Alexander Keller
Jean-Philippe Martin-Flatin (Eds.)

LNCS 3996

Self-Managed Networks, Systems, and Services

Second IEEE International Workshop, SelfMan 2006
Dublin, Ireland, June 2006
Proceedings

Springer

Alexander Keller
Jean-Philippe Martin-Flatin (Eds.)

Self-Managed Networks, Systems, and Services

Second IEEE International Workshop, SelfMan 2006
Dublin, Ireland, June 16, 2006
Proceedings

Springer

Volume Editors

Alexander Keller
IBM T.J. Watson Research Center
P.O. Box 704, Yorktown Heights, NY 10598, USA
E-mail: alexk@us.ibm.com

Jean-Philippe Martin-Flatin
UQAM, Laboratoire de Téléinformatique
Département d'Informatique
Case postale 8888, Succursale Centre-Ville, Montréal, Québec H3C 3P8, Canada
E-mail: jp.martin-flatin@ieee.org

Library of Congress Control Number: 2006926662

CR Subject Classification (1998): C.2, D.4.4, H.4.3, I.2.11

LNCS Sublibrary: SL 5 – Computer Communication Networks and Telecommunications

ISSN 0302-9743
ISBN-10 3-540-34739-9 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-34739-2 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11767886 06/3142 5 4 3 2 1 0

Preface

This volume of the *Lecture Notes in Computer Science* series contains all the papers accepted for presentation at the second IEEE International Workshop on Self-Managed Networks, Systems and Services (SelfMan 2006), which was held at University College Dublin, Ireland on June 16, 2006.

This workshop follows up on a very successful edition that took place last year in Nice, France. The online proceedings of SelfMan 2005 are available at *http://madynes.loria.fr/selfman2005/*.

The objectives of this year's edition were to bring together people from different communities (networking, distributed systems, software engineering, P2P, service engineering, distributed artificial intelligence, robotics, etc.) and cross-pollinate their experience in designing and implementing self-managed networks, systems and services.

We received 51 papers from 21 countries, of which 12 were selected. The acceptance ratio was below 24%. In addition, we selected three work-in-progress papers for short presentations. This one-day event was structured so as to encourage discussions and foster collaborations.

The breadth of the topics presented herein reflects the current interest and developments in this rapidly growing field. It is also a testimony to the promises of self-management to design, operate and manage today's increasingly complex and heterogeneous networks, systems and services.

SelfMan 2006 was co-located with the third IEEE International Conference on Autonomic Computing (ICAC 2006). It was sponsored by the IEEE Computer Society's Task Force on Autonomous and Autonomic Systems (TFAAS) and Technical Committee on Parallel Processing (TCPP), in cooperation with the ACM Special Interest Groups on Operating Systems (SIGOPS) and Artificial Intelligence (SIGART), the IEEE Systems, Man, and Cybernetics Society (SMC), and the IFIP Working Group 6.6 on Management of Networks and Distributed Systems (WG6.6).

The outstanding quality of this workshop's technical program owes a good deal to the members of the Technical Program Committee, who encouraged colleagues in the field to submit papers and devoted much time to review papers. We sincerely thank them, as well as the few external reviewers who also took part in the review process. Finally, we are grateful to the corporate patrons of SelfMan 2006, Cisco and BT, for their generous donations.

New York and Montreal, June 2006

Alexander Keller
Jean-Philippe Martin-Flatin

Organization

Conference Chairs

Alexander Keller — *IBM T.J. Watson Research Center, Yorktown Heights, NY, USA*

Jean-Philippe Martin-Flatin — *University of Quebec in Montreal, Canada*

Sponsored by

Institute of Electrical and Electronics Engineers (IEEE)

IEEE Computer Society

In cooperation with

ACM SIGOPS, ACM SIGART, IEEE SMC and IFIP WG6.6

Corporate Patrons

Steering Committee

Kurt Geihs, *University of Kassel, Germany*
Joe Sventek, *University of Glasgow, UK*

