Operational Troubleshooting-enabled Coordination in Self-Organizing Networks

Christoph Frenzel[1 3], Tsvetko Tsvetkov[2], Henning Sanneck[3], Bernhard Bauer[1], and Georg Carle[2]

[1] Department of Computer Science, University of Augsburg, Germany
{frenzel,bauer}@informatik.uni-augsburg.de
[2] Department of Computer Science, Technische Universität München, Germany
{tsvetko.tsvetkov,carle}@in.tum.de
[3] Nokia Solutions and Networks Research, Munich, Germany
henning.sanneck@nsn.com

Summary. A Self-Organizing Network (SON) performs automated network management through the coordinated execution of autonomous functions, each aiming to achieve a specific objective like the optimization of a network Key Performance Indicator (KPI) value. However, there are situations in which a SON function cannot achieve its objective which can lead to disturbed SON operation and inferior performance. We present a SON Operational Troubleshooting (SONOT) SON function that is able to detect such problematic situations and trigger respective countermeasures. Thereby, it can exploit regular SON functions as probes in order to improve problem detection. Simulations show that the tight integration of the SONOT function with SON coordination provides means to automatically overcome the problems and improve overall network performance.

Key words: Self-Organizing Network, SON operation, SON coordination, troubleshooting

1 Introduction

The Self-Organizing Network (SON) paradigm is an automated network operations approach which provides self-configuration, self-optimization, and self-healing capabilities for next generation mobile communication networks including Long Term Evolution (LTE) [6]. This is achieved by a collection of autonomous SON functions, each observing Performance Management (PM), Fault Management (FM), and Configuration Management (CM) data and changing network parameters in order to achieve a specific operator given objective or target [2], e.g., the reduction of the number of Handover (HO) failures between two network cells. However, the SON function objectives are connected and, so, the SON functions can interfere with each other at run-time, e.g., by contrary adjustments of the same network parameters. Such conflicts hamper seamless SON operations and can lead to inferior performance, hence, they are prevented

before or resolved after they happen by SON coordination [3]. Among the numerous approaches for SON coordination, run-time action coordination [11] is very common. It requires all SON functions to request for permission to change some network parameter at a SON coordination function. This function then determines conflicting requests, e.g., contrary changes of the same network configuration parameter, computes a set of non-conflicting SON functions that can be executed at the same time in the same area, and triggers their execution. Thereby, the conflict resolution may be based on operator priorities [3].

There can be network conditions and situations in which a SON function might not be able to achieve its targets [5]. Although, the reasons for this are often outside of the objective scope of the particular SON function, the problems that cause the function failure can often be resolved by another SON function. For instance, on the one hand, an Mobility Robustness Optimization (MRO) function running in an area with a coverage hole might not be able to reduce HO failures, however, on the other hand, the coverage problem can be handled by a Coverage and Capacity Optimization (CCO) function. The main problem of an ineffective SON function is that it may affect other SON functions due to the coordinated execution. An example for such an negative effect on other SON functions is network monopolization: a high priority function is constantly running because it encountered an unresolvable problem and, thereby, blocks the execution of other functions leading to a deadlock. In such problem situations, a SON function might need assistance by another SON function or the operator. Currently, the operation of SON functions is usually not monitored, thus, leaving such problems unnoticed and making the affected SON functions valueless.

In order to overcome these problems, we have sketched a preliminary concept in [5] that is able to detect conditions in which a SON function cannot achieve its objectives and mitigates this problem. Thereby, a SON function, namely the SON Operational Troubleshooting (SONOT) function, is proposed that can analyze the problem using a network-wide view and determine possible remedy actions, e.g., blocking functions that cannot achieve their objectives as well as triggering other functions that might resolve the problem.

In this paper, we present an approach for troubleshooting a SON that is based on the concept presented in [5]. In contrast to the previous work, we discuss the approach and its design aspects, especially the detection of ineffective functions, in much more depth including a detailed evaluation of the approach using an LTE network simulator. Additionally, we have extended the initial concept with the ability to exploit SON functions as probes, thereby, increasing accuracy and decreasing delay of problem detection. We present results of simulations which show the positive impact of the approach on network performance.

2 SON Operational Troubleshooting Approach

The SONOT approach presented in this paper consists of two steps, as depicted in Figure 1. First, there is an *alarm generation* step which detects that a SON function ran into some problematic state which it cannot handle by itself and

raises an alarm. Second, this alarm is evaluated and a corrective action is taken in the *alarm analysis and remedy* step.

