Sven A. Brueckner
Giovanna Di Marzo Serugendo
David Hales
Franco Zambonelli (Eds.)

LNAI 3910

Engineering Self-Organising Systems

Third International Workshop, ESOA 2005
Utrecht, The Netherlands, July 2005
Revised Selected Papers

Springer

Sven A. Brueckner
Giovanna Di Marzo Serugendo David Hales
Franco Zambonelli (Eds.)

Engineering Self-Organising Systems

Third International Workshop, ESOA 2005
Utrecht, The Netherlands, July 25, 2005
Revised Selected Papers

Springer

Series Editors

Jaime G. Carbonell, Carnegie Mellon University, Pittsburgh, PA, USA
Jörg Siekmann, University of Saarland, Saarbrücken, Germany

Volume Editors

Sven A. Brueckner
Altarum Institute
3520 Green Court, Suite 300, Ann Arbor, MI 48105-1579, USA
E-mail: sven.brueckner@altarum.org

Giovanna Di Marzo Serugendo
Birkback (University of London), Computer Science and Information Systems
Malet Street, London WC1E 7HX, UK
E-mail: dimarzo@dcs.bbk.ac.uk

David Hales
University of Bologna, Department of Computer Science
Mura Anteo Zamboni 7, 40127 Bologna, Italy
E-mail: hales@cs.unibo.it

Franco Zambonelli
Università di Modena e Reggio Emilia, Dipartimento di Scienze e Metodi dell'Ingegneria
Via Allegri 13, 42100 Reggio Emilia, Italy
E-mail: franco.zambonelli@unimore.it

Library of Congress Control Number: 2006923000

CR Subject Classification (1998): I.2.11, C.2.4, C.2, D.2.12, D.1.3, H.3, H.4

LNCS Sublibrary: SL 7 – Artificial Intelligence

ISSN 0302-9743
ISBN-10 3-540-33342-8 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-33342-5 Springer Berlin Heidelberg New York

Springer is a part of Springer Science+Business Media

springer.com

Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11734697 06/3142 5 4 3 2 1 0

Lecture Notes in Artificial Intelligence 3910
Edited by J. G. Carbonell and J. Siekmann

Subseries of Lecture Notes in Computer Science

Preface

The idea that self-organisation and emergence can be harnessed for the purpose of solving tricky engineering problems is becoming increasingly accepted. Researchers working in many diverse fields (such as networks, distributed systems, operating systems and agent systems) are beginning to apply this new approach. This book contains recent work from a broad range of areas with the common theme of utilising self-organisation productively.

As distributed information infrastructures continue to spread (such as the Internet, wireless and mobile systems), new challenges have arisen demanding robust and scalable solutions. In these new challenging environments the designers and engineers of global applications and services can seldom rely on centralised control or management, high reliability of devices, or secure environments. At the other end of the scale, ad-hoc sensor networks and ubiquitous computing devices are making it possible to embed millions of smart computing agents into the local environment. Here too systems need to adapt to constant failures and replacement of agents and changes in the environment, without human intervention or centralised management.

Self-organising applications (SOAs) are able to dynamically change their functionality and structure without direct user intervention to meet changes in requirements and their environment. The overall functionality delivered by SOAs typically changes progressively, mainly in a non-linear fashion, until it reaches (emerges to) a state where it satisfies the current system requirements and therefore it is termed self-organising or emergent behaviour. Self-organising behaviour is often the result of the execution of a number of individual application components that locally interact with each other aiming to achieve their local goals, for example, systems that are based on agents or distributed objects. The main characteristic of such systems is their ability to achieve complex collective tasks with relatively simple individual behaviours, without central or hierarchical control.

However, in artificial systems, environmental pressures and local interactions and control may lead to unpredicted or undesirable behaviour. A major open issue is therefore how to engineer desirable emergent behaviour in SOAs and how to avoid undesirable ones given the requirements and the application environment. To address this issue, approaches originating from diverse areas such as non-linear optimisation, knowledge-based programming and constraint problem solving are currently been explored. Furthermore, SOA engineers often take inspiration from the real world, for example from biology, chemistry, sociology and the physical world. Typical examples of SOAs are systems that reproduce socially based insect behaviour, such as ants-based systems, artificial life, or robots. Although the results achieved so far are promising, further work is required until the problem is sufficiently addressed.