Technical Program Committee

Ozalp Babaoglu, *University of Bologna, Italy*
Raouf Boutaba, *University of Waterloo, Canada*
Geoff Coulson, *Lancaster University, UK*
Giovanna Di Marzo Serugendo, *Birkbeck College, University of London, UK*
Jim Dowling, *MySQL, Sweden*

David Garlan, *Carnegie Mellon University, USA*
Joseph L. Hellerstein, *IBM T.J. Watson Research Center, USA*
Michael Hinchey, *NASA, USA*
Kazuo Iwano, *IBM, Japan*
Mark Jelasity, *University of Bologna, Italy*
Randy Katz, *University of California, Berkeley, USA*
Robert Laddaga, *Massachusetts Institute of Technology, USA*
Ian Marshall, *University of Kent, UK*
Radhika Nagpal, *Harvard University, USA*
George Pavlou, *University of Surrey, UK*
Paul Robertson, *Massachusetts Institute of Technology, USA*
Jerry Rolia, *HP Labs Palo Alto, USA*
Fabrice Saffre, *BT Research & Venturing, UK*
Jürgen Schönwälder, *International University Bremen, Germany*
Karsten Schwan, *Georgia Institute of Technology, USA*
Morris Sloman, *Imperial College London, UK*
Mikhail Smirnov, *Fraunhofer FOKUS, Germany*
Roy Sterritt, *University of Ulster, UK*
John Strassner, *Motorola Labs, USA*
Joe Sventek, *University of Glasgow, UK*
Aad van Moorsel, *Newcastle University, UK*
Maarten van Steen, *Vrije Universiteit Amsterdam, The Netherlands*
Franco Zambonelli, *Università di Modena e Reggio Emilia, Italy*
Zheng Zhang, *Microsoft Research Asia, China*

Reviewers

The task of reviewing the papers submitted to SelfMan 2006 was extremely important. It is therefore a great pleasure to thank the additional reviewers listed below for their constructive and detailed comments. Their efforts were key in assuring the high quality of the workshop.

Sharad Agarwal, *Microsoft Research, USA*
Matt Caesar, *University of California, Berkeley, USA*
Nikolaos Chatzis, *Fraunhofer FOKUS, Germany*
Markus Huebscher, *Imperial College London, UK*
Lutz Mark, *Fraunhofer FOKUS, Germany*
George Porter, *University of California, Berkeley, USA*
Christoph Reichert, *Fraunhofer FOKUS, Germany*
Giuseppe Valetto, *IBM T.J. Watson Research Center, USA*
Tanja Zseby, *Fraunhofer FOKUS, Germany*

Author Index

Lecture Notes in Computer Science 3996

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

'Eds.),
006.
tinelli,
pages.
J. K.
Com-
2006,
J. K.
Com-
2006,
J. K.
Com-
2006,
J. K.
Com-
2006,
J. K.
Com-
2006,
vić, M.
Vision
2006.
(Eds.),
2006.
. West-
twork-
mance
ile and
pages.
hurais-
rity In-
H. Yin
6, Part
H. Yin
6, Part
H. Yin
6, Part
eryavy
s. XIV,

Lecture Notes in Computer Science

For information about Vols. 1–3914

please contact your bookseller or Springer

Vol. 4039: M. Morisio (Ed.), Reuse of Off-the-Shelf Components. XI, 444 pages. 2006.

Vol. 4027: H.L. Larsen, G. Pasi, D. Ortiz-Arroyo, T. Andreasen, H. Christiansen (Eds.), Flexible Query Answering Systems. XVIII, 714 pages. 2006. (Sublibrary LNAI).

Vol. 4024: S. Donatelli, P. S. Thiagarajan (Eds.), Petri Nets and Other Models of Concurrency - ICATPN 2006. XI, 441 pages. 2006.

Vol. 4011: Y. Sure, J. Domingue (Eds.), The Semantic Web: Research and Applications. XIX, 726 pages. 2006.

Vol. 4007: C. Àlvarez, M. Serna (Eds.), Experimental Algorithms. XI, 329 pages. 2006.

Vol. 4006: L.M. Pinho, M. González Harbour (Eds.), Reliable Software Technology – Ada-Europe 2006. XII, 241 pages. 2006.

