be introduced to the next release. The features in the frozen release do not yet have to be fully standardized. In the beginning the releases were named after the year they were introduced, but this naming scheme was discontinued with Release 4. Thus, the releases 98 and 99 have been published prior to Release 4.

For each release the 3GPP maintains a document with a high-level description of the features that are part of the respective release. For example, in January 2012 the document for Release 10 [3GP12c] provides an overview of the new features per WG. It names the state of the standardization process, the important Technical Specifications (TSs), reflecting the standards, and Technical Reports (TRs) and also dependencies to other standardization processes for example within the IETF.

2.1.1.2 Technical Reports and Specifications

The work of the WGs is assembled in a set of TRs. Those documents contain all proposals for standardization from the contributors. If a proposal is recognized by the TSG the work on the particular topic is continued. Each TSG publishes its result in a set of TSs. Both TSs and TR are assigned a number code, that allows to relate the content to the topic and which phase it is in. Detailed information about the handling of the reports and specifications as well as the numbering is also published as a TR [3GP11d].

2.1.2 Mobile Network Architecture

The architecture of the 3GPP based mobile communication networks is very hierarchical. It is in general separated into the RAN and the Core Network (CN) and their respective network elements as shown in Figure 2.1. The difference between the different RANs is the usage of different RATs. For the user of a mobile network the usage of a different RAT becomes apparent through the different data rates that can be reached. In the language use

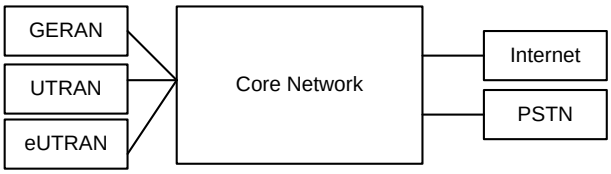


Fig. 2.1: Architecture of 3GPP based Mobile Communication Networks

the different deployed RATs are referred to via the terms 2G for GSM, 3G for UMTS and 4G mostly for LTE. For marketing reasons some operators call their UMTS networks that support High Speed Packet Access (HSPA) and HSPA+ connections also for 4G networks.

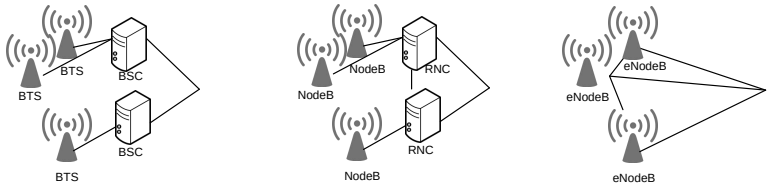
2.1.2.1 Radio Access Networks

The lowest level is formed by the base stations that serve the cells of the network and communicate directly with the mobile handsets using a particular RAT. The base stations in GSM are called Base Tranceiver Station (BTS) and NodeB for UMTS, respectively Evolved NodeB (eNodeB) for LTE RANs. Depending on the RAN a different number of intermediate network elements are situated between the base stations and the first core network nodes. The number of intermediate nodes has been reduced with every newly introduced RAT, on the one hand due to the improved performance of the hardware on the other hand to reduce the complexity of the network. Functionality that has been distributed to multiple network elements in GSM has been merged into fewer in UMTS and then into a single network element, the eNodeB in LTE. The following listing gives a short overview on the RAN network elements for the different RATs.

- **GSM:** A Base Station Controller (BSC) controls several BTSs, it is, among other things, responsible for the outer power control loop. The BSC serve as concentrators for the communication of the BTSs towards the Mobile Switching Centers (MSCs) in the Core Network.

- **UMTS:** The network elements of the UMTS Terrestrial Radio Access Network (UTRAN) are the NodeB which replace the BTSs and the Radio Network Controllers (RNCs) that controls several eNodeBs. Already for UTRAN more functionality has been shifted to the NodeB which in GSM resided in higher level network elements. As a difference to the BTSs in GSM, RNCs can be connected directly to each other which is an important requirement for the support of soft handovers.

- **LTE:** The architecture of the Evolved Universal Terrestrial Radio Access Network (E-UTRAN) is different to GSM and UMTS. There is



(a) GERAN (b) UTRAN (c) E-UTRAN

Fig. 2.2: Radio Access Networks

only a single NE type deployed, the so called eNodeB. It contains all functionality that has previously been provided by two individual NEs types in GERAN and UTRAN, therefore direct connections between the eNodeBs that serve neighboring cells are used.

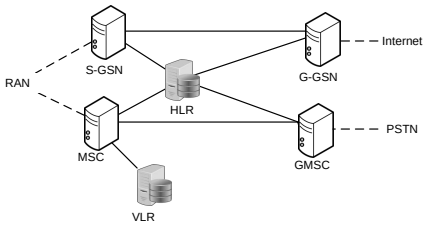
An overall overview of the different RANs is given in the respective TSs that also provide further references for more detailed specifications. (GERAN [3GP11c], UTRAN [3GP11i], E-UTRAN [3GP08])

2.1.2.2 Core Networks

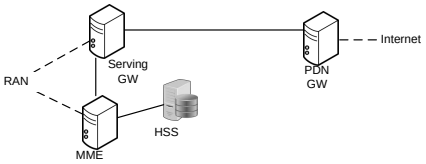
GSM and UMTS share an identical core network. RNCs and BSCs connect to the same core network elements. The MSC for circuit switched communication and the SGSN for packet switched communication. The usage of the common core network had the advantage that only minor adaptations were required for the introduction of UMTS. While for UMTS networks there is only a migration path towards an All-IP network. LTE has been designed to be an All-IP network only, which mandates the introduction of a newly designed core network architecture, thereby changes the networking paradigms prevalent in GSM and UMTS [Alc09]. The LTE architecture with EPC offers a high degree of flexibility to increase the networks performance and reliability, for example, through Serving Gateway (SGW) or Mobility Management Entity (MME) pooling [Mot]. Special interfaces between the different core network architectures have been designed, to allow an interoperation. Figure 2.3 gives an overview on the GSM / UMTS and LTE core networks. For a deeper understanding of the different network architectures and the technologies used in RANs and CNs it is referred to a set of good textbooks which are available on the topic, for example [HT11, Sau11, Sau09, STB11].

2.1.2.3 Network Deployments

From the very beginning of the first GSM networks, deployments always followed the same scheme. At first the networks were deployed in areas with a high demand for the service. Usually in large cities or other high traffic



(a) Combine 2G / 3G Core Network



(b) Evolved Packet Core (EPC) Network

Fig. 2.3: Core Networks

hotspots, or in areas where legal regulation required network coverage. From these so called island deployments the networks are step-by-step extended until full coverage is reached. There are only very few deployments where a complete network was deployed before it went operational. The evolution of the networks targets not only full coverage, but networks are also enhanced to provide more capacity when needed. There are in general several means to increase the capacity from a deployment point of view. For example by changing the deployment from three sector to six sector cells. In a next step the size of deployed cells is decreased and new cells are introduced into the network. The last possibility that can be used without deploying new RATs for a higher network capacity is to introduce very small micro or pico cells in addition to the general coverage provided by macro cells.

As already stated, there is a general trend for an increasing usage of the mobile wireless networks that drives their fast evolution. Therefore the mobile network operators deploy new networking technologies in addition to already deployed networks. In current networks GSM, UMTS and LTE are deployed in parallel. Initially the new network elements are deployed to already existing sites, since it is a very time-consuming and expensive process to acquire the permissions required to introduce new sites. As a result network equipment for different RANs is present at most sites and at least initially the cells served from a particular site cover roughly the same area.

Most operators use equipment from different vendors within their network. For operational and management reasons they are deployed in clusters within the networks, they form the so called vendor domains or vendor

clouds. Especially in GSM and UMTS networks such a setup was enforced by the hierarchical architecture of the network and the interdependence between BTS and BSC or NodeB and RNC respectively. The reduction to a single NE type in the RAN and the introduced capability of the eNodeBs to interact directly eliminates a structural requirement for vendor cloud deployments in LTE, but due to organizational reasons there will still be vendor clouds which cover a closed geographical area. The deployment of HetNets will also lead to the deployment of NEs from different vendors for the same RAT in the same geographical area [ea11, Net12]. But of course it can be expected that there is still some sort of vendor cloud due to organizational reasons.

2.2 Mobile Network Management

This section introduces several important aspects of the management for 3GPP based wireless mobile communication networks. At first an overview over the management architecture in a multi RAT, multi vendor network is provided. Subsequently an introduction to todays used network management approaches is given and important characteristics that are substantial to mobile network management are highlighted .

2.2.1 Management Architecture

The strong hierarchical structure of the 3GPP network architecture is also visible in the management architecture standardized by the 3GPP [3GP11e].

Several NEs are combined and managed through an Element Management System (EMS). Several EMSs are combined and managed through Domain Managers (DMs). Above the DMs a central Network Manager (NM) system is used to control the overall network.

Not only the hierarchical structure, but also the deployment structure is reflected in the management hierarchy, especially the deployed vendor clouds have their direct counterparts.

All NEs of a vendor cloud are managed through an EMS of that particular vendor which in term is managed by vendor specific DMs and a respective Network Management System (NMS).

Such a setup allows to control all aspects of a vendor cloud through a single NMS. But it is also enforced through the degree of standardization that exists at different levels of the network. Up to the DM level most interfaces and parameters are highly proprietary and only little is standardized. Above domain level a higher degree of standardization is applied, which theoretically allows to control the DMs from different vendors up to a certain level through a single multi vendor NMS. But, due to the high degree of proprietary interfaces and non-standardized functions and also different levels of available functionality at NE level, it is often not possible to fully configure

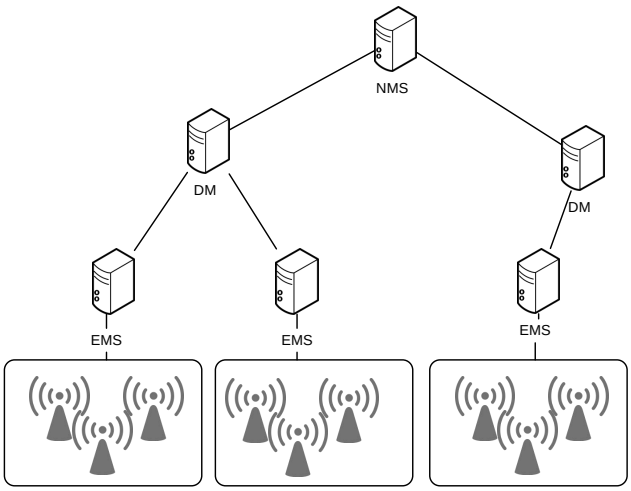


Fig. 2.4: Management Architecture of 3GPP based Mobile Communication Networks

all aspects of the network through the NMS. Thus, additional vendor specific management systems and tools have to be used to perform these vendor specific configuration steps. In addition, not all vendors provide the means to configure network elements for different RATs through the same management infrastructure, which then requires the usage of different management systems within a single vendor domain.

A very tight alignment of the configurations throughout a mobile network is required to provide the expected seamless user-experience and very high levels of service quality. It is not sufficient to provide this close alignment only within the individual vendor domains or different deployed RATs. A mobile operator has to assure seamless operation within and between different vendor domains as well as between different RANs, which creates a lot of interdependencies between the configurations of different NEs. For example, the configurations of NEs that are responsible for the operation of neighboring cells, independent from vendor and RAT have to be adjusted to support vertical and horizontal handovers. This results in a very high configuration complexity which is further increased with the introduction of HetNets. While in traditional deployments the configuration of NEs from different vendors only have to be aligned at the vendor cloud boundaries or between different RATs this changes drastically in HetNet deployments. In these future deployment scenarios NEs from different vendors will be deployed that serve cells of the same RAT within the same area. This basically means, there is a vendor domain border at most of the deployed cells.

In common networks this overall high configuration complexity has to be handled through multiple separate management systems with no or only minimal multivendor support. This is one of the main reasons making mobile

network management that labor respectively cost intensive and highly error prone.

2.2.2 Generic / Traditional Mobile Network Management

In the previous section an insight into the overall mobile network management architecture was provided and it was shown how multi RAT and multi vendor deployments contribute to the very high configuration complexity.

This Section provides a deeper knowledge about the principles of mobile network management to give a baseline for the understanding of the necessity for network management automation in general and SON based management in particular.

The management of mobile networks follows the classical Monitor, Analyze, Plan, Execute cycle. Performance measurement data and network event traces are analyzed to detect suboptimal (PM) or erroneous (FM) behavior, depending on the discovered situation countermeasures are planned and subsequently executed. Compared to other communication networks, the management of wireless mobile communication networks shows some particularities, which are partly caused by their size and complexity.

In order to be able to perform network management, to operate a network, detect failures and tune the configuration for optimized performance it is important to know the state of the overall network and its mostly several thousand individual network elements. Each of the NEs provides hundreds or even thousands of managed objects and counters which are used to configure it or represent its state.

The most important input for the network management processes is PM data. Extensive numbers of performance measurements have to be collected, processed and analysed. They are used as input to assess the current state of the network and detect failures and unwanted behavior. They are also used together with additional information, like current and historical configurations, dependencies between NEs and network behavior, to derive new configuration settings.

Instead of the raw performance measurement values mostly KPIs and Key Quality Indicators (KQIs) are used, which aggregate individual measurement values into more meaningful abstract values. The handover success rate KPI, which is used to rate the handover performance in a network, is constructed from a large set of different performance measurement values. Each possible cause for a handover failure is reflected by a certain counter. This detailed information is irrelevant for most of the cases, and therefore the aggregate KPI is used. 3GPP specifications define a set of required KPI values that have to be provided, which is mostly extended by a larger number of vendor specific KPIs. The specifications [3GP09g, 3GP09e, 3GP11f, 3GP09f, 3GP11h, 3GP11g, 3GP09d] give an impression of the large number of available KPIs. It is noteworthy that each of the KPIs is combined

from a set of performance measurement values which are provided through individual counters at NE level.

To speed up the management processes some of the analyses, mainly failure and error detection based on KPI threshold analysis, are done automatically. Data is aggregated and alarms are raised to indicate missbehavior whenever predefined thresholds for particular KPI values are crossed. The procedure is identical for FM and PM. Analysis and interpretation of the data in order to track down the root cause for an observed behavior is complex and therefore usually done by the human operators, who put the available data into context. When the detected situation is identified and decisions on appropriate countermeasures are taken the resulting reconfiguration requests are enforced through the respective CM systems.

2.2.2.1 Timing in Mobile Network Management

A mobile network consists of several thousand NEs. At each of the NEs up to several thousand performance counters are maintained to reflect the operating point. These performance counters are then, at NM level, aggregated and combined into KPIs and KQIs. Up to now there is no comprehensive real-time access to performance data at the NMS but the data is provided at the end of predefined time intervals, so called Granularity Periods (GPs), for two basic reasons:

- **Management Link Capacity Limitations:** From the beginning of mobile networks the backhaul link capacity from the base stations to the core network has always been limited, especially for the transmission of performance data. The main focus of the backhaul obviously lies on the transmission of user traffic. For a GSM base station usually only a 64 kBit/s share of the link capacity was available for management data. Today with the availability of multi MBit/s or even GBit/s backhaul connections the available data rate for management data has also been increased but is still limited.

- **Statistical Relevance:** Another reason for the usage of GPs is the need for statistically significant performance measurements. An example is the monitoring of the handover failure rate between two cells. Whenever the configuration is adjusted to optimize the handover behavior, this configuration change has an immediate effect on the handover performance. But to reliably assess whether the new configuration shows the expected effects it is not sufficient to base the analysis on a single or a small number of samples. For a meaningful handover failure rate information collected over a predefined time period is required.

2.2.2.1.1 Granularity Periods are used to handle as well the link capacity limitations as also the data collection requirements.

In order to cope with the link limitations the performance measurement values have to be prioritized and assigned to so called GPs. A GP defines a time interval during which a performance counter is maintained at NE level and at the end of which the data is uploaded to the NMS [3GP05]. The 3GPP lists examples for the duration of GPs as 5, 10, 15 or 30 minutes as well as multiples of an hour [3GP05]. Depending on the importance of a particular performance measurement the assigned GP is longer or shorter, and it is therefore more or less frequently updated at NM level. The beginning of the GPs is synchronized throughout the network, for example, all start at the full hour. This is done to allow the network operator to get a consistent view on the state of the network, through synchronous availability of particular performance measurements from all NEs.

The assigned GP is not only determined by the importance of the particular performance measurement value but also by the time required to collect a statistically significant number of samples. It is almost impossible to base the requirements for the statistical relevance on absolute numbers of samples. The number of available samples is highly dependent on the number of users and the overall traffic pattern for a particular cell. For cells in a high traffic area the required number of samples could be collected in a very short time, in other areas of the network it would take much longer. Instead of absolute numbers, specific GPs are used which are selected based on operational experience.

The management of mobile networks is strongly influenced through the GPs as they determine the availability of data that is required to analyze the behavior and overall performance of the network. Successful analysis at NM level is only possible when all required data is available.

One of the major drawbacks is the delay that is induced through the GPs. Situations that are characterized through performance measurement values which are collected over very long time intervals can only be detected at the end of the particular GP. Hence, performance degradations, outages or system failures might remain undetected for a long time. The same applies for the verification of configuration changes. In the best case, information about the reconfiguration effects are available at the end of the next GP. But, since error detection, root cause analysis, and finding and enforcing counter measures can be time-consuming there is a possibility that the effects become not visible before the end of the next but one GP. The reason is the time period that is required to reach statistical relevance of the collected data. Due to the delay until the enforcement of the configuration changes, the remaining time in the current GP is not sufficient to collect enough samples. In that case, the performance measurements that are aggregated at the end of the GP do not fully reflect the current state of the network.

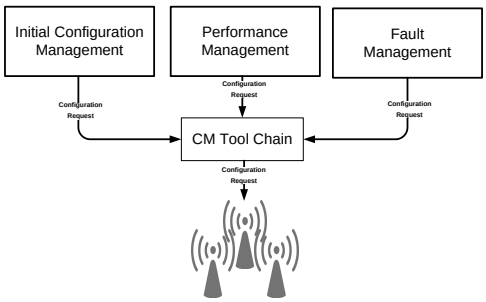


Fig. 2.5: Initial CM, FM and PM use the same CM Toochain

2.2.2.2 Management Tasks Interdependencies

Due to the complexity of the networks and the large number of associated management tasks, the different network management areas are often considered separately. In general, initial deployment and CM [3GP10a] is done separately from network PM [3GP05] and optimization which is performed independently from FM [3GP11] and failure recovery. Although the management areas are treated separately, the performed management tasks can have strong effects on each other across all management areas. It is important to note, that the configuration tasks requested by initial CM, PM and FM are performed through the network wide CM toolchain as shown in Figure 2.5.

For example the introduction of a new cell in a network needs an initial configuration of the base station that provides the cell. The new cell has an impact on the performance of the surrounding cells, which will mandate an optimization of their configurations. Another example is the resolution of a coverage hole that was created through the failure of a base station and its associated cells. The immediate means that are used to compensate a cell failure is to increase the coverage area of the surrounding cells, which has an effect on several KPIs. Through the increased distance between the served User Equipments (UEs) and the serving antenna the measured Channel Quality Indicator (CQI) can be reduced as well as the average Throughput per Resource Block. In general the load of the remaining cells will be increased as they have to serve more users. These KPI changes can cause alarms that trigger optimization actions. A simple way to reduce the load and improve the CQI values is to reduce the size of an overloaded cell. It is obvious that such interdependencies require some sort of coordination between PM and FM. It is an important part of the OAM for the operational staff to know how to handle such situations, for example by notifying the PM people about an ongoing failure recovery so that they can ignore an indicated overload situation.

There is a set of reasons that can cause such interdependencies. The following listing gives a short introduction on the reasons, a more detailed discussion is part of the conflict analysis in Chapter 5.

- **Shared Configuration Parameters:** Multiple tasks require the modification of the identical configuration parameters
- **Shared Input Values:** Several configuration tasks use the same Input values, but have different objectives
- **Impact on Input values:** The actions performed for one management task affect the input values of other management tasks. That was shown above where the resolution of a coverage hole triggered the optimization which can cause a coverage hole.

It is impossible to separate the management tasks in a way that resolves all interdependencies, therefore each tasks that is performed can implicitly raise subsequent alarms and trigger opposing actions. Thus, detailed operational experiences and a good overview on the currently ongoing management processes is required to handle management task interdependencies and avoid the negative effects they can cause. Due to the timing in mobile network management and the usage of GPs, an operator has to be aware of the fact that the performance measurement results that are provided at the end of GP potentially do not reflect the current state of the network. Configuration changes that have been performed within the last GP might not have shown all their effects. That means not only the configuration changes performed in currently ongoing processes are of interest but also those that have been performed in the past.

2.2.2.3 Summary Network Management

As shown in the previous sections, the management of wireless mobile networks is very complex. There are several reasons for this large complexity. The networks consist of thousands of network elements from different vendors with only partially standardized and otherwise highly proprietary management interfaces. Multiple technologies and network generations are operated in parallel with the requirement to provide a seamless inter-technology operation which mandates a high degree of configuration alignment. The configuration is done via a large number of configuration parameters, a single reconfiguration can have multiple side-effects that need to be considered. There is a tremendous amount of network performance measurements that can be used to assess the state of the network which has to be aggregated and processed.

It is obvious that the management paradigms with its architecture, processes and interfaces which were introduced with the specification of the GSM networks and then evolved over time are not sufficient anymore. Especially now, when the fourth network generation is deployed which introduces even more complexity through radically changed RAN and CM architecture. In addition mobile network operators experience the pressure to reduce their OPEX and at the same time increase the overall service quality to meet customer expectations.

3. RELATED WORK

This chapter provides an overview of related work, especially on the coordination framework of the SOCRATES [Sch08] project, the results of other relevant research projects as AUTOI, UniverSelf and EFIPSANS and in addition on important work in the area of autonomic computing and autonomic networking.

3.1 SOCRATES

SOCRATES (Self-Optimisation and self-ConfiguRATion in wirelEss networkS) was a pan-European FP7 project, which aimed to enhance the operation of wireless access networks through the introduction of self-organization methods into network planning, configuration and optimization. All studies within the project were done based on the 3GPP LTE radio interface and the NGMN list of SON use cases [Leh07a]. The most important contribution was the focus on the integration of all self-organization methods into a single framework, independent from their particular management area.

The work performed within SOCRATES is of special interest as it, for the first time, identified the need of coordination to prevent conflicting behavior between SON-Function instances with negative effects on the network operation. It also provides a first categorization of SON-Function conflict types [KAB+10]. Based on their findings the project developed a generic coordination concept including an architecture.

[Ban11] provides a detailed analysis of the SOCRATES coordination framework, therefore only a summary of the results is given here.

The development of the coordination concept and also the framework was driven by the goal to ensure a joint operation of all deployed SON-Functions towards high level operational goals. To reach these goals the framework aims to provide capabilities to detect undesired behavior and harmonize the operations of the SON-Function instances.

SOCRATES deliverables [SZSS10] provide, in detail information on nine SON use cases that have been selected as representative SON-Functions, their implementation and behavior. SOCRATES provides no generic definition of a SON-Function, their anticipated structure, or required common behavior.

The harmonization itself is separated into two phases:

- Heading Harmonization: Design-time harmonization provides non-conflicting SON-Functions to avoid potential conflicts.
- Tailing Harmonization: Run-time conflict detection and resolution based on the configuration requests of the SON-Functions.

The coordination is based on two main building blocks, a set of so called policies and a set of functions that form the coordinators architecture. Policies are used to transform high level operational goals in a way that they can be enforced during the harmonization process. The understanding of policies in SOCRATES [AEL+11] shows a difference to the more common policy understanding as defined for example in [Str04]. But the way how high level Operator Policies are stepwise refined into SON-Function or SON coordinator specific policies resembles in many ways the well known policy continuum introduced in [Str04].

In the SON coordinator, the coordination process is performed by four functions. Policy, Alignment, Guard, and Autognostic function jointly contribute to the tailing coordination. The main drawback of the proposed architecture and its functions is the replication of identical functionality in multiple functions, which increases the overall complexity and will complicate the maintenance of the system when new SON-Functions are introduced. For example the estimation of an effect caused by a configuration change is done in the SON-Functions but also within the Alignment function. Thus, each introduction of a new SON-Function requires an update of the Alignment function. This issue has also been described in [RH11].

The SOCRATES project did not provide a proof-of-concept implementation or simulations of the complete coordination concept nor a detailed description of the coordination process itself. The studies that were performed [AEL+11] included only two SON-Functions which directly contained the functionality of the Autognostic function as well as the policies for the Alignment function. The implementation of the SON coordinator provided only small parts of the Alignment function. Autognostic, Policy and Guard function were not implemented at all.

The work of the SOCRATES project provided some important results for the area of SON-Function coordination but still leaves open questions. The missing implementation together with the complexity of the concept, especially due to the redundant distribution of functionality and the policy refinement process, leaves doubts on the scalability and maintainability of the proposed SON coordinator concept. The provided studies do not provide sufficient support to resolve the doubts that the proposed SON Coordinator will be able to perform the anticipated detection and resolution of SON-Function conflicts. No studies have been performed on the complexity of the policy refinement process, which is a central part of the overall concept.

The SOCRATES coordination concept focuses on the resolution of conflicts that are caused by parallel reconfiguration requests but provides no means to align a request to previously performed configuration change. In addition there are also no means to assure the validity of used input data. For example to prevent the enforcement of configurations that were computed on input data that does not yet reflect the changes performed by a previous SON-Function.

3.2 AUTOI

Autonomic Internet (AutoI) was a EU funded research project targeting a future Internet of Networks and Service as a optimized network and service solution.

The main problem identified by the project is that the internet as it is today shows to a large extend a high degree of ossification. The Internet was always seen as a way to interconnect intelligent end systems, thus being a simple network service and all complexity is pushed to the end systems. The result is a relatively inflexible system with no or only little built-in support for:

- Services and their management
- Network management
- QoS and security features
- Orchestration of different aspects of the network layers to support the demands for new services

As a solution the project proposes an autonomous overlay over the existing networks. Not only the internet but all kinds of wired and wireless networks, thus covering a larger number of heterogeneous networks. This overlay provides different means to support the deployment of new services with specific demands. For example the setup of VPN connections with guaranteed network characteristics including support for vertical handovers between different access networks for mobile nodes.

Different techniques are combined to be able to provide this autonomic overlay. The main contribution is a central management system consisting of five different planes [ABB+08] (Orchestration, Service Enable, Knowledge, Management, Virtualization) which uses a multitude of techniques to provide the required features. This central entity communicates with different Autonomic Management Systems in the networks that are used in the underlay. It has all knowledge about how the capabilities of these systems have to be combined in order to satisfy the demands of the new service.

It uses for example a subset of the DENng modelling language citeStrassner to enable a model based translator to provide device specific configurations from generic AutoI configuration files. Policy refinement according to the principle of the policy continuum is used to allow decision taking at system level based on abstract operator policies. Together with the information

provided from the Knowledge Plane, an autonomic policy system is used to support the orchestration process or trigger specific actions whenever the context of a consumer changes and an adaptation of the provided virtual network is required.

The AutoI concept [RLS10] uses a multi round negotiation approach to determine a solution that satisfies the requested network characteristics and at the same time can be supported by all used subsystems in the underlay. To perform these negotiations the Orchestration Plane has a market place where each participating subsystem registers its capabilities. The negotiation approach combines the offered capabilities in a way that all requirements are satisfied. From a conceptual point of view it has several similarities with the conflict detection and prevention approach that has been proposed within the SOCRATES project, although it has not explicitly designed to resolve conflicts. What makes the initial approach of AutoI interesting is that it shows that different concepts from the area of autonomic computing, the semantic web and policy based network management can be successfully combined to create a network management system for end-to-end use cases.

3.3 UniverSelf

UniverSelf [Pro] is a research project funded by the European Union under the 7th Framework Programme for Research. The main goal of the project is to target the growing management complexity of future networks. The basics of the project lie in the general autonomics research of the past years but the project has the objective to bring the self-management approach a big step forward.

It aims to provide a management framework, which is applicable to both, existing and future autonomic management architectures. All functionality that is required to enable self-management will be designed in a way that it can be embedded directly into the equipment that forms the network infrastructure.

Based on the analysis of the different use cases the project identified the need for the coordination of SON-Functions. Different coordination strategies are proposed [GBK11] which aim to avoid the negative impact of SON-Function conflicts. Coordination is not performed by a central entity but is inherently performed through the interacting optimization processes. Two different basic approaches are introduced:

- **Coordination by Information:** Active optimization processes inform other processes about their activity and therefore prevent conflicting actions
- **Coordination by Control:** This approach is based on the assignment of priority to the optimization processes. A process with higher priority is able to overrule or block the actions performed by optimization processes with a lower priority. It is also possible that only

a limited parameter range is allowed by the higher priority process.

Apart from this active coordination [GBK11] proposes also to prevent conflicts by separating the optimization processes. For example processes with different timescales (e.g. optimization on a monthly vs. an hourly basis) are considered a priori separated and therefore not conflicting. In case of identical timescales the execution of these optimization processes should be performed at the same point in time to make conflicting behavior observable. Several constraints need to be considered to determine the timescale of an optimization process.

A first case study presented by the UniverSelf Project [Uni12] states that coordination and conflict resolution of SON functionalities is a key enabler for the introduction of SON based network operation. The research questions raised in the study motivate the need for coordination and conflict resolution in the danger of conflicting operational targets and the existence of high level operational goals that autonomic optimization processes have to follow.

The coordination and conflict avoidance approaches proposed by the UniverSelf project aim at resolving conflicts either on a timescale basis or by empowering the optimization processes to solve the conflicts through direct interaction. This approach requires that the optimization processes are aware of all potentially conflicting processes that are active in a shared target area. It also requires a high level of simultaneity for the execution of optimization processes, which puts additional constraints on the processes. There is no information provided, if information about executed configuration changes is somehow maintained in order to be provided to other optimization processes, or if the optimization processes are somehow in an always on state, allowing the persistent storage of the required information. A last difficulty that has to be solved is that the presented concepts require some sort of common interface what makes it hard to include SON-Functions from 3rd parties.

At the time of writing, the case studies are not yet completed, therefore it is possible that the open questions are targeted in the final versions.

3.4 EFIPSANS

The EU funded EFIPSANS (**E**xposing the **F**eatures in **IP** version **S** ix protocols that can be exploited/extended for the purposes of designing/building **A**utonomic **N**etworks and **S**ervices) project [EFI] was dedicated to research within the Future Internet and IPv6 area. Its main goal was to provide the basis for autonomic networks and systems using IPv6 based networks.

The main idea was to study emerging research areas and projects with a focus on autonomics to identify a catalogue of desirable autonomic behavior within different systems. Based on this information respective features

within the IPv6 protocol family should be exposed and used to build autonomic networks. Wherever required functionality was missing, it should be provided as extension to the IPv6 standards.

Although the main topic was relatively far away from Self-Organizing Networks, the results of the research have a certain relevance.

There was a lot of research performed in the area of autonomic Radio Resource Management [KKP08a, KKP08b, KTP08, KP09] especially on different approaches for power and rate control. These results can be used for the development of individual SON-Functions.

The project developed also a theoretic approach for the design of Self-Managing entities which was combined within a single reference model called: Generic Autonomic Networking Architecture (GANA) [Cha10]. The core idea is to use this reference model to establish autonomic behavior on different system levels. From Level 1 (Protocol Level) up to Level 4 (Network Level). At each of the level there are control loops containing managed entities and decision elements which are influence by higher level objectives. It is even possible to build up a hierarchy that spans multiple levels.

This architecture is generic enough to be instantiated for different network types and their particular applications domains [ea09a], for example, for traffic monitoring and shaping in fixed networks as well as for QoS and mobility management in heterogeneous wireless networks or auto configuration in Mobile Ad-Hoc Networks.

Based on the GANA architecture Tcholtchev and Chaparadza [TC10] propose an autonomic fault management system and discuss its limitations. They highlight the need to still have the operator in the loop to handle those management tasks an autonomic system can for whatever not handle, for example, the handling of unknown incidents.

With respect to Self-Organizing networks, the project provides important insights. Many details of the approaches developed by the EFIPSANS project can be used as an direct input to the development of SON-Functions. Especially the need to have a clear separation of functionality and the possibility to handle unforeseen situations by escalating them to human operators. But also the solutions presented that are directly applicable to problems in mobile network management.

3.5 Autonomic Computing

In 2001 IBM research layed the foundation of all approaches that try to make systems more autonomic. Paul Horn published a manifesto [Hor01] to show IBMs view of the state of IT systems. The manifesto highlights the benefits that come with the automation of processes but also the drawbacks from the increasing complexity, especially with regard to an increased management complexity that becomes harder and harder to handle. It envisions self-governing autonomic systems that adapt their configuration to changing

conditions in their environment and continuously try to optimize themselves following high level goals or Service Level Aggreements (SLAs). IBM research continued the work in the area of autonomic computing and in 2003 presented their vision of Autonomic Computing [KC03b] which was complemented by an architectural blue print for autonomic computing [C+06]. IBM presented the four main aspects of Self-Management that became known as Self-* aspects: Self-configuration, -optimization, -healing and protection. In a next step five evolutionary stages of self-management processes were introduced, in each step the processes become more autonomic until they reach the fully autonomic level where the role of the humans is restricted on the definition of high level operational goals. Another important concept that was introduced by the work of IBM research is the MAPE loop (cf. 3.1) that describes the operation of an autonomic system.

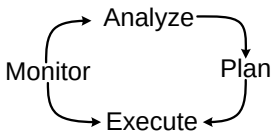


Fig. 3.1: MAPE Loop

The systems **monitors** its environment and performance in order to **analyze** whether it still conforms with the operational guidelines or if counteractive measures have to be taken. In that case it will **plan** a set of actions that are required to bring the behavior in compliance with the operational guidelines. These actions are subsequently **executed**. After the execution the loop is at its beginning and the monitoring phase is used to collect data that can be used to determine the results of the performed changes.

The work of IBM and the Autonomic Computing Initiative (ACI) provided the basic terms and concepts for all management automation approaches and all approaches that try to introduce advanced autonomic behavior into systems.

3.6 Autonomic Networking

Autonomic networking is based on similar principles as Autonomic Computing. Its main goal is to develop communication architectures with NEs and network protocols which autonomically adapt their operation to the changing environment. Each NE senses the environment and then, based on a MAPE loop tries to optimize its own behavior.For a successful autonomic operation each NE needs to maintain a dedicated knowledge base. Due to the very complex interdependencies in a network where a single configuration change on an NE can negatively affect multiple other NEs a joint cooperation of all autonomic NEs is required. One important aspect of this

joint operation is the exchange of knowledge to coordinate the autonomic operation.

3.6.1 Autonomic Network Management

Autonomic Network Management aims to provide solutions that enable a self-management of the network, where the interaction of the human operators with the network is limited to the provisioning of high level operational goals. The components of the systems will use autonomic and self-* concepts to assure a network operation in compliance with the operational goals.

One of the most important aspects of the introduction of Autonomic Network Management is a stepwise introduction of higher levels of autonomic behavior to assure the operators confidence in the performance of autonomic management. Similar to the five stages of autonomic computing, a first step will be the introduction of individual autonomic functionalities which are then gradually extended until full autonomic behavior is reached. It is important that even at the highest level of autonomic behavior, the system is still under full operator control. Another aspect is the requirement of a stepwise and smooth transition from manual to autonomic management that is performed along with the introduction of higher level of autonomic behavior.

There has been a major research focus on the introduction of semantically enhanced data and information models that are required to provide a consistent and foremost vendor independent view on the network [DJ07, DJS08]. This information is required to allow the automatic mapping of abstract operational goals into device specific configurations and policies.

3.7 Bio Inspired Networking

There are some approaches that use the behavior of living things as a blueprint for the operation of a network [BBD+06b, BBD+06a, DA10]. There are complex systems in nature that show a highly efficient behavior without any kind of centralized control. For example ant hives, the movement of shoals or processes within the human body. Analysis of their behavior showed that it is sufficient if each individual follows a simple set of rules. Based on these rules the overall system quickly reaches an operational equilibrium and continuously contributes to maintain it. The idea is to use these bio inspired self-organizing principles in order to optimize the network operation. Each NE follows a simple set of rules that creates and maintains an optimized operation. There are two main problems with this approach. On the one hand it is not sure if the equilibrium actually represents the anticipated state of operation, especially because each NE can only take decisions based on local knowledge. It is difficult for a network operator to shift capacities from one area into another to cope with a

temporarily high demand if the bio inspired optimization targets an equal capacity distribution. On the other hand, if the approach is used to optimize different, potentially conflicting requirements. The equilibrium that results from this multi target optimization can represent a state which is acceptable for the individual aspects but be far from the reachable optimum for a higher priority goal.

The basic ideas introduced by Bio Inspired Networking are an important contribution to the further development of SON but the applicability of the concrete technologies is debatable.

3.8 Policy Based Network Management

The usage of policies for the operation of networks and especially in network management emerged from the area of access control. In the beginning access to a resource was defined through an access control matrix, which contained information about the accessrights for each individual. Such an approach will reach its limits as soon as the number of individuals is growing. In order to be able to cope with a large number of individuals a characteristic of the access matrix was exploited. Many of the individuals were assigned identical access rights, therefore it was possible to assign the same rights to a group of individuals. Role Based Access Control (RBAC) [FK92] is the approach that manages access rights based on roles which are assigned to individuals. From these basis policies developed into a management paradigm where resources which were no longer only used to manage access rights but to efficiently manage resources in large scale distributed systems. The usage of SLAs in networks and thus the need to enforce Quality of Service (QoS) guarantees lead to the usage of policies in communication networks [FTP02, SRS+03]. The usage of cross-organizational SLAs, for example to give end-to-end service quality guarantees brought up the need for a standardized policy framework. The Internet Engineering Task Force (IETF) and the Distributed Management Task Force (DMTF) provided a set of standards, Request for Comments (RFC) and protocols with a strong focus of the application of policies in the network management domain [DBC+00, YPG00, MESW01, WSS+01, Moo03, SRS+03]. The standardized framework introduced a reference model with different entities which are assigned dedicated tasks:

- Policy Repository to store policies
- Policy Decision Point (PDP) which takes decisions based on the provided policies and the state of the network
- Policy Enforcement Point (PEP) which is responsible to enforce the decisions for example by assigning traffic classes or rejecting network access

The initial focus of policies in network management on access control and QoS shifted gradually, and policies became a more versatile tool. The great benefit of policies is the possibility to easily change the system behavior at run-time, which makes it interesting also for network management [Ver02, Wie94]. A network operator can influence the behavior of the system through an adaptation of the policies or by assigning a different role to a group of users. Today the usage of a policy based network management approach can be used to allow a business driven operation of the network, where business rules are used to specify technical policies that are used to take the decisions on network level [Wie94, Str04]. Research tries to use techniques from the semantic web to bridge the gap between the abstract specification of business policies and the the required level of detail for the policies at the network domain[DJ07, DJS08, TK07]. Another important area of research within the policy community is the detection and resolution of conflicts between policies [LS99]. The policy systems today already use techniques to prevent policy conflicts [RDD07], but also in this area modelling and semantic web techniques are used to detect the conflicts already at design-time [DJ07].

For the development of SON the usage of techniques from Policy Based Network Management are an important part. Policies provide a basic building block for the automation of single tasks and larger processes, especially through the capability to take context specific decisions. Equally important is their deterministic behavior, which is a key requirement for the usage in mobile communication networks.

Part II

SON-FUNCTION COORDINATION CONCEPT

The following chapters present the conceptual approach to conflict-free SON-Function execution that follows the overall operational guidelines of a specific network. They present the requirements on the design of SON-Functions that are necessary to perform an efficient and vendor independent coordination of SON-Functions. A classification of SON-Function conflicts is an important input to the design process of the individual SON-Functions and their respective coordination logic is also introduced. Before the presentation of the coordination concept and different variations of the coordination process, spatial and temporal characteristics of the SON-Function instances are introduced, which are important for the coordination process.

The last part shows the SON-Function design process that was derived from the findings made. It is highly recommended to follow the process, as it will not only provide the SON-Function itself in the recommended structure but it will also provide information on the temporal requirements and the coordination logic is required to prevent and resolve SON-Function conflicts.

4. CONCEPTUAL VIEW ON SON-FUNCTIONS

In general, a SON-Function is the implementation of the functionalities specified by a particular SON use case. For example the Automated Neighbor Relationship Assessment (ANR) SON-Function is the implementation of the ANR use case as specified by the 3GPP [3GP09c]. The SON-Function is deployed into the network and whenever its triggering situation is detected it will perform its particular tasks. In case of the ANR function it will add a neighborship to the Neighbor Relation Table (NRT).

From the already large number of SON use cases that have been specified [Leh07a, Leh07b] it becomes obvious that there will be a large number of deployed SON-Functions. In addition to the standardized functions there will also be vendor and operator specific SON-Functions as well as vendor specific implementations of the standardized SON-Functions.

In order to be able to provide an efficient and effective SON-Function coordination for all deployed SON-Functions the usage of a basic scheme that separates the functionality of the SON-Functions into three parts as shown in Figure 4.1 is proposed.

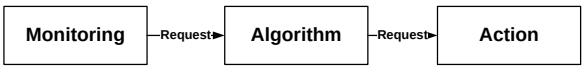


Fig. 4.1: Generic SON-Function scheme

The applied separation of concerns resembles the well known Monitor-Analyze-Plan-Execute (MAPE) loop but directly reflects the way the SON use cases have been specified by the NGMN Alliance. The relation is explained in Table 4.1. However, the separation of the functionality has to be flexible enough to perform it for any SON-Function. It has to provide the function designer with the possibility to shift the functionality within certain limits to different parts of the SON-Function in order to map the functionality in a way that it meets the requirements. For example, to perform the PM data analysis already completely within the Monitoring-part, or to restrict the functionality of the Monitoring-part to receive a triggering event and then immediately trigger the Algorithm-part that performs an extensive analysis of the PM data.

Tab. 4.1: Relation between MAPE, SON-Function scheme and use case specification

MAPE Phase	SON-Function Part	Task to be performed
Monitoring	Monitoring	Detect triggering situation
Analyse and Plan	Algorithm	Analyze available data and determine reaction (compute configuration change)
Execute	Action	Enforce the configuration changes

Each SON-Function is seen as an independent entity in the network, which operates autonomously without direct connections to other SON-Functions. Additional connections to other SON-Functions, for example, through communication and interaction capabilities, will introduce another dimension of complexity that complicates the maintenance of a system with a large number of concurrently deployed and active SON-Functions. Each introduction or removal of a SON-Function will then require the adaptation of all functions to which it has a relationship.

A SON-Function will not explicitly trigger a subsequent SON-Function but implicitly through an event message that notifies the system about the results or the failure to provide a resolution that resolves the detected situation. It is an intentional design decision to avoid direct interrelations between SON-Functions. A network operator thereby has the possibility to deploy and modify the responding SON-Functions without the need to update the functions that send the request. It also allows the operator to flexibly change the system behavior. SON-Functions that respond to an event can either be deactivated and the notifications directly forwarded to the operator or the operator can choose to be notified in parallel. It is also possible to deploy different SON-Functions that react on a combination of a notification event and additional context information.

SON-Functions are fully operational as soon as they are deployed within the network and have the capability to interact with the NEs. That does though, not mean that they are actively performing any tasks. Even a SON-Function that is waiting for the availability of performance data or a particular alarm message is operational. Non-operational SON-Functions can not interact with the network at all, they will not process input data, execute their Algorithm-part or perform actions in the network.

4.1 Monitoring-part of a SON-Function

The Monitoring-part is the initial part of the SON-Function. Its main task is to detect the SON-Function triggering situations. For many SON use cases the triggering situations are already described in the NGMN use case definitions [Leh07b, Leh07a].

Triggering situations are characterized through specific events or KPI values or through combinations of KPI values and events. A typical way to describe a triggering situation is to define thresholds for KPI values that may not be crossed. The Monitoring-part of each SON-Function receives and evaluates the specific input data, whenever it indicates the presence of a triggering situation, the Monitoring-part will trigger the Algorithm-part of the SON-Function.

Depending on the triggering situation the tasks that have to be performed and the amount of data that needs to be analyzed can strongly differ. For some functions the Monitoring-part performs an extensive trend analysis for a set of KPIs or only a basic comparison of the observed KPI values with predefined thresholds.

The trigger that requests the algorithm execution of a SON-Function can contain different amounts of information. From a very basic trigger that only indicates the need for the algorithm execution to the full results of the performed analysis. The minimal required information is the area where the triggering situation has been detected. This spatial information can refer to a geographical area or also a more abstract spatial description as a network domain or a single or a set of NEs.

As shown in Figure 4.1 there is only a uni-directional relationship between the Monitoring-part and the rest of the SON-Function. There is no explicit feedback loop from the subsequent parts of the SON-Function back to the Monitoring-part. A SON-Function therefore has no information if the requested algorithm execution has been performed and if it resulted in actions.

If a triggering situation is not resolved by the requested algorithm and the subsequent Action-part the Monitoring-part will re-detect the situation and then re-trigger the algorithm execution until the situation is resolved. The operation of the Monitoring-part has therefore to be designed in a way that the detected issues can be resolved before it re-requests another algorithm execution. It makes no sense if the algorithm execution is requested every five seconds, although the algorithm and action execution will take at least several minutes to run.

This requirement shows the dependency of the Monitoring-part on valid input data. If the used data is affected by other SON-Functions or not yet affected by a previously triggered Algorithm-part a non-existing triggering situation could be detected and unnecessary countermeasures could be requested.

4.1.1 Activity Schemes

In a future SON a large number of SON-Functions will be deployed that cover almost any aspect of network management, from fine grain, low level optimization to large scale recovery and auto-configuration. Depending on their specific tasks different modes of operation need to be supported, some SON-Functions react on individual alarm events, others perform data analysis at predefined points in time in order to evaluate the state of the network and detect their triggering situations. Another class of SON-Functions will need to continuously monitor available performance data and track the changes over a long time period, in order to detect misbehavior from the observed KPI trends.

The presented generic SON-Function concept supports all of these activity schemes. The Monitoring-part is defined as the part of the SON-Function that is used to realize the respective activity scheme. Three different activity schemes are defined here that can be assigned to the Monitoring-part of a SON-Function.

- **On demand:** The Monitoring-part receives an explicit triggering event, for example, a particular alarm message or another notification. Usually it will not perform further monitoring tasks but directly trigger the algorithm execution. However, it is possible it will not directly trigger the Action-part, but first evaluates the network context to detect a potential triggering situation. For example, if one SON-Function performs changes that can have negative effects on the network the Monitoring-parts of additional SON-Functions can be triggered. This guarantees a minimal response time to treat those negative effects.
- **Timed:** Some tasks have to be performed at certain points in time, for example general optimizations are done repeatedly in fixed time intervals. Therefore, the Monitoring-part of the SON-Functions can be requested to evaluate the network context at predefined times, and if required, optimize the configuration.
- **Continuous:** This is probably the most common activity mode for SON-Functions. The Monitoring-part will always be active and evaluate available data. In case it detects its triggering situation it will initiate the SON-Function Algorithm-part to resolve the detected situation.

Which activity scheme is used for a particular SON-Function is, as described, highly dependent on the tasks that it should perform. Therefore the selection and assignment of an appropriate activity scheme has to be performed as part of the SON-Function design process.

4.2 Algorithm-part of a SON-Function

The Algorithm-part is the main part of a SON-Function. It is triggered by the Monitoring-part and is responsible for the computation of the actions that are required to resolve the triggering situation. Similar to the Monitoring-part, the functionality of the Algorithm-part can strongly vary. From a single step that directly transforms the input from the monitoring into actions to extensive, long-running analysis based on state and configuration of the network.

The input used by the Algorithm-part can be restricted to the input provided by the monitoring function, but depending on the requirements of the performed computations can also include data from multiple data sources within the network. That means the Algorithm-part of a SON-Function can have the capabilities to acquire additional data, for example, additional performance measurement values or the current configuration of NEs as well as historical configuration and performance data, or additional context information. The Algorithm-part has also the possibility to trigger supporting SON-Functions to retrieve additional information.

