

**Stream Data Processing:
A Quality of Service Perspective
Modeling, Scheduling, Load Shedding, and
Complex Event Processing**

ADVANCES IN DATABASE SYSTEMS

Volume 36

Series Editors

Ahmed K. Elmagarmid

*Purdue University
West Lafayette, IN 47907*

Amit P. Sheth

*Wright State University
Dayton, Ohio 45435*

For other titles published in this series, go to
www.springer.com/series/5573

Stream Data Processing: A Quality of Service Perspective Modeling, Scheduling, Load Shedding, and Complex Event Processing

by

Sharma Chakravarthy
*The University of Texas at Arlington
Arlington, TX, USA*

Qingchun Jiang
*Oracle Corporation
Redwood Shores, CA, USA*



Springer

Sharma Chakravarthy
Department of Computer Science
& Engineering
University of Texas at Arlington
Arlington, TX 76019, USA
sharmac@uta.edu

Qingchun Jiang
Oracle Corporation
500 Oracle Parkway
Redwood Shores, CA 94065, USA
dustin.jiang@gmail.com

ISBN: 978-0-387-71002-0 e-ISBN: 978-0-387-71003-7
DOI: 10.1007/978-0-387-71003-7

Library of Congress Control Number: 2009920957

© Springer Science+Business Media, LLC 2009

All rights reserved. This work may not be translated or copied in whole or in part without the written permission of the publisher (Springer Science+Business Media, LLC, 233 Spring Street, New York, NY 10013, USA), except for brief excerpts in connection with reviews or scholarly analysis. Use in connection with any form of information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed is forbidden.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

Printed on acid-free paper

springer.com

In Loving Memory of my father **Vidwan Chakravarthy Ananthachar** (27th June 1916 to 9th December 2000) who instilled in me an appreciation for learning and a thirst for knowledge

– Sharma Chakravarthy

To my wife **Ailing Wang**, and kids **George and Kevin** for their unconditional love and support

– Qingchun Jiang

Preface

In recent years, a new class of applications has come to the forefront – primarily due to the advancement in our ability to collect data from multitudes of devices, and process them efficiently. These include homeland security applications, sensor/pervasive computing applications, various kinds of monitoring applications, and even traditional applications belonging to financial, computer network management, and telecommunication domains. These applications need to process data continuously (and as long as data is available) from one or more sources. The sequence of data items continuously generated by sources is termed a data stream. Because of the possible never-ending nature of a data stream, the amount of data to be processed is likely to be unbounded. In addition, timely detection of interesting changes or patterns or aggregations over incoming data is critical for many of these applications. Furthermore, the data arrival rates may fluctuate over a period of time and may be bursty at times. For most of these applications, Quality of Service (or QoS) requirements, such as response time, memory usage, and throughput are extremely important. These application requirements make it infeasible to simply load the incoming data streams into a persistent store and process them effectively using currently available database management techniques.

As the currently used data processing paradigm does not fully meet the new requirements of these stream applications, the paradigm shift needed for the effective management of stream data needs to be understood, and new techniques/approaches have to be developed. As stream data is handled by applications in disparate domains, stream data processing can be addressed at different levels and in different contexts: processing of sensor data, pervasive computing, situation monitoring, real-time response, approximate algorithms, on-the-fly mining, complex event processing, or a combination thereof. Although the applications are diverse, some of the fundamental characteristics of stream data and their processing requirements remain common to most applications. Time-critical processing which can be generalized to a QoS requirement is one of the fundamental requirements and forms the thrust of this book. Prior to the advent of stream processing systems as a research area,

database management systems (DBMSs) and their variants were used for some of the above applications. Hence, stream processing can also be related to earlier work on main-memory (or in-memory) databases, embedded databases, and real-time transaction processing systems. This book focuses on the QoS aspects of stream processing and concomitant extensions/modifications that are needed in comparison to the features supported by traditional DBMSs.