Vol. 4004: S. Vaudenay (Ed.), Advances in Cryptology - EUROCRYPT 2006. XIV, 613 pages. 2006.

Vol. 4003: Y. Koucheryavy, J. Harju, V.B. Iversen (Eds.), Next Generation Teletraffic and Wired/Wireless Advanced Networking. XVI, 582 pages. 2006.

Vol. 4001: E. Dubois, K. Pohl (Eds.), Advanced Information Systems Engineering. XVI, 560 pages. 2006.

Vol. 3999: C. Kop, G. Fliedl, H.C. Mayr, E. Métais (Eds.), Natural Language Processing and Information Systems. XIII, 227 pages. 2006.

Vol. 3998: T. Calamoneri, I. Finocchi, G.F. Italiano (Eds.), Algorithms and Complexity. XII, 394 pages. 2006.

Vol. 3997: W. Grieskamp, C. Weise (Eds.), Formal Approaches to Software Testing. XII, 219 pages. 2006.

Vol. 3996: A. Keller, J.-P. Martin-Flatin (Eds.), Self-Managed Networks, Systems, and Services. X, 185 pages. 2006.

Vol. 3995: G. Müller (Ed.), Emerging Trends in Information and Communication Security. XX, 524 pages. 2006.

Vol. 3994: V.N. Alexandrov, G.D. van Albada, P.M.A. Sloot, J. Dongarra (Eds.), Computational Science – ICCS 2006, Part IV. XXXV, 1096 pages. 2006.

Vol. 3993: V.N. Alexandrov, G.D. van Albada, P.M.A. Sloot, J. Dongarra (Eds.), Computational Science – ICCS 2006, Part III. XXXVI, 1136 pages. 2006.

Vol. 3992: V.N. Alexandrov, G.D. van Albada, P.M.A. Sloot, J. Dongarra (Eds.), Computational Science – ICCS 2006, Part II. XXXV, 1122 pages. 2006.

Vol. 3991: V.N. Alexandrov, G.D. van Albada, P.M.A. Sloot, J. Dongarra (Eds.), Computational Science – ICCS 2006, Part I. LXXXI, 1096 pages. 2006.

Vol. 3990: J. C. Beck, B.M. Smith (Eds.), Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems. X, 301 pages. 2006.

Vol. 3987: M. Hazas, J. Krumm, T. Strang (Eds.), Location- and Context-Awareness. X, 289 pages. 2006.

Vol. 3986: K. Stølen, W.H. Winsborough, F. Martinelli, F. Massacci (Eds.), Trust Management. XIV, 474 pages. 2006.

Vol. 3984: M. Gavrilova, O. Gervasi, V. Kumar, C.J. K. Tan, D. Taniar, A. Laganà, Y. Mun, H. Choo (Eds.), Computational Science and Its Applications - ICCSA 2006, Part V. XXV, 1045 pages. 2006.

Vol. 3983: M. Gavrilova, O. Gervasi, V. Kumar, C.J. K. Tan, D. Taniar, A. Laganà, Y. Mun, H. Choo (Eds.), Computational Science and Its Applications - ICCSA 2006, Part IV. XXVI, 1191 pages. 2006.

Vol. 3982: M. Gavrilova, O. Gervasi, V. Kumar, C.J. K. Tan, D. Taniar, A. Laganà, Y. Mun, H. Choo (Eds.), Computational Science and Its Applications - ICCSA 2006, Part III. XXV, 1243 pages. 2006.

Vol. 3981: M. Gavrilova, O. Gervasi, V. Kumar, C.J. K. Tan, D. Taniar, A. Laganà, Y. Mun, H. Choo (Eds.), Computational Science and Its Applications - ICCSA 2006, Part II. XXVI, 1255 pages. 2006.

Vol. 3980: M. Gavrilova, O. Gervasi, V. Kumar, C.J. K. Tan, D. Taniar, A. Laganà, Y. Mun, H. Choo (Eds.), Computational Science and Its Applications - ICCSA 2006, Part I. LXXV, 1199 pages. 2006.

