A Post-Action Verification Approach for Automatic Configuration Parameter Changes in Self-Organizing Networks

Tsvetko Tsvetkov[1], Szabolcs Nováczki[2], Henning Sanneck[3], and Georg Carle[1]

[1] Department of Computer Science, Technische Universität München
{tsvetko.tsvetkov,carle}@in.tum.de
[2] Nokia, Budapest, Hungary
szabolcs.novaczki@nsn.com
[3] Nokia, Munich, Germany
henning.sanneck@nsn.com

Abstract. In a mobile Self-Organizing Network (SON) a SON coordinator is required to prevent the execution of conflicting SON function instances. Usually, such a coordinator is responsible for conflict prevention and resolution and does not consider the fact that the activity of SON function instances may induce an undesired network behavior, like a performance degradation. In this paper, we propose an approach for the verification of Configuration Management (CM) changes induced by the activity of function instances. We have developed the SON verification function which is triggered when CM change requests get acknowledged by a SON coordinator. It analyses the resulting Performance Management (PM) data and in the case it detects an undesired network behavior it sends a request to the coordinator to revert the changes responsible for that to happen. Furthermore, our function takes the impact area of a SON function instance into account to determine the scope of verification. It also takes the impact time of function instances that have been active into consideration as it tries to identify the CM changes that have possibly caused an undesired network behavior. Simulations have shown that the tight integration of our function with a SON coordinator provides a solution for overcoming such problems and improving the overall network performance.

Key words: Self-Organizing Networks, Long Term Evolution, SON Verification, SON Coordination

1 Introduction

SONs are seen today as a key enabler for automated network management in next generation mobile communication networks such as Long Term Evolution (LTE) and LTE-Advanced. SON areas include self-configuration, self-optimization and self-healing [1]. The first area typically focuses on the initial configuration and auto-connectivity of newly deployed Network Elements (NEs). The second one targets the optimal operation of the network. A network enabled for

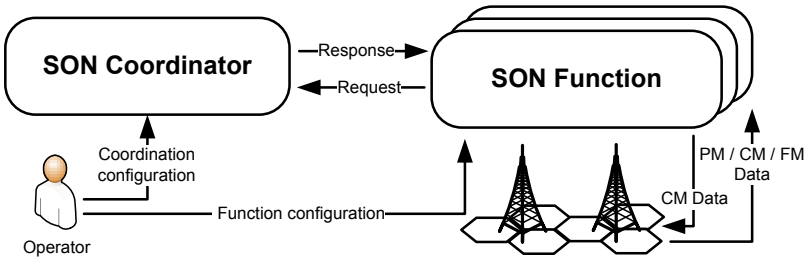

Fig. 1. Overview of a SON

self-optimization automatically adapts network parameters which should lead to improved robustness, reliability and throughput. The third area, self-healing, is responsible for fault detection and resolution caused, for example, by malfunctioning hardware or faulty software.

A SON-enabled network is managed by a set of autonomous functions performing specific Network Management (NM) tasks, as shown in Fig. 1. These SON functions are often designed as control loops which monitor PM and Fault Management (FM) data, and based on their objectives perform changes of CM parameters. Usually, a SON function's objective is given by the operator through a function's configuration [2]. For instance, the objective of the Mobility Robustness Optimization (MRO) function is to keep the rate of Handover (HO) failures below a given threshold.

Since SON function instances may perform changes to network configuration parameters during their operation, a SON coordinator is required to reject the requests which would cause or engage in conflicts and allow those which would guarantee a flawless network operation. Usually, such type of coordination is referred to as *pre-action SON coordination* and is based on rules used to anticipate and avoid known conflicts between SON function instances. Extensive research has led to several coordination approaches which can be performed either at design-time or at run-time [3–6].

Nevertheless, approved network configuration changes may not necessarily lead to the performance targeted by the corresponding SON functions or by the human operator himself. A SON coordinator does not consider the fact that the actions of a large number of deployed SON functions may cause an undesired behavior [4]. Usually, a SON coordinator is designed for conflict prevention and resolution between SON function instances, e.g., taking care that two instances are not modifying the same CM parameter on the same cell. However, it does not observe whether approved changes have had a negative impact on the network performance.

