RELIABILITY AND TIMELINESS ANALYSIS OF CONTENT-BASED PUBLISH/SUBSCRIBE SYSTEMS

BY

THADPONG PONGTHAWORNKAMOL

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2010

Urbana, Illinois

ABSTRACT

Content-based Publish/subscribe systems (CBPS) is a simple yet powerful communication paradigm. Its content-centric nature is suitable for a wide spectrum of today's content-centric applications such as stock market quote exchange, remote monitoring and surveillance, RSS news feed, and online gaming. As the trend shows that the amount of information along with its producers become astonishingly increasing everyday, a publish/subscribe system seems to be one of only a few viable choices that could govern the next-generation world of communication. However, the content-centric nature of a publish/subscribe system also poses difficulty in analyzing or assessing its performance. Moreover, the complexity increases when deploying a publish/subscribe system on top of best-effort, unreliable wide-area networks. Such uncertainty and complexity become a hindrance to apply content-based publish/subscribe systems to delay-sensitive applications that require reliable/timely event delivery and tight resource control such as soft real-time systems or cyber-physical systems. The need to solve such problem calls for a good *analytical model* that could capture both expressiveness and uncertainty nature of distributed CBPS systems yet predict the system behavior accurately.

This dissertation is, to the best our knowledge, the first attempt to analyze the reliability/timeliness performance of distributed content-based publish/subscribe systems under best-effort networks. It proposes a probabilistic, analytical framework of content-based publish/subscribe systems under different dynamism for the purpose of performance analysis. Specifically, given a publish/subscribe system configuration and dynamism parameters, it estimates event delivery probability and timeliness received by each subscriber in the publish/subscribe system. The dissertation also presents evaluation results of the proposed predictive model via simulations with both synthetic traces and real-world traces. The results yield prediction accuracy and effectiveness of the proposed framework. The proposed analytical framework can be used as a tool for performance assessment or as a building block for publish/subscribe system optimizations such as subscriber admission control, subscriber allocation, broker capacity planning, and broker network planning.

There are several factors, which are termed *dynamism* in this dissertation, that affect the performance of distributed content-based publish/subscribe systems. The proposed analytical framework first addresses each type of dynamism separately in order to avoid the modeling complexity and to study the effect of each type of dynamism individually. The proposed analytical model then relaxes each assumption and combine several types of dynamism altogether under one integrated framework.

There are three major types of dynamism considered in the analytical framework : *content dynamism*, *overlay dynamism*, and *mobility dynamism*. Content dynamism means the uncertainty in determining the amount of data from an arbitrary publisher to an arbitrary subscriber due to the publisher-subscriber decoupling nature of the content-based publish/subscribe systems. Overlay dynamism means the uncertainty from publish/subscribe internal broker network, including broker failures and link failures. Finally, mobility dynamism refers to the uncertainty from users' changes of location and content interest. We first propose a probabilistic analytical model for each type of dynamism separately before discussing the framework that integrates all separate analytical models together. We also present validation results for each dynamism-specific analytical model, which prove the accuracy and effectiveness of its corresponding analytical model.

This thesis makes contributions in the following areas. First, it proposes a detailed analytical model of content-based publish/subscribe systems from all possible aspects, providing a complete analysis in systematic manner. Second, it incorporates delay and reliability into one single analytical framework, which makes it suitable for delay-sensitive publish/subscribe applications. Third, it discusses and proposes some examples of possible publish/subscribe optimizations on top of such analytical model. Finally, it proves the applicability of the proposed analytical model via simulations with both synthetic and real-world traces.

To my family, teachers, and friends.

ACKNOWLEDGMENTS

First of all, I would like to thank my Ph.D. advisor, Professor Klara Nahrstedt for her dedication, advice, and support during all of my Ph.D. years. This dissertation and all my Ph.D. research would not have been done without her guidance and suggestions. Her limitless enthusiasm to pursue novel academic research and her kind understanding in her students have made her my role model of how a great researcher/professor can be. I will always be grateful to her.

At the same time, I would like to thank all of my Ph.D. committee members as follows. I would like to thank Professor Indranil Gupta for being both my M.S. thesis advisor and Ph.D. committee member. His insightful suggestions have helped improve my research in both M.S. and Ph.D. levels. I would like to thank Professor Matthew Caesar for his kind help and comments on my Ph.D. thesis. I also would like to express my gratitude to Doctor Ludmila Cherkasova for Hewlett-Packard Laboratories for all technical discussions and suggestions. Besides my Ph.D. committee members, I would not have reached to this point without all the coursework taught by various professors in the department of Computer Science, University of Illinois. I would like to thank all of them for their teachings and trainings. Apart from academic support, I would like to thank Doctor Guijun Wang along with his affiliation, Boeing Research & Technology, for all financial support during my Ph.D. years and useful research suggestions that have ultimately motivated this Ph.D. dissertation.

Having been a member in Multimedia Operating Systems and Networking (MONET) group for several years, I have come to know and received many supports from my past and present colleagues, which include Jingwen Jin, Won J. Jeon, Bin Yu, Jin Liang, Zhenyu Yang, Wenbo He, William Conner, Hoang Nguyen, Ying Huang, Long Vu, Muyuan Wang, Jigar Doshi, Wanmin Wu, Ravishankar Sathyam, Roger Cheng, Qiyan Wang, Raoul Rivas, Claudio Campeggi, Rahul Malik, Shameem Ahmed, Ahsan Arefin, Debessay Fesehaye, Zixia Huang, Rini Kaushik, Pooja Agarwal, Kurchi Hazra, Naveen Cherukuri, and Angali Sridhar. I would like to thank all of them for their kind feedback and energetic support that make my life in MONET group a very pleasant one.

I also would like to thank all of the administrative staffs in the department of Computer Science, University of Illinois at Urbana-Champaign for their kind assistance in various aspects. The staffs include, but are not limited to, Barb Cicone, Mary Beth Kelly, Shirley Finke, Anda Ohlsson, and Lynette Lubben.

Many thanks also go to my friends outside the department of Computer Science. I would like to thank everyone from Thai Student Association for making my Ph.D. life warm, enjoyable, and less stressful than it could be. Spending time with them has made me feel like home. I also wish them all best of luck in pursuing their current studies and future careers.

Last, but not least, I would like to thank my family for their unconditional love and support. I have been blessed to be born and raised up in a warm, happy family that gives me freedom to pursue my life in every aspect with full support and understanding. I would like to thank my parents for everything they have given in order to raise me up to the way I currently am. I would like to thank my brother and all the enjoyable moments we share together. I would like to thank my grandparents for their inspiration and dedication all along. My gratitude also goes to all relatives and their kind support. Finally, I would like to thank my special person for understanding me, being with me for better or for worse, and accepting me the way I am.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

$e \in \mathbb{E}$	An event in the set of all system events
$E \subseteq \mathbb{E}$	An arbitrary set of events
d_e	Event e's lifetime
$\mathbb{D}$	Set of all events' lifetime values
a_e	Event e's content
v_{ie}	Event e's i'th attribute
τ_e	Event e's topic (v_{1e})
$\mathbb{V}_i$	Value space of i^{th} attribute
$\mathbb{T}$	Topic space of all events and subscriptions ($\mathbb{V}_i$)
V	Content value space of all events
$s \in \mathbb{S}$	A subscriber in the set of all subscribers
$S \subseteq \mathbb{S}$	An arbitrary set of subscribers
$f_s \in V$	Subscriber s's content of interest filter
$F_s(E)$	Events in set E that matches s's interest
d_e^s	End-to-end delivery delay of event e to subscriber s
$p \in \mathbb{P}$	An arbitrary publisher in the set of all publishers
$P \subseteq \mathbb{P}$	An arbitrary set of publishers
t_p	The topic published by publisher p
$b \in \mathbb{B}$	An arbitrary broker in the set of all brokers
$B \subseteq \mathbb{B}$	An arbitrary set of brokers
M_b	Broker b's event processing time (distribution)
n	An arbitrary node in the system (subscriber/publisher/broker)

$\mathbb{N}$	The set of all nodes in the system ($\mathbb{S} \cup \mathbb{P} \cup \mathbb{B}$)
$l \in \mathbb{L}$	An arbitrary link in the set of all links
l_{xy}	The link connecting node x and node y
r_l	Link l's transmission probability
D_l	Link l's transmission delay (distribution)
$\mathbb{G} = (\mathbb{N}, \mathbb{L})$	Publish/subscribe overlay graph
r_s	Subscriber s's real-time reliability
r'_s	Subscriber s's estimated real-time reliability

CHAPTER 1

INTRODUCTION

1.1 Motivation

Over the past few years, publish/subscribe systems have recently become an emerging paradigm for large-scale information dissemination. The nature of publish/subscribe where the producers of the information (i.e., publishers) and the consumers of the information (i.e., subscribers) are interacting via intermediaries (i.e., brokers) allows both sides to be decoupled in space, time, and synchronization[1]. Such flexibility and scalability make publish/subscribe paradigm one of few viable choices for designing and building large-scale data dissemination systems.

So far, there have been significant efforts to design standards and build implementations of scalable and efficient distributed publish/subscribe systems[2, 3, 4, 5, 6, 7, 8, 9, 10, 11]. Based on commonly accepted taxonomy, publish/subscribe systems can be categorized into *topic-based* [5, 6, 7] and *content-based* publish/subscribe systems[2, 3, 4, 8, 9, 10]. In Topic-Based Publish/Subscribe (TBPS) systems, the events from publishers are delivered to subscribers that share the same single interest value, called *topic*. In Content-Based Publish/Subscribe (CBPS) systems, each event can contain multiple attributes. Any subscriber that is interested in a topic can further specify, at the attribute level, which portion of the topic events that it wants to receive. Hence, CBPS systems give more flexibility to the subscribers at the cost of increasing processing complexity at the brokers.

Beside the increasing complexity of CBPS systems compared to TBPS systems, another drawback of CBPS systems is the lack of predictability. In CBPS systems, since each subscriber has flexibility to choose the information it wants at fine-grained attribute level, it is non-trivial to determine event flow from each publisher to each subscriber. Hence, it is also non-trivial to analyze quality-of-service performance and correctness of a content-based publish/subscribe system when compared to its topic-based counterpart. For example, it is non-trivial to check how much resource is needed to service each subscriber properly, or to verify if the system's current state is stable. Moreover, deploying distributed publish/subscribe systems over unreliable networks further decreases system determinism and predictability. Such uncertainty and complexity become a hindrance to apply content-based publish/subscribe systems to delay-sensitive applications that require quality of service and tight resource control such as soft real-time systems or cyber-physical systems. The need to solve such problems calls for a strong *analytical model* that could capture both expressiveness and uncertainty nature of distributed CBPS systems yet predict the system behavior accurately.

However, while it is infeasible to calculate the *exact* resource consumption and quality of service each subscriber receives in content-based publish/subscribe systems, it is still possible to do so in a probabilistic manner when the *partial* information of each component in the system is given. The term *partial* information refers to the trend or behavior patterns of each component, such as *underlying networks* (e.g., how likely that the event transmission delay over a link will be within 5 seconds), *hardware capabilities* (e.g., the average broker event processing time), *user mobility pattern* (e.g., how likely a subscriber will move from one point to another point), and *publishing pattern* (e.g., how likely a publisher will publish a value or how likely that a publisher will publish the next message within a specific time). Such pattern information can be either explicitly given by or implicitly observed statistically from each component[12], thus making it possible to model and predict behavior of the publish/subscribe system. The performance prediction can then be used as a building block for autonomic QoS control such as *subscriber admission control*, *broker capacity planning*, *overload management*, and *resource adaptation*. To the best of our knowledge, there is no work to date that discusses the usage of such fuzzy and partial information to predict the reliability/timeliness behavior of distributed content-based publish/subscribe systems over best-effort networks. In other words, we make the following thesis statement.

> We propose an analytical model that allows subscriber-oriented, reliability/timeliness prediction and optimization in distributed content-based publish/subscribe systems with high probability.

In this thesis, we categorize the information of publish/subscribe components into three groups. We use the term *dynamism* to describe such information, as they are dynamic factors that affect the performance of publish/subscribe systems. Each type of dynamism is addressed by each Chapter of this thesis. The first form of dynamism is called *content dynamism*, which is the uncertainty in determining the amount of traffic that flows through a path between each publisher-subscriber pair. Such information is essential in determining the load of each component in publish/subscribe systems. The second form of dynamism is called *overlay dynamism*, which is the change in broker overlay networks caused by broker failures and link failures. The overlay dynamism can cause service outage or disruption from each subscriber's point of view. The third form of dynamism is called *mobility dynamism*, which is the change of subscribers' locations caused by subscribers' movements. Given the information of such dynamism, the proposed analytical model can predict the performance of the publish/subscribe systems under the dynamic factors.

1.2 State of the Art

This sections presents the state of the art of quality of service in publish/subscribe systems. We first present the literature survey of generic distributed publish/subscribe systems in Section 1.2.1. Then, the related works in the qualify of service aspect is presented in Section 1.2.2. Finally, Section 1.2.3 discusses existing works in formal model, analysis and verification of publish/subscribe systems. Note that each category of the works presented in this section is not mutually exclusive, as some works may fall into multiple categories. In each subsequent chapter, we also give related works that are specific to that chapter.

1.2.1 Distributed Publish/Subscribe Systems

Over the past decade, publish/subscribe systems have emerged as a multi-source, multi-sink communication paradigm. The main concept of publish/subscribe paradigm is that senders and receivers of information are connected loosely based on the *content* of the information. Specifically, in a publish/subscribe system, a *publisher* can produce its information (i.e. *messages* or *events*) without specifying the set of *subscribers*. Instead, each subscriber specifies the content of information it is interested to receive. All messages produced from publishers containing content that matches a subscriber's interest are then delivered to that subscriber via a network of publish/subscribe intermediaries called *brokers*. Since the information flows based on the content of the information, publishers and subscribers are decoupled in space, time, and synchronization[1]. Such transparency allows the system to scale and adapt well under dynamic environments.

Generally, publish/subscribe systems can be categorized into two types based on the granularity of the content matching and routing. The first type is called *topic-based* publish/subscribe, while the second type is called *content-based* publish/subscribe. The other types of categorization and complete taxonomy of publish/subscribe systems can be found in several survey papers[11, 13, 1, 14].

1.2.1.1 Topic-based Publish/Subscribe

Topic-based publish/subscribe systems are the first set of incarnation and the simplest form of publish/subscribe systems where publishers and subscribers are linked via a single keyword called *topic*. That is, each subscriber specifies a single topic keyword of its interest. Likewise, each publisher associates a single topic keyword to each of its published message. Hence, a subscriber will receive all messages associated with a topic keyword that matches the subscriber's topic keyword. Hence, topic keyword in topic-based publish/subscribe systems is analogous to session identifier in multicast systems. Some examples of topic-based publish/subscribe systems are SCRIBE[5], Herald[6], and Bayeux[7].

The advantages of topic-based publish/subscribe systems are simplicity and efficiency in message matching and routing at each broker in the network. The matching process in topic-based publish/subscribe systems only involves exact keyword matching, incurring small overhead at each broker. However, the main disadvantage of the topic-based publish/subscribe systems is the lack of expressiveness, since each subscriber will always receive *all* messages with the topic of its interest even when that subscriber is interested only in a *subset* of messages from that topic. The limitation of expressiveness than motivates the second-generation publish/subscribe systems called *content-based* publish/subscribe systems.

1.2.1.2 Content-based Publish/Subscribe

To achieve higher expressiveness in publish/subscribe systems, content-based publish/subscribe systems are introduced. The main difference between topic-based publish/subscribe systems and content-based publish/subscribe systems are the level of granularity in message structure and topic matching. In contrast to topic-based publish/subscribe systems where subscribers specify their interests in the form of topic keywords, subscribers in content-based publish/subscribe systems can specify, in addition to the topic keyword, conditions of attributes of messages that match the subscribers' interests. Hence, not all messages with the matching topic will be delivered to the subscriber. Instead, only messages with both matching topic and matching attributes will be delivered to the subscriber. Content-based publish/subscribe systems thus assume the attribute structure of messages of each topic to be known by both publishers and subscribers. Some examples of content-based publish/subscribe systems include Siena[2], Gryphon[3], JEDI[4], Hermes[8], CORBA standard[9], and DDS standard[10].

There are several advantages of content-based publish/subscribe systems over topic-based publish/subscribe systems such as higher level of expressiveness for subscribers in selecting messages of their interests. Another advantage is less bandwidth consumption as the unwanted portion of messages will be filtered out before reaching subscribers. However, such advantages come at the cost of additional processing and storage overhead at the brokers, as now the message matching involves more complex attribute matching. Another disadvantage of content-based publish/subscribe systems is less predictability in resource consumption, as it is harder to determine the amount of traffic between a pair of publisher and subscriber, making it harder to predict the behavior of the system especially in quality of service aspect.

1.2.2 Quality of Service in Publish/Subscribe Systems

Quality of service in publish/subscribe systems has been discussed in several survey works[15, 16, 17]. We categorize works in this area in two categories based on application requirements and QoS support from underlying hardware and networks. The first category represents hard real-time publish/subscribe systems and the second category represents soft real-time publish/subscribe systems.

1.2.2.1 Hard Real-time Publish/Subscribe

The first category of QoS in publish/subscribe systems is hard real-time publish/subscribe systems[18, 19, 20]. Works in this area discuss the problem of designing publish/subscribe systems that support hard real-time applications with critical deadlines. However, these works require QoS support from underlying hardware and networks (i.e., hard real-time processor scheduling and real-time networks). This thesis, however, focuses on another scenario where an application deadline is not critical.

1.2.2.2 Soft Real-time Publish/Subscribe

In contrast to hard real-time publish/subscribe systems, soft real-time publish/subscribe systems aim to support non-critical applications that allow QoS violation to some extent[21, 22, 23, 24, 25]. Soft real-time publish/subscribe systems usually do not require real-time support from underlying components, but instead implement quality of service mechanisms at broker level. Some works in this category focus on event routing decision[22, 23] while some works achieve quality of service by event scheduling[21]. Recently, there are several works that address quality of service when clients are mobile[24, 25].

While this thesis proposes an analytical model for best-effort content-based publish/subscribe system, it can be categorized soft real-time publish/subscribe category as the proposed model targets delay-sensitive applications. However, unlike existing works in soft real-time publish/subscribe systems, this thesis proposes the generic analytical model in order to quantitatively predict QoS behaviors of standard, existing publish/subscribe systems.

1.2.3 Formal Modeling and Analysis of Distributed Publish/Subscribe Systems

There have been significant efforts to model and analyze publish/subscribe systems along with their correctness properties and performance aspects. In his dissertation, Muhl[26] proposed a generic content-based publish/subscribe frameworks and a class of subscription/publication matching/routing algorithms (such as flooding, subscription covering, subscription merging) along with proof of correctness and performance analysis. Jaeger[27] also proposed the analytical model for self-stabilizing publish/subscribe under transient failures. Baldoni et al[28] also proposed correctness proof of publish/subscribe systems when subscription propagation delay is not negligible. However, all works assume do not address timeliness aspect in event delivery. He et al[29] proposed a publish/subscribe model checker based on probabilistic timed automata. However, the computational overhead associated with the automata due to state explosion may limit the usage of such approach to only small-sized problems. Baresi et al[30] addressed the state explosion problem in by adding publish/subscribe APIs into model checkers. Recently, Schröter et al have proposed the stochastic analytical model for content-based publish/subscribe that addresses event queuing/processing and link transmission delay[31, 32], which bears some similarity with a part of our thesis[33, 34]. However, our work differ from their work in analysis granularity and queuing model.

1.3 Thesis Scope and Contribution

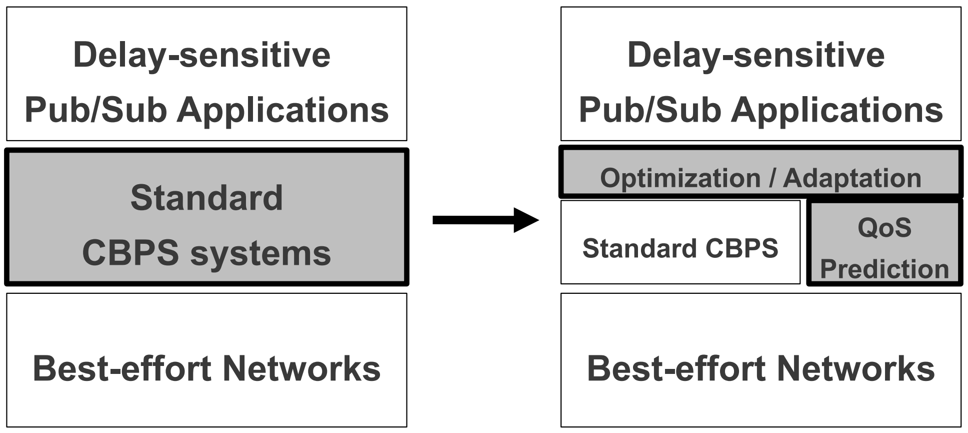

Delay-sensitive Pub/Sub Applications
Standard CBPS systems
Best-effort Networks
Delay-sensitive Pub/Sub Applications
Optimization / Adaptation
Standard CBPS
QoS Prediction
Best-effort Networks

Figure 1.1: Diagram showing the scope of this thesis in the communication protocol stack. The right figure shows the break-down of each component addressed in this thesis.

1.3.1 Thesis Contributions

To the best of our knowledge, this thesis has the following contributions towards research areas in content-based publish/subscribe systems.

Reliability/timeliness analytical model for content-based publish/subscribe systems : We propose an analytical model framework to estimate reliability and timeliness quality of service for each subscriber in content-based publish/subscribe systems. The framework abstracts any generic content-based publish/subscribe protocols and their underlying best-effort networks. To the best of our knowledge, this dissertation is the first work to analyze reliability and timeliness aspects of publish/subscribe systems under various types of uncertainty, which represent many Internet-scale, time-sensitive, content-based applications[35, 36, 37, 38, 39]. The framework accounts for different types of uncertainty, termed *dynamism*, that affects the performance of the publish/subscribe systems. For each type of dynamism, we propose an analytical model that specifically abstracts such type of dynamism and estimate the performance of the publish/subscribe systems. We also present the integrated model framework that combines all types of dynamism altogether.

Validation of the proposed model via synthetic/trace-based simulations : For each type of dynamism-specific analytical model, we present the validation of such model via simulations using realistic configuration parameters. All the results have less then 10% error, with average error less than 5%, yielding the accuracy of the proposed predictive models.

Study of the impact of different types of dynamism to publish/subscribe systems : Using the proposed analytical model, we study the effect from each type of dynamism to the reliability/timeliness quality of service each subscriber receives. The study gives several insights for the purpose of publish/subscribe system design and deployment.

Performance comparison between different Publish/subscribe protocols and configurations : Using the proposed analytical model, we do the performance comparison between different publish/subscribe protocols and configurations The results show that the performance of each protocol also depends on the external settings such as subscriber mobility and broker reliability.

Publish/subscribe optimization techniques based on the proposed analytical model : Based on the proposed analytical model, we suggest several possible performance optimization techniques that can be used to improve reliability/timeliness quality of service in content-based publish/subscribe systems.

1.3.2 Thesis Scope

Publish/subscribe is a powerful and broad communication paradigm that covers a huge variety of protocols and implementations. While this dissertation proposes the analytical model for publish/subscribe systems that is as generic as possible, it does not claim a universal analytical model that covers all publish/subscribe systems and scenarios. This section presents the scope of content-based publish/subscribe systems along with the types of applications that are addressed by this thesis. More details about the exact publish/subscribe model and assumptions assumed by this thesis can be found in Chapter 2.

Delay-sensitive, event-based applications : From the application perspective, this thesis focuses on delay-sensitive event-based applications such as real-time traffic report or remote sensing.

Such applications are delay-intensive but not bandwidth-intensive. Each event or message is self-contained and has its validity period within which it needs to be delivered to the target subscribers. Also, the target applications allow some event losses without catastrophic consequences. This thesis does not consider hard real-time critical applications, which have been proved impossible to achieve perfect reliability in asynchronous networks[40].

Best-effort, wide-area overlay networks : This thesis focuses on distributed, content-based publish/subscribe systems that are deployed over best-effort, wide-area networks such as Internet. Over the past decade, the quality of public, wide-area networks has been significantly improved. It has become common for soft real-time applications to be deployed in such asynchronous networks while getting acceptable performance.

Analysis of *existing* publish/subscribe systems : This thesis does not propose any brand-new publish/subscribe protocol. This thesis presents a complete analytical model for existing publish/subscribe systems and protocols. Given the system settings, the proposed analytical model estimates the performance of the corresponding publish/subscribe systems in terms of reliability and timeliness in event delivery. Figure 1.1 depicts the scope of this thesis in the communication protocol stack.

Client-server model : This thesis assumes clear distinction between clients (publishers and subscribers) and servers (brokers). Each client is connected only to any single broker at one time and there is no ad hoc communication directly among clients. This thesis does not propose the analytical model for large-scale peer-to-peer publish subscribe systems[5, 41], although some parts of the framework may be applicable to the peer-to-peer approaches.

Single broker administrative domain : This thesis assumes that all brokers are managed under the same policy and level of security, which is the case for most real-time distributed event-based applications. This thesis does not address any issues in security among brokers nor between brokers and clients.

1.4 Thesis Outline

This dissertation is organized as follows. Chapter 2 discusses the detailed and complete model of soft real-time, distributed content-based publish/subscribe systems along with all assumptions used in the proposed analytical model. Chapter 3 then proposes the reliability/timeliness analytical model to address content dynamism under the assumption of static, reliable broker networks. Chapter 4 then presents the reliability/timeliness analytical model for mobility dynamism caused by users under the same assumption. Chapter 5 then relaxes the static, reliable broker assumption and presents the reliability/timeliness analytical model for non-reliable broker networks. Chapter 6 then discusses the integration of the proposed three separate models. Finally, Chapter 7 concludes the thesis and suggests possible future research directions.

CHAPTER 2

PUBLISH/SUBSCRIBE SYSTEM MODEL AND ASSUMPTIONS

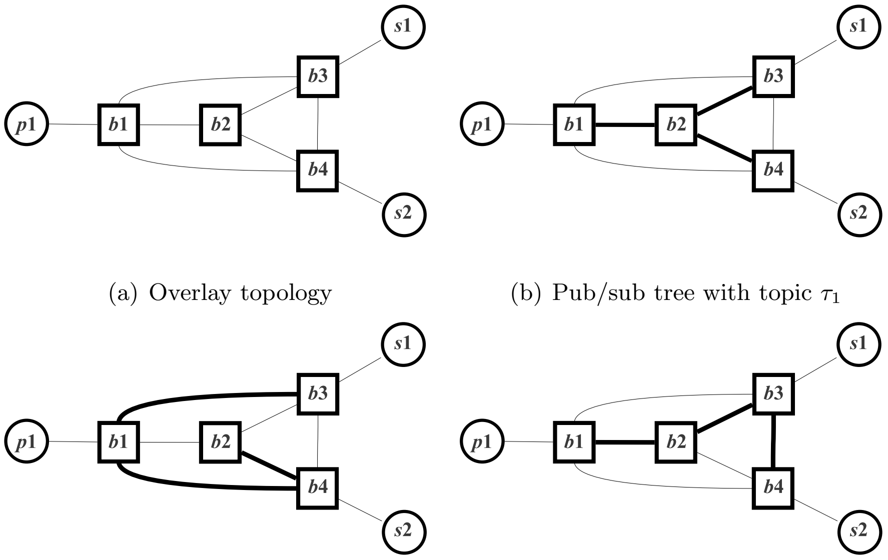
This chapter presents the complete details of the distributed content-based publish/subscribe model, the formal definition of reliability and timeliness metrics, along with all the assumptions used in the proposed analytical model.

2.1 Content-based Publish/Subscribe Architecture

In this work, we assume acyclic publish/subscribe broker tree networks connecting publishers and subscribers. This acyclic broker tree model is commonly adopted by existing publish/subscribe systems[2, 3, 4, 8, 42]. Each broker is connected to at least one neighbor to form a tree network (i.e. there is only one path between each pair of broker). Each subscriber/publisher is connected to only one of the brokers in the system. The broker that is connected to a subscriber/publisher is called the *home broker* with respect to that subscriber/publisher. Each publisher publishes *events* or *messages* to its home broker, which then forwards the events to other brokers until the events are propagated to the designated subscribers.

Note that in general, it is possible for a broker to be connected to several neighboring brokers, resulting in a generic, non-tree broker overlay. In such case, we assume the use of a *per-topic* tree, in which brokers form a spanning tree to disseminate events per each topic. Figure 2.1 depicts an example of how a broker constructs three per-topic trees corresponding to three different topics. There are one publisher ($p1$), two subscribers ($s1$ and $s2$), and four brokers ($b1$, $b2$, $b3$, $b4$) in the overlay topology (Figure 2.1(a)). However, the brokers use only a subset of links to form a spanning tree among themselves to disseminate events of each topic. The spanning tree construction is based on the overlay topology and the corresponding topic. We assume each broker has the global knowledge of the overlay broker network topology. Hence, given the same topic identifier, each broker can construct the same tree by using the topic identifier as a random seed to construct a spanning tree. For example, the corresponding tree for topic τ_1 consists of links l_{b1b2}, l_{b2b3}, and

l_{b2b4} as depicted in Figure 2.1(b). Note that each broker can be associated with many topics and thus will form many broker spanning trees over the same non-tree broker overlay. The use of a per-topic tree helps spread the load across the broker network as no single broker will become the bottleneck. From now on, we omit the per-topic keyword and use the term *tree* to represent a per-topic tree with the focus on only one topic.



(a) Overlay topology

(b) Pub/sub tree with topic τ_1

(c) Pub/sub tree with topic τ_2

(d) Pub/sub tree with topic τ_3

Figure 2.1: Example of subscription propagation and event routing in a publish/subscribe system

2.2 Content-based Publish/Subscribe Entities

Here, we present the basic model of each component in distributed publish/subscribe systems. We also present each component's formal notation, which will be used in the analytical model throughout this thesis. The summary of basic notations can be found in the list of abbreviations section.

2.2.1 Real-time Events

Each published event has one or more *attributes* with the associated *value.* Each event also has its *lifetime* value, which is the duration between the time the event was published and the time the event is expired. An event is said to be delivered to a subscriber *on time* if the end-to-end delivery delay is *less* than its lifetime[1].

We use the notation e to denote an event in the system. We use the notation E to represent an arbitrary set of events and the notation $\mathbb{E}$ to represent the set of all existing events in the system. An event $e \in \mathbb{E}$ is defined as (id_e, a_e, d_e), which represent event's identifier, content, and lifetime duration, respectively. The content of an event e, denoted by a_e, is defined as a tuple

$$a_e = (v_{1e}, v_{2e}, .., v_{ke})$$

, where v_{ie} is the value of the i^{th} attribute of the event e. For simplicity of the analysis, we assume that the event topic (τ_e) is always the first attribute (v_{1e}) and the rest $k-1$ attributes are *the union of all per-topic attributes* in the system in an arbitrary, but globally consistent order. Hence, an event of any topic in the system can be expressed with such k attributes by setting irrelevant attributes from other topics to null value.

Let $\mathbb{V}_i$ be the value space of the i^{th} attribute of any event $(\forall e \in \mathbb{E} : v_{ie} \in \mathbb{V}_i)$. Let $\mathbb{T}$ be the set of all topics in the system (i.e., $\mathbb{T} = \mathbb{V}_1$). Let $\mathbb{D}$ be the set of all possible lifetime duration values of events in the system. Note that $\mathbb{V}_i$ and $\mathbb{D}$ can be either discrete or continuous. Without loss of generality in the analysis, we assume $\mathbb{V}_i$ to be discrete in this work. However, the proof also applies to the continuous case.

We define

$$\mathbb{V} = \mathbb{T} \times \mathbb{V}_2 \times .. \times \mathbb{V}_k$$

as the event content space of the system (i.e., $\forall e \in \mathbb{E} : a_e \in \mathbb{V}$).

