

DYNAMIC MANAGEMENT OF NETWORK SYSTEMS

By

Syed Yousaf Shah

A Dissertation Submitted to the Graduate
Faculty of Rensselaer Polytechnic Institute
in Partial Fulfillment of the
Requirements for the Degree of
DOCTOR OF PHILOSOPHY
Major Subject: **COMPUTER SCIENCE**

Approved by the
Examining Committee:

Prof. Boleslaw K. Szymanski, Dissertation Adviser

Prof. Christopher D. Carothers, Member

Prof. Malik Magdon-Ismail, Member

Dr. Petros Zerfos, Member

Rensselaer Polytechnic Institute
Troy, New York

November 2014
(For Graduation December 2014)

© Copyright 2014
by
Syed Yousaf Shah
All Rights Reserved

CONTENTS

LIST OF TABLES	vi
LIST OF FIGURES	vii
ACKNOWLEDGMENT	ix
ABSTRACT	xi
1. Introduction	1
1.1 Research Problem	2
1.1.1 Resource Management in Networks	3
1.1.2 Data Relevancy-Aware Sensor Service Configuration	3
1.1.3 Policy Enforcement in Sensor Service Configuration	3
1.2 Thesis Outline	4
2. Background and Related Literature	5
2.1 Network Congestion Control	5
2.2 Service Oriented Sensor Service Configuration	8
2.3 Policy Enforcement in Service Configuration	10
3. Resource Management in Networks	12
3.1 Introduction	13
3.2 Events and Traffic	14
3.2.1 Event-Driven Traffic	14
3.2.2 Quality of Information (QoI)	15
3.2.3 Motivating Event-Driven Case Scenarios	16
3.3 Dynamic Price Based Routing (PBR)	17
3.3.1 The Approach	18
3.3.1.1 PBR Example	19
3.3.2 Dynamic Path Selection with Predicted QoI loss	20
3.4 Prioritized Random Oblivious Routing (PROR)	23
3.4.1 Experimental Evaluations	24
3.4.1.1 Experimental Setup	24
3.4.2 Fully Functional Network	26
3.4.2.1 Evaluating SPR, PBR and PROR	26

3.4.2.2	Comparison of PBR with SPR	27
3.4.3	Partially Damaged Network	29
3.5	Price Based Routing for Network Resource Management	33
3.6	Concluding Remarks	35
4.	Service Configuration in Constrained Environment: Data Relevancy	36
4.1	Introduction	36
4.2	Application Scenarios for Relevancy Aware Services	39
4.2.1	Information Relevancy Maximization Scenario	39
4.2.2	Wildfire Modeling	40
4.3	Service Configuration with Spatial Relevancy	41
4.3.1	The Coverage Model for Relevancy	41
4.3.2	The Price of Relevancy	42
4.3.3	Cost Based Optimization of Relevancy	43
4.3.4	Gain Based Optimization of Relevancy	45
4.4	System Design and Implementation	49
4.4.1	The WSN Framework	49
4.4.2	Service Provisioning	50
4.4.3	Service Maintenance	52
4.4.4	Modes of Configuration	53
4.4.4.1	Centralized Mode of Configuration	53
4.4.4.2	Distributed Mode of Configuration	54
4.4.4.3	Hybrid Mode of Configuration	55
4.5	Evaluation and Experimentation	56
4.5.1	Performance Comparison of the Configuration Modes	57
4.5.2	Results of Cost Based Model (CBM)	58
4.5.3	Results of Cost Based Model (CBM) vs. Gain Based Model (GBM)	60
4.5.4	ROI Based Service Configuration with Budget Constraints	62
4.6	Concluding Remarks	63
5.	Service Configuration in Constrained Environments: Policies	64
5.1	Introduction	65
5.2	Restriction Set-Theoretic Expressions (RSTE) for Service Configuration	67
5.2.1	The Semantics and Operations of RSTE Language	68

5.2.2	Definitions of the Sets	69
5.2.3	The RSTE Operations	71
5.2.4	Operations for Policy Restrictions	73
5.3	RSTE Implementation	74
5.3.1	RSTE Backtracking	75
5.3.2	RSTE Example	76
5.4	RSTE Evaluation	78
5.5	Dynamic Information Flow Control using RSTE in Multiple administrative Domain Environments	81
5.5.1	The Data Sharing Problem	81
5.5.2	Proposed Solution	82
5.5.3	System Design and Implementation	84
5.6	Concluding Remarks	84
6.	Conclusion	86
6.1	Congestion Control in Event-Driven Scenarios	86
6.2	Service Management of Sensor Services	87
6.3	Resource sharing using Policies	88
	REFERENCES	90

LIST OF TABLES

3.1 Priorities 26

5.1 Service Set 69

5.2 Node Set 69

5.3 Service Request Set 70

5.4 Policy Set 70

5.5 User Set 71

5.6 RSTE Respense Set 72

5.7 Service Response Set 72

LIST OF FIGURES

3.1	Route selection based on path price	19
3.2	PBR compared to SPR and PROR with 2 applications	27
3.3	PBR compared to SPR and PROR with 3 Applications	27
3.4	Delay ratio of PROR to PBR	28
3.5	Decrease in QoI loss in PBR vs. SPR	29
3.6	QoI losses for two priorities in SPR and PBR	29
3.7	Percentage decrease in packet delivery delay in price based routing compared to SPR under different mobility models and multiple applications . . .	30
3.8	Percentage increase in packet received in price based routing compared to SPR under different mobility models and multiple applications	30
3.9	Decrease in QoI loss in 5% damaged network	31
3.10	Decrease in delay and increase in packet delivery in 5% damaged network .	31
3.11	Decrease in QoI loss in 10% damaged network	32
3.12	Decrease in delay and Increase in packet delivery in 10% damaged network	33
3.13	Percentage gain over SPR under different mobility models and multiple applications in 30% damaged network	33
3.14	Decrease in delay and increase in packet delivery in 30% damaged network .	34
4.1	Sensor service configuration hierarchy	39
4.2	Sensor service configuration map GUI	40
4.3	Disk coverage model	42
4.4	Price with changing Alpha	45
4.5	Distributed mode of configuration	51
4.6	Node level information flow	52
4.7	Configuration times in various modes	57
4.8	Prices in CBM	59

4.9	Gain in Cost Based Model (CBM)	59
4.10	Price comparison in two models	60
4.11	Relevancy comparison in two models	61
4.12	Comparison of two models	62
4.13	ROI based service configuration for a given budget	63
5.1	RSTE layer in the system	67
5.2	Backtracking in for policy negotiation in case of overly restricted policies . .	76
5.3	RSTE Response Set entry for John Smith's request	77
5.4	Scalability comparison of RSTE based service configuration with WPML Based PASC	79
5.5	Dynamics of services availability with RSTE based policy enforcement and relaxation	80
5.6	Hybrid operator in a workflow graph	83
5.7	Hybrid operator and fabric node interaction	84
5.8	Detailed node level architecture	85

ACKNOWLEDGMENT

I am humbled to have reached towards the conclusion of my PhD degree, a learning process full of challenges, excitements, difficulties and rewards. It is an honor to acknowledge efforts and support of all those from around the world who stood by me from the very first day throughout my studies.

I would like to thank my thesis advisor Prof. Boleslaw K. Szymanski for his continuous support, guidance and encouragement through the course of my degree. The continuous supervision of Prof. Szymanski was vital in my professional as well as personal grooming. I would also like to thank Dr. Petros Zerfos for his mentorship and guidance as an external committee member. Dr. Zerfos's suggestions and feedback on my work has been instrumental in shaping and broadening my research horizon. I am also thankful to my internal committee members Prof. Malik Magdon-Ismael and Prof. Christopher Carothers for their valuable feedback. Along with my committee, I am also grateful to Prof. Carlos Varela for his supervision during early times of my degree.

I also owe gratitude to administrative and technical staff of the Computer Science Department for their help. In particular thanks to awesome people like Pamela Paslow, Shannon Carrothers, Katie Marie, Diane Torres for taking care of all the administrative arrangements. Thanks to all my friends and colleagues at RPI for making my stay enjoyable at RPI.

I am fortunate to have loving parents who have always been my source of inspiration. I will be always thankful to them for their tremendous unconditional, encouragement, support and prayers without which any achievement would have been impossible for me. I would also like to thank my brothers and sister for being there for me whenever I needed them. Hats off to my wonderful younger brother Ismael for taking care of the family when I could not be there because of my academic commitments. I am grateful to have marvelous wife, I am indebted to her for being there through the times of hardships and joys and taking care of our adorable little Uzair, thanks for keeping me going.

Last but never the least, I would thank some amazing people from CERN, Geneva, who encouraged me and utterly supported me in applying to graduate school. Gratitude

to awe-inspiring Dr. Andromachi Tsirou, who actually persuaded me to apply to RPI and have always stood by me, thanks Dr. Tsirou for your priceless beneficence. I would also thank other colleagues from CERN including Dr. Piero Giorgio, Dr. Duccio Abbaneo, Prof. Joseph Incandela, Dr. Jorma Tuominiemi, Dr. Frank Hartmann, Dr. Lorenzo Massetti, Patrice Siegrist, Shahzad Muzzafar, Imtiaz Ahmed and Prof. Arshad Ali (NUST).

ABSTRACT

Networks are becoming integral part of our lives. Services hosted on our mobile devices, network hubs and networked servers provide us with functionalities that were never so ubiquitous. These services are customized to our requirements more than ever and provides us with the intelligence and support that we need. Researchers are innovating cutting edge technologies and services to leverage the smart networked devices. With extensive functionalities the smart phones, tablets and other mobile and sensing devices are posing new challenges. These challenges span over various aspects i.e., from morphology of these networked devices to the mobility pattern of these devices and from policy enforcing the data shared by these devices to the data transport protocols and congestion control mechanisms.

With growing ubiquity of the mobile devices, the next generation service oriented systems face various challenges due to increasing number of services involved in each system and volume of data produced by them. The quality of the produced data or information can be sensitive to many application specific parameters and this Quality of Information (QoI) is represented by application specific utility function. Moreover, if a user is interested in getting information from a specific area then the information from or about such an area is highly relevant to the user's interest. This thesis focuses on three major challenges: (1) How to efficiently manage resources such as network routes, among multi-priority applications in resource-constrained scenarios, such as congestion, to reduce the loss of QoI; (2) how to configure and compose services that produce information which is relevant to the users interest? (3) How to share network resources efficiently by mapping human readable policies to low-level system constraints.

To address the first challenge, this thesis explores pricing mechanism for route allocation in event-driven scenarios. Multi-priority traffic suffers from loss of QoI in event-driven scenarios. Such scenarios trigger bursts of data when an event happens, e.g., sensors sensing a target. We research how to utilize pricing mechanism in routing and show that by dynamically pricing the network routes, they can be efficiently assigned to multi-priority traffic to avoid congestion thus lowering the overall QoI losses.

Sensor Networks, mobile devices and online services are producing huge amount of data these days. From plethora of such data sources, fusing data from most relevant sources to produce information that is relevant to the users interest is a challenging task. Thus, this thesis also explores how can we dynamically compose complex sensor services that produce information relevant to the users interest? We present three models to capture the concept of relevancy in composite services and present the design of a service-oriented system that uses these models to compose complex services from services hosted on sensor nodes, conforming to operational and relevancy constraints.

For better resource management and security of the network infrastructure, access to its resources is often regulated via policies based on user's credentials. Moreover, if the network infrastructure is shared among multiple organizations then resources are not always equally available to all the users. In order to regulate access to the network resources frameworks and mechanisms are needed to define, modify and enforce policies. In the last part of this thesis, we present a policy enforcement framework that enables resource-sharing among multiple parties in a shared network infrastructure. We explore how to enable end-users and policy makers to use human readable formats for system management and transparently transform human understandable policies to machine-level constraints for application configuration. We present a set language based approach to produce restricted set of resources for executable application configuration of a user request based on declared policies and user's privileges.

CHAPTER 1

Introduction

Modern day mobile gadgets have knitted us in so many networks that were not even existing in the past; one such network is certainly the communication network. With new smart devices, networks are defining new fields of science and uncovering new dimensions of our daily life. Although computer networks sprung more than half a century ago [1], the Wireless Networks are relatively recent. Mobile and wireless networks that connect mobile and sensing devices together and to the rest of the world offer extensive functionalities as a result of years of research. As a matter of fact such wireless networks impose unique challenges due to their fragile nature. Not only these networks have limited resources (e.g., limited bandwidth and battery life of the network devices etc.), they operate in very challenging physical environments such as forests, war zones, mountains etc., which makes the design and management of these networks much more challenging.

Sensor Networks are networks of sensing devices usually referred to as sensing nodes. These sensing nodes are networked together for various purposes ranging from environmental control to early warning systems in the battle fields. The sensing nodes and sensor networks have historically been used in the military, but lately the use of sensor networks has become widespread. With new sensing devices, wireless and mobile ad-hoc networks, these sensor networks are now embedded in our social fabric. When sensor nodes are connected via wireless networks they are commonly referred as *Wireless Sensor Networks (WSNs)*.

Networks today are much more dynamic than ever in the recent past. With increase in size, smartness and dynamism, managing networks is increasingly more challenging. The network management considered here aims at managing low-level network resources such as network routes and high-level network resources such as services and network assets. On the low-level, network management tries to optimize the delivery of data packets (routes) among network nodes and on higher level it performs service management which is tasked with making sure that the configuration of applications and network routes abides by the operational constraints imposed by the administrators (e.g. access policies,

resource utilization policies, etc.). Such network management needs to take care of many issues that were non-existent in the past. Network management today faces challenges such as near real-time network congestion resolution, on-the-fly network configuration, autonomous recovery, fault resilience, self-diagnostics and intelligent resource management. In this thesis, we explore techniques to dynamically manage network systems. We mainly focus on WSNs because they serve as a concrete target area for the application of our techniques. Moreover, WSNs are essentially resource constrained and are meant to operate unattended, therefore they impose challenges that we do not face in other types of networks. However, some of the techniques presented here are equally applicable to other networks.

In this thesis, we focus on some of the emerging research problems in context of network management of wireless sensor networks (WSNs).

1.1 Research Problem

The main focus of this thesis is on *Dynamic Management of Network Systems*. This thesis explores three important problems. First, we focus on lowest level of network management where routing packets produced by applications is addressed. In WSNs when an event of interest happens, burst of data is produced by applications hosted on WSNs; such bursts of data can cause congestion and thus increase packet delivery delays. Congestion management of routing packets produced as a result of occurring of events is challenging and arises very often in WSNs. We have tried to address this challenging issue via a reactive routing algorithm named *Price Based Routing*.

On the application level of WSN architecture, we focus on service configuration and management. This issue is particularly challenging as optimizing and providing services based on user defined constraints needs exploration of new models to satisfy user requests under user specified constraints. On application and user interface level, we focus on access control of network resources. This is a challenging problem as well because managing policies for shared resources in dynamically changing environment like WSN using human understandable language has not been extensively researched. The three main topics explored in this thesis are listed below.

1.1.1 Resource Management in Networks

We explore how to efficiently manage resources such as multiple routes in the network. We also explore how to efficiently route multi-priority traffic from one point to another. This work focuses on computer networks, but could be applicable to other types of networks such as road networks etc. The work is also applicable to evacuation route management for emergency situations. We present *Price Based Routing* for dynamically managing network routes in event-driven network congestion.

1.1.2 Data Relevancy-Aware Sensor Service Configuration

In sensor service configuration users might specify requirements and constraints on the requested service. Such constraints are generally called as operational constraints and include limitation on cost of service or energy consumption or combination of various constraints. However, user might also be interested in data from a specific area or location. In such case services producing data from/about such a location are more relevant to user's interest. Therefore, user can also specify constraints on the relevancy of the data along with operational constraints while requesting a service. In such a case, services should obey not only the operational constraints but also fulfill data relevancy restrictions. We explore how to dynamically configure networks based on the user defined data relevancy constraints. Composite Services that provide complex functionalities are configured and composed of various other services (component services), thus forming a hierarchical graph of services. While configuring such hierarchical services both the operational and data relevancy constraints on the component services need to be considered in order to optimize the requested composite service. Therefore, we also explore how to capture the data-relevancy aspect in providing hierarchically configured network services. We present various models for incorporating data and service relevancy requirements in dynamic network service configurations.

1.1.3 Policy Enforcement in Sensor Service Configuration

Under this topic, we explore mapping of high-level user requirements to low level system constraints and policies for sensor service configuration. We show how network assets and services can be shared among multiple parties via policies using our proposed

policy enforcement mechanism. Policies that regulate access to WSN resources in real-time situations, like rescue missions, change with on-ground situations, for example, resources that are not available to a user might be released to the user after changes occur in the on-ground situation. Therefore, the policy enforcement mechanism should be dynamic to accommodate such changes. Moreover it should be easily usable by user that are actually part of the real-time situation. In order to address policy enforcement in such dynamic situations, we propose a mechanism that on higher level uses human understandable language such as Controlled English to interact with humans and on lower level interacts with system level applications. We also explore how information can be shared on ad-hoc and dynamic basis among multiple administrative domains and heterogeneous platforms.

1.2 Thesis Outline

The remainder of this thesis is organized in the following chapters.

- **Chapter-2: Background and Related Literature:** This chapter presents summary of literature related to the work presented in this thesis.
- **Chapter-3: Resource Management in Networks:** This chapter discusses in more detail the issue of resource management in networks. It provides details on our proposed *Price Based Routing* mechanism and its evaluations.
- **Chapter-4: Service Configuration in Constrained Environment: Data Relevance:** This chapter provides details on researching principles for designing service oriented systems that can provide composite sensor services based on operational and spatial constraint.
- **Chapter-5: Service Configuration in Constrained Environments: Policies:** This chapter describes mechanisms for dynamic policy enforcement and mapping human readable user requirements to low-level system constraints for service configurations.
- **Chapter-6: Conclusion:** This chapter summarizes the research contributions of this thesis and proposes future directions in the areas explored in this thesis.

CHAPTER 2

Background and Related Literature

This chapter summarizes related work on the research presented in this thesis. There are three main sections of this chapter which present work closely related to our work described in the following chapters.

2.1 Network Congestion Control

Congestion control is well studied area of computer networks; however, congestion control in event driven scenarios is still an open challenge. Various routing protocols have been devised to efficiently route traffic to destination. Researchers have studied feedback loop based protocols and buffer control protocols to alleviate congestion. The congestion control techniques that use feedback loops to adjust data generation rate are not applicable to event driven scenarios because the event may disappear by the time rate is adjusted. Moreover, in scenarios where network is partially damaged, the feedback loop generated control traffic in the network may lead to lower throughput of the network. It is important to note that multi-path routing for congestion control has been studied in the past but most of the proposed approaches do not assume event driven traffic. Moreover, to the best of our knowledge none of the proposed approaches dynamically route prioritized traffic to multiple paths based on auction winning prices. Our route assignment approach is different from pure link delay based route assignment such as Etx [2]. We take into

Portions of this chapter previously appeared as: S. Y. Shah and B. K. Szymanski, “Price based routing for event driven prioritized traffic in wireless sensor networks,” in Network Sci. Workshop (NSW), 2013, pp. 1–8.

Portions of this chapter previously appeared as: Y. S. Shah and B. K. Szymanski. (2012) “*Dynamic Multipath Routing of Multi-Priority Traffic in Wireless Sensor Network*”. [Online]. Available: <https://www.usukitacs.com/sites/default/files/multi-path.pdf> (Date Last Accessed, Oct., 28, 2014).

Portions of this chapter previously appeared as: S. Shah, B. Szymanski, P. Zerfos, and C. Gibson, “Towards relevancy aware service oriented system in wsns,” IEEE Trans. on Services Computing (TSC), vol. PP, no. 99, p. 1–13, Oct. 2014.

Portions of this chapter previously appeared as: S. Y. Shah and B. Szymanski, “Dynamic policy enforcement using restriction set theoretic expressions (rste),” in IEEE Conf. on Military Commun. (MILCOM), 2014, pp. 198–203.

account the nature of traffic on the link, which is why a link may cause delay to a low priority traffic, but can be faster for high priority traffic. So each link has different delay for applications of different priorities.

A distributed auction based mechanism to resolve congestion in distributed manner is proposed in [3]. The approach is designed to deal with multiple applications of different priorities. In case of congestion, the node runs auctions to decide the winner packet that will be transmitted. Every candidate packet bids its predicted utility loss which it will incur, if it is delayed for a transmission slot. The packet of application with highest bid is chosen for transmission at each time slot. However, the approach proposed in [3] makes a local decision to get an allocation slot and the routing decision is not the focus of the work. Moreover, the authors also do not address congestion control in case of network failures. In [4], the authors propose a receiver-assisted congestion control mechanism using Transmission Control Protocol (TCP) [5]. In their approach the receiver estimates the transmission rate from packet inter-arrival time. The receiver then notifies the sender about the rate appropriate to avoid congestion so that sender can adapt its rate accordingly. Such a rate control mechanism or other TCP based approaches are not feasible for event driven traffic as it may take time to adjust transmission rate using feedback loops and control messages.

In buffer management technique as proposed in [6], nodes maintain buffer states which are piggybacked to their neighbors. The buffer state of a node tells its neighbors if they should send packet to it. A non-full buffer state of a node signifies that it can receive packets. In such a way, packet transmission rate in whole network is adjusted. The authors have also incorporated multi-path routing in their mechanism but in their approach the priorities are statically assigned based on link conditions. A problem with such a multi-path approach is that in case of congestion, all the nodes will send packets on link with less delay and thus overload it. Moreover, this approach treats all packets as of same value which may not be the case in event driven prioritized traffic.

In [7], the authors present a congestion control technique for prioritized traffic that uses piggybacking of congestion notification embedded in packet's header. Each node is assigned certain priority index. Once congestion is detected, bandwidth proportional to the priority index of the node is allocated to the traffic generated by that node. This

approach also uses rate control mechanism to avoid congestion and may suffer from issues mentioned earlier for rate control approaches. The disadvantage of such assignment is that a low priority node may receive packets that are of high value that will incur large delays due to node's low priority. A prioritized multi-path approach is suggested in [8] to minimize end to end delay. In this approach, every node has multiple shortest paths to destination. Sink propagates back end-to-end delay on link between each node and sink, thus establishing a graph where every path has associated delay. The paths are assigned to packets in order of their priorities, where highest priority packets get routes with lowest delay and so on. In such a way, paths are statically allocated to different priority packets based on delay. Since paths are statically assigned and the delay on paths may change overtime (which is highly expected in event driven scenarios), statically assigned paths are not feasible for event-driven scenarios.

Route recovery protocols to alleviate failures have also been studied in the past. In [9], authors propose a protocol which detects the network failures by packets losses from the sender nodes. Once the failure is detected and failed nodes are identified, the routing interval is increased to notify the neighboring nodes of the failure. After that, alternate paths to the child nodes of the failed parent nodes are discovered which are then used for further communication. In such an approach, alternate routes are discovered only in case of failures. A path with congestion is still considered usable and the traffic is not rerouted in such cases. In our approach, we not only reroute packets in case of failures, but also in case of congestion. Moreover, routing packets from multiple applications of various priorities, which we are focusing on, is not addressed in this reviewed approach. Other multi-path routing approaches based on link weight and dynamic link weight proposed in [10, 11] are highly dependent on the quality of weight assigned to the links. Oblivious routing approaches have also been devised in the past for congestion management. In such approaches, the packets are first sent to a random node before sending it to actual destination to distribute traffic in the network [12, 13].

Research community working on building evacuation and road traffic congestion management in event driven scenarios is facing similar challenges as computer network researchers. In [14], researchers have devised navigation system to enable efficient evacuation of buildings in case of emergency. In their approach two kinds of nodes (sensing

and decision nodes) facilitate the evacuation of habitants from a building. Based on information collected from sensing nodes, decisions nodes compute effective shortest path to guide people out of the building.

2.2 Service Oriented Sensor Service Configuration

The service oriented architecture and service composition on resource rich platforms has been well studied and has been present in the literature for quite some time. However, the area of service oriented WSNs is getting significant attention only recently. Researchers are already investigating SOA for sensor networks but still the area is largely unexplored. Various aspects of the service oriented WSNs have been most studied so far include service composition and service management but not spatial relevancy. In this section we describe the most relevant work that has been done in the areas relevant to this thesis.

Service composition for web service have been explored in the past [15–17]. Reference [18] provides an overview of techniques used for web service composition. In [19], the authors have proposed cost effective algorithms for mapping services in clouds. They have addressed the service mapping problem as a service composition problem. In the researched approaches a service is composed from various other web services hosted on servers in different locations. Such approaches are suitable for world wide web but WSNs are different in terms of network structure, computing resources and computing models. The web service compositions are designed based on some salient requirements, their functionalities are predefined and their composition relies mostly on a static graph of different functionalities. Even though the functionalities can be dynamically bound, the overall functional capability of the system is restricted. Researchers have also explored workflows for service composition [20–22]. Reference [23], also utilizes workflow representation for service compositions and presents an approach that is similar to web service composition. Workflows are typical example for static composition of services and are very useful in situations where the composition graph does not change during service execution. The nesC [24] was an effort to make component based system for mobile devices by connecting various reusable components. The approach is similar to CORBA [25] that was proposed for distributed system. However, in nesC the graphs and trees that are used

to bind different components of the system are static and predefined before the system can start working. This is why the nesC is missing dynamic reconfiguration capabilities which is one of the main features proposed in our work. Model for mobile services proposed in [26] suggests connecting services that are semantically and syntactically similar. The model is oriented around mobile users. Change in location of a mobile user initiates connection/disconnection with a server and triggers composition of services which may not be the case in WSN system. The approach does not compare costs or attempt any spatial relevancy check to find out most appropriate services from a bag of available services. In our approach, the system takes into account the cost and relevancy of services. Moreover, the composite service graph is dynamic and automatically reconfigures in case any changes occur in the network.

