Miroslaw Malek
Edgar Nett
Neeraj Suri (Eds.)

LNCS 3694

Service Availability

Second International Service Availability Symposium, ISAS 2005
Berlin, Germany, April 2005
Revised Selected Papers

Springer

Miroslaw Malek Edgar Nett
Neeraj Suri (Eds.)

Service Availability

Second International Service Availability Symposium, ISAS 2005
Berlin, Germany, April 25 – 26, 2005
Revised Selected Papers

江苏工业学院图书馆
藏书章

Springer

Volume Editors

Miroslaw Malek
Humboldt-Universität zu Berlin
Institut für Informatik
Rechnerorganisation und Kommunikation
Rudower Chaussee 25, 12489 Berlin, Germany
E-mail: malek@informatik.hu-berlin.de

Edgar Nett
Otto-von-Guericke-Universität Magdeburg
Institut für Verteilte Systeme
Universitätsplatz 2, 39106 Magdeburg, Germany
E-mail: nett@ivs.cs.uni-magdeburg.de

Neeraj Suri
Technical University Darmstadt
Department of Computer Science
Dependable Embedded Systems and Software
Hochschulstr. 10, 64289 Darmstadt, Germany
E-mail: suri@informatik.tu-darmstadt.de

Library of Congress Control Number: 2005932545

CR Subject Classification (1998): C.2, H.4, H.3, I.2.11, D.2, H.5, K.4.4, K.6

ISSN 0302-9743
ISBN-10 3-540-29103-2 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-29103-9 Springer Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

Springer is a part of Springer Science+Business Media

springeronline.com

© Springer-Verlag Berlin Heidelberg 2005
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11560333 06/3142 5 4 3 2 1 0

Lecture Notes in Computer Science 3694

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

Editorial Board

David Hutchison
Lancaster University, UK
Takeo Kanade
Carnegie Mellon University, Pittsburgh, PA, USA
Josef Kittler
University of Surrey, Guildford, UK
Jon M. Kleinberg
Cornell University, Ithaca, NY, USA
Friedemann Mattern
ETH Zurich, Switzerland
John C. Mitchell
Stanford University, CA, USA
Moni Naor
Weizmann Institute of Science, Rehovot, Israel
Oscar Nierstrasz
University of Bern, Switzerland
C. Pandu Rangan
Indian Institute of Technology, Madras, India
Bernhard Steffen
University of Dortmund, Germany
Madhu Sudan
Massachusetts Institute of Technology, MA, USA
Demetri Terzopoulos
New York University, NY, USA
Doug Tygar
University of California, Berkeley, CA, USA
Moshe Y. Vardi
Rice University, Houston, TX, USA
Gerhard Weikum
Max-Planck Institute of Computer Science, Saarbruecken, Germany

Lecture Notes in Computer Science

For information about Vols. 1–3615

please contact your bookseller or Springer

Vol. 3728: V. Paliouras, J. Vounckx, D. Verkest (Eds.), Integrated Circuit and System Design. XV, 753 pages. 2005.

Vol. 3718: V.G. Ganzha, E.W. Mayr, E.V. Vorozhtsov (Eds.), Computer Algebra in Scientific Computing. XII, 502 pages. 2005.

Vol. 3715: E. Dawson, S. Vaudenay (Eds.), Progress in Cryptology – Mycrypt 2005. XI, 329 pages. 2005.

Vol. 3714: H. Obbink, K. Pohl (Eds.), Software Product Lines. XIII, 235 pages. 2005.

Vol. 3713: L. Briand, C. Williams (Eds.), Model Driven Engineering Languages and Systems. XV, 722 pages. 2005.

Vol. 3712: R. Reussner, J. Mayer, J.A. Stafford, S. Overhage, S. Becker, P.J. Schroeder (Eds.), Quality of Software Architectures and Software Quality. XIII, 289 pages. 2005.

Vol. 3711: F. Kishino, Y. Kitamura, H. Kato, N. Nagata (Eds.), Entertainment Computing - ICEC 2005. XXIV, 540 pages. 2005.

Vol. 3710: M. Barni, I. Cox, T. Kalker, H.J. Kim (Eds.), Digital Watermarking. XII, 485 pages. 2005.

