Towards Automated Verification of Autonomous Networks: A Case Study in Self-Configuration

JaeSeung Song, Tiejun Ma, Peter Pietzuch
Department of Computing
Imperial College London
London SW7 2AZ, United Kingdom
{*jsong, tma, prp*}*@doc.ic.ac.uk*

***Abstract*—In autonomic networks, the self-configuration of network entities is one of the most desirable properties. In this paper, we show how formal verification techniques can verify the correctness of self-configuration. As a case study, we describe the configuration of physical cell identifiers (PCIs), a radio configuration parameter in cellular base stations. We provide formal models of PCI assignment algorithms and their desired properties. We then demonstrate how the potential for conflicting PCI assignments can be detected using model checking and resolved in the design stage. Through this case study, we argue that both simulation and verification should be adopted and highlight the potential of runtime verification approaches in this space.**

***Keywords*-network self-configuration; verification; model-checking; autonomous networks**

I. Introduction

The complexity of current communication networks is increasing due to their scale, heterogeneity and new requirements imposed by users. Recently the wireless communication industry has introduced new technologies, such as the *Long Term Evolution* [1], as the next step in the evolution of mobile networks. These networks require network operators to deploy new cellular base stations and core network entities in order to cover an equivalent geographical area. For example, a new type of cellular home base station may be installed by home users to extend coverage and capacity [2]. A solution to these management challenges in future networks is to apply concepts from autonomic computing to design *autonomous networks* [3]. Such networks include functionality for self-configuration, self-optimisation and self-healing. This leads not only to efficient and reliable operation but also a reduction in costs to network operators.

We argue that *formal verification* should play a central role in the design and operation of future autonomous networks. Formal verification techniques based on model-checking have been successfully used for the verification of hardware designs, communication protocols and safety-critical systems. Such approaches explore the entire state space of a system to ensure that there are no violations of correctness properties. For network operators, formal verification can provide correctness assurances for autonomous networks throughout the design and operation.

In this paper, we compare formal verification with simulation for analysing the correctness of self-configuration in autonomous networks. As a case study, we focus on a specific management problem, namely the self-configuration of cellular base stations. We show how it can benefit from model-checking. We formalise the correct assignment of physical cell identifiers (PCIs) to base stations and verify assignment algorithms using model-checking.

Our verification results show that current PCI assignment algorithms cannot guarantee correctness due to two types of violation. First, violations occur when available PCIs are fewer than the number of base stations. Second, violations also can be caused by concurrent deployment of adjacent base stations. However, verification based on model-checking has limited scalability due to the large number of possible states. Simulation does not suffer from this short-coming but it only explores a small portion of the possible state space. It is less likely to detect violations in specific conditions. To combine the exhaustive analysis of verification while retaining the scalability of simulation, we propose a network verification approach based on *runtime verification* that searches for error states starting from partial snapshots of the configuration state.

In the next section, we survey related research on autonomous networks and formal network verification. §III describes the PCI assignment problem and algorithms for solving it. We then present formal models and the associated verification approach using model-checking in §IV. In §V, we discuss the experimental results from verification and simulation. To address the limited scalability of verification, we propose runtime verification in autonomous networks in §VI. The paper finishes with conclusions in §VII.

II. Background

Research into *autonomous networks* comes from the idea of autonomic computing initially proposed by IBM [4] and extends its scope to networking. There are a number of ongoing research projects in this area and different architectures for autonomous networks have been proposed and analysed [3], [5]. The *Self-NET* project [6] describes and validates an innovative architecture for cognitive self-managed

