Efficient, Dynamic Coordination of Request Batches in C-SON Systems

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

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

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

***Abstract*—Self-Organizing Mobile Networks contain a potentially large number of SON function instances that need to be coordinated in order to achieve system-level operational goals. Many SON functions (and thus their coordination) will be realized in a centralized way (C-SON) due to the required integration with a (centralized) legacy OAM system. In such systems, many configuration requests ("batches") are treated in regular time intervals For SON functions, this may lead to undesired effects (monopolization by high priority functions and frequent rollbacks of configuration actions). This paper proposes a novel coordination approach combining an output buffer of pending configuration requests with dynamic priorities for different SON function types. A simulated network scenario exhibiting a specific optimization problem (closure of a coverage hole) is used to evaluate the proposed method. The method is able to avoid the undesired effects and approximate legacy workflows in network optimization while still keeping the autonomous characteristic of SON functions.**

Keywords—centralized self-organizing networks; SON coordination; run-time coordination; optimization workflows; configuration batches

I. INTRODUCTION

In traditional network operation and optimization, data of an entire network domain is "aligned" to the OAM system and then modified (optimized) within a single "offline" function. When new network element configurations have been computed, they are "rolled out" in the next step. The execution of this alignment / rollout cycle is planned and supported by a human operator.

Self-Organizing Network (SON [1]) functions realize an individual use case (rather than e.g., fulfilling self-optimization as a whole) respectively. This allows for the development of functions in a focused and thus quick way, independently of other individual function types. Furthermore, an operator can choose to deploy (and combine) a selection of individual functions coming from potentially various sources (different vendors). Apart from this selectivity, the separation into a set of functions allows to efficiently manage the individual functions over time.

SON functions have a generic function area associated with them. The function area comprises all network resources to be manipulated by a SON function in order to achieve the desired goal. SON function instances are the run-time instantiation of a SON function. They act on actual network resources in a certain area at a certain time. While the function area introduced above is generic (e.g., just implies that a function works on a(ny) pair of two adjacent cells), the function instance area is a concrete instantiation (e.g., a pair of the cells with IDs X and Y being adjacent to each other). Thus, SON function instances have a spatial scope (e.g., a set of cells, a set of network interfaces) and a temporal scope (activity in certain time intervals). Furthermore, SON function instances may get active at any time (e.g., triggered by a network measurement crossing a threshold) without any involvement by a human operator or a conventional OAM function. Thus, SON function instances run "inside" the OAM system and/or the respective network elements. In contrast to traditional network operation, the execution of SON function instances is individual and dynamic (i.e., not planned). Therefore SON function instances can have run-time interactions with each other ([2],[3], [4] Chapter 9). A negative interaction is called a *conflict*.

A simple solution for conflict avoidance (but bringing SON basically back to the traditional network operation and optimization cycle described above) is full serialization, i.e., executing SON functions one after another (in a fixed workflow) for an entire network domain. On the one hand, a full serialization is appealing as existing workflows could be implemented easily, but, on the other hand, it severely limits the flexibility and efficiency of the automated OAM system, compromising many of the desired SON benefits.

II. PRE-ACTION COORDINATION

An option to dynamically detect and prevent conflicts is *pre-action coordination* of the execution of SON function instances [5]. This approach aims at avoiding actions with a negative impact on the overall system at system run-time (rather than avoiding conflicts in the function design process or embedding coordination knowledge into the respective functions [6]). Fig. 1 shows an example of this category which aims at a virtual locking of network resources in order to avoid conflicts [7].

For the execution of such a coordination layer at runtime, some preparation at "design-time" is required. For any existing SON function, the following set of information needs to be analyzed, prepared, and configured:

- Generic *coordination logic*: expresses the conflict resolution strategy desired by the network operator, e.g., giving a certain type of function *priority* over an active function of another type.
- Generic *impact area:* comprises the generic function area and in addition the network resources on which the function has impacts on. The function instance only changes configurations within its function instance area, but these changes may impact other resources of the impact area (typically resources geographically or topologically adjacent to the function area).
- Generic *impact time:* expresses the mentioned temporal scope of a SON function. The impact time is defined for pairs of function types, i.e., the type of the considered function and each other potentially conflicting function type .

