

بسم الله الرحمن الرحيم





شبكة المعلومات الجامعية التوثيق الالكتروني والميكرو فيلم



جامعة عين شمس

التوثيق الإلكتروني والميكروفيلم

قسم

نقسم بالله العظيم أن المادة التي تم توثيقها وتسجيلها
علي هذه الأقراص المدمجة قد أعدت دون أية تغيرات



يجب أن

تحفظ هذه الأقراص المدمجة بعيدا عن الغبار





بالرسالة صفحات
لم ترد بالأصل





بعض الوثائق الأصلية تالفة



B 11241

A Single System Image Supporting Distributed Objects

Usama A. M. Badawi
Department of Mathematics
Faculty of Science
Cairo University
Giza – Egypt

For the Fulfilment of
the degree of Ph.D.
in Computer Science

Supervised By

Prof. Dr. Laila F. Abdel Aal
Department of Mathematics
Faculty of Science
Cairo University
Giza – Egypt

Prof. Dr. Salwa M. Nassar
Parallel Systems Division
Computer and Control Department
Electronic Research Institute
Giza – Egypt

Dr.-Ing. Thomas Setz
Theoretic Computer Science
Institute of Computer Science
Technical University of Darmstadt
Darmstadt – Germany

Dr. Hatem M. Moharram
Department of Mathematics
Faculty of Science
Cairo University
Giza – Egypt

November 2000

Approval Sheet
For the Ph.D. Thesis Titled
A Single System Image Supporting
Distributed Objects

Presented By
Usama A. M. Badawi

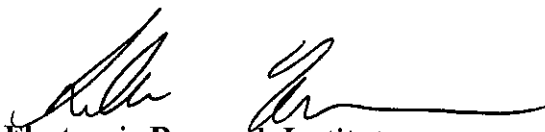
Submitted to
Faculty of Science, Cairo University

Supervised By

Prof. Dr. Laila F. Abdel Aal
Department of Mathematics, Faculty of Science, Cairo University.



Prof. Dr. Salwa M. Nassar
Computer and Control Department., Electronic Research Institute.



Prof. Dr. B. T. Hassan

Head of the Dept. of Mathematics
Faculty of Science, Cairo University

2000

Abstract

Nowadays, parallel systems, such as networks of heterogeneous workstations, are widely used to solve distributed applications. An easy way to implement such applications is the use of a distributed memory which is shared among all machines participating in the application. In such environment, failures, such as machine crashes, are harmful because of the possibly lost data due to the failure. In most cases the application is to be restarted from scratch.

If the application is a computational intensive one and if the handled data must remain alive to survive the application, failures must be handled dynamically.

Our solution to such problem adds an additional layer, called the *high availability layer*, to the shared memory architecture. Such layer includes mechanisms to handle the failures dynamically.

To implement such layer, the protocols introduced by the TRansis¹ system are used as well as the shared memory supported by the LIPS² system. Hence, we named the resulting system **TRIPS**. TRIPS is a way to achieve a unified view³ to the shared memory among all nodes in the parallel machine in spite of the failures. Its aim is to establish a protocol to achieve a *Single System Image*.

¹A Group Communication system implemented at the labs of the Hebrew University of Jursalem.

²Library for parallel systems, A system designed and implemented in Saarbruecken university, Germany. It is used to distribute computational intensive applications on a net of approximately 250 machines.

³If the machines have identical copies of the shared memory, they have a unified view to it.

Dedications are not enough ...

To my *mother* for her love To my wife *Nesma* for her big heart

Acknowledgment

Many people have done a lot for me during my research. I wish to express my deep thanks to all of them ...

To *Dr.-Ing. Thomas Setz* who has given me all the possible support, cooperation, and meaningful help during my work under his supervision in Darmstadt - Germany.

To *Prof. Dr. Johannes Buchmann*, the head of our chair in the Technical university of Darmstadt for his kind hospitality.

To my supervisors in Egypt, *Prof. Dr. Laila F. Abdel Aal*, Faculty of Science, Cairo University, *Prof. Dr. Salwa M. Nassar*, Electronic Research Institute, National Research Center, and *Dr. Hatem Moharram*, Faculty of Science, Cairo University for there valuable encourage and support.

To *Dr. A.M. El-Asrag* for his valuable advice as a father and friend.

To my colleague *Mr. Thomas Liefke*, and our Secretary *Frau Marita Skrobic* for their meaningful help during my stay in Germany.