2.2.2 Subscribers

A subscriber is an information consumer that would like to receive the events of certain attributes published from the systems. In topic-based publish/subscribe systems, a subscriber specifies its topic of interest, which is a single identifier such as a number or a string keyword. Any event with the specified topic is then forwarded to that subscriber. In content-based publish/subscribe systems, however, a subscriber can specify a *content filter* to the system to inform the system of

[1]We assume that there exists a mechanism such as clock synchronization among brokers/publishers/subscribers that allows a node to verify whether the event from a publisher is expired or not.

certain event attribute patterns it would like to receive. Hence, content-based publish/subscribe systems provide more flexibility for subscribers in terms of event selectivity.

We use the notation s to describe an arbitrary subscriber in the system and the notation S to represent an arbitrary set of subscribers. We also use the notation $\mathbb{S}$ to denote the set of all possible subscribers in the system. A subscriber $s \in \mathbb{S}$ is defined as a tuple

$$s = (id_s, f_s)$$

where id_s is the subscriber's identifier, $f_s \subseteq \mathbb{V}$ is the content filter defining the content of interest for s. That is, f_s is the set of all event content that matches s's interest. We define a filter set $F_s(E)$ of an event set $E \subseteq \mathbb{E}$ with respect to subscriber s as

$$F_s(E) = \{e \in E : a_e \in f_s\}$$

We assume that a subscription's content filter is static in the sense that it does not change over time once it is submitted. If a subscriber wish to change its subscription, it needs to unsubscribe its previous content filter and subscribe with its new content filter. Modeling dynamic subscriptions such as context-aware subscriptions[43] and parametric subscriptions[44] are left as future work.

2.2.3 Publishers

A publisher is an information producer that publishes events (i.e., generates information) to the publish/subscribe systems. In many scenarios, a publisher can be a computer or a smart device with sensing capability. We assume each publisher is independent from each other in terms of the content of its published events and its publishing frequency.

We use the notation p to denote an arbitrary publisher in the system and the notation P to denote an arbitrary set of publishers. A notation $\mathbb{P}$ is used to represent the set of all publishers in the system. In its most basic form, a publisher $p \in \mathbb{P}$ is defined as a tuple

$$p = (id_p, t_p)$$

where id_s is the publisher's identifier and $t_p \in \mathbb{T}$ is the topic the publisher p publishes. Note that this publisher model represents a basic publisher. Note that in later chapters such as Chapter

3, we will incorporate additional variables into the publisher model in order to reflect additional information and assumptions used for the purpose of performance analysis in each chapter.

2.2.4 Brokers

A broker is a service component in publish/subscribe systems which helps forward events from publishers to designated subscribers. Each broker is connected to at least one neighboring broker to form the distributed publish/subscribe network. A broker can be called a content-based router, which routes the event based on the event content.

In our content-based publish/subscribe model, a broker is represented by a single server and an event queue. Upon receiving a new event, the broker stores the incoming event in its event queue on a first-come-first-served basis. When the broker is ready, it fetches the event from the head of the queue and performs the matching and forwarding process, during which the broker is considered busy. Once the event is forwarded, the broker then repeats the process until the queue is empty. The details of the broker event matching process will be presented in Section 2.3.

We use the notation b to denote an arbitrary broker in the system, the notation B to denote an arbitrary set of brokers, and the notation $\mathbb{B}$ to denote the set of all brokers in the system. A broker $b \in \mathbb{B}$ in the system is defined as a tuple

$$b = (id_b, M_b(d))$$

where id_b is the broker's identifier, and M_b is the probability mass/density function of broker b's event processing (matching and routing) delay[2]. For example, $M_b(100\text{ms}) = 0.2$ means with 20% probability, the delay the broker b will take to retrieve an event from its queue and route the event through the appropriate links is 100 milliseconds. We assume that the average event matching time $\mathrm{E}[M_b]$ is a function of the number of subscriptions stored in broker b's routing table. $M_b(d)$ can be obtained from periodic performance profiling at the broker node b. As of now, we assume each broker does not fail. This assumption will be relaxed in Chapter 5, in which the broker failure and availability parameters are incorporated into the broker model.

We use the term *node*, denoted by n, to refer to either a subscriber, a publisher, or a broker in the system. Hence, we use notation $\mathbb{N}$ to represent all nodes in the system (i.e., $\mathbb{N} = \mathbb{P} \cup \mathbb{S} \cup \mathbb{B}$).

[2] $M_b(d)$ is either a probability mass function in case of discrete distribution or a probability density function in case of continuous distribution.

2.2.5 Links

Each broker is linked via asynchronous, non real-time, wired communication link. Inter-broker links can fail with some probability. The broker/publisher and broker/subscriber links can be either wired or wireless links. As mentioned in Section 2.1, we assume a per-topic broker tree, which means for each topic, there is only one communication path between each pair of broker.

We use the notation l to represent an arbitrary link, L to represent an arbitrary set of links, and $\mathbb{L}$ to represent the set of all links in the system (i.e., $L \subseteq \mathbb{L}$). The notation $l_{xy} \in \mathbb{L}$ is also used to represent the link connecting node x and node y. We assume each link to be directional (i.e., $l_{xy} = l_{yx})$). A link $l \in \mathbb{L}$ can connect either a publisher to a broker (i.e., *publisher-broker* link), a broker to another broker (i.e., *broker-broker* link), or a broker to a subscriber (i.e., *broker-subscriber* link).

Each link $l \in \mathbb{L}$ is defined as a tuple $(r_l, D_l(d))$, where r_l is the link l's transmission success probability and $D_l(d)$ is link l's transmission delay distribution. Both parameters which can be estimated periodically via active or passive probing. We believe that the two parameters r_l and $D_l(d)$ are sufficient to capture the characteristic of most point-to-point overlay links and transmission protocols. For example, a UDP link will have lower transmission success probability r_l and lower delay distribution $D_l(d)$ while a TCP link will have higher transmission success probability r_l, but also higher delay distribution $D_l(d)$ due to multiple retransmission. Note that $(1 - r_l)$ represents *transient* failure loss rate of the link. Long-term link failure will be discussed in Chapter 5.

Finally, we define a *publish/subscribe graph* $\mathbb{G} = (\mathbb{N}, \mathbb{L})$ to represent the complete view of the system that includes all nodes and links. Unless specified otherwise, we assume the publish/subscribe graph $\mathbb{G}$ to be static. Such assumption will be relaxed when user mobility and node/link failure are considered in Chapter 4 and Chapter 5 respectively.

2.3 Content-based Publish/Subscribe Operations

The subscriber/publisher joining process and event/subscription matching process in the publish/subscribe systems are shown in Figure 2.2 as follows.

2.3.1 Subscription

When a new subscriber joins the system, it sends its subscription to one of the brokers (Figure 2.2(b)). A subscription contains *predicate filter* specifying the event attribute content that the subscriber wants to receive. In Figure 2.2(b), each predicate filter is in conjunctive form consisting of per-attribute min-max clauses. However, our analytical model supports all possible forms of filter as long as the filter can be expressed a a subset of the attribute content space.

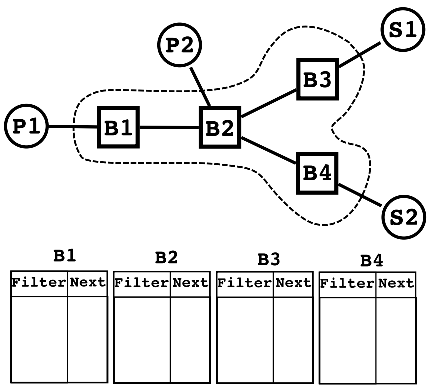
Upon receiving the subscription from the subscriber, the broker stores the subscription into its routing table and propagates the new subscription to its adjacent brokers, which in turn repeat the process until all brokers receive the subscription (Figure 2.2(b) and 2.2(c)). When storing a new subscription into its routing table, each broker also stores the link information to the broker which it receives the subscription from.

2.3.2 Event Matching and Forwarding

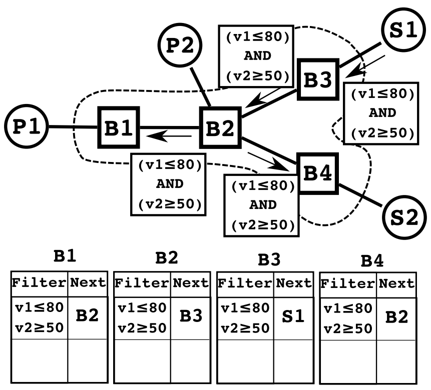
When a broker receives a newly-published event (Figure 2.2(d)), it checks the event with each subscription stored in its routing table. For each matching subscription, the broker forwards the event to the link which it receives that subscription from. Note that an event is forwarded *once* per link even though there are multiple matching subscriptions from that link. The process then continues, and the event is propagated hop-by-hop in the reverse direction of the subscription until it reaches the designated subscribers. The mentioned publish/subscribe model is simple yet generic enough to represent a variety of existing publish/subscribe system works[2, 3, 4, 8]. Further optimization techniques on top of this basic model such as subscription covering or subscription merging[45] are beyond the scope of this paper and considered as future directions.

2.3.3 Unsubscription

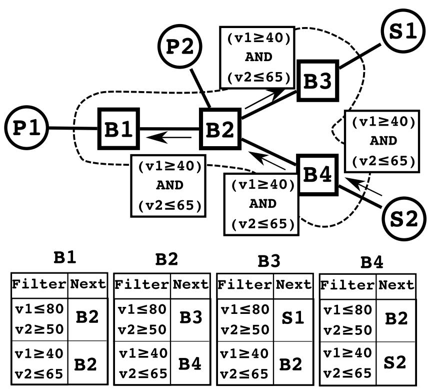
We assume the use of *soft state maintenance*[27, 46] in our publish/subscribe model. Specifically, any connecting and non-faulty subscriber must periodically sends its subscription message to the broker network to maintain its subscription. Any subscription stored in a broker's subscription table that is not renewed within a specified period of time will be considered expired and removed from the table. We do not discuss the appropriate renewal period and expiration period. Such research direction can be found in existing works[27, 46].



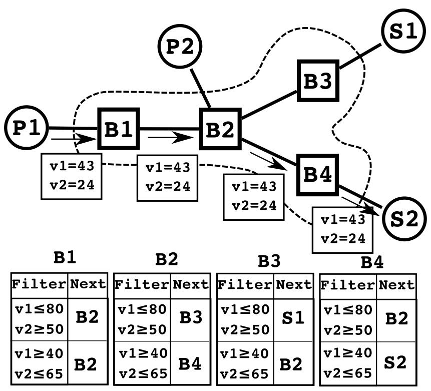
(a) Tree topology



(b) Subscriber S_1 subscribes



(c) Subscriber S_2 subscribes



(d) Publisher P_1 publishes an event

Figure 2.2: Example of subscription propagation and event routing in publish/subscribe broker tree

2.4 Quality of Service Metrics

To quantify quality of service, we define a subscriber-level metric called *subscriber real-time reliability* as follows.

Subscriber Real-time Reliability : A subscriber s is said to receive the service with real-time reliability $r_s \in [0, 1]$, where r_s is defined as the fraction of all events of s's interest that arrives at s before its deadline (i.e., delivery delay less than the message lifetime).

In the other word, r_s is the fraction of all messages matching s's interest that are delivered to s on time.

Subscriber real-time reliability can be analytically defined as follows. Let $\mathbb{E}_s$ be the set of all events that are published during a subscriber s's lifetime in the system. Hence, $F_s(\mathbb{E}_s)$ is the set of all events that match s's interest during its lifetime in the system. For each event $e \in F_s(\mathbb{E}_s)$, let d_e^s be the *delivery delay* of event e to subscriber s (the time period between e's publishing time and e's delivery time to s). Thus, the real-time reliability at a subscriber s, denoted by r_s, can be expressed as

$$r_s = \frac{|\{e \in F_s(\mathbb{E}_s) : d_e^s \leq d_e\}|}{|F_s(\mathbb{E}_s)|}$$

Since the proposed real-time subscriber reliability combines the concept of standard reliability with the concept of timeliness property, it can be used as a good indicator how much quality of service each subscriber receives. It is the user-oriented QoS metric that affects directly to clients' satisfaction with the publish/subscribe service. Throughout this thesis, we use the term *subscriber reliability* and *subscriber real-time reliability* interchangeably.

2.5 Subscriber Reliability Estimation Problem

As mentioned, subscriber reliability is an important user-oriented metric from each subscriber's perspective. Hence, we would like to estimate r_s for each subscriber s, which leads to the subscriber real-time reliability estimation problem.

Subscriber Real-time Reliability Estimation Problem: Given a publish/subscribe overlay network $\mathbb{G} = (\mathbb{N}, \mathbb{L})$, where $\mathbb{N} = \mathbb{B} \cup \mathbb{P} \cup \mathbb{S}$, find the estimated value of r_s, denoted by r'_s, for each subscriber $s \in \mathbb{S}$.

The subscriber real-time reliability estimation algorithm is the core part of this thesis. Through Chapter 3, 4, and 5, this thesis presents subscriber reliability estimation algorithms in different settings and assumptions. Finally, Chapter 6 discusses the possibility to integrate all models from each chapter together.

Once each subscriber's real-time reliability is estimated, the system can employ autonomic adjustment to the system in order to increase the subscribers' reliability. For example, in Figure 2.2(d), the system may increase the capacity of brokers $B4$ when the subscriber $S2$'s reliability level is too low. Some other types of performance optimizations based on the proposed analytical model are also discussed throughout various chapters.

CHAPTER 3

RELIABILITY AND TIMELINESS ANALYSIS WITH CONTENT DYNAMISM

3.1 Introduction

The most fundamental distinction between content-based publish/subscribe systems and topic-based publish/subscribe systems, or even conventional point-to-point routing protocols, is traffic predictability. In topic-based publish/subscribe systems and point-to-point routing protocols, once the traffic rate from each information producer is known, it is trivial to calculate the amount of traffic that flows to each path to each consumer. This is because there is a clear, deterministic mapping whether a publisher's (sender's) generated content should be sent to a subscriber (receiver) or not. In point-to-point routing protocols, the mapping is done by the receiver field in the protocol header. In topic-based publish/subscribe systems, the mapping is achieved by a single identifier called *topic*. In contrast, the mapping between each publisher to each receiver in content-based publish/subscribe systems is not clear as a subscriber may specify, in addition to its topic of interest, its *content filter* which allows the subscriber to specify its interest to receive only certain events containing certain attributes in a more fine-grained manner. Such technique provides more flexibility to subscribers but imposes more complexity and uncertainty at the brokers. Specifically, it increases the event processing time at each broker to math each whole event to complex content filters. Also, it is hard to determine exact amount of traffic that flows through the path from each publisher to each subscriber (see Figure 3.1 for examples), making it hard to calculate each broker load and queuing delay. Conversely, if the amount of traffic through the path from each publisher to each subscriber is known, or even predicted with some accuracy, we could then analyze the performance of content-based publish/subscribe systems by existing approaches in traditional point-to-point routing analysis such as stochastic and queuing theory.

As mentioned, while it is not possible to deterministically specify the amount of traffic between each publisher-subscriber pair, it is sufficient ,for the purpose of publish/subscribe performance

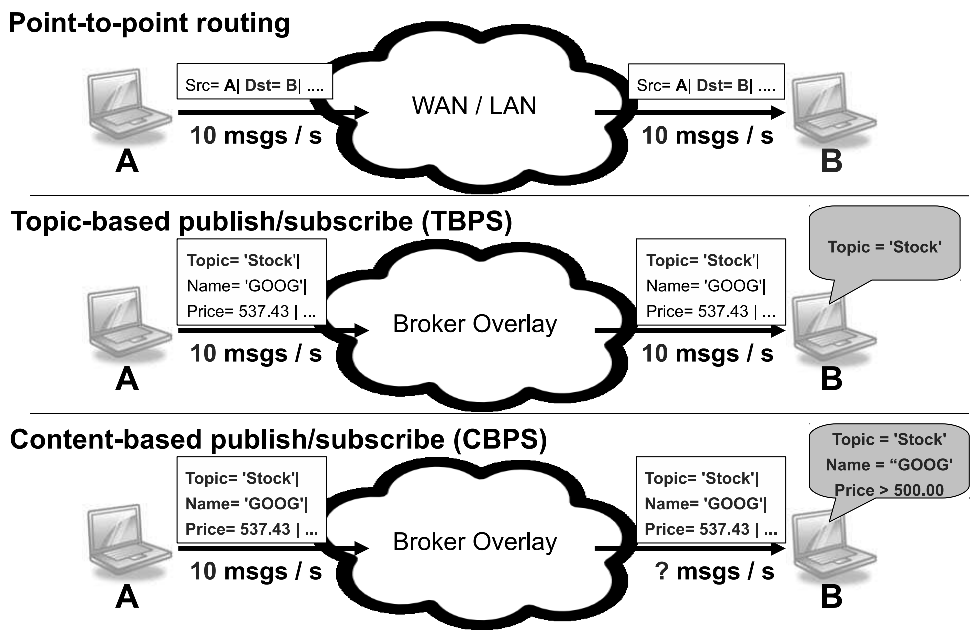
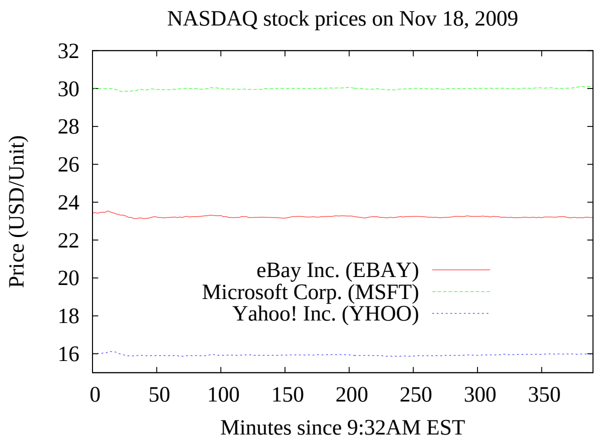

Point-to-point routing
Src= A| Dst= B| ....
WAN / LAN
10 msgs / s
A
B
Topic-based publish/subscribe (TBPS)
Topic= 'Stock'| Name= 'GOOG'| Price= 537.43 | ...
Broker Overlay
Topic = 'Stock'
Content-based publish/subscribe (CBPS)
Topic = 'Stock' Name = "GOOG' Price > 500.00
? msgs / s

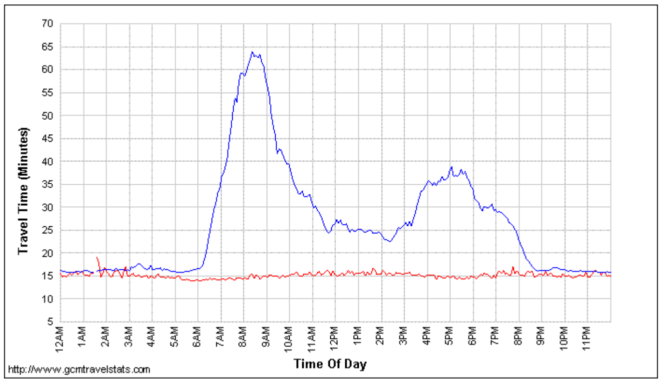
Figure 3.1: Examples of load calculation in each communication paradigms

analysis, to estimate the amount of such pair-wise traffic. In order to estimate each publisher-subscriber pairwise traffic, it is necessary to know or estimate the publisher's *content distribution*, which is the distribution of event attributes the publisher publishes. Note that estimating publisher content distribution is feasible, as many real-world event publishers exhibit temporal locality such that its content distribution can be predicted based on its publishing history (see Figure 3.2 for examples). Given the estimated content distribution of a publisher and the content filter of a subscriber, the average amount of traffic that flows between such publisher-subscriber pair can be calculated.

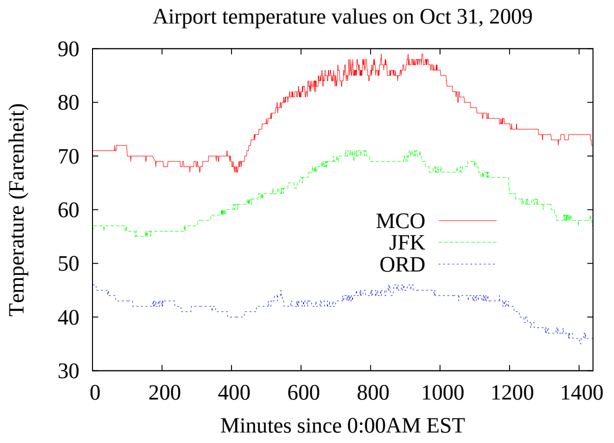
Yet, once the average traffic rate for each publisher/subscriber pair is determined, special care must be taken to calculate the load of each link and broker. This is because the calculation of broker/link traffic cannot use simple average arithmetics such as summation, but also needs to consider subscription filter from subscribers as well. As an example how subscription filters affect broker/link load calculation, consider a simple publish/subscribe system topology in Figure 3.3. Let publisher p_1 publish events in topic τ_1. Let subscriber s_1 and s_2 subscribe events of topic τ_1 (the same topic as p_1 topic) with subscription content filter f_{s_1} and f_{s_2} respectively. Assuming we know the content distribution of publisher p_1, we can calculate $\lambda_{p_1 s_1}$ and $\lambda_{p_1 s_2}$, which are the estimated amounts of traffic from p_1 to s_1 and s_2, respectively. However, it is not trivial to calculate the



(a) NASDAQ stock prices (Source: Google Finance[35])



(b) Chicago highway daily traveling time (Source: GCM Travel[36])



(c) Airport temperature report (Source: NCDC[37])

Figure 3.2: Example of real-world event streams and their temporal locality

amount of traffic that pass through link $l_{b_1 b_2}$, denoted by λ_l. For example, if both subscribers have exactly the same subscription filter (i.e., $f_{s_1} = f_{s_2}$), then $\lambda_l = \lambda_{p_1 s_1} = \lambda_{p_1 s_2}$ as s_1 and s_2 will receive the same set of events, which will be transmitted only once at broker b_1. At the other spectrum, if subscribers have completely disjoint subscription filter (i.e., $f_{s_1} \cap f_{s_2} = \emptyset$), then $\lambda_l = \lambda_{p_1 s_1} + \lambda_{p_1 s_2}$, as both subscribers receive totally different sets of events. Thus, the effect of publish/subscribe content-centric nature becomes a challenge to its load and performance calculation.

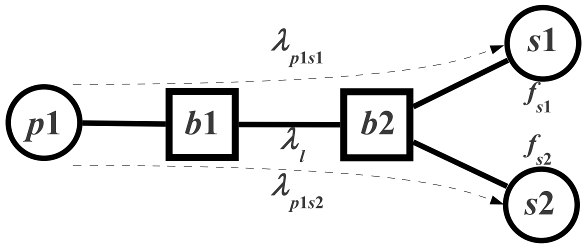


Figure 3.3: Example of the effect of subscription filter to rate calculation

To overcome such challenge, we extend use the concept of publisher content distribution to links and brokers as well, resulting in the concept of *entity content distribution*. The proposed analysis makes use of the entity content distribution to calculate the load at each part of the system, and then applies stochastic techniques and queuing theory to calculate the end-to-end delivery delay, which is then used to calculate subscriber real-time reliability.

In this chapter, we propose the analytical model to estimate subscriber reliability with the focus on predict event queuing and processing delay. We first define the problem of static subscriber real-time reliability, which has the assumption of reliable broker overlay and static broker tree (these assumptions will be relaxed in Chapter 5). We then present the algorithm to estimate subscriber reliability with the emphasis on broker/link queuing/processing delay distribution. The algorithm is based on the use of entity content distribution. Then, we present the subscriber admission control algorithm an example of performance optimizations that leverage the use of the predictive model. Finally, we present the extensive simulation results of the proposed system under realistic parameters and real-world event traces. The evaluation results yield strong accuracy for the prediction algorithm even when the statistical information of each component is inaccurate.

3.2 Static Subscriber Real-time Reliability

3.2.1 Formulation

As mentioned, this chapter of thesis focuses on the estimation of subscriber reliability with respect to broker/link queuing/processing delay. Hence, in this chapter, we make the following additional assumptions to filter out the effects from other types of dynamism and reduce the complexity of the analysis.

Single, static broker tree : As mentioned in Chapter 2.1, a broker overlay can be a generic connected graph, and each broker generally maintains multiple spanning trees with each tree corresponding to each topic. However, to facilitate the analysis in this chapter, we consider the case where all publishers, subscribers, and brokers use the same, single broker tree. This can be found in the scenarios where the entire broker overlay graph itself is a tree, or all publishers and subscribers are interested in a single topic. However, the analysis in this chapter can be easily generalized to multiple-tree scenarios.

No publisher/subscriber mobility : In this chapter, each subscriber is assumed to stay with one local broker during its lifetime. There is no change in subscriber set $\mathbb{S}$ and publisher set $\mathbb{P}$ in this analysis. In practice, the change of subscriber set $\mathbb{S}$ and publisher set $\mathbb{P}$ can be handled by running the analysis algorithm whenever such changes occur.

No long-term broker/link failure : We assume each broker and its links do not crash. Hence, the only cause of event loss in this chapter are probabilistic link loss and event expiration. Long-term broker/link failures will be considered in Chapter 5.

Known publisher content distribution : We assume each publisher's publishing content distribution is given or estimated. This assumption is necessary as it is not possible at all to determine the load without it. With this assumption, we revise the publisher model previously defined in Chapter 2.2.3 to incorporate the content distribution as follows.

Publisher Model : A publisher $p \in \mathbb{P}$ is defined as a tuple

$$p = (id_p, C_p(a, d), I_p(d))$$

where $C_p : \mathbb{V} \times \mathbb{D} \to [0, 1]$ is the content-lifetime joint distribution function of events that p publishes (i.e., $C_p(a, d)$ is the probability that p will publish an event with content $a \in \mathbb{V}$ and lifetime $d \in \mathbb{D}$), $I_p : \mathbb{R} \to [0, 1]$ is the inter-event publishing time distribution (i.e., $I_p(d)$ is the probability that the time between p's successive event publications is d), and id_p is the publisher's identifier. Thus,

$$\textstyle\sum_{(a,d)\in\mathbb{V}\times\mathbb{D}} C_p(a, d) = 1 \text{ and } \sum_{d>0}^{\infty} I_p(d) = 1$$

Both Content-lifetime distribution function $C_p(a, d)$ and inter-event publishing time distribution function $I_p(d)$ can be obtained via statistical estimation from publisher p's publishing history[12].

Static Subscriber Real-time Reliability Estimation Problem: Given a static, reliable publish/subscribe tree $\mathbb{G} = (\mathbb{N}, \mathbb{L})$ where $\mathbb{N} = \mathbb{B} \cup \mathbb{P} \cup \mathbb{S}$, find the estimated value of r_s, denoted by r'_s, for each subscriber $s \in \mathbb{S}$.

3.2.2 M/M/1 Estimation Algorithm

In this section, we present the subscriber reliability estimation algorithm. The estimation algorithm takes the publish/subscribe tree $\mathbb{G} = (\mathbb{B} \cup \mathbb{P} \cup \mathbb{S}, \mathbb{L})$ along with the statistical information of each component as the input and estimates the real-time reliability value r'_s for each subscriber $s \in \mathbb{S}$. To do so, it is necessary to estimate the end-to-end delivery delay distributions and path reliability distributions of all s's matching events when they arrive at s. Hence, we introduce another set of variables in Table 3.1 for the purpose of the analysis. These variables are not parts of the problem definition, but are defined as intermediate variables in order to solve the estimation problem. The overall estimation process, depicted in Figure 3.4, consists of four steps : propagating subscription filters, calculating per-link event flow rate, calculating broker queuing/processing delay, and calculating per-link content-remain time distribution.

Symbol	Definition
$C_p(a,d)$	content-lifetime distribution of events published by p
$I_p(d)$	publisher p's event publishing interval distribution
f_l	union of all subscription filters propagated via link l
$in(l)$	link l's sink node
$out(l)$	link l's source node
λ_l	estimated event flow rate through link l
λ_p	estimated event flow rate from publisher p
$C_l(a)$	estimated content distribution of events through link l
$up(l)$	upstream link set of link l (Equation (3.3))
λ_b	estimated event flow rate to broker b
μ_b	estimated event processing rate at broker b
q_b	estimated queuing delay at broker b
$D_b(d)$	estimated total delay distribution at broker b
$C_l(a,d)$	estimated content-remain time distribution of events through link l
z_p^2	publisher p's event traffic flow burstiness
z_l^2	estimated event flow burstiness through link l
z_b^2	broker b's service burstiness
z_{bi}^2	broker b's incoming flow burstiness
z_{bo}^2	broker b's outgoing flow burstiness
r_s^*	subscriber s's requested real-time reliability
$U(G)$	publish/subscribe network G's utility
$\phi(s)$	subscriber s's utility score
α	publisher information skewness

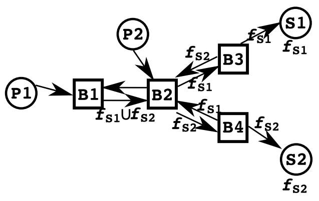
Table 3.1: Content dynamism analysis variables' notation

Note that in this chapter, A bi-directional link between two brokers will be modeled as two directional links for the purpose of event flow calculation in each direction of the link. We define $out(l) \in (\mathbb{P} \cup \mathbb{B})$ and $in(l) \in (\mathbb{B} \cup \mathbb{S})$ as the source node and the sink node of link l, respectively. For example, Figure 3.4 shows the analytical view of the publish/subscribe system from Figure 2.2.

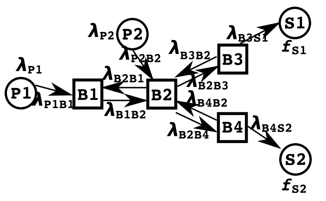
3.2.2.1 Subscription Filter Propagation

In this step, the subscription filters are propagated from subscribers to each broker in the system in the same manner as subscription propagation process discussed in Section 2.3.1. As shown in Figure 3.4(a), each subscription is propagated in the reverse direction of the event flow direction (i.e., reversed to the direction of the arrows). When a subscription filter f is propagated to a broker b via one of b's outgoing links l, the subscription will be propagated to all other incoming links of b. At the same time, the subscription filter f will be merged into l's *link filter*, denoted by f_l. The link filter $f_l \subseteq \mathbb{V}$ can be viewed as the union of filters from each subscription that is propagated through link l and hence represents the content space of the events that should be forwarded to link l.

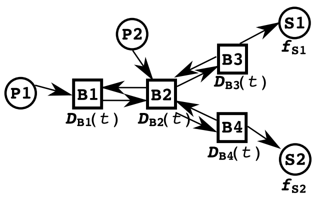
That is, at the beginning of this step, each link l has its link filter empty (i.e. $f_l = \emptyset$). For each new subscription filter f_s that is propagated to link l, $f_l = f_l \cup f_s$. The process continues until all



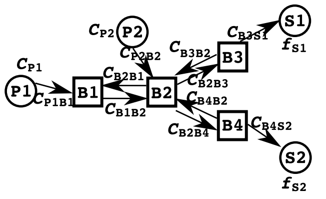
(a) Propagating subscription (reversed direction of arrows)



(b) Calculating link event rate



(c) Calculating broker total delay distribution



(d) Calculating per-link content-remain time distribution

Figure 3.4: Steps for subscriber reliability estimation

subscriptions are propagated to all brokers in the system. At the end of this step, if any link l's filter set still remains empty, then it means that there will be no event sent over that link l.

