Knowledge Modeling for Conflict Detection in Self-organized Networks

Vilho Räisänen and Haitao Tang

Nokia Siemens Networks, 02600 Espoo, Finland
{vilho.raisanen,haitao.tang}@nsn.com

Abstract. In this article, conflict detection between functions in self-organizing networks (SON) is reviewed. SON coordination is of crucial importance to management automation of fourth-generation networks. In particular, conflict detection is studied from knowledge management perspective. The advantages of model-based conflict detection over algorithmic alternatives are analyzed.

Keywords: Self-organized networks, knowledge management, reasoning.

1 Introduction

Analogously to "plug and play" concept used in computing, self-organizing is perceived as an increasingly important capability for networks. A network with self-organizing capability is called as Self-Organizing Network (SON), where installation and operation are mainly done through a set of automatic and mostly autonomous self-organizing capabilities such as self-configuration, self-optimization, and self-healing. The self-organizing capability is expected to reduce Capital Expenditure (CAPEX) and Operational Expenditure (OPEX) of the network by making it easier to realize potentially time-varying operational objectives.

Self-organization tasks are carried out by instances of SON functions in the radio access network. The type of a SON function describes its capability, an example of the type being. Transmitter/Receiver (TRX) tilt angle adjustment. The presence of multiple concurrent function instances gives rise to the need of coordination. For example, one self-organizing function instance may update a parameter with a particular value. Immediately afterwards, another SON function instance may change the parameter with another value. At worst, the two functions can make conflicting adjustments. This is one type of conflict between SON functions.

The coordination can be built based a complete set of pre-defined decision trees (i.e., the "hard-coded") to resolve or prevent all known conflicts between SON functions. However, the weakness of such an approach is that it is not scalable. A change in the SON functions could require the change of the complete set of the decision tree at worst. The new SON functions are coming. Any update or additions of the SON functions will require the change on the decision trees as well. The motivation of this article is thus to investigate a coordination approach which is scalable and future proof in resolving or preventing the SON conflicts.

K. Pentikousis et al. (Eds.): MONAMI 2011, LNICST 97, pp. 107–119, 2012.

Below, SON functions and conflicts between them are introduced, followed by discussion about conflict detection for SON functions from information management perspective. It is argued that the use of suitably chosen knowledge models and reasoning is better than “hard-coded” algorithmic detection logic. The choice of conflict detection methodology affects not only implementation and testing of SON system, but also maintenance of the system.

This article is organized as follows: in Section 2, characteristics of SON functions are described followed by a Section on operations architecture for a SON-capable network. In Section 4, the interactions between SON functions are discussed. Sections 5-7 present approaches for conflict detection and introduce the use of knowledge management and reasoning in SON coordination.

2 Scopes and Locations of SON Functions in a Network

A SON function instance in a mobile network [1-7] can be characterized in terms of its type, scopes and location. Relevant scopes are input scope, impact area, and impact time. An input scope of a SON function is the scope in which a SON function instance collects the inputs required for its execution. An impact area of a SON function is the scope in affected by an action of a SON function instance. The input scope and the impact area can each consist of a cell, a cell pair, cell neighbours, a cell cluster, sub-network, or the network. Furthermore, an impact time of a SON function is the time period during which an action of the SON function instance has an effect upon other related SON function instances. The location of a SON function consists of the entity or entities where a SON function instance is executed.

The aforementioned scopes and locations may have a significant effect to the network in terms of stability, reliability, scalability, and performance. Usually, a function executed frequently requires more scalable architecture. The function might benefit from a de-centralized architecture in terms of performance. In such a case, the location of the function should be small.

On the other hand, a centralized approach is suitable for SON functions where the scalability and performance requirements are not high, and where the scopes are large. Typically the scope in such a case might vary from a cell cluster to network.

