

# Component Service Optimization and Verification for Web Service Composition

( Web服务组合中组件服务的优化和验证技术研究 )

---

Chen Liping (陈莉萍)



科学出版社

# Component Service Optimization and Verification for Web Service Composition

(Web 服务组合中组件服务的优化和验证技术研究)

Chen Liping (陈莉萍)

Responsible Editor: Xiangping Yang and Wuhan Song

Copyright© 2016 by Science Press  
Published by Science Press  
16 Donghuangchenggen North Street  
Beijing 100717, P. R. China

Printed in Beijing

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior written permission of the copyright owner.

ISBN 978-7-03-049877-9

## Preface

Today, the need for advanced services is rapidly increasing, as consumers' needs become more demanding and complex, most notably as a result of the growth in demand for mobile devices and ubiquitous computing environments. Web service composition aims to respond to today's challenges, as it combines several atomic services to form a unique service composition that provides a solution to these complex demands. Service-oriented architecture (SOA) represents an appropriate architectural model for composite services, enabling, even dynamically, combinations of a variety of independently developed services to form distributed, software intensive systems.

A key plank of the service computing agenda is Web service composition: it aims at providing techniques, models, and architectures for the automation of multiple, autonomous, and dissimilar services to produce new and novel services. Web service composition benefits include better techniques for service outsourcing and innovative and serendipitous services. Applications abound and span almost numerous areas, including e-government, life sciences, hospitality, disaster management, education, health, IT outsourcing, cloud computing, and many more. A key technology enabler for services is Web services which is tightly congruent with the service paradigm. There have been tremendous activities around Web service standardization which must be said, has probably gone beyond what was needed. Without any doubt, the problem of Web service composition is viewed by many as the "holy grail" in Services Computing. There have been many attempts by researchers from various domains to perform research on this highly relevant and timely subject. The optimization selection and reliability verification played a more prominent role in Web service composition research for the following reasons: On the one hand, with the proliferation of Web services offering similar functionality, developed on various platforms and in various programming languages, the problem of how to choose optimal service for composition from a given class has become increasingly important. On the other hand, service reliability verification becomes a prerequisite as these new kinds of modern services are present in our everyday life affecting several everyday functions and situations. Web services are usually managed by different companies and may not have been developed for composition. Analogous to other distributed systems based on asynchronous communication, it

is difficult to anticipate how Web service compositions behave during execution and whether they conform to the functional requirements identified during requirements engineering. Errors, however, may violate service level agreements. This may cause losses or penalties and have negative impact on the reputation of the companies involved. Therefore, it is an important task to make sure that service compositions are reliable at run-time.

This book is to the best of our knowledge, the first of its kind to address service optimization by using aiNet. The aiNet is capable of finding optimal services with different QoS (Quality of Service) properties from similar functionality services, which provides a new approach for service selection. The service optimization selection implements not only QoS properties but also transactional properties, which ensure reliable execution of composite Web service and construct the optimal composite Web service. Reliability verification is another cool feature in this book. The conformance verification approach with temporal logic is implemented. The expected interaction is usually defined by business process execution language (BPEL), which is translated into a colour Petri net model. We use scenarios from e-government (social services) and life sciences (analysis of protein sequence information) to illustrate the concepts and techniques discussed in this book.

This work is partially supported by Science and Technology Plan Projects of Shaanxi Education Department, No. 15jk1237, by the projects of characteristic discipline construction of Weinan Normal University, No. 14TSXK09, by the projects of Weinan Normal University, No. 13YKP013 and 13YKS006, by Science and Technology Plan Projects of Weinan, No 2015KYJ-2-3.

Chen Liping  
June 20, 2016

# Contents

## Preface

<b>Chapter 1</b>	<b>Introduction</b>	1
1.1	Background	1
1.1.1	Web service model and framework	1
1.1.2	Web service composition definition	3
1.1.3	Motivations and goals of Web service standards	4
1.2	Related standards and technologies	5
1.2.1	Web service-related standards and technologies	5
1.2.2	Web service composition standards and technologies	9
1.2.3	Sample for different standards supporting Web service composition	16
1.3	Web service composition methods	20
1.3.1	Web service composition methods based on workflow	21
1.3.2	Web service composition methods based on semantics	23
1.4	References	28
<b>Chapter 2</b>	<b>Different Phases of Web Service Composition</b>	30
2.1	Composite process design	30
2.2	Deployment	31
2.3	Component service discovery	31
2.4	Component service selection	33
2.4.1	Personal service selection	33
2.4.2	Cooperative service selection	35
2.4.3	Service selection based on QoS	37
2.5	Web service composition verification	44
2.5.1	Requirements for verification process	45
2.5.2	Modeling the composition in MSCs	46
2.6	Composite Web service execution	48
2.6.1	An example for execution of service composition	49
2.6.2	Compatible executions	51