Vol. 3708: J. Blanc-Talon, W. Philips, D. Popescu, P. Scheunders (Eds.), Advanced Concepts for Intelligent Vision Systems. XXII, 725 pages. 2005.

Vol. 3706: H. Fuks, S. Lukosch, A.C. Salgado (Eds.), Groupware: Design, Implementation, and Use. XII, 378 pages. 2005.

Vol. 3703: F. Fages, S. Soliman (Eds.), Principles and Practice of Semantic Web Reasoning. VIII, 163 pages. 2005.

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

Vol. 3699: C.S. Calude, M.J. Dinneen, G. Paun, M.J. Pérez-Jiménez, G. Rozenberg (Eds.), Unconventional Computation. XI, 267 pages. 2005.

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

Vol. 3697: W. Duch, J. Kacprzyk, E. Oja, S. Zadrożny (Eds.), Artificial Neural Networks: Formal Models and Their Applications – ICANN 2005, Part II. XXXII, 1045 pages. 2005.

Vol. 3696: W. Duch, J. Kacprzyk, E. Oja, S. Zadrożny (Eds.), Artificial Neural Networks: Biological Inspirations – ICANN 2005, Part I. XXXI, 703 pages. 2005.

Vol. 3695: M.R. Berthold, R. Glen, K. Diederichs, O. Kohlbacher, I. Fischer (Eds.), Computational Life Sciences. XI, 277 pages. 2005. (Subseries LNBI).

Vol. 3694: M. Malek, E. Nett, N. Suri (Eds.), Service Availability. VIII, 213 pages. 2005.

Vol. 3693: A.G. Cohn, D.M. Mark (Eds.), Spatial Information Theory. XII, 493 pages. 2005.

Vol. 3691: A. Gagalowicz, W. Philips (Eds.), Computer Analysis of Images and Patterns. XIX, 865 pages. 2005.

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

Vol. 3687: S. Singh, M. Singh, C. Apte, P. Perner (Eds.), Pattern Recognition and Image Analysis, Part II. XXV, 809 pages. 2005.

Vol. 3686: S. Singh, M. Singh, C. Apte, P. Perner (Eds.), Pattern Recognition and Data Mining, Part I. XXVI, 689 pages. 2005.

Vol. 3685: V. Gorodetsky, I. Kotenko, V. Skormin (Eds.), Computer Network Security. XIV, 480 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. (Subseries LNAI).

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

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

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

Vol. 3679: S.d.C. di Vimercati, P. Syverson, D. Gollmann (Eds.), Computer Security – ESORICS 2005. XI, 509 pages. 2005.

Vol. 3678: A. McLysaght, D.H. Huson (Eds.), Comparative Genomics. VIII, 167 pages. 2005. (Subseries LNBI).

Vol. 3677: J. Dittmann, S. Katzenbeisser, A. Uhl (Eds.), Communications and Multimedia Security. XIII, 360 pages. 2005.

Vol. 3675: Y. Luo (Ed.), Cooperative Design, Visualization, and Engineering. XI, 264 pages. 2005.

Vol. 3674: W. Jonker, M. Petković (Eds.), Secure Data Management. X, 241 pages. 2005.

Vol. 3672: C. Hankin, I. Siveroni (Eds.), Static Analysis. X, 369 pages. 2005.

Vol. 3671: S. Bressan, S. Ceri, E. Hunt, Z.G. Ives, Z. Bellahsène, M. Rys, R. Unland (Eds.), Database and XML Technologies. X, 239 pages. 2005.

此为试读，需要完整PDF请访问：www.ertongbook.com

Vol. 3670: M. Bravetti, L. Kloul, G. Zavattaro (Eds.), Formal Techniques for Computer Systems and Business Processes. XIII, 349 pages. 2005.

Vol. 3666: B.D. Martino, D. Kranzlmüller, J. Dongarra (Eds.), Recent Advances in Parallel Virtual Machine and Message Passing Interface. XVII, 546 pages. 2005.

Vol. 3665: K. S. Candan, A. Celentano (Eds.), Advances in Multimedia Information Systems. X, 221 pages. 2005.

Vol. 3664: C. Türker, M. Agosti, H.-J. Schek (Eds.), Peer-to-Peer, Grid, and Service-Orientation in Digital Library Architectures. X, 261 pages. 2005.