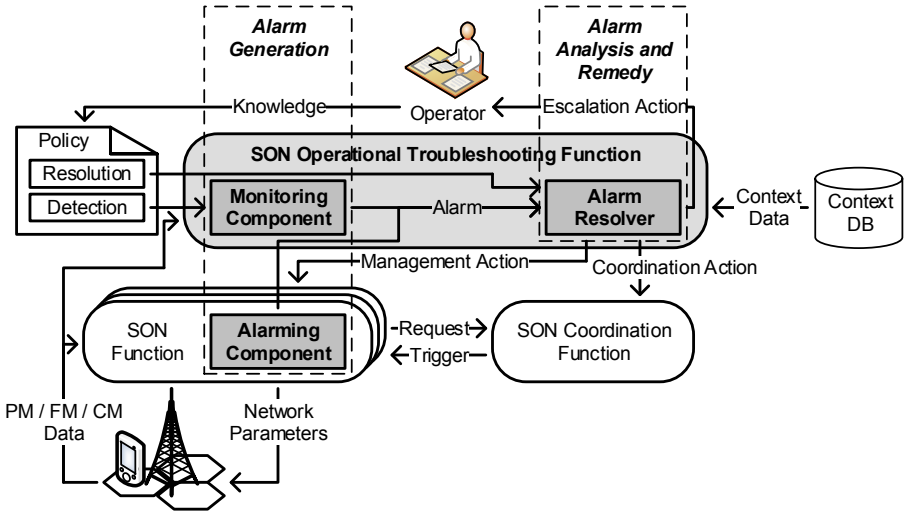


Fig. 1. The SONOT function and its interaction with other SON functions

Alarms may be generated by two different sources: the normal SON functions and the SONOT function. In the first case, each function is extended with an *alarming component* which raises an alarm if it encounters a problem during execution. In the second case, the SONOT function, more specifically its *monitoring component*, continuously analyses the network and generates an alarm if it encounters undesired behavior, controlled by the operator through a policy. The alarms trigger the analysis and remedy of the problem which produces a corrective action, e.g., a coordination action, a management action, or an escalation action. This process is based on a policy which captures the operational experience of the operator and the SON vendor. During execution, the SONOT function can access comprehensive contextual information about the operational status of the network, e.g., the current date and time, CM data like the network topology, PM data like Key Performance Indicators (KPIs), and FM data like technical alarms from the Network Elements (NEs).

2.1 Alarm Generation

In this paper, an alarm is an indication that a SON function is not able to achieve its objective, i.e., an alarm is not supposed to directly indicate hardware or software failures of NEs. Before generating an alarm, the system has to detect the problem that hinders a SON function from fulfilling its objective. There are two general approaches for this: the *state-based* approach considers the current

network state, whereas the *history-based* approach additionally considers previous network states and allows time series-based analyses of the system behavior. Hence, the latter can monitor the impact of network parameters changes, analyze performance trend in the network, and make predictions about the ability of a SON function to satisfy its objective. For instance, a history-based method can detect SON functions that are continuously modifying parameters without performance improvements, i.e., configuration oscillations.

Alarming Component The primary idea behind the alarm generation within the SON functions is to reuse their existing sophisticated monitoring and analysis capabilities, i.e., they are exploited as probes. On the one hand, this avoids the collection and transfer of PM, FM, and CM data that is used by the SON function anyway, thus, leading to less management overhead in the network and less complex processing of the data. On the other hand, SON functions are usually self-contained entities which are provided without any specification of the internal algorithms, i.e., like black boxes. Hence, the function itself may be the only entity which can directly detect a problem during the execution of the algorithm.

Usually, a SON function is designed for the detection of a specific problem based on KPIs from the network and the determination and execution of corrective actions in form of changes of network parameters. Nevertheless, it is often able to also detect problematic situations that it cannot correct by itself. Thereby, state-based detection approaches should be preferred because they are less complex. In that way, a SON function might detect a problem from anomalous PM data from a NE that is not foreseen by the algorithm so that it runs into an exception. An example for such an error state is when the MRO function monitors a lot of too late and too early HOs or when it attempts to set the value of an HO parameter outside a limit defined by the operator.

However, in some cases history-based detection methods may also be possible to use. A SON function can learn the effects of network parameter changes on network performance and, if the reaction differs significantly from the learned behavior, it raises an alarm. The reason for a significant deviation from the normal network behavior can be numerous, e.g., a severe hardware fault.