More specific fundamental questions that need an answer are: How do we structure the application components and their interactions, so that the self-organisation process results in the desired functionality? How do we validate that the application performs to the requirements within the range of scenarios expected during deployment? What means of influencing the dynamics of the application do we have available and how effective are they? On the one hand, multi-agent simulations and analytic modelling can be used to study emergent behaviour in real systems. On the other hand, results from complexity theory can be applied in engineering of both multi-agent systems and self-organising systems.

To address these issues the ESOA series of workshops was established. The aim is to open a dialog among practitioners from diverse fields, including: agent-based systems, software engineering, information systems, distributed systems, complex systems, optimisation theory and non-linear systems, neural networks, and evolutionary computation. Although backgrounds are diverse, the focus is always clear – to harness self-organising principles to solve difficult engineering problems.

This book includes revised and extended papers presented at the Third ESOA workshop held during the 4th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS) conference held in Utrecht, The Netherlands in July 2005. The workshop received 25 submissions, out of which 12 papers were selected for a long presentation and 6 papers for short presentation.

The first workshop (ESOA 2003) followed a theme of applying nature-inspired models to fields as diverse as network security, manufacturing control, and electronic markets. The second workshop (ESOA 2004) included papers on self-assembly of software, robots task allocations, design methods, and stigmergy-based applications. Both workshops were held during the AAMAS conferences in 2003 and 2004 respectively and post-proceedings are published by Springer, (volumes LNAI 2977 and 3464).

ESOA 2005 included a number of papers related to methodologies and engineering practices. This shows that research in the field of self-organising applications is maturing from novel techniques that work in specific contexts to more general engineering proposals. This book is structured into three parts reflecting the workshop session themes.

Part I presents novel self-organising mechanisms. Jelasity et al. present a self-organising mechanism for maintaining and controlling topology in overlay networks based on gossiping. Georgé et al. describe "emergent programming" through self-organisation of a program's instructions. Picard et al. show how cooperation among agents serves as a self-organisation mechanism in the framework of a distributed timetabling problem. Nowostaswski et al. present the concept of "evolvable virtual machines" architecture for independent programs to evolve into higher levels of hierarchical complexity; Hales presents a P2P re-wiring protocol that allows peers with different skills to spontaneously self-organise into cooperative groups. Dimuro et al. present a self-regulation algorithm for multi-agent systems based on a sociological model of social exchanges.

Armetta et al. discuss a protocol for sharing critical resources based on a two-level self-organised coordination schema.

In Part II methodologies, models and tools for self-organising applications are presented. Brueckner et al. present an agent-based graph colouring model favouring distributed coordination among agents with limited resources in a real-world environment. Marrow et al. describe applications using self-organisation based upon the DIET multi-agent platform. Saenchai et al. present a multi-agent-based algorithm solving the dynamic distributed constraint satisfaction problem. De Wolf et al. present an approach combining simulation and numerical analysis for engineering self-organising systems with some guaranteed macroscopic behaviour. Gardelli et al. discuss self-organising security mechanisms based on the human immune system, and their verification through simulation. Renz et al. discuss the need of using mesoscopic modeling to provide descriptions of emergent behaviour.

Part III presents specific applications of self-organising mechanisms. Ando et al. apply the stigmergy paradigm to automated road traffic management. Fabregas et al. discuss a model inspired from bee behaviour and apply this model to an example of cultural heritage. Van Parunak et al. discuss a sift and sort algorithm for information processing inspired by ants sorting and foraging. Tatara et al. present an agent-based adaptive control approach where local control objectives can be changed in order to obtain global control objectives. Hadeli et al. discuss measures of reactivity of agents in a multi-agent and control approach based on stigmergy.

Finally, we wish to thank all members of the Programme Committee for returning their reviews on time (all papers submitted to the workshop were reviewed by two to three members of the Programme Committee) and for offering useful suggestions on improving the workshop event. Also we thank all those who attended the workshop and contributed to the lively discussions and question and answer sessions.

January 2006

Sven Brueckner
Giovanna Di Marzo Serugendo
David Hales
Franco Zambonelli
Organising Committee
ESOA 2005