In this book, we first discuss the characteristics of stream data and its processing requirements to arrive at a paradigm shift, cull out some of the challenges of a data stream processing system (or a DSMS), and propose our solutions to the challenges. We develop an architecture along with the functional components required for processing data streams and meet the requirements of stream applications. After a broad literature survey to provide a bird's-eye view of the research and development efforts in the thrust areas of stream and complex event processing, the book focuses on functionalities of a DSMS from a *QoS* perspective. In other words, techniques and approaches required for guaranteeing QoS requirements are presented and elaborated in this book. This leads us to focus on the topics of: (a) continuous query modeling, (b) scheduling strategies for improving QoS, (c) load shedding to satisfy QoS requirements with minimal deviation in accuracy of results, and (d) synergistic integration of stream and complex event processing to satisfy end-to-end QoS requirements. In addition, we analyze a complex, real-life network fault management application in detail to infer the extensions needed for integrating stream and complex event processing. Finally, we describe the design and implementation of a stream processing system and its synergistic integration with an existing Complex Event Processing (CEP) system.

The organization of the book is designed to facilitate reading of groups of chapters depending upon reader's interest. By choosing appropriate chapters, readers of this book can obtain: an overview of the thrust areas (stream and complex event processing), technical details on important issues of focus topics, and insights into the architecture, design, and implementation of stream & CEP systems. The discussions are cross-referenced at the chapter and section level to help the reader.

Objectives

The objective of this book is to provide a comprehensive understanding of stream processing and its relationship with complex event processing. We start with an overview of these areas, a clear understanding of the processing requirements, and challenges leading to a review of the large body of work that exists in these two areas. We then provide a technical discussion of topics from a QoS perspective. This book also introduces readers to a new paradigm, its relevance, importance, and the theoretical underpinnings of the abstractions needed for the new paradigm. The book also addresses the transition of techniques into prototypes along with design and implementation details. The

contents of this book have been chosen to highlight the QoS aspects of stream processing. The techniques presented here can be applied to other domains and applications as well. In summary, this book provides the state of the art in the areas of stream and complex event processing, presents technical details on selected topics, and discusses design and implementation of prototypes to make the book useful for a diverse audience.

Intended Audience

What background should one have to appreciate this book and benefit the most? Someone who has an undergraduate or a graduate degree in computer science has that background. Those who have taken a course on or have worked with database management systems can readily relate to the paradigm shift and its details. An understanding of relational databases and mathematical inclination will help the reader in enjoying the technical contents of this book. We provide adequate discussion of related work so that the reader can understand and appreciate the approaches and techniques presented in the book. The discussion of prototypes provides insights into the design and implementation issues of a data stream management system that integrates complex event processing in a seamless manner.

This book would be of interest to: senior undergraduate and graduate students, computer professionals, IT managers, database management system users & administrators, time-critical or event-driven application developers, and developers dealing with publish/subscribe systems.

Acknowledgements

This book owes its existence to a large number of people who have played an active role throughout the development of the work presented in the book. The support from National Science Foundation (NSF) for the **MavHome** project (ITR grant 0121297) at The University of Texas at Arlington (UTA) was instrumental in getting us started on this problem. Qingchun's background in managing network applications and the frustrations he faced in the absence of a data stream processing system provided some of the impetus for this work. Another NSF grant (IIS 0534611) for the integration of stream and event processing helped us continue the work and complete the **MavStream** prototype.

The following students at UTA enthusiastically worked on the **MavStream** and **MavESstream** prototypes as part of their theses: Altaf Gilani, Satyaajeet Sonune, Vamshi Pajjuri, Bala Kumar Kendai, and Vihang Garg. Their contributions to this book are very much appreciated. Raman Adaikkalavan has contributed extensively to the contents of chapters on the integration of events with streams and prototype implementations. We sincerely thank him for his

contributions and for his time during the semester on this book. Raman has also edited the entire book, added most of the index, and has improved its technical presentation significantly. It is also our pleasure to thank our colleagues at UTA who were part of the **MavHome** project, especially Diane Cook and Larry Holder.

Proofreading is an arduous and time-consuming task. We were lucky to have several volunteers helping us out on this. We want to thank the following for improving the readability and the quality of presentation of the book: Raman Adaikkalavan, Reuben Brooks, Tejas Chakravarthy, Mohan Kumar, Jonathan Ndede, Roochi Mishra, Charu Puri, and Aditya Telang. Sincere thanks to Aditya Telang for also helping us extensively with figures and references.