3.2.2.2 Per-link Event Flow Rate Calculation

After each link's filter set is identified, the next step is to calculate each link l's average event flow rate λ_l, as depicted in Figure 3.4(b). This step starts by calculating the average event generation rate at each publisher p, denoted by λ_p, as the inverse of average inter-event generation time $I_p(d)$ as follows.

$$\lambda_p = \frac{1}{\mathrm{E}[I_p(d)]} = \frac{1}{\sum_{d:I_p(d)>0}(d.I_p(d))}$$

The average event flow rate of a *publisher-broker* link l is then equal to the event flow rate of l's source publisher, multiplied by the link's reliability r_l as follows.

$$\lambda_l = r_l.\lambda_p \tag{3.1}$$

, where $p = out(l)$

The process continues until the event flow rates of all *publisher-broker* links are determined. Then, the event flow rates of the other links (i.e. *broker-broker* links and *broker-subscriber* links) are calculated. To do so, the *content distribution* of each publisher-broker link is needed. The content distribution of a link l, denoted by $C_l : \mathbb{V} \rightarrow [0,1]$, is the probability distribution of the event content that passes through link l (i.e., $C_l(a)$ is the probability that an event passing through link l contains attributes $a \in \mathbb{V}$). For each publisher-broker link l that connects a publisher p, the content distribution is equal to the content projection of the p's content-lifetime distribution as follows.

$$C_l(a) = \sum_{d \in \mathbb{D}} C_p(a,d), \forall a \in \mathbb{V} \tag{3.2}$$

where $p = out(l)$

A link l is called *resolved* if its average flow rate λ_l and content distribution $C_l(a)$ are calculated. Hence, each publisher-broker link can be resolved using Equation (3.2). After all publisher-broker links are resolved, links of other types (i.e., broker-broker links and broker-subscriber links) will be resolved as follows. We define the *upstream link set* of a link l, denoted by $up(l)$, as the set of all incoming links to l's source broker $out(l)$, excluding l's reversed link. In other words,

$$up(l) = \{l' \in \mathbb{L} : in(l') = out(l) \wedge out(l') \neq in(l)\} \tag{3.3}$$

That is, $up(l)$ contains all l's adjacent links from which events potentially flow to l. For example, in Figure 3.4, link *B2B1*'s upstream link set consists of link *P2B2*, *B3B2*, and *B4B2*.

Any broker-broker or broker-subscriber link l is defined as *resolvable* if and only if all links in l's upstream link set $up(l)$ are resolved. For each resolvable link l, its average flow rate λ_l and content distribution $C_l(a)$ can be calculated from its upstream link set by the following equation.

$$\lambda_l = r_l.\lambda_{up(l)}.\sum_{a \in f_l} C(a) \tag{3.4}$$

and

$$C_l(a) = \frac{r_l.\lambda.C_{up(l)}(a)}{\lambda_l}, \forall a \in f_l$$

where $\lambda_{up(l)}$ and $C_{up(l)}(a)$ are the rate and content distribution of all l's total upstream event flows. Specifically,

$$\lambda_{up(l)} = \sum_{l' \in up(l)} \lambda_{l'} \tag{3.5}$$

and

$$C_{up(l)}(a) = \frac{\sum_{l' \in up(l)} \lambda_{l'}.C_{l'}(a)}{\lambda_{up(l)}}$$

That is, link l's flow rate λ_l is calculated from the total rate of all l's incoming event flows that match the filter set f_l. The content distribution $C_l(a)$ is then calculated in the same manner. Note that for each link l, $\sum_{a \in \mathbb{V}} C_l(a) = 1$ and $\sum_{a \in \mathbb{V}} C_{up(l)}(a) = 1$.

Once a resolvable link's flow rate λ_l and content distribution $C_l(a)$ are identified, that link then becomes a resolved link. The process then continues to resolve the remaining links until all links are resolved. Since we assume the broker network to be acyclic, it is guaranteed that the process always finds a new resolvable link until all links are resolved.

3.2.2.3 Broker Total Delay Calculation

After all the links are resolved, we then determine the average queuing delay at each broker. Since we model each broker as an event matching server with a single queue, we can apply queuing theory techniques to determine broker queuing delay as follows. A broker b's average queuing delay, denoted by q_b, can be calculated based on M/M/1 queuing model as follows.

$$q_b = \frac{\lambda_b}{\mu_b(\mu_b - \lambda_b)} \tag{3.6}$$

where

$$\lambda_b = \sum_{l \in \mathbb{L}: in(l) = b} \lambda_l \tag{3.7}$$

and

$$\mu_b = \frac{1}{\mathrm{E}[M_b(d)]} = \frac{1}{\sum_{d: M_b(d) > 0} (d.M_b(d))} \tag{3.8}$$

In other words, λ_b is the total event flow rate from all of b's incoming links, and μ_b is b's average matching rate.

Note that if the event flow rate λ_b is more than the average matching rate μ_b, then the broker b

is overloaded. In such a case, the queuing delay at broker b will be equal to infinity, as the broker will never reach the stable state.

Once b's average queuing delay is determined, we then estimate b's total broker delay distribution $D_b(d)$ as

$$\forall d : M_b(d) > 0, D_b(d + q_b) = M_b(d)$$

That is, the total broker delay distribution is estimated as the event processing delay distribution plus the average queuing delay. Although the proposed approach is a simple delay distribution estimation based on the assumption of M/M/1 queue model, the evaluation result presented in Section 3.4.1.1 yields accurate results for non-exponential traffic rate and service rate as well. However, we further improve the delay estimation accuracy using more sophisticated techniques in queuing theory in Section 3.2.3.

3.2.2.4 Per-link Content-remain time Distribution Calculation

After the queuing and matching delay distributions at all brokers are identified, the last step is to calculate the content and lifetime distribution at each link. To do so, we define content-remain time joint distribution at each link l, denoted by $C_l(a, d)$, as the joint probability of the content and remaining lifetime of each event that passes through link l. Note that it is possible that $C_l(a, d) > 0$ when d is negative, which means that such fraction of events is already expired after they pass through link l.

As shown in Figure 3.4(d), the process at this step is similar to the per-link event flow rate calculation described in Section 3.2.2.2, except that both content and lifetime are now considered in the calculation. Specifically, for each publisher-broker link l, the content-remain time distribution $C_l(a, d)$ is calculated as

$$C_l(a, d) = \sum_{d' : D_l(d') > 0} (D_l(d').C_p(a, d + d')), \forall a \in \mathbb{V} \tag{3.9}$$

, where $p = out(l)$ and $D_l(d)$ is l's link delay distribution. The reason behind Equation (3.9) is that once an event is transmitted via link l, its remaining lifetime is shortened by link l's transmission delay.

Here we once again use the concept of resolved link and resolvable link from Section 3.2.2.2, except that in this section, a link l is resolved when its content-delay distribution is identified. Hence, we apply Equation (3.9) to all publisher-broker links, making all of them resolved. We then repetitively find a resolvable link l and calculate its content-remain time distribution as follows.

$$C_l(a, d) = \frac{r_l.\lambda_{up(l)}}{\lambda_l}. \sum_{d':D_l(d')>0} (D_l(d').C_{up(l)}(a, d + d')), \forall a \in f_l \tag{3.10}$$

, where

$$C_{up(l)}(a, d) = \sum_{d':D_b(d')>0} \frac{D_b(d'). \sum_{l' \in up(l)} \lambda_l.C_l(a, d + d')}{\lambda_{up(l)}} \tag{3.11}$$

, and $\lambda_{up(l)}$ is calculated from Equation (3.5).

Hence, the estimated reliability r'_s can then be calculated as

$$\begin{aligned} r'_s &= \frac{\text{rate of unexpired matching events delivered to } s}{\text{total rate of all events that match } s\text{'s interest}} \\ &= \frac{\lambda_l. \sum_{(a \in f_s, d>0)} C_l(a, d)}{\sum_{p \in \mathbb{P}} \lambda_{ps}} \end{aligned} \tag{3.12}$$

where l is the link to s (i.e., $s = in(l)$) and λ_{ps} is the *pairwise average event flow rate* between publisher p and subscriber s and can be calculated as

$$\lambda_{ps} = \sum_{a \in f_s} C_p(a).\lambda_p \tag{3.13}$$

With Equation (3.12) and Equation (3.13), we can calculate the estimated real-time reliability R'_s at each subscriber s from static publish/subscribe tree $\mathbb{G} = (\mathbb{N}, \mathbb{L})$.

3.2.3 G/G/1 Estimation Algorithm

So far, the load estimation at each broker presented in this section uses M/M/1 queue model, which assumes event inter-arrival time distribution and broker processing time distribution to be exponential random variables. Such assumption may not result in accurate subscriber reliability estimation as each event inter-arrival time and broker processing time may be drawn from other distributions than exponential distribution. For example, the event inter-arrival time may be deterministic (i.e. publishers with periodic sensors) or the broker event processing time may be uniform (i.e. brokers matching a random event with an array of subscriptions). To address complex time distribution for more accurate reliability estimation, this section presents the improved estimation algorithm based on G/G/1 queue model.

3.2.3.1 Publisher Flow Burstiness Calculation

To model the system using G/G/1 model, we follow the approach by Whitt[47] and introduce additional analytical variables as follows. Apart from event flow rate λ_p at each publisher p, another variable called event *flow burstiness*, denoted by z_p^2, is calculated from p's event inter-arrival time distribution $I_p(d)$ as

$$z_p^2 = \frac{\mathrm{Var}[I_p(d)]}{\mathrm{E}[I_p(d)]^2} = \frac{\sum_{d:I_p(d)>0} I_p(d).(d - \frac{1}{\lambda_p})^2}{(\frac{1}{\lambda_p})^2} \tag{3.14}$$

the burstiness variable z_p^2 hence represents the uniformity level of event generation interval at p. For example, $z_p^2 = 0$ when $I_p(d)$ is a uniform distribution and $z_p^2 = 1$ when $I_p(d)$ is an exponential distribution.

Also, at each pub/sub broker b, the burstiness variable z_b^2 is calculated from its event matching time distribution $M_b(d)$ in the same way z_p^2 is calculated at each publisher p. That is,

$$z_b^2 = \frac{\mathrm{Var}[M_b(d)]}{\mathrm{E}[M_b(d)]^2} = \frac{\sum_{d:M_b(d)>0} M_b(d).(d - \frac{1}{\mu_b})^2}{(\frac{1}{\mu_b})^2} \tag{3.15}$$

With the event generation burstiness variable z_p^2 at each publisher p and the event matching burstiness variable z_b^2 at each broker b, a more accurate subscriber reliability estimation algorithm can be done by the approaches presented in Section 3.2.2.1 through Section 3.2.2.4 but with one additional step between the step in Section 3.2.2.2 and the step in Section 3.2.2.3 in order to calculate link and broker flow burstiness.

3.2.3.2 Broker/link Flow Burstiness Calculation

The process of per-link event flow burstiness calculation starts after the process of per-link event flow rate calculation (Section 3.2.2.2) is done. After the flow rate calculation process, the per-link event flow rate λ_l and content distribution $C_l'(a)$ is known for each link l. Also, the per-publisher event flow burstiness z_p^2 for each publisher p and per-broker event matching burstiness z_b^2 for each publisher b are known via equation (3.14) and (3.15) respectively. The per-link event flow burstiness calculation process aims to calculate per-link event flow burstiness z_l^2 for each link l. The techniques used in the calculation are adopted from traditional queuing network theory[47].

The process starts by calculating z_l^2 for each *publisher-broker* link l using the asymptotic method[47] as follows.

$$\forall l \in \mathbb{L} : out(l) \in P, z_l^2 = r_l.z_p^2 + 1 - r_l \tag{3.16}$$

, where $p = out(l)$

To calculate per-link event flow burstiness for *broker-broker* and *broker-subscriber* links, a set of linear equations must be solved according to the following set of rules.

Incoming Flow Superposition: we define *per-broker incoming flow burstiness*, denoted by z_{bi}^2 for each broker b, as the burstiness of the total event flow coming from all b's incoming links. Using the superposition rule and the asymptotic method, the per-broker incoming flow burstiness is the convex combination of each per-link flow burstiness as follows.

$$\forall b \in \mathbb{B}, z_{bi}^2 = \sum_{l \in \mathbb{L}:in(l)=b} \left(\frac{\lambda_l}{\lambda_b}.z_l^2 \right) \tag{3.17}$$

, where λ_b is the total incoming event flow rate at broker b calculated from Equation (3.7).

Equation (3.17) takes place at each broker $b \in \mathbb{B}$ in the system. Hence, there are $|\mathbb{B}|$ incoming flow equations.

Broker Incoming-Outgoing Flow Transformation: we define *per-broker outgoing flow burstiness*, denoted by z_{bo}^2 for each broker b, as the burstiness of the total event flow going out from broker b to all b's outgoing links. Using Marshall's formula[47], the per-broker outgoing flow burstiness z_{bo}^2 is a function of total incoming flow burstiness z_{bi}^2, total incoming flow rate λ_b (Equation (3.7), broker average event matching rate μ_b (Equation (3.8)), broker event matching burstiness z_b^2 (Equation (3.15)) as follows.

$$\forall b \in \mathbb{B}, z_{bo}^2 = (\rho_b^2.z_b^2 + (1 - \rho_b^2).z_{bi}^2 \tag{3.18}$$

, where $\rho_b = \frac{\lambda_b}{\mu_b}$

Since Equation 3.18 takes place at each broker $b \in \mathbb{B}$, there are $|\mathbb{B}|$ incoming-outgoing flow equations.

Broker Outgoing Flow Splitting: after a broker fetches the incoming event from the head of the queue, it routes the event to each outgoing link with the subscription that matches the event. Hence, the per-link event flow burstiness of each outgoing link z_l^2 is a function of its source broker's incoming traffic rate λ_b (From Equation (3.7)) and its own traffic rate λ_l (From Equation (3.4)) as follows.

$$\forall l \in \mathbb{L} : out(l) \in \mathbb{B}, z_l^2 = \frac{\lambda_l}{\lambda_b}.z_{bo}^2 + 1 - \frac{\lambda_l}{\lambda_b} \tag{3.19}$$

, where $b = out(l)$

From the three equations (Equation (3.17), (3.18), and (3.19)), there are three forms of unknown variables (z_{bi}^2,z_{bo}^2, and z_l^2). All other variables are known from previous calculations. Since each unknown variable z_l^2 can be written in a linear form of some variable z_{bo}^2 using Equation (3.19) and each unknown variable z_{bo}^2 can be written in a linear form of some variable z_{bi}^2 using Equation (3.18), there are $|\mathbb{B}|$ unknown variables left, which are in the form of z_{bi}^2. Also, there are $|\mathbb{B}|$ equations left (Equation (3.17)). Since there are $|\mathbb{B}|$ unknown variables left with $|\mathbb{B}|$ linear equations, each variable z_{bi}^2 for each broker $b \in \mathbb{B}$ can be solved by using standard matrix operations. Once variables in the form of z_{bi}^2 are solved, other unknown variables in the forms of z_{bo}^2 and z_l^2 are also solved using Equation (3.18) and (3.19). However, only variables in the form of z_{bi}^2 are needed in the next step to calculate the queuing delay at each broker.

3.2.3.3 Improved G/G/1 Broker Delay Estimation

Once the per-broker total incoming flow burstiness z_{bi}^2 is calculated, a more accurate estimation of the average queuing delay q_b for each broker $b \in \mathbb{B}$ is then calculated as a function of total incoming flow burstiness z_{bi}^2, total incoming flow rate λ_b (Equation (3.7)), broker average event matching rate μ_b (Equation (3.8)), broker event matching burstiness z_b^2 (Equation (3.15)) as follows.

$$q_b = \frac{\rho_b.(z_{bi}^2 + z_b^2).g(\rho_b, z_{bi}^2, z_b^2)}{2.\mu_b.(1 - \rho_b)} \tag{3.20}$$

where $\rho_b = \frac{\lambda_b}{\mu_b}$ and

$$g(\rho_b, z_{bi}^2, z_b^2) = \begin{cases} \exp(-\frac{2(1-\rho_b).(1-z_{bi}^2)^2}{3\rho_b.(z_{bi}^2+z_b^2)}) & \text{if } z_{bi}^2 < 1 \\ 1 & \text{if } z_{bi}^2 \geq 1 \end{cases}$$

Thus, we replace Equation (3.6) with Equation (3.20) to calculate the average broker queuing delay, which is then used to calculate content-remain time distribution and finally the subscriber reliability estimation as stated in Section 3.2.2.4. Note that when the incoming event flow rate and the event matching rate of a broker are exponentially distributed (i.e. $z_{bi}^2 = 1$ and $z_b^2 = 1$), then Equation (3.20) is reduced to Equation (3.6). The proposed G/G/1 model reliability estimation yields better estimation accuracy when compared to the M/M/1 model presented in Section 3.2.2.3.

3.3 QoS-aware Subscriber Admission Control

From the analytical framework presented in Section 3.2.2, we can conclude that the delivery delay can increase along with the load in the system. Hence, it is not efficient to admit to many subscribers into the system as the competing load will degrade subscriber reliability. Rather, the publish/subscribe system should admit only a subset of subscribers so that it has sufficient resource to satisfy the admitted subscribers' requirements.

In this section, we first introduce subscriber requested reliability model and system utility model. We then formally define the maximum-utility subscriber admission control problem, which is NP-hard in terms of complexity. Finally, we propose a set of heuristic algorithms to solve such problem.

3.3.1 Formulation

3.3.1.1 Publish/Subscribe Utility Model

Let each subscriber $s \in \mathbb{S}$ have its own real-time reliability requirement (i.e., requested reliability) $r_s^* = [0, 1]$, a subscriber s is said to have its requirement satisfied if $r_s \geq r_s^*$. We define the set of *satisfied subscribers*, denoted by $S' \in \mathbb{S}$, as the set of subscribers in $\mathbb{S}$ that have their reliability requirements satisfied (i.e. $S' = \{s \in \mathbb{S} : r_s \geq r_s^*\}$). Let S“ be the set of admitted subscribers in the system. Obviously, $S' \subseteq S$“. Let G“ $= (\mathbb{P} \cup \mathbb{B} \cup S$“$, \mathbb{L} - (\mathbb{B} \times (\mathbb{S} - S$“$)))$ be the publish/subscribe tree with only admitted subscribers. We define the utility of the publish/subscribe tree G'', denoted by

$U(G'')$ as the number of satisfied subscribers. That is $U(G``) = |S'|$.

Using the proposed utility function $U(G'')$, it is more beneficial not to admit the whole subscriber set $\mathbb{S}$ into the system if we know in advance that some subscribers will not meet their requirements, since those unsatisfied subscribers will only waste system resources without adding any benefit to the system. Instead, a subscriber s should be admitted to the system only when it is likely to have its requirement satisfied.

3.3.1.2 Maximum-utility Subscriber Admission Control Problem

As mentioned in Section 3.3.1.1, admitting all subscribers in the systems may result in bad system utility. Thus, the system should pick only a subset of subscribers that will maximize the system utility. Hence, we define maximum-utility subscriber admission control problem as follows.

Definition *Maximum-utility Subscriber Admission Control Problem:* Given a static, reliable publish/subscribe tree $\mathbb{G} = (\mathbb{N}, \mathbb{L})$, find the largest subscriber subset $S^* \subseteq \mathbb{S}$ that maximize the utility of the system (i.e. $S^* = \arg\max_{S``\subseteq \mathbb{S}} U(G''))$.

It is known that the subscriber admission control problem is an NP-Hard problem, since the problem can be mapped to other NP-hard optimization problems such as multicast admission control or multi-commodity flow problems, which were shown to be NP-complete[48, 49]. However, this work discusses a set of greedy, heuristic-based algorithms to solve the subscriber admission control problem in Section 3.3.2.

3.3.2 Heuristic Algorithms

In this Section, we propose the heuristic-based algorithm to solve the subscriber admission problem. That is, given a publish/subscribe network $\mathbb{G} = (\mathbb{N}, \mathbb{L})$, find the subset of subscriber set $S \in \mathbb{S}$, denoted by S^*, that will maximize system utility. In other words, $S^* = \arg\max_{S``\subseteq \mathbb{S}} U(G``)$. This algorithm runs in a centralized fashion at a control center node[1], which periodically collects monitoring status from each publisher/broker entities in the network and uses such collected status to run the subscriber reliability estimation and admission control every time a new subscriber joins the system.

[1]A control center node can be an arbitrary broker that is elected by leader election algorithms among brokers.

As mentioned, the subscriber admission problem is an NP-hard problem with respect to the total number of subscribers ($|\mathbb{S}|$). However, since we can estimate the system utility $U(G'')$ for any publish/subscribe tree G'' based on the approach presented in Section 3.2.2, we now then propose the heuristic-based, greedy algorithm framework, denoted by $A^*(\mathbb{G})$ to for the subscriber admission control problem (i.e., $A^*(\mathbb{G})$ approximates S^* for $\mathbb{G} = (\mathbb{N}, \mathbb{L})$).

Algorithm 1 Function $A^*(\mathbb{G} = (\mathbb{N}, \mathbb{L}))$

```
$S \Leftarrow \mathbb{S}$
$S" \Leftarrow \emptyset$
$U^* \Leftarrow 0$
**while** $S \neq \emptyset$ **do**
  $s \Leftarrow \arg\max_{s' \in S} \phi(s')$
  $G'' = (\mathbb{B} \cup \mathbb{P} \cup S'' \cup \{s\}, \mathbb{L} - (\mathbb{B} \times (\mathbb{S} - (S" \cup \{s\}))))$
  **if** $U(G") > U^*$ **then**
    $S" \Leftarrow S'' \cup \{s\}$
    $U^* \Leftarrow U(G")$
  **end if**
  $S \Leftarrow S - \{s\}$
**end while**
**return** $S"$
```

The Algorithm 1 presents the details of the greedy, heuristic-based subscriber admission control algorithm A^* to approximate the maximum-utility subscriber set S^*. The basic concept of the algorithm A^* is to initially set the admitted subscriber set S'' to empty set, and then grows the set S'' progressively by including each subscriber $s \in \mathbb{S}$ only when the addition of s can increase the system utility. The system utility can be approximated based on the analytical framework described in Section 3.2.2. The order of subscribers in the addition process is obtained on the priority function $\phi(s)$, which gives a priority value to each subscriber s. Since each subscriber is considered only once in the addition process, the priority function $\phi(s)$ must be chosen carefully to achieve near-optimum maximum-utility subscriber set.

In this work, we pick a set of heuristic subscriber priority functions $\phi(s)$ to be used with the maximum-utility subscriber admission algorithm framework A^* as follows.

Random Priority (random): The priority value of each subscriber is determined randomly based on its identification number (i.e. $\phi(s) = id_s$).

Requirement Priority (hi-req-first): The priority value of each subscriber is equal to the reliability requirement of itself (i.e., $\phi(s) = r_s^*$). Hence, the subscriber with higher reliability requirement will be considered before the one with lower reliability requirement in this priority function.

Inversed Requirement Priority (low-req-first): The priority value of each subscriber is equal to the inverse of the reliability requirement of itself (i.e. $\phi(s) = 1 - r_s^*$). This scheme is the opposite of the requirement priority scheme, as the subscriber with lower reliability will be considered first in this scheme.

Additional Content Priority (overlap-first): In this scheme, the first $\log_2|\mathbb{S}|$ subscribers will have random priority (i.e. $\phi(s) = id_s$). However, after $\log_2|\mathbb{S}|$ subscribers, the subscriber priority s will be calculated as the inverse of the size of additional filter space incurred by adding such subscriber (i.e. $\phi(s) = \frac{1}{|f_s - f_{s^*}|}$ where $f_{s^*} = \bigcup_{s \in s^*} f_s$).

In Section 3.4.3, we will evaluate and compare the effectiveness of each subscriber priority function to approximate the maximum-utility subscriber set in the publish/subscribe system.

3.4 Evaluation

In this section, we present the evaluation results of our proposed analytical framework. The evaluation is done via simulation using ns-2 network simulator[50]. The simulation uses both synthetic event trace (Section 3.4.1) and real-world event trace (Section 3.4.2). Both experiments take place in a broker tree consisting of 20 broker nodes. The link delay between broker nodes are derived from Planetlab delay and bandwidth traces that were collected by Ripeanu et al[51, 52]. The event processing delay distribution at each broker is interpolated from broker performance report in publish/subscribe event matching algorithm works[53, 54]. Specifically, the average event matching time at each broker is linearly proportional to the number of subscriptions stored in that broker's routing table, with the increase rate roughly equal to 1 millisecond per 1 additional stored subscription.

3.4.1 Simulations with Synthetic Trace

This section presents simulation results of the prediction algorithm with the synthetic trace. The purpose of the evaluation with the synthetic trace is to observe the accuracy of the prediction algorithm under different event traffic patterns and broker service time distributions. Unless explicitly specified, each synthetic trace simulation is run with the parameters presented in the second column

Parameters	Synthetic	Real-world
#brokers	20	20
#topics	4	100
#publishers	8	100
#subscribers	100	2000
#event attributes (k)	21	see Table 3.3
event lifetime	1 second	1 second
event attribute distribution	Zipf-like	see Figure 3.8(b)
subscription filter distribution	Zipf-like	see Table 3.4
avg publishing rate	1 event/sec	see Figure 3.8(a)
Message size	64 bytes	64 bytes
Simulation Time	10000 seconds	23400 seconds
#Runs	5	5

Table 3.2: Simulation parameters

of Table 3.2. We vary publishers' publishing interval distribution between exponential, deterministic (i.e. periodic), and uniform publishing distributions. Also, we vary brokers' event matching distribution between exponential and uniform distributions.

3.4.1.1 Prediction with M/M/1 Broker Model

Figure 3.5 presents the accuracy of the subscriber reliability estimation algorithm using M/M/1 broker model presented in Section 3.2.2 under different distributions of each publisher's publishing interval and each broker's event processing interval. The y-axis of each graph represents the values of actual subscriber real-time reliability while the x-axis of the graph represents the values of predicted real-time reliability. Each single point in each graph represents one subscriber in one run of simulation. As shown in the result, our algorithm can predict subscriber reliability values accurately in all scenarios. The prediction is most accurate when publishing interval and event processing delay are both exponentially distributed (Figure 3.5(a)). While the results in non-exponential settings are less accurate, almost all predicted values are less than or equal to the actual reliability values. Hence, the prediction can still be used as tight upper bound of actual reliability.

3.4.1.2 Prediction with G/G/1 Broker Model

This section presents the accuracy of the subscriber reliability estimation using G/G/1 broker model. The experimental setting is the same as the setting in Section 3.4.1.1 except the estimation algorithm, which includes the flow burstiness calculation described in Section 3.2.3. Figure 3.6 shows the result of G/G/1 prediction. As seen from the result, the prediction algorithm with

G/G/1 model is more accurate than the one with M/M/1 model when the publication interval and matching interval are not exponentially distributed. When both publication interval and matching interval are exponentially distributed, both M/M/1 model and G/G/1 model produce the same result as explained in Section 3.2.3.

3.4.1.3 Effect of Imprecise Publisher Information

The reliability prediction results shown in Section 3.4.1.1 and Section 3.4.1.2 are based on the experiments with perfectly accurate publisher content-lifetime distributions. However, such assumption is not true in practice as the approximation of publisher's characteristic may not be accurate. This section presents the accuracy of the subscriber reliability prediction algorithm with imprecise publisher information. Specifically, we define *distribution skewness*, denoted by α, as the level of inaccuracy in the observed publisher content-lifetime distribution. Let $\tilde{C}_p(a,d)$ be the actual, hidden content-lifetime distribution of a publisher p, then the observed content-lifetime $C_p(a,d)$ of publisher p with skewness α is

$$C_p(a,d) = \frac{\tilde{C}_p(a,d)^\alpha}{\sum_{a\in\mathbb{V},d\geq 0} \tilde{C}_p(a,d)^\alpha}$$

That is, the observed probability that a publisher p will publish an event with content a and lifetime d will be equal to the actual probability of such event to the power of α, normalized by the total transformed weight. Hence, $\alpha = 1$ represents the scenario of perfectly precise publisher information.

Figure 3.7 presents the result of subscriber reliability prediction algorithm with the same parameter configuration as Section 3.4.1.1 and Section 3.4.1.2, but with different values of skewness (α). The results, shown in Figure 3.7, are based on exponentially distributed publishers' publication interval and brokers' event processing delay, so both M/M/1 model and G/G/1 model produce the same results. It can be seen that the accuracy of the prediction algorithm slightly decreases when $\alpha > 1$, but significantly decreases when $\alpha < 1$. The conclusion is that $\alpha < 1$ reduces the difference of content popularity in Zipf-like distribution, and thus affects flow estimation accuracy more than when $\alpha > 1$. However, the overall prediction accuracy is acceptable.

Field	Type	Min	Max	Example
Symbol	STRING	AAA	ZZZZZ	GOOG
Price	FLOAT	0.01	3000.00	593.56
Vol	INTEGER	0	4000000	18,378

Table 3.3: NASDAQ stock event structure

3.4.2 Simulation with Real-world Trace

This section presents the evaluation of the subscriber reliability prediction algorithm with the real-world publisher event trace. The simulation depicts the real-time stock market quote service, where each publisher publishes real-time quotes of a stock to subscribers that are interested in that stock. Each simulation is done with the parameters shown in the third column of Table 3.2. We obtain two consecutive days' worth of NASDAQ stock quote event trace from Google Finance[35] between Thursday December 4, 2009 and Friday December 5, 2009. The trace from each day starts from 9:30 AM ET to 4:00 PM ET, which are NASDAQ stock market hours of operation. The trace consists of 258,853 events from 2,792 stocks on the first day, and 272,974 events from 2,832 stocks on the second day. There are 2,733 overlapping stocks from both days. In each simulation run, we randomly pick up 100 stock traces from the overlapping stocks and assign each of them to each of 100 publishers. We use the first day's trace as the publishers' statistical information to predict the real-time reliability that each of 2,000 subscribers will receive on the second day. We then run the simulation using the trace of the second day (23,400 seconds duration), and compare the predicted reliability and the actual reliability each subscriber receives.

3.4.2.1 Event/Subscription Trace Characteristic

Each event in the NASDAQ stock trace consists of three attributes as shown in Table 3.3. We use the stock symbol as the topic of the event. However, since there is no actual subscription trace, we follow the approach by Gupta et al[41] to construct the subscriptions for each subscriber as follows. There are 5 types of subscriptions as shown in Table 3.4. Each subscription is drawn from each subscription type with its corresponding probability. For any single-topic subscription (i.e., subscription type 1-4), its topic is picked up from the 100 stocks with weights equal to each stock's market capitalization value in order to reflect differences in each stock's popularity.

Figure 3.8 shows the *temporal correlation* between the first day traces and the second day traces. Specifically, Figure 3.8(a) shows the temporal correlation between the first day event average rate and the second day event average rate. Each point in the figure represents the average publishing

Type	Filter	Meaning	Probability
1	Symbol = $P1$	stock $P1$	20%
2	(Symbol = $P1$) $\wedge$ (Price $\leq P2$)	a stock $P1$'s value is less than or equal to $P2$	35%
3	(Symbol = $P1$) $\wedge$ (Price $\geq P2$)	a stock $P1$'s value is more than or equal to $P2$	35%
4	(Symbol = $P1$) $\wedge$ (Vol $\geq P2$)	a stock $P1$'s volume is more than or equal to $P2$	5%
5	Vol $\geq P1$	any stock's trade volume is more than $P1$	5%

Table 3.4: NASDAQ stock subscription type distribution

rate of one distinct publisher that publishes a stock. As seen from the figure, each stock's rate changes only slightly across days, which supports the *temporal locality assumption* suggested in Section 1. The temporal locality assumption is then further confirmed by Figure 3.8(b), which shows the CDF distributions of the percentage differences of each average event attribute across days. The figure shows that while the average volume of each stock fluctuates more then the average price, the average volume changes at most 100% across days for 80% of all stocks. Hence, from both Figure 3.8(a) and Figure 3.8(b), we can conclude that temporal locality exists in the stock event trace. This fact allows us to use the past event trace history to predict the current event distribution accurately.

3.4.2.2 Prediction Accuracy

This section presents the *accuracy of the subscriber reliability prediction algorithm* with the stock event traces described in Section 3.4.2.1. In this experiment, we use exponential matching time distribution at each broker and G/G/1 prediction model presented in Section 3.2.3. To observe the effect of imperfect publisher information, we run the simulation in two modes. The first mode is called *perfect information* mode, where the publisher event profile is known in advance (i.e., use the second day event trace to predict the second day subscriber reliability). The second mode is called *historical information* mode, where the publisher event profile is calculated from the past history (i.e., use the first day event trace to predict the second day subscriber reliability). While the perfect information mode is unrealistic, it can be considered as the ideal case to be compared with the realistic historical information mode.