Organization

Programme Committee

Yaneer Bar-Yam, New England Complex Systems Institute, USA
Sergio Camorlinga, University of Manitoba, Canada
Vincent Cicirello, Drexel University, USA
Marco Dorigo, IRIDIA, Université Libre de Bruxelles, Belgium
Noria Foukia, University of Southern California, USA
Maria Gini, University of Minnesota, USA
Marie-Pierre Gleizes, IRIT Toulouse, France
Salima Hassas, University of Lyon, France
Manfred Hauswirth, Swiss Federal Institute of Technology, Switzerland
Mark Jelasity, University of Bologna, Italy
Margaret Jefferies, The University of Waikato, New Zealand
Manolis Koubarakis, Technical University of Crete, Greece
Mark Klein, MIT Sloan School of Management, USA
Ghita Kouadri Mostefaoui, University of Fribourg, Switzerland
Soraya Kouadri Mostefaoui, University of Fribourg, Switzerland
Marco Mamei, University of Modena and Reggio Emilia, Italy
Paul Marrow, BT, UK
Philippe Massonet, CETIC, Belgium
Alberto Montresor, University of Bologna, Italy
Andrea Omicini, University of Bologna, Italy
Daniel Polani, University of Hertfordshire, UK
Martin Purvis, University of Otago, New Zealand
Mikhail Smirnov, Fraunhofer Fokus, Berlin, Germany
Paul Valckenaers, Katholieke Universiteit Leuven, Belgium

Lecture Notes in Artificial Intelligence (LNAI)

Vol. 3910: S.A. Brueckner, G. Di Marzo Serugendo, D. Hales, F. Zambonelli (Eds.), Engineering Self-Organising Systems. XII, 245 pages. 2006.

Vol. 3904: M. Baldoni, U. Endriss, A. Omicini, P. Torroni (Eds.), Declarative Agent Languages and Technologies III. XII, 245 pages. 2006.

Vol. 3899: S. Frintrop, VOCUS: A Visual Attention System for Object Detection and Goal-Directed Search. XIV, 216 pages. 2006.

Vol. 3890: S.G. Thompson, R. Ghanea-Hercock (Eds.), Defence Applications of Multi-Agent Systems. XII, 141 pages. 2006.

Vol. 3885: V. Torra, Y. Narukawa, A. Valls, J. Domingo-Ferrer (Eds.), Modeling Decisions for Artificial Intelligence. XII, 374 pages. 2006.

Vol. 3881: S. Gibet, N. Courty, J.-F. Kamp (Eds.), Gesture in Human-Computer Interaction and Simulation. XIII, 344 pages. 2006.

Vol. 3874: R. Missaoui, J. Schmidt (Eds.), Formal Concept Analysis. X, 309 pages. 2006.

Vol. 3873: L. Maicher, J. Park (Eds.), Charting the Topic Maps Research and Applications Landscape. VIII, 281 pages. 2006.

Vol. 3863: M. Kohlhase (Ed.), Mathematical Knowledge Management. XI, 405 pages. 2006.

Vol. 3862: R.H. Bordini, M. Dastani, J. Dix, A.E.F. Seghrouchni (Eds.), Programming Multi-Agent Systems. XIV, 267 pages. 2006.

Vol. 3849: I. Bloch, A. Petrosino, A.G.B. Tettamanzi (Eds.), Fuzzy Logic and Applications. XIV, 438 pages. 2006.

Vol. 3848: J.-F. Boulicaut, L. De Raedt, H. Mannila (Eds.), Constraint-Based Mining and Inductive Databases. X, 401 pages. 2006.

Vol. 3847: K.P. Jantke, A. Lunzer, N. Spyratos, Y. Tanaka (Eds.), Federation over the Web. X, 215 pages. 2006.

Vol. 3835: G. Sutcliffe, A. Voronkov (Eds.), Logic for Programming, Artificial Intelligence, and Reasoning. XIV, 744 pages. 2005.

Vol. 3830: D. Weyns, H. V.D. Parunak, F. Michel (Eds.), Environments for Multi-Agent Systems II. VIII, 291 pages. 2006.

Vol. 3817: M. Faundez-Zanuy, L. Janer, A. Esposito, A. Satue-Villar, J. Roure, V. Espinosa-Duro (Eds.), Nonlinear Analyses and Algorithms for Speech Processing. XII, 380 pages. 2006.

Vol. 3814: M. Maybury, O. Stock, W. Wahlster (Eds.), Intelligent Technologies for Interactive Entertainment. XV, 342 pages. 2005.

Vol. 3809: S. Zhang, R. Jarvis (Eds.), AI 2005: Advances in Artificial Intelligence. XXVII, 1344 pages. 2005.

Vol. 3808: C. Bento, A. Cardoso, G. Dias (Eds.), Progress in Artificial Intelligence. XVIII, 704 pages. 2005.

Vol. 3802: Y. Hao, J. Liu, Y.-P. Wang, Y.-m. Cheung, H. Yin, L. Jiao, J. Ma, Y.-C. Jiao (Eds.), Computational Intelligence and Security, Part II. XLII, 1166 pages. 2005.

