Self-Organizing Systems: New Trends in Architectures and Performance Modeling

H. de Meer, P. Wüchner, A. Houyou (Eds.)
International Workshop on Self-Organizing Systems (IWSOS 2006)
September 17–21, 2006, University of Passau, Germany

PREFACE

Self-organization plays a key architectural role for the future Internet. Self-organization will enhance flexibility and evolvability of organically growing, large-scale distributed systems, e.g., of large-scale pervasive computing systems such as wireless sensor networks.

These statements could be proved by the technical program of the IWSOS 2006. Sixteen high-quality papers were selected by a thorough review process out of more than 70 submissions from 21 different countries. The technical program of the IWSOS 2006 was particularly covering application-oriented topics like: the dynamics of structured and unstructured overlays; self-organization in grids, peer-to-peer networks, wireless environments, and autonomic computing; and the application of self-organization for enhancing network management and routing.

The program of the IWSOS 2006 has been supplemented, besides the social events and tutorials, by a poster session on the role of *"Self-Organization in European Next Generation Internet"* and also by a Technical Discussion on *"Performance Modeling of Self-Organizing Systems"*.

The two parts of the technical report on hand is used to make the contributions to the poster session (Part I) and technical discussion (Part II) available to a broader community.

We are thankful to the *EuroNGI – Network of Excellence* for their financial support of IWSOS 2006 and to the organizations *GI*, *ITG*, *MMB*, and *KuVS* for ideally supporting the Technical Discussion.

An electronic version of this technical report can be found at:
`http://www.fmi.uni-passau.de/forschung/mip-berichte/MIP-0609.html`

Passau, September 2006
Hermann de Meer, Patrick Wüchner, Amine Houyou

CONTENTS

Posters on Self-Organization in European Next Generation Internet

H. de Meer, A. Houyou (Eds.)
International Workshop on Self-Organizing Systems (IWSOS 2006)
September 17–21, 2006, University of Passau, Germany

PREFACE

IWSOS 2006 has been organized in a highly interactive workshop specific fashion. Participants have been encouraged to be prepared for a more extensive discussion of the presented work than usually common at conferences. To complement the exciting program of IWSOS 2006, a poster session has been planned following the invited panel. The poster session includes reviewed posters that are also part of the IWSOS 2006 main proceedings (LNCS 4124).

These include the following posters: *"Active Element Network with P2P Control Plane"* by Michal Procházka, Petr Holub, and Eva Hladká from both Masaryk University, Botanická Brno and CESNET, Prag, Czech Republic, *"A Monitoring Infrastructure for the Digital on-demand Computing Organism (DodOrg)"* by Rainer Buchty from University of Karlsruhe, Germany, and a third poster *"Autonomic Network Management for Wireless Mesh and MANETs"* by Shafique Ahmad Chaudhry, Ali Hammad Akbar, Faisal Siddiqui, & Ki-Hyung Kim from Ajou University, Korea.

Furthermore, some additional posters have been invited to the program due to their high quality content. These posters were especially solicited from members of the European Next Generation Internet (EuroNGI) Network of Excellence. The network gathers major leading European institutions and researchers. Previously, two EuroNGI workshops on "New Trends of Network Architectures and Services" took place in Würzburg, Germany, and then in Villa Vigoni at Lake Como, Italy. The third edition of this workshop has emerged as the first IWSOS 2006.

Passau, September 2006
Hermann de Meer, Amine Houyou

CONTENTS

Designing Rules for Self-Organizing Protocols for A Cooperative Communication Architecture

Qi Zhang
COM · Denmark Technical University
Email: qz@com.dtu.dk

Frank H.P. Fitzek
Aalborg University
Email: ff@kom.aau.dk

***Abstract*— Cooperation based on hybrid cooperative communication architecture has demonstrated great potential of cooperative communication in our research. In order to achieve more cooperative gain and widely spread and deploy cooperative communication, efficient and generic self-organizing protocol has to be designed for cooperative communication.**

I. Introduction

Cooperative communication is an upcoming communication paradigm for future wireless networks. Cooperative communication has potential to resolve some crucial issues in wireless network, such as prolong standby time of the battery, increase bandwidth, enhance spectrum efficiency, improve reliability of wireless channel, and more. However the final network architecture is not clear. The basic idea of cooperation is that a group of entities working together to achieve a common or individual goal [1]. It originates from natural and human sciences and is applied in wireless communication now. The understanding of cooperative communication varies with different implementation approaches. It is classified as implicit cooperation, explicit macro cooperation and explicit micro cooperation [1]. ”Cooperation” mentioned in the paper is explicit micro cooperation.

Explicit micro cooperation exploits the potential of combined data reception/transmission, battery, and processing unit to achieve cooperation gain. It is implemented based on the hybrid cooperative communication architecture, which is given in Section II.

The communication among peer terminals in short range link works in autonomous way, without any infrastructure. So far there is no particular communication protocol or mechanism designed for short range link in cooperative network. Although Wireless Local Network (WLAN) and Bluetooth are given as candidate technologies for local short range link in [1], they have many drawbacks. Therefore, in order to work properly in tandem with cellular link in the hybrid cooperative communication architecture or to achieve higher cooperative gain, efficient self-organizing mechanism has to be designed. The cooperative communication driven self-organization design issues are discussed in Section III. Finally, conclusion is reached in Section IV.

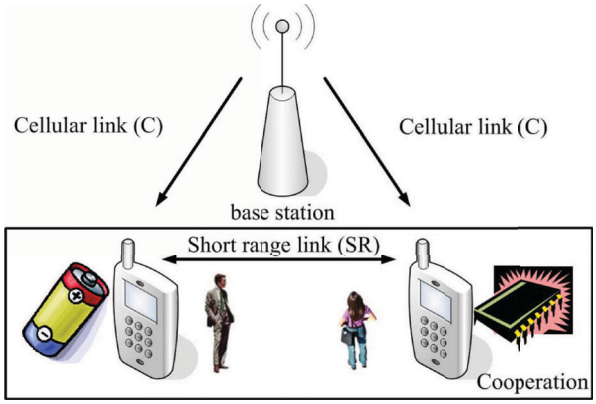

Fig. 1. Hybrid Cooperative Communication Architecture [1]

II. Cooperative Communication Architecture

A cooperative communication architecture is advocated as shown in Fig. 1 in [1]. In this architecture, terminals are capable to communicate with the base station (or access point) using the cellular link and simultaneously communicate with other peer terminals on short range links. In state of the art, the cellular link can be implemented by technologies, such as Global System for Mobility (GSM) or Universal Mobile Telecommunication System (UMTS). Bluetooth or WLAN are two candidate technologies for the short range link. In the future, a unified air interface might be used for both cellular link and short range link.

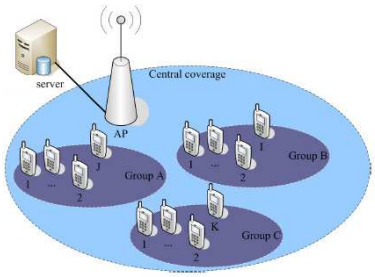

Fig. 2. Cluster based Cooperation Scenario

Based on the special characteristics of short range link (e.g. high reliability, higher data rate) and different location of peer terminals (e.g. independent propagation paths between terminals and base station), the proposed cooperative communication architecture exploits the potential of combined data reception/transmission, battery and processing unit to achieve cooperation gain.
The local cooperative communication on short range link is cluster based communication. The cluster is composed of several peer terminals, as shown in Fig. 2. The forming of cluster depends on the cooperation strategy and terminals' situations and circumstance (e.g. location, applications and services, battery status, and so on). According to the principle of cooperation, each terminal has capability of evaluating each situation and making cooperative decision (i.e. joining cooperative cluster or refusing to cooperate).

III. Cooperative Communication Driven Self-Organization Design Issues

In our hybrid cooperation architecture, due to terminal's dynamic situation and dynamic circumstance, terminal needs to adapt its cooperative decision from time to time in autonomous way. Consequently, the size of cluster varies, correspondingly. It means the communication mechanism in cluster should be very flexible, adaptive, scalable, failure robust.
Although many cooperation scenarios (e.g. Multiple Description Coding (MDC)/ Multiple Layer Coding (MLC) video services, Cooperative header compression, and so on) have shown great potential of cooperative gain in this novel communication paradigm, they rely on the assumption that efficient cooperative protocol (or mechanism) exists in short range link. The key issue here is that, so far, no particular cooperative protocol has been designed for short range link.
Currently, WLAN and Bluetooth technologies are used in our cooperative communication testbed or prototype demonstration, due to the state of art technology limitation. However, they are inefficient when applied in short range link communication, because of their intrinsic features, which affect achievable cooperative gain. For instance, one master and seven slaves piconet structure in Bluetooth limits the size of cluster. CSMA/CA medium access mechanism used in WLAN has bad performance (high conflict) and deteriorates cooperative gain, when the terminals increase.
Therefore, efficient self-organizing protocols supporting cooperation within the cluster is needed. The protocol should address the following actions.

- Cluster forming and discovery: it represents how terminal finding possible partner or group of partners. Which spectrum to use in the cluster is also an important issue here. Terminals can use predefined spectrum or might use cognitive radio later on.
- Cluster splitting and merge: it relates cluster's scalability and interference issue. Hence, intelligent power control algorithm should be investigated here.
- Cluster maintenance: it means that members of cluster how to evaluate a new applicant (members of cluster can refuse new applicant to join in the cluster) and how to update cluster membership if accepting the new member or some members leaving the cluster.
- Adaptive cooperative decision: it requires terminal is capable of doing timely cooperative gain evaluation and making corresponding decision.

IV. Conclusion

Based on the proposed hybrid cooperative communication architecture and our research results in the field, we have seen great potential of cooperative communication for future wireless networks. However, the success of cooperative communication relies on efficient self-organizing protocols for communication on short range links. Therefore, designing efficient self-organizing protocol is of significant importance to promote development and wide application of cooperative communication in wireless network.

References

[1] Frank H. P. Fitzek and Marcos D. Katz, "Cooperation in Wireless Networks: Principle and Applications", 2006, ISBN-10 1-4020-4710-X

AntPing - prototype demonstrator of swarm based path management and monitoring

Poul E. Heegaard*/**, Ingebrigt Fuglem*
* Telenor R&D
** Department of Telematics, Norwegian University of Science and Technology, Norway

***Abstract*— AntPing is a prototype implementation of a distributed and robust path management and monitoring system. The AntPing is running on small home routers and visualises the inner workings of the ant algorithm that establishes, maintains, and monitors virtual paths. The demo network used is a replica of the Norwegian ISP Telenor's core IP network with 10 nodes. The demonstrator visualises how ants are moving and dropping in the network. The animation is live and the audience is allowed to plug and unplug the links between nodes and the power supply to the nodes. The changes in current and historical cost values of each virtual path are plotted as a function over time.**

***Index Terms*— prototype, demonstrator, openwrt, click, path management, monitoring**

I. Introduction

VIRTUAL path management in dynamic networks poses a number of challenges related to combinatorial optimisation, fault and traffic handling. Ideally such management should react immediately on changes in the operational conditions, and be autonomous, inherently robust and distributed to ensure operational simplicity and network resilience. Swarm intelligence [1] based self-management is a candidate potentially able to fulfil these requirements. This was first introduced in Schoonderwoerd [2] where multiple with a behaviour inspired by ants were used to solve problems in telecommunication networks, and later pursued further by others, see for instance [3], [4], [5], [6] and references therein.

The *CE-ants* implemented in the demonstrator in this paper, is a distributed system designed for dealing with path management in communication networks based on swarm intelligence. CE ants and their application are presented in [7]. A CE-ants system is inspired by the foraging behaviour of ants. The idea is to let ants iteratively search for paths from a source to a destination node in a network. When a path is found the ant returns to the source on the reversed path and leaves markings on every intermediate node, denoted *pheromones* (resembling the chemicals left by real ants during ant trail development). The strength of the pheromones depends on the quality of the path found. The following searching ants stochastically select the next hop based on the current pheromone distribution. The overall process converges quickly toward a near optimal path. The CE-ants applies a distributed variant of Rubinstein's method developed for stochastic optimisation [8] to determine the best possible updates of pheromones based on the cost value of the last path found combined with recent historical cost values. See [6] for more details.

Proper path management with service guarantees requires high quality measurement data. It is a great challenge to obtain sufficient, necessary, correct, and fresh data for various network management and traffic engineering aspects, security and fraud detection, and accounting. Performance monitoring collects and aggregates quality attributes about services. This is of interest to verify that a service is delivered according to the specifications as given in Service Level Agreements. As a supplement to best practice in performance monitoring the demonstrator in this paper illustrates the potential in using the indices of a complex adaptive system as CE-ants. The idea is to observe temporal variables of CE-ants and illustrate how they change as the network changes and, hence, as the corresponding quality of the service delivered on this network changes. Previous studies [9] identified that several adaptive components of such a system can be used as indicators of the network condition status with respect to traffic load level and topology. The indicators include the stochastic routing matrix (the pheromone), routing path probability, cost values, and grade of convergence (denoted temperature in CE-ants).

This paper describes an prototype implementation of CE-ants. The technical demo implementation is briefly described in Section II while Section III explains what is demonstrated. Comments on possible extensions are found in Section IV.

II. AntPing implementation

Implementing a working prototype provides useful insights in the complexity of swarm-based methods in real routers, and detects potential performance bottleneck. The first approach was based on *Click* Modular software router system [10]. The prototype in [11] is based on a Mobile Agent System (Java based). However, the solution suffered from severe performance limitations even in a small-scale demo network.

To improve the performance, the *AntPing* is implemented without the mobile agent system. The ants are no longer mobile agents but simple IP packets. AntPing extends the *Click* Modular software router system, and uses *hping3* [12] to generate and receive ant-packets from source to destination. The AntPing is running on home routers, see Figure 1 for a picture from the lab) with OpenWRT Linux [13], see [14] for more details. This implementation has quite modest hardware and software requirements, which makes the demo inexpensive, flexible, and easily portable.

This work was also partially supported by the Future & Emerging Technologies unit of the European Commission through Project BISON (IST-2001-38923).

Fig. 1. Setup from the lab (LinkSys WRT54GS(v4.0) routers).

III. Visualisation & interaction in AntPing

The purpose of the demonstrator is to illustrate the inner workings of a swarm based method and to provide an interactive technical installation. The ant algorithm is animated live by use of Network Animator (*nam* [15]) showing how ants are moving and dropping in the network, and how the topology is changing with link and node failures and restorations. The animation shows ants that do not find the destination but are dropped because the TTL is expired. In addition, the changes in cost values of each virtual path are plotted live by use of *gnuplot* [16] as a function over time, both the cost of the current best path, and the last cost. In Figure 2 an illustration of the demo network topology is given, together with two screen snapshots showing the nam and gnuplot visualisations.

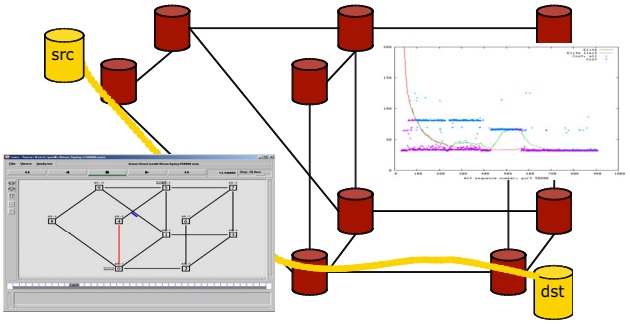

Fig. 2. Demonstrating virtual path detection, management, and monitoring

It is up to the audience to introduce the network dynamics. They may unplug and plug the cables between the nodes, and the power supply to the nodes. Adding new interfaces or links that are not predefined is currently not possible because a discovery protocol is not yet implemented.

IV. Conclusions

The AntPing visualises the inner working of a distributed path management and monitoring system. It is executed on a small, 10-node network, and no scalability or performance testing is conducted. However, the CE-ants method that AntPing implements is thoroughly tested on a number of applications using a *ns-2* simulator, see [17], [5], [18]. Furthermore, a series of simulations is conducted test the scalability and to compare with ICMP Ping over link state routing (OSPF, ISIS) [14], and to reduce the overhead by focusing on updating only the best paths [19] and only when the network state has changed [20], [21].

References

[1] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm Intelligence: From Natural to Artifical Systems*. Oxford University Press, 1999.

[2] R. Schoonderwoerd, O. Holland, J. Bruten, and L. Rothkrantz, "Ant-based load balancing in telecommunications networks," HP Labs, Tech. Rep. HPL-96-76, May 1996.

[3] M. Dorigo and G. Di Caro, "Ant Algorithms for Discrete Optimization," *Artificial Life*, vol. 5, no. 3, pp. 137–172, 1999.

[4] G. Di Caro and M. Dorigo, "AntNet: Distributed Stigmergetic Control for Communications Networks," *Journal of Artificial Intelligence Research*, vol. 9, pp. 317–365, Dec 1998.

[5] O. Wittner and B. E. Helvik, "Distributed soft policy enforcement by swarm intelligence; application to load sharing and protection," *Annals of Telecommunications*, vol. 59, no. 1-2, pp. 10–24, Jan/Feb 2004.

[6] O. Wittner, "Emergent behavior based implements for distributed network management," Ph.D. dissertation, Norwegian University of Science and Technology, NTNU, Department of Telematics, November 2003.

[7] P. E. Heegaard, O. Wittner, and B. E. Helvik, "Self-managed virtual path management in dynamic networks," in *Self-* Properties in Complex Information Systems*, ser. Lecture Notes in Computer Science, LNCS 3460 (ISSN 0302-9743), O. Babaoglu, M. Jelasity, A. Montresor, A. van Moorsel, and M. van Steen, Eds. Springer-Verlag GmbH, 2005, pp. 417–432.

[8] R. Y. Rubinstein, "The Cross-Entropy Method for Combinatorial and Continuous Optimization," *Methodology and Computing in Applied Probability*, pp. 127–190, 1999.

[9] P. Heegaard, "Performance monitoring of routing stability in dynamic networks," Section 5 in Deliverable 05: "Models for basic services in AHN, P2P and Grid networks" in IST-FET Project BISON (IST-2001-38923), BISON (IST-2001-38923), December 2003.

[10] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, "The click modular router," *ACM Transactions on Computer Systems*, vol. 18, no. 3, pp. 263–297, August 2000.

[11] A. Mykkeltveit, P. Heegaard, and O. Wittner, "Realization of a distributed route management system on software routers," in *Proceedings of Norsk Informatikkonferanse*, Stavanger, Norway, 29. Nov - 1. Dec 2004.

[12] hping. [Online]. Available: http://wiki.hping.org/

[13] OpenWRT. [Online]. Available: http://wiki.openwrt.org/FrontPage

[14] P. Heegaard and I. Fuglem, "Demonstrator 1: Ant-based monitoring on software ip routers," BISON (IST-2001-38923), Tech. Rep., 2006. [Online]. Available: http://www.cs.unibo.it/bison/deliverables/D14.pdf

[15] Nam: Network animator. [Online]. Available: http://www.isi.edu/nsnam/nam/

[16] Gnuplot. [Online]. Available: http://www.gnuplot.info/

[17] O. Wittner, P. E. Heegaard, and B. E. Helvik, "Swarm based distributed search in the amigos environment," Department of Telematics, Norwegian University of Science and Technology, AVANTEL Technical Report ISSN 1503-4097, December 2003.