As a summary, when considering the optimal location for a given SON function, one may start with the rule of thumb "the faster the SON function is, the more decentralized its location is in the network." However, this rule of thumb may be overly simplistic for some SON functions so it should not be assumed to apply in all cases.

3 Reference Architecture of a SON-Capable LTE Network and Its O&M

Figure 1 shows SON-capable multi-vendor reference architecture for LTE [8] and its operations and management (O&M). It is in accord with the LTE and O&M architecture made at 3GPP standardization body, where a NE (such as an eNodeB, eNB for short) is managed through the Domain Manager (DM) of the same vendor.

The network elements under a DM form a vendor domain. Different vendor domains can interwork via open interfaces (e.g., X2 and Itf-N). In the Figure, one can see that some SON functions are located in network management layer, some in DM, and some in NEs (e.g., eNBs). Itf-N and Itf-S are interfaces used for the management of eNBs and other management elements (e.g., mobility management entity, MME) of the LTE network. The X2 interface is the control interface between eNBs. The control interface between an eNB and MME is the S1 interface. More details of the interfaces can be found in the standard [8]. The SON function instances reside in network manager (NM), DM, or NEs, and their locations are selected according to their input and impact scopes.

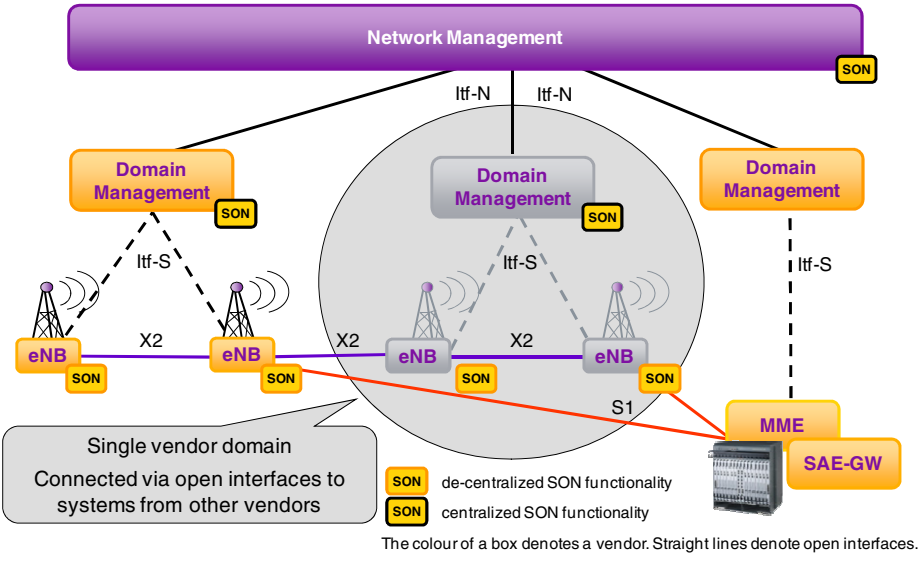

Fig. 1. Multi-vendor reference architecture of a SON-capable LTE network and its O&M, where interfaces Itf-N, Itf-S, S1, and X2 are defined in [8]

SON functions of an LTE network can reside in different network entities. This leads to the architecture as shown in Figure 1, which can be called multi-layer SON architecture, where some SON functions are realized with a decentralized approach and others are implemented with a centralized approach. In this sense, the reference architecture is a hybrid multi-layer SON architecture consisting of both decentralized and centralized SON functions. It would be inefficient to centralize a function where most of the required data is available in eNB. It would be equally inefficient to distribute functions which are dependent on large quantities of X2 data, since in most cases the X2 will share the same physical routing with S1.

This hybrid SON architecture requires a standard SON management framework made for O&M, including its standard interface Itf-N. This standard SON management framework can be that shown in Figure 2. It proposes a distributed SON coordination function to support the operator in the operation of the whole network.