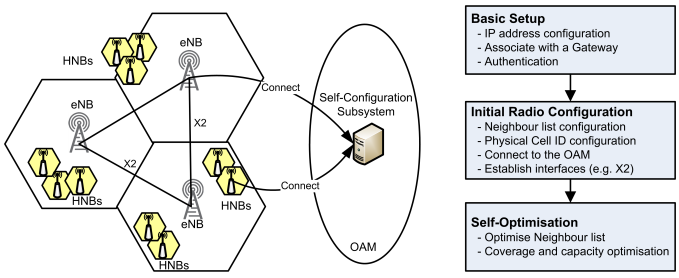


Figure 1. Overview of self-configuration in cellular base stations

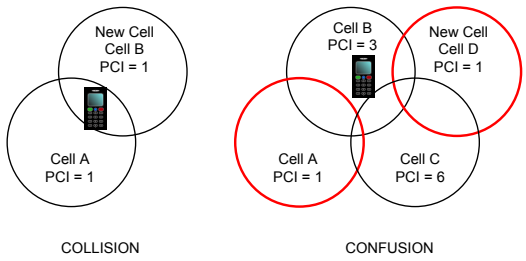


Figure 2. Examples of collision and confusion in PCI assignment

elements for the future Internet. Standardisation efforts, such as the Third Generation Partnership Project (3GPP) for the telecommunication industry, have started work on a standard for the *Self-Organising Network* (SON) that has the autonomous features [7].

Formal methods can guarantee the correctness of systems. They can be used to model and verify existing protocols and algorithms. Architectures or protocols are designed first and verified after the design phase to check for correctness according to a specification. *Model checking* [8] has been used successfully in various software systems and protocol specifications to discover design bugs, verify correctness and analyse systems. Examples include the analysis of the Dynamic Host Configuration Protocol (DHCP) [9], the FireWire tree identity protocol [10] and the root contention protocol [11]. As part of such verification, new bugs have been discovered.

III. Self-Configuration of Base Stations

As a case study, we describe self-configuration in a self-organising network. We show how formal verification techniques can be adopted to analyse algorithms in self-organising network. A *self-organising network* (SON) is an autonomous network specified by the 3GPP that configures, optimises, heals and protects itself to help operators reduce operational expenses [7]. Figure 1 shows its architecture. In a SON, *evolved Node B* (eNBs) and *Home Node B* (HNBs)—also called Femtocells or home base stations—are cellular base stations that cover macro and small (home-range) areas, respectively. A *cell* is the basic geographic unit covered by a base station. A cell has a unique *physical-layer cell identifier* (PCI) as a basic configuration parameter that identifies the cell. As depicted in Figure 1, a macrocell created by an eNB includes multiple small cells created by HNBs. An eNB has a logical *X2 interface* to exchange configuration data with adjacent eNBs. An *Operation and Management Unit* (OAM) is responsible for a set of management functions. It incorporates a self-configuration subsystem for configuring parameters of eNBs and HNBs.

In a SON, self-configuration enables a newly-deployed base station to configure radio parameters automatically. During startup, base stations connect to the self-configuration subsystem to configure their initial parameters. This includes three steps: *basic setup*, *initial radio configuration* and *self-optimisation*. During the basic setup, an IP address is allocated to the new base station and a gateway is configured. After the basic setup, the base station starts to set its radio parameters. This includes managing a neighbour list and allocating unique PCIs to provide mobility management and avoid interference with adjacent cells.

A. PCI assignment problem

In a SON, the PCI is an essential configuration parameter for a cell and corresponds to a unique combination of one orthogonal and one pseudo-random sequence for data encoding. Only 504 unique PCIs are supported because of compatibility with legacy base stations [7]. When a new base station is deployed, a PCI needs to be selected. In anticipation of the large number of base stations to be deployed in the future, the reuse of PCIs by different base stations becomes unavoidable. In currently deployed networks, PCIs of cells are configured by means of static planning [12]. Besides being expensive, this approach does not scale to a large number of home base stations.

According to the SON specifications [7], an automated PCI selection algorithm should fulfil the following two requirements regardless of deployment strategies:

- **collision-free:** a PCI should be unique in the area that the cell covers;
- **confusion-free:** a cell should not have neighbour cells with identical PCIs.

Figure 2 illustrates PCI collision and confusion. If the PCI of Cell B is equal to 1, there is a collision between Cells A and B because the PCI of Cell A is also 1. If the PCI of a new Cell D is selected to be 1, it will lead to confusion of Cells B and C with A. This would cause hand-over procedures from B to the new Cell D to fail.

The PCI selection problem is equivalent to the distance-2 vertex colouring problem where the PCI is a colour. It has been shown that this problem is NP-complete [13]. There exist randomised distributed algorithms for solving it in wireless ad-hoc networks [14]. The time complexity of such algorithms is $O(\Delta \log^2 n)$ where Δ is the maximum degree and n is the number of network nodes.