Vol. 3801: Y. Hao, J. Liu, Y.-P. Wang, Y.-m. Cheung, H. Yin, L. Jiao, J. Ma, Y.-C. Jiao (Eds.), Computational Intelligence and Security, Part I. XLI, 1122 pages. 2005.

Vol. 3789: A. Gelbukh, Á. de Albornoz, H. Terashima-Marín (Eds.), MICAI 2005: Advances in Artificial Intelligence. XXVI, 1198 pages. 2005.

Vol. 3782: K.-D. Althoff, A. Dengel, R. Bergmann, M. Nick, T.R. Roth-Berghofer (Eds.), Professional Knowledge Management. XXIII, 739 pages. 2005.

Vol. 3763: H. Hong, D. Wang (Eds.), Automated Deduction in Geometry. X, 213 pages. 2006.

Vol. 3755: G.J. Williams, S.J. Simoff (Eds.), Data Mining. XI, 331 pages. 2006.

Vol. 3735: A. Hoffmann, H. Motoda, T. Scheffer (Eds.), Discovery Science. XVI, 400 pages. 2005.

Vol. 3734: S. Jain, H.U. Simon, E. Tomita (Eds.), Algorithmic Learning Theory. XII, 490 pages. 2005.

Vol. 3721: A.M. Jorge, L. Torgo, P.B. Brazdil, R. Camacho, J. Gama (Eds.), Knowledge Discovery in Databases: PKDD 2005. XXIII, 719 pages. 2005.

Vol. 3720: J. Gama, R. Camacho, P.B. Brazdil, A.M. Jorge, L. Torgo (Eds.), Machine Learning: ECML 2005. XXIII, 769 pages. 2005.

Vol. 3717: B. Gramlich (Ed.), Frontiers of Combining Systems. X, 321 pages. 2005.

Vol. 3702: B. Beckert (Ed.), Automated Reasoning with Analytic Tableaux and Related Methods. XIII, 343 pages. 2005.

Vol. 3698: U. Furbach (Ed.), KI 2005: Advances in Artificial Intelligence. XIII, 409 pages. 2005.

Vol. 3690: M. Pěchouček, P. Petta, L.Z. Varga (Eds.), Multi-Agent Systems and Applications IV. XVII, 667 pages. 2005.

Vol. 3684: R. Khosla, R.J. Howlett, L.C. Jain (Eds.), Knowledge-Based Intelligent Information and Engineering Systems, Part IV. LXXIX, 933 pages. 2005.

Vol. 3683: R. Khosla, R.J. Howlett, L.C. Jain (Eds.), Knowledge-Based Intelligent Information and Engineering Systems, Part III. LXXX, 1397 pages. 2005.

Vol. 3682: R. Khosla, R.J. Howlett, L.C. Jain (Eds.), Knowledge-Based Intelligent Information and Engineering Systems, Part II. LXXIX, 1371 pages. 2005.

Vol. 3681: R. Khosla, R.J. Howlett, L.C. Jain (Eds.), Knowledge-Based Intelligent Information and Engineering Systems, Part I. LXXX, 1319 pages. 2005.

Vol. 3673: S. Bandini, S. Manzoni (Eds.), AI*IA 2005: Advances in Artificial Intelligence. XIV, 614 pages. 2005.

Vol. 3662: C. Baral, G. Greco, N. Leone, G. Terracina (Eds.), Logic Programming and Nonmonotonic Reasoning. XIII, 454 pages. 2005.

Vol. 3661: T. Panayiotopoulos, J. Gratch, R.S. Aylett, D. Ballin, P. Olivier, T. Rist (Eds.), Intelligent Virtual Agents. XIII, 506 pages. 2005.

Vol. 3658: V. Matoušek, P. Mautner, T. Pavelka (Eds.), Text, Speech and Dialogue. XV, 460 pages. 2005.

Vol. 3651: R. Dale, K.-F. Wong, J. Su, O.Y. Kwong (Eds.), Natural Language Processing – IJCNLP 2005. XXI, 1031 pages. 2005.

Vol. 3642: D. Ślęzak, J. Yao, J.F. Peters, W. Ziarko, X. Hu (Eds.), Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing, Part II. XXIII, 738 pages. 2005.

Vol. 3641: D. Ślęzak, G. Wang, M. Szczuka, I. Düntsch, Y. Yao (Eds.), Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing, Part I. XXIV, 742 pages. 2005.

Vol. 3635: J.R. Winkler, M. Niranjan, N.D. Lawrence (Eds.), Deterministic and Statistical Methods in Machine Learning. VIII, 341 pages. 2005.