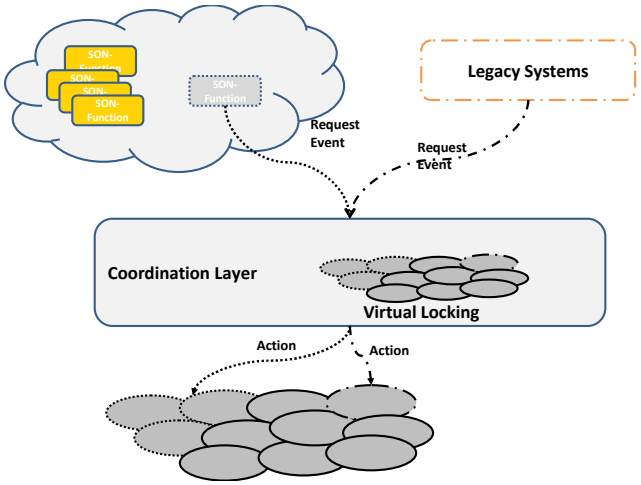

Fig. 1. Pre-action SON coordination

At run-time, the above information is evaluated by the coordination logic instantiated on the coordination layer (cf. Fig. 1) together with the actual system state (context and actual events, i.e., requests by SON function instances). Finally, a decision of how to proceed with a specific request is made:

- *Acknowledge (ACK):* The request can be executed on the network. For this purpose, the event is forwarded to the network and the contextual information on the resources within the impact area of the requested function are updated.
- *Reject (NACK):* The request information is deleted and the event is not forwarded to the network (thereby reacting to the activity of another function).
- *Rollback (RB):* The actions performed by the previous function instance (stored in the context) are undone and the contextual information about this function is deleted.

SON functions operate rather differently than traditional network operation and optimization. While operations should be as distributed as possible, all known SON functions have at least some centralized component, cf., e.g., Automatic Neighbor Relationship (ANR) setup. Some functions (like the tilt and power optimization functions presented later in this paper) need to acquire information on a number of cells and are thus more amenable to centralized ("C-SON") realization. Also, the transition phase moving from a (centralized) existing OAM system to a full deployment of SON advocates to add C-SON functions step-by-step to the legacy OAM system. Integrating C-SON functions and their coordination with such systems, however, means that the *Performance Management (PM) data, required as input to the SON functions, is available only in certain time intervals called Granularity Period (GP)* with a lower bound of typically 15 minutes. The partitioning of time into GPs is due to the way of downloading (rather than streaming) the PM data. The lower bound of the GP comes from the fact that the relative overhead for the download (in terms of network and processing overhead) is significantly increasing for further decreasing GPs.

This means that the longer the GP time interval is chosen, the higher the probability is that *many SON function instances become active at the same time*, because they may detect a condition, to which they need to respond to, in the data. Note that the actual points in time in the past when those conditions have actually happened may not at all overlap. When many SON function instances become active at the same time and subsequently aim to change network configuration parameters, *batches of SON requests* have to be processed. Due to the correlation in time (for a given network area) of the respective configuration change requests, there is also a *higher probability for conflicts, due to the temporal relationship of SON functions (impact time)* introduced above. Hence, it is crucial that a SON coordination scheme can effectively work on those request batches.

The described situation leads to the following two issues which are addressed in this paper:

- *Higher priority functions can monopolize the network* (in particular if they are greedy, i.e., never satisfied and therefore permanently issue change requests). This may lead to an undesired behavior deviating significantly from the operational goals of the network operator.
- *Rollbacks* on actual physical resources may occur (relatively frequently) for the following case: immediately after a change requested by a lower priority function instance has been accepted and deployed to the network, a higher priority one appears such that the lower priority one has to be rolled back. As certain configuration actions may incur significant costs (e.g., resetting a cell with service interruption), it may be costly to do an action, undo it, and finally do something else.