Vol. 3663: W.G. Kropatsch, R. Sablatnig, A. Hanbury (Eds.), Pattern Recognition. XIV, 512 pages. 2005.

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

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

Vol. 3660: M. Beigl, S. Intille, J. Rekimoto, H. Tokuda (Eds.), UbiComp 2005: Ubiquitous Computing. XVII, 394 pages. 2005.

Vol. 3659: J.R. Rao, B. Sunar (Eds.), Cryptographic Hardware and Embedded Systems – CHES 2005. XIV, 458 pages. 2005.

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

Vol. 3655: A. Aldini, R. Gorrieri, F. Martinelli (Eds.), Foundations of Security Analysis and Design III. VII, 273 pages. 2005.

Vol. 3654: S. Jajodia, D. Wijesekera (Eds.), Data and Applications Security XIX. X, 353 pages. 2005.

Vol. 3653: M. Abadi, L. de Alfaro (Eds.), CONCUR 2005 – Concurrency Theory. XIV, 578 pages. 2005.

Vol. 3652: A. Rauber, S. Christodoulakis, A M. Tjoa (Eds.), Research and Advanced Technology for Digital Libraries. XVIII, 545 pages. 2005.

Vol. 3650: J. Zhou, J. Lopez, R.H. Deng, F. Bao (Eds.), Information Security. XII, 516 pages. 2005.

Vol. 3649: W.M. P. van der Aalst, B. Benatallah, F. Casati, F. Curbera (Eds.), Business Process Management. XII, 472 pages. 2005.

Vol. 3648: J.C. Cunha, P.D. Medeiros (Eds.), Euro-Par 2005 Parallel Processing. XXXVI, 1299 pages. 2005.

Vol. 3646: A. F. Famili, J.N. Kok, J.M. Peña, A. Siebes, A. Feelders (Eds.), Advances in Intelligent Data Analysis VI. XIV, 522 pages. 2005.

Vol. 3645: D.-S. Huang, X.-P. Zhang, G.-B. Huang (Eds.), Advances in Intelligent Computing, Part II. XIII, 1010 pages. 2005.

Vol. 3644: D.-S. Huang, X.-P. Zhang, G.-B. Huang (Eds.), Advances in Intelligent Computing, Part I. XXVII, 1101 pages. 2005.

Vol. 3642: D. Ślezak, 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. (Subseries LNAI).

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

Vol. 3639: P. Godefroid (Ed.), Model Checking Software. XI, 289 pages. 2005.

Vol. 3638: A. Butz, B. Fisher, A. Krüger, P. Olivier (Eds.), Smart Graphics. XI, 269 pages. 2005.

Vol. 3637: J. M. Moreno, J. Madrenas, J. Cosp (Eds.), Evolvable Systems: From Biology to Hardware. XI, 227 pages. 2005.

Vol. 3636: M.J. Blesa, C. Blum, A. Roli, M. Sampels (Eds.), Hybrid Metaheuristics. XII, 155 pages. 2005.

Vol. 3634: L. Ong (Ed.), Computer Science Logic. XI, 567 pages. 2005.

Vol. 3633: C. Bauzer Medeiros, M. Egenhofer, E. Bertino (Eds.), Advances in Spatial and Temporal Databases. XIII, 433 pages. 2005.

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

Vol. 3631: J. Eder, H.-M. Haav, A. Kalja, J. Penjam (Eds.), Advances in Databases and Information Systems. XIII, 393 pages. 2005.

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

Vol. 3629: J.L. Fiadeiro, N. Harman, M. Roggenbach, J. Rutten (Eds.), Algebra and Coalgebra in Computer Science. XI, 457 pages. 2005.

Vol. 3628: T. Gschwind, U. Aßmann, O. Nierstrasz (Eds.), Software Composition. X, 199 pages. 2005.

Vol. 3627: C. Jacob, M.L. Pilat, P.J. Bentley, J. Timmis (Eds.), Artificial Immune Systems. XII, 500 pages. 2005.

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

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

Vol. 3624: C. Chekuri, K. Jansen, J.D. P. Rolim, L. Trevisan (Eds.), Approximation, Randomization and Combinatorial Optimization. XI, 495 pages. 2005.