Vol. 3979: T.S. Huang, N. Sebe, M.S. Lew, V. Pavlović, M. Kölsch, A. Galata, B. Kisačanin (Eds.), Computer Vision in Human-Computer Interaction. XII, 121 pages. 2006.

Vol. 3978: B. Hnich, M. Carlsson, F. Fages, F. Rossi (Eds.), Recent Advances in Constraints. VIII, 179 pages. 2006. (Sublibrary LNAI).

Vol. 3976: F. Boavida, T. Plagemann, B. Stiller, C. Westphal, E. Monteiro (Eds.), Networking 2006. Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications Systems. XXVI, 1276 pages. 2006.

Vol. 3975: S. Mehrotra, D.D. Zeng, H. Chen, B. Thuraisingham, F.-Y. Wang (Eds.), Intelligence and Security Informatics. XXII, 772 pages. 2006.

Vol. 3973: J. Wang, Z. Yi, J.M. Zurada, B.-L. Lu, H. Yin (Eds.), Advances in Neural Networks - ISNN 2006, Part III. XXIX, 1402 pages. 2006.

Vol. 3972: J. Wang, Z. Yi, J.M. Zurada, B.-L. Lu, H. Yin (Eds.), Advances in Neural Networks - ISNN 2006, Part II. XXVII, 1444 pages. 2006.

Vol. 3971: J. Wang, Z. Yi, J.M. Zurada, B.-L. Lu, H. Yin (Eds.), Advances in Neural Networks - ISNN 2006, Part I. LXVII, 1442 pages. 2006.

Vol. 3970: T. Braun, G. Carle, S. Fahmy, Y. Koucheryavy (Eds.), Wired/Wireless Internet Communications. XIV, 350 pages. 2006.

Vol. 3968: K.P. Fishkin, B. Schiele, P. Nixon, A. Quigley (Eds.), Pervasive Computing. XV, 402 pages. 2006.

Vol. 3967: D. Grigoriev, J. Harrison, E.A. Hirsch (Eds.), Computer Science – Theory and Applications. XVI, 684 pages. 2006.

Vol. 3966: Q. Wang, D. Pfahl, D.M. Raffo, P. Wernick (Eds.), Software Process Change. XIV, 356 pages. 2006.

Vol. 3965: M. Bernardo, A. Cimatti (Eds.), Formal Methods for Hardware Verification. VII, 243 pages. 2006.

Vol. 3964: M. Ü. Uyar, A.Y. Duale, M.A. Fecko (Eds.), Testing of Communicating Systems. XI, 373 pages. 2006.

Vol. 3963: O. Dikenelli, M.-P. Gleizes, A. Ricci (Eds.), Engineering Societies in the Agents World VI. X, 303 pages. 2006. (Sublibrary LNAI).

Vol. 3962: W. IJsselsteijn, Y. de Kort, C. Midden, B. Eggen, E. van den Hoven (Eds.), Persuasive Technology. XII, 216 pages. 2006.

Vol. 3960: R. Vieira, P. Quaresma, M.d.G.V. Nunes, N.J. Mamede, C. Oliveira, M.C. Dias (Eds.), Computational Processing of the Portuguese Language. XII, 274 pages. 2006. (Sublibrary LNAI).

Vol. 3959: J.-Y. Cai, S. B. Cooper, A. Li (Eds.), Theory and Applications of Models of Computation. XV, 794 pages. 2006.

Vol. 3958: M. Yung, Y. Dodis, A. Kiayias, T. Malkin (Eds.), Public Key Cryptography - PKC 2006. XIV, 543 pages. 2006.

Vol. 3956: G. Barthe, B. Grégoire, M. Huisman, J.-L. Lanet (Eds.), Construction and Analysis of Safe, Secure, and Interoperable Smart Devices. IX, 175 pages. 2006.

Vol. 3955: G. Antoniou, G. Potamias, C. Spyropoulos, D. Plexousakis (Eds.), Advances in Artificial Intelligence. XVII, 611 pages. 2006. (Sublibrary LNAI).