A possible solution is to employ an anomaly detection and diagnosis framework, as defined in [7, 8]. The purpose of such a mechanism is to detect an undesired network behavior, perform root cause analysis and provide the corresponding corrective action. Such a system includes an anomaly detector that learns the faultless behavior of the network. The gained knowledge is used at

a later point in time as a basis of comparison to identify significant deviations from the usual behavior. There are several ways of how the required performance data can be collected. For instance, each NE can monitor its own operation by measuring several types of performance indicators and upload the result to the Operations Support System (OSS) database. This stored data can be then fed into the anomaly detector. In order to provide the corrective action the diagnosis part may learn the impact of different faults on the performance indicators. For example, it may employ a scoring system that rewards a given action if it has had a positive effect on the network.

Inspired by the ideas of anomaly detection and diagnosis we have developed the SON verification function. The purpose of our function is to assess the impact of SON-induced CM changes and provide the corrective action in case they have caused an undesired or unusual network behavior. When action requests get acknowledged, the SON coordinator delegates the task of observing the performance impact of those changes to our verification function. The coordinator sends for this purpose a verification request message identifying the cells that have been reconfigured by a SON function instance. Furthermore, the coordinator informs our function about the area influenced by that reconfiguration as well as the time the change has an effect on other running function instances. Based on this information, our verification function determines where to look for an anomaly and find the changes responsible for an undesired behavior to occur. In this document, we classify the workflow of our function as *post-action verification*.

The rest of the paper is organized as follows. In Section 2 we give a general overview of coordination and verification in SON. In Section 3 we present our SON verification function, including all main building blocks. In Section 4 we outline the results from our experimental case study as well as include a description of the used simulation system. Our paper concludes with the related work and a summary.

2 Coordination and Verification in SON

In a SON-enabled network, functions are designed to work independently from each other. Moreover, a very common approach is to split a SON function into three major parts [3]: (1) a monitoring phase, (2) an algorithm execution phase, and (3) an action execution phase. During the monitoring phase a SON function instance observes certain Key Performance Indicators (KPIs) and collects information about the network, such as configuration changes and fault occurrences. After gathering the required amount of information, the algorithm part of a SON function instance may get triggered. Its purpose is to compute new CM parameters which are then applied during the action execution phase. In addition, there are two important properties of a SON function instance required for coordination: the impact time and impact area. As defined in [3], a SON function instance has to be considered by a SON coordinator for the whole time period it has an influence on the network. This includes not only the delay required to perform

measurements, run the algorithm and compute new configuration parameters, but also the time required to deploy the new configurations and the time until they become relevant for subsequently active functions. The impact area on the other side is the spatial scope within which a SON function instance modifies configuration parameters or takes measurements. More precisely, it contains the function area (area that is directly configured), the input area (area where the measurements are taken from), and the effect area (the area that contains the NEs that are affected by a CM change). Furthermore, it may include a safety margin which is an extension to the impact area. Its purpose is to provide a higher degree of protection against undesired effects.

The workflow of a post-action verification approach resembles the one of a SON function. It begins with the analysis of PM data required for the detection of significant performance changes. This phase can be represented as the monitoring phase of a SON function. It continues by identifying the affected NEs and suggesting the corrective configuration action. This part can be classified as the algorithm phase. The last phase, namely the action execution, takes place when the selected CM changes are enforced on the corresponding NEs. It should be noted, though, that unlike the CM configuration assembled by typical SON function, the configuration here is constructed by taking past CM settings. In this paper, we call such an action a *CM undo operation*.

However, having such a workflow immediately raises the question about the coordination, more precisely, the selection of the impact area and time. In general, a SON function instance that is not properly coordinated (e.g., due to an inappropriate selection of an impact area) may lead to configuration and measurement conflicts. As stated in [6], this can further cause undesired network behavior like performance drops. For instance, if two functions are trying to modify the same CM parameters at a given cell at the same time, configurations can get overwritten. The first function instance can simply modify the setting of the parameter that is considered to be unchanged by the second one. In fact, even if two function instances are not adjusting the same parameters, they might still be in a logical, but not direct conflict. If, for example, the Coverage and Capacity Optimization (CCO) function modifies the antenna tilt, the cell border changes physically which means that the received signal quality changes as well. Obviously, this affects the HO performance of neighboring cells which is monitored by a function like MRO.

