

Distributed Event-Based Systems

Gero Mühl · Ludger Fiege
Peter Pietzuch

Distributed Event-Based Systems

With 158 Figures and 17 Tables

Authors

Gero Mühl

Fakultät IV Elektrotechnik und Informatik
Technische Universität Berlin
Einsteinufer 17
10587 Berlin, Germany
g_muehl@acm.org

Ludger Fiege

Siemens AG
CT SE2
Otto-Hahn-Ring 6
81730 München, Germany
ludger.fiege@siemens.com

Peter Pietzuch

Div. of Engineering and Applied Sciences
Harvard University
33 Oxford Street
Cambridge, MA 02138, USA
prp@eecs.harvard.edu

Library of Congress Control Number: 2006927041

ACM Computing Classification (1998): C.2.4, C.3, D.2.11, D.2.12

ISBN-10 3-540-32651-0 Springer Berlin Heidelberg New York

ISBN-13 978-3-540-32651-9 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, reuse of illustrations, recitation, broadcasting, reproduction on microfilm 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 for prosecution under the German Copyright Law.

Springer is a part of Springer Science+Business Media

springer.com

© Springer-Verlag Berlin Heidelberg 2006

Printed in Germany

The use of general descriptive names, registered names, trademarks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

Typeset by the authors using a Springer $\text{\TeX}$ macro package

Production: LE- $\text{\TeX}$ Jelonek, Schmidt & Vöckler GbR, Leipzig

Cover design: KünkelLopka Werbeagentur, Heidelberg

Printed on acid-free paper 45/3100/YL - 5 4 3 2 1 0

For Regina.
— Gero

For Biggi,
Franzi, Flori, and Johanna.
— Ludger

For Bohdan Bańkowski.
— Peter

Preface

The field of event-based systems is surprisingly broad. In many scientific communities, technical talks, commercial products, and industrial projects people think about asynchronous computations and messaging, scalability and maintainability, stepwise evolution and loose coupling. Most likely, these people are discussing event-based systems, even if they use other terms.

When we began investigating event-based systems some years ago, we were surprised to see that *eventing* was scattered among many disciplines of computer science. There were no workshops or conferences dedicated to this topic, for example, although many aspects of event-based systems cannot be assessed from a database, network, or software engineering perspective alone. In the same sense, commercially available products that could help solving problems of event-based architectures are often bundled and marketed in solutions of a specific domain.

In order to channel some of the attention, the Distributed Event-Based Systems (DEBS) workshop series was created. It attracts people from distributed computing, database, and software engineering audiences, and it demonstrates the wide variety of facets event-based systems have. After having heard about and being engaged in interesting discussions about allegedly “academic” and “real-world” problems, in investigating many findings, and after creating many solutions in both academic and industrial environments, we decided to write this book to present both the current state-of-the-art and its base concepts.

The book takes a distributed system’s point of view. This is, of course, partly due to our own background, but more importantly we believe a solid understanding of distributed event-based systems is a good starting point for building modern computing systems. It lets you integrate sophisticated filter and data processing capabilities as well as new network topologies and routing algorithms.

Acknowledgements

We want to thank our colleagues, coauthors, and friends who discussed and developed most of the ideas presented in this book with us. Without being able to name all, we want to thank Jean M. Bacon, Alejandro P. Buchmann, Frank Buschmann, Mariano Cilia, Felix C. Freiling, Rachid Gerraoui, Michael A. Jaeger, Arno Jacobsen, Mira Mezini, Ken Moody, Joe Sventek, Andreas Ulbrich, Andreas Zeidler, and many others we worked and talked with in universities and companies, at conferences, and via email. The good thing about writing a book is that you gain so many new insights into already known topics.

We also want to thank Ralf Gerstner from Springer Verlag for his patience and continuous support, and the reviewers and proofreaders who helped us improve the book.

Last but not least, we are grateful to our families and friends for their patience and understanding for yet another evening being occupied with this “nonsense”. Thanks.