Figure 3.9(a) shows the prediction accuracy of the algorithm in perfect information mode. As seen from the figure, the algorithm can estimate most subscribers' reliability values accurately. The result, however, is slightly worse than the one with synthetic trace due to heterogeneity in publishers' rates in real-world event traces.

Figure 3.9(b) shows the prediction accuracy of the algorithm in historical information mode. The graph shows that the algorithm can still predict most subscribers' reliability values accurately. The CDF distribution of prediction error in Figure 3.9(c) also shows that there is only slight difference between the perfect information mode and the historical information mode. This confirms the fact that the proposed subscriber reliability prediction algorithm can achieve good accuracy with imperfect publisher profile information.

3.4.3 Subscriber Admission Control

We evaluate the heuristic-based admission control algorithms discussed in Section 3.3 in a smaller-scale setting due to time constraint in exhaustively exploring all possible subscriber sets to find the optimal solution. The publish/subscribe system in the setting consists of 4 brokers, 8 publishers, and 10 requested subscribers. The event publishing interval and event processing time are exponentially distributed, resulting in no difference between results from M/M/1 model and G/G/1 model.

Figure 3.10 shows the fraction of subscribers that have their requirements satisfied. As shown from the figure, the publish/subscribe system without admission control performs the worst, since all subscribers are admitted to the system and contend for resources. On the other hand, the proposed heuristic–based algorithms give satisfaction rates that are closed to the optimal subscriber selection, yielding the effectiveness of the algorithm. Each algorithm performs close to each other without clear extinction, although the *low-req-first* heuristic policy performs slightly better than others as the load increases.

3.5 Related Work

3.5.1 Content Uncertainty in Publish/Subscribe Systems

Liu and Jacobsen[53] addressed the uncertainty in terms of imprecise knowledge in subscriptions and events in CBPS systems. By expressing subscriptions and events in the form of fuzzy sets, the work proposes the publish/subscribe systems that allow approximate matching between subscriptions and events with vague attributes. The concept of publication uncertainty in their work can be considered equivalent to the concept of publisher content-lifetime probability distribution in our work. However, their work focuses on the aspect of subscription uncertainty and correctness in event matching while our work focuses on uncertainty coming from underlying networks, event delivery probability and timeliness.

3.5.2 Timing Analysis in Publish/Subscribe Systems

To the best of our knowledge, the first work to propose timing analysis in publish/subscribe systems is by Baldoni et al[55], which formulates the event delivery probability during the subscription phase (i.e., the probability that a newly joining subscriber receives an event while its subscription is not finalized throughout the system.) and during the un-subscription phase (i.e., the probability that a leaving subscriber receives an event published shortly before its un-subscription time). Their concept of persistent event is similar to event lifetime in our model. However, their model focuses on transient-state behavior (i.e., joining phase and leaving phase) while our model focuses on the steady-state behavior of the publish/subscribe systems where event delivery probability is affected by the event's lifetime, link delay, and broker queuing/matching delay. Our model thus is suitable for delay-sensitve content-based publish/subscribe systems as we emphasize the concept of per-event lifetime.

Another work that resembles our work in modeling publish/subscribe system integration and timeliness is the work done by Kounev et al[56]. The work analyzes mean delivery delay of distributed event-based systems with the use of rate calculation and queuing theory. While our work also uses the queuing theory to calculate delivery delay, our work presents the model that abstracts content-based events and subscriptions, and allows fine-grained prediction of reliability and delay. We also propose a heuristic-based admission control algorithm on top of such a model.

Schröter et al also proposes the use of stochastic model to compute the performance of the CBPS systems for the purpose of performance prediction and capacity planning[31, 32]. Their work addresses content uncertainty by assuming each subscriber to belong to one of the traffic classes, and assuming the local incoming traffic rate for each traffic class at each broker is known. Their work discuss several advanced subscription routing algorithms such as identity-based and merging-based routings[26]. Based on the notion of traffic class, the average subscription and notification traffic at each broker is calculated, which is then used to estimate per-broker service delay and end-to-end notification/subscription delays using M/G/1 queuing model. While our work is similar to their works in terms of stochastic analysis, we propose the use of G/G/1 queuing model, which supports more generalized traffic distribution. Another difference is that their work aims to provide coarse-grain, system-level performance prediction while our work aims to provide fine-grain, subscriber-level performance prediction.

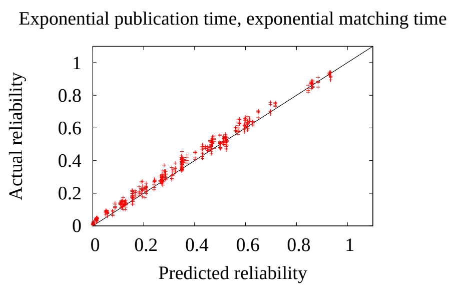
3.5.3 Overload Management in Publish/Subscribe Systems

There have been several works which propose techniques to handle overload in publish/subscribe systems[12, 57]. Guo et al propose a subscriber admission control scheme where each publisher advertises its publishing rate and content pattern. Each publisher's advertisement is propagated along the broker tree and stored at each broker. When a subscriber joins the system by sending its subscription, its subscription is forwarded along the broker tree. Upon receiving the new subscription, each broker calculates the additional resources needed to support the new subscription and then either accepts the subscription or rejects the subscription. Their admission control protocol is similar in concept to our admission control protocol. However, their works does not address delay calculation and timeliness constraints. Our work, on the other hand, addresses fine-grained end-to-end delay calculation, which is essential for delay-sensitive content-based applications.

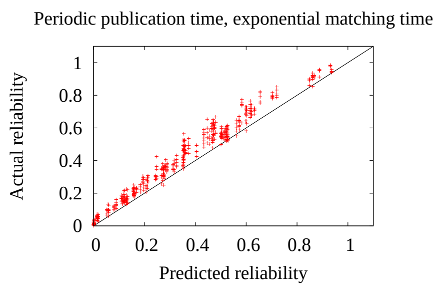
Jerzak et al propose the load shedding scheme for overloaded content-based publish/subscribe systems[57]. In their scheme, each broker continuously monitors its local event processing delay and the each of its links' available bandwidth. When a broker detects a broker/link overload, it starts to drop events. They introduce a pricing model for each event/subscription, and propose an event dropping policy which will maximize the total revenue generated by the system. Their approach detects system overload on-the-fly while our approach detects system overload by off-line model analysis. The two approaches can be combined altogether for higher accuracy and responsiveness. The on-line overload detection technique is also adopted in Arianfar's master thesis[58] to detect congested brokers and re-construct the publish/subscribe tree to avoid congested brokers.

3.6 Discussion

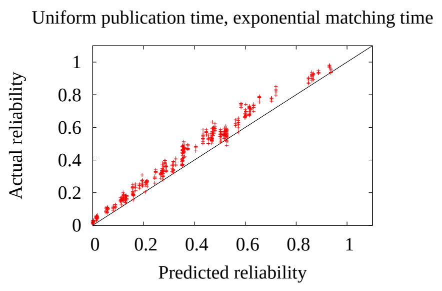
In this chapter, we investigate the possibility of subscriber real-time reliability estimation in content-based publish/subscribe systems, with the emphasis on content dynamism. The finding is that, given the publisher content distribution, it is thus possible to calculate the amount of traffic load between each publisher-subscriber pair and load along the path between them. The use of novel entity content distribution and content-lifetime distribution allow us to estimate the fraction of events delivered at each subscriber on time, which is the definition of subscriber reliability. We then propose the subscriber admission control algorithm to allow only a subset of subscribers that maximally utilize the system while still receive good performance. While the proposed algorithm is limited on a single static publish/subscribe tree, a simple extension can be done to adopt the solution for scenarios with multiple topic trees. The evaluation results have shown that our predictive model can estimate subscriber reliability accurately.



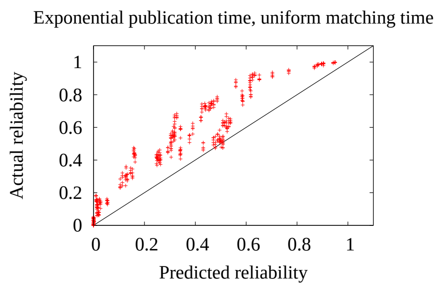
(a) Exponential publication time, exponential matching time



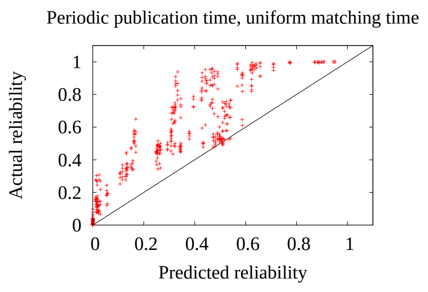
(b) Periodic publication time, exponential matching time



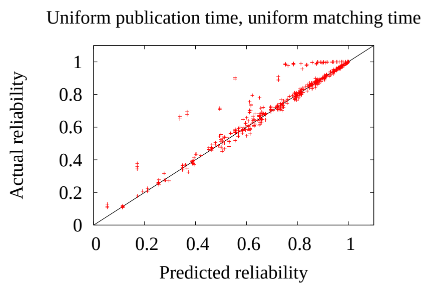
(c) Uniform publication time, exponential matching time



(d) Exponential publication time, uniform matching time

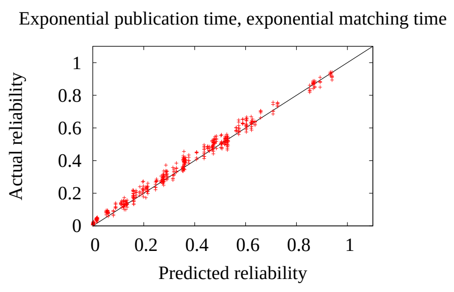


(e) Periodic publication time, uniform matching time

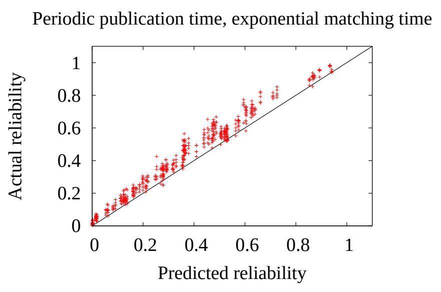


(f) Uniform publication time, uniform matching time

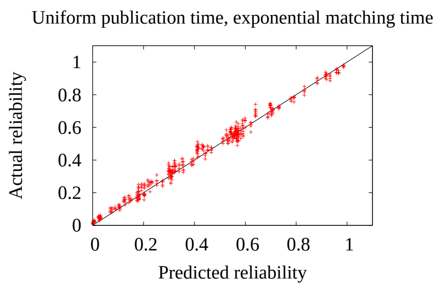
Figure 3.5: M/M/1 model predicted subscriber reliability compared to actual reliability under different event traffic patterns



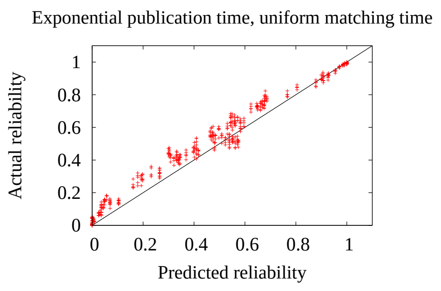
(a) Exponential publication time, exponential matching time



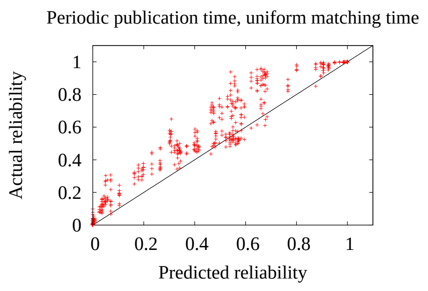
(b) Periodic publication time, exponential matching time



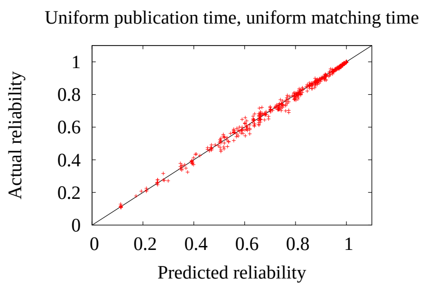
(c) Uniform publication time, exponential matching time



(d) Exponential publication time, uniform matching time

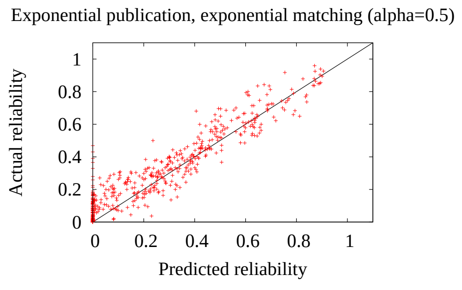


(e) Periodic publication time, uniform matching time

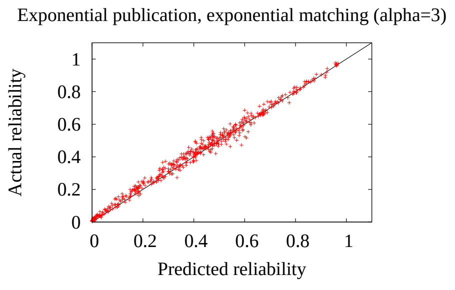


(f) Uniform publication time, uniform matching time

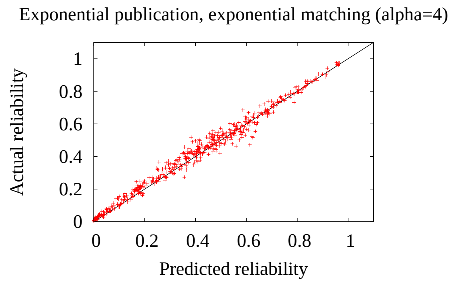
Figure 3.6: G/G/1 model predicted subscriber reliability compared to actual reliability under different event traffic patterns



(a) α=0.5

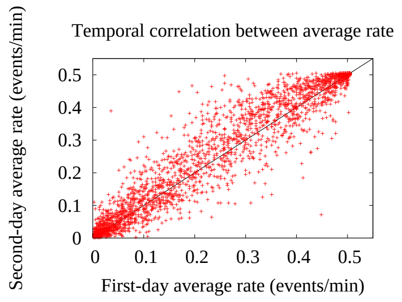


(b) α=3

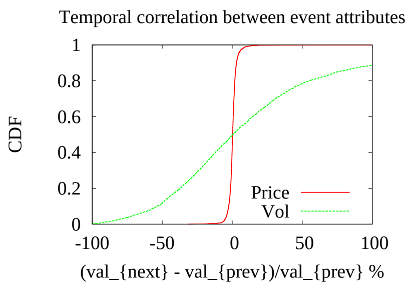


(c) α=4

Figure 3.7: Predicted reliability compared to actual reliability with inaccurate content distribution information

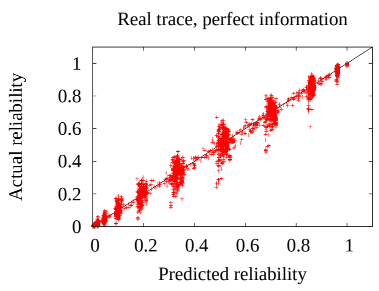


(a) Event average rate

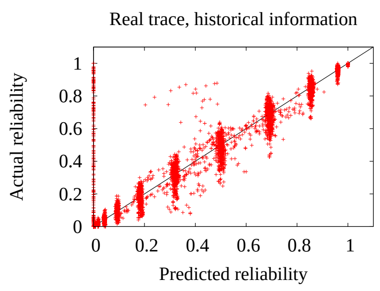


(b) Event attributes

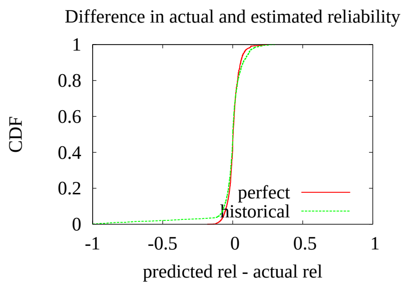
Figure 3.8: Temporal locality between first-day event trace and second-day event trace



(a) Perfect Publisher Information



(b) Historical Publisher Information



(c) Prediction Error

Figure 3.9: Predicted reliability compared to actual reliability with real-world stock traces

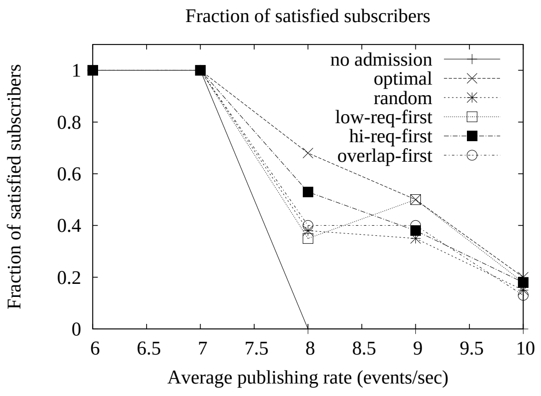


Figure 3.10: Fraction of satisfied subscribers with different admission control algorithms

CHAPTER 4

RELIABILITY AND TIMELINESS ANALYSIS WITH MOBILITY DYNAMISM

4.1 Introduction

Over the past decade, 802.11 wireless networks (i.e., Wi-Fi networks) have reached popularity and become the common standard for data-plane mobile device communication. With rapid development in terms of connection quality and power consumption, Wi-Fi technology has bridged the gap between computer networks and cellular networks, allowing its users to enjoy Internet access even while they are on the move. Moreover, with the advancements and economy of scale in core network infrastructure, the cost for Internet connectivity has become cheaper as well. Likewise, in the context of publish/subscribe systems, it has now been common for users to receive real-time events such as news or traffic report from their mobile devices. Hence, to reflect the real-world scenarios of mobile subscribers, it is necessary for the analytical model to incorporate user mobility into the analysis as well.

To incorporate mobile subscriber into the publish/subscribe model, the relationship between a subscriber and its home broker must be re-defined. So far, we assumed each subscriber to be static and connected to only one home broker during its lifetime in the system. Now that mobile subscriber is considered, each subscriber's home broker can change over time, depending on the physical locations of subscribers and brokers. Thus, we envision the *two-tiered* hierarchical content-based publish/subscribe architecture, where each subscriber can be either static or mobile. Each broker also has the capability to communicate with mobile subscribers via wireless channels. Mobile subscribers can move around and interact with any broker within its communication range. However, we do not consider ad hoc communication among subscribers. Figure 4.1 gives an illustration of such two-tiered hierarchical content-based publish/subscribe architecture.

While mobile publish/subscribe systems provide flexibility and convenience to subscribers, it is even harder to predict subscriber reliability due to uncertainty in subscriber mobility over time (i.e.,

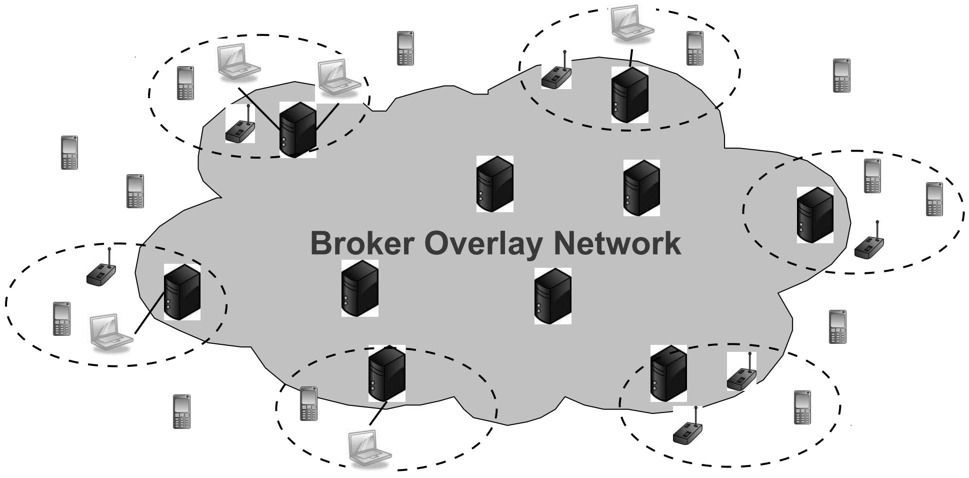


Figure 4.1: An illustrative scenario of a publish/subscribe system with mobile subscribers

mobility dynamism). This is because a subscriber can move out of any broker's communication range and fail to receive any events during that outage period. To cope with such problem, many mobility management schemes have been proposed to allow brokers to buffer the events on behalf of a subscriber during the subscriber's disconnection period, and transmit those buffered events to the subscriber once it reconnects to the publish/subscribe network. However, buffered real-time events can also be expired during the buffering period. The effect of such mobility dynamism to the subscriber reliability must be quantified for the purpose of performance assessment and optimization to maintain service satisfaction to the users.

This chapter proposes an analytical model to predict mobile subscriber reliability with respect to its mobility pattern and broker network deployment. Specifically, each mobile subscriber's reliability is quantified as a function of four parameters : *event lifetime* (the period an event is valid after its publishing time), protocol *handoff latency* (the time it takes for the system to detect the change of subscriber's location), *contact duration* (the period of time the subscriber is within brokers' communication ranges), and *outage duration* (the period of time the subscriber is not within any broker's communication range). Based on such four parameters, subscriber reliability can be estimated with acceptable accuracy. Contact/outage duration parameters, however, must be further calculated from broker physical topology and user mobility. Hence, another part of this chapter focuses on estimating contact/outage duration distribution based on various types of mobility models. Finally, simulation results with various broker topologies and subscriber mobility models yield the effectiveness of the proposed analytical model.

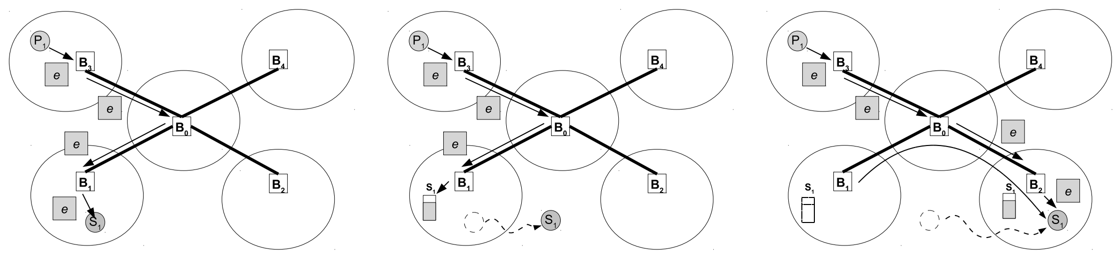
This chapter is organized as follows. Section 4.2 first presents the existing subscriber mobility management protocols along with how to model such protocols in the analysis. Section 4.3 then formally defines the problem of mobile subscriber reliability estimation problem and proposes the analytical framework to solve such problem. Section 4.4 focuses on the calculation of contact/outage duration distributions, which are important parameters in determining subscriber reliability in Section 4.3. Section 4.5 shows the validation results of the proposed analytical model via simulations. Section 4.6 presents related works in mobility handoff analysis. Finally, Section 4.7 discusses the future research direction of mobile dynamism and publish/subscribe systems.

4.2 Publish/Subscribe Mobility Management

In this section, we investigate two families of publish/subscribe mobility management protocols that exist in the literature[59, 60, 61, 24]. The two families share one similar approach to handle disconnected subscribers, which is to have at least one *proxy broker* to store incoming events on behalf of each disconnected subscriber and transfer the buffered events to the subscriber once the subscriber reconnects. The two schemes differ in the way the proxy broker set is chosen for each disconnected subscriber. The first mobility management scheme is called *reactive handoff* and the second scheme is called *proactive prefetching.*

4.2.1 Reactive Handoff

Reactive handoff scheme is a classical approach to buffer events for a subscriber during its disconnection period[59, 60, 61]. In the reactive handoff scheme, the last broker serving a subscriber detects the subscriber's disconnection either via explicit *MoveOut* message from the subscriber or via connection timeout. Once subscriber disconnection is detected, the last serving broker continues to receive events on behalf of the disconnected subscriber. When the subscriber reconnects at a new broker, the new broker knows of the subscriber's previous broker either by explicit *MoveIn* message or by querying the broker network. The new broker then resubscribes on behalf of the subscriber and, at the same time, notify the subscriber's previous broker about the subscriber's reconnection. Once notified, the old broker un-subscribes on behalf of the subscriber and, at the same time, transfers all buffered events to the new broker. The new broker then forwards all the events received from the old broker to the subscriber. Figure 4.2 shows an example of the reactive handoff protocol.



(a) Normal Operation (b) Disconnection (c) Reconnection

Figure 4.2: Example of reactive handoff mobility management

The advantage of the reactive handoff approach is simplicity in implementation. Most traditional publish/subscribe systems use this approach to enhance their supports of mobile clients[59, 62, 63]. However, the drawback of this approach is high delay when the new broker contacts and receives the buffered events from the old broker. The effect of such high delay to the users depend on the application delay requirement.

4.2.2 Proactive Prefetching

Recently, a new publish/subscribe mobility scheme called *proactive prefetching* has been proposed[24]. The proactive prefetching scheme exploits the spatial locality nature of user mobility. That is, a subscriber that is disconnected from one broker usually reconnects at another broker near the old broker. Hence, in the proactive prefetching scheme, each broker maintains a data structure called *neighbor set*, which contains a set of brokers to which a subscriber potentially reconnects next after its disconnection from the current broker. When a subscriber subscribes to a broker, the current broker also forwards the *passive subscription* to its neighbor set. A passive subscription at each neighboring brokers is not activated until the subscriber is disconnected from the current broker. Once the current broker detects the subscriber's disconnection, it sends the signal to all neighboring brokers to activate the previously passive subscription and start to buffer incoming events on behalf of the subscriber. If a subscriber reconnects at one of the neighboring brokers, that broker becomes the subscriber's new home broker. The new home broker then transfers the locally buffered events to the subscriber and, at the same time, notifies the old home broker and other neighboring brokers about the subscriber's reconnection. The old home broker and other neighboring brokers then discard the buffered events and deactivate the subscriber's subscription. If, in any circumstance, the disconnected subscriber reconnects at another broker which is not the old home broker's neighbor, the new home broker will use the reactive handoff scheme to retrieve the buffered events from the old home broker. Figure 4.3 shows an example of the proactive prefetching protocol.

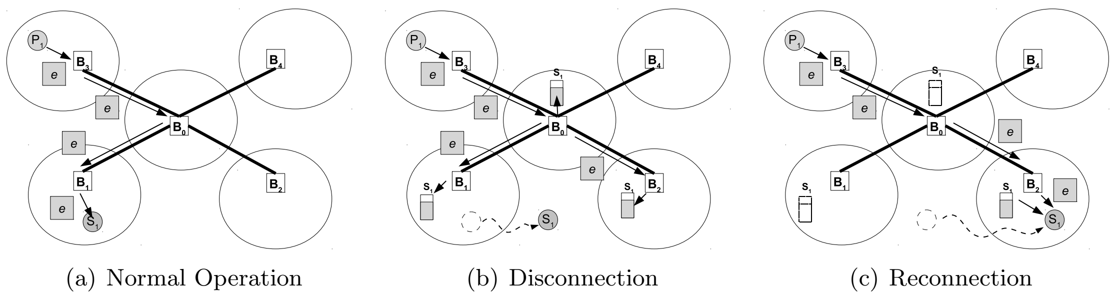


Figure 4.3: Example of proactive prefetching mobility management

The performance of the proactive prefetching scheme heavily depends on the accuracy in determining the neighbor set of each broker[24]. If the subscriber mobility is deterministic (e.g. street mobility) and each broker's neighbor set is small, then the proactive scheme incurs less handoff delay and comparable handoff overhead when compared to the reactive handoff. However, if the subscriber mobility is not deterministic (e.g. random waypoint mobility) and each broker's neighbor set is large, the proactive scheme could incur higher delay and overhead.

In our proposed analytical model, we model the two different mobility management protocols in the form of *handoff latency*, which is the delay it takes for the new broker to detects the subscriber's reconnection and delivers the buffered events to the subscriber. The detail of the handoff latency will be presented in Section 4.3.2.2.

In the next section, we will formulate the mobile subscriber real-time reliability estimation problem. We then propose the analytical model to solve such problem.

4.3 Mobile Subscriber Real-time Reliability

4.3.1 Formulation

As mentioned, this chapter studies the effect of subscriber mobility to subscriber real-time reliability. Hence, we use the following assumptions in this chapter to mask the effects from other factors.

Reliable broker network : We assume that the broker overlay network does not have failure.

Any event from publisher will be delivered to the designated home broker within a very short period (several milliseconds). Compared to user mobility, which is in the scale of several seconds or minutes, we can consider transmission and processing delay within the broker network to be negligible.

No publisher mobility : While it is possible for a publisher to be mobile and publishing its events to its local broker via wireless communication, we do not consider scenarios where publishers move across different brokers over time. It is usually typical in the real-world scenario for a publisher to have no or very slow mobility, since its change of location may affect the semantic of its published events. The problem of publisher mobility in publish/subscribe systems has been investigated but without quantitative analysis[64]. Hence, it can be considered the future research direction of this thesis.

No broker overload : In Chapter 3, we explored the effect of traffic load to the subscriber reliability. As any broker is overloaded, the broker's average queuing delay will become unbounded. We assume such situation does not occur. Such assumption can be achieved by leveraging the subscriber admission control algorithm proposed in Section 3.3.

Physical model : As mentioned, the performance of publish/subscribe systems for mobile subscribers now also depend on the physical location of both brokers and subscribers. Hence, it is necessary to incorporate physical location into the broker model and the publisher model. Therefore, we redefine a broker $b \in \mathbb{B}$ as a tuple

$$b = (id_b, M_b(t), c_b)$$

, where id_b and $M_b(t)$ are the broker b's identifier and event processing delay defined in Section 2.2.3, respectively, and $c_b = (x_b, y_b)$ is broker b's 2-dimensional, geographical location. At the same time, a subscriber $s \in \mathbb{S}$ is re-defined as a tuple

$$s = (id_s, f_s, W_s, \overline{W}_s)$$

, where id_s and f_s are subscriber s's identifier and content filter defined in Section 2.2.4 respectively. The additional variables W_s and $\overline{W}_s$ are subscriber s's contact and outage duration random variables, which will be explained in Section 4.3.2.1.

With such assumptions and revised models, we define the mobile subscriber real-time reliability estimation problem as follows.

Mobile Subscriber Real-time Reliability Estimation Problem: Given a reliable publish/subscribe overlay network $\mathbb{G} = (\mathbb{N}, \mathbb{L})$ where $\mathbb{N} = \mathbb{B} \cup \mathbb{P} \cup \mathbb{S}$, find the estimated value of r_s, denoted by r'_s, for each mobile subscriber $s \in \mathbb{S}$.

4.3.2 Analytical Framework

As mentioned in Section 4.1, a mobile subscriber's reliability can be modeled as a function of four parameters, which are contact duration distribution, outage duration distribution, protocol handoff latency, and event lifetime. This section describes each parameters in detail.