An alternative post-action verification approach is to employ SON functions that track their own CM changes and trigger an undo in case they have caused an undesired network behavior. However, a SON function following such an approach suffers from one major weakness. In a SON-enabled network where several SON functions are actively running, a single function has a rather limited view on the network to determine whether exactly its change is causing an abnormal behavior. The impact of each function's action on the environment simply depends upon the actions of other functions. For instance, an inappropriate change of the physical cell borders induced by CCO may negatively impact the HO performance and, therefore, the upcoming decisions taken by MRO as well. Since

SON functions do not exchange context information, there will be always an uncertainty when a function performs a CM undo on its own. Typically, functions are considered as black-boxes which means that no one except the vendor is able to perform changes to the function itself (e.g., adding an interface for such context information exchange).

3 Concept Overview

Our post-action verification approach involves a tight integration with SON coordination. The SON verification function we propose analyzes the network performance for acknowledged action requests of SON function instances. In case the activity of a given instance causes an undesired network behavior, our function requests a CM undo from the SON coordinator for the affected area. To achieve its task, though, the SON verification function makes use of four helper functions: (1) an anomaly level, (2) a cell level, (3) an area resolver, and (4) an area analyzer function. The anomaly level function allows us to differentiate between normal and abnormal cell KPI values. The cell level function creates an overall performance metric of individual cells. The area resolver function defines the spatial scope we are going to observe for anomalies. The area analyzer function determines whether the cells within that scope are showing abnormal behavior, identifies the responsible CM changes for that to happen and sends a request to the SON coordinator to undo these changes. In the following, we are going to describe each function in detail and provide information of how they interact with each other.

3.1 Anomaly Level Function

Just monitoring a given cell KPI and observing whether it is above or below a threshold is not sufficient to determine whether it is showing anomalous values. Usually, a supervised anomaly detection technique requires the training and computation of a classifier that allows us to differentiate between a normal and abnormal state. If we take the KPI terminology used in this paper, such a method would allow us to compute a reference state from which a given cell KPI may deviate. In this way, we can analyze whether a given KPI data set is conform to an expected pattern or not.

The anomaly level function presented in this section is responsible to calculate this difference. Its output is a *KPI anomaly level* which depicts the deviation of a KPI from its expectation. To do so, we standardize a KPI dataset by taking the z-score of each point. A z-score is a measure of how many standard deviations a data point is away from the mean of a given KPI data set. Any data point that has a z-score lower, for example, than -2 or higher than 2 is an outlier, and likely to be an anomaly. The actual process of computing an anomaly level consists of two steps.

First, we collect samples $X_1, \ldots, X_t$ during the *verification training* phase for each KPI. Here, we use t to mark a training period. Depending on the granularity

at which we are able to get PM data from the network, a training period may correspond to an hour, a day, a week and so on. Second, we compute the z-score for each KPI sample $X_1, \ldots, X_t, X_{t+1}$. Note that X_{t+1} corresponds to the KPI sample from the current (non-training) sampling period. Let us give an example of how this may look like when we observe the Handover Success Rate (HOSR) for a given cell. Suppose that a cell has reported a success rate of 98.1 %, 97.6 %, and 98.5 % during the training phase. Furthermore, let us assume 95.2 % is the result from the current sampling period. The normalized result of all four samples would be 0.46, 0.21, 0.78, and -1.46. The HOSR anomaly level equals to -1.46, which is the z-score of the current sampling period.

Furthermore, it should be noted that during the verification training phase our verification function collects KPI data and does not verify any CM changes. It is of high importance for the verification function to be supplied with faultless data during this phase, i.e., the network must show an expected behavior.

Research has shown that there are several other ways of designing an anomaly level function. For instance, we may use a two-sample Kolmogorov-Smirnov test to compare the distributions of two sets of KPI samples [8]. Another example is the approach followed in [9] where an ensemble method is suggested to calculate KPI degradation levels.

3.2 Cell Level Function

The definition of KPI anomaly levels is not sufficient for depicting the state of any part of the network. The question that arises here is how we can use the result provided by the anomaly function so we can compute an overall performance metric of individual cells. In addition, we may desire to test a given cell for different anomaly types. For instance, we may wish to take only HO related KPI anomaly levels into account so we can assess the impact of a CM change induced by the MRO function.