It is obvious that the Algorithm-part requires valid input data, which is not affected by other SON-Functions or manual reconfigurations. This is not only important if new configuration parameter values are derived from current configuration parameter values but also in case performance measurements are used. If the detected triggering situation is not caused by the evaluated configuration parameter values or the used performance measurement values do not actually reflect the current situation, the requested countermeasures can deteriorate instead of resolve the detected triggering situation.

The Algorithm-part uses its input to compute new configuration parameter settings in order to resolve the detected triggering situation. For example, the Cell-Outage Compensation (COC) SON-Function evaluates the effects of a cell outage based on the available performance measurements and cell planning data. It computes new configuration parameter values that increase the coverage area of the cells surrounding the cell in outage to compensate the lost network coverage.

The Algorithm-part of the SON-Function will not enforce the computed configurations but will trigger the Action-part of the SON-Function and provide the configurations as input together with the trigger. In case the Algorithm-part is not able not find a resolution for the detected situation it can either request no action or escalate the responsibility by issuing an appropriate alarm message via the Action-part, towards the human operator.

4.3 Action-part of a SON-Function

The responsibility of the Action-part of a SON-Function is to enforce the configuration parameter settings that were computed by the Algorithm-part or notify the system about the algorithm results.

The Algorithm-part provides all input that is required by the Action-part, as for example the target NEs and the computed configurations. Depending on the setup of the network and the SON-Functions the Action-part either has the capability to interact directly with the targeted NEs or employ the networks CM toolchain to enforce the configuration changes. The time required to perform the reconfiguration can vary depending on the available reconfiguration means. Using the CM system to reconfigure the NEs can cause longer delays until the changes are propagated to the target NE compared to the direct interaction between the Action-part and the NE.

The reconfiguration method influences also the execution duration of the Action-part. Transmitting reconfiguration requests to the CM system is usually much faster than performing the changes directly on the NEs. That means that the runtime of the Action-part is much shorter if the CM toolchain is used, but the configuration parameter settings are not enforced yet when the Action-part ends.

4.4 SON-Functions Summary and Findings

This chapter introduced a uniform design scheme for SON-Functions that is based on a detailed analysis of the existing SON use cases. The analysis revealed the functional requirements of the different SON use cases and their SON-Functions. From this starting point the results where generalized and a broadened design scheme was derived that could be used for all of the SON-Functions. Following the presented conceptual tripartite separation of a SON-Function into Monitoring-part, Algorithm-part, and Action-part, provides a functional partitioning similar to the well known MAPE loop. Although the design scheme requires a strict separation of the SON-Function functionality into Monitoring-, Algorithm- and Action-part, it is constructed in a way that the SON-Functions for the existing SON use-cases can be mapped to it. This is possible because there is still a high degree of freedom on how the assignment to the individual parts is done. The Monitoring-part can perform, for example, only the detection for a single alarm or it can gather a large amount of data, which is preprocessed and evaluated before the transition to the Algorithm-part. As long as there is no general change for the mode of operation of SON-Functions the presented design scheme will be applicable for future SON-Functions.

This is highlighted by the possibility to tailor the Monitoring-part to the needs of a use-case as shown in Section 4.1. The development of a SON-Function based on a 3GPP SON use case in accordance to the proposed

design scheme is shown in Chapter 11.

The presented conceptual design scheme for SON-Functions contributes to several research questions that were introduced in Chapter 1.3:

Concerning the research area Reliability and Robustness, a contribution to the following research question has been made:

R5 How can the structure of a SON-Function support the detection and resolution of conflicts?

The tripartite functional partitioning of SON-Functions is a key building block for the detection and resolution of run-time conflicts. A proper assignment of the performed tasks to the respective SON-Function parts supports the identification of the tasks within the SON-Functions that can be affected by other SON-Functions, or have the potential to negatively affect other SON-Functions. Due to the transitions between the SON-Function parts it is possible to know which tasks have been performed and how they might have been affected by, or have affected tasks of other SON-Functions.

Concerning the research area Efficiency, a contribution to the following research question has been made:

E1 How can new SON-Functions be designed and developed such that they can be easily integrated into the SON enabled network?

A uniform design of all SON-Functions, according to the introduced common design scheme, facilitates the introduction of SON-Functions into a SON enabled network. All systems can expect an identical behavior of the SON-Functions and therefore no additional adaptation for new SON-Functions is required.

Concerning the research area Flexibility, contributions to the following research questions have been made:

F1 Is there a uniform, future proof design scheme for SON-Functions?

The presented uniform SON-Function design scheme is sufficiently flexible to be applied to the SON-Functions of the proposed SON use cases [Leh07a, Leh07b]. It does not restrict the functionality or mode of operation of individual SON-Functions. The possibility of flexible assignment of functionality to the building blocks of a SON-Function allows that the SON-Functions for future SON use cases can be developed according to the proposed scheme.

F3 How can the system be designed such that it will be able, without problems, to cope with newly introduced SON-Functions?

This question relates directly back to the research question F1. The design of the systems in a SON enabled network is closely connected to the design of the SON-Functions. A uniform SON-Function design scheme, thus the semantic clustering of tasks into the different SON-Function parts, provides abstract knowledge about the SON-Functions. Based on this knowledge all supporting parts of the SON enabled network can be structured, designed and developed to support the execution of these SON-Functions. Especially the run-time conflict detection and resolution benefits from the uniform SON-Function design and the associated knowledge about the SON-Functions, since it provides the clear separation of affected and affecting tasks. That means, as long as it is possible to develop a SON-Function according to the tripartite design scheme, it can be used within the SON enabled network without requiring any changes to the conceptual operation of the system. This does not mean that no adaptations or enhancements to the used APIs or similar might be required.

5. SON-FUNCTION CONFLICTS

The detection, prevention and resolution of conflicts between executed instances of different SON-Functions is an important aspect of SON operation. Executed SON-Function instances can show conflicting behavior towards SON-Function instances of the same or different type, and they can also be in conflict with operational guidelines. Both types of conflicting behavior have to be detected and treated properly to assure efficient network operation. SON-Function instances use measurement values and configuration parameter values in order to identify operational issues (for example underperformance) and compute new configuration parameter settings to resolve the detected issues [ea09b]. Tasks that are performed are the configuration of newly inserted NEs or optimizing existing NE configuration parameter settings. If functions operate on the same configuration parameters, influence identical measurements, or change the same characteristics, there is a possibility of a conflict between those functions. For the design of the SON-Function coordination logic that is used to handle SON-Function conflicts it is important to identify potential conflicts between SON-Functions already at design-time. Multiple reasons that can cause conflicts between SON-Functions, are identified and described in Section 5.1.

An introduction to different conflict causes and provide, based on the causes, a conflict classification is given. This classification separates the conflicts into three basic classes and, if plausible, into multiple subclasses. The goal is to allow a fast identification of potential conflicts between SON-Functions based on the classification. For the detection of conflicts and the design of the coordination logic the classification can be used as a checklist to assure that all conflicts are properly detected and handled.

5.1 Conflict Description and Classification

An analysis of SON-Functions identified three conflict classes that cover the major conflict types and additional subclasses for the fine grain conflict categorization.

- Configuration Conflicts
- Measurement Conflicts
- Characteristic Conflicts

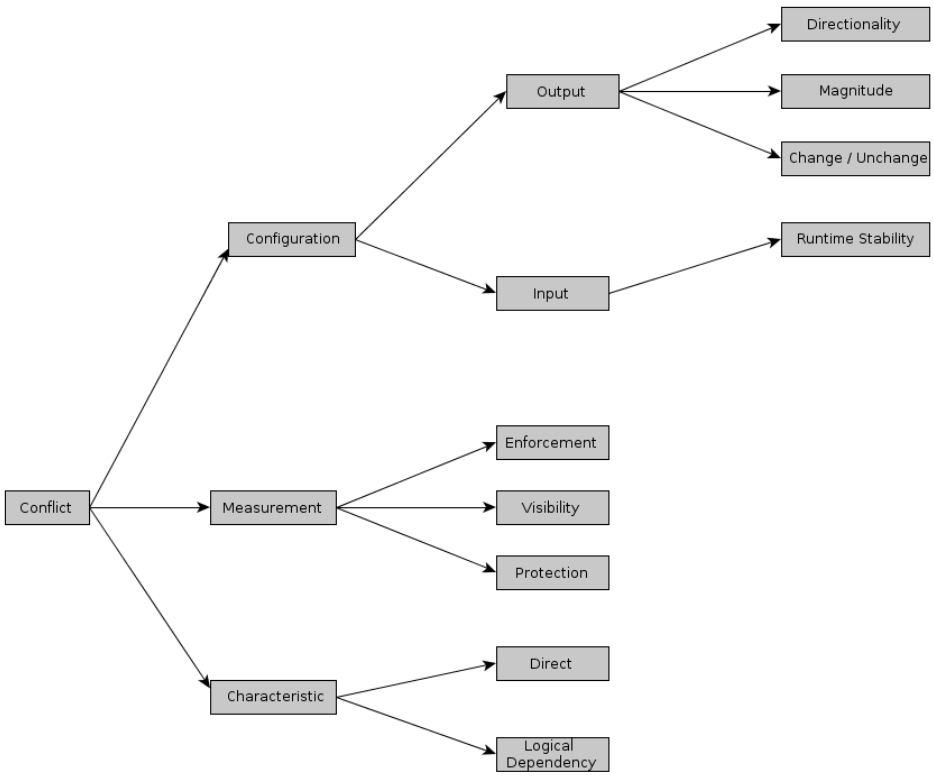


Fig. 5.1: SON-Function Conflict Classification Graph

The level of complexity required to detect the conflicts increases from configuration conflicts to characteristic conflicts. The detection of configuration conflicts can be performed with automated tools while for characteristic conflicts, at the moment, still operational experience and a good knowledge about the network, the relations between entities within the network and the relations between their configurations are required. The complexity of detecting measurement conflicts lies between configuration and characteristic conflicts, they can partly be detected automatically and partly require operational experience.

A detailed introduction to the basic conflict classes and their specific subclasses is given in the following sections. The graph in Figure 5.1 shows the three basic classes and all their subclasses.

5.2 Configuration Conflicts

Configuration conflicts are the most obvious conflicts between two SON-Functions. A conflict is categorized as a configuration conflict whenever the

functions operate on at least a single or a set of shared configuration parameters. Figure 5.2 provides a high level overview on the usage of configuration parameters by different SON use cases. There is not a single configuration parameter which is not targeted by multiple SON uses cases. An automatic analysis of the SON-Functions can be used to identify all function pairs that operate on the same configuration parameters. The operation on shared configuration parameters is a necessary but not always a sufficient condition to classify the functions as potentially conflicting. If two functions only consume the same configuration parameter value but do not change this particular configuration parameter and do not operate on additional shared configuration parameters they are not conflicting. In order to determine whether two functions that operate on identical configuration parameters show conflicting behavior, a more detailed analysis including operational experience might be required.

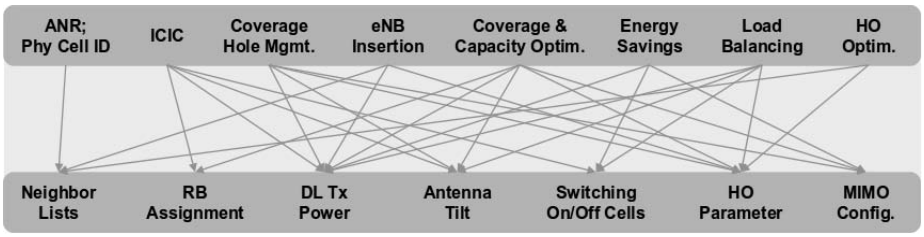


Fig. 5.2: Configuration Parameters are influenced by multiple SON-Functions of multiple SON use cases

The configuration conflict class is subdivided into output and input parameter conflicts.

5.2.1 Output Parameter Conflict

The most common configuration parameter conflict is the output conflict. A conflict falls into this category whenever both conflicting SON-Function instances try to modify at least a single shared configuration parameter. This conflict category is further subdivided to provide for the enforcement of additional operator policies. An operator could allow configuration changes by multiple SON-Function instances within certain limits, but wants to avoid and prevent strong or oscillating changes. The assignment to one of the subcategories does not have to be exclusive, two SON-Functions could have, for example, both a magnitude and a directionality conflict. In such a case the coordination logic that is used to determine the treatment of the conflicting functions has to be defined in a way to cover both conflict types.

- **Directionality:** A directionality conflict occurs in case the SON-Function instances try to modify the shared configuration parameter setting in opposite directions. A typical example is the increase or

decrease of the remote electrical tilt angle or the transmission power of an antenna. Directionality conflicts are used when changes are allowed but the coordination logic has to check for oscillating reconfigurations, to prevent, for example, power up followed by either a power or tilt down, followed by a power up...

- **Magnitude:** Even if the setting of the shared configuration parameter is changed in the same direction, there is still the possibility for a conflict. The conflict originates in different magnitudes of the requested changes. In case one SON-Function instance requests a change that is much larger than the change requested by the other SON-Function instance, the expected effects might be mutually cancelled or even be reversed. In addition multiple gradual changes are often preferred over abrupt large changes which could cause a lot of side effects.

- **Change / Unchange Conflict:** A conflict falls into this category if one of the SON-Function instances determines that one of the shared configuration parameters has already the optimal value and does not need to be changed, while the other shared configuration parameters have to be changed in order to match the already optimal setting of the unchanged parameter. The conflicting function then modifies the setting of the parameter that is considered to be unchanged by the first SON-Function instance. As soon as the parameter setting is modified the goals of the first function will not be met anymore.

5.2.2 Input Parameter Conflict

SON-Functions often use configuration parameter values as an input. Either because the configuration parameters that are affected have to be configured dependent on values of other configuration parameters or because the input is used to derive the new configuration parameter settings. SON-Functions that operate on configuration parameters whose setting is dependent on the values of other configuration parameters can suffer from an input conflict. In order to compute new values the SON-Functions will read particular configuration parameter values and then have to rely on their stability until the end of the operation of the Action-part. The allocation of PCIs [3GP10b] is an example for such a type of SON-Function. It will gather the PCI configuration of the neighboring cells and based on this information compute the new configuration for the target cell. In case the PCI of one of the neighboring cells is changed during the runtime of the PCI allocation SON-Function the resulting network configuration can result in a collision or confusion, which leads to service degradations until an overall collision and confusion free configuration has been performed.

5.3 Measurement Conflicts

Many SON-Functions use performance measurement values, such as counters, timers, radio measurements etc., as input. In general, performance measurements are used to characterize the state of the network, either directly as raw values or aggregated into more abstract KPIs or KQIs. States of the network are described through sets of measurements and their values with respect to reference values. There are different ways how measurements are used by the SON-Functions. Either to characterize and detect triggering situations, or as input to the Algorithm-part for the evaluation of the state of the system and the deduction of appropriate actions or configurations parameters settings to resolve the current situation.

Even if both, monitoring and Algorithm-part of a SON-Function use performance measurements as input, the set of used measurements can be partly or completely disjoint. For example, the Monitoring-part makes use of a minimal set of performance measurements that is sufficient to characterize the triggering situation. Then, for the Algorithm-part, an additional or even completely disjoint set of performance measurements is used for a detailed analysis of the situation and to compute new configurations.

In general the result of almost any SON-Function execution whose triggering situation is defined by performance measurements is evaluated through the effects on these measurements. A SON-Function execution will not necessarily affect all performance measurements that have been used, but, due to the conceptual foundation of SON-Function operation, it will at least affect the measurements that are used to define its triggering situation as shown in Figure 5.3.

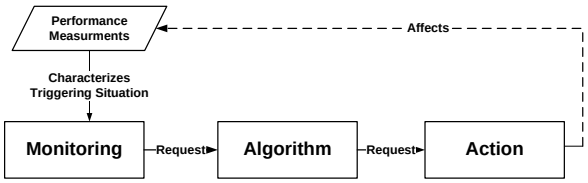


Fig. 5.3: Monitoring Input Measurements are affected by SON-Function Actions

In the same way configuration parameters can be used by multiple SON-Functions, the same performance measurements are also often used by multiple SON-Functions. A single executed SON-Function action can therefore have negative effects on other SON-Functions. It can affect performance measurements that are either used by the monitoring or Algorithm-part of other SON-Functions. Figure 5.4 shows an example where one SON-Function affects the performance measurements used by the Algorithm-part of another SON-Function.

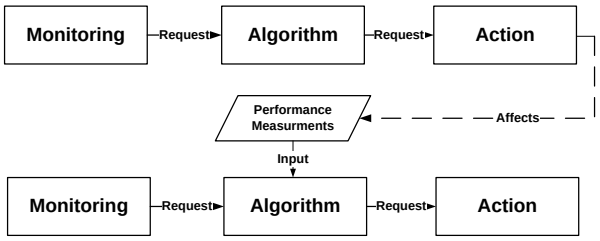


Fig. 5.4: Algorithm Input Measurements are affected by SON-Function Action

If the performance measurements are affected after they have been used as input but before the end of the SON-Function execution the results of the Algorithm-part are not based on the current state of the network anymore. The enforced configuration parameter settings can then have negative effects on the network performance.

For the measurement conflicts it is important to note that changes in measurements can have different reasons. On the one hand, they can be caused by configuration changes, or on the other hand by changes in the traffic patterns in the network. For measurement conflicts only changes which are caused by configuration changes are relevant.

The conflicts of the measurement conflict class are subdivided based on the reasons for the conflict. Three subclasses have been identified that aim, to facilitate the identification of measurement conflicts and the development of a respective coordination logic:

- **Enforcement:** Information about the configuration of an NE is in general available at, at least, two separate entities within the network. Once directly at the NE but also at the CM systems, possibly also at intermediate DM systems. For the enforcement conflict the configurations directly at the NEs are relevant. Reconfiguration requests in the network are often not instantaneously enforced at the target NE as shown in Figure 5.5. Depending on the network setup and the access possibilities of a SON-Function changing a configuration takes some time between the reconfiguration decision and the actual enforcement of the configuration change. Especially if a SON-Function interacts with the NE through an intermediate configuration tool chain there might be a noticeable delay. While the configuration process is executed the already "performed" changes are not yet enforced at the NEs and therefore not yet visible to other SON-Function instances. While those changes have not been enforced other SON-Functions could trigger conflicting actions. The conflicting behavior either results in contradicting configuration requests or additional unnecessary reconfigurations.

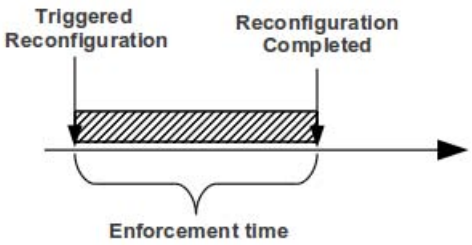


Fig. 5.5: Time required to enforce Configuration Changes

- **Visibility:** Changes of configuration parameter settings have an instant effect on the related performance measurement values that are produced and recorded. But it is possible that configurations that have been configured at the NEs are only gradually enforced towards the UEs. Changed handover parameters are, for example, not instantaneously pushed to all UEs but gradually over time, whenever needed. Therefore the performed configuration changes do not immediately show their full effects and thus are not reflected in the performance measurement data. SON-Functions that operate on those values might produce erroneous results.

  Visibility conflicts can occur between the enforcement of the changes to the configuration parameter settings at the NE and the end of the Visibility delay, as shown in Figure 5.6. In the example above, this interval has to cover the time until the changes to the configuration settings have been communicated to a statistically significant fraction of the served UEs.

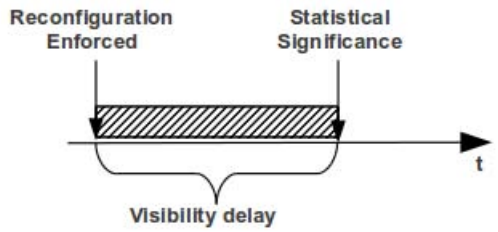


Fig. 5.6: Visibility Delay

- **Protection:** To get a realistic view of the network, measurements are often collected over a certain time interval. The collection interval assures that a statistically significant number of measurements is included into KPIs that are used to asses the state of the network. These measurement values are usually collected using a sliding window type of method. This means that at any time they reflect the situation of the network during the past measurement interval. Protection conflicts

occur, if SON-Function instances, when started, access such measurements and the visibility delay of a previously executed SON-Function instance has ended within collection interval of these measurements.

Especially if a function retrieves data right after the end of the visibility delay, invalid data will be consumed. The protection conflict is visualized in Figure 5.7. A SON-Function instance execution is started at some point, and the SON-Function instance uses data collected during the visibility delay of a previous function.

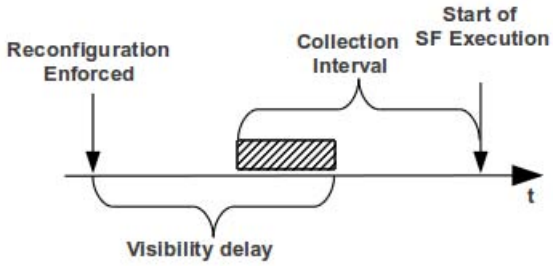


Fig. 5.7: Protection Conflict

The possibility of successful automatic detection of measurement conflicts is restricted. On the one hand to conflicts where the same measurements are used to identify the triggering situations of different SON-Functions, because of the direct interrelation between the input for the Monitoring-part and the affected measurements.

On the other hand conflicts can be automatically detected if SON-Functions use measurements as input to the Algorithm-part which are also used to characterize triggering situations of other SON-Functions.

In case a SON-Function affects measurements, which it does not use as input to monitoring or algorithm, conflicts with other SON-Functions are possible, which cannot be detected automatically through an analysis of the used performance measurements.

The detection of measurement conflicts gets even more complicated if KPIs are used which share some of the measurements from which they are constructed. If a SON-Function designer is not or not completely aware of the employed measurements, measurement conflicts can be overseen.

For a fully automatic detection of all measurement conflicts means have to be used which allow to integrate operational experience into the conflict detection process. Advanced semantic modelling techniques could be used to provide the required information to an automated conflict detection process.

5.4 Characteristic Conflicts

Cells in a network have certain characteristics, whereof the cell size or the coverage area of a cell are the most prominent ones.

Cell characteristics, often, cannot be measured directly but are reflected in a multitude of measurements and are influenced by multiple configuration parameters. KPIs like the cell load and the handover success rate are often directly coupled to the location of the cell border. A cell border can be changed by adapting the antenna tilt or the transmission power but also logically through adaptations of different handover parameter settings, for example, the handover hysteresis values. If tilt or power are changed, the cell border changes physically, meaning that the received signal quality at a given position is changed, which affects the handovers to neighboring cells. If the handover parameters are changed, the received signal quality stays identical but handovers to neighboring cells are initiated sooner or later.

For this reason, it is possible that SON-Functions which impact the same specific cell characteristic consume fully disjoint sets of performance measurements and adapt a disjoint set of configuration parameters. Two subclasses of the characteristic conflicts were identified:

- **Direct:** SON-Functions with a direct characteristic conflict target the same characteristic. For example two functions aiming to change the cell coverage area are in a direct characteristic conflict. Detection of a direct characteristics conflict has to rely on operational knowledge at design-time. Typically the target "cell size or coverage area" should be included in the function description and a direct characteristic conflict can therefore be revealed by a full text search of the function descriptions.

- **Logical Dependency:** conflicts on the characteristics in contrast are much harder to detect and require more operational experience. Such a conflict appears if there is a logical dependency between the performance metrics influenced respectively used by a SON-Function. For example one SON-Function instance changes the size of a cell to optimize the coverage while another function changes the handover hysteresis triggers to perform load balancing at a neighboring cell. In that case the conflict is not very obvious. By changing the cell size, the overlap in the shared coverage area is changed, the physical cell boundaries are moved. This change will instantaneously also have an impact on the load of the cell under load optimization, since the UEs will start the handover procedure either earlier or later although the hysteresis values have not yet been adapted. The measurements used as input to the load balancing SON-Function are not valid anymore and the optimization will probably not meet its goals.

All automatic detection measures which have been introduced for configuration and measurement conflicts will fail for this kind of conflicts.

To be able to perform automatic characteristics conflict detection additional information is required, which is today mostly only available as operational knowledge. Advanced semantic modelling could be used as an approach to close the information gap. Such models need to include, apart from the used measurements and configuration parameters, also dependencies and relationships between characteristics, configuration parameters, and performance measurements. Semantic modelling also allows the specification of synonyms, which is important because a search for the target coverage could miss functions whose description uses cell size instead. In order to detect characteristic conflicts reasoning on semantic models is employed.

5.5 Naming of Conflict Categories

The introduced categorization is mapped into a compact table representation and a naming scheme based on letters and numbers. The main conflict class is represented by a letter (A-C) and the subclasses are identified by numbers. In case there are multiple levels of subcategories, those subclass identifiers are separated by a dot. Table 5.1 shows the main conflict classes and a first level of subclasses. The details of the second level subclasses of the Configuration conflicts are shown in Table 5.2.

Tab. 5.1: Overview and Naming of Conflict Classes

Conflict Category	Sub-Category	
	Sub Category	Short Description
A	**Configuration Conflicts**	
	A1:	Input Parameter Conflict
	A2:	Output Parameter Conflict (Details in Table 5.2)
B	**Measurement Conflicts**	
	B1:	Enforcement Conflict
	B2:	Visibility Conflict
	B3:	Configuration Conflict
C	**Characteristic Conflicts**	
	C1:	Direct Characteristic Conflict
	C2:	Logical Dependency Conflict

Tab. 5.2: Detailed Specification of Parameter Conflicts

Conflict Category	Sub-Category	
	Sub Category	Short Description
A2	**Output Parameter Conflicts**	
	A2.1:	Directionality Conflict
	A2.2:	Magnitude Conflict
	A2.3:	Change / Unchange Conflict

5.6 Conflict Categorization Summary and Findings

A sound detection and classification of potential conflicts between SON-Functions is crucial for the design of appropriate means to detect and resolve conflicts between instances of SON-Functions at run-time. The categorization of all possible conflict types supports this important task, since the classification can be used as a guideline or check list when designing new SON-Functions.

Depending on the type of conflict, automatic detection is already today easily possible, either through an analysis of the SON-Function implementations or through an analysis of the available information and data models.

For some conflicts types, especially the characteristic conflicts, more elaborate models are required which can fill the information gap between available and required information. In the future, semantic modelling and advanced machine learning technologies can largely contribute to a more comprehensive automation of conflict detection and categorization.

Sometimes it is not easy to decide into which category a given conflict falls, especially if a large number of different performance measurement values are used as input, or are affected by the SON-Function instances. In this case it might be better to categorize a conflict as a characteristics conflict, even though its automatic detection is more complex. For the mapping of a given conflict to a particular conflict category, often operational knowledge is required to determine that the conflict can better be assigned to a more abstract conflict class.

The category of a SON-Function conflict provides indications on how a given conflict can be treated. In case of a parameter conflict exclusive access to the configuration parameters for a single SON-Function is helpful, while a temporal alignment of SON-Function execution can sometimes be used to prevent conflicting behavior that causes measurement or characteristic conflicts.

The conflict categorization for SON-Functions contributes to several research questions that were introduced in Chapter 1.3:

Concerning the research area Reliability and Robustness, a contribution to the following research question has been made:

R4 Which parts of the SON-Functions are affected by conflicts?

The analysis of the interaction of SON-Functions and the categorization of the conflict types reveals directly how conflicts affect particular parts of a SON-Function. The categorization is not limited to the area of SON, but can be applied in all systems where autonomic functions change the configuration based on input data from the system, which affects the behavior of the system. This is possible because the categorization provides conflict categories for all phases of the SON-Function execution, the possible interactions between SON-Functions and their effects on the network.

Concerning the research area Efficiency, a contribution to the following research question has been made:

E2 How can conflicting behavior be detected and prevented or resolved?

The conflict categorization provides a guideline on how to detect potential conflicts already at design-time. This guideline results from how the conflicts are categorized. If a conflict is, for example, categorized as an input conflict it is mandatory to check for potential modifications of the input data. Similar guidelines result from the other conflict categories. Hence, as soon as a conflict is assigned to a conflict category it is possible to provide specific run-time means that detect and prevent or resolve this conflict. This approach can be generally used in systems where MAPE-like operation is used, because each of the functions monitors, analyses and processes some sort of input (Configuration or measurement conflict) performs actions by changing the configuration of NEs, which affects the behavior and the characteristics of the network.

6. SON-FUNCTION INSTANCES

In the same way as an operator can focus on individual or a small number of NEs, SON-Functions can also operate on a limited subset of the network. An operational SON-Function is called a SON-Function instance. Each SON-Function instance is characterized through the generic SON-Function type as well as its spatial and temporal extension. Each SON-Function instance has an individual Impact-area and Impact-time. Through the spatial separation of SON-Function instances a conflict-free, concurrent execution of multiple SON-Function instances within the same network is possible. For example, a COC SON-Function instance can be executed in one part of the network while Coverage and Capacity Optimization (CCO) SON-Function instances optimize other parts of the same network.

6.1 SON-Function Impact-area

The Impact-area of a SON-Function instance describes its spatial scope. Spatial scope does not necessarily refer to a geographical area but can also be described through network characteristics as, for example, targeted NEs or network domains. In general the Impact-area of a SON-Function instance provides information about which parts of the network are affected through the execution of this particular SON-Function instance. The term affected does not only refer to a configuration change, but covers also locking to assure input parameter stability or other effects caused by a changed environment. For example the load of a cell might be changed through an antenna tilt change in a neighboring cell or changed handover parameters.

The Impact-area is an important information, required to detect conflicting SON-Function instances. To turn a potential SON-Function conflict into an actual conflict the potentially conflicting SON-Function instances have to have an overlapping Impact-area.

A SON-Functions's Impact-area can be constructed from several individual components that are based on the input and output requirements of the SON-Function and the operational guidelines within the network. The four components are:

- Function-area
- Input-area
- Effect-area
- Safety-margin

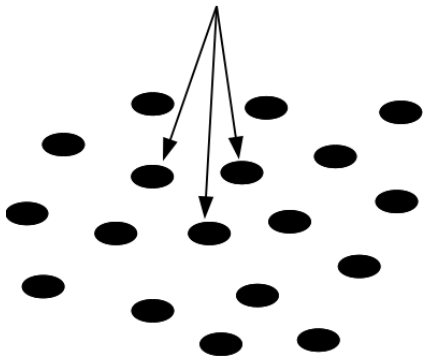


Fig. 6.1: Function-area

The named components of the Impact-area are not necessarily fully disjoint. A cell falling into the Input-area can also experience some effects through the executed SON-Function and would therefore also be part of the Effect-area. The separation into the different areas gives the function designer a guideline for the analysis that needs to be performed to come to a complete Impact-area definition.

The following sections introduce the four components of the Impact-area and how they are specified for each SON-Function. Section 6.1.5 gives recommendations on how to define an abstract Impact-area and, how such an abstract description is mapped at run-time to the actual network deployment.

6.1.1 Function-area

The Function-area is the minimal core of the Impact-area. It is formed by a single or multiple NEs or cells and comprises those NEs that are directly configured or reconfigured by the SON-Function instance. In order to facilitate the further understanding, the targets of the SON-Function instances in the example are only referred to as cells.

The following examples are all based on the same set of cells which are shown as circles. Figure 6.1 shows this set of cells with the Function-area indicated by arrows pointing at three cells. In the description of the other Impact-area components the Function-area will always be shown as non-filled circles.

6.1.2 Input-area

Some SON-Functions, as for example the PCI function, include information from cells that are not directly targeted, that means reconfigured, into the algorithm execution. This information can contain, for example, the configuration of surrounding cells or performance measurement values that they

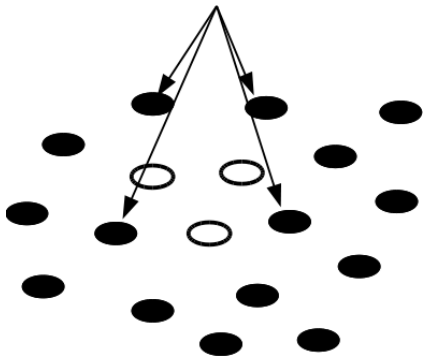


Fig. 6.2: Input-area

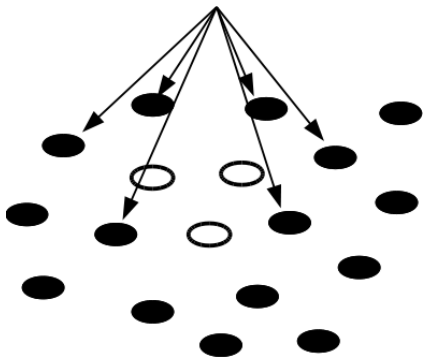


Fig. 6.3: Effect-Area

have gathered. Those cells that provide input to a SON-Function instance form the Input-area. In Figure 6.2 the Function-area is represented by circles and the Input-area is indicated through the filled circles with arrows pointing to them.

6.1.3 Effect-area

The changes enforced to the network during the execution of a SON-Function instance can have effects on other cells, even if those cells are part of the Function-area. Typically functions that optimize the load of one cell will have effects on the load conditions of the neighboring cells as well. The Effect-area covers all cells that experience (side-)effects through the executed function instance. Figure 6.3 depicts the Effect-area (indicated by arrows) of the executed SON-Function instance.

6.1.4 Safety-margin

The Safety-margin serves different purposes. At first it allows the function designer to include an area into the Impact-area that does not fall into

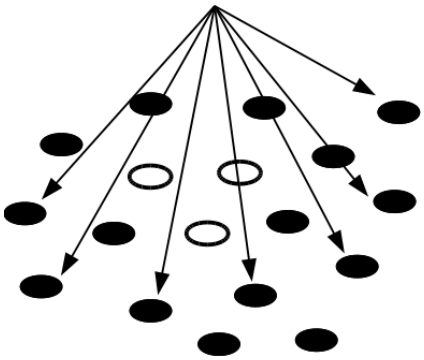


Fig. 6.4: Safety-margin

one of the previously described areas. Especially the Effect-area could be larger than assumed. Due to unpredictable radio propagation effects, more cells could be affected by the performed changes. The inclusion of a Safety-margin extends the border of the Impact-area and therefore provides a higher degree of protection against undesired effects. With an increased Impact-area two potentially conflicting SON-Function instances which are executed in close spatial proximity could then be considered conflicting since their extended Impact-areas overlap. A scenario where the usage of a Safety-margin is beneficial is the execution of a SON-Function instance within an urban network with a very dense layout of small cells. In such network deployments it is often impossible to determine exactly all cell neighborships and therefore the set of affected cells.

The second purpose of the Safety-margin is to use the extended Impact-area to pro-actively protect subsequently triggered SON-Function instances against conflicts. Such a protection is useful if it is known in advance that after the execution of the current SON-Function instance an instance of another SON-Function for the identical cells needs to be triggered that has a larger Impact-area. A good example is the coverage and capacity optimization that has to be performed after the introduction of a new cell. The Impact-area for the introduction of a new cell is equal to this single cell. But the Safety-margin can already contain all neighboring cells. The extended Impact-area will then, automatically, block these cells for other SON-Function instances through the conflict detection mechanisms.

An example of the Safety-margin is shown in Figure 6.4.

6.1.5 Specification of the Impact-area

As shown in the introduction, the Impact-area is an important information to detect whether two potentially conflicting SON-Function instances are actually conflicting. This capability is an important pre-condition to perform efficient SON-Function instance coordination. The specification of an

abstract Impact-area for a SON-Function is part of the SON-Function design process. At run-time, in the moment when the SON Coordinator has to decide, whether the Impact-area of the requested SON-Function instance is overlapping with the Impact-area of a potentially conflicting SON-Function instance, a simple but reliable way of mapping the abstract specification to the actual network layout is required.

This section introduces the specification process and discusses the possibility of a pair-wise Impact-area definition for potentially conflicting SON-Functions instead of the definition of a single Impact-area per SON-Function.

6.1.5.1 Abstract Definition of a SON-Function Impact-area

At design-time an abstract Impact-area specification for each SON-Function has to be provided, which can be used at run-time to identify the Impact-area of the SON-Function instance within the actual network deployment.

Several requirements have to be fulfilled for reasonable abstract Impact-area specification:

- **Design-time specification:** The most important requirement is the possibility to specify the SON-Function Impact-area at design-time without any knowledge of the concrete network deployment and cell layout
- **Common, function independent specification:** The same Impact-area specification approach has to be applicable to all types of SON-Functions. The approach has to be independent from the entities in the network that form the Impact-area for a particular SON-Function
- **Reliable and simple mapping:** The mapping of the abstractly defined Impact-area to the network setup respectively the cell layout has to be a simple and fast to perform task. It has to provide a reliable identification of the entities that form the Impact-area
- **Simple overlap detection:** For the automatic conflict detection it is important, to allow a fast Impact-area overlap detection

6.1.5.1.1 Analysis of the Impact-area Description For the abstract description of an Impact-area it is important to look at the way how the concrete Function-area of a SON-Function instance can be identified. A SON-Function itself is provided in an abstract form, which is instantiated at run-time to perform its tasks, for a mostly limited Function-area. If its Function-area consists of more than a single NE a mechanism is required to identify all NEs that form the Function-area. Looking at the description of the SON use cases [Leh07a], for example, the Mobility Robustness Optimization (MRO) SON use case [HSS12], shows that the Function-area for SON-Functions is often described based on cell neighborships. In general, the entities that are comprised in the Impact-area can be described through specific relationships between those entities, which exist in the network.

A multitude of relationships exists that can be used to identify the entities in the Impact-area, which are also indicated in the conflict classification and the parameter categorization [SBT10]. The following listing gives an overview on the most common relationships which are used.

Neighborships are the most commonly used relationship to define the Function-area of a SON-Function. They are not limited to the physical neighborships between cells, other neighborships, for example between eNodeBs, are also often used. Neighborships between two cells can be identified by:

- Overlapping coverage areas
- The existence of handover associations between the cells
- Geographical distance

eNodeBs are neighboring if:

- They are connected to each other through a X2 interface [3GP11b]
- Association to the same MME or MME Pool

In the SON use case descriptions not only direct neighborships but also neighborships of a higher degree like second and third degree are used.

Apart from neighborships, other relationships are used in SON use case descriptions to describe the Function-area. A SON-Function can, for example, operate on:

- All cells that are assigned to a specific Traffic Area (TA)
- All NEs required to setup a particular bearer
- All NEs within a specific network or vendor domain
- All cells within a network layer (coverage or capacity layer)

Since relationships between entities in the network are already used in the SON use case descriptions it is beneficial to use them also for the abstract description of Impact-area of a SON-Function. The Function-area as the root for the description of the Impact-area, because it is a required input for the instantiation of the SON-Function, and will therefore always be available for each SON-Function instance.

The Impact-area of a SON-Function could, for example, be described as: *"The Function-area is a single cell, the Input-area consists of the neighboring cells to which handovers can be performed. The Effect-area contains all cells that are connected to the neighboring eNodeBs of the eNodeB of the targeted cell."*

This example shows how several relationships are combined to describe the SON-Function's Impact-area:

- Neighboring Cells (handover association)
- Cells connected to eNodeBs
- Neighboring eNodeBs (defined by X2)

The resulting Impact-area specification for a SON-Function will consist of a set of descriptions to identify the impacted entities based on multiple different relationships.

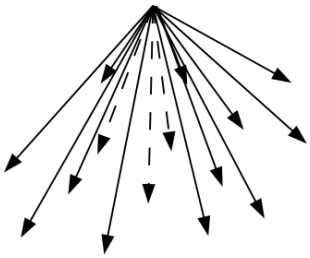
Since the components of the Impact-area are not necessarily disjoint, individual entities could be contained in multiple Impact-area components, for example the Input- and the Effect-area. Even within a single Impact-area component it is possible that individual entities are multiply addressed through the usage of different relationships. Such a redundancy should be avoided, to reduce the description complexity. These redundancies can be reduced if only a minimal number of different relationships is used for the description of the Impact-area components. In a final step, all duplicate elements can easily be removed from the Impact-area description.

In order to map an abstract Impact-area description to the real network deployment, the Function-area is identified by the SON-Function instance. The abstract Impact-area description together with knowledge about the relationships in the network are then used to identify the impacted entities in the network.

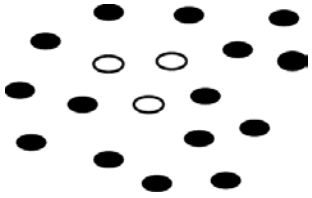
Figure 6.5 shows a simplified visual example for the mapping of an abstract Impact-area description to a concrete network deployment.

- **Figure 6.5(a):** shows the abstract definition of the Impact-area. The dashed arrows indicate the Function-area, the other arrows represent the remaining components of the Impact-area
- **Figure 6.5(b):** shows the cells of a network. The non-filled circles indicate the cells of the Function-area that were identified by the SON-Function instance
- **Figure 6.5(c):** shows the abstract Impact-area description mapped to the network. The dashed arrows point to the non-filled circles. The Impact-area is then automatically revealed through the arrows pointing to all cells that form the Impact-area

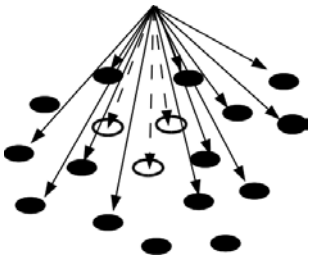
6.1.5.1.2 Specification Approach For the usability in a SON a formal way of specifying the Impact-area is required which can be used by the SON coordinator to automatically identify the concrete Impact-area of a SON-Function instance in a fast an reliable way. This section introduces first a formal, graph based approach for the specification of the Impact-area and subsequently shows how it can be used at run-time to identify the concrete entities in the network deployment.



(a) Abstract Impact-area Description



(b) Function-area in Network



(c) Instantiated Impact-area

Fig. 6.5: Impact-area Instantiation

Graph Based Formal Specification: As introduced several relationships between entities in the network exist that can be used to describe the Impact-area of a SON-Function in relation to the Function-area. Such relationships in the network can be represented as graphs. Entities, for example cells, are depicted as nodes and each node is connected with an edge to all other nodes with which it shares the relationship. This will result in one graph for each relationship that can be used to describe the Impact-area. Some entities will be represented in multiple graphs. There is a multitude of possible relationships in the network, but the usage of a relationship for the Impact-area specification only makes sense if it is directly accessible by the SON Coordinator or can be easily computed from the available information. Neighborships between cells are good relationships, since they are directly available from the network planning databases or from the NRTs which are maintained in the Evolved NodeBs (eNodeBs).

The benefit of the graph based representation of relationships is that the Impact-area of a SON-Function can be described as a combination of sub-graphs of the relationship graphs. These sub-graphs are defined through graph-operations that reflect the literal descriptions from the SON use case descriptions. These operations are based on the Function-area of the SON-Function, but until the concrete Impact-area of a SON-Function instance needs to be identified, the descriptions are still abstract and no knowledge about the actual network deployment is required.

To define the Impact-area, the function designer performs several steps:

1. Definition of the SON-Function and the results of the conflict analysis are used as input
2. For each of the conflicts it is determined whether Input-area, Effect-area or a Safety-margin is required
3. Relationships that can be used for the specification are identified. Information about potential conflicts can support this step.
4. Selection of the graphs that represent the required relationships.
5. Definition of operations on the graphs to produce subgraphs, which allow to identify the needed entities. The root for each operation is the Function-area of a SON-Function. If for example only the direct neighbors of a target cell are required, the cell-neighborship-graph is used as input to an operation that produces a subgraph that only contains the node representing the target cells and those of the direct neighbors.
6. Removal of redundant operations. Redundancies in the specifications are removed through the unification of subgraphs which are based on the same underlying graph. If different subgraphs have been derived from the cell-neighborship-graph, they can be unified into a single graph that covers all entities identified by the individual graphs. For the abstract specification, the unification of subgraphs is performed by providing new graph operations, which, when applied to the underlying graph return the unified subgraph. This step is performed across the different Impact-area components.

After performing these steps, the abstract formal description of an Impact-area consists of a set of operations for different relationship graphs.

Instantiation of the Impact-area: To determine the concrete Impact-area of a SON-Function instance, the information provided about the Function-area of the SON-Function instance is used as a starting point. It contains already the SON-Function identifier and provides also the identifiers of the entities that form the Function-area.

To identify the entities of the Impact-area the following steps are performed:

1. Extraction of function instance specific information
2. Retrieval of abstract Impact-area description for the requested SON-Function
3. Retrieval of required relationship graphs
4. Execution of the operations from the abstract Impact-area specification on the graphs with the Function-area as starting point
5. Computation of a set of nodes that comprises all nodes of the resulting sub-graphs.
6. As an optional step, removal of duplicate entries

After these steps the resulting set contains the identifiers of the Impact-area of the SON-Function instance. To determine a run-time conflict between potentially conflicting SON-Function instances the Impact-areas are intersected. If the intersection results in a non-empty set, one necessary but not sufficient requirement for a SON-Function instance conflict has been fulfilled.

Specification Example: Here, a simple scenario is used to give an example of the formal specification of the Impact-area for a SON-Function. The SON-Function targets only a single cell, which is denoted as a. The Effect-area are all direct neighboring cells of a and input is provided from all neighbors of the Effect-Area, thus the Input-area consists of all second degree neighbors of a in the cell layout.

The underlying graph $G = \{V, E\}$ used to specify the Impact-area for this SON-Function is the cell-neighborship graph. The nodes represent the cells in the network and edges connect two nodes if the respective cells have an entry in the neighbor relationship table. The information to compute this graph can either be taken from the configuration databases or directly from the NRTs of the eNodeBs.

Two subgraphs are required to specify the Impact-area. The subgraph for the Effect-area is specified by Equation 6.1 and the Input-area by 6.2.

$$G' = \{V', E\} \qquad V' = \{x \in V | dist(a, x) = 1\} \tag{6.1}$$

$$G'' = \{V'', E\} \qquad V'' = \{x \in V | dist(a, x) = 2\} \tag{6.2}$$

Since both subgraphs are specified based on the cell-neighborship graph it is possible to provide a single, unified description that covers both as in Equation 6.3, which is used as a formal specification of the Impact-area.

$$G''' = \{V''', E\} \qquad V''' = \{x \in V | 1 \leq dist(a, x) \leq 2\} \tag{6.3}$$

At run-time the SON Coordinator uses the node identifier of the target cell a and the actual neighborship-graph of the network to compute the subgraph that identifies the cells which form the Impact-area. The function will return a subgraph containing all nodes which are the first and second level degree neighbors of the cell a in the neighborship-graph.

6.1.5.2 Pairwise specification of Impact-Areas

There is the possibility that the Impact-area has a negative effect on the efficiency of the SON-Function instance execution. If the Impact-area is used to align SON-Function instance execution in a sequential order, the Impact-area can potentially cause a blocking of a large number of cells or NEs for the operation of other SON-Function instances. The Impact-area

describes which NEs or cells are affected by an executed SON-Function instance independent from the actual effects. An executed SON-Function instance could affect only a single configuration parameter or a complete cell. Therefore, for different conflicts of a given SON-Function with other SON-Functions the actually required size of the Impact-area could vary. In the best case, if a pair-wise definition of the Impact-area would be used, individually fitted Impact-areas for each pair of potentially conflicting SON-Functions could be defined. That could lower the blocking probability for some conflicts and therefore increase the efficiency of the SON-Function instance execution.

If a (SON-Function) pair-wise definition of the Impact-area should be provided, the following process needs to be performed for each pair of potentially conflicting SON-Functions:

1. Identification of all possible conflicts between the two functions
2. Mapping of the conflict specific spatial requirements on Impact-area components
3. Provisioning of a function-pair specific Impact-area

While those steps are relatively simple for the first three components of the Impact-area, it is more complicated for the Safety-margin. This might require the same process to be performed for all of the subsequent functions that should be pro-actively protected through the Impact-area. It requires therefore a conflict analysis with all of the potentially subsequently triggered SON-Functions.

An analysis of the set of available SON-Functions namely, ANR, MRO, Coverage and Capacity Optimization via Remote Electrical Tilt Adaptation (CCO(RET)), Coverage and Capacity Optimization via Antenna Transmission Power Adaptation (CCO(TXP)), Energy Saving Management (ESM) and PCI, showed no benefits of a pair-wise Impact-area definition. The reason is that most of the Impact-area is specific to the requirements of the individual SON-Function. For the Impact-area of a particular SON-Function all information about the affected area is available already at the design-time of each function.