Berlin, Germany
Munich, Germany
Cambridge, MA, USA

Gero Mühl
Ludger Fiege
Peter Pietzuch

May 2006

Contents

1	Introduction	1
1.1	Networked Computing	1
1.2	Middleware	2
1.3	Event-Based Systems	3
1.4	Application Scenarios	4
1.4.1	Information Dissemination	4
1.4.2	Network Monitoring	4
1.4.3	Enterprise Application Integration	5
1.4.4	Mobile Systems	6
1.4.5	Ubiquitous systems	6
1.5	Putting Event-Based Systems Into Context	7
1.6	From Centralized to Internet-Scale Event Systems	8
1.7	Structure of the Book	8
2	Basics	11
2.1	Terminology	11
2.1.1	Events and Notifications	11
2.1.2	Producers and Consumers	12
2.1.3	Subscriptions and Filters	13
2.1.4	Event Notification Service	13
2.2	Models of Interaction	14
2.2.1	Request/Reply	15
2.2.2	Anonymous Request/Reply	15
2.2.3	Callback	16
2.2.4	Event-Based	16
2.2.5	Comparison	17
2.2.6	Interaction vs. Implementation	17
2.3	Notification Filtering Mechanisms	19
2.3.1	Channels	19
2.3.2	Subject-Based Filtering	19
2.3.3	Type-Based Filtering	19

2.3.4	Content-Based Filtering	20
2.4	A Model Distributed Notification Service	20
2.4.1	System Model	20
2.4.2	Architecture	21
2.4.3	Distributed Notification Routing	22
2.5	Specification of Event Systems	23
2.5.1	Formal Background	24
2.5.2	A Simple Event System	26
2.5.3	A Simple Event System With Ordering Requirements ..	30
2.5.4	Simple Event System With Advertisements	31
2.6	Further Reading	33
3	Content-Based Models and Matching	35
3.1	Content-Based Data and Filter Models	35
3.1.1	Tuples	35
3.1.2	Structured Records	36
3.1.3	Semistructured Records	52
3.1.4	Objects	56
3.2	Matching Algorithms	57
3.2.1	Brute Force	59
3.2.2	Counting Algorithm	59
3.2.3	Decision Trees	60
3.2.4	Binary Decision Diagrams	61
3.2.5	Efficient XML Matching	63
3.3	Further Reading	64
4	Distributed Notification Routing	67
4.1	System Model	67
4.2	Routing Algorithm Framework	69
4.2.1	Atomic Steps of the Implementation	69
4.2.2	Notification Forwarding and Delivery	72
4.2.3	Avoidance of Duplicate and Spurious Notifications	73
4.2.4	Routing Table Updates	73
4.3	Valid and Monotone Valid Routing Algorithms	74
4.3.1	Valid Routing Algorithms	74
4.3.2	Monotone Valid Routing Algorithms	76
4.4	Valid Framework Instantiations	77
4.5	Content-Based Routing Algorithms	80
4.5.1	Flooding	81
4.5.2	Simple Routing	82
4.5.3	Identity-Based Routing	85
4.5.4	Covering-Based Routing	91
4.5.5	Merging-Based Routing	98
4.5.6	Discussion	104
4.6	Extensions of the Basic Routing Framework	107

4.6.1	Routing With Advertisements	107
4.6.2	Hierarchical Routing Algorithms	112
4.6.3	Rendezvous-Based Routing	115
4.6.4	Topology Changes	117
4.6.5	Joining and Leaving Clients	119
4.6.6	Routing in Cyclic Topologies	120
4.6.7	Exploiting IP Multicast	122
4.6.8	Topology Maintenance	123
4.7	Further Reading	125
5	Engineering of Event-Based Systems	129
5.1	Engineering Requirements	129
5.1.1	Application Examples	130
5.1.2	Requirements	132
5.1.3	Existing Support	136
5.2	Accessing Publish/Subscribe Functionality	137
5.2.1	Generic APIs	137
5.2.2	Domain-Specific APIs	139
5.3	Using the API	140
5.3.1	Patterns and Idioms	141
5.3.2	Emitting Notifications	143
5.4	Further Reading	147
6	Scoping	149
6.1	Controlling Cooperation	150
6.1.1	Implicit Coordination and Visibility	150
6.1.2	Explicit Control of Visibility	151
6.1.3	The Role of Administrators	151
6.2	Event-Based Systems With Scopes	152
6.2.1	Visibility and Scopes	152
6.2.2	Specification	153
6.2.3	Notification Dissemination	156
6.2.4	Duplicate Notifications	158
6.2.5	Dynamic Scopes	159
6.2.6	Attributes and Abstract Scopes	161
6.2.7	A Correct Implementation	161
6.3	Event-Based Components	164
6.3.1	Component Interfaces	164
6.3.2	Scope Interfaces	164
6.3.3	Event-Based Components	167
6.3.4	Example	167
6.4	Notification Mappings	169
6.4.1	Specification	169
6.4.2	A Correct Implementation	173
6.4.3	Example	176