To be able to provide such a flexibility, our verification approach makes use of a *cell level function*, denoted as φ. This function is responsible for the aggregation of the KPI anomaly levels of a single cell. In this paper, we call such an aggregation a *cell level*. For a given set K of KPIs and an anomaly level function ρ, we multiply the resulting KPI anomaly levels with a weighting factor α. The sum of the weighted anomaly levels corresponds to the cell level.

$$\varphi\left(K, \rho\right) = \sum_{k \in K} \alpha_k \rho(k) \tag{1}$$

3.3 Area Resolver Function

A cell level function is an indicator for abnormal network behavior of a cell. However, such a function becomes useful only when we know when and where to apply it. In other words, we need a mechanism that allows us to assess the impacts of SON-induced CM changes. For this reason, we have defined the *area*

resolver function that identifies the cell or a set of cells affected by a (set of) CM change(s). The output of this function is a *target tuple* (Σ, Ω). It consists of a set Σ that includes the cells that have been reconfigured by a SON function instance and a set of cells Ω that have been possibly influenced by that reconfiguration process. In this paper, we call Σ the *CM change base* and Ω the *CM change extension area*. Altogether (i.e., $\Sigma \cup \Omega$) they compose the *verification area* V which is the spatial scope we observe for anomalies.

Our area resolver function performs the computation based on the impact area of the SON function instance whose activity has triggered the verification process. As mentioned in [3], the impact area of a SON function instance provides us information about which cells are affected after its execution. In a similar manner as described in [10], we compute the CM change base by taking the function area. We see the cells that have been reconfigured by a SON function instance as most prone for experiencing anomalies. Furthermore, we compute the CM change extension area by taking the effect area and the safety margin. The main reason why we consider the effect area is because it includes all cells that are supposed to experience side-effects after the execution of a SON function instance. For instance, the load of a cell may change if the transmission power of a neighboring cell has been adjusted. However, the effect area can differ from its original definition. For example, due to an increased network density the effect area can be much larger than assumed. This is why we take the safety margin as well.

3.4 Area Analyzer Function

So far, we have described how our verification function assesses anomalous changes in performance and how it determines the area affected by a particular CM change. However, we did not mention how an undesired network behavior is actually detected and how the corrective action is provided. These two tasks are accomplished by the *area analyzer function* δ as follows.

First, it evaluates whether a given target tuple is showing an anomalous behavior. For each cell within the verification area it applies the cell level function φ and observes whether the result falls within the acceptable range defined by c_{min} and c_{max}. A tuple is considered as such when the following condition is met: $\exists v \in V \colon \delta\left(v, \varphi\right) \rightarrow (-\infty, c_{min}] \cup [c_{max}, \infty)$. Should this be the case, it goes to the second step: triggering a CM undo for the CM change base. Such an operation can be triggered when we have only one verification area to observe. However, being able to instantly verify the action of each SON function instance is rather ambitious. For example, the SON coordinator and our verification function may not placed at the same level, e.g., the latter one is located at the NM level whereas the coordinator is settled at the Domain Management (DM) level, several verification requests can get aggregated and sent at once. As a result, we can face the situation where we have overlapping verification areas, i.e., we have at least one cell that is part of two target tuples. In case this cell is the one that is experiencing anomalous behavior, i.e., it has triggered the creation of both target tuples, we cannot simply undo the changes at both CM change

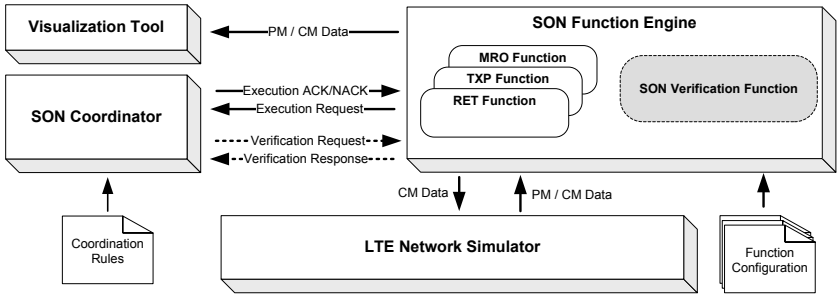

Fig. 2. S3 Overview

bases. The induced reconfiguration at one of the CM changes bases might have had a positive impact, i.e., it is not responsible for the degradation of that cell. In our concept we call such a situation a *verification collision*. Our proposal is to stepwise revert the changes made by the SON function instances. We first start to undo the CM changes triggered by the instance whose impact time has been most recently completed. Then, we observe the impact of the undo operation. Should the result lead to an improvement but still indicate that the target tuple is showing anomalous behavior, we continue by undoing the changes made by the next function instance that has been most recently executed. The undo process terminates as soon as we fall again in the acceptable range, as defined by c_{min} and c_{max}.