Vol. 3954: A. Leonardis, H. Bischof, A. Pinz (Eds.), Computer Vision – ECCV 2006, Part IV. XVII, 613 pages. 2006.

Vol. 3953: A. Leonardis, H. Bischof, A. Pinz (Eds.), Computer Vision – ECCV 2006, Part III. XVII, 649 pages. 2006.

Vol. 3952: A. Leonardis, H. Bischof, A. Pinz (Eds.), Computer Vision – ECCV 2006, Part II. XVII, 661 pages. 2006.

Vol. 3951: A. Leonardis, H. Bischof, A. Pinz (Eds.), Computer Vision – ECCV 2006, Part I. XXXV, 639 pages. 2006.

Vol. 3950: J.P. Müller, F. Zambonelli (Eds.), Agent-Oriented Software Engineering VI. XVI, 249 pages. 2006.

Vol. 3947: Y.-C. Chung, J.E. Moreira (Eds.), Advances in Grid and Pervasive Computing. XXI, 667 pages. 2006.

Vol. 3946: T.R. Roth-Berghofer, S. Schulz, D.B. Leake (Eds.), Modeling and Retrieval of Context. XI, 149 pages. 2006. (Sublibrary LNAI).

Vol. 3945: M. Hagiya, P. Wadler (Eds.), Functional and Logic Programming. X, 295 pages. 2006.

Vol. 3944: J. Quiñonero-Candela, I. Dagan, B. Magnini, F. d'Alché-Buc (Eds.), Machine Learning Challenges. XIII, 462 pages. 2006. (Sublibrary LNAI).

Vol. 3943: N. Guelfi, A. Savidis (Eds.), Rapid Integration of Software Engineering Techniques. X, 289 pages. 2006.

Vol. 3942: Z. Pan, R. Aylett, H. Diener, X. Jin, S. Göbel, L. Li (Eds.), Technologies for E-Learning and Digital Entertainment. XXV, 1396 pages. 2006.

Vol. 3941: S.W. Gilroy, M.D. Harrison (Eds.), Interactive Systems. XI, 267 pages. 2006.

Vol. 3940: C. Saunders, M. Grobelnik, S. Gunn, J. Shawe-Taylor (Eds.), Subspace, Latent Structure and Feature Selection. X, 209 pages. 2006.

Vol. 3939: C. Priami, L. Cardelli, S. Emmott (Eds.), Transactions on Computational Systems Biology IV. VII, 141 pages. 2006. (Sublibrary LNBI).

Vol. 3936: M. Lalmas, A. MacFarlane, S. Rüger, A. Tombros, T. Tsikrika, A. Yavlinsky (Eds.), Advances in Information Retrieval. XIX, 584 pages. 2006.

Vol. 3935: D. Won, S. Kim (Eds.), Information Security and Cryptology - ICISC 2005. XIV, 458 pages. 2006.

Vol. 3934: J.A. Clark, R.F. Paige, F.A. C. Polack, P.J. Brooke (Eds.), Security in Pervasive Computing. X, 243 pages. 2006.

Vol. 3933: F. Bonchi, J.-F. Boulicaut (Eds.), Knowledge Discovery in Inductive Databases. VIII, 251 pages. 2006.

Vol. 3931: B. Apolloni, M. Marinaro, G. Nicosia, R. Tagliaferri (Eds.), Neural Nets. XIII, 370 pages. 2006.

Vol. 3930: D.S. Yeung, Z.-Q. Liu, X.-Z. Wang, H. Yan (Eds.), Advances in Machine Learning and Cybernetics. XXI, 1110 pages. 2006. (Sublibrary LNAI).

Vol. 3929: W. MacCaull, M. Winter, I. Düntsch (Eds.), Relational Methods in Computer Science. VIII, 263 pages. 2006.

Vol. 3928: J. Domingo-Ferrer, J. Posegga, D. Schreckling (Eds.), Smart Card Research and Advanced Applications. XI, 359 pages. 2006.

Vol. 3927: J. Hespanha, A. Tiwari (Eds.), Hybrid Systems: Computation and Control. XII, 584 pages. 2006.

Vol. 3925: A. Valmari (Ed.), Model Checking Software. X, 307 pages. 2006.