Only if the different conflict types result in completely different Impact-area requirements and the conflict types are separated between function pairs the operational efficiency could benefit from individual Impact-areas for function pairs.

As an example three SON-Functions are considered A, B and C, with potential conflicts between A and B as well as between A and C. The Impact-area between A and B consists only of directly neighboring cells, while the Impact-area between A and C consists of all entities between the cell and the responsible MME. In such an extreme case it would make sense to treat the Impact-area separately instead of unifying it into a single Impact-area which is used to coordinate both conflicts.

In case new SON-Functions would introduce the need for such different Impact-areas, the pair-wise definition of Impact-areas can easily be introduced. Instead of a single abstract Impact-area definition per SON-Function, multiple definitions would be provided together with the information about SON-Function types which mandate their usage.

6.2 SON-Function Impact-time

An important dimension of the SON-Function instance is the timing information. A SON-Function instance operates only during a limited time-interval. The interaction between SON-Function instances is therefore restricted to their individual time-intervals. Only if two potentially conflicting SON-Function instances are executed with an overlapping time-interval and overlapping Impact-area the potential conflict becomes an actual conflict.

The most obvious and most important time related information is the execution time of a SON-Function instance. It is the time interval during which the SON-Function instance gathers all relevant information, computes new configurations, and request the enforcement of the results, thus the time required to execute algorithm and Action-part.

Considering only the execution-time of the SON-Function instances is not sufficient, since changes performed by a SON-Function instance can negatively affect other SON-Function instances for some time after the end of the SON-Function instance execution. Therefore, the Impact-time is defined as the additional time interval after the execution time, during which a SON-Function instance needs to be considered to allow for an efficient and successful conflict detection and prevention.

The Impact-time together with the execution time describes the complete time interval during which a SON-Function instance can affect other SON-Functions and therefore has to be considered for conflict detection and prevention. The Impact-time is not the time during which a SON-Function has an impact on the network.

Whenever a SON-Function instance execution is requested it has to be evaluated whether another potentially conflicting function instance with an overlapping Impact-area is either currently executed or has been previously executed and its Impact-time has not yet expired. While the Impact-time of a given function instance has not yet timed out the function is considered as active.

In an initial concept only an individual, fixed Impact-time for each SON-Function was defined, but the deeper understanding of SON-Functions, SON-Function conflict types, and the requirements of the conflict detection and resolution lead to the conclusion that this was not sufficient. The following section presents a much more flexible and dynamic Impact-time definition.

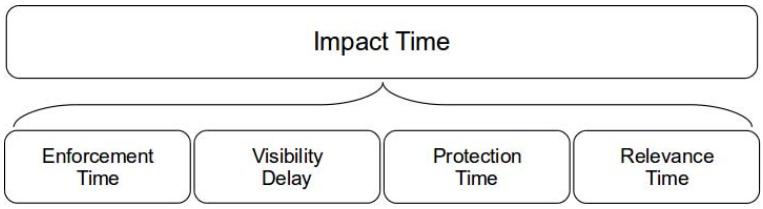


Fig. 6.6: Separation of Impact-time into Multiple Components

6.2.1 Components of the Impact-time

The analysis of SON-Function conflict classes lead us to the conclusion that the Impact-time should not be defined as a single monolithic time interval but rather be defined based on multiple time intervals that dynamically add up to the Impact-time. Based on the findings of the conflict analysis and the experience with the definition of the Impact-area an Impact-time definition is presented, which is based on four basic components that are related to the conflict categories introduced in Chapter 5.

As shown in Figure 6.6 the Impact-time is subdivided into:

- Enforcement-time
- Visibility-delay
- Protection-time
- Relevance-time

The components and their close alignment to the conflict categories are explained in the following sections.

As introduced above, the main reason for the introduction of the Impact-time is the fact that subsequent SON-Functions with an overlapping Impact-area can be negatively affected by the executed function instance. One root cause for these negative effects lies mostly in the delays between a performed configuration change and visibility of the changes. Due to such delays the input values for a SON-Function instance do not reflect the reality which can lead to erroneous results.

There are also other timing related constellations between SON-Function instances that can have negative effects on the network and are therefore regulated through operational guidelines.

For example, repeated execution of SON-Functions with opposing actions on the same targets can lead to oscillating reconfiguration of network elements, which can have major negative effects on the operation reliability and efficiency. If only the execution time is considered during the function coordination, the malicious behavior would only be detected as long as the functions are concurrently executed. But often such negative effects are also visible in cases when the execution times do not overlap. Actions performed by one function can be relevant for the execution of subsequent functions over a longer time period.

In order to detect and prevent SON-Function instance conflicts information about SON-Function instances, especially their Impact-area, is required. Usually this information is lost, as soon as the SON-Function instance terminated. The introduction of the Impact-time can be used to prolong the availability of this important context information. The system is forced, at the end of the SON-Function instance execution, to maintain the information until the end of the SON-Function instance's Impact-time. The introduction of the Impact-time and the prolonged information availability allows the SON Coordinator to include information about previously executed SON-Function instances into the coordination decision as long as they are relevant for other SON-Function instances. There are several reasons why a previous SON-Function instance is still relevant for the execution of a subsequent SON-Function instance, for example, the performed changes need to be treated specially or because operational guidelines restrict subsequent configurations dependent on previous changes.

At the design-time an appropriate Impact-time has to be chosen for each SON-Function. In Section 6.2.2 it is shown how a detailed analysis of pairs of conflicting SON-Functions allows for the definition of the Impact-times.

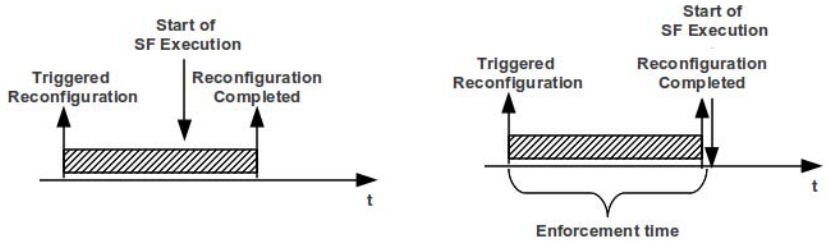
6.2.1.1 Enforcement-time

The Enforcement-time is used to prevent enforcement conflicts (cf. Section 5.3). As indicated in the conflict description, the reason for the conflict is the delay between the requested configuration change and its final enforcement.

The shaded box in Figure 6.7(a) shows the time span between the reconfiguration request and the completion of the reconfiguration. To prevent the situation that the changes performed by the first function are not visible to the second function, the Impact-time has to contain a time interval that reflects the delay between a configuration request and the final enforcement of configuration parameter values. Figure 6.7(b) shows the identical scenario as Figure 6.7(a) but with a delay enforced through the configured Enforcement-time. The execution of the requested SON-Function instance execution is delayed until the previous reconfigurations have been performed. The duration of the Enforcement-time is dependent on how NEs can be reconfigured and the way configuration management is performed. It can be assumed that for most of the SON-Functions within a network the same Enforcement-time can be used.

In order to minimize the negative effects for the users, network operators collect non-time-critical configuration changes during the day and execute them in batches during a particular maintenance window, mostly in low-traffic hours at night. In those cases larger variations of the Enforcement-times will occur. Such a mode of operation can have extensive negative consequences on the behavior and the efficiency of the SON system and is

therefore not recommended. It is much better to shift not only the reconfigurations but the complete SON-Function instance executions to low traffic hours, in that case an Enforcement-time can be used that reflects the actual enforcement delay.



(a) Without Enforcement-time (b) With Enforcement-time

Fig. 6.7: Enforcement-time

6.2.1.2 Visibility-delay

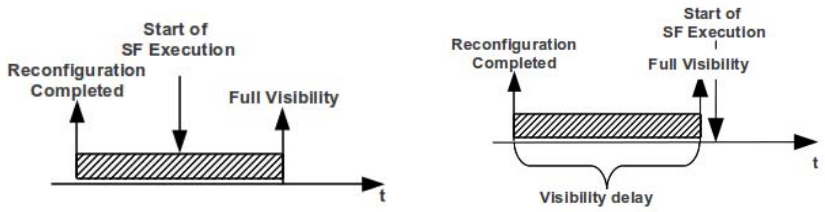
For the prevention of visibility conflicts the Impact-time also plays a central role. The reason for visibility conflicts (cf. Section 5.3) is the delay until an enforced configuration change is actually showing effects, due to a gradual communication towards the UEs. Such conflicts can be solved through a temporal alignment of the SON-Function execution that assures the usage of data that reliably reflects the performed configuration changes.

The main reason why the Visibility-delay needs to be represented in the Impact-time is the:

Input value protection: In case other SON-Function instances use the performance measurement values that are affected they could detect a non-existing triggering situation or the Algorithm-part processes non-up-to-date input values and produces erroneous results. This is especially important for subsequent instances of the same SON-Function with identical targets.

Figure 6.8 shows the difference between SON-Function execution with and without Visibility-delay. The Visibility-delay is added to the Impact-time directly after the Enforcement-time, it starts as soon as the reconfiguration is complete. The shaded box indicates the time while the measurement values do not sufficiently reflect the performed changes. In case no Visibility-delay is added to the Impact-time (cf. Figure 6.8(a)) the subsequent function uses erroneous input data, which is not the case if a Visibility-delay is used to align SON-Function execution (cf. Figure 6.8(b)).

For the specification of the Visibility-delay knowledge about the affected measurement values, how they are collected and how long it takes until they have stabilized after the enforcement of a configuration change is required. Based on this information, the SON-Function designer is able to define the



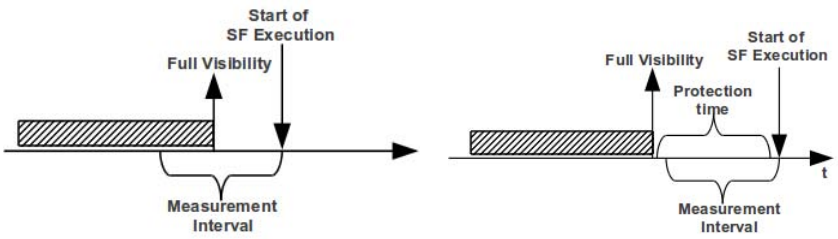
(a) Without Visibility-delay (b) With Visibility-delay

Fig. 6.8: Visibility-delay

Visibility-delay in an optimal length.

6.2.1.3 Protection-time

Measurement conflicts can also be resolved with additional timing information added to the Impact-time. The introduction of the Protection-time allows the function designer to specify the time over which the used performance measurement values need to reflect the actual state of the network. Through the added Protection-time it is possible to delay the SON-Function execution until the full effects of the previous configuration changes are statistically relevant reflected in the used performance measurement values as shown in Figure 6.9. The shaded box visualizes the time while the applied changes are not fully visible in the measurement values, and the measurement interval shows the time interval that is required to collect statistically significant data to be used by the SON-Function. Usually the Protection-



(a) Without Protection-time (b) With Protection-time

Fig. 6.9: Protection-time

time will be chosen in accordance to the time interval of the sliding window that is used to aggregate the measurement values. This allows to minimize the delay induced by the Protection-time.

6.2.1.4 Relevance-time

The Relevance-time is an important mean to allow the enforcement of operator guidelines. As the context information about the executed SON-Function instance is kept until the end of the Impact-time the Relevance-time can be used to extend the duration of the visibility. Through an extended visibility it is possible enforce operational goals, for example,to avoid negative behavior like oscillating reconfigurations, or similar. The length of the Relevance-time has therefore to be chosen as long as the context information is required to detect the violation of operational guidelines.

Figure 6.10 shows how the Relevance-time extends the visibility of a SON-Function. In contrast to the other parts of the Impact-time the length of the Relevance-time varies strongly, therefore it is almost impossible to give a single Relevance-time definition for a particular SON-Function but definitions for pairs of functions are required.

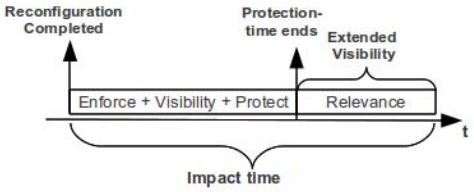


Fig. 6.10: Extended visibility through added Relevance-time

6.2.2 Definition of the Impact-time Components

As already indicated for the Relevance-time in Section 6.2.1.4 with the subdivided definition of the Impact-time it is not possible to specify "the" Impact-time for a given SON-Function. Impact-times need to be specified on the basis of pairs of potentially conflicting SON-Functions. At design-time it is therefore only possible to fully specify the Impact-times between already existing SON-Functions. Whenever new potentially conflicting SON-Functions are introduced the missing Impact-times have to be defined. The proposed method for the definition of Impact-times restricts the adaptation effort when new SON-Functions are introduced. Whenever new SON-Functions are deployed, only their requirements have to be defined at design-time and the final Impact-time can then be derived at run-time.

Table 6.1 gives an overview over the constituent parts of the Impact-time, the reasons they are used and where respectively for which SON-Function of the following scenario they are specified. To simplify the understanding, the table is based on a scenario with two potentially conflicting SON-Functions: SF1 and SF2. SF1 has been executed and the execution of SF2 is requested. The Impact-time for SF1 is defined from the view of SF2, which means how long has SF1 an impact on SF2 and how long ist the Impact-time that is required to prevent and resolve the conflict.

Tab. 6.1: Definition of the Impact-time for SON-Function SF1 from the Viewpoint of SF2

Impact-time Component	Reason	Conflict class (Chapter: 5)	Defined at
Enforcement-time	Time required to enforce new configurations requested by SF1. Protects other functions from using not yet updated configuration parameter values as input.	A	Defined at design-time for function SF1
Visibility-delay	Protects other SON-Functions to use measurement values that do yet fully reflect configuration changes performed by SF1	B	Defined for function SF1, with respect to known network and measurement characteristics
Protection-time	Protects Function SF2 from using erroneous measurement values in case SF2 uses measurement values aggregated over a sliding window time interval	B.2	Defined for SF2, only if the function requires measurements collected over some time. The Protection-time is selected in accordance to the collection interval. It is defined for SF2 with respect to SF1, as the the requirements on the measurement interval are only known for SF2.
Relevance-time	Assures the visibility of function SF1 as long as it is relevant for the execution of SF2. This information is only known during the design-time of SF2, and can be different for all SON-Function pairs of SF1 and another function.	A-C	Defined for function SF2.

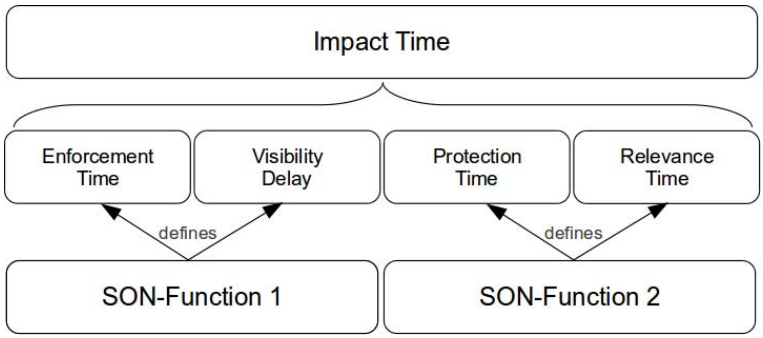


Fig. 6.11: Definition of the Impact-time of SF1 on SF2

Table 6.1 is structured as follows:

- **Impact-time Component:** States which part of the Impact-time is described
- **Reason** Specifies why this part of the Impact-time is required. Usually the Impact-time is used as a protector against a certain conflict or a conflict class
- **Conflict Class:** States the related conflict class according to the categorization introduced in Chapter 5
- **Defined at:** This column states for each Impact-time component whether it is specified during the design process of SF1 or SF2

6.2.2.1 Impact-time Definition Process

In the previous section it was shown that even though the Impact-time describes a time interval during which one function has an impact on another function, the components of the Impact-time come from both, the affecting and the affected, function. Figure 6.11 visualizes which part of the Impact-time are defined from which SON-Function in the above described scenario. This section introduces the design-time process that leads to the specification of the function specific Impact-times.

The Impact-time between two SON-Functions is specified at design-time as soon as a conflict or a relevance between two SON-Functions is found. During the design-time of SF1, three parts of the Impact-time are defined. A designer knows how long the enforcement of the configuration changes takes and he is able to estimate the time required for the affected measurement values to show the performed changes, which provides the possibility to already define Enforcement-time and Visibility-delay.

Additionally, the function designer knows the Protection-time requirements for SF1. It is important to remember that this is not the Protection-time used for the Impact-time of SF1 on SF2. The Protection-time for this

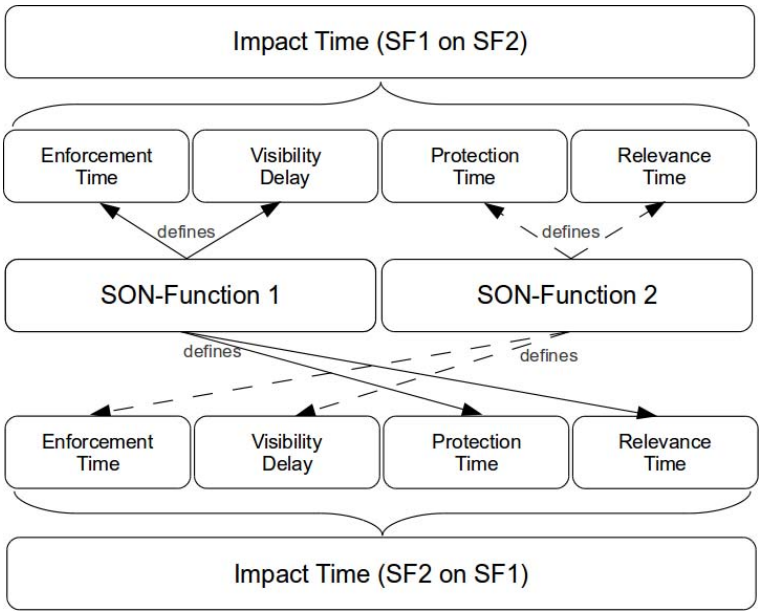


Fig. 6.12: Pairwise Definition of Impact-times for two SON-Functions

Impact-time is defined as part of the SF2 design process. Figure 6.12 shows an example with two conflicting SON-Functions. It shows which parts of the respective Impact-times are defined by which function.

The last missing part is the Relevance-time, which is specified as soon as an additional relevance from SF1 to SF2 is discovered for example as part of the operational goals.

Depending on the conflict type between the functions only particular components of the Impact-time are specified, which are combined at run-time according to the scheme shown in Figure 6.12. In the following listing it is shown for a set of conflicts which Impact-time components are required. The description uses the conflict naming scheme introduced in Section 5.5.

- **Configuration Conflict:** For a conflict of type A there is no Visibility-delay or Protection-time required, as there is no time delay until performance measurement values show meaningful results. For a pure parameter conflict, the Impact-time could consist of only the Enforcement-time. Relevance-time is an optional part, for example to be able to prevent oscillating reconfigurations.

- **Measurement Conflict:** Class B conflicts are rooted in the usage of measurements. In case SF1 affects measurements that are used by SF2, the Impact-time is built from the Enforcement-time and the Visibility-delay of SF1 and in case SF2 uses measurements collected over some time interval before the start of the execution, also the

Protection-time. The Relevance-time is again an optional part in this example.

6.2.3 Impact-time in combination with Granularity Periods

SON-Functions can be executed in a distributed way at NE level but also centrally at NM level, depending on the input information required and the overall setup of the SON System. This input information can thereby include configuration parameter values or performance measurement values from the NE, the UE, or core elements like the MME, or the Operations Support System (OSS) in general. Most of this input information used by SON-Functions is generated at NE level, independent from the execution location of the SON-Function itself. This can impose some difficulties when SON-Functions are executed at NM level.

In current networks performance measurement data coming from the NEs are not available in real-time at NM level for processing as there is no direct, immediate transmission. The data transmissions for real-time representation of performance measurement results at NM level would overload the capacity of the data connections dedicated for management tasks. To prevent such overload situations, the collected data are aggregated at NE level for a certain time interval and then, at the end of the so-called Granularity Period (cf. Section 2.2.2.1), compressed and uploaded to the NM systems. Typical granularity periods range between 5 minutes for important counters and KPIs up to several days for long-term statistics.

In some cases the collected data are provided with information on the temporal resolution to allow an analysis over time, but mostly only aggregated values are provided. For example, the overall number of failed handover attempts during the GP is provided but without a timestamp for each of the failed handover attempts. If a SON- Function has performed changes affecting the handover failure rate in the middle of a GP, the benefits of the executed function may be very hard to assess immediately, since the collected data provided at the end of this granularity period might still indicate a handover problem even though it has already been solved. The information on handover attempts, failures and successes are average values computed over the complete GP, therefore, in such a case, they contain also information about the state of the system before the configuration change. The full visibility of the performed changes within the collected data is only visible after the next complete GP cycle, when the effects are visible from the begin to the end of the GP.

This fact has to be considered for the definition of the Impact-times and the coordination logic used to resolve conflicts between the SON-Functions that operate at NM level, but target the NE level. Visibility-delay and Protection-time are the parts of the Impact-time that can be used to overcome the difficulties.

When specifying the components of the Impact-time and the respective coordination logic it is important to keep in mind which goals shall be reached: either to protect the monitoring results, or to protect the operation of the Algorithm-part. To protect the monitoring results it is necessary to guarantee that the collected data on which the monitoring bases its decision that a trigger situation has been identified, already fully reflects all configuration changes. For the execution of the Algorithm-part of the SON-Function instance it is important that it operates on clean input data.

6.2.3.1 Protection of Monitoring Decisions

The Monitoring-part of a SON-Function monitors a set of counters, KPIs or configuration parameter values in order to detect the trigger situation of the SON-Function. As a result of the monitoring process, the Monitoring-part will decide whether a trigger situation has occurred or not. For the soundness of the monitoring decision it is important that the previously performed configuration changes are long enough reflected within the input data that is used to determine the trigger situation. Regarding collected measurement and performance data this can either be a complete granularity period, but also long enough parts of the granularity period. "Long enough" denotes an operator specified time span within which the performed configuration changes have already had a strong impact on the aggregated data so that the effects of the configuration change can reliably be assessed. Obviously this is only required when no temporal resolution of the provided data is available.

The Monitoring-part of a SON-Function operates independently, and usually there is no possibility to inform it about performed configuration changes which affect its monitored measurement values. Therefore, the only possibility to assure that a triggering situation has been correctly detected, is to reject the requests for algorithm execution permission sent by the Monitoring-part until the usage of sound input data can be assured.

For the handling of these conflicts Impact-time plays a central role. Specifically Visibility-delay and Protection-time are used to enforce the usage of clean input data and have to be configured appropriately.

Figure 6.13 shows an example scenario. During the first GP between $t0$ and $t1$ the Enforcement-time of a SON-Function ends. From this moment the changes affect the measurement and performance data. The data that is collected until $t1$ contains performance data before and after the configuration change. The first complete data set that contains only aggregated values reflecting the state of the network after the performed changes is available from $t2$. If a full GP of data after a configuration change is required it has to be assured that requests from a Monitoring-part of a SON-Function are not acknowledged before $t2$. Therefore, Visibility-delay and Protection-time can be set at least equally long as a GP, which is shown in Figure 6.14. With

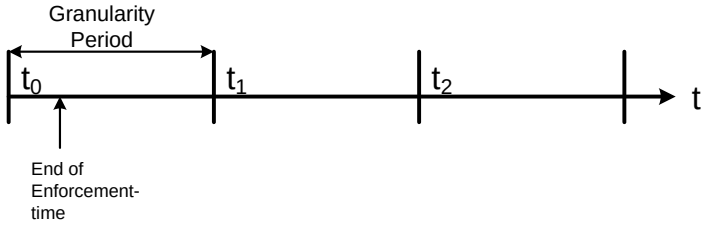


Fig. 6.13: Execution of a SON-Function instance during the Granularity Period

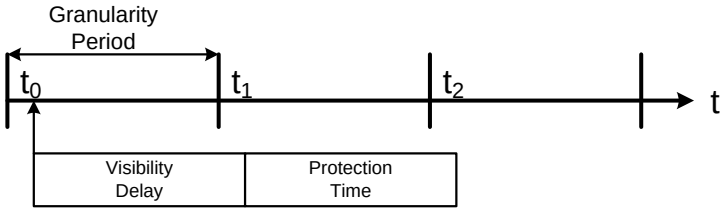


Fig. 6.14: Visibility-delay and Protection-time after Configuration enforcement

this setting the validity of the input data for the monitoring is guaranteed, even if the change is performed just after the beginning of the GP.

The coordination logic that is applied whenever algorithm execution requests have to be handled, needs to reject all requests until the end of the Protection-time. Figure 6.14 shows an issue that is caused by that way of dealing with GPs and SON-Function instance execution. Immediately with the availability of the clean data at $t2$ the SON Coordinator could acknowledge algorithm execution requests. But the Protection-time lasts a little bit longer. Between $t2$ and the end of the Protection-time requests are rejected, although the Monitoring-part identified the trigger situation based on valid input data and therefore the requested Algorithm-part should be executed. This situation gets worse if the end of the Enforcement-time is closer to $t1$, as shown in Figure 6.15. For almost a complete GP algorithm execution requests will be rejected, based on the assumption that the input data to the Monitoring-part used for the identification of the triggering situation is not valid. Depending on the frequency at which the Monitoring-part of the SON-Function sends algorithm execution requests, the overall SON system efficiency can be reduced as it may take a long time until the state of the network is evaluated the next time and the algorithm execution request is sent the next time .

To resolve the problem of erroneously rejected algorithm execution requests an extension to the way the Visibility-delay is set has to be introduced. Usually the Visibility-delay is a constant value that is defined at the

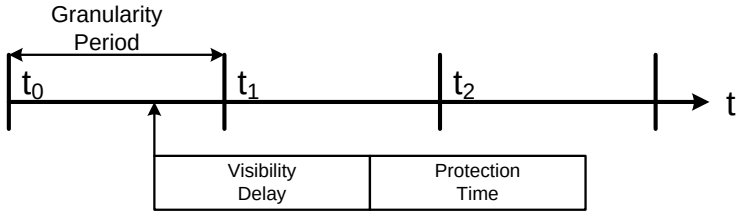


Fig. 6.15: Blocked time interval through Protection-time

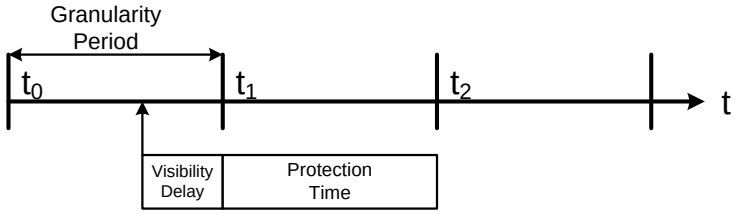


Fig. 6.16: Dynamic Visibility-delay ends at end of Granularity Period

design-time of the SON-Functions, which causes the unnecessary rejection of SON-Function algorithm execution requests. To circumvent potential blocking of other algorithm execution requests, a dynamic Visibility-delay which is always extended to the end of the GP during which the minimal required Visibility-delay ends is proposed. This approach requires additional knowledge about the length of the used GPs and their timing. Subsequently the Protection-time assures the collection of valid measurement and performance data during the next GP. The Monitoring-part of the SON-Function can base its decision on this new data and new algorithm execution requests will not be rejected due to a pending Protection-time. This mode of operation is shown in Figure 6.16.

6.2.3.2 Protection of Algorithm Execution

For the protection of the algorithm execution it has to be assured that the Algorithm-part of the SON-Function does not operate on data, which does not or not fully reflect the current state of the network. Therfore, the algorithm execution is not allowed to start before the end of a GP during which only measurement and performance data were collected that reflect all previous configuration changes. In contrast to the protection of the monitoring decision, there is the possibility to reschedule the algorithm execution. In that case, the request is not directly rejected or acknowledged but stored for some time. At the end of the rescheduling interval, the request is re-evaluated and an appropriate coordination decision based on the network context is taken.

For the protection of the collected input data for the algorithm execution, rescheduling does not provide any benefit. It is no solution to assume that it is possible to just reschedule the algorithm execution request to a point in time when the clean input data is available. In order to reliably take a rescheduling decision sound context information is required. This data has to provide information about previously executed SON-Functions and whether or not the collected data does reflect the current state of the network. Since this information is only available if the Impact-time has not yet expired, Visibility-delay and Protection-time have to cover the GP during which the clean measurements and performance data are collected and aggregated.

6.3 SON-Function Instances Summary and Findings

The introduced spatial and temporal characteristics of SON-Functions are key information for the detection and resolution of SON-Function conflicts and therefore provide an important contribution to the research questions. The combination of Impact-area and Impact-time allows the system to detect conflicting SON-Function instances and prevent the conflicting behavior.

Without a proper definition of the Impact-area it is impossible to determine whether conflict free execution of SON-Function instances can be performed, since the SON-Function instances operating in the same area cannot be identified. To guarantee conflict operation without the definition of Impact-areas a full serialization of SON-Function instance execution would be the only possible solution.

Therefore, the definition of Impact-areas for SON-Function instances contributes to the efficiency and the robustness of the operation of a SON enabled network.

The same applies to the temporal characteristic of the SON-Function instances. The Impact-time definition extends the visibility of the SON-Function instances which allows to detect and prevent some conflict types which would be otherwise not detectable.

The presented characteristics of SON-Function instances contributes to several research questions that were introduced in Chapter 1.3:

Concerning the research area Reliability and Robustness, a contribution to the following research question has been made:

R6 Are there characteristics of SON-Functions that can support the detection of run-time conflicts?

The specification of Impact-time and Impact-area as the temporal and spatial characteristics of a SON-Function instance is an im-

portant contribution to the run-time conflict detection and prevention. Only if this information is available it is possible to determine if two potentially conflicting SON-Functions are actually conflicting. The specification approaches have been designed to cover the requirements of the SON-Functions, the network and the systems involved into the coordination process. For example, the Impact-area specification allows to consider not only directly but also indirectly affected areas. Mapping the description of the Impact-area, based on existing relationships in the network, into a graph representation allows the usage of well known, formal methods for automatic computation of the concrete Impact-area. This approach can be used generally for Impact-areas that can be described based on relationships between entities. Since it is possible to use very generic relationships, like the geographical distance, there is no situation where no Impact-area can be specified. With the provided specification approaches a simple run-time computation of Impact-area and Impact-time for each running SON-Function instance is possible, with only minimal additional information required.

Concerning the research area Efficiency, a contribution to the following research question has been made:

E6 How can the number of concurrently, conflict free executed SON-Functions be maximized?

Parallel conflict free execution of SON-Functions is important for the efficiency of the management processes. The knowledge about the spatial and temporal characteristics of SON-Function instances allows to acknowledge the parallel execution of all SON-Function instances that have a non-overlapping spatial extent and maximize the number of concurrently running SON-Function instances. If this information would not be available, the only way to prevent conflicting behavior is the full serialization of SON-Function instance executions. If there was a common starting time for all SON-Function instances, the knowledge about the Impact-area could be used to find the best combination of SON-Function instances to reach the absolute possible maximum of concurrently running SON-Functions. This is not the case in a fully SON enabled network, therefore it is possible to be in a situation where no more SON-Function instances can be acknowledged, but due to SON-Function instances with very large Impact-areas, many SON-Function instances execution requests are rejected.

Concerning the research area Flexibility, a contribution to the following research question has been made:

F4 Is the concept sufficiently flexible to be used in networks with different operational modes?

There is a main difference between the operation of a fully SON enabled and a typical network today. In a fully SON enabled network it is assumed that the SON-Function instances operate absolutely independent. They have the possibility to continuously monitor their input KPIs and will request algorithm execution as soon as a triggering situation is detected. They will also directly, without any further delay, enforce the required configuration changes to resolve the detected issues. This is not possible in todays networks. Input data as, for example KPIs, is provided at the end of fixed GPs and the configuration changes have to be performed through dedicated CM systems, potentially during dedicated maintenance windows. Section 6.2.3 shows how the Impact-time can be defined in a way to cope with this particular mode of operation. Only by using an appropriate specification of the Impact-time, this radically different mode of operation can be covered. The flexibility of the Impact-time allows to use the presented approach with all types of operational modes, from todays, GP controlled operation to a fully SON enabled network, therefore it can be used throughout the transition period when more an more of the SON concepts are introduced into network management.

7. SON-FUNCTION COORDINATION CONCEPT

The previous chapters showed that there exists a considerable number of potential conflicts between different SON-Functions. In order to benefit from the advantages of autonomic SON-Function operation, conflicting behavior between concurrently executed SON-Functions has to be detected, prevented or resolved at run-time.

Today, in traditional network management and operation, human network operators have the duty to guarantee the conflict-free execution of management tasks. This requires a very good understanding of the networks and the interdependencies of the different management tasks and also a detailed insight into the ongoing and previously performed management tasks in the network. A network operator will combine his operational background knowledge with performance measurements to assess the current state of the network and, if required, trigger appropriate management actions which are not conflicting. In the same way as conflict-free execution has to be assured, high efficiency of the network management is required. While failures within the network have to be treated with minimal delay, also network optimization is equally important to satisfy the user requirements. The operational staff has to take care that all failures are treated and the network operation is continuously optimized without any conflicting tasks being performed.

7.1 Conflict Resolution Approaches

In a SON enabled network instances of SON-Functions will be autonomically triggered to perform specific tasks that resolve detected triggering situations. Instances of different SON-Functions which have been categorized as conflicting are called potentially conflicting SON-Function instances. As long as there is no spatial or temporal overlap between the individual SON-Function instances a large number of potentially conflicting SON-Function instances can be concurrently and fully conflict-free executed. Only if there is an overlap and therefore a conflict between two SON-Function instances this needs to be resolved. The challenges of conflict-free operation for autonomic network management have been identified by earlier research [BSDB06, BK10, DV09, SAE+11] and therefore it is widely accepted that automatic methods are required to govern the execution of individual SON-Functions in a wider system-level context.

There are several approaches to avoid conflicting behavior which are shortly introduced in the following sections:

- **SON-Function Co-design**
- **SON-Function Harmonization**
- **SON-Function Coordination**

7.1.1 Design-time SON-Function Co-Design

Run-time prevention and resolution of SON-Function conflicts can be difficult to accomplish especially if there is a larger number of potentially conflicting functions deployed within a network. Therefore a resolution of potential conflicts at design-time of the SON-Functions is a promising approach. The conflict analysis in Chapter 5 showed that most of the conflicts between SON-Functions are rooted in the operation on a shared set of configuration parameters, performance measurement values or characteristics of a cell. Co-design tries to resolve this situation by providing functions that operate on disjoint input values and do not affect the same performance measurements or configuration parameters. The resulting functions will not be conflicting at run-time.

The following guidelines have to be followed for SON-Function Co-design:

- Reduction of the number of shared parameters by pre-assigning clear responsibilities for parameters to SON-Functions to the maximal possible extent. The goal is that SON-Function instances operate on fully disjoint parameter sets
- If required, the functionality of several SON-Functions should be merged into a single function, for example to create an optimization function that optimises several target parameters at the same time instead of having multiple single target parameter SON-Functions
- Exclusion of SON conflicts a priori by co-designing SON-Functions as a "function group". Within a function group, there is a defined run-time interaction between the individual functions which assures conflict-free behaviour. The co-design of the Mobility Load Balancing (MLB) and MRO SON-Functions is an example of this type of conflict prevention. Indepentend function execution could cancel the intended handover performance gain. After the co-design MLB will set limiting values within which MRO is allowed to operate.
- Assurance of required SON-Function interactions in a way that they are not excluded by some other constraints.

Figures 7.1 and 7.2 visualize an example of SON-Function co-design. In the beginning there is a set of SON-Functions as shown in Figure 7.1 the parameters are represented by different shapes. After the co-design process following the above guidelines, the number of shared parameters has been reduced. The result is a function group with dedicated interaction (cf. Figure 7.2(a)). Other functions that still share a single parameter have been

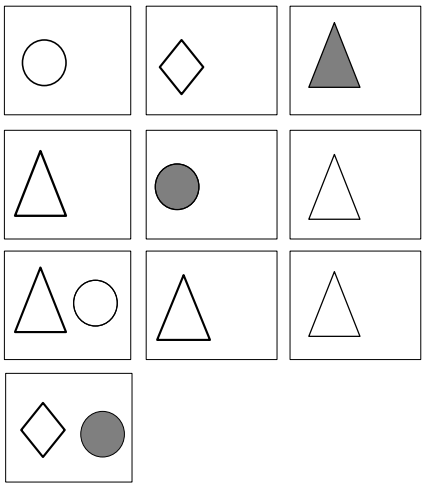
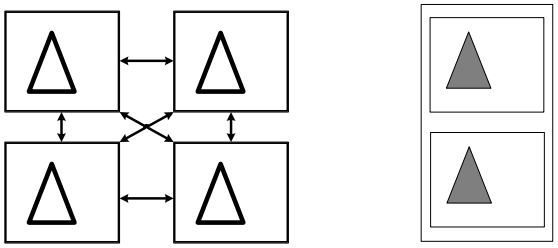


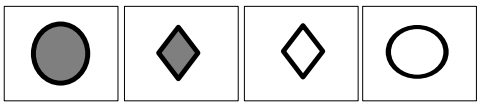
Fig. 7.1: Set of Non Co-designed SON-Functions

merged into a single SON-Function (Figure 7.2(b)). The remaining SON-Function could be designed in a way that they do not share any parameters with other SON-Functions (Figure 7.2(c)).



(a) SON-Function Group

(b) Merged SON-Functions



(c) SON-Functions with disjoint paramter sets

Fig. 7.2: Result after Co-Design

The benefit of SON-Function co-design is that it is often a simple and foremost feasible approach when there are only a few SON-Functions from a single provider in the network. SON-Function co-design as a method to avoid run-time conflicts reaches its limits as soon as the overall number of deployed SON-Function increases and especially if the set of deployed functions is changed at run-time. The introduction of additional functions could

require a re-design of already deployed functions, which increases the overall overhead and possibly creates new dependencies between functions. Even with a theoretic possibility to design only SON-Functions with disjoint parameter sets, a major problem persists. Co-design of SON-Functions which are provided from different vendors (which did only marginally or not at all align their work) is not possible. There are additional properties and requirements of SON-Functions and SON in general which can hardly be covered by co-design. SON-Function co-design relies mainly on the systematic analysis of the configuration parameters, KPIs and performance measurements used or targeted by the functions, which can even be done automatically but fails for conflicts that are not directly reflected in the parameters that are related to a given SON-Function.

There is also a situation where some interrelations that a parameter analysis would reveal as conflicts are fully acceptable or even required for the proper operation of the network. Typical examples are SON-Functions that target network optimisation and failure recovery functions. There is a high probability that those functions do not only operate on identical parameter but actually are in conflict. Cell outage compensation functions will try to re-establish full coverage as fast as possible. This is conflicting with functions that try to reach an equal load distribution within all cells. This conflict should not be removed but needs to be handled otherwise. Either by creating a combined SON-Function that merges failure recovery and optimization or integrates run-time interaction capabilities to allow the functions to solve the conflicts. Again it is important to note that this approach is only feasible with a few functions from a single function provider.

But even if SON-Function co-design is not applicable as the exclusive conflict prevention solution, the principles can be used to significantly reduce the amount of potential SON conflicts. This, in turn reduces the number of conflicts that need to be handled separately and facilitates run-time conflict resolution by SON-Function coordination.

7.1.2 SON-Function Harmonization

Harmonization of SON-Function actions is a method for run-time conflict resolution. The approach follows the idea to find a solution for a conflict that satisfies the requirements of all conflicting SON-Function instances. It can be performed either by a central entity or through interaction between the conflicting functions or even hybrid solution where a central entity mediates between the conflicting SON-Function instances. In order to satisfy all requirements the configuration requests of all conflicting parties are evaluated to find a configuration that can be enforced. The obvious benefit is that all conflicting function instances are able to perform their tasks and there is no need to give precedence to one or another function. Harmonization is the conflict resolution approach used within the SOCRATES Project [Sch08].

There are some central shortcomings in this approach, due to which it is not feasible within a network with a larger number of deployed NEs and SON-Functions. If harmonization relies only on a central mediator, much of the functionality of the SON-Functions has to be replicated in the mediator to allow this entity to determine which configurations are still acceptable for the SON-Functions. In this case, each time new SON-Functions are being introduced the mediator has to be adapted. With a centralized mediator the requests have to be available at the same time to allow the harmonization of the requests, which does not comply with the general mode of SON-Function operation where requests are sent whenever the triggering situation has been detected.

A way to circumvent the concentration of functionality and the very strict timing requirement is to employ a negotiation method between the potentially conflicting SON-Function instances. Whenever a SON-Function instance is triggered it will communicate with all other potentially conflicting SON-Function instances to determine if the proposed configuration changes are acceptable. Apart from the need to include extended communication capabilities into the SON-Functions it will also mandate to trigger instances for all potentially conflicting SON-Functions whenever a single SON-Function requires configuration changes to guarantee consistent results. The execution time of the SON-Functions can vary largely. Therefore the enforcement of a high priority failure recovery configuration could be delayed by a long running potentially conflicting optimization SON-Function.

Independently from which harmonization approach is chosen, the scalability and efficiency of the approach is questionable. Either a central function unifies a lot of functionality or a large number of potentially conflicting SON-Functions has to be triggered each time a triggering situation fo a single SON-Function is detected. Looking at the set of potential conflicts between SON-Functions (cf. Table 5.1) it becomes obvious that the focus of harmonization lies on A2 conflicts. There is no explicit possibility to resolve A1 or category B conflicts. When looking at the SON use case descriptions it becomes obvious that especially category B conflicts are the most common conflicts between SON-Functions.

7.1.3 SON-Function Coordination

The goal of the presented SON-Function coordination approach is to provide a functionality that acts similar to todas human operator. It is a logical next step if the co-design is not capable to provide fully conflict-free SON-Functions. Coordination operates on a per SON-Function instance basis. Whenever the Monitoring-part of a SON-Function detects a triggering situation, the SON Coordinator evaluates the current state of the network and information about previously executed SON-Functions to determine if it is safe to run a particular SON-Function instance, as shown in Figure 7.3.

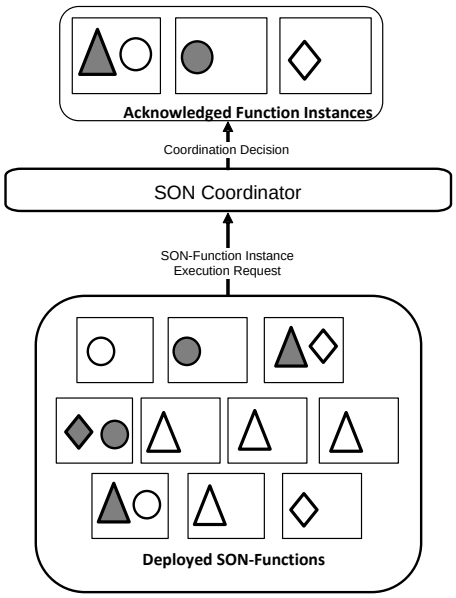


Fig. 7.3: Operation of the SON Coordinator

Coordinating SON-Function instances provides benefits in addition to the conflict detection and resolution. It combines to basic ideas: Low-level coordination of autonomically operating SON-Function instances for conflict resolution and prevention, and at the same time give full control over the SON enabled system to the human operator to govern the system behavior. This operator control is realized through the enforcement of high level operator requirements without the need for continuous management interactions. The SON Coordinator enforces operational guidelines which are, for example, based on KPI thresholds, detected conflicts between SON-Function instances, and the general network context of a requested SON-Function instance. SON-Function instances request the permission to execute their tasks after their Monitoring-part has detected a triggering situation, they are not triggered by the SON Coordinator.
The SON-Function instance execution coordination concept is based on the following principles:

- Prevention of undesired SON-Function conflicts
- Support of required SON-Function interactions
- SON-Function instance execution is under full SON Coordinator control - no SON-Function instance is executed without being explicitly acknowledged
- The SON Coordinator must be able to prioritize function instances and also preempt already running function instances
- All SON Coordinator decisions are based on the operator requirements,

state of the network, and the characteristics of the requested SON-Functions

SON-Function instance execution coordination based on these principles provides a powerful tool to detect, prevent and resolve SON-Function instance conflicts and enforce operational goals and guidelines while assuring a very high efficiency of the overall SON system. The coordination of SON-Function instances is the key for enabling full automated network management with SON-Functions. It allows:

- **Active SON-Function instance Protection:** is the most obvious task that is performed. Running SON-Function instances need to be protected against negative impacts caused by conflicting instances by accepting or rejecting SON-Function instance execution requests.
- **Pro-active SON-Function instance Protection:** is performed to avoid future conflicts. Pro-active protection targets future SON-Function instances. As a result of changes performed by a SON-Function instance often some particular function instances need to be executed subsequently to guarantee the overall consistency of the network configuration.
  Conflicts between currently executed and future functions (cf. Figure 7.4) can prevent the successful execution of the future functions. A coordination decision based on knowledge about such dependencies allows to pro-actively protect subsequent function instances. Deadlocks or priority-based function termination due to increased Impact-areas are typical examples for future conflicts. If a currently running SON-Function instance will trigger the execution of a SON-Function instance with a larger Impact-area, this can cause a conflict with the requested or one of the subsequent SON-Function instances. Such a future conflict can be omitted through an appropriate coordination decision. Future conflicts are obviously much harder to assess at run-time than conflicts between active and requested SON-Functions instances. To be able to use the coordination function to perform this kind of pro-active protection, knowledge about typical sequences of functions and their characteristics has to be used as input to the design-process of the coordination logic.
- **Enforcement of management goals:** Each network operator has high level management goals which are used to govern the overall network operation, for example: "prioritize full coverage availability over load optimization." Such a general policy is also required to resolve conflicts between failure recovery and optimisation functions. The SON Coordinator is used to enforce those high level management goals by giving priority to function instances in a way that those management goals are met. But also other operator policies like maximum change values for certain parameters can be enforced via the SON Coordinator. The applied coordination logic has to assure that in the

long run all operational goals are met, and that there are no functions which are blocked forever.

- A **high efficiency of the overall SON system** strongly supported through SON-Function coordination. Several individual parts of the coordination contribute to the efficiency of the system. A first important part is parallelisation: the SON Coordinator acts as a kind of smart scheduler. In a network with thousands of network elements, spread over a wide area, several potentially conflicting functions can safely be executed in parallel as long as their Impact-areas do not intersect. The SON Coordinator will try to acknowledge as many concurrent non-conflicting SON-Function instances.
  A second important part that determines the efficiency of the system is whether a SON-Function instance has all intended effects, or if the effects are reduced or even eliminated through conflicting functions. Active and pro-active protection together with operational guidelines are used to assure that intended effects are reached.
  - Protection of already reached effects by preventing execution of function instances that counteract the reached effects
  - Rejection of function instances whose changes will not be enforced due to or undone by other (potentially future) functions.

  If such effects can be foreseen, for instance, due to higher priorities of subsequent functions, blocking function instances will serve the overall efficiency.