6.5	Transmission Policies	176
6.5.1	Publishing Policy	177
6.5.2	Delivery Policy	179
6.5.3	Traverse Policy	180
6.5.4	Influencing Notification Dissemination	181
6.6	Engineering With Scopes	182
6.6.1	Development Process	182
6.6.2	Scope Graph Handling	183
6.6.3	Scope Graph Language	187
6.7	Implementation Strategies for Scoping	196
6.7.1	Scope Architectures	197
6.7.2	Comparing Architectures	209
6.7.3	Implement Scopes as Event Brokers	210
6.7.4	Integrate Scoping and Routing	213
6.8	Combining Different Implementations	225
6.8.1	Architectures and Scope Graphs	226
6.8.2	Bridging Architectures	227
6.8.3	Integration With Other Notification Services	228
6.9	Further Reading	228
7	Composite Events	231
7.1	Application Scenarios	231
7.2	Requirements	234
7.3	Composite Events	234
7.4	Composite Event Detection	236
7.4.1	Composite Event Detectors	236
7.4.2	Composite Event Language	238
7.5	Detection Architectures	242
7.5.1	Centralized Detection	243
7.5.2	Distributed Detection	244
7.6	Further Reading	250
8	Advanced Topics	253
8.1	Security	253
8.1.1	Application Scenarios	254
8.1.2	Requirements	255
8.1.3	Access Control Techniques	256
8.1.4	Secure Publish/Subscribe Model	258
8.1.5	Further Reading	264
8.2	Fault Tolerance	264
8.2.1	Fault Masking	265
8.2.2	Self-Stabilizing Publish/Subscribe Systems	265
8.2.3	Self-Stabilizing Content-Based Routing	266
8.2.4	Generic Self-Stabilization Through Periodic Rebuild ...	273
8.2.5	Further Reading	276

8.3	Congestion Control	276
8.3.1	The Congestion Problem	277
8.3.2	Requirements	277
8.3.3	Congestion Control Algorithms.....	279
8.3.4	Further Reading	285
8.4	Mobility.....	287
8.4.1	Mobility Issues in Publish/Subscribe Middleware	289
8.4.2	Physical Mobility.....	290
8.4.3	Logical Mobility.....	295
8.4.4	Further Reading	302
9	Existing Notification Services	305
9.1	Standards	305
9.1.1	CORBA Event and Notification Service	305
9.1.2	Jini.....	310
9.1.3	Java Message Service (JMS)	311
9.1.4	Data Distribution for Real-Time Systems (DDS)	313
9.1.5	WS Eventing and WS Notification.....	317
9.1.6	The High-Level Architecture (HLA)	317
9.2	Commercial Systems	318
9.2.1	IBM WebSphere MQ	318
9.2.2	TIBCO Rendezvous	320
9.2.3	Oracle Streams Advanced Queuing	322
9.3	Research Prototypes	324
9.3.1	Gryphon	324
9.3.2	SIENA.....	326
9.3.3	JEDI	329
9.3.4	REBECA	331
9.3.5	Hermes	334
9.3.6	Cambridge Event Architecture (CEA).....	337
9.3.7	Elvin	340
9.3.8	READY.....	340
9.3.9	Narada Brokering	340
10	Outlook	343
	References	349
	Index	379