Symbol	Definition
$c = (x, y)$	An arbitrary 2-dimensional geographical location
$\mathbb{C}$	The set of all possible locations of the system
W	Contact duration (distribution)
$\overline{W}$	Outage duration (distribution)
H	Handoff latency (distribution)
$H_{\max}$	Maximum handoff latency (constant)
D	Lifetime of all events (constant)
β	A basic trajectory
k	A generic trajectory
$\Omega : \mathbb{C} \to \{0, 1\}$	Broker connectivity function
ϕ	An arbitrary phase
$\|\phi\|$	phase ϕ's duration
$\omega(\phi)$	phase ϕ's type (0=outage, 1=contact)
Φ	An interleaving sequence of contact phases and outage phases
Φ_W	The subset of Φ that contains only contact phases
$\Phi_{\overline{W}}$	The subset of Φ that contains only outage phases
$\dot{G} = (\dot{V}, \dot{E})$	A semi-Markov mobility graph

Table 4.1: Mobility dynamism analysis variables' notation

4.3.2.1 Contact/Outage Duration

In a nutshell, many mobile connectivity metrics can be estimated from two basic temporal metrics : *contact phase* duration and *outage phase* duration. A *contact phase* is a consecutive period of time a mobile device is connected to at least one wireless access point (i.e., cell residence time or association time). On the other hand, an *outage phase* is a period of time a mobile device is disconnected from wireless infrastructure. Both metrics are complementary to each other as a mobile device will experience an interleaving sequence of contact phase and outage phase throughout its traveling time. A more protocol-dependent performance metric can then be calculated on top of such two basic metrics. For example, given its Wi-Fi contact/outage duration distributions, a mobile device can calculate the period of time (and hence the air time cost) it needs to resort to the cellular network due to its disconnection from Wi-Fi networks.

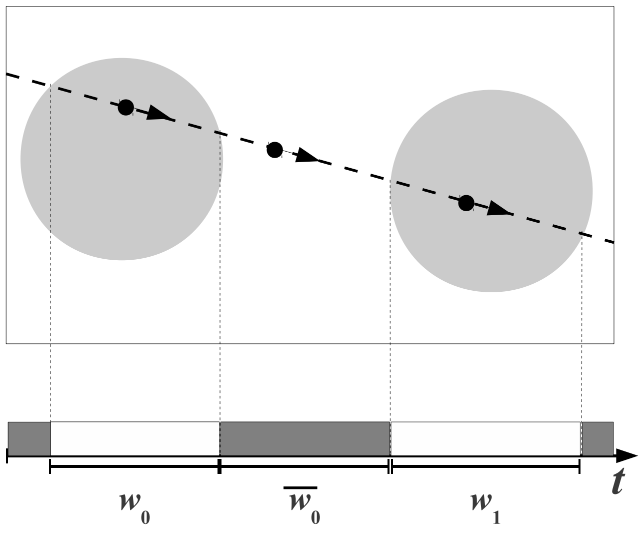


Figure 4.4: Timing diagram showing a subscriber's mobility and contact/outage phases

In the context of mobile publish/subscribe systems, a subscriber s's contact duration distribution, denoted by W_s, is a random variable defining the time period when subscriber s is within at least one broker's communication range. A subscriber s's outage duration distribution, denoted by $\overline{W}_s$, is a random variable defining the time period when subscriber s is not within any broker's communication range. For example, Figure 4.4 shows the timing diagram of a moving subscriber across two brokers' ranges. The white timing regions are contact phases and the gray timing regions are outage phases. The distributions of both W_s and $\overline{W}_s$ can be obtained from association history between subscriber s and each broker. However, if such history is not available, the two distributions can still be estimated from subscriber s's mobility function and broker set $\mathbb{B}$ geographical deployment. The estimation of contact/outage duration distributions will be described in Section 4.4.

4.3.2.2 Handoff Latency

We abstract the effect of mobility management protocols, which are described in Section 4.2, in the form of *handoff latency*, denoted by H. The handoff latency represents the duration of time it takes for any broker to detects a subscriber within its range, plus the time it takes for the broker network to transfer all buffered events to the reconnecting subscriber. The handoff latency distribution can be calculated from the mobility management protocol and broker's beacon period (e.g., wireless access point advertisement period). In usual cases, handoff latency H is uniformly distributed between 0 and $H_{\max}$, where $H_{\max}$ is the beacon period of each broker.

4.3.2.3 Event Lifetime

In Section 2, we modelled event lifetime as a per-publisher distribution function. In this Chapter, we assume all events have a single lifetime value, denoted by D, for simplicity of analysis. However, our analytical model can be simply modified for the cases where event lifetime is a distribution as well.

4.3.3 Mobile Reliability Estimation Algorithm

This section presents the algorithm to estimate subscriber reliability for any mobile subscriber $s \in \mathbb{S}$. We first present the estimation algorithm for mobile publish/subscribe systems without any mobility management protocol presented in Section 4.2. We then present the estimation algorithm for mobile publish/subscribe systems with mobility management protocol.

4.3.3.1 Subscriber Reliability without Buffering

This section presents the mobile subscriber reliability estimation algorithm for publish/subscribe systems without mobility management for the purpose of performance comparison. Since there is no buffering and we assume that an event can be published at any arbitrary time, a mobile subscriber s's reliability, denoted by r_s, can be estimated as the fraction of time the subscriber s can communicate with at least one broker, which defined as *live period*. Such live period is indeed equal to the contact period excluding the handoff delay, which is the time the broker takes to register the subscriber to the system. For example, Figure 4.5 shows the timing diagram of a

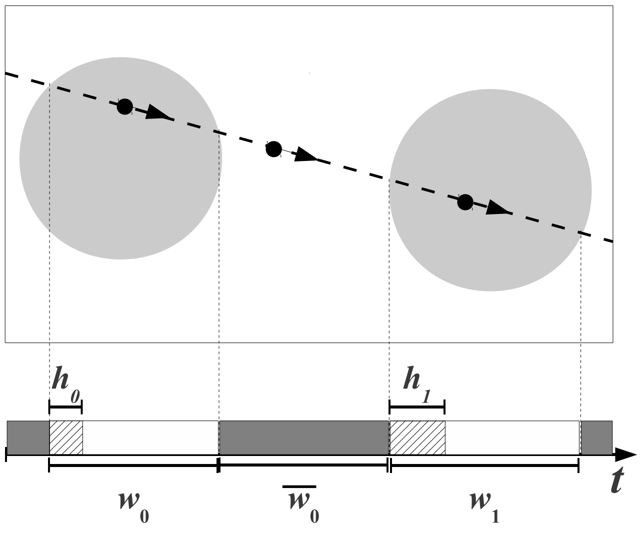


Figure 4.5: Timing diagram showing a subscriber's mobility and live period without buffering

subscriber moving across two brokers' ranges. The white period represents the live period during which a published event will be delivered to the subscriber.

Hence, given subscriber s contact/outage duration distributions $f_{W_s}(t)$ and $f_{\overline{W}_s}(t)$, the generic event lifetime D, and protocol handoff latency distribution $f_H(t)$, we can estimate the mobile subscriber reliability r_s as follows.

$$\begin{aligned} r'_s &= \frac{\text{mean contact duration after associating with broker}}{\text{mean time between two successive brokers}} \\ &= \frac{E[\max(W_s - H, 0)]}{E[W_s] + E[\overline{W}_s]} \\ &= \frac{\int_{t=0}^{\infty}\int_{t'=0}^{\infty} f_{W_s}(t).f_H(t')\max(t-t',0)dtdt'}{\int_{t=0}^{\infty} t.f_{W_s}(t)dt + \int_{t=0}^{\infty} t.f_{\overline{W}_s}(t)dt} \end{aligned} \tag{4.1}$$

If we assume handoff delay H is uniformly distributed between 0 and $H_{\max}(H_{\max} << E[\overline{W}_s])$, then Equation (4.1) becomes as follows.

$$
\begin{aligned}
r'_s &= \frac{\int_{t=0}^{\infty}\int_{t'=0}^{H_{\max}} \frac{f_{W_s}(t)}{H_{\max}}.\max(t-t',0)dtdt'}{\int_{t=0}^{\infty} t.f_{W_s}(t)dt + \int_{t=0}^{\infty} t.f_{\overline{W}_s}(t)dt} \\
&= \frac{\int_{t=0}^{H_{\max}} f_{W_s}(t).\int_{t'=0}^{t} \frac{t-t'}{H_{\max}}dt'dt + \int_{t=H_{\max}}^{\infty} f_{W_s}(t).\int_{t'=0}^{H_{\max}} \frac{t-t'}{H_{\max}}dt'dt}{\int_{t=0}^{\infty} t.f_{W_s}(t)dt + \int_{t=0}^{\infty} t.f_{\overline{W}_s}(t)dt} \\
&= \frac{\int_{t=0}^{H_{\max}} f_{W_s}(t).\frac{t^2}{2H_{\max}}dt + \int_{t=H_{\max}}^{\infty} f_{W_s}(t).(t-\frac{H_{\max}}{2})dt}{\int_{t=0}^{\infty} t.f_{W_s}(t)dt + \int_{t=0}^{\infty} t.f_{\overline{W}_s}(t)dt} \qquad (4.2)
\end{aligned}
$$

4.3.3.2 Subscriber Reliability with Buffering

If a publish/subscribe system has support for mobile subscribers by buffering events on behalf of each disconnected subscriber (i.e., mobility-aware publish/subscribe), then an event published during a subscriber's disconnection period will be buffered in the broker network and transferred to the subscriber once it reconnects to a broker. Thus, given that every event has the same lifetime D, an event published during a subscriber's disconnection period can be successfully delivered to the subscriber if and only if it is published within the period D before the subscriber's next reconnection point. The subscriber s's live period for a mobility-aware publish/subscribe system is then equal to its live period without buffering (i.e., Section 4.3.3.1) prefixed by the event lifetime duration D. Figure 4.6 shows a timing diagram of mobile subscriber in a mobility-aware publish/subscribe system. The white period shows the live period of the subscriber.

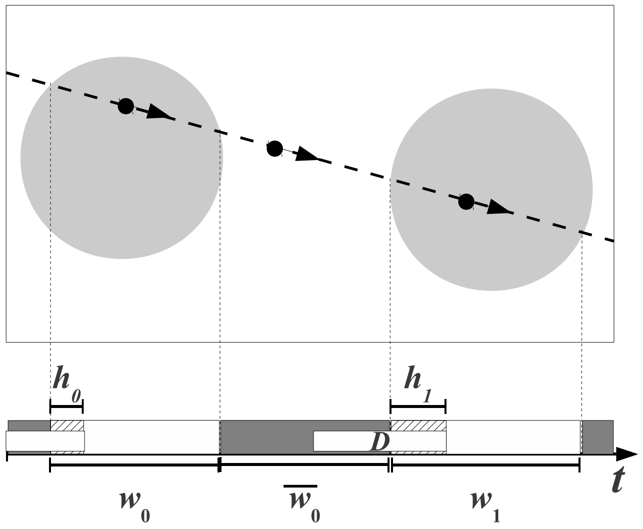


Figure 4.6: Timing diagram showing a subscriber's mobility and live period with buffering

Hence, we can calculate subscriber s's real-time reliability r_s in a mobility-aware publish/subscribe system as follows.

$$\begin{aligned} r'_s &= \frac{\text{mean live period}}{\text{mean time between two successive APs}} \\ &= \frac{E[\min(\overline{W}_s + H_s, D) + \max(W_s - H_s, 0)]}{E[W_s] + E[\overline{W}_s]} \\ &= \frac{\int_{t=0}^{\infty}\int_{t'=0}^{\infty} f_{\overline{W}}(t).f_H(t')\min(t'+t, D)dt'dt}{\int_{t=0}^{\infty} t.f_{W_s}(t)dt + \int_{t=0}^{\infty} t.f_{\overline{W}_s}(t)dt} + \\ &\quad \frac{\int_{t=0}^{\infty}\int_{t'=0}^{\infty} f_W(t).f_H(t')\max(t-t', 0)dt'dt}{\int_{t=0}^{\infty} t.f_{W_s}(t)dt + \int_{t=0}^{\infty} t.f_{\overline{W}_s}(t)dt} \end{aligned} \tag{4.3}$$

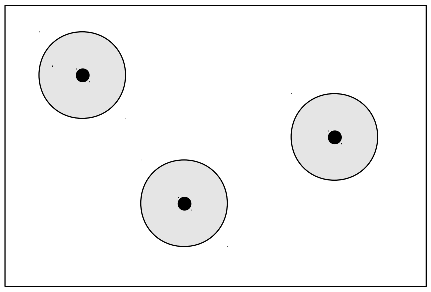
4.4 Estimating Contact/Outage Duration Distributions

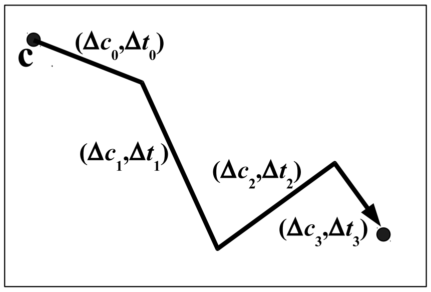
So far, we can calculate mobile subscriber reliability, given the contact/outage duration distributions, protocol handoff delay, and event lifetime. While protocol handoff delay and event lifetime can be estimated from the mobility management protocol and event specification, it is sometimes not trivial to obtain contact/outage duration distributions as the contact/outage duration history of the subscriber may not be available. This section shows that, if the subscriber mobility pattern and broker locations are known, it is possible to estimate contact/outage duration distributions.

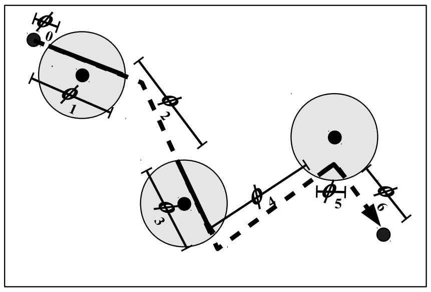
However, mobility pattern is a very vague term as there are many mobility models existing in the literature. In this section, we first present a mobility abstraction model that covers several classes of existing mobility models. The mobility abstraction starts from a very simple, deterministic mobility pattern to more complex, probabilistic mobility patterns. We also present how to calculate contact/outage duration distributions from each type of mobility pattern.

4.4.1 Broker Map Model

First, we must define each broker's physical location and communication range. Let a geographical region $\mathbb{C}$ be the set of all possible geographical coordinates $c = (x, y)$ that a mobile subscriber can reach. Without loss of generality, we assume $\mathbb{C}$ to be a convex geographical region such as a circle or a rectangle. We then define a connectivity function Ω over a geographical region $\mathbb{C}$ as follows.


(a) Map connectivity function $\Omega()$. The gray regions denote the areas with Wi-Fi coverage


c
$(\Delta c_0,\Delta t_0)$
$(\Delta c_1,\Delta t_1)$
$(\Delta c_2,\Delta t_2)$
$(\Delta c_3,\Delta t_3)$
(b) Trajectory k



(c) $\Phi^{c,k,\Omega}$ as an interleaving sequence of contact phases (solid lines) and outage phases (dash lines)

Figure 4.7: Example of a subscriber's trajectory in a map with three APs

Definition A connectivity function $\Omega : \mathbb{C} \rightarrow \{0, 1\}$ is defined by

$$\Omega(c) = \begin{cases} 1 & \text{if } c = (x, y) \text{ is within at least one broker's communication range} \\ 0 & \text{otherwise} \end{cases}$$

That is, a connectivity function $\Omega(c)$ returns 1 if the point c is within the transmission range of at least one broker, and returns zero otherwise. Without loss of generality, we assume a simple, circular broker coverage where a point c is within a broker b's communication range if the Euclidean distance between point c and the broker b's position c_b is less than some constant radius. However, a more complex Wi-Fi coverage that considers more accurate signal and noise propagation models can be applied to function $\Omega(c)$ as well. Figure 4.7(b) shows an example of a connectivity function defined over a rectangular region with brokers.

4.4.2 Basic Mobility Model

4.4.2.1 Trajectory

Here, we define a subscriber's *trajectory* as the mobility primitive which represents the subscriber's geographical movement over a period of time. We will show later in Section 4.4.2 and 4.4.3.1 that more complex mobility models can be constructed on top of the trajectory model.

The simplest form of a trajectory called *basic trajectory* is defined as follows.

Definition A basic trajectory $\beta = (\Delta c, \Delta t)$, represents a straight, constant-speed movement that results in a subscriber's position change equal to $\Delta c = (\Delta x, \Delta y)$ within the time $\Delta t > 0$. That is, a subscriber with its original position $c = (x, y) \in \mathbb{C}$ at time t taking a basic trajectory $\beta = (\Delta c, \Delta t)$ will be arriving at the new position $c' = c + \Delta c = (x + \Delta x, y + \Delta y)$ at time $t' = t + \Delta t$. We use the notations $c \xrightarrow{\beta} c'$ and $t \overset{\beta}{\Longrightarrow} t'$ to denote the spatial and temporal transitions caused by β respectively.

Once the basic trajectory is defined, we can define a generic trajectory as a sequence of basic trajectories as follows.

Definition A generic trajectory k is defined as a sequence containing at least one basic trajectories $[\beta_0, \beta_1, .., \beta_{|k|-1}]$ where $\beta_i = (\Delta c_i, \Delta t_i)$ for all $0 \leq i < |k|$. A subscriber with an original position $c = (x, y) \in \mathbb{C}$ at time t taking a generic trajectory $k = (\beta_0, \beta_1, .., \beta_{|k|-1})$ will be arriving at the position $c' = c + \sum_{i=0}^{|k|-1} \Delta c_i$ at time $t' = t + \sum_{i=0}^{|k|-1} \Delta t_i$. Again, we use the notations $c \xrightarrow{k} c'$ and $t \overset{k}{\Longrightarrow} t'$ to denote the spatial and temporal transitions caused by k respectively.

With such definition, we can define trajectory of any shape (including curve) or speed distribution by breaking the trajectory into several basic trajectories, each with straight movement and constant speed. Figure 4.7(b) depicts one example of a generic trajectory defined by a sequence of four basic trajectories.

4.4.2.2 Contact/Outage Phases

We define a *contact phase* as a period of time a mobile client remains connected to at least one Wi-Fi access point. Conversely, we define an *outage phase* as a period of time a mobile client remains disconnected from any Wi-Fi access point.

Definition A *phase* ϕ is either a contact phase or an outage phase. We define phase type $\omega(\phi)$ to be equal to 0 if ϕ is an outage phase, and equal to 1 if ϕ is a contact phase. We also define phase length $|\phi|$ to be the time duration of the phase ϕ.

It can be easily seen that starting from an initial position c, a subscriber's trajectory k over a connectivity map Ω will produce an interleaving sequence of contact phases and outage phases (e.g., Figure 4.7(c)).

Definition A *phase sequence* $\Phi^{c,k,\Omega}$ is an interleaving sequence of phases $[\phi_0, \phi_1, .., \phi_{|\Phi^{c,k,\Omega}|-1}]$ where $\omega(\phi_i) \neq \omega(\phi_{i+1})$ for all $0 \leq i < |\Phi^{c,k,\Omega}|$ that results from a subscriber starting from the initial location c and moving with the trajectory k over the connectivity map Ω. We also define the contact set $\Phi_W^{c,k,\Omega} \subseteq \Phi^{c,k,\Omega}$ and the outage set $\Phi_{\overline{W}}^{c,k,\Omega} \subseteq \Phi^{c,k,\Omega}$ to be the set containing contact phases and outage phases from $\Phi^{c,k,\Omega}$ respectively.

For example, with a map connectivity function Ω from Figure 4.7(a), a subscriber's initial position c and generic trajectory k from Figure 4.7(b), the phase sequence $\Phi^{c,k,\Omega} = [\phi_0,\phi_1,\phi_2,\phi_3,\phi_4,\phi_5,\phi_6]$, the contact set $\Phi_W^{c,k,\Omega} = \{\phi_1, \phi_3, \phi_5\}$, and the outage set $\Phi_{\overline{W}}^{c,k,\Omega} = \{\phi_0, \phi_2, \phi_4, \phi_6\}$ as shown in Figure 4.7(c).

4.4.2.3 Deterministic Mobility

In the simplest form, a subscriber s's mobility can be described as its initial position c_s and a single generic trajectory k_s. Such mobility model represents the scenario where a subscriber's movement is deterministic. In such case, the contact/outage duration estimation problem can be defined as follows.

Problem 1 *Given subscriber s's mobility as (c_s, k_s) and a connectivity map Ω, estimate the corresponding contact duration distribution, denoted by $f_W^{c_s,k_s,\Omega}(t)$, and outage duration distribution, denoted by $f_{\overline{W}}^{c_s,k_s,\Omega}(t)$.*

Problem 1 can be solved trivially by counting the occurrences of contact/outage phases with specific duration t in the contact/outage set defined in Section 4.4.2.2 as follows.

$$\begin{aligned} f_W^{c_s,k_s,\Omega}(t) &= \text{P}[s\text{'s contact duration } = t] \\ &= \frac{\text{size of } s\text{'s contact subset with duration } t}{\text{size of } s\text{'s contact set}} \\ &= \frac{|\{\phi \in \Phi_W^{c_s,k_s,\Omega} : |\phi| = t\}|}{|\Phi_W^{c_s,k_s,\Omega}|} \end{aligned} \tag{4.4}$$

Likewise, the corresponding outage duration distribution, denoted by $f_{\overline{W}}^{c_s,k_s,\Omega}(t)$ can be calculated as

$$f_{\overline{W}}^{c_s,k_s,\Omega}(t) = \frac{|\{\phi \in \Phi_{\overline{W}}^{c_s,k_s,\Omega} : |\phi| = t\}|}{|\Phi_{\overline{W}}^{c_s,k_s,\Omega}|} \tag{4.5}$$

4.4.2.4 Probabilistic Mobility

In many scenarios, a subscriber's movement is not known deterministically. Rather, there can be several possible trajectories and initial positions for a subscriber. Such uncertainty cannot be described by the model in Section 4.4.2.3. Instead, we can model a subscriber s's mobility as a probabilistic distribution function q_s as follows.

Definition A mobile client s's mobility model is represented by a coordinate-trajectory joint distribution function $q_s : \mathbb{C} \times \mathbb{K} \to [0,1]$. Where $\mathbb{C}$ and $\mathbb{K}$ are all possible valid points and trajectories respectively. That is,

$$\sum_{(c,k)\in\mathbb{C}\times\mathbb{K}} q_s(c,k) = 1$$

We also define subscriber s's *mobility space* $Q_s \subseteq \mathbb{C} \times \mathbb{K}$ as the set containing all possible (c,k) pairs a subscriber s can take (e.g., $Q_s = \{(c,k) \in \mathbb{C} \times \mathbb{K} : q_s(c,k) > 0\}$).

In other words, the coordinate-trajectory joint distribution q_s describes how likely subscriber k will move from a starting point with a trajectory. Hence, the contact/outage duration estimation problem can be redefined as follows.

Problem 2 *Given subscriber s's mobility as a joint distribution function $q_s(c,k)$ and a connectivity map Ω, estimate the corresponding contact duration distribution $f_W^{q_s,\Omega}(t)$, and outage duration distribution $f_{\overline{W}}^{q_s,\Omega}(t)$.*

Again, Problem 2 can be solved by considering the mobility distribution function q_s as a mixture of different trajectories with different weights. The probability of s's contact phase with duration t, denoted by $f_W^{q_s,\Omega}(t)$, can then be estimated by calculating the expected number of contact phases with duration t from all subscriber s's possible initial positions c and trajectories k. Specifically,

$$\begin{aligned} f_W^{q_s,\Omega}(t) &= \mathrm{P}[s\text{'s contact duration } = t] \\ &= \frac{\mathrm{E}[\text{size of } s\text{'s contact subset with duration } t]}{\mathrm{E}[\text{size of } s\text{'s contact set}]} \\ &= \frac{\sum_{(c,k)\in Q_s} q_s(c,k).|\{\phi \in \Phi_W^{c,k,\Omega} : |\phi| = t\}|}{\sum_{(c,k)\in Q_s} q_s(c,k).|\Phi_W^{c,k,\Omega}|} \end{aligned} \tag{4.6}$$

where $Q_s = \{(c,k) \in \mathbb{C} \times \mathbb{K} : q_s(c,k) > 0\}$

For simplicity, we define $|W^{q_s,\Omega}(t)|$ and $|W^{q_s,\Omega}|$ as the expected size of subscriber s's contact subset with duration t and the expected size of s's contact set (i.e., the upper part and the lower part of Equation (4.6)) respectively.

Similarly, subscriber s's outage duration distribution $f_{\overline{W}}^{q_s,\Omega}(t)$ can be estimated as

$$f_{\overline{W}}^{q_s,\Omega}(t) = \frac{|\overline{W}^{q_s,\Omega}(t)|}{|\overline{W}^{q_s,\Omega}|} \tag{4.7}$$

where

$$|\overline{W}^{q_s,\Omega}(t)| = \sum_{(c,k)\in Q_s} q_s(c,k).|\{\phi \in \Phi_{\overline{W}}^{c,k,\Omega} : |\phi| = t\}|$$

and

$$|\overline{W}^{q_s,\Omega}| = \sum_{(c,k)\in Q_s} q_s(c,k).|\Phi_{\overline{W}}^{c,k,\Omega}|$$

.

It can be seen that Equation (4.6) and (4.7) are the generalized Equation (4.4) and (4.5) respectively. Hence, Equation (4.6) and (4.7) can be used to estimate the contact/outage duration distributions of a subscriber s for any mobility model as long as all of its potential starting points and trajectories can be enumerated (e.g., as long as s's mobility space Q_s is finite).

However, there are scenarios where a subscriber s has infinitely possible numbers of potential starting points or trajectories (e.g., Q_s is infinite). Such scenarios include long-term mobility (subscribers moving for infinitely long or very long time) and recurrent mobility (subscribers moving back to previous states with some probability). In such cases, a subscriber s's corresponding contact/outage duration distributions cannot be simply estimated by previous models. In the next Section, we will address the case of infinite mobility space by abstracting subscriber mobility with a stochastic technique called *Semi-Markov* model.

4.4.3 Semi-Markov Mobility Model

As previously discussed, there are scenarios where a mobile subscriber has infinitely many possible movement trajectories throughout its lifetime in the system. The contact/outage duration distributions of a subscriber in such scenarios cannot be estimated by the methods proposed in Section 4.4.2.

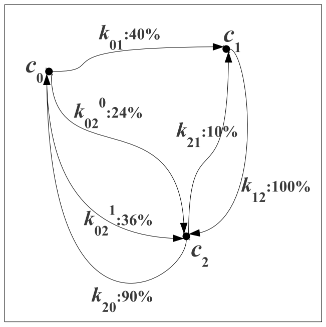
However, despite having indefinitely many possible trajectory paths, many mobility models exhibit the behavior where the infinitely many trajectories can be decomposed into finite trajectories. Some examples of mobility models with such recurrent behavior include random waypoint mobility, random walk mobility, Manhattan mobility. In order to estimate the contact/outage duration distributions of a mobile subscriber with such recurring mobility, we propose the use of *Semi-Markov* model to decompose the infinitely many possible trajectories into independent but finite "patterns". The contact/outage duration distributions for each pattern are then computed independently.

In Section 4.4.3.1, we first define the Semi-Markov mobility model and show examples of how it can be used to describe many widely-used mobility models. We then propose the contact/outage duration estimation algorithms for the Semi-Markov mobility model in Section 4.4.3.4.

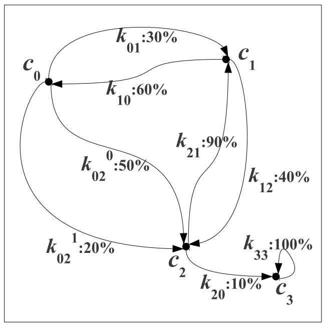
4.4.3.1 Semi-Markov Graph

Here, we define Semi-Markov mobility model as follows.

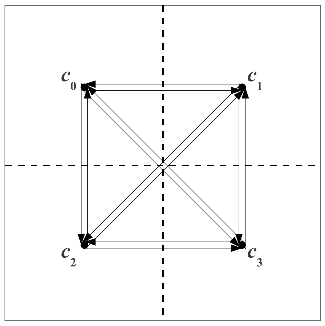
Definition A Semi-Markov mobility graph is defined as a directed graph $\dot{G} = (\dot{V}, \dot{E})$ where $\dot{V}$ represents the set of the mobility *renewal states* and $\dot{E}$ denotes the set of *transitions* between any two renewal states. A *renewal state* $\dot{v}_i \in \dot{V}$ represents an independent state on which the mobile subscriber's subsequent actions only depend. That is, the subsequent actions of a subscriber currently at state $\dot{v}_i$ will only depend on the state $\dot{v}_i$ without depending on the subscriber's previous states so far. Each renewal state $\dot{v}_i$ has its corresponding coordinate c_i, representing the geographical location of that state. A renewal state $\dot{v}_i$ with location c_i can be, for example, a point of interest or a point of operation for a mobile subscriber. A state transition $\dot{e}_{ij} \in \dot{E} \subseteq (\dot{V} \times \dot{V})$, denoting a transition between state $\dot{v}_i$ to state $\dot{v}_j$, is defined as $(\dot{p}_{ij}, q_{ij})$ where $\dot{p}_{ij}$ is the transition probability from state $\dot{v}_i$ to $\dot{v}_j$ and q_{ij} is the corresponding coordinate-trajectory joint probability (defined in Section 4.4.2.4) that describes all possible trajectories for the transition from $\dot{v}_i$ to $\dot{v}_j$. Likewise, transition $\dot{e}_{ij}$'s mobility space $Q_{ij} = \{(c,k) \in \mathbb{C} \times \mathbb{K} : q_{ij}(c,k) > 0\}$



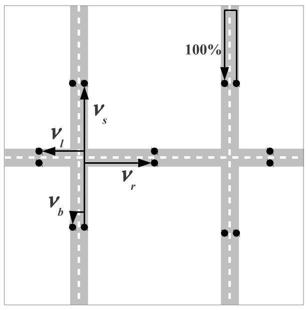
(a) A 3-state Semi-Markov graph



(b) A 4-state Semi-Markov graph with absorbing state



(c) Mapping a Random waypoint model to a Semi-Markov graph with $|\dot{V}| = 4$



(d) Mapping Manhattan model to Semi-Markov graph

Figure 4.8: Semi-Markov mobility graph and its mapping from other models

Note that the total transition probability from each state must be equal to one (i.e., $\sum_{0 \leq j < |\dot{V}|} \dot{p}_{ij} = 1$), and all possible trajectories in a transition $\dot{e}_{ij}$ must start from state $\dot{v}_i$'s location and end at state $\dot{v}_j$'s location (i.e., $\forall (c,k) \in Q_{ij} : (c = c_i) \wedge (c \xrightarrow{k} c_j)$). It can be seen that the Semi-Markov mobility graph exhibits the behavior of a *Semi-Markov process* (i.e., a Markov chain where each inter-state transition time is a random variable of any distribution), from which the Semi-Markov name is motivated. Figure 4.8(a) shows an example of a Semi-Markov graph with 3 states. When a mobile subscriber reaches state $\dot{v}_0$, it chooses to go to state $\dot{v}_1$ with probability $\dot{p}_{01} = 0.4$ and to state $\dot{v}_2$ with probability $\dot{p}_{02} = 0.6$ If the mobile subscriber from state $\dot{v}_0$ decides to move to state $\dot{v}_1$, it will picks trajectory k_{01} with probability 1.0. However, if it decides to move to state $\dot{v}_2$, it can either picks trajectory k_{02}^0 with probability 0.5 or trajectory k_{02}^1 with probability 0.5. Hence, the trajectory transition probability of trajectories k_{02}^0 and k_{02}^1 are $0.6 \times 0.4 = 0.24$ and $0.6 \times 0.6 = 0.36$ respectively. Also, it is possible for a Semi-Markov graph to contain *absorbing* states (i.e., terminal state without transition to other states but themselves). Absorbing states can represent, for example, the user's terminal locations in the map. Figure 4.8(b) shows an example of a Semi-Markov graph with one absorbing state $\dot{v}_3$.

As seen from the example, a mobile subscriber with Semi-Markov mobility can move with infinitely many possible trajectories. However, each movement is repetitive and consists of only a limited set of trajectories. A Semi-Markov graph can be used to describe many steady-state mobility models. To apply Semi-Markov model to a mobility model, it is essential to identify the renewal states of such model. Here, we present examples how to use Semi-Markov mobility to describe Random Waypoint model and Manhattan model.

4.4.3.2 Semi-Markov Mobility Mapping : Random Waypoint Model

Random waypoint model is one of the classic mobility models investigated in the literature. A subscriber in random waypoint mobility model picks a location randomly and uniformly from the map, and picks a speed value u uniformly from $[u_{\min}, u_{\max}]$ range. It then travels straight to the chosen location with the constant, chosen speed u. Upon its arrival to the destination, it stops at the destination for a constant period called pause time before choosing a new destination and speed value. The process then repeats indefinitely.