B. PCI assignment algorithm

To achieve collision- and confusion-free assignments, the 3GPP proposed a PCI selection algorithm with several optional extensions, which Figure 1 summarises. The algorithm is divided into four main steps, with steps 2–3 being optional. Network operators have the flexibility to implement the optional steps to improve the PCI selection process.

(1) A base station tries to get a valid range of PCIs from the OAM. The list of returned PCIs depends on the location of the deployment and the operator's planning policies.
(2) The base station performs neighbour discovery through a broadcasting mechanism to detect the PCIs of its neighbour cells, thus avoiding selecting these PCIs.
(3) The X2 interface enables neighbours to exchange a *neighbour relation table* that contains information about neighbours of neighbours. Therefore, the base station may avoid selecting PCIs that result in confusion.
(4) The base station selects a random PCI from the list of candidate PCIs. The base station then sends the selected PCI to the OAM that records this configuration.

IV. Verification Approach

To assess the satisfaction of the collision- and confusion-free properties, we adopt model-checking as a rigorous method to evaluate the current PCI selection algorithm. We chose to use the SPIN model-checker [15] because it supports the verification of asynchronous and distributed process system. It uses the *PROcess Meta LAnguage* (PROMELA) to describe the algorithm and its assignment policies. PROMELA can be translated into a C program for efficient verification using SPIN. It supports the specification of safety properties in *linear temporal logic* (LTL). The semantics of LTL provides temporal modal operators that can make statements about properties that are globally true or eventually true. This is sufficient to describe collision- and confusion-free PCI assignments.

A. Modelling PCI assignment

We consider an abstract model of the PCI assignment problem. We view the SON as a distributed system that includes base stations and the OAM. We only model the configuration procedure and do not consider failure of any process. All communication channels are assumed to be reliable and do not duplicate messages. The local clocks of processes are synchronised and clock drifts can be ignored. With these simplifying assumptions, we can focus on the fundamentals of the PCI selection algorithm.

To define neighbour relationships and detect collision and confusion, we need to model the spatial layout of cells. A matrix position $(i,\ j)$ (where $(i,\ j)$ are non-negative integers) is a coordinate that represents the spatial location of a cell. A cell with its PCI located at $(i,\ j)$ can be represented as $\mathcal{P}_{(i,\ j)}$. We assume that all places that can be reached directly from $(i,\ j)$ are neighbours of $(i,\ j)$; all places that can be reached directly from neighbours of $(i,\ j)$ are neighbours of neighbours of $(i,\ j)$. Any deployed base station can have a maximum of 8 neighbours and 16 neighbours of neighbours.

In practice, HNBs can have a varying number of neighbour cells depending on the deployment environment, the density of HNBs and radio propagation properties. However, the principle of PCI selection remains the same irrespective of the numbers of neighbours. We adopt this simple model to illustrate the problem and investigate the feasibility of formal verification. As we show in §V, the scalability of model-checking is limited even in this simple scenario.

B. Verification goals

Let $\mathbb{N}^+$ represent the positive integers and let $\varnothing$ be the empty set. $N \times N$ represents the size of the two-dimensional matrix where $N \in \mathbb{N}^+$ and $N \geq 2$. Let $(i,\ j)$ be the coordinates of a cell where $i \in [0,\ N]$, $j \in [0,\ N]$. Let $\mathcal{P}_{(i,\ j)}$ represents the PCI of a cell at position $(i,\ j)$ in the matrix where $0 < i \pm 2 < N$, $0 < j \pm 2 < N$. $\mathcal{P}_{(i,\ j)} \in [1,\ 504]$. Initially $\mathcal{P}_{(i,\ j)} = \varnothing$.