List of Figures

1.1	A news story dissemination system	5
1.2	The Active Office ubiquitous environment	6
1.3	The structure of the book	9
2.1	Event-based systems: interaction versus implementation	12
2.2	Taxonomy of cooperation models	15
2.3	The router network of REBECA	21
2.4	A simple event system	26
3.1	Identity of filters consisting of attribute filters	44
3.2	$F_1 \supseteq F_2$ although neither $F_1^1 \supseteq F_2^1$ nor $F_1^1 \supseteq F_2^2$ (two examples)	45
3.3	Covering of filters consisting of attribute filters	46
3.4	Disjoint filters consisting of attribute filters	47
3.5	Overlapping filters consisting of attribute filters	47
3.6	Matching algorithm based on counting satisfied attribute filters	50
3.7	Covering algorithm that determines all <i>covering</i> filters	51
3.8	Covering algorithm that determines all <i>covered</i> filters	51
3.9	Merging algorithm based on counting identical attribute filters	52
3.10	A simple notification	54
3.11	Implementation of a <code>ClassFilter</code> in Java	58
3.12	Implementation of a <code>QuoteFilter</code> in Java	58
3.13	Using a multilevel index structure for the counting algorithm	60
3.14	An exemplary decision tree	60
3.15	An exemplary binary decision diagram	61
3.16	Evaluating a filter using a binary decision diagram	62
3.17	Evaluating an ordered binary decision diagram	62
3.18	XPath Queries and their corresponding finite state automaton	63
3.19	Combined nondeterministic finite state automaton	64
4.1	Content-based routing framework, part I	70
4.2	Content-based routing framework, part II	71

4.3	Diagram explaining notification forwarding	73
4.4	Flooding	81
4.5	Simple routing	83
4.6	Diagram explaining simple routing (new subscription)	84
4.7	Relation among α and β for simple routing	84
4.8	Identity-based routing	87
4.9	Identity-based routing: Processing a new subscription from a neighbor	87
4.10	Identity-based routing: Processing a new subscription from a client	88
4.11	Relation among α and β for identity-based routing	88
4.12	Covering-based routing	90
4.13	Covering-based routing: Processing of a new subscription from a client	93
4.14	Covering-based routing: Processing of a new subscription from a neighbor	94
4.15	Covering-based routing: Processing of an unsubscription from a neighbor	94
4.16	Covering-based routing: Processing of an unsubscription from a client	95
4.17	Covering-based routing: Processing of an unsubscription from a client	95
4.18	Covering-based routing: Processing of an unsubscription from a neighbor, example 2	96
4.19	Relation among α and β for covering-based routing	97
4.20	Merging-based routing	99
4.21	Merging: deletion of covering filters	99
4.22	Merging: searching for a covering merger	100
4.23	Merging: handling of subscriptions	101
4.24	Merging: handling of unsubscriptions	103
4.25	Circular evolution of CBR algorithms	106
4.26	Routing using advertisements, part I	108
4.27	Routing using advertisements, part II	109
4.28	prune for simple routing	110
4.29	prune for identity-based routing	111
4.30	Hierarchical covering-based routing	112
4.31	Hybrid routing	113
4.32	Rendezvous-based routing	116
4.33	Managing connects and disconnects	119
4.34	Simple routing in cyclic topologies: algorithm	121
4.35	Example of simple routing in cyclic topologies	122
4.36	Routing a message in a Pastry network	125
5.1	Data flow graphs of applications: bipartite single (a) and mult source (b), and a general group (c)	130