A closely relevant work in [27] proposes dynamic composition of service. The work focuses on composing services from various other services but the work does not consider spatial relevancy of the services during composition. The authors assume in their work that all the services that provide certain output have the same relevancy. They do not consider user's desired area of interest in their composition algorithm, the composition is done based on functional capabilities of the services. In contrast, in our work the spatial relevancy has a major role in the resultant composite service. Moreover, the authors have tested their centralized composition approach in a simulated environment and have provided theoretical results for the distributed version. We provide fully emulated centralized, distributed and hybrid sensor service configuration with full awareness of the geospatial locality of the service. To the best of our knowledge, this problem with all the aspects that we explore in this thesis has not been investigated in the past. Service composition for MANETs has been researched in [28], however the work does not investigate the aspects we are looking at, such as controlled cost, service relevancy, service configuration management and emulation.

In [29], the authors have considered problem of information provider selection and have presented numerical results. However, the authors do not consider the relationship among the providers and do not address the problem of sensor service composition and management. In their work, the authors has focused on only on selection of services based on their Quality of Information (QoI). The authors do not address the relevancy

of composite services in a sensor service graph which we have addressed in this thesis in service configuration process. In addition, we provide different models for capturing the relevancy in hierarchical services. These models enable users to achieve the suitable balance of cost and relevancy for their applications.

Researchers have also explored the coverage problem from perspective of sensor deployment and sensing algorithms. In [30], a good survey, of work on coverage in WSNs, is presented. The literature that focuses on coverage provided by the complete WSN and tries to cover maximum possible area with the sensors. In our work, we focus on dynamically configuring complex services; these complex service in our case might use only few of the sensors (that are valuable to user's request) of the complete WSN to provide relevant information about a small part of the complete area covered by a WSN.

Most of wireless sensor network research (e.g., [31–34]) has historically been simulated or experimented either using TinyOS [35] or some specialized hardware software stack. We employ a novel technique of emulating WSNs applications. Our testbed not only enables network emulation but also enables integration of emulated network with real time network.

2.3 Policy Enforcement in Service Configuration

Asset sharing and management in coalition environments has been studied in the past and various techniques for resource allocation have been suggested by researchers [36, 37]. Policy enforcement and management techniques for different computing paradigms have been proposed in the past [38–40]. In [39, 40], more programatic and standardized approaches have been proposed, however, the policy description and representation is closer to computer programs thus not very friendly to non-technical users. Goal oriented and formal logic based frameworks have also been proposed in the past for policy management [41], but their capabilities are limited by the underlying OWL version. Other semantic representation based frameworks such as [42, 43] have also been proposed. Deontic logic based approaches to address paradoxes in Standard Deontic Logic (SDL) have been proposed in [44], however, our approach differs from deontic logic based approaches and other formal logic based approaches such as [41, 45] in policy representation as well as enforcement because the latter are not feasible for dynamic compositions of services.

In [46] attribute based policy enforcement has been investigated; this approach also utilizes [39] and has more programatic syntax for policy representation, moreover, it does not support backtracking and suggestions for policy negotiation and relaxation.

CHAPTER 3

Resource Management in Networks

Resource management in networks is a challenging issue. Researchers from various fields including networks have been working on issue of resource management [11, 47, 48]. This chapter primarily focuses on how to efficiently utilize the available bandwidth of a network and minimize delays resulting from congestion in Wireless Sensor Networks (WSNs). Networks can have many different topologies as well as traffic patterns that differ from network to network. Traffic in some networks is more continuous than others; also one kind of traffic may tolerate delays others (e.g., event-driven traffic) may not, therefore, the resource management mechanism needs to be carefully designed, so that it can better meet the requirements of different kinds of traffic flowing in the network.

Event-driven traffic poses new challenges for bandwidth allocation and congestion control. In an event-driven traffic, the occurrence of events is mostly unpredictable, therefore networks observe bursts of traffic at different times due to occurrence of events. Traditional approaches for congestion control and bandwidth allocation in WSNs do not take into account such bursts of data and address the bandwidth allocation problem assuming continuous flow of packets with all packets of equal importance. They do not differentiate packets generated at the time of event from the more predictably generated packets. This chapter specifically targets event-driven scenarios and provides mechanism to improve performance of the network in such situations. The *Price Based Routing (PBR)* mechanism proposed in this chapter also addresses the multi-priority traffic challenge [49, 50]. The mechanism is explained for WSNs, but it is also applicable to cyber-physical systems and road networks, examples of which are briefly discussed in the later part of the chapter. The chapter also introduces *Prioritized Random Oblivious Routing (PROR)*, a multi-priority version of Valiant's randomized routing algorithm [12] and compares it

Portions of this chapter previously appeared as: S. Y. Shah and B. K. Szymanski, "Price based routing for event driven prioritized traffic in wireless sensor networks," in *Network Sci. Workshop (NSW)*, 2013, pp. 1–8.

Portions of this chapter previously appeared as: Y. S. Shah and B. K. Szymanski. (2012) "*Dynamic Multipath Routing of Multi-Priority Traffic in Wireless Sensor Network*". [Online]. Available: <https://www.usukitacs.com/sites/default/files/multi-path.pdf> (Date Last Accessed, Oct., 28, 2014).

with *Price Based Routing (PBR)* .

3.1 Introduction

In the research literature, various techniques have been proposed for congestion resolution to minimize delays. Most of the proposed approaches focus on congestion resolution for continuous stream of data in which some of the data is lost in congestion during congestion resolution or rate adjustment phase. In event-driven traffic case, such techniques might take longer to converge to the solution than the actual duration of event and important data related to the event may get lost before congestion is resolved or data rate is adjusted. Moreover, there might be traffic of various priorities and data traffic from high priority applications should be routed with lowest delay. For example, data feed from various sources may have different importance and thus different priority. The priority of an application is driven by various factors and may change over time depending upon the application and the network situations. Often times, the priority reflects tolerance to the delays and importance of the application to the end-user, applications with less tolerance to the delays have higher priority.

Routing prioritized traffic in event driven situations is relatively less explored area. However, researchers have recently started exploring this area in the context of safety and emergency services [9, 51]. In this chapter, we propose mechanism to efficiently deal with congestion caused by event-driven traffic. Our proposed mechanism specifically addresses the issue of multi-priority traffic in event driven scenarios and lowers delays by distributing traffic from different applications over separate paths. Such distributed transportation of traffic is of high importance in public safety and emergency scenarios in which event-driven traffic is generated at large and robustness against failures caused by such situations is required. Emergency situations, such as earthquakes, floods or fires etc., can also damage the communication infrastructure, therefore routing algorithms should be able to deal with such partial failures of networks and keep the *Critical Infrastructure and Key Resources (CIKR)* connected. We show that in our mechanism, the packets intelligently choose time efficient paths for their route to the destination and can dynamically re-route in case of partially damaged network.

Traditionally, ad-hoc routing protocols route packets via shortest path to destina-

tion, e.g., DSDV (Destination-Sequenced Distance Vector) or most efficient source route, such as Dynamic Source Routing (DSR) [52] etc., route packets via shortest or most efficient path. When such protocols are used for routing prioritized traffic generated by multiple applications, traffic from all the applications go via the same shortest or most efficient path. Such a *Shortest Path Routing (SPR)* makes shortest paths congested as well as delay high priority traffic as much as low priority traffic. Priority based assignment of paths has also been suggested in literature, but such fixed assignment [8] can lead to starvation of low priority traffic, because the high priority traffic always gets the routes and when traffic belonging to certain priority subsides the paths assigned to that priority are under utilized. In contrast, routing protocols that are designed for ad-hoc or on demand routing need to update the paths frequently. Such frequent updates unnecessarily exhaust battery, and depending upon the size of the network the updates can take long time to complete.

In the past, different approaches have been proposed for differentiated services. Various multi-path approaches dedicate routing paths to applications based on their priorities. Such static allocation may lead to under utilization of the network because in many sensor network applications the traffic pattern constantly change. It is often hard to predict volume of data transfer between network nodes specially in event-driven scenarios. Therefore, if shorter paths are statically assigned to high priority traffic and at time t there is minimal or no high priority traffic in the network but huge volume of low priority traffic, the low priority traffic will incur huge delays whereas shortest paths remain underutilized. Therefore, there is a need for a reactive routing mechanism that can respond to the varying state of the network by dynamically adjusting routing paths for multi priority applications. Multi-path routing is also another widely used technique for congestion resolution and increasing utilization of the network, in the *Related Work* chapter, we have discussed the proposed approaches and their limitations.

3.2 Events and Traffic

3.2.1 Event-Driven Traffic

Event driven traffic is the traffic generated when an event of interest to the user of the network happens. Suppose, a sensor network is running target tracking application to

track an object of interest, when the object enters the sensing range of sensing nodes, the nodes start sending object trajectory data, once the object leaves the sensor network which marks the end of the event, the sensing nodes go to sleep state sending occasional packets. In case of cyber-physical systems, the event driven traffic is the burst of people evacuating a building in an event of emergency or a burst may consist of large number of cars leaving a disaster stricken area. All such scenarios, produce event-driven traffic and the load in the network usually subsides when the event finishes. The nature of event driven traffic makes them different from a continuously flowing traffic because event-driven traffic is bursty and can produce short lived congestions. In case of sensor networks the event-driven scenarios produce bursts of packets that produce congestion, also in emergency evacuation scenarios burst of people leaving a building generate congestion on exit routes. There are two possible consequences of the events in event-driven scenarios;

1. The event has passed and traffic in the network is back to normal and the congestion has disappeared.
2. The event has passed and traffic in the network is back to normal but the congestion still exists, the event has had physical effects and partly damaged the network. In such a case, the network has now limited number of nodes that are able to route the traffic, therefore although the event has passed the congestion still remains.

3.2.2 Quality of Information (QoI)

In data communication networks such as in WSNs, the packets carry information from source to the destination. *QoI* is the measurement of the utility that the packets bring to the end-user. The *QoI* can be represented by a utility function that are specific to the application. The *QoI* loss is the loss that the quality of the information incurs when packets carrying them are delayed. The gain in *QoI* is complex to measure, but we can measure *QoI* loss as consequence of delay in the network. The *QoI* can be very sensitive to the timeliness. Moreover, the *QoI* cannot increase, it can only decrease with the passage of time, so it is reasonable to measure the *QoI* loss.

3.2.3 Motivating Event-Driven Case Scenarios

Event-driven scenarios are often observed in our daily life in our office buildings, houses, road networks and surveillance systems. Following are some of the scenarios that motivated the research presented in this chapter.

Various cities often host festivals and games in which large crowds of people participate. Imagine a city hosting a festival that attracts large audience. At the end of the festival, the festival area empties and participants try to leave through their nearest exits and via shortest paths. There are limited number of exits. Moreover, some exits might be closer to car parking lots/garages or bus stations which, if not controlled, will draw more crowd. Therefore, such exits are more prone to congestion and hence any mismanagement of these exits can easily cause stampede. However, all exits can be monitored via video cameras and exit ingress and egress ratios can easily be computed. The movement of the people provides an estimate of how fast people are passing through the exit. Thus, our mechanism can predict congestion by dynamically computing the total passage time on each exit route. Using this information, electronic displays strategically placed at cross-paths can guide dynamically groups of people to an exit that currently offers the fastest passage time. Displays will periodically update their recommended route as the exits change their passage time according to the numbers of people passing through them, thus avoiding stampedes. Such event-driven routing guidance and management of physical infrastructure can save human lives. Recently, a stampede in a music festival in Germany killed a number of people, leaving scores of people injured. The stampede occurred when the crowd departing festival followed the shortest path and the exit was too narrow to cope with huge traffic [53]. An early divergence of people to different exits and effective monitoring could have saved precious lives. Unfortunately, such disasters are quite frequent [54,55], as evidenced by [56], that reviews major disasters in mass gatherings over past decades caused by stampedes and mismanagement of evacuation routes [49].

A more WSN oriented scenario is the target tracking scenario. Consider vehicles moving on roads in a city and being tracked by wireless sensors e.g., video cameras stationed at different locations. The cars move in and out of the sensing areas of different cameras. The target trajectory information is sent over the wireless network to the surveillance center where it is displayed to the security agencies of the city. Depending upon

the driving pattern of the cars the security chief assigns priorities to suspicious vehicles to keep close eye on them. Whenever the target vehicles enter the sensing range of the camera, the camera starts sending large number of packets to the monitoring center to provide precise information about the location of the vehicles. When the target vehicles leave the sensing range of the camera, the camera switches back to normal mode and sends low fidelity data to control center. Similarly, every camera stationed in the sensor network produces large number of packets whenever target vehicles enter their sensing range. Such bursts of data create congestion on downstream nodes in the sensor network causing packet loss and increased delay in packet delivery [49].

Above mentioned are two motivating scenarios, but there are other similar scenarios in which event-driven traffic with different priorities need to be transmitted from one point to another with minimal delay. In cyber-physical systems, prioritized evacuation includes hospital evacuation, disaster stricken area evacuation and distributed transmission of traffic in road networks to decrease load on heavily used highways. There is a need for a mechanism that is capable of handling event-driven traffic in different scenarios and can segregate multi-priority traffic. Because some traffic might be more sensitive to delays, therefore it should given more importance without starving low priority traffic. The mechanism should be capable of distributing the traffic in the network to utilize bandwidth on various routes of the network. Such a mechanism should also be resilient to network failures such that it can transport traffic in case of devastating events that may damage the network infrastructure.

3.3 Dynamic Price Based Routing (PBR)

Motivated by scenarios described in section 3.2.3, we propose a *Price Based Routing (PBR)* mechanism that addresses the routing issue in event-driven traffic both in data communication networks and cyber-physical networks. In the following section we explain our mechanism by considering target tracking scenario in sensor networks. Later we show how the same mechanism can be used in other types of networks.

3.3.1 The Approach

In order to maximize the network throughput and decrease transmission delay for all the applications, we propose the following mechanism. Whenever a router has a packet of certain application to deliver to the next hop towards destination, it looks at all of its neighbors which can lead a route to the destination without looping. The router then uses the price for each path to estimate the total delay that the packet will encounter if this path is selected. Then, the router sends the packet on the path on which the delay is the smallest. An auction is held and path with lowest price is selected and packet is forwarded to the neighbor leading that path to destination.

The path prices are proportional to the predicted time delay for a packet of a unit priority taking the path. This time delay is a function of the path congestion. The path price can be translated into waiting time of a packet with a given priority, but the mapping depends on routing and transmission protocols and technical details of the network used. It is important to notice that path price computation is based only on information obtained from the immediate neighbors of the router. Each router calculates prices for route originating from each of its neighbors but not from all nodes in the network. The same processing repeats on every hop and a minimal cost route is selected on hop by hop bases which leads to overall minimal delay path.

A path may have different delays for applications with different priorities. A path that imposes long delay for a low priority application may impose short delay for a high priority application which eventually leads to segregation of different types of traffic in the network. Moreover, as the state of the network changes over time and congestion appears and disappears at different points of the network, the path prices follow and change dynamically. This is why in our proposed mechanism paths are not statically allocated to different traffic categories and traffic pertaining to an application does not follow a single path but changes its path according to the network conditions. As our protocol routes each packet to next-hop which leads to lowest cost route at each step, there is a chance that the packet gets into a routing loop, in which it keeps oscillating between some nodes and never reaches the destination. In order to avoid such routing loops, each node checks the number of hops to destination from its neighboring nodes. The candidates for next hop are restricted to those neighboring nodes which have fewer hops to destination than the

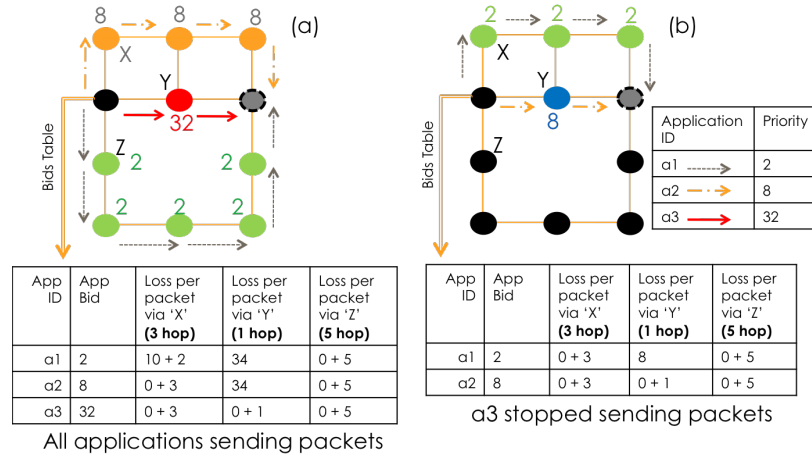


Figure 3.1: Route selection based on path price

current node. Thus the packet always moves forward towards destination (downstream movement) strictly decreasing its number of hops to destination and avoiding routing loops. It is important to note that the packet may choose longer path over shorter available path but in both cases the next hop has to have fewer hops to destination than the current one to avoid routing loop. In our protocol, the main goal is to find the path with minimal cost, therefore each node has a destination distance table which contains distances to destination via its neighbors. This table can be easily populated only once using Dijkstra's algorithm or any other path finding algorithm. With such a destination-distance table, every node knows its distance in hops to each destination. This requires storing only $O(n)$ data in each node. However, in sensor networks only a subset of nodes can be destinations, so the storage requirement usually reduces to $O(1)$.

3.3.1.1 PBR Example

We demonstrate the *Price Based Routing* using an example network. Figure 3.1(a) shows a simple network with three applications of priorities represented by number 2, 8 and 32, where higher number represents higher priority. Packets generated for three applications are routed to the same destination. There are three routes of different lengths from source to destination (i.e. via nodes X, Y and Z). In such a simplistic scenario the routing protocol works as follows. At each routing node (or router), if there are packets competing for transmission slot, they bid for the right to use this slot. The highest bid wins and becomes the path price for the path originating from this node. The initial bid

of each packet is its priority. Each packet losing the auctions increases its bid for the next auction by adding to the old bid its priority. For nodes on which the packet does not compete or wins auction, we assume the loss is only “1” for pass through the node. For this scenario, the application bids and losses are shown in the Figure 3.1. As shown in tables in Figure 3.1, going through each node i.e., X, Y or Z, there is loss associated with taking that path, which in this case is price for taking the path. If we look at the bids table, ‘a3’ has the highest bid (so can pay high prices) and therefore lowest loss if its packet is routed via node ‘Y’, hence its packets go through the shortest path. Slightly longer paths are taken by applications ‘a1’ and ‘a2’. Node ‘Y’ that leads to shortest path to the destination will be avoided by applications ‘a1’ and ‘a2’, because packets from ‘a3’ which are high priority will most of the time win the transmission slot on node ‘Y’ and other applications would need to wait till through the auction losses they accumulate a bid with which they can win auction on node ‘Y’, losses needed to win slot on each node are shown in the table. This is an unnecessary contention as other routes to destination are available. Therefore, our mechanism wisely sends packets of other applications through alternative routes as shown in Figure 3.1. If the network state changes and for certain period there are no more packets of application ‘a3’ flowing through node ‘Y’, routing changes. The winning bid on node ‘Y’ changes as the router automatically reconfigures the routes. As seen in Figure 3.1(b), packets of ‘a2’ which have the second highest priority go through the shortest path and packets from ‘a1’ go through the route previously used by ‘a2’. Thus, the routing nodes dynamically reconfigure paths based on the network conditions indicated by prices on neighboring nodes. Consequently, the routes are not underutilized like in the case of statically assigned routes.

3.3.2 Dynamic Path Selection with Predicted QoI loss

In section 3.3.1.1, we described the *Price Based Routing* protocol for a simple scenario in which *QoI* function for every application is equal to their priority. In this section, we show how the *Price Based Routing* works for situation in which the *QoI* loss is proportional to both priority and timeliness of the information. Suppose there are several object tracking applications hosted on a WSN, monitoring different objects of different priorities. Every application has a utility function which represents its Quality of Informa-

tion (QoI) that decreases proportionally to the product of the accumulated communication delay and priority of the applications. Every time unit, the router transmits one packet. We incorporate auction mechanism described in [3] for selecting winner of the transmission slot. Assuming TDMA as medium sharing protocol, application i bids its *QoI* loss represented by the following utility function.

$$Bid_i = \Delta t \times p_i \times t_c \quad (3.1)$$

In equation (3.1), p_i is the priority of the application i , t_c is the TDMA cycle-time and Δt represents the difference between packet generation time of current bidding packet and the packet that the router transmitted previously for application i . Please note that the Δt increases as consequence of losing auctions and therefore the bid also increases for application i . This way we prevent starvation of the low priority packet, because with each losing auction their bid increases. Whenever there is congestion on the routing node, the auction winning bid increases and applications with higher bids go through while applications with lower bids have to wait and compete in upcoming auctions. In case a node receives new packet for an application for which it already has packets waiting for transmission, the old packets are replaced with new ones in the queue as transmitting fresher packets will prevent more *QoI* losses compared to sending older packets with or without new ones [57].

Once the winner of transmission slot is selected, it needs to be transmitted to one of the downstream neighbors. As discussed above, as congestion arises on a downstream node, the winning bid increases on that node. In this scenario, the auction winning bid acts as the price of a path originating in this node because the packet has to wait on the node and win auction before it can take path originating from this node. Using this price, we can estimate the transmission delay on routes originating from this node. Use of TDMA protocol with its transmission cycles makes the path delays well predictable, but other techniques for delay prediction can be used for protocols other than TDMA. The path price divided by the packet priority in this case represents the number of TDMA cycles that the packet needs to wait until it gets a transmission slot for the next hop.

In other words, when a packet attempts to make the next hop, it may need to compete with packets from other applications for transmission slot on that node. This com-

petition can be based on any mechanism, here we assume auctions as described above to estimate waiting time on the node. In order to win auction on this new node, the packet has to wait and accumulate *QoI* losses proportional to packet's application priority before it can win the auction and get the transmission slot. This is important to note that, if the next-hop is congested and high priority packets are bidding in auction, then a low priority packet will have to wait longer and suffer bigger delays before it can make up a winning bid. Therefore, selecting next hop for a packet is crucial and can lead to longer delays if low priority packets join auction with high priority packets. We have noticed that this problem arises when all the packets try to go through the shortest path and end up in high *QoI* loss. Our model intelligently tackles this situation by separating high priority packets from low priority packets. In our model each packet compares its priority with the winning bids on the candidate next-hop; from this comparison, it can calculate how long it would need to wait until it wins auction for the transmission slot. Moreover, it also takes into account how many more hops it needs to travel from the next hop onwards, before it can reach the destination. This gives us predicted delay on the candidate next-hop and thus on the path to destination originating from that hop. The path prices are reflection of such delays. The following equation represents path price denoted as P_i^k on neighbor node k for application i .

$$P_i^k = H^k \times \left(\frac{Hbid^k}{Mbid_i^k} + \frac{1}{2} \right) \quad (3.2)$$

$$Next\ hop = k\ with\ \min P_i^k \quad (3.3)$$

where H^k is the number of hops to destination from neighbor node k , $Hbid^k$ is the price i.e., the highest or winning bid on node k , $Mbid_i^k$ is the bid of the application i (hence the bid of the packet seeking the next hop) on node k . There is an underlying assumption that a node overhears its neighbor's highest bid at the time this neighbor sends forwarded packet for application i that enables calculation of $Mbid_i^k$ on node k . These parameters can be embedded in packet's header in forwarding procedure. This way, whenever one node forwards packet to another, the neighboring nodes (which may not be the destination nodes for this packet) can also read these parameters from header. This is a reasonable

procedure because all the neighboring nodes need to read header for destination address, to find out if the packet is destined to them. Therefore reading two extra parameters will consume little energy and processing power. There are efficient overhearing techniques already proposed in the literature which can be incorporated for overhearing.

In our case, the *QoI* function is linear in priority, therefore in our scenario, the ratio of the application i 's bid to the winning bid on the neighboring node acts as good predictor of waiting time for transmission slot. Consequently, we can predict the waiting time on next node using $H^k \times (\frac{Hbid^k}{Mbid^k})$, because depending upon what is the highest winning price on the next node, the packet will need to wait accordingly. The average future delay without any congestion is $H^k/2$. In other words, the highest price in auction on the next node translate to the waiting time on that node. The same mechanism can be easily generalized for other *QoI* functions based on the priority of the application, so the prices on the next node can be estimated. Another important point to notice in equation (3.2) is that as more and more packets follow certain path, the winning price on that node will increase and this price will reflect the predicted delay on the node. Eventually, routers will start diverting low priority applications packets to other nodes where the auction prices are lower. In such a way, the dynamic prices on nodes will automatically redirect the traffic as congestion increase on the nodes.

