An Experimental System for SON Function Coordination

Tobias Bandh
Network Architectures and Services
Technische Universität München
bandh@net.in.tum.de

Henning Sanneck
Nokia Siemens Networks Research
henning.sanneck@nsn.com

Raphael Romeikat
Institute of Computer Science
University of Augsburg
romeikat@ds-lab.org

***Abstract*—Through the introduction of Self-Organizing Networks (SON) into mobile networks, a potentially large number of SON functions are available. These SON functions automatically perfom management actions. There will be SON functions for many aspects of fault, configuration, accounting, performance, and security (FCAPS) management. The functions are executed based on monitored network behavior, which may lead to several functions being active concurrently in the same network area. Simultaneous execution of different functions with contradicting goals may lead to oscillating function execution and service degradation in the worst case. Therefore, SON function coordination is indispensable for SON-enabled networks in order to align the executed SON funtions and thus assure that they take full effect (improved performance and fault handling). Coordination mechanisms need to be developed and verified before they are deployed into the network. For this purpose, an experimental system realizes SON function coordination based on flexible policy-based decisions. Coverage and Capacity Optimization (CCO) is presented as use case to demonstrate succesful coordination of multiple independent SON functions.**

I. Introduction

A Self-Organizing Network (SON) will contain a large number of independently acting SON functions across various use cases from all areas of network management [1]. Many of those functions will have an impact on each other and thus the network when triggered concurrently [2]. Furthermore, it is obvious that e.g. fault recovery functions have to be priorized over network optimization functions. Unwanted effects may may also arise within the same SON function type.

The risk for such effects increases with the number of deployed SON functions. For this reason, coordination mechanisms are necessary in order to avoid errors, oscillations, or even deadlocks in the configuration [3]. A preventive coordination mechanism that uses policy-based decision taking has already been developed [4], [5]. An architecture to which these policy-based (or other) methods can be embedded is shown in [6]. Coordination also needs to be thoroughly tested and verified, which is not possible in an operational network. Experimental systems are required as a proof of concept that allow to simulate coordination cases but also integration with real or simulated SON functions.

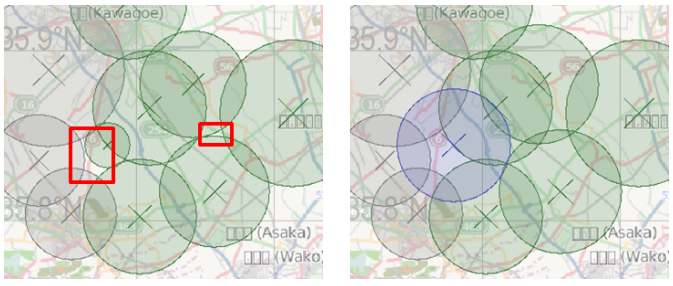
Section II describes particular SON functions and showcases and illustrates why coordination of those functions is necessary. Section III presents an experimental system used to implement coordination and to demonstrate the showcases.

II. Use Cases

Various SON functions have been analyzed and introduced so far. Part of the analysis focused on the interrelations between SON functions to determine how functions have to be coordinated with each other. The focus of this section is on the functions for Coverage and Capacity Optimization (CCO). CCO is performed through adaptations of throughput power (TXP) and antenna tilt (RET). Subsequent CCO(TXP) and CCO(RET) requests for changing the respective parameters need to be coordinated because they depend on the changes previously performed at a cell.

A. CCO without Coordination

In the first showcase a sequence of CCO requests is applied without coordination to a network. The network is initially configured with coverage holes and unbalanced cell sizes. As the CCO functions are not coordinated, some function instances are executed on the same or adjacent cells closely correlated in time. Hence, some function instances are executed without any effect as their effects are overridden by other function instances. This results in an optimized network configuration as shown in Figure 1a, but still two coverage holes remain due to uncoordinated function execution. The desired state with full coverage cannot not be reached.



(a) Without coordination (b) With coordination

Fig. 1: CCO without and with coordination

B. CCO with Coordination

In the second showcase the same sequence of CCO requests is applied with coordination. The coordination decisions are taken by policies that are visualized as decision trees in

Figure 2. Operational knowledge on function coordination is used to define the CCO decision trees. The logic expressed within them is mapped into three policies for each decision tree. Coordination of the CCO functions ensures that function instances do not override the effects of other function instances and any function instance has its desired effect. As a result, CCO is successful in the end and coverage holes are eliminated completely. The resulting network configuration without coverage holes is shown in Figure 1b.