Vol. 3632: R. Nieuwenhuis (Ed.), Automated Deduction – CADE-20. XIII, 459 pages. 2005.

Vol. 3630: M.S. Capcarrère, A.A. Freitas, P.J. Bentley, C.G. Johnson, J. Timmis (Eds.), Advances in Artificial Life. XIX, 949 pages. 2005.

Vol. 3626: B. Ganter, G. Stumme, R. Wille (Eds.), Formal Concept Analysis. X, 349 pages. 2005.

Vol. 3625: S. Kramer, B. Pfahringer (Eds.), Inductive Logic Programming. XIII, 427 pages. 2005.

Vol. 3620: H. Muñoz-Ávila, F. Ricci (Eds.), Case-Based Reasoning Research and Development. XV, 654 pages. 2005.

Vol. 3614: L. Wang, Y. Jin (Eds.), Fuzzy Systems and Knowledge Discovery, Part II. XLI, 1314 pages. 2005.

Vol. 3613: L. Wang, Y. Jin (Eds.), Fuzzy Systems and Knowledge Discovery, Part I. XLI, 1334 pages. 2005.

Vol. 3607: J.-D. Zucker, L. Saitta (Eds.), Abstraction, Reformulation and Approximation. XII, 376 pages. 2005.

Vol. 3601: G. Moro, S. Bergamaschi, K. Aberer (Eds.), Agents and Peer-to-Peer Computing. XII, 245 pages. 2005.

Vol. 3600: F. Wiedijk (Ed.), The Seventeen Provers of the World. XVI, 159 pages. 2006.

Vol. 3596: F. Dau, M.-L. Mugnier, G. Stumme (Eds.), Conceptual Structures: Common Semantics for Sharing Knowledge. XI, 467 pages. 2005.

Vol. 3593: V. Mařík, R. W. Brennan, M. Pěchouček (Eds.), Holonic and Multi-Agent Systems for Manufacturing. XI, 269 pages. 2005.

Vol. 3587: P. Perner, A. Imiya (Eds.), Machine Learning and Data Mining in Pattern Recognition. XVII, 695 pages. 2005.

Vol. 3584: X. Li, S. Wang, Z.Y. Dong (Eds.), Advanced Data Mining and Applications. XIX, 835 pages. 2005.

Vol. 3581: S. Miksch, J. Hunter, E.T. Keravnou (Eds.), Artificial Intelligence in Medicine. XVII, 547 pages. 2005.

Vol. 3577: R. Falcone, S. Barber, J. Sabater-Mir, M.P. Singh (Eds.), Trusting Agents for Trusting Electronic Societies. VIII, 235 pages. 2005.

Vol. 3575: S. Wermter, G. Palm, M. Elshaw (Eds.), Biomimetic Neural Learning for Intelligent Robots. IX, 383 pages. 2005.

Vol. 3571: L. Godo (Ed.), Symbolic and Quantitative Approaches to Reasoning with Uncertainty. XVI, 1028 pages. 2005.

Vol. 3559: P. Auer, R. Meir (Eds.), Learning Theory. XI, 692 pages. 2005.

Vol. 3558: V. Torra, Y. Narukawa, S. Miyamoto (Eds.), Modeling Decisions for Artificial Intelligence. XII, 470 pages. 2005.

Vol. 3554: A.K. Dey, B. Kokinov, D.B. Leake, R. Turner (Eds.), Modeling and Using Context. XIV, 572 pages. 2005.

Vol. 3550: T. Eymann, F. Klügl, W. Lamersdorf, M. Klusch, M.N. Huhns (Eds.), Multiagent System Technologies. XI, 246 pages. 2005.

Vol. 3539: K. Morik, J.-F. Boulicaut, A. Siebes (Eds.), Local Pattern Detection. XI, 233 pages. 2005.

Vol. 3538: L. Ardissono, P. Brna, A. Mitrović (Eds.), User Modeling 2005. XVI, 533 pages. 2005.

Vol. 3533: M. Ali, F. Esposito (Eds.), Innovations in Applied Artificial Intelligence. XX, 858 pages. 2005.

Vol. 3528: P.S. Szczepaniak, J. Kacprzyk, A. Niewiadomski (Eds.), Advances in Web Intelligence. XVII, 513 pages. 2005.

Vol. 3518: T.-B. Ho, D. Cheung, H. Liu (Eds.), Advances in Knowledge Discovery and Data Mining. XXI, 864 pages. 2005.