In our proposed mechanism the starvation of the packets is avoided by using aging effect. For each application, the bid is defined by the priority as well as the waiting time. So, as the packet of an application loses auctions, its loss of *QoI* increases, increasing the bid of the application. After losing some auctions, the application bid eventually rises enough (due to aging) that it will beat other (even high priority) applications in the auction and the corresponding packet is transmitted.

3.4 Prioritized Random Oblivious Routing (PROR)

The *PBR* algorithm is oblivious to the state of the complete network, it decides paths for packets based on information from local neighbors. Therefore, we think it would be useful to compare the *PBR* with oblivious routing approach. In this section, we present a prioritized version of the Valiant's random oblivious algorithm [12]. The original random oblivious routing presented in [12] does not account for multiple applications with

different priorities. Hence, we revised the original random oblivious algorithm to apply it to multi-application scenario in which applications have different priorities. Algorithm 1 presents the random oblivious algorithm for multi-priority applications. In the algorithm 1, $jumpUL$ is the upper bound on distance from current node. The number $Z_{(2,jumpUL)}$ is a random number drawn from uniform distribution between 2 and $jumpUL$. The function $getNode(Z_{(2,jumpUL)})$ returns a node that is Z hops away from the current node. Each packet Pkt_i goes through shortest path with probability $Prob_i$ and with probability $(1 - Prob_i)$, it is sent towards random node $RandNode_i$ which is Z hops away from current node ($Node_k$). From there it is sent towards sink. The probability for an application i is calculated using Eq. 3.4, where n is total number of applications. In this approach *SPR* (*DSDV in this case*) is used as default routing algorithm for sending packet to intermediate random node and finally to the *Sink*.

$$Prob_i = \frac{P_i}{\sum_{i=1}^n P_i} \quad (3.4)$$

3.4.1 Experimental Evaluations

In order to evaluate our mechanism, NS2 [58] simulator was used to implement our proposed protocols represented by *PBR* and *PROR*. We compared performance of these protocols with the shortest path routing (*SPR*) represented by the native DSDV protocol in NS2 distribution.

3.4.1.1 Experimental Setup

In our experiments for WSNs, we use *TDMA* as MAC level protocol and also used *TwoRayGround* as radio propagation model. We simulate the target tracking scenario presented in section 3.2.3 using nodes in NS2. We used 44 nodes one of which is destination node (sink node) and multiple mobile objects served by different tracking applications. These mobile objects were represented by NS2 nodes with routing and sensing capabilities disabled. During simulation, the mobile nodes move in the sensing field to mimic real life objects (e.g. cars). We experimented with number of objects varying between 2 and 4. The sensing nodes were fixed for all experiments whereas the mobile objects moved

Algorithm 1 Prioritized Random Oblivious Routing on source node $Node_k$

Require: $jumpUL > 2$ /*upper bound on random jump*/

```

Pkti ← Packet for application i
DPkti ← Final Destination of Packet Pkti
if (Pi is generated by Nodek) then
  RandNodel ← getNode(Z(2,jumpUL))
  X ← UniformDistribution(0,1)
  if ( $0 < X \leq Prob_i$ ) then
    DPkti ← Sink
    Send Pkti on shortest path to DPkti
  else
    DPkti ← RandNodel
    Send Pkti on path to RandNodel
  end if
else
  if ( $D_{Pkt_i} = Node_k \&\& Node_k \neq Sink$ ) then
    DPkti ← Sink
    Forward Pkti on path to DPkti
  else
    Forward Pkti on path to DPkti
  end if
end if

```

around with specified mobility patterns while being tracked by fixed sensing nodes. For our experiments we simulated three different mobility models: “Random Waypoint Model (RWP)”, “Pursue Mobility Model (PMM)” [59] and “Manhattan Grid Model (MGM)”. We ran simulations for 500 seconds for each mobility model with different priorities shown in table 3.1. We have used auction mechanism for slot winner selection for all the protocols. In simulation of all the protocols i.e., *SPR*, *PBR* and *PROR*, the winner of transmission slot is selected using auction mechanism [3], the packet is then routed using *SPR*, *PBR* or *PROR* protocol. The importance of an application increases with the number that represents that priority the application. The priority of an application increases with the number that represents it and is set by the consumer of the information. The results presented in this section are averaged over 7 different runs.

We tested our *PBR* for target tracking event-driven scenario with two cases;

1. situation in which event happens and network is fully functional
2. situation in which the network is damaged and is partially unavailable while target

Table 3.1: Priorities

#Applications	Priorities
2	(5,10), (9,36), (8,64), (16,64), (27,216), (4,16), (10,20)
3	(5,10,15), (9,36,81), (8,64,216), (16,64,144), (27,216,729), (4,16,36), (10,20,30)
4	(5,10,15,20), (9,36,81,144), (8,64,216,512), (16,64,144,256), (27,216,729,1728), (4,16,36,64), (10,20,30,40)

tracking being carried on by the WSN.

In the following sections (3.4.2, 3.4.3), we show how our system performs well in both the situations.

3.4.2 Fully Functional Network

3.4.2.1 Evaluating SPR, PBR and PROR

We evaluated the *PROR* and compared the *QoI* losses of various applications in target tracking scenario. We evaluated the protocols with three different mobility models. We noticed that *PROR* incurs higher losses as compared to other models in the target tracking scenario. We believe this is because *PROR* sends low priority packet through longer paths even though there may not be congestion on shortest paths. Moreover, since we are simulating the target tracking scenario in which the sources of the packets change as the target moves in the sensor field, every source node routes traffic through longer paths thus increasing packet delivery delays. The results show that *PBR* performs better than *PROR*, as we can see from the Figure 3.2 and Figure 3.3, the *PBR* incurs lower losses and thus has higher gains over *PROR*. The gains are calculated in same way as shown in Eq. 3.5.

We also measured the end-to-end delay d and computed the *delay_ratio* which is the ratio of delay in *PROR* and *PBR* i.e., $\frac{d_{PROR}}{d_{PBR}}$, shown in Figure 3.4. The *delay_ratio* > 1 means that *PROR* incurs higher delays as compared to *PBR*. The *delay_ratio* in Figure 3.4 shows that *PROR* incurs higher delays in target tracking scenario under three different

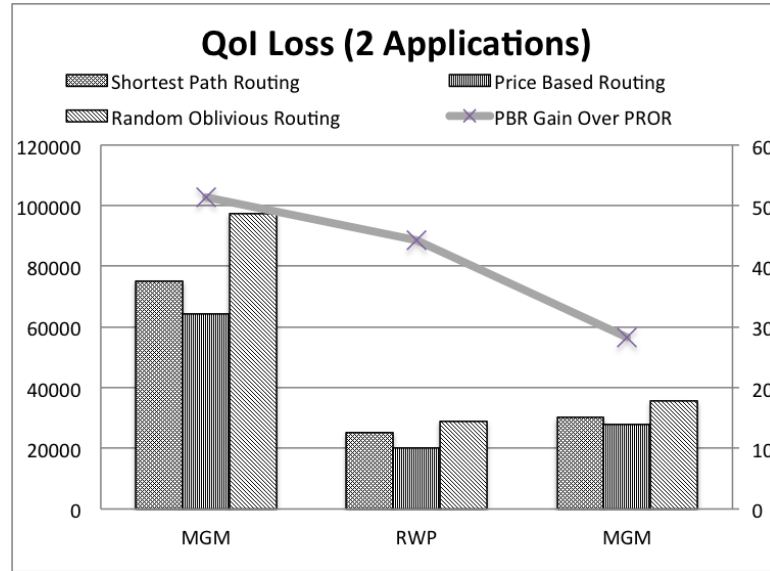


Figure 3.2: Average QoI losses in three models with 2 applications and percentage gain of *PBR* over *PROR*

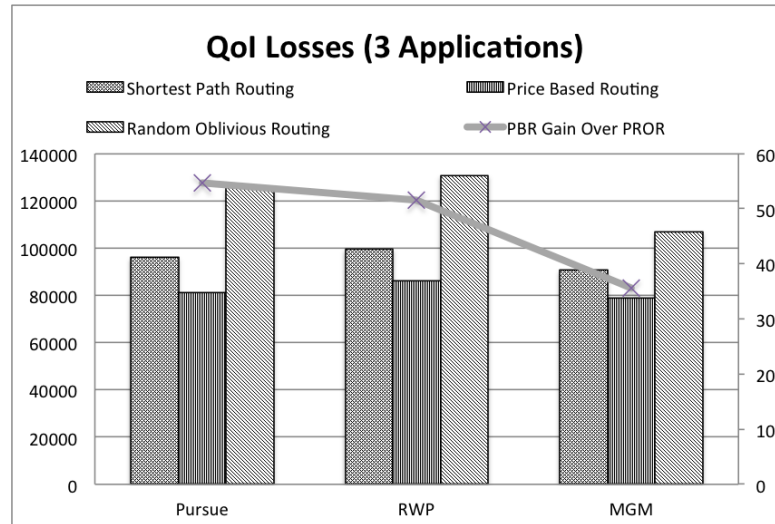


Figure 3.3: Average QoI losses in three models with three applications and percentage gain of *PBR* over *PROR*

mobility models assumed by targets.

3.4.2.2 Comparison of PBR with SPR

In order to show the efficiency of our routing protocol in a fully functional network, we simulated network without any failed nodes or broken paths. We measured the sum of *QoI* losses for all the applications for both *SPR* protocol and our dynamic price based

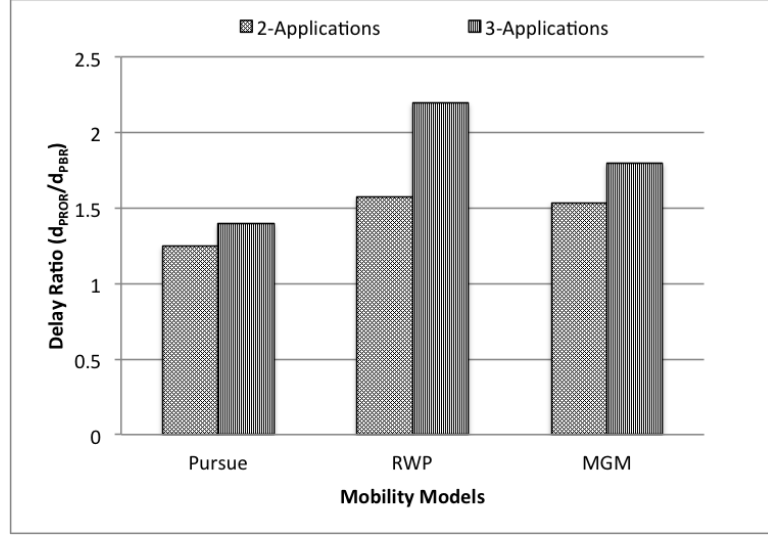


Figure 3.4: Delay ratio of PROR to PBR

routing protocol (PBR). Figure 3.5 presents the percentage decrease in *QoI* loss when our proposed mechanism is used as compared to *SPR*. As the figure shows, our approach decreases the utility loss by one third, in some cases. We observed that our method saves *QoI* losses. The percentage gain over *SPR* in our results was calculated using equation 3.5.

$$Gain = 100 \times \frac{QoILoss_{SPR} - QoILoss_{PBR}}{QoILoss_{PBR}} \quad (3.5)$$

where $QoILoss_{PBR}$ is the *QoI* loss in case of our protocol and $QoILoss_{SPR}$ is the *QoI* loss observed in case of *SPR*.

Figure 3.6 shows the decrease of the *QoI* losses for applications under our *PBR* protocol as compared to *SPR*. Here, measurements for only one priority set and *MGM* model are presented but same behavior was observed under different other priority sets and mobility models.

Figure 3.7 shows decrease in average packet delivery delay. The figure shows that our intuition of dynamically routing traffic with different priorities on different paths actually decreases delay for all the applications in different scenarios. This is because when the mechanism notices that the auction winning prices on certain route are increasing, it diverts traffic from that route to alternative routes to prevent the increase in congestion on that route. This leads to increased packet delivery. As shown in the figure 3.8, our

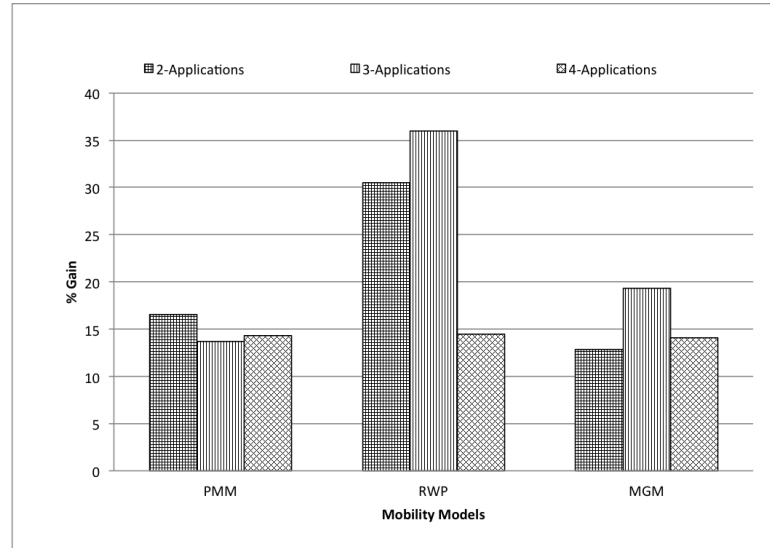


Figure 3.5: Percentage decrease in QoI loss in price based routing compared to SPR under different mobility models and multiple applications

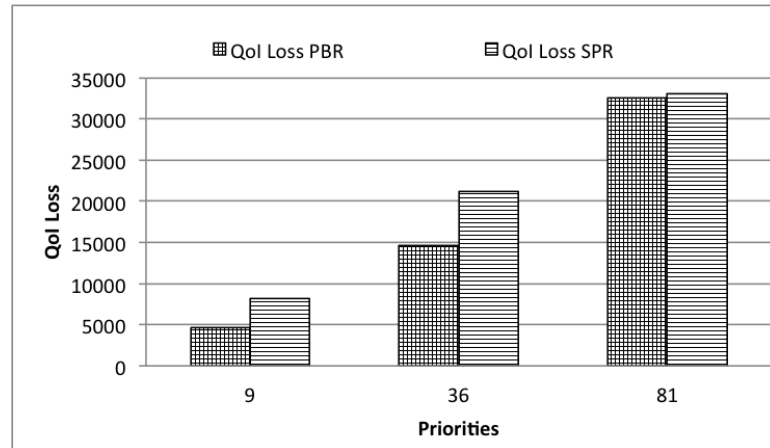


Figure 3.6: QoI losses for three applications under dynamic price based routing and SPR protocol (note that that total loss of QoI in this scenario is the sum of losses for all three applications, so clearly lower priority application gain the most from our mechanism).

algorithm delivers 15% more packets than *SPR* protocol can.

3.4.3 Partially Damaged Network

Our proposed mechanism is resilient to network changes by dynamically adjusting paths in case some portion of the network fails and some nodes are not operational. To demonstrate this capability, we simulated a scenario in which the network partially fails

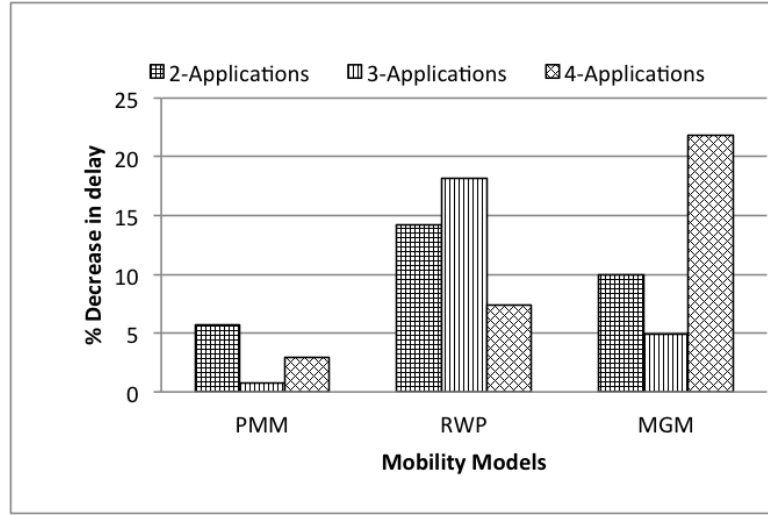


Figure 3.7: Percentage decrease in packet delivery delay in price based routing compared to SPR under different mobility models and multiple applications

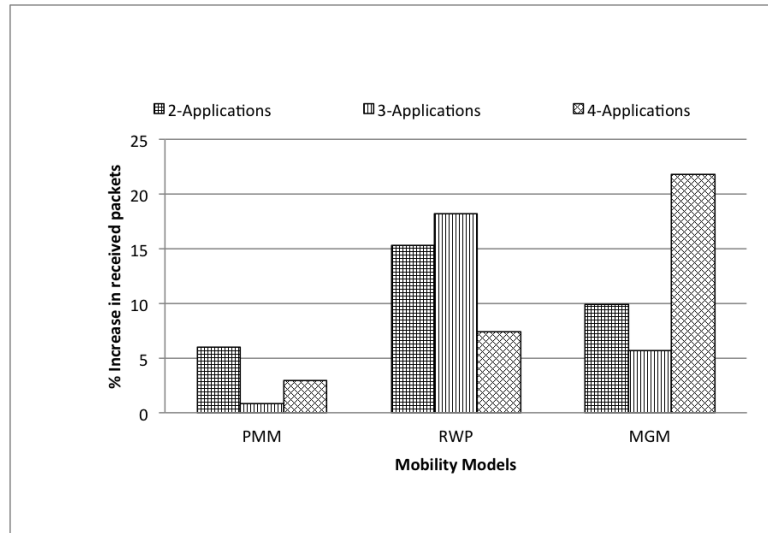


Figure 3.8: Percentage increase in packet received in price based routing compared to SPR under different mobility models and multiple applications

during normal operation because of some emergency or component failure. We evaluated the performance of our mechanism in cases where 5%, 10% and 30% of the network is damaged and nonfunctional, in order to test the responsiveness and resilience of our protocol with increasing network failures. In this section, we present simulation results showing that our algorithm performs well in a partially damaged network. Figure 3.9 shows performance comparison for a network in which 5% of the nodes fail in the middle

of the simulation. We can see in the Figure 3.10 that our protocol performs better than *SPR* and even with 5% damaged network; moreover the performance is comparable to fully functional network. With our approach we can increase the number of delivered packets and decrease delay.

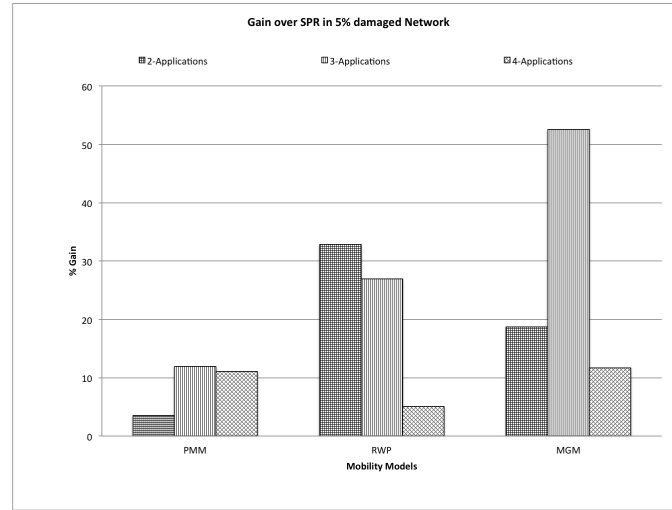


Figure 3.9: Percentage gain in dynamic priced based routing compared to SPR under different mobility models and multiple applications in 5% damaged network

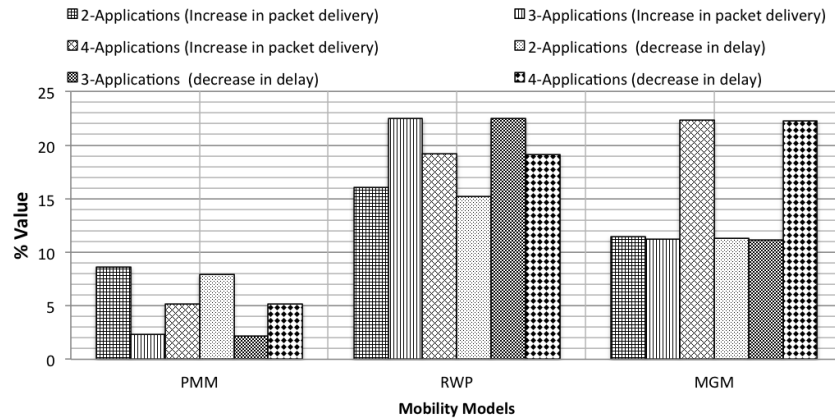


Figure 3.10: Percentage increase in packet delivery and decrease in delay, in dynamic price based routing compared to SPR under different mobility models and multiple applications in 5% damaged network

As we scale the network failure, the performance of our protocol decreases as expected. As more and more sensing nodes fail, the network has fewer tracking packets gen-

erated as well as fewer routing nodes. Therefore, the throughput of the network decreases, but by leveraging the available network, the effect of the damage can be decreased. With 10% damage to the network, our protocol still reduces *QoI* losses and decreases the delay by approximately 15%. Please note, that the gain, decrease in delay and increase in packet delivery also depends on the mobility model assumed by the moving objects. For example, we have lower gains in *PMM* model because objects in pursue model follow each other, thus they move in a group. All the objects in the group are usually detected by the same group of sensing nodes, therefore, the sum of *QoI* loss is smaller when the network is fully operational. When the network is damaged, and all the objects move to damaged part of the network, there is much less information about all the objects, therefore, the network incurs high *QoI* losses which may be unavoidable. When objects move in an uncoordinated fashion, like in *RWP* and *MGM* models, there may be one object in damaged part of the network but others detected by the operational part of the network, which reduces the overall *QoI* loss. This enables our routing algorithm to disperse the traffic via different routes.

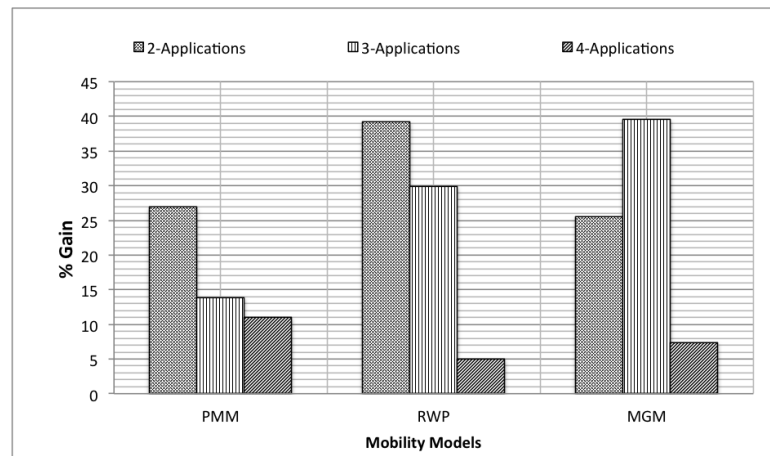


Figure 3.11: Percentage gain over SPR under different mobility models and multiple applications in 10% damaged network

Please note that in 30% highly damaged network, we can still reduced packet delivery delay and improve packet delivery significantly in *RWP* and *MGM* models, but the performance with *PMM* model is lower as explained earlier.