[18] O. Wittner, B. E. Helvik, and V. F. Nicola, "Internet failure protection using hamiltonian p-cycles found by ant-like agents," *Journal of Network and System Management, Special issue on Self-Managing Systems and Networks*, vol. Submitted, 2005.

[19] P. E. Heegaard, O. Wittner, V. F. Nicola, and B. E. Helvik, "Distributed asynchronous algorithm for cross-entropy-based combinatorial optimization," in *Rare Event Simulation and Combinatorial Optimization (RESIM/COP 2004)*, Budapest, Hungary, September 7-8 2004.

[20] P. E. Heegaard and O. J. Wittner, "Self-tuned refresh rate in a swarm intelligence path management system," in *Proceedings of the EuroNGI International Workshop on Self-Organizing Systems (IWSOS 2006)*, ser. LNCS. University of Passau, Germany: Springer, September 18 - 20 2006.

[21] P. E. Heegaard and O. Wittner, "Restoration performance vs. overhead in a swarm intelligence path management system," in *Proceedings of the Fifth International Workshop on Ant Colony Optimization and Swarm Intelligence (ANTS2006)*, ser. LNCS, M. Dorigo and L. M. Gambardella, Eds. Brussels, Belgium: Springer, September 4-7 2006.

Stability and Systemic Threats in Ecologies of Services

Elpida S. Tzafestas
Institute of Communication and Computer Systems
National Technical University of Athens
Zographou Campus 15773, Athens, GREECE.
brensham@softlab.ece.ntua.gr

***Index terms*—artificial immune system, virtual organization, antibodies, specificities, stability**

EXTENDED ABSTRACT

We are addressing the class of virtual organizations where an entrepreneurial or other activity acts as an interface between a number of existing remote services or applications and a number of customers that should interact with them in complex ways. For example, such activities include private e-Government middle services that specialize in carrying out complex tasks with many different state administration services, or commercial houses specializing in centralizing and organizing large complicated orders that involve a number of different vendors scattered around the electronic world. Because the integrated services/applications are remote and external to the middle entity responsible for the organization, the current practice is to manage the customer tasks mostly manually or in conventional ways, such as calling external entities and negotiating prices and delivery delays.

The INFRAWEBS[1] framework is being designed and developed with the vision to help business people build composite applications that tackle much of this interaction complexity as automatically as possible, with the role of human intervention not necessarily being fully dismissed. Such virtual organizations are characterized by operational and performance criteria that are independent and even sometimes incompatible with those of individual components (external services/applications) involved. This parallels the organization of multi-cellular biological organisms where the integrity of the organism as a whole is defined differently than for each of its constituent cells, and indeed many individually effective functionalities may be harmful from a systemic point of view, such as for example tumor cells, parasites etc. Because of this property, it is expected that an artificial immune system can be designed to treat integrity issues in the same way that a natural immune system does so within a multi-cellular organism.

Within such an "ecology" of services, it makes no sense to regard individual components (services) as safe or unsafe, which is the usual approach in classical ICT security research. Instead, the same service may behave safely or unsafely for the overall organization (and even for itself) according to internal state, specifics of the requests processed, timings etc. The security or functionality management problem in this framework can be formulated now as a problem of managing information about information, i.e. information about message from service to service and is therefore declarative, in the sense of understanding abstract relations and flow.

Because of the above, our approach focuses on the detailed examination of the history of exchanged messages with services to identify potential threats and only a little if any explicit component authentication is necessary. This approach sets itself in the trend of information flow regulation at runtime.

The artificial immune system is defined as a security "shell" that constitutes the control system for the autonomous virtual ecology and filters incoming and outcoming information to external services or other types of active resources. This shell functions in parallel with the regular activities of the virtual organization and is responsible for recognizing hostile resources, i.e. resources that are malicious, malfunctioning or just slow. Attacks and insecure cases such as, but not limited to, unavailability, improper treatment, unacceptability of communicated information, denial of service etc. can in principle be smoothly integrated in this configuration.

The operation of the artificial immune system mirrors exactly that of the natural immune system of first order. The natural immune system recognizes and attacks alien bodies by not allowing them to be metabolized by the body of the organism. In the same way, the security shell recognizes and captures alien or suspect messages before they are processed by the organization. A population of specialized artificial cells (that correspond to the antibodies), are triggered by the presence of the alien message and proliferate very rapidly and only as much as necessary to ensure clobbering of the alien population. Certainly, under certain conditions, for example when the alien attack is extremely severe or when there are multiple attacks concurrently, the immune system occupies a large amount of the resources of the central organization and as a result the latter is hindered in its normal operation. In nature, advanced

[1] www.infrawebs.org

organisms, such as vertebrates, possess also a second order level of operation, in which the population of the antibodies records information from past experience, i.e. alien body features or "templates" used for identification, so as to make resistance automatic the next time the same threat appears.

The virtual organization (the body) is a highly connected network of representations or "images" of external components (somatic cells), where for each link a "preference degree" or "degree of trust" is present. This metric is continuously monitored and updated by the security shell, according to the immune algorithm. While particular details are bound to be case-specific, in general the immune algorithm has the following properties :

- The artificial immune network regulates input-output flow between component images. It is noteworthy that this flow happens only and is meaningful only within the particular virtual organization concerned. Unlike natural organisms, several organizations (equivalent to bodies) may include the same or some of the same services (equivalent to cells), so that the individual functionality of one particular service may be benevolent for one organization and malevolent for another one.
- The immune network consists of a number of nodes ("cells" or "antibodies"), each responding to one type of service message. Types are to be defined within the scope of particular applications, but as a general rule they use actual service properties or metrics. Two general-purpose antibody classes are price-related ones (e.g. "price > 100" or "price between 80 and 120") and delay-related ones (e.g. "delay < 3h").
- Many such different types have to be present and/or generated for the algorithm to work properly. For example, in the above case many variants of the type "price > X" have to be present and/or generated for different values of X. The types correspond to the biological notion of **specificity**.
- The algorithm relies on emergent population effects, hence a number of clones for each antibody may be present at any moment. Each antibody has an **activation** level, that expresses a deviation for a target value. The relations between different activation levels are dynamic through external (service-specific) activity and through individual antibody adaptation.

Global consistency can be achieved and monitored by the artificial immune system that self-organizes according to application-specific criteria. In normal conditions, an organization stabilizes and remains in a global safe state where individual antibodies have obtained stable activation levels. Any abrupt change in one monitored condition will cause corresponding abrupt changes in the activation level of one or more antibodies and this in turn will cause intensive re-organizational activity. Self-organization algorithms used for this purpose will necessarily integrate both absolute global properties (for example, antibodies of the same type but far away in the network may be correlated and co-develop) and inter-service properties (for example, a message not confirmed by the recipient virtual node may be weaker next time).

The goal of the organization is to find reliable and secure, hence stable, pathways that fulfill the user-requested goal. Appropriate antibody pools can be designed off-line by the application designer and with the aid of simulations to monitor such stability. Unexpected changes in the global activation state compromise the organization's functionality and make it appear insecure to the user. However, persistent changes are gradually incorporated by the system and drive it to a new functional equilibrium.

In principle, the artificial immune system can discover also novel systemic threats by correlating apparently disparate antibody activation patterns. One prerequisite to this is to be able to describe computationally many conditions and translate them to an antibody formalism – the INFRAWEBS semantic web derived-approach allows precisely to exploit and share arbitrary semantic service descriptions. It would also be beneficial if some of the antibodies could be defined recursively to "reason" on the state of other antibodies and not on the state of services themselves. Within this extended framework we can expect an artificial immune system that continuously generates new antibodies to discover dysfunctions, threats and deficiencies. Current virtual organizations are becoming more and more complex. Future research will focus on specifics of a few of them to demonstrate the behavior and the potential of this artificial immune systems approach to organization management and protection.

REFERENCES

[1] H. Bersini, "The Endogenous Double Plasticity of the Immune Network and the Inspiration to be drawn for Engineering Artifacts", in *Artificial Immune Systems and Their Applications*, Springer Verlag, 1999, pp.22–44.

[2] D. Dasgupta, F. Gonzalez, "Artificial Immune Systems in Intrusion Detection", Chapter 7 in *Enhancing Computer Security with Smart Technology* by V. Rao Vemuri (Ed.), Auerbach Publications, November 2005, pp. 165-208.

[3] J. Goldenberg, Y. Shavitt, E. Shir, S. Solomon, "Distributive immunization of networks against viruses using the 'honey-pot' architecture", *Nature*, December 2005.

GI/ITG/MMB/KuVS Technical Discussion (Fachgespräch) on

Performance Modeling of Self-Organizing Systems

H. de Meer, P. Wüchner, J. B. Schmitt, M. Hollick (Eds.)

September 21, 2006, University of Passau, Germany
co-located with:
EuroNGI IA.8.2 – New Trends in Network Architectures and Services:
International Workshop on Self-Organizing Systems (IWSOS 2006)
September 18–20, 2006, University of Passau, Germany

PREFACE

To supplement the program of the IWSOS 2006, amongst other events, the technical discussion on performance modeling of self-organizing systems was initiated. The technical discussion was in contrast to the workshop less focused on the practicality of self-organizing appliances but more on the mathematical tractability of self-organizing systems.

The particular challenge in building mathematical models for self-organizing systems lies in their complexity. Driven by randomness and feedback, the relation between cause and effect of these systems may appear chaotic: minor causes may have severe impact, whereas seemingly major causes may only have a small effect. Traditional mathematical models tend to be unsuitable to model self-organizing systems. Models to be developed should be simple to remain scalable to the huge number of entities in the systems under investigation and to be generally applicable.

The technical discussion aimed at joining together different points of view provided by the participating researchers. Especially discussions on the methodology for modeling and evaluating the performance and reliability of complex self-organizing systems were deepened.

TECHNICAL PROGRAM CO-CHAIRS

- Hermann de Meer, University of Passau, Germany
- Patrick Wüchner, University of Passau, Germany
- Jens B. Schmitt, University of Kaiserslautern, Germany
- Matthias Hollick, University of Technology Darmstadt, Germany

PROGRAM

09:00–09:10	Welcome Address
09:10–10:00	P. Wüchner, H. de Meer (University of Passau): *Discrete-Event System Performance Modeling of Self-Organizing Systems*
10:00–10:20	Coffee Break
10:20–11:10	P. Merz (University of Kaiserslautern): *Self-Organizing Networks for Performance Optimization*
11:10–12:00	T. Krop, T. Lange, M. Hollik, R. Steinmetz (University of Darmstadt): *Modell zur Analyse des Routenfortschritts in Ad Hoc Netzen*
12:00–13:00	Lunch Break
13:00–13:50	P. Becker, R. Gotzhein, T. Kuhn (University of Kaiserslautern): *Performance Simulation of Distributed Embedded Self-Organizing Systems Modeled with SDL*
13:50–14:40	J. B. Schmitt (University of Kaiserslautern): *Worst Case Modelling of Wireless Sensor Networks*
14:40–15:00	Coffee Break
15:00–15:50	E. S. Tzafestas (University of Athens, Greece): *Measures of Developing Stability and Perturbations in Ecologies of Semantic Web Services* (not included in technical report)
15:50–16:15	Closing Discussion

CONTENTS

Discrete-Event System Performance Modeling of Self-Organizing Systems

Patrick Wüchner and Hermann de Meer
Chair of Computer Networks and Computer Communications,
Faculty of Computer Science and Mathematics,
University of Passau,
Innstr. 43, 94034 Passau, Germany
Email: {Patrick.Wuechner, Hermann.DeMeer}@Uni-Passau.de

***Abstract*— The contribution of this paper is twofold. On the one hand the authors give a brief survey on existing publications on the performance modeling of discrete-event systems and on the existing literature on the modeling of self-organizing systems. On the other hand an evaluation is started towards answering the question if modeling techniques designed for discrete-event systems are capable of describing self-organizing systems. We demonstrate that self-organizing systems share many properties of discrete-event systems. Thus, the well-known methods developed for the performance investigation of discrete-event system performance seem to be attractive for the performance evaluation of self-organizing systems.**

I. Introduction

Self-organization is supposed to play a decisive role in future communication systems, especially in the Internet. With the help of self-organizing systems, developers of already complex and still further growing communication infrastructures hope to keep their system manageable, scalable, configurable, repairable, and last but not least high-performing.

Thus, to see the benefit of self-organizing systems it is worthwhile to investigate modeling techniques suitable for evaluating the performance of these systems. Particularly with regard to complexity and nonlinearity, the modeling of self-organizing systems and the evaluation of these models is a non-trivial task.

This paper lists some ideas how to deal with this problem and discusses to which degree discrete-event system models are suitable to model the behavior of self-organizing systems. The investigation is consciously kept general and abstract to provide a common universal basis for multiple applications like sensor networks and ubiquitous computing, ad-hoc and group-forming networks, P2P networks, and others.

The paper is organized as follows. In Section II modeling techniques for discrete-event systems are reviewed. In Section III the paper introduces the properties of self-organizing systems. Section IV is used to discuss which properties complicate the modeling of self-organizing systems. Furthermore, Section IV investigates if self-organizing systems can be modeled and evaluated easily using the well-known methods developed for discrete-event systems.

II. Discrete-Event Systems

To set a basis for discussion, we will recapitulate the definition and different modeling and evaluation techniques for discrete-event systems (DESs).

A. Working Definition

Referring to a point of view which is widely accepted in large parts of the research community (see, e.g., [1]), a system in general can be described as a set of objects that depend on each other or interact with each other in some way to fulfill some task. While doing this, the system might be influenced by the system's environment. It is essential to set clear boundaries between the system and its environment.

At this point it is worthwhile to define some system components (see [1]):

- *Entity:* An entity is an object of interest in the system, whereby the selection of the object of interests depends on the purpose and level of abstraction of the study.
- *Attribute:* Attributes are used to describe the properties of the system's entities.
- *State:* The state of the system is a set of variables that is capable of characterizing the system at any time.

- *Event:* An event is an instantaneous incidence that might result in a state change. Events can be endogenous (generated by the system itself) or exogenous (induced by the system's environment).

Systems can be classified as continuous-state or as discrete-state systems (see [1]). In *continuous-state* systems the state variables change continuously over time. By contrast, the state of *discrete-state* systems only changes at arbitrary but instantaneous points in time.

An orthogonal classification of systems is based on the distinction if the progress of the dynamic system state is pushed forward by time or by the occurrence of events (see [2]). While in *time-driven* systems all state changes are synchronized by the system clock, in *event-driven* systems events occur asynchronously and possibly concurrently/simultaneously.

Thus, a DES (see, e.g., [1] or [2]) is a discrete-state, event-driven (not time-driven) system, i.e., the state changes of the system depend completely on the appearance of discrete events over time.

B. Modeling Techniques

Various approaches to the modeling of (discrete-event) dynamic systems exist (see [3]). These approaches can be divided into *logical models* and *timed models*. The latter are also known as *performance models* and can further be split into the classes of *deterministic* (or *non-stochastic*) and *stochastic* models.

Logical models and non-stochastic timed models are primarily used in control theory (see, e.g., [4]). These models mainly consider the logical aspects of the system, e.g., the order of events. Although it might be interesting to apply control theory to self-organizing systems, these models are not within the scope of this paper.

Thus, in this paper we focus on stochastic models of discrete-event systems. Event-driven systems may be modeled in *continuous time* or in *discrete time*. Several modeling techniques exist, developed especially for handling performance aspects of discrete-event systems, including *Markov chains*, *queueing networks*, and stochastic *Petri nets*. A thorough investigation of these well-established techniques can be found in [5].

C. Evaluation Techniques

Various evaluation techniques exist for the modeling techniques mentioned in Section II-B (see, e.g., [5]). These evaluation techniques include analytical/numerical methods (e.g., Markovian analysis, algorithms for product-form queueing networks) as well as (discrete-event) simulation.

Especially when dealing with very complex systems the following drawbacks can be observed:

- With Markovian analysis, it is necessary to create the whole state space (e.g., in form of generator matrices of continuous-time Markov chains). Due to the finite memory of computer systems, the calculation of results heavily suffers from state-space explosion. Furthermore, Markovian analysis can only be done for models that fulfill the Markovian property (i.e., state sojourn times have to be memoryless, i.e., have to be exponentially or geometrically distributed).
- Algorithms for product-form queueing networks require queueing network models in product form, which is a hard constraint. Furthermore, queueing networks lack means for handling synchronization and concurrency of system components.
- To get narrow confidence intervals it is necessary to simulate complex systems with long simulation runs and/or with a large number of simulation runs, especially in the presence of rare events. Thus, although simulation is a versatile tool because it can handle stochastic processes that fall into the class of generalized semi-Markov processes, it is a very time-consuming method for deriving results.

Of course, workarounds have been presented to these and further problems (see, e.g., [5], [6], or [7]). These include largeness avoidance, state space truncation, lumping, decomposition, fluid models, stiffness avoidance, stiffness tolerance, phase approximations, and simulation speed-up techniques.

Especially this plentifulness of methods makes it worthwhile to investigate if they can be used efficiently in the modeling process and evaluation of self-organizing systems.

III. Self-Organizing Systems

The term "self-organization" (SO) is widespread in science among various disciplines. Interdisciplinary lectures at the University of Passau have shown that it is difficult to find a common interpretation.

A. Working Definition

We utilize the view on self-organization that emerged within the USENET Newsgroup *comp.theory.self-org-sys* and was summarized in the "Self-Organizing Systems (SOS) FAQ"[1].

[1]Version 2.99, July 2006, http://www.calresco.org/sos/sosfaq.htm

There, SO is defined as follows:

- The evolution of a system into an organized form in the absence of external pressures.
- A move from a large region of state space to a persistent smaller one, under the control of the system itself. This smaller region of state space is called an attractor.
- The introduction of correlations (pattern) over time or space for previously independent variables operating under local rules.

The following typical features of SO are listed in rough order of generality:

- Absence of external control (autonomy)
- Dynamic operation (time evolution)
- Fluctuations (noise/searches through options)
- Symmetry breaking (loss of freedom/heterogeneity)
- Global order (emergence from local interactions)
- Dissipation (energy usage/far-from-equilibrium)
- Instability (self-reinforcing choices/nonlinearity)
- Multiple equilibria (many possible attractors)
- Criticality (threshold effects/phase changes)
- Redundancy (insensitivity to damage)
- Self-maintenance (repair/reproduction metabolisms)
- Adaptation (functionality/tracking of external variations)
- Complexity (multiple concurrent values or objectives)
- Hierarchies (multiple nested self-organized levels)

Similar views are (partly) shared by several researchers including, e.g., [8], [9], and [10].