Vol. 3508: P. Bresciani, P. Giorgini, B. Henderson-Sellers, G. Low, M. Winikoff (Eds.), Agent-Oriented Information Systems II. X, 227 pages. 2005.

Vol. 3505: V. Gorodetsky, J. Liu, V.A. Skormin (Eds.), Autonomous Intelligent Systems: Agents and Data Mining. XIII, 303 pages. 2005.

Vol. 3501: B. Kégl, G. Lapalme (Eds.), Advances in Artificial Intelligence. XV, 458 pages. 2005.

Vol. 3492: P. Blache, E.P. Stabler, J.V. Busquets, R. Moot (Eds.), Logical Aspects of Computational Linguistics. X, 363 pages. 2005.

Vol. 3490: L. Bolc, Z. Michalewicz, T. Nishida (Eds.), Intelligent Media Technology for Communicative Intelligence. X, 259 pages. 2005.

Vol. 3488: M.-S. Hacid, N.V. Murray, Z.W. Raś, S. Tsumoto (Eds.), Foundations of Intelligent Systems. XIII, 700 pages. 2005.

Vol. 3487: J.A. Leite, P. Torroni (Eds.), Computational Logic in Multi-Agent Systems. XII, 281 pages. 2005.

Vol. 3476: J.A. Leite, A. Omicini, P. Torroni, P. Yolum (Eds.), Declarative Agent Languages and Technologies II. XII, 289 pages. 2005.

¥385.00元

Table of Contents

Part I: Self-organising Mechanisms

Part II: Methodologies, Models and Tools

Part III: Applications

T-Man: Gossip-Based Overlay Topology Management*

Márk Jelasity** and Ozalp Babaoglu

University of Bologna,
Dipartimento di Scienze dell'Informazione,
Mura Anteo Zamboni 7, 40126 Bologna, Italy
{jelasity, babaoglu}@cs.unibo.it

Abstract. Overlay topology plays an important role in P2P systems. Topology serves as a basis for achieving functions such as routing, searching and information dissemination, and it has a major impact on their efficiency, cost and robustness. Furthermore, the solution to problems such as sorting and clustering of nodes can also be interpreted as a topology. In this paper we propose a generic protocol, T-MAN, for constructing and maintaining a large class of topologies. In the proposed framework, a topology is defined with the help of a *ranking function*. The nodes participating in the protocol can use this ranking function to order any set of other nodes according to preference for choosing them as a neighbor. This simple abstraction makes it possible to control the self-organization process of topologies in a straightforward, intuitive and flexible manner. At the same time, the T-MAN protocol involves only local communication to increase the quality of the current set of neighbors of each node. We show that this bottom-up approach results in fast convergence and high robustness in dynamic environments. The protocol can be applied as a standalone solution as well as a component for recovery or bootstrapping of other protocols.

1 Introduction

In large, dynamic, fully distributed systems, such as peer-to-peer (P2P) networks, nodes (peers) must be organized in a connected network to be able to communicate with each other and to implement functions and services. The neighbors of the nodes—the "who is connected to whom", or "who knows whom" relation—define the *overlay topology* of the distributed system in question. This topology can dynamically change in time, and in every time point, it defines the possible interactions between the nodes.

Although it would be desirable, it is typically very difficult to ensure that all nodes are aware of every other participating node in the system. The reason is

* This work was partially supported by the Future and Emerging Technologies unit of the European Commission through Project BISON (IST-2001-38923) and DELIS (IST-2002-001907).

** Also with MTA RGAI, SZTE, Szeged, Hungary.

S.A. Brueckner et al. (Eds.): ESOA 2005, LNAI 3910, pp. 1–15, 2006.

that the set of participating nodes changes quickly, and (due to the large number of nodes) it is not feasible to maintain a complete list of the nodes. This means that all nodes are aware of only a limited subset of other nodes, so efficient and robust algorithms are necessary to create, maintain and optimize the topology.

Overlay topology forms the basis for, or has a major impact on many functions. It is well known that functions such as searching, routing, information dissemination, data aggregation, etc, need special topologies for good performance and high efficiency. Furthermore, solutions to other problems including sorting and clustering can be readily expressed as topologies. For example, in the case of sorting, we are looking for a linear structure that represents some total ordering relation. For all these functions, numerous topologies have been suggested and even more protocols to construct and repair them have been proposed.