This distributed SON coordination function is responsible on (1) reacting to operational instructions accordingly, (2) managing the conflicts between the SON functions in the network and resolving them, and (3) running operational workflows to pursue operational goals.

The principles of this SON management framework can be summarized as follows:

- Individual SON functions are located at NE, DM, and NM.
- A distributed SON coordination function coordinates them.
- Cross-Itf-N SON conflict/coordination should be standardized (i.e., the blue part in the Figure).
- A vendor-specific SON function can work in NM if it supports the standard interface.
- A vendor-specific SON coordination function can work across NM if it supports the standard interface.

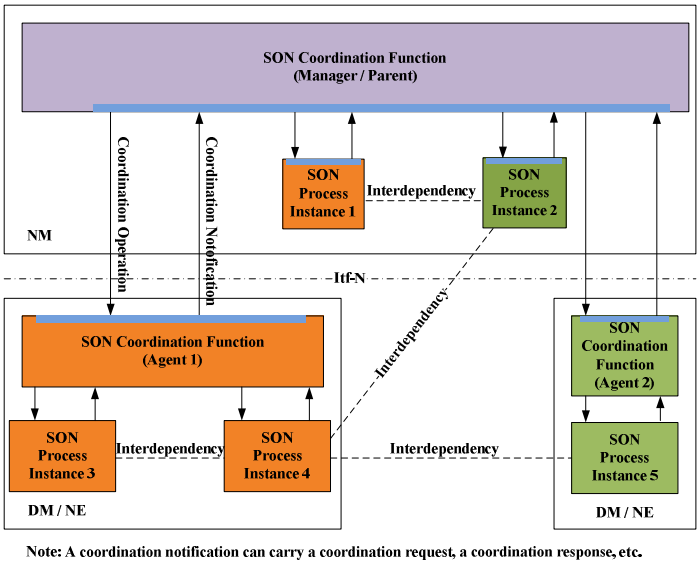

Fig. 2. An architecture framework for the distributed SON coordination function

4 Functional Conflicts in a Self-organizing Network

This section introduces SON conflicts and the situations in which they arise. Examples of SON conflicts are provided below. A good understanding of SON conflicts is the basis for correctly operating a SON network with multiple SON functions integrated.

4.1 General Nature of SON Function Interactions

A SON interaction takes place between two SON function instances. Such an interaction can only appear when there is a dependency either directly or indirectly between the two instances. Some SON interactions may actually boost the system performance of the network, whereas other SON interactions may have an adverse effect on overall performance. The first type of SON interaction could be called as positive SON interactions, and the second type negative SON interactions or SON conflicts. The latter are the main focus of this article.

A SON takes place between two or more SON function instances in a given time period, where one instance (say A) has an impact on another instance (say B) affecting the originally intended operation of the latter instance and thus lower the related system performance. The impact of a SON conflict can (1) distort the input for Instance B, (2) block the execution of Instance B, (3) cancel the intended action of Instance B, (4) cancel the change made by instance B, (5) delete / diminish the performance gain achievable by Instance B, and/or (6) compete with Instance B to solve the problem that should be solved originally by Instance B alone. Thus, by definition, a SON conflict is always directional; Instance A having conflict with Instance B in a particular way does not mean that Instance B would conflict with Instance A in the same way if at all.

Specific SON conflicts (more general, SON interactions) can only occur between specific SON function instances under an assumption of specific system integration and operation environment (including specific priority assumption for the specific SON function instances).

A SON conflict can be further categorized as a direct SON conflict or an indirect SON conflict. A direct SON conflict takes place between two or more SON function instances on the same network entity in a given time period, as shown in Figure 3. As shown in Figure 4 and Figure 5, an indirect SON conflict takes place happens between two or more SON function instances that take effect on (1) the same network entity at different points of time beyond the above time period (as shown in Figure 4) and/or (2) different network entities that would lead to a conflict at certain network entity at certain time later (as shown in Figure 5).