Sharma Chakravarthy would like to thank his wife *Shubha* and son *Tejas*, and Qingchun Jiang would like to thank his wife *Ailing* and kids *Kevin* and *George* for their patience, love, encouragement, and support during the preparation of this book.

Last but not least, we would like to thank Ahmed Elmagarmid for suggesting that we consider writing a book on stream processing. This book would not have been possible without the patience, gentle nudging, and timely support of Susan Lagerstrom-Fife and Sharon Palleschi of Springer throughout the preparation of this book. Our sincere thanks to Deborah from Springer author support for resolving several issues on formatting and font usage.

Southlake, Texas
Foster City, California

Sharma Chakravarthy
Qingchun Jiang
November 2008

How to Use the Book

This book can be used in several ways. It can be used as a reference book, but the contents and organization have been successfully used for a graduate course as well. Organization of this book is meant to help readers maximize their benefit based on their interest in the areas of stream and complex event processing.

It is important to read the first three chapters (Introduction, Overview of Data Stream Processing, and DSMS Challenges) as it sets the stage for the rest of the contents of the book. These chapters introduce the problem and characteristics of stream data processing, continuous queries, need for window-based computation, and differences between stream processing & conventional data processing paradigms leading to the challenges arising from the new requirements. The DSMS challenges chapter also includes a concise overview of the material covered in each chapter of the book to provide a sneak peek into what is coming.

The chapter on literature survey (Chapter 4) provides a broad overview of the work carried out in stream and complex event processing along with an exhaustive bibliography. Most of the research systems and some commercial systems are introduced. In addition, detailed discussion of related work in focus topics is provided. The reader can gauge the interest of the research and vendor community on the thrust areas by the number of currently available prototypes and systems. The large number of references provided in the book is meant to help the reader access additional materials as needed. Internet links are also provided, where possible, for theses, technical reports, and home pages of commercial systems.

Each of the next three chapters (Modeling Continuous Queries over Data Streams, Scheduling Strategies for Continuous Queries, and Load Shedding in a DSMS) are fairly technical and self-contained with respect to its topic. These can be read selectively based on the interest of the reader. Each chapter discusses a problem and our proposed solution to that problem. Some of the theoretical aspects can be skipped without sacrificing the understandability

of the chapter contents. A concise non-technical summary of these chapters is provided in Chapter 3.

The purpose of Chapter 8 (*NFMⁱ* : An Inter-Domain Network Fault Management System) is primarily to analyze a complex, real-life application to cull out the critical requirements beyond stream processing. This chapter brings out the need for integrating complex event processing with stream processing.

Those who are mainly interested in complex event processing can read the first and the fourth chapters to obtain an understanding of CEP development, and continue with Chapters 8 on the need for complex event processing. Chapter 9 analyzes the similarities and differences between the two areas before providing a synergistic, integrated solution. If the design and implementation of systems is of interest, read Chapters 10 and 11 after reading the first four setting-the-stage chapters.

We have summarized the usage of the book into an easy-to-use format, based on the areas of interest, in the Table below.

Areas \ Chapters	1	2	3	4	5	6	7	8	9	10	11	12
Stream Processing (SP)	✓	✓	✓	✓	✓ ¹	✓ ¹	✓ ¹					✓
SP Design & Implementation	✓	✓	✓							✓		✓
Complex Event Processing (CEP)	✓			✓				✓	✓		✓	✓
SP & CEP Integration	✓	✓	✓	✓				✓	✓		✓	✓
SP & CEP	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

¹ These chapters can be read selectively

Mapping of Areas of Discussion to Chapter Contents

Sharma Chakravarthy continues to use the contents of this book (sans some of the theoretical details) to teach a one-semester graduate course on stream and complex event processing at UTA. Hands-on implementation projects are included to appreciate and apply the concepts taught, and to enhance the course experience. Students implement the same stream and event processing application using different freely available systems to analyze: expressiveness, ease of application development, QoS evaluation, and the ability to handle large data sets. Students discuss their project experience with the rest of the class through an in-class presentation. Most of the stream and complex event processing systems used for the course, such as StreamBase, Aleri, AMiT, Snoop/Sentinel, Coral8, Esper, and RuleCore are available free-of-charge for teaching purposes or as trial/lite/open source versions.