III. Efficient Coordination of Request Batches

This paper proposes an output buffer of pending requests, i.e., those already coordinated but not yet deployed to the network. Whenever there is no conflict, the coordinator replies with an acknowledgement, or it applies the coordination logic and sends the respective reply (acknowledge, reject, rollback).

Fig. 2 shows a flowchart of the batch coordination scheme (area shaded in grey). The links to a concrete coordination

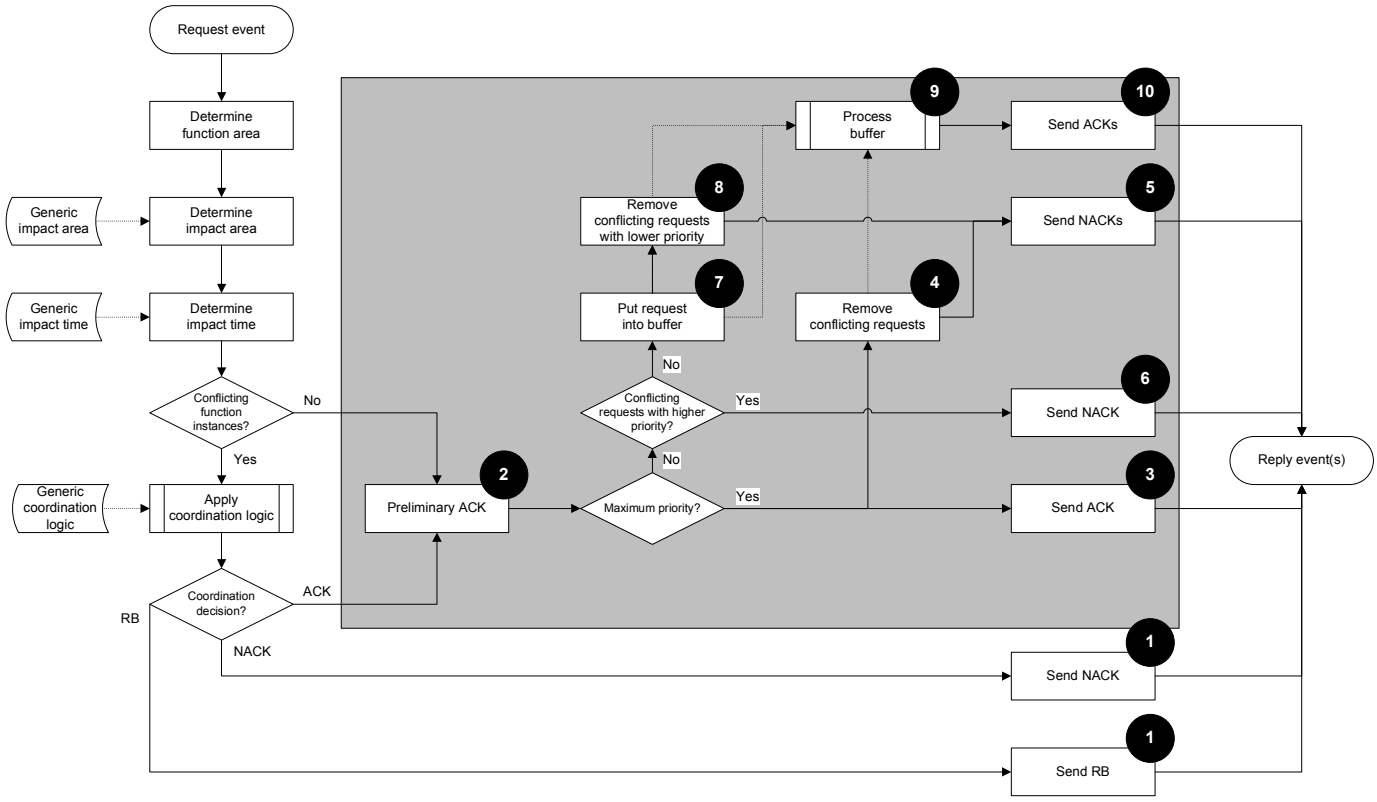

Fig. 2. Batch request coordination using an output buffer

scheme (described in [7]) are shown (non-shaded area), yet the concept is not dependent on a specific coordination scheme.