Motivated by these observations, we consider topology management as a general purpose function that is desirable in distributed systems. In this paper we specifically target very large scale and highly dynamic systems. Key requirements of topology management in such environments include robustness, scalability, flexibility and simplicity. Besides, it is a great advantage if a topology manager is flexible enough to allow for changing the managed topology at run time *on demand*, without having to develop a new protocol for each possible topology from scratch. Since topology is a very general abstraction, that can be used to express solutions to problems and to enhance and support other functions, such functionality would allow us to increase the efficiency of deploying fully distributed application dramatically. We would need only one running topology component and the application area of the system could be changed at run time whenever necessary. With a protocol that supports quickly changing topologies, it even becomes possible to automatically *evolve* topologies through, for example, an evolutionary process.

In this paper we propose a generic protocol, T-MAN, with the aim of fulfilling the requirements outlined above. The desired topology is described using a single ranking function that all nodes can apply to order any subset of potential neighbors according to preference for actually being selected as a neighbor. Using only local gossip messages, T-MAN gradually evolves the current topology towards the desired target structure with the help of the ranking function. We show experimentally that the protocol is scalable and fast, with convergence times that grow only as the logarithm of the network size. These properties allow T-MAN to be practical even when several different topologies have to be created on demand, and also in dynamic systems where the set of nodes or their properties change rapidly. Additionally, the general formulation of the ranking function allows us to deal with a wide range of different topologies.

Although this work is concerned mainly with exploring the basic properties of T-MAN by examining simple topologies like ring, mesh and binary tree, it is possible to illustrate its practicality with more realistic applications. We briefly outline three such applications: sorting, clustering and a distributed hash table (DHT).

Related work includes gossip-based protocols, that have gained notable popularity in various contexts [1, 2, 14]. In this paper we suggest a novel application of the gossip communication model to solve the topology management problem. Issues related to topology management itself have also received considerable attention. Examples from the vast literature include DHTs [7, 11, 13], unstructured overlays [9,3], and superpeer topologies [16]. As for topology construction, Massoulié and Kermarrec [6] propose a protocol to evolve a topology that reflects proximity, Voulgaris and van Steen [15] propose a method to jump-start Pastry. Unlike these specific solutions, T-MAN is a generic framework and can be used to construct and maintain a large class of different topologies quickly in a simple and scalable manner.

2 The Problem

We assume that we are given a (perhaps random) overlay network, and we are interested in constructing some desirable topology by connecting all nodes in the network to the right neighbors. The topology can be defined in many different ways and it will typically depend on some properties of the nodes like geographical location, semantic description of stored content, storage capacity, etc. We need a formal framework that is simple yet powerful enough to be able to capture most of the interesting structures. Our proposal is the *ranking function* that defines the target topology through allowing all nodes to sort any subset of nodes (potential neighbors) according to preference to be selected as their neighbor.

For a more formal definition, let us first define some basic concepts. We consider a set of nodes connected through a routed network. Each node has an address that is necessary and sufficient for sending it a message. Nodes maintain addresses of other nodes through *partial views* (*views* for short), which are sets of c *node descriptors*. In addition to an address, a node descriptor contains a *profile*, which contains those properties of the nodes that are relevant for defining the topology, such as ID, geographical location, etc. The addresses contained in views at nodes define the links of the *overlay network topology*, or simply the *topology*. Note that parameter c defines the node degree of the overlay network and is uniform for all nodes.

We can now define the *topology construction problem*. The input of the problem is a set of N nodes, the view size c and a *ranking function* R that can order a list of nodes according to preference from a given node. The ranking function R takes as parameters a base node x and a set of nodes $\{y_1, \ldots, y_m\}$ and outputs a set of orderings of these m nodes. The task is to construct the views of the nodes such that the view of node x, denoted view_x, contains exactly the first c elements of a "good" ranking of the entire node set, that is, $R(x, \{\text{all nodes except } x\})$ contains a ranking that starts with the elements of view_x. We will call this topology the *target topology*.

In the presence of churn (ie, when nodes constantly join and leave the overlay network) we talk about maintenance of the target topology instead of construction. Instead of a formal definition, we define the problem as staying "as close as possible" to the target topology. The actual figures of merit to characterize maintenance can be largely application dependent in this case.

One (but not the only) way of obtaining ranking functions is through a distance function that defines a metric space over the set of nodes. The ranking function can simply order the given set according to increasing distance from the base node. Let us define some example distance-based topologies of different characteristics. From now on, to simplify our language and notation, we use the nodes and their profiles interchangeably.

Line and ring. The profile of a node is a real number. The distance function for the line is $d(a,b) = |a-b|$. In the case of a ring, profiles are from an interval $[0,N]$ and distance is defined by $d(a,b) = \min(N-|a-b|, |a-b|)$ Ranking is defined through this distance function as described above.