According to random waypoint mobility, it can be seen that once a mobile subscriber stops at the current destination, the subsequent actions depend only on the current stopping point and do not depend on how far or which trajectories the subscriber has been traveled. This memoryless property allows us to map random waypoint mobility model to Semi-Markov model by constructing a Semi-Markov graph as follows. First, we divide the map into small rectangular cells of equal size. For each cell, we adds a renewal state $\dot{v}_i$ with the corresponding location c_i equal to the center of the cell. Hence, each state represents the state where a mobile subscriber stops at the cell before selecting the next destination. The smaller size the cell is, the more accurately the graph represents the random waypoint model. Let $|\dot{V}|$ be the total number of cells, and hence total number of renewal states. For each state $\dot{v}_i$, we adds a transition $\dot{e}_{ij}$ to every other state $\dot{v}_j$ with equal transition probability $\dot{p}_{ij} = \frac{1}{|\dot{V}|-1}$ to reflects the uniform destination selection. Also, the transition distribution q_{ij} consists of one straight trajectory from state $\dot{v}_i$'s location c_i to state $\dot{v}_j$'s location c_j with traveling time equal to the Euclidean distance between c_i and c_j divided by the subscriber's speed[1]. Hence, the mapping will result in a complete directed graph. Figure 4.8(c) shows the example of the mapping from random waypoint mobility model to a Semi-Markov graph when $|\dot{V}| = 4$ for the purpose of visualization. Note that the number of states are usually much larger than 4 in order to get higher estimation accuracy.

4.4.3.3 Semi-Markov Mobility Mapping : Manhattan Model

Manhattan model is one of the first mobility models to describe the movement of vehicles in metropolitan road networks. In Manhattan model, the map consists of several vertical and horizontal roads. Each road has two lanes, one in each direction. Each subscriber in the map continuously moves along the roads with pre-defined, constant speed u. A junction is defined as a crosspoint between a horizontal road and another vertical road. A road segment is defined a fraction of road between any two adjacent junctions. As the subscriber reaches any junction, it decides to cross straight the junction with probability ν_s, or turn left at the junction with probability ν_l, or turn right at the junction with probability ν_r, or make a back turn with probability ν_b. The turning probability values $\nu_s, \nu_l, \nu_r, \nu_u$ are pre-defined constants that sum up to 1. A vehicle that reaches the map border will bounce back to the reverse direction of the same road with probability 1.

To map Manhattan model into Semi-Markov model, renewal states must be defined. It is not hard to see that a pair of $(\dot{r}, \dot{d})$, where $\dot{r} \in \dot{R}$ is a road segment and $\dot{d}$ is the subscriber direction, forms an independent renewal state. Since there are only two possible two directions for each road segment (East/West for horizontal road segments and North/South for vertical road segments), There will be $2|\dot{R}|$ renewal states in the corresponding Semi-Markov graph, where $|\dot{R}|$ is the total number of road segments. Each state $\dot{v}_i$ has its corresponding location c_i equal to the middle point of the corresponding road segment. Each state has four transitions, representing the scenarios where the subscriber in that state travels to the next junction and go straight/left/right/ back with transition probability ν_s, ν_l, ν_r, and ν_b respectively. Each transition $\dot{e}_{ij}$'s mobility space also

[1]Here we assume the pause time to be zero and there is only one possible subscriber speed (i.e., $u_{\min} = u_{\max}$). However, it is easy to modify q_{ij} to reflects the case of non-zero pause time and multiple possible speed values.

contains one trajectory from state $\dot{v}_i$'s location c_i to the next junction and then to state $\dot{v}_j$'s location c_j (and thus having the duration equal to the average length between $\dot{v}_i$'s road segment length and $\dot{v}_j$'s road segment length, divided by the speed u). Figure 4.8(d) shows an example of such mapping in a simple Manhattan model with two junctions.

4.4.3.4 Contact/Outage Duration in Semi-Markov Model

Once a Semi-Markov mobility graph is defined, we can then formulate the contact/outage duration estimation problem by modeling a subscriber s's mobility as a Semi-Markov graph as follows.

Problem 3 *Given subscriber s's mobility as a Semi-Markov graph $\dot{G}_s(\dot{V}, \dot{E})$ and a connectivity map Ω, estimate the corresponding contact duration distribution $f_W^{\dot{G}_s,\Omega}(t)$, and outage duration distribution $f_{\overline{W}}^{\dot{G}_s,\Omega}(t)$.*

While Problem 3 involves a more complex and generic mobility abstraction than the problems previously defined in Section 4.4.2, it can be solved by the same principle with other problems. Specifically, there are two steps in the estimation algorithms. First, we need to find the average fraction of time each possible trajectory is visited. Second, the contact/outage duration distributions are then computed by estimating the average number of occurrences of contact/outage phases with certain duration.

In the first step (i.e., finding the fraction of time each trajectory is visited), we need to calculate the fraction of time each renewal state $\dot{v}_i \in \dot{V}$ is visited, denoted by π_i. There are two separate cases depending on whether the Semi-Markov graph contains any absorbing state. If there is any absorbing state in its Semi-Markov graph, a mobile client will end up staying in the absorbing state eventually. In such case, the transition probability matrix $\dot{P} = [\dot{p}_{ij}]$ is an absorbing Markov chain[65], which can be rearranged into its *canonical form* as,

$$\dot{P} = \left(\begin{array}{c|c} \dot{\mathbb{T}} & \dot{\mathbb{R}} \\ \hline 0 & I \end{array}\right)$$

where $\dot{\mathbb{T}}$ is the transition probability matrix between any two transient (non-absorbing) states, $\dot{\mathbb{R}}$ is the transition probability matrix from each transient state to each absorbing state, and I is the identity matrix (since each absorbing state only has a transition to itself). From transient matrix

$\dot{\mathbb{T}}$, we can calculate the corresponding *fundamental matrix* Λ as

$$\Lambda = (I - \dot{\mathbb{T}})^{-1}$$

The fundamental matrix Λ has the same dimension as the transient matrix $\dot{\mathbb{T}}$. Each entry $\dot{\lambda}_{ij} \in \Lambda$ denotes the expected number of times state $\dot{v}_j$ is visited before absorption, given that state $\dot{v}_i$ is the starting state. Hence, given the starting state $\dot{v}_s$, we can calculate the expected fraction of occurrences π_i for each transient state $\dot{v}_i$ as

$$\begin{aligned}\pi_i &= \frac{\text{E[\#visit at state } \dot{v}_i \text{ before absorption]}}{\text{E[\#visit at all states before the absorption]}} \\ &= \frac{\dot{\lambda}_{si}}{\sum_{0\le j<|V|} \dot{\lambda}_{sj}} \end{aligned} \tag{4.8}$$

Note that $\pi_i = 0$ if state $\dot{v}_i$ is an absorbing state because we assume that the subscriber's movement terminates at the absorbing state.

For example, consider the Semi-Markov graph in Figure 4.8(b). Its corresponding transition matrix P is

$$\dot{P} = \begin{pmatrix} \dot{p}_{00} & \dot{p}_{01} & \dot{p}_{02} & \dot{p}_{03} \\ \dot{p}_{10} & \dot{p}_{11} & \dot{p}_{12} & \dot{p}_{13} \\ \dot{p}_{20} & \dot{p}_{21} & \dot{p}_{22} & \dot{p}_{23} \\ \dot{p}_{20} & \dot{p}_{21} & \dot{p}_{22} & \dot{p}_{23} \end{pmatrix} = \begin{pmatrix} 0 & 0.3 & 0.7 & 0 \\ 0.6 & 0 & 0.4 & 0 \\ 0.0 & 0.9 & 0 & 0.1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Hence, its corresponding transient matrix $\dot{\mathbb{T}}$ is the upper-left 3×3 sub-matrix of $\dot{P}$ and the fundamental matrix is

$$\Lambda = \begin{pmatrix} 7.805 & 11.341 & 10 \\ 7.317 & 12.195 & 10 \\ 6.585 & 10.976 & 10 \end{pmatrix}$$

Assuming that $\dot{v}_0$ is the starting state, then we can calculate the state occurrence vector π as

$$\pi = \begin{pmatrix} 0.36 & 0.34 & 0.3 & 0 \end{pmatrix}$$

If the Semi-Markov mobility graph has no absorbing state (e.g., Figure 4.8(a)), which is the case for Random waypoint model and Manhattan model, then the graph's corresponding Markov chain is ergodic[65]. In such case, the average fraction of time a state $\dot{v}_i$ is visited π_i can be calculated from the steady-state equation

$$\pi = \pi \dot{P} \tag{4.9}$$

with the constraint

$$\sum_{0 \leq i < |\pi|} \pi_i = 1$$

, where $\dot{P}$ is the state transition matrix $[\dot{p}_{ij}]$ and π is the state vector $[\pi_i]$.

For example, the state transition matrix P for the Semi-Markov graph in Figure 4.8(a) is

$$\dot{P} = \begin{pmatrix} \dot{p}_{00} & \dot{p}_{01} & \dot{p}_{02} \\ \dot{p}_{10} & \dot{p}_{11} & \dot{p}_{12} \\ \dot{p}_{20} & \dot{p}_{21} & \dot{p}_{22} \end{pmatrix} = \begin{pmatrix} 0 & 0.4 & 0.6 \\ 0 & 0 & 1 \\ 0.9 & 0.1 & 0 \end{pmatrix}$$

Solving Equation (4.9) with the matrix $\dot{P}$ yields the steady-state vector

$$\pi = \begin{pmatrix} \pi_0 & \pi_1 & \pi_2 \end{pmatrix} = \begin{pmatrix} 0.38 & 0.2 & 0.42 \end{pmatrix}$$

Once the expected fraction of occurrence π_i for each state $\dot{v}_i \in \dot{V}$ is calculated either by Equation (4.8) or Equation (4.9), we can calculate the expected fraction of occurrence for each transition $\dot{e}_{ij}$, denoted by π_{ij}, as

$$\begin{aligned} \pi_{ij} &= \text{E[fraction of time transition } \dot{e}_{ij} \text{ is visited]} \\ &= \text{E[fraction of time } \dot{v}_i \text{ is visited].P[subscriber in } \dot{v}_i \text{ jumps to } \dot{v}_j] \\ &= \pi_i . \dot{p}_{ij} \end{aligned} \tag{4.10}$$

where $\dot{p}_{ij}$ is the corresponding state transition probability for transition $\dot{e}_{ij} \in E$.

Once the expected fraction of occurrence π_{ij} for each transition $\dot{e}_{ij} \in E$ is calculated at the end of the first step, we can then calculate the contact duration distribution $f_W^{\dot{G}_s,\Omega}$ for subscriber s's Semi-Markov mobility graph $\dot{G}_s = (\dot{V}, \dot{E})$ and connectivity map Ω as follows.

$$\begin{aligned} f_W^{\dot{G}_s,\Omega}(t) &= \text{P}[s\text{'s contact duration } = t] \\ &= \frac{\text{E[size of } s\text{'s contact subset with duration } t]}{\text{E[size of } s\text{'s contact set]}} \\ &= \frac{\sum_{\dot{e}_{ij}\in\dot{E}} \pi_{ij}.|W^{q_{ij},\Omega}(t)|}{\sum_{\dot{e}_{ij}\in\dot{E}} \pi_{ij}.|W^{q_{ij},\Omega}|} \end{aligned} \tag{4.11}$$

where $|W^{q_{ij},\Omega}(t)|$ and $|W^{q_{ij},\Omega}|$ are defined in Section 4.4.2.4. Similarly, the outage duration distribution $f_{\overline{W}}^{\dot{G}_s,\Omega}$ for subscriber s's Semi-Markov mobility graph $\dot{G}_s = (\dot{V}, \dot{E})$ and connectivity map Ω can be calculated as

$$f_{\overline{W}}^{\dot{G}_s,\Omega} = \frac{\sum_{\dot{e}_{ij}\in\dot{E}} \pi_{ij}.|\overline{W}^{q_{ij},\Omega}(t)|}{\sum_{\dot{e}_{ij}\in E} \pi_{ij}.|\overline{W}^{q_{ij},\Omega}|} \tag{4.12}$$

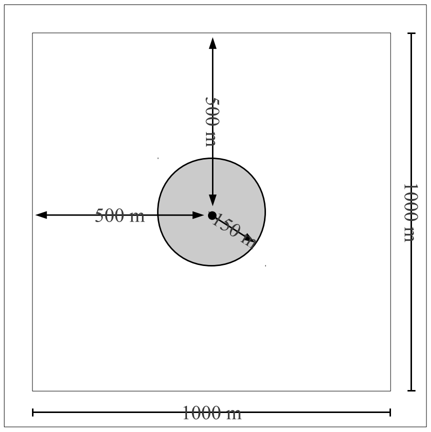
where $|\overline{W}^{q_{ij},\Omega}(t)|$ and $|\overline{W}^{q_{ij},\Omega}|$ are defined in Section 4.4.2.4.

Note that Equation (4.11) and Equation (4.12) estimates the contact/outage duration distribution of a Semi-Markov mobility graph with the assumption that each location-trajectory pair (c, k) in each state transition $\dot{e}_{ij}$ contains more than one phases (i.e., $|\Phi^{c,k,\Omega}| \geq 2$ for all $(c, k) \in Q_{ij}$, for all $\dot{e}_{ij} \in \dot{E}$). If there exists a trajectory with only one phase between two states, it means that the corresponding phase length cannot be computed from that trajectory alone (since the duration is not bounded). In such case, the graph must be traversed and the phase duration is aggregated until a trajectory with more than one phase is met. For graph traversal, we use exhaustive search on the graph until the aggregate probability is less than a constant bound.

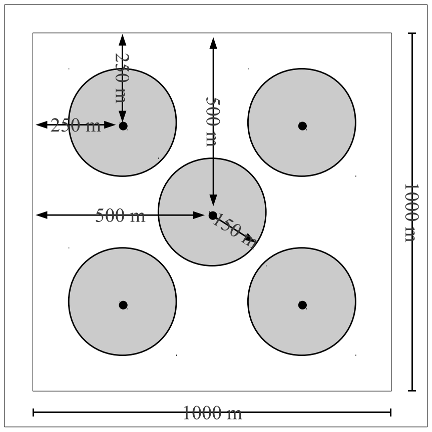
4.5 Evaluation

This section presents the validation of the proposed analytical model via simulations. We first describe simulation parameter settings in Section 4.5.1. We then present the evaluation results for the proposed contact/outage duration estimation algorithms (i.e., Section 4.4.3) in Section 4.5.2. Finally, we present the evaluation results of the protocol analysis (i.e., Section 4.2) in Section 4.5.3.

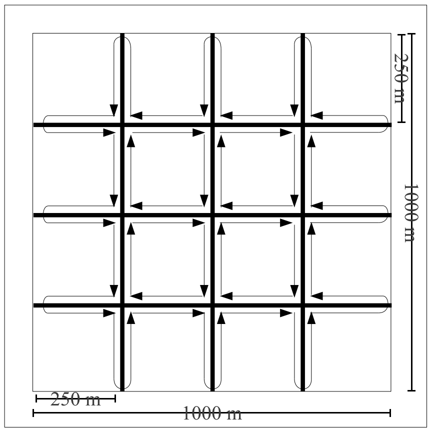
4.5.1 Parameter Settings



(a) 1 AP scenario



(b) 5 APs scenario



(c) Manhattan scenario

Figure 4.9: Simulation Map

We validate our proposed analytical model via simulations with NS-2 network simulator[50]. Each simulation runs in a 1000-meter x 1000-meter square region with 150 meter radio transmission range. We test the model with two mobility models, Random waypoint model and Manhattan model mentioned in Section 4.4.3.2 and Section 4.4.3.3 respectively. The number of mobile subscribers in each experiment is 40. In each experiment, there are 10 publishers publishing 4 topics in total. In the experiment, the subscribers move in the map according the specified mobility model. Each broker in the map periodically sends out the beacon message and the mobile subscribers detect the access point when it receives the beacon message. Each mobile subscriber records the duration of the contact phases and outage phases it experiences throughout the simulation. Each simulation runs for 1 hour. The contact/outage duration distributions and subscribers' reliability values are then constructed and averaged from 5 simulation runs for each parameter setting.

There are two broker deployment scenarios used in the experiments. The first scenario consists of only one broker in the middle of the map (Figure 4.9(a)) and the second scenario consists of 5 brokers distributed evenly (Figure 4.9(b)). For Random waypoint mobility model, the mobile subscribers move freely in the simulation area. For Manhattan mobility model, the subscribers move only along the junction specified in Figure 4.9(c). We vary subscriber speed from 1 m/s to 25 m/s in Random waypoint model and from 1 m/s to 60 m/s in Manhattan model. We vary subscriber pause time from 1s to 60s in both mobility models.

4.5.2 Contact/Outage Duration Estimation

Figure 4.10 and Figure 4.11 show the CDF of contact/outage duration distributions for various simulation parameters. The plots show both analytical result and simulation result (denoted by anl and sim in each figure respectively). As seen from the figures, each analytical CDF graph exhibits similarity to its simulation counterpart, yielding accuracy in our proposed contact/outage duration estimation algorithm.

4.5.2.1 Effect of Mobility Pattern

By comparing the contact/outage duration distributions between Random waypoint model and Manhattan model in the same setting (e.g., Figure 4.10(a) and Figure 4.10(e)), we can conclude that the contact/outage duration distributions of Manhattan model are more deterministic than their Random waypoint model counterparts (i.e., contact/outage distributions from Random waypoint model are more long-tailed than their Manhattan counterparts). That is, the contact/outage duration distributions share the same level of determinism from its corresponding mobility model. This finding can be useful when designing or deploying the publish/subscribe system for different mobility models. For example, it is relatively easier to find good locations to deploy brokers when node mobility follows Manhattan model than when node mobility follows Random waypoint model, because there will be relatively fewer "good" locations for Manhattan scenario.

4.5.2.2 Effect of Broker Deployment

One could simply conclude by looking the graphs that it is trivially better to have more brokers deployed on the map. For example, comparing Figure 4.10(a) to Figure 4.10(c) and comparing Figure 4.10(b) to Figure 4.10(d), having more deployed brokers generally increases contact duration while decreasing outage duration, which are good from subscriber reliability's perspective. However, it is possible that a system with fewer brokers deployed in good locations may yield longer contact duration (or shorter outage duration) than another system that deploys more brokers in bad locations. The effect of location to the contact/outage duration distributions has not been explored in this work and can be considered as future research direction.

4.5.2.3 Effect of Subscriber Speed

Each sub-figure in Figure 4.10 shows contact or outage duration distributions for each setting with zero pause time and different subscriber speed values. As seen from the figures, increasing node speed while fixing pause time generally decreases both contact/outage duration lengths. Hence, it is not clear how speed affects the performance of the protocol by analyzing contact/outage duration distributions. The effect of subscriber speed to protocol performance will be discussed in Section 4.5.3.

4.5.2.4 Effect of Subscriber Pause Time

Each sub-figure in Figure 4.11 shows contact or outage duration distributions for each setting with 10 m/s subscriber speed and different pause time values. In contrast to subscriber speed, increasing subscriber pause time generally increases both contact/outage duration lengths. However, we cannot make any conclusion from the subscriber reliability perspective as subscriber pause time has the same effect on the contact duration length and outage duration length. Section 4.5.3 will discuss such effect in detail.

4.5.3 Protocol Performance

This section presents the evaluation in terms of average subscriber real-time reliability. Unless specified otherwise, each simulation is run with 10m/s subscriber speed with zero pause time. The default event lifetime is 60 seconds and the maximum protocol handoff delay equal to 1 second.

Again, we use either Random waypoint or Manhattan model on either 1-broker map or 5-broker map for each simulation.

There are two modes of publish/subscribe used in the simulations. The first mode is basic publish/subscribe without mobility management. Thus, each broker does not buffer any incoming event on behalf of disconnected subscribers. The second mode is mobility-aware publish/subscribe (denoted by term "+broker" in each graph), where each broker buffers incoming events on behalf of disconnected subscribers and transfer the buffered events to a disconnected subscriber when it reconnects the broker network.

Figure 4.12, Figure 4.13, and Figure 4.14 show the average subscriber real-time reliability for various system parameter settings. Each of the plots consists of both analytical results and simulation results. In general, our analytical model accurately predict subscriber real-time reliability.

4.5.3.1 Effect of Mobility Management

Figure 4.12 and Figure 4.13 compare the subscriber reliability between basic mode and mobility-aware mode (i.e., buffer mode) for each simulation setting. Note that the mobility-aware mode significantly outperforms the basic mode in all cases. This performance gain comes at the cost of buffering at each broker during the subscriber's disconnection period.

4.5.3.2 Effect of Broker Coverage

Figure 4.12 and Figure 4.13 also compare the subscrber reliability between different broker maps for each parameter setting. As expected, adding more brokers into the map generally increases the broker coverage and hence subscriber reliability.

4.5.3.3 Effect of Subscriber Speed

Figure 4.12 shows the effect of subscriber speed to the subscriber real-time reliability. From the figure, we can conclude that speed generally has no effect in the basic publish/subscribe settings. This is because node speed does not change the fraction of disconnection time of the subscriber. This explanation is consistent with the results from Section 4.5.2, as the increasing speed decreases both contact duration and outage duration.

For the mobility-aware publish/subscribe settings, however, increasing subscriber speed results in significant increase in average subscriber real-time reliability. This can be explained as follows. Although increasing subscriber speed does not change the fraction of time a subscriber is disconnected, it shortens duration of each outage phase (the subscriber get disconnected faster, but also get reconnected faster as well). According to Equation (4.3), decreasing outage duration will cause each buffered event to spend less time in the buffer before getting transferred, resulting in higher real-time reliability.

4.5.3.4 Effect of Subscriber Pause Time

The explanation of the effect of speed to subscriber reliability also applies to the effect of subscriber pause time as well. Specifically, increasing subscriber pause time does not change subscriber reliability in basic publish/subscribe protocols because it does not change the fraction of time each subscriber is disconnected. However, increasing subscriber pause time will increase subscriber outage duration, resulting in longer time each buffered event needs to stay in the buffer. Thus increasing subscriber pause time generally decreases subscriber real-time reliability in mobility-aware publish/subscribe systems.

4.5.3.5 Effect of Event Life time

Figure 4.14 shows the effect of event lifetime to the subscriber real-time reliability in mobility-aware publish/subscribe systems. We do not show the results for basic publish/subscribe systems as the events are not buffered at all in such mode. We can conclude that increasing event lifetime generally increases subscriber reliability in mobility-aware publish/subscribe systems. However, the event lifetime is application-dependent and could not be changed flexibly. However, the result shows that the same publish/subscribe system may have different effect to different applications with different requirements.

4.5.3.6 Effect of Handoff Delay

Figure 4.15 shows the effect of maximum handoff delay value ($H_{\max}$) to the prediction accuracy of the model. As seen from Figure 4.15, as compared to Figure 4.12, the more the $H_{\max}$ is, the less accurate the predictive model becomes. This is because the assumption ($H_{\max} << E[\overline{W}_s]$) made in the analysis (Equation (4.2)) is no longer valid. Note that the accuracy decreases when the subscriber speed increases due to the same reason.

4.6 Related Work

Mobility handoff and its effect on publish/subscribe systems is not a new research area and has been investigated by numerous works[59, 60, 61, 24]. However, to the best of our knowledge, none of such works explores the effect of subscriber mobility model to the publish/subscribe systems from the analytical point of view. This chapter of thesis first proposes a generic mobility model that can abstract various existing mobility models and yet fit into the analytical framework for publish/subscribe performance estimation.

From a broader perspective of mobile networking, there have been numerous works to analyze and optimize the performance of mobile networks where users can roam around and change their association with the network infrastructure. In fact, the analytical results of user mobility and its effect to network performance in cellular networks have been long well established with the conclusion that most user mobility metrics such as cell residence time or cell hand-off rate follow some certain distributions (e.g., gamma distribution or negative exponential distribution)[66]. However, such findings are based on the assumption of cellular network topology (i.e., multi-kilometer cell size, dense/grid cell deployment), which is different from our sparse broker deployment scenarios.

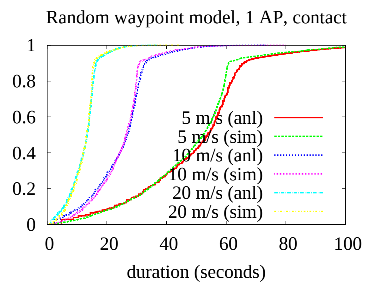
Recently, there has been an emergence of a new phenomenon called *opportunistic Wi-Fi* network (i.e., *in situ* network) where a mobile device opportunistically discovers and connects to a free 802.11 access point for free network connectivity[67, 68]. Our analytical model can also be used to calculate contact/outage duration in such scenarios. In fact, a few works have proposed the estimation of contact/outage duration in the context of opportunistic Wi-Fi networks[69, 70, 71]. However, such works focus on finding the maximum contact/outage duration. Our model estimates contact/outage duration at the distribution granularity, which is more accurate and useful for quality of service analysis.

4.7 Discussion

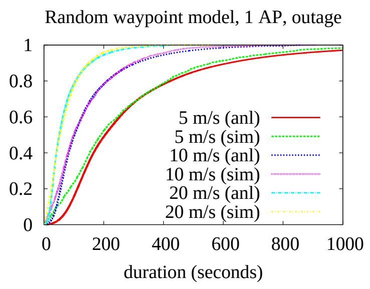
This chapter of thesis explored the effect of subscriber mobility (i.e., mobility dynamism) to the performance of distributed, content-based publish/subscribe systems. We first formulated the problem of mobile subscribe real-time reliability estimation problem. We then modeled the subscriber real-time reliability as a function of four parameters : subscriber contact duration, subscriber outage duration, protocol handoff delay, and event lifetime. We then proposed the algorithm to estimate subscriber real-time reliability from such four parameters. Furthermore, we proposed the analytical model to estimate subscriber contact/outage duration distributions as a function of broker deployment and subscriber mobility model. Our analytical model abstracts a wide range of mobility models, from simple deterministic mobility to complex Semi-Markov mobility. Finally, we proved the accuracy of our predictive model via simulations.

There are few issues in this work that worth exploring in the future. First, a more accurate model that accounts for more fine-grained parameters and corner cases can be built on top of the proposed model. For example, the current model does not consider the region covered by several overlapping brokers. In such scenario, the current model will treat such region as if it is covered by one long-range broker. In reality, a subscriber may switch among such overlapping brokers, leading to unnecessary traffic and synchronization overhead. A more accurate model for such scenario is left as future work.

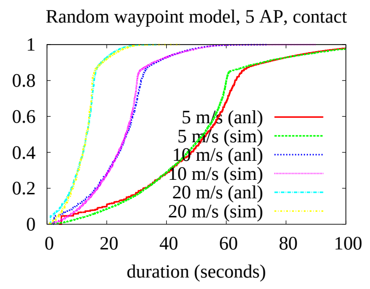
Another research direction is how to leverage the proposed prediction model for performance improvement in mobile publish/subscribe systems. For example, one can use the proposed model to adjust broker location, broker transmission range, or broker capability in order to maximize subscriber reliability. In fact, a few works have started to explore such direction in the context of generic wireless networks [70, 71]. However, a more accurate optimization based on our proposed model can be done in the future.