Vol. 3924: P. Sestoft (Ed.), Programming Languages and Systems. XII, 343 pages. 2006.

Vol. 3923: A. Mycroft, A. Zeller (Eds.), Compiler Construction. XIII, 277 pages. 2006.

Vol. 3922: L. Baresi, R. Heckel (Eds.), Fundamental Approaches to Software Engineering. XIII, 427 pages. 2006.

Vol. 3921: L. Aceto, A. Ingólfsdóttir (Eds.), Foundations of Software Science and Computation Structures. XV, 447 pages. 2006.

Vol. 3920: H. Hermanns, J. Palsberg (Eds.), Tools and Algorithms for the Construction and Analysis of Systems. XIV, 506 pages. 2006.

Vol. 3918: W.K. Ng, M. Kitsuregawa, J. Li, K. Chang (Eds.), Advances in Knowledge Discovery and Data Mining. XXIV, 879 pages. 2006. (Sublibrary LNAI).

Vol. 3917: H. Chen, F.-Y. Wang, C.C. Yang, D. Zeng, M. Chau, K. Chang (Eds.), Intelligence and Security Informatics. XII, 186 pages. 2006.

Vol. 3916: J. Li, Q. Yang, A.-H. Tan (Eds.), Data Mining for Biomedical Applications. VIII, 155 pages. 2006. (Sublibrary LNBI).

Vol. 3915: R. Nayak, M.J. Zaki (Eds.), Knowledge Discovery from XML Documents. VIII, 105 pages. 2006.

Table of Contents

Networking

Work-in-Progress Papers

Implementation and Evaluation of a Middleware for Self-Organizing Decentralized Web Services

Constantin Adam and Rolf Stadler

KTH Royal Institute of Technology, Laboratory for Communication Networks, Stockholm, Sweden
{ctin, stadler}@kth.se

Abstract. We present the implementation of Chameleon, a peer-to-peer middleware for self-organizing web services, and we provide evaluation results from a test bed. The novel aspect of Chameleon is that key functions, including resource allocation, are decentralized, which facilitates scalability and robustness of the overall system. Chameleon is implemented in Java on the Tomcat web server environment. The implementation is non-intrusive in the sense that it does not require code modifications in Tomcat or in the underlying operating system. We evaluate the system by running the TPC-W benchmark. We show that the middleware dynamically and effectively reconfigures in response to changes in load patterns and server failures, while enforcing operating policies, namely, QoS objectives and service differentiation under overload.

1 Introduction

Large-scale web services, such as on-line shopping, auctioning, and webcasting, rapidly expand in geographical coverage and number of users. Current systems that support such services, including commercial solutions (IBM WebSphere, BEA WebLogic) and research prototypes (Ninja [1], Neptune [2]), are based on centralized designs, which limit their scalability in terms of efficient operation, low configuration complexity and robustness.

To address these limitations, we have developed Chameleon, a decentralized middleware design that dynamically allocates resources to multiple services classes inside a global server cluster. Chameleon has three features characteristic of peer-to-peer systems. First, the server cluster consists of a set of functionally identical nodes, which simplifies the design and configuration. Second, the design is decentralized and the cluster dynamically re-organizes after changes or failures. Third, each node maintains only a partial view of the system, which facilitates scalability.

Chameleon supports QoS objectives for each service class. Its distinctive features are the use of an epidemic protocol [3] to disseminate state and control information, as well as the decentralized evaluation of utility functions to control resource partitioning among service classes. In [4], we have presented our design in detail and evaluated it through extensive simulations. This work complements simulation studies we have conducted earlier to validate the scalability

A. Keller and J.-P. Martin-Flatin (Eds.): SelfMan 2006, LNCS 3996, pp. 1–14, 2006.

of the Chameleon design. The implementation presented in this paper further validates the design and the system model we have used in our simulations.

The rest of this paper is structured as follows: Section 2 reviews the design of our peer-to-peer middleware. Section 3 describes the implementation of the design on an experimental test bed. Section 4 evaluates the implementation. Section 5 reviews related work. Finally, Section 6 contains additional observations and outlines future work.