2.6.3	Guaranteeing a composite service execution	53
2.7	References	55
<b>Chapter 3</b>	<b>What is Component Service Optimization and Verification for Web Service Composition</b>	<b>56</b>
3.1	Component service optimization	56
3.1.1	Service optimization challenges	57
3.1.2	Service optimization spectrum	59
3.1.3	Optimization approaches	60
3.1.4	Negotiation-based optimization	64
3.2	Verification of Web service composition	68
3.2.1	Verification architecture	69
3.2.2	Verification properties	70
3.2.3	Operationalization correctness verification	70
3.3	References	72
<b>Chapter 4</b>	<b>A New Web Service Optimization with Memory Classifier</b>	<b>74</b>
4.1	Introduction	74
4.2	Fuzzy C-means clustering algorithm	75
4.2.1	Fuzzy C-means functional	76
4.2.2	Fuzzy C-means clustering algorithm	77
4.2.3	Parameters of the FCM algorithm	78
4.3	Artificial immune network	80
4.3.1	Resource limited artificial immune system	81
4.3.2	AiNet	82
4.3.3	iNet	83
4.3.4	IPAIsys	84
4.4	Ontology-oriented evaluation model description of semantic Web service based QoS	85
4.5	Evaluation algorithm of Web service based QoS	86
4.5.1	The objective and subjective synthetic approach for weigh of evaluation attribute	86
4.5.2	Fuzzy C-means artificial immune network memory classifier (FCMAINMC)	87
4.6	Evaluation prototype system of Web service based QoS	91
4.7	Simulation	93
4.8	Conclusion	94
4.9	References	94
<b>Chapter 5</b>	<b>Evaluation Model of Web Service Health Level on End-to-End Network Based on Artificial Immune</b>	<b>96</b>
5.1	Introduction	96
5.2	Optimization using artificial immune systems	96

5.3	ENHMM evaluation model construction .....	97
5.3.1	Formation of dynamic evaluation tree .....	97
5.3.2	Evaluation data acquired based on evaluation tree .....	98
5.3.3	End-to-end network service health evaluation level designed .....	98
5.3.4	Evaluation of network health level based on new aiNet immune network .....	99
5.4	ENHMM simulations .....	102
5.5	Conclusion .....	103
5.6	References .....	103
<b>Chapter 6</b>	<b>Extended Evaluation for Quality of Service in the Community of Web Service .....</b>	<b>104</b>
6.1	Introduction .....	104
6.2	Building of Web service community .....	105
6.2.1	Basic concepts .....	105
6.2.2	Dynamic building of Web service community .....	105
6.3	Ontology-oriented extended evaluation model description of semantic Web service .....	106
6.4	Triangular fuzzy analytic hierarchy process .....	107
6.4.1	Triangular fuzzy numbers (TFNs) .....	107
6.4.2	Algebraic operations on TFNs .....	107
6.4.3	Construction of the FAHP comparison matrices .....	108
6.4.4	Value of fuzzy synthetic extent .....	108
6.4.5	Calculation of the sets of weight values of the FAHP .....	109
6.5	A new evaluation algorithm based on triangular fuzzy number .....	110
6.5.1	New triangular fuzzy analytic hierarchy process (NTFAHP) .....	110
6.5.2	Assessing of improved fuzzy comprehensive evaluation method .....	111
6.6	Evaluation examples .....	113
6.6.1	Building of extended evaluation tree in the train booking service community .....	113
6.6.2	Service evaluation .....	114
6.7	Conclusion .....	114
6.8	References .....	114
<b>Chapter 7</b>	<b>Adaptive Evaluation and Selection of Information System by Triangular Fuzzy Number .....</b>	<b>116</b>
7.1	Introduction .....	116
7.2	Reviews on information system evaluation theory .....	118
7.2.1	User satisfaction research stream .....	119
7.2.2	Taxonomy framework of information system evaluation methods .....	121
7.3	A new information systems evaluation algorithm based on triangular fuzzy numbers .....	123