The neighbour relationship can be expressed using cell's coordinates. Let $(x,\ y)$ be the neighbour coordinates of $(i,\ j)$ where $x,\ y \in \mathbb{N}^+$, $x \in [0,\ N]$, $y \in [0,\ N]$ and $x \neq i$, $y \neq j$. If Σ_n is a finite set containing a list of $\mathcal{P}_{(i,j)}$ neighbours' PCIs, then

$$\Sigma_n = \{\mathcal{P}_{(x,y)} \mid x \in [i-1,\ i+1],\ y \in [j-1,\ j+1]\}.$$

Let Σ_m be a finite set containing a list of $\mathcal{P}_{(i,j)}$ neighbours of neighbours' PCIs, then

$$\Sigma_m = \{\mathcal{P}_{(x,y)} \mid x \in [i-2,\ \ i+2] \wedge x \notin [i-1,\ \ i+1],\\ y \in [j-2,\ j+2] \wedge y \notin [j-1,\ j+1]\}.$$

With these definitions, we define the collision-free and confusion-free properties. Let Ψ and Φ be defined as:

$$\Psi := \mathcal{P}_{(i,j)} \in \Sigma_n,\ \Phi := \ \ \mathcal{P}_{(i,j)} \in \Sigma_m$$

Let $\mathcal{M}_s$ represent a SON, then the safety properties can be stated as:

$$\begin{aligned} \text{collision-free:} \quad & \mathcal{M}_s \models \Box\ \neg\Psi \\ \text{confusion-free:} \quad & \mathcal{M}_s \models \Box\ \neg\Phi \end{aligned} \tag{1}$$

We use the assert command in PROMELA to express the violation of our safety properties (see Equation (1)). The assert statement can take any valid PROMELA expression as its argument. The expression is evaluated each time the statement is executed. If the expression evaluates to false, an assertion violation is reported.

V. Experimental Results

The goal of our experimental evaluation is to evaluate the feasibility of formal verification compared to simulation and examine how it can ensure the correctness of future autonomous networks. We use the SPIN to exhaustively

search the state space for collision or confusion violations of the PCI selection algorithm described in §III-B. In addition, we take advantage of SPIN's support for simulating a model represented in PROMELA. The non-deterministic choices in the model are simulated as random choices and processes communication is simulated as message passing. The SPIN model-checker ran on a 2.4 Ghz Intel Core2 Duo machine with 2 GB of RAM.

We chose the minimum number of available PCIs to be 30 to ensure enough PCIs for the 5 × 5 matrix. For the first scenario, we set the number of available PCIs to 200 and tried to deploy two eNBs in the matrix. However, the verification could not be completed because the model-checker exhausted memory after 31,272,100 state transitions. Therefore, we limited the maximum number of available PCIs to 110 to reduce the number of verifiable states. We use this range of PCIs for the experiments unless otherwise stated. For simulation results, we run each experiment 500 times to ensure a reasonable number of samples.

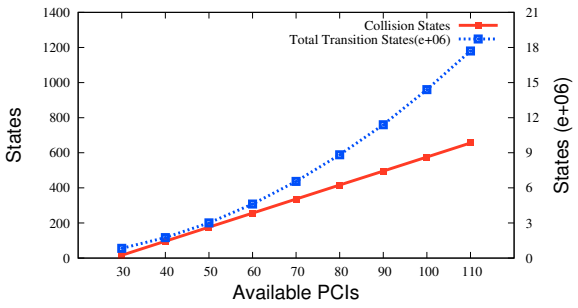
A. Single base station scenario

In this scenario, one eNB is deployed in the 5 × 5 matrix and the other 24 places have pre-deployed eNBs. The self-configuration requirement is that the PCI of the new eNB is collision- and confusion-free in relation to the other cells. We evaluate this scenario according to the PCI assignment algorithm (described in §III-B) in three cases: (1) PCI LIST: the eNB can only receive a valid list of available PCIs from the OAM; (2) NEIGHBOUR DETECT: the eNB can detect its neighbours' PCIs; (3) X2: eNBs can exchange neighbour information over the X2 interface.

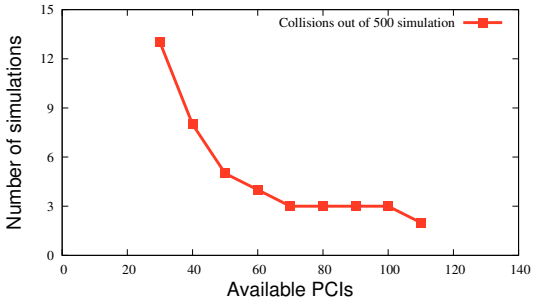
The verification results show that for (1) PCI LIST, there are 8 collisions and 16 confusions; for (2) NEIGHBOUR DETECT, there are no collisions and 16 confusions; and for (3) X2, there are no collisions and confusions. It shows the potential number of states that the deployed eNB has with collisions and confusions. When the eNB cannot detect neighbours' PCIs and communicate with other eNBs through the X2 interface (PCI LIST), both collisions and confusions may occur. When the eNB is capable of detecting its neighbours' PCIs (NEIGHBOUR DETECT), the collision-free property can be guaranteed but confusions may still happen due to lack of PCI information about neighbours' neighbours. When the X2 interface is added (X2), there are no collisions or confusions. This result provides evidence for the usefulness of the 3GPP's recommendation of neighbour detection and the X2 interface.