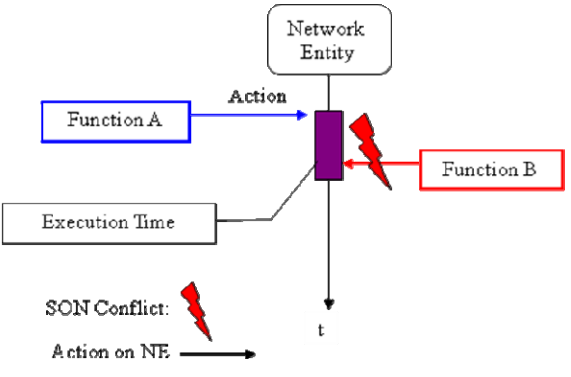

Fig. 3. An example of a direct SON conflict between Function B and Function A. From [9]

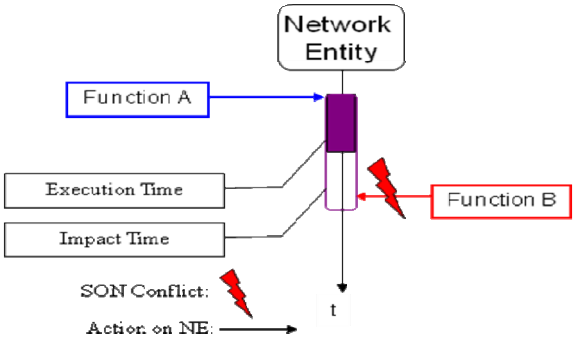

Fig. 4. An example of an indirect SON conflict between Function B and Function A, which happens beyond the execution period of Function A. From [9]

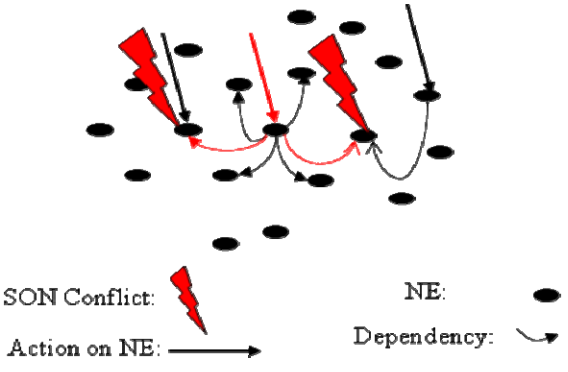

Fig. 5. An example of an indirect SON conflict between the functional actions at different NEs

4.2 Network Parameters Related to SON Functions

There are many network parameters (a few hundreds, some of which only rarely change in the course of network operation) related to SON functions in a self-organizing LTE network. These parameters include the cell-related identities (IDs), radio transmission power and other antenna parameters, radio channel parameters, neighbor cell parameters, mobility parameters, etc. As shown in Figure 6, many of the network parameters are directly related to two or more different SON function types, as their shared inputs, their shared outputs, or both.

In Figure 6, SON function interacting with another SON function through their directly shared network parameters and their indirectly related parameters are illustrated. This issue will be discussed in more detail in the next sections.

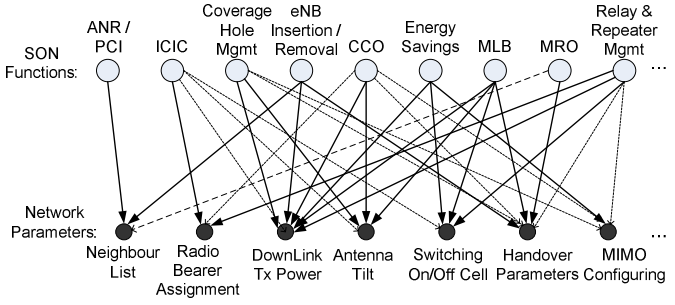

Fig. 6. The network parameters directly shared by different SON functions in a self-organizing LTE network. Solid line means a major parameter, while the dashed arrow line means a secondary parameter. From [10].