Mesh, tube and torus. The 1-dimensional topology defined above can be easily generalized to arbitrary dimensions to get for example a mesh or a torus. The profiles are two-dimensional real vectors. The distance for the mesh is the Manhattan distance. It is given by calculating the 1-dimensional distance described above along the two coordinates and returning the sum of these distances. Applying the periodic boundary condition (as for the ring) results in a tube for one coordinate and a three dimensional torus for both coordinates.

Binary tree. A low diameter topology can be constructed from a binary tree: the profiles are binary strings of length m, excluding the all zero string. Distance is defined as the shortest path length between the two nodes in the following undirected rooted binary tree. The string $0\ldots01$ is the root. Any string $0a_2\ldots a_m$ has two children $a_2\ldots a_m0$ and $a_2\ldots a_m1$. Strings starting with 1 are leafs. This topology is of interest because (unlike the previous ones) it has a very short (logarithmic) diameter of $2m$.

There are very important ranking functions that cannot be defined by a global distance function, therefore the ranking function is a more general concept than distance. The ranking functions that define sorting or proximity topologies belong to this category. Examples will be given in Section 6.1.

3 The Proposed Solution

The topology construction problem becomes interesting when c is small and the number of nodes is very large. Randomized, gossip-based approaches in similar settings, but for other problem domains like information dissemination or data aggregation, have proven to be successful [2, 4]. Our solution to topology construction is also based on a gossip communication scheme.

3.1 The Protocol

Each node executes the same protocol shown in Figure 1. The protocol consists of two threads: an active thread initiating communication with other nodes, and a passive thread waiting for incoming messages.

Each nodes maintains a view. The view is a set of node descriptors. A call to MERGE(view_1,view_2) returns the union of view_1 and view_2.

do at a random time once in each
consecutive interval of T time units
  $p \leftarrow$ selectPeer()
  myDescriptor $\leftarrow$ (myAddress,myProfile)
  buffer $\leftarrow$ merge(view,{myDescriptor})
  buffer $\leftarrow$ merge(buffer,rnd.view)
  send buffer to p
  receive buffer$_p$ from p
  buffer $\leftarrow$ merge(buffer$_p$,view)
  view $\leftarrow$ selectView(buffer)

(a) active thread

do forever
  receive buffer$_q$ from q
  myDescriptor $\leftarrow$ (myAddress,myprofile)
  buffer $\leftarrow$ merge(view,{myDescriptor})
  buffer $\leftarrow$ merge(buffer,rnd.view)
  send buffer to q
  buffer $\leftarrow$ merge(buffer$_q$,view)
  view $\leftarrow$ selectView(buffer)

(b) passive thread

Fig. 1. The T-MAN protocol

The two key methods are SELECTPEER and SELECTVIEW. Method SELECTPEER uses the current view to return an address. First, it applies the ranking function to order the elements in the view. Next, it returns the first descriptor (according to this ordering) that belongs to a live node. Method SELECTVIEW(BUFFER) also applies the ranking function to order the elements in the buffer. Subsequently, it returns the *first c elements* of the buffer according to ranking order.

The underlying idea is that in this manner nodes improve their views using the views of their current neighbors, so that their new neighbors will be "closer" according to the target topology. Since all nodes do the same concurrently, neighbors in the subsequent topologies will be gradually closer and closer. This also means that the views of the neighbors will keep serving as a useful source of additional, even better links for the next iteration.

Last but not least, we need to explain the origin and role of the buffer RND.VIEW. This buffer contains a random sample of the nodes from the entire network. It is provided by a *peer sampling service* [3]. The peer sampling service described in [3] is implemented in a very similar fashion: nodes periodically exchange their random views and update their local views thereby creating a new random sample. These random views define an approximately random overlay network. The buffer RND.VIEW is the current set of neighbors in this random overlay network. The peer sampling service is extremely robust to failure and maintains a connected network with a very high probability.

The role of the random buffer is most important in large diameter topologies. In this case, if a node has a low quality neighbor set and if most of the rest of the nodes have a high quality neighbor set (forming a large diameter topology, e.g., a ring), then this node needs to perform many exchanges until it can reach the optimal set of neighbors, because the speed of "finding its neighborhood" is related to the diameter of the topology. The random buffer adds long range links that help speeding up convergence.

Although the protocol is not synchronous, it is often convenient to refer to *cycles* of the protocol. We define a cycle to be a time interval of $T/2$ time units where T is the parameter of the protocol in Figure 1. Note that during a cycle, each node is updated once on the average.