2 Overview of the Chameleon Middleware Design

We have developed our design for large-scale systems. A typical deployment scenario that contains several data centers and many entry points is shown in Fig. 1. We assume that the data centers are connected through high-capacity links and the networking delays are much smaller than the processing delays. We refer to the collection of servers in these centers as the *cluster*.

Note that the design in this paper covers the tier of the application servers, but not the database tier. Specifically, we do not provide an approach to scale the database tier, which is an active area of current research. This is also reflected in the evaluation of the design, where we use database operations in browsing mode for read-only data, which do not have transaction requirements.

Service requests enter the cluster through the entry points, which function in our design as layer 7 switches. An entry point associates an incoming request with a service class and directs it to a server assigned to that class. An epidemic protocol runs in the cluster to provide the entry points with information about servers assigned to each service class.

We associate two performance targets with each service: the maximum response time (defined per individual request), and the maximum drop rate (de-

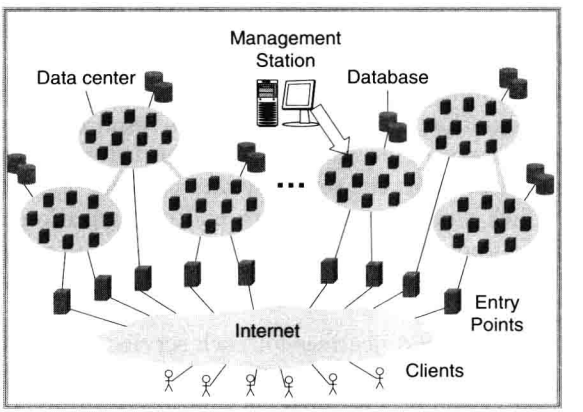

Fig. 1. We have developed a design for large-scale, self-configuring server clusters. The design includes the servers in data centers, the entry points and a management station.

fined over all the requests for a specific service). In this paper, we use the terms service and service class interchangeably.

Upon receiving a request, a server schedules it for execution, using a FIFO policy. In case of overload, or high CPU utilization, the server redirects the request to one of its neighbors that runs the same service. A request that cannot be processed within the maximum response time objective is dropped by the system.

2.1 Cluster Services and Utility Functions

The objective of the system is to maximize a *cluster utility function* that measures how well it meets the performance targets for the offered services. We define the cluster utility function as the sum of the service utility functions. A service utility function specifies the rewards for meeting and the penalties for missing the performance targets for a given service.

Let ρ denote the maximum allowed drop rate and r represent the experienced drop rate. Meeting or exceeding the QoS objectives for a service yields a reward:

$$U^+ = \alpha(\rho - r), \text{ if } r \leq \rho.$$

The violation of the QoS objectives for a service results in a penalty. In order to avoid starvation of a service, the penalty increases exponentially past the point $r^+ = \rho + \alpha^{1/(\beta-1)}$:

$$U^- = \begin{cases} -\alpha(r-\rho), & \rho < r \leq r^+ \\ -(r-\rho)^\beta, & r > r^+ \end{cases}$$

The control parameters α and β are positive numbers that define the shape of the graph and determine the relative importance of a service.

As the cluster utility is the sum of service utilities, the system will attempt to maximize the cluster utility by allocating cluster resources to services in such a way that the performance targets of services with higher values for α and β are more likely to be met than those of services with lower values. This enables service differentiation in case of overload. Note that the behavior of a system depends on the specific choice of the cluster utility function. For example, a system for which the cluster utility function is defined as the minimum of the service utility functions will attempt to provide fair allocation to all cluster services.

2.2 Decentralized Control Mechanisms

Three distributed mechanisms, shown in Fig. 2, form the core of our design. Topology construction, based on Newscast, an epidemic protocol, organizes the cluster nodes into dynamic overlays, which are used to disseminate state and control information in a scalable and robust manner. Request routing directs service requests towards available resources, subject to response time constraints. Service selection dynamically partitions the cluster resources between services in response to external events, by periodically maximizing utility functions. These mechanisms run independently and asynchronously on each server.