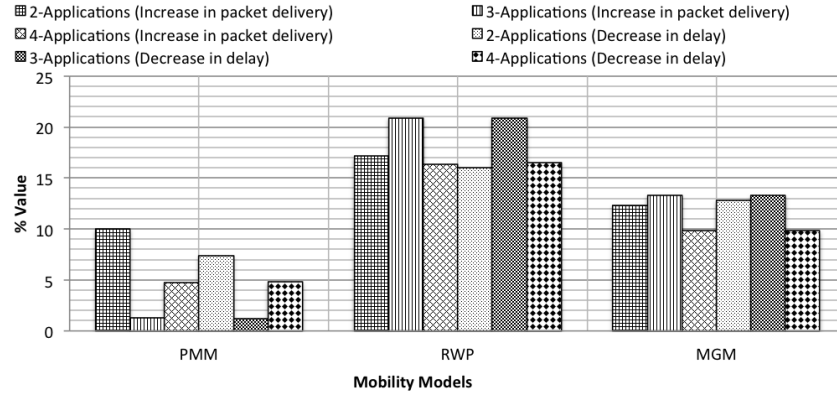


Figure 3.12: Percentage increase in packet delivery and decrease in delay, in dynamic price based routing compared to SPR under different mobility models and multiple applications in 10% damaged network

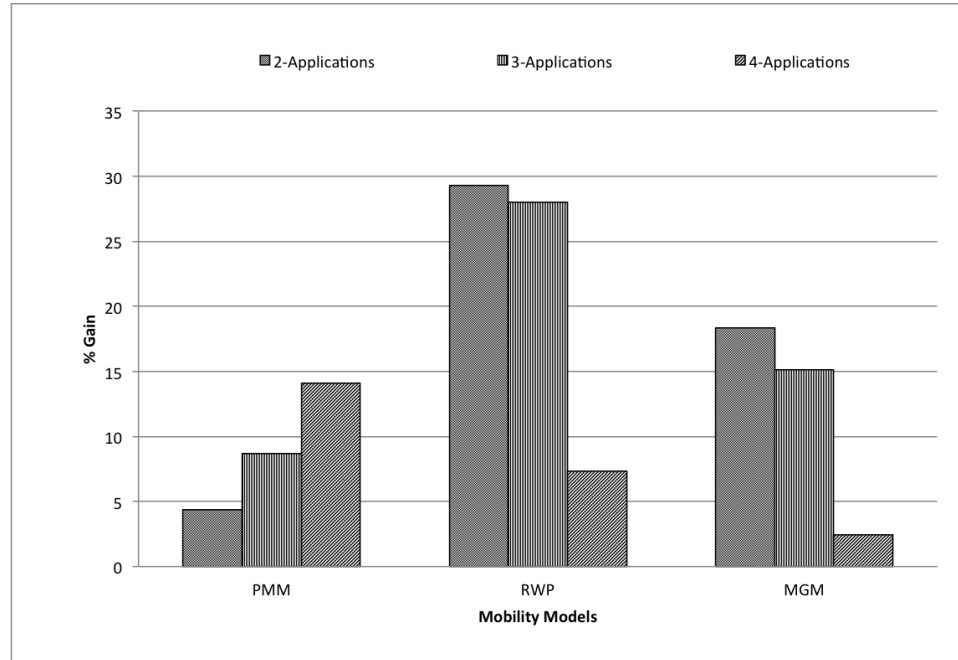


Figure 3.13: Percentage gain over SPR under different mobility models and multiple applications in 30% damaged network

3.5 Price Based Routing for Network Resource Management

In previous sections, we have used WSNs as example to explain and evaluate our approach but our approach is also applicable to other type of networks, such as road networks and building evacuation routes. Our proposed approach can be used to manage

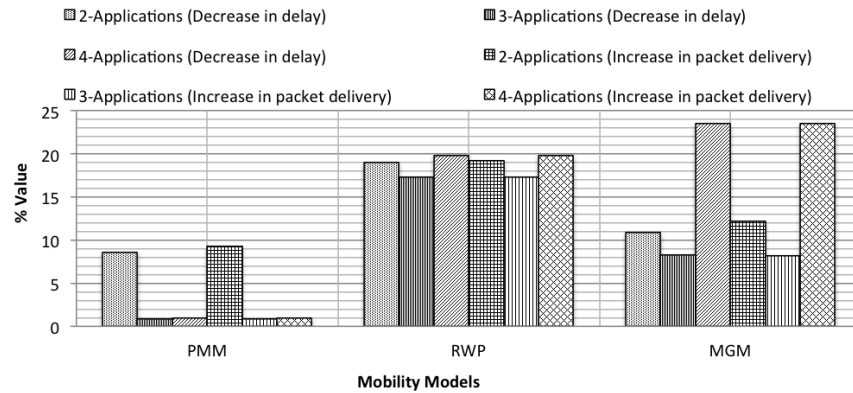


Figure 3.14: Percentage increase in packet delivery and decrease in delay in dynamic price based routing compared to SPR under different mobility models and multiple applications in 30% damaged network

other networks which are prone to congestion.

For example, in case of a heavily used road network our mechanism can be used to dynamically set toll prices on heavily used shorter routes to encourage people to use alternative longer routes to decrease possible congestion on shorter routes. Also, in disaster scenarios or construction work in which road network is partially damaged or under construction, our mechanism can efficiently route traffic on multiple paths to avoid congestion.

Our mechanism can also be used in emergency evacuation of buildings and towns; the mechanism can help to evacuate habitants through different exits to avoid stampede and congestion on commonly used exits routes. Analogical to previously described target tracking example in which distinguishing data packets from different applications having different priorities is important in building evacuations, such as evacuation of hospitals, segregating different types of patients and guiding them to different exits is crucial. Senior or handicapped patients and patients needing special assistance should be evacuated via shortest path (sometimes dedicated paths with special arrangements) and their flow cannot be mixed with others. Evacuating patients without segregation may create convoy effect or lead to situation in which slow moving patients are overrun by fast moving people. Patients can be divided in different groups with different priorities and our mechanism can effectively evacuate these groups via different routes. The priority of the patient

(reflecting patient's physical fitness) will decide which way they should follow. In such a case the delay on the link can be predicted by the difference between the arrival and departure rate of people at the starting and ending point of a route. Patients that are able to move faster than the speed on the route will dynamically be routed to different paths which are longer but faster due to type of patients leaving via that route. Thus, patients who share similar physical conditions would be grouped together and flow through different paths without hindering each other. For such evacuation purposes, dedicated screens or other methods can be used to direct people to different exits.

3.6 Concluding Remarks

In this chapter, we presented dynamic *Price Based Routing* protocol for prioritized traffic in both communication networks and physical infrastructure networks. We show that routing packets based on path prices improves routing decisions and helps to reduce congestion in the network, by dynamically establishing different paths for multiple applications. The packets with lower priority may incur higher delays if routed on shorter paths which are congested. Therefore, our mechanism enables low priority packets to go through alternate paths which may be longer, but due to lighter traffic impose shorter delays. We also show that our approach dynamically segregates multi-priority traffic in network making it robust against partial network failures. We also showed in section 3.5 that our mechanism is applicable to different types of networks such as WSNs, cyber-physical systems and emergency evacuations. We presented prioritized version of random oblivious protocol and compared it with our price based routing and found that our protocol works better.

We found that our protocol works much better than shortest path routing in event-driven scenarios, such as target tracking in which multiple sources at different locations, generate data packets, but in cases where only one source continuously sends an infinite stream of packets and network is fairly stable, our protocol yields marginally lower average delay but still efficiently allocates bandwidth to multi-priority application. As our mechanism is not tightly coupled with data communication network protocols, it is equally applicable to physical infrastructure networks for routing spontaneous bursts of people and vehicles from one point to another.

CHAPTER 4

Service Configuration in Constrained Environment: Data Relevancy

The emergence of smart mobile devices is making WSNs much more intelligent than ever. These smart devices operate at the edge of the network and host services to perform certain tasks. When these services are composed together they can perform computationally complex tasks. The networks in sensory systems are ad-hoc and can change over time. In this chapter, we show how services hosted at the edge of the network can be dynamically configured to perform complex tasks. We present a service-oriented system for WSNs that is capable of performing service configuration under data relevancy constraints. We present and evaluate “*Cost Based Model (CBM)*”, “*Gain Based Model (GBM)*” and “Return on Investment (ROI)” approaches to capture the relevancy of services hosted on WSN nodes in composite service configuration [60]. The system is resilient to failures and can operate in *manual* or *autonomous* recovery modes. The system supports three service configuration methods namely, *distributed*, *centralized* and *hybrid*. Furthermore, we present a novel emulation mechanism for testing the performance of our proposed relevancy models and show that our system efficiently configures services.

4.1 Introduction

Service oriented architecture (SOA) for WSNs has got significant attention of researchers. The SOA for WSNs focuses on how applications for WSNs can be designed and developed as services so that they can be interconnected and composed in service oriented fashion to create complex services. With enhancing hardware capabilities, sensors can execute services on the edge of the network. These services can be interconnected in order to perform intelligent tasks. However, there are still open research questions in this

Portions of this chapter previously appeared as: S. Shah, B. Szymanski, P. Zerfos, and C. Gibson, “Towards relevancy aware service oriented system in wsns,” IEEE Trans. on Services Computing (TSC), vol. PP, no. 99, p. 1–13, Oct. 2014.

Portions of this chapter previously appeared as: S. Y. Shah, B. Szymanski, P. Zerfos, C. Bisdikian, C. Gibson, and D. Harries, “Autonomous configuration of spatially aware sensor services in service oriented wsns,” in Int. Conf. on Pervasive Computing and Commun. (PERCOM Demos), 2013, pp. 312–314.

emerging field of service orientation. Tools and techniques for WSNs that would enable secure, fault-tolerant and robust configuration of complex services are still in developing stages. The geospatial aspect of the data that is collected and transferred via WSNs is adding a totally new innovative aspect to the design of complex services in WSNs. This geospatial tagging of data was never design factor in service orientation earlier, neither in WSNs nor in the web-services.

In a sensor oriented WSNs, the nodes host services and these services can be configured on-the-fly to perform more sophisticated operations. The service configuration in pervasive wireless sensory systems (WSNs) is quite challenging as the requirements of the applications/services hosted on WSNs change over time and these changes must be reflected in the system configuration. As events in WSNs (e.g., node failures making services residing on the node unavailable etc.) occur over time, the configuration mechanism should dynamically reconfigure the system according to the new requirements. An efficient configuration mechanism should be able to configure services in a way that ensures their inputs and outputs to be interoperable during instrumentation and execution of complex tasks. Moreover, the selection of services in service configuration should be taking into account various constraints (including configuration and operational policies) and various performance metrics. The constraints can be divided into two main categories, hard constraints and soft constraints. Hard constraints include those operational constraints without adhering to which the service cannot be configured and composed, e.g., without fulfilling policy requirements services cannot be configured, other forms of hard constraints are budget constraints and energy consumption limits etc. These policies and constraints can be specified in various formats i.e., from natural language to ontologies and computer programs. Soft constraints are on the other hand constraints which are not critical in configuring services but are focused on optimization of the services, e.g., relevancy of information produced by the service to user's area of interest. Service composition in WSNs as well as in web services has been studied in the past. A closely relevant work in [27] proposes dynamic service composition, but in this prior work only cost is considered as metrics for optimizing service composition [27], however the relevancy of services plays a vital role in service composition in WSNs.

Intuitively, relevancy of a service is the volume of relevant information that the ser-

vice produces to the user's request. In other words, information in which user is actually interested. Relevancy of a service can be defined along various dimensions such as, coverage of the service in the area of interest, accuracy of the measurements provided by the service, latency, temporal utility of a service or geospatial/spatiotemporal closeness of a service to the area of interest. Typically, in service configuration mechanisms, the relevancy that a service brings to the overall system is ignored and only the flat cost of using a service is considered. In contrast, our service configuration approach considers both aspects, cost and relevancy.

In this chapter, we focus on the design of a service-oriented system that can capture the concept of relevancy and can compose complex services from services hosted on sensor or mobile nodes. The sensor service configuration is more elaborate form of service composition. Service composition is a problem of composing services together to form a composite service with complex functionality. The process of service configuration not only performs service composition but also takes into account the implicit and explicit requirements specified by the user of the service, and incorporate these requirements into the system as hard and soft constraints. Some of such requirements include, user specified policies and restrictions, spatiotemporal relevancy constraints on the services hosted at the edge of the network and modes of operation of the service. In this chapter, we focus mostly on relevancy of services and geo-spatial relevancy of services to the area of interest defined by the user. Yet, the concept of relevancy and the models for capturing relevancy presented in this chapter are more general. These models can be applied to any user defined relevancy metric which can then be modeled as a soft constraint in the configuration of services. Figure 4.1, shows an example of the a service hierarchy and how services at the edge of the network are configured to form a complex service.

Throughout this chapter, we use the word "source service/source node" as in Figure 4.1 for a service provided by a sensor or a group of sensors without any input from other services and we require that the relevancy is defined for each source service. Composite services are the services that require input from other services. All services produce output and their relevancy is established by their service providers which directly or indirectly provide input to them. Therefore, a service can be either composite service or a source service.

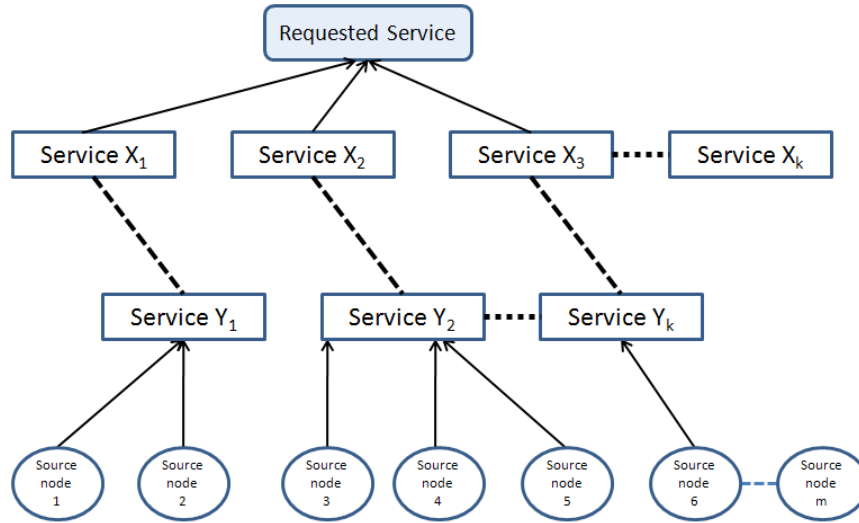


Figure 4.1: Sensor service configuration hierarchy

4.2 Application Scenarios for Relevancy Aware Services

Many applications require relevancy aware services and in such applications maximizing relevancy to user's interest or intent is very critical. In this section, we present application scenarios in which relevancy plays an important role.

4.2.1 Information Relevancy Maximization Scenario

Suppose a user is interested in a composite service, that is capable of monitoring an area for activity and visualizing the location of events. Our system configures the composite service that uses component services such as a camera service, a set of acoustic detection services and a localization service to geolocate the events. For such a scenario, we have developed a map interface to the system. The user is provided with a map of the area with services hosted at different locations. Using this GUI, the user encircles the area which the user wishes to monitor. The system automatically selects services that maximize the coverage in the area of interest and filters out any unrelated services. After selecting appropriate services, the system configures those services and links them together to create a composite service that performs the monitoring task. Figure 4.2 shows the map client also accessible on an iPad connected to the system. The interface displays services on the map and the links between services denote the wiring among different services. The system shows how spatially relevant services are composed together to

configure a complex service that utilizes three 'Line Of Bearing (LOBR)' readings using "JOIN" service that calculates 'Line Of Camera Reading (LOC)' . The big yellow circle shows the area of interest specified by the user. As we can see in the Figure 4.2 the service "LOBR_4" is not used by the composite services as it does not link to any other services on the map. This is because "LOBR_4" is out of the area of interest specified by the user, so an alternate service provider of the same outputs, in this case "LOBR_3" is selected for configuring the required monitoring service.

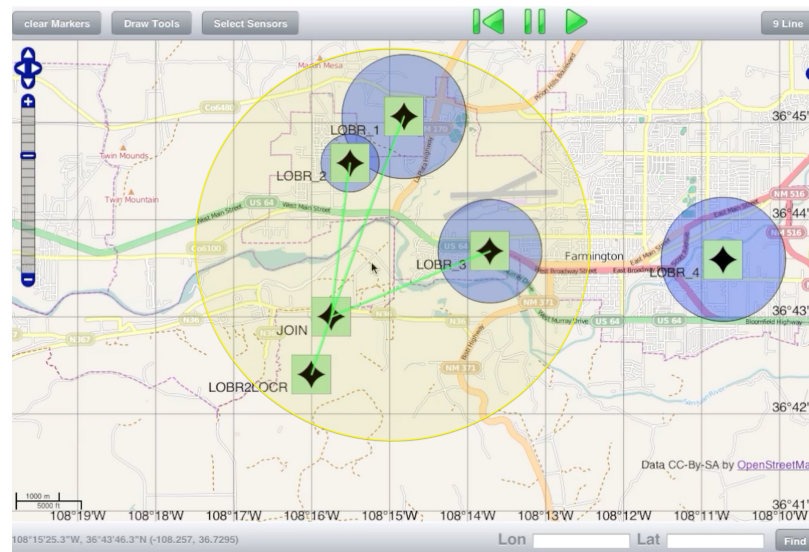


Figure 4.2: Map client of service configuration showing services and their interaction.

4.2.2 Wildfire Modeling

Consider example of *wildfire modeling* in a forest. Suppose there are various kinds of sensors, such as relative humidity (RH) sensors, temperature sensors, anemometer (for wind speed) and wind direction sensors installed in the forest. In order to model the wildfire, we need complex services that take input from various sensors and produce the required fire spreading models. In this case, a RH sensor will require input from the temperature sensors in the fire area; the *originLocator* service will require input from RH sensors as well as various temperature sensors to calculate the origin of the fire. The *wildfire modeler* service will require input from various sensors such as wind direction sensor, anemometer, RH sensors and temperature sensors to model the path and speed of the fire

spreading. In such a way, different services are configured together to produce information about wildfire. The accuracy of wildfire modeling highly depends on the efficient selection of services. These sensors should not only produce accurate information, but information that is highly relevant to the area of interest, which in this case is the area on fire. Services in close vicinity of the wildfire are of higher relevance than those far away from the fire, even though they produce the same kind of outputs but are not relevant to the actual fire. In this example, the semantic matching of services as well as the relevancy of services both are of high importance for configuring and composing complex services.

4.3 Service Configuration with Spatial Relevancy

Suppose a WSN is deployed as a support system for a disaster relief effort. A monitoring system configured in such a scenario might use audio and video feeds produced by other services to surveil the area for operation coordinators. The service configuration in such a scenario should not only consider input/output portability [27], but also other factors, such as energy cost and spatial relevancy of services to the area of interest. In such a scenario, services that are more relevant (e.g., have a larger sensing range in the operation area) are more useful than services that provide the same outputs but with lower relevancy. In this section, we present two different models for relevancy incorporation in service configuration. Based on user's preferences, these models can be used to incorporate service relevancy in the service configuration.

4.3.1 The Coverage Model for Relevancy

Sensing and mobile devices can collect various kinds of data from the environment and the relevancy of data produced by a sensor, a mobile device or a service can be defined accordingly. Here, we model relevancy in terms of geospatial coverage that a sensor offers in the area of interest. For the scope of this chapter, we model the area covered by a sensor in the form of a disk of radius r around the sensor location, which is defined by latitude, longitude and altitude. Each user's service request also specifies the area of interest using two parameters, a point (latitude, longitude, altitude) on the map and radius R . The area of interest is calculated as a circular disk of radius R around the specified point. Our design and relevancy utility models are not restricted to a disk coverage model; the same

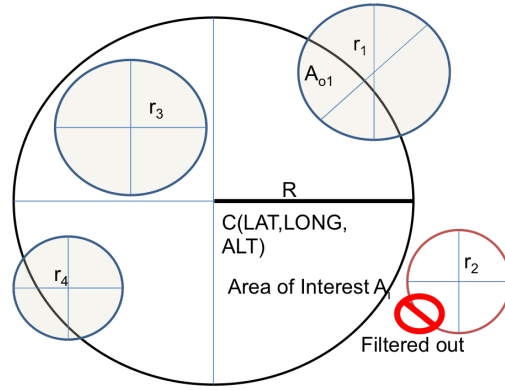


Figure 4.3: Disk coverage model

mechanism can be applied to different coverage models (hexagonal, polygon etc.). Our system is designed in such a way that new coverage/overlap formulations can easily be plugged into the system.

If a source service/sensor has no coverage in the area of interest then the node is not considered in the configuration process. The relevancy of a service increases with its coverage in the area of interest. Figure 4.3 shows the disk coverage model, the large disk centered at a particular *LAT, LONG, ALT* with radius R represents the area of interest to the user and the small circles depict the sensing area of different services. As shown in the figure, services with sensing radius r_2 is filtered out of candidate set of services for configuration because it does not provide any coverage in the area of interest. In Figure 4.3, A_{o1} is the overlapping area of service with radius r_1 . If R is the radius of the area (disk) of interest A_i centered at $C(LAT, LONG, ALT)$ and r_x is the radius of the disk covered by a service x centered at $c_x(lat, long, alt)$, then the relevancy of a service x is defined by equation 4.1.

$$Relevancy = \Gamma(x) = \frac{|C_R \cap c_{r,x}|}{|C_R|} \quad (4.1)$$

where C_R denotes area of interest with radius R and $c_{r,x}$ denotes the area of coverage of service x with radius r .

4.3.2 The Price of Relevancy

We define the *Price of Relevancy* as the price that the user is willing to pay for getting certain value from a service; this “value” is a function of relevancy. We believe,

such an approach is important for evaluating models for relevancy as it gives the users a way to customize used services according to their needs. Depending upon the constraints that user might have in the application space such as budget or amount of relevancy, user can control budget spending and achieve the required relevancy. The *Price of Relevancy* is represented by Eq. (4.2).

$$\text{Price of Relevancy} = \frac{\text{AdditiveBaseCost}}{\text{Value}} \quad (4.2)$$

The “value” in a simplest case can be just the relevancy that the configured service provides in the area of interest. Then the equation can be written as,

$$\text{Price of Relevancy} = \frac{\text{AdditiveBaseCost}}{\text{Relevancy}} \quad (4.3)$$

The values of *AdditiveBaseCost* and *Relevancy* are normalized to same range. The *Relevancy* here is the overall relevancy of the configured service to the area of interest and the *AdditiveBaseCost* is the total cost i.e., accumulated cost of all the services used in the service configuration. In a service configuration, use of a particular service incurs certain cost to the hosting node. This cost can be singular, such as energy consumed, or a combination of different factors such as edge transmission delay, battery consumption and processing time delay; we refer to such a cost as the *BaseCost*. The *BaseCost* of a service can be measured in different ways, it can be monetary cost of using the service, energy consumed, communication and processing cost etc., or composition of various such costs. As we can see from the price formulation (Eq. 4.3), if the prices are very high that means the cost incurred to achieve certain relevancy is high and a user may not achieve appropriate level of relevancy for the cost to be justified. Therefore, according to our price formulation, low prices are beneficial. The best suitable price also depends on the user specific situation as we will see later in the results sections; we use it as one of the evaluation criteria for our relevancy models.

4.3.3 Cost Based Optimization of Relevancy

Cost and relevancy of a service play major roles in the configuration of services in mobile and sensor networks. Users often are sensitive to the cost of composite services

but still require relevant information, and want to control the weight they put on to cost and relevancy; for such users we propose the *Cost Based Mode (CBM)*, represented by Eq. (4.4). This utility model aims to configure services that are both low-cost and spatially relevant to the requested area of interest. Every service has a *BaseCost* associated with it, which is incurred when the service is used, since we defined relevancy in Eq. (4.1) as ratio, we can easily convert it to irrelevancy to the area of interest as shown below. The irrelevancy of a service is also a cost, thus we combine both *BaseCost* and *Irrelevancy* into an *AggregatedCost* using Eq. (4.4). Using a balancing coefficient α where $0 \leq \alpha \leq 1$, user can adjust the model to its needs with lower cost or higher relevancy. Both the relevancy cost and base cost are normalized to the same range before the aggregated cost is calculated.

$$\begin{aligned} Irrelevancy(x) &= \bar{\Gamma}_x = 1 - \Gamma_x \\ AggregatedCost &= \alpha \times BaseCost_x + (1 - \alpha) \times \bar{\Gamma}_x \end{aligned} \quad (4.4)$$

Under *CBM*, the system uses *AggregatedCost* of services in service selection process and uses greedy heuristics of *Set Cover* problem (following Geyik et al. [27]) to select service with minimum *Aggregated Cost* at each step of service composition. The minimum cost service composition and achieving maximal relevancy in services are NP-hard problems ([27], [29]) which is why we use the greedy heuristics.

Figure 4.4 shows the affect of α on the relevancy. As we can see from the figure that as we increase the value of α in the *CBM*, the price of the relevancy increases and it goes to infinity at $\alpha = 1$. This shows that as user puts more weight on *BaseCost* in the model the system selects services with lower relevancy and as the value of α goes to 1 the service is composed with minimum *BaseCost* but may produce totally irrelevant information. Please note that at $\alpha = 0.5$, the price of relevancy remains below 1 and there is a good balance between both the *BaseCost* and *Irrelevancy*. In general user can set α according to the application scenario, 0.5 is good value in cases where user wants both the low cost as well as relevant services. If user wants lower cost or more relevancy this can be accomplished by adjusting value of α above or below 0.5.

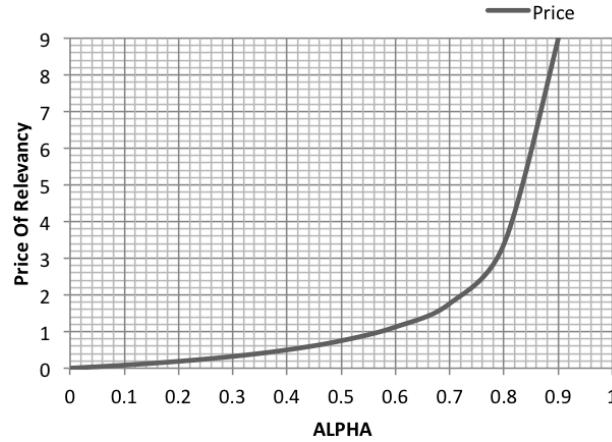


Figure 4.4: Price with changing α

CBM is useful in many scenarios in which minimizing the overall cost of services is needed but reasonable relevancy of information is also desired. For example, if a user is interested in overall temperature trend in an geographical area (e.g., New York City) or a forest, *CBM* can configure services with low cost yet maintaining reasonable relevancy to the specified area. Other applications of this model are, water quality monitoring of streams in the city, marketing surveys in the cities etc., in such scenarios relevant information from the area of interest is required but the information does not need to be precisely localized.