CONTENTS

Contents

Overview	1
1 Introduction to Tuple-space based Parallel Systems	6
1.1 Parallel Conceptual Classes	8
1.2 Distributed Shared Memory Architecture	10
1.3 Generative Communication	12
1.3.1 The Linda Tuple-Space Model	13
1.3.2 Benefits of Tuple-Space Based Systems	14
1.3.3 Examples on Tuple-Space Based Systems	17
1.3.3.1 IBM T Spaces	17
1.3.3.2 SUN JavaSpaces Service	18
1.4 Single System Image: Needs and Approaches	20
1.4.1 The Operating System Point of View	20
1.4.2 Benefits of Single System Image	21
1.4.3 Examples of Systems Supporting Single System Image	22
1.4.3.1 Solaris MC: A Distributed Operating System for Clusters	23
1.4.3.2 GLUnix: A Global Layer Unix for NOW	23
1.4.3.3 MOSIX	24
1.4.4 Tuple-space Systems Support of Single System Image	24
1.4.4.1 A General View	25
1.4.4.2 The Possible Problems and Their Solutions	25
1.5 Summary	29

2	TRIPS Cooperative Systems	31
2.1	The Transis Group Communication System	33
2.1.1	Transis System Overview	33
2.1.2	Transis System Structure	36
2.1.3	Transis Support of Extended Virtual Synchrony	39
2.1.3.1	Virtual Synchrony	39
2.1.3.2	Extended Virtual Synchrony	40
2.1.3.3	An Example on Extended Virtual Synchrony	42
2.2	The LiPS System	43
2.2.1	LiPS Development Environment	44
2.2.2	The LiPS Tuple-space	45
2.2.2.1	LiPS Tuple-space Structure	45
2.2.2.2	LiPS Tuple-space Access Operations	46
2.2.3	LiPS Runtime Systems	47
2.2.4	LiPS Runtime Systems Enhancements	50
2.3	Summary	52
3	The TRIPS System Design	53
3.1	TRIPS System Basic Concepts	56
3.1.1	Trips-Box	56
3.1.2	The Trips-Daemon	59
3.1.3	The Replica	62
3.1.4	The Representative	63
3.2	The TRIPS System Layers	65
3.2.1	Local Tuple-Space Layer	65
3.2.2	TRIPS Middle Layer 1	66
3.2.3	TRIPS Middle Layer 2	69
3.2.4	Transis Events Layer	69
3.3	Summary	70

4	TRIPS System Implementation	71
4.1	TRIPS Data Structures	73
4.2	TRIPS System Libraries	76
4.2.1	TRIPS External Libraries	78
4.2.1.1	External Libraries from LiPS	78
4.2.1.2	External Libraries from Transis	82
4.2.2	The Kernel Library	82
4.2.2.1	The Scheduler (<i>myconnect</i>) Function Profile	84
4.2.2.2	The tran Module Functions Profiles	84
4.2.2.3	The lipsth Module Functions Profiles	85
4.3	TRIPS System Flow Control	87
4.4	The State Change Handling Algorithm	89
4.5	Summary	93
5	Testing the TRIPS System	94
5.1	TRIPS Coverage Tests	96
5.1.1	LiPS Test Environment	96
5.1.2	Coverage Tests to TRIPS Functions	97
5.1.2.1	The SendBroadcastMessage() Function Coverage	98
5.1.2.2	The ReceiveBroadcastMessage() Function Coverage	99
5.1.2.3	The ReceiveMessage() Function Coverage	99
5.1.2.4	The Enqueue() Function Coverage	101
5.1.2.5	The Dequeue() Function Coverage	102
5.2	TRIPS Performance Tests	103
5.2.1	Test Conditions	104
5.2.2	Singleton Trips-Box Tests	105
5.2.2.1	Singleton Trips-Box Send-Only Test	105
5.2.2.2	Singleton-Trips-Box Ping-Pong Test	107
5.2.3	Two Members Trips-Box Tests	108
5.2.3.1	Two Members Trips-Box Send-Only Test	109
5.2.3.2	Two Members Trips-Box Ping-Pong Test	110
5.2.4	Three Members Trips-Box Tests	112
5.2.4.1	Three Members Trips-Box Send-Only Test	112
5.2.4.2	Three Members Trips-Box Ping-Pong Test	113

5.2.5	Comparing Performance Results	114
5.2.6	More Members Test (The Ring Test)	116
5.3	Application Tests	116
5.3.1	Linear System of Equations Application	116
5.3.1.1	Test Scenario	117
5.3.1.2	Test Results	117
5.3.2	Distributed Counter Application	118
5.3.2.1	Test Scenario	118
5.3.2.2	Test Results	118
5.4	Different Join/Disjoin Scenarios	118
5.4.1	Singleton Trips-Box State Change Test	120
5.4.2	Two-Members Trips-Box State Change Tests	121
5.4.2.1	One Member Joins The Trips-Box	121
5.4.2.2	Two Members Join The Trips-Box	123
5.5	Summary	126
6	Conclusion and Future Work	127
A	Index	132
B	Technical Index	134
C	Glossary	136