The *topology construction* mechanism organizes the servers of a cluster with s service classes into $s + 1$ logical networks: one system overlay and s service

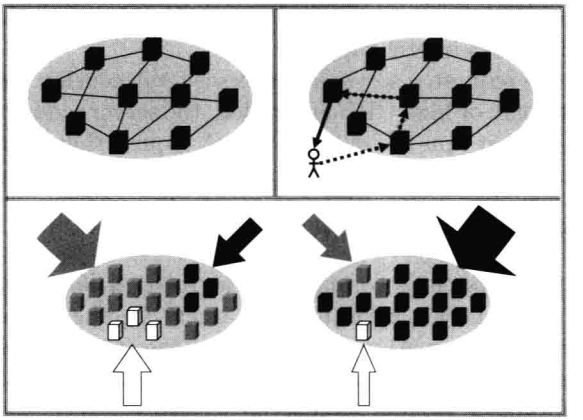

Fig. 2. Three decentralized mechanisms control the system behavior: (a) topology construction, (b) request routing, and (c) service selection

overlays, one per service class. Each server runs, at any time, a single service S and is a member of two logical networks, the system overlay and the overlay for the service S. For each of these networks, a server has a neighbor table, with the structure shown in Tables 1 and 2. As we will explain below, the request routing mechanism uses the information in the service table, while the service selection mechanism uses the information in the system table.

Table 1. Structure of the service neighborhood table

ID	Timestamp	Utilization	Processed Requests	Dropped Requests

Table 2. Structure of the system neighborhood table

ID	Timestamp	Service	Processed Requests	Dropped Requests

The *request routing* mechanism directs incoming requests along the service overlays toward available resources, subject to the maximum response time objective. It is invoked when a server is overloaded and does not have sufficient local resources to process a request in time. If the server has light neighbors (i.e. with utilization below a threshold cpu_{max}) in its service neighborhood table, it forwards the request to a randomly selected light neighbor. Otherwise, it forwards the request to a random neighbor. In order to avoid routing loops, each request keeps in its header the addresses of the servers it has visited. If a node cannot process the request and the request has already visited all of its neighbors, the request is dropped.

```
1. Service_id my_service, s, new_service;
2. Hashtable  known_services;
```

```
 3. Double     max_util;
 4. while (true) {
 5.   max_util=estimateNhoodUtility(my_service);
 6.   new_service=my_service;
 7.   for each s in known_services {
 8.     estimateCurrentDropRate(s);
 9.     predictDropRateAfterSwitch(s);
10.   }
11.   for each s in known_services {
12.     U_s= predictNhoodUtilityAfterSwitchTo(s);
13.     if(U_s > max_util) {
14.       max_util=U_s;
15.       new_service=s;
16.     }
17.   }
18.   if(new_service!=my_service) {
19.     if(local_utilization<random(0%..100%)) {
20.       switchService(new_service);
21.       my_service=new_service;
22.     }
23.   }
24.   wait(delta_t);
25. }
```

The *service selection* mechanism partitions the system resources between services. Each server starts the service selection mechanism at a random time and performs an *execution cycle* with period Δt. A server uses the information in its system table (Table 2) to estimate the utility currently produced by its neighborhood (line 5) and compares that to the utility the neighborhood would generate in the event the node would switch to a different service (lines 7-17). The server then probabilistically switches to the service it predicts will yield the highest utility for the neighborhood. The probability for switching increases with decreasing server utilization (line 19). We choose a probabilistic approach, because applying deterministic switching can lead to oscillations, whereby several servers periodically switch between two or more services.

3 Implementation: Integrating Chameleon into the Tomcat Framework

We have implemented our middleware in Java, within the Tomcat environment [6]. We have chosen Tomcat because it is simple and it is widely available as open source. We believe that Chameleon could be integrated as well with other web server environments that support application filters, such as IBM WebSphere.

To let the reader better understand the integration, we provide a short description of Tomcat; Fig. 3 shows its internal architecture. The components of the architecture are called containers. The service container has a set of connectors that handle the communication with clients on a specified port (80 or 8080 for