The alarming component extends a SON function by allowing it to inform the SONOT function about an issue through an alarm. However, if an alarm is raised, the SON function should try to continue its operation. This is because a SON function typically has a limited view on the network and, therefore, is not able to make informed decisions about the remedy of the problem.

SON Operational Troubleshooting It is not reasonable to assume that all SON functions in a mobile network will have an alarming component. It is more likely that there will always be a mixture of alarming-enabled and traditional SON functions in a real network. Hence, the SONOT function needs to monitor and analyze the behavior of all SON functions as well.

The monitoring component in the SONOT function has an advantage over the alarming components in the SON functions: a broad view on the network

regarding PM, FM, and CM data. SON functions often focus on single NEs and solely monitor data that is necessary for their task due to time and memory constraints. In contrast, the SONOT function can collect and accumulate a broad range of data that is not accessible to regular SON functions, e.g., the performance of a group of NEs in a specific area, system-level KPIs or Minimization of Drive Test (MDT) [6] data.

The disadvantage of external monitoring is that the SONOT usually has no information about the algorithm or the internal status of the SON function. Hence, its analysis must always be based on assumptions about the logic of the functions. If these are not correct then this can lead to false inferences and, consequently, false alarms or unnoticed problems. For instance, a continuously running CCO function might indicate a coverage hole produced by a broken NE. Conversely, if an Mobility Load Balancing (MLB) function is often executed, this does not necessarily indicate a problem because MLB is heavily dependent on user behavior which might change often. Therefore, the monitoring component needs to be configured for a concrete SON. As depicted in Figure 1, this configuration is given in form of a policy.

The SONOT function should employ complex, history-based approaches since the indirect identification of problems requires sophisticated, knowledge-based inference mechanisms. In this way, the monitoring component can detect oscillations produced by an ineffective SON function through a statistical, time-series analysis. However, notice that even complex detection approaches need to be configured for a specific SON. For example, oscillations can also be caused by a SON function that attempts to find an optimal value for a network parameter using some hill climbing algorithm.

2.2 Alarm Analysis and Remedy

The alarm resolver component performs an analysis of the alarms and determines suitable countermeasures. For example, the analysis of an CCO function alarm that indicates an unrecoverable coverage hole by the alarm resolver can produce an equipment failure as the root cause. As a result, the SONOT function blocks the execution of the CCO and triggers a self-healing Cell Outage Compensation (COC) function. For a comprehensive analysis and well-informed decision making, the SONOT function can draw on contextual information about the current status of the network. Thereby, it is possible to employ simple reasoning approaches like production or fuzzy rule systems, or sophisticated Artificial Intelligence (AI) systems like influence diagrams or planners [10].

Remedy Actions The actions that the alarm resolver might execute can be classified into three categories:

– *SON coordination actions* are directly interfering with the execution of SON functions. Examples are the blocking or preempting of the execution of a SON function, or the active requesting of the execution of a SON function. Thereby, it is also imaginable to not just request the execution of a single function but to carry out a complex workflow with several functions.

- *SON management actions* are changing the configuration of the SON system itself, e.g., by changing the configuration of the SON functions such that their objectives change.
- *Escalation actions* are triggered if the SON troubleshooting function cannot or should not perform an action. For instance, an alarm can be escalated to a human operator as a trouble ticket for further inspection. Then, the operator can decide for a remedy. By utilizing machine learning techniques, it is possible to extend the expert knowledge of the SON troubleshooting function based on the operator response.

Interaction with Coordination If the SONOT function requests the execution of a SON function, this request needs to be coordinated against other SON function requests [3]. Consequently, these requests need to by prioritized. A simple option is to give the SONOT requests maximum priority. However, this can result in unintended behavior. For instance, consider that the MRO function detects some coverage hole and sends an alarm to the alarm resolver which requests the execution of the CCO function. In parallel, a Cell Outage Detection (COD) function detects a severe outage of an NE and request the execution of a self-healing function. It is obvious that, in this case, the operator prefers the execution of the self-healing function.

In the following, an advantageous prioritization scheme proposal is derived. Therefore, consider an CCO(RET) function instance I_{RET} which performs CCO by adapting the Remote Electrical Tilt (RET). This instance has a priority P_{RET} and raises an alarm $A_{\mathrm{RET}\to\mathrm{TXP}}$. The alarm leads to the execution of a CCO(TXP) function instance I_{TXP} which performs CCO by adjusting the Transmission Power (TXP). It has the priority P_{TXP} and requests an adaptation of the TXP R_{TXP}. Since I_{RET} continues to run after raising the alarm (cf. Section 2.1), I_{RET} also requests a new RET value R_{RET} at the SON coordination function. Furthermore, imagine that an MRO function instance I_{MRO} with priority P_{MRO} requests an adjustment of the HO parameters R_{MRO}. The priorities are $P_{\mathrm{RET}} > P_{\mathrm{MRO}} > P_{\mathrm{TXP}}$, i.e., I_{RET} has the highest and I_{TXP} the lowest priority. In summary, there are three action request, R_{RET}, R_{TXP}, and R_{MRO}, and one alarm, $A_{\mathrm{RET}\to\mathrm{TXP}}$.