4 Evaluation

In this section we present the results of our experimental case study with the presented SON verification approach. We also give an overview of the simulation system we use for evaluation.

4.1 SON Simulation System

To evaluate the behavior of the introduced concept, we have developed the SON Simulation System (S3). The corresponding structure is outlined in Fig. 2. Our system consists of a state of the art LTE radio network simulator which allows us to configure 12 Evolved NodeBs (eNBs) at run-time. The configuration itself is done by a set of SON functions as well as our SON verification function. All functions are running in a SON Function Engine (SFE) and are coordinated by using a priority-based SON coordinator. In addition, our system includes a visualization tool that depicts the state of the network.

LTE Network Simulator The simulator periodically computes and exports PM data for an LTE macro cell network, consisting of 12 eNBs spread over an area of 50 km^2. The simulation is performed in rounds each corresponding

to approximately 100 minutes in real time. At the beginning of a round, the simulator configures the network according to the CM parameters defined by the selected scenario. During a round, 1500 uniformly distributed mobile users follow a random walk mobility model and use the network. The constant bit rate requirement of each user is set to 256 kpbs. In addition, the speed of the users is set to 6 km/h. At the end of a round, the simulator computes the performance statistics for each cell which are forwarded to the SFE. Furthermore, the simulator makes use of the path loss radio propagation model. A HO happens immediately when a User Equipment (UE) crosses the hysteresis threshold of 2.0 dB. A Radio Link Failure (RLF) happens based on a signal-to-interference-plus-noise ratio comparison to a threshold of -6.0 dB.

SON Function Engine The SFE is a runtime environment for SON functions which handles their communication and configuration. Every time the LTE network simulator completes a round, i.e., it exports new PM data, the SFE triggers the monitoring phase of all SON functions. Should a CM change request be generated, it is immediately forwarded to the SON coordinator. Based on the coordinator's decision, the SFE deploys the requested CM parameter changes to the simulator. For all simulation test runs, we employ our verification function as well as three optimization functions: the MRO, Remote Electrical Tilt (RET), and Transmission Power (TXP) function. Note that the latter two are a special CCO type, as defined in [1]. The RET function adapts only the antenna tilt whereas TXP adjusts solely the transmission power within a cell.

Furthermore, an instance of MRO, RET and TXP is running on each cell in the network. The function and input area of every function instance is set to a single cell. The impact time of every instance is set to one simulation round.

SON Coordinator The used SON coordinator performs pre-action coordination [4] by employing the batch coordination concept with dynamic priorities, as defined in [6]. The concept is designed for batch processing of SON function requests. More precisely, every SON function instance has an assigned bucket and dynamic priority. The bucket initially contains a number of tokens that are reduced every time a request by the SON function instance is accepted and increased otherwise. In case the bucket gets empty, the priority is changed to minimum. The priority can be increased again if requests start being rejected. The coordinator collects all requests for a round, determines the conflicts and sends an Acknowledgment (Ack) for the requests with the highest priority and a Negative-Acknowledgment (Nack) for the others.

SON Verification Function In our system a sampling period corresponds to a simulation round. The total number of training periods equals to 70 rounds. The KPIs we take into account for computing a cell level are the Channel Quality Indicator (CQI) and the HOSR. The CQI is computed as the weighted harmonic mean of the CQI channel efficiency. The efficiency values are defined in [11]. Furthermore, the cell level function weights these two KPIs with a factor of 0.5. A cell is considered to be experiencing anomalous behavior if the cell level function returns a value that falls in the range of $(-\infty, -2.0]$.

4.2 Simulation Results

Each scenario consist of 5 test runs each of which is lasting 18 simulation rounds. The 70 training rounds have been recorded beforehand. Each test run starts with a standard setup, as defined by the network planning phase. Furthermore, we define the effect area of each function instance to include only the cell on which the instance is running, i.e., the effect area equals the function area. In addition, we allow only one CM parameter per cell to be changed at the same time, i.e., only one function instance is allowed to adjust the configuration of a cell during a simulation round. The initial function coordination priority P is set as follows: $P_{RET} > P_{TXP} > P_{MRO}$.