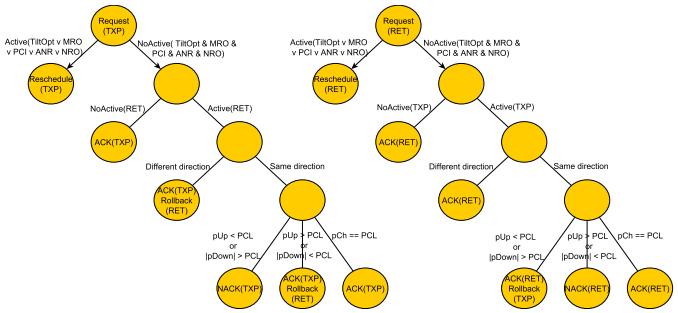


Fig. 2: Decision trees for CCO function coordination

If other functions such as Mobility Robustness Optimization (MRO) or Cell Outage Compensation (COC) are integrated, only those policies which have a dependency with the new function need to be changed. Function classification and grouping are additionally used to reduce the number of policies and dependencies between them.

III. Experimental System

In order to implement, demonstrate, and evaluate policy-based coordination, an experimental system was developed as a proof of concept for the approach presented in [4], [5]. It is built upon a distributed infrastructure and consists of different modules that are decoupled from each other and represent functional blocks. The modules can be combined in a flexible way as they communicate with each other through events over a message bus. The message bus represents the connecting point for the modules and is shown in Figure 3. The following modules were developed:

- **Visualization:** The network layout is displayed graphically on a map by this module. The module represents a management application for the operator that allows to interact with network elements (NEs). The operator can request configuration information about the cells, change the network layout, and trigger SON functions that operate on the NEs.
- **Execution:** This module implements the coordination mechanism and decides which SON functions to execute under which curcumstances. The decisions are actually taken by policies that evaluate events based on conditions and finally trigger actions. The module contains a policy engine to perform decisions automatically but under the control of the operator. It allows the operator to influence the behavior of the decision taking by e.g. activating, deactivating, or changing policies at any time.
- **Knowledge:** Coordination of SON functions also depends on configuration information. For this purpose, the Knowledge module represents a configuration database that stores information about the network elements and their current configuration. Other modules can request and change configuration information via certain events.
- **Event Generator:** Repeatable event sequences can be generated by this module to demonstrate and evaluate the coordination mechanism. This allows to create both synthetic scenarios to evaluate how coordination is performed in special cases, and real-world scenarios where the event pattern is taken from an actual network trace.
- **Event Monitor:** This module analyzes the event flow in the background, shows which SON functions are being executed, and logs earlier executed functions. The logical dependencies between the functions are visualized graphically in a diagram that traces the event flow over time.

The modular architecture makes the experimental system highly extensible. New SON functions can be integrated by extending the respective policies. Other simulators and systems (implementing the actual SON functions) can be integrated by simply attaching them to the message bus via message wrappers, e.g. integrating an existing alarm system as shown in Figure 3. Thus the experimental system can work on simulated or real events or a combination of them.

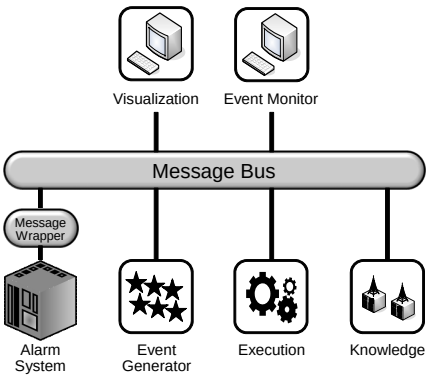


Fig. 3: Architecture of the experimental system

References

[1] 3GPP, "TR 36.902 – Self-configuring and self-optimizing network (SON) use cases and solutions," Technical Specification, 2010.

[2] L.-C. Schmelz, K. Spaey, H. van den Berg, L. Jorguseski, O. Linnell, T. Krner, and N. Scully, "Self-organisation in Wireless Networks – Use Cases and their Interrelation," WWRF Meeting 22, 2009.

[3] T. Jansen, M. Amirijoo, U. Türke, L. Jorguseski, K. Zetterberg, R. Nascimento, L.-C. Schmelz, J. Turk, and I. Balan, "Embedding Multiple Self-Organisation Functionalities in Future Radio Access Networks," in *VTC Spring 2009*. IEEE Vehicular Technology Society, 2009.

[4] R. Romeikat, B. Bauer, T. Bandh, G. Carle, H. Sanneck, and L.-C. Schmelz, "Policy-driven Workflows for Mobile Network Management Automation," in *IWCMC 2010*. ACM, 2010.

[5] T. Bandh, R. Romeikat, and H. Sanneck, "Policy-based Coordination and Management of SON Functions," in *IM 2011*, 2011, to be published.

[6] M. Amirijoo, A. Eisenblaetter, L. Remco, M. Neuland, L.-C. Schmelz, and J. Turk, "A Coordination Framework for Self-Organisation in LTE Networks," in *IM 2011*, 2011, submitted for publication.