A first simple approach is to prioritize the alarm-triggered requests like regular ones, i.e., R_{TXP} has priority P_{TXP}. This does not require an adaptation of the SON coordination function. In this case, the coordination function triggers R_{RET} and blocks R_{TXP} since $P_{\mathrm{RET}} > P_{\mathrm{TXP}}$, though, it is obvious that it should be the other way around because I_{RET} is actually in a problematic state. Therefore, the coordination function additionally needs to block the requests by alarming functions, i.e., R_{RET} would not be considered for coordination. Consequently, the request R_{MRO} would be triggered since $P_{\mathrm{MRO}} > P_{\mathrm{TXP}}$. However, since the highly important I_{RET} encountered a problem which blocked it from satisfying its objective, it seems reasonable that the solution to this problem, R_{TXP}, should actually be executed. So, it is furthermore required to adapt the priority of the request by alarm-triggered functions such that their priority is the maximum of the priority of the alarming and the alarm-triggered function,

i.e., R_{TXP} gets the priority $\max(P_{TXP}, P_{RET})$. As a result, R_{TXP} would be triggered since $P_{\mathrm{TXP}} > P_{\mathrm{MRO}}$. This finally leads to the desired behavior.

3 Evaluation

In order to evaluate the behavior of the SONOT concept, an extended simulation environment based on [14] has been developed. It consists of a state-of-the-art LTE network simulator which simulates an LTE macro cell network with an area of 50 km^2 and 1500 uniformly distributed mobile users moving randomly around at 6 km/h. The simulation is performed in periodic time-slices, called rounds, of approximately 100 minutes in real time.

The following three SON functions are considered:

- *MRO function*: Its objective is to minimize HO problems, e.g., too late and too early HOs [6], by altering the HO offset parameter between a pair of cells.
- *CCO(RET) function*: As a CCO function, it aims to maximize a cell's coverage and capacity. It adapts the antenna tilt in order to minimize interference.
- *CCO(TXP) function*: It is also a CCO function which adapts the transmission power.

3.1 SON Coordination

The SON coordination function implements an adaptation of the pre-action batch coordination concept with dynamic priorities [9]. In this concept, the priorities represent the importance of a SON function for the overall system performance, i.e., the higher the priority of a function, the more important it is. Every SON function instance, i.e., a running SON function in a concrete area, has an assigned bucket and dynamic priority. The bucket initially contains a number of tokens which are reduced every time a request by the function instance is triggered and increased if a request is rejected. If the bucket gets empty, the priority is set to minimum. However, the priority can be increased again if a request is rejected. In sum, the SON coordination function collects all requests by SON functions with non-empty buckets in a round, and computes and resolves the conflicts between them by accepting the requests with the highest priority. For simplicity, the SON coordination function has been configured to consider all requested actions concerning the same cell as conflicting. In other words, only one network parameter per cell can be changed in each simulation round.

3.2 SON Operational Troubleshooting Implementation

Alarm Generation The alarming component of each SON function employs a network parameter limit check which generates an alarm if the SON function algorithm wants to cross it. This state-based problem detection is motivated by the fact that the function attempts to achieve it objective but is prevented by its configuration. However, notice that the reason for this might not be a wrong

```
1  rule "RETAlarm+TXPAlarm->Reset"
   when
3   $alarm: RETAlarm()
    $txp: SonFuncInst() from new SonFuncInst(TXP(), $alarm.targetCell)
5   eval(context.isAlarmInCurrentRound($txp))
    $neighbor: Cell() from context.getNeighborsOfCell($alarm.targetCell)
7   $reset: SonFuncInst() from new SonFuncInst(SonFunction.RESET(), $neighbor)
   then
9   sfe.requestSonFunction($reset);
    sonco.addAlarm($alarm, $reset);
11 end
```

Listing 1. Exemplary rule of the policy for the alarm resolver

configuration but an environmental problem. For instance, the CCO(RET) function is configured with some upper and lower limit on the tilt of the antenna within which it tries to find an optimal value.