Scenario "Function Dependency" As stated in Section 2, a SON function instance is only able to partially perform verification on its own, i.e., monitoring whether its CM changes are causing an undesired network behavior. The RET and TXP functions employed by S3 are fitting perfectly well for recreating and evaluating such a scenario. Both functions are monitoring the same set of KPIs, having the same objective, but modifying different CM parameters. In other words, if TXP changes the transmission power in such a way that it induces a negative impact on the KPIs monitored by the RET function, the latter one may try to provide a corrective action which may not necessarily lead to an improvement. Moreover, since we employ a pre-action coordination mechanism that dynamically adapts the priority of the running function instances, a high-prioritized RET function may even prevent a low-prioritized TXP from undoing its change. The scenario presented in this section investigates the advantages of employing our SON verification function in such a situation. The used safety margin does not increase the impact area.

At round 5 of every test run, the TXP instances running on two neighboring cells decrease the transmission power in such a way that it negatively impacts the KPIs monitored by the RET function instances running on these cells as well. Figure 3 and 4 outline the aggregated result of the HOSR and CQI of the two cells as well as three of their direct neighbors. Up to round 5, the observed set of five cells is showing a normal and usual behavior which is also confirmed by the resulting cell level, as shown in Fig. 5. At the time where the two TXP instances make a wrong decision by decreasing the transmission power of the two cells, all five cells begin to experience an anomalous behavior. At this point, the advantage of employing our function can be seen. Instead of letting the functions slowly provide a solution, i.e., let RET and TXP adapt the coverage and the let MRO adjust the HO parameters, our function triggers a CM undo which returns the set of cells to the normal state experienced between round 1 and 4. The same observation can be made for another KPI, namely the RLFs, as shown in Fig. 6. The experienced level of RLFs before round 5 and after round 6 matches. Furthermore, another important observation can be made here. The SON system is not able to completely return the performance state experienced before round 5 if we disable our SON verification function. This is caused by the

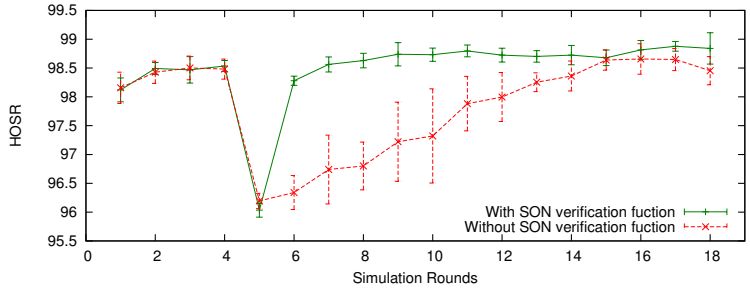

Fig. 3. HOSR Result for Scenario "Function Dependency"

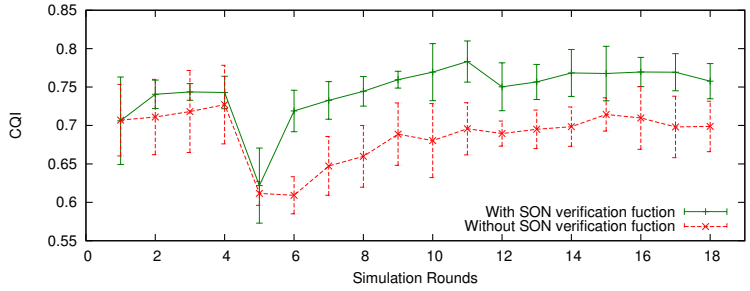

Fig. 4. CQI Result for Scenario "Function Dependency"

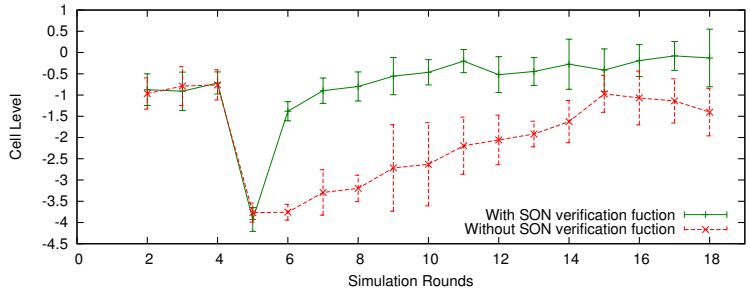

Fig. 5. Cell Level Result for Scenario "Function Dependency"

dynamic coordination mechanism. At some point in time, the coordinator starts to reject the requests of a function instance if it has frequently been executed.