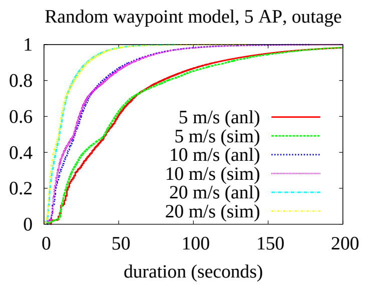
(a) 1 AP, Random waypoint (contact)



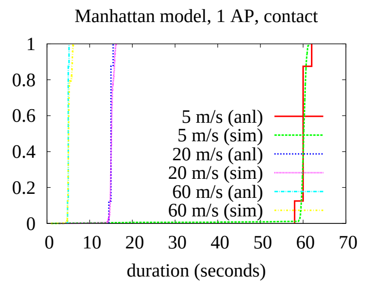
(b) 1 AP, Random waypoint (outage)



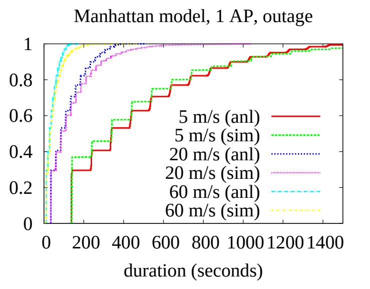
(c) 5 AP, Random waypoint (contact)



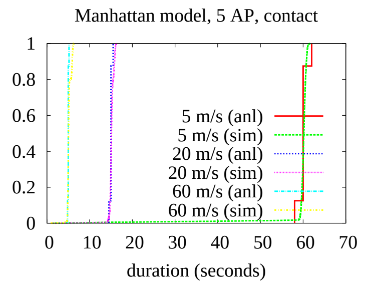
(d) 5 AP, Random waypoint (outage)



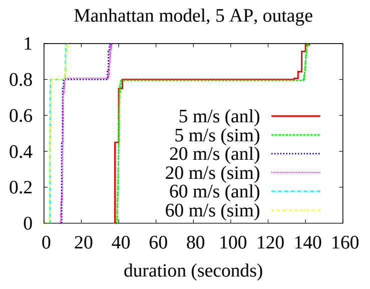
(e) 1 AP, Manhattan (contact)



(f) 1 AP, Manhattan (outage)

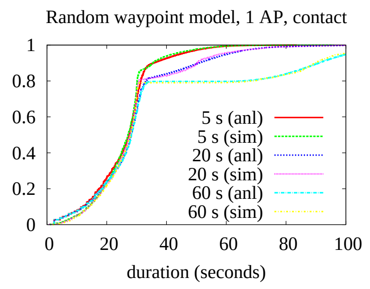


(g) 5 AP, Manhattan (contact)

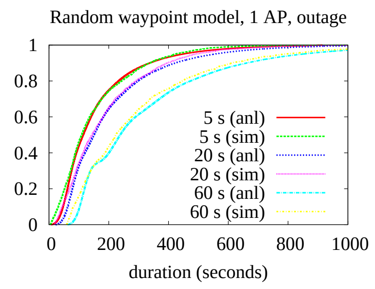


(h) 5 AP, Manhattan (outage)

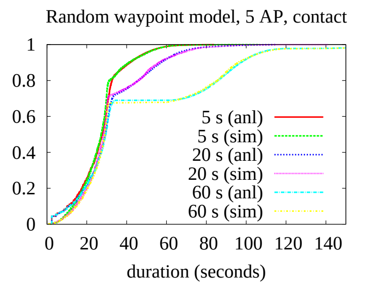
Figure 4.10: Effect of different subscriber speed when pause time = 0 s



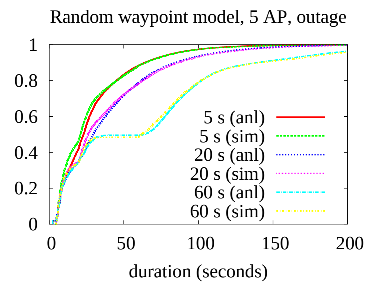
(a) 1 AP, Random waypoint (contact)



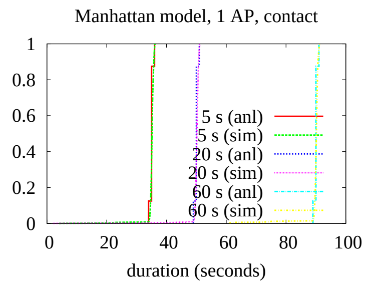
(b) 1 AP, Random waypoint (outage)



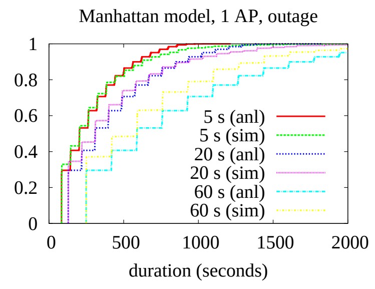
(c) 5 AP, Random waypoint (contact)



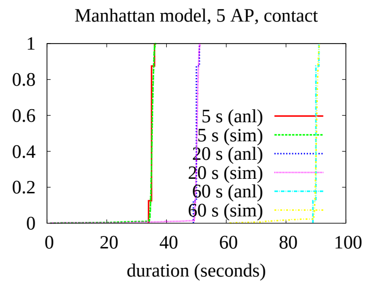
(d) 5 AP, Random waypoint (outage)



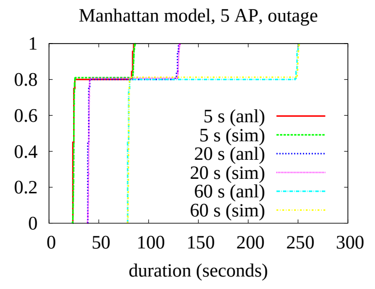
(e) 1 AP, Manhattan (contact)



(f) 1 AP, Manhattan (outage)

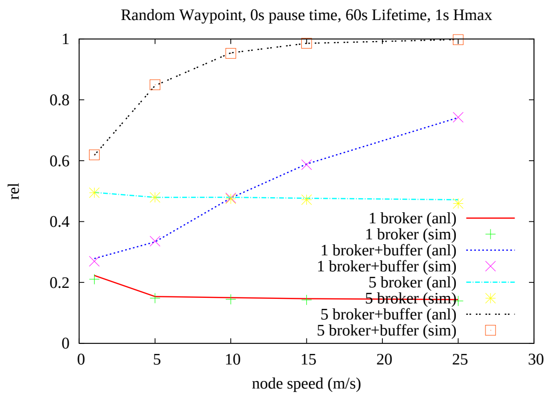


(g) 5 AP, Manhattan (contact)

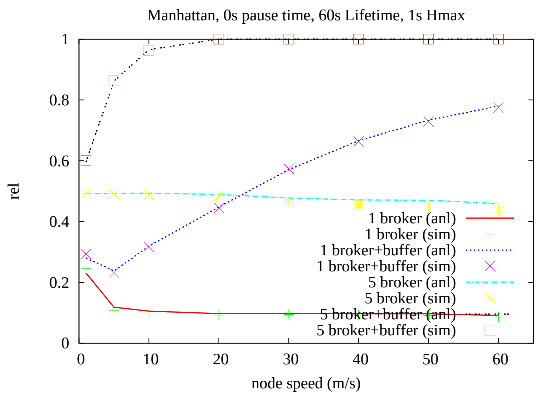


(h) 5 AP, Manhattan (outage)

Figure 4.11: Effect of different subscriber pause time with 10 m/s speed

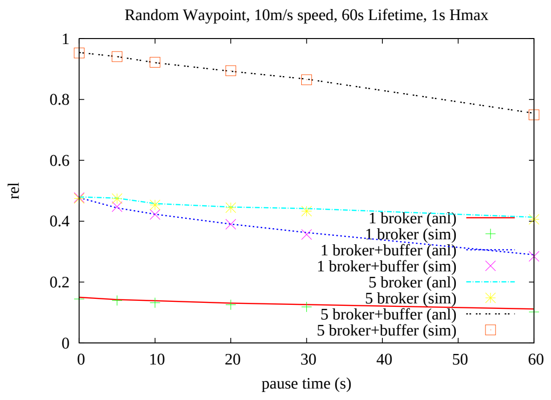


(a) Random waypoint model

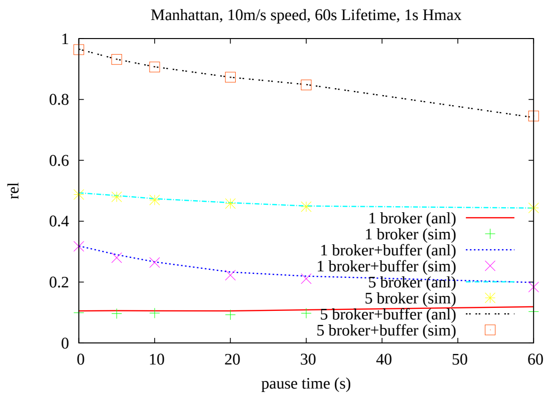


(b) Manhattan model

Figure 4.12: Average subscriber real-time reliability with 0s pause time, 60s event lifetime, 1s handoff latency



(a) Random waypoint model



(b) Manhattan model

Figure 4.13: Average subscriber real-time reliability with 10m/s speed, 60s event lifetime, 1s handoff latency

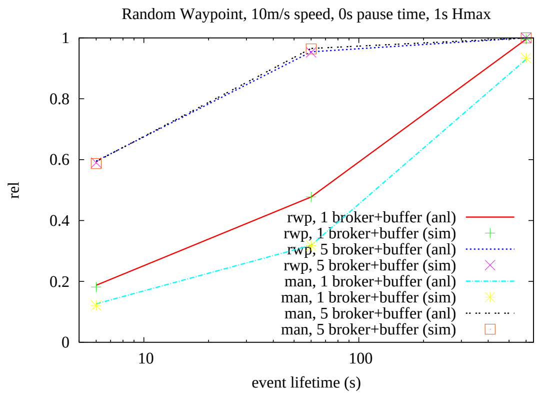
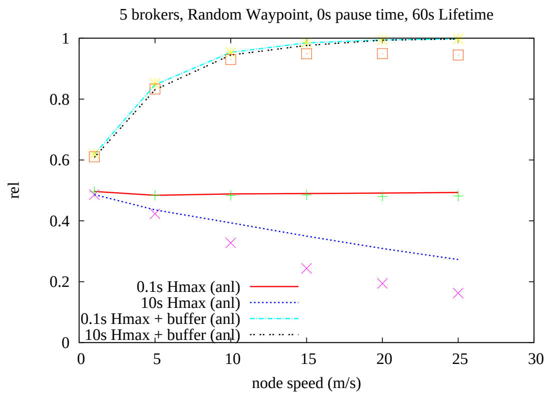
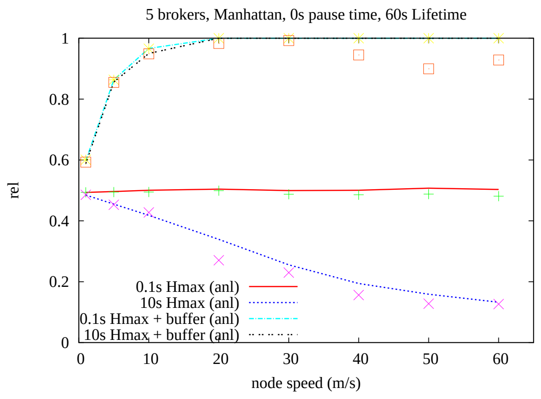


Figure 4.14: Average subscriber real-time reliability with 10m/s speed, 0s pause time, 1s handoff latency



(a) Random waypoint model



(b) Manhattan model

Figure 4.15: Average subscriber real-time reliability with 5 brokers, 0s pause time, 60s event lifetime

CHAPTER 5

RELIABILITY AND TIMELINESS ANALYSIS WITH OVERLAY DYNAMISM

5.1 Introduction

So far in previous chapter, we have proposed the analytical model to analyze subscribe reliability in content-based publish/subscribe systems with the assumption of no failures inside the overlay broker. Such assumption, however, does not reflect the real-world system where server and network failures are possible, though probably not frequent. While high-end commercial servers with high maintenance generally achieve at least 99.9% availability (i.e., available 99.9% of the time)[72], most standard, off-the-shelf commodity servers with low to moderate maintenance may have less than 90% availability[73]. A distributed system that consist of multiple servers connected via best-effort communication links could have lower service availability than its single server component[74]. To cope with such issue in the context of publish/subscribe systems, several fault-tolerance/fault-recovery techniques have been proposed to increase service availability in distributed publish/subscribe systems[75, 76]. However, to the best of our knowledge, the effect of such availability-enhancement techniques to a publish/subscribe system's availability, specifically in real-time aspect, has not been analyzed yet.

This thesis chapter quantitatively analyzes the effect of failures and recovery techniques to real-time content-based publish/subscribe systems. Specifically, it proposes an analytical model that captures component failures and various recovery techniques. The primary goal of such analytical model is to estimate each subscriber's real-time reliability, which was generally defined in Section 2.4. The analytical model covers common component failures and recovery mechanisms, resulting in the model's high applicability. The analysis's key concept is to estimate pairwise reliability for each publisher-subscriber pair and then use such pairwise reliability to calculate each subscriber's real-time reliability. The calculation of publisher-subscriber pairwise reliability, however, depends on fault-tolerance/fault-recovery mechanisms that are used by the system. This chapter also presents the calculation of such pairwise reliability for each fault-tolerance/fault-recovery mechanism. Evaluation results via simulations yield accuracy and effectiveness of the proposed analytical model.

The organization of this chapter is as follows. We first give the overview of potential failures in the broker overlay in Section 5.2 and the overview of various fault tolerance/recovery mechanisms in Section 5.3. In Section 5.4, we then formulate the problem of subscriber real-time reliability estimation with unreliable broker networks and propose a generic, protocol-independent analytical framework to solve such problem. In Section 5.5, we propose a set of protocol-dependent, publisher-subscriber pairwise reliability estimation models for each fault tolerance/recovery mechanism. Section 5.6 presents a simpole optimization called *home broker selection* on top of the proposed analytical model. Section 5.7 then presents the evaluation results to validate the proposed analytical model via simulations. Section 5.8 presents related work in fault tolerance/recovery mechanisms in publish/subscribe systems and their analysis. Finally, Section 5.9 discusses future directions of overlay analysis in publish/subscribe systems.

5.2 Overlay Dynamism

This section describes different types of failure that potentially occur in a broker overlay network. There are two types of failure that are considered in the proposed analytical model, which are broker failure, link failure.

5.2.1 Broker Failure

A broker's failure can be either hardware failure (e.g., motherboard, power supply, network card) or software failure (e.g., operating system, device driver). However, we assume crash-recovery failure model, where each broker is assumed to be either *on* or *off*. We assume that a broker could crash at any time. When a broker crashes, it stops its activity and loses all of its soft-state routing information, including subscription routing table and event in the queue. However, we assume that the information about its neighboring broker is still maintained in non-volatile storage. Such assumption is valid for our target scenarios, where broker set and its neighboring relationship are static. We do not assume byzantine failure such as misconfiguration or security breach.

5.2.2 Link Failure

We also assume an overlay link that follows crash-recovery failure model. Thus, a link's status can be either *on* or *off*. A link in *off* state drops any event that is sent to it. When a link is *on*, however, it delivers each packet successfully. If an event is successfully transmitted, the transmission delay is bounded by some constant (i.e., within several seconds). Several results have shown that almost of the time, the overlay link exhibits such bimodal behavior (i.e., a packet is either dropped, or transmitted within seconds)[77, 78]. The studies also have shown that overlay link failures tend to have small period (e.g., a few minutes).

Both broker failure and link failure can cause three possible application-level failures : *event loss*, *subscription loss*, and *unsubscription loss*. Event loss is the loss of a single event while being transmitted from a publisher to a subscriber via the broker overlay network. Subscription loss is either the loss of a subscription message from a subscriber, or the loss of subscription routing information at any broker due to that broker's failure. Unsubscription loss is the lost of unsubscription message from a subscriber, resulting in stale routing information in broker networks that amount to unnecessary traffic load. In general, subscription loss has more impact on the subscriber reliability, as the loss of a subscriber's subscription may prevent any subsequent events from reaching that subscriber.

In the next section, we will describe several existing reliability-enhancement mechanisms to prevent event/subscription/unsubscription loss under broker failure and link failure.

5.3 Fault Tolerance/Recovery Mechanisms

This section presents several fault tolerance/recovery schemes existing in the literature for publish/subscribe systems[75, 27, 58, 79, 80]. Instead of discussing each individual works, we summarize all the works, and discuss techniques and concepts commonly used by such works. Many of these techniques are not limited to publish/subscribe systems, but are also applicable to generic distributed systems. However, this thesis focus on the impact of such techniques in the context of publish/subscribe systems.

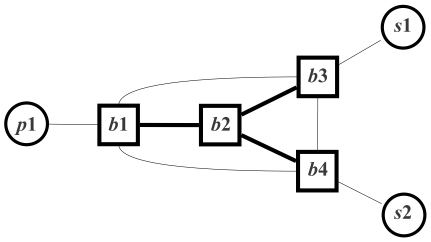
5.3.1 Periodic Subscription

Periodic subscription is one form of soft-state maintenance, which one basic mechanism to ensure safety and liveness of generic distributed systems[81]. In periodic subscription scheme, each subscriber periodically re-issues its subscription message to its home broker, which then propagates the subscription to other brokers in the network. Each broker also maintains a timestamp for each subscription entry in its routing table. The timestamp is refreshed every time the broker receives the corresponding subscription. The broker discards any subscription from its routing table if the subscription is not refreshed within a period of time (i.e., timeout). More details about periodic subscription can be found by several previous works[27, 46].

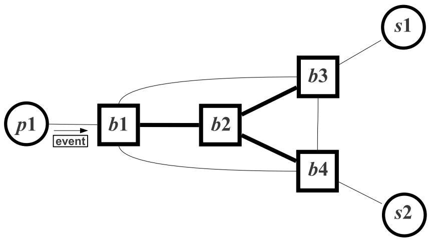
The periodic subscription scheme addresses the problem of unsubscription loss by eliminating the need of unsubscription message, since a subscriber can unsubscribe by just stop sending the periodic subscription. It also limits the effect of subscription loss by periodically refreshing the subscription message, which guarantees eventual subscription delivery when the system recovers. However, it does not address the problem of event loss, as any events generated during tree partition will be lost.

5.3.2 Event Buffering/Retransmission

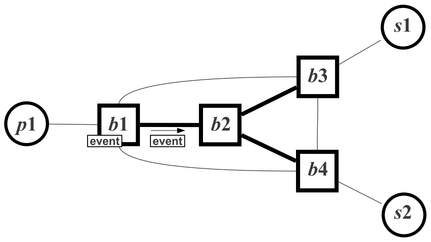
Reliable retransmission and acknowledgement are also basic link-level and application techniques in conventional point-to-point communication[82]. These techniques, when modified, can be done to improve system reliability in publish/subscribe systems as well[75, 79]. The reliable communication makes use of the event acknowledgment message (ACK) as follows. When a broker receives an event from one of its immediate neighbors, it performs the event matching and calculates the event's forwarding set (i.e., the set of immediate neighbors to forward the event to). The broker then stores the event and its forwarding set into the broker's non-volatile storage and sends the acknowledgment message (ACK) containing the event sequence number back to its upstream neighbor. The broker then forwards the event to the event's forwarding set. The broker then waits for the ACK message from each next-hop neighbor in the event's forwarding set. The broker discards the event from its non-volatile storage once it collects all the ACK messages from all brokers in the forward set, as now it is certain that the event has been received by all of the next-hop brokers. If, due to failures, the broker does not receive ACK messages from some next-hop neighbors, it retransmits the event to each of such neighbors until all ACK messages are collected or the buffered event becomes expired. Figure 5.1 illustrates an example of event buffering/retransmission scheme.



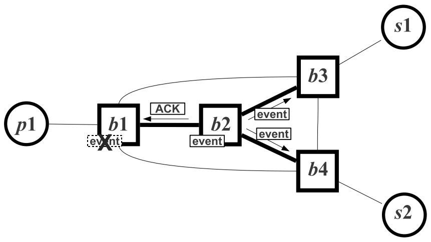
(a) Pub/sub tree topology



(b) Publisher publishes an event



(c) Local broker stores and forwards event



(d) Broker receives ACK and discards event

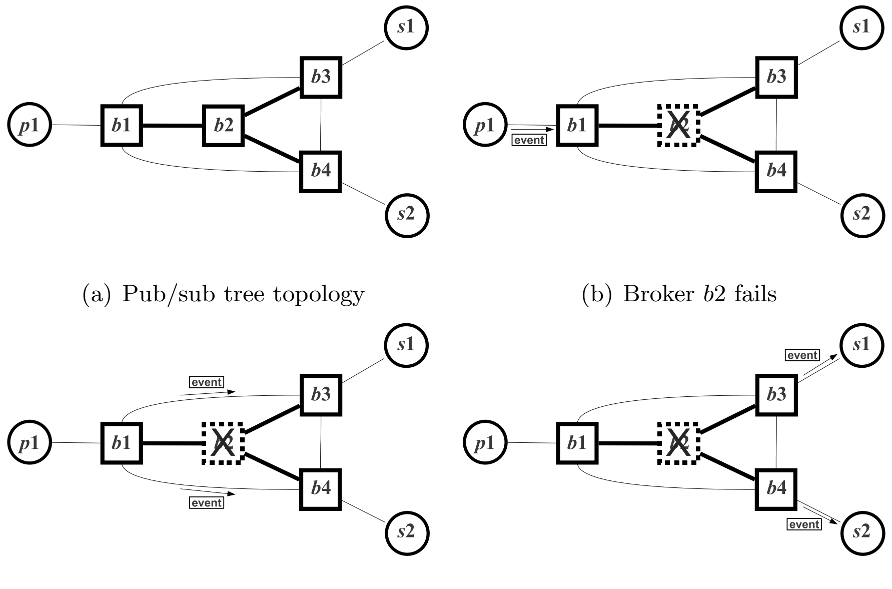
Figure 5.1: Example of event buffering/retransmission scheme

The event buffering/retransmission guarantees eventual event delivery to all designated receivers, given that the per-topic broker tree does not change over time. However, the event may expire during the buffering period. Note that this mechanism does not prevent event duplication due to the loss of ACK messages. To prevent event duplication, each downstream broker has to permanently maintain the sequence numbers of all events received so far in its persistent storage. However, this needs to be done only at the last-hop broker. The buffering/retransmission scheme can also guarantee eventual subscription/unsubscription delivery by treating each subscription/unsubscription message as a normal event to be delivered to all brokers.

5.3.3 Redundant Path Bypassing

According to our broker network model presented in Section 2.1, it is possible that there is more than one path between any two brokers in the overlay graph. In such case, if a per-topic publish/subscribe tree is disconnected due to broker/link failures, it is then possible for the remaining brokers to reconstruct a new per-topic publish/subscribe tree that excludes the failed brokers/links to maintain the service operation[27, 79, 80].

The detail of the automatic tree construction/repair is as follows. The system employs link-state protocol among routers[77]. Each broker in the overlay network runs a failure detector protocol for each of its neighbors (e.g., via periodic heartbeat messages). Whenever a broker detects a change of its neighbor's state (e.g., neighbor fails, neighbor recovers), it broadcasts the update message containing its local view to all other reachable brokers. With this mechanism, each broker can maintain a global, up-to-date view of the entire broker network[1]. With the up-to-date global view of the network, each broker can identify the set of *immediately reachable children* of a failed broker along the tree. The immediately reachable children of a broker b is the set of b's next-hop brokers that are available and reachable. For example in Figure 5.2(c), the immediately reachable children set of failed broker $b2$ are $b3$ and $b4$. Hence, using the link-state routing protocol, the broker forwards events to all of the brokers in the failed broker's immediately reachable children set by using point-to-point routing protocol (i.e., *bypass routing*). The same approach applies to subscription forwarding as well. That is, any subscription that is supposed to be sent to a failed broker will be routed to all brokers in the failed broker's immediately reachable children set instead. Figure 5.2 shows an example of the path bypassing scheme.



(a) Pub/sub tree topology

(b) Broker $b2$ fails

(c) Broker $b1$ forwards event to $b2$'s immediately reable children ($b3$ and $b4$)

(d) $b3$ and $b4$ forward event as if it was sent from $b2$

Figure 5.2: Example of path bypassing scheme

With path bypassing mechanism, any pair of brokers is guaranteed to be connected as long as the overlay graph is not partitioned. This mechanism can augment the periodic subscription mechanism

[1]In case of network partition, each broker in a partition will maintain the global view of its partition.

to ensure eventual subscription delivery. The performance analysis of the path bypassing scheme will be given in Section 5.5.4.

Depending on the reliability protocols used, the subscriber reliability will be calculated differently. However, the reliability calculation will be based on the same algorithmic framework. The next section will be a formal definition of subscriber real-time reliability estimation problem with unreliable broker networks.

5.4 Subscriber Real-time Reliability with Unreliable Broker Networks

5.4.1 Formulation

Crash-recovery failure model : As mentioned, we assume each component (broker or link) to be either *on* or *off* over time. A component that is off fails silently and drops any incoming packet. A failed broker will lose all of its routing state information except the data stored in its non-volatile (persistent) storage.

Independent broker/link failure : We assume each broker and link failure to be independent from each other. Several results have shown that such assumption holds in wide-area distributed systems where each broker is not physically co-located[78]. However, different overlay-level links may share some underlying physical-level links, which invalidates the independent assumption[83]. We solve such issue by assuming that the broker overlay graph has been carefully designed to avoid dependency among links by picking only independent links to be in the part of the overlay[75].

Exponentially distributed failure/recovery time : Currently, we assume that each component's time between failures and time to recovery (i.e, times spending in each *on* and *off* state) are exponentially distributed. Previous studies have shown that the assumption of exponential time between failures is true in many distributed systems[84, 73]. While the same set of studies have shown that the exponential distribution assumption does not generally hold for time to repair, we assume it holds in our scenarios for the sake of analysis feasibility. Performance analysis of publish/subscribe systems with non-exponential repair time will be left as future direction of this thesis.

Negligible event queuing/processing/transmission delay : Based on several traces[84, 77, 78], the uptime and downtime durations tend to have the scale of minutes, if not hours or days. Hence, the event transmission, queuing, or processing delay of a normal broker, which have the scale of milliseconds or a few seconds, can be negligible in our analysis. However, as stated in Chapter 3, an overloaded broker can have unbounded queuing delay. We avoid such case in this chapter by assuming some admission control such as the one presented in Section 3.3 to prevent broker overload.

With the assumptions mentioned above, we revise the broker model and link model previously described in Section 2.2 as follows.

5.4.1.1 Broker Model

We assume each broker $b \in \mathbb{B}$ is defined as a tuple

$$(id_b, \gamma_b, \sigma_b)$$

where γ_b and σ_b are exponentially distributed failure rate and repair rate respectively. That is, the broker b has exponentially distributed time between failures and time to repair with mean $\frac{1}{\gamma_b}$ and $\frac{1}{\sigma_b}$ respectively. Hence, we define broker b's availability, denoted by a_b, as the average fraction of time the broker b is on. Hence, b's availability a_b can be calculated as

$$a_b = \frac{\frac{1}{\gamma_b}}{\frac{1}{\gamma_b} + \frac{1}{\sigma_b}}$$

5.4.1.2 Link Model

Similar to the revised broker model, a link $l \in \mathbb{L}$ will have exponentially distributed time between failure and time to repair with rate γ_l and σ_l respectively. Link l's availability value a_l is also calculated as

$$a_l = \frac{\frac{1}{\gamma_l}}{\frac{1}{\gamma_l} + \frac{1}{\sigma_l}}$$

Without loss of generality, we assume that the local link connected between publish/subscriber to its local broker does not fail in order to reflect the fact that intra-domain links have high availability. Our scheme can be simply modified for non-reliable local link scenarios as well.

With the revised broker and link model, we formulate the subscriber real-time reliability estimation as follows.

Subscriber Real-time Reliability Estimation Problem with Unreliable Broker Network: Given an unreliable publish/subscribe overlay network $\mathbb{G} = (\mathbb{N}, \mathbb{L})$ where $\mathbb{N} = \mathbb{B} \cup \mathbb{P} \cup \mathbb{S}$, find the estimated value of r_s, denoted by r'_s, for each mobile subscriber $s \in \mathbb{S}$.

5.4.2 Analytical Framework

The subscriber real-time reliability estimation problem in unreliable broker network can be generally broken down into two sub-problems, which are estimating publisher-subscriber pairwise reliability and estimating publisher-subscriber pairwise flow rate.

Symbol	Definition
γ_x	Component (broker or link) x's failure rate
σ_x	Component (broker or link) x's repair rate
a_x	Component (broker or link) x's availability
λ_{ps}	Average pairwise traffic flow rate between publisher p and subscriber s
δ_{ps}	The path connecting publisher p and subscriber s
$\|\delta_{ps}\|$	The length of path δ_{ps}
d_{ps}	Total end-to-end delay (including failure) between publisher p and subscriber s
D	The lifetime of all events (constant)
d_b	Event delay at broker b (distribution)
d_b^+	Conditional event delay at broker b (distribution)
Q	Transition rate matrix between two successive brokers
$\delta_{ps}^{(i)}$	The i^{th} disjoint paths between p's local broker and s's local broker
PR_x	Component x's period (mean failure-repair cycle)
$MTBF_x$	Component x's mean time between failures
$MTTR_x$	Component x's mean time to repair

Table 5.1: Overlay dynamism analysis variables' notation

5.4.2.1 Publisher-Subscriber Pairwise Reliability

The publisher-subscriber pairwise reliability is the probability that a publisher's event of a subscriber's interest will be delivered to that subscriber before its expiration time. We use the notation $r'_{ps} \in [0, 1]$ to denote the pairwise reliability between publisher $p \in \mathbb{P}$ and subscriber $s \in \mathbb{S}$. As mentioned, the pairwise reliability depends on the reliability protocol used in the publish/subscribe system. Section 5.5 will present the calculation or pairwise reliability for each reliability protocol discussed in Section 5.3.

5.4.2.2 Publisher-Subscriber Pairwise Flow Rate

The publisher-subscriber pairwise flow rate is the average event traffic flow rate from a publisher to a subscriber when no failure occurs. The publisher-subscriber pairwise flow rate between a publisher $p \in \mathbb{P}$ and a subscriber $s \in \mathbb{S}$ can be calculated using Equation (3.13) described in Section 3.2.2.4.

5.4.3 Generic Estimation Algorithm

Subscriber s's real-time reliability r_s is the probability that s will receive an event of its interest successfully before the event's deadline. Hence, the estimated value of r_s, denoted by r_s can be calculated as the weighted average of publisher-subscriber pairwise reliability between each publisher to that subscriber, with the weight equal to the pairwise event flow rate from the corresponding publisher to that subscriber. That is,

$$\begin{aligned} r'_s &= \frac{\text{E[rate of events delivered on time to } s \text{]}}{\text{E[total rate of events of } s\text{'s interest]}} \\ &= \frac{\sum_{p \in \mathbb{P}} r'_{ps} . \lambda_{ps}}{\sum_{p \in \mathbb{P}} \lambda_{ps}} \end{aligned} \tag{5.1}$$

5.5 Publisher-Subscriber Pairwise Reliability Estimation

This section proposes an analytical model to calculate publisher-subscriber pairwise reliability for each different fault tolerance/recovery protocols presented in Section 5.3.

5.5.1 Static Tree

Without any reliable mechanism, the subscription information stored at each broker about the subscriber will be eventually lost when that broker fails. If a subscriber does not have reliable subscription or periodic subscription mechanisms, its subscription along the routing path will be eventually lost, preventing any subsequently published event to be delivered to the subscriber. Hence, the steady-state pairwise reliability for bare-bone publish/subscribe will be zero (i.e., $r'_{ps} = 0$).

5.5.2 Static Tree + Periodic Subscription

With the use of periodic subscription (i.e., Section 5.3.1), the system is guaranteed to eventually recover its routing information to the correct state once the system recovers from its failure. However, without the path bypassing scheme, the per-topic broker tree is fixed. Hence, If brokers fail, the tree is partitioned until all the failed brokers are successfully repaired. During the time the tree is partitioned, a subscriber s will only receives the messages from the publishers in the same partition. All events published at other partitions will be considered lost to that subscriber s.

To analyze the pairwise reliability of each publisher-subscriber pair in static broker tree, We use the concept of *path*. Let δ_{ps} denote the path connecting a publisher p to a subscriber s. We define *path length*, denoted by $|\delta_{ps}|$, as the number of brokers in the path. Hence, a path δ_{ps} can be expressed as the sequence $(p, l_0, b_0, l_1, b_1, ..., l_{|\delta_{ps}|-1}, b_{|\delta_{ps}|-1}, l_|\delta_{ps}|, s)$. Figure 5.3 shows an example of a path of length 3.

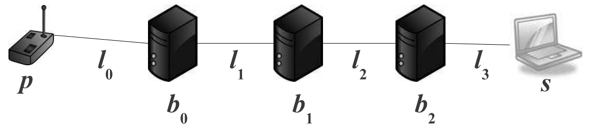


Figure 5.3: Example of a publisher-subscriber path with length 3

Since we consider the static, per-topic broker tree, there is only one unique path δ_{ps} used between each publisher-subscriber pair (p, s). Such path δ_{ps} can be identified from the overlay graph $\mathbb{G} = (\mathbb{N}, \mathbb{L})$ and subscriber's topic τ_s. If the path δ_{ps} is disconnected[2], any subsequent packets transmitted over that path will be lost until the path is connected again. As we consider static broker tree, there is no recovery mechanism to choose a new path. Assuming an event's arrival time is independent from the system state, we can calculate r'_{ps} for static tree with periodic subscription mechanism as the fraction of time that the path is connected as follows.

$$\begin{aligned} r'_{ps} &= \text{P[path } \delta_{ps} \text{ is connected]} \\ &= a_{b_0} \Pi_{i=1}^{|\delta_{ps}|-1} a_{l_i}.a_{b_i} \end{aligned} \tag{5.2}$$

where a_x is the availability of component (broker or link) x.

The overhead of the static tree and periodic subscription scheme depends on the frequency of the subscription renewal at each subscriber. Increasing subscription renewal frequency results in quicker system convergence but also incurs higher overhead. In our scenarios, component failure

[2]Without loss of generality, we assume that the first link (i.e., publisher-broker local link l_0) and the last link (i.e., broker-subscriber local link $l_{|\delta_s^p|}$) do not fail.

periods are in the scale of several minutes to a few hours. Hence, we set the subscription renewal period to be in the scale of several seconds to one minute, which renders its effect to the protocol performance negligible. The problem of finding the optimal subscription renewal frequency has been investigated by several existing works[27, 46] and is beyond the scope of this thesis.

5.5.3 Static Tree + Event Buffering/Retransmission

With the reliable acknowledgment protocol (i.e., Section 5.3.2) in static per-topic tree, an event of a subscriber s's interest that is published by a publisher p will be eventually delivered to s, given that p's local broker is available when p publishes the event (since we assume that p does not have retransmission capability). This is because the event will always be buffered at some broker along the path between p and s, even when the path is disconnected[3]. The event will then be forwarded when the next-hop broker and link are available, and eventually delivered to the subscriber. However, the delay the event spends in the buffer may be longer than its lifetime, which results in late delivery.

To analyze the pairwise reliability r'_{ps} between a publisher p and a subscriber s under static tree with event buffering scheme, consider the single, unique path δ_{ps} connecting p and s. Assuming the event arrival time to be independent from the path δ_{ps}'s state, we estimate the path real-time reliability as follows.

$$\begin{aligned} r'_{ps} &= \text{P[an event from } p \text{ arrives at } s \text{ before the deadline]} \\ &= \text{P[}p\text{'s local broker is on].P[end-to-end delay less than event lifetime]} \\ &= a_{b_0}.\text{P}[d_{ps} < d_e] \end{aligned} \tag{5.3}$$

where d_{ps} is the end-to-end delivery delay and d_e is the event lifetime. Without loss of generality, we assume that d_e is a globally defined constant D for all events. Thus, it is necessary to calculate the end-to-end delivery delay d_{ps} first in order to estimate path reliability r'_{ps}.