The SONOT function's monitoring component performs oscillation detection of a SON function's behavior. This history-based approach observes whether a function monopolizes the network by continuously adjusting network parameters. Furthermore, it applies exponential smoothing on the KPI data. With this history-based problem detection, they can make a prediction about the effectiveness of their proposed network parameter change: if a SON function does not manage to improve a KPI within a specific number of rounds, it generates an alarm.

Alarm Analysis and Remedy The alarm resolver is controlled by a policy which is a set of rules mapping an alarm in a specific operational context to a request for a SON function. Listing 1 shows a complex rule taken from the Scenario "Sleeping Cells" in Section 3.3 encoded in the Drools rule language [7]: if there are two alarms, one by the CCO(RET) function (Line 3) and one by the CCO(TXP) function(Line 5), then all neighbors of the cell are determined (Line 6) and the reset function is triggered for each of them (Line 9). Additionally, the SON coordination function is informed about the alarm (Line 10) in order to block the CCO(RET) function and set the priorities of the reset requests.

The definition of such rules needs to be well thought through. On the one hand, the system may face a deadlock if the mapping in the policy is circular. On the other hand, in case a function in a cell is triggered because of an alarm, its impact area [3] needs to be taken into account. More precisely, all other functions on neighboring cells which have an impact area overlapping with the impact area of the triggered function have to be blocked.

3.3 Simulation Results

The presented results in this section include the evaluation of three simulation scenarios. For each of them, the SONOT concept is compared with the batch SON coordination with dynamic priorities (cf. Section 3.1). In the former case,

the priority adaptation of the coordination mechanism is disabled, i.e., the request having the highest initial priority will be always triggered and the SONOT function has to block the execution of SON functions. In the latter setup, the SONOT function is disabled and, so, the SON coordination performs coordination with dynamic priorities.

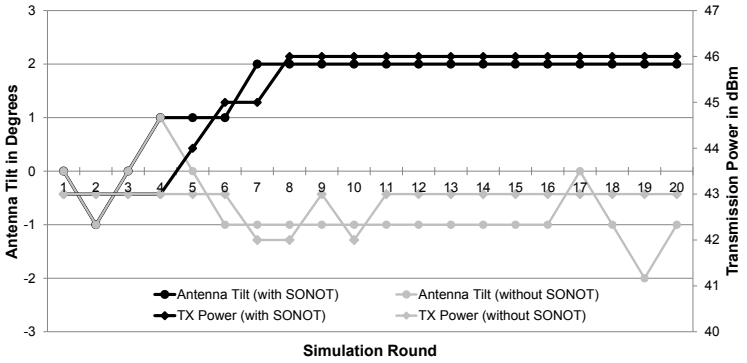
Scenario "Inability of CCO(RET)" In the first scenario, it can be observed that the CCO(RET) function detects a coverage hole, tries to close it and, due to its inability to achieve its objective, an alarm is generated which triggers the execution of the CCO(TXP) function. Such unexpected changes in coverage may occur due to several reasons [6], e.g., the demolition or construction of buildings, the insertion or deletion of base stations, or the misconfiguration of a cell during network planning.

In order to reproduce such a scenario, a coverage hole is manually created before the start of the simulation. Furthermore, the order of the SON functions regarding their priority P is $P_{\mathrm{RET}} > P_{\mathrm{TXP}} > P_{\mathrm{MRO}}$, i.e., CCO(RET) is the most important function and MRO the least important one. Regarding the resolution policy, an CCO(RET) alarm is mapped to a request for CCO(TXP) function execution and a CCO(TXP) alarm is mapped to a request for MRO function execution.

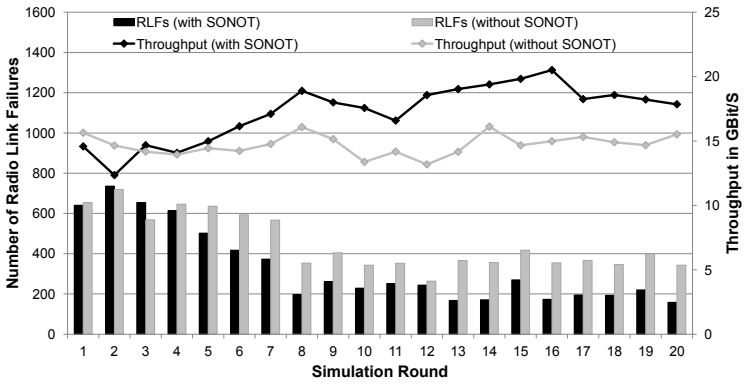
Figure 2(a) and Fig. 2(b) depict the changes in the network parameters and the interesting KPIs of a cell that has been affected by the coverage problem for both the SONOT approach and the coordination with dynamic priorities. In the former case, the CCO(RET) function first performs several adjustments to the antenna tilt (up to round 4) which do not lead to a significant improvement of the Radio Link Failures (RLFs) throughput, though. As soon as the SONOT functions detects this it sends an alarm that leads to the trigger of the CCO(TXP) function. As a result, the transmission power is increased twice which improves the RLFs by a factor of 1.5 and the throughput by a factor of 1.1.