If you are interested in adapting this book for teaching, preliminary slides for the contents of the book can be obtained by sending email to sharma@cse.uta.edu with course information.

Contents

List of Figures	XIX
List of Tables	XXIII
List of Algorithms	XXV
1 INTRODUCTION	1
1.1 Paradigm Shift	3
1.2 Data Stream Applications	5
1.3 Book Organization	6
2 OVERVIEW OF DATA STREAM PROCESSING	9
2.1 Data Stream Characteristics	9
2.2 Data Stream Application Characteristics	10
2.3 Continuous Queries	12
2.3.1 Window Specification	14
2.3.2 Examples of Continuous Queries	16
2.3.3 QoS Metrics	18
2.4 Data Stream Management System Architecture	19
2.5 Summary of Chapter 2	20
3 DSMS CHALLENGES	23
3.1 QoS-Related Challenges	23
3.1.1 Capacity Planning and QoS Verification	23
3.1.2 Scheduling Strategies for CQs	24
3.1.3 Load Shedding and Run-Time Optimization	25
3.1.4 Complex Event and Rule Processing	26
3.1.5 Design and Implementation of a DSMS with CEP	27
3.2 Concise Overview of Book Chapters	27
3.2.1 Literature Review	27
3.2.2 Continuous Query Modeling	28

3.2.3	Scheduling Strategies for CQs	28
3.2.4	Load Shedding in a DSMS	29
3.2.5	<i>NFMⁱ</i> : A Motivating Application	30
3.2.6	DSMS and Complex Event Processing	30
3.2.7	Design and Implementation of Prototypes	31
4	LITERATURE REVIEW	33
4.1	Data Stream Management Systems	33
4.1.1	Aurora and Borealis	33
4.1.2	STREAM	34
4.1.3	<i>TelegraphCQ</i>	35
4.1.4	MavStream	36
4.1.5	Others	37
4.2	QoS-Related Issues	38
4.2.1	Continuous Query Modeling for Capacity Planning	38
4.2.2	Scheduling Strategies for CQs	39
4.2.3	Load Shedding in a DSMS	40
4.2.4	Design and Implementation of Prototypes	41
4.3	Complex Event Processing	41
4.3.1	Mid- to Late-Eighties: Active Databases	41
4.3.2	Nineties: Active Object-Oriented Databases	42
4.3.3	Beyond 2000: (Distributed) Complex Event Processing .	45
4.4	Commercial and Open Source Stream and CEP Systems	47
5	MODELING CONTINUOUS QUERIES OVER DATA STREAMS	49
5.1	Continuous Query Processing	50
5.1.1	Operator Path	50
5.1.2	Operator Modeling	52
5.1.3	Scheduling and Service Discipline	53
5.2	Problem Definition	54
5.2.1	Notations and Assumptions	56
5.2.2	Stability and Performance Metrics	57
5.3	Modeling Relational Operators	57
5.3.1	Modeling Select and Project Operators	58
5.3.2	Modeling Window-Based Symmetric Hash join	60
5.3.3	Steady State Processing Cost	60
5.3.4	Handling Bursty Inputs and Disk-Resident Data	68
5.4	Modeling Continuous Queries	69
5.4.1	Modeling Operators with External Input(s)	71
5.4.2	Modeling Operators with Internal Input(s)	75
5.4.3	Modeling Operators with External and Internal Inputs .	79
5.4.4	Scheduling Strategy and Vacation Period	79
5.4.5	Computing Memory Usage and Tuple Latency	82
5.5	Intuitive Observations	82