Any reply by the coordination scheme that is not an algorithm acknowledgement needs not be handled by the buffer and is directly forwarded to the network (1). Each acknowledged request is regarded as preliminary (2) and compared to the other requests in the output buffer.

If the preliminary request has the maximum priority possible at the respective target, it does not need to be considered further and is directly accepted and sent to the network (3). In addition, all requests for conflicting function instances in the buffer are removed and rejected (4,5). If the buffer already contains another request with a higher priority, the preliminary request is turned into a rejection (6). Otherwise, the preliminary request is put into the buffer (7) and all conflicting requests with a lower priority are removed (8) and rejected (5).

The buffer processing (9) is an independent process. It monitors the system behavior and detects the end of an input batch on the basis of the monitored request events. At the end of a batch, the remaining requests in the buffer are acknowledged and sent to the network (10).

IV. Dynamic Coordination

Fig. 3 shows an example for an improved request coordination scheme with a *dynamic adaptation of the priorities.* In this example, different function types for a target cell are each associated with a token bucket containing a number of tokens of a certain priority. For any request acknowledged after the buffer processing, a number of tokens are removed from the bucket; if requests are rejected, tokens are added to the bucket and the priority of the contained tokens is increased again.

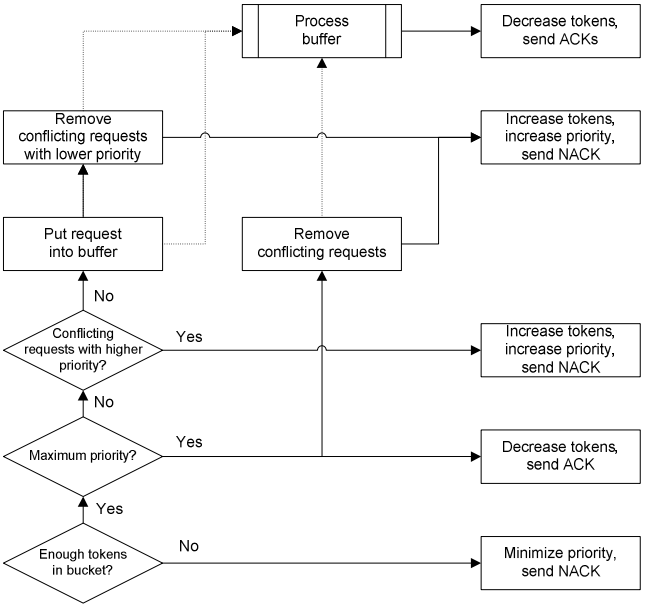

Fig. 3. Combined priority / token bucket coordination

If the token bucket is empty (i.e., a burst of requests of a single function type have been issued), the priority of the (to-be-added) tokens is reset to its respective minimum.

Initial priorities for SON functions are assigned based on the (observed / expected) positive impact of the respective SON function on the overall network performance. The algorithm that controls the adaptation of the priorities is thus highly specific to the involved SON functions and the respective operational goals.

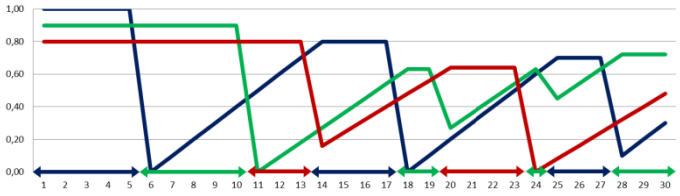

Fig. 4. Dynamic priorities for a target cell

Fig. 4 shows a typical development of dynamic priorities for three function types at a target cell that are requested in any GP. As a result, the function instances become active in an alternating fashion, depending on their initial priorities.