However, in round 7 the CCO(RET) function reaches the antenna tilt limit. This results in the generation of an alarm by the function itself which results in triggering CCO(TXP) This final parameter change improves the cell performance such that the objectives of both functions are achieved and they stay inactive until the end of the experiment. Hence, the MRO function gets its turn to adapt the HO parameters which is not shown.

In contrast to that, the coordination mechanism with dynamic priorities is not able to improve the performance of the cell in such a manner. This is mainly caused by the coordination mechanism itself. Each SON function is allowed to try to improve the performance on its own for 6 rounds until the bucket is empty: first the CCO(RET), then the CCO(TXP), and finally the MRO function caused by the priorities. As can be seen, this coordination does not allow blocking a function if it is not doing any good leading to the case that the function actually decreases performance. This trend can be also be seen in the performance measurements from all cells affected by the coverage hole as shown in Figure 3(a).



(a) Single cell's network parameters

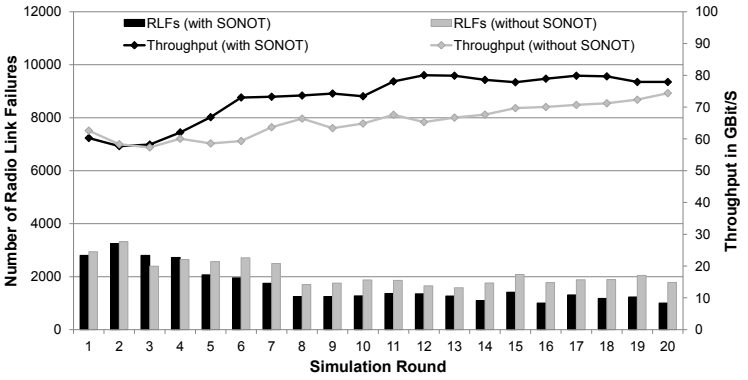


(b) Single cell's performance

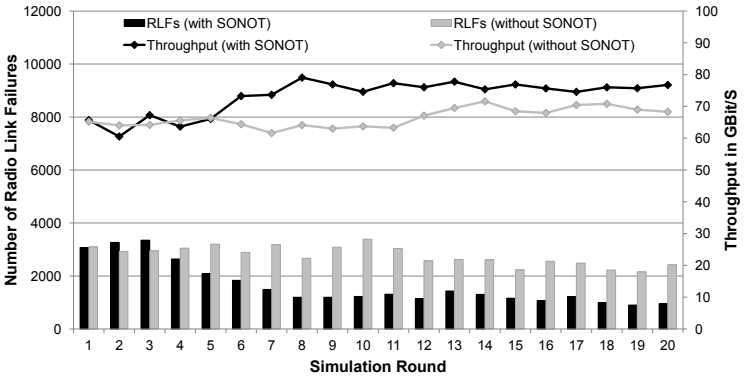
Fig. 2. Comparison between SONOT function and SON coordination with dynamic priorities for scenario "Inability of CCO(RET)"

Scenario "Inability of MRO" In the second scenario, the MRO function spots a coverage problem and sends an alarm so that one of the other two functions can resolve it. In this scenario, the same coverage problem as before is induced, but the priority P of the SON functions is set to $P_{\mathrm{MRO}} > P_{\mathrm{TXP}} > P_{\mathrm{RET}}$. The resolution policy describes that an MRO alarm is mapped to a request for CCO(TXP) function execution and a CCO(TXP) alarm is mapped to a request for CCO(RET) function execution.