B. Application Areas of Self-Organizing Systems

As stated before, "self-organization" was introduced into the vocabulary of researchers from many disciplines. Correspondingly, the field of application areas for self-organizing systems (SOSs) has grown quite large. For example, SOSs were related to: peer-to-peer systems (see [11] or [12]), social networks (see [13]), artificial neural networks (see [14]), and many more.

C. Performance Modeling Techniques

The approaches towards modeling self-organizing systems are mostly application area specific. They reach from very abstract, mathematical models based on Markov chains (like given in [15]) to application-oriented models, e.g., based on network calculus (see, e.g., [16]).

A widely preferred evaluation technique of SOS models is simulation, used, e.g., in [13] for studying social networks, in [17] for studying sensor networks, in [18] for studying neural networks, in [19] for studying mesh transport networks, in [20] for studying neuro mechanical networks, or in [21] for studying load balancing in grids. It is also interesting that although simulation is used a lot for evaluating SO models, the formal specifications of the underlying (discrete-event) system models are given rarely. Furthermore, no discussion could be found within these publications, why simulation was preferred to analytical solutions.

IV. Discrete-Event Self-Organizing Systems

To check if SOSs systems can be modeled appropriately by DESs, selected features of SOSs that were listed in Section III-A will now be discussed and related to DESs in more detail.

A. Autonomy

The autonomy of an SOS has to be distinguished from total autarchy. The latter does not exist in SOSs (see [9]). In contrast to total autarchy, interchanges (of, e.g., information, space, energy, or perturbations) between the SOS and its surrounding environment occur. In addition to these stimulations, the environment must not impose any control upon the SOS.

In terms of DESs, these stimulations can be modeled as exogenous events, e.g., in form of workload, job, token, or customer arrivals, as known from higher-level DES modeling formalisms like queueing networks or stochastic Petri nets.

B. Dynamics

As an event-driven system, a DES will have the facility to proceed as long as exogenous events occur. If no events are invoked by the environment, the DES's progress will depend on the existence of endogenous events. Regarding this, DESs show the same properties as SOSs.

C. Fluctuations

The term fluctuation may refer to two different aspects.

On the one hand, it describes fluctuation in the timing of events. This can quite easily be handled by stochastic DES performance models to some extend via defining stochastic distributions for inter-event times.

On the other hand, the means for modeling fluctuation of the SOS's structure is limited in DES models. One could discuss whether fluctuation in structure can be modeled by using routing probabilities as known from queueing networks or by using some construct of immediate transitions with marking dependent weights

(in the following abbreviated as MDWITs) in stochastic Petri nets to let the workload choose only entities (paths/transitions and nodes) that are set active for the current structure.

Usually, SOSs are too complex to know all possible shapes of the SOS's structure in advance. Thus, the approach will not be viable without vast abstractions.

D. Symmetry Breaking

Symmetry breaking is closely related to bifurcation. Bifurcation causes sudden qualitative or topological changes of the SOS's behavior. Analogously to fluctuations, bifurcations could — respecting the limitations — be modeled using concepts like MDWITs.

E. Local interactions

To ensure the locality of interactions, MDWITs could be defined in a way that they depend on the states of near-by entities only. For this, however, a metric for the term "near-by" has to be defined. Furthermore, it has to be discussed how this can be modeled dynamically and conveniently using high-level model descriptions.

F. Emergence

The investigation of emergence of the SOS towards a global order is the purpose of the study of SOS models. It is particularly interesting to investigate the properties of the attractors the SOS may move to.

Transient evaluation of short, mid, and long-term behavior could be observed using well-known analytical/numerical or simulation methods – presuming SOSs can be modeled as DESs.

G. Energy Usage

In the scope of DESs, energy usage could be modeled as stimulations from the environment that get dissipated by the SOSs. As already mentioned in Section IV-A, these stimulations can be modeled as exogenous events.

The dissipation of the energy can be modeled, e.g., by join constructs (known from fork-join systems described in [5]) that can be easily modeled using, e.g., stochastic Petri nets.

H. Nonlinearity

Nonlinearity results from feedback. Feedback can be positive or negative (see [9]). Positive and negative feedback could be modeled by fork and join constructs, respectively, eventually combined with state or marking-dependent transition rates and/or arc multiplicities.

Nevertheless, the nonlinear behavior of SOSs is usually more complex and this makes it hard to model SOSs and to evaluate these models. It is unlikely that this complex behavior can be (approximatively) described appropriately by linear systems of first order differential equations as it is done for DESs.

I. Further Features

The investigation of *multiple equilibria* is related to the investigation of the SOS's behavior (see IV-F).

Criticality (in particular *threshold effects*) could be modeled using MDWITs (see IV-C).

Redundancy and *repair mechanisms* could be handled by well-known methods taken from performability modeling (see, e.g., [6] or [22]).

Model *hierarchies* are also discussed in literature for several years (see, e.g., [5]).

V. Conclusion and Future Work

Although initial results for the investigation of correlations of DESs and SOSs were presented in this paper, the modeling of SOSs in an universal way and the evaluation of these models remains a non-trivial task.

Not all features of SOSs could be checked in detail yet whether they restrict or even prohibit the modeling and evaluation of SOSs with methods originally deployed for DESs. More time and research effort will be needed to complete a thorough investigation.

In near future, we plan to build a more detailed survey and classification of models used by researchers dealing with SOSs. From these models we might be able to learn how to extend the capabilities of DESs to increase their flexibility while modeling SOSs.

Particularly, we will have a look at modeling and evaluation techniques for nonlinear systems.

Acknowledgment

Parts of this work have been accomplished under support of the Euro-NGI – Network of Excellence, EC grant IST–507613.

References

[1] J. Banks, J. S. Carson II, B. L. Nelson, and D. M. Nicol, *Discrete-Event System Simulation*, 3rd ed. Prentice-Hall, 2001.

[2] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. Kluwer Academic Publishers, September 1999.

[3] P. J. G. Ramadge and W. M. Wonham, "The control of discrete event systems," in *Proceedings of the IEEE*, vol. 77, no. 1. IEEE Computer Society, jan 1989.

[4] J. G. Thistle, "Logical aspects of control of discrete event systems: A survey of tools and techniques," in *Proc. 11th International Conference on Analysis and Optimization of Systems – Discrete Event Systems*, ser. Lecture Notes in Control and Information Sciences, G. Cohen and J.-P. Quadrat, Eds., vol. 199. Berlin: Springer-Verlag, June 1994, pp. 3–15.

[5] G. Bolch, S. Greiner, H. de Meer, and K. Trivedi, *Queueing Networks and Markov Chains*, 2nd ed. New York: John Wiley & Sons, 2006.

[6] K. S. Trivedi, G. Ciardo, M. Malhotra, and S. Garg, "Dependability and performability analysis using stochastic Petri nets," in *Proc. 11th International Conference on Analysis and Optimization of Systems – Discrete Event Systems*, ser. Lecture Notes in Control and Information Sciences, G. Cohen and J.-P. Quadrat, Eds., vol. 199. Berlin: Springer-Verlag, jun 1994, pp. 144–157.

[7] P. Wuechner, H. de Meer, J. Barner, and G. Bolch, "MOSEL-2 – a compact but versatile model description language and its evaluation environment," in *Proc. MMBnet'05, Hamburg*, B. Wolfinger and K. Heidtmann, Eds. University of Hamburg, sep 2005, pp. 51–59.

[8] G. Di Marzo Serugendo, N. Foukia, S. Hassas, A. Karageorgos, S. K. Mostéfaoui, O. F. Rana, M. Ulieru, P. Valckenaers, and C. Van Aart, "Self-organisation: Paradigms and applications," in *Proc. 11th International Conference on Analysis and Optimization of Systems – Discrete Event Systems*, ser. Lecture Notes in Computer Science, vol. 2977. Springer-Verlag, 2004, pp. 1–19.

[9] H. De Meer and C. Koppen, "Characterization of self-organization," in *Peer-to-Peer Systems and Applications*, ser. Lecture Notes in Computer Science, R. Steinmetz and K. Wehrle, Eds., vol. 3485. Springer-Verlag, 2005, pp. 227–246.

[10] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biological Cybernetics*, vol. 43, no. 1, pp. 59–69, jan 1982.

[11] H. De Meer and C. Koppen, "Self-organization in peer-to-peer systems," in *Peer-to-Peer Systems and Applications*, ser. Lecture Notes in Computer Science, R. Steinmetz and K. Wehrle, Eds., vol. 3485. Springer-Verlag, 2005, pp. 247–266.

[12] J. Ledlie, J. Taylor, L. Serban, and M. Seltzer, "Self-organization in peer-to-peer systems," in *Proc. 10th European SIGOPS Workshop*, September 2002.

[13] M. Rosvall and K. Sneppen, "Modeling self-organization of communication and topology in social networks," *Phys. Rev. E*, vol. 74, no. 1, p. 016108, jul 2006.

[14] T. Kohonen, *Self-Organizing Maps*, 3rd ed., ser. Springer Series in Information Sciences. Springer-Verlag, 2001, vol. 30.

[15] T. M. Liggett and S. W. W. Rolles, "An infinite stochastic model of social network formation," *An infinite stochastic model of social network formation*, vol. 113, no. 1, pp. 65–80, sep 2004.

[16] J. B. Schmitt and U. Roedig, "Sensor network calculus Ű a framework for worst case analysis," in *Proc. First IEEE International Conference on Distributed Computing in Sensor Systems,*, ser. Lecture Notes in Computer Science, vol. 3560, IEEE. Springer-Verlag, 2005, pp. 141–154.

[17] T. H. Labella, G. Fuchs, and F. Dressler, "A simulation model for self-organised management of sensor/actuator networks," in *Proc. GI/ITG KuVS Fachgespräch Selbstorganisierende, Adaptive, Kontextsensitive verteilte Systeme (SAKS)*, mar 2006.

[18] J. Starzyk and Z. Zhu, "Software simulation of a self-organizing learning array system," in *Proc. 6th IASTED Int. Conf. Artificial Intelligence & Soft Comp. (ASC 2002)*, H. Leung, Ed. IASTED/ACTA Press, jul 2002.

[19] D. Stamatelakis and W. D. Grover, "Opnet simulation of self-organizing restorable sonet mesh transport networks," in *Proc. OPNETWORKS '98 Conference (CD-ROM)*, apr 1998, p. paper 04.

[20] M. Sethson, P. Krus, and M. Karlsson, "Simulation of self organizing structures using neuro mechanical networks," in *Foundations for Successful Modelling & Simulation : Proc. 17th European Simulation Multiconferece (Esm2003)*, D. Al-Dabass, Ed., jun 2003, pp. 532–537.

[21] J. Cao, "Self-organizing agents for grid load balancing," in *Proc. 5th IEEE/ACM International Workshop on Grid Computing (GRID 2004)*. IEEE/ACM, nov 2004, pp. 388–395.

[22] K. S. Trivedi, *Probability and Statistics with Reliability, Queuing, and Computer Science Applications*, 2nd ed. John Wiley & Sons, nov 2001.

Self-Organizing, Evolving Networks for Performance Optimization

Peter Merz
Department of Computer Science
University of Kaiserslautern
D-67653 Kaiserslautern, GERMANY
Email: peter.merz@ieee.org

***Abstract*— Self-organization plays an important role in the design of protocols/algorithms for computer networks, such as peer-to-peer overlays, ad-hoc mobile networks, and sensor networks. However, it it is not trivial to design protocols or algorithms in a way that a desired behavior emerges.**

By taking the point of view of optimization, we demonstrate how algorithms can be designed to optimize predefined performance metrics and hence exhibit desired emergent behavior. We show in the case of peer-to-peer overlay topologies that considering an approximation to the performance objective can yield fast convergence to a stable state of the network.

I. Introduction

Self-organizing systems (SOS) are systems comprised of independent interacting entities often acting on simple rules. These systems have no external control but may exhibit emergent behavior. It is generally not obvious how the system as a whole behaves looking at the rules describing the behavior of the entities. On the other hand, it is not obvious how to define the rules of the entities in order to achieve a desired (emergent) behavior. In the case of self-organizing networks it is desirable to choose the rules (protocols/algorithm) such that a highly fault-tolerant and efficient network emerges in respect to predefined performance metrics.

II. Modeling of Self-Organizing Networks

A self-organizing network can be modeled as a set of nodes and a set of links. Hence, the state of the network is comprised of the individual states of the nodes and the states of the links. The nodes of the network are the active entities and hence their protocols (rules) define the overall behavior of the network. Regarding the system, an action according to the rules of a node may yield a transition from one state of the system to a new state.

In the following we consider the simplified model where a network is defined as a graph $G = (V, E, d)$ denoting a set of nodes V, a set of edges E, and a function d assigning a weight to each edge $e \in E$. An edge weight $d(i, j)$ of edge (i, j) denotes the communication cost over the edge. In static networks, the node set V remains constant over time, while in dynamic networks nodes may join and leave. As a consequence, the edge set may also change. The state of the network can be described as the graph $G_t = (V_t, E_t, d_t)$ at time t. A transition rule of the overall system is defined as a function T with $G_{t+1} := T(G_t)$. Since in a self-organizing network, the nodes are active elements, the transition rule is defined by the rules implemented in the nodes: each node may actively change the network by establishing and removing links. Hence, it removes edges from or adds edges to the edge set.

The dynamics of self-organizing networks can be observed by looking at a sequence of graphs $G_1, G_2, \ldots, G_{t-1}, G_t$ produced by the transition rule given a start configuration G_1. If the graph does no longer change from one time step to the other $G_{t+1} = T(G_t)\ \forall t >= t_c$, the systems is said to be converged, it has reached a stable state.

III. Performance Modeling of Self-Organizing Networks and Network Evolution

Performance modeling in the above model simply means assigning a performance value to each possible state of the system $p_t := P(G_t)$. Hence, we can observe the evolution of the system regarding its performance over time. Since, there may be more than one performance metric, $P(G_t)$ yields a vector of performance values: $(p_1, p_2, p_3, \ldots)_t := P(G_t)$.

The performance dynamics of self-organizing networks can be observed by looking at a sequence of performance values $P(G_1), P(G_2), \ldots, P(G_t)$ produced by the transition rule given a start configuration G_1. Since there may be too many possible starting configurations, considering all possible sequences is not practical. Instead, one may focus on some other important properties of these self-organizing networks.

The main goal of the development of self-organizing networks besides their ability to recover from faults is their ability to optimize their properties regarding predefined performance measures. From this point-of-view, we are interested in self-optimizing, evolving networks which maximize the desired performance metrics. More precisely, let $f(G_t)$ be a function to be optimized. If the transition rule allows to select only those transitions with $f(G_{t+1}) > f(G_t)$, a sequence of transitions terminates eventually by converging to a local or global optimum of f. Ideally, we are interested in transition rules that optimize our desired performance measures ($f = P$).

By treating the process of self-organization also as a process of optimizing the performance, techniques from the field of combinatorial optimization may be applied. The concept of local search appears therefore to be promising. In local search, a current solution to the optimization problem is improved by

searching for a better solution in the neighborhood of the current solution. If a better solution is found, it is accepted as new current solution and the search is repeated. If no better solution is found, the search terminates since a local optimum has been found. The neighborhood is defined as the set of solutions that can be reached from the current solution by making small changes. In case of a network problem, where the solution is a graph this means exchanging one or two edges against one or two new edges, for example. This way, the network is allowed to improve stepwise with? respect to some objective function. This basic optimization scheme can be realized as a distributed algorithm and is hence suitable for self-organizing networks. However, not any performance measure can be used as the objective in a distributed local search, due to several reasons: the individual nodes have no global view, nodes and links may fail, nodes may act selfishly, the communication overhead for an improvement step becomes to high, or the objective is a hard optimization problem (NP-hard in case of combinatorial optimization). Therefore, we propose to select a different objective for which an effective distributed local search exists and which approximates the desired performance metrics. In the following we will demonstrate this for peer-to-peer overlays.

IV. Example: Peer-to-Peer Overlays

Peer-to-peer overlays are self-organizing networks on top of the Internet. Nodes may join and leave at any time, hence these networks are highly dynamic. In general (disregarding firewall issues), any peer may establish a connection to any other peer. The challenge is to find a topology connecting all peers and enabling efficient routing (resource discovery). More formally considering the above model, from a fully connected graph, some edges have to be selected such that the resulting subgraph containing all nodes of the original is connected. Moreover, it is preferable that the underlying network is respected in a way that neighboring peers in the overlay are also close on the Internet level (short message delays).

The simplest topology with the least number of edges is a spanning tree. In the following, we consider therefore self-organizing spanning tree networks. Moreover, we consider only performance issues and not fault tolerance.

1) Performance Measures / Optimization Objectives: A performance measure for a tree may be the total edge cost, where the edge costs are defined by the round-trip time (delay). Hence a minimum spanning tree is desired for the overlay:

$$\min C_{MST}(T) = \sum_{(i,j)\in T} d(i,j), \tag{1}$$

where $d(i,j)$ denotes the message delay between peer i and peer j. A self-stabilizing distributed algorithm and hence transition rules for converging to an optimum minimum spanning tree (MST) is presented in [1].

The above objective is not well-suited for P2P overlays although it respects the underlying network as mentioned above. A better performance measure is the communication cost / routing cost in the tree. The following objective expresses this:

$$\min C_{MRT}(T) = \sum_{i\in V}\sum_{j\in V} d_T(i,j), \tag{2}$$

with $d_T(i,j)$ denoting the sum of the edge weights (delays) of the edges along the path in the tree T from node i to j. This problem is known as the Minimum Routing Cost Spanning Tree Problem (MRT) and is known to be NP-hard [2], [3].

An alternative to the MRT problem objective may be the shortest path tree (SPT) objective:

$$\min C_{SPT}(T,r) = \sum_{i\in V} d_T(i,r), \tag{3}$$

where r is the root node in the (directed) tree. Since the SPT may degenerate to a star, a degree constraint has to be added. Each node is allowed to have at most k children. The degree constrained minimum SPT serves as an approximation to the MRT objective, hence a self-organizing network optimizing the SPT objective also implicitly optimizes the MRT objective.

Optimum trees with the three objectives are shown in Figure 1. The figure demonstrates the high impact of the choice of the objective on the tree. Clearly, the MST is not a good approximation to the MRT.

2) Transition Rules / Local Search Moves: In the distributed local search algorithm, each node has transition rules which define the local transformations on the graph that should be considered. Each node selects an action randomly in an optimization step which optimizes his local objective (gain function). In combinatorial optimization, the chosen action is called a move. In case of spanning trees, we consider two types of moves shown in Fig. 2.