7.3.1	Index of information system evaluation	123
7.3.2	The weight obtained from NTFAHP	127
7.3.3	Numerical examples	130
7.4	Framework of evaluation system	131
7.5	Discussion	132
7.6	Conclusion	134
7.7	References	134
<b>Chapter 8 Service Selection of Ensuring Transactional Reliability and QoS for Web Service Composition</b>		136
8.1	Introduction	136
8.2	Web service transaction descriptions	138
8.2.1	Transactions overview	138
8.2.2	Web service transactional property	139
8.2.3	Composite Web service transactional property	139
8.3	Transactional automaton services selection	140
8.3.1	Workflow patterns	140
8.3.2	Definition of risk tolerance	147
8.3.3	Transactional automaton services selection	147
8.4	Transactional automaton model for services selection to Web service composition	150
8.4.1	I/O automata	150
8.4.2	Modelling transaction systems	150
8.4.3	Transactional automaton model for services selection	152
8.4.4	Example of service selection driven by transactional automaton model	153
8.5	QoS-based Web service selection	154
8.5.1	QoS-based Web service model	154
8.5.2	Qos-based composite Web service	155
8.5.3	QoS-based service selection for CWS	155
8.6	Experimentation	157
8.7	Conclusion	160
8.8	References	160
<b>Chapter 9 A Formal Transaction Model for Reliable Web Service Composition</b>		162
9.1	Introduction	162
9.2	A formal description to Web services composition	164
9.3	Transaction model for WSC	165
9.3.1	Transactional properties of Web service	166
9.3.2	The set of control flow and transaction relationship	167

9.4	Generating of transaction relationship in the workgroup	167
9.4.1	Complete set of TF	168
9.4.2	Automatic generating of TF	169
9.4.3	Validation of transaction model for Web service composition	169
9.5	Application of transaction model for Web service composition	170
9.6	Conclusion	171
9.7	References	171
<b>Chapter 10</b>	<b>Reliable Execution Based on CPN and Skyline Optimization for Web Service Composition</b>	<b>172</b>
10.1	Introduction	172
10.2	Related work	174
10.3	Reviews on the methodologies for reliable Web services	176
10.3.1	Fault tolerance	176
10.3.2	Redundancy	177
10.3.3	Diversity	178
10.3.4	Reliable Web services and composition	178
10.4	A colored Petri-net model of Web service composition	180
10.4.1	Colored Petri-net	180
10.4.2	Formal definition of CP-nets	182
10.4.3	Transactional property description	183
10.4.4	Tolerance level	183
10.4.5	TCWS-CPN definition	184
10.4.6	Services selection of transactional property in the TCWS-CPN	185
10.4.7	Composite sequence in the CPN	186
10.5	Execution framework architecture of TCWS-CPN	187
10.5.1	Execution framework architecture	187
10.5.2	Example	189
10.6	Skyline computation intruduction	193
10.7	QoS-based skyline Web services	195
10.7.1	The skyline computation problem	195
10.7.2	Skyline Web services for QoS-based composition	195
10.7.3	Skyline algorithm of QoS-based Web service selection	197
10.8	Experimentation	198
10.9	Conclusion	199
10.10	References	200
<b>Chapter 11</b>	<b>Conformance Checking for Interaction of Web Service Composition with Temporal Logic</b>	<b>202</b>

- 11.1 Introduction .....202
- 11.2 Conformance checking approach .....205
- 11.3 Conformance verification and interaction of Web service composition .....207
  - 11.3.1 Conformance verification ..... 207
  - 11.3.2 Web service interactions ..... 209
- 11.4 Conformance checking framework of for Web service interaction behaviours.....211
- 11.5 Web service composition interaction modeling: BPEL-CPN model .....212
  - 11.5.1 Definition of BCPN model ..... 213
  - 11.5.2 Transformation of BPEL into BCPN model ..... 213
- 11.6 Conformance checking for interaction of Web service composition .....217
  - 11.6.1 Interaction fitness and appropriateness computation..... 218
  - 11.6.2 Temporal properties of BCPN model..... 219
  - 11.6.3 Conformance checking algorithm for interaction of Web service composition..... 220
- 11.7 Examples and experiment .....222
  - 11.7.1 Fitness and appropriateness computation of example..... 224
  - 11.7.2 Conformance checking with temporal logic ..... 225
- 11.8 Conclusion .....225
- 11.9 References.....229