The concept of "dynamic coordination" is generic. However, the concept introduced in section III., allows to run the dynamic coordination efficiently on a batch of requests concurrently present in the buffer.

V. EVALUATION

In the following, the network scenario introduced in [8] exhibiting a coverage hole is evaluated using an experimental SON coordinator implementation, C-SON functions, and a radio network simulator. The experiment is partitioned into "rounds" corresponding to the acquisition of PM data for one GP and the deployment of a set of new configurations. Two SON function types are deployed in the system: an antenna tilt and a transmission power optimization function (characterized in Fig. 5-8 by their respective configuration parameter). Details on those functions can be found in [4], section 5.4. Several other SON functions, e.g., Mobility Robustness Optimization (MRO), Physical Cell ID (PCI) allocation, are also deployed and coordinated in the system but not considered further here wrt. the specific network scenario. KPIs characterizing SON function, and in particular (because the SON functions themselves are unchanged throughout the experiments) coordination performance, are characterized by (coverage-induced) Radio Link Failures (RLFs) and cell throughput, both aggregated over five cells adjacent to the coverage hole.

Fig. 5 shows the uncoordinated case as the baseline. From the configuration parameters, it can be seen that for all rounds of the experiment there is activity of functions instances of both types. For cell 2, the transmission power is significantly reduced which means (together with the further down-tilt) that a stable but undesirable operating point has been reached. The coverage area of cell 2 has been reduced and the coverage hole has not been closed (hence RLFs remain high and the throughput is basically unchanged) clearly showing that uncoordinated concurrent changes have lead to a detrimental effect.

Fig. 6 shows the coordinated case (without the output buffer) where each SON function type is associated with priority (tilt changes having higher priority than power changes). It can be seen that permanent activity of tilt optimization instances results in no activity for the power changes (i.e., all request are rejected; cf. the "monopolization" problem introduced in Section II). Again, the coverage hole is not closed, though for another reason as in the case before, because the tilt optimization function on its own is not able to solve the problem completely. No real improvement in performance can be achieved.

Fig. 7 shows a fixed workflow where tilt changes are done for 10 rounds at first, then transmission power is optimized for 10 rounds, and finally other SON functions (e.g., MRO) are admitted. The coverage hole can be closed and a corresponding significant improvement of both the RLFs and the throughput is depicted.

Finally, Fig. 8 shows the dynamic coordination case. As it can be seen, all functions are active all the time, though (through the dynamic priority setting) more tilt changes are accepted initially.