Vol. 3623: M. Liśkiewicz, R. Reischuk (Eds.), Fundamentals of Computation Theory. XV, 576 pages. 2005.

Vol. 3622: V. Vene, T. Uustalu (Eds.), Advanced Functional Programming. IX, 359 pages. 2005.

Vol. 3621: V. Shoup (Ed.), Advances in Cryptology – CRYPTO 2005. XI, 568 pages. 2005.

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

Vol. 3619: X. Lu, W. Zhao (Eds.), Networking and Mobile Computing. XXIV, 1299 pages. 2005.

Vol. 3618: J. Jedrzejowicz, A. Szepietowski (Eds.), Mathematical Foundations of Computer Science 2005. XVI, 814 pages. 2005.

Vol. 3617: F. Roli, S. Vitulano (Eds.), Image Analysis and Processing – ICIAP 2005. XXIV, 1219 pages. 2005.

General and Program Chairs' Message

The 2nd International Service Availability Symposium (ISAS 2005) provided a unique forum for academia and industry researchers who focus not only on developing next generation solutions but also on standards for today's market. Given the pervasive interweaving of computing devices, increasingly it is "services" rather than "systems" that warrant our attention. As services emerge as the primary vehicle for information acquisition, processing and delivery, the demands for dependability become of primary concern. Needless to add, the expectations from users' with respect to trust and reliance of such systems will only continue to grow.

As computers already pervade almost all walks of our lives, significantly increased interest in dependable computing should not be a surprise as the industry leaders and main computer companies are searching for innovative ways of enhancing the dependability of systems that are increasingly more complex and networked. With the paradigm shift where "everything" may become a service, it is not an option but an imperative to address the questions of service availability. From humble beginnings of dealing with types and formats, later with tasks and processes, then with objects and components, we have arrived to service and peer-to-peer computing. Over 8.5 billion processors are produced each year and 98.5% end up in geographically distributed and interconnected embedded systems. The challenge is to design services and systems that are highly available, reliable and secure. As the number of 7×24 applications continuously increases this is an ambitious challenge that will have to be met. Service availability cannot be compromised. It will have to be delivered as the economic and social impacts of unreliable, incorrect services might range from minor inconveniences to losses of human lives and unpredictable costs.

This year's ISAS represented an excellent mix of academic and industrial contributions as well as participation.

The eight sessions featured truly distinguished academics and industrial leaders as well as some new researchers in the field. We had an outstanding Keynote Speaker Prof. Hermann Kopetz from TU Vienna who is a pioneer in the field of dependable real-time computing, actively contributing to the field for almost 30 years. A distinguished panel featuring representatives from academia and industry, two invited sessions, and regular papers that were subject to a rigorous review process constituted the overall ISAS program. Each paper was reviewed by at least three Program Committee members. We would wholeheartedly like to thank our PC members for their guidance and diligent reviewing. Our thanks go to Prof. Edgar Nett, Nikola Milanovic and Christine Henze for editing the proceedings. Nikola and Christine also helped together with Sabine Becker and Steffen Tschirpke of Humboldt University Berlin, Susan Morgner and Dr. Christine Titel from Congressa GmbH the organization and we do appreciate it very much! Last but not least we would like to thank Manfred Reitenspieß who has been the guiding force behind ISAS and the Service Availability Forum.

I hope that the attendees enjoyed the final program, enjoyed the presentations, got involved in the discussions, struck up new friendships, and got inspiration for contributions to the next year's symposium which will be hosted by Kimmo Raatikainen, University of Helsinki and Francis Tam of Nokia in Helsinki during May 15–16, 2006.

Miroslaw Malek
Humboldt Universität Berlin
Institut für Informatik
malek@informatik.hu-berlin.de
ISAS 2005 General Chair

Neeraj Suri
Technische Universität Darmstadt
Institut fuȓ Informatik
suri@informatik.tu-darmstadt.de
ISAS 2005 Program Chair

Table of Contents

TTA Supported Service Availability

H. Kopetz

Institut für Technische Informatik TU Wien,
A 1040 Wien, Treitlstrasse 3
hk@vmars.tuwien.ac.at

Abstract. The Time-Triggered Architecture (TTA) is a distributed architecture for high-dependability real-time applications. In this paper the mechanisms that guarantee a high availability of TTA services are presented. The paper starts with a deliberation on the fault-hypothesis of the TTA and discusses the partitioning of a TTA system into independent fault-containment regions, their failure modes and their failure frequencies. In the second part the structure of the TTA is explained and the mechanisms that handle the specified faults are outlined. The role of the TTA-inherent sparse time base for the consistent ordering of messages and the solution of the simultaneity problem is explained. Finally, the third part speculates on the vision of a highly integrated TTA-giga-chip that acts as a self-contained TTA node and could be implemented on a single silicon die.