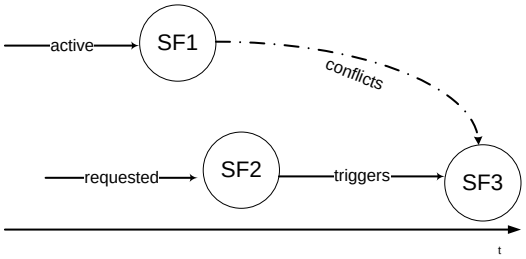


Fig. 7.4: Future SON-Function instance conflict

7.2 Information Requirements for SON Coordination

In order to allow the SON Coordinator to efficiently take the right coordination decisions some prerequisites have to be fulfilled. Important input to the SON Coordinator has to be prepared as for example the coordination logic which is applied whenever a SON-Function instance execution is requested. There are different ways how a SON-Function can be coordinated, which means that already at design-time an appropriate coordination scheme has to be assigned to each SON-Function. An important input to the

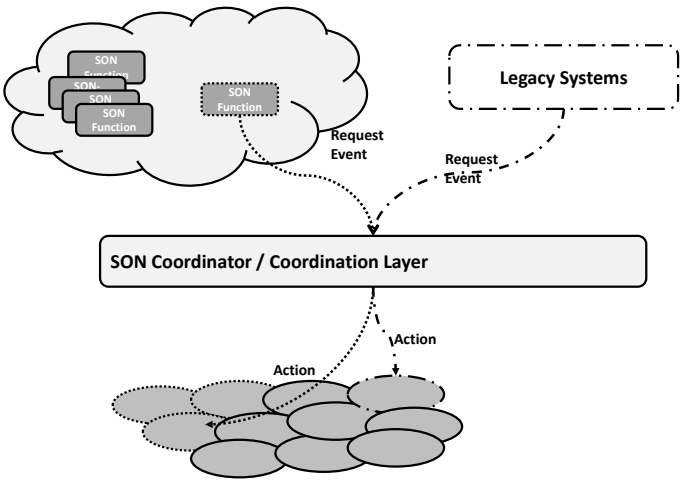


Fig. 7.5: SON Coordinator as Intermediate Layer between SON functions, operators and NEs

coordination process is also the context information that is combined with the coordination logic to reach a decision whether the requested function instance should be executed or not. Used context information is mainly information about previously executed SON-Function instances with a shared Impact-area and an Impact-time that has not yet expired but also additional information can be required if for example there are variations of the coordination logic for specific situations (time, region, RAN,...).

Although this input is required for the SON-Function instance execution coordination at run-time, it can be prepared at design-time to a large extent. Preparing the coordination logic and assigning the coordination schemes are mainly design-time tasks but they can be adapted at run-time to satisfy new requirements. Efficient context information handling is an important run-time task. It has to be assured that all required information is available, but not-needed information should be discarded as early as possible to avoid unnecessary operational overhead.

For consistent coordination it is important that all management interactions are made visible to the SON Coordinator, independent if they are performed by SON-Functions or the human operator through the legacy network management systems. The SON Coordinator will operate as an intermediate layer between SON-Functions, human network operators and the network itself as shown in Figure 7.5.

This setup gives the SON Coordinator not only all required information but also the possibility to even coordinate the tasks performed by the human operator. It is assumed that tasks from the operational staff will usually be given highest priority but the network operator has the possibility to change this accordingly and prioritize them differently.

7.2.1 Coordination Logic

An important part of the SON-Function design-process is the definition of the coordination logic. The goal is to provide the SON Coordinator with the instructions that are required to take the coordination decision. The coordination logic is defined after the conflict analysis, and the specification of the Impact-area and the Impact-time for the new SON-Function.

The results of the conflict analysis form the starting point for the specification of the coordination logic, as it has to cover all conflicts revealed. The list of potential conflicts is combined with operational knowledge and the questions: "How should a given conflict be resolved? Which of the involved SON-Functions has a higher priority and should therefore be given precedence, or how can it be handled if both functions have the same priority?"

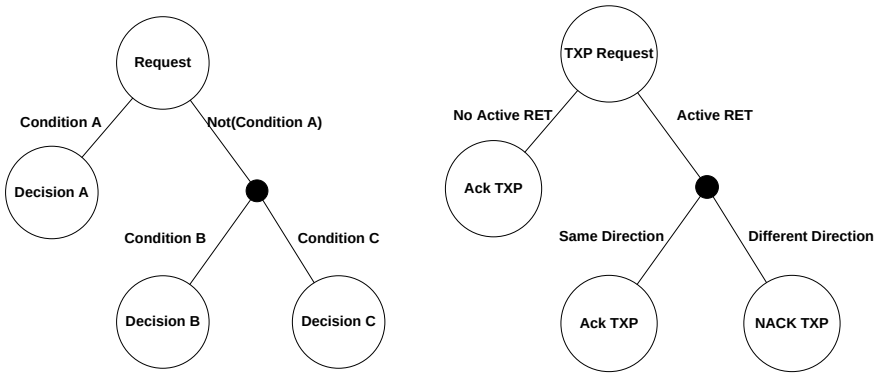
To keep the development complexity at a manageable level the set of possible coordination decisions is restricted to:

- **Acknowledge** the execution of the requested SON-Function instance algorithm or action respectively, if no conflicting SON-Function instances have been detected.
- **Reject** the request, either because a conflicting SON-Function instance has been detected or the operational guidelines object the algorithm or action execution.
- **Pre-empt** the conflicting running SON-Function instance. This decision is mostly combined with an subsequent acknowledgement of the current request.
- **Reschedule** the request. This decision has been introduced to increase the efficiency of the system. It is used in two situations. First, if a SON-Function instance is pre-empted but should be executed as soon as possible after the higher priority SON-Function instance has been executed. Second, if the requested instance should be rejected but it is known that the Monitoring-part of this SON-Function will not re-evaluate the situation for a very long time. The solution is to schedule a SON-Function instance execution request that is re-inserted into the SON Coordinator after a predefined rescheduling time. This subsequent request will be treated by the SON Coordinator in the same way as a new request.

For each SON-Function an appropriate coordination logic has to be developed that covers all the conflicts and all respective operational guidelines.

7.2.1.1 Representation of the Coordination Logic

It is an important question how the developed coordination logic is stored and represented in a solution agnostic way. The techniques used within the SON Coordinator to take the coordination decisions may be changed over time, but the designed coordination logic should still be usable with new



(a) Generic Decision Tree

(b) SON-Function Specific Decision Tree (CCO(TXP))

Fig. 7.6: Decision Trees representing Coordination Logic

decision making technologies.

The decision logic that is applied in response to a SON-Function instance execution request is represented in solution-agnostic decision trees. They capture dependencies and interactions of a given function with other deployed functions. For each SON-Function they provide a sequence of conditions that need to be evaluated to take a coordination decision. An example of a generic decision tree is shown in Figure 7.6(a). The root of the tree represents the SON-Function instance execution request. The decision logic is annotated at the edges between the nodes. Typically the decision logic forces the SON Coordinator to perform different types of evaluations:

- Check whether one or more SON-Function instances are currently active within the Impact-area of the requested SON-Function instance
- Evaluate the requested changes whether they are in conflict with previously executed changes to the targeted network elements or any operational guidelines

The leaves of the decision trees contain the possible results of the decision process. The tree representation of the decision logic gives a good indication on how decision trees should be constructed to support efficient decision making. Decisions that lead to an instantaneous decision should be placed in the upper part of the tree. The more information is required for a particular decision the lower it should be placed within the tree. For example if there is a conflicting SON-Function, which, if an instance is present, leads to a direct rejection of the requested SON-Function instance execution, the respective check should be placed close to the root of the decision tree. Such a tree organisation facilitates the understanding of the decision logic and speeds up the overall coordination process.

Independent from the used technology, the availability of decision trees simplifies the actual implementation of the decision making. On the occurrence of a SON-Function instance execution request, the SON Coordinator only has to traverse the respective decision tree until it reaches a leaf. The result contained in the leaf is then enforced as coordination decision.

In order to allow successful SON-Function instance execution coordination, the construction of the decision trees has to be an integral part of the SON-Function design process. If new SON-Functions are introduced into a SON enabled network, the initially created decision trees have to be adapted to fit into the environment. This concept reduces the information required to take a decision to a minimal level and targets to minimize the overhead for the introduction of new SON-Functions.

In case the conflict analysis of a new SON-Function reveals conflicts with already deployed SON-Functions only the decision trees of those conflicting functions have to be re-evaluated whether they need to be adapted, while the decision trees for all other SON-Functions stays untouched.

The SON-Function instance specific information required in the SON Coordinator for the decision making is minimal. Therefore it can be provided together with the SON-Function instance execution request to the SON Coordinator. For most of the SON-Function instances it is sufficient to provide only information about the directly targeted NEs, given that the SON Coordinator has the ability to access the abstract Impact-area and Impact-time definitions and use them to derive concrete information for the SON-Function instance. The required information can even be further reduced if individual SON-Functions with identical Impact-time or Impact-area requirements are assigned to SON-Function classes. This reduces the overall number of abstract definitions which need to be provided, maintained and stored.

7.2.1.1.1 Coordination Logic Example The conflicting behavior of the CCO functions (CCO(RET),CCO(TXP)) is easy to understand. Both functions try to optimize coverage and capacity of a cell. To reach their goal, they modify the size of the coverage area of cells either by adapting the antenna tilt angle or the transmission power. Due to the higher impact, the CCO(RET) function should be assigned the higher priority. An additional condition for the execution of the CCO(TXP) function is postulated based on operational experience: If a CCO(RET) function is active only power changes in the same direction as the previous tilt change is allowed. A SON-Function instance is considered active if its Impact-time has not yet expired. The resulting decision tree for CCO(TXP) shown in Figure 7.6(b).

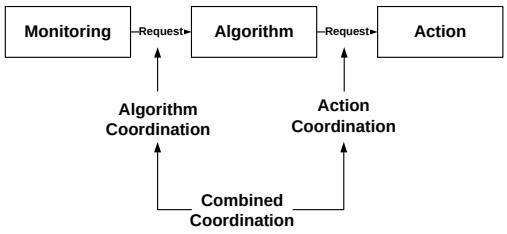


Fig. 7.7: Interaction Points between SON-Function and Coordination Scheme

7.2.2 Coordination Schemes

The example in Section 7.2.1.1.1 shows that the coordination of SON-Function instances requires information about the coordinated SON-Function instance which is not available in any phase of the SON-Function instance execution. For the initial decision it is sufficient to know the targeted NEs to be able to compute the concrete Impact-area. But for the second decision, information is required that is not available before the end of the Algorithm-part. Due to the availability of instance specific data the coordination can only be performed at the transition between two SON-Function parts when all required information is available.

Coordination schemes describe at which transition between the functional building blocks of a SON-Function the coordination process can provide a valid decision. This section gives an overview on the three generic types of coordination schemes that are defined. Two single-phase coordination schemes (Algorithm and Action Coordination) and a multi-phase coordination scheme that combines the two single-phase schemes. The advantages and disadvantages of each of the coordination schemes are presented. It is shown, why based on the conceptual setup of a SON-Function (cf. Figure 4.1), Impact-time, Impact-area, and conflict analysis, a particular coordination scheme should be applied to a particular SON-Function.

Figure 7.7 shows the logical separation of a SON-Function and indicates at which of the transitions the respective coordination schemes interact with the SON-Function.

7.2.2.1 Action Coordination

The most obvious way of coordinating SON-Functions is to coordinate the action execution, since the actions show the largest conflict potential. Therefore it is beneficial to be able to block conflicting actions. All potential conflict types (cf. Chapter 5) can be related to action execution. Actions are mostly reconfigurations and are therefore subject to output configuration conflicts (Type A.2 conflict) but also input parameter conflicts are possible (Type A.1 conflict) in case the configuration parameter values being

input to the Algorithm-part are subject to change during the run-time of the algorithm. Apart from parameter conflicts, also measurement and characteristic conflicts are possible, as the Algorithm-part potentially consumes measurements values and affects network and cell characteristics as well as measurements by requesting the enforcement of new configurations through the Action-part.

If action coordination is used, a SON-Function instance requests approval to execute its action from the SON Coordinator. Typically such a request contains at least the new configurations that should be enforced. This allows the SON Coordinator to compare the request against the current configuration or, if available, to previously requested configuration changes. Today, the comparisons done by the SON Coordinator are rather simple, they are mainly used to enforce operator policies which prevent abrupt or oscillating configuration changes. It is important to note that, in contrast to the harmonization approach, the SON Coordinator does not need to estimate the effects of the requested changes on the network and use them for the coordination decision.

The possibility to access all results of the algorithm and evaluate the requested configuration changes is a major benefit of the action coordination. This possibility in combination with a long relevance time allows the SON Coordinator to base the coordination decision on detailed information not only from the current request but also on previously applied configuration changes. It opens the possibility to perform sanity checks and suppress sudden strong or subsequent opposing configuration changes.

This is especially important for SON-Functions that influence important characteristics within the network by changing different configuration parameter values as, for example, SON-Functions that have an influence on the coverage area of cells. If repeatedly the transmission power of a base station is increased followed by a down-tilt of the antenna this is a strong indication of oscillating reconfigurations, which can easily be detected if action coordination is used.

Also from an efficiency viewpoint, action coordination is very appealing, as it reduces the time interval during which the SON-Function instance is considered to be active. The time required to execute the Algorithm-part does not have to be considered in addition to the Impact-time of the SON-Function instance.

The main disadvantage of action coordination in terms of operational efficiency is the fact that the Algorithm-parts of the SON-Functions are always executed. Only after receiving an action execution request, the SON Coordinator will evaluate which potentially conflicting functions have been executed in the same Impact-area, whose Impact-time has not yet expired.

The execution of SON-Function Algorithm-parts whose results are not enforced cause a higher load on the overall system which reduces the operational efficiency. The time required to execute a SON-Function instance can

differ strongly. Some SON-Function instances require only a few minutes or seconds to execute Algorithm- and Action-part, others can take much longer, for example, several hours. The disadvantage of the action coordination is increased if the results of long-running SON-Function instances are not enforced due to short-running, high priority SON-Function instances.

The problem is that the short-running functions can be started after the start of the execution of a long-running function, but they will still request the enforcement of their algorithm results before the long-running function. If no information about previous long-running function instances is available, the action execution request of the short-running instance will be acknowledged by the SON Coordinator. Even if the long-running SON-Function instance has a higher priority there is a possibility that it will be overruled by the short-running, lower priority SON-Function instance.

This behavior is rooted in the fact that information about running SON-Function instances is not included into the context information before the action execution request. If only action coordination is used, the SON Coordinator performs two tasks: firstly to evaluate the context on potential conflicts with other SON-Function instances and secondly to evaluate the configuration change values on their compliance with previous configuration changes and operator guidelines. If there is no reason to reject the coordination request for the Action-part, action execution will be admitted.

For several reasons this is not the best mode of operation.:

- Computational overhead depending on the number of rejected action requests. A large number of algorithms could have been executed without any benefit. This is especially the case whenever algorithm results are irrelevant for the coordination decision. The analysis of known SON-Functions has shown that for many combinations of conflicting SON-Functions the only information required for the decision making is whether another SON-Function instance with an intersecting Impact-area is active.
- The soundness of input data either for identifying triggering situations or for computing new configuration parameter settings is something that is neglected or at least hard to reach with action coordination. The Impact-time of another SON-Function instance that has been active at the point in time when the algorithm has been triggered can easily time out until the coordination is performed. Therefore, it is possible that measurement values which caused the other algorithm to be executed or are used as input to the other algorithm did not fully reflect the already performed changes. If the Impact-time of the first SON-Function has already timed out the potentially erroneous changes requested by the action of the later function will be enforced. Therefore the requirement that a monitoring function does not trig-

ger a SON-Function algorithm based on measurement values that do not fully reflect the current state of the network, cannot be met with action coordination.

7.2.2.2 Algorithm Coordination

For many SON-Function conflicts knowledge about the concrete changes of the configuration parameter values is not required to be able to take a coordination decision. It is mostly sufficient to know whether a potentially conflicting SON-Function instance is active on a shared target.

Algorithm coordination addresses the action coordination drawback of unnecessarily executed Algorithm-parts. Coordination is done on the transition between the monitoring and the Algorithm-part of a SON-Function instance. The main benefit is the operation on the algorithm execution requests, which allows coordination decisions **before** the algorithm execution. Apart from the earlier decision making, information about active SON-Function instances is available from the earliest possible point in time.

This early acquisition of information allows to prevent the blocking of high-priority long-running SON-Function instances through short-running, low-priority SON-Function instances. As soon as the SON Coordinator has acknowledged the algorithm execution of a long-running SON-Function, this information is available for further coordination decisions. If a low priority SON-Function instance requests the acknowledgement for algorithm execution, the request will be rejected while the Impact-time of the long-running function has not yet timed out. In combination with a well specified Impact-area, algorithm coordination allows the protection of subsequently triggered SON-Functions as it protects an area which is wider than the spatial requirements of the currently executed SON-Function.

Another issue that is targeted is the protection on input values, especially of the input values for the algorithm execution. In case a SON-Function operates on performance measurement values that are also affected by another function, visibility delay and protection time are used to guarantee that only stable measurement values are used. When an algorithm coordination request is received, the SON Coordinator has the possibility to on the one hand evaluate whether the monitoring has detected the triggering situation based on stable values, but on the other hand also whether it is safe to acknowledge the algorithm execution or not.

The main disadvantage of algorithm coordination is the lacking ability to take coordination decisions based on the details of requested configuration changes. As soon as an algorithm execution has been acknowledged the action will also be executed. Therefore, algorithm coordination has no possibility to protect the system against contradicting or even oscillating configurations.

7.2.2.3 Combined Coordination Scheme

Depending on the potential conflicts that have been identified for a given SON-Function it might not be sufficient to use either algorithm coordination or action coordination. None of the introduced coordination scheme provides the required functionality to handle the case, if the identification of a trigger situation is based on performance measurement values that are influenced by another SON-Function instance, and, in addition, the operator has defined maximal difference with respect to previously performed configuration changes.

If the SON Coordinator has the ability to use a combined coordination scheme which performs both, algorithm and action coordination, the benefits of both coordination schemes can be utilized at the price of higher overall coordination complexity. It is then possible to guarantee that SON-Function instances are not executed before reliable input values to both monitoring and Algorithm-part are available, and also to protect the network against contradicting reconfigurations.

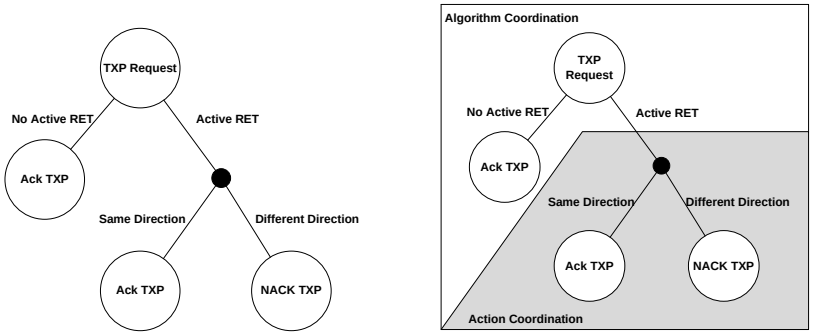
If a combined coordination scheme is applied to a SON-Function most of the coordination is performed by the algorithm coordination part, which protects monitoring and algorithm execution against erroneous input values.

Action coordination is only performed for a small number of SON-Functions and a limited set of conflicts. For example, if SON-Function instances with the same goals are executed as shown in the CCO example in Section 7.2.1.1.1, or for intra-function coordination when multiple instances of the same SON-Function are subsequently executed. Action coordination plays also an important role if extended operational requirements have to be fulfilled for the requested configuration changes.

7.2.2.3.1 Coordination Logic for Combined Coordination Schemes

In a combined coordination scheme the coordination logic for the algorithm coordination part can be designed to allow parallel execution of potentially conflicting SON-Function instances, while there is no input conflict between those functions. If there is only an output parameter conflict, and the SON Coordinator required knowledge about the algorithm results, their Algorithm-parts can be executed in parallel. Algorithm coordination acknowledges the algorithm execution for both functions and the conflicts are then handled by the action coordination.

The usage of a combined coordination scheme has also effects on the used coordination logic. Figure7.8(a) shows the decision tree from the CCO(TXP) example. For the usage with a combined coordination scheme, dedicated decision logic for each of the coordination parts has to be provided. The coordination logic for each SON-Function can still be represented within a single decision tree, but the decision tree is logically split into two parts (one part for the algorithm and one for action coordination, respectively, as



(a) Decision Tree for Algorithm or Action Coordination

(b) Decision Tree for Combined Coordination Scheme

Fig. 7.8: Decision Trees reflecting the Coordination Logic of the CCO(TXP) SON-Function

shown in Figure 7.8(b)). The upper part of the decision tree is processed in the moment the algorithm coordination request is received. The second part requires information that is only available during action coordination, therefore it is not processed until an action coordination request which contains the required information is received by the SON Coordinator.

7.2.2.4 Selection of appropriate Coordination Schemes

Depending on the potential conflicts with other SON-Functions and the operational requirements, a particular coordination scheme is selected at design-time for each SON-Function.

As an example for a small set of SON-Functions, the selected coordination scheme is presented along with a reasoning why this particular coordination scheme has been chosen. The operational guidelines such as, for example the priorities of the SON-Functions, have been taken from the experimental setup. Although those guidelines are aligned to those in real-world deployments they still should be considered as examples.

- **CCO Functions:** There are two SON-Functions that are used for Coverage and Capacity Optimization. One makes use of remote electrical antenna tilt changes (CCO(RET)) while the other reconfigures the transmission power at the antenna of the targeted cell (CCO(TXP)). For the selection of the coordination schemes two conflict types are relevant. At first, compared to other SON-Functions, the CCO functions are assigned a rather low priority, and secondly the changes have to be aligned with potential previous changes of a CCO function of different type. In case of a requested tilt change this is a power change and vice versa. From these conflict types it is easy to see that the combined coordination scheme is required. Algorithm coordination is needed to

prevent potential conflicts with higher priority SON-Functions. Action coordination is used to guarantee compliance with operational goals for subsequent execution of CCO functions.

- **Physical Cell ID Assignment:** The PCI SON-Function is an important example for the selection of coordination schemes, because although it requires coordination towards other functions as well as intra function coordination, no combined coordination scheme is required. The intra function conflicts are Category A conflicts and can therefore also be prevented by using algorithm coordination.

- **MRO Function:** For Mobility Robustness Optimization the SON Coordinator has to assure that no other conflicting function operates within the same target area. MRO is one of the functions with the highest priority, thus algorithm coordination is chosen. Before the algorithm execution is acknowledged all conflicting lower priority function instances need to be pre-empted, and during the SON-Function instance execution new lower priority function instances need to be blocked. These requirements impose the usage of algorithm coordination.

These examples show how, based on the conflict analysis and the operational guidelines, an appropriate coordination scheme can be chosen for a given SON-Function. If the introduction of new SON-Functions introduces additional potential conflicts to existing functions, it is possible to change the selected coordination scheme. Usually such a change is done from a single phase towards the combined coordination scheme. The change from the combined to a single-phase coordination scheme is only possible if SON-Functions or operational requirements are removed from the system in a way that detailed information about performed or requested changes are no longer needed for a coordination decision.

7.2.3 Context Information

Context information comprises all run-time information that is relevant for the coordination process. The SON Coordinator needs to have access to the current state of the network, information about which SON-Function instances need to be considered, changes performed by the SON-Function instances if relevant, and their Impact-area. Additionally also information whether special operational guidelines have to be considered, for example, due to the location of the targeted NEs, date or time, or other factors that can influence the coordination decision are part of the context information.

Information about the network, like topology information, types or locations of targeted NEs can easily be retrieved from the respective CM

databases. For special requirements like a date or event specific coordination logic, the operator needs to supply this information to the SON Coordinator.

Information on executed SON-Function instances and performed management tasks are directly available due to the conceptual placement (cf. Figure 7.5) of the SON Coordinator as a coordination. All relevant information is directly provided through the coordination requests, therefore no additional data sources are required.

Some data, as, for example, the concrete Impact-area of a SON-Function instance is neither directly provided nor directly available from any database. This information is derived per coordination request through a combination of the provided SON-Function instance specific data and the SON-Function specific abstract data, created at design-time, like the abstract Impact-area description.

In a large network with thousands of NEs and a multitude of deployed SON-Functions the analysis of the context data can become a limiting factor for the coordination speed. To guarantee a highly efficient coordination process the proper management of the context data located directly at the SON Coordinator is an important task. For this reason, the SON Coordinator should maintain all SON-Function instance specific information, especially the concrete Impact-area. If all information that has been provided or generated for a specific SON-Function instance is kept, subsequent coordination processes benefit as the information is directly available and no additional, time-consuming computations are required. Hence, context information about executed SON-Function instances is only relevant and required as long as their Impact-time has not expired. After that point in time it should therefore be discarded to avoid an unnecessary slowdown of the coordination request processing. Therefore, the SON Coordinator has to determine the longest possible Impact-time for each SON-Function instance, which is dependent on the relevance-time requirements of potentially conflicting SON-Functions (cf. Section 6.2.2).

7.3 SON-Function Instance Coordination Process

This section shows the SON-Function instance execution coordination process. After the description of the generic coordination process an example based on a concrete SON-Function is given.

7.3.1 Generic Coordination Process

The previous sections provided an overview on all components that are required for a successful coordination. All theses individual parts play an important role for the successful coordination of a SON-Function instance. The overall process that is performed for each received coordination request is shown in Figure 7.9. The process always starts with a coordination request

event that is sent from the SON-Function instance to the SON Coordinator. Depending on the assigned coordination scheme it can either be an algorithm execution request sent by the Monitoring-part of a SON-Function instance to request algorithm coordination, or an action execution coordination request from the Algorithm-part to trigger the action execution.

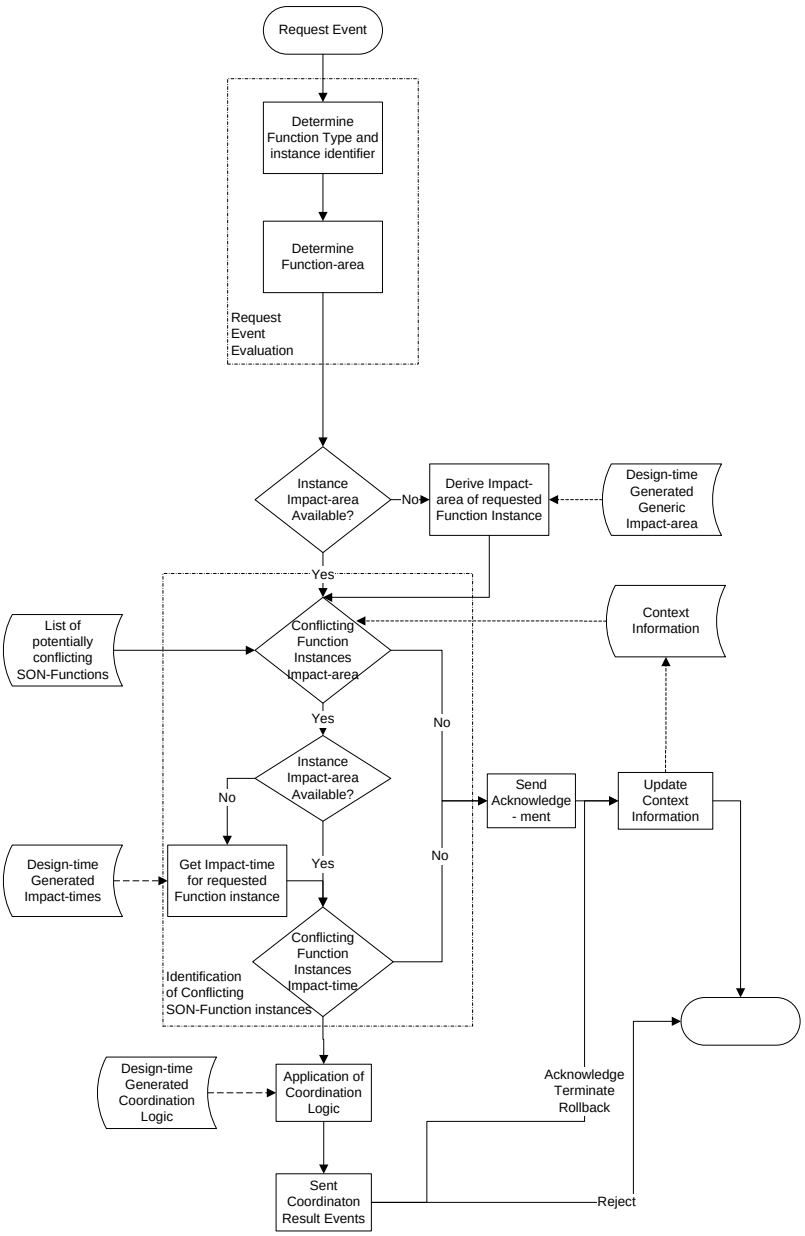


Fig. 7.9: SON-Function instance Coordination Process

This request and the contained information is used to determine SON-Function instance specific information. Apart from the unique SON-Function instance identifier two important informations are taken from the request. At first, the SON-Function type is determined, for example MRO, ANR etc. Additionally the coordination request contains information about the Function-area of the SON-Function instance.

In a next step the SON Coordinator checks whether the Impact-area information for this particular SON-Function instance is already available, which is the case if it is the action request for a SON-Function instance that is coordinated with the combined coordination scheme. If this is not the case the Impact-area is derived through the combination of the Function-area and the generic Impact-area definition.

The subsequent steps are performed to determine whether there are conflicting SON-Function instances. Basically this is a two-step process. At first potentially conflicting SON-Function instances with overlapping Impact-area are identified. Then, if required the Impact-time of the requested SON-Function instance is computed and with this information an evaluation is performed whether the identified potentially conflicting SON-Function instances are actually conflicting. If there are no potentially conflicting SON-Function instances with an overlapping Impact-area or if their Impact-time has already expired the request can immediately be acknowledged without any further processing. The acknowledgement is sent and the context information is updated accordingly.

The actual coordination is the last step of the process. In case conflicting SON-Function instances have been identified, the SON Coordinator applies the coordination logic. The result of the coordination is sent and, if required the context information is updated.

The same process is executed for each incoming coordination request event. Independent if it is an algorithm or action coordination request. Using a single coordination process that implicitly differentiates between the different coordination schemes through the availability of context information and the application of an appropriate coordination logic contributes to the high efficiency of the SON Coordinator. The functionality of the SON Coordinator is restricted to a small number of core functionalities, which do not have to be adapted to the applied SON-Functions. The SON-Function specific information is dynamically retrieved at run-time and can easily be adapted, if required, without interrupting the operation of the SON Coordinator.

In addition the SON-Functions are fully decoupled. There is no direct interaction between SON-Function instances required which facilitates the SON-Function deployment and maintenance. If SON-Functions are added to or removed from the system, their specific information such as generic Impact-area or Impact-time, potentially conflicting SON-Functions provided or removed and the coordination logic has to be updated.

7.3.2 Coordination Example

For this example the CCO(TXP) SON-Function is used. As shown in Chapter 7.2.2 the CCO(TXP) is assigned the combined coordination scheme, as the coordination requires knowledge about the performed changes.

Therefore the first coordination request event that is received by the SON Coordinator is the algorithm execution request event, which contains, apart from SON-Function and SON-Function instance identifier, the information about the Function-area which consists only of the targeted cell. As it is the first request for this particular SON-Function instance there is no information about the Impact-area available. The SON Coordinator needs to retrieve the generic Impact-area specification and derive the concrete Impact-area as proposed in Chapter 6.1. In combination with the list of potentially conflicting SON-Function types and the context information about active SON-Function instances, potentially conflicting SON-Function instance with an intersecting Impact-area are identified. This example considers an existing conflicting CCO(RET) SON-Function instance. The Impact-time between the two SON-Function instances has not yet expired and the SON Coordinator has to apply the coordination logic (cf. Figure 7.8(b)). Since the existence of a CCO(RET) SON-Function instance is not a blocking criteria, the SON Coordinator will acknowledge the requested algorithm execution. The SON-Function instance identifier and the respective Impact-area and Impact-time are added to the context information.

After the execution of the Algorithm-part, the SON Coordinator will receive an action execution request event. The first steps of the coordination process are performed very quickly, as the Impact-time and Impact-area are directly available from the context information. The conflicting CCO(RET) SON-Function instance is still active, therefore the coordination logic for the action coordination has to be applied. The SON Coordinator evaluates the requested changes, in order to determine whether the power change should be performed in the same direction as the previously performed tilt change. In this example, an uptilting has been performed and a power down is requested. Thus, the SON Coordinator does not permit the requested action execution. The SON-Function instance is terminated and all instance specific context information is deleted.

7.4 Coordination Concept Summary and Findings

This chapter introduced the main concept for the coordination of SON-Function instance execution. The initial overview over different conflict prevention and resolution approaches showed the necessity for an efficient and robust run-time approach which allows active and pro-active protection of SON-Function instances as well as the enforcement of network management

guidelines. The presented concept introduces the SON Coordinator which acts as an intermediate coordination layer between SON-Functions and the affected NEs. It combines policy based decision taking which operates on a pre-defined coordination logic and interacts with the SON-Functions according to assigned coordination schemes. The coordination scheme defines between which parts of the SON-Function the SON-Function instance has to interact with the SON Coordinator. The presented coordination approach has minimal run-time information requirements. Most of the run-time information can easily be derived from a combination of SON-Function instance specific information and information that has been defined at design-time. For example, to identify the Impact-area and Impact-time of a particular SON-Function instance the coordination request contains only the a unique SON-Function identifier and the Function-area of the SON-Function instance. The SON-Function identifier defines on the one hand the SON-Function type (e.g. ANR) and on the other hand the SON-Function instance itself. The identifier allow the SON Coordinator to recognize SON-Function instances when subsequent coordination requests are sent. The minimization, and simple provisioning of required information contributes to the efficiency of the approach.

Through the coordination logic, it is possible to enforce additional operational guidelines on the execution of the SON-Function instances which contributes to the robustness and reliability of the SON enabled network (Research Questions R2 and R3).

In order to speedup the coordination process, the steps that have to be performed for the coordination are put into a particular order that allows to skip steps if they are not required. For example, if the SON Coordinator detects that the Impact-area of the requested SON-Function instance does not overlap with the Impact-areas of other SON-Functions it will directly acknowledge the request without any further processing. This increases the efficiency and speed of the coordination, which is especially important in a network with potentially thousands of concurrently active SON-Function instances.

The presented conceptual design scheme for SON-Functions contributes to several research questions that were introduced in Chapter 1.3:

Concerning the research area Reliability and Robustness, a contribution to the following research question has been made:

R1 How can the danger of negative effects through the execution of conflicting SON-Function instances be minimized?

The danger of negative effects of the conflicting SON-Function instance execution can be reduced if the conflicts and thus the effects

do not occur, that means if the concurrent execution of conflicting SON-Function instances is avoided.

Since all interactions with the network have to be acknowledged by the SON Coordinator, conflicting SON-Function instance execution can be prevented or resolved by the SON Coordinator. Run-time conflict detection and resolution can always be performed if the conflicts have been revealed and categorized during design-time and the appropriate coordination and decision logic has been specified. This allows the SON Coordinator to intervene and block particular SON-Function instances to prevent conflicting behavior.

R2 How can the network operator, despite the automation, still be in full control over the ongoing processes in the network?

The presented SON-Function instance execution coordination approach resides in the network as an intermediate coordination layer, therefore no SON-Function instance is able to perform configuration changes in the network without the permission of the SON Coordinator. If operational guidelines can be specified as part of the coordination and decision logic, they will automatically be enforced by the SON Coordinator. Thus, the operator has full implicit control on the network behavior. It is important that the operator can monitor the decisions of the SON Coordinator and overrule any decision that has automatically been taken. This can, for example, be guaranteed by giving manual interaction the highest priority.

R3 How can the enforcement of and the compliance with all operational guidelines be assured?

The compliance with the operational guidelines can be assured if they can be specified as part of the coordination logic. Then, the SON Coordinator automatically takes care of their enforcement, since it has to acknowledge all configuration requests. This requires operational guidelines to be specified either in relation to fixed values (e.g. allowed parameter setting interval) or to properties which can be easily assessed by the SON Coordinator. This can be easily covered for the existing SON use cases, either by expressing operational guidelines in terms of priorities (e.g. failure recovery before optimization), in relation to the time (e.g. no energy saving between 6 am and 10 pm) or as fixed or dynamic thresholds (e.g. allowed parameter setting between -10 and 10 but change not more than 10% of the previous value).

R4 Which parts of the SON-Functions are affected by conflicts?

Based on the conflict categorization it is possible to derive which parts of the SON-Function are affected by potential conflicts. The

presented coordination schemes allow the SON Coordinator to particularly address these conflicts and protect the SON-Function instance execution. If the correct coordination scheme is assigned to a SON-Function, that has been developed according to the tripartite design scheme, the SON Coordinator will be aware thereof, and is therefore capable of preventing or resolving of conflicting behavior.

Concerning the research area Efficiency, a contribution to the following research question has been made:

E4 Which is the most efficient way to take the required decisions?

The execution of SON-Function instances is highly event driven, therefore a decision making approach that supports this mode of operation is required. The presented decision tree based approach directly reflects the event based operation, thus is directly usable in a SON enabled network. A SON-Function instance execution request can be directly mapped to a particular decision tree, and the coordination decision can be made by traversing the tree.

The decision making should always be aligned to the prevailing mode of operation in a system. Due to the properties of SON and SON-Function execution, an approach, like the presented decision tree based decision making, is potentially most efficient.

E5 How can the required context information be provided efficiently to the system controlling the SON-Function instance execution?

Two separate approaches have been presented for the efficient provisioning of information to the coordination process. First, the combination of run-time information that is derived from abstract information that has been defined at design-time (cf. the instantiation of the Impact-area in Chapter 6.1) with a minimal amount of information that is easily accessible at run-time.

Second, a coordination process that minimizes the required amount of information. Detailed information is only assessed and stored when needed. As soon as the information is no longer required it is deleted, which increases the efficiency to manage and operate on the run-time information. The minimization of used information is always beneficial to the efficiency of the data processing.

The combination of both approaches leads to a very efficient provisioning of context information, based on the requirement that there are methods to easily compute required information (e.g. Impact-area), and the remaining information can easily be retrieved.

For the coordination of SON-Function instance execution it has been shown that complex information like Impact-area and Impact-time

can be derived from abstract specifications created at design-time and only. The remaining information requirements like date, time, and NE type can easily provided by fast information retrieval methods, for example, data base lookups.

Concerning the research area Flexibility, contributions to the following research questions have been made:

F2 How can the decision logic be provided in an easily adaptable way?

Decision making based on decision trees makes the SON Coordinator very efficient and allows a flexible adaptation and extension of the coordination logic. New SON-Functions and their coordination logic can be introduced at run-time with only minimal effects on other SON-Functions. Based on solution agnostic decision trees it is easily possible to derive appropriate solution specific decision making methods. Decision trees can, for example, be directly mapped into a set of policies as a run-time representation of the decision logic.

This approach can always be used if sequential decision making is an option. For the usage in network management, and especially for the coordination of SON-Function instance execution, it has been shown that this kind of decision making is closely aligned to the general mode of operation and therefore is preferable. If the decision trees are well built, they will even contribute to the speed of the decision process and the reduction of required information. To reach this goal, the decisions that are more likely have to be located in the upper part of the decision tree.

8. SON-FUNCTION DESIGN PROCESS

For the development of a SON-Function, based on a specified SON use case, this thesis proposes a structured development process that provides all components required for a successful coordination of instances of the resulting SON-Functions. The sequence of the development steps guarantees the availability of the information required for each step.

- The **Analysis** of the specification of the SON use case is an important first step. SON use case descriptions [Leh07b] often provide important information about the tasks that should be performed, the prerequisites and the expected results. Thus a proper analysis provides the understanding that is required to develop the SON-Function.

- The **development** of the concrete SON-Function is based on the information that was gained through the use case analysis and the network specifics. The development process turns the abstract use case description into an executable SON-Function. At the end of the development process all parts of the SON-Function are available, which includes the specification of the triggering situation, the algorithm that is executed and the actions that have to be performed to reach the goals. As a part of the development process a co-design based on information of already existing SON-Functions can be performed. This initial co-design requires no detailed conflict analysis but relies only on the available information of parameters and input values used by other SON-Functions.

- In order to support the run-time coordination of SON-Function instances, the **separation** of the SON-Function and the assignment of functionality to monitoring, algorithm and Action-part is required.

- After the assignment of functionality to the three parts of the SON-Function (monitoring, algorithm, action cf. 4.1) the **Conflict Analysis** can be performed. Based on the results the implementation of the SON-Function can be adapted to remove potential conflicts with other SON-Functions. For the co-design based on the results of the conflict analysis the development step is repeated. Potentially the assignment of functionality to SON-Function part can be changed.

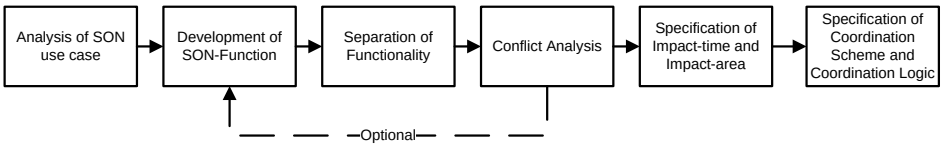


Fig. 8.1: SON-Function Development Process

- Definition of the SON-Function specific **Impact-area and Impact-time** is the step that can be performed as soon as the requirements of the newly developed SON-Function itself and the requirements of the potentially conflicting SON-Functions are known.
- The specification of the **Coordination Logic** and assignment of an appropriate **Coordination Scheme** is the final step of the SON-Function development process. The coordination logic unifies the knowledge about the specific SON-Function like the tasks that are performed, its Impact-area and Impact-time requirements, the potentially conflicting SON-Functions, as well as the network specific operational guidelines and requirements.

8.1 Design-Process Summary and Findings

The described development process is shown in Figure 8.1. It is created to speed up the SON-Function development and the provisioning of all parts that are required for the coordination. To increase the system efficiency and reduce potential conflicts between deployed SON-Functions it is possible to refine the developed SON-Function after the conflict analysis. It is also possible to adapt the potentially conflicting SON-Functions, but it is not required to change deployed SON-Functions. If there are potential conflicts, they can be handled through an appropriate coordination of the SON-Function instances, which requires only an adaptation of their coordination logic.

Part III

CONCEPT IMPLEMENTATION, VERIFICATION AND EVALUATION

9. SON COORDINATOR IMPLEMENTATION

For both, the verification and also for the evaluation of the SON-Function coordination concept a dedicated Demonstrator- and Experimental System was implemented. On the one hand, the goal was to have an independent experimental system as a framework that allows to implement the conceptual key elements for further development and testing. On the other hand, it was designed to be able to connect the experimental system to other systems, for example network simulators or performance measurement sources from a real network, in order to allow an examination under more realistic conditions and also the comparison to other coordination approaches.

For this dual purpose it was important to design the system in a way that allows the implementation of all required algorithms and also the implementation of the behavior of the elements that are not present within the experimental system. Another important characteristic is the possibility to switch off most of the emulated functionalities and delegate it to other systems. With such an architecture it is possible to gradually evolve from a pure proof of concept implementation to a Demonstrator System that is closely integrated to a productive environment.

In the initial development phase most of the required functionality can be simulated and a very high abstraction of a realistic network is used, which represents only those characteristics that are required for ongoing evaluation and presentation tasks. For example, instead of a realistic multi-sector cell layout which is based on radio propagation theory and underlying physical models, an abstract approximation of a network is used. In this approximation, the cells are represented by circles whose center positions were chosen to reflect a typical Inter Site Distance (ISD). The cell radius is chosen according to typical cell sizes in the respective deployment scenarios, such as, for example, urban or rural deployments. The information available on the NEs would also be restricted to an absolute minimum.
In an initial implementation the information that was available for the cell was restricted to:

- Unique identifier of the cell
- Cell center position
- Cell Radius
- Physical Cell ID

Additional details were added later, to be able to test additional use cases.

- Operational state (Active or Inactive)
- Cell type (Macro, Micro, ...)
- Neigboring cells

In the same way an abstract and generalized representation of the network is used, other functionality is also only provided as far as required. For example, simplified SON-Functions with limited functionality that show only the required behavior are used. Such functions send only coordination requests for target cells but do not perform any actual configuration changes.

For the general evaluation of the implemented concepts such an abstract representation of the network and the functions within the network is fully sufficient. But for a more thorough evaluation under more realistic conditions more advanced and realistic systems are needed. This is the reason why the possibility to exchange parts of the system with other systems was a top requirement. For the evaluation of the coordination concept, a full-fledged LTE network simulator was attached that served as a data source and was also capable of running the SON-Function algorithms. Therefore, the only functionality that was provided through the experimental system is the coordination of the SON-Function instance execution requests.

This chapter is structured as follows: First an introduction to the overall architecture of the Demonstrator- and Experimental System is given, which is complemented by information about the different components that are used. In Section 9.3 the basic operational concepts of the system are explained. The last section presents the results of the evaluation of the coordination concept that has been performed within the implemented Demonstrator- and Experimental System.

9.1 Demonstrator System Architecture

The architecture of the Demonstrator- and Experimental system is rooted in two basic assumptions which were present from the very beginning. First, the overall system would be under constant further development in order to satisfy the developing requirements and to allow the implementation and evaluation of new concepts and additional required functionalities. Second, at some point, some of the functionality that has initially been implemented to facilitate and speed-up the overall development process will be replaced by other simulation systems or even real commercial products. Therefore a very modular architecture that is connected by a generic message bus has been chosen which is shown in Figure 9.1.

The generic message bus is the glue between the modules that forms the basis of the experimental system. A detailed description on how it has been realized is given in Section 9.2.1. Its most important characteristic is that it imposes only minimal requirements and restrictions on how the modules connect to the message bus and on how the modules communicate.

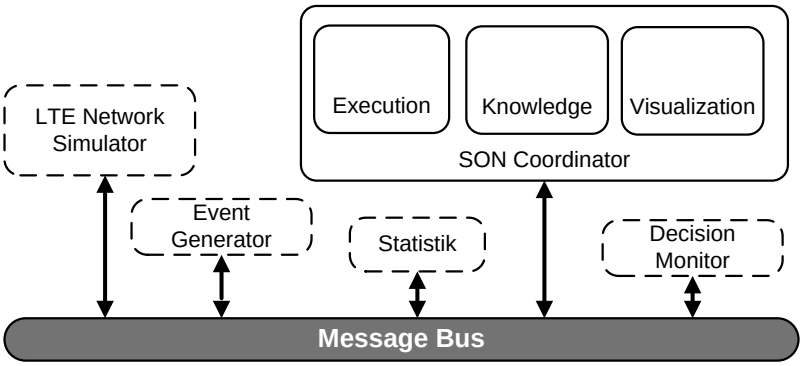


Fig. 9.1: Architecture of Demonstrator- and Experimental System

It is designed to allow a simple integration of new modules with minimal overhead.

All modules have a very dedicated but limited task range. Although this design decision increases the communication load between the modules it is mandatory to satisfy the requirement that a module which provides a particular functionality can easily be replaced.

The overview in Figure 9.1 shows the most important modules. The core SON Coordinator consists of three modules, Execution, Knowledge and Visualization, which are mandatory. Each of the modules is individually connected to the message bus, although the Figure shows only a single connection. The Visualization Module allows to interact with the system and visualize ongoing SON-Function operation. SON-Function algorithms and the coordination logic are located in the Execution Module. All required information about the network is provided by the Knowledge Module. Apart from these three core modules which provide the main functionality there are additional optional modules that are used for concept verification and evaluation, like the Statistic Module, Event Generator and Workflow and Decision Monitor. Optional modules are marked by dashed lines in the figure. Detailed descriptions of the module functionalities are given in Section 9.2

9.2 Components and Modules

Several central components and modules form the core for the Demonstrator- and Experimental System. They provide the basic functionality that allows to easily implement and evaluate different use cases and scenarios. A detailed introduction to these modules is given in the following sections.

9.2.1 Message Bus

As stated above the Message Bus is the central part of the Experimental- and Demonstrator System. The main requirement was a high degree of flexibility that it needs to offer. It should impose only minimal restrictions on the attached modules in the way how they are connected to the message bus and how they communicate between each other. It should be possible to add and remove modules dynamically without the need to perform major changes to the other modules.

From these requirements it became obvious that a publish and subscribe based system should be used as message bus. Especially as the publish and subscribe mode of operation directly satisfies multiple requirements. New modules can be introduced by connecting them to the message bus. They can immediately take part in the communication by subscribing to relevant topics and publishing information under certain topics. That way it is also possible to introduce additional functionality, like extensive analysis and monitoring capabilities, simply by connecting a respective module to the message bus which subscribes to the relevant topics. No changes at the sending and receiving modules that are evaluated are required.

Using a central common message bus opens up the possibility to either execute all modules on a single machine or distribute them to different machines as long all modules can connect to a common message bus. This is in particular an interesting feature as it allows to run important modules like the Knowledge or Execution Module on a central server to keep the information within the system consistent and at the same time allow different users to perform different tasks concurrently. It also allows to model a complex distributed network setup, where functionality is provided at different locations. In an experimental setup additional constraints as, for example, limited available data rates can be realized by imposing such constraints on network links between the machines.