4.3.4 Gain Based Optimization of Relevancy

Some users might be sensitive to the overall cost of the service but others might be very sensitive to the actual gain in terms of relevancy they get from the applications. Users sensitive to gain need to maximize the relevancy of information to their area of interests and may not be restricted by cost constraints. For such users, we propose a *Gain Based Model (GBM)*. *GBM* enables users to control the amount of relevancy that they get from the service and adjust according to their needs. In addition to relevancy *GBM* incorporates the idea that information outside the area of interest may not be totally irrelevant, thus allows users to specify how much they value such information. *GBM* is represented by Eq. (4.5).

$$Gain = (1 - \kappa) \times \Gamma_x + \kappa - BaseCost_x$$

or

$$Gain = \Gamma_x + \kappa \times \bar{\Gamma}_x - BaseCost_x \quad (4.5)$$

The coefficient κ where $0 \leq \kappa \leq 1$, is the ratio of value of irrelevant information to relevant information. Using κ users can specify how much they value the information that is outside of the direct interest. The total value of all the information produced by service x is one unit, the $BaseCost_x$ of the useful service x is a fraction of this value, i.e., $0 \leq BaseCost_x < 1$ (otherwise service is unusable because its cost exceeds its value even if fully relevant). The solution is to include in the configuration all services for which Eq. (4.5) is positive. This model tries to configure highly relevant services thus maximizing gain while taking into account the value of κ . Please note that this balancing coefficient κ is different from that of *CBM* which is α . The two models serve users with different concerns. The coefficient α in *CBM* enables user to control “Cost” whereas κ enables customization of gain in terms of relevancy.

For example, if the data outside of the area of direct interest of the user are worthless, then $\kappa = 0$ and the service x is included *if and only if*,

$$Gain_x(0) = \Gamma_x - BaseCost_x > 0$$

when $\Gamma_x > BaseCost_x$

On the other hand if irrelevant information is as valuable as the relevant one then $\kappa = 1$ and we have,

$$Gain_x(1) = 1 - BaseCost_x > 0$$

so correctly all useful services qualify if their cost is less than 1.

Finally, if irrelevant information is half of the value of the relevant one, we get the

following condition.

$$\begin{aligned} Gain_x(1/2) &= (1 + \Gamma_x)/2 - BaseCost_x > 0 \\ \Gamma_x &> 2 \times BaseCost_x - 1 \end{aligned}$$

Thus, more services will qualify than in case of $\kappa = 0$ because $2 \times BaseCost_x - 1 < BaseCost_x$. In general the qualifying condition is directly derivable from equation 4.5,

$$\Gamma_x > \frac{(BaseCost_x - \kappa)}{1 - \kappa} \quad (4.6)$$

So when $BaseCost_x < \kappa$ and $\kappa > 0$ the service qualifies no matter how relevant it is. Since for each service we know κ , Γ_x and $BaseCost_x$ the optimization is very simple, for each service we just compute if inequality (4.6) is satisfied. The value of κ is application dependent, as shown above it can be set to, above or below 0.5. Setting κ above 0.5 will make more services qualify for configuration giving increasing importance to information from outside the direct area of interest. The system allows user to adjust value of κ , so user can always try different values of κ in order to choose most suitable value for the application.

GBM is useful in many scenario where the relevancy of information is of prime importance. In wildfire scenario mentioned in section 4.2.2, the information from services in the wildfire region is of high value, therefore services should be configured and optimized to produce highly relevant information about the fire area as well as its immediate surroundings. Monitoring water contamination in a localized area (e.g., a county) or information regarding spreading of virus infection in a county are other strong candidate scenarios for *GBM*. In such life-threatening scenarios, highly relevant information is very valuable as opposed to cost factor, precise information is needed to track and stop spread of disease locally.

Return on Investment (ROI)

With *GBM*, we can measure the gain achieved from using certain service. If a user has multiple services running and wants to balance and optimize spending on them, *GBM* can be used to calculate the *Return on Investment (ROI)* of each service. The *ROI* of

services is very useful measure for users wanting to create a portfolio of services within the budget constraints. Using Eq. (4.5), we can define *ROI* of a service as,

$$ROI_x(\kappa) = \frac{Gain(\kappa)}{BaseCost_x} = \frac{(1 - \kappa) \times \Gamma + \kappa}{BaseCost_x} - 1 \quad (4.7)$$

where $ROI_x(\kappa)$ is the ROI of service x with coefficient κ .

In order to maximize gain for a given budget based on *ROI*, let *RealCost* denote the monetary measure in the same units as Budget and let *BaseCost* be defined as the ratio of the *RealCost* to the value of a service. Then for the given κ , we can sort all services in the decreasing order of their $ROI_x(\kappa)$ renumbering services so the services with highest *ROI* has lowest index and so on. If two services have the same *ROI*, then the larger $RealCost_x$ service is placed before lower one. Then in a loop we do,

```

index = 0
while(Budget > 0)
{
    index ++;
    Budget -= RealCostindex
    Add Serviceindex to configuration
}

```

As an example, suppose we have two services *Service(1)* and *Service(2)*. *Service(1)* is worth \$100 if all of its information is relevant and \$20 if nothing is relevant, and it costs \$80 to run and has relevancy of 90%. So,

$$RealCost_1 = \$80, BaseCost_1 = \$80/\$100 = 0.8, \kappa = \$20/\$100 = 0.2$$

$$ROI_1 = (0.8 * 0.9 + 0.2)/0.8 - 1 = 0.15$$

Service(2) is worth \$200 if all of its information is relevant and \$50 if nothing is, and it costs \$160 to run and has relevancy of 80%. So,

$$RealCost_2 = \$160, BaseCost_2 = \$160/\$200 = 0.8, \kappa = \$50/\$200 = 0.25$$

$$ROI_2 = (0.75 * 0.8 + 0.25)/0.8 - 1 = 0.0625$$

If our budget is \$120 then, if *Service(2)* is chosen, it will bring \$7.5 gain and exhaust the budget. If instead we select *Service(1)* for \$80 and *Service(2)* for the remaining \$40, we get \$12 gain from *Service(1)* and \$2.5 from *Service(2)* so in total \$14.5 > \$7.5.

It is important to notice that the *ROI* allows the user to assign resources across several applications. With many applications, setting the minimum *ROI* for all applications tells the user how to assign resources (but the budget could be high) or the procedure that we showed could be used, where we allocate first the highest *ROI* assets for all applications and stop when entire budget is optimized. The optimization based on *ROI* has its benefits both in case of atomic resources as well as resources which can be partially shared. In the above example, though *Service(2)* is worth \$200, but the *ROI* is low if *Service(2)* is used alone. On the other hand the *ROI* of *Service(1)* and partial use of *Service(2)* leads to higher *ROI*. The partial use of services is possible in different situations. Consider example of a 'video camera' that provides high definition video for a certain cost (that can be cost of bandwidth), it can also provide lower resolution video for a lower cost i.e., lower usage bandwidth. In such a way camera is partially used for lower cost.

4.4 System Design and Implementation

We present a service-oriented system for service configuration with relevancy constraints. Our proposed design provides user with the flexibility to configure system in various modes (*Centralized, Hybrid and Distributed Modes*) as well as run system in *Autonomous Recovery Mode* and *Manual Recovery Mode*.

4.4.1 The WSN Framework

We prototyped our system in Java using the ITA Information Fabric [61], a SOA-based middleware for sensor networks. A sensor network built using the Information Fabric consists of set of fabric nodes, each of which manages a set of assets and offers a set of services. Fabric is a fully distributed infrastructure with federated nodes (WSN nodes) that form a service bus across the WSN. The information fabric enables users to develop and deploy applications on sensor nodes and as a framework it provides basic functionalities, such as message routing, node discovery and connectivity services among the sensor nodes. Fabric uses Gaian Database [62] as a backend database for storing assets and services available across different parts of the network.

The service configuration with spatial constraints checking is implemented as fab-

ric service that runs on every Fabric node. Service configuration is designed with full consideration of extensibility. New components (e.g., a new coverage model, a more elaborate cost function, an additional system checker component) can be easily plugged into the system as services. Moreover, the service oriented design of the system enables the user to customize various aspects of the system such as enable/disable the spatial constraints, the depth of service composition on certain node, the cost function, the meta-data functionality, system recovery and configuration modes.

To achieve the flavor of realistic system, we run Fabric nodes inside Common Open Research Emulator (CORE [63]) nodes. Each *Core* node emulates a virtual machine with separate process space and independent network stack. Using *CORE*, we emulate the network layer of our WSN testbed. In order to emulate the data link and physical layer in combination with *Core*, we use Extendable Mobile Ad-hoc Network Emulator (EMANE [64]). Figure 4.5 shows the overview of the emulation testbed and how different components of the system integrate. As shown in the figure, the Fabric node runs inside a CORE node, which is an independent Linux container. Each CORE node is integrated with the EMANE in order to emulate *WIFI* communication among Fabric nodes. Our system is novel in a way that it uses current state-of-the-art simulation and emulation platforms to prototype the sensor service configuration. We believe that this is a novel way to test WSN applications and architecture because the emulation platform provide hardware like support to the Information Fabric framework within which we run our service configuration system. Moreover, with such a testbed it is possible to emulate large WSNs as well as emulate path losses and physical layer errors.

4.4.2 Service Provisioning

The user sends service configuration request to system through a client. One such client is the map interface shown in Figure 4.2. The *Configuration Service* handles the request accordingly based on the mode of configuration. Figure 4.6 shows how service configuration request is handled on node level. After the request is received from the user, the system retrieves services from the Fabric registry. The services set retrieved from *registry* is based on input/output parameter matching therefore the set may contain services that do not provide coverage in the requested area of interest. Services that

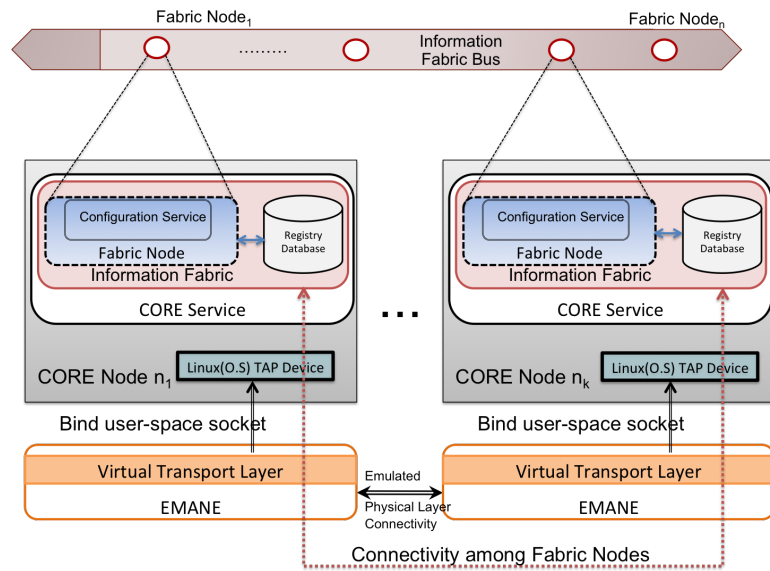


Figure 4.5: Distributed mode of configuration

do not provide coverage in the area of interest to the user are filtered out (except the critical services), thus leaving only those which have non-empty coverage in the area of interest. Then, a Set Cover heuristic is applied to the services, we implemented Set Cover heuristic algorithm similar to the one described in [27] for optimization based on *GBM* or *CBM*. If there are any hard constraints applied to services, such as policies then the corresponding policies are fetched from the *Policy Repository*. Only policy compliant services are selected for configuration and the final service composition graph is written back to the registry [46]. If the policies cannot be satisfied, the service is not configured and user is notified. The policies in our system are enforced as *hard constraints* while spatial constraints are enforced as *soft constraints*. Failure to meet *hard constraints* results in failure of configuration whereas if *soft constraints* are not met, user of the system can still get service configured, as user can choose to ignore spatial constraints. Policies that are *hard constraints* can be relaxed or negotiated using policy relaxation and negotiation techniques [65], however policy relaxation and negotiation are out of scope of this chapter.

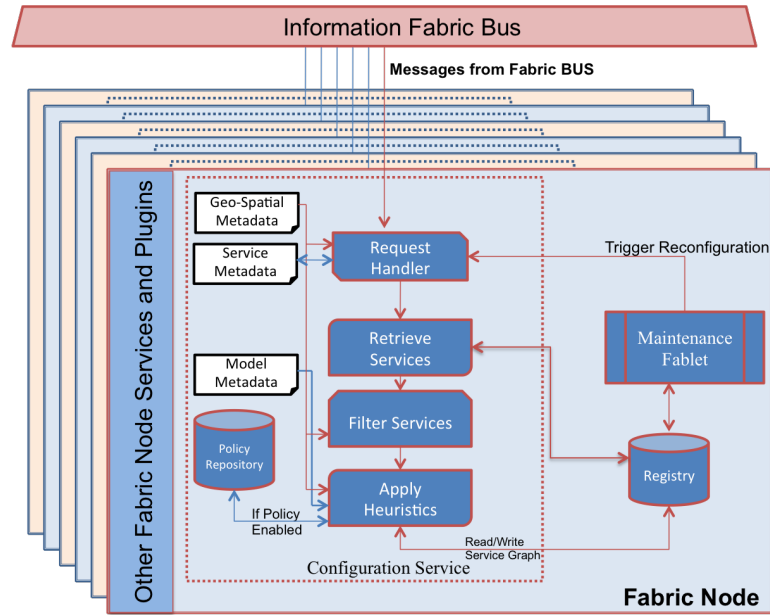


Figure 4.6: Node level information flow

4.4.3 Service Maintenance

Each node runs a *Maintenance Fablet* along with the *Configuration Service*. The *Maintenance Fablet* is a fabric bus plug-in that runs as an independent thread and is responsible for checking the composition status of the service hosted on the node. It makes sure that all the services which are known to be composed are indeed composed and in case any of the services becomes unavailable, the *Maintenance Fablet* detects the failure. The failure detection is done through monitoring, at specified intervals the composition graph of each service is retrieved from *Registry* and the availability of the node hosting this service is verified. The mapping of what service is hosted on which node and the current status of the node is stored and maintained by Fabric middleware in the registry, so *Maintenance Fablet* can easily use that information. In case node hosting the service has failed, the *Maintenance Fablet* triggers reconfiguration of the services affected by the failed service(s) or reports the failure in case auto reconfiguration is not possible. In such a way the state of the system is checked and maintained. The monitoring parameters (such as frequency of the system checks, enable/disable system checking) of the *Maintenance Fablet* are configurable and can even be specified by the user in the request, if the user wants to disable the automatic reconfiguration or wants to modify other configuration parameters.

The user can request to run the system in *Autonomous Recovery Mode* or in *Manual Recovery Mode*. In *Autonomous Recovery Mode*, the system is self-resilient to different kinds of failures, such as malfunctioning of a service node, failure of service on a node or service node going out of range etc. When any of such failures is detected, the system automatically reconfigures itself with a valid most cost-effective solution or reports a problem in case configuration cannot be run due to failures or user constraints. In *Manual Recovery Mode*, the system will not reconfigure itself automatically in case of failures. This mode is designed to allow human intervention and the reconfiguration is triggered by the user.

4.4.4 Modes of Configuration

The system is designed to operate in *Centralized*, *Distributed* and *Hybrid* mode. In order to make service composition faster and more efficient, we incorporate caching mechanism for composite service graph. The composite service parts of composite service graph are cached in the fabric registry for later use. These cached service graphs are periodically validated by the *Maintenance Fablet*; if the composite service graphs are not valid due to unavailability of their service providers, they are removed from the registry database. Users can compose service by utilizing the cached services or opt to generate fresh configuration.

4.4.4.1 Centralized Mode of Configuration

In the centralized mode of configuration, all the services are configured using a centralized node. All the nodes store information about services they are hosting in a centralized registry. All the services are serially configured in bottom up approach by the centralized node. This mode is prone to single point of failure and can be inefficient as every single change (in any component service) will trigger recomposition of the complete composite service graph. However, this mode of configuration incurs low bandwidth cost because configuration messages are sent to a central node which then communicates with other nodes.

4.4.4.2 Distributed Mode of Configuration

The distributed mode of service configuration makes the system robust against single point of failure. In this mode each service is hosted by a separate node. In order to configure services in a distributed manner, all the component services are sent configuration request as message via the node hosting these services. In our emulation setup, these services are hosted on different *Core* nodes and are connected to the registry via Fabric middleware. At the system startup, every node publishes its functional capabilities, location information and its sensing range to the registry. Upon receiving request for configuration, a service looks for input services that it needs for getting configured. If all the services required by the service are configured, it configures itself otherwise it retries after a specified sleep time τ . After the timer τ expires, the service checks again for the required services and attempts reconfiguration for a specified number of tries denoted by χ , before terminating the configuration process. In such a way, all the services independently configure themselves and finally the requested service composed of other services, is configured. Whenever a service is configured, an entry is inserted in the registry database which is available to other services to peek at and check status of this service. The registry is used for status synchronization among services. In case any service fails, all those services that are not dependent on the failed service are still available and can be utilized without interruption while *Maintenance Fablet* tries to reconfigure the services that are dependent on the failed service.

The benefit of distributed mode is that it adds high level of robustness and configuration granularity to the system and improves the efficiency. Moreover, the capabilities of Gaian database can also be used to seamlessly access disparate databases which increases the reliability. The distributed mode is the most scalable, yet its scalability is limited by the network bandwidth. As the number of services grows, so does data replication (if fully distributed registry is used) and the number of messages sent during reconfiguration process increasing burden on networking. In this mode one configuration message per service is sent, therefore, the number of messages is dependent on the number of services in the system. In a large and fully distributed system, not only the number of messages passed among distributed Gaian database instances will be large (if distributed Gaian DB instances are used), but also service configuration request messages will be large lead-

ing to significant bandwidth consumption in a resource constrained environment. This mode uses Gaian DB at no extra cost because Fabric already uses Gaian DB as backend database (registry) and maintains as well as optimizes the lifecycle of connections with it. Details on performance of Gaian DB can be found here [66].

4.4.4.3 Hybrid Mode of Configuration

The hybrid mode is an attempt to achieve the best of both centralized and distributed modes of configuration. The aim of this mode is to make the system robust against single point of failure while reducing the system specific messages. In this mode, there are more than one configuration nodes that are responsible for configuration of different sets of services. Moreover, if a distributed registry is used, these sets of services are served by separate registry instances which reduces total number of registry instances. This way we can increase the node-to-services ratio (number of services configured by a node) as well as the nodes to registry database ratio (number of nodes served by one instance of registry). Having one registry database serve more than one node decreases the level of data and query distribution and consequently the number of messages transferred across distributed databases. With one node hosting more than one services, we reduce the number of messages sent to the nodes as only one service request message per node is sent to the node for all the services configured through the node.

Along with the benefits, this mode introduces new challenges and solutions to these challenges is out of scope of this chapter. One such challenge is the optimization of parameters, i.e., how many services should be configured by a single node, how the services should be co-located on nodes, how many registry databases should there be in the system. Another problem is, if one registry serves more than one nodes then the registry can also become a single point of failure for all the nodes dependent on this registry instance. We handle this specific challenge by introducing a backup channel to the registry hosting nodes to make the connection of nodes with registry more reliable and resilient. This backup channel is an alternative connection to the registry which is used only for registry related communication, in case the main connection breaks. The backup channel may not offer the same set of functionalities as the main channel but maintains connectivity with nodes and other distributed databases in case main communication channel fails. Such a

backup communication channel enables the distributed database to stay alive and respond to queries.

In our experimental testbed, we configure the CORE nodes hosting registry with two interfaces. One channel which is used as main emulated channel and all the communication (service specific, e.g., sensor readings) among the nodes is done through the emulated channel. For establishing backup channel in our testbed, we use the *CORE* control channel to the registry hosting nodes. The networking of the experimental setup is done in such a way that nodes can connect to registry database via the CORE control channel in case of failure of main channel, but use the main channel for other communication. In such a way, even if the main channel gets disconnected and services on that node fail, the backup channel (*CORE* control channel) provides connectivity to registry database.

4.5 Evaluation and Experimentation

We have performed experimentation on the distributed emulated testbed described in section 4.4. For experimentation, we have used the monitoring scenario and evaluated the models presented in this chapter. We also conducted experiments to judge the efficiency of our system. In our experiments, we configured and composed six different composite services shown in charts as SINK-0-0 to SINK-0-5. These are the sink services providing spatially constrained information to the user. Each sink service is composed of various other component services in fashion depicted in Figure 4.1, a hierarchy with six levels. On average a single sink service is composed of 23 other services selected out of various alternatives services based on the relevancy model in use. In evaluating our relevancy models, we have compared the *Price of Relevancy* of services in addition to *Additive Cost* and *Increase in Relevancy*. In our experiments α and κ were set to 0.5.

We compared *CBM* and *GBM* with a *Simple Relevancy Model (SRM)* for service configuration. The *SRM* essentially minimizes the overall base cost of the configured services but, the *SRM* only selects services with non-empty coverage in the area of interest during configuration process. Therefore, *SRM* does have basic notion of relevancy because all those services which have no coverage in the area of interest are already filtered out before actual configuration takes place using any of the models. We measured the rel-

evancy of the configured services in *SRM* case and compared it with our proposed models to verify that our models actually improve the level of relevancy.

4.5.1 Performance Comparison of the Configuration Modes

We compared the running time of *Centralized*, *Distributed* and *Hybrid* service configuration modes on our emulation testbed. Plots in Figure 4.7 show configuration time of three different modes of configurations. As we can see, the configuration time in all configuration modes remains about the same for configuration of individual service, but time for both *Centralized* and *Hybrid* modes increase as the number of services in a configuration grow (or the composite graph enlarges). In our experiment, there were various composite services with same number of component services (such as 1, 4, 5 etc.), therefore, we see repeated X-axis labels in plot 4.7, but they are actually distinct configured services.

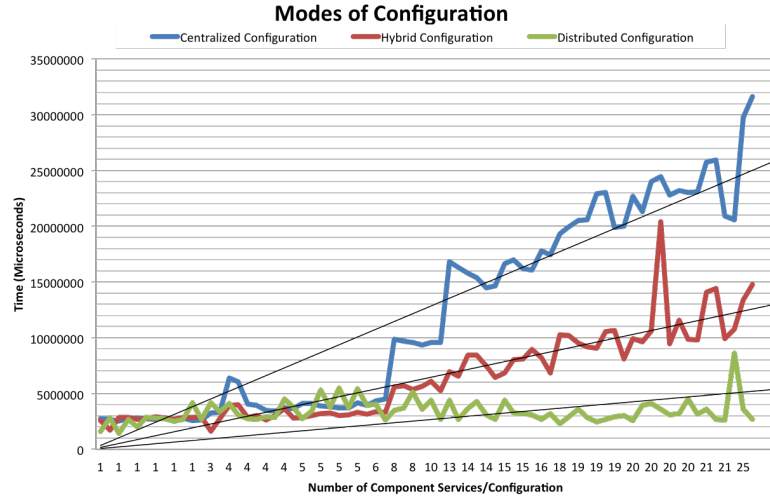


Figure 4.7: Configuration times in various modes

As the composite graph gets bigger, the configuration times in *Distributed* mode slightly increases but not as much as in the other modes. The behavior is in agreement with our earlier discussion (in section 4.4.4) on advantages of *Distributed Mode*. Each service is hosted on a separate node and individually configures itself without depending on a centralized node. This is why services can configure themselves in parallel as long as they do not depend on the output of others. Even those dependent on the output of others configure themselves quicker than in *Centralized* or *Hybrid* mode because all services

start configuring themselves as soon as the request is received. Moreover, this enables distributed mode of configuration to scale well with increasing number of services, as there is minimal or no dependency on other services and nodes. As discussed in section 4.4.4, the *Hybrid Mode* performs midway between *Centralized* and *Distributed*. In *Centralized Mode*, services are configured by a centralized node in *Bottom-up* approach and as the number of services in a composite graph increase the execution time increases as configurations are composed in serial fashion. In *Hybrid Mode*, services are divided in different groups which are configured by nodes responsible for their configuration, therefore a group of services is configured in parallel decreasing the overall configuration time.

To decrease the overall configuration time in *Centralized and Hybrid Modes*, we have incorporated *service caching* mechanism as described earlier. This saves time by storing frequently used services or services configured in recent past, in the registry database. These services act as pre-configured part of other complex services thereby saving time which will otherwise be spent on their configuration. We have noticed in our experiments that *caching services* reduces configuration time, but it depends on the granularity of the cached services, i.e., component services with larger number of services (larger part of the service graph) save more time than caching non-composite single services or in our case the source services.