4.3 Examples of Potential SON Conflicts between Selected SON Functions

Examples of SON conflicts between selected SON functions are studied below.

4.3.1 Potential SON Conflicts between Physical Cell ID Function Instances

Two or more instances of physical cell ID (PCI) function can be active in a network or network area at the same time. They can be triggered by the insertion of cells and the need to update certain assigned cell IDs. Therefore, there are potential conflicts of cell-ID collision caused by two individual instances from assigning their cell IDs. For example shown in Figure 7, Cell Z is confused by two of its neighbor cells that have the same Physical Cell ID "Cell X" after the PCI instance B assigns the same physical cell ID to the latest inserted cell, if they are not coordinated. There is thus the need of a conflict solution.

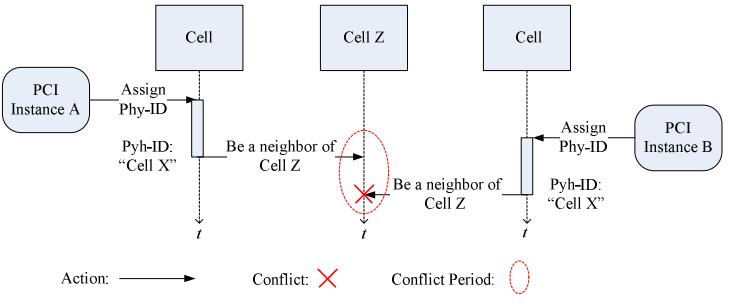

Fig. 7. A potential indirect conflict between two PCI instances (A and B) over different time and from different cells, where Cell Z is confused by two of its neighbor cells that have the same Physical Cell ID "Cell X" after the PCI instance B assigns the same physical cell ID to the latest inserted cell, if they are not coordinated

4.3.2 Potential SON Conflict between Two Cell Outage Compensation (COC) Function Instances in Two Different Domains

As shown in Figure 8, Cells A1 and A2 belong to Domain A managed by Domain Manager A. Cells B1 and B2 belong to Domain B managed by Domain Manager B. Cell A2 is neighbour to both Cell A1 and Cell B1. There is an individual COC function in each domain, which is running in the domain manager of that domain. There is a limit in the reality: Each domain has only visibility to its own domain.

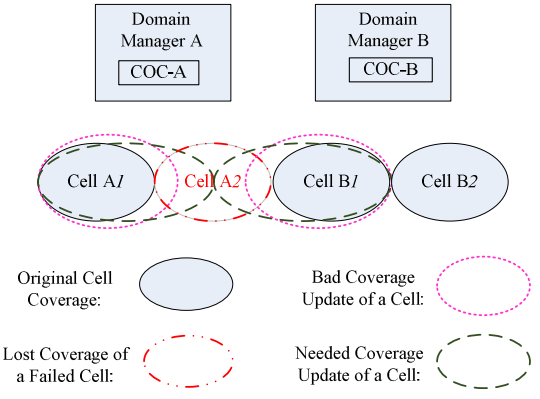

Fig. 8. An example cell outage at the border between two domains, and the problem for compensating its coverage by the two COC functions in Domain A and B, where Cell A1 and A2 are of Domain A and Cell B1 and B2 are of Domain B

Now, let us assume that Cell A2 fails, and that the failure creates a coverage hole that needs to be fixed with the extended coverage from both Cell A1 and Cell B1. Thus, each of the domains should have to activate its COC function. The following cross-domain conflicts between the COC functions in the different domains can take place:

- If the neighbouring Domain B finds the outage, the neighbouring Domain B may need to activate its COC function and inform Domain A (with the cell outage) to activate its COC function.
- If Domain A (with the cell outage) finds the outage, this domain may need to activate its COC function and inform the neighbouring Domain B to activate its COC function.
- When both domains decide their specific cell compensations (e.g., increasing a cell coverage towards the outage area), they need to coordinate or be coordinated so that to make the specific cell compensations consistent, which fixes the outage problem properly.