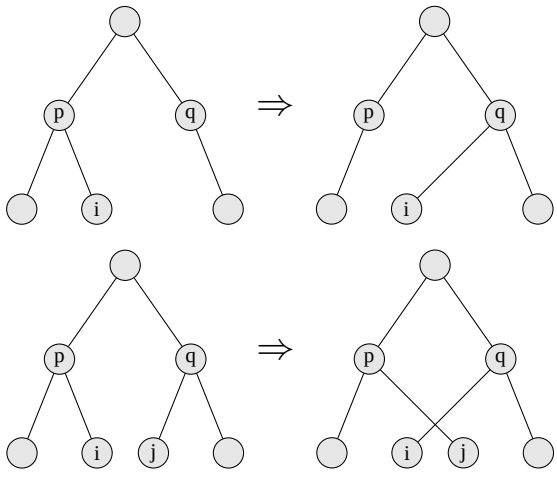

Fig. 2. Two types of tree moves

In the first move (a), a node (i) tries to connect to a new parent (q). If the node q accepts further children, and if the distance from node i to the root is reduced, the move is accepted. Let us assume the distance from node x to the root in terms of the sum of the weights of the path to the root is

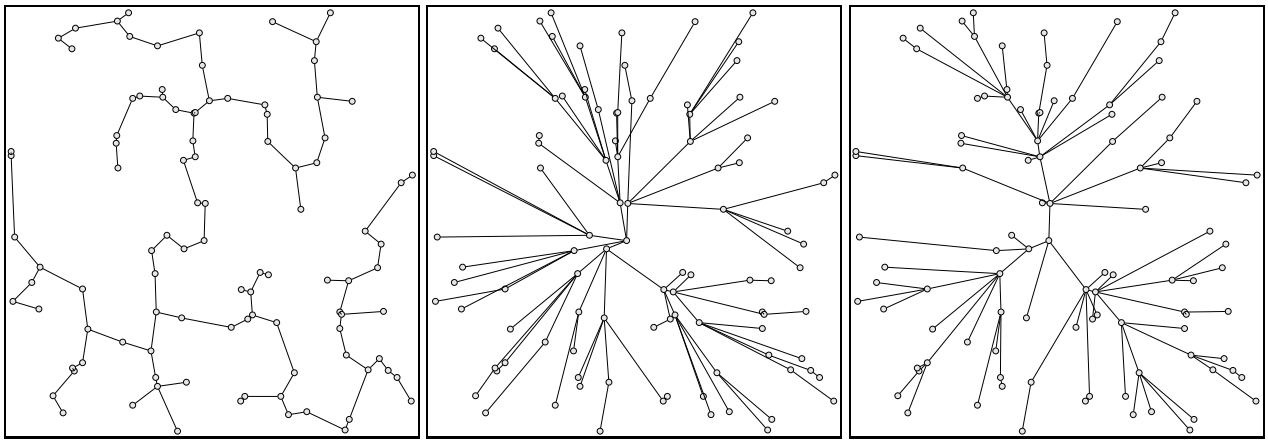

Fig. 1. A minimum spanning tree (right), a degree constrained shortest path tree (middle), and a minimum routing cost tree (right) in the Euclidean plane.

denoted by $d_r(x)$. Hence the gain in cost of choosing node q as the new parent for i is

$$g(i,p,q) = d(i,p) + d_r(p) - d(i,q) - d_r(q), \qquad (4)$$

denoting $d(x,y)$ the round-trip-time from peer x to y (the weight of edge (x,y)). If the gain is greater than zero, the move is accepted. The second move (b) is a little bit more complicated. If node q already has the maximum number of children (2 in the figure), node q may decide to accept the new child i and redirect its child j with maximum distance to itself (j with $\max d(j,q)$) if $d(i,q) < d(j,q)$. Hence, the distance to the root of one node (i) is reduced, possibly at the cost of the other (node j).

Since each node periodically probes for an improving move, the global objective is optimized over time. Hence the optimization is performed in rounds. In case of the moves described above, this objective is the $C_{SPT}(T)$, the degree constrained shortest path objective. This objective provides a good approximation of the minimum routing cost objective, our chosen performance metric. In Fig. 3, the convergence of the self-organizing, evolving network towards the minimum routing cost objective is shown. The results are derived from simulation of a overlay network with approx. 400 peers. The normalized cost $C(T)$ is the $C_{MRT}/C_{bestSPT}$ with $C_{bestSPT}$ denoting the globally best SPT in terms of the MRT objective. Instead of providing the simulation time, the number of optimization rounds is shown on the x-axis. The round trip times used in the simulations were taken from PlanetLab ping measurements [4]. The ping values were taken as message delay values for the overlay network. In the simulations, the network remained static (no joining and leaving of peers) and it was assumed that the delays remain constant. Moreover, an external mechanism is required for discovering peers in the overlay (this can be realized with an epidemic algorithm). The plot demonstrates the fast convergence towards a local optimum depending on the maximum degree. A degree of 5 appears to be a good choice for shortest path trees. Since a good approximating SPT has the root in the center of the network additional rules are required for moving the root towards the center. A more detailed evaluation of the algorithm and the problem can be found in [5], [6].

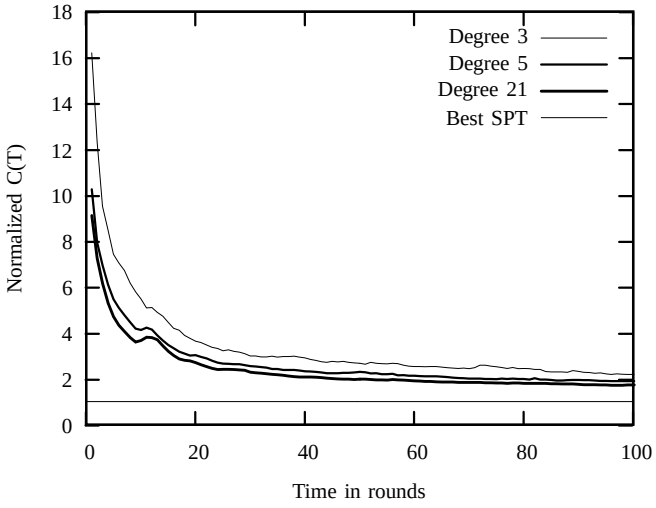

Fig. 3. Convergence with respect to minimum routing costs.

The study described above only focuses on the static behavior of the evolutionary SOS. In a dynamic scenario, the graph G_t changes every time a node joins or leaves and also if the message delay d_t changes due to traffic changes in the underlying Internet. Hence, the optimization problem itself changes. The algorithm described above automatically adapts to the new situation. However, if changes are too frequent, the algorithm may never converge. Since the optimization is performed in rounds, the message overhead of the optimization algorithm remains constant even in case of highly dynamic systems. This would not be the case if the algorithm was fully asynchronous. Clearly, there is a point of diminishing returns where the optimization cannot do any longer improvements due to the high frequency of changes. However, the analysis of the static case is highly important since it demonstrates the ability of the SOS to converge to (near)optimal configurations. Here, two aspects are especially important: (a) the speed of

convergence, and (b) the expected quality of the converged stable states in respect to the performance measures.

V. Example: Ad-hoc Sensor Networks

Ad-hoc sensor networks are in several aspects different from peer-to-peer overlay networks. Not any node may communicate directly with each other node. However, nearby nodes can be easily discovered in wireless networks. Regarding performance measures, also routing/communication efficiency is important but communication costs may be measured by the number of hops required for routing packets from source to destination nodes as well as by the energy consumed by the routing operation. In [7], the energy stretch factor and the distance stretch factor are proposed as performance measures. Similar objectives are also feasible. Besides these properties of wireless ad-hoc sensor networks, also the *throughput* in terms of bit-meters per second and the *robustness to mobility*, given by the number of nodes that need to change their topology information as a result of a movement of a node, are of importance [7].

From the optimization point of view, these performance measures may be optimized at the same time leading to a multi-objective optimization problem. For example, one might be interested in minimizing routing costs (in terms of number of hobs over all routing paths), minimizing energy consumption, and minimizing robustness to mobility simultaneously.

VI. Aspects of Self-Organized Evolution

The above local search based optimization can be considered as a distributed evolutionary algorithm. Formally, it resembles a (1+1)-EA but due to nature of self-organization there is no global selection. Considering static networks (no joining and leaving of peers) allows us to find an objective to optimize and therefore provides a way to prove convergence and hence self-stabilization of the distributed algorithm, which is an important aspect in theory of distributed algorithms. The speed of convergence can be analyzed with methods from the theory of evolutionary algorithms [8].

Note that the local search can be generalized to optimize more than one objective. In such a multi-objective optimization [9] scenario, the concept of Pareto dominance and optimality provides a suitable approach to dealing with several performance metrics.

A given distributed algorithm can be assessed by his ability to approach (approximate) the optimum regarding the performance metric. In cases where the optimum is not known, .i.e., NP-hard problems, lower/upper bounds on the optimum may be used instead. Here, approaches from combinatorial optimization apply.

While generally the system tends to self-stabilize in a local optimum, this might not be the case in a highly dynamic scenario where nodes join and leave with high frequency. If this happens, moves no longer improve the objective and the system becomes chaotic. The phase transition where the system/network turns its ordered behavior into a chaotic behavior is an important characteristic of the system.

Since each node can be regarded as a selfishly acting agent, game theory plays an important role here. The local optimum to which the network converges is a Nash equilibrium from the game theoretic point of view. If all nodes act selfish, the objective functions to optimize by such a system are restricted. Game theory may help to decide which function can be optimized and which not, see [10], [11] for a discussion.

The ”price of anarchy” [12] is the relative cost of agents acting selfish instead of altruistic regarding some predefined objective/performance measure. This property has been considered in network creation games [13], [14] and is also an important property in self-organizing, evolving networks since it indicates the relative effectiveness in case of selfish, greedy behavior.

VII. Conclusions

Instead of utilizing self-organization only to recover from failure, we propose to use self-organization to increase efficiency with respect to desired performance metrics. This enables networks to evolve, e.g. to improve over time and to converge to a stable state that can be respected as a (local) optimum of the performance objective(s). We discussed the use of objectives as approximations to the real performance functions due to the fact that not all functions in a distributed scenario with possibly selfish acting nodes can be optimized. We demonstrated the feasibility of the approach for peer-to-peer overlays, where a topology is desired in which communication/routing cost is minimal. Furthermore, we discussed some issues relevant for the analysis of self-organizing, evolving networks.

References

[1] L. Higham and Z. Liang, “Self-stabilizing minimum spanning tree construction on message-passing networks,” in *Distributed Computing, 15th International Conference, DISC 2001, Lisbon, Portugal, October 3-5, 2001, Proceedings*. 2001, vol. 2180 of *Lecture Notes in Computer Science*, pp. 194–208, Springer.

[2] D. S. Johnson, J. K. Lenstra, and A. H. G. Rinnooy Kan, “The Complexity of the Network Design Problem,” *Networks*, vol. 8, pp. 279–285, 1978.

[3] B. Y. Wu, G. Lancia, V. Bafna, K.-M. Chao, R. Ravi, and C. Y. Tang, “A Polynomial-Time Approximation Scheme for Minimum Routing Cost Spanning Trees,” *SIAM Journal on Computing*, vol. 29, no. 3, pp. 761–778, 1999.

[4] S. Banerjee, T. Griffin, and M. Pias, “The Interdomain Connectivity of PlanetLab Nodes,” in *Proceedings of the 5th International Workshop on Passive and Active Network Measurement, (PAM 2004)*, C. Barakat and I. Pratt, Eds. 2004, vol. 3015 of *Lecture Notes in Computer Science*, Springer.

[5] P. Merz and S. Wolf, “Evolutionary Local Search for Designing Peer-to-Peer Overlay Topologies Based on Minimum Routing Cost Spanning Trees,” in *Proceedings of the 9th International Conference On Problem Solving from Nature (PPSN IX)*. 2006, vol. 4193 of *Lecture Notes in Computer Science*, pp. 272–281, Springer.

[6] P. Merz and S. Wolf, “TreeOpt: Self-Organizing, Evolving P2P Overlay Topologies Based On Spanning Trees,” Working paper, University of Kaiserslautern, Kaiserslautern, Germany, 2006.

[7] R. Rajaraman, “Topology control and routing in ad hoc networks: a survey,” *SIGACT News*, pp. 60–73, 2002.

[8] I. Wegener, “On the Design and Analysis of Evolutionary Algorithms,” in *Proceedings of the Workshop on Algorithm Engineering as a New Paradigm*, 2000, pp. 36–47.

[9] C. A. Coello, "Special Issue on Evolutionary Multiobjective Optimization," *IEEE Transactions on Evolutionary Computation*, vol. 7, pp. 97–99, 2003.
[10] D. Wolpert and K. Tumer, "An Introduction to Collective Intelligence," *CoRR*, vol. cs.LG/9908014, 1999.
[11] D. H. Wolpert and K. Tumer, "Optimal Payoff Functions for Members of Collectives," *Advances in Complex Systems*, vol. 4, no. 2–3, pp. 265–279, 2001.
[12] E. Koutsoupias and C. Papadimitriou, "Worst-Case Equilibria," in *STACS: Annual Symposium on Theoretical Aspects of Computer Science*, 1999.
[13] A. Fabrikant, A. Luthra, E. Maneva, C. Papadimitriou, and S. Shenker, "On a Network Creation Game," in *22th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC)*, 2003.
[14] T. Moscibroda, S. Schmid, and R. Wattenhofer, "On the Topologies Formed by Selfish Peers," in *25th Annual Symposium on Principles of Distributed Computing (PODC)*, Denver, Colorado, USA, 2006.

Modell zur Analyse des Routenfortschritts in Ad Hoc Netzen

Tronje Krop, Tobias Lange, Matthias Hollick und Ralf Steinmetz
Technische Universität Darmstadt, Merkstrasse 25, 64289 Darmstadt
{tronje.krop}@kom.tu-darmstadt.de

***Kurzfassung*—Selbstorganisation in mobilen Ad hoc Netzen ist ein hoch aktuelles Forschungsthema. Die Fähigkeit solcher Netze spontane Verbindungen ohne Infrastruktur zu ermöglichen ist dabei von besonderem Interesse. Die Leistungsfähigkeit dieser Verbindungen hängt dabei stark von den verwendeten Technologien ab. Die Protokolle zur Routenfindung und zur Koordination des Zugriffs auf das Medium sind dabei entscheidend für die Stabilität und Zuverlässigkeit der Verbindungen. Hierbei kommen häufig Mechanismen der Selbstorganisation zum Einsatz, welche Aussagen über globalen Eigenschaften von Protokollen in Ad hoc Netzen schwierig machen. Im Folgenden entwickeln wir ein simulatives Model zur Analyse der Selbstorganisation der Routenfindung in Ad hoc Netzen um deren Leistungsfähigkeit zu evaluieren.**

***Schlagwörter*—Ad hoc Netze, Routenfindung, Leistungsbewertung.**

A. EINFÜHRUNG

Ad hoc Netze sind in den letzen Jahren vermehrt in das Interesse der Forschung getreten. Obwohl die Idee Endgeräte drahtlos und ohne Infrastruktur miteinander kommunizieren zu lassen nicht neu ist, wurden erst in den letzten Jahren Endgeräte entwickelt, mit denen dieses Paradigma der Selbstorganisation voll nutzbar ist. Insbesondere die Möglichkeit eine Route über mehrere Zwischenknoten aufzubauen, ermöglicht es, Endgeräten mit anderen außerhalb ihrer Sendereichweite in Verbindung zu treten. Die Leistungsfähigkeit solcher Routen hängt im Wesentlichen von den verwendeten Technologien und vor allem von dem Verfahren zur Findung und zum Aufbau der Routen ab. Hierbei kommen häufig Mechanismen der Selbstorganisation zum Einsatz, welche den Prozess der Routenfindung, des Routenaufbaus und der Medienzugriffskontrolle steuern.

Bei Protokollen zum Finden und Aufbauen von Routen unterscheidet man drei Protokolltypen, welche auch in Kombination auftreten können:

- *Proaktive* Protokolle führen den Prozess der Routenfindung und des Routenaufbaus mittels Link-Zustands-Nachrichten durch bevor eine Routenanfrage vorliegt, z.B. OLSR [1] oder TBRPF [2].
- *Reaktive* Protokolle führen den Prozess der Routenfindung und des Routenaufbaus erst auf Bedarf mittels eines Anfrage/Antwort-Verfahrens durch, z.B. AODV [3], DSR [4] oder DYMO [5].
- *Passive* Protokolle verwenden eine Route auf Grund von implizit vorliegenden Informationen, wie dem Ort des Zielknoten, z.B. LAR [6].

In allen Fällen unterscheidet man zwischen *flachen* und *hierarchischen* Protokollen zur Routenfindung. Wir konzentrieren uns im Weiteren auf flache, reaktive Protokolle zur Routenfindung, da hier aussagekräftige Modelle fehlen und sie ein prototypisches Suchverhalten aufweisen, das sich ähnlich in passiven Protokollen wiederfindet. Hierbei wird das Netz mit Anfragen geflutet, um auf effiziente Weise eine Kombination aus schnellster und kürzester Route aufzubauen. Dabei ist die zeitliche Abfolge der Nachrichten entscheidend, da häufig nur die erste Nachricht Berücksichtigung findet.

Beim Fluten der Anfragen sind Protokolle zur Medienzugriffskontrolle entscheidend, z.B. IEEE 802.11/15/16, da diese die Ausbreitung der Routenanfragen festlegen. Das in IEEE 802.11 [7] verwendete CSMA Broadcast unterstützt z.B. den Prozess des Flutens in Ad hoc Netzen, während der fehlende Broadcast in IEEE 802.16 eher hinderlich ist und einen störenden Einfluss auf die zeitliche Abfolge des Flutens hat. Auch IEEE 802.15 hat nur eine beschränkte Broadcast-Fähigkeit, die von vielen reaktiven Protokollen vorausgesetzt wird. Umgekehrt erzwingt die Verwendung des Broadcast gewöhnlich, dass Vorgänger und Nachfolger eines Knotens in der Route nicht direkt miteinander Kommunizieren können.

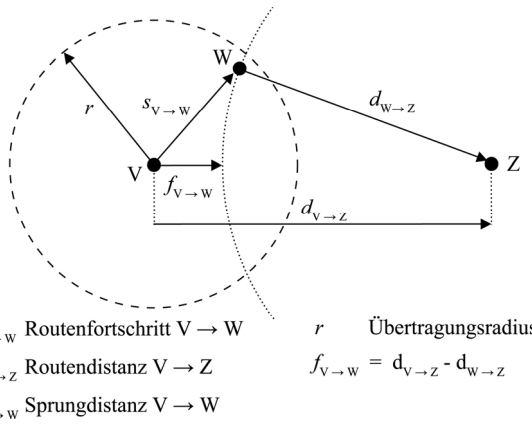

Abbildung 1: Definition von Routenfortschritt, Routendistanz, und Sprungdistanz.

Um die Eigenschaften von Verbindungen zu beschreiben verwenden wir die Konzepte des *Routenfortschritts f*, der *Routendistanz d* und der *Sprungdistanz s* (siehe Abbildung 1). Diese sind verwandt zur Sprungzahl einer Verbindung, ermöglichen jedoch eine bessere Charakterisierung um später Qualität und Lebenszeit einer Route zu bestimmen.