1 Introduction

The Time-Triggered Architecture (TTA) [1] is an *integrated distributed* computer architecture, designed to provide a continuous *timely* services with an MBTF of better than 10^9 hours in the presence of component failures, *provided that* the occurrences of component failures are in agreement with the stated fault hypothesis. The TTA is intended for applications that require utmost availability even in the presence of a fault in any of its components: examples of such applications are the control of a nuclear power plant, the flight control system of an airplane or a computer-based brake control system within an automobile that does not contain a mechanical backup.

Such a high reliability can only be achieved by the provision of redundancy in the hardware, since the observed component (chip) failure rates are orders of magnitudes lower [2] than the desired system reliability. Every redundancy scheme is based on a number of assumptions--*the fault hypothesis*--about the types and frequency of faults that the system is supposed to handle. In case that all fault-handling mechanisms are perfect and cover all scenarios that are listed in the fault-hypothesis, the probability of system failure is reduced to the *assumption coverage*[3], i.e., the probability that the assumptions made in the fault hypothesis are met by reality. The fault hypothesis of any fault-tolerant system is a critical document in the design process. The fault hypothesis of the TTA is discussed in more detail in Section two.

One common technique to implement fault-masking by redundancy is called *triple-modular redundancy* (TMR). In a TMR system fault-tolerant units (FTUs) are formed by placing three synchronized deterministic replicas of every critical component into a new distributed unit--the FTU. An incoming message is distributed

© Springer-Verlag Berlin Heidelberg 2005

此为试读，需要完整PDF请访问：www.ertongbook.com

to all three units of the FTU and the result message (and the internal state) is outputted to a voter that makes a majority decision based on at least two identical results. If one of the components of FTU produces no result or a result that is different from the result of the other two components, this component is considered to have failed. TMR structures will only succeed if the redundant components fail *independently*, i.e. if there is no correlation between the failures of components that form a fault-tolerant unit. Correlated failures can occur because of external causes (a single external event, e.g., a lightning stroke, causes the failure of more than one component) or by error propagation, i.e. an erroneous component sends a faulty message to an up to that instant correctly operating component and thus corrupts the internal state of this component. The issues of fault isolation and error propagation in the TTA are covered in Sections three and four.

Finally in Section five and six we speculate about the future hardware implementation of the TTA. Considering the tremendous advances in the field of semiconductor technology, which is expected to give us billion-transistor giga chips (system-on-a-chip: SoC) by the end of this decade, we outline the structure of a generic TTA-SoC that can be used in many different application domains.

2 Fault Hypothesis of the TTA

In the following paragraphs we discuss the fault hypothesis of the TTA with respect to hardware faults. We assume that the hardware design and the basic fault-handling mechanisms are free of design faults.

2.1 Fault-Containment Regions

The first step in the specification of a fault-hypothesis is concerned with the establishment of a the fault-containment regions (FCR), i.e. the *units of failure*. An FCR is a subsystem that is considered to fail independently from any other FCR. If we must tolerate the physical destruction of a hardware component (e.g., in an accident), then different FCRs must be in different physical locations, i.e. the computer system must be distributed. In the TTA we assume that every node of the distributed system forms an FCR.

2.2 Failure Modes

In the next step we must specify the *critical failure modes* of FCRs. Any restriction of the tolerated failure modes must be considered as an *additional assumption* that has a negative effect on the assumption coverage. In the optimal case no restriction of the failure modes are made, i.e. a failing component can manifest an arbitrary behavior.

We consider a failure mode of an FCR as *critical*, if it impacts the remaining correct nodes of the distributed system in such a way that the functionality or the consistency of the distributed computing base among the nodes that are outside the affected FCR is lost. We focus on a single fault during a fault-recovery interval Δd. After the recovery interval Δd the architecture has recovered from the consequences of this fault and can tolerate a further fault (provided enough resources remain operational).