Exchanging modules that provide a certain functionality is equally simple. The original module is removed and the new module is inserted. In case the new module does not provide the expected message format, additional message wrappers can be introduced into the system. The data source A1 is exchanged through the data source A2. But the message format that is provided by A2 is not compatible to the one formerly used between A1 and the data source B2. Therefore the messages are routed through the message format converter C1. This rerouting of messages is done by using the basic functionality of the publish subscribe system. A1 and B1 were communicating under the topic T1, which means A1 was publishing its messages under the topic T1 and B1 was subscribed to the same topic and therefore was provided with the messages. Instead of publishing the messages directly under T1, which would result in a communication error, A2 publishes its messages under the topic T2. C1 is subscribed to T2 and converts the re-

ceived messages into the format understandable by B1. After the conversion the messages are published under T1 which B1 is subscribed to.

What makes this approach very appealing is that absolutely no changes are required, not only at B1 but also at A2. In a complex system where the same message format is used to communicate between a multitude of modules a change of the message format could introduce a large overhead as many modules have to be adapted to the new message format. This is not the case when using the publish subscribe based communication approach, which allows to route message through intermediate format converters.

For the implementation of the message bus in the Demonstrator and Experimental System an open source system was used. xmlBlaster [xml08] is a very versatile platform independent publish subscribe system. It offers a wide array of APIs through which messages can be sent and received, and therefore does not restrict the attached modules in any way. By using XML based messages the full power of the XML filtering and querying capabilities can be used to subscribe to certain message types and topics, especially if a structured and consistent naming and addressing scheme for the message topics is used. In Section 9.2.1.1 the hierarchical naming scheme used within the message format is introduced that, in combination with the XML filtering capabilities, allows to subscribe to either particular messages or groups of messages.

To be able to include modules in the way as described in the example for the data source A2, modules have to be capable to provide the data in an XML based format and to use one of the APIs provided by xmlBlaster. If this is not the case, external message wrappers or format converters can be used. These can on the one side directly interact with the external module and on the other side with the xmlBlaster.

xmlBlaster imposes only very little restrictions on the used message format apart from the message being represented in an XML structure, and also this requirement is limited to a particular outer XML element. This allows to also transmit data which is not available in XML, as long as the sender and the recipients of the messages have a common understanding of the message format.

9.2.1.1 Message Format

The implementation uses JAXB [MS06], a Java based library to create the XML messages. JAXB can be used to serialize Java objects into XML code and de-serialize received XML into Java objects. Therefore no specific XML generators or parsers are required. In order to have a common message format for all attached modules, a basic Java Class as a representation for the messages was provided. All specific messages are created from objects that are assigned classes which are derived from this basic class. Thereby messages based on a common message format are created when the objects are

serialized with JAXB. The basic message format provides some important features that facilitate the handling of the messages, such as a message type (request, response, informational) or a structured message ID.

9.2.2 Knowledge Module

All information about the networks that serve as basis for the evaluation and the showcases is stored within the Knowledge Module. Other modules can retrieve, store and update information by publishing appropriate messages on the message bus. Communication with the module is performed through dedicated request and response events. Additionally, the Knowledge Module exploits the event based mode of operation, that is, it evaluates configuration update event messages even when they are not directly addressed to the Knowledge Module. This makes the development and introduction of new functionality much easier, since no explicit interaction with the Knowledge Module is required to maintain the information, as long as specific events are used to communicate configuration updates. This functionality is extensively used whenever a data source, as for example, the LTE network simulator, is used and, in the beginning no information about the underlying network is available. Whenever the coordination function acknowledges a configuration update request, these event messages are evaluated and the information about the targeted NE and its configuration is stored within the Knowledge Module. This behavior allows the Knowledge Module to learn information, which was not explicitly provided at the startup of the system, at run-time.

The Knowledge Module allows to make information about a network consistent by storing it locally and loading it whenever required, for example, evaluating the effects of different versions of the coordination logic starting from an identical network scenario. Not only different network scenarios can be prepared and loaded on demand, but it is also possible to store the information about the modified network at the end of an experiment.

In order to be able to work with realistic network layouts, the Knowledge Module also allows to import network planning files from different sources. Data can be imported from commercially used network planning tools or freely available data sources such as [Sch09]. The import process, ignores all information that is not required and imports only relevant data.

9.2.3 Visualization

The Visualization Module with its GUI (cf. Figure 9.2.3) serves as the primary interface to the Demonstrator and Experimental System. Networks can be loaded, stored or created from scratch. It allows to visualize the network setup and provides different views for different showcases. For example it allows to present the network for the PCI use case [3GP10b] in a special

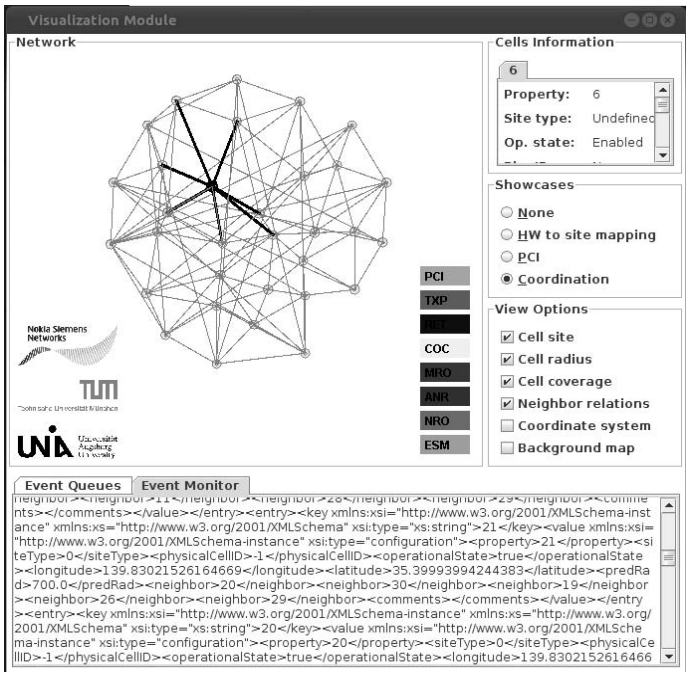


Fig. 9.2: Visualization GUI in Coordination View

color coding, which represents the PCIs configured to the cells in different colors. To understand the coordination processes and the SON-Functions that are currently executed within the network the coordination view is provided that encodes the active SON-Functions and the operational state they are currently in, through different colors and line styles, in addition to the representation of the neighborships which are important to understand the system behavior. Additional pop-up messages can be activated to inform the user about the decisions that were taken

The user has the possibility to retrieve information about the network in general and the cells in particular through the GUI, by selecting the elements he is interested in on the map. The configuration of the network can be directly manipulated through the GUI, cells can be added, or their configuration can be changed. Such possibilities are in particular important to evaluate the behavior of SON-Functions which react on certain configuration changes. Am example is the PCI function which needs to evaluate the PCI assignment whenever there is a possibility for new or changed neighborships.

Through the context menu of the GUI it is possible to trigger showcases that represent key SON use cases, to show and evaluate changed behavior that could come from an adapted coordination logic, SON-Functions, or activated or deactivated policies that control the system behavior.

To visualize ongoing processes in the network the Visualization Mod-

ule makes extensive use of the event based publish-subscribe message bus. At startup it will automatically subscribe to a range of topics to become aware of acknowledged and rejected SON-Function instance executions or configuration changes within the network. This information will then be automatically represented within the GUI, so there is mostly no need for a direct communication with the visualization module. This functionality is realized without a single directly addressed message from the coordination function to the Visualization Module. For example, when the coordination view is activated, the Visualization Module subscribes to all coordination requests and coordination decision events. Those events are issued by the SON-Functions and the SON Coordinator. To be able to represent the configuration changes, a subscription on the configuration-change-events is also made. The Action-parts of the SON-Function instances issue these events to reconfigure the network elements. Through the evaluation of these events it is possible to update the representation of the network in the GUI, and hence the information that can be shown to the user.

9.2.4 Execution Module

The main module of the SON Coordinator is the Execution Module. Within this module most of the coordination logic is contained and executed. Depending on the mode of operation different functionalities can be activated. Basically it has the ability to execute SON-Function instances and perform the coordination. If there is an external entity for the execution of SON-Function instances the internal SON-Function instance execution is disabled and the functionality is restricted to the coordination.

From the Execution Module GUI (cf. Figure 9.3) SON-Functions can be activated or deactivated. If the SON-Function execution is performed within the Execution Module, it is possible to change the configuration for the SON-Functions. For example, for the Hardware-to-Site Mapping SON-Function [BS11] it is possible to determine which mapping strategies should be available, and in which sequence they should be enforced. All changes that are performed in the GUI become instantaneously active and influence the behavior of the system.

The most important functionality is the coordination of the SON-Function instance execution. If the coordination is enabled the coordination policies are used to enforce the cordination logic and take coordination decisions. For a fine grain control over the coordination logic, the user has the ability to activate and deactivate individual policies.

It is also possible to determine the Impact-time. The Execution module can be configured to use a global Impact-time which is identical for all SON-Functions, an individual fixed Impact-time for each SON-Function type or the pair-wise Impact-time as specified in Chapter 6.2. The respective Impact-times are specified in a configuration file and can also be changed at

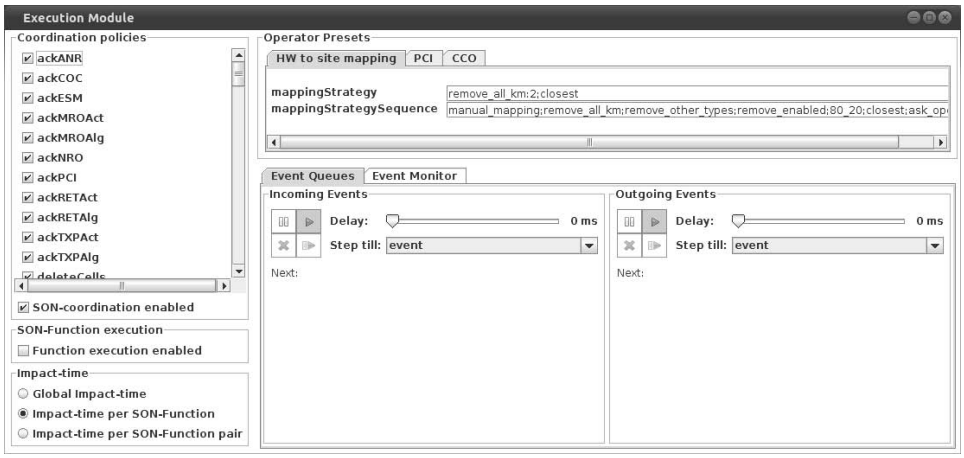


Fig. 9.3: GUI of the Execution Module

run-time.

9.2.4.1 Request Coordination and Coordination Logic

The coordination of the SON-Function instance execution follows the overall event based mode of operation. All requests are communicated through respective events to the Execution Module, which evaluates these request events through a set of policies that represent the coordination logic. Within the Execution Module an instance of Ponder2 [TLDS08] is used. Ponder2 is an Open Source policy engine, which stands out compared to other alternatives through its clean design and sound conceptual foundation. It emerged from a research project as the successor of the well known Ponder [DSESitDoC02] policy framework that served as a reference model for policy research.

The implementation of the SON Coordinator makes use of two important features of the Ponder2 framework to the coordination of SON-Function algorithm and action requests. On the one hand, management objects to maintain a consistent storage of context data. On the other hand the Event Condition Action (ECA) policies to represent the coordination logic. The policies forming the coordination logic are provided as a Ponder2 policy file, which is loaded during the startup of the module. Ponder2 as a policy engine follows the ECA scheme and can therefore be used without any further adaptations within the highly event based mode of operation of a network management system. The policies are an implementation specific representation of the solution agnostic decision trees that were generated as part of the SON-Function design process as described in Section 8. An example is shown in Figure 9.4 and the corresponding policy code in Listing 9.1. The decision tree in Figure 9.4 shows the part of the decision logic that leads

to an acknowledgement of the coordination request for the Algorithm-part of the CCO(TXP) SON-Function. The policy code represents exactly that part of the SON-Function decision logic.

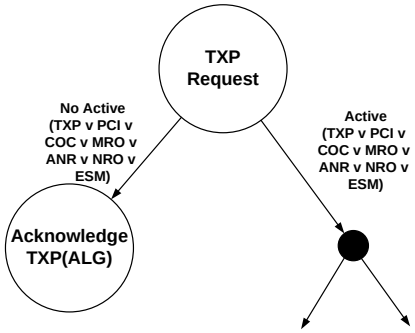


Fig. 9.4: Partial Decision Tree for a CCO(TXP) Algorithm Coordination Request

Listing 9.1: Example Policy Code to acknowledge a TXP Algorithm execution request

```
policy event: root/event/workflowExec/ExecuteTXPAlg;
       condition: [ :id :property :changeVal |
            root/config/coordination not |
           ((root/conditionChecker noActiveTXP:id
              property:property) &
            (root/conditionChecker noActivePCI:id
              property:property) &
            (root/conditionChecker noActiveCOC:id
              property:property) &
            (root/conditionChecker noActiveMRO:id
              property:property) &
            (root/conditionChecker noActiveANR:id
              property:property) &
            (root/conditionChecker noActiveNRO:id
              property:property) &
            (root/conditionChecker noActiveESM:id
              property:property))
       ];
       action: [ :id :property :changeVal |
            root/actionExecutor ackTXPAlg:id
            property:property changeVal:changeVal
       ];
```

The coordination of SON-Function instance execution requests is kept as simple as possible. As soon as they are received by the Execution Module,

the coordination request events are fed into the Ponder2 instance which acts as the coordination function. The policies react instantaneously on the incoming events. The actions of the policies represent directly the result of the coordination and are communicated back to the requesting functional entity, like SON-Function instances. In case of a rescheduling decision the events are buffered internally and re-inserted into the coordination process at the end of the rescheduling delay.

The capabilities of the policies in Ponder2 are rather simplistic, for example, the condition part of a policy allows only simple comparisons with a boolean result. Ponder2 therefore provides the possibility to use so called Managed Objects. Managed Objects are persistent Java objects that can be used to provide missing extended functionality. For example to perform analysis of data as part of the policy condition part, communicate with external data sources, or persistently store information. Managed Objects can be used within both, the conditions and the Action-part of a policy. The combination of simple policies with the full capabilities of a Java program forms a very powerful approach. An important side-effect is the maintained readability of the policy files.

There are several ways how Managed Objects are used within the coordination process, mainly to manage network context information. Since all SON-Function instance execution requests pass through the Execution Module it is a well suited place to keep track of all ongoing processes. Concrete information about the executed SON-Function instances are contained in the coordination requests and are therefore available within the Ponder2 instance to take coordination decisions.

Dedicated Managed Objects are used to store information on which SON-Function instance execution coordination requests have been acknowledged, their Impact-area and, if available, which configuration values have been enforced. All information that is required for further coordination decisions is stored. For the maintenance of the context information, time-stamps are used, as described in Section 7.2.3 that represent the end of the function instances Impact-time. At the end of the Impact-time the information is deleted within the Managed Object and is therefore no longer available for further coordination decisions. This assures that information can be taken into account for coordination decisions only as long as it is relevant.

Another application of Managed Objects is the rescheduling of request events. Instead of requiring additional communication capabilities and respective protocols to inform the requesting entity about a rescheduling decision the respective event is stored within a Managed Object. At the end of the rescheduling interval, the request event is re-inserted into the Ponder2 instance and treated equally to a newly received coordination request event. Using Managed Objects and their capability to perform tasks can be used to assure that requests are not rescheduled indefinitely, for example, by incrementing a rescheduling counter within the request event, which is evaluated

by the policies as part of the coordination logic. In case the rescheduling counter exceeds a preconfigured limit an appropriate coordination decision can be taken.

Ponder2 offers an interface for run-time interaction with the active policy engine instance. New policies can be introduced, existing policies can be adapted, enabled or disabled. This capability is used to allow a run-time adaptation of the coordination logic through the GUI of the Execution Module. Each change of the configuration in the GUI is directly reflected within the Ponder2 instance and has therefore instantaneous effects on the coordination process.

9.2.5 Statistics Module

An independent Statistics Module is provided for the evaluation of different coordination approaches with different coordination logic, or coordination logic that is differently parameterized. It makes use of the possibility to monitor ongoing processes by subscribing to their communication.

The efficiency of the coordination logic is evaluated through an analysis of the messages exchanged between the SON-Function instances and the SON Coordinator. As the Statistic Module operates independently from the other modules it does not affect the coordination in any way. It provides an overview on the number of coordination requests, and furthermore on how many of these requests have been acknowledged or rejected. For the acknowledge algorithm execution requests it will show how many of the action execution requests are acknowledge or rejected or in a subsequent step even rolled-back. The Statistics Module can additionally be used to provide real-time statistics about requests and coordination decisions structured by targets. A user can directly see how many requests of which type have been issued for a particular target and which coordination decisions have been taken.

Based on the detailed information contained in the request events it is possible to compute statistics on which SON-Functions are requested how often for which targets, and how the SON Coordinator responds to these requests. It also allows to trace the sequence of SON-Function requests for individual targets and determine whether there are unknown dependencies between SON-Functions or if there is an oscillating behavior that needs to be treated by changing the coordination logic.

The Statistics Module is an open framework for the generation of statistics, which can easily be extended to satisfy all information requirements. For a general evaluation of coordination logic it is preconfigured with 13 different metrics, among others:

- Number of Algorithm and Action requests
- Number of different types of coordination result events
- Proportion of Algorithm and Action requests

- Distribution of reschedule decisions on different SON-Function types
- Average number of rescheduling decisions per rescheduled request
- Average number of subsequent requests caused by an acknowledged SON-Function execution

Apart from the basic statistics, additional output is produced for the further processing of the data. For the preconfigured metrics, different charts can be generated. The user has the possibility to regenerate the charts at any point in time during the evaluation, which allows to compare the behavior of the system at different stages during the operation. The module also provides a raw event dump of all events that have been received, which allows a further processing at a later point in time, for example, to generate additional statistics which have not been available during the initial evaluation run.

9.2.6 Event Generator Module

Using a network simulator that includes a lot of randomness, for example, in the user movements, as source of input data, is beneficial if realistic scenarios are to be evaluated. For repeatable experiments though, or to thoroughly test a given setup or coordination logic, another input source with predictable behavior is required. The Event Generator Module provides the possibility to inject predefined event sequences into the Demonstrator and Experimental System. It is capable to issue any event that is used by any of the modules within the system.

Different methods regarding how the events can be sent to the system are available. Events are either individually sent by the user or according to a predefined schedule. It is also possible to switch between the different methods at run-time. Additionally, breakpoints can be set, for example, to send all events up to a certain event type according to the predefined timing and then switching for a few events to a stepwise, user triggered sending of events.

The graphical user interface of the Event Generator can be used to create and modify event sequences. All relevant parameters of an event can be directly edited. Additionally, the delay between two subsequent events is configurable. Event sequences can be stored to and loaded from XML based files, which, apart from reusing event sequences, allows us to process the event sequences with other tools.

The Event Generator Module is used even in addition to another active event source, for example a network simulator. An event stream from an arbitrary source can then be enhanced with additional events. Receiving entities are not able to differentiate the source of the incoming events. Combining multiple event sources allows us to use one event stream that reflects usual behavior and imposes a background load to the systems, and a second event stream that provokes critical situations within the system or just produces particular scenarios.

9.3 Operation of the Demonstrator and Experimental System

The behavior of SON-Function instances in particular and the SON enabled network in general is highly event driven. Self-healing, -optimization, and -configuration run in parallel, for example, cells fail and the failure has to be treated, at the same time suboptimal operation of other parts of the network is detected which mandates optimization, and in addition new network equipment is introduced and needs to be integrated into the network. To reflect this behavior the event based operation scheme was applied to the operation of the Demonstrator- and Experimental System and all of its components. Also the SON-Functions as described in Section 4 and their event based behavior benefits from this operational paradigm. Events and event flows characterize and structure SON-Function instance operation. Such an event flow can run, for example, from an initial event, when the monitoring part detects a triggering situation via the event that is used to request the execution of an algorithm to the configuration request events which are used to enforce the newly computed configuration parameter settings.

The purely event based communication imposes also a highly asynchronous, unidirectional communication scheme. A basic assumption behind the selection of the event based communication scheme is that most of the entities and processes within the system require only unidirectional communication. Events transmit information and trigger actions. Processes and workflows are modelled through sequences of tasks that are triggered through an event from the previous task. This requires that all required information is contained in the event.

Additional means, like specific message IDs, are used to provide the possibility for message correlation and tracing whenever required.

9.3.1 Traceability of Event Correlations

An event based communication system needs to provide capabilities that allow both, an efficient processing of the transmitted events, and to easily correlate outgoing and incoming events. That applies in particular if request-response communication should still be supported via the stateless publish-subscribe message bus.

For the operation of the Demonstrator- and Experimental System the possibility to visualize relationships between observed events and the corresponding SON-Function instances was needed. For example, to determine which additional SON-Function instances are triggered by a particular instance or to allow monitoring and evaluation of the behavior of the modules in the system.

The message format as described in Section 9.2.1.1 already allows to classify events into different groups based on their type. But in a large

network with potentially thousands of concurrently active SON-Function instances there will be a multitude of events of the same type at any given point in time. Therefore an additional method is required to assure a reliable event correlation.

The Demonstrator- and Experimental System makes use of so called Structured Workflow IDs for event correlation. It realizes the concept of a logical workflow. A logical workflow is a sequence of tasks that is performed due to an initial trigger. An example for such a trigger is a SON-Function instance algorithm coordination request event. This initial event causes a set of subsequent events, at least a single coordination decision event, but potentially also events that are used by the Execution Module to request additional information from the Knowledge Module or the subsequent SON-Function instance algorithm, or action execution request events.

All these events characterize the logical workflow and it is important to have the possibility to correlate them. A traceabilty of event messages is reached through the Workflow ID that uniquely identifies each event and binds it to the logical workflow. The workflow ID is constructed after the following scheme:

- Each entity type has an individual Type ID, which allows to identify a SON-Function like the PCI Functions in general.
- The initial part of the Workflow ID is the Instance ID of the first entity that contributes to the logical workflow. The Instance ID consists of the Type ID which is extended with an instance specific suffix. This allows to identify type and instance at the same time.
- The current Workflow ID is inserted into an outgoing event.
- Each entity that participates in this logical workflow adds its own Instance ID if it initiates further requests. If not, it sends a response event with the received Workflow ID copied unchanged into the response event.
- Entities can identify particular response events they are waiting for through the Workflow ID which has been inserted to the request it has sent.
- When sending a response, the entity just assures that the Workflow ID is restored to the one that has been provided through the request event.

Figure 9.5 shows a simple example for the handling of Workflow IDs. For the better understanding only simple IDs are used. The Type ID for the entities consists of the letters a, b and c. The instance ID is represented by a single digit. For the usage in the experimental system long hexadecimal encodings are used generated in such a way that the generation of identical Instance IDs is prevented.

In a first step the initial entity sends only a request to a single entity and initiates the Basic Workflow based on ID A. In the next step a Sub-Workflow is triggered with ID B, which triggers an additional Sub-Workflow

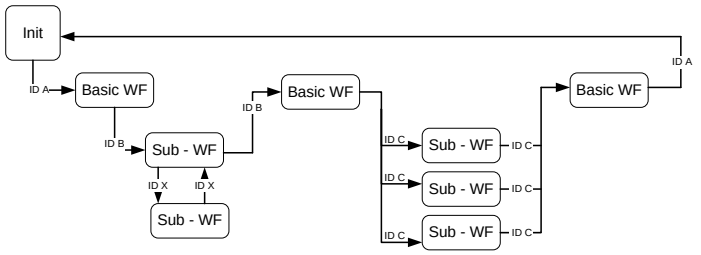


Fig. 9.5: Structured Workflow ID

via ID X. After the Sub-Workflow has received the information through a response with ID X it will provide its results via a message with ID B. The Basic Workflow continues and triggers then three Sub-Workflows with ID C. After these Sub-Workflows performed their tasks, the Basic Workflow completes and communicates the completion through a message with ID A.

This ID scheme allows the operating entities in the system to identify specific messages and communicate synchronously over an asynchronous communication system, it also allows extended monitoring and tracing through entities that are only subscribed to the message bus.

The Statistics Module (cf. Section 9.2.5) computes its results by evaluating the Structured Workflow IDs to correlate coordination request and response events. Additional modules as, for example, a monitoring module that shows the relationships between executed SON-Function instances and plots the logical workflows, also make use of the information provided by the ID scheme.

9.4 Implementation Summary and Findings

This chapter provided a detailed overview on the implementation of the SON coordination concept into a Demonstrator- and Experimental System. Section 9.1 introduces the general architecture, which consists of core modules that make up the SON Coordinator (Execution, Knowledge, Visualization) and a publish-subscribe based message bus. It is possible to attach additional modules to the message bus that provide extended functionality as, for example, network simulators or statistic modules.

Section 9.2.4.1 describes how coordination of the requests is performed and how the coordination logic is mapped from solution agnostic decision trees into Ponder2 policies that are used within the Execution Module.

It is shown how the conceptual baseline is mapped into an implementation and how the individual parts form a flexible and highly efficient SON Coordinator. This implementation is used for the evaluation of the concept. The flexible extensibility which is shown by the possibility to replace existing modules with new modules or other system components is important

during the transition phase from traditionally managed networks towards fully SON enabled networks.

Concerning the research area Reliability and Robustness, a contribution to the following research question has been made:

R3 How can the enforcement of and the compliance with all operational guidelines be assured?

The usage of a policy based decision making system shows, how it is possible to control the SON-Function execution in practice. It is demonstrated that the theoretical approach of representing the operator defined coordination logic within decision trees can directly be mapped to the policies that determine the automatic decision making process.

Concerning the research area Efficiency, a contribution to the following research question has been made:

E4 Which is the most efficient way to take the required coordination decisions?

The implementation takes up the question and by using using a policy based system proves that there is a very efficient way to evaluate and enforce a complex coordination logic. The efficiency comes partly from the efficiency of the policy based decision making itself and partly from the perfect match with the operation of the SON-Functions.

Concerning the research area Flexibility, a contribution to the following research question has been made:

F2 How can the decision logic be provided in a easily adaptable way?

The usage of a policy based system shows that complex coordination logic can be represented and evaluated in a flexible and efficient way. Policy based decision making systems are designed to separate the decision making process from the actual decision logic. The implementation makes use of that feature, and shows that it is possible to have a system that allows a flexible adaptation of the decision logic at run-time, without interruptions of the system operation.

F3 How can the system be designed such that it will be able, without problems, to cope with newly introduced SON-Functions?

There are two design-decisions that allow the system to easily cope with newly introduced SON-Functions. First, the fact that information like the definition of Impact-area, Impact-time, or coordination logic is only evaluated when required. This allows to provide new or updated information at run-time whenever needed. The second feature is the message based communication system that connects the individual modules, thus new functionality like new SON-Functions can be introduced at run-time without interrupting the rest of the system.

10. CONCEPT EVALUATION

For the evaluation the implementation of the SON Coordinator was used together with a commercial LTE Radio Network simulator and SON-Functions based on real-world algorithms. The simulated environment consists of 32 cells with 1500 moving users. It reflects the situation in a larger city including typical radio propagation issues as, for example, shadowing caused by buildings. The simulator allows a high degree of freedom for the setup of specific simulation scenarios. It is possible to change the configuration of the network by adapting tilt or throughput power settings for each cell and modify the handover parameters. All changes are performed with minimal delay and affect directly the recorded KPIs. The simulator provides a total of ten different KPIs which can be used as input to the SON-Functions and also for the evaluation of the effects of the performed configuration changes.

The demo setup contains a larger coverage hole that is surrounded by five cells. The effects of both, the coverage hole and the performed changes, will be visible in the KPIs. The effects of the SON-Function instance execution coordination were shown with a set of SON-Functions that perform coverage and capacity optimization and the optimization of the handover parameters. A detailed overview on the selected SON-Functions and KPIs is given in the following sections.

To show the effects of the coordination, the effects of the SON-Function instances on the system were observed, both with and without active coordination. The results without the coordination form a worst case base-line, which is used for comparison with the results when the coordination is enabled. Several experiments with variations of the coordination parameters like Impact-time, Impact-area and coordination logic were made, which allows to show the impact of the different components that are important for a successful coordination.

10.1 Experimental Setup

The evaluation was performed with a dedicated setup in order to be able to compare the results.

10.1.1 LTE Radio Network Simulator

The GP within the simulated environment is set to 5400 seconds simulation time. That means every 5400 simulated seconds updated KPI values are available. One 5400 second interval is referred to as one round. One experiment is run over 30 rounds. To reduce the time required for the experiments the possibility to accelerate the simulation is taken advantage of. With the maximum acceleration new KPI data is provided every 360 seconds instead of every 5400 seconds. The contained data still reflects the measurements of 5400 simulation seconds.

The enforcement of new configuration parameter values is instantly performed. Configuration updates are not dependent on any GP or enforcement delay. The simulated environment contains a coverage hole and a set of unbalanced cells with non-optimal coverage and capacity situation. Additionally, the settings of the handover parameters are not fully optimized. The system was reset after each experiment to start each time with an identical scenario. The movement of the users is based on a random walk scheme that causes the variations in the KPIs and therefore also the behavior of the SON-Function instances.

10.1.2 Used SON-Functions

To be able to resolve the coverage hole and optimize the handover parameter settings, CCO(RET), CCO(TXP) and MRO [3GP10b] were selected as the deployed SON-Functions. The implementations of the SON-Functions use algorithms from commercial deployments and are executed whenever their Monitoring-part detects a triggering situation within the provided KPI data. Further details on the respective SON use cases are provided in the given citations.

The combined coordination scheme was assigned to all used SON-Functions, which provides the possibility to modify the coordination logic according to the requirements without the need to adapt the implementation of the SON-Functions.

10.1.3 Used KPIs

For the evaluation of the effects of the executed SON-Function instances an appropriate set of KPIs is used. Not only the resolution of the coverage hole but also the effects of the capacity and handover optimization should be reflected in the KPIs.

- **Radio Link Failures:** Whenever a user moves from a cell into a coverage hole the connection will be terminated and the incident will be counted as a Radio Link Failure (RLF). Although there are also other reasons for RLFs, like missing coverage in buildings, a reduction of RLFs in the cells around coverage hole is expected, as soon as the coverage optimization succeeds.
- **Throughput per cell:** The CCOs functions are expected not only to close the coverage hole but also to optimize the capacity of the targeted cells. The expectation is to see an improved throughput in the optimized cells. The KPI shows the throughput in MBit/s per cell.
- **Handover Drops:** Are the measurement that provides the possibility to assess the results of the handover parameter optimization performed by the MRO SON-Function. Although the network simulator provides separate KPIs that show the number of too-early and too-late Handover (HO) drops only the aggregated number of HO drops is used, which is sufficient to assess the optimization results.

To show the effects of the coordination on the behavior of the SON-Function instances the changes performed by the CCO(RET) and CCO(TXP) function instances were monitored.

10.1.4 Coordination Logic

As indicated above, variations of the coordination logic and also variations of the coordination parameters were used. Apart from prioritizing SON-Function instances the possibility to block the SON-Function instance execution for a certain time was employed. This allows to overrule the fixed priorities if required. The concrete coordination logic is introduced for the individual experiments in their respective subsections.

10.2 Experiments

Several experiments to evaluate the effects of the SON-Function instance execution coordination were performed. Each experiment ran for 30 rounds and has been repeated for several runs.

10.2.1 Without Coordination

It is important to have a base line as a reference for the evaluation of the experimental results. The used base-line is an uncoordinated execution of the SON-Function instances. For this experiment all requests passed through the SON Coordinator but were not coordinated at all.

10.2.1.1 SON Coordinator settings

The SON Coordinator settings were chosen in a way that requests are not coordinated at all. The coordination logic for each of the three SON-Functions consisted of a single leaf that acknowledged all incoming requests. Due to this setting Impact-time and Impact-area were not relevant for the coordination decision and were set to zero.

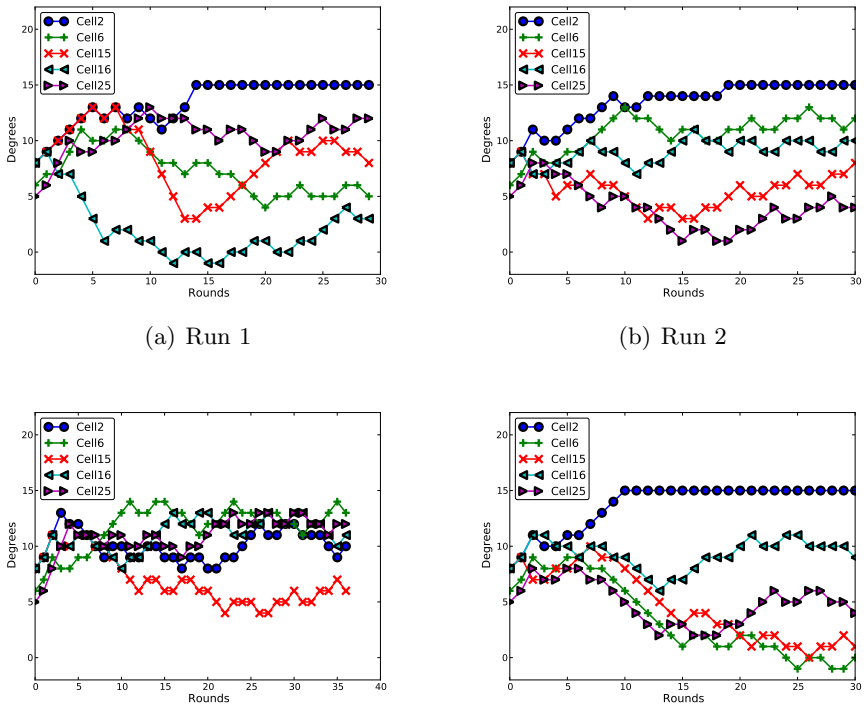
10.2.1.2 Results of uncoordinated SON-Function instance execution

The graphs in the following sections show the behavior of the system if the execution of the SON-Function instances is not coordinated at all. Results for four runs are given to show the differences in the behavior.

10.2.1.2.1 Antenna Tilt Changes Figure 10.1 shows the tilt changes for the cells around the coverage hole for four different runs of the experiment. It is important to know that increasing the values corresponds to tilting the antenna down. What is easily visible is that after some time the tilts are either not changing any more or are changing in a way that indicates that the algorithms are not able to find a better configuration. The tilt of some of the cells are changed forth and back repeatedly, while others oscillate on a very long time scale. The tilt changes of the cells are never identical, though in three of four cases cell two is heavily tilted down. Repeated changes in one direction followed by a subsequent change into the opposite direction indicates that the algorithm is basically no longer able to determine which change would improve the situation. Such behavior can be observed for Cell 15 in Figures 10.1(b) and 10.1(c).

10.2.1.2.2 Transmission Power Changes The changes of the Transmission Power for the cells around the coverage hole are shown in Figure 10.2. Compared to the antenna tilt changes it is evident that the overall number of changes is much smaller, and that even if the Transmission Power has been stable for several rounds, changes are performed. In the Runs 1 2 and 4 all changes are performed during the first twelve rounds, only in the third Run a power change is performed after 17 rounds. A big difference to the tilt changes is also that there is no phase where the changes start to oscillate. The Monitoring-part of this SON-Function seems to perform better when detecting the triggering situations.

10.2.1.2.3 Radio Link Failures are a typical side effect of coverage holes. Whenever a user with an active connections leaves the coverage area of a cell into an area without coverage an RLF will be detected, thus no handover can be initiated. Although RLFs occur also in situations when a user enters a building without coverage, or there is no neighbor relation



(a) Run 1

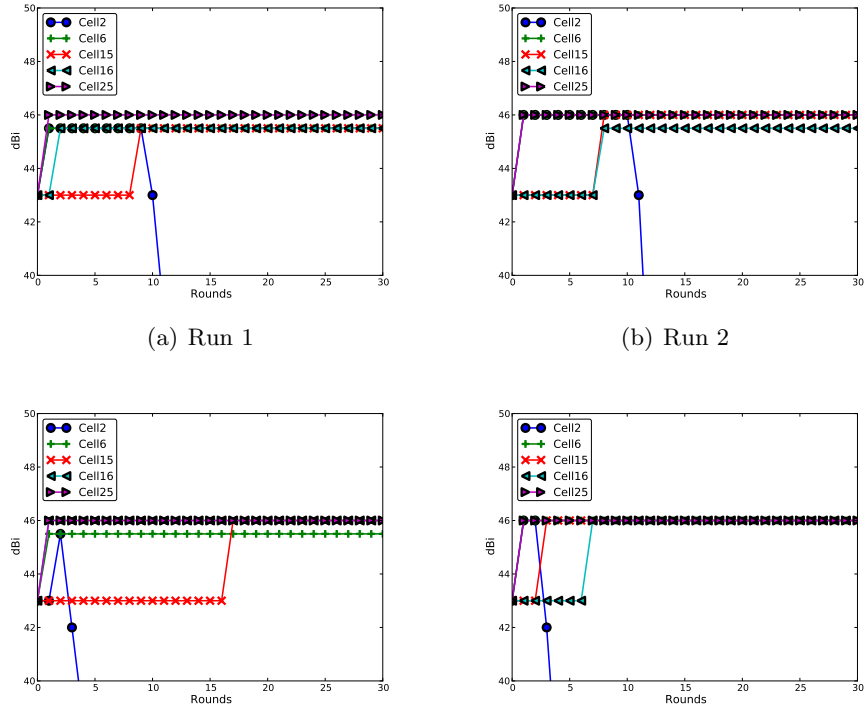
(b) Run 2

(c) Run 3

(d) Run 4

Fig. 10.1: Tilt changes without coordination

between neighboring cells, the KPI is of special interest for the evaluation as the overall number of RLFs should be reduced as soon as the coverage hole is closed. The KPI changes displayed in Figure 10.3 show not always satisfying results. In terms of RLFs the second run (cf. Figure 10.3(a)) shows the best results. For four of the five cells the number of RLFs is strongly reduced. The caveat to the result is that cell 25 experiences over 50% more RLFs compared to before the optimization. A similar result is visible in Figure 10.3(c) where the number of RLFs is almost 10 times higher than at the beginning. As a baseline for the uncoordinated execution of the SON-Function instances it was observed that it is possible to reduce the number of RLFs for the majority of the cells at the cost of increasing them for at least one cell. The result indicates that it was not possible to fully close the coverage hole, for some runs it can be assumed that it was increased in parts of the network. This it the case in Run 1 (cf. 10.3(a)) between Cells 6 and 16 and in the run displayed in Figure 10.3(d) between 6, 15 and 25.

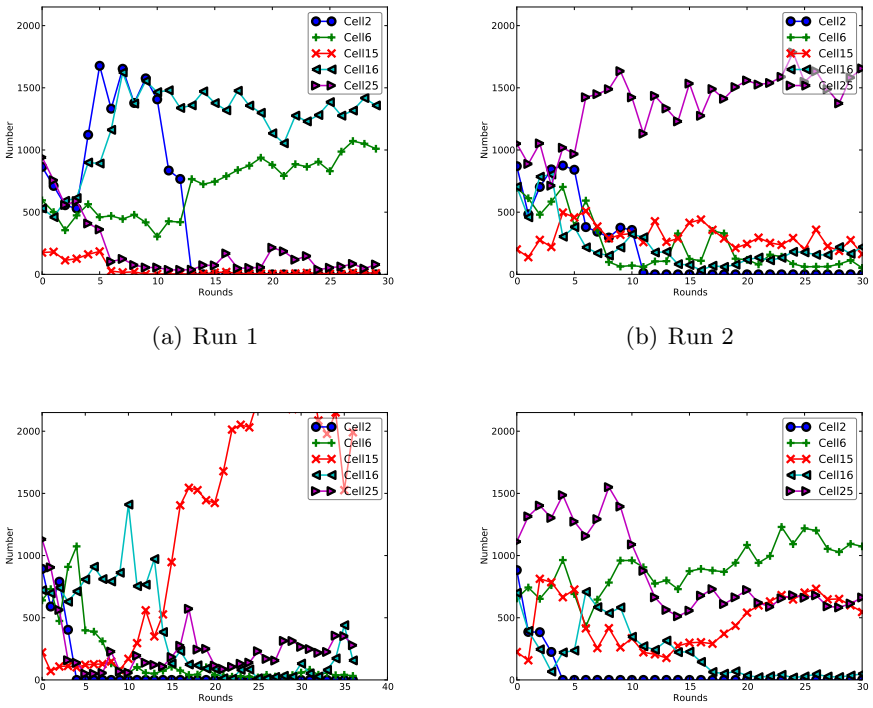


(a) Run 1 (b) Run 2

(c) Run 3 (d) Run 4

Fig. 10.2: Transmission Power changes without coordination

10.2.1.2.4 Throughput per cell is an important measure for the success of the Coverage and Capacity optimization, therefore the changes of the throughput KPI were monitored. The results in Figure 10.4 do clearly not indicate a success for CCO. Although the throughput could be increased for individual cells it is even reduced for most of the cells. Especially in Cell 2 the throughput is massively reduced. In all experiments it was reduced to almost zero which means that Cell 2 did not carry any traffic anymore. In an optimized network the average throughput of the neighboring cells should stay the same. Optimization attempts would only be able to equal out the load differences in the cells caused by the underlying traffic patterns. Looking at the changes clearly shows a degradation of the overall performance. This is even visible in Figure 10.5 where the average for the five cells of each Run is shown together with a plot of an overall average throughput. Only for Run 4 (cf. Figure 10.4(d)) the overall used capacity was slightly improved. In average the throughput was reduced from an average of 13 MBit/s down to about 10 MBit/s with extreme differences between the individual cells.



(a) Run 1 (b) Run 2

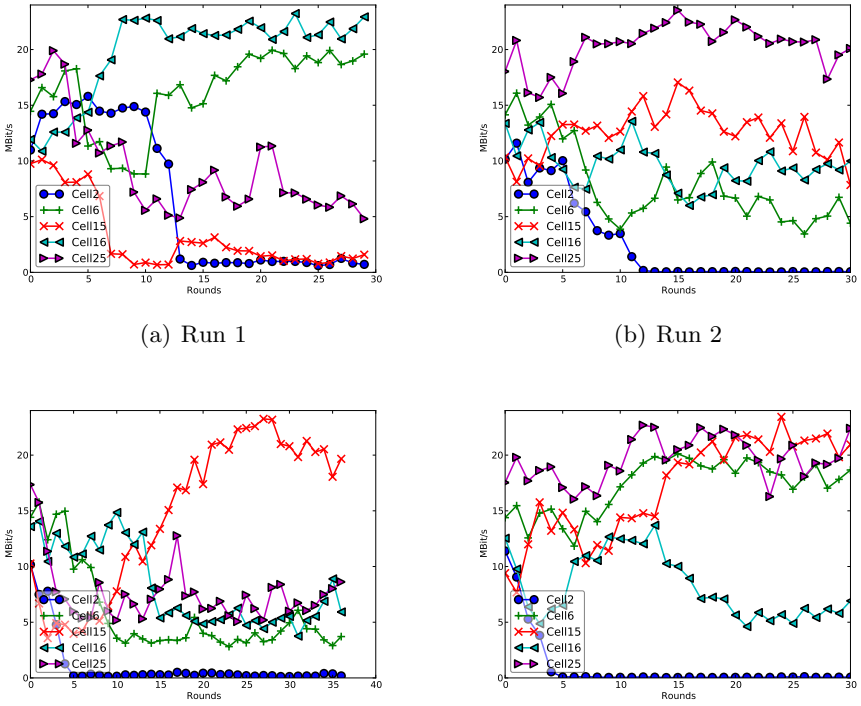
(c) Run 3 (d) Run 4

Fig. 10.3: Radio Link Failures per cell without coordination

10.2.1.2.5 HO drops reflect the success of the MRO SON-Function instance execution. Figure 10.6 that shows the changes in the number of handover drops uses a logarithmic scale for the y-axis to be able to show the behavior when the number of handover drops is reduced to an area around ten. During the first five to ten rounds there is a strong decline from a maximal number of 482 drops in Cell 6 down to a maximal number of 26 handover drops. What can also be observed is that the numbers never stabilize but start varying in an interval between zero and 22 handover drops per round. There are several potential reasons for this behavior. Either normal network behavior where changing traffic patterns lead to a varying number of users in the cells and a usual low percentage of handover drops. Another possibility is a characteristic conflict between the CCO SON-Functions and MRO. The continuous changes of the cell borders require a continuous adaptation of the handover parameter settings which cause the variations.

10.2.2 Individual SON-Functions

In order to determine an optimal coordination logic, the behavior of the individual SON-Functions was analyzed. The main focus was to find out how



(a) Run 1

(b) Run 2

(c) Run 3

(d) Run 4

Fig. 10.4: Througput in MBit/s per cell without coordination

many simulation rounds in average are required until no further improvements can be reached through a single SON-Function type. Therefore, test runs with instances of only a single SON-Function were performed.

10.2.2.1 CCO(RET) only

In a first setup the coordinator acknowledged only the execution of CCO(RET) SON-Function instances. Figure 10.7(a) shows that SON-Function instances are triggered continuously for all cells around the coverage hole. Most of the significant tilt changes are performed within the first ten rounds. After round ten the tilt changes either stabilize or start to oscillate. Especially the oscillating reconfigurations indicate a situation where the Monitoring-parts still detect a non-optimal situation but the Algorithm-parts of the SON-Function instances can not compute configuration changes that significantly improve coverage or capacity. This observation is supported by the changes of the KPIs for RLFs (cf. Figure 10.7(b)), the average throughput in the cells around the coverage shown in Figure 10.7(c), and the reduction of HO drops (cf. Figure 10.7(d)). The reduction of the HO drops shows that the CCO(RET) SON-Function, although it does not directly target

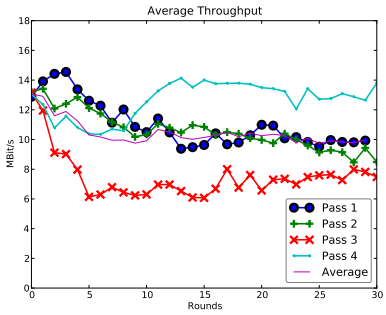
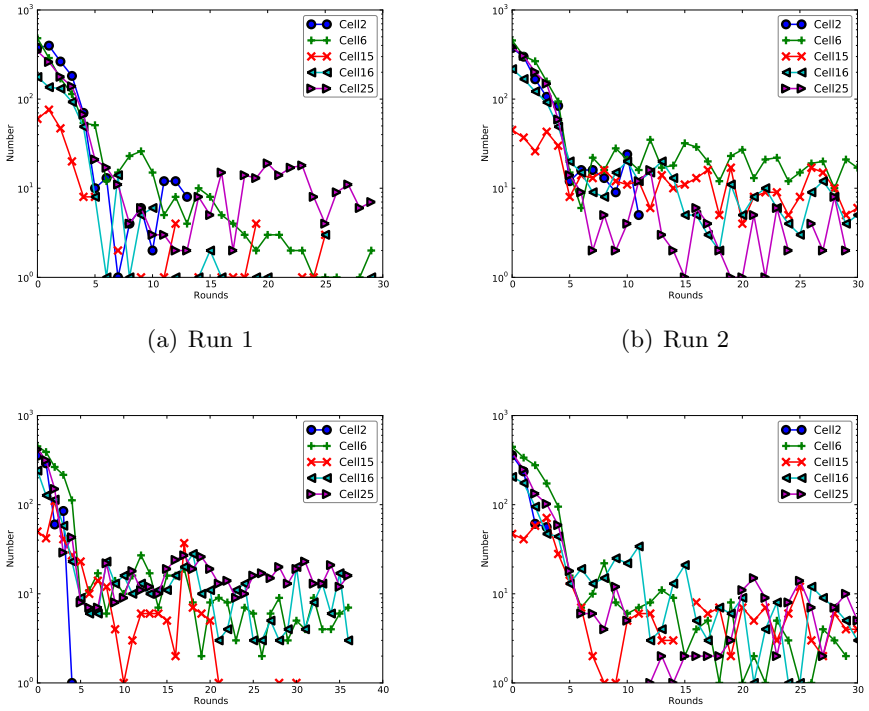


Fig. 10.5: Aggregated and Average Throughput for cells around the Coverage Hole without coordination

the HO optimization, has a significant impact. The adaptation of the antenna tilts changes the cell structure, especially the area which is covered by two neighboring cells. The quality of the non-optimized HO parameters settings can be significantly better after a change of the cell borders. This behavior points towards a characteristic conflict between the CCO and the MRO SON-Function. If an operator is not aware of this conflict, such an experiment can help to detect unknown conflicts.