# Chapter 1

## Introduction

Web services are considered as self-contained, self-describing, modular applications that can be published, located, and invoked across the Web, which have generated great interests in both vendors and researchers. Web services, based on existing Internet protocols and open standards, can provide a flexible solution to the problem of application integration. With the help of WSDL, SOAP, and UDDI, Web services are becoming popular in Web applications. Nowadays, an increasing amount of companies and organizations implement their core business and outsource other applications as Web services over Internet. However, single Web service cannot satisfy all the functionality needed by the user. It is imperative task to study how to compose Web services into a new service. Web service composition focuses on how to integrate existing Web services in diverse and heterogeneous distributed environments, providing different functional, nonfunctional, and behavioral features, to quickly construct workable applications or software for satisfying the requirements which are requested by users and unable to be fulfilled by any single Web service. This book focuses on how to find and select optimal component services form same functionality candidate services, and verify reliability of Web service composition. This chapter discusses the state of Web service composition.

### 1.1 Background

#### 1.1.1 Web service model and framework

The sharing of data and application system resources among collaborating organizations is the key to achieving their common goals. With the Internet, especially the Web, becoming the fastest growing collection of data and software programs, more and more academia, business and government organizations have focused their efforts on promoting the sharing of these Web resources.

A Web service is defined by the W3C <sup>[1]</sup> as "a software system designed to support interoperable Machine to Machine interaction over a network." The Web services

model proposed by IBM and Microsoft is a framework designed to facilitate the sharing of a tremendous amount of data and software resources on the Web. It allows software/application systems' functionalities to be defined as Web services, also referred to as services hereafter, and makes them programmatically accessible over the Internet. This model allows the publishing of business functions to the Web and enables universal access to these functions. Both developers and end-users can enjoy the benefits of Web services. The Web service model simplifies business application development and interoperation. Additionally, it greatly serves end-user needs by enabling them through an intuitive, browser-based interface to choose, configure and assemble their own Web services [2].

The Web service model shown in the Figure 1.1 contains 3 roles: service provider, service requester and service broker, and the following basic activities: describing, publish/unpublish/update, discover and invoke/bind [3].

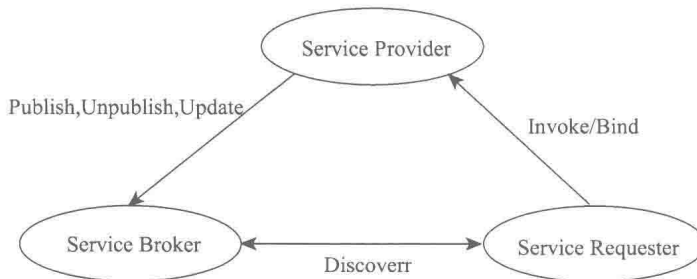


Figure 1.1 Web service model

### 1) Service provider

A service provider is the party that provides software applications for specific needs as services. Service providers publish, unpublish and update their services so that they are available on the Internet. From a business perspective, this is the owner of the service. From an architectural perspective, this is the platform that holds the implementation of the service.

### 2) Service requester

A requester is the party that has a need that can be fulfilled by a service available on the Internet. From a business perspective, this is the business that requires certain function to be fulfilled. From an architectural perspective, this is the application that is looking for and invoking a service. A requester could be a human user accessing the service through a desktop or a wireless browser; it could be an application program; or it could be another Web service. A requester finds the required services via a service broker and binds to services via the service provider.

### 3) Service broker

This party provides a searchable repository of service descriptions where service providers publish their services and service requesters find services and obtain binding

information for these services. It is like telephone yellow pages. Such examples of service brokers are UDDI (Universal Description, Discovery, Integration) and Xmethods.

It is clear that since the service provider, the service broker and the service requester interact with each other they should use standard technologies for service description, communication and data formats.

There are generally three roles involved in Web service applications: service provider, service broker, and service requester. The interactions between these three roles are published, find and invoke/bind. Web services are implemented and published by service providers. They are discovered and invoked by service requesters. Information about a Web service (i.e., service descriptions) is kept within a service broker.