B. Multiple base station scenario

First we consider two eNBs that are deployed concurrently as neighbours. We assume that all optional extensions are implemented by each eNB, which is why we do not expect collisions or confusions.



(a) Verification of two concurrent eNBs deployment



(b) Simulation of two concurrent eNBs deployment

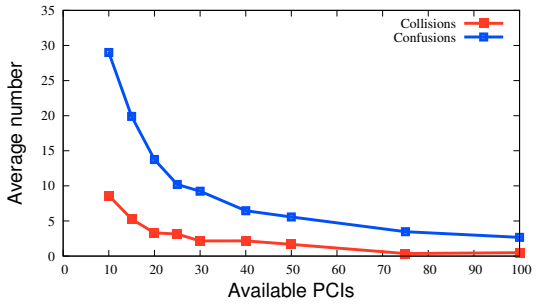
Figure 3. Verification and simulation results for two deployed eNBs

Figure 3(a) presents the verification results in terms of potential collisions and the total number of state transition. The x-axis shows the number of available PCIs, the y-axis shows the collisions (left) and the total state transition (right). As the number of available PCIs increases, the number of collision states also increases linearly and the total transition states of the model increase faster than the collision states. This indicates that, with more PCIs, the possibility of collisions occurring is lower. However, collisions still exist.

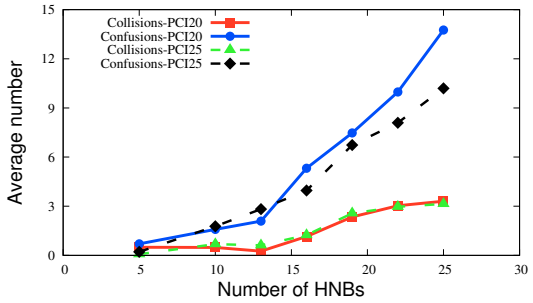
For each violation, SPIN generates a counterexample trace that includes a path to the violation states. It shows that when two neighbour eNBs select their PCIs concurrently to be the same, they are not aware each others PCIs. This causes a violation of the collision-free property. We can expect that the confusion-free property can also be violated when two neighbour of neighbour eNBs are deployed concurrently.

Figure 3(b) shows the simulation results of the average number of occurred collisions. We can see that, as we increase the number of available PCIs, the number of detected collisions decreases. This result implies that when there are more PCIs, the possibility of collisions occurring can be reduced. Simulation cannot detect all potential collisions when there are many candidate PCIs, while verification can always detect all possible collision states. For assessing the collision-free property, verification is more rigorous.

In addition, we also evaluated the effect of varying the number of HNBs and available PCIs. To avoid the scalability



(a) 25 HNBs concurrent deployment



(b) 20 and 25 available PCIs

Figure 4. Simulation results for multiple deployed HNBs

limitations of verification, we constructed two simulation experiments with a limited number of HNBs. In the first experiment, we fixed the number of concurrently deployed HNBs to 25. All HNBs within the 5×5 matrix are deployed concurrently. Each HNB can have up to eight neighbours, which means that the number of available PCIs should be more than that to make a collision-free assignment possible. We then varied the value of available PCIs from 10 to 100. In the second experiment, we keep the number of available PCIs constant at 20 and 25, respectively. This simulates the case when there are not enough PCIs available. For this experiment, we varied the number of HNBs from 5–25.

Figure 4 shows the number of occurred collisions and confusions for the multiple HNBs deployment. We can see that, as the number of available PCIs increases with a fixed number of HNBs, collisions decrease. Figure 4(b) shows that as the number of HNBs increases with a fixed number of PCIs, collisions increase. We conclude that the ratio between the number of HNBs and available PCIs is an important factor that determines the collisions and confusions.