This raises the question: how should the coverage of cells A1 and B1 be adjusted correctly by the two separated domains? Neither of them has visibility beyond their own domain. This problem can only be solved by the cooperation of the COC function instances in Domain A and Domain B in run time. They need thus be coordinated in run time, so that a potential conflict can be identified, and prevented or resolved.

5 Information Management Requirements

Abstracting from previous discussion, the logic for detecting conflicts can be summarized as follows:

- A prevents the execution of B (e.g. number of concurrent instances is limited).
- The output of A changes data in the input scope of B so that execution of B is no longer possible.
- The output of A acts in an opposite direction than that of B.

Information needed to detect these conflicts require

1. Type of function instance.
2. Inputs and outputs associated with a particular function type.
3. Location of function instance.
4. Scopes of function instance.

Of the types of information listed above, #1 is readily known, for #2, the types of parameters can be determined in design time (but corresponding scopes only in run time), and #3 and #4 are determined in run time. As described above, #4 may involve cross-vendor domain operations.

Due to the number of SON function types, as well as combinations of their relative locations and scopes, an exhaustive conflict matrix would be large. Consequently, an algorithmic implementation would be equally sizable. Such implementations not only need to be implemented and tested, but also maintenance aspects need to be taken into account. For instance, a change in the definition of a SON function type could at worst require reassessment of the entire conflict matrix.

Below, an alternative is studied, namely a means of generating interaction analysis automatically from small number of more easily verifiable constructs.

6 Use of Models for Conflict Analysis Generation

The basis for generation of the conflict is the use of models. Function types are modeled in terms of its inputs and outputs, whereas function instances are characterized by their type, location, and scopes. Conflict analysis can then be generated from small number of rules related to types, inputs, outputs, locations, and scopes of function instances.

Modeling described above can be based on different modeling paradigms. Simple Entity-Relationship (ER) model is conceptually the simplest, but ensuring the consistency of the model is a challenge. The use of a modeling language alone is not enough, but needs to be accompanied by processes and best current practices (BCPs). This is also true for more complex variants of ER such as the class diagram of Unified Modelling Language (UML). Furthermore, the analysis of ER-type models requires

bespoke algorithm which also needs to be maintained and conform to the same set of processes and BCPs as the model. Indeed, one might say that the algorithmic complexity of conflict analysis has been traded for complexity of model and analysis algorithm maintenance.

These shortcomings can be addressed by using models the consistency of which can be readily established, and which facilitate reasoning to at least partly substitute bespoke algorithms. This necessitates the use of formal models. We have opted to use knowledge modeling paradigm which provide formal semantics, supports reasoning, consistency checking, and – as a bonus – also can deal with imperfect information. More precisely, we are using Description Logic (DL) models, the properties of which are well known [11].

With DL models, Knowledge Base (KB) is the basis for reasoning. The creation and governance of KB should be designed to minimize effort required and support participation of knowledge management roles [12]. In view of these goals, the contents of knowledge base can be reduced to the following constituents:

1. Ontology of SON function types and their inputs and outputs.
2. Rules for identifying conflicts.
3. System state (types/locations/scopes of SON function instances).

Of the above, #1 originates from standards and systems design, #2 is expert knowledge, and #3 is imported from the run-time system. As discussed in [12], model transformations can be utilized to create KB contents corresponding to #1 and #3. The middle constituent needs to be created by expert, but the crucial difference is that related modeling can be performed in terms of higher-level domain concepts, as illustrated in the example below. The creation of model transformations in #1 and #3 may require expert knowledge, but their operation does not.

7 Example

In this Section, a simple example is studied to illustrate our concepts introduced above.