5.2	An example stock trading application	133
5.3	Generic publish/subscribe interface	138
5.4	The structure of the observer pattern	141
5.5	Event and notification in a UML class diagram	144
6.1	A metamodel of scopes	153
6.2	An exemplary scope graph	154
6.3	Outgoing and incoming notifications	157
6.4	Two ways of generating duplicates	158
6.5	A possible implementation of a scoped event system	162
6.6	Different scope interfaces	165
6.7	The graph of the stock application	168
6.8	Interfaces of the components in the example application	168
6.9	Recursive definition of the relation $(n_1, X) \rightsquigarrow (n_2, Y)$	170
6.10	Transformation of mappings into components	174
6.11	Architecture of scoped event system with mappings	174
6.12	Three important transmission policies in scope graphs	177
6.13	Scope definition accuracy	196
6.14	Design dimensions of scope architectures	197
6.15	Implicit implementation shifts visibility control into application components	201
6.16	A comparison of scope architectures	203
6.17	Steps of scoped notification delivery	207
6.18	Types of architectures, their characteristics, and examples	208
6.19	Comparison of scope architectures	210
6.20	An exemplary scope graph	214
6.21	Scopes as overlays within the broker topology	214
6.22	A flat routing table for broker B_1	215
6.23	Enhanced routing tables of B_1 incorporating scopes	216
6.24	Scope lookup tables	217
6.25	Overall routing algorithm	220
6.26	The naïve matching algorithm with mappings	221
6.27	Interscope forwarding	222
6.28	Duplicate scopes to separate QoS requirements	226
7.1	The Active Office with different sensors	232
7.2	A system for monitoring faults in a network	233
7.3	The components of the composite event detection service	236
7.4	The states in a composite event detection automaton	237
7.5	The transitions in a composite event detection automaton	237
7.6	A composite event detection automaton	238
7.7	The architecture for the composite event detection service	243
7.8	Illustration of centralized composite event detection	243
7.9	Illustration of distributed composite event detection	244

7.10	Two cooperating composite event detectors for distributed detection	245
7.11	The life cycle of a mobile composite event detector	245
7.12	The design space for distribution policies	247
8.1	An event type hierarchy for the Active City	255
8.2	Illustration of the secure publish/subscribe model	258
8.3	An event type hierarchy with attribute encryption	262
8.4	Subscription coverage with attribute encryption	263
8.5	Deriving the minimum leasing time	269
8.6	Notification bandwidth saved by doing filtering instead of flooding	272
8.7	Choosing π such that “old” and “new” update messages do not interleave	274
8.8	Derivation of the maximum stabilization time	276
8.9	Flow of DCQ and UCA messages	280
8.10	Processing of DCQ and UCA messages at IBs	283
8.11	Consolidation of UCA messages at IBs	283
8.12	Missing notifications in a flooding scenario	291
8.13	Moving client scenarios with one and multiple producers	293
8.14	Blackout period after subscribing with simple routing	297
8.15	Blackout period with flooding and client-side filtering	297
8.16	Defining the quality of service for logical mobility	298
8.17	Network setting for the example	299
8.18	Movement graph defining movement restrictions of a consumer	299
8.19	Total number of messages generated for flooding and two scenarios of the new algorithm	301
9.1	Internal structure of an object request broker (ORB)	305
9.2	Push mode vs. pull mode (typed event communication)	308
9.3	Typed event communication using an event channel	308
9.4	The structure of a structured event (from [287])	309
9.5	Conceptual overview of data-centric publish/subscribe (DCPS)	314
9.6	A Gryphon network with virtual event brokers	325
9.7	A hierarchical topology in SIENA	327
9.8	An acyclic peer-to-peer topology in SIENA	327
9.9	A generic peer-to-peer topology in SIENA	328
9.10	Hierarchical event routing in JEDI	329
9.11	Substituting one link with another link	330
9.12	An exemplary router network of REBECA	331
9.13	The filtering framework of REBECA	332
9.14	Layered networks in HERMES	335
9.15	Overview of the HERMES architecture	336
9.16	The publish–register–notify paradigm in the CEA	338
9.17	An ODL definition of event types in ODL-COBEA	339

List of Tables

2.1	Some exemplary temporal formulas and their informal meaning	26
2.2	Interface operations of a simple event system	27
2.3	Changes of the state variables caused by interface operations...	27
2.4	Additional interface operations for advertisements	32
2.5	Changes of the state variables caused by the additional interface operations for advertisements	32
3.1	Covering among notification types	40
3.2	Covering among (in)equality constraints on simple values	40
3.3	Covering among comparison constraint on simple values	41
3.4	Covering among interval constraints on simple values	41
3.5	Covering among constraints on strings	41
3.6	Covering among set constraints on simple values	42
3.7	Covering among set constraints on multi values	42
3.8	Perfect merging rules for attribute filters	49
4.1	Portfolio of content-based routing algorithms	104
7.1	Example of five distribution policies	248
8.1	Values of $ploc(x, t)$ for the example setting	300
8.2	Values of filters in example setting	301