Scenario "Verification Collision" The purpose of this scenario is to show the importance of resolving verification collisions. To do so, we configured the SON coordinator to send a verification request not immediately after it acknowledges an action request, but collect and send them in batches every odd-numbered round. To get a verification collision, though, we configure the safety margin of RET and TXP to include the direct neighbors as well. Furthermore, we force RET to change the tilt of one cell (denoted as A) and TXP to adjust the transmission power of another one (denoted as B). Note that we do the latter change

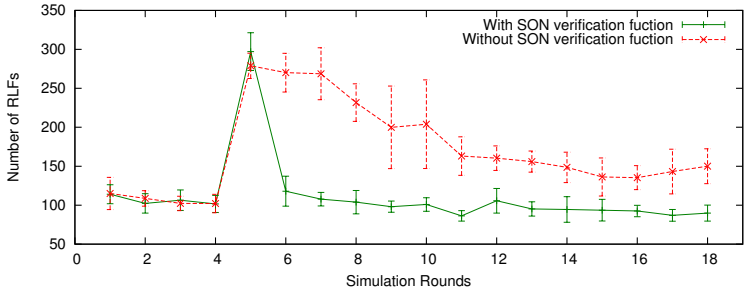

Fig. 6. RLF Result for Scenario "Function Dependency"

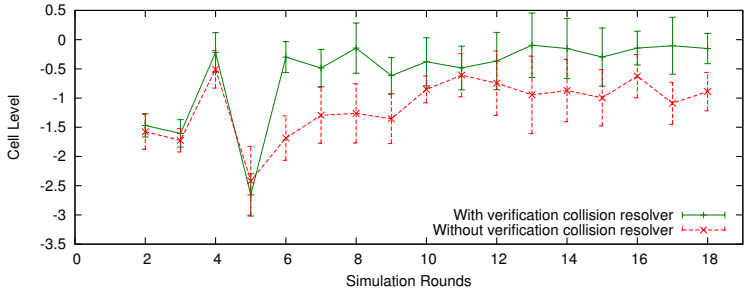

Fig. 7. Cell Level for Scenario "Verification Collision"

in a way that it negatively impacts the performance of cell C which is a common neighbor of cells A and B. The tilt change is done in round 4 on cell A whereas the transmission power change is triggered in round 5 on cell B.

Figure 7 shows the cell level at cell C, the cell that has triggered the creation of the two target tuples. If we simply disable the verification collision resolving capability, i.e., our function does not consider verification collisions, we will undo all changes made after simulation round 3. This would mean that we will revert a CM change that has had a positive impact (the tilt adjustment on cell A) and one that has had a negative influence (the power change on cell B) on cell C. As a result, cell C returns to the state before the tilt change was triggered which leads to a much lower cell level compared to the results from the test runs where we have enabled the verification collision resolver. In the latter case, our function reverts only the changes made by the TXP function instance running on cell B.

5 Related Work

Within the SOCRATES project [4] ideas have been developed about how undesired behavior can be detected and resolved in a SON-enabled network. The authors introduce a so-called Guard function whose purpose is to detect unexpected and undesirable network performance. They define two types of undesirable behavior: oscillations and unexpected absolute performance. Into the first

category usually fall CM parameter oscillations. The second category includes unexpected KPI combinations such as high Random Access Channel (RACH) rate and low carried traffic. The Guard function itself follows only the directive of the operator defined through policies, i.e., it requires knowledge about expected anomalies. In case such a behavior has been detected, the Guard function calls another function, the Alignment function, to take countermeasures. The latter one is divided into two sub-functions: an Arbitration and an Activation function. The first one is responsible for the detection and resolution of conflicting action execution requests. The second one is responsible for enforcing parameter changes, undoing them in case the Guard function detects an undesired behavior, and even suggesting SON function parameter changes.

Despite the given ideas, no detailed concepts are provided. Furthermore, the questions of how we should select the spatial scope that we are going to observe for anomalies, how we should determine the impact area when we are triggering a CM undo and how we should handle verification collisions remain unanswered.