### 1.1.2 Web service composition definition

With the rapid expansion of Web service related applications in fields such as e-business, e-government and e-health, there is a clear need for infrastructures and frameworks that can be used to develop applications on the basis of Web service compositions. The majority of SOA publications are concentrating on definition and implementations of the individual business services. Building enterprise solution(s) typically requires combining multiple existing enterprise services. These composite services can be in turn recursively composed with other services into higher level solutions, and so on. Such recursive composition of business services is one of the most important features of SOA, allowing to rapidly building new solutions based on the existing business services. As the amount of individual business services (and their compositions) grows, the easier it becomes to implement new enterprise solutions.

The main drivers for the creation of composite-services are:

#### 1) Usage simplicity

When several business services are used together by multiple consumers, exposing knowledge about all participating services and their coordination rules to every consumer can make consumer implementation more complex. Creation of composite service, both encapsulating participating services and enforcing the rules of their invocation can significantly simplify their usage.

#### 2) Improved reusability

New unplanned solutions can often be assembled from available services. Even though they have been designed for the construction of a specific set of solutions, existing business services can be combined in other ways to implement solutions that had not been anticipated. In addition, the availability of services suggests new solutions that might otherwise not be considered. These new solutions can often be created inexpensively and quickly through development or enhancement of relatively few services.

#### 3) Solution partitioning, visibility, control and change management

Composite services can serve as a partitioning mechanism for overall solution. Similar

to the case of local and remote interfaces in EJBs, introducing composite services and exposing only some of their interfaces to external users allows controlling what is visible to the consumer. This supports the ability of underlying software architectures (composite services implementations) to rapidly respond to changing requirements by changing the implementations of its subordinate services, as well as the interconnection between them with minimal or no impact to the consumers.

There is currently no single definition of Web service composition, and we collect some existing definitions. The Web service composition, according to Jonatha et al.<sup>[4]</sup>, can be defined as follows: The Web service composition is the process of building an instance of an abstract workflow by combining appropriate Web services that satisfies given QoS requirements. In general, QoS requirements consist of a number of constraints. The selection process requires global optimization and can be formalized as a mixed integer linear programming problem which cannot be solved in polynomial time.

Two other definitions are as follows:

Web services composition is an emerging paradigm for application integration within and across organizational boundaries. A landscape of languages and techniques for Web services composition has emerged and is continuously being enriched with new proposals from different vendors and coalitions.

Web services composition is an emerging paradigm for enabling application integration within and across organizational boundaries. Current Web services composition proposals, such as BPML, WSBPEL, WSCI and OWL-S, provide solutions for describing the control and data flows in Web service composition.

Taking into account the above definitions of Web service composition, we define Web service composition: Provides an open, standards-based approach for connecting Web services together to create higher-level business processes. Standards are designed to reduce the complexity required to compose Web services, hence reducing time and costs, and increase overall efficiency in businesses.

### 1.1.3 Motivations and goals of Web service standards

A good starting point for understanding the Web services paradigm is to consider the stated goals, as found in the literature and the standards communities. The basic motivation of standards such as SOAP and WSDL is to allow a high degree of flexibility in combining Web services to create more complex ones, often in a dynamic fashion. The current dream behind UDDI is to enable both manual and automated discovery of Web services, and to facilitate the construction of composite Web services. Building on these, the BPEL standard provides the basis for manually specifying composite Web services using a procedural language that coordinates the activities of other Web services.

Much more ambitious goals are espoused by the OWL-S coalition (formally known as DAML-S), and more broadly the semantic Web services community<sup>[5]</sup>. These goals are to provide machine-readable descriptions of Web services, which will enable automated discovery, negotiation with, composition, enactment, and monitoring of Web services.

A kind of middle ground is also emerging, which provides abstract signatures of Web services that are richer than WSDL but retain a declarative flavor. Most popular here is the use of automata based descriptions of permitted sequencing patterns of the Web services, with a focus on either activities performed<sup>[6]</sup> or messages passed<sup>[7]</sup>.