Overall, both our verification and simulation results provide evidence that collisions and confusions cannot be avoided due to the concurrent PCI selection. which has not been addressed in the 3GPP specification.

C. Discussion

Our evaluation of verification and simulation has yielded several insights. Both verification and simulation are useful for detecting hidden design flaws and finding correctness violations. Simulation is easier to understand and perform. It scales better because it avoids the problem of state space explosion. However, simulation has lower state space coverage and only tests scenarios under predefined conditions, thus may not discover infrequently occurring problems.

Formal verification techniques are more rigours when assessing required properties. However, modelling and verification of a complex network is challenging and requires a certain level of abstraction. The problem of state space explosion is the main obstacle when applying formal verification using model-checking. Even our simple deployment model with simplifying assumptions had limited scalable below what would be considered a realistic deployment.

VI. Future Work

We will study more realistic models that consider failure of base stations, message errors and other probabilistic behaviour. We will adopt quantitative verification techniques to assist the design of algorithms and protocols according to the desired overall performance. We also plan to look into reconfiguration approaches to avoid collisions and confusions after they have happened. In addition, we plan to address the scalability limitations of verification at a more fundamental level, as follows:

Runtime verification. Based on the lessons that we learnt as part of this work, we propose a framework for the verification in future autonomous networks. Our approach aims at combining the scalability of simulation with the rigour of model-checking. The main idea is to apply *runtime verification* techniques [16], [17] to verify autonomous networks. Runtime verification adopts model checking during program execution to detect faults. In runtime verification, a runtime verifier periodically checks the correctness of a protocol and algorithm implementation at runtime. During normal system operation, runtime verification observes the system's input and output behaviour to verify given properties. Desired correctness properties are only checked for potential future states in the network model. Since this results in a bounded state depth without searching the entire state space, it minimises the number of explored states and thus avoids the state explosion problem. When a potential violation is detected, a fault-avoidance mechanism can be used to influence the operation of the network avoiding states that may lead to incorrect behaviour.

Figure 5 shows how an autonomous network may include a runtime verification mechanism. A runtime verifier is embedded into the network architecture to provide correctness guarantees. It continuously monitors and checks the network state against given desired properties. Such properties can be described as logical representations using a formal specification and are embedded as monitors into the network implementation. Correctness properties can be functional and non-functional properties, such as consistency, quality

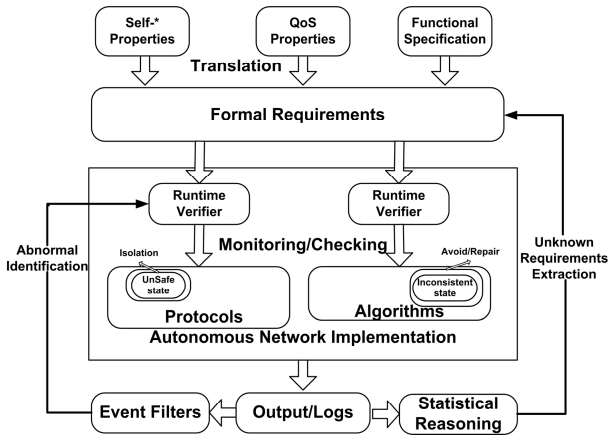


Figure 5. Runtime verification in future autonomous networks

of service and system resiliency etc. During runtime, unsafe states should be *globally* avoided and desired properties should be *eventually* achieved. When unsafe or inconsistent states are detected by the verifier, it avoids such states to ensure that the network continues operating correctly.

To cope with potentially unknown conditions and requirements, verification goals must evolve. Statistical reasoning techniques can be adopted to learn from the system output and provide feedback to verification requirements. We believe that such runtime verification can be applied as an effective and scalable technique for the verification of correctness and performance properties, for example, relating to Femtocell coverage optimisation, adaptive interference management and dynamic load balancing.

VII. Conclusions

In this paper, we show how to use verification based on model-checking and simulation to assess the collision-free and confusion-free properties when assigning PCIs to cellular base station in a SON. We specify the self-configuration procedure as a model checking problem. For this, we implement the PCI selection algorithm using PROMELA and verify the model using the SPIN model checker. Our results reveal potential assignment issues when base stations are deployed concurrently. A simple solution based on serial processing of request can alleviate this problem.