4.5.2 Results of Cost Based Model (CBM)

We evaluated the *Cost Based Model (CBM)* for relevancy and compared it with *SRM*. Figure 4.8 shows the comparison of prices of configured services in *CBM* and *SRM*. As seen in the plots the *Price of Relevancy* is lower in *Cost Based Model (CBM)*; lower price means that by using *CBM* the user is getting more relevant information (or value from service) for the payment made. In other words, *CBM* provides more relevant information for the cost incurred as compared to *SRM*.

We further analyzed the relevancies yielded by both models and compared the increase in “relevancy” and decrease in “prices” to judge the advantage that our *CBM* has over the *SRM* model. The gains are defined by Eqs. (4.8-4.9). The Γ_{CBM} , Γ_{SRM} represent relevancies and P_{CBM} , P_{SRM} represent prices in *CBM* and *SRM* configurations, respec-

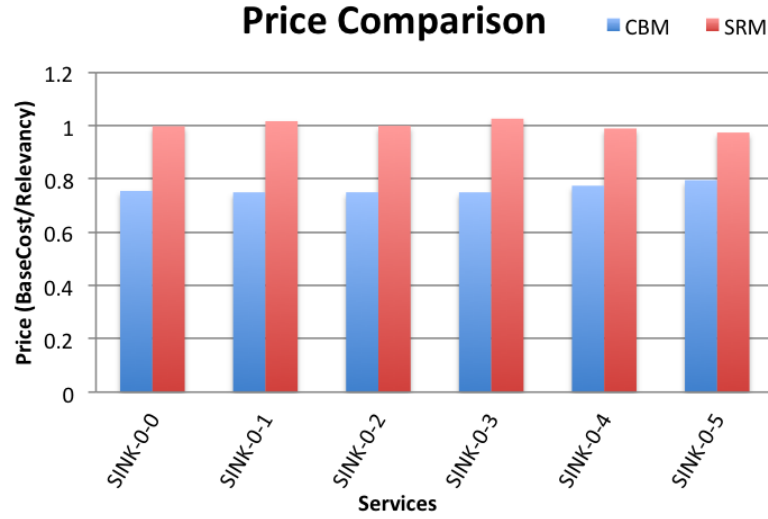


Figure 4.8: Prices in CBM

tively.

$$\text{Increase in Relevancy} = 100 \times \frac{\Gamma_{CBM} - \Gamma_{SRM}}{\Gamma_{SRM}} \quad (4.8)$$

$$\text{Decrease in Price} = 100 \times \frac{P_{SRM} - P_{CBM}}{P_{CBM}} \quad (4.9)$$



Figure 4.9: Price and relevancy change in CBM

Figure 4.9 shows that *CBM* configured services produce up to 50% more relevant information than those of *SRM*. Moreover, the *CBM* configurations incur on average 30% lower prices than *SRM* configurations. These results demonstrate that our *Cost Based*

Model leads to low prices and configures highly relevant services as compared to services that are configured just based on the minimizing the base costs.

4.5.3 Results of Cost Based Model (CBM) vs. Gain Based Model (GBM)

We evaluated *GBM* and compared it with both *CBM* and *SRM* approaches. Figure 4.10 shows price comparison among models being compared. We see that both *GBM* and *CBM* incur low prices, the prices incurred by *CBM* are slightly lower than those of *GBM*, but both of these approaches lead to highly relevant information for the cost incurred as opposed to *SRM* approach which minimizes base cost but leads to low relevancy as well.



Figure 4.10: Price comparison in two models

Figure 4.11 compares the relevancy gains that the two models *CBM* and *GBM* yield compared to *SRM* and shows that the two models are complementary. We can see that despite slightly higher prices *GBM* produces up-to 250% more relevant information compared to 50% of *CBM* which meets the purpose of the *GBM* model. *GBM* is designed to deliver high relevancy services and optimize services such that the configured services yield high gains. Please note that the formulation of *GBM* is oriented towards gain achieved without putting any limit on the base cost therefore for this model, we expect the base cost to go very high.

Figure 4.12 shows *Additive Base Cost* and *Price* comparison of the three models being compared. It is important to notice that the three models behave as expected. Consider the *SRM* approach, it is designed to incur low additive cost without optimizing for relevancy of individual services and the composite services, so it yields the lowest ad-

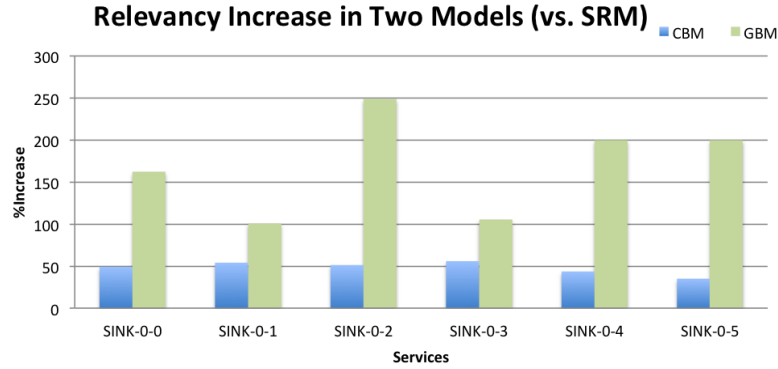


Figure 4.11: Comparison of relevancy in two models

ditive base cost but at the same time incurs very high prices due to low relevancy. The *GBM* on the other hand is designed to deliver highly relevant services which it achieves but at the same time, it incurs high base cost. This is because while selecting services for configuration the model does not explicitly take into account the base cost of the component services, it only optimizes gain based on the relevancy that a service brings, which is why it selects highly relevant services even if they have high base cost. In contrast the *CBM* optimizes a generic cost that is composite of both base cost as well as the cost of irrelevancy of a service. This results in selecting services that are both relevant and cost effective. That is why we see the additive base cost in *CBM* is slightly higher than *SRM* but is much lower than *GBM*.

The results show that both models, *CBM* and *GBM*, effectively capture the concept of relevancy and configure services with high relevancy to the area of interest to the user. The results also show that without optimizing the relevancy of services during the configuration process results in configurations with poor relevancy. Therefore, such service configuration will result in delivery of information that is not relevant to the user's interest. On the other hand, our relevancy aware service configuration models capture the concept of relevancy while fulfilling the service configuration requests. However, as we observed in the results, different relevancy models are appropriate in different application scenarios. When highly relevant information is needed and there is no limit on budget or the *BaseCost* of alternative services do not vary much, *GBM* is a good model to use. Nevertheless, if there are budget restrictions or the *BaseCost* variation is high among services of different relevancies, the *CBM* is the way to go, because it strikes a good balance

between additive base cost and overall relevancy of the configured service.

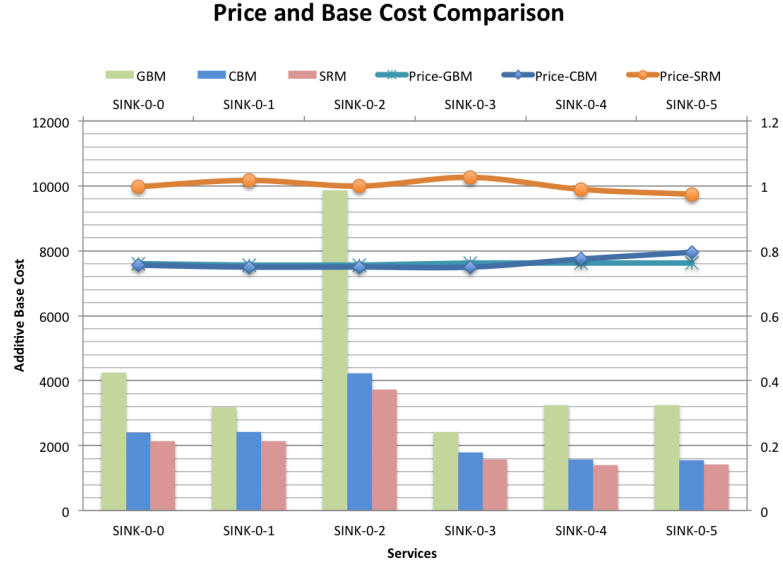


Figure 4.12: Comparison of two models

4.5.4 ROI Based Service Configuration with Budget Constraints

We conducted simulation to evaluate the *ROI* based approach for a given constraint. We fixed the budget to 1000 and tried to efficiently select services that can be configured within the budget limits. Figure 4.13 shows the gain achieved and the number of services used in a configuration. It is clear from the figure that the number of services in a configuration and the total gain achieved from the configuration increases as the budget limit increases. However, services are added only when they lead to high return relative to their cost, this rule causes empty pockets in the graph where no additional services are added, i.e., between 92 – 199, 201 – 223. In our simulation, we select a service if it can be fully obtained within the available budget limitations. The gain shown in the Figure 4.13, is the total gain from all services in a configuration and is calculated using Eq. (4.10). As already explained through examples in the section 4.3.4, this mechanism selects services based on the “ROI” and optimizes the overall return on investment.

$$Total\ Gain = \sum_{x=1}^n (ROI_x \times cost_x) \quad (4.10)$$

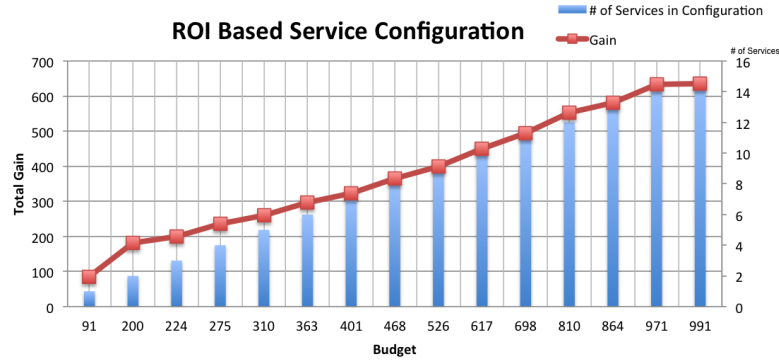


Figure 4.13: ROI based service configuration for a given budget

4.6 Concluding Remarks

In this chapter, we explored how SOA design can be utilized to manage networks. We investigated how services can be configured together to perform complex tasks. We also studied how spatial relevancy of the sensor services can be incorporated in service configuration. In this chapter, we have mainly focused on geospatial relevancy of services but the techniques and models presented in this chapter are applicable to user defined relevancy models. We have proposed two models namely “Cost Based Model (CBM)” and “Gain Based Model (GBM)” for capturing relevancy of services. The “Cost Based Model (CBM)” uses both cost and relevancy of the service to optimize the configured service, whereas the “Gain Based Model (GBM)” focuses more on the gain that services bring in terms of relevancy to the configured service. We also present “Return On Investment (ROI)” based service configuration which optimizes service selection for a given budget based on “ROI”. The main advantage of “ROI” based mechanism is that the user can optimize budget over multiple applications. We have shown that these models produce services that are highly relevant to the area of interest of the user as compared to models that do not consider the spatial relevancy of the services in configuration process. Moreover, we demonstrate the architecture of our service oriented system is self-recovering and capable of performing configuration in various modes. The system is highly extendable and allows new models and services to be integrated in the system.

CHAPTER 5

Service Configuration in Constrained Environments: Policies

We explore how to manage networks using policies and how network resources (e.g., sensor services, sensor nodes etc.) can be shared among various parties. The network resources may be owned by different entities and organizations forming coalition networks. We study how to map and model various high-level configuration restrictions into sets of policies and rules, that would empower network managers of multiparty coalitions to define limits on the administrative and user level capabilities of the individuals or parties. In the past, researchers have suggested various tools and techniques for defining policies and access control mechanisms. These mechanisms are designed to be used by programmers and technical experts, however, a general user of the network or high-level strategic decision makers might not be acquainted with specific tools or programming languages. Often times, the decision makers are higher in managerial hierarchy of the organization (e.g., generals in army) so policy tools requiring expertise might not be very useful to such people. Moreover, the user of a network might be on the edge of a network (e.g., squad leader in army) and may not have expertise in handling complicated policy management and negotiation tools.

We present “Restriction Set Theoretic Expressions (RSTE)” to represent assets and policies in the form of sets at the system level, therefore *RSTE* is independent of high-level representation of policies and assets [67]. Yet, such high-level representation can be easily translated to semantically equivalent *RSTE* sets and then different *RSTE* operations are applied to restrict or release access to resources. *RSTE* defines sets and operations that can be performed on the sets to implement policies. We describe semantics of *RSTE* sets and operations for assets in service configuration in WSNs and show how the services

Portions of this chapter previously appeared as: S. Y. Shah and B. Szymanski, “Dynamic policy enforcement using restriction set theoretic expressions (rste),” in IEEE Conf. on Military Commun. (MILCOM), 2014, pp. 198–203.

Portions of this chapter previously appeared as: S. Y. Shah, B. Szymanski, P. Zerfos, M. Srivatsa, C. Gibson, and D. Harries, “Policy enabled real-time information flow control across multiple administrative domains,” 2013. [Online]. Available: <https://www.usukita.org/sites/default/files/ITA-ShortPaper-2013.pdf> (Date Last Accessed, Oct., 28, 2014).

and policies can be represented as sets. We also show how *RSTE* can control information flow among multiple administrative domains.

5.1 Introduction

Various languages and frameworks have been proposed for policy description, management and application [46, 68]. Every approach has its features and shortcomings. Recently, research on expressing and managing policies in human understandable language is getting significant traction. Researchers have used controlled natural language (CNL) to express user requirements and efforts are being made to devise frameworks that can manage policies using human readable languages [69]. However, translating these requirements from human readable format into system level constraints is a challenging issue. In general, policy representation frameworks tightly couple the actual implementation of policies with the higher level policy representation. This tight coupling leaves minimal flexibility for using alternative low-level policy implementation. Policy languages that are highly humanly expressive such as CNL based policies, have lower policy enforcement capabilities. On the other hand policy languages that support very sophisticated operations on system level are coupled with representation layer which is very programming oriented and too complex for an end user. Therefore, there is a need for decoupling the presentation layer from low level policy execution, so that policies can be expressed in user friendly format at presentation level and can be easily transformed into lower level policies for actual enforcement. Thus, maintaining a required level of semantic compatibility, the low-level policy enforcement and higher level representation should be as independent as possible to achieve design goal at two separate levels.

In this chapter, we consider system for service configuration in a mobile sensor network at the application layer [70]. Each service hosted in the sensor network produces one or more outputs and requires zero or more inputs from other services, thus composite services are hierarchical in nature forming a workflow graph for data processing. In a coalition environment, services are owned by different collaborating organizations or partners, these services are hosted on sensor nodes owned by different partners. Policy makers define policy rules to restrict or allow access to the resources in the network. Users make requests for services with certain requirements, the requests are handled by

the configuration system and based on policies and users' authorization corresponding services are configured for the requestor. In this chapter, we show how *Restriction Set Theoretic Expressions (RSTE)* works for such a service configuration scenario. Details of *Application Layer* were discussed in the previous chapter.

Here, we focus on producing policy complaint services for mission specific applications [70]. Such missions include, multi-organization led rescue and recovery missions, military mission, monitoring and surveillance etc. Policy enforcement in application used in such dynamic scenarios are more challenging than traditional policy enforcement where all assets are indigenously owned by a single organization and the flow of data is pre-defined for applications. In dynamic service composition, the service graph is not known apriori as the status of the mission at a particular time defines the architecture of the services, moreover, services which are owned by different organizations dynamically appear and disappear during the course of a mission. Another important aspect of such systems is the nature of the policies. Policies in such cases can be temporal in nature and as they are updated/negotiated by policy makers the effect should be instant. So, the challenges are:

1. how to enforce such dynamic policies for equally dynamic services with minimal delay after a change in policy and report over restricting conditions for negotiation;
2. how to make policy description and management user friendly without overly limiting the capabilities of policy enforcement engine.

Emergency Response Coalition Scenario

Imagine an emergency response coalition formed between US and UK forces to evacuate a disaster stricken area. A sensor network is setup to monitor the area as well as to provide support to the rescue forces using composite sensor services. Network assets are owned by either US or UK, therefore, sharing policies have been defined to allow partners to access each others' resources. As other nations join the rescue mission, more policies are defined to manage resources in coalition. Resources are not always equally accessible to all partners but as the situation evolves policies are defined to provide access to coalition partners. We show policy enforced services sharing using *RSTE* for such a scenario.

5.2 Restriction Set-Theoretic Expressions (RSTE) for Service Configuration

We present a *Restriction Set Theoretic Expressions (RSTE)* language, that is simple yet expressive enough to map user requirements to lower level system constraints so that lower level execution engines can produce desired result. This set language is understandable by both higher layers of framework dealing directly with the user and low level system engines. Therefore, this language enables two way information flow from user to the system level objects and back. Figure 5.1 shows layered view of three major components of the system.

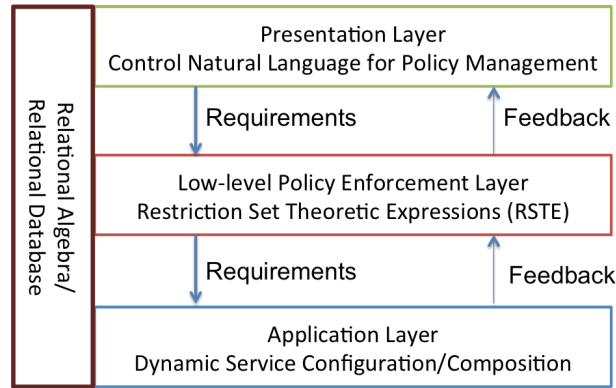


Figure 5.1: Set language is incorporated as layer between presentation layer and application

For high level of user friendliness in our approach, we propose that policies are represented in Controlled English [69] as Controlled Natural Language (CNL) on presentation layer, as shown in Figure 5.1, however this is not a requirement for *RSTE*. Policies can be represented in any language at *Policy Presentation Layer*, but the policies and constraints should be compliant with or transformable to semantics of *RSTE*.

The *CNL for Policy Management* layer is responsible for policy management on presentation layer. Policies are defined and modified at this level. If there are any conflicts between different policies, this layer also resolves them by running conflict resolution algorithms. Moreover, this layer performs negotiations in case conflicts cannot be resolved between policies or no valid system configuration can be produced under specified policies.

The *Dynamic Service Configuration/Composition* layer is responsible for producing

policy enforced low cost composite services based on user requirements. This layer deals with input/output matching of different services as well as spatial relevancy constraints to produce composite services that are cost effective and geo-spatially relevant to the user's interests. In case this layer fails to produce service configuration requested by the user, it notifies the *Controlled Natural Language for Policy Management* via *RSTE* layer about the need for appropriate actions. The *Dynamic Service Configuration/Composition* layer accesses resources that are policy constrained by the *RSTE* layer, therefore this layer considers only resources that are accessible by the requesting user in compliance with policies. If the resources are too scarce for service composition, it generates corresponding output message that is sent to the presentation layer. We have provided detailed description of the *Service Configuration* in the previous chapter.

5.2.1 The Semantics and Operations of RSTE Language

In this section we define *RSTE* operations and semantics for *RSTE* sets. A request for service is represented by *SReq* set and its corresponding service response is represented by *SRes*. Following are the sets used by *RSTE* for representing assets, policies, requests/response and users. Alongside these sets we define various set operations, such as *Union*, *Intersection*, *Subtraction*, *Cardinality*, that are performed on these sets to produce a restricted set of assets which is then provided to the system layer for configuration.

Service Set = $SS = \{\{ServiceName, NodeId, GeoSpatialCoverage, Capabilities, Ownership\}\}$

Node Set = $NS = \{\{NodeId, Ownership, Location, Conditions, Permissions\}\}$

Service Request Set = $SReq = \{\{RequestId, UserId, ServiceName, Capabilities, Properties, Restrictions\}\}$

Policy Set = $PS = \{\{PolicyId, ServiceOwnership, ServiceName, Conditions, UserAffiliation, Restrictions, Action\}\}$

User Set = $US = \{\{UserId, UserName, UserAffiliation, Role, UserProperties\}\}$

RSTE Response Set = $RRS = \{\{ServiceName, NodeId, GeoSpatialCoverage, ServiceCapabilities, ServiceOwnership, PolicyId, PolicyConditions, UserAffiliation, PolicyRestrictions, RequestId, UserId, SReqCapabilities,$

$SReqRestrictions, Action\}\}$

Service Response Set = $SRes = \{\{RequestId, Logs, ReturnValue, Failures\}\}$

5.2.2 Definitions of the Sets

1. *Service Set (SS)*: This set represents the snapshot of the available services. This set is provided by the upper layer i.e., the *CNL Layer*. This set is subset of all the services hosted in the asset database. During the set operations this set is filtered based on policies and restrictions applied by the user.

Table 5.1: Service Set

Set Element Name	Description
Service Name	Name of the service, e.g., LOBR
NodeId	Unique ID of node hosting the service
GeoSpatialCoverage	Geospatial coverage provided by the service
Capabilities	Capabilities provided by the service, e.g., $\{<Sensor, Video>, <Resolution, HD>\}$
Ownership	Organization that owns the service, e.g., US, UK.

2. *Node Set*: This set represents nodes that host member services of *SS*. This set is used to access detailed information about the infrastructure hosting the service and will be used as needed.

Table 5.2: Node Set

Set Element Name	Description
NodeId	Unique ID of node hosting the service
Ownership	Organization that owns the node, e.g., US, UK.
Location	Physical Location of the Node, e.g., $<LAT, LONG, ALT>$
Conditions	Conditions that applies to usage of the Node.
Permissions	Permissions on this node

3. *Service Request Set*: This set describes service requests made by users.

Table 5.3: Service Request Set

Set Element Name	Description
RequestId	Unique ID of the service requests to distinguish different requests.
UserId	Unique ID of the user who requested the service.
ServiceName	Name of the service that has been request.
Capabilities	Capabilities that are required by the service, e.g., High resolution video
Properties	Configuration properties of the service, e.g., {< <i>Mode of Operation, Distributed</i> >}
Restrictions	Restrictions on the requested service, e.g., {< <i>Ownership, US</i> >}

4. *Policy Set*: The *Policy Set* represents policies that need to be applied to the *SS* using various *RSTE* operations. This set captures various other aspects of the policies (e.g., restrictions) along with *conditions* and *actions* policy enforcement paradigm.

Table 5.4: Policy Set

Set Element Name	Description
PolicyId	Unique ID of a policy
ServiceOwnership	Owner organization of the service affected by the policy, e.g., UK.
ServiceName	Name of the service affected by the policy.
Conditions	Conditions on the use of the service, e.g., {#service instances < 5}
UserAffiliation	Affiliation of the user requesting the service, e.g., US or UK
Restrictions	Restrictions on the use of the service, e.g., {< <i>Role, Commander</i> >}
Action	Access to service based on the policy, i.e., Allow or Deny

5. *User Set*: User Set describes user of the network. Each user has certain properties as well as associated organization. The *UserId* enables the set to access more de-

tailed information about the user from the detailed assets database in case details are needed. This set is primarily used for policy validation through *RSTE* operations. Various conditions are matched based on the user profile before authorizing user to access services.

Table 5.5: User Set

Set Element Name	Description
UserId	Unique ID of a user.
UserName	Name of the user.
UserAffiliation	Affiliated organization of the user, e.g., US, UK
Role	Role of the user in organization or in a mission, e.g., $\langle Role, Commander \rangle$
UserProperties	Other properties of the user, e.g., $\langle SkillSet, Expert \rangle$

6. *RSTE Response Set*: The *RSTE Response Set* is the set that is supplied to system layer along with *Service Request Set (SReq)* for final service configuration. This set contains services that fulfill the policy and user requirements.
7. *Service Response Set*: The service response set is primarily for feedback purposes. This set is supplied to the *CNL Layer* in order to be presented to the user or to negotiate/relax policies.

5.2.3 The RSTE Operations

The *RSTE* operations are basically normal *Set Operations* with the extension that these operations can also be performed based on a specific condition (which can be imposed on set members), which is checked in the corresponding sets. Consider two sets *A* and *B*, members of these sets are also sets of same semantics. Let $\{e-1, e-2, e-3\}$ be the semantics of sets *A* and *B*.

$$A = \{\{1, 2, 3\}, \{a, b, c\}\}$$

$$B = \{\{1, 5, 10\}, \{e, f, g\}, \{a, x, y\}, \{a, b, c\}\}$$

1. **Union (\cup)** The union operation combines two sets into one. An union operation is defined as,

Table 5.6: RSTE Respnse Set

Set Element Name	Description
Service Name	Name of the service, e.g., LOBR
NodeId	Unique ID of node hosting the service
GeoSpatialCoverage	Geospatial coverage provided by the service
ServiceCapabilities	Capabilities provided by the service, e.g., {<Sensor, Video>, <Resolution, HD>}
ServiceOwnership	Organization that owns the service, e.g., US, UK.
PolicyId	ID of policy applied to this service.
PolicyConditions	Conditions from policy applied to this service.
UserAffiliation	Affiliation of the user, e.g., US, UK.
PolicyRestrictions	Restrictions specified by policy.
RequestId	ID of the user's request.
UserId	ID of the user of this service.
SReqCapabilities	Capabilities requested in the Service Request
SReqRestrictions	Restrictions specified in the Service Request.
Action	Authorization action of policy on this service.