10.2.2.2 CCO(TXP) only

Similar to the previous setup, the simulation was run for 30 rounds acknowledging only the execution of CCO(RET) SON-Function instances. The results of the example shown in Figure 10.8 are typical for this setup. There are only few configuration changes performed by CCO(TXP) SON-Function instances. These changes are always performed within the first ten simulation rounds, but not always as soon as it is shown in Figure 10.8(a). The Monitoring-part of the CCO(TXP) SON-Function seems to request only the execution of SON-Function instances when there is a high probability that performed changes contribute to a major positive effect. Another visible result, is that the execution of only CCO(TXP) SON-Function instances does not significantly contribute to the resolution of the coverage hole or the optimization of the number of HO failures. Due to these results, the CCO(TXP) SON-Function instances are used as a means to fine tune the changes performed by the CCO(RET) SON-Function instance. This results in a prioritization of the SON-Function instances where changes of the antenna tilt (CCO(RET)) receive a higher priority than changes of the transmission power (CCO(TXP)).



(a) Run 1 (b) Run 2

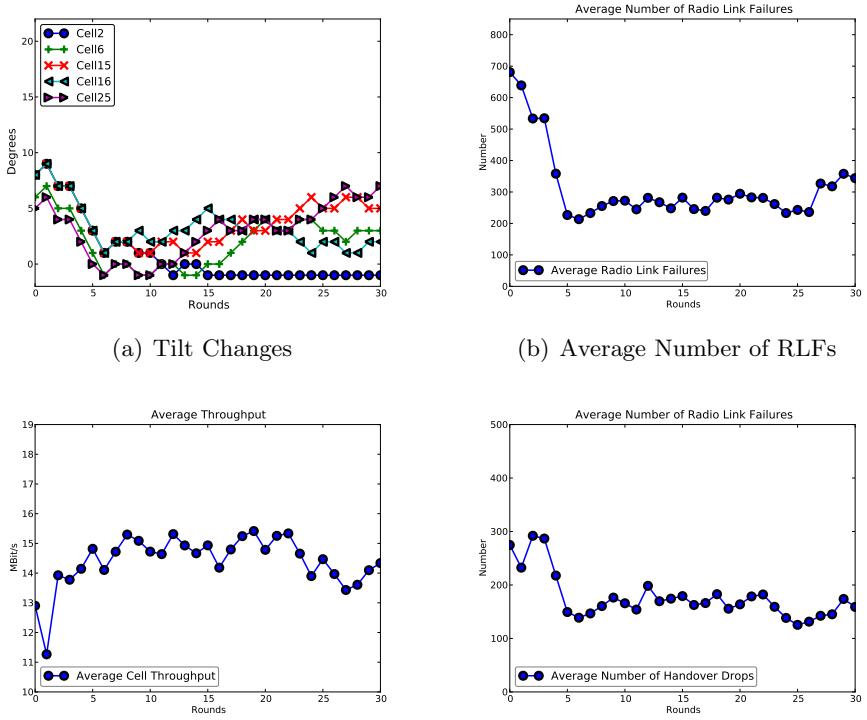
(c) Run 3 (d) Run 4

Fig. 10.6: Handover drops per cell without coordination

10.2.2.3 MRO only

For the last series of single SON-Function experiments only the execution of MRO SON-Function instances were acknowledged. The focus of this experiment was to determine how many rounds are required to optimize the HO parameter settings between the cells. It is important to note that the MRO function does not affect the coverage area of a cell, and therefore no reduction of RLFs or an increased throughput caused by the resolution of the coverage hole is expected. The KPI changes shown in Figure 10.9 result from the variations in the underlying traffic patterns. The optimized HO behavior has no significant impact. MRO optimizes three separate HO parameters, Handover Offset Max, Handover Offset Min and Handover Offset Bias. MRO SON-Function instances perform the optimization on cell pairs, however, due to the large number of cell pairs and the required individual plots for the parameters no explicit figure for the handover parameter changes is given, instead the affected KPI is used.

Figure 10.9(d) presents the changes in the number of HO drops. Similar to the previous experiments the results converge within the first ten rounds. The analysis of the SON-Function execution requests handled by the SON



(a) Tilt Changes

(b) Average Number of RLFs

(c) Average Throughput

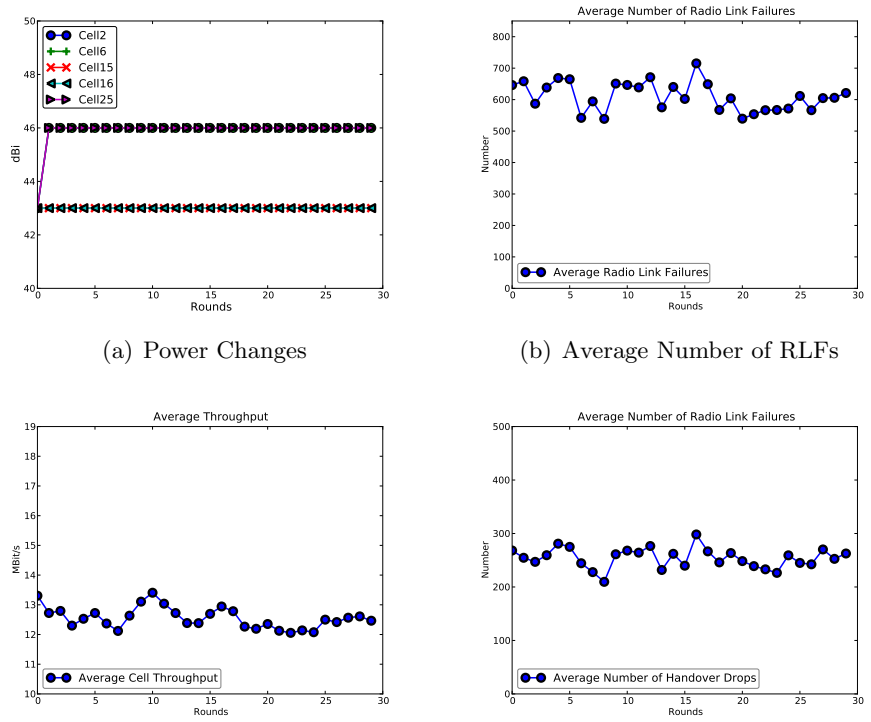
(d) Average Number of Handover Drops

Fig. 10.7: Results of a CCO(RET) only experiment

Coordinator showed that requests for the monitored five cells were received for all 30 rounds. In Figure 10.9(c) the same KPI changes are visualized on a logarithmic y-axis. This shows that there is a small variation of the HO drops throughout the experiment. The MRO SON-Function instances try to eliminate this minimal number of HO drops but do not reach their targets. There is probably an area between the cells with only minimal overlap, where no setting of the HO parameters will prevent all HO drops. In the end, the changing traffic patterns cause the variations in the number of HO drops.

10.2.3 With Coordination

After determining the coordination logic for the deployed SON-Functions and the network specific coordination parameters like the Impact-time an analysis of the effects of the coordinated SON-Function instance execution on the network was performed.



(a) Power Changes

(b) Average Number of RLFs

(c) Average Throughput

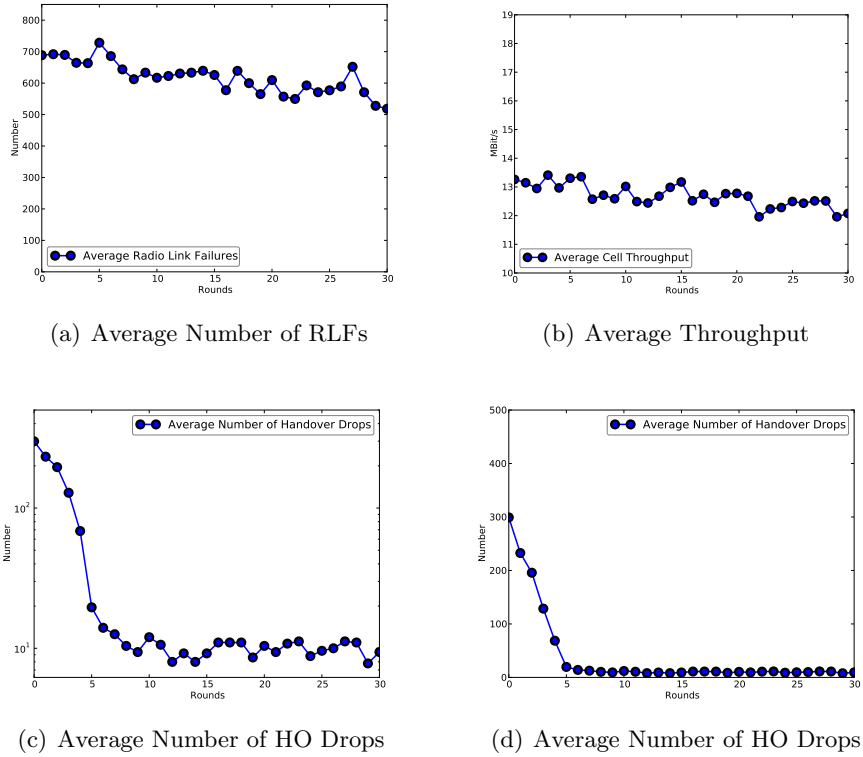
(d) Average Number of Handover Drops

Fig. 10.8: Results of a CCO(TXP) only experiment

10.2.3.1 SON Coordinator settings

For the coordination of the SON-Function execution requests the following settings were chosen for the SON Coordinator.

10.2.3.1.1 Coordination Logic: The configured coordination logic was rather simple and employed only Algorithm Coordination. Since the SON-Functions were built with an assigned combined coordination scheme, all action execution requests were simply acknowledged. Due to their strong effects, the CCO(RET) SON-Function instances were given the highest priority. Then in a decreasing order priorities were assigned to CCO(TXP) and MRO. With only a priority based coordination, it can be observed that, for some cells, there will be continuous tilt changes during the complete experiment. These results showed that these changes did not only not contribute to any major changes, or were often oscillating, but also blocked the required CCO(TXP) and MRO SON-Function instances. The assessment of the behavior is that the Monitoring-part of the SON-Functions is not able to fully detect whether an additional configuration change would positively contribute to the ongoing optimization. Therefore the configura-



(a) Average Number of RLFs

(b) Average Throughput

(c) Average Number of HO Drops

(d) Average Number of HO Drops

Fig. 10.9: Results of a MRO only experiment

tion changes are continuously requested. In order to avoid this behavior, an operational guideline was introduced to block the tilt changes after they would not longer contribute to the optimization. Similar effects could be observed in the relation between CCO(TXP) and MRO SON-Function instances. The resulting coordination logic enforces both, the prioritization and the the SON-Function instance blocking, to allow the execution of lower priority SON-Function instances.

From previous experiments it was assessed that after eight to ten executions of a particular SON-Function, the subsequent executions do no longer contribute to the optimization. As a consequence of this result the coordination logic will for the first ten simulation rounds exclusively allow tilt changes, for the ten subsequent rounds only CCO(TXP) SON-Function instances are permitted followed, by ten rounds of handover parameter optimization through MRO SON-Function instances.

10.2.3.1.2 Impact-time through the analysis of the uncoordinated system behavior it was detected that the SON-Function instances are sensitive to the input data they were provided. The results of the SON-Function in-

stance execution depended on when during the GP the configuration changes were performed. To maximize the positive effects the Impact-time was adapted in a way that if a configuration change had been enforced within a GP no further configuration changes were allowed during the subsequent GP (cf. Section 6.2.3). In the setup the enforcement of the configuration changes is almost instantaneous and the performed changes show immediate effects. For this reason Enforcement-time and Visibility-delay were set to zero. In order to make sure that only valid input data was used the protection time was set to 1.14 times the duration of the GP. The duration of the 5400 seconds simulation time for a GP is reduced through the simulation speedup down to 360 seconds. To block requests in subsequent GPs an appropriate protection time has to be chosen, which is long enough to block a request in a subsequent GP, but not too long such that it blocks even requests in the next but one GP. An analysis of the temporal characteristics of the coordination was performed. The timing is counted in seconds after the end of the previous GP.

- All algorithm execution requests arrive within 20 seconds
- Depending on the number of incoming requests, the processing within the SON Coordinator takes up to 35 seconds (cf. Figure 10.10).
- Although only algorithm coordination is applied the SON-Function instances request permission for both, algorithm and action execution
- Therefore, in a worst case, the run-time of a SON-Function instance is 70 seconds plus the processing time required for monitoring and Algorithm-part
- In the best case the coordination induces a delay of only a couple of seconds
- The execution of monitoring and Algorithm-part does in total not take more than 10 seconds
- The earliest start of the Impact-time is at 13 seconds
- The latest start of the Impact-time is at 100 seconds

Based on this analysis the minimal and maximal length of the protection time was determined as shown in Figure 10.11. To block all algorithm requests in the subsequent GP it has to cover at least the first 20 seconds of the GP. That leads to a minimal Impact-time of 380 seconds. To avoid blocking SON-Function instances for more than one GP it may not cover more than the full subsequent GP. From the analysis of the starting times of the Impact-times a maximal length of 620 seconds was assessed. For the evaluation the Impact-time was set to 410 seconds, which is long enough to cover even delayed requests but not unnecessarily long.

10.2.3.1.3 Impact-area The analysis of the behavior of the SON-Functions showed that there is little or no interference between instances of the same SON-Function at neighboring cells. Therefore the Impact-area was restricted

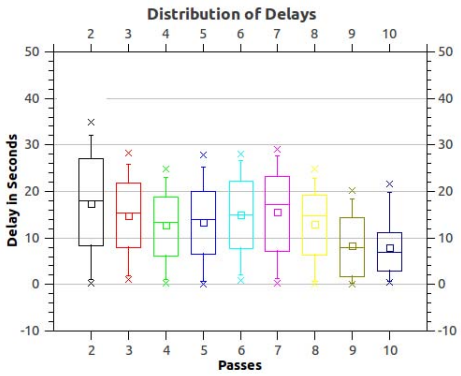


Fig. 10.10: Processing delay induced by the SON Coordinator

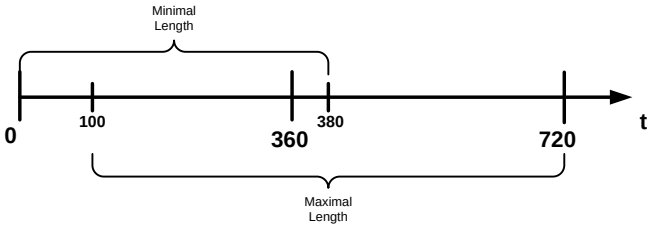
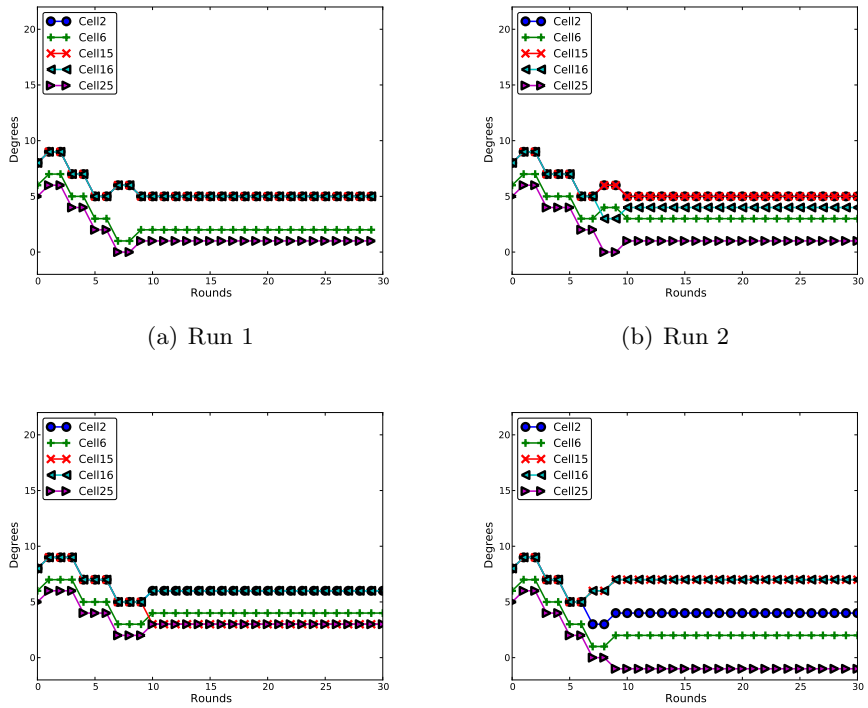


Fig. 10.11: Minmal and Maximal Length of Protection-time

to the Function-area of the SON-Function instance. For the CCO function instances this is only the targeted cell and for the MRO function instances the optimized cell pairs. It is important to note that it is only the cell pair that is under optimization. If cell A has the neighbors B,C, and D, which generates three neighborships to a (A-B, A-C, and A-D). If in the first GP MRO optimizes the settings for a-b, it will block in the subsequent GP optimizations for a-b but not for a-c or a-d.

10.2.3.2 Results of coordinated SON-Function instance execution

This section presents the results of four examples of coordinated SON-Function instance execution. The results clearly show a much more consistent behavior of the SON-Function instances in the individual passes and the resulting development of the KPIs. This clearly indicates that the improved consistency of the input data, which is reached through the extended Impact-time has the intended effects. The differences between the individual passes are caused through the independent operation of the SON-Function instances which take different approaches to reach their part of the global target to close the coverage hole and balance the load between the cells.

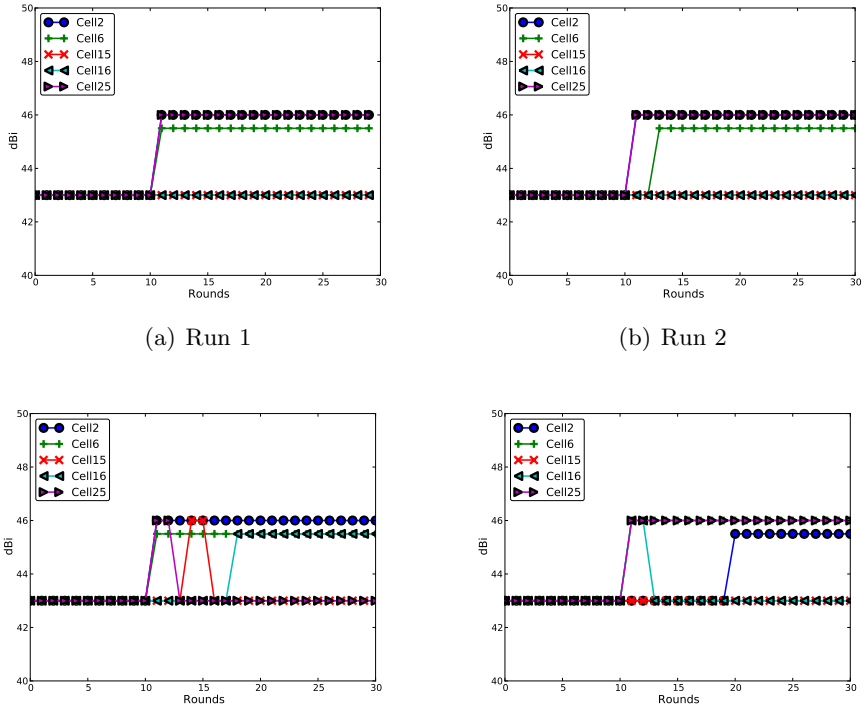


(a) Run 1 (b) Run 2

(c) Run 3 (d) Run 4

Fig. 10.12: Tilt changes with coordination

10.2.3.2.1 Antenna Tilt Changes The effects of the coordination are directly visible in plots of the antenna tilt changes (cf. Figure 10.12). The antenna tilt is never reconfigured in two subsequent rounds and latest after ten simulation rounds no further tilt changes are performed, since they are blocked through the coordination logic. The tilt changes between the different runs show large similarity. In the beginning, all considered cells are tilted down, after this initial change a series of up-tiltings are performed. After the coverage hole has been closed, the SON-Function instances start to balance the coverage and capacity of the cells. This optimizations results in a different tilting behavior for the individual cells. In Run 3 the configuration of the cells is very similar, while in Run 4 (cf. Figure 10.12(d)) the configurations are more widespread. But it still results in a balanced configuration with the antenna of the Cells 6 and 25 relatively strong uptilted and the antenna of Cell 15 and 16 only little uptilted compared to the initial configuration.



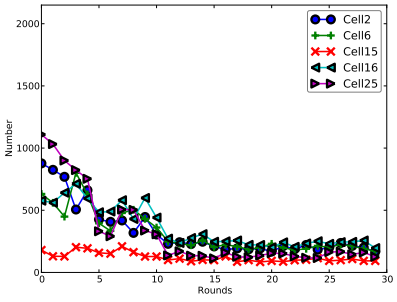
(a) Run 1 (b) Run 2

(c) Run 3 (d) Run 4

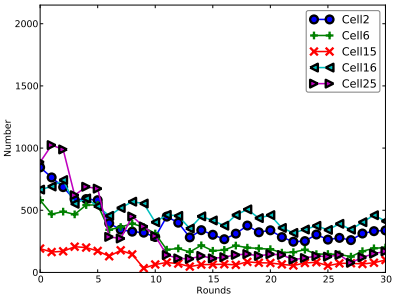
Fig. 10.13: Transmission Power changes with coordination

10.2.3.2.2 Transmission Power Changes The configuration changes for the transmission power confirm the assessment from the CCO(TXP) only experiments, in so far that the Monitoring-part more reliably determines the triggering situations for the SON-Function instance execution. There are only a few configuration changes during the ten rounds. In addition it can take some rounds until a triggering situation is detected and the configuration is changed. The adaptation of the transmission power is used to fine-tune the coverage and capacity optimization performed by the CCO(RET) SON-Function instances

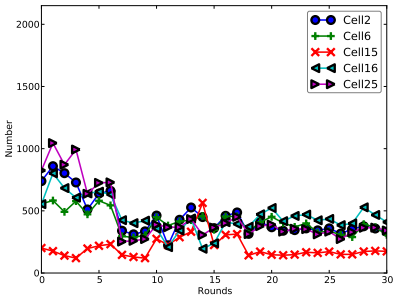
10.2.3.2.3 Radio Link Failures Compared to the RLFs results when no coordination was applied the coordinated SON-Function instance execution produces much better results. In each of the passes shown in Figure 10.14 the number of RLFs is clearly reduced or at least kept at the same level. A direct comparison of the trends of the aggregated average RLFs in Figure 10.15 shows a clear reduction of the RLFs. The figure shows plots of the average number of RLFs for the five cells and an average of the performed passes. The remaining RLFs do not necessarily originate from the original coverage hole but can also be caused by shadowing through buildings.



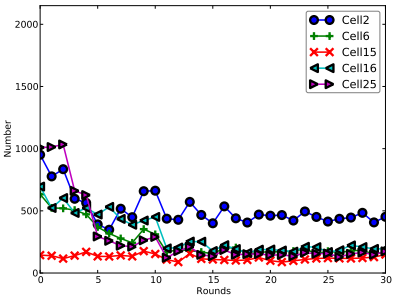
(a) Run 1



(b) Run 2



(c) Run 3



(d) Run 4

Fig. 10.14: Radio Link Failures per cell with coordination

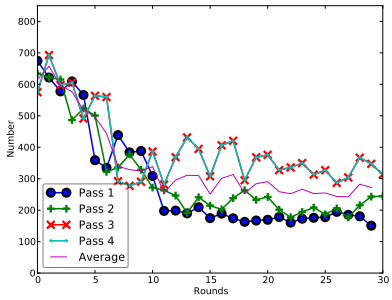
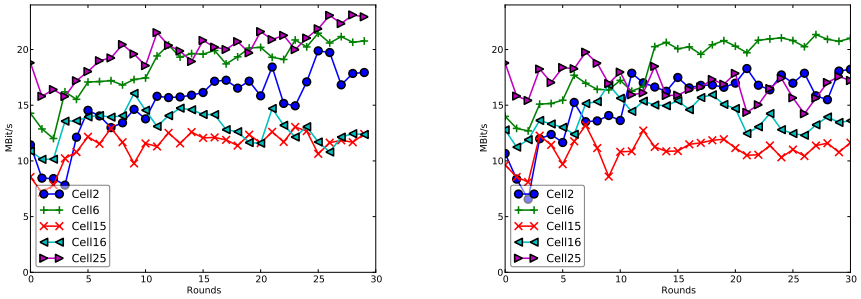


Fig. 10.15: Aggregated Average of Radio Links Failures

10.2.3.2.4 Throughput per cell The results of the capacity optimization is visible in the plots in Figure 10.16. In each of the passes the throughput for each of the cells was increased. The main improvement is reached through the resolution of the coverage hole. After the coverage hole has been closed the CCO SON-Function instances optimize the load for the individual cells which results in an additional increased average throughput. The coordinated execution of the SON-Function instances leads to a 20% overall increased throughput, while the uncoordinated execution resulted in a 23% reduction.

10.2.3.2.5 Handover Drops The number of HO drops in the experiments with an active coordination develops as expected. The antenna tilt and transmission power changes lead to slight decline of HO drops, similar to the CCO(RET) only setup. The number of HO drops is reduced from a maximum of 476 and a median of 275 in the beginning of the CCO phase to a maximum of 200 and a median of 159. The major reduction of HO drops is reached through the execution of the MRO SON-Function instances after round 20. The positive effects of the coordinated SON-Function instance execution are visible when looking at the comparison between the uncoordinated and coordinated plots in Figure 10.19. The figure shows boxplots for a set of 10 experiments with and without coordination. The figures show the numbers of the HO drops starting from the round in the experiments where the maximal number of HO drops for the first time drops below 25. In both setups only 5 rounds are required to reach this threshold. The difference is that for the uncoordinated execution MRO instances are executed from round 1 and only from round 21 in the experiments where coordination is used. Compared to the setup without coordination the overall number of HO drops stabilizes much faster and the number of maximal HO drops decreases more consistently if active SON-Function instance execution is performed. While instances of the CCO and MRO functions are executed concurrently the applied, optimized, HO parameter settings can be inval-



(a) Run 1

(b) Run 2

(c) Run 3

(d) Run 4

Fig. 10.16: Througput in MBit/s per cell with coordination

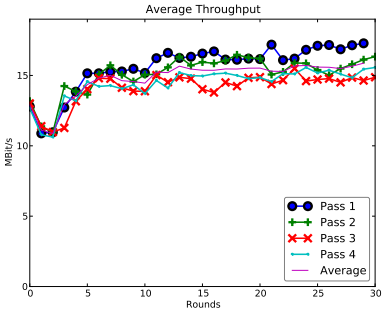
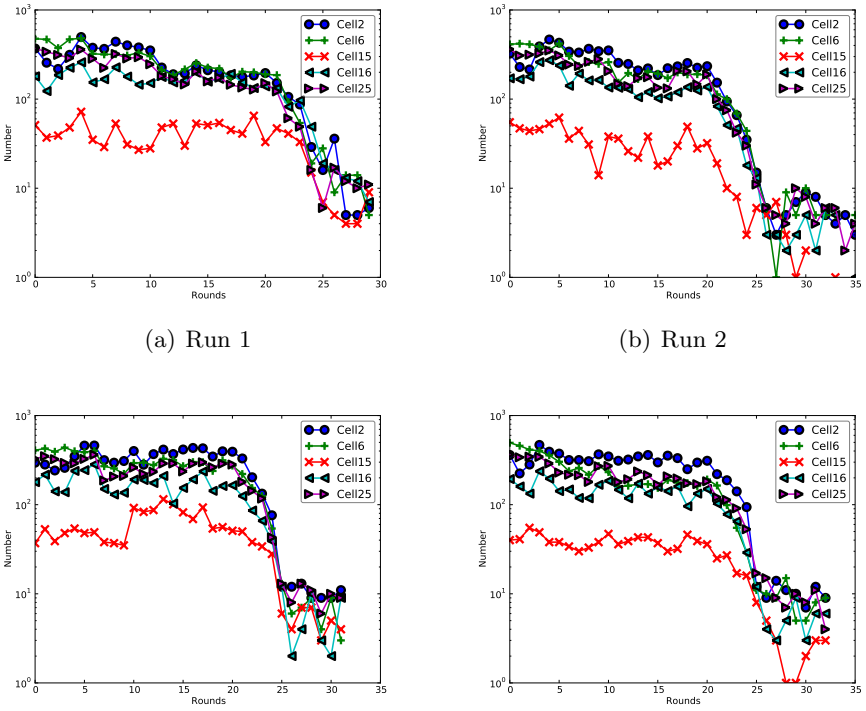


Fig. 10.17: Aggregated and Average Throughput



(a) Run 1

(b) Run 2

(c) Run 3

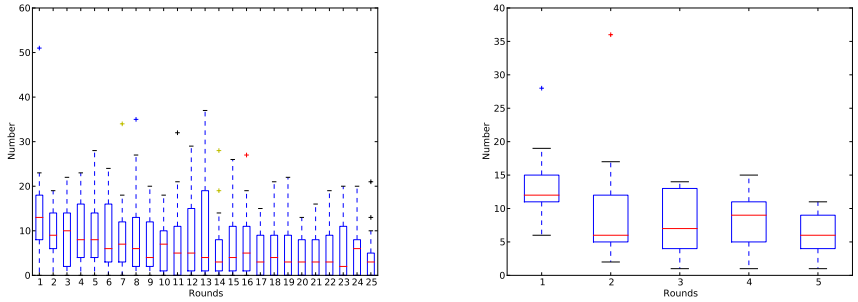
(d) Run 4

Fig. 10.18: HO drops per cell with coordination

idated when CCO SON-Function instances change the coverage area and therefore the border of the cells. These changes affect the HO drops, which needs to be resolved through the execution of additional MRO instances. The coordination resolves this implicit characteristic conflict, which results in a lower number of required MRO SON-Function instances to reach the same results, thus in fewer rounds.

10.3 Evaluation Summary and Findings

The experiments show the benefits and the positive effects of the coordinated execution of SON-Function instances on the overall performance of the network. Especially the differences between the uncoordinated and coordinated execution of CCO(RET) SON-Functions highlight the benefits of the coordination, thus the resolution and prevention of intra and inter SON-Function conflicts through the application of a coordination logic and a set of predefined operational guidelines (cf. research questions R1, R3. After the the coordinated execution of SON-Function instances the network shows a strongly improved cell throughput with a good balance of the load



(a) Without Coordination (Round 6-30) (b) With Coordination (Round 26-30)

Fig. 10.19: Development of the Number of HO drops after the maximum is for the first time below 25

to the different cells around the former coverage hole. The coverage hole can only be closed with an coordinated execution, which manifests itself in the comparison of the number of RLFs. Handover parameter settings are more consistently optimized with a much smaller number of executed MRO SON-Function instances. This reduces not only the load in the system but additionally allows the execution of SON-Function instances that would otherwise be blocked by active MRO instances. The experiments have shown different positive aspects of the coordinated execution of SON-Function instances, which are directly reflected in the better results of the used KPIs. In general, the results highlight the benefits for the reliability and the robustness of the SON-Function execution through the coordination (cf. Chapter 1.3.1). Another aspect that has been shown are the benefits of the policy based decision making, where the coordination logic of the solution agnostic decision trees has been represented as policies. The minimal delay of the request processing shows that the decision making process really benefits from the an appropriately defined decision tree. Decision can be taken very fast if the important parts of the decision logic are placed close to the source of the decision tree, especially if the additional information that is required is minimized and provided in a highly efficient way (cf. research question E4 and E5).

Two important aspects are presented in more detail in the following sections, the conflict resolution and the possibility to optimize the results of give SON-Functions without changing the SON-Functions themselves but influencing their behavior through specific coordination logic and operational guidelines.

10.3.1 Conflict Resolution

In a network with only these three deployed SON-Functions (CCO(RET), CCO(TXP) and MRO) there is already conflicting behavior which can have a negative impact on the network performance (cf. research question R1). The observed conflicts cannot be resolved through co-design, since the functions should be kept separated and not being merged into a single super-function in order maintain the possibility to execute their functionality separately. Each of the SON-Functions shows both intra- and inter-function conflicts. The intra-function conflicts are caused by the need to process only input data where the changes performed by a previous SON-Function instance with the same targets are reflected for a full GP. The selection of an appropriate Impact-time allows to assure that this requirement is fulfilled each time the execution of a SON-Function instance is acknowledged.

The behavior of the deployed SON-Functions show the importance of good understanding of the structure and the operation of a SON-Function, and a sound conflict categorization (cf. research questions R4, E2, F1). Between the two CCO functions the conflict is an explicit characteristic conflict: instances of both SON-Function types modify the coverage area of a cell. Such a concurrent modification of the cell size can negatively affect the results of the individual SON-Function instances, even though tilt changes have much stronger effects compared to the modification of the transmission power. Through the assigned priorities to the SON-Functions and the sequence of the execution of instances of the two CCO SON-Function types, the conflicts can not only be resolved but the combined positive effects are maximized. The initial antenna tilt changes resolve the coverage hole and provide a balanced load in the cells, and the coverage area of the cells is subsequently fine-tuned through a small number of transmission power adjustments.

Each modification of the physical cell borders during coverage and capacity optimization can potentially invalidate previously optimized handover parameter settings. The employed coordination logic prevents these indirect characteristic conflicts and provides a stable cell layout, where the MRO SON-Function instances quickly can reduce the remaining number of HO drops.

10.3.2 SON-Function Optimizations

The analysis of the execution of instances of a single SON-Function type showed that, although no further major positive effects could be reached, the monitoring-parts of the SON-Functions continuously requested the execution of additional SON-Function instances. Due to the different priorities this can lead to situations where instances of a particular, high priority, SON-Function type are executed without any positive effects and at the

same time these instances block the execution of required instances of other SON-Functions. The behavior of the SON-Function Monitoring-parts indicates either an erroneous implementation or a situation where it is no longer possible to reliably asses the expected effects. In future SON enabled networks, the SON-Functions can be provided through multiple sources, for example, equipment vendors, the network operators or specialized SON-Function providers.

During the experiments it was impossible to change the implementation of the deployed SON-Functions. This situation is similar to the situation of a network operator for a future, SON enabled network. An adaptation of the functionality of the deployed SON-Functions requires the function developer to provide a new version. This is a suboptimal solution if the SON-Functions do not perform as expected, since it means either to deactivate the deployed SON-Functions or accept and manually correct their negative effects. If such a behavior is detected, the network operators could introduce specific operational guidelines to prevent the observed behavior. The SON Coordinator with an optimized coordination logic that contains these operational guidelines can be used to circumvent this problem (cf. research question R2, R3). The results of the behavior analysis in Section 10.2.2 were used to define an execution sequence, and enforce a strict serialization of the SON-Function instance execution through the SON Coordinator. This serialization takes two very important facts into account:

- **SON-Function Priorities:** Respect the priorities of the SON-Functions
- **Covergence time:** How many rounds are required until the SON-Function instances have stabilized

The active suppression of unnecessary SON-Function instances allows the execution of required but lower priority SON-Function instances. It also assures the exclusive execution of a single SON-Function as long as it is required to reach stable results.

10.3.3 Summary of Contributions to the Research Questions

This Section provides a summary of the contributions to the individual research questions The results of the evaluation are used to show the effectiveness of contributions, especially the concepts ,provided as answers to individual research questions.

Concerning the research area Reliability and Robustness, a contribution to the following research question has been made:

R1 How can the danger of negative effects through the execution of conflicting SON-Function instances be minimized?

The results of the evaluation show, that the concepts introduced for this research question actually provide the intended benefits. The

conflicts, that caused the negative effects that became visible in the uncoordinated function execution, were fully prevented.

R2 How can the network operator, despite the automation, still be in full control over the ongoing processes in the network? and

R3 How can the enforcement of and the compliance with all operational guidelines be assured?

The benefits of the possibility to provide operational guidelines that prioritize the different SON-Functions and also to suppress SON-Function instance execution to allow the configuration changes performed to show their full effects are demonstrated with the performed evaluation. Without this possibility it would not have been possible to reach the presented results.

Concerning the research area Efficiency, a contribution to the following research question has been made:

E4 Which is the most efficient way to take the required coordination decisions?

The efficiency of the usage of a decision tree based coordination logic represented as policies is directly reflected in the evaluation results. They clearly show, that the negative effects of intra- and inter function conflicts can be successfully prevented.

E5 How can the required context information be provided efficiently to the system controlling the SON-Function instance execution?

The coordination logic for the used scenario requires different types of context information. On the one hand the information about the active SON-Function instances and their Impact-area and Impact-time but on the other hand also additional information about the number of optimization attempts for particular cells is used. Since the policy system has immediate access to this context information it has he possibility to efficiently prevent the conflict behavior between the individual SON-Function instances.

Part IV

CASE STUDY: SON-Function DEVELOPMENT

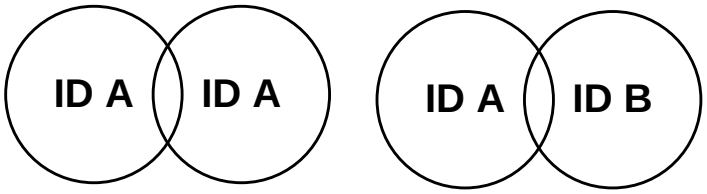
11. CASE STUDY: PHYSICAL CELL ID ASSIGNMENT SON-FUNCTION

This Chapter presents a case study based on a single SON use case. PCI assignment [3GP10b] is used to show the mapping of a use case to an implementation of the SON-Function and the design-time steps to prepare the coordination of the SON-Function once it is deployed. This particular SON-Function is required for self-configuration during the initial network deployment and the evolution of the network but also for self-optimization since a sound PCI assignment is relevant for multiple other SON-Functions.

11.1 Introduction to Physical ID Assignment

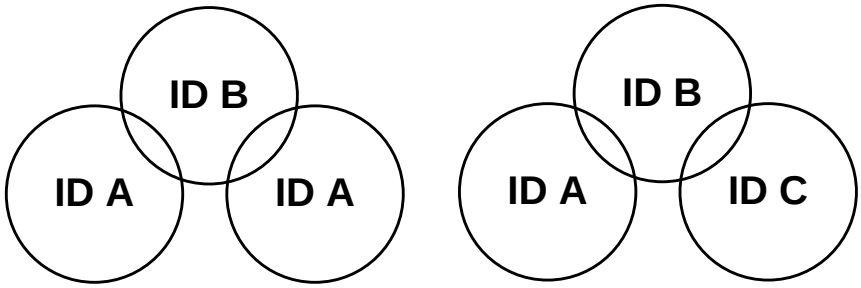
A fundamental parameter for the LTE radio configuration is the LTE reference signal sequence. It is called the Physical Cell ID and used as regionally unique identifiers on the physical layer. There are two reasons why these reference signal sequences are used as identifiers. First, they can be read within a very short time interval (5 ms) but more importantly they are constructed in a very robust way against interference which is important for their reliable identification. The automated configuration of physical cell identities is one of the key SON use cases defined by the 3GPP [3GP10b]. Only a small subset of the NGMN use cases [Leh07a] were rated to be relevant by the 3GPP. In general these are those use cases that can not or only hardly be solved with vendor proprietary solutions.

The starting point for the construction of PCIs are 168 pseudo random sequences [3GP12a], the 'cell identity groups'. For each of the cell identity groups three orthogonal sequences are constructed which results in a total of 504 usable PCIs [3GP08]). In an LTE network domain there are obviously more than 504 cells, which necessitates the reuse of PCIs. For a sound PCI assignment two basic requirements have to be fulfilled. In [3GP09a] network configuration parameters are categorized based on how they can be configured and if their configuration is dependent on the configuration of other parameters. PCIs are classified as B3 parameters therefore they need to be configured in a **'collision-free'** way, which means that 'neighboring cells' need to be assigned different PCIs. Neighboring cells are defined here as a set of cells which can be received by an UE at a specific location. To be able to receive multiple cells at a specific location the coverage areas of the cells need to overlap at the specific location. Figure 11.1 shows both a



(a) Colliding PCI Assignment (b) Collision Free PCI Assignment

Fig. 11.1: Colliding and Collision Free PCI Assignment



(a) Confused PCI Assignment (b) Confusion Free PCI Assignment

Fig. 11.2: Confused and Confusion Free PCI Assignment

colliding and a collision free PCI assignment. Neighboring cells are represented through overlapping circles. The PCIs are identified by letters. A colliding assignment is shown in Figure 11.1(a) where both cells have been assigned the PCI A. To resolve this issue a different PCI has to be assigned to one of the cells, as shown in Figure 11.1(b).

A collision free assignment is a necessary but not sufficient requirement for the proper assignment of PCIs. In addition the PCI assignment has to be **'confusion free'**.

An assignment is confusion free if there is no cell in the network that has two or more neighboring cells with identical PCIs.

Figure 11.2 shows both, a confused (Figure 11.2(a)) and a confusion-free assignment (Figure 11.2(b)).

The reason for the confusion free requirement is rooted in the way handovers of UEs between cells are done in LTE. As introduced, reliably reading a PCI takes 5ms and is part of the regular operation of a UE. Therefore the PCI used to identify the target cell for a handover. Reading the globally unique E-UTRAN Cell Global ID (ECGI) requires additional signalling and measurements on the UE side which introduces additional overhead and therefore is avoided whenever possible. A UE will only be requested to read a cells ECGI in rare situations, for example during an initial network

setup phase or after the introduction of a new cell in order to populate the eNodeBs NRT. The NRT is heavily used for handover management. It contains the PCIs of the neighboring cells and all additional information that is required to perform a handover to the cell identified by a given PCI.

In case of a non-confusion free assignment, a UE could requests a handover to a cell identified by a particular PCI. If the NRT maintained by the serving eNodeB already contains an entry for the provided PCI it will not request the UE to provide the ECGI of the target cell. The handover will be prepared based on the available information. Since LTE uses hard handovers, the PCI confusion will result in a handover failure if the targeted cell is not identical with the one with the identical PCI in the NRT.

11.1.1 Restrictions to PCI Assignment

Operators can have different views on how to assign PCIs in the network. In order to carefully deal with a 'limited resource' a high 'reuse-rate' could be applied. In this case as few PCIs as possible are assigned and the same PCIs are repeatedly configured to most of the cells. There are also other possible assignment strategies like assigning PCIs in a way that two cells with the same ID have a maximal distance from each other.

Although, the complexity of a collision and confusion free PCI assignment seems to be quite reasonable with the availability of 504 IDs the impact of a PCI assignment should be considered: every reconfiguration of a cell's PCI may require a reinitialization of the cell or even a restart of the responsible eNodeB, which causes a service interruption. Therefore it is important to identify reasons why the configuration needs to be changed on the one hand to be able to act if a reconfiguration might be required and on the other hand provide an assignment that proactively reduces the number of reconfigurations.

11.1.1.1 Related Radio Parameters

The configuration of other radio parameters is directly dependent on the configuration of the PCI. The assigned PCI directly influences the structure of the Uplink and Downlink Reference Signals of a cell [3GP12a]. Especially if collaborative multipoint transmission with joint beamforming is used the downlink reference signals of neighboring cells should be orthogonal to each other. Since there are six sub-carrier groups, an assignment NID [1] mod 6 should be used. For the uplink reference signals, 30 sequence groups are defined. Neighboring cells should not be assigned identical sequence groups, which implies that neighboring cells should have different PCIs according to NID mod 30. This is an important example how other configuration param-

[1] Numerical Representation of the Physical Cell ID (NID)

eters which are derived from the PCI configuration impose requirements on the assignment of PCIs in the network.

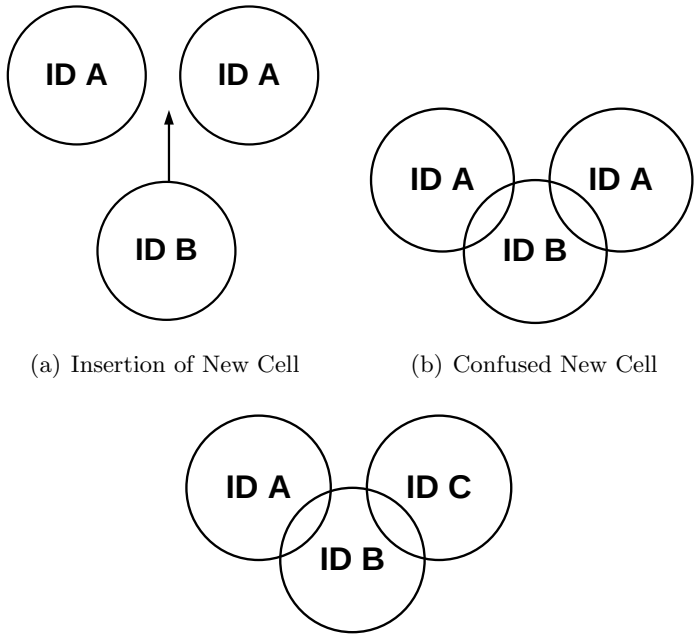
11.1.1.2 Changes of Neighborships

For an initial assignment in a non-operational network there is plenty of time to evaluate the planned but theoretical neighbor structure of each cell and assign the PCI in a collision and confusion free way. But there is not only a high probability that in reality not all neighborships are as planned but they can also easily change due to variations in the radio propagation properties, which is a typical phenomenon observed between the seasons or if buildings are modified or newly constructed. Such deviations from the planned network layout and the involved mal-configurations will only be detectable in the operational phase of the network. Whenever such an erroneous configuration is detected one or more cells have to be reconfigured.

11.1.1.3 Network Evolution

Apart from the difficulties to determine the actual network layout in a pre-operational phase the layout of the cell structure changes as the network evolves over time. New cells are added to either close coverage holes or to provide more capacity at high traffic hotspots. Cells are replaced by a set of smaller cells or removed when the coverage is provided through larger cells. Another important fact that has to be kept in mind is that especially in the beginning of LTE network rollouts there will only be LTE island deployments, typically in areas where high capacity is required, for example, at airports, trade fair areas or within large cities. These islands are then step-wise extended with additional cells until a full coverage is reached. To configure the PCIs of the cells added to such an island properly, the PCI assignment of the neighboring cells has to be analyzed. The results of the analysis are used to assure that the PCI of the newly introduced cell does not collide with any of the neighboring cells. But especially when new cells are added to an existing cluster of cells, the newly added cell is potentially already confused by two or more neighboring cells which use identical PCIs as shown in Figure 11.3. During the evolutionary growth of the network it is important to detect and resolve such mis-configurations as soon as possible.

The confusion probability increases when the coverage areas of multiple deployment islands start to overlap. Each of the islands has been configured collision and confusion free, but if the same PCIs have been used in all islands the probability for the cells 'connecting' islands to be confused is rather high. Since the confusion is caused by already configured cells that have previously neither been direct neighbors nor neighbors of neighbors, which means this is not a mis-configuration until they are connected by the



(a) Insertion of New Cell (b) Confused New Cell

(c) Resolved Confusion

Fig. 11.3: Confusion of a newly inserted cell

new cell. There is no possibility to avoid this confusion by selecting another PCI for the newly introduced cell. In Figure 11.3(a) the new cell with PCI B 'connects' two cells which belong to previously unconnected coverage islands. Each of the cells use the PCI A. This cell insertion causes an instantaneous confusion of the new cell (cf. Figure 11.3(b)). An additional reconfiguration of the PCI of one of the cells that cause the confusion is required to resolve this confusion, shown in Figure 11.3(c).

Another source of collision and confusion is whenever a high capacity island deployment which consists of multiple small cells is complemented by larger macro cells. With the advent of HetNets whith their layered cell structure where the capacity of macro cells is complemented with numerous small pico, micro and femto cells this scenario will become an everyday scenario [ea11]. In such deployments macro cells have a common coverage area with several of the small cells. In case the PCIs within the island deployment are assigned using a high reuse rate, the new macro cell will be confused by a larger set of cells.