Our comparison of simulation and verification reveals their respective strengths. Verification is more rigorous when discovering incorrect behaviour but its limited scalability precludes its use in non-trivial deployment scenarios. As a solution to this problem, we describe a verification architecture for autonomous networks that exploits runtime verification. This approach reduces the number of states to be considered by the verifier but nevertheless can discover and avoid correctness problems at operation time. As future work, we plan to explore the potential of this approach. We hope that it is a first step towards provably-correct autonomous networks.

Acknowledgements

The work reported in this paper has formed part of the Flexible Networks area of the Core 5 Research Programme of the Virtual Centre of Excellence in Mobile & Personal Communications, Mobile VCE (www.mobilevce.com), and has been jointly funded by Mobile VCE's industrial member companies and the UK Government, via the Engineering and Physical Sciences Research Council (EPSRC).

References

[1] H. Ekstrom, A. Furuskar, J. Karlsson, M. Meyer, S. Parkvall, J. Torsner, and M. Wahlqvist, "Technical Solutions for the 3G Long-Term Evolution," *Communications Magazine, IEEE*, vol. 44, no. 3, pp. 38–45, March 2006.

[2] V. Chandrasekhar, J. Andrews, and A. Gatherer, "Femtocell Networks: a Survey," *Communications Magazine, IEEE*, vol. 46, no. 9, pp. 59–67, September 2008.

[3] S. Dobson, S. Denazis, A. Fernández, and other, "A Survey of Autonomic Communications," *ACM Trans. on Autonomous and Adaptive Systems (TAAS)*, vol. 1, no. 2, 2006.

[4] IBM, *An Architectural Blueprint for Autonomic Computing*. IBM, June 2005.

[5] M. Randles, D. Lamb, and A. Taleb-Bendiab, "Engineering Autonomic Systems Self-Organisation," in *5th Workshop on Eng. of Autonomic and Autonomous Systems (EASE)*, 2008.

[6] "Self-Net Project," http://www.ict-selfnet.eu/.

[7] 3GPP TS 32.500, "Telecommunication Management; Self-Organizing Networks (SON); Concepts and Requirements (Rel. 8)," 2008.

[8] E. M. Clarke, O. Grumberg, and D. E. Long, "Model Checking and Abstraction," *Principles of PL*, 1992.

[9] S. Islam, M. Sqalli, and S. Khan, "Modeling and Formal Verification of DHCP Using SPIN," *Int. Journal of Computer Science and Applications (IJCSA)*, vol. 3, Jun. 2006.

[10] V. Schuppan and A. Biere, "Verifying the IEEE 1394 FireWire Tree Identify Protocol with SMV," *Formal Aspects of Computing*, vol. 14, no. 3, pp. 267–280, 2003.

[11] C. Daws, M. Kwiatkowska, and G. Norman, "Automatic Verification of the IEEE 1394 Root Contention Protocol with KRONOS and PRISM," *Int. Journal on Software Tools for Technology Transfer (STTT)*, vol. 5, no. 2–3, 2004.

[12] 3GPP TR 32.816, "Study on Management of Evolved Universal Terrestrial Radio Access Network (E-UTRAN) and Evolved Packet Core (EPC)," Tech. Rep. 8.0.0, Dec. 2008.

[13] E. Lloyd and S. Ramanathan, "On the Complexity of Distance-2 Coloring," in *4th Conf. on Computing and Information (ICCI)*, May 1992.

[14] S. Parthasarathy and R. Gandhi, "Distributed Algorithms for Coloring and Domination in Wireless Ad Hoc Networks," in *Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, Dec. 2004.

[15] G. Holzmann, "The Model Checker SPIN," *IEEE Transactions on Software Engineering*, vol. 23, no. 5, 1997.

[16] M. Barnett and W. Schulte, "Spying on Components: A Runtime Verification Technique," in *Workshop on Spec. and Verification of Component-Based Systems (SAVCBS)*, 2001.

[17] M. Kim, I. Lee, U. Sammapun, J. Shin, and O. Sokolsky, "Monitoring, Checking, and Steering of Real-time Systems," *Electronic Notes in Theoretical CS*, vol. 70, no. 4, 2002.