We define a set of nodes as *Δd-consistent* if *Δd* time units after the occurrence of failure all remaining correct nodes have the same view about this failure event.

In the TTA we have identified the following critical failure modes of an FCR that must be addressed at the level of the architecture:

(i) Crash/Omission (CO) failures
(ii) Babbling idiot failures
(iii) Slightly-off-specification (SOS) failures
(iv) Masquerading failures
(v) Massive transient disturbances

In the analysis of failure mode (i) to (iv) we assume that a fault impacts a single FCR only. Failure mode (v), special case that affects more than one FCR, is explained at the end of this Section.

Crash/Omission Failures: A widely accepted fault-model in a distributed system assumes a fault that manifests itself as either a crash failure of a node or an omission failure of the communication channel (CO failure). CO failures are the most common failures in distributed system--close to 99% of the failures are of the CO type[4]. According to this fault model a node either operates correctly or crashes. The communication system either transports a message correctly, produces a detectably corrupted message, or fails to transport a message. Most of the available communication protocols, such as for example TCP/IP, are designed to detect and, if possible, to correct CO failures. The consequence of a CO failures is a loss of consistency of the distributed computing base. In a point-to-point communication system an acknowledgement service is provided to detect CO failures. In multi-cast communication system, such as the TTA, a membership service can is available to detect and identify CO failures. Another mechanism for CO failure detection is the acknowledgement mechanism of the CAN protocol[5].

In a multicast environment it is important to distinguish between an omission failure at the sender and an omission failure at one of the receivers. If the sender learns promptly about a local omission failure it can often undo the state-change assumed to have taken place by rolling back to the state before the send operation. In case of an omission failure at one of the receivers, such a rollback is not possible.

Prompt CO failure detection and diagnosis at the architecture level is important in order to inform the application that consistency has been lost, and which unit is responsible for the loss of consistency. The application can then decide what corrective action must be taken.

Babbling idiot failures: A babbling idiot failure of an FCR occurs, if the FCR starts sending *untimely messages*. In a multicast time-triggered communication topology that contains a broadcast channel such a babbling FCR can interfere with the communication of the correct nodes. If an FCR exhibits permanent babbling-idiot failures on both channels (this is in principle possible, since both channels are in the same FCR) any further communication among the correct nodes becomes impossible. The TTA detects and handles babbling-idiot failures of FCRs by the guardians in the communication system. The guardian will only open the sending channels during the *a priori* known time-interval that has been allocated to a node.

Slightly-off-Specification (SOS) Failures: Slightly-off-specification failures are an important special case of Byzantine failures. They can occur at the interface between the analog and the digital world. Assume the situation as depicted in Figure 1. The specification requires that every correct node must accept analog input signals if they are within a specified receive window of a parameter (e.g., timing, frequency, or voltage). Every individual node will have a wider actual receive window than the one specified in order to ensure that even if there are slight variations in manufacturing it can accept all correct input signals as required by the specification. These actual receive windows will be slightly different for the individual nodes, as shown in Fig 4. If an erroneous FCR produces an output signal (in time or value) slightly outside the specified window, some nodes will correctly receive this signal, while others might fail to receive this signal. Such a scenario will result in an inconsistent state of the distributed system.

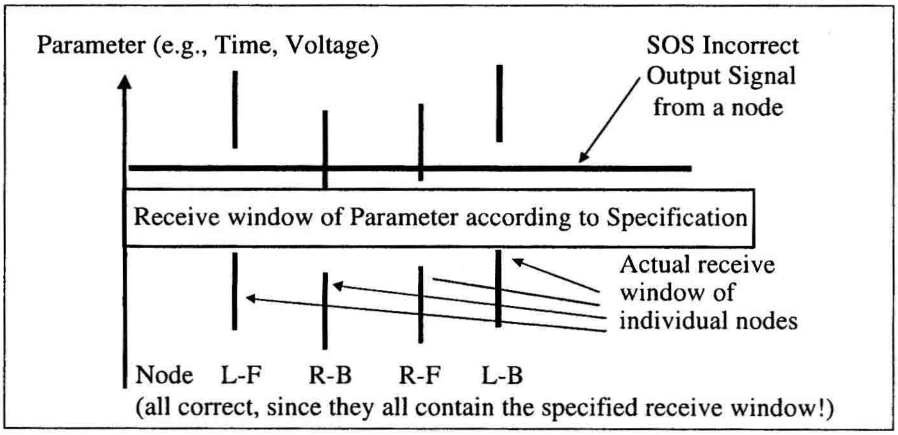
Parameter (e.g., Time, Voltage)
SOS Incorrect Output Signal from a node
Receive window of Parameter according to Specification
Actual receive window of individual nodes
Node L-F R-B R-F L-B
(all correct, since they all contain the specified receive window!)