In Abschnitt B geben wir eine kurze Übersicht über verwandte Arbeiten. In Abschnitt C erklären wir unseren Ansatz zur Analyse der Routenfindung und beschreiben ein Modell zu Simulation. In Abschnitt D präsentieren wir schließlich die Ergebnisse der Evaluation unserer Modellierung, um den Beitrag in Abschnitt E mit unseren Schlussfolgerungen zu schließen.

B. Verwandte Arbeiten

Unter den vielen Arbeiten, die sich mit den Ad hoc Netzen beschäftigen, gibt es nur wenige, die versuchen die Eigenschaften der Selbstorganisation zu analysieren. Die meisten Arbeiten evaluieren die Leistungsfähigkeit des Datentransports ohne die zugrunde liegende Struktur des Netzes bezüglich Routendistanz, Sprungdistanz oder Routenfortschritt zu betrachten. Einen ersten Ansatz für die Wahrscheinlichkeitsverteilung von Routendistanzen in geometrischen Flächen enthält [8], welcher in [9] durch eine approximative Lösung weiterentwickelt wird.

Andere Arbeiten adressieren die Sprungdistanz indirekt durch die Untersuchung der erwarteten Lebenszeit einer Route. So werden in [10] verschiedene Metriken evaluiert, welche die Linkstabilität bei Routenfindung berücksichtigen und zu verbessern suchen. Dagegen versucht [11] die Lebenszeiten von Links in Ad hoc Netzen mittels Simulation zu ermitteln. [12] entwickelt für gleichförmige Bewegungen ein geometrisches Modell mit unabhängigen Linklebenszeiten und nutzt dieses zur Optimierung von Update-Intervallen. In [13] wird ein ähnliches Modell für den 2-Sprung-Spezialfall entwickelt, welches Abhängigkeiten zwischen den Links berücksichtigt. Dieses Modell wird in [14] rekursiv weiterentwickelt, wobei indirekte Abhängigkeiten vernachlässigt werden.

Alle Arbeiten gehen nur bedingt auf den Einfluss der Routenfindung ein und machen nicht klar, in welcher Form diese die Ergebnisse von Routenfortschritt und Sprungzahl abhängen. Einzig die letzten beiden Arbeiten gehen darauf ein, dass die Wahl eines Vorgängers die Wahl des Nachfolgers und damit die Sprungdistanz und den Routenfortschritt bedingt, ohne dieses jedoch konsequent zu untersuchen.

C. Modellansatz

Das Ziel unserer Modellbildung ist eine Analyse der Qualität und Lebenszeit von Routen in Ad hoc Netzen mittels der Verteilungsfunktionen von Routenfortschritt und Sprungdistanz durchzuführen. Dabei beschreibt die Sprungdistanz die Entfernung von zwei aufeinander folgenden Knoten in einer Verbindung, während der Routenfortschritt die Differenz der beiden Knoten in Relation zum Ziel bezeichnet (siehe Abbildung 1).

Das Modell soll den Prozess der Findung und des Aufbaus von Routen realistisch abbilden und dabei die Eigenschaften existierender Technologien berücksichtigen. Die modellierte Realität wird dabei durch das Drei-Gespann IEEE 802.11 für den Medienzugriff, IP für die Paketvermittlung und AODV für Routenfindung und -aufbau vorgegeben. Für die Modellierung treffen wir folgende Annahmen:

- *Übertragung:* Die drahtlose Übertragung wird durch eine zuverlässige und kollisionsfreie aber zufällige Zugriffskontrolle mit einheitlicher Reichweite modelliert. Die Pakete werden unendlich schnell und ohne Verzögerung übermittelt. Diese Abstraktion ist möglich, da die Routenfindung in IEEE 802.11 lediglich auf lokalen Broadcasts mit Random Backoff basiert.
- *Paketvermittlung:* Für die Paketvermittlung nehmen wir vereinfachend an, dass alle Knoten im Netz dem gleichen Subnetz angehören. Die Paketlänge spielt auf Grund der unendlich schnellen Übertragung keine Rolle. Ebenso erfolgt die Verarbeitung der Pakete unendlich schnell.
- *Routenfindung:* Die Routenfindung erfolgt durch einfaches Fluten des Netzes, wobei nur die als erst eintreffende Request an jedem Knoten berücksichtigt werden um eine Route zu etablieren. Die in AODV üblichen Verfahren zur Vor- und Nachoptimierung der Route werden dadurch vernachlässigt (siehe Abbildung 2).

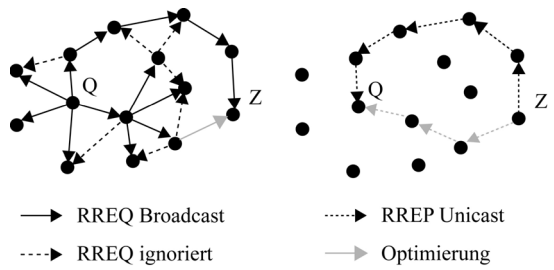

Abbildung 2: Schematische Darstellung der Routenfindung in AODV mit Optimierungen.

- *Topologie:* Da Findung und Aufbau von Routen über einen sehr kurzen Zeitraum ablaufen, sind Änderungen der Topologie zwar möglich aber unwahrscheinlich, da diese zwischen dem Empfang der Request- und dem Weiterleiten der Reply-Nachricht erfolgen müssten. Aus diesem Grund können wir dynamische Bedingungen durch statische Topologien modellieren.
- *Knotendichte:* Die Verteilung von Knoten hängt in der Realität von Szenario, Situation und Mobilität ab. Das Modell abstrahiert diesen Aspekt durch die

Annahme einer zufälligen Gleichverteilung mit der Knotendichte ρ.

Als Ergebnis erhalten wir ein einfaches Modell das statisch Knoten mit konstanter Reichweite *r* zuverlässig, bidirektional verbindet. Zur Kontrolle von Randeffekten verwenden wir eine elliptische Fläche in deren beider Foki Quell- und Zielknoten aufgehängt sind (siehe Abbildung 3). Die Routenfindung garantiert, dass jeweils der schnellste Pfad gefunden wird.

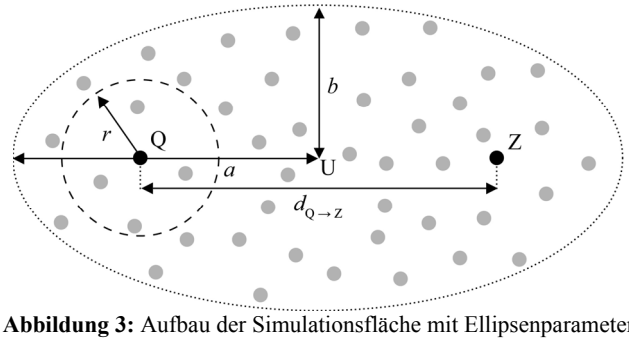

Abbildung 3: Aufbau der Simulationsfläche mit Ellipsenparameter *a*,*b*, Kommunikationsradius *r*, Routendistanz *d*, für den Ursprung *U*, und die Foki *Q (Quelle)*, *Z (Ziel)*.

D. ERGEBNISSE

Auf Basis dieses einfachen Modells haben wir eine systematische Untersuchung zur Findung und zum Aufbau von Routen in Ad hoc Netzen durchgeführt. In den Experimenten wurde die Übertragungsreichweite auf *r=1* normiert, wobei die Knotendichte ρ und die Distanz *d* zwischen Quell- und Zielknoten variiert wurde (siehe Tabelle 1).

Jedes Experiment wurde mit genau 50.000 Replikationen durchgeführt, was erst durch die einfache Struktur des Modells möglich wird. Bei der Analyse haben wir uns auf die Verteilung von Routenhäufigkeit (Anteil der erfolgreichen Routenanfragen in einer Topologie), Routenfortschritt, Sprungzahl und Sprungdistanz konzentriert (siehe Abbildung 1). Die Verteilung von Routenfortschritt und Sprungdistanz kann man sehr gut beobachten indem man für jeden Sprung die z-Achse auf den Zielknoten normiert, d.h. es wird der Nachfolger in Relation der Achse vom aktuellen Konten zum Ziel bestimmt (siehe Abbildung 4).

Ermittelt man genauer, wo die an einer erfolgreichen Route beteiligten Knoten liegen, so fällt auf, dass sich die Routenfindung für den ersten und den letzten Sprung deutlich von den restlichen unterscheiden (siehe Abbildung 5), was sich auch in der Sprungdistanz und im Routenfortschritt ausdrückt. Während dieses für den Sprung zum Ziel offensichtlich ist, erstaunt jedoch die Symmetrie im Verhalten für den Sprung von der Quelle.

Interessant ist auch, dass sich der durchschnittliche Routenfortschritt in dem Modell abhängig von Dichte und Distanz sehr unterschiedlich entwickelt (siehe Abbildung 6). Bei kleinen Distanzen ist unabhängig von der Dichte die Wahrscheinlichkeit bestimmend, dass ein Knoten für eine optimale Route im Empfangsradius von Quelle und Ziel liegt. Für hohe Dichten setzt sich dieses Verhalten für große Distanzen fort was zu Schwingungen führt. Bei geringen Dichten koppelt sich der Fortschritt von diesem Einfluss ab. Außerdem ist bei zunehmender Entfernung ein leichter Abfall des Routenfortschritts zu erkennen.

Tabelle 1: Experimentelles Design

	Start	Ende	Schrittweite
Knotendichte ρ	0.25	3.00	0.25
	3.50	12.00	0.50
Distanz *d*	1.10	3.00	0.10
	3.50	12.00	0.50

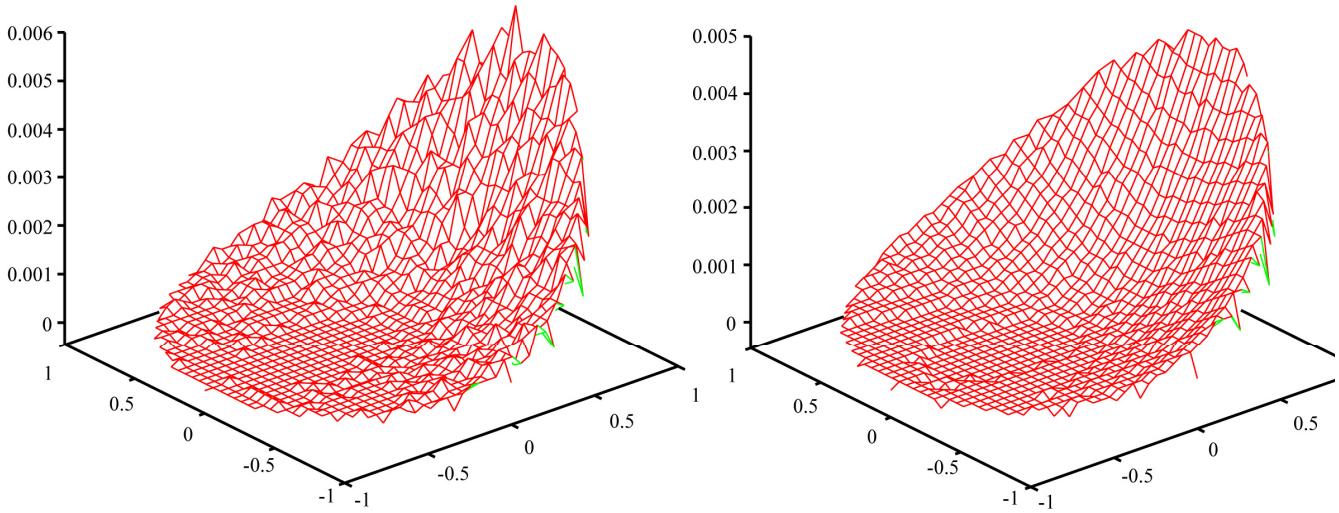

Abbildung 4: Verteilung von Routenfortschritt und Sprungdistanz für Knotendichten $\rho=0.75$ (links) und $\rho=1.25$ (rechts).

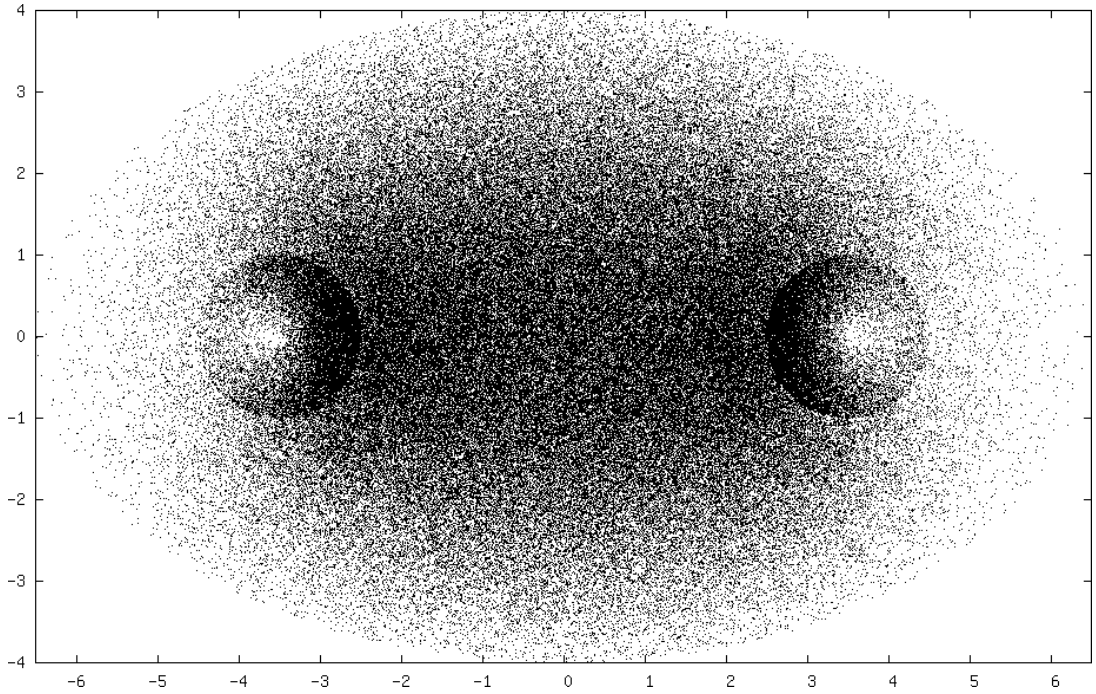

Abbildung 5: Position aller in Routen beteiligten Knoten für *d=7.0*, ρ=*1.5* von zufälligen Topologien aus *50.000* Experimenten.

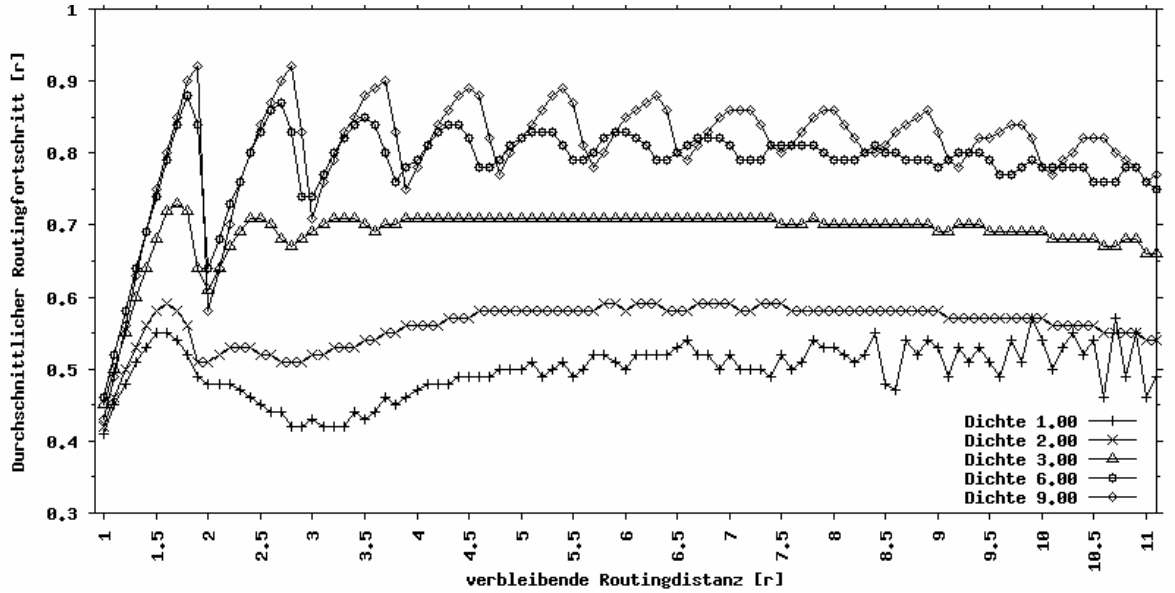

Abbildung 6: Durchschnittlicher Routenfortschritt über Routendistanz ohne Fortschritt des ersten und letzten Sprungs.

E. SCHLUSSFOLGERUNG UND AUSBLICK

Die Ergebnisse zeigen, dass bereits ein einfaches Modell der Selbstorganisation in Ad hoc Netzen ausreicht um wichtige Einsichten in den Prozess der Routenfindung zu erhalten. Dabei spielen die Routendistanz, Sprungdistanz und der Routenfortschritt eine bedeutende Rolle. Denn erst das Verständnis der Ausbreitung von Routenanfragen und der Eigenschaften von Verbindungen bezüglich Qualität und Lebenszeit erlaubt es die Leistungsfähigkeit der Selbstorganisation in Ad hoc Netzen zu analysieren und zu optimieren.

In weiteren Arbeiten wollen wir das Modell in der Übertragungsschicht und der Routenfindung verfeinern. Dabei wollen wir Bandbreite und Kollisionen in das Modell der Übertragung aufnehmen, sowie Verfahren der Optimierung von Routen berücksichtigen.

F. REFERENZEN

[1] T. Clausen and P. Jacquet. Optimized Link State Routing Protocol (OLSR). IETF Request for Comments 2501, October 2003.

[2] R. Ogier, F. Templin, and M. Lewis. Topology Dissemination Based on Reverse-Path Forwarding (TBRPF). IETF Request for Comments 2501, February 2004.

[3] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc On-Demand Distance Vector (AODV) Routing. IETF Request for Comments 3561, July 2003.

[4] D. B. Johnson, D. A. Maltz, and Y.-C. Hu. The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR). IETF Internet-Draft draft-ietf-manet-dsr-10.txt, July 2004.

[5] Ian D. Chakeres, Elizabeth M. Royer, and Charles E. Perkins. Dynamic MANET On-demand Routing Protocol. IETF Internet Draft, draft-ietf-manet-dymo-03.txt, October 2005 (Work in Progress)

[6] Y.-B. Ko and N. H. Vaidya. Location-Aided Routing (LAR) in Mobile Ad Hoc Networks. In Proceedings of 4. ACM/IEEE International Conference Mobile Computing and Networking (MobiCom 1998), Dallas, Texas, USA, pages 66-75, October 1998.