The example relates to two SON function types: CCO-RET adjusts TRX tilt angle and CCO-PWR adjusts transmission power. The SON coordinator is assessing the risk of conflict of starting CCO-PWR in cells C2 and C3 while CCO-RET is running in cell C1. The sector S1a of cell C1 is neighbor to sectors S2a of cell C2 and S3a of cell C3. We shall assume that CCO-RET has been started first, and has output parameter TRX-tilt. For simplicity we shall assume that the input and output scopes of both functions are the same as their locations, so that e.g. CCO-RET collects its input from C1 and also performs adjustments in C1.

The outputs of both of the functions potentially affect the cell size as well as interference caused to neighboring cells. Relevant to the analysis, CCO-RET is assumed to have an output parameter *TRX-tilt* and CCO-power an output *TX-power*.

Both parameters *TRX-tilt* and *TX-power* relate to coverage. The two functions may be in conflict e.g. so that CCO-RET attempts to reduce interference by tilting its TRX down, whereas CCO-power simultaneous decides to increase output power for its TRX in a neighboring sector.

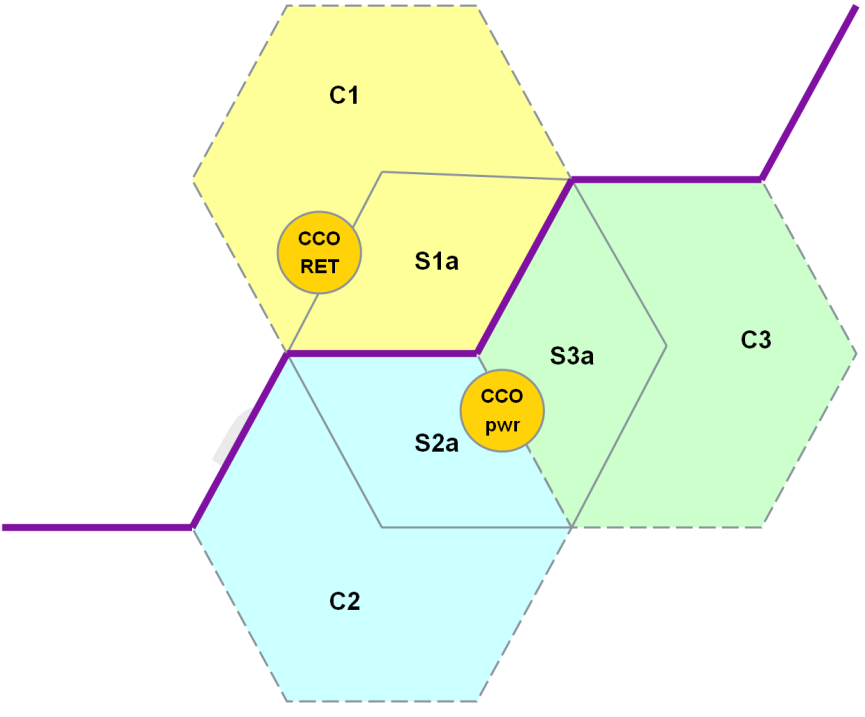

Fig. 9. Topology for the example

Let us next consider how this situation can be modeled with knowledge management technologies. We shall use OWL/RDF-like syntax [13] for the illustration. In the interests of space, we leave out some details (e.g. relation between sectors and cells as well as neighbor relations).

CCO-RET hasScope C1
CCO-power hasScope C2
CCO-power hasScope C3
CCO-RET hasOutput TRX-tilt
CCO-power hasOutput TRX-power
TRX-tilt relatesTo Coverage
TRX-power relatesTo Coverage
Coverage isA DomainConcept

Above, the three first assertions define the scopes of the functions, and the two next ones establish the relevant input and output parameters for conflict detection. The three last assertions, in turn, say that both *TRX-tilt* and *TRX-power* relate to the domain concept called *Coverage*. Next we proceed to define the conflict detection rule:

if (Function1 hasOutput which relatesTo DomainConcept D) and
(Function2 hasIntput which relatesTo DomainConcept D) and
((scopeOf Function1) AdjacentTo (ScopeOf Function2)) then
PotentialConflict

Above, a pseudo-logic description has been used instead of actual OWL rule to enhance readability. For simplicity, the rule above assumes that parameter based conflict detection is performed in "worst case" manner, i.e., if the parameters of two function instances are related to the same domain concept, they always result in a conflict. Modelling can be made more accurate by considering parameter value ranges.