Independent of the reason for the confusion of the new cell, all but one of the cells that cause the confusion have to be assigned a new PCI which can have a major impact on the operational efficiency.

11.1.1.4 Reduction of available PCIs through partitioning of the ID Space

Instead of operating with the full set of possible PCIs there are many reasons why an assignment is only done with a subset of the available PCIs. In order to pro-actively avoid repetitive reconfigurations (and the service interruptions they cause), network operators can divide the available ID space into separate blocks and use them for assignment. Sets of IDs could be reserved for different cell types, like one block for macro cells, one for pico cells and so on. Especially for the usage with Closed Subscriber Group (CSG) femto cells, the 3GPP has defined the possibility to exclusively reserve and signal a group of PCIs [3GP10b]. This is of special interest as a dedicated subgroup of PCIs for femto cells helps to decrease the power consumption of the UEs. Femto cells broadcast additional information to allow the UEs the detection of potential target cells and avoid attempts to camp on a CSG cell where the UE has no access rights. The broadcasted information contains a special CSG ID to identify a particular CSG and an access flag. The access flag allows general differentiation between different access schemes like CSG, open or mixed. An open cell operates like any other cell and is accessible by any UE. Those cells which indicate mixed access, allow in general the usage by any UE but give precendence to a set of registered UEs. The CSG ID and the access flag can be obtained by the UEs through additional data acquisition which uses additional battery resources. To avoid those kinds of measurements, Wu et. al. [WJWZ10] propose the usage of groups of PCIs to identify Femto Cells and the access scheme they use without any additional UE based measurements.

If in a network that makes use of NE from multiple vendors a single NM system for all NEs is available, the PCI allocation can be coordinated at the NM level; PCIs for the respective vendor domains are then configured using the Automatic Radio Configuration Function (ARCF) being part of the Self-Configuration Management Integration Reference Point (IRP) defined in [3GP09b]. However, groups of IDs could also be assigned to different network domains. In case several domain management systems are used, each of the systems that are responsible for the different vendor clouds could be assigned such a separate group of IDs. This distribution of ID ranges thus avoids the problem of the coordination of PCI configuration across vendor domain borders.

Another important reason for the usage of PCI groups are cells located at spectrum license border locations. In these areas a collision and confusion free PCI assignment with respect to a neighboring network operating on the same frequency has to be found. The mobile network operators are not able to influence the ID assignments of the respective neighboring networks. Therefore they could agree on which operator configures which PCI group to the cells in the border area, so that no further coordination is required.

As it has been shown there are many reasons for separation of the available PCIs into subsets. Finding a way to provide a set of reasonable subgroups of PCIs is a complex task. Up to now there is no final solution to the problem and research in this area is still going on. Especially with the introduction of femto cells a static assignment of PCIs will not be sufficient. Research tries to find ways for dynamic sub-partitioning that meets the requirements of the networks. Lee et al. [LJSS09] give an example how PCI groups could be dynamically reserved for the usage with a varying number of femto cells. Dividing the available IDs into smaller groups of IDs leads to the conclusion that it is important to think about the way the IDs are assigned. It requires a strategy that performs an assignment with small number of IDs which still can handle changes in the cell layout, caused especially by the introduction of new cells, to a large degree without requiring any reconfiguration of the already active cells.

The following paragraph summarises the assignment criteria which have been derived from all the factors that have an influence on the assignment and shows examples of different assignment approaches that can be used for SON.

11.2 PCI Assignment SON-Function

The previous Section already described the most important initial step for the development of a SON-Function. Section 11.1 provides a thorough analysis of the PCI assignment use case. Section 11.2.1 summarizes the criteria for a proper PCI assignment. The next steps that need to be performed is the assignment of functionality to the conceptual building blocks of a SON-Function, the development of the used algorithm that considers all aspects and requirements of the assignment with an appropriate Impact-area and the specification of the coordination logic as well as the selection of an appropriate coordination scheme.

11.2.1 Criteria for PCI Assignment

To reach a well defined assignment of PCIs which is in addition also very robust against changes of the cell layout, several requirements have to be satisfied. This Section gives a general overview on the criteria derived and raised from the previous Sections.

- **Collision- and confusion-free assignment:** This is the most obvious requirement on the assignment of the PCIs, which has already been emphasised.
- **Avoid reconfigurations:** The introduction of new cells or the removal of existing cells should not cause reconfiguration of other cells. In case reconfiguration is inevitable the number of reconfigurations should be minimized.

- **Adaptability to different network deployments:** There is no one-fits-all assignment scheme. The used scheme must be adaptable to take the actual context of the new cell and the surrounding network into account.
- **Applicability to initial and evolutionary deployment scenarios:** There should be no major difference if PCIs are assigned to a newly deployed network or to new cells during the network evolution of an operational network. The used approach should be the same or at least be based on the same principles in order to give similar results.
- **Useable in the presence of additional constraints:** A PCI assignment scheme has to have the possibility to support operator policy constraints but also other constraints like NID mod 6, NID mod 30 as shown in Section 11.1.

11.2.2 Assignment of Functionality to SON-Function Building Blocks

The PCI assignment SON-Function is basically a self-configuration function, which needs to be executed for the initial network rollout and whenever new cells are introcuded in the network. In addition it can also be used as a self-optimization function either if a PCI collision or confusion is detected in the network or to perform an optimization of the assignment on a larger scale for example for a sub-network or even the complete network. Such optimizations are performed to improve an assignment that has been reached as a result of the evolutionary growth of the network, either to reduce the number of assigned PCIs or to make the assignment more robust. If the assignment is robust, the insertion of new cells will cause no or only very few reconfigurations.

Due to the characteristics of the function, the tasks assigned to the Monitoring-part are rather simple. It is an on-demand function. The triggering situation is reduced to explicit requests. The function execution is requested whenever potentially new physical neighborships between cells are introduced or the assignment should be optimized. Whenever the PCI function is executed as part of the self-configuration process, an explicit notification is received by the Monitoring-part. In addition to that it will monitor the network to detect configuration changes that affect the cell sizes and therefore potentially create new neighborships and cause confusions or collisions. An additional analysis that tries to detect erroneous PCI configurations based on observations like an accumulation of failed handover attempts can either be integrated into the Monitoring-part or implemented as a separate SON-Function.

The Algorithm-part has to compute a collision and confusion free assignment that additionally considers all additional requirements. Depending on the situation the computation is done, either for a single newly inserted

cell, for a single or a set of cells that cause confusion or for a sub-network or the complete network for optimization reasons. In any case the requirements are the same but the way to reach the goal can be different which allows variations in the algorithm. In case the PCI assignment has been triggered as a self-configuration function an anlysis of the IDs of the neighboring cells to check for confusions has to be performed. In case confusions are detected a new PCI assignment for the cells that cause the confusion is performed. An important requirement, especially if cells in an operational network are reconfigured is to limit the number or reconfigurations to an absolute minimum to restrict the negative effects that are caused by a PCI reconfiguration.

Enforcing the newly computed PCI is subsequently performed in the Action-part of the SON-Function. As shown in the previous paragraph the reconfigurations can target single cells as well as sets of neighboring cells or complete sub-networks or networks. The Action-part of the PCI function has to assure a close to simultaneous enforcement of all new or changed PCIs in order to reduce the negative impact on the network operation.

11.2.3 PCI Assignment Algorithm

All basic requirements for the PCI assignment algorithm have been presented in the previous Section. For a SON-Function algorithm that is executed without further human control reliable operation with predictable output is mandatory. In the best case the validity of the results can be formally proved. In order to provide a reliable PCI assignment algorithm that is useable for both, initial ID assignment in a non-operational network and ID assignment during an evolutionary growth of an operational network the underlying problem has been mapped to a problem with known properties and solution approaches.

11.2.3.1 Mapping of PCI Assignment to Graph Coloring Problem

The graph coloring problem is a well understood problem from the world of mathemathics and computer science [JT94] which already has been used for frequency planning in GSM networks [Eis03]. Although the problem of finding the minimal set of colors for a given non-planar graph is known to be NP-complete there are multiple solutions that deliver close to optimal results with a justifiable complexity. The main idea of the approach to map the given problem of PCI assignment to graph coloring is to use the available algorithms and assess the characteristics of the PCI assignment even before it is done. It allows to provide a worst case assumption about the required number of IDs and much more important whether an assignment is possible or not with the available set of PCIs.

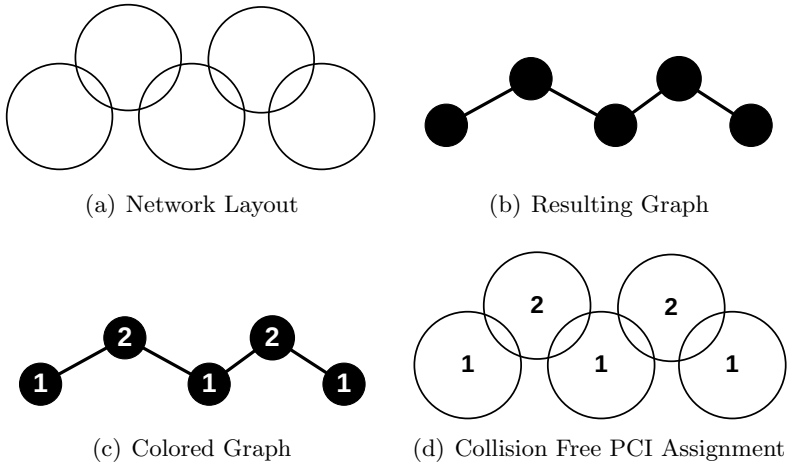
When looking at the task descriptions of PCI assignment and graph coloring there is a very high similarity. Assigning colors to a graph in a way that there are no nodes with the same color that are connected by an edge and assigning IDs to a network of cells in a collision and confusion free way.

The task to be performed is to translate requirements like collision and confusion free to configurations for a graph and then select coloring solutions for the different application scenarios.

11.2.3.1.1 Collsion Free Providing a mapping between the cell layout of a network and a graph to satisfy the collision free requirement is very straightforward. The steps are shown in Figure 11.4. Collision free is equal to the basic definition of the graph coloring problem. Two neighbors may not have the same color or ID respectively. As a first step the given cell layout with its neighbors has to be mapped to an appropriate graph. Figure 11.4(a) shows an example of a network with five cells. Cells with an overlapping coverage area are considered to be neighbors. The considered neighborships of the cells are assessed from the coverage area assessment in network planning. In an operational network these estimated neighborships can be improved through different methods for example by evaluating the results of SON-Functions like the ANR function, or performed drive tests. Cells are depicted as nodes in a graph, and in case they are neighbors in the network deployment they are connected by an edge. The resulting graph is shown in Figure 11.4(b). If a generic coloring algorihtm like the 'Greedy Algorithm' proposed by Welsh and Powell [WP67] is applied to the graph an assignment as shown in Figure 11.4(c) is returned. For simplicity reasons the colors are denoted by numbers. If this coloring is, in a last step, mapped back to the network deployment and the colors are translated to PCIs a collision free assignment has been reached (cf. Figure 11.4(d)). Since the graph coloring algorithms are trying to reach a minimal number of used colors, the PCI assignment shows a very high re-use rate, where the same IDs are re-used with the minimal possible distance and which is collision but not confusion free.

The mapping function that maps the colors in the graph to PCIs can consider additional operator requirements as for example a restricted availability of PCIs as long as it consistently maps a color to the same PCI.

11.2.3.1.2 Confusion Free To reach a confusion free PCI assignment without changing the applied algorithm, the graph has to be adapted in a way that the assignment becomes confusion free. Based on the definition that in a confusion free assignment no cell has two neighbors with identical PCIs additional edges are added to the graph to reflect the requirement on the level of direct neighbors. Therefore edges from each node to all neighboring nodes of his neighbors are added. The resulting graph (cf. Figure 11.5(b)



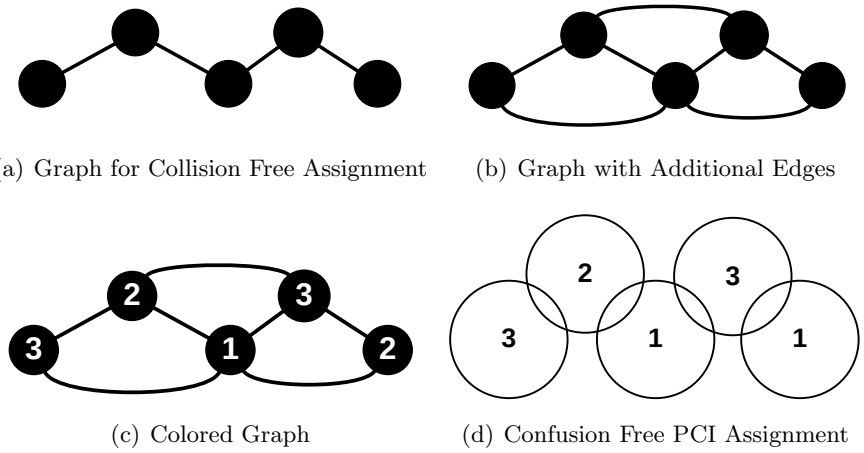
(a) Network Layout (b) Resulting Graph

(c) Colored Graph (d) Collision Free PCI Assignment

Fig. 11.4: Network Layout to Graph Mapping for Collision Free PCI Assignment

is colored with the same algorithm as before. Due to the additional edges more colors are required. If the colors of this graph (Figure 11.5(c)) are translated to a PCI assignment for the cell layout it is not only collision but also confusion free (Figure 11.5(d)).

11.2.3.1.3 Increased Robustness through Safety-margins The PCI assignment that is reached when the graph introduced in Section 11.2.3.1.2 is combined with a generic graph coloring algorithm is very dense as shown in Figure 11.5(d). A very high re-use rate is used for the distribution of the PCIs. In an optimal and completely stable network this is the best assignment as it preserves as many PCIs as possible. In an optimal network the cells have exactly the sizes as assessed in the network planning phase, yet such a deployment is rather unrealistic. Unplannable radio propagation will always establish unplanned neighborships, an example from real-world networks are so called overshooting cells, which are cells that can be received in areas where they are not expected to be receivable. This happens often due to changes in the radio propagation caused by unforeseen constructions and buildings, or change of attenuation due to humidity, or the change of seasons which has a major effect on the radio propagation [CSW02]. There are also major differences between the precision of the network planning between urban an rural areas. While in rural areas with rather larger cells, planning and reality are mostly consistent this is not true in urban areas with small cells with a typical ISD of around 1000m. With the introduction of HetNets the ISD within urban areas will even become much smaller. MNOs have a strong interest to provide continuous coverage to their users especially in urban areas even if individual cells fail, or are turned of due to maintenance, or for power saving. In areas with a very dense deployment



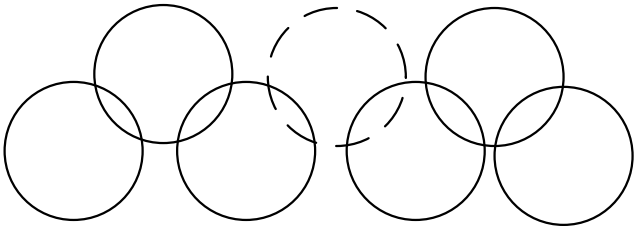
(a) Graph for Collision Free Assignment (b) Graph with Additional Edges

(c) Colored Graph (d) Confusion Free PCI Assignment

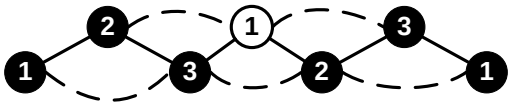
Fig. 11.5: Network Layout to Graph Mapping for Confusion Free PCI Assignment

COC is used to increase the coverage area of surrounding cells to cover the area of a cell that is currently not available. This functionality also increases the risk of the introduction of unplanned neighborships as shown in Figure 11.6. Figure 11.6(a) shows a cell layout, its graph representation and PCI assignment in Figure 11.6(d). For the layout the PCI assignment is collision and confusion free. The cell that is going to fail is represented by a dashed circle in the cell layout and a non-filled circle in the graph representation. In the graph edges that represent direct neighborships in the cell layout have solid lines and second degree neighborships are indicated through dashed lines. After the cell failure, COC adapts the cell sizes of the neighboring cells to resolve the coverage hole, which obviously changes the neighborships in the network as shown in Figure 11.6(c) and immediately causes a confusion. Figure 11.6(d) shows the graph after the COC. Two cells are confused (highlighted by arrows), which is directly visible from the graph. There are two node pairs with identical PCIs which are connected by a dashed line. If two or more nodes, called the confusing nodes, or cells that are second degree neighbors in the network have an identical PCI they cause a confusion. The confused cells are those who are direct neighbors to the confusing cells. In the graph those cells are confused which are connected with solid lines to both nodes of a confusing node pair. This confusion which was introduced by COC requires additional reconfigurations which has an additional negative impact on the network performance.

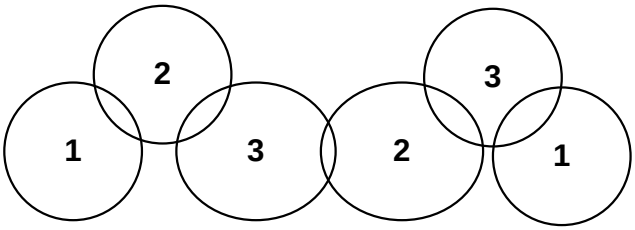
As shown, a very dense PCI assignment is a disadvantage in networks with potentially many unplanned neighborships or strong neighborship fluctuations due to the fast evolution of the network with a constant introduction of new cells to provide sufficient coverage and capacitity to the users.



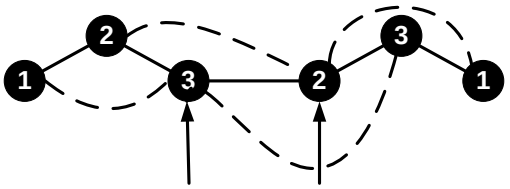
(a) Cell Layout



(b) Graph Representation and PCI Assignment



(c) Cell Layout and PCI Assignment after COC



(d) Graph Representation and PCI Assignment after COC

Fig. 11.6: PCI Confusion caused by Cell-Outage Compensation

If there are neighborships which have not been considered during the assignment, the risk for a collision or confusion is very high, therefore a more robust assignment needs to be chosen.

A PCI assignment is called **robust** if changes of the neighborships do not mandate the reconfiguration of the PCI assignment. Maximal robustness is reached if each cell in a network can be assigned an individual PCI, since this is not possible an alternative solution has to be found. In order to keep the possibility to use the underlying graph coloring approach changes to the graph have to be made that lead to the intented results. The re-use rate can be influenced if cells which are not direct or second level neighbors in the network deployment are made direct neighbors in the graph. Basically it is the same principle that was used to assure a confusion free PCI assignment. Which neighborships are added within the graph is controlled by a configurable paramter called the **Safety-margin** .

The Safety-margin is configured as follows:

- **Safety-margin 0:** No neighborships are represented in the graph. Each cell in the network deployment results in a single graph with a single node.
- **Safety-margin 1:** Direct neighbors in the cell layout become direct neighbors in the graph. This configuration will result in a collision free PCI assignment.
- **Safety-margin 2:** Cells which are neighbors of neighbors in the network are direct neighbors in the graphs. As shown in Section 11.2.3.1.2 the resulting PCI assignment is then collision and confusion free.
- **Safety-margin >2:** Neighboring cells of the n^{th} level become direct neighbors in the graph. This increases the robustness of the PCI assignment but also the risk of PCI depletion either because of a too high Safety-margin in a very dense network or due to the very limited number of available PCIs.

The definition of the Safety-margin has been chosen that way to make the graph based approach useable also for other types configuration parameters as for example those described in [3GP09a].

In order to examine the positive effects of the Safety-margin, the graph used in Figure 11.6 has been extended to represent the nework with a Safety-margin of three. The results are shown in Figure 11.7. As a first step the previously used graph (Figure 11.7(a)) is extended with edges between the nodes representing the 3^{rd} degree neighbors in the network deployment (Figure 11.6(a)). To make Figure 11.7(b) more legible all edges for Safety-margin (SM) 2 are greyed out and the additional edges for SM 3 are shown as dashed lines. Due to the additional edges the PCI assignment changes and more PCIs are used, the assignment is given in Figure 11.7(c). The effect of a

cell failure is shown in Figure 11.7(d). As in the previous example (Figure 11.6(a)) the cell visualized as a ring fails and the failure is compensated using the COC SON-Function. Previously two cell were confused after the COC which is now no longer the case. Due to the higher Safety-margin, this neighborship was already considered by the initial PCI assignment and therefore no PCI reconfiguration is required. The resulting PCI assignment does not longer fully comply to the required SM 3 which means that in case of an additional cell failure confusions need to be removed. A PCI assignment with SM 3 compensates the failure of a single neighboring cell. If one of the cells that have become direct neighbors fails and if the failure is compensated in a way that introduces new direct neighborships between cells that were neighbors of the failed cell a confusion is created.

MNOs can choose an appropriate SM depending on the required robustness of the PCI assignment and the number of available PCIs. If the SM is increased by one degree one additional cell failure can be tolerated without the danger of an introduced confusion.

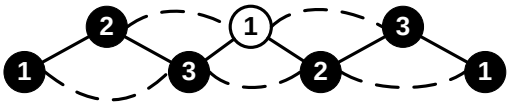
This additional robustness is also beneficial in future LTE HetNet deployments, where a cell deployment of small high capacity cells is complemented with an additional layer through a large macro cell. Such a macro cell covers the coverage area of multiple small cells and can easily be confused by the cells of the underlying pico, micro or femto layer. A PCI assignment with a higher SM reduces the number of required PCI reconfigurations required to restore a confusion free assignment.

11.2.3.2 Adaptation of Algorithm to Evolutionary Network Growth

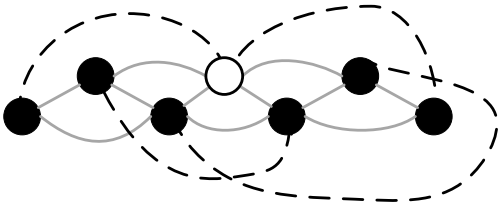
During the evolutionary growth of an operational network individual cells are added to enhance the network. Either to provide additional coverage or network capacity. A collision and confusion free assignment of a PCI has to be performed for each newly deployed cell. The application of the presented approach is not reasonable, as it always operates on the complete network. The introduction of single cells could therefore cause PCI reconfigurations for multiple cells which has a negative impact on the network operation.

For PCI assignment to new cells during the evolutionary growth of a network a variation of the presented approach without the potentially large overhead is required, which restricts possible changes to small parts of the network arround the newly introduced cell.

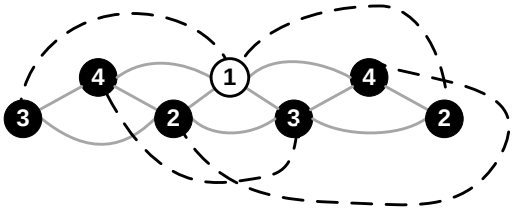
An iterative process is defined for the assignment of individual PCIs that operates on limited parts of the network and therefore only small partitions of the graph, while still retaining the properties of the graph coloring based approach that performs PCI assignment for the complete network, for example the minimal usage of colors respectively PCIs.



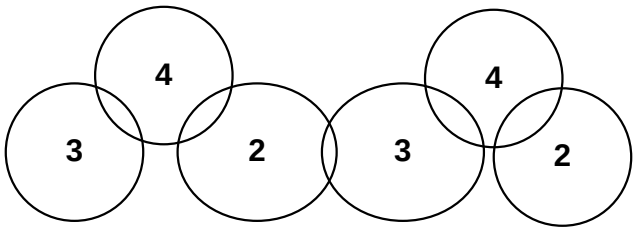
(a) PCI Assignment with Safety-margin 2



(b) Additional Edges(Dashed) for Safety-margin 3



(c) PCI Assignment with Safety-margin 3



(d) PCI Assignment after COC with Safety-margin 3

Fig. 11.7: PCI Assignment and COC with Safety-margin 3

The basic requirement for this approach is that the initial PCI assignment has been performed with the graph coloring based approach. For the assignment of a PCI to a newly introduced cell during the growth of a network only a subpart of the network is considered. When a new cell is introduced into the network it is also added to the graph representation of the network as a new node. Edges are added according to the required Safety-margin. For the following example only confusion and collision free assignment is required thus a Safety-margin of two is used. The subgraph of the network graph that is considered to be able to perform a proper PCI assignment consists therefore only of the new cell, its direct neighbors, the neighbors of its neighbors and the third degree neighbors in the cell layout. All operations are based on the respective sub-graphs, therefore, in the following the term neighbor denotes a neighboring node in the graph. In case a larger Safety-margin is required the additional cells have to be considered accordingly.

For the further understanding the used PCIs are grouped into ID Sets, where the contained IDs are ordered in an increasing order.

The following ID Sets are defined:

- **NSet** : The PCIs assigned to the direct neighbors of the new cell in the graph (direct and second degree neighbors in the cell layout)
- **NNNSet** : The PCIs assigned to the third degree neighbors of the new cell in the cell layout.
- $\mathbf{IDSet_u}$: All assigned PCIs in the network
- $\mathbf{IDSet_a}$: All PCIs that are available for an assignment

To retain a minimal ID usage, the new PCI should ideally be chosen from the $NNNSet$. If this is not possible and $\mathbf{IDSet_u}$ is not equal $\mathbf{IDSet_a}$ then the lowest PCI from $\mathbf{IDSet_a}$ that has not been used in $NSet$ or $NNNSet$ is assigned.

The following steps are performed for the PCI assignment:

1. Check if there are PCIs in the $NNNSet$ that have not been used in the $NSet$ of the new cell. $NNNSet \setminus NSet \neq \emptyset$
   (a) If the result set is nonempty choose the first PCIs from the result set. END
   (b) If the result set is empty perform next step.
2. Check if the $IDSet_u$ contains IDs that have not been used in the $NSet$ or $NNNSet$ of the new cell $IDSet_u \setminus \{NSet \cup NNNSet\} \neq \emptyset$
   (a) If the result set is nonempty select the first PCI from the result set. END
   (b) If the result set is empty, continue
3. If $IDSet_a$ and $IDset_u$ are identical no PCI can be assigned. An initial assignment for the complete network is triggered.
   (a) Otherwise introduce a new PCI by choosing the first PCI from $IDSet_a \setminus IDSet_u$

Through the composition of the $IDSet_a$ the operator can influence which PCIs are assigned in the network, similar to the configuration of the mapping function for the network wide assignment. For example by only adding a subset of all possible PCIs to $IDSet_a$.

11.2.3.2.1 Confusion Resolution The PCI assignment that is reached by following the algorithm is guaranteed to be collision free and not confusing any of the neighboring cells, but the newly introduced cell can already be confused as described in Section 11.1.1.3 and shown in Figure 11.3. Therefore, after each assignment of a new PCI an evaluation of the neighboring cells that could cause a confusion is required and in case of a confusion re-configurations of the confusing cells have to be performed.

The cells which have to be considered for the detection of a confusion is dependent on the Safety-margin required for the PCI assignment. Basically, in a network which should only be configured collision and confusion free (Safety-margin 2) without additional protection, only cells that are direct neighbors in the cell layout can cause a confusion. This changes in case a higher Safety-margin has been chosen for a more robust assignment that can cope with unplanned neighborships, for example due to potentially overshooting cells, then the range of cells that have to be evaluated also has to be extended.

When looking at the graphs representing the network, the sub-graph that can be used to identify the cells that can cause a confusion is the graph that represents a configuration of a $Safety - margin - 1$. In this graph the PCIs of all cells that are represented by nodes that are connected with an edge to the node of the new cell have to be evaluated. Only if all of these cells have unique PCIs the assignment is confusion free. In case of a Safety-margin of two, it is the graph for Safety-margin of one, or only the direct neighbors in the cell layout. If a Safety-margin larger than two is required the same evaluation is performed for the cells identified by $Safety - margin - 1$ in order to guarantee an assignment with the required Safety-margin.

In the example in Figure 11.8(d) the cells that have to be evaluated for a potential confusion of the new cell, represented by the non-filled ring, are those that are connected by the dashed edge. In this example those cells cause no confusion of the new cell.

In case a confusion of the new cell is detected it is resolved through a simple mechanism. New PCI values are sequentially assigned to all but one of the cells that cause the confusion. For the reassignment the cells are treated as if they were newly introduced to the network and the identical PCI assignment algorithm is applied. If the previous PCI assignment has been performed using the graph based approach it is already guaranteed that these cells are not confused by their neighboring cells.

11.2.3.2.2 Example of PCI Assignment for a newly introduced Cell Figure 11.8 shows an example of the PCI assignment to a newly introduced cell during the evolutionary growth of a network. The initial cell layout and the corresponding graph and PCI assignment are shown in the Figures 11.8(a) and 11.8(c). The assignment is only confusion and collision free without an additional Safety-margin. The position of the new cell is shown in Figure 11.8(b) by the dashed circle. The position of the new cell is also reflected in the correspondig graph representation of the network (Figure 11.8(d) as a non-filled circle. The dashed edges connect the new node to the nodes that represent the direct neighbors in the cell layout. Second degree neighbors are connected by the dotted edges.

After the introduction of the new node into the graph the PCI assignment algorithm for evolutionary growth is performed.

Definition of ID Sets:

- $NSet = \{1, 2, 3\}$
- $NNNSet = \{2\}$
- $IDSet_a = \{1, 2, .., 503\}$
- $IDSet_u = \{1, 2, 3\}$

Selection of a PCI for the new cell:

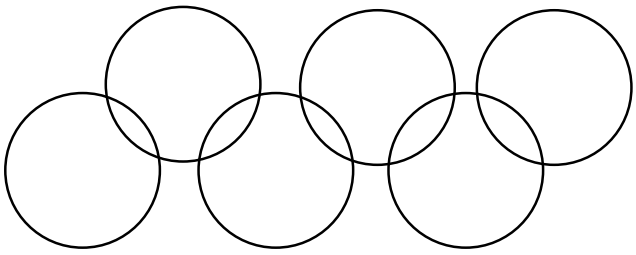
- $NNNSet \setminus NSet = \emptyset \Rightarrow$ Continue
- $IDSet_u \setminus NSet = \emptyset \Rightarrow$ Continue
- $IDSet_a \setminus NSet \neq \emptyset$
  $\Rightarrow$ Select $PCI \in \{IDSet_a \setminus IDSet_u\} = \{4, 5, ..., 503\}$
- Assign new PCI = 4

Confusion dection and resolution:

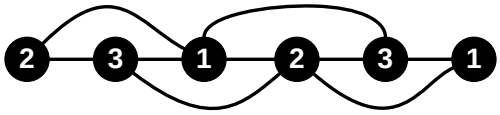
After the new PCI has been assigned the PCIs of the direct neighboring cells have to be examined in order to detect a potential confusion of the new cell. The new cell would be confused if the nodes connected by dashed edges in the graph (cf. Figure 11.8(e)) had identical PCIs. In the example the direct neighboring cells had been assigned different PCIs and the new cell is therefore not confused.

11.2.3.3 Evaluation of the PCI Assignment Scheme

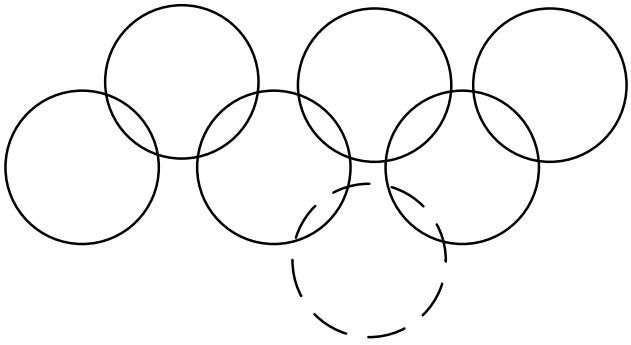
In order to validate the theoretical approach the algorithms for initial PCI assignment and an assignment for evolutionary growth of the network were implemented and tested against multiple network setups, both realistic and artificial. The evaluation is subdivided into two parts. First the analysis of



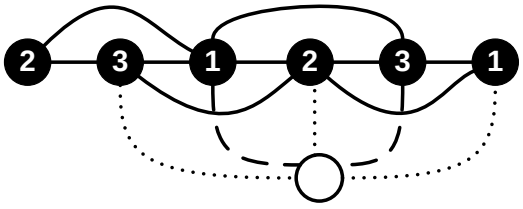
(a) Initial Cell Layout



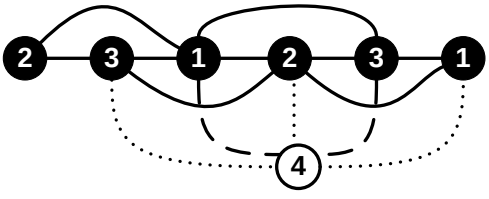
(b) Graph with PCI Assignment for initial Cell Layout



(c) New Cell Layout with additional Cell



(d) Extended Graph for new Cell Layout



(e) PCI Assignment for new Cell Layout

Fig. 11.8: PCI Assignment during evolutionary Network Growth

the basic approach and its applicability within a network. Second the analysis of the effects on the number of required PCIs caused by the introduction of Safety-margins of different degree.

11.2.3.3.1 Used Datasets Three data sets were used for the evaluation of the effectiveness of the proposed PCI assignment algorithm.

The first data set, is based on a layout of hexagonal cells. Each cell has six hexagonal neighboring cells of the same size. This very large network with a total of 10981 cells is particularly suited for the evaluation of the effects of PCI assignments with very large Safety-margins. The network has a ring-like structure. Starting with a single cell, which is surrounded by six cells. The next ring consists of 12 cells. In total the network consists of the center cell and 60 additional cell rings.

A second data set provides a semi-artificial macro-only scenario. It is based on the geo positions of 229 of Vodafone Germanys' 3G sites for parts of eastern Germany especially in the greater area of Berlin. This information is available as a kmz-file[2] from [BSL]. For the evaluation each site was treated as a single cell. The available information was extended with an estimated but realistic cell radius according to their surrounding area. For example a radius between 1000 and 3500 meters for urban areas like the greater area of Berlin and cell radiuses up to 6000 m for rural areas. The cells were approximated by circles.

The cell radius is important as it is used together with the geographical position of the sites to compute coverage areas and from that neighbors in the network. These extension to the data results in a total of 23 cell clusters. A cell cluster is a set of cells which provide contiguous coverage. The largest cluster consist of 87 cells and an average cluster size is 10 cells. This very low average cluster size results from clusters that consist of only a single or very few cells in rural areas, for example to cover only small villages or a theme park.

The third data set is a realistic HetNet scenario consisting of a total of 336 cells. The macro layer consists of 236 cells which is extended by a pico layer with 100 omnidirectional pico cells. The adjacencies between the cells have been determined in a simulation using a set of basic radio parameters and the Okumura-Hata pathloss model [SBS12]. The characteristics of this scenario correspond to what is expected for a HetNet deployment in a major European city.

[2] Zipped Keyhole Markup Language Format

11.2.3.3.2 Evaluation of the Graph Coloring based Approach of PCI Assignment The basic evaluation of the graph coloring based approach targets the following questions:

- How many PCIs are used for a confusion and collision free assignment?
- Is a proper assignment still possible with an increasing SM

To answer these questions, several rounds of PCI assignment have been performed with an increasing Safety-margin. Starting from one to a maximal Safety-margin of 12. Table 11.1 provides an overview on the results for the examined scenarios. For each scenario it shows the basic information about the number of cells and the number of cells in the largest cluster. The per cluster information is only relevant for semi-artifical network scenario, since it is the only one that contains multiple cell clusters. For each of the scenarios, the number of required PCIs for assignments with different SM requirements is given, starting with SM2 which guarantees collision and confusion free PCI assignment. The results for SM3 and SM4 shows the impact of the selected SM in dependency to the deployment scenario.

For the hexagonal network layout each cell has six neighbors. A collision and confusion free PCI assignment required in theory a minimal number of eight different PCIs, which is independent from the number of cells in the cluster. Using the greedy algorithm proposed in [WP67] this minimal number is not always reached, especially if a large number of nodes in the graph have the same degree. For a close to optimal assignment additional requirements would be needed to identify which of those nodes with an identical number of neighbors should be used as a starting point. Since there is no such criteria, the algorithm requires eleven different PCIs for a collision and confusion free assignment. With an increasing SM the number of required PCIs increases only slowly. For this network it was possible to compute an assignment with SM 12 without ending a with a full-mesh graph. For a PCI assignment with SM 12 only 171 PCIs are required.

Tab. 11.1: Results of PCI Assignment

Scenario	Number of Cells	Max Cells in Cluster	Number of PCIs used		
			SM2	SM3	SM4
Hexagon	145861	145861	11	19	28
Semi-Artificial	229	87	15	20	24
HetNet	336	336	112	314	335
HetNet Pico Only	100	42	9	14	18
HetNet Macro Only	236	236	41	114	236

The hexagonal cell layout is rather unrealistic, therefore it is interesting to see how the numbers change when a more realistic setup is used. In the semi-artificial network layout the network is combined from several cell clusters. The largest cluster consists of 87 cells. When looking at the required PCIs it becomes obvious that the network layout has a big influence on the number of required PCIs. The higher number of neighbors in a part of a network leads to a higher number of assigned PCIs.

This effect becomes even more visible in the HetNet scenario. In order to provide a valid, collision and confusion free assignment for the 336 macro and pico cells already 112 PCIs are required. The reason is that a single macro cell covers a large number of pico cells, and are additionally neighbors to other macro cells. This causes a very larger number of neighbors in the neighborship graph that is used as input to compute the PCI assignment. When increasing the SM, the graph results very fast into a full mesh where are cells are neighbors in the graph. Already with SM 3 almost as many PCIs as cells are required for proper assignment. The conclusion from this result is that for a HetNet scenario it does not make sense to treat macro and pico cells equally. One proposal to circumvent the situation that there are not enough PCIs to assign them properly in a dense urban scenario is to use separate ID ranges for the different cell layers. To have one set only for macro and another set of IDs only for the pico cells. To show the benefits of such a range situation, the HetNet scenario has been split into a macro and a pico only network.

When considering only the macro layer, the network consists of 236 cells and for a proper assignment only 41 PCIs are required. The pico layer does not provide contiguous coverage, the 100 cells are distributed to seven cell clusters. The largest cluster consists of only 42 cells, so that for a collision and confusion free assignment only nine PCIs are required and even for an assignment with SM 4 the number increases only to 18.

Looking at the number of required PCIs even in total the range separated approach requires less PCIs compared to the assignment that treats macro and pico cells equally. This trend is even stronger when looking at the assignment for higher SM. Only for SM 4 there is a strong increase of required PCIs in the macro layer. The macro deployment is still pretty dense, therefore the assignment with SM 4 is already based on a full mesh graph and requires as many PCIs as there are cells.

In total the evaluation shows that the proposed PCI assignment approach satisfies the requirements for the usage within a SON-Function. Through the theoretic foundation in graph coloring, the results are highly deterministic and a close to optimal assignment can be reached by using well known algorithms with proven characteristics. The graph coloring basis also guarantees the minimal usage of PCIs even if larger SMs are used to make the assignment more robust against unexpected radio propagation behavior and changes in the network layout that might be caused by network evolution

or failures. For very dense network layouts, for example, in future urban HetNets additional means have to be used to prevent situations where more than available PCIs are required for an assignment with the requested characteristics. When using, for example, range separation the operator can even choose to impose different SM requirements to different cell layers. A higher SM is then chosen for the more dynamic pico layer and a lower SM for the macro layer, where even the failure or the addition of new cells do not have such a strong impact on the neighborships.

11.3 Conflict Analysis for the PCI SON-Function

For the design of the decision logic for the PCI SON-Function a thorough conflict analysis with other deployed SON-Functions has to be performed. Six SON-Functions have been chosen to show an example of a conflict analysis. The conflict analysis follows the conflict classification as it has been described in Section 5 in order to identify all potentially conflicting SON-Functions.

11.3.1 Considered SON-Functions

For the conflict analysis PCI, COC, ANR, MRO, CCO(RET) and CCO(TXP) serve as SON-Function examples. The following listing provides a short description of the functions further information about the SON-Functions can be found in [Leh10, Leh07a]. The description of the PCI SON-Function is not given as it has been extensively discussed.

- **COC:** Cell Outage Compensation tries to compensate loss of coverage and capacity caused by a failed cell. Usually it enforces an enlargement of the neighboring cells to reach its targets.
- **ANR:** The Automatic Neighborship Relation SON-Function triggers UEs to provide information about receivable cell in its vicinity to update the NRT for the serving cell. Neighboring cells are identified by their PCI.
- **MRO:** Mobility Robustness Optimization evaluates the handover performance in the network. In case it detects a sub-optimal behavior the SON-Function will try to optimize parameter settings that influence the handover behavior in the network. For example adapting the handover hysteresis or the Time-to-Trigger.
- **CCO(RET) and CCO(TXP)**: Two individual SON-Functions are responsible for adaptation of the cell layout in order to optimize the overall coverage and capacity of the network. CCO(RET) changes the antenna tilt in order to increase or decrease the coverage area of a

cell. CCO(TXP) reaches similar results through the adaptation of the antenna transmission power.

11.3.2 Conflict Analysis

All available SON-Functions are examined whether there is a parameter, measurement or characteristic conflict between them and the PCI SON-Function. Each conflict has to be appropriately considered within the decision logic.

11.3.2.1 Configuration Conflicts

A configuration conflict obviously exists between two instances of the PCI function. Each instance has the ability to change the PCI of a cell, which could cause both an output conflict on the target cell whenever both functions try to modify the PCI of the same cell and also an input conflict as the function requires run-time stability for all cells which are direct neighbors in the network graph.

A configuration conflicts exists also with the ANR function, which identifies potential neighbors via the PCIs reported by the UEs. In how far this conflict is relevant for the coordination of a PCI function execution request has to be determined when designing the decision logic.

11.3.2.2 Measurement Conflicts

The PCI function does not operate on performance measurements, therefore there is no measurement conflict from other SON-Functions towards the PCI function. In the other direction, the PCI function does not change configuration parameters which show their effect over time in performance measurements that are collected. The change of the PCI is instantly visible. In case the PCI is used to identify a counter or a statistic, the identificator can then easily be updated but the change itself does not affect the measurements.

11.3.2.3 Characteristics Conflicts

There is a constant danger of an invalidation of the PCI assignment whenever the cell layout changes in a way that it potentially introduces new neighborships. Therefore all SON-Functions that can affect the cell layout are potentially conflicting with the PCI function. Especially the CCO functions are in a characteristics conflict with the PCI function. Through the adaptations that are performed by the CCO functions new neighborships can be created, which need to be considered when assigning a new collision and confusion free PCI to a cell.

Tab. 11.2: SON-Functions and their conflicts with the PCI Function

SON-Function	Configuration Conflict	Measurement Conflict	Characteristics Conflict
ANR	x	-	-
CCO(TXP)	-	-	x
CCO(RET)	-	-	x
COC	-	-	x
MRO	-	-	-
PCI	x	-	-

Coverage Outage Compensation is also a function that can easily create new neighborships as shown in Section 11.2.3.1.3 therefore it has to be considered as a characteristics conflict in the decision logic.

11.3.2.4 Conflict Overview

Table 11.2 gives an overview on the SON-Functions and their conflicts with the PCI function. The conflict types are shown in the columns the SON-Functions in rows. In case a SON-Function has a conflict of a given type it is marked with an "x" in the respective column. If not it is marked with a "-".

11.4 PCI Assignment Impact-area and Impact-time

11.4.1 Abstract Specification of the Impact-area

The sound definition of the Impact-area is an important step for a successful coordination of SON-Function instances. Chapter 6.1 names four building blocks for the Impact-area of a SON-Function. A specification of the Impact-area can be performed by specifying the building blocks and subsequently unifying those parts.

For the initial definition of the abstract Impact-area for the PCI SON-Function no additional Safety-margin for the assignment is assumed, but only the SON-Function that performs a PCI assignment with a maximal re-use of PCIs.

The Graph $G = \{V, E\}$ that is used for the specification of the Impact-area is the cell-neigborship graph without any additions. Nodes V in the graph represent the cells in the network and the edges E represent a neigborship between two cells. If two nodes are connected by an edge, the respective cells are neighbors in the network deployment.

The parts of the Impact-area are defined as follows:

- **Function-area:** The PCI SON-Function for the evolutionary growth of a network is always triggered for a single cell. Therefore, the

Function-area consists only of a single cell. Referenced by the node a with $a \in V with \quad G = \{V, E\}$

- **Input-area:** In order to compute a PCI for the new cell that neither collides with the PCIs of the neighboring cells nor causes a confusion for the neighboring cells the function needs to access the PCI configuration of the first and second degree neighbors. This input provides the information on which PCIs may not be configured to the new cell. For a minimal usage of PCIs the new PCI should be selected from those that are configured at the third degree neighbors. This fact increases the Input-area to the third degree neighbors, represented by Equation 11.1. The function $dist(x, y)$ is the distance function for nodes in the graph. With two nodes in the graph as input it returns the distance between those nodes. That means the minimal number of edges between the two nodes. If $x == y$ the distance is 0 if they are direct neighbors the distance is 1.

$$G_{InpA} = \{V_{InpA}, E\} \quad V_{InpA} = \{x \in V | 0 < dist(a, x) <= 3\} \quad (11.1)$$

- **Effect-area:** A result of the conflict analysis in Section 11.3 is that a PCI change does not have any measurement conflicts, which are important for the specification of the Effect-area. The only conflict type that is relevant are configuration conflicts. This conflict type is relevant for other PCI SON-Function instances and instances of the ANR function. ANR functions can be affected either when they are running on direct neighboring cells or on second degree neighbors. For the first degree neighbors the problem is obvious. Therefore the Effect-area comprises the targeted cell itself and all first degree neighbors.

$$G_{EffA} = \{V_{EffA}, E\} \qquad V_{EffA} = \{x \in V | dist(a, x) <= 1\} \quad (11.2)$$

- **Safety-Margin:** Even if initially stated that for this example no additional Safety-margin is required, there is also a Safety-margin that needs to be considered, since a PCI function instance can also be relevant for ANR instances on second degree neighbors in case the targeted cell is confused and therefore a subsequent PCI change on direct neighboring cells is required. This introduces a Safety-margin that consists of all seconds degree neighbors of the affected cell.

$$G_{SM} = \{V_{SM}, E\} \qquad V_{SM} = \{x \in V | dist(a, x) = 2\} \qquad (11.3)$$

From the definitions of the building blocks of the Impact-area, shown in Equations 11.1, 11.2 and 11.3, the abstract Impact-area for the PCI function can be derived and is then defined as follows:

- **Graph:** $G = V, E$ is the Cell neighborship Graph without additional Safety-margin
- **Function-area:** Single cell denoted as a
- **Abstract Impact-area definition:**

$$G_{ImpA} = \{V_{ImpA}, E\} \qquad V_{ImpA} = \{x \in V | dist(a, x) <= 3\} \quad (11.4)$$

11.4.1.1 Impact-area Definition with additional Safety-margin

In case additional Safety-margins have to be considered for the PCI assignment, this has to be reflected in the abstract Impact-area specification. There are several ways to do this, two possibilities are discussed here.

11.4.1.1.1 Modified Equations: For a larger Safety-margin, the PCI assignment algorithm evaluates the PCIs of neighbors of a higher degree, for example instead of selecting from the third degree neighbors, a PCI is selected that is configured to one of the fourth degree neighbors but not to any of the lower degree neighbors. From this condition it becomes obvious, that the Input-area definition has to be adapted to reflect this requirement. Also the Safety-margin definition has to be extended to assure that required Safety-margin is enforced. With an increased Safety-margin a confusion like situation can be caused not only by the first degree but also by higher degree neighbors. In case of an assignment with a Safety-margin of three all first and second degree neighbors need to be evaluated and where required reconfigured.