[7] IEEE Computer Society. IEEE Standard 802.11-1999 for Local and Metropolitan Area Networks. *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, Reference number ISO/IEC 8802-11:1999(E), August 1999.

[8] L. E. Miller. Distribution of Link Distances in a Wireless Network. *Journal of Research of the National Institute of Standards and Technology*, 106(2):401-412, March 2001.

[9] J. P. Mullen. Robust Approximations to the Distribution of Link Distances in a Wireless Network Occupying a Rectangular Region. *ACM SIGMOBILE Mobile Computing and Communications Review*, 7(2):80-91, April 2003.

[10] G. Lim, K. Shin, S. Lee, H. Yoon, and J. S. Ma. Link Stability and Route Lifetime in Ad-hoc Wireless Networks. In Proceedings of International Conference on Parallel Processing Workshops, pages 116-123, August 2002.

[11] N. Sadagopan, F. Bai, B. Krishnamachari, and A. Helmy. PATHS: Analysis of PATH Duration Statistics and their Impact on Reactive MANET Routing Protocols. In Proceedings of 4th ACM International Symposium on Mobile Ad Hoc Networking & Computing (MobiHoc 2003), Annapolis, Maryland, USA, pages 245-256, June 2003.

[12] P. Samar and S. B. Wicker. On the Behavior of Communication Links of a Node in a Multi-Hop Mobile Environment. In Proceedings of 5th ACM International Symposium on Mobile Ad hoc Networking & Computing (MobiHoc 2004), Roppongi Hills, Tokyo, Japan, pages 145-156, May 2004.

[13] I. Gruber and H. Li. Link Expiration Times in Mobile Ad Hoc Networks. In Proceedings of 27th IEEE Conference on Local Computer Networks (LCN 2002), Bonn/Königswinter, Germany, pages 743-750, November 2002.

[14] I. Gruber and H. Li. Path Expiration Times in Mobile Ad Hoc Networks. In 5th European Personal Mobile Communications Conference, (EPMCC 2003), Glasgow, Scotland, pages 27-31, April 2003.

Performance Simulation of Distributed Embedded Self-Organizing Systems Modeled with SDL

P. Becker, R. Gotzhein, T. Kuhn
University of Kaiserslautern
Postfach 3049, D-67653 Kaiserslautern, Germany

***Abstract*--SDL is one of the formal design languages used for model-driven development of distributed systems. Models specified with SDL can be used directly for performance simulations. To obtain accurate performance assessments, all resources influencing system performance must be simulated together. In this work, we present PartsSim, a tool that is capable of simulating the performance of SDL models, taking limitations of hardware platform and network into account. Several performance simulations of a simple self-organizing MicaZ network scenario provide evidence for the additional accuracy achieved with PartsSim.**

***Index terms*-- performance modeling, simulation, SDL**

A. INTRODUCTION

ITU-T's formal description technique SDL [1] is one of the design languages used for model-driven development of distributed systems. SDL is supported by commercial tool suites (e.g., [2]), and applied in the telecommunications industry. In previous work, we have shown that SDL models can be used for performance simulation. For this purpose, we have developed *ns*+SDL [3], an extension of the network simulator *ns-2* [4] that is capable of loading SDL models. For related work on the performance simulation based on SDL models, refer to [3].

To obtain accurate performance assessments in general and of SDL models in particular, all resources influencing system performance must be simulated together. For instance, in a distributed embedded self-organizing system, both the hardware platform and the network are resource bottlenecks. For an accurate performance assessment, the simulator must support realistic timing models of processor, memory, and transceiver chip. Furthermore, it must support realistic propagation models, considering the possibility of message collisions and transmission ranges under varying channel conditions. Further factors are mobility and energy consumption.

In this work, we present our tool PartsSim, which is capable of simulating the performance of SDL models, incorporating several resource bottlenecks. In Section B, we briefly survey the architecture and functionalities of PartsSim. In Section C, we present simulation results that demonstrate the additional accuracy of simulating several resources together. We present conclusions and an outlook in Section D.

B. THE PERFORMANCE SIMULATOR PARTSSIM

PartsSim is a modular performance simulator for SDL models that incorporates accurate simulation models of the MicaZ [5] hardware platform including hardware interrupts, processor timing, and transceiver chip, and of the wireless network. It has been created out of several existing simulators for the simulation of single resources. An important advantage of PartsSim is that the same SDL model is used as code base for the simulation and for the target system, and that the same compiler is used to generate this code. Thus, it can be expected that the results of the performance simulation faithfully reflect the behavior of the system in operation.

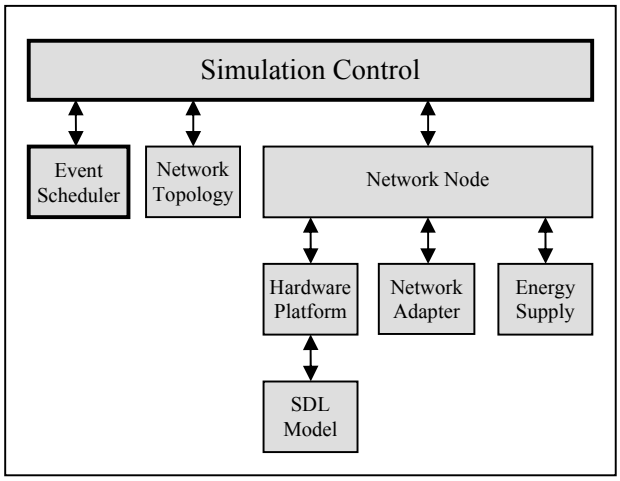

Fig. 1 Structure of PartsSim

The core components of PartsSim are simulation control and event scheduler (see Fig. 1), which have been extracted from the *ns-2* [4] and modified. Boxes represent simulator components, arrows denote the control flow. The *simulation control* instantiates all other simulation components and triggers their simulation steps in an orderly fashion. The *event scheduler* ensures that all relevant simulation events of different simulator

components are recorded in a centralized event list and processed in the correct order. Both components are generic in the sense that they support the integration of specialized simulators.

In a distributed scenario, several network nodes and a network topology are instantiated by the simulation control (see Fig. 1). A *network node* consists of a set of *platform resources* (e.g. hardware platform, simulated devices, energy supply) and an *SDL model*. Currently, PartsSim supports one type of nodes, namely MicaZ motes produced by Crossbow Technologies [5]. This type of node is used, for instance, in self-organizing ad-hoc network scenarios for ubiquitous computing.

- Main processor of MicaZ motes is the ATMEL ATMega 128L, with 4 KB RAM and 128 KB ROM. This part of the platform is simulated by a PartsSim component that is based on Avrora [6].
- For wireless communication, MicaZ uses the transceiver chip Chipcon CC2420 with the MAC protocol IEEE 802.15.4 (ZigBee), supporting a transmission rate of up to 250 Kbps. This part of the platform is simulated by a modified *ns-2* component.
- The behavior of the system under simulation is defined by an SDL model, which is translated to binaries for MicaZ and loaded into the simulator.
- Nodes are arranged in a network, which in particular determines topology, transmission ranges, mobility of nodes, and propagation model.

By extracting components of existing simulators, where possible, and systematically integrating them into PartsSim, we have saved an enormous amount of development effort. For the integration, several problems had to be solved:

- A simulator interconnection framework defining an overall structure and generic interfaces of PartsSim components had to be devised.
- To convert specialized simulators into a PartsSim component, the event scheduler had to be removed, and an interface to the centralized PartsSim event scheduler had to be added.
- From the SDL model, binaries running on MicaZ motes had to be created. This required the development of a specialized runtime system that both fitted the available SDL-to-C compiler and the target platform.

As mentioned before, PartsSim has a modular structure and can be modified and extended by exchanging simulator components and integrating further specialized simulators in the described fashion, respectively. For instance, by replacing the Avrora simulator, other types of nodes may be supported.

C. SIMULATION WITH PARTSSIM

In order to get a feeling for the additional accuracy of simulating several resources together, we have run a simple scenario with different simulator configurations. The simulated system consists of three MicaZ motes, which are in transmission range of each other. The master node is responsible for synchronizing the client nodes, using a beacon frame sent every 8ms. Transmission slots of the two clients start 2 ms and 4 ms after the beacon was received, respectively. The two clients resume listening for the beacon 4 ms and 3 ms after their own transmission, respectively.

For the performance simulation of this scenario, we have used three simulator configurations: TAU [2] simulating the SDL model only, *ns*+SDL [3] to include network characteristics, and PartsSim that in addition simulates the hardware platform.

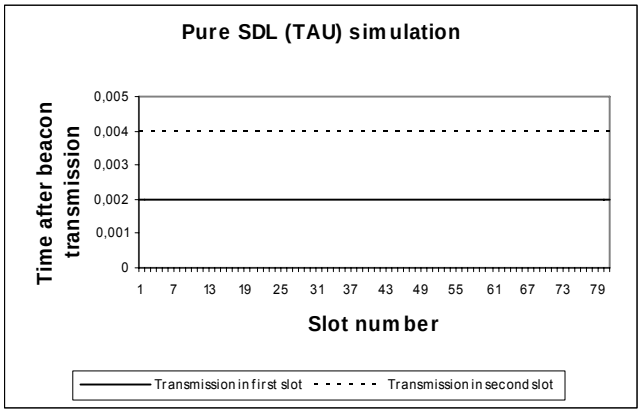

Fig. 2: Simulation with TAU Generation 1 [2]

For the simulation with the TAU SDL simulator, we have specified a global SDL model by instantiating nodes and connecting them by SDL channels. As the TAU SDL simulator does not consider resource bottlenecks, the SDL timers are the only sources of delays. Fig. 2 shows the results of the performance simulation. The delay between transmission start of the master node's beacon frame and the transmission start of corresponding data frames is 2 ms and 4 ms, respectively, depending on the slot used by the client node. This means that in the simulation, packets are transmitted on schedule, with no further propagation or reception delays. With no resources being simulated, this is the expected timing behavior.

For the simulation with the *ns*+SDL simulator, we have instantiated a distributed simulation scenario, consisting of three nodes, each simulating an SDL model. In this configuration, the network characteristics including transmission ranges, network delay, and frame collisions are incorporated. Due to the synchronized medium access in the given scenario, no collisions can

occur. Also, all nodes are within range of each other. Thus, the additional effect that shows up in the simulation is the (constant) transmission delay, which is about 0.45 ms per frame. Thus, transmission of data frames is delayed by the transmission time of the beacon frame (see Fig. 3).

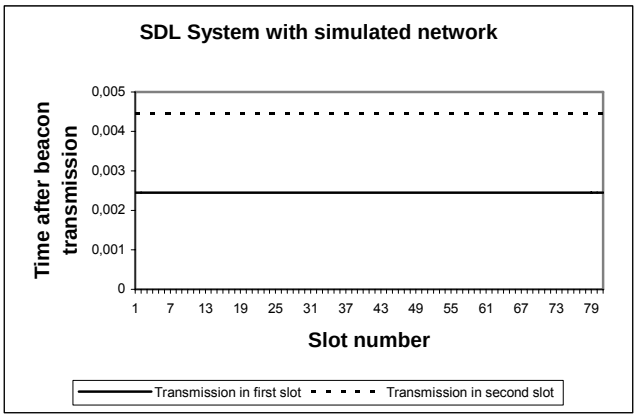

Fig. 3: Simulation with *ns*+SDL [3]

The third simulation is run using PartsSim, with the same scenario as before. In addition to *ns*+SDL, PartsSim considers the MicaZ hardware platform. The binary code generated from the SDL model is now executed under timing control of the simulator, which produces additional delays for reacting to hardware interrupts and for processing sending and reception of frames. The timing behavior of this scenario is shown in Fig. 4. In addition to the network delay, there is a platform delay of about 1.5 ms per frame.

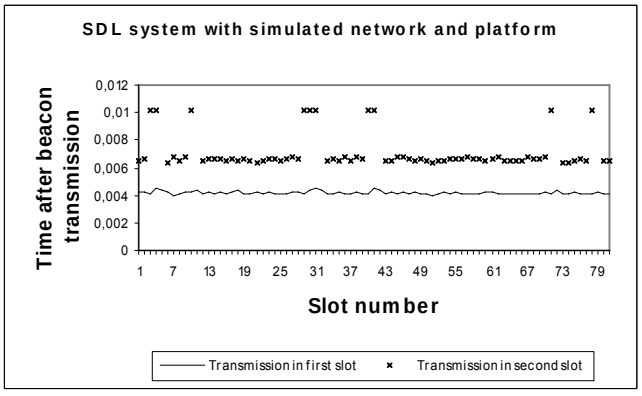

Fig. 4: Simulation with PartsSim

Fig. 4 reveals timing problems that can only be detected if all resource bottlenecks are simulated together. First, transmission is delayed by about 2 ms w. r. t. to the assigned data slot. Second, due to internal jitter, the second client does not always receive the beacon frame while listening. In these cases, it interprets the data frame of the first client as beacon frame, and synchronizes to this event. This causes an additional transmission delay of 4 ms w. r. t. the corresponding beacon transmission. When missing the next data frame of the first client, the second client is resynchronized to the beacon period, however, by losing one data slot.

With more simulated resources, the simulation time in the above simple scenario increases substantially. The simulation with TAU runs for about 2 sec, the *ns*+SDL simulation takes about 10 sec, and the simulation time with PartsSim is 2:07 min.

As already pointed out, the purpose of the above simulations is to show the additional accuracy that is gained by simulating several resources together. From the results of the simulation with PartsSim, we can conclude that there are timing problems caused by resource bottlenecks. From the scenario, it is obvious how these problems can be avoided. However, without an accurate performance simulation, it would be difficult to predict this timing behavior even for this simple scenario.

D. Conclusions and Outlook

In this work, we have presented PartsSim, a performance simulator for SDL models that takes the limitations of the MicaZ hardware platform and of the underlying network into account. We have provided first evidence for the additional accuracy achieved with PartsSim.

The accuracy achievable in performance simulations depends on several factors. One limiting factor is the precision of the resource models. To this end, the results obtained with PartsSim rely on the accuracy of the platform model provided by Avrora, and of the network characteristics as modeled by *ns2*. These parts of the simulator must be calibrated by exactly following the available data sheets, and by real experiments. Another crucial factor is the behavior of system under simulation. Here, it is crucial that the same binary code is used for simulation and production. While we rely on the work of others regarding the resource models, we have solved the latter problem in our approach.

Our future work will be focused on two aspects. First, we will use PartsSim to study the performance of several networked systems in the ubiquitous computing domain that are based on MicaZ hardware, including an assisted bicycle training scenario [7] and an assisted living system [8]. Also, we will assess the performance of MacZ, a MAC layer for enhanced best effort guarantees [9]. Second, we will incorporate further simulators into PartsSim, in particular, simulators for energy consumption. This aspect is of particular importance for self-organizing, mobile networks.

E. REFERENCES

[1] ITU – International Telecommunications Union: *Specification and Description Language (SDL)*, ITU-T Recommendation Z.100, August 2002

[2] Telelogic AB: *TAU Generation 1*, Telelogic, www.telelogic.com/products/tau/index.cfm

[3] T. Kuhn, A. Geraldy, R. Gotzhein, F. Rothländer: *ns+SDL – The Network Simulator for SDL Systems*, in: A. Prinz, R. Reed, and J. Reed (Eds.), SDL 2005 – Model Driven, LNCS 3530, Springer, 2005, pp. 103-116

[4] *The Network Simulator ns-2*, Information Sciences Institute, University of Southern California, www.isi.edu/nsnam/ns/

[5] Crossbow Technologies Inc.: *MicaZ Wireless Measurement System*, www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAz_Datasheet.pdf

[6] B. L. Titzer, D. K. Lee, J. Palsberg: *Avrora: Scalable Sensor Network Simulation with Precise Timing*, Prod. Of the 4th International Symposium on Information Processing in Sensor Networks (IPSN 2005), Los Angeles, USA, 2005

[7] I. Fliege, A. Geraldy, R. Gotzhein, T. Jaitner, T. Kuhn, C. Webel: *An Ambient Intelligence System to Assist Team Training and Competition in Cycling*, 6th Conference of the International Sports Engineering Association, Germany, 2006

[8] J. Nehmer, A. I. Karshmer, M. Becker: *Living Assistance Systems – An Ambient Intelligence Approach.* Invited talk on 28th International Conference on Software Engineering (ICSE), Shanghai, China, 2006

[9] T. Kuhn: *MacZ – a QoS MAC Layer for Ambient Intelligence Systems*. Adjunct Proceedings of the 4th International Conference on Pervasive Computing, pp. 69-72, Dublin, Ireland, 2006

Worst Case Modelling of Wireless Sensor Networks

Jens B. Schmitt
disco | Distributed Computer Systems Lab, University of Kaiserslautern, Germany
jschmitt@informatik.uni-kl.de

***Abstract*— At the current state of affairs it is hard to obtain a predictable performance from wireless sensor networks, not to mention performance guarantees. In particular, a widely accepted and established methodology for modelling the performance of wireless sensor networks is missing. In the last two years we have tried to make a step into the direction of an analytical framework for the performance modelling of wireless sensor networks based on the theory of network calculus, which we customized towards a so-called *sensor network calculus* [1]. We believe the sensor network calculus to be especially useful for applications which have timing requirements. Examples for this class of applications are factory control, nuclear power plant control, medical applications, and any alerting systems. In general, whenever the sensed input may necessitate immediate actions the sensor network calculus may be the way to go. In this paper we summarize these activities and discuss the open issues for such an analytical framework to be widely accepted.**

I. Introduction

Decisions in daily life are based on the accuracy and availability of information. Sensor networks can significantly improve the quality of information as well as the ways gathering it. For example, sensor networks can help to get higher fidelity information, acquire information in real time, get hard-to-obtain information, and reduce the cost of obtaining information. Application areas for sensor networks might be production surveillance, traffic management, medical care, or military systems. In these areas it is crucial to ensure that the sensor network is functioning even in a worst case scenario. If a sensor network is used for example for production surveillance, it must be ensured that messages indicating a dangerous condition are not dropped, thus avoiding costly production outages. If functionality in worst case scenarios cannot be proven, people might be in danger and the production system might not be certified by authorities.

As it may be difficult or even impossible to produce the worst case in a real world scenario or in a simulation in a controlled fashion, an analytical framework is desirable that allows a worst case performance analysis in sensor networks. Network calculus [2] is a relatively new tool that allows worst case analysis of packet-switched communication networks. In [1] a framework for worst case analysis of wireless sensor networks based on network calculus is presented and called *sensor network calculus*. This framework has further been extended towards random deployments [3] and the case of multiple sinks in [4]. The goal of this paper is to summarize these activities and show the usefulness of the sensor network calculus as well as opportunities for future work along this avenue.

II. Sensor Network Calculus: A Brief Walk-Through

In this section we use the notation and the basic results provided in [1] (a summary of the most important notions of network calculus are given in the Appendix), furthermore a single sink communication pattern is assumed. It is assumed that the routing protocol being used forms a tree in the sensor network. Hence, N sensor nodes arranged in a directed acyclic graph are given.