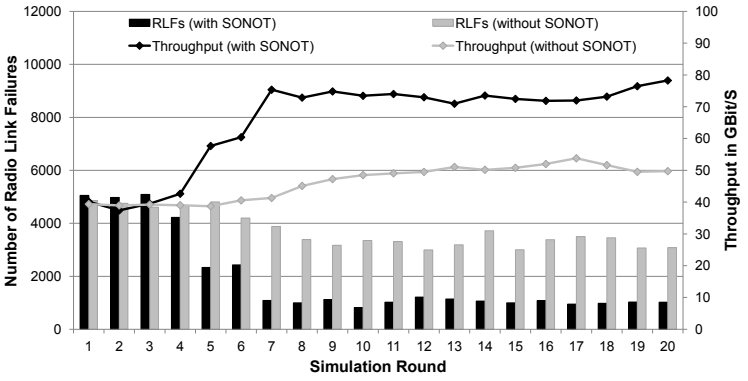
Figure 3(b) depicts the performance of the cell around the coverage hole for of the SONOT-enabled coordination and the SON coordination with dynamic priorities. In the first three simulation rounds, the RLFs and the throughput stay constant and do not improve although the MRO function is always changing the HO offsets. In the SONOT case, however, the SONOT function detects in round 4 that the performance drop is not related to mobility and generates an alarm.



(a) Scenario "Inability of CCO(RET)"



(b) Scenario "Inability of MRO"



(c) Scenario "Sleeping Cells"

Fig. 3. Overall performance with the SONOT function and SON coordination with dynamic priorities

This triggers the CCO(TXP) function which changes the transmission power similar to the previous scenario, leading to a significant decrease of the RLFs and an increase of the throughput. In contrast to that, the coordination with dynamic priorities behaves very similar to the previous scenario and produces a non-optimal situation.

Scenario "Sleeping Cells" The third scenario investigates the case where both the CCO(RET) function and the CCO(TXP) function are not able to close a coverage hole caused by a sleeping cell and, therefore, an alarm is generated which triggers a reset function. Sleeping cells are a serious problem in mobile networks since they are performing poorly without generating any failure alarms [11]. Hence, they often remain undetected for hours or even days. Sleeping cells can be caused by software failures, in which case the remedy can be the reset of the cell's software configuration. In this scenario, two sleeping cells, cause by a software upgrade, produce a coverage hole. Moreover, the SON function priorities P are set as follows: $P_{\mathrm{RET}} > P_{\mathrm{TXP}} > P_{\mathrm{MRO}}$. The resolution policy triggers a reset function, i.e., the restoration of a previous software version, if an CCO(RET) alarm and a CCO(TXP) alarm occur together (cf. List. 1).

The results of this experiment are shown in Figure 3(c). In case the SONOT function is employed, the same positive impact on the performance as shown in the previous two scenarios can be observed. Since the CCO(RET) and the CCO(TXP) functions are not able to achieve their objective, an alarm is generated in round 3 which leads to the execution of the reset function that restores the sleeping cells. In the coordination with dynamic priorities case, the CCO(RET) and CCO(TXP) functions are continuously adapting the tilts and transmission powers of the neighbors of the two sleeping cells which does not lead to the desired effect.

4 Related Work

Within SON research, troubleshooting is traditionally considered in the self-healing area, i.e., the automatic detection, diagnosis, and recovery of faults of NEs [1]. Although self-healing approaches combine alarms from NEs with other fault indicators like abnormal KPI values in order to diagnose the root cause of a problem [12], they usually do not analyze the operational state of SON functions. However, there is some related work which identified the problem of troubleshooting a SON as a part of SON coordination.

In [9], the authors present a batch coordination approach with dynamic priorities which prevents the monopolization of the network by a SON function, i.e., that a SON function with a high priority permanently issues requests which block other functions. However, the reason for this abnormal behavior of the function is not particularly analyzed. Furthermore, it does not allow to distinguish between different problems one SON function might encounter and it does not allow triggering another SON function to solve a problem.

The Self-NET project provides a framework for the self-management of cognitive NEs by introducing a hierarchical architecture of cognitive managers [8]. Thereby, a low-level cognitive manager can delegate a problem to a higher-level manager if it does not match its local resolution rules. The UniverSelf project aims in a similar direction [13]. Thereby, the SON functions are able to alert a coordination block about a "situation where they are not able to fulfil the specified goals" [13]. If the problem cannot be solved by coordination, it is escalated to the human operator. The SOCRATES project developed, among others, an extensive SON coordination concept introducing several functions which together perform coordination [11]. Thereby, the concept describes a Guard function which detects undesired performance and behavior in the SON and, hence, compares to the monitoring component of the SONOT function. The authors particularly mention the detection of oscillations and poor absolute performance of the network, i.e., not achieved objectives. The Guard function notifies the Alignment function about detected problems, which determines suitable countermeasures, e.g., undo previous changes of network parameters, blocking of SON functions, or adapting the configuration of SON functions, by analyzing the cause of the problem. The three projects consider operational troubleshooting as a side note and, thus, do not provide further details, e.g., an implementation or case study.