For a Safety-margin of three, the equation that specifies the Impact-area is Equation 11.5.

$$G_{ImpA} = \{V_{ImpA}, E\} \qquad V_{ImpA} = \{x \in V | dist(a, x) <= 4\} \qquad (11.5)$$

11.4.1.1.2 Modified Graphs: Instead of reflecting the increased Safety-margin through the adaptation of the equations it is also possible to change the graphs that are used. For example, the neighborship graph that is used, depicts already second degree cell neighbors as direct neighbors in the graph. Then the Impact-area definition in Equation 11.4 can be applied. It is important to note that the resulting Impact-area is larger than only the Function-area increased by a single degree. The algorithm will not re-use PCIs that have been configured up to the fourth degree cell neighbors, and it will choose a PCI from one of the fifth or sixth degree cell neighbors. What makes this approach appealing, is that an operator could provide different cell-neighborship graphs for example for different cell types, and then use the same abstract Impact-area definition.

11.4.2 Specification of Impact-time

In the same way the Impact-area is built, the Impact-time consists of several components (cf. Section 6.2), which allow a fast definition of the overall Impact-time individually for each SON-Function.

- **Enforcement-time:** The Enforcement-time for the PCI function is dependent on the way how a SON-Function instance in the network can interact with the targeted NEs.
- **Visibility Delay:** The PCI is a configuration parameter that is instantaneously visible once the configuration is enforced, therefore no additional Visibility Delay is required.
- **Protection Time:** Since the algorithm to compute a new PCI configuration for a cell, does not consume any performance measurements, no Protection Time is required.
- **Relevance Time:** From the conflict anlysis in Section 11.3 no function could be identified that considers the acutal change of a PCI over a longer time period. Not even for the intra function coordination, the previous changes are relevant. The graph based foundation and the general characteristic of the PCI SON-Function excludes the possibility of oscillating reconfigurations therefore the previous PCI changes are not relevant for subsequent instances operating on the same cell.

In the end, the Impact-time, that is added to the Execution-time of the PCI-Function, consists of only a single building block, the Enforcement-time.

11.5 Coordination Logic and Coordination Scheme

For a succesful coordination, the coordination logic for each SON-Function has to be designed and an appropriate coordination scheme has to be chosen.

11.5.1 Coordination Logic for the PCI SON-Function

The coordination logic for the PCI SON-Function is based on the results of the conflict analysis in Section 11.3.

The coordination logic has to reflect and handle all potential conflicts with other SON-Functions that have been detected. For the PCI SON-Function it has to cover the conflicts with ANR, CCO(TXP), CCO(RET), COC and also with other PCI functions. Figure 11.9 shows a graphical representation of the decision logic as a decision tree. The root of the tree is the PCI request that should be coordinated. The first check is the evaluation of the context to determine whether there is an instance of a PCI or COC that conflicts with the requested PCI instance. In that case the request for

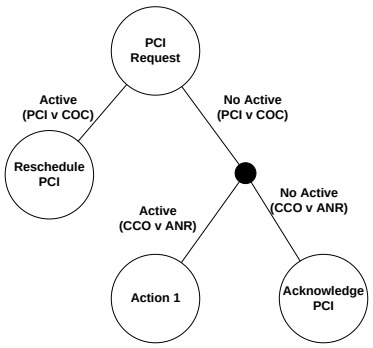


Fig. 11.9: Decision Tree for the PCI SON-Function

the new PCI instance is rescheduled. If there are no conflicting PCI or COC instances it has to be evaluated whether conflicting CCO or ANR instances are present. The action (Denoted as Action 1 in Figure 11.9) that has to be performed when there are conflicting CCO or ANR functions are:

1. **Terminate** the conflicting active CCO or ANR functions
2. **Reschedule** the terminated Function instances
3. **Acknowledge** the requested PCI function instance

If there is also no conflicting CCO or ANR function instance, the PCI SON-Function request is directly acknowledged and executed.

11.5.2 Coordination Scheme for the PCI SON-Function

The last question that has to be answered is which of the possible coordination schemes (cf. Section 7.2.2) should be applied for the PCI SON-Function. The answer can be directly derived from the coordination logic for the function. As shown in Figure 11.9 the coordination logic requires only information if there are conflicting SON-Functions. No information about performed configuration changes is required, therefore neither Action Coordination nor the Combined Coordination scheme comes into consideration. The only coordination scheme that makes sense is Algorithm Coordination, which is therefore chosen.

11.6 PCI SON-Function Summary and Findings

This chapter provided an example of the development of a SON-Function including all parts that are required to enable a reliable and efficient coordination of function instances. As a first step the SON use case was evaluated to determine the functionality that should be provided by the SON-Function. Based on this initial analysis a SON-Function was defined and the functionality was distributed to Monitoring-, Algorithm- and Action-part of the function. The requirements on a sound PCI assignment were used to define

a reliable and efficient algorithm that is based on well known graph coloring problems. For the phase of the evolutionary growth of the network, an adapted algorithm that operates on only a small set of cells was derived. All PCI assignment approaches where designed to support additional operator requirements as for example extended Safety-margins. An evaluation of the assignments produced by the PCI SON-Function showed that even in complex and large networks good results are reached.

The next steps provided the input that is required for the coordination of the PCI SON-Function to guarantee an efficient and conflict-free operation even if multiple potentially conflicting SON-Functions are deployed and active in the same network. At first a conflict analysis revealed all potentially conflicting SON-Functions and their conflict types with respect to the PCI function. In a next step the abstract Impact-area and Impact-time definitions where provided. From the available knowledge about the potentially conflicting SON-Functions and their conflicts the coordination logic was defined. As a last step, an appropriate coordination scheme was chosen, that introduces enough interaction points between the SON-Function instances and the coordination function to satisfy the coordination requirements.

This exemplary processing of a SON use case can serve as a template for the introduction of new SON-Functions. What has not been shown is the adaptation of the decision logic of other functions, the PCI SON-Function potentially is in conflict with. But the tasks that need to be performed are identical to the steps that have been performed for the PCI function itself. At first, a quick conflict analysis has to be performed to show, whether the new PCI function conflicts with existing SON-Functions. In case such an additional conflict is detected, it has to be considered in the coordination logic, which has to be extended accordingly. For the PCI function the additions to the coordination logic of other functions is only an existence check, whether a conflicting PCI function is currently active with a shared Impact-area. For the SON-Functions evaluated in this example, no changes of the Coordination Schemes would be required, since the actual change performed by the PCI function is not of interest and none of the other functions is coordinated using Action Coordination only.

11.6.1 Contributions to the Research Questions

This case study contributes to answering the research questions by proving that the developed concepts can actually be applied to a real-world use case.

R2 How can the network operator, despite the automation, still be in full control over the ongoing processes in the network?

There are several parts of the PCI assignment process, that the operator implicitly controls. First of all, there is the re-use distance of the used PCIs, which is controlled by the design-time definition of

the used Safety-margin. Second, the specification of which PCIs are actually available allows the operator to control the ID assignment. Another way of implicit control is the selection of a deterministic algorithm which is executed by the SON-Function. Through the selection of an appropriate algorithm, the network operator controls the results.

R3 How can the enforcement of and the compliance with all operational guidelines be assured?

The presented coordination logic shows how all operational guidelines for the assignment of PCI assignment are enforced. The information that is used to check the conditions is an important ingredient to assure the compliance with the operational guidelines. For example the definition of an appropriate Impact-area assures the spatial distribution of active PCI SON-Function instances.

Concerning the research area Efficiency, a contribution to the following research question has been made:

E5 How can the required context information be provided efficiently to the system controlling the SON-Function instance execution?

The process of abstractly specifying the Impact-area and then at run-time mapping it to a concrete spatial extension is a very good example on how information is efficiently provided. A specification based on mathematical operations allows is very efficient, since, in combination with appropriate input, it can be easily evaluated.

E6 How can the number of concurrently, conflict free executed SON-Functions be maximized?

The concept of maximizing the number of concurrently executed SON-Functions is based on the evaluation of their respective Impact-areas. For the assignment of PCIs the network operator has the ability to fine tune the size of the Impact-area according to the network characteristics. In a very dense urban network a larger Impact-area, thus a reduced number of concurrent PCI SON-Function instance executions, is required to prevent negative effects. In rural areas the Impact-area can be reduced to the absolute minimal size and allow more concurrent active SON-Function instances. This example shows that the concept of controlling the number of parallel SON-Function instances through the Impact-area is a very powerful and efficient concept.

Concerning the research area Flexibility, a contribution to the following research question has been made:

F1 Is there a uniform, future proof design scheme for SON-Functions?

The approach for a uniform design of SON-Functions presented in this thesis is the assignment of functionality to particular functional blocks to reach a well-defined SON-Function behavior on an abstract level. The PCI SON-Function is not the typical SON-Function as it does not process a lot of performance data to detect a triggering situation. But even this SON-Function can be designed according to the proposed design-scheme, which shows the applicability of the approach.

F2 How can the decision logic be provided in a easily adaptable way?

The provided decision trees proof the applicability of the developed concepts for having an easily adaptable decision logic. Whenever new conflicts with existing or additionally introduced SON-Functions would be detected an adaptation could easily be performed.

Part V

CONCLUSION

12. CONCLUSIONS AND FUTURE DIRECTIONS

This thesis introduced SON-Function instance execution coordination as an efficient approach for run-time SON-Function instance conflict resolution and prevention. This approach is not only capable of resolving or preventing conflicts between SON-Function instances but allows also to enforce operational guidelines to control the behavior of the SON-Function instances and the complete network. The behavior of the SON Coordinator can even be easily modified at run-time to adapt its behavior to the needs and requirements of the network operator. An evaluation of the coordination concept based on network simulation and first-generation SON-Functions proved the effectiveness of the presented approach. The case study showed how, based on the presented theoretical concepts, a SON-Function for a given SON use case can be developed.

12.1 Results

As an enabler for the development of the presented coordination concept, the requirements imposed on the SON-Functions by the SON use cases were analysed and the results formed the basis to develop a formal SON-Function architecture. In this architecture, the functionality of a SON-Function is assigned to its Monitoring-, Algorithm- and Action-parts. While this formal specification is intended to still be generic enough for being applied to all SON-Functions it provides also the structure that is required for further analysis of the SON-Functions and their functionality. It also provides the basic means to coordinate each SON-Function instance according to its requirements.

With this initial structure of a SON-Function and the available description of the SON use cases a categorization of SON-Function conflicts was performed. The categories describe how two SON-Functions are conflicting with each other, therefore each potential conflict can be assigned to a Configuration, Measurement or Characteristic conflict group or one of their respective sub-groups. If a conflict between two SON-Functions is categorized as a configuration conflict, both of the SON-Functions operate on a shared set of configuration parameters. In case of a measurement conflict, performance measurements are affected by one SON-Function which are used by the other as input to either its monitoring or Algorithm-part. Characteristics conflicts cover conflicts which are not directly based on a

shared input or output but on the modification of network characteristics, for example, the coverage area of a cell. The assignment of a particular conflict between to SON-Functions already provides an indication on how it can be resolved or prevented at run-time.

In order to propose a solution for conflict prevention and resolution with a much higher efficiency than SON-Function co-design or full SON-Function instance execution serialization an analysis of the operation of SON-Functions was performed.

The detailed analysis of the SON-Function characteristics and how SON-Functions are executed in large mobile communication networks revealed two important facts:

- It is impossible to provide only non-conflicting SON-Functions. The co-design of SON-Functions is an important tool to reduce the number of conflicts but it is impossible to remove all conflicts
- Most SON-Functions are executed with a rather small target scope. Often they affect only a single or a small set of NEs in a network with potentially several thousands of NEs

These results led to the introduction of the concept of a SON-Function instance. An instance of a SON-Function is a run-time instantiation of a particular SON-Function type with a limited target scope. Parallel execution of potentially conflicting SON-Function instances is possible as long as their Impact-areas do not intersect.

For the conflict prevention and solution not only the spatial scope of a SON-Function instance but also its temporal scope is relevant. A SON-Function instance can affect other SON-Function instances for some time interval after it has performed its configuration changes and therefore has to be considered for this time interval after its execution. In Addition a SON-Function instance can have requirements about the stability of configuration parameter settings, which are used, or affect the used performance measurement values for some time before its actual execution.

Thus, a SON-Function instance is characterized by its SON-Function type, the Impact-area, and its Impact-time. These proposed characteristics of a SON-Function instance are used by within 3GPP standardization processes to describe the behavior of SON-Functions in SON enabled networks.

This thesis presents a modular definition of both Impact-area and Impact-time. The parts of the Impact-time are specified based on the conflict types between two SON-Functions. For the definition of the Impact-time the conflict categorization can directly be used to determine which parts of the Impact-time are required. The length of the parts of the Impact-time are then defined according to the SON-Function and network specifics.

For the definition of the Impact-area the information about Function-, Input-, and Effect-area are considered together with a Safety-margin.

Conflict categorization and the Impact-area and Impact-time as charac-

teristics form the foundation of the SON-Function instance coordination as an approach for conflict prevention and resolution.

The introduced SON coordination approach makes use of the highly event based operation of mobile network operation and management, which is becoming prevalent through new possibilities caused the changes introduced in the networks, as for example the availability of real-time performance data. Coordination decisions are taken within the SON Coordinator with help of a policy based coordination logic. The ECA policies are triggered by events that correspond to the SON-Function instance execution. Therefore coordination decisions are taken at the transitions between the monitoring and algorithm or algorithm and Action-part of the SON-Function instance. The transition at which the coordination is performed is defined by the coordination scheme that has been assigned to a SON-Function. Apart from algorithm and action coordination it is also possible to assign a combined coordination scheme to a SON-Function so that coordination is performed at both transitions. The SON Coordinator uses the available context information when a respective (algorithm- or action-) coordination request event is received together with the SON-Function instance specific information, which is provided by the request to evaluate the current situation. If there is no conflict, the next step in the SON-Function instance execution will be acknowledged. In case of a detected conflict, the request will be rejected to prevent a conflict, or necessary conflict resolution measures will be triggered.

The applied coordination logic resolves SON-Function instance conflicts not only based on absolute priorities but also on operational guidelines. The approach to have the coordination logic external to the deployed SON-Functions provides multiple benefits. SON-Function developers provide a generic coordination logic along with the SON-Function which is then refined whenever required, thus operator can modify the provided coordination logic according to the concrete needs in his network. The representation of the coordination logic, first as solution agnostic decision trees, and subsequently in simple ECA Policies, is important for the understandability and maintainability of the coordination logic.

The presented concept with the provided blueprint for SON-Function development is an enabler for SON in multi-vendor networks. As long as a SON-Function developer follows the proposed monitoring-algorithm-action structure and assures the interaction with the SON Coordinator according to the assigned coordination scheme, the presented SON Coordinator can coordinate any kind of SON-Function. The network operator has even the possibility to influence the behavior of the SON-Functions through modifications of the coordination logic, as shown in the concept evaluation chapter. Instead of waiting for updated versions of the deployed SON-Functions, an operator can instantaneously influence the behavior of the SON-Function instances and the enforced configuration changes to prevent negative effects

on the network performance. For example if the Monitoring-part of a SON-Function has to many false positives when detecting triggering situations, it is possible to cross-check the result and potentially reject an unnecessary algorithm execution request.

The evaluation of the approach based on the proof of concept implementation of the SON Coordinator, a commercial LTE network simulator, and first generation SON-Functions showed the significant positive effects of SON-Function instance execution coordination on the resulting network performance. It was demonstrated that the coordination of SON-Function instances is even required to prevent the negative effects of the concurrent execution of SON-Function instances. The presented results prove that coordination is required not only to prevent inter-function and intra-function conflicts but also to protect the network from unwanted behavior, as for example, oscillating reconfigurations.

Based on a basic knowledge about the characteristics of the SON-Functions, an analysis of their behavior, and the behavior of the network simulator, the coordination logic for the evaluation was developed. It was shown that, compared to the uncoordinated execution of the SON-Function instances the resulting performance of the network was significantly better with active SON-Function instance coordination. An existing coverage hole was reliably closed, which was reflected in the improved cell throughput, as well as a major reduction in RLFs and HO drops. If no coordination was applied the cell throughput and the number of RLFs would not be improved but degraded.

Although, in this work, coordination has been used exclusively to handle conflicts within Self-Organizing Networks it is designed as a generic conflict prevention and resolution approach. As long as some basic requirements are satisfied it can be also used for conflict prevention and resolution in other application domains. These requirements are, for example, the full control of the coordination entity over the active functions and the possibility to specify an appropriate coordination logic. The event based operation is taken for granted since it is considered to be the standard mode of operation for network management systems that are based on a MAPE loop.

The performed case study demonstrates the development of a SON-Function for the Physical Cell ID Assignment use case. It follows exactly the proposed SON-Function development process and shows how the individual steps are built on top of each other and how they complement each other. After following the process, the SON-Function including Impact-area and Impact-time definition and an appropriate coordination logic were available and could be deployed in the network, allowing the coordination through a SON Coordinator. The developed SON-Function algorithm and the presented results serve, in the meanwhile, as a basic reference for multiple Physical Cell ID assignment approaches.

12.2 Future Work

There are several open questions that have to be treated in future work.

Most important are the design-time processes for the development of the SON-Functions and the coordination logic. The presented design-time process can be used as a blueprint but some parts have to be further developed. First, this applies to the refinement and implementation of high-level operational goals into a machine executable coordination logic. In this work this step has been performed manually, but as soon as there is a large number of deployed SON-Functions a reliable process that formally refines high level operational goals and prioritizations between different SON-Functions is required. Such a system would allow an abstract specification of all SON-Functions and network specific requirements, which are then unified and compiled to executable policies for the specific network management system used in the network they are deployed in. This provides the network operator with the opportunity to adapt the coordination logic which was provided by the SON-Function vendors according to his requirements.

Second, a more dynamic coordination logic needs to be developed. Networks show different behavior at different points in time. Instead of configuring fixed thresholds within the coordination logic, statistical and semantic methods could be used in the future. Seasonal trends would be incorporated automatically and the coordination logic could be adapted to match cause and effects that have been unknown at design-time.

Such a learning coordination logic could also be used to detect and prevent undesired behavior like oscillating reconfiguration. To achieve such results the coordination logic needs to include means to analyse the performed changes of the executed SON-Function instances (from the context information) and the caused effects.

A last important field is the semantic modelling of the deployed NEs and SON-Functions together with their interrelations, which would contribute to the possibility of a fully automatic design-time conflict analysis. If semantically enhanced models are available a reasoner can be used to find conflicts that today are not detected at all or only detected with lots of operational experience, as for example Characteristic conflicts. But even for the detection of other conflict types, semantic modelling can be beneficial, for example, if different vendors use different naming schemes for the same KPIs, or different sets of configuration parameters are used to reach the same goal.

The goal of SON is not only to reduce OPEX through automation but also to improve the overall network performance and minimize the reaction delay for failure recovery. There are two approaches to reach these goals, speed up the execution of management processes, and increase the overall number of performed management processes. Management tasks, which are done today in several day intervals due to their complexity, will be performed

in much shorter intervals for example several times a day. The presented coordination concept supports this future operation scheme already, but further experiments with more dynamic SON-Function instance execution should be performed to validate the expected behavior.

The experimental system can play an important role for the development of future SON-Functions, which are not only automated versions of today's management tasks, but make full use of the available degrees of freedom. An example is the real-time access to Performance Measurement data instead of their tight coupling to GPs and instant enforcement of configuration changes. Developers can not only develop and evaluate new SON-Functions within the experimental system but also the respective coordination logic. New functionality as for example statistical oscillation detection or a coordination logic based on uncertainty can easily be added and tested.

The developed coordination concept and the implemented SON Coordinator is considered to be a solid basis for future developments in the area of SON. For the usage in other application domains it is up to the experts of the different fields to determine how and in which configuration coordination can best be used for conflict prevention and resolution.

APPENDIX

GLOSSARY

3GPP Third Generation Partnership Project.

ACI Autonomic Computing Initiative.

Action-part The final part of a SON-Function which performs the action which have been prepared through the Algorithm-part, for example reconfiguration of NEs, triggering other SON-Functions or interacting with other network management systems..

Algorithm-part The second component of a SON-Function. Evaluation of the current situation and preparation of the actions to be performed. The Algorithm-part is triggered through an algorithm execution request issued by the Monitoring-part.

ANR Automated Neighbor Relationship Assessment.

ARCF Automatic Radio Configuration Function.

BSC Base Station Controller.

BTS Base Tranceiver Station.

CCO Coverage and Capacity Optimization.

CCO(RET) Coverage and Capacity Optimization via Remote Electrical Tilt Adaptation.

CCO(TXP) Coverage and Capacity Optimization via Antenna Transmission Power Adaptation.

CEPT Conférence Européenne des Administrations des Postes et des Télécommunications.

CM Configuration Management.

CN Core Network.

COC Cell-Outage Compensation.

CQI Channel Quality Indicator.

CSG Closed Subscriber Group.

DM Domain Manager.

DMTF Distributed Management Task Force.

E-UTRA Evolved Universal Terrestrial Radio Access.

E-UTRAN Evolved Universal Terrestrial Radio Access Network.

ECA Event Condition Action.

ECGI E-UTRAN Cell Global ID.

EDGE Enhanced Data Rates for GSM Evolution.

EMS Element Management System.

eNodeB Evolved NodeB.

eNodeB Evolved NodeB.

EPC Evolved Packet Core.

ESM Energy Saving Management.

FM Fault Management.

GERAN GSM EDGE Radio Access Network.

GP Granularity Period.

GPRS General Packet Radio Service.

GSM Global System for Mobile Communications.

HetNet Heterogeneous Network.

HO Handover.

HSPA High Speed Packet Access.

IETF Internet Engineering Task Force.

IETF Internet Engineering Task Force.

IRP Integration Reference Point.

ISD Inter Site Distance.

KPI Key Performance Indicator.

KQI Key Quality Indicator.

LTE Long Term Evolution.

MAPE Monitor-Analyze-Plan-Execute.

MLB Mobility Load Balancing.

MME Mobility Management Entity.

MNO Mobile Network Operator.

Monitoring-part The initial part of a SON-Function, that is used to detect a triggering situation, either by analysing performance measurements, configuration settings or particular triggering events.

MRO Mobility Robustness Optimization.

MSC Mobile Switching Center.

NE Network Element.

NGMN Next Generation Mobile Networks.

NID Numerical Representation of the Physical Cell ID.

NM Network Manager.

NMS Network Management System.

NRT Neighbor Relation Table.

OAM Operation Administration Maintenance.

OM Operation and Maintenance.

OPEX Operational Expenditures.

OSS Operations Support System.

PCI Physical Cell ID.

PDP Policy Decision Point.

PEP Policy Enforcement Point.

PM Performance Management.

QoS Quality of Service.

RAN Radio Access Network.

RAT Radio Access Technology.

RBAC Role Based Access Control.

RFC Request for Comments.

RLF Radio Link Failure.

RNC Radio Network Controller.

SGSN Serving General Packet Radio Service (GPRS) Support Node.

SGW Serving Gateway.

SLA Service Level Aggreement.

SM Safety-margin.

SON Self-Organizing Network.

TA Traffic Area.

TR Technical Report.

TS Technical Specification.

TSG Technical Specification Group.

UE User Equipment.

UMTS Universal Mobile Telecommunications System.

UTRA UMTS Terrestrial Radio Access.

UTRAN UMTS Terrestrial Radio Access Network.

WG Working Group.

LIST OF FIGURES

BIBLIOGRAPHY

[3GP] 3GPP. 3GPP Scope and Objectives, 08.

[3GP11] 3GPP. Telecommunication management; Fault Management; Part 1: 3G fault management requirements; V10.1.0, September 20011.

[3GP05] 3GPP. 3gpp ts 32.401 v6.4.1. Technical Specification Concepts and Requirements - Version 6, 3GPP, 02 2005.

[3GP08] 3GPP. 3GPP TS 36.300 - Evolved Universal Terrestrial Radio Access (E-UTRA) and Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Overall description; Stage 2 V.11. Technical Specification Release 8, 3GPP, 09 2008.

[3GP09a] 3GPP. 3GPP S5-092289. Technical Report Parameters Subject to Dynamic Radio Configuration, 3GPP, 05 2009.

[3GP09b] 3GPP. 3gpp ts 32.502. Technical Specification Concepts and Requirements - Version 6, 3GPP, 02 2009.

[3GP09c] 3GPP. 3gpp ts 32.511. Technical Specification Concepts and Requirements - Version 6, 3GPP, 02 2009.

[3GP09d] 3GPP. Telecommunication management; Performance Management (PM); Performance measurements - GSM, December 2009.

[3GP09e] 3GPP. Telecommunication management; Performance Management (PM); Performance measurements; Core Network (CN) Packet Switched (PS) domain, December 2009.

[3GP09f] 3GPP. Telecommunication management; Performance Management (PM); Performance measurements; Teleservice, December 2009.

[3GP09g] 3GPP. Telecommunication management; Performance Management (PM); Performance measurements; Universal Terrestrial Radio Access Network (UTRAN), December 2009.

[3GP10a] 3GPP. Telecommunication management; Configuration Management (CM); Concept and high-level requirements, May 2010.

[3GP10b] 3GPP. TR36.902 Self-configuring and self-optimizing network (SON) use cases and solutions. Technical report, 3GPP, 2010.

[3GP11a] 3GPP. 3GPP work items on Self-Organizing Networks v0.0.7. Technical report, 3GPP, 2011.

[3GP11b] 3GPP. Evolved Universal Terrestrial Radio Access Network (E-UTRAN); X2 general aspects and principles. TS 36.420, 3rd Generation Partnership Project (3GPP), April 2011.

[3GP11c] 3GPP. GSM/EDGE Radio Access Network (GERAN) overall description; Stage 2, V.10. TS 43.051, 3rd Generation Partnership Project (3GPP), December 2011.

[3GP11d] 3GPP. Technical Specification Group working methods. TR 21.900, 3rd Generation Partnership Project (3GPP), June 2011.

[3GP11e] 3GPP. Telecommunication management; Architecture V10.0.0, MAR 2011.

[3GP11f] 3GPP. Telecommunication management; Performance Management (PM); Performance measurements; Core Network (CN) Circuit Switched (CS) domain; UMTS and combined UMTS/GSM, June 2011.

[3GP11g] 3GPP. Telecommunication management; Performance Management (PM); Performance measurements Evolved Universal Terrestrial Radio Access Network (E-UTRAN), June 2011.

[3GP11h] 3GPP. Telecommunication management; Performance Management (PM); Performance measurements; IP Multimedia Subsystem (IMS), April 2011.

[3GP11i] 3GPP. UTRAN overall description. TS 25.401, 3rd Generation Partnership Project (3GPP), April 2011.

[3GP12a] 3GPP. Evolved Universal Terrestrial Radio Access (E-UTRA); Physical channels and modulation, March 2012.

[3GP12b] 3GPP. Overview of 3GPP Release 10 V0.1.3. Technical report, 3GPP, 2012.

[3GP12c] 3GPP. Overview of 3GPP Release 10 V0.1.3Overview of 3GPP Release 10 V0.1.3. Technical report, 3GPP, 01 2012.

[3GP12d] 3GPP. Overview of 3GPP Release 8 - V0.2.5. Technical report, 3GPP, 2012.

[3GP12e] 3GPP. Overview of 3GPP Release 9 V0.2.4. Technical report, 3GPP, 2012.

[ABB+08] Meriem Abid, Andreas Berl, Zohra Boudjemil, Abderhaman Cheniour, Steven Davy, Spyros Denazis, Alex Galis, Jean-Patrick Gelas, Claire Fahy, Andreas Fischer, Javier Rubio Loyola, Laurent Lefevre, Daniel Macedo, Lefteris Mamatas, Hermann de Meer, Zeinab Movahedi, John Strassner, and Hubert Zimmermann. Autonomic internet - initial framework - deliverable d6.1. Technical report, Autonomic Internet (AutoI) Project, 2008.

[ABG+06] Hamid Akhavan, Vivek Badrinath, Thomas Geitner, Dr. Horst Lennertz, Yuejia Sha, Takanori Utano, and Barry West. Next Generation Mobile Networks - Beyond HSPA & EVDO. Technical report, NGMN Project, December 2006.

[AEL+11] Mehdi Amirijoo, Andreas Eisenblaetter, Remco Litjens, Michaela Neuland, Lars-Christoph Schmelz, and John Turk. A Coordination Framework for Self-Organisation in LTE Networks. In *IM 2011*, 2011. submitted for publication.

[AJLS11] M. Amirijoo, L. Jorguseski, R. Litjens, and L. C. Schmelz. Cell Outage Compensation in LTE Networks: Algorithms and Performance Assessment. In *Proc. IEEE 73rd Vehicular Technology Conf. (VTC Spring)*, pages 1–5, 2011.

[Alc09] Alcatel-Lucent. Introduction to Evolved Packet Core. Technical report, Alcatel-Lucent, 2009.

[All11] NGMN Alliance. The NGMN Alliance - At a glance, 2011.

[Ban11] Tobias Bandh. The SOCRATES SON-Function Coordination Concept. Technical report, TUM, November 2011.

[BBD+06a] S. Balasubramaniam, K. Barrett, W. Donnelly, S. van der Meer, and J. Strassner. Bio-inspired policy based management (biopbm) for autonomic bio-inspired policy based management (biopbm) for autonomic. In *Policies for Distributed Systems and Networks, 2006. Policy 2006. Seventh IEEE International Workshop on*, pages 3 –12, june 2006.

[BBD+06b] Sasitharan Balasubramaniam, Dmitri Botvich, William Donnelly, Mícheál Ó Foghlú, and John Strassner. Biologically inspired self-governance and self-organisation for autonomic networks. In *Proceedings of the 1st international conference on Bio inspired models of network, information and computing systems*, BIONETICS '06, New York, NY, USA, 2006. ACM.

[BK10] U. Barth and E. Kuehn. Self-organization in 4G mobile networks: Motivation and vision. In *Wireless Communication Systems (ISWCS), 2010 7th International Symposium on*, pages 731–735. IEEE, 2010.

[BRB+10] Tobias Bandh, Raphael Romeikat, Bernhard Bauer, Georg Carle, Henning Sanneck, and Lars Christoph Schmelz. Policy-driven workflows for mobile network management automation. In *Proceedings of the 6th International Wireless Communications and Mobile Computing Conference*, IWCMC '10, pages 1111–1115, New York, NY, USA, 2010. ACM.

[BRS+10] Tobias Bandh, Raphael Romeikat, Henning Sanneck, Lars Christoph Schmelz, Bernhard Bauer, and Georg Carle. Optimized Network Configuration Parameter Assignment Based on Graph Coloring. In *NOMS 2010*, 2010.

[BRS11] Tobias Bandh, Raphael Romeikat, and Henning Sanneck. Policy-based Coordination and Management of SON Functions. In *IM 2011*, 2011.

[BRS12] Tobias Bandh, Raphael Romeikat, and Henning Sanneck. An Integrated SON Experimental System for Self-Optimization and SON Coordination. In *International Workshop on Self-Organizing Networks (IWSON 2012)*, Paris,France, August 2012.

[BRST10] Tobias Bandh, Raphael Romeikat, Henning Sanneck, and Haitao Tang. Policy-based coordination and management of SON-functions. In *VDE/ITG Fachgruppengespräche, Consistent System Optimization in 3G/4G Wireless Networks*, Stuttgart, Germany, October 2010.

[BS11] Tobias Bandh and Henning Sanneck. Automatic Site Identification and Hardware-to-Site-Mapping for Base Station Self-Configuration. In *Vehicular Technology Conference (VTC Spring), 2011 IEEE 73rd*, pages 1 –5, may 2011.

[BS12] Tobias Bandh and Christoph Schmelz. Impact-time Concept for SON-Function Coordination. In *International Workshop on Self-Organizing Networks (IWSON 2012)*, Paris,France, August 2012.

[BSC09] Tobias Bandh, Henning Sanneck, and Georg Carle. Graph Coloring Based Physical Cell ID Assignment for LTE Networks. In *IWCMC2009*, 2009.

[BSDB06] J. Baliosian, F. Sailhan, A. Devitt, and A.M. Bosneag. The Omega Architecture: Towards Adaptable, Self-Managed Networks. In *proceedings of the 1st Annual Workshop on Distributed Autonomous Network Management Systems*, 2006.

[BSL] Base Station List - http://gsm.schnurstein.de/download/senderliste_obdg.kmz. 01.09.2008.

[BSR11] Tobias Bandh, Henning Sanneck, and Raphael Romeikat. An Experimental System for SON Function Coordination. In *Vehicular Technology Conference (VTC Spring), 2011 IEEE 73rd*, pages 1 –2, May 2011.

[BSS+11] Tobias Bandh, Cinzia Sartori, Péter Szilágyi, Yves Bouwen, Eddy Troch, Jürgen Goerge, Simone Redana, Raimund Kausl, Christoph Schmelz, and Henning Sanneck. Self-Configuration (Plug-and-Play). In *Chapter in Book: LTE Self-Organising Networks (SON): Network Management Automation for Operational Efficiency, Seppo Hämäläinen and Henning Sanneck and Cinzia Sartori (Eds.), Wiley.* December 2011.

[BTSS11] Tobias Bandh, Haitao Tang, Christoph Schmelz, and Henning Sanneck. SON Operation. In *Chapter in Book: LTE Self-Organising Networks (SON): Network Management Automation for Operational Efficiency, Seppo Hämäläinen and Henning Sanneck and Cinzia Sartori (Eds.), Wiley.* December 2011.

[C+06] A. Computing et al. An architectural blueprint for autonomic computing. *IBM White Paper*, 2006.

[Cha10] Ranganai Chaparadz. Final Draft of Autonomic Behaviours Specifications (ABs) for Selected Diverse Networking Environments (Core Document). Technical report, EFIPSANS, 2010.

[Cis11] Cisco. Cisco visual networking index: Global mobile data traffic forecast 2010 - 2015. Technical report, Cisco, 2011.

[CSW02] M. Coinchon, A. P. Salovaara, and J. F. Wagen. The impact of radio propagation predictions on urban umts planning. In *Broadband Communications*, 2002.

[DA10] F. Dressler and O.B. Akan. A survey on bio-inspired networking. *Computer Networks*, 54(6):881–900, 2010.

[DBC+00] D. Durham, J. Boyle, R. Cohen, S. Herzog, R. Rajan, and A. Sastry. The COPS (Common Open Policy Service) Protocol. RFC 2748 (Proposed Standard), January 2000. Updated by RFC 4261.

[DJ07] Steven Davy and Brendan Jennings. Harnessing Models for Policy Conflict Analysis. In *AIMS '07: Proceedings of the 1st international conference on Autonomous Infrastructure, Management and Security*, pages 176–179, Berlin, Heidelberg, 2007. Springer-Verlag.

[DJS08] Steven Davy, Brendan Jennings, and John Strassner. The policy continuum-Policy authoring and conflict analysis. volume 31, pages 2981–2995, Newton, MA, USA, 2008. Butterworth-Heinemann.

[DSESitDoC02] UK. Distributed Software Engineering Section in the Department of Computing, Imperial College London. The PONDER Policy Based Management Toolkit. Technical report, Distributed Software Engineering Section in the Department of Computing, Imperial College London, UK., 2002.

[DV09] M. Dottling and I. Viering. Challenges in mobile network operation: Towards self-optimizing networks. In *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, pages 3609–3612. Ieee, 2009.

[ea09a] Ranganai Chaparadza et al. Creating a viable Evolution Path towards Self-Managing Future Internet via a Standardizable Reference Model for Autonomic Network Engineering. In *IEEE Signal Process. Lett.'09*, pages 136–147, 2009.

[ea09b] Schmelz et al. Self-organisation in wireless networks–use cases and their interrelation. In *WWRF Meeting*, volume 22, 2009.

[ea11] Sara Landstrom et al. Heterogeneous networks - increasing cellular capacity. *Ericsson Review*, 2011.

[EFI] EFIPSANS. Efipsans www.efipsans.org (2013.04.20). 2013-04-20.

[Eis03] Andreas Eisenblätter. Assigning Frequencies in GSM Networks. Technical report, Zuse Institut Berlin, 2003.

[EKG11] H. Eckhardt, S. Klein, and M. Gruber. Vertical Antenna Tilt Optimization for LTE Base Stations. In *Proc. IEEE 73rd Vehicular Technology Conf. (VTC Spring)*, pages 1–5, 2011.

[FK92] David Ferraiolo and Richard Kuhn. Role-based access control. In *In 15th NIST-NCSC National Computer Security Conference*, pages 554–563, 1992.

[FTP02] Paris Flegkas, Panos Trimintzios, and George Pavlou. A Policy-Based Quality of Service Management System for IP DiffServ Networks, 2002.

[GBK11] M. Gruber, S. Borst, and E. Kuehn. Stable Interaction of Self-Optimization Processes in Wireless Networks. In *Communications Workshops (ICC), 2011 IEEE International Conference on*, pages 1–6, 2011.

[HL11] Gao Hui and P. Legg. LTE Handover Optimisation Using Uplink ICIC. In *Proc. IEEE 73rd Vehicular Technology Conf. (VTC Spring)*, pages 1–5, 2011.

[Hor01] P. Horn. Autonomic computing: Ibm's perspective on the state of information technology. *Computing Systems*, 15(Jan):1–40, 2001.

[HSS12] S. Hämäläinen, H. Sanneck, and C. Sartori. *LTE Self-Organising Networks (SON): Network Management Automation for Operational Efficiency.* John Wiley & Sons, 2012.

[HT11] Harri Holma and Antti Toskala. *LTE for UMTS: Evolution to LTE-Advanced.* Wiley, 2 edition, 5 2011.

[JBS+11] T. Jansen, I. Balan, S. Stefanski, I. Moerman, and T. Kurner. Weighted Performance Based Handover Parameter Optimization in LTE. In *Proc. IEEE 73rd Vehicular Technology Conf. (VTC Spring)*, pages 1–5, 2011.

[JT94] Tommy R. Jensen and Bjarne Toft. *Graph Coloring Problems (Wiley Series in Discrete Mathematics and Optimization)*. Wiley-Interscience, 1 edition, 12 1994.

[KAB+10] Thomas Kürner, Mehdi Amirijoo, Irina Balan, Hans van den Berg, Andreas EisenblÃ¤tter, Thomas Jansen, Ljupco Jorguseski, Remco Litjens, Ove Linnell, Andreas Lobinger, Michaela Neuland, Frank Phillipson, Lars Christoph Schmelz, Bart Sas, Neil Scully, Kathleen Spaey, Szymon Stefanski, John Turk, Ulrich Türke, and Kristina Zetterberg. SOCRATES Deliverable 5.9 Final report on self-organisation and its implications in wireless access networks. Technical report, SOCRATES FP7 Project, 2010.

[KC03a] Jeffrey O. Kephart and David M. Chess. IBM ACI - The Vision of Autonomic Computing . *IEEE Computer magazine*, 1:41–50, 2003.

[KC03b] Jeffrey O. Kephart and David M. Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, January 2003.

[KKP08a] T. Kastrinogiannis, V. Karyotis, and S. Papavassiliou. An Opportunistic Combined Power and Rate Allocation Approach in CDMA Ad Hoc Networks. In *Sarnoff Symposium, 2008 IEEE*, pages 1–5, 2008.

[KKP08b] T. Kastrinogiannis, V. Karyotis, and S. Papavassiliou. On the problem of joint power and rate control in CDMA ad hoc networks. In *Wireless Pervasive Computing, 2008. ISWPC 2008. 3rd International Symposium on*, pages 78–82, 2008.

[KP09] Timotheos Kastrinogiannis and Symeon Papavassiliou. Game theoretic distributed uplink power control for CDMA networks with real-time services. *Computer Communications*, 32(2):376–385, 2009.

[KTP08] Timotheos Kastrinogiannis, Eirini-Eleni Tsiropoulou, and Symeon Papavassiliou. Utility-Based Uplink Power Control in CDMA Wireless Networks with Real-Time Services. In David Coudert, David Simplot-Ryl, and Ivan Stojmenovic, editors, *Ad-hoc, Mobile and Wireless Networks*, volume 5198 of *Lecture Notes in Computer Science*, pages 307–320. Springer Berlin Heidelberg, 2008.

[Leh07a] Frank Lehser. Next Generation Mobile Networks - Informative List of SON Use Cases. Technical report, NGMN Alliance, 2007.

[Leh07b] Frank Lehser. Next Generation Mobile Networks - Use Cases related to Self Organising Network, Overall Description. Technical report, NGMN Alliance, 2007.

[Leh08] NGMN Frank Lehser. Next Generation Mobile Networks Recommendation on SON and O&M Requirements. Technical report, NGMN, 2008.

[Leh10] Frank Lehser. Ngmn top ope recommendations. Technical report, NGMN, 2010.

[LJSS09] Poongup Lee, Jangkeun Jeong, Navrati Saxena, and Jitae Shin. Dynamic reservation scheme of physical cell identity for 3gpp lte femtocell systems. *JIPS*, pages 207–220, 2009.

[LS99] Emil Lupu and Morris Sloman. Conflicts in policy-based distributed system management. *IEEE Transactions on Software Engineering, Special Issue on Inconsistency Management*, 25(6):852–869, 1999.

[LSJB11] A. Lobinger, S. Stefanski, T. Jansen, and I. Balan. Coordinating Handover Parameter Optimization and Load Balancing in LTE Self-Optimizing Networks. In *Proc. IEEE 73rd Vehicular Technology Conf. (VTC Spring)*, pages 1–5, 2011.

[Mer09] Mercedes Benz USA. MBRACE Brochure, 2009. http://www.mbusa.com/vcm/MB/DigitalAssets/pdfmb/brochures/mbrace-Brochure.pdf.

[MESW01] B. Moore, E. Ellesson, J. Strassner, and A. Westerinen. Policy Core Information Model – Version 1 Specification. RFC 3060 (Proposed Standard), February 2001. Updated by RFC 3460.

[Moo03] B. Moore. Policy Core Information Model (PCIM) Extensions. RFC 3460 (Proposed Standard), January 2003.

[Mot] Inc Motorola. Long Term Evolution (LTE): A Technical Overview. Technical report, Motorola Inc, http://www.motorola.com/web/Business/Solutions/Industry20Providers/Wirelessp Retrieved December 2011.

[MS06] Samuel Michaelis and Wolfgang Schmiesing. *JAXB 2.0: Ein Programmiertutorial fÃ¼r die Java Architecture for XML Binding*. Hanser Fachbuchverlag, 1 edition, 2006.

[Net12] Nokia Siemens Networks. Deployment Strategies for Heterogeneous Networks. Technical report, Nokia Siemens Networks, 2012.

[Plc11] Vodafone Group Plc. Annual report - for the year ended 31 march 2011, March 2011.

[Pro] UniverSelf Project. www.univerself-project.eu. Project Website. 2013-04-07.

[RDD07] G. Russello, Changyu Dong, and N. Dulay. Authorisation and conflict resolution for hierarchical domains. In *Policies for Distributed Systems and Networks, 2007. POLICY '07. Eighth IEEE International Workshop on*, pages 201 –210, june 2007.

[RdlBMB11] J. Rodriguez, I. de la Bandera, P. Munoz, and R. Barco. Load Balancing in a Realistic Urban Scenario for LTE Networks. In *Proc. IEEE 73rd Vehicular Technology Conf. (VTC Spring)*, pages 1–5, 2011.

[RH11] J. Ramiro and K. Hamied. *Self-Organizing Networks (SON): Self-Planning, Self-Optimization and Self-Healing for GSM, UMTS and LTE*. Wiley, 2011.

[RLS10] Javier Rubio-Loyola and Joan Serrat. Final Results of the Autonomic Internet Approach - Deliverable D6.3. Technical report, Autonomic Internet (AUTOI) Project, 2010.

[SAE+11] Lars Christoph Schmelz, Mehdi Amirijoo, Andreas Eisenblaetter, Remco Litjens, Michaela Neuland, and John Turk. A Coordination Framework for Self-Organisation in LTE Networks. In *Proceedings of the 12th IFIP/IEEE International Symposium on Integrated Network Management 2011*, 2011.

[Sau09] Martin Sauter. *Beyond 3G - Bringing Networks, Terminals and the Web Together: LTE, WiMAX, IMS, 4G Devices and the Mobile Web 2.0*. Wiley, 1 edition, 2 2009.

[Sau11] Martin Sauter. *From GSM to LTE: An Introduction to Mobile Networks and Mobile Broadband*. Wiley, 1 edition, 3 2011.

[SBS12] Péter Szilágyi, Tobias Bandh, and Henning Sanneck. Physical Cell ID Allocation in Multi-layer, Multi-vendor LTE Networks. In *Proceedings of 4th International Conference on Mobile Networks and Management*, Hamburg, Germany, September 2012. **Best Paper Award**.

[SBT10] H. Sanneck, Y. Bouwen, and E. Troch. Dynamic radio configuration of Self-Organizing base stations. In *Wireless Communication Systems (ISWCS), 2010 7th International Symposium on*, pages 716 –720, sept. 2010.

[Sch08] Christoph Schmelz. Socrates - FP7 - http://www.fp7-socrates.org. Website, 12 2008. 02.12.2008.

[Sch09] Thomas Schnurstein. Base Station List - http://gsm.schnurstein.de/download/senderliste_obdg.kmz. Website, August 2009. 01.09.2008.

[Sma12] Small Cell Forum. Small Cells - What's the big Idea. Technical report, Small Cell Forum, 02 2012.

[SRS+03] Y. Snir, Y. Ramberg, J. Strassner, R. Cohen, and B. Moore. Policy Quality of Service (QoS) Information Model. RFC 3644 (Proposed Standard), November 2003.

[STB11] Stefania Sesia, Issam Toufik, and Matthew Baker. *LTE - The UMTS Long Term Evolution: From Theory to Practice*. Wiley, 2 edition, 9 2011.

[Str04] John C. Strassner. *Policy-Based Network Management: Solutions for the Next Generation*. Morgan Kaufmann Publishers, San Francisco, CA, USA, 2004.

[SZSS10] Neil Scully, Kristina Zetterberg, Szymon Stefanski, and Lars Christoph Schmelz. Socrates deliverable 5.10 measurements, architecture and interfaces for self-organising networks. Technical report, Socrates FP7 Project, 2010.

[TC10] N. Tcholtchev and R. Chaparadza. Autonomic Fault-Management and resilience from the perspective of the network operation personnel. In *GLOBECOM Workshops (GC Wkshps), 2010 IEEE*, pages 469–474, 2010.

[Tem10] Stephen Temple. *Inside the Mobile Revolution - A Political History of GSM*. 2010.

[TK07] Lars Wolf Torsten Klie, Benjamin Ernst. Automatic policy refinement using owl-s and semantic infrastructure information. In *MACE 2007 - 2nd IEEE International Workshop on Modelling Autonomic Communications Environments*, October 2007.

[TLDS08] Kevin Twidle, Emil Lupu, Naranker Dulay, and Morris Sloman. Ponder2 - a policy environment for autonomous pervasive systems. In *Workshop on Policies for Distributed Systems and Networks (POLICY '08)*, pages 245–246. IEEE Computer Society, 2008.

[Uni12] UniverSelf. CASE STUDY - PART I SON and SON collaboration according to operator policies. Technical report, UniverSelf, 2012.

[Ver02] D.C. Verma. Simplifying network administration using policy-based management. *IEEE Network*, 16(2):20–26, 2002.

[Wie94] Rene Wies. Policies in Network and Systems Management - Formal Definition and Architecture -. *Journal of Network and Systems Management*, 2:63–83, 1994.

[WJWZ10] Yi Wu, Hai Jiang, Ye Wu, and Dongmei Zhang. Physical cell identity self-organization for home enodeb deployment in lte. In *Wireless Communications Networking and Mobile Computing (WiCOM), 2010 6th International Conference on*, pages 1 –6, 2010.

[WP67] D. J. A. Welsh and M. B. Powell. An upper bound for the chromatic number of a graph and its application to timetabling problems. *The Computer Journal*, 10(1):85–86, 1967.

[WSS+01] A. Westerinen, J. Schnizlein, J. Strassner, M. Scherling, B. Quinn, S. Herzog, A. Huynh, M. Carlson, J. Perry, and S. Waldbusser. Terminology for Policy-Based Management. RFC 3198 (Informational), November 2001.

[xml08] xmlBlaster.org. xmlblaster. `http://www.xmlBlaster.org`, November 2008.

[YPG00] R. Yavatkar, D. Pendarakis, and R. Guerin. A Framework for Policy-based Admission Control. RFC 2753 (Informational), January 2000.

ISBN 3-937201-34-3

ISSN 1868-2634 (print)
ISSN 1868-2642 (electronic)