## 1.2 Related standards and technologies

### 1.2.1 Web service-related standards and technologies

Today there is a lot of activity in the Web service area. Services can assume different roles when involved in various interaction scenarios. Depending on the context with which it's viewed, as well as the state of the currently running task, the same Web service can change roles or be assigned multiple roles at the same time. Thomas Erl delves into the concepts and technology behind Web services, including Web services and the service-oriented architecture (SOA), Web services Description Language (WSDL), Simple Object Access Protocol (SOAP), and Universal Description, Discovery, and Integration (UDDI). We are currently witnessing the rapid development and maturation of a stack of interrelated standards that are defining the Web service infrastructure along with a great number of development tools that support the Web service development. The following standards play key roles in Web services: Universal Description, Discovery and Integration (UDDI), Web Services Description Language (WSDL), Web Services Inspection Language (WSIL), SOAP, and Web Services Interoperability (WS-I). The relationship between these standards is described in Figure 1.2.

#### 1. UDDI

UDDI is a protocol that allows businesses to promote, use, and share services over the Internet. It is an OASIS Standard, which is supported by several major technology companies. Members include Microsoft, Cisco, Oracle, Avaya, Sun Microsystems, and others. The UDDI specification defines open, platform-independent standards that enable businesses to share information in a global business registry, discover services on the registry, and define how they interact over the Internet. It is a platform-independent, Extensible Markup Language (XML)-based registry by which businesses worldwide can list themselves on the Internet, and a mechanism to register and locate Web service



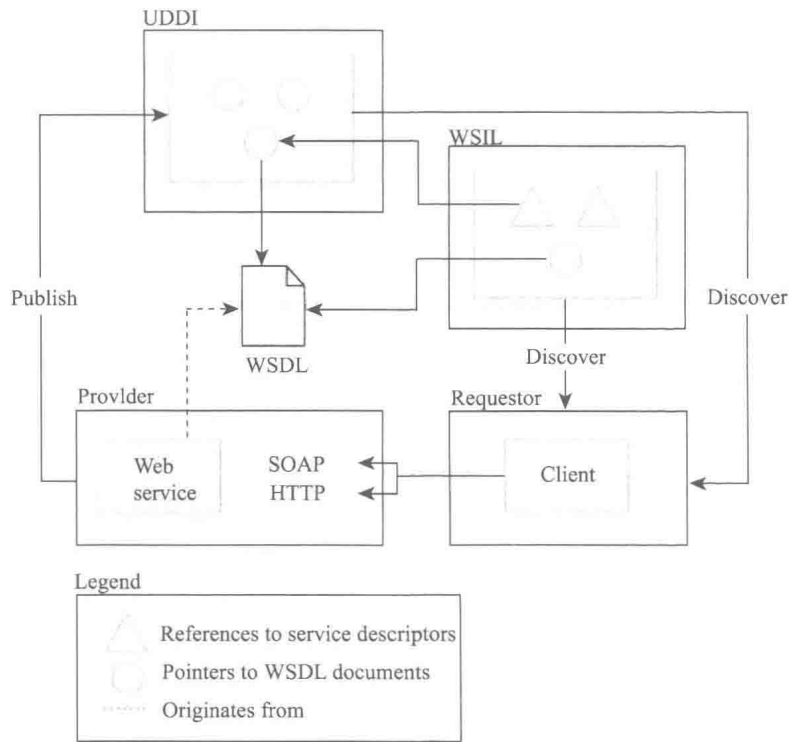


Figure 1.2 Relationships between SOAP, UDDI, WSIL and WSDL

applications. The UDDI protocol serves as a foundational tool that allows businesses to find each other and complete transactions quickly and easily. Companies that use the UDDI protocol can extend their market reach and find new customers while also finding other businesses that offer useful services to them. Because UDDI uses a standard format for describing business services, it is easy to search and find useful services offered from other businesses. UDDI was originally proposed as a core Web service standard<sup>[1]</sup>. It is designed to be interrogated by SOAP messages and to provide access to Web Services Description Language (WSDL) documents describing the protocol bindings and message formats required to interact with the Web services listed in its directory.

Figure 1.3 depicts the makeup of the UDDI project. The UDDI Business Registry (UBR), also known as the Public Cloud, is a conceptually single system built from multiple nodes that has their data synchronized through replication. A series of operator nodes each hosts a copy of the content. The global grouping of operator nodes is jointly known as the UBR. Operator nodes replicate content among one another. Accessing any individual operator node provides the same information and quality of service as any other operator node. Content inserted into the UBR is done at a single node, and that operator node becomes the master owner of that content. Any subsequent updates or deletes of the data must occur at the operator node where the data was inserted.