Each sensor node i senses its environment and thus is exposed to an input function R_i corresponding to its sensed input traffic. If sensor node i is not a leaf node of the tree then it also receives sensed data from all of its child nodes $child(i,1),\ldots,child(i,n_i)$, where n_i is the number of child nodes of sensor node i. Sensor node i forwards/processes its input which results in an output function R_i^* from node i towards its parent node.

Now the basic network calculus components, arrival and service curve, have to be incorporated. First, the arrival curve $\bar{\alpha}_i$ of each sensor node in the field has to be derived. The input of each sensor node in the field, taking into account its sensed input and its children's input, is

$$\bar{R}_i = R_i + \sum_{j=1}^{n_i} R^*_{child(i,j)} \tag{1}$$

Thus, an arrival curve for the total input function for sensor node i is

$$\bar{\alpha}_i = \alpha_i + \sum_{j=1}^{n_i} \alpha^*_{child(i,j)} \tag{2}$$

A. Maximum Sensing Rate Arrival Curve

The simplest option in bounding the sensing input at a given sensor node is based on its maximum sensing rate. This may either be due to the way the sensing unit is designed or due to a limitation on the sensing rate to a certain value by the sensor network application's task in observing a certain phenomenon. For example, it might be known that in a temperature surveillance sensor system, the temperature does not have to be reported more than once per second at most. The arrival curve for a sensor node i corresponding to simply putting a bound on the maximum sensing rate is

$$\alpha_i(t) = p_i t = \gamma_{p_i,0}(t) \tag{3}$$

Here, $\gamma_{r,b} = \begin{cases} rt+b & t>0 \\ 0 & t\leq 0 \end{cases}$ denotes an affine arrival curve. This maximum sensing rate arrival curve can be used

in situations where all sensor nodes are set up to periodically report the condition in a sensor field. The set of sensible arrival curve candidates is certainly larger than the arrival curves described above. The more knowledge on the sensing operation and its characteristics is incorporated into the arrival curve for the sensing input the better the performance bounds become.

B. Rate-Latency Service Curve

Next, the service curve has to be specified. The service curve depends on the way packets are scheduled in a sensor node, which mainly depends on link layer characteristics. More specific, the service curve depends on how the duty cycle and therefore the energy-efficiency goals are set.

The service curve captures the characteristics with which sensor data is forwarded by the sensor nodes towards the sink. It abstracts from the specifics and idiosyncrasies of the link layer and makes a statement on the minimum service that can be assumed even in the worst case. A typical and well-known example of a service curve from traditional traffic control in a packet-switched network is

$$\beta_{R,T}(t) = R\,[t-T]^+ \tag{4}$$

where the notation $[x]^+$ equals x if $x \geq 0$ and 0 otherwise. This is often also called a rate-latency service curve. The latency term T nicely captures the characteristics induced by the application of a duty cycle concept, i.e., the sensor nodes periodically fall asleep for a certain amount of time if they are idle. Whenever the duty cycle approach is applied there is the chance that sensed data or data to be forwarded arrives after the last duty cycle (of the next hop!) is just over and thus a fixed latency occurs until the forwarding capacity is available again. In a simple duty cycle scheme this latency would need to be accounted for for all data transfers. For the forwarding capacity it is assumed that it can be lower bounded by a fixed rate which depends on transceiver speed, the chosen link layer protocol and the duty cycle. So, with some new parameters the following service curve at sensor node i is obtained:

$$\beta_i(t) = \beta_{f_i,l_i}(t) = f_i[t-l_i]^+ \tag{5}$$

Here f_i and l_i denote the forwarding rate and forwarding latency for sensor node i.

C. Network Flow Analysis

Finally, the output of sensor node i, i.e., the traffic which it forwards to its parent in the tree, is constrained by the following arrival curve (see Appendix):

$$\alpha_i^* = \bar{\alpha}_i \oslash \beta_i = \left(\alpha_i + \sum_{j=1}^{n_i} \alpha^*_{child(i,j)}\right) \oslash \beta_i \tag{6}$$

In order to calculate a network-wide characteristic like the maximum information transfer delay or local buffer requirements especially at the most challenged sensor node just below the sink (which is called node 1 from now on) an iterative procedure to calculate the network internal flows is required:

1) Let us assume that arrival curves for the sensed input α_i and service curves β_i for sensor node i, $i = 1, \ldots, N$, are given.
2) For all leaf nodes the output bound α_i^* can be calculated according to (6). Each leaf node is now marked as "calculated".
3) For all nodes only having children which are marked "calculated" the output bound α_i^* can be calculated according to (6) and they can again be marked "calculated".
4) If node 1 is marked "calculated" the algorithm terminates, otherwise go to step 3.

After the network internal flows are computed according to this procedure, the local per node delay bounds D_i for each sensor node i can be calculated according to a basic network calculus result (see appendix):

$$D_i = h(\bar{\alpha}_i, \beta_i) = \sup_{s \geq 0}\{\inf\{\tau \geq 0 : \bar{\alpha}_i(s) \leq \beta_i(s+\tau)\}\} \tag{7}$$

To compute the total information transfer delay $\bar{D}_i$ for a given sensor node i the per node delay bounds on the path $P(i)$ to the sink need to be added:

$$\bar{D}_i = \sum_{i \in P(i)} D_i \tag{8}$$

The maximum information transfer delay in the sensor network can then obviously be calculated as $\bar{D} = \max_{i=1,\ldots,N} \bar{D}_i$. Note that this kind of analysis assumes FIFO scheduling at the sensor nodes, which however should be the case in most practical cases.

III. Sensor Network Calculus at Work

In this section some numerical examples for the previously presented sensor network calculus framework are described. These examples are chosen with the intention of describing realistic and common application scenarios, yet they are certainly simplifying matters to some degree for illustrative purposes. The sensor network calculus framework allows, from a worst case perspective, to relate the following local characteristics:

- *Sensing Activity*: this parameter is described in the framework by the *arrival curv*e concept;
- *Buffer Requirements*: the buffer requirements of each node are described by the *backlog bound;*

to the following global characteristics:

- *Information Transfer Delay*: the delay in each node is described by the *delay bound;*
- *Network Lifetime*: the energy consumption is described by the *duty cycle* represented in the *service curve.*

The goal in using sensor network calculus is to determine specific values for these characteristics for a given application scenario. The scenario itself is characterized by further constraints such as *topology* and *routing*.

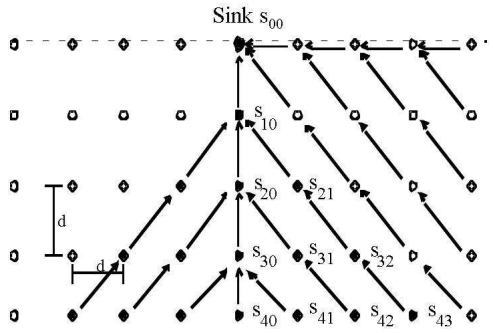

Fig. 1. Sensor Field with Grid Layout.

A. Basic Scenario

The intention of this example is to analytically explore the possible range of the characteristics discussed above in a realistic scenario. Thereafter it is analysed in which operation range a state of the art sensor node could be used to form the sensor field.

1) Topology and Routing: The sensor field is assumed to be a 9x9 grid, the distance between the sensors is d. Fig. 1 shows the lower half of a grid shaped sensor field with the base station (sink) located in its center (node s_{00}). The size of the field is $8d \times 8d$, containing $N = 80$ sensors each with an idealized transmission range of $\sqrt{2}d$.

For the routing protocol, the Greedy Perimeter Stateless Routing (GPSR) protocol is used [6]. All nodes in GPSR must be aware of their position within a sensor field. Each node communicates its current position periodically to its neighbors through beacon packets. In the given static scenario, these beacons have to be transmitted only once. Upon receiving a data packet, a node analyses its geographic destination. If possible, the node always forwards the packet to the neighbor geographically closest to the packet destination. If there is no neighbor geographically closer to the destination, the protocol tries to route around the hole in the sensor field. This routing around a hole is not used in the described topology. In Fig. 1 the resulting structure of the communication paths is shown.

2) Sensing Activity: It is assumed that the sensor field is used to collect data periodically from each of the sensors. Each sensor can report with a maximum report frequency of p. Thus, the maximum sensing rate arrival curve described by (3) is used to model the upper bound of the sensing activity of each node in the sensor field. A homogeneous field is assumed, hence

$$\alpha_i(t) = pt = \gamma_{p,0}(t) \tag{9}$$

Each node additionally receives traffic from its child nodes according to the traffic pattern implied by the topology and the routing protocol (see Figure 1). Therefore, the arrival curve $\bar{\alpha}_i$ for the total input of a sensor node i is given by (2). Later it will be shown in detail how the relevant $\bar{\alpha}_i$ can be calculated.

3) Network Lifetime: To achieve a high network lifetime a duty cycle of $\delta = 1\%$ is set for the nodes in the network. As a sensor node, the Mica-2 [13] platform is assumed. Mica-2 supports a link speed of 19.2 kbit/s. The minimum idle time of the transceiver is $T_1 = 11$[ms] (3ms to begin sampling, 8ms minimum preamble length), the corresponding sleep time is $T_2 = 1085$[ms]. Thus, a maximum packet forwarding rate of 0.89[packets/s] ($f = 258$[bit/s]) can be achieved.[1]The resulting latency for the packet forwarding is $l = T_1 + T_2$. This packet forwarding scheme can be described by the rate-latency service curve as described by equation (5) in Section II-B:

$$\beta_i(t) = \beta_{f,l}(t) = f(t-l)^+ = 258(t - 1.096)^+[bit] \tag{10}$$

4) Calculation: After defining the scenario, the sensor network calculus framework can now be used to evaluate the characteristics of interest and their interdependencies. Goal of the calculation is to determine these characteristics at the sensor node with the worst possible traffic conditions. In this example this is the node s_{10}. If the characteristics in this node are determined and the node is dimensioned to cope with them, all other nodes in the field (assuming homogeneity) are dimensioned properly as well.

To calculate the total traffic pattern, the algorithm described in Section II-C has to be used. First, the output bound α^*_{40} of the leaf node s_{40} has to be calculated using (9), (10) and (16):

$$\alpha_{40} = \gamma_{p,0},\ \beta_{40} = \beta_{f,l},\ \alpha^*_{40} = \alpha_{40} \oslash \beta_{40} = \gamma_{p,pl} \tag{11}$$

The output bound for node s_{40} is also the output bound for the other leaf nodes (e.g., $\alpha^*_{40} = \alpha^*_{41} = \alpha^*_{42} = \alpha^*_{43}$). Now the output bounds for the nodes one level higher in the tree can be calculated using equation (11), (9), (10) and (6):

$$\bar{\alpha}_{30} = \gamma_{p,0} + 3\alpha^*_{40} = \gamma_{p,0} + 3\gamma_{p,pl} = \gamma_{4p,3pl}$$

$$\alpha^*_{30} = \bar{\alpha}_{30} \oslash \beta = \gamma_{4p,7pl} \tag{12}$$

The calculation can now be repeated until node s_{10} is reached: ...

$$\bar{\alpha}_{10} = \gamma_{p,0} + 2\alpha^*_{21} + \alpha^*_{20} = \gamma_{16p,34pl}$$

$$\alpha^*_{10} = \bar{\alpha}_{10} \oslash \beta = \gamma_{16p,50pl} \tag{13}$$

After the arrival curve for node s_{10} is calculated, the worst case buffer requirements B_{10} and the information transfer delay D can be calculated according to equation (14) and (7):

$$B_{10} = v(\bar{\alpha}_{10}, \beta) = 50pl$$

$$D_{10} = h(\bar{\alpha}_{10}, \beta) = l + \frac{34pl}{f},\ D_{20} = h(\bar{\alpha}_{20}, \beta) = l + \frac{13pl}{f}$$

$$D_{30} = h(\bar{\alpha}_{30}, \beta) = l + \frac{3pl}{f},\ D_{40} = h(\bar{\alpha}_{40}, \beta) = l$$

$$D = D_{40} + D_{30} + D_{20} + D_{10} = 4l + \frac{50pl}{f}$$

[1]Values are taken from the TinyOS code (CC1000Const.h). The packet length is 36 bytes, the preamble length for 1% duty cycle is 2654 bytes.

5) Discussion: Now, after all nodes are calculated, it is possible to determine specific values for the characteristics of interest for the given application scenario. Furthermore it is possible to evaluate how these factors influence each other. As mentioned above, due to the channel speed and the selected duty cycle, the effective maximum forwarding speed is $f = 258$[bit/s]. The arrival rate of packets cannot be higher than the maximum forwarding speed. A higher arrival rate would result in an infinite queueing of packets. Therefore, the sensing rate must be set such that $16p \leq f$. In the following, the highest possible integral sensing rate is assumed: $p = \lfloor f/16 \rfloor = 16$[bit/s]. This first result already shows the limits of this specific sensor field regarding its maximum sensing frequency. Translated in TinyOS packets with a standard size of 36 byte, the result shows that each sensor can only send a packet every 18 seconds.

The backlog bound at node s_{10} is now given by: $B_{10} = 50pl = 876.8$[bit]. This result can be translated into TinyOS packets with the standard size of 36 byte. In this case, $\lceil 3.04 \rceil = 4$ packets must be stored in the worst case in node s_{10}. As a Mica-2 node provides per default only a buffer space of one, a node modification would be necessary to support the described scenario in the worst case. The maximum information transfer delay is given by: $D = 4l + \frac{51pl}{f} = 7.85$[s].

To improve the backlog bound and the information transfer delay, the duty cycle used in the nodes can be modified. Of course the improvements have to be paid in this case with a higher energy consumption in the nodes and thus a shorter network lifetime. If the duty cycle is set to 11.5%[2], a maximum packet forwarding rate of 0.54[packets/s] ($f = 2488$[bit/s]) can be achieved. The resulting delay for the packet forwarding is $l = T_1 + T_2 = 11 + 85 = 96$[ms]. Now the following is obtained: $B_{10} = 50pl = 76.8$. In this case now, only 1 TinyOS packets needs to be stored in node s_{10} even under worst case conditions. The information transfer delay is now given by: $D = 4l + \frac{51pl}{f} = 0.41$[s].

IV. Advanced Sensor Network Calculus

After the brief walk-through of the sensor network calculus basics and the illustrative example of its operation, we will discuss some of the more advanced techniques we have developed to further customize network calculus to the wireless sensor network setting as well as some of the applications of the framework we have proposed.

We have seen in the previous sections how the single sink communication pattern typically found in wireless sensor networks was used to iteratively work out the internal traffic flow bounds inside the network and use these to calculate delay bounds in an additive fashion. However, one of the strengths of network calculus is its powerful concatenation result, which allows in general to achieve better bounds when a tandem of servers is first min-plus convoluted to a single system compared to an additive analysis of the separate servers. This concatenation result is not directly applicable in a wireless sensor network scenario even when only considering the simple single sink case. Therefore, we have generalized the concatenation result for general feedforward networks in [5], introducing a principle called "pay multiplexing only once" which makes optimal use of sub-paths shared between flows and achieves improvements over the additive bounds, which may be on the order of magnitudes depending on the scenario. A further extension of the basic sensor network calculus, which we also describe in [5], is the integration of maximum service curves into the sensor network calculus, which allows to improve the bounds on the network-internal flows and thus in turn lowers the performance bounds, again often very considerably. All these techniques, among other general network calculus techniques, have been implemented in the so-called DISCO Network Calculator. As we believe that tool support is of great importance for a wide acceptance of the sensor network calculus we provide the DISCO Network Calculator in the public domain[3].

Apart from trying to push the sensor network calculus forward methodically, we have also illustrated how to apply it for several design and control issues in wireless sensor networks. In [1] we have shown how a buffer dimensioning of the sensor nodes may be performed based on the worst case analyses of sensor network calculus such that no information is lost due to buffer overflow inside the network. Furthermore, we also discussed in [1] how different choices of duty cycles affect the information transfer delay. In [3], we considered the case of a randomly deployed sensor network and how this further dimension of uncertainty can be factored into the sensor network calculus. In particular we discussed how constraints from topology control may be used to improve the performance bounds from the sensor network calculus. Thus, we proposed to guide topology control decisions based on the sensor network calculus models. In [4] we used the advanced sensor network calculus result discussed in the previous paragraph to investigate scenarios with multiple sinks. In particular we demonstrated how sensor network calculus can be used to dimension the number of sinks as well as their placement in the sensor field.

V. Open Issues and Future Work Items

While we believe the sensor network calculus to have potential, there are still many open issues and correspondingly opportunities for future work. One immediate issue arising from the use of a deterministic analytical framework is the question how to capture the inherently stochastic nature of wireless communications. Here, we plan to integrate lately upcoming stochastic extensions of network calculus [7], which however again need to be customized for the sensor network case. Another issue is how to take in-network processing as is frequently proposed for wireless sensor networks into account. In [8] we have proposed a network calculus that allows for the scaling of data flows. This development should enable modelling of typical in-network processing techniques as for example aggregation of information. Furthermore, it should also be possible to accomodate the mobility of sensor nodes and/or sinks. As in many scenarios this is a kind of

[2] A duty cycle value offered by the TinyOS code for the Mica-2.

[3] See http://disco.informatik.uni-kl.de/content/Downloads.

controlled mobility there is hope to capture even this difficult characteristic of advanced wireless sensor network scenarios.

Apart from these fundamental issues for the sensor network calculus, it is also important to demonstrate its usefulness in further applications. At the moment we design a task admission control scheme based on sensor network calculus for sensor networks that may have several concurrent tasks. Another work item could be a scheme where sleeping nodes are activated such that certain performance bounds can still be satisfied. Apart from these issues the presented framework should also be validated by packet-level simulations in order to increase the fidelity in the predictive power of our models. Especially this last point deserves our immediate attention and is already currently under investigation.

REFERENCES

[1] Jens Schmitt and Utz Roedig. Sensor Network Calculus - A Framework for Worst Case Analysis. In Proc. of IEEE/ACM Int. Conf. on Distributed Computing in Sensor Systems (DCOSS'05), pages 141-154. Springer, LNCS 3560, June 2005.

[2] J.-Y. Le Boudec and P. Thiran. Network Calculus - A Theory of Deterministic Queueing Systems for the Internet. Springer, LNCS 2050, 2001.

[3] Jens Schmitt and Utz Roedig. Worst Case Dimensioning of Wireless Sensor Networks under Uncertain Topologies. In Proc. of 3rd International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt'05), Workshop on Resource Allocation in Wireless Networks, Riva del Garda, Italy. IEEE, April 2005.

[4] Jens B. Schmitt, Frank A. Zdarsky, and Utz Roedig. Sensor Network Calculus with Multiple Sinks. In Proc. of IFIP Networking 2006, Workshop on Performance Control in Wireless Sensor Networks, Coimbra, Portugal. Springer LNCS, May 2006.

[5] Jens B. Schmitt, Frank A. Zdarsky, and Ivan Martinovic. Performance Bounds in Feed-Forward Networks under Blind Multiplexing. Technical Report 349/06. University of Kaiserslautern, Germany, April 2006.