5.5.1	Tuple Latency	82
5.5.2	Service Discipline.....	82
5.5.3	Scheduling Algorithms	83
5.5.4	Choice of Query Plans	84
5.5.5	Input Rate	84
5.6	Experimental Validation.....	85
5.6.1	Validation of Operator Models	86
5.6.2	Validation of Continuous Query Plan Models	89
5.7	Summary of Chapter 5	93
6	SCHEDULING STRATEGIES FOR CQs.....	95
6.1	Scheduling Model and Terminology	96
6.1.1	Scheduling Model	97
6.1.2	Notations	99
6.2	Impact of Scheduling Strategies on QoS	103
6.3	Novel Scheduling Strategies for CQs	105
6.3.1	Path Capacity Strategy	106
6.3.2	Analysis of CQ Scheduling Strategies	108
6.3.3	Segment Strategy and Its Variants.....	111
6.3.4	Hybrid Threshold Scheduling Strategy	122
6.3.5	CQ Plan Characteristics.....	124
6.3.6	Starvation-Free Scheduling	125
6.4	Experimental Validation.....	126
6.4.1	Setup	126
6.4.2	Evaluation of Scheduling Strategies	127
6.5	Summary of Chapter 6	136
7	LOAD SHEDDING IN DATA STREAM MANAGEMENT SYSTEMS	137
7.1	The Load Shedding Problem	138
7.2	Integrating Load Shedders	140
7.2.1	Load Shedder as Part of a Buffer	142
7.2.2	Types of Load Shedders	143
7.3	Load Shedding Framework.....	143
7.3.1	Prediction of Query Processing Congestion	144
7.3.2	Placement of Load Shedders	151
7.3.3	Allocation of Load for Shedding	156
7.3.4	Load Shedding Overhead	157
7.4	Experimental Validation.....	158
7.4.1	Prototype Implementation	158
7.4.2	Experiment Setup	158
7.4.3	Load Shedding with Path capacity strategy	160
7.4.4	Load Shedding with EDF scheduling strategy	163
7.5	Summary of Chapter 7	165

8	<i>NFMⁱ</i>: AN INTER-DOMAIN NETWORK FAULT MANAGEMENT SYSTEM	167
8.1	Network Fault Management Problem	168
8.2	Data Processing Challenges for Fault Management	170
8.2.1	Semi-structured Text Messages	171
8.2.2	Large Number of Messages	172
8.2.3	Complex Data Processing	172
8.2.4	Online Processing and Response Time	172
8.3	Stream- and Event-Based <i>NFMⁱ</i> Architecture	173
8.3.1	Message Splitter	175
8.3.2	Message Filter and Information Extractor	175
8.3.3	Alarm Processing	178
8.4	Three-Phase Processing Model for <i>NFMⁱ</i>	178
8.4.1	Continuous Query (CQ) Processing Phase	178
8.4.2	Complex Event Processing Phase	181
8.4.3	Rule Processing Phase	182
8.4.4	Summary	183
8.5	Transactional Needs of Network Management Applications	184
8.5.1	Updates and Views	185
8.6	Summary of Chapter 8	186
9	INTEGRATING STREAM AND COMPLEX EVENT PROCESSING	187
9.1	Motivation	188
9.2	Event Processing Model	191
9.2.1	Event Detection Graphs	192
9.2.2	Event Consumption Modes	192
9.2.3	Event Detection and Rule Execution	194
9.3	Complex Event Vs. Stream Processing	195
9.3.1	Inputs and Outputs	195
9.3.2	Consumption Modes Vs. Window Types	196
9.3.3	Event Operators Vs. CQ Operators	197
9.3.4	Best-Effort Vs. QoS	197
9.3.5	Optimization and Scheduling	198
9.3.6	Buffer Management and Load Shedding	198
9.3.7	Rule Execution Semantics	199
9.3.8	Summary	199
9.4	MavEStrEam: An Integrated Architecture	200
9.4.1	Strengths of the Architecture	201
9.5	Stream-Side Extensions	203
9.5.1	Named Continuous Queries	203
9.5.2	Stream Modifiers	205
9.6	Event-Side Extensions	207
9.6.1	Generalization of Event Specification	207
9.6.2	Event Specification using Extended SQL	208