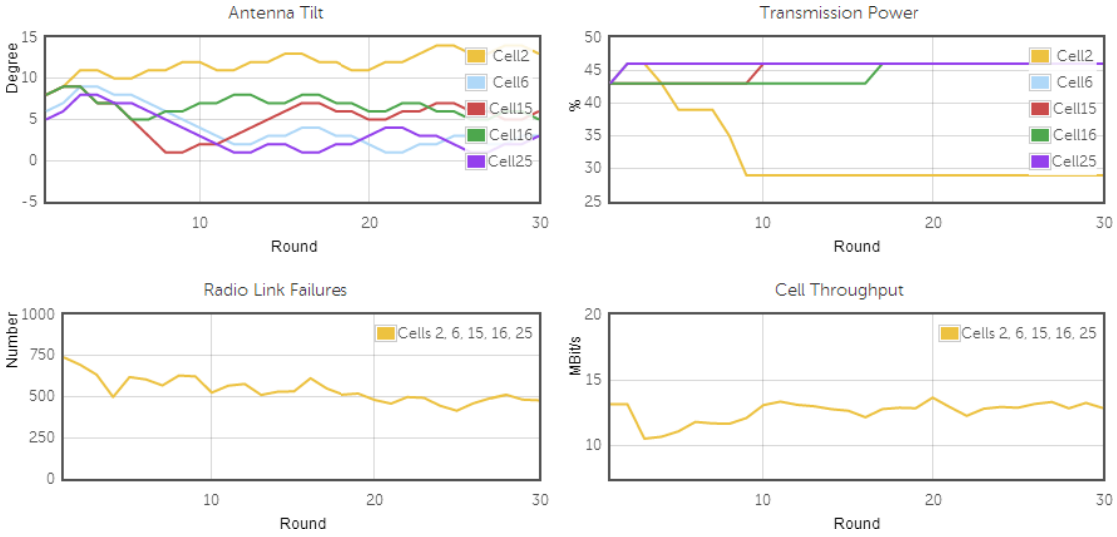

Fig. 5. Uncoordinated case

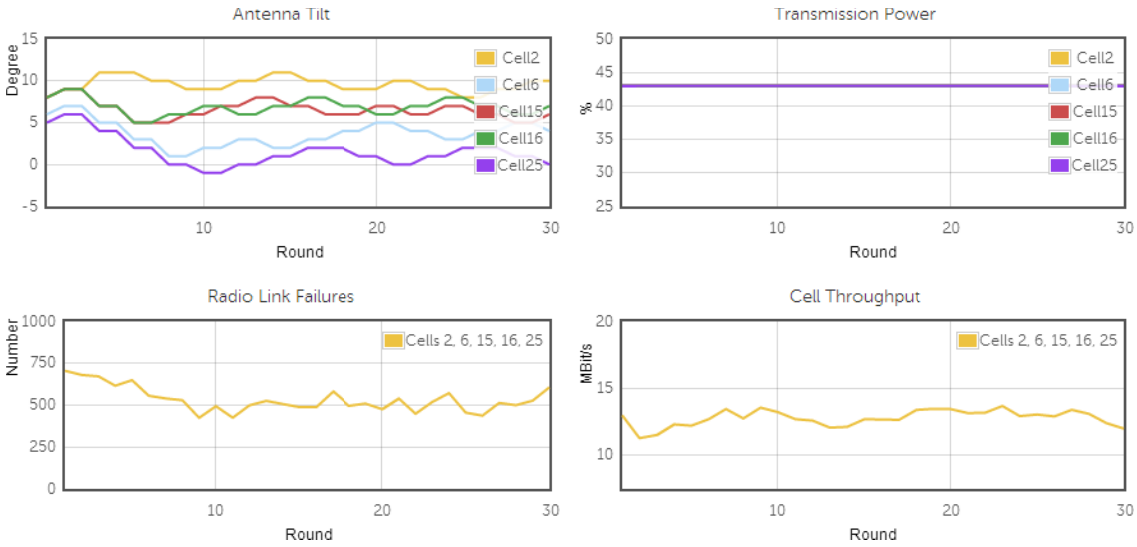

Fig. 6. Coordinated case with fixed priorities

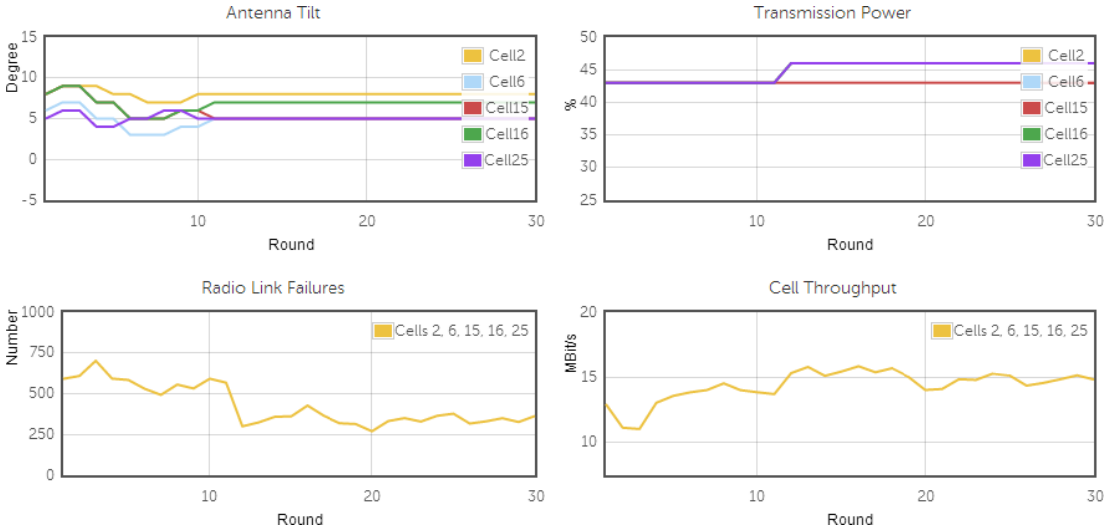

Fig. 7. Coordinated case with fixed workflow

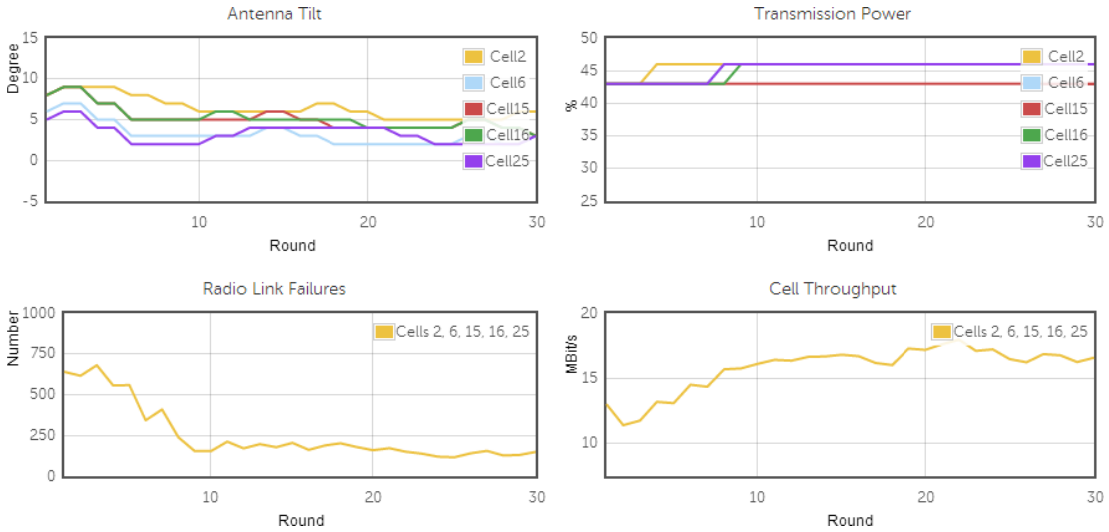

Fig. 8. Coordinated case with dynamic priorities

This gradually shifts over time to power optimization (and later to other SON function) changes. Here, we see the best performance of all cases as the SON function instances can act autonomously according to the detected conditions. On the other hand, the logical steps implemented via the fixed workflow of Fig. 7 can be approximated through the priorities while at the same time avoiding the monopolization visible in Fig. 6. Moreover, this method is more flexible and automated than a fixed workflow as it adapts to the actual network conditions.

Fig. 9 shows the comparison of an event-based scheme with batch input excluding (a) and including (b) batch request coordination. It can be seen that the issue of accepting and immediately rolling back a request can be avoided through the output buffer and its management Thus, efficiency can be improved, replacing an actual rollback with a remove action from the buffer (cf. the "rollback" problem identified in section II).

Comparing the experiment of Fig. 6 (coordination without buffer) with Fig. 8 (dynamic coordination with buffer), the number of rollbacks that can be avoided for the experiment is 605 (that amounts to 24.5 % of the total number of configuration actions). Even with the proposed output buffer, there may be cases where the preemption (rollback) decision is made, when the output buffer has just been processed, i.e., a rollback on the actual physical resource is required (yet due to the specific settings of GP and impact time, such cases do not occur in the experiment).