In [12] a concept for operational troubleshooting-enabled SON coordination is given. If a SON function encounters a problem and has at the same time a high assigned priority by the SON coordinator, it may block other functions from being executed. Thus, it can monopolize the network which can result in an unusable SON that is trapped in a deadlock. In such a case a SON function might need assistance by another SON function. The approach proposed by the authors includes new SON troubleshooting function that analyses whether SON functions are able to achieve their objectives. If a function encounters a problem that hinders it from achieving its task, the troubleshooting function may trigger another one that might provide a solution.

6 Conclusion

In this paper we proposed an approach for verifying CM changes induced by the activity of SON function instances. We realized it by defining a new SON function: the SON verification function. It is requested by a SON coordinator to observe the impact of acknowledged CM change requests. The SON verification function determines whether a cell or set of cells is experiencing an undesired behavior and triggers a so-called CM undo request to the SON coordinator in case such a state can be confirmed. The purpose of this message is to revert the changes made by the SON function instance responsible for that to happen.

The results from our experimental case study show that the tight integration with a SON coordinator is quite advantageous when performing post-action verification. During the anomaly detection process the usage of the impact area of SON function instances proves to be a reliable starting point where to search for an unusual network behavior. Furthermore, the experiments show that if we stepwise undo CM changes based on another function property, namely the impact time, we are able to prevent positive (i.e., such having a positive effect on the network) CM changes from being undone.

Our future work will be devoted to further evaluation including more KPIs and more complex fault cases. We also plan to study alternative anomaly detection and diagnosis techniques. The link between several CM undo operations and performing the corrective action if they start repeating will be also one of our future research topics.

References

1. Hämäläinen, S., Sanneck, H., Sartori, C., eds.: LTE Self-Organising Networks (SON): Network Management Automation for Operational Efficiency. John Wiley & Sons, Chichester, UK (December 2011)
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) (September 2013)
3. Bandh, T.: Coordination of autonomic function execution in Self-Organizing Networks. Phd thesis, Technische Universität München (April 2013)
4. Kürner, T., Amirijoo, M., Balan, I., van den Berg, H., Eisenblätter, A., et al.: Final Report on Self-Organisation and its Implications in Wireless Access Networks. Deliverable d5.9, Self-Optimisation and self-ConfiguRATion in wirelEss networkS (SOCRATES) (January 2010)
5. Tsagkaris, K., Galani, A., Koutsouris, N., Demestichas, P., Bantouna, A., et al.: Unified Management Framework (UMF) Specifications Release 3. Deliverable d2.4, UniverSelf (November 2013)
6. Romeikat, R., Sanneck, H., Bandh, T.: Efficient , Dynamic Coordination of Request Batches in C-SON Systems. In: IEEE Veh. Technol. Conf. (VTC Spring 2013), Dresden, Germany (June 2013)
7. Szilágyi, P., Nováczki, S.: An Automatic Detection and Diagnosis Framework for Mobile Communication Systems. IEEE Trans. Netw. Serv. Manag. **9**(2) (June 2012) 184–197
8. Nováczki, S.: An Improved Anomaly Detection and Diagnosis Framework for Mobile Network Operators. In: 9th International Conference on Design of Reliable Communication Networks (DRCN 2013). (March 2013)
9. Ciocarlie, G., Lindqvist, U., Nitz, K., Nováczki, S., Sanneck, H.: On the Feasibility of Deploying Cell Anomaly Detection in Operational Cellular Networks. In: IEEE/IFIP Network Operations and Management Symposium (NOMS 2014). (May 2014)
10. Tsvetkov, T., Nováczki, S., Sanneck, H., Carle, G.: A Configuration Management Assessment Method for SON Verification. In: International Workshop on Self-Organizing Networks (IWSON 2014), Barcelona, Spain (August 2014)
11. 3GPP: Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer procedures. Technical specification 36.213 v12.1.0, 3rd Generation Partnership Project (3GPP) (March 2014)
12. Frenzel, C., Tsvetkov, T., Sanneck, H., Bauer, B., Carle, G.: Detection and Resolution of Ineffective Function Behavior in Self-Organizing Networks. In: IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks (WoWMoM 2014), Sydney, Australia (June 2014)