[6] B. Karp and H. T. Kung, "GPSR : greedy perimeter stateless routing for wireless networks," in Mobile Computing and Networking, pp. 243-254, 2000.

[7] Yuming Jiang. A Basic Stochastic Network Calculus. In Proc. ACM SIGCOMM 2006, Pisa, Italy, September 2006.

[8] Markus Fidler and Jens B. Schmitt. On the Way to a Distributed Systems Calculus: An End-to-End Network Calculus with Data Scaling. In Proc. ACM SIGMETRICS/Performance 2006 (SIGMETRICS'06), St. Malo, France, June 2006.

APPENDIX:BACKGROUND ON NETWORK CALCULUS

Network calculus is *the* tool to analyse flow control problems in networks with particular focus on determination of bounds on worst case performance. It has been successfully applied as a framework to derive deterministic guarantees on throughput, delay, and to ensure zero loss in packet-switched networks. Network calculus can also be interpreted as a system theory for *deterministic* queueing systems, based on min-plus algebra. What makes it different from traditional queueing theory is that it is concerned with worst case rather than average case or equilibrium behaviour. It thus deals with bounding processes called arrival and service curves rather than arrival and departure processes themselves.

Next some basic definitions and notations are provided before some basic results from network calculus are summarized.

Definition 1: The input function $R(t)$ of an arrival process is the number of bits that arrive in the interval $[0, t]$. In particular $R(0) = 0$, and R is wide-sense increasing, i.e., $R(t_1) \leq R(t_2)$ for all $t_1 \leq t_2$.

Definition 2: The output function $R^*(t)$ of a system S is the number of bits that have left S in the interval $[0, t]$. In particular $R^*(0) = 0$, and R is wide-sense increasing, i.e., $R^*(t_1) \leq R^*(t_2)$ for all $t_1 \leq t_2$.

Definition 3: Min-Plus Convolution. Let f and g be wide-sense increasing and $f(0) = g(0) = 0$. Then their convolution under min-plus algebra is defined as

$$(f \otimes g)(t) = \inf_{0 \leq s \leq t} \{f(t-s) + g(s)\}$$

Definition 4: Min-Plus Deconvolution. Let f and g be wide-sense increasing and $f(0) = g(0) = 0$. Then their deconvolution under min-plus algebra is defined as

$$(f \oslash g)(t) = \sup_{s \geq 0} \{f(t+s) - g(s)\}$$

Now, by means of the min-plus convolution, the arrival and service curve are defined.

Definition 5: Arrival Curve. Let α be a wide-sense increasing function such that $\alpha(t) = 0$ for $t < 0$. α is an arrival curve for an input function R iff $R \leq R \otimes \alpha$. It is also said that R is α-smooth or R is constrained by α.

Definition 6: Service Curve. Consider a system S and a flow through S with R and R^*. S offers a service curve β to the flow iff β is wide-sense increasing and $R^* \geq R \otimes \beta$.

From these, it is now possible to capture the major worst-case properties for data flows: maximum delay and maximum backlog. These are stated in the following theorems.

Theorem 1: Backlog Bound. Let a flow $R(t)$, constrained by an arrival curve α, traverse a system S that offers a service curve β. The backlog $x(t)$ for all t satisfies

$$x(t) \leq \sup_{s \geq 0} \{\alpha(s) - \beta(s)\} = v(\alpha, \beta) \quad (14)$$

$v(\alpha, \beta)$ is also often called the vertical deviation between α and β.

Theorem 2: Delay Bound. Assume a flow $R(t)$, constrained by arrival curve α, traverses a system S that offers a service curve β. At any time t, the virtual delay $d(t)$ satisfies

$$d(t) \leq \sup_{s \geq 0} \{\inf\{\tau \geq 0 : \alpha(s) \leq \beta(s+\tau)\}\} = h(\alpha, \beta) \quad (15)$$

$v(\alpha, \beta)$ is also often called the vertical deviation between α and β.

As a system theory network calculus offers further results on the concatenation of network nodes as well as the output when traversing a single node. Especially the latter for which now the min-plus deconvolution is used will be of high importance in the sensor network setting as it potentially involves a so-called *burstiness increase* when a node is traversed by a data flow.

Theorem 3: Output Bound. Assume a flow $R(t)$ constrained by arrival curve α traverses a system S that offers a service curve β. Then the output function is constrained by the following arrival curve

$$\alpha^* = \alpha \oslash \beta \geq \alpha \quad (16)$$

Theorem 4: Concatenation of Nodes. Assume a flow $R(t)$ traverses systems S_1 and S_2 in sequence where S_1 offers service curve β_1 and S_2 offers β_2. Then the resulting system S, defined by the concatenation of the two systems offers the following service curve to the flow:

$$\beta = \beta_1 \otimes \beta_2 \quad (17)$$

Measures of Developing Stability and Perturbations in Ecologies of Semantic Web Services

Elpida S. Tzafestas
Institute of Communication and Computer Systems
National Technical University of Athens
Zographou Campus 15773, Athens, GREECE.
brensham@softlab.ece.ntua.gr

***Abstract*--We are discussing the issue of performance of engineered self-organizing systems. We are arguing that the qualitative concepts of stability, resistance to perturbations, predictability and development, that are drawn from the natural self-organizing systems, can serve the role of measure classes. We subsequently briefly show how these classes can be instantiated to specific measures within an INFRAWEBS application.**

***Index terms*— self-organization, stability, equilibrium, attractor, perturbation, development, virtual organization**

A. Introduction

We are addressing the class of virtual organizations where an entrepreneurial or other activity acts as an interface between a number of existing remote services or applications and a number of customers that interact with them in complex ways. For example, such activities include private e-government middle services that specialize in carrying out complex tasks with many different state administration services, or commercial houses specializing in centralizing and organizing large complicated orders that involve a number of different vendors scattered around the electronic world.

The INFRAWEBS[1] framework is being designed and developed with the vision to help business people build composite distributed applications that tackle much of this interaction complexity as automatically as possible, with the role of human intervention not necessarily being fully dismissed. Such virtual organizations are characterized by operational and performance criteria that are independent and even sometimes incompatible with those of individual components (external services/applications) involved.

Within this global ecology of services, an INFRAWEBS application will be able to act consistently and achieve reliable performance for the user, if it manages to find stable pathways that fulfill the user-requested goal. It needs therefore to maintain an "image" of the service ecology that will represent at any moment the state of the external world with regard to the application functionality and commercial or other goals. This image or representation of the world has to be self-organizing in order to reflect the autonomous character of the external services.

B. Self-Organization in INFRAWEBS

INFRAWEBS applications exploit semantic web technologies to automate much of the work necessary to identify and locate web services in order to dynamically compose meaningful functionality. More precisely, applications are built and executed with the aid of a number of tools that retrieve, select, match, etc. *semantic* web services, i.e., web services whose requirements, abilities and behavior are described in one of the WSMO-supported formats (cf. Web Service Modelling Ontology website, http://www.wsmo.org/). Two use cases are being developed, one on e-tourism and one on e-government.

Within this effort, we are developing a security reasoner, which is responsible for maintaining an image of the network of external semantic web services used by an application and which follows the artificial immune systems approach.

From a design and development point of view, the usage of semantic web technologies allows the reasoner to automatically import, parse and match web service requirements (actually in WSDL format) and to offer to the system designer a straightforward possibility to select relevant "attributes" to protect.

The INFRAWEBS artificial immune system has been designed on a set of concepts that directly match those features and facilities of semantic web services. Furthermore, it uses a specialized algorithmic setup that differs from the usual setup of artificial immune systems for security.

Each semantic web service participating to an INFRAWEBS application is monitored for a set of attributes by the artificial immune system. The attributes can be either semantically described in the service description or user-defined (the user of the security reasoner is the application designer) and will be continuously given new values during service retrieval, invocation and execution. For example, in the case of e-tourism, it makes sense to use the "price" property of all airline, car rental and hotel services as a monitored or

[1] www.infrawebs.org

protected attribute. In general, global attributes can be also defined that refer to many services together. For example, in the case of e-tourism, the "average price" property of an airliner (that refers to all the individual flights it offers) is a measure of company price policy and is not expected to fluctuate as much as an individual price property of one particular flight. In principle, we can also generate new simple or composite local and global attributes that will be monitored for meaningfulness. For example, such a composite property of the type "price (Air France) > X **and** price (Iberia) > Y" could mean that price rises for the two companies are positively correlated, which could be an indication of collusion. In this case, the level of trust to these two airline companies would have to be negatively affected.

For each of the monitored attributes, an artificial antibody (protector cell) is created that monitors the attribute. Each antibody defines among its other data a target value and an activation level, which express respectively a "normal" value and the deviation from regular behavior. This setup has the following features:

- For each service and for each monitored attribute, there will always be some degree of deviation (for example, prices are never constant). Thus, the problem of detecting abnormalities and dysfunctions translates into how to detect persistent or unusual deviations, by reasoning on the history of the activation level of the antibodies.
- The normal or usual behavior of services may not be (and is usually not) known beforehand (e.g. the average normal price of a flight). Moreover, the so-called normal and usual behavior itself changes, albeit much slower than the local fluctuations. For example, average prices can slightly change (generally increase) with time, but the fluctuations around the average will be much larger than that.

These features imply that the immune network, i.e., the network of antibodies, will have to be in continuous self-organizational activity in order to approximate the, otherwise unknown, normal behavior of the service network. A network that self-organizes and reaches an equilibrium can be said to have discovered the normal behavior of the target system, i.e., of the corresponding service network. From our experience with self-organized systems, we can predict the following dynamic behavior of the immune system:

- More than one immune network can discover the normal state of the same service network.
- For this reason, some of the exact values of the various antibodies of the system may be meaningless to the application designer or may not correspond to his/her own idea about the functioning of the application.
- Continuous operation of the immune system in parallel with the corresponding service network will induce continuous change in the antibody parameters.

While these appear as drawbacks to non-informed application designers, in reality they are advantageous from a self-organization and stability point-of-view, because they allow an equilibrium to be discovered, if it exists. Continuous change in antibody parameters is the underlying activity for self-organization and occurs through two mechanisms: individual antibody adaptation and inter-antibody competition within the application [2].

C. BEHAVIOR, PERFORMANCE, MEASURES

Measures of performance for the distributed application have to be sought that ensure the reliability of the virtual organization and its commercial or other value. To this end, we draw inspiration from biological analogs, and especially ecological networks (food webs), genetic networks, and insect socioregulatory processes.

In natural self-organizing systems, such as the above the quality of behavior "emerges" from the interactions between the components of the system. As such, it is difficult to assess computationally, let alone to predict automatically. Therefore, the quality or performance of such systems is judged on the basis of human observation in simulation ("in vitro") or, if possible, "in vivo". However, the common denominator in all natural self-organizing systems, independently of how well their macro behavior is understood, remains their remarkable ability to recover from perturbations of all kinds, to adapt temporarily to novel conditions and to develop new emergent behaviors with time as response/internalization of persistent changes in their environment. All the emergent behaviors and all the macroscopic qualities that are attributed to them, are only side-effects of this self-organization process [3].

If we want to import this approach to the design and engineering of artifacts, such as distributed computer applications, mobile networks, etc., then we should re-design these artifacts as loosely-connected systems (networks in the general sense) of ill-defined behavior. The desirable functionality would not be explicitly perceived by or described as goal of any one of the components, but it would be only visible to an external observer (human or agent at a higher level of reasoning). One of the first examples of such a system is the artificial ant-based network routing system of [6], where the optimal routes computed are not visible as a whole to any of the individual constituent agents (artificial ants),

but only to an external agent, such as a human application designer.

The concept of macroscopic performance of artificial self-organizing systems as perceived by external entities may be relatively well-defined, contrary to what happens in natural systems. For example, in the above optimal routing problem, the length of the path to a destination constitutes a measure of performance. In INFRAWEBS applications, things will be more complex, because, for example, an absolute safe criterion of quality of an e-tourism plan is hard to formalize. Moreover, different users will assign different values to the same plan, so there are difficulties in assessing quality and proposing plans globally by the application.

Even more difficult, if we want a self-organizing system to develop new behaviors with time, then we should endow it with the possibility to somehow observe itself. Because by definition its constituent parts will only have a limited view of the system, then the behavioral development will have to be emergent as well and rely on individual criteria that differ from what an external observer would perceive.

Finally, our experience with natural self-organizing systems implies that their behavior under stable conditions is generally not optimal because of continuous intrinsic self-organization not triggered by external stress. This feature is the price to pay for the potential of recovery from perturbations, resistance to stress and behavioral development.

As a result of all the above, the first step to successful design of performance measures for a self-organizing system, is the successful design of the system itself as a system demonstrating emergence instead of optimality. The subsequent design of specific measures will have to comply with the ill-defined nature of self-organizing systems and would better be based on generic concepts of self-organization appropriately instantiated.

One characteristic set of such concepts is the quadruple (*stability*, *resistance to perturbations*, *predictability*, *development*). We proceed below to describe in brief how this approach is treated within INFRAWEBS application development.

D. Design of Measures

The most evident manifestation of ***stability*** in a self-organizing system is the existence of an equilibrium. Because this will be emergent, we cannot in general predict in advance which measure will reach an equilibrium. Thus, we should discover this measure during online trials at design time. One approach compatible with the overall line of thought is the selectionist one, according to which many measures can be *generated* and tried for equilibria, and the measures corresponding to the best equilibria selected for adoption.

In INFRAWEBS applications, individual antibodies used should be more-or-less in equilibrium, just like their biological counterparts, and the same should apply to generic population measures (in our case we have used the "average activation level" measure). For example, in an e-commerce application, there exist antibodies for the average price exposed by each service and/or for the average total price. Although we don't expect *any* price to be constant in time, it is reasonable to assume that a constant average price would be one safe criterion of a stable organization, i.e., of one organization in ecological equilibrium. Thus average prices in this case can be used as measures of stability.

As a rule, and from previous experience with biological self-organizing systems, we can expect many global measures to be in equilibrium, even if they don't directly represent real functionalities in the virtual organization, for example the number of services *changing* their exposed price. This is an advantage to us designers, because we can derive many stability criteria for the same organization. This can be also a disadvantage, because the observed variables may not be meaningful from the application's point of view. One design guideline for equilibria measure design is to search ***behavioral attractors*** of the system, i.e., attractors for derived values and measures that express the functionality and operational goals of the system (such as average price), rather than purely structural parameters (such as connectivity metrics).

The second concept of ***resistance to perturbations*** can also be instantiated by systematic trial and error. For example, in an e-commerce application, if suddenly the number of available services for one task doubles, this constitutes a perturbation. If the overall system can regain stability after a while, this is an indication of an operational, "healthy" application. In such cases, the matter of study is the relation between the former and the new equilibria. The analysis of the perturbations goes in two directions: (i) we don't expect *any* perturbation to allow the system to find a new equilibrium, and (ii) sometimes the new equilibrium is unacceptable for the organization.

Thus we need to find ways to ***predict*** system behavior in presence of a number of perturbations and this can be accomplished in off-line ***simulation***. For example, in the previous e-commerce case, we can simulate the system to find the threshold above which the number of newly inserted services will be unmanageable by the system (will not lead to an equilibrium or will lead to an unacceptable equilibrium). The study can be thought to be over when (i) the behavior of the system can be at

least qualitatively predicted within a range of operation, and (ii) the major perturbations considered can be accomodated by the system within acceptable limits (for example, if the number of new services is up to a quarter of the existing services).

We also need to devise and study counter-measures to specific perturbations, i.e., ***counter-perturbations*** that can make up for potentially harmful ones. For example, in the above case we can think of introducing virtual services of other types, or one virtual service that will act as an interface (filter) between the organization and the potentially umlimited number of external services of the critical type.

Lessons from highly perturbed biological systems can be imported, for example from host-parasite ecologies, ecologies with an externally manipulated food source, etc.

The most important aspect of such biologically-inspired equilibria remains however the dynamic nature of the equilibria themselves. For example, in e-commerce applications, we know from everyday experience that, in the long term, average prices are not constant but change as a result of many factors, including inflation, technological upgrades, competition, etc. A virtual organization should be able to maintain by self-organization a correct image of the ***developing equilibrium***, in the same way that a real biological system learns from experience. Although the study of developing, dynamic equilibria depends directly on real data of external world dynamics, we can predict one mechanism that allows this kind of learning. Namely, a second organization, parallel with the first and of the same nature, can reason on medium-term parameters or measures of the organization (and not directly on the state of external services themselves) and self-organize accordingly. ***Meta-reasoning*** of this type can act as a self-regulatory process that allows the organization to maintain a dynamic representation of the global world, much in the way that an insect society regulates the allocation of its work force to different social tasks.

In sum, the qualitative concepts of stability, resistance to perturbations, predictability and development, which are well-defined within self-organization research, have to be instantiated for a particular self-organized system in, possibly many, measures, sometimes overlapping or contradictory. In the case of INFRAWEBS applications, the measures can be generated and the corresponding data collected semi-automatically with only little additional effort, thanks to the use of semantic technologies to describe the capabilities, goals and behaviors of the various services involved.

E. PERSPECTIVES

The study of performance and the definition of measures for ill-defined systems is an open research theme. Previous discussions [4][5] in the domain of intelligent systems that are hard to qualify concluded as well that performance should be qualitatively assessed to fit the ill-defined nature of the domain. In this respect, self-organized systems are much worse than intelligent systems, because they are by definition distributed, mostly large-scale and they exhibit emergent behavior. In another line of research, the formal attributes of particular self-organized systems are studied, outside any application realm, for example see [1][7] for artificial immune systems.

As far as INFRAWEBS applications are concerned, the adopted direction of work allows in the middle term rapid development and prototyping. Farther in this direction, more advanced issues will come into play: incorporation of potentially harmful features (such as viruses) in the system after persistent external manipulation, stabilization to suboptimal equilibria (from an application perspective), cyclic equilibria, etc. In the longer term we can think of cancerous organizations, threats spread, deception of one service by another, etc.

REFERENCES

[1] Garrett, S., "How do we evaluate artificial immune systems?", *Evolutionary Computation*, 13(2):145-178, 2005.

[2] INFRAWEBS Deliverable 8.3.2, Design security and privacy models and application, July 2006.

[3] Kauffman, S.A. *The origins of order - Self-organization and selection in evolution*, Oxford University Press, 1993.

[4] Panel discussion "Performance Metrics for Intelligent Systems" (IEEE *Intelligent Control Systems Conference 2000*, Patras, Greece, July 2000).

[5] *Proceedings of the Workshop on Performance Metrics for Intelligent Systems*, Washington, DC, August 2000, edited by A. Meystel.

[6] Schoonderwoerd, R. Holland, O.E., Bruten, J.L., Rothkrantz, L., "Ant-Based Load Balancing in Telecommunications Networks", *Adaptive Behavior*, 5(2): 169-207, 1996.

[7] Network for Artificial Immune Systems (funded by EPSRC), http://www.elec.york.ac.uk/ARTIST/.