Table 5.7: Service Response Set

Set Element Name	Description
RequestId	Unique ID of the service request
Logs	Any logs produced by the service configuration at the system level.
ReturnValue	Exit state of the service configuration, e.g, 0,1,-1.
Failures	In case of failed configuration, reason of failure.

$$A \cup B = \{x : x \in A \text{ or } x \in B\}$$

A union with a condition v can be written as,

$$(A \cup B)_v = \{x : x \in A \text{ or } x \in B \text{ and } v = \text{true}\}$$

A simple union of the two sets will be as follows,

$$A \cup B = \{\{1, 2, 3\}, \{a, b, c\}, \{a, x, y\}, \{1, 5, 10\}, \{e, f, g\}\}$$

A union operation defined on a condition " $v : (A.e-1=B.e-1)$ ", produces following set,

$$(A \cup B)_v = \{\{1, 2, 3\}, \{1, 5, 10\}, \{a, b, c\}, \{a, x, y\}\}$$

2. **Intersection** (\cap) The intersection of two sets results into a set with common members among the sets. An intersection operation is defined as,

$$A \cap B = \{x : x \in A \text{ and } x \in B\}$$

An intersection with a condition v is written as,

$$(A \cap B)_v = \{x : x \in A \text{ and } x \in B \text{ and } v = \text{true}\}$$

A simple union of the two sets is,

$$A \cap B = \{\{a, b, c\}\}$$

An union operation defined on a condition “ $v : (A.e-1=1)$ ”, produces an empty set.

$$(A \cap B)_v = \{\}$$

3. **Difference** ($-$) The difference or subtraction is the relative complement of set B in A . The difference operation is defined as,

$$A - B = \{x : x \in A \text{ and } x \notin B\}$$

A difference operation with a condition v is written as,

$$(A - B)_v = \{x : x \in A \text{ and } x \notin B \text{ and } v = \text{true}\}$$

A simple difference of the two sets will be as follows,

$$B - A = \{\{1, 5, 10\}, \{e, f, g\}, \{a, x, y\}\}$$

A difference operation defined on a condition “ $v : (A.e-1=a)$ ”, produces set.

$$(B - A)_v = \{\{a, x, y\}\}$$

4. **Cardinality** ($| \ |$) Cardinality of a set gives the size of a set. In above example, $|A| = 2$ and $|B| = 4$.

5.2.4 Operations for Policy Restrictions

First we need to find out policies that apply to certain user, then we will be able to find out services that a specific user is authorized to access contingent on conditions and restrictions. Moreover, there may be services that are not policy enforced, such services should be available to all the users to use. In order to find policies that need to be applied to a service configuration request, we apply series of operations to the policy set (PS).

1. $PS_1 = (PS \cap US)_v \quad v = UserAffiliation$

$$2. PS_2 = (PS_1 \cap US)_v \quad v = Role \wedge UserProperties$$

$$3. SS_{PolicyEnforced} = (SS \cap PS)_v \\ v = ServiceName \wedge ServiceOwnership$$

$$4. SS_{PolicyAllowed} = (PS_2 \cup SS_{PolicyEnforced})_v \\ v = ServiceName \wedge Ownership$$

Now we find all those services, which have no restrictions on them or there is no policy defined to restrict them. These services are available to all the users. To find services with no policy restrictions, we perform following operations.

$$5. SS_{NoPolicyEnforced} = (SS - SS_{PolicyEnforced})$$

$$6. SS_{AllAllowedServices} = (SS_{PolicyAllowed} \cup SS_{NoPolicyEnforced})$$

$$7. SS_1 = (SS_{AllAllowedServices} \cup SReq)_v \\ v = Capabilities$$

$$8. RSTE_SRes = SS_1 \quad \text{Process based on restrictions and conditions}$$

5.3 RSTE Implementation

We have implemented the prototype of *RSTE* language using JAVA and relational database. The *Relational Algebra* in relational databases provides a natural support for *RSTE* sets and operations, therefore the capabilities of relational databases can be used for efficient implementation of *RSTE*. The sets in the language and tables in a relational database are analogous. We represent every set by a corresponding table. We then perform different operations on the tables using relational algebra and create views for intermediate steps. These views are then systematically processed to produce other views and finally the view that represents the *RSTE Response Set*, which is further processed before providing it to the service configuration layer. For some of the operations shown in section 5.2.4, we have created “Views” in the database, whereas the operations that deal with temporal requirement or that need more fine grain processing e.g., Step-6 are done via post-processing on *RSTE Response Set*.

When a service request is received from the user, it is represented by *SReq* set and corresponding entry is made in the database table. The views automatically get updated and set of all allowed services are filtered out. Now, if the user has defined any specific restrictions or there are policies related to the user requesting a service, further processing is done via JAVA libraries, after which allowed services are provided to the *Application Layer* for service configuration. For certain services or users very restrictive sharing policies may be defined by the policy makers. Such overly restrictive policies limit the services available for configuration and these services might not be sufficient for a successful configuration leading to failure in meeting the user's request. In case the service configuration returns failure status, the service configuration at *Application Layer* makes procedure calls to *RSTE* to invoke backtracking. The backtracking mechanism finds out condition(s) that can be relaxed in order to produce a configuration. The condition(s) are then sent to policy management layer for policy negotiation. When such a condition is relaxed in the policy, the resources are immediately available to the user and service can then be configured enforcing the updated policy.

5.3.1 RSTE Backtracking

Policies specified by policy makers might be very restrictive and may constrain resources necessary for service configuration. In a very restrictive case, operations performed in step by step policy application by *RSTE* may lead to empty sets or empty views in our implementation. This means that there are no services available for configuration. At any such step where we detect an empty set we back track to previous step and negate the conditions to see if that leads to a non-empty set. If so we can suggest this negation to *CNL layer* as relaxation for policies involved. In case there are more than one conditions applied at a particular step, each condition is negated separately and the resultant set is checked for not being empty, this way negotiation can be performed at more granular level and only over-restrictive conditions will be negotiated/relaxed.

Figure 5.2 shows the backtracking mechanism in the *RSTE* implementation. As shown in the figure 5.2, when condition "C4" of the policy is applied at "Step4" all services are filtered out yielding an empty set; at this stage the policy enforcement backtracks and applies negotiation of condition "C4" to go to "Step4N". If at this stage the result set

and (partner ‘US’ owns service S) then (the user U hasAccess to service ‘CAMERASERVICE’)

User Request

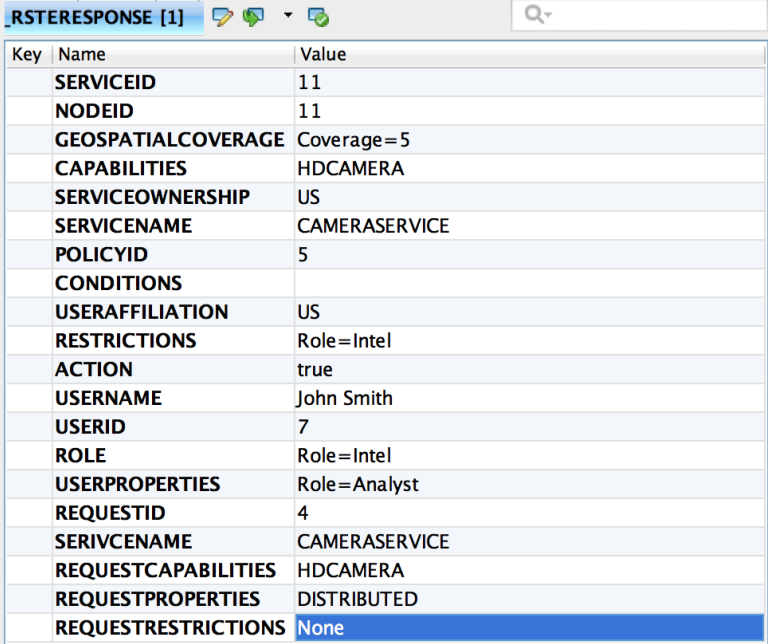
A user “John Smith” requests for a service, the user “John Smith” is represented in *User Set* in the system as follows,

$$User\ Set = US = \{7, JohnSmith, US, Intel, Analyst\}$$

User’s request is represented by following set in the *RSTE*,

$$SReq = \{4, 7, CAMERASERVICE, HDCAMERA, Distributed, None\}$$

As soon as the above request is made by “John Smith”, the system enforces policy and produces filtered services that this user is authorized to access based on user’s particulars. Following is the corresponding *RSTE_RESPONSE* entry produced by the system. As



Key	Name	Value
SERVICEID		11
NODEID		11
GEOSPATIALCOVERAGE		Coverage=5
CAPABILITIES		HDCAMERA
SERVICEOWNERSHIP		US
SERVICENAME		CAMERASERVICE
POLICYID		5
CONDITIONS		
USERAFFILIATION		US
RESTRICTIONS		Role=Intel
ACTION		true
USERNAME		John Smith
USERID		7
ROLE		Role=Intel
USERPROPERTIES		Role=Analyst
REQUESTID		4
SERVICENAME		CAMERASERVICE
REQUESTCAPABILITIES		HDCAMERA
REQUESTPROPERTIES		DISTRIBUTED
REQUESTRESTRICTIONS		None

Figure 5.3: RSTE Response Set entry for John Smith’s request

shown in the figure 5.3 there is only one service that is available for the user, this is because the policies in the system allowed only a subset of services to be available to this user and then the service request had restriction on the services to have capability of a “HDCAMERA” which further reduced the number of services to only one, which

satisfies both policies and user's criteria.

To test backtracking mechanism, we modified the role of user "John Smith" from "Intel" to "Soldier". This produced an empty set and the user no longer can access "CAM-ERASERVICE" that he could with "Intel" role. The backtracking mechanism backtracks to condition which leads to empty set. The backtracking is only performed on operations that filter services based on policies. So, when we ran queries with negated conditions, we found following query that restricts services based on roles. When that query was negated a non-empty set was produced which signified condition based on roles as over-restricting and the condition was reported for negotiation. The original query was

```
SELECT "PS.ID", "PS.PolicyId", "PS.ServiceOwnership", "PS.ServiceName",
"PS.conditions", "PS.UserAffiliation", "PS.Restrictions", "PS.Action", "US.UserName", "US.UserId"
FROM "PS", "US" WHERE "PS.UserAffiliation" = "US.UserAffiliation" AND "US.UserName"
= 'John Smith' AND ( "PS.Restrictions" = "US.Role" OR
"PS.Restrictions" = "US.UserProperties" );
```

when condition on roles is negated as

```
... WHERE "PS.UserAffiliation" = "US.UserAffiliation" AND "US.UserName"
= 'John Smith' AND not ( "PS.Restrictions" = "US.Role"
OR "PS.Restrictions" = "US.UserProperties" );
```

the query produced results. The condition "PS.Restrictions" = "US.Role" OR "PS.Restrictions" = "US.UserProperties" is reported for negotiation. This meets our expectations as we changed the role of John Smith from "Intel" to a "Soldier".

5.4 RSTE Evaluation

In this section, we present results of *RSTE* evaluation. We evaluated the policy enforcement, backtracking aspects as well as the policy execution time of *RSTE*. As mentioned in section 5.3, we have used JAVA and Derby Relational Database to implement *RSTE*. In our experiments, we used *RSTE* to policy enforce sensor service configuration and restrict access to the WSN assets and services. In total, we used 50 services and 0 – 50 policies for our experiments.

In order to test the runtime performance of *RSTE* enabled service configuration that restricts access to services based on the attributes specified in the policy, we benchmarked

the performance of our mechanism against IBM WPML (Watson Policy Management Library) [39] based policy enabled service configuration published as *PASC* [46] in the literature. IBM-WPML is professionally built and optimized framework for policy enforcement in networks. We inserted several policies in the policy set and configured services based on the restricted set produced by *RSTE* and then compared the configuration time of services in both the approaches. Figure 5.4 shows configuration time of services with increasing number of policies relative to 1 policy. The results shown in the Figure 5.4 are averaged over several experimental runs. For plotting performance of PASC/WPML, experimental data from their publication [46] was used.

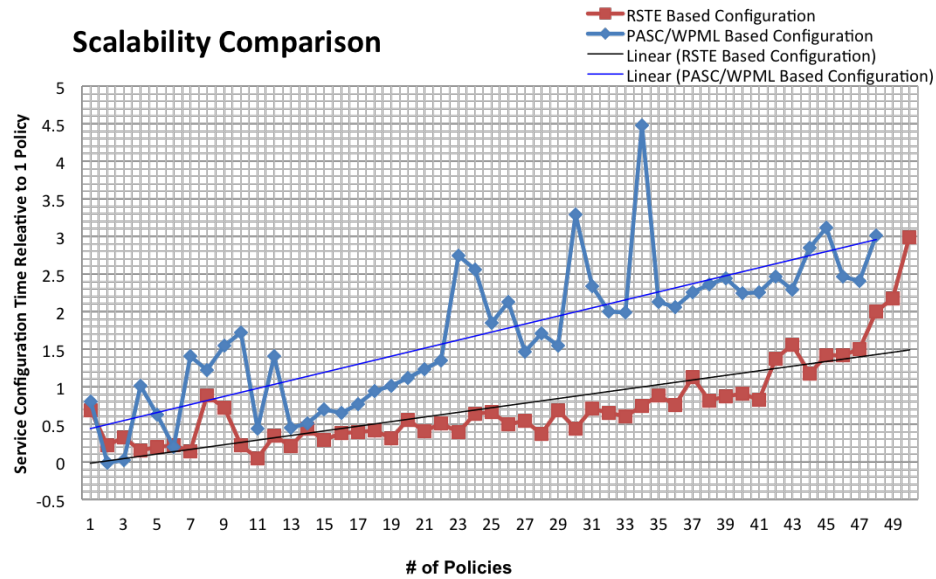


Figure 5.4: Scalability comparison of RSTE based service configuration with WPML Based PASC

We can see from Figure 5.4, that the performance of RSTE is comparable and even slightly better than WPML based policy enforcement. Both the approaches have linearly increasing configuration time with the increase in the number of policies, but the slope of *RSTE* trend-line is slightly lower than its counterpart. We use relative performance for scalability because the absolute configuration time for the two comparative approaches was measured on different hardware, therefore, it makes more sense to plot relative performance rather than absolute values.

We conducted experiments to evaluate the policy enforcement and backtracking capabilities of *RSTE*. We restricted access to services using policies, with every policy

enforcement there is one less service available which is what we were expecting as each policy was affecting only one services (except for one wildcard policy). We enforced policies ranging from 1 to 37 and as a result the number of services was reducing, in addition we reported the *RSTE* backtracking information to notify the restricting conditions. The backtracking correctly reported *Policy Ids* along with the restricting conditions that were blocking access to the services.

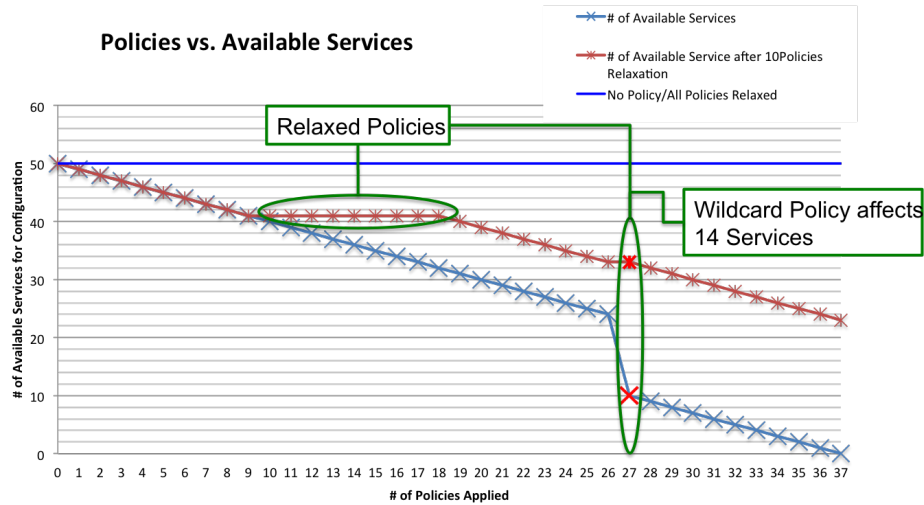


Figure 5.5: Dynamics of services availability with RSTE based policy enforcement and relaxation

Figure 5.5 shows the number of services that are available for configuration under various number of policies. The results are averaged over several runs. If there are no policies then all the 50 services are available for configuration, this is depicted by solid blue line. As we add policies that restrict access to the services, the number of available services decreases linearly as shown by crossed blue line in the Figure 5.5. There is a sharp dip in the number of available services as a result of policy 27, this is because *RSTE* supports wildcard policies which apply to more than one services. The wildcard policies in *RSTE* may apply to multiple services using *, e.g., the corresponding entry in the policy set for Policy 27 is {27, “KWC”, “*”, “None”, “CW”, “Role=LocalCoordinator”, “true”}, which simply means for all services owned by organization “KWC”, allow access to users affiliated with organization “CW”, if they have the role of a “LocalCoordinator”. Policy 27 restricts access to 14 services, therefore as soon as this policy is inserted

in the database/Policy Set the number of available services drop from 24 to 10 and finally to 0.

With each policy application, we also reported restricting conditions and based on that information we relaxed 10 policies. The brown crossed line shows that as we relax policies the number of available services stays constant. Please note that Policy 27 was also relaxed which is why we do not see the sharp dip and with 10 policies relaxed out of 37, we have 23 services available for configuration. This shows that *RSTE* effectively enforces policies and correctly reports restricting conditions. Such restricting conditions can be used to negotiate or relax policies to provide access to services in overly restricted environments.

5.5 Dynamic Information Flow Control using RSTE in Multiple administrative Domain Environments

5.5.1 The Data Sharing Problem

Information sharing and data distribution among multiple administrative domains is a well known issue. Research has been done on policies and policy aware data distribution in many fields of computer science including sensor networks [46]. However, the policy enforcement of the prior work on policy based data distribution assumes that the data producers and consumers are known a-priori so the rules and connections among data sources and data consumers can be predefined. In such a scenario, the consumers of the data have prior knowledge of the kind of data they need and also know the data providers. Similarly, the producers know their consumers so they can predefine rules for data sharing based on the attributes of the consumers. Such static linking of producers and consumers are feasible for scenarios where data sharing parties and administrative domains are well established and the parties do not change once the data sharing process has started and the data starts to flow. Such kinds of mechanisms are established between large organizations and data banks, e.g., hospitals, insurance companies, telecommunication companies and other government and private organizations. Therefore, graphs and networks for such data flows are well-defined. Other examples of such static linking of parties are workflows. Workflows are steps of statically linked operators, where each operator performs some function on data and sends the result to next the operator. Therefore, the data follows

a defined path from producer to consumer. Data rates in such flows can be very fast as they are designed to analyze realtime data such as network analytics and real-time data analysis etc., [71]. All the operators of a workflow are defined at compile time and the workflow graph remains typically static once the execution starts.

For more dynamic situations like ad-hoc networks, where data consumers and producers continuously join and leave the network, mechanisms like publish-subscribe (pub-sub) have been developed to decouple producers and consumers [61]. Therefore, the producers of the data need not to know the consumers of the data and vice-versa. Such a data sharing mechanism is largely used in mobile ad-hoc networks and wireless sensor networks. However, these pub-sub platforms do not provide extensive functionalities including high speed data analytics, real-time data processing, automatic load distribution etc., that are provided by the workflow based frameworks. Some of the workflow based analytics systems such as IBM InfoSphereStreams [71] and Microsoft Workflow Manager [21] provide pub-sub mechanism among operators of the workflow graph but these kind pub-sub mechanisms are static in nature. All the publisher and subscribers need to be defined at compile time and cannot be changed once the execution starts. In the following sections, we will refer to such a system as *Static Pub-Sub Systems*.

In the remainder of this chapter, we explore how *RSTE* can be used to distribute data among multi-administrative domain environments. We focus on scenarios in which the statically linked producer/consumers and ad-hoc producer/consumers merge. Specifically, we explore a mechanism that can dynamically enforce policies restricting the information flow among multiple administrative domains and heterogeneous platforms in a controlled fashion using *RSTE*. We further explore how workflows spanning over multiple administrative domains can be scaled at runtime during execution without recompiling and interrupting the current workflow.

5.5.2 Proposed Solution

We propose a system that leverages the advantages of a dynamic publish subscribe (pub/sub) mechanism in static workflow graphs to dynamically share data among multiple parties. We propose the use of such dynamic pub/sub mechanism to make static workflow graphs extendible by using specialized *Hybrid Operators* [72]. These operators function

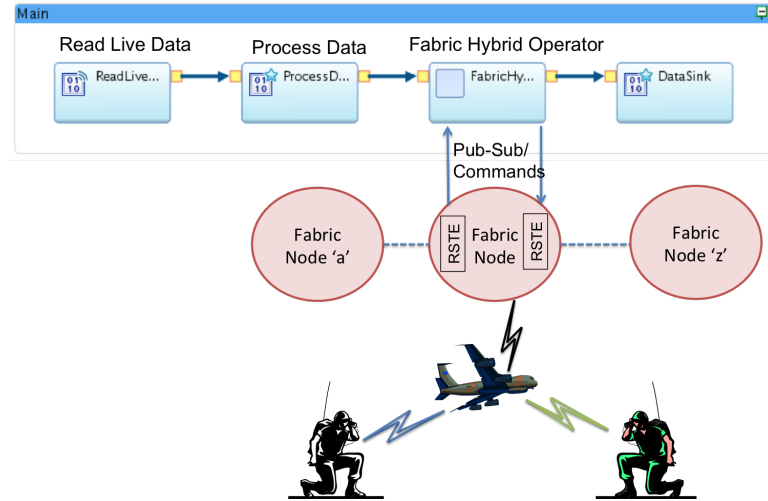


Figure 5.6: Hybrid operator in a workflow graph

like normal workflow operators but they have the capability to control the flow of data in the workflow graphs. These *Hybrid Operators* are simultaneously part of both the workflow execution graph and a dynamic multi-hop pub/sub bus. The *Hybrid Operators* are coupled with pub/sub-nodes which are part of the multi-hop pub/sub bus and we can use *RSTE* at these pub/sub nodes to enforce policies restricting information that is flowing out of these nodes. This is also the way in which the data that flows in a static workflow graph can be published over dynamic multi-hop pub/sub bus using the pub/sub-nodes. This way the publish subscribe mechanism enables flow control through pub/sub nodes where the *RSTE* policy enforcement mechanism is applied to them before sharing with other nodes on the bus and subscribers.

The *Hybrid Operators* also enable injection of new data sources into workflow graphs through new publishers which can become part of the pub/sub bus on an ad-hoc basis. Figure 5.6 shows the concept of an elastic workflow graph in which operators can run on fabric nodes that are hosted on heterogeneous distributed platforms. The challenge in such a hybrid system is the event rate mismatch. When the operators of a workflow span over heterogeneous devices and frameworks, the event rate of one device/framework can be very different from the other framework. Therefore, this data-rate mismatch has to be continuously adjusted at runtime to avoid overflowing frameworks with slow event processing rates.

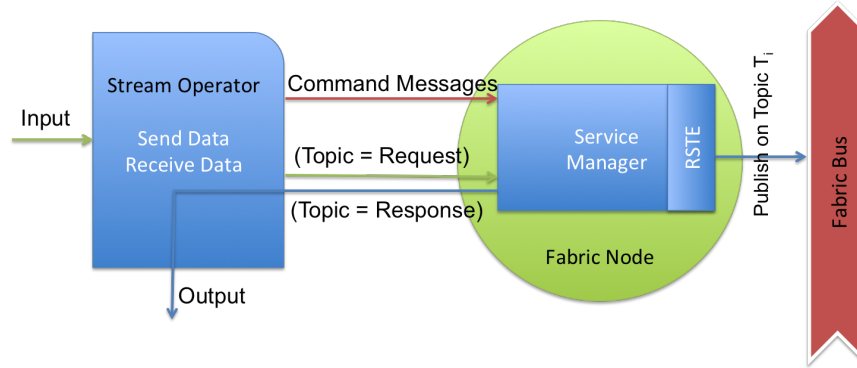


Figure 5.7: Hybrid operator and fabric node interaction

5.5.3 System Design and Implementation

As a proof of concept, we have developed the dynamic information flow control system using *IBM InfoSphere Streams* [71], the ITA Information Fabric [61] and our proposed *RSTE* language. *InfoSphere Streams* is capable of high speed data processing using distributed workflow graphs and the ITA Information Fabric is a dynamic multi-hop publish subscribe bus supporting sensor devices. Figure 5.6 shows the system level architecture in which the *Hybrid Operator* is coupled with a *Fabric Node*. Each ‘Stream Primitive Operator’ has a data and command path established with the *Fabric Node* through which the services and operations that take place on *Fabric Node* are configured. The data that is received by the ‘Stream Primitive Operator’ from predecessor operators is sent to the coupled *Fabric Node*. The *Fabric Node* applies policy enforcement to the data using *RSTE* and dynamically generates a new topic specific to the user/organization on which this data is published. This data can also be shared with other *Fabric Nodes* on the bus as needed, this way a workflow branch can also be created on the *Fabric Bus*.