The definition of conflict presented above is generic, and is not specific to the types of the functions used in the example. The same rule can be readily be used for output parameter scope adjacency based conflict detection for any SON functions, provided that their parameters are mapped to domain model concepts. Similar generic rules can be defined for detecting other types of conflicts (e.g., impact of an output parameter of one function instance on an input parameter of another function).

Some of the advantages of the model-based approach described above are:

- Domain expert defining conflict detection rules can operate with high-level conflicts rather than dealing with individual pairs of functions.
- Reasoner ensures that definitions in the knowledge base are internally consistent.
- If function definition is added or modified, only mapping of parameters to domain model need to be checked.

8 Summary and Outlook

Concepts related to coordination of self-organized functions have been described with a focus on SON conflicts between function instances in run time. SON coordination has been analyzed from knowledge management perspective, and summarized the benefits of model-based approach as compared to "hard-coded" algorithms. The main advantages of the use knowledge management models are consistency, change management, and the ability to work on higher conceptual level in defining conflicts. The knowledge models for conflict detection uses existing information models and expert knowledge as inputs in creating the class model used by the reasoner.

In production systems, the information management flexibility provided by model-driven methods needs to be balanced with other considerations, such as real-time

performance. The knowledge base / reasoner approach described above is not to be taken to substitute other paradigms, but rather to provide a complementary functionality where it makes sense.

References

1. Dottling, M., Viering, I.: Challenges in mobile network operation: Towards self-optimizing networks. In: Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (April 2009)
2. Deliverable D2.2: Requirements for Self-organizing Networks. INFSO-ICT-216284, SOCRATES (June 2008), http://www.fp7-socrates.eu/
3. Deliverable D2.1: Use Cases for Self-Organizing Networks. INFSO-ICT-216284 SOCRATES (March 2008), http://www.fp7-socrates.eu/
4. NGMN Deliverable: NGMN Use Cases related to Self Organising Network, Overall Description. NGMN (2008), http://www.ngmn.org
5. 3GPP TR 36.902: Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Self-configuration and self-optimizing network use cases and solutions, V1.0.1 (September 2008)
6. 3GPP TR 36.902: Evolved Universal Terrestrial Radio Access Network(E-UTRAN); Self-configuration and self-optimizing network use cases and solutions. V9.3.0 (December 21, 2010)
7. Hämäläinen, S.: Self-Organizing Networks in 3GPP LTE. In: Proceedings of Portable 2009 (September 2009), http://www.ieeevtc.org/portable2009/portable2009-finalprog02.pdf
8. 3GPP TS 36.300: Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Overall Description. V10.2.0 (December 21, 2010)
9. Bandh, T., Romeikat, R., Sanneck, H., Tang, H.: Policy-based coordination and management of SON functions. In: Proc. IM 2011, Dublin, Ireland (2011)
10. Tang, H., Hämäläinen, S.: Self-organizing functions and their coordination in self-organizing communication networks. Systemics and Informatics World Network 11, 77 (2010)
11. Baader, F., Calvanese, D., et al. (eds.): The description logic handbook, 2nd edn. Cambridge University Press, Cambridge (2007)
12. Räisänen, V.: Semantic aspects of system integration. In: Proc. 6th International Workshop on Vocabularies, Ontologies, and Rules for the Enterprise, Helsinki, Finland (August 2011)
13. See OWL and RDF definitions at W3C website, http://www.w3.org (tested April 2011)