Fig. 1. Slightly off Specification (SOS) failure

Example: Consider a brake-by-wire system where four receiving nodes are at the four wheels of a car (L-F: left front, R-B: right back, R-F: right front, L-B: Left back). In this example an SOS output failure of the "brake master" will cause confusion in the distributed system. According to this example, the *L-Front* and the *L-Back* node will accept the SOS message, while the *R-Back* and *R-Front* node will discard this message. In a brake-by-wire system, such an inconsistency can become safety relevant.

In the TTA we must address the following three types of SOS failures:

(i) SOS value failure
(ii) SOS coding failures
(iii) SOS send-instant failures.

An *SOS value failure* occurs, if the signal level of the outgoing message is SOS faulty. Some receivers may still correctly decode such an SOS faulty signal, while others may not be able to decode this signal. Since both outgoing channels of an FCR depend on the same power supply, the probability that SOS value failures on both channels are correlated.

An *SOS coding failure* occurs, if the bit stream from the sender is at the border of the coding specification, e.g., the frequency is SOS faulty. Since both channels are driven by the same oscillator the probabilities for the occurrence of an SOS coding failure on both channels are not independent.

An *SOS send-instant failure* occurs, if the send-instant of a message transmission (see Section 3.1) is SOS faulty. A message that is SOS send-instant faulty may be accepted by some nodes and rejected by others. Again, SOS send-instant failures on the two channels are correlated.

Masquerading Failures: A masquerading failure occurs if an erroneous node assumes the identity of another node and causes harm to the system. Systems that rely on names stored in a message to identify the transported message and the information contained therein are vulnerable to masquerading failures. It opens the possibility that a single faulty node can masquerade other nodes, without the receivers having a chance to detect the fault. For example, if a bit in the name of a message to-be-sent that is stored in the sending node is incorrect, this message could, after arrival at its destination, overwrite correct messages at correct receivers. This problem is discussed at some length in the safety-critical SafeBus protocol [6] p.36: *Any protocol that includes a destination memory address in a message is a space-partitioning problem.*

Massive Transient Disturbances: Another important fault class in a distributed embedded system, particularly in the automotive domain, is concerned with massive transient disturbances, e.g., those caused by electromagnetic emission (EMI). A massive transient disturbance can cause the temporary loss of communication among otherwise correct nodes that reside in different FCRs or cause state-corruptions within more than one node. Based on available failure data [2] it is reasonable to assume that the multiple correlated faults produced by a massive transient disturbance are transient, i.e. that the hardware is not faulted by the massive transient disturbance. In such a situation the architecture can provide the service of prompt error detection in order that the nodes may take some local corrective action until the transient disturbance has disappeared and the communication service and the consistency of the nodes is reestablished by a fast restart. For example, [7] report that in an automotive environment a temporary loss of communication of up to 50 msec can be tolerated by freezing the actuators in the positions that were taken before the onset of the transient disturbance. The probability of occurrence of transient disturbances must be reduced by proper quality engineering, e.g., by shielding the cables or installing fiber optics instead of copper. In a safety-critical distributed system massive transient disturbances must be rare events. From the point of view of the communication system, fast detection of a transient disturbance and fast recovery after the transient has disappeared are important.

2.3 Frequency of Faults

The assumptions about the frequency of fault occurrence are depicted in Table 1. We distinguish between transient failures and permanent failures as well as between fail-silent failures and Byzantine failures.