5.6 Concluding Remarks

In this chapter, we present a novel mechanism for dynamic policy enforcement on services in sensor networks. We present *RSTE* language and we show that the network assets, user requirements and policy constraints can be represented as sets. We then show that we can apply set operations to these sets to enforce policy constraints and report any overly strict constraints to the user. We demonstrate that *RSTE* can be implemented using *RDBMS* and various *Relational Algebra* operations can be utilized to implement *RSTE*

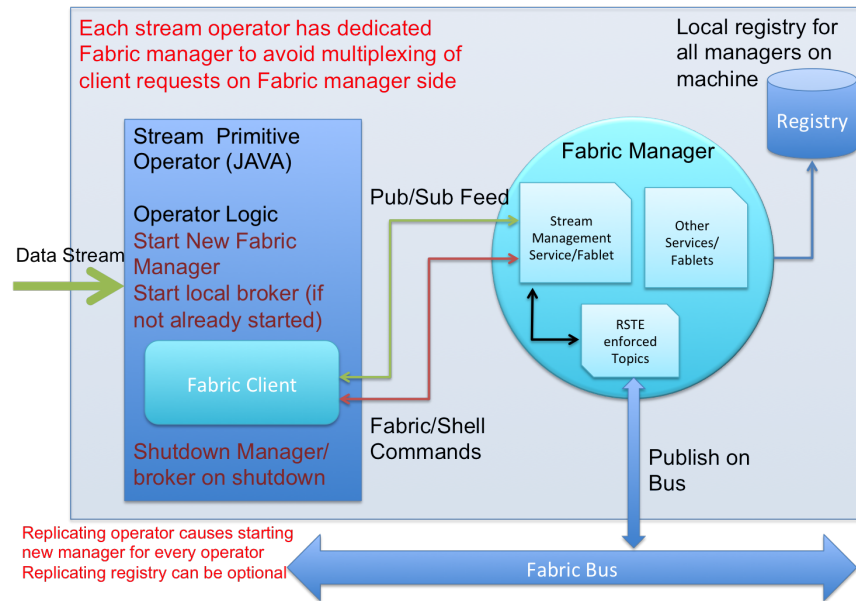


Figure 5.8: Detailed node level architecture

operations. We show that *RSTE* can enforce policies and perform backtracking.

CHAPTER 6

Conclusion

In this thesis, we have explored the area of network management for dynamically changing applications in general and focused on three topics within this area in particular. We have explored management of network routes in case of event-driven congestion and explored how to efficiently allot network routes to multi-priority traffic. We also studied the management of services hosted on network nodes and explored how to dynamically configured services that produce data that is highly relevant to end-user's area of interest. For the ease of managing shared resources, we investigated a policy enforcement mechanism that uses human understandable languages, such as Controlled English, for policy representation and for translating high-level policies to low-level system representation for application configuration.

6.1 Congestion Control in Event-Driven Scenarios

Congestion control in networks is a well-studied research area but congestion control in event-driven situations and routing in partially damaged network has still open challenges. We have addressed the issue of network routes management in event-driven scenarios that generate bursty traffic. We have devised a dynamic *Price Based Routing* protocol which not only reduces packet delivery delays and *Quality of Information (QoI)* losses for multi-priority traffic but is also capable of handling network traffic in situations where network infrastructure might be partially damaged or unavailable. Our mechanism efficiently manages multi-priority traffic by dynamically allocating routes to prioritized traffic without starving the lowest priority traffic. The mechanism allocates network routes to applications based on current state of the network i.e., predicted delays on network nodes, thus routes are assigned on-the-fly. Such assignment enables our mechanism to tackle the failed network nodes/routes and route traffic around the failed nodes/routes. Our mechanism is not only usable for congestion control in data communication network but is also applicable to other types of networks such as road networks and cyber-physical systems for building evacuations. We have conducted series of exper-

iments to show the performance of our protocol and prove that our mechanism reduces packet delivery delays as well as *QoI* losses of applications. We also devised a prioritized version of the *Random Oblivious Protocol* and compared it with our price based routing, details on this research topic are presented in *Chapter-3* of this dissertation.

Future Work:

For future, exploring more sophisticated utility functions for route pricing is a promising direction of research. Also, extending the current work beyond event-driven traffic and applying it to actual cyber-physical system would be interesting as new challenges from cyber-physical system may arise and help to improve the current protocol.

6.2 Service Management of Sensor Services

Under this topic, we have explored *Service Oriented Architecture (SOA)* for sensor services in WSNs and investigated how to dynamically configured complex services from services hosted at the edge of the network. Sensing nodes and services produces data but configuring services that produce data that is relevant to the user's interest is challenging issue. We have devised utility models to incorporate data relevancy in complex sensor service configuration to produces highly relevant services. We have proposed three models for incorporating relevancy in service configurations, the "Cost Based Model (CBM)", "Gain Based Model (GBM)" and "Return On Investment (ROI)" based model. The *GBM* and *CBM* are complementary, *CBM* uses both cost of a services as well as their relevancy to produce efficient service configuration whereas *GBM* focuses entirely on producing highly relevant services without trying to minimize overall cost of the service. The *ROI* based model enables user to utilize its budget on multiple services optimizing the return on investment of each individual service. Using these models, we incorporated data relevancy as a soft constraint on service selection during service composition. We have demonstrated a service oriented framework that provides such a relevancy aware service configuration. Using our system we conducted experiments to evaluate our utility models, further details on this topic are presented in *Chapter-4* of this thesis.

Future Work:

The current state of the work provides a solid foundation for future work. In future, more relevancy models can be explored to be incorporated in system, one such extension would be defining relevancy based on actual data flow rather than the coverage of the a service. A challenging task under this research topic would be to measure the data relevancy of the actual data produced by each individual service and then the data-relevancy of the composite service in terms of *Quality of Information*.

6.3 Resource sharing using Policies

Network services and resources in large coalition environment are established and shared by multiple organizations. Users in each organization in such coalitions have various roles and privileges; enabling access to the shared infrastructure based on user's privileges is often regularized by policies. We proposed a dynamic policy enforcement mechanism called *Restriction Set Theoretic Expressions (RSTE)* for sharing network resources. The novelty of such a mechanism lies in the fact that it uses human understandable format for policy representation and management and then converts such high-level policy representation to low-level system constraints. It does so by transforming all the data about the network resources, user requests and policies to set representation and then performs operation on sets to produces restricted set of resources. The restricted set contains resources available to a user's request based on policies and user's requirements. We have utilized relational database to prototype the *RSTE*, the use of *RDBMS* enables fast execution of queries and pre-calculated views which make the set operations of *RSTE* as fast as the speed of query execution on *RDBMS*. Our proposed *RSTE* language also enables backtracking mechanism which is very useful in tracking the over restricting policies for policy negotiation and relaxation. Further details and experimentation results on this topic can be found in *Chapter-5* of this thesis.

Future Work:

Currently *RSTE* supports policy enforcement on basic attributes such as user's role, properties, affiliation etc., along with basic constraints. In future, extensive support for multiple generic attributes along with full support of temporal constraints is a promising

direction. *RSTE* currently supports *Authorization Policies*, so adding *Obligation Policies* to the framework would be a good extension. The current prototype that is based on *RDBMS* can be extended to compute on-the-fly joins and views, create views based on set-operations and remove them dynamically after the use. Currently, *RSTE* has been tested with *IBM Controlled English* as a human-face of the system, but in future other Natural Language Processing platform can be supported. The backtracking in *RSTE* also has a promising future and extension can be done to optimize backtracking of policy enforcement which could be used for policy relaxation and negotiation on policy representation level.

REFERENCES

- [1] D. Ritchie, *The Computer Pioneers: The Making of the Modern Computer*. New York, NY, USA: Simon & Schuster, 1986.
- [2] D. S. De Couto, D. Aguayo, J. Bicket, and R. Morris, "A high-throughput path metric for multi-hop wireless routing," *Wireless Networks*, vol. 11, no. 4, pp. 419–434, Jul. 2005.
- [3] L. Chen, Z. Wang, B. Szymanski, J. Branch, D. Verma, R. Damarla, and J. Ibbotson, "Dynamic service execution in sensor networks," *The Comput. J.*, vol. 53, no. 5, pp. 513–527, Jun. 2010.
- [4] K. Shi, Y. Shu, O. Yang, and J. Luo, "Receiver-assisted congestion control to achieve high throughput in lossy wireless networks," *IEEE Trans. on Nucl. Sci.*, vol. 57, no. 2, pp. 491–496, Apr. 2010.
- [5] J. Postel. (1981) *RFC 793-Transmission Control Protocol*. [Online]. Available: <http://tools.ietf.org/html/rfc793> (Date Last Accessed, Oct., 28, 2014)
- [6] S. Chen and N. Yang, "Congestion avoidance based on lightweight buffer management in sensor networks," *IEEE Trans. on Parallel and Distributed Syst.*, vol. 17, no. 9, pp. 934–946, Sep. 2006.
- [7] C. Wang, B. Li, K. Sohraby, M. Daneshmand, and Y. Hu, "Upstream congestion control in wireless sensor networks through cross-layer optimization," *IEEE J. on Selected Areas in Commun.*, vol. 25, no. 4, pp. 786–795, May 2007.
- [8] X. Huang and Y. Fang, "End-to-end delay differentiation by prioritized multipath routing in wireless sensor networks," in *Military Commun. Conf. (MILCOM)*, 2005, pp. 1277–1283.
- [9] J. Lee and I. Jung, "Speedy routing recovery protocol for large failure tolerance in wireless sensor networks," *Sensors*, vol. 10, no. 4, pp. 3389–3410, Mar. 2010.
- [10] B. Fortz and M. Thorup, "Optimizing ospf/isis weights in a changing world," *IEEE J. on Selec. Areas in Commun.*, vol. 20, no. 4, pp. 756–767, May 2002.
- [11] S. Liu, X. Zhang, J. Yu, Y. Liu, and X. Chen, "A dynamic traffic distribution strategy for multipath routing," in *7th Int. Conf. on Inform., Commun. and Signal Process.*, 2009, pp. 1–5.
- [12] L. G. Valiant and G. J. Brebner, "Universal schemes for parallel communication," in *13th Annu. ACM Symp. on Theory of Computing*, 1981, pp. 263–277.

- [13] C. Busch, M. Magdon-Ismail, and J. Xi, "Oblivious routing on geometric networks," in *17th Annu. ACM Symp. on Parallelism in Algorithms and Architectures*, 2005, pp. 316–324.
- [14] G. Gorbil, A. Filippoupolitis, and E. Gelenbe, "Intelligent navigation syst. for building evacuation," in *Comput. and Inform. Sci. II: 26th Int. Symp. on Computer and Inform. Sci.*, 2011, pp. 339–349.
- [15] L. Zeng, B. Benatallah, A. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "Qos-aware middleware for web services composition," *IEEE Trans. on Software Eng.*, vol. 30, no. 5, pp. 311–327, May 2004.
- [16] S. Narayanan and S. A. McIlraith, "Simulation, verification and automated composition of web services," in *11th Int. Conf. on World Wide Web (WWW)*, 2002, pp. 77–88.
- [17] M. Alrifai and T. Risse, "Combining global optimization with local selection for efficient qos-aware service composition," in *18th Int. Conf. on World Wide Web (WWW)*, 2009, pp. 881–890.
- [18] N. Milanovic and M. Malek, "Current solutions for web service composition," *IEEE Internet Computing*, vol. 8, no. 6, pp. 51–59, Nov. 2004.
- [19] K.-T. Tran, N. Agoulmine, and Y. Iraqi, "Cost-effective complex service mapping in cloud infrastructures," in *Network Operations and Manage. Symp. (NOMS)*, 2012, pp. 1–8.
- [20] D. Jordan, J. Evdemon, A. Alves, A. Arkin, S. Askary, C. Barreto, B. Bloch, F. Curbera, M. Ford, Y. Goland *et al.* (2007, Apr.) *Web Services Business Process Execution Language*. [Online]. Available: <https://www.oasis-open.org/committees/download.php/12791> (Date Last Accessed, Oct., 28, 2014)
- [21] Microsoft. (2014, Oct.) *Windows Workflow Foundation*. [Online]. Available: <http://msdn.microsoft.com/en-us/vstudio/jj684582.aspx> (Date Last Accessed, Oct., 28, 2014)
- [22] A. Biem, E. Bouillet, H. Feng, A. Ranganathan, A. Riabov, O. Verscheure, H. Koutsopoulos, and C. Moran, "IBM infosphere streams for scalable, real-time, intelligent transportation services," in *2010 ACM SIGMOD Int. Conf. on Manage. of Data*, 2010, pp. 1093–1104.
- [23] F. Rosenberg, P. Leitner, A. Michlmayr, P. Celikovic, and S. Dustdar, "Towards composition as a service - a quality of service driven approach," in *IEEE 25th Int. Conf. on Data Eng. (ICDE)*, 2009, pp. 1733–1740.

- [24] D. Gay, P. Levis, R. Von Behren, M. Welsh, E. Brewer, and D. Culler, "The nesc language: A holistic approach to networked embedded systems," in *ACM SIGPLAN Conf. on Programming Language Design and Implementation*, 2003, pp. 1–11.
- [25] S. Vinoski, "Corba: Integrating diverse applications within distributed heterogeneous environments," *IEEE Commun. Mag.*, vol. 35, no. 2, pp. 46–55, Feb. 1997.
- [26] I. Jorstad and T. Do van, "A service-oriented architecture framework for mobile services," in *Advanced Ind. Conf. on Telecommun./Service Assurance with Partial and Intermittent Resources Conf./e-learning on Telecommun. Workshop*, 2005, pp. 65–70.
- [27] S. Geyik, B. Szymanski, and P. Zerfos, "Robust dynamic service composition in sensor networks," *IEEE Trans. on Services Computing (TSC)*, vol. 6, no. 4, pp. 560–572, Oct. 2013.
- [28] D. Chakraborty, A. Joshi, T. Finin, and Y. Yesha, "Service composition for mobile environments," *Mobile Networks and Appl. (MNA)*, vol. 10, no. 4, pp. 435–451, Aug. 2005.
- [29] G. Tychogiorgos and C. Bisdikian, "Selecting relevant sensor providers for meeting your quality information needs," in *12th IEEE Int. Conf. on Mobile Data Manage.*, vol. 1, 2011, pp. 200–205.
- [30] R. Mulligan and H. M. Ammari, "Coverage in wireless sensor networks: A survey," *Network Protocols & Algorithms*, vol. 2, no. 2, pp. 27–53, Apr. 2010.
- [31] I. Leontiadis, C. Efstratiou, C. Mascolo, and J. Crowcroft, "Senshare: Transforming sensor networks into multi-application sensing infrastructures," in *9th Eur. Conf. on Wireless Sensor Networks (EWSN)*, 2012, pp. 65–81.
- [32] D. Georgoulas and K. Blow, "In-motes: An intelligent agent based middleware for wireless sensor networks," in *5th WSEAS Int. Conf. on Appl. of Elect. Eng.*, 2006, pp. 225–231.
- [33] R. Smith, "Spotworld and the sun spot," in *6th Int. Conf. on Inform. Process. in Sensor Networks (IPSN)*, 2007, pp. 565–566.
- [34] P. Levis and D. Culler, "Maté: A tiny virtual machine for sensor networks," in *SIGOPS Operating Syst. (SIGOPS)*, vol. 36, no. 5, Oct. 2002, pp. 85–95.
- [35] TinyOS. (2014) *TinyOS Home Page*. [Online]. Available: <http://www.tinyos.net> (Date Last Accessed, Oct., 28, 2014)
- [36] T. Pham, G. H. Cirincione, D. Verma, and G. Pearson, "Intelligence, surveillance, and reconnaissance fusion for coalition operations," in *11th Int. Conf. on Inform. Fusion (FUSION)*, 2008, pp. 1–8.

- [37] A. Preece, T. Norman, G. de Mel, D. Pizzocaro, M. Sensoy, and T. Pham, "Agilely assigning sensing assets to mission tasks in a coalition context," *IEEE Intelligent Syst. (IS)*, vol. 28, no. 1, pp. 57–63, Jan. 2013.
- [38] G. Karjoth, "Access control with ibm tivoli access manager," *ACM Trans. on Inform. and Syst. Security*, vol. 6, no. 2, pp. 232–257, May 2003.
- [39] IBM. (2014, Oct.) *Policy Management Library*. [Online]. Available: <https://www.ibm.com/developerworks/community/groups/service/html/communityview?communityUuid=ed556565-1d91-4289-94ae-213df1340350> (Date Last Accessed, Oct., 28, 2014)
- [40] J. Lobo, "Cim simplified policy language (cim-spl)," *Specification DSP0231 v1. 0.0 a, Distributed Management Task Force (DMTF)*, vol. 10, no. 1, pp. 1–55, Jul. 2007.
- [41] J. Rubio-Loyola, J. Serrat, M. Charalambides, P. Flegkas, G. Pavlou, and A. L. Lafuente, "Using linear temporal model checking for goal-oriented policy refinement frameworks," in *6th IEEE Int. Workshop on Policies for Distributed Syst. and Networks (POLICY)*, 2005, pp. 181–190.
- [42] M. Sensoy, T. J. Norman, W. W. Vasconcelos, and K. Sycara, "Owl-polar: A framework for semantic policy representation and reasoning," *Web Semantics: Sci., Services and Agents on the World Wide Web*, vol. 12, no. 0, pp. 148–160, Apr. 2012.
- [43] S. A. Chun, V. Atluri, and N. R. Adam, "Using semantics for policy-based web service composition," *Distributed and Parallel Databases*, vol. 18, no. 1, pp. 37–64, Jul. 2005.
- [44] H. Prakken and M. Sergot, "Dyadic deontic logic and contrary-to-duty obligations," *Defeasible Deontic Logic*, vol. 263, pp. 223–262, 1997.
- [45] A. K. Bandara, E. C. Lupu, J. Moffett, and A. Russo, "A goal-based approach to policy refinement," in *5th IEEE Int. Workshop on Policies for Distributed Syst. and Networks (POLICY)*, 2004, pp. 229–239.
- [46] R. Dilmaghani, S. Geyik, K. Grueneberg, J. Lobo, S. Y. Shah, B. K. Szymanski, and P. Zerfos, "Policy-aware service composition in sensor networks," in *IEEE 9th Int. Conf. on Services Computing (SCC)*, 2012, pp. 186–193.
- [47] T. A. Ramrekha and C. Politis, "An adaptive qos routing solution for manet based multimedia communications in emergency cases," *Mobile Lightweight Wireless Syst. (MLWS)*, vol. 13, pp. 74–84, 2009.
- [48] L. Anderegg and S. Eidenbenz, "Ad hoc-vcg: a truthful and cost-efficient routing protocol for mobile ad hoc networks with selfish agents," in *9th Annu. Int. Conf. on Mobile Computing and Networking (MobiCom)*, 2003, pp. 245–259.

- [49] S. Y. Shah and B. K. Szymanski, "Price based routing for event driven prioritized traffic in wireless sensor networks," in *Network Sci. Workshop (NSW)*, 2013, pp. 1–8.
- [50] Y. S. Shah and B. K. Szymanski. (2012) *Dynamic Multipath Routing of Multi-Priority Traffic in Wireless Sensor Network*. [Online]. Available: <https://www.usukitacs.com/sites/default/files/multi-path.pdf> (Date Last Accessed, Oct., 28, 2014)
- [51] E. Gelenbe, G. Gorbil, and F. Wu, "Emergency cyber-physical-human systems," in *21st Int. Conf. on Comput. Commun. and Networks*, 2012, pp. 1–7.
- [52] D. Johnson and D. Maltz, "Dynamic source routing in ad hoc wireless networks," *Mobile Computing*, vol. 353, pp. 153–181, 1996.
- [53] K. Connolly. (2010) *Festivalgoers killed in stampede at Love Parade in Germany*. [Online]. Available: <http://www.guardian.co.uk/world/2010/jul/24/love-parade-festival-tunnel-stampede> (Date Last Accessed, Oct., 28, 2014)
- [54] Reuters. (2013) *At least 61 crushed to death in Ivory Coast stampede*. [Online]. Available: <http://www.reuters.com/article/2013/01/01/us-ivorycoast-stadium-idUSBRE90003Z20130101> (Date Last Accessed, Oct., 28, 2014)
- [55] Wikipedia. (2014) *List of human stampedes*. [Online]. Available: http://en.wikipedia.org/wiki/List_of_human_stampedes (Date Last Accessed, Oct., 28, 2014).
- [56] L. Soomaroo and V. Murray, "Disasters at mass gatherings: Lessons from history," *PLoS currents (PLOS)*, vol. 4, no. 1, Feb. 2012.
- [57] S. Y. Shah and B. Szymanski, "Transient traffic congestion control with traveling auctions," in *IEEE Int. Conf. on Pervasive Computing and Commun. Workshops (PERCOM Workshops)*, 2012, pp. 14–19.
- [58] I. S. Institute. (2011) *User Information*. [Online]. Available: http://nslam.isi.edu/nslam/index.php/User_Information (Date Last Accessed, Oct., 28, 2014)
- [59] T. Camp, J. Boleng, and V. Davies, "A survey of mobility models for ad-hoc network research," *Wireless Communications and Mobile Computing (WCM)*, vol. 2, no. 5, pp. 483–502, Aug. 2002.
- [60] S. Shah, B. Szymanski, P. Zerfos, and C. Gibson, "Towards relevancy aware service oriented system in wsns," *IEEE Trans. on Services Computing (TSC)*, vol. PP, no. 99, p. 113, Oct. 2014.

- [61] J. Wright, C. Gibson, F. Bergamaschi, K. Marcus, R. Pressley, G. Verma, and G. Whipps, "A dynamic infrastructure for interconnecting disparate isr/istar assets (the ita sensor fabric)," in *12th Int. Conf. on Inform. Fusion (FUSION)*, 2009, pp. 1393–1400.
- [62] G. Bent, P. Dantressangle, D. Vyvyan, A. Mowshowitz, and V. Mitsou. (2008) *A dynamic distributed federated database*. [Online]. Available: <https://www.usukitacs.com/papers> (Date Last Accessed, Oct., 28, 2014).
- [63] J. Ahrenholz, "Comparison of core network emulation platforms," in *Military Commun. Conf. (MILCOM)*, 2010, pp. 166–171.
- [64] NRL. (2014) *Extendable Mobile Ad-hoc Network Emulator (EMANE)*. [Online]. Available: <http://www.nrl.navy.mil/itd/ncs/products/eman> (Date Last Accessed, Oct., 28, 2014)
- [65] C. Sakama and K. Inoue, "Negotiation by abduction and relaxation," in *6th Int. Joint Conf. on Autonomous Agents and Multiagent Syst. (AAMAS)*, 2007, pp. 1–8.
- [66] P. D. Stone, P. Dantressangle, G. Bent, A. Mowshowitz, A. Toce, and B. Szymanski. (2012) *Query propagation behaviour in gaian database networks*. [Online]. Available: https://www.usukita.org/sites/default/files/P5_pstone_query_propagation_gaian.pdf (Date Last Accessed, Oct, 28, 2014)
- [67] S. Y. Shah and B. Szymanski, "Dynamic policy enforcement using restriction set theoretic expressions (rste)," in *IEEE Conf. on Military Commun. (MILCOM)*, 2014, pp. 198–203.
- [68] R. Boutaba and I. Aib, "Policy-based management: A historical perspective," *J. of Network and Syst. Manage.*, vol. 15, no. 4, pp. 447–480, Dec. 2007.
- [69] *IBM Controlled Natural Language Processing Enviornment*. [Online]. Available: <https://www.ibm.com/developerworks/community/groups/service/html/communityview?communityUuid=558d55b6-78b6-43e6-9c14-0792481e4532> (Date Last Accessed, Oct., 28, 2014)
- [70] S. Y. Shah, B. Szymanski, P. Zerfos, C. Bisdikian, C. Gibson, and D. Harries, "Autonomous configuration of spatially aware sensor services in service oriented wsns," in *Int. Conf. on Pervasive Computing and Commun. (PERCOM Demos)*, 2013, pp. 312–314.
- [71] IBM. (2013) *IBM InfoSphere Streams*. [Online]. Available: <http://www-03.ibm.com/software/products/us/en/infosphere-streams> (Date Last Accessed, Oct., 28, 2014)

- [72] S. Y. Shah, B. Szymanski, P. Zerfos, M. Srivatsa, C. Gibson, and D. Harries, “*Policy Enabled Real-time Information Flow Control Across Multiple Administrative Domains*,” 2013. [Online]. Available: <https://www.usukita.org/sites/default/files/ITA-ShortPaper-2013.pdf> (Date Last Accessed, Oct., 28, 2014)