The SEMAFOUR project aims at a unified self-management system and extensively works on a SON coordination concept [4]. Its SON coordination function is able to detect oscillations and frequent requests by functions which the authors infer to be due to a misconfiguration of the functions. Hence, the coordination can trigger a management component to adapt the configuration. However, due to its recent start, there are currently no further details.

5 Conclusion

This paper presented the details of a new operational troubleshooting approach for a Self-Organizing Network (SON). It allows the detection of situations in which a SON function is not able to achieve its objectives. This monitoring is performed by the SON functions themselves as well as the monitoring component of the SON Operational Troubleshooting (SONOT). This approach allows, on the one hand, the exploitation of the sophisticated detection methods employed by the SON functions and, on the other hand, the utilization of complex algorithms and network-wide data in the SONOT function. Based on the detected problems, the alarm resolver can determines possible countermeasures like the preemption and triggering of SON functions. In simulations it has been shown that the presented approach remarkably improves the network performance in terms of Key Performance Indicators (KPIs) like Radio Link Failures (RLFs) and cell throughput, and outperforms traditional coordination approaches like a batch-based coordination scheme with dynamic priorities.

In the future, further research is necessary for the development of improved methods for the detection of problematic situations, the diagnosis of problems,

and the determination of countermeasures. Particularly, it seems promising to make use of machine learning in order to enable a high degree of automation.

References

1. 3GPP: Telecommunication management; Fault Management; Part 1: 3G fault management requirements. Technical specification 32.111-1 v12.0.0, 3rd Generation Partnership Project (3GPP) (Jun 2013)
2. 3GPP: Telecommunication management; Self-Organizing Networks (SON) Policy Network Resource Model (NRM) Integration Reference Point (IRP); Information Service (IS). Technical specification 32.522 v11.7.0, 3rd Generation Partnership Project (3GPP) (Sep 2013)
3. Bandh, T.: Coordination of autonomic function execution in Self-Organizing Networks. Phd thesis, Technische Universität München (Apr 2013)
4. Ben Jemaa, S., Frenzel, C., Dario, G., et al.: Integrated SON Management - Requirements and Basic Concepts. Deliverable d5.1, SEMAFOUR Project (Dec 2013)
5. Frenzel, C., Tsvetkov, T., Sanneck, H., Bauer, B., Carle, G.: Detection and Resolution of Ineffective Function Behavior in Self-Organizing Networks. In: Proc. 15th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM 2014). Sydney, Australia (Jun 2014)
6. Hämäläinen, S., Sanneck, H., Sartori, C. (eds.): LTE Self-Organising Networks (SON): Network Management Automation for Operational Efficiency. John Wiley & Sons, Chichester, UK (Dec 2011)
7. JBoss Community: Drools Expert, `http://www.jboss.org/drools/drools-expert`
8. Kousaridas, A., Nguengang, G.: Final Report on Self-Management Artefacts. Deliverable d2.3, Self-NET Project (Apr 2010)
9. Romeikat, R., Sanneck, H., Bandh, T.: Efficient, Dynamic Coordination of Request Batches in C-SON Systems. In: Proc. IEEE 77th Vehicular Technology Conference (VTC Spring 2013). pp. 1–6. Dresden, Germany (Jun 2013)
10. Russell, S.J., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice Hall, Upper Saddle River, NJ, USA, 2 edn. (2003)
11. Schmelz, L.C., Amirijoo, M., Eisenblaetter, A., et al.: A Coordination Framework for Self-Organisation in LTE Networks. In: Proc. 12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops. pp. 193–200. Dublin, Ireland (May 2011)
12. Szilágyi, P., Nováczki, S.: An Automatic Detection and Diagnosis Framework for Mobile Communication Systems. IEEE Transactions on Network and Service Management 9(2), 184–197 (Jun 2012)
13. Tsagkaris, K., Galani, A., Koutsouris, N., et al.: Unified Management Framework (UMF) Specifications Release 3. Deliverable d2.4, UniverSelf Project (Nov 2013)
14. Tsvetkov, T., Nováczki, S., Sanneck, H., Carle, G.: A Post-Action Verification Approach for Automatic Configuration Parameter Changes in Self-Organizing Networks. In: 6th International Conference on Mobile Networks and Management (MONAMI 2014). Wuerzburg, Germany (Sep 2014)