9.6.3	Mask Optimization	210
9.6.4	Enhanced Event Consumption Modes	210
9.6.5	Rule Processing	211
9.7	Summary of Chapter 9	213
10	MavStream: DEVELOPMENT OF A DSMS PROTOTYPE	215
10.1	MavStream Architecture	216
10.1.1	Functionality	216
10.1.2	MavStream Server Design	217
10.1.3	MavStream Server Implementation	219
10.2	Window Types	220
10.2.1	Functionality	220
10.2.2	Design	220
10.2.3	Implementation	222
10.3	Stream Operators and CQs	222
10.3.1	Functionality	222
10.3.2	Design of Operators	223
10.3.3	Implementation	225
10.4	Buffers and Archiving	229
10.4.1	Functionality	229
10.4.2	Buffer Manager Design	230
10.4.3	Buffer Manager Implementation	231
10.5	Run-time Optimizer	231
10.5.1	Functionality	231
10.5.2	Run-time Optimizer Design	232
10.5.3	Run-time Optimizer Implementation	234
10.6	QoS-Delivery Mechanisms	243
10.6.1	Functionality	243
10.6.2	Scheduler Design	243
10.6.3	Scheduler Implementation	245
10.6.4	Load Shedder Design	246
10.6.5	Load Shedder Implementation	246
10.7	System Evaluation	248
10.7.1	Single QoS Measure Violation	248
10.7.2	Multiple QoS Measures Violation	249
10.7.3	Effect of Load Shedding on QoS Measures	253
10.7.4	Effect of Load Shedding on Error in Results	257
11	INTEGRATING CEP WITH A DSMS	261
11.1	MavEStrEam: Integration Issues	262
11.1.1	Event Generation	263
11.1.2	Continuous Event Query (CEQ) Specification	264
11.1.3	Events and Masks	264
11.1.4	Address Space Issues	265
11.1.5	Summary	265

11.2 Design of the Integrated System	266
11.2.1 Address Space	266
11.2.2 Continuous Event Queries	266
11.2.3 Events and Masks	268
11.2.4 Event Generator Interface	271
11.2.5 Need for a Common Buffer for All Events	272
11.2.6 Complex Events and Rule Management	274
11.3 Implementation Details of Integration	275
11.3.1 Input Processor	276
11.3.2 Event and Rule <i>Instantiator</i>	278
11.3.3 Event Generator Interface	278
11.4 Stream Modifiers	281
11.4.1 Tuple-Based Stream Modifiers	281
11.4.2 Window-Based Stream Modifiers	282
11.4.3 Implementation	282
11.5 Additional Benefits of CEP Integration	284
11.6 Summary of Chapter 11	285
12 CONCLUSIONS AND FUTURE DIRECTIONS	287
12.1 Looking Ahead	287
12.2 Stream Processing	288
12.2.1 Continuous Query Modeling	289
12.2.2 Scheduling	289
12.2.3 Load Shedding	290
12.3 Integration of Stream and Event Processing	291
12.4 Epilogue	293
References	295
Index	315

List of Figures

1.1	Data Stream Processing Using a DBMS	3
1.2	Data Stream Processing Using a DSMS	4
2.1	Query Processing in a DSMS Vs. DBMS	14
2.2	A Window With Start and End Boundaries	15
2.3	Window Types and Their Specification Using Start and End Boundaries	15
2.4	A Sliding (Overlapping) Window	16
2.5	Data Streams in a Telecommunication Service Provider	17
2.6	Architecture of the Proposed QoS-Aware DSMS	20
5.1	Operator Path	51
5.2	Modeling of a Continuous Query	52
5.3	Modeling of Select Operator	58
5.4	Window-based Symmetric Hash Join	61
5.5	Queueing Model of Window-based Symmetric Hash Join Operator	63
5.6	Three Classes of Queueing Models	70
5.7	Input Process Of Operator (Internal Input)	76
5.8	Push-Up Operator Scheduling Strategy	80
5.9	CDF of Delay In Queue (Same Window Size)	87
5.10	CDF of Delay In Queue (Different Window Size)	88
5.11	Target Query Plan	90
5.12	Tuple Latency Vs. System Load	92
6.1	Scheduling Model	97
6.2	A Complex Operator Path/Segment	101
6.3	A Query Execution Plan	104
6.4	Bottom-up Scheduling Strategy	106
6.5	Example of Complex MOS Construction	116
6.6	Example of Segmentation	118