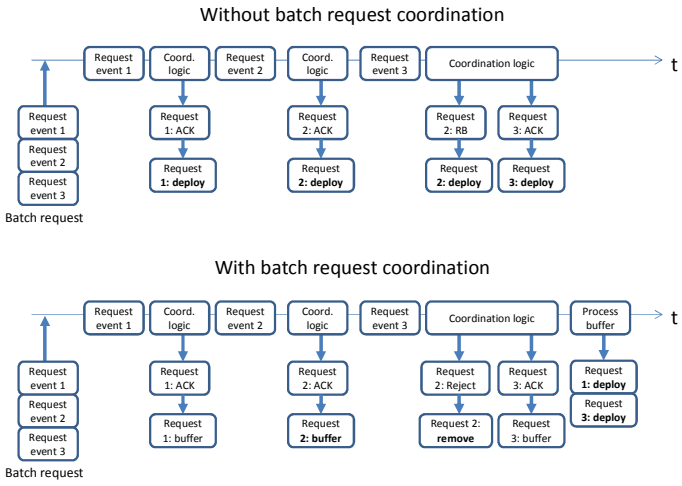

Fig. 9. Comparison of individual vs. batch request coordination

VI. CONCLUSION

SON functions realize individual use cases in network operation and optimization. While this "toolbox" approach has a lot of advantages for network operators and equipment vendors, an efficient (potentially multi-vendor) system integration and operation is required. If many individually operating SON function instances are deployed in the network, the risk of conflicts and thus the risk of sub-optimal and undesired network behavior increases. Hence, it is widely agreed now (cf. [2]-[6]) that a coordination solution must address the tradeoff between "safety" (avoiding basically all conflicts) vs. "efficiency" (fast, parallelized execution of SON function instances).

This paper proposed an output buffer of pending requests that basically eliminates rollbacks on network resources due to coordination decisions, thereby improving efficiency. The buffer is also the baseline for improved coordination decision making methods. As an example, a combined priority / token bucket method has been proposed, which is able to:

- avoid monopolization by "greedy" high priority functions
- approximate "workflows" where the respective function type priority setting determines the sequence of function types to be executed and the respective number of tokens determines the duration of the execution for each function type in the "workflow"
- keep the autonomous characteristic of SON function instances, i.e., a function instance and corresponding configuration action is only executed when there is a need, i.e., a specific condition has been detected in the network

Thus, operational constraints can be addressed together with the desire to increase the level of automation. These characteristics have been proven by evaluation within an experimental system simulating a specific self-optimization case.

REFERENCES

[1] NGMN Alliance, "NGMN Use Cases related to Self Organising Network, Overall Description," December 2008.

[2] T. Jansen et al., "Embedding Multiple Self-Organisation Functionalities in Future Radio Access Networks," 69th Vehicular Technology Conference (VTC Spring), Barcelona, Spain, April 2009 , pp. 1-5.

[3] U. Barth and E. Kuehn, "Self-Organization in 4G Mobile Networks: Motivation and Vision," 7th International Symposium on Wireless Communication Systems (ISWCS), York, UK, September 2010, pp. 731-735.

[4] S. Hämäläinen, H. Sanneck, and C. Sartori (eds.), "LTE Self-Organizing Networks (SON): Network Management Automation for Operational Efficiency," Wiley, December 2011.

[5] M. Amirijoo et al. "A Coordination Framework for Self-Organisation in LTE Networks," 12th IFIP/IEEE International Symposium on Integrated Network Management (IM), Dublin, Ireland, May 2011, pp. 193-200.

[6] X. Gelabert, B. Sayrac, and S. B. Jemaa, "A Performance Evaluation Framework for Control Loop Interaction in Self Organizing Networks," 22nd Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), Toronto, Canada, September 2011, pp. 263-267.

[7] T. Bandh, R. Romeikat, H. Sanneck, and H. Tang, "Policy-based Coordination and Management of SON Functions," 12th IFIP/IEEE International Symposium on Integrated Network Management (IM), Dublin, Ireland, May 2011, pp. 827-840.

[8] T. Bandh, R. Romeikat, and H. Sanneck, "An Integrated SON Experimental System for Self-Optimization and SON Coordination," 2nd International Workshop on Self-Organizing Networks (IWSON), Paris, France, August 2012.