Table 1. Assumed failure rates

Type of Failure	Failure Rate	Source
permanent fail silent	< 100 FIT (MTTF > 1 000 000 hours)	Field data from the automotive industry[2]
transient fail silent	< 100 000 FIT (MTTF > 1000 hours)	SEUs caused by neutrons[8]
permanent Byzantine	< 2 FIT (MTTF 50 000 000 hours)	Fault injection experiments[4]
transient Byzantine	< 2 000 FIT (MTTF > 50 000 hours)	Fault injection experiments[4]

Whereas the data in line one--permanent failures--is derived from extensive field data, the assumptions of line two, three and four are not as well supported by experimental data and field evidence. In particular it is very difficult to find a good estimate for the transient failure rates, because these failure are very dependent upon the environmental conditions (e.g., geometry of the setup determines the susceptibility with respect to EMI, geographical position and altitude determines the rate of SEUs etc..) of the unit under observation. The failure rates of Table 1 are our best estimates and are used in our reliability models to calculate the service availability of the TTA.

3 Structure of the TTA

The time-triggered architecture (TTA) is a distributed architecture for the implementation of hard real-time applications. It consists of a set of nodes interconnected by a TDMA (time-division multiple access) based real-time communication system. The TTA provides the following services to the application at the architecture level

(i) a consistent distributed computing platform with prompt error detection if consistency is lost by a failure that can be detected at the architecture level.
(ii) a fault-tolerant global sparse time base of known precision at all nodes
(iii) mechanisms for the precise operational specification of the interfaces among the nodes in the domains of time and value. These interfaces are called "temporal firewalls".
(iv) error containment such that arbitrary node failures can be tolerated
(v) mechanisms that support the transparent implementation of fault-tolerance.

In the following section we will discuss two essential characteristics of the TTA, the TTA view of time and state.

3.1 Global Sparse Time

For most applications, a model of time based on Newtonian physics is adequate. In this model, real time progresses along a *dense* timeline, consisting of an infinite set of *instants*, from the past to the future. A *duration (or interval)* is a section of the timeline, delimited by two instants. A happening that occurs at an instant (i.e., a cut of the timeline) is called an *event*. An *observation* of the state of the world at an instant is thus an event. The *time-stamp* of an event is established by assigning the state of the local clock of the observer to the event immediately after the event occurrence. Due to the impossibility of synchronizing clocks perfectly and the denseness property of real time, there is always the possibility of the following sequence of events occurring: clock in component *j* ticks, event *e* occurs, clock in component *k* ticks. In such a situation, the single event *e* is time-stamped by the two clocks *j* and *k* with a difference of one tick. The finite precision of the global time-base and the digitalization of the time make it impossible in a distributed system to order events consistently on the basis of their global time-stamps based on a *dense* time. This problem is solved by the introduction of a *sparse time base* in the TTA. In the sparse-time model the continuum of time is partitioned into an infinite sequence of alternating durations of *activity* and *silence* as shown in Figure 2. The activity intervals form a synchronized system-wide *action lattice*. From the point of view of temporal ordering, all events that occur within a duration of activity of the action lattice are considered to happen *at the same time*. Events that happen in the distributed system at different components at the same global clock-tick are thus considered *simultaneous*. Events that happen during different durations of activity (at different points of the action lattice) and are separated by the required interval of silence (the duration of this silence interval depends among others, on the precision of the clock synchronization [9]) can be temporally ordered on the basis of their global timestamps. The architecture must make sure that significant events, such as the sending of a message, or the observation of the environment, occur only during an interval of activity of the action lattice. The time-stamps of events that are based on a sparse time base can be mapped on the set of positive integers. It is then possible to establish the temporal order of events by integer arithmetic.

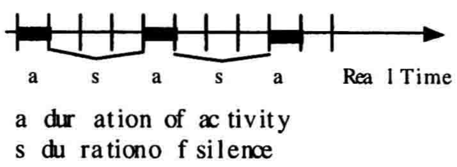

Fig. 2. Sparse time base

The timestamps of events that are outside the control of the distributed computer system (and therefore happen on a dense timeline) must be assigned to an agreed lattice point of the action lattice by an *agreement protocol*. Agreement protocols are also needed to come to a system-wide consistent view of analogue values that are digitized by more than one analogue-to-digital converter.

此为试读,需要完整PDF请访问: www.ertongbook.com