6.7	Two Cases For The Memory Release Capacity Of The Operator O_i	119
6.8	Tuple Latency Vs. Time	128
6.9	Tuple Latency Vs. Time (Detailed)	130
6.10	Throughput Vs. Time	131
6.11	Throughput Vs. Time	132
6.12	Total Memory Requirement Vs. Time	134
6.13	Total Memory Requirement Vs. Time	135
7.1	Load Shedders	141
7.2	Cost of Load Shedders	142
7.3	A shared-Segment	146
7.4	The Global Operator Path	148
7.5	Locations Of Load Shedders	153
7.6	Tuple Latency Under Path (group1)	160
7.7	Tuple Latency Under Path (group3)	161
7.8	Moving Average Under PC strategy	162
7.9	Total Internal Queue Size Under PC strategy	163
7.10	Tuple Latency Under EDF (group1)	164
7.11	Total Internal Queue Size Under EDF	165
8.1	A Typical Telecommunication Network	168
8.2	A Fault Propagation Scenario	169
8.3	Architecture of the Proposed Inter-Domain Network Fault Management System	174
9.1	A Simple Event Detection Graph (EDG)	192
9.2	EStreams: Four Stage Stream and Complex Event Integration Model	201
9.3	CarADN Example using MavESStream Architecture	213
10.1	Functional Components of MavStream System	217
10.2	Query Plan Object With Data Structures	220
10.3	GHashTable Details	228
10.4	Latency Graph	232
10.5	Latency Graph: Piecewise Approximation	232
10.6	Run-time Optimizer Flow Chart	242
10.7	Latency: Single QoS Measure	250
10.8	Latency: Single QoS Measure – Details	251
10.9	Latency: Different Measures with Different Priorities	252
10.10	Latency: Different Measures With Same Priority	254
10.11	Memory Utilization: Different Measures with Same Priority	255
10.12	Effect of Load Shedding on Tuple Latency	256
10.13	Effect of Load Shedding on Memory Utilization	258
10.14	Random Shedders: Error in Average	259

10.15 Semantic Shedders: Error in Average	260
11.1 Event Generation and Propagation	273
11.2 MavEStream Architecture	275
11.3 Input Processor	277
12.1 General Architecture of Situation Monitoring Applications.....	288

List of Tables

- 2.1 Back-of-the Envelope Comparison of a DBMS with a DSMS ... 21
- 5.1 Delay & Queue Size (Same Window Size) 87
- 5.2 Delay & Queue Size (Different Window Size) 88
- 5.3 Parameters for Target Query Plan 90
- 5.4 Tuple Latency (in seconds) Under Hierarchical Scheduling Strategy 91
- 5.5 Tuple Latency (in seconds) Under Global Scheduling Strategy . 91
- 6.1 Operator Properties 104
- 6.2 Performance (F:FIFO, C:Chain)..... 105
- 10.1 Decision Table (Showing Relative Strategy Rank) 235
- 10.2 Single QoS Measure Violated 237
- 10.3 Multiple QoS Measures Violated 237
- 10.4 QoS Measure, Priority Class, and Weight 238
- 10.5 Violation of Measures with Different Priority Classes..... 239

List of Algorithms

1	Greedy Segment Construction Algorithm	113
2	Memory-Optimal Segment Construction Algorithm For a Simple Operator Path.....	115
3	Memory-Optimal Segment Construction Algorithm For a Complex Operator Path.....	117
4	Sharing Segment Search Algorithm	149
5	SEARCH($\mathcal{PC}$)	150
6	Group By Computation	228
7	Run-time Optimizer Algorithm.....	241
8	<i>System Monitor</i> Algorithm	242
9	Master Scheduler Algorithm	244
10	Event Generator Operator	280
11	Tuple-based Stream Modifier	282
12	Window-Based Stream Modifier	283