Under the static per-topic broker tree with event buffering scheme, the path end-to-end delivery delay d_{ps} of a path δ_{ps} can be broken down into *link delay* (i.e., the time to transmit an event over each link), *broker queuing/processing* delay (i.e., the time the event is queued and processed

[3]In the analysis, we assume each broker to have unbounded buffer such that it can always store any incoming event.

at each broker), and *buffering delay* (i.e., the time the event is stored at the broker due to next hop's failure). Link delay and broker queuing/processing delay usually range from milliseconds to a few seconds while buffering delay usually has the scale in minutes (since the broker downtime is in the scale of minutes). Hence, we assume link delay and broker queuing/processing delay are negligible and focus on the buffering delay. To calculate the distribution of d_{ps} for path δ_{ps}, we need to calculate the buffering delay at each broker $b_i(0 \leq i < |\delta_{ps}|)$ in the path. Consider when the event is received successfully at broker b_i and hence broker b_i will try to retransmit the event to broker b_{i+1}. If both link l_{i+1} and broker b_{i+1} are up at the moment, the event will be transmitted successfully to broker b_{i+1} immediately, thus incurring zero buffering delay at broker b_i. However, if either link l_{i+1} or broker b_{i+1} is down at the moment, the event will be buffered at the broker b_i, which will keep retransmitting the event until the event gets through to broker b_{i+1}. The broker b_i discards the event if the event expires. Note that the event will get through only when all b_i, l_{i+1}, and b_{i+1} are up at the same time.

Let d_{b_i} be the buffering delay at each broker $b_i(0 \leq i < |\delta_{ps}|)$. We first calculate the probability that $d_{b_i} = 0$ (i.e., the probability that the event is successfully delivered to b_{i+1} immediately), which can be calculated as

$$\begin{aligned}
\mathrm{P}[d_{b_i} = 0] &= \mathrm{P}[b_i, l_{i+1}, b_{i+1} \text{ are available} | b_i \text{ is available}] \\
&= \mathrm{P}[l_{i+1}, b_{i+1} \text{ are available}] \\
&= a_{l_{i+1}}.a_{b_{i+1}} \qquad (5.4)
\end{aligned}$$

Given that delay is always non-negative, we have

$$\begin{aligned}
\mathrm{P}[d_{b_i} > \ 0] &= 1 - \mathrm{P}[d_{b_i} = 0] \\
&= 1 - a_{l_{i+1}}.a_{b_{i+1}} \qquad (5.5)
\end{aligned}$$

Now, in the case that the buffering delay at each broker b_i is not zero (with probability $1 - a_{l_{i+1}}.a_{b_{i+1}}$), we need to find the delay distribution in such case. Let $d^+_{b_i}$ be the conditional buffering delay at broker b_i under the condition that $d_{b_i} > 0$. Assuming the event arrives at arbitrary time at broker b_i, the conditional buffering delay $d^+_{b_i}$ is equal to the time it takes for the next-hop path to be repaired (i.e., time until l_{i+1} and b_{i+1} are both in *on* state). Assuming each component's time between failure and time to repair to be exponentially distributed, we can calculate such delay distribution by using continuous-time Markov process diagram that represents the state of broker b_i, link l_{i+1}, and b_{i+1}. The diagram is shown in Figure 5.4. Each of 8 states depicts each possible state of sub-path (b_i,l_{i+1},b_{i+1}), with each bit representing each individual component's state (0 = *off*, 1 = *on*). The first bit (least significant bit) represents b_i's state. The second bit represents l_{i+1}'s state. The third bit (most significant bit) represent b_{i+1}'s state. For example, state "011" represents the state where broker b_i is on, link l_{i+1} is on, and broker b_{i+1} is off. Note that in the scenario where an event arrives at broker b_i and needs to be buffered at b_i, an event will find the system state in either state "001", "011", or "101" with probability $\frac{(1-a_{l_{i+1}})(1-a_{b_{i+1}})}{1-a_{l_{i+1}} \cdot a_{b_{i+1}}}$, $\frac{a_{l_{i+1}}(1-a_{b_{i+1}})}{1-a_{l_{i+1}} \cdot a_{b_{i+1}}}$, and $\frac{(1-a_{l_{i+1}})a_{b_{i+1}}}{1-a_{l_{i+1}} \cdot a_{b_{i+1}}}$ respectively. The event will continue to be buffered at broker b_i (note that b_i can also fail but the event is kept in its non-volatile storage) until the state becomes "111", which the event will be transmitted to broker b_{i+1} successfully. Hence, the diagram depicts the absorbing Markov process with three start states = "001", "011", "101" and one absorbing state "111"with the corresponding transition rate matrix $\dot{Q}$ as

$$\dot{Q} = \begin{pmatrix} -\dot{q}_0 & \sigma_{b_i} & \sigma_{l_{i+1}} & 0 & \sigma_{b_{i+1}} & 0 & 0 & 0 \\ \gamma_{b_i} & -\dot{q}_1 & 0 & \sigma_{l_{i+1}} & 0 & \sigma_{b_{i+1}} & 0 & 0 \\ \gamma_{l_{i+1}} & 0 & -\dot{q}_2 & \sigma_{b_i} & 0 & 0 & \sigma_{b_{i+1}} & 0 \\ 0 & \gamma_{l_{i+1}} & \gamma_{b_i} & -\dot{q}_3 & 0 & 0 & 0 & \sigma_{b_{i+1}} \\ \gamma_{b_{i+1}} & 0 & 0 & 0 & -\dot{q}_4 & \sigma_{b_i} & \sigma_{l_{i+1}} & 0 \\ 0 & \gamma_{b_{i+1}} & 0 & 0 & \gamma_{b_i} & -\dot{q}_5 & 0 & \sigma_{l_{i+1}} \\ 0 & 0 & \gamma_{b_{i+1}} & 0 & \gamma_{l_{i+1}} & 0 & -\dot{q}_6 & \sigma_{b_i} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad (5.6)$$

where γ_x and σ_x are component x's exponential failure rate and exponential repair rate described in Section 5.4.1, and $\dot{q}_i$ is state i's total outgoing rate. For example, $\dot{q}_0 = (\sigma_{b_i} + \sigma_{l_i} + \sigma_{b_{i+1}})$. Thus, the conditional buffering delay at broker b_i is equal to the time to absorption of the absorbing matrix $\dot{Q}$, which is a phase-type distribution[85] and can be calculated by breaking down the matrix $\dot{Q}$ in to the form of

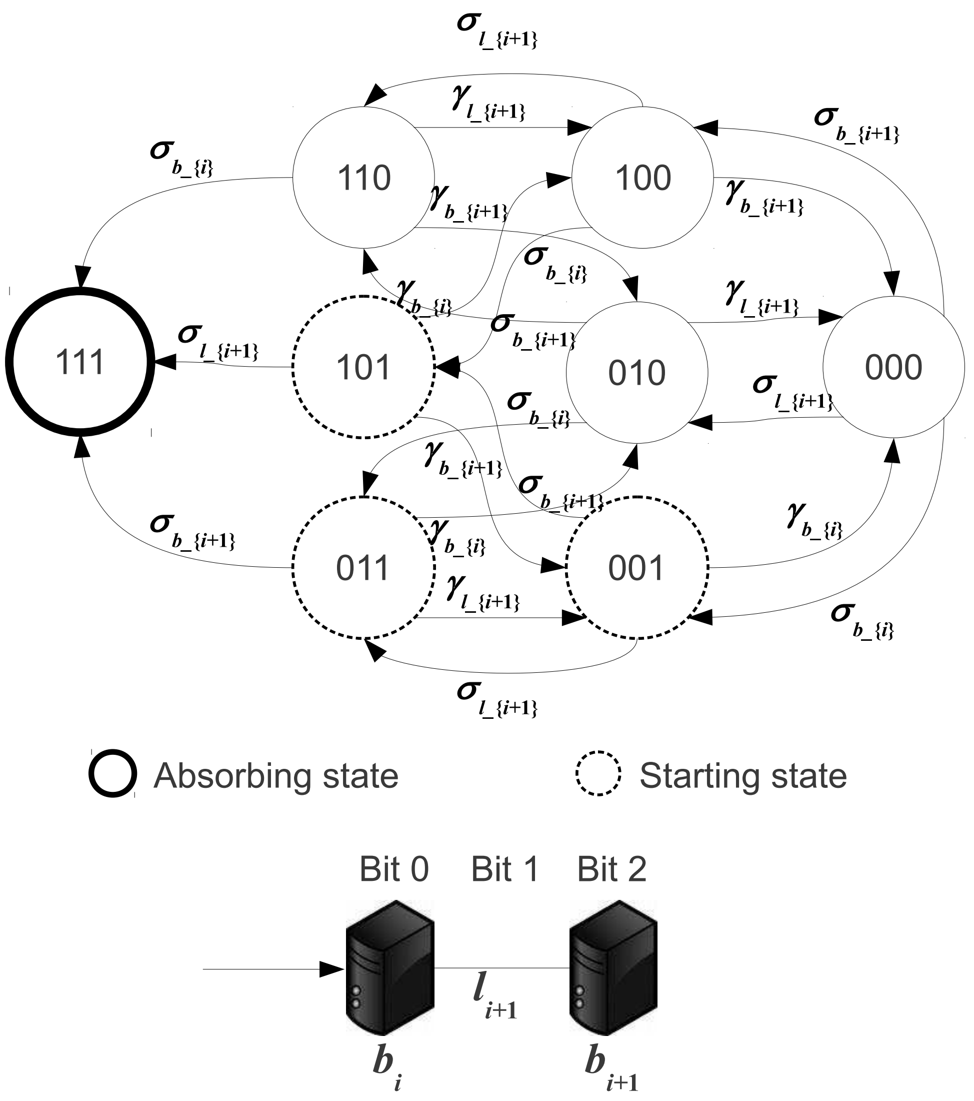


Figure 5.4: 8-state continuous, absorbing Markov process diagram for per-hop buffering delay analysis

$$\dot{Q} = \left(\begin{array}{c|c} \dot{S} & \dot{\mathbf{S}}^0 \\ \hline \mathbf{0} & 0 \end{array} \right) \tag{5.7}$$

Where $\dot{S}$ and $\dot{\mathbf{S}}^0$ are the 7x7 top-left sub-matrix and the 7x1 top-right sub-vector of $\dot{Q}$ defined in Equation (5.6) respectively. Hence, the cumulative distribution of $d_{b_i}^+$ can be calculated as

$$\mathrm{P}[d_{b_i}^+ < t] = 1 - \alpha.\exp(\dot{S}t)\mathbf{1} \tag{5.8}$$

where $\exp(\dot{S})$ is the matrix exponential[86] of $\dot{S}$, α is the 1x7 starting state vector

$$\alpha = \begin{pmatrix} 0 & \frac{(1-a_{l_{i+1}})(1-a_{b_{i+1}})}{1-a_{l_{i+1}}.a_{b_{i+1}}} & 0 & \frac{a_{l_{i+1}}(1-a_{b_{i+1}})}{1-a_{l_{i+1}}.a_{b_{i+1}}} & 0 & \frac{(1-a_{l_{i+1}})a_{b_{i+1}}}{1-a_{l_{i+1}}.a_{b_{i+1}}} & 0 \end{pmatrix}$$

and $\mathbf{1}$ is an 7x1 vector with every element being 1.

Thus, with Equation (5.8), we can calculate the distribution of conditional buffering delay $d_{b_i}^+$ at broker b_i. Hence, we can estimate the buffering delay d_{b_i} at broker b_i as

$$d_{b_i} = \begin{cases} d_{b_i}^+ & \text{with probability } 1 - a_{l_{i+1}}.a_{b_{i+1}} \\ 0 & \text{with probability } a_{l_{i+1}}.a_{b_{i+1}} \end{cases} \tag{5.9}$$

Once we calculate per-hop buffering delay d_{b_i} with Equation (5.9), we then can calculate the end-to-end buffering delay d_{ps} for path $\delta_{ps} = (p, l_0, b_0, l_1, b_1, ..., l_{|\delta_{ps}|-1}, b_{|\delta_{ps}|-1}, l_{|\delta_{ps}|}, s)$ as

$$d_{ps} = \sum_{i=0}^{|\delta_{ps}|-1} d_{b_i} \tag{5.10}$$

Hence, Equation (5.10) completes the calculation of pairwise reliability for static per-topic tree with event buffering scheme in Equation (5.3).

The overhead of the static tree with event buffering/retransmission comes in the forms of buffer overhead at each broker, and retransmission traffic. To calculate the maximum buffer size at a broker b, denoted by BS_b, we need to first calculate the maximum buffer size that broker b requires to store the events to be forwarded to its neighbor b'. Let $BS_{bb'}$ denote such per-neighbor buffer. $BS_{bb'}$ can be calculated as

$$BS_{bb'} = \lambda_{bb'}.\min(d^{+}_{bb'}, d_e)$$

where $\lambda_{bb'}$ is the traffic rate from broker b to b' and can be calculated by techniques in Chapter 3 (i.e., Section 3.2.2.2), $d^{+}_{bb'}$ is the conditional buffering delay calculated from Equation (5.8), and d_e is the event lifetime. Once the per-neighbor buffer overhead is calculated, we can calculate the total maximum buffer overhead at broker b as

$$BS_b = \sum_{b':(b,b')\in\mathbb{L}} BS_{bb'}$$

The retransmission overhead can be calculated per event. For each event transmitted along the path δ_{ps}, the number of retransmission is equal to $\frac{\min(d_{ps},d_e)}{\Delta_T}$ where Δ_T is the retransmission period[4].

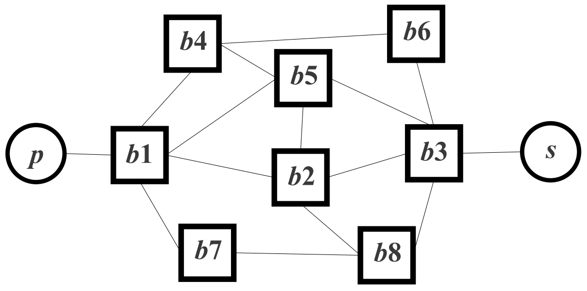
5.5.4 Path Bypassing + Periodic Subscription

With the path bypassing scheme discussed in Section 5.3.3, a new path will be generated between p and s if the old path fails. Hence, an event of a subscriber s's interest that is published by a publisher p will be delivered to s as long as the entire overlay graph $\mathbb{G}$ is not disconnected between p and s. Thus, pairwise reliability r'_{ps} is then the graph $\mathbb{G}$'s connection probability between p and s. However, the calculation of such connection probability for any generic overlay graph is considered to be a #P-complete problem[87], which has higher complexity that a NP-complete problem.

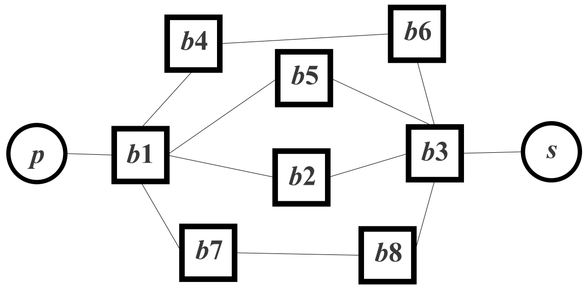
Due to such computational complexity, we propose an algorithm to approximate the lower bound of graph $\mathbb{G}$'s connection probability between any publisher-subscriber pair (p, s) by constructing a subgraph $\mathbb{G}' \subseteq \mathbb{G}$ that consists only parallel, broker-disjoint paths between p's local broker and s's local broker (for example, see Figure 5.5).

That is, the multi-path subgraph $\mathbb{G}'$ contains multiple, broker-disjoint path between p's lo-

[4]In our simulation, we assume the use of failure detector, which will notify each broker when its failed neighbors recover. The broker then retransmits pending events once a failed neighbor recovers.



(a) Exact broker overlay graph $\mathbb{G}$



(b) Subgraph $\mathbb{G}'$ for r'_{ps} approximation with 4 disjoint paths

Figure 5.5: Example of reduced subgraph to estimate the lower bound of pairwise reliability r'_{ps}

cal broker and s's local broker, assuming there are m of such paths in subgraph $\mathbb{G}'$, namely $\delta_{ps}^{(0)}, \delta_{ps}^{(1)}, ..., \delta_{ps}^{(m-1)}$ where

$$\delta_{ps}^{(i)} = (p, l_0^{(i)}, b_0^{(i)}, l_1^{(i)}, b_1^{(i)}, ..., l^{(i)}_{|\delta_{ps}^{(i)}|-1}, b^{(i)}_{|\delta_{ps}^{(i)}|-1}, l^{(i)}_{|\delta_{ps}^{(i)}|}, s)$$

Note that $b_0^{(i)}$ refers to the same broker for all $0 \leq i < m$, which is publisher p's local broker. Likewise, $b^{(i)}_{|\delta_{ps}^{(i)}|-1}$ refers to the same broker, which is the subscriber s's local broker as well. Let b_0 and $b_{|\delta_{ps}|-1}$ denote publisher p's local broker and subscriber s's local broker respectively. Hence, the pairwise reliability r'_{ps} between publisher p and subscriber s in dynamic tree scheme is approximated as

$$\begin{aligned}
r'_{ps} &= \text{P}[\mathbb{G} \text{ is connected between } p \text{ and } s] \\
&\geq \text{P}[\mathbb{G}' \text{ is connected between } p \text{ and } s] \\
&\geq \text{P}[p\text{'s local broker is on}].\text{P}[s\text{'s local broker is on}].\text{P}[\text{at least one path is connected}] \\
&\geq a_{b_0}.a_{b_{|\delta_{ps}|-1}}.(1 - \Pi_{i=0}^{m-1}(1 - \frac{r_{ps}^{(i)}}{a_{b_0}.a_{b_{|\delta_{ps}|-1}}})) \qquad (5.11)
\end{aligned}$$

where $r_{ps}^{(i)}$ is the pairwise reliability of each path $\delta_{ps}^{(i)}$ in subgraph $\mathbb{G}'$, which can be calculated by Equation (5.2).

The overhead of the path bypassing scheme depends on the overhead of the failure detector and link-state protocol that updates system topology view at each broker. The overhead of the failure detector comes in the form of periodic heartbeat message among neighbors. Hence, the failure detector control message rate is equal to $\frac{2.|\mathbb{L}|}{\Delta_f}$, where Δ_f is the failure detector heartbeat message retransmission interval. The overhead of the link-state protocol occurs in the form of view update packets that are generated from each broker to all other brokers whenever a network topology change is detected. For every change in the network topology view, the link-state protocol incurrs $|\mathbb{B}|^2$ view update packets to the system. Hence, the total view update packet generation rate is equal to

$$|\mathbb{B}|^2.(\sum_{b\in\mathbb{B}}(\gamma_b+\sigma_b)+\sum_{l\in\mathbb{L}}(\gamma_l+\sigma_l))$$

5.5.5 Path Bypassing + Event Buffering

The path bypassing scheme guarantees that an event from a publisher will be delivered to a subscriber as long as there is at least one path between them. However, it cannot handle network partitions. In such case, we can combine the path bypassing scheme with the event buffering scheme to ensure that an event will be bufferred during network partition and delivered eventually after the partition is repaired. The detail of the scheme is as follows. Each broker uses the event acknowledgement/buffering scheme as mentioned in Section 5.5.3. When a broker b_1 detects its neighbor b_2's failure, it uses the bypass routing *without* acknowledgement to forward the event to the failed broker b_2's immediately reachable children. The broker b_1 also keeps the event in its buffer and keeps retransmitting the event to the failed broker b_2 until b_2 recovers, receives the event, and sends the acknowledgement back to b_1. b_1 then discards the event. This scheme combines eventual delivery guarantee of the retransmission scheme with timeliness delivery of the path bypassing scheme. That is, an event is guaranteed to progress along the per-topic dissemination tree while all the currently reachable subscribers will receive the event immediately without buffering delay. The drawback of this approach is the additional overhead and potential event duplication at the subscribers. Event duplication, however, can be filtered out at the last-hop broker.

We can calculate the publisher-subscriber pairwise reliability for the path bypassing with event buffering scheme as follows. Let r'^A_{ps} be the estimated publisher-subscriber pairwise reliability for the path bypassing scheme (i.e., Equation (5.11) and d_{ps} be the end-to-end buffering delay for the event buffering scheme (i.e., Equation (5.10)). We can calculate the estimated pairwise reliability for the combined scheme as

$$\begin{aligned} r'_{ps} &= \text{P[event delivered immediately]} + \text{P[partition].P[event delivered on time]} \\ &= r'^{A}_{ps} + (1 - r'^{A}_{ps}).\frac{\text{P}[d_{ps} \leq d_e]}{\text{P}[d_{ps} > 0]} \end{aligned} \qquad (5.12)$$

where d_e is event lifetime. That is the total reliability is the probability that either the event can be delivered immediately via automatic tree scheme, or the event suffers network partition but the buffering delay is still less than the event lifetime.

Note that estimated pairwise reliability r'_{ps} can be either calculated by Equation (5.2), Equation (5.3), Equation (5.11), or Equation (5.12), depending on the reliable protocol used. Once the estimated reliability r'_{ps} values of all publisher-subscriber pairs are calculated, they can be used to calculate the estimated subscriber reliability r'_s using Equation (5.1).

5.6 Home Broker Selection Optimization

Based on the overlay dynamism analytical model proposed in Section 5.4, several performance optimization/improvement can be achieved, such as broker quality planning, broker overlay planning, or reliability protocol selection. In this section, we present one simple performance optimization/improvement called *home broker selection* optimization. The subscriber allocation problem is motivated from the fact that a joining subscriber usually has the freedom to choose its home broker from a subset of brokers. The subset of brokers which a subscriber can join can be calculated from various QoS or locality constraints. Hence, a subscriber should choose its home broker such that the subscriber gets the best reliability.

Based on the proposed model, home broker selection algorithm is straightforward as follows. Consider a new subscriber s that wants to join a publish/subscribe overlay graph $\mathbb{G} = (\mathbb{N}, \mathbb{L})$ where $\mathbb{N} = \mathbb{P} \cup \mathbb{B} \cup \mathbb{S}$. Let $B(s) \subseteq \mathbb{B}$ be the home broker candidate set that a new subscriber s can choose. For each broker $b \in B(s)$, we construct a new graph $\mathbb{G}' = (\mathbb{N} \cup \{s\}, \mathbb{L} \cup \{(b, s)\})$. That is, the graph $\mathbb{G}'$ represents the new overlay when s chooses b as its home broker. Hence, for each broker $b \in B(s)$, we generate the corresponding graph $\mathbb{G}'$ and calculate the estimated reliability r'_s of subscriber s out of $\mathbb{G}'$. The new subscriber s then chooses to associate with broker b such that its corresponding graph gives the highest value of r'_s.

5.7 Evaluation

We evaluate the proposed analytical model via simulations with NS-2 network simulator[50]. All the event-related and subscription-related simulation parameters are the same with the simulation from Chapter 3. However, we assume the broker to have unlimited processing capability in this simulation to hide the effect of event queuing/processing delay and focus on event delay/loss caused by unreliable broker networks.

5.7.1 Parameter Settings

We investigated several host availability traces and reports, ranging from commercial server log to distributed testbed log[73, 84, 78, 72]. In most cases, server's time between failures tends to range from several days to weeks, while time to repair usually range within hours. Also, high-end and well-maintained commercial servers usually have availability more than 0.99 while standard, off-the-shelf server under low-to-moderate maintenance barely have availability a little more than 0.9. Meanwhile, link availability tends to have shorter time between failures and time to repair when compared to host availability[73].

Motivated by such finding, we describe a component[5] from availability perspective by two metrics, *period* and *availability*. We define the term *period* of a component as the summation of the component's mean time between failure and the components mean time to repair (i.e., mean failure-repair cycle length) and the term *availability* as the fraction of time the component is on. Thus, given a component x's period PR_x and availability a_x, we can calculate x's mean time between failures $MTBF_x$ and mean time to repair $MTTR_x$ as

$$MTBF_x = a_x.PR_x$$

$$MTTR_x = (1 - a_x).PR_x$$

In the simulation, a component will switch its state between *on* and *off* state. A component

[5]A component means a link or a broker

x will be on for the time period which is drawn from the exponential distribution with mean $MTBF_x$ before going to *off* state. Likewise, the component x will then be off for the time period drawn from the exponential distribution with mean $MTTR_x$ before going to *on* state again. Thus, such component x will have exponential failure rate $\gamma_x = \frac{1}{MTBF_x}$ and exponential repair rate $\sigma_x = \frac{1}{MTTR_x}$.

We run the simulation with different broker network topologies. We vary the number of brokers in the system from 3 to 10. Unless specified otherwise, each broker's default period is 60,000 seconds (16.67 hours). We vary each component's availability from 0.9 to 0.999. Based on the previous work[88], we set each overlay link's availability set to 0.99 and period to 60,000 seconds. Each publisher has default publishing interval equal to 1 minute. Each event has default lifetime equal to 3,600 seconds (1 hours). Each simulation is run for 14 days of simulation time. The evaluation result of each simulation parameter set is averaged from 10 runs.

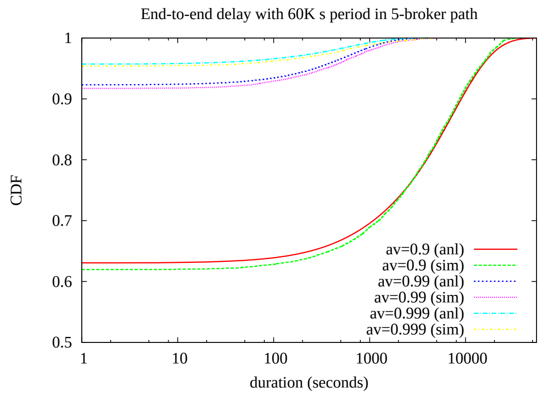
5.7.2 Evaluation Results

We implement three different reliability mechanisms described in Section 5.3 and analyzed in Section 5.5. The three different mechanisms are static tree with periodic subscription, static tree with event buffering, and dynamic tree with periodic subscription. We run the simulation with different broker overlay graph topologies in order to validate various aspect of our proposed analytical model.

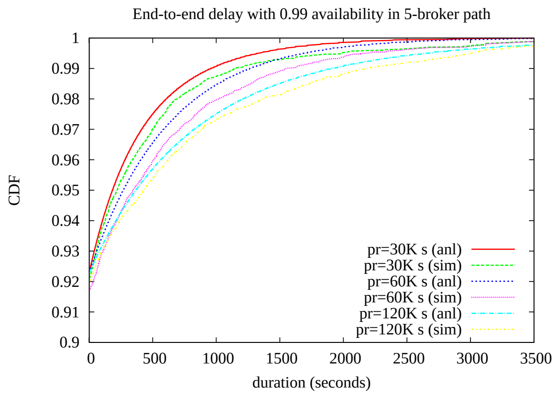
5.7.2.1 Straight Line Topology

We start to evaluate our model with straight line topology where the overlay graph contains one publisher and one subscriber connected to each other via a single path of brokers (e.g., Figure 5.3). In this setting, in order to study the effect of path length, all brokers and links have the same period and availability value.

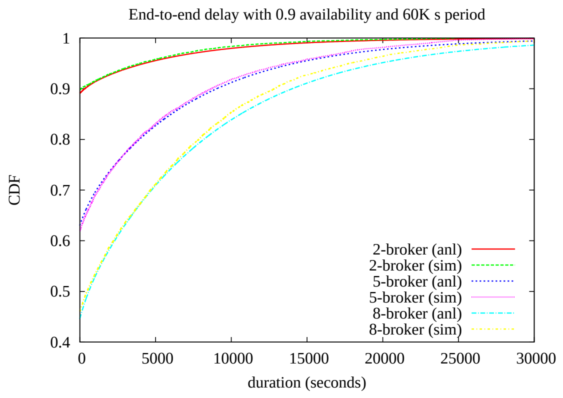
We first evaluate the result of event-buffering scheme. Figure 5.6 shows the cumulative distribution of end-to-end buffering delay for event-buffering scheme (i.e., d_{ps} analyzed in Equation (5.10) of Section 5.5.3) while Figure 5.7 shows the cumulative distribution of end-to-end conditional delay (i.e, the distribution of d_{ps}, given that $d_{ps} > 0$). Notice that our proposed analytical model could predict the end-to-end buffering delay and conditional buffering delay accurately for any publisher-subscriber path when event buffering scheme is used. Figure 5.8 shows the average subscriber real-time reliability in straight line topology with different settings. Again, our proposed analytical model could predict the results accurately.



(a) 60K s period, 5-broker



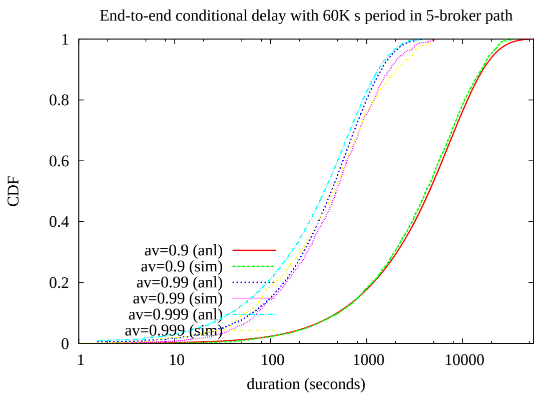
(b) 0.99 availability, 5-broker



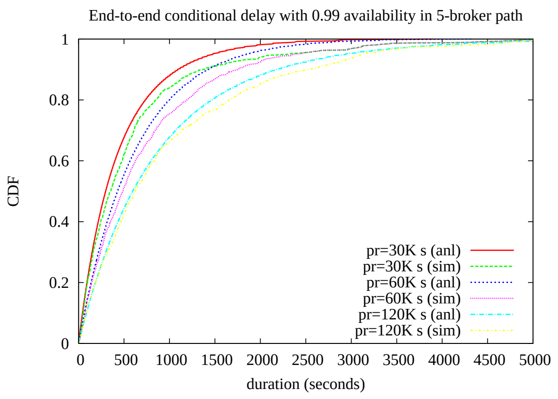
(c) 0.9 availability, 60K s period

Figure 5.6: End-to-end buffering delay distribution of a single publisher-subscriber path

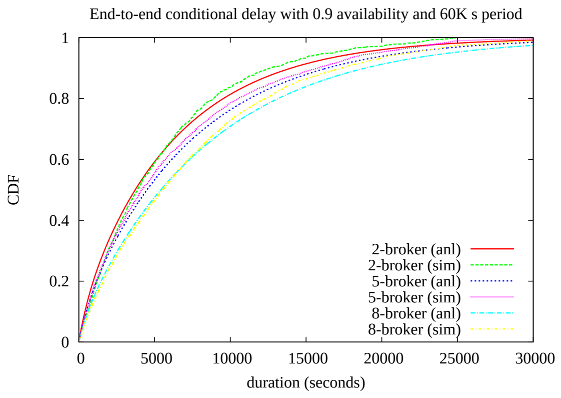
Periodic subscription VS Event buffering : Figure 5.8(a) and Figure 5.8(c) shows performance comparison between periodic subscription scheme and event buffering scheme. Note that under the same setting, event buffering scheme usually achieves higher path reliability when compared to the periodic subscription scheme. This is not surprising, as the event buffering scheme keeps the event within the buffer when the path is disconnected while the periodic subscription scheme just drops the event. Note that the performance gain from the event buffering scheme also depends on the event lifetime. If each event has very short lifetime, the event buffering scheme would achieve the same performance to the periodic subscription scheme. On the other hand if each event has very long lifetime, the event buffering scheme would achieve much higher performance than the periodic subscription scheme.



(a) 60K s period, 5-broker



(b) 0.99 availability, 5-broker



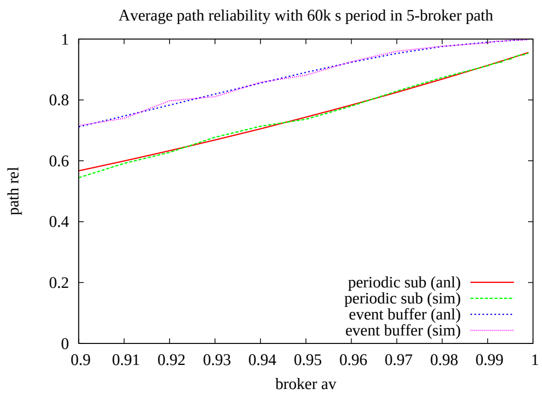
(c) 0.9 availability, 60K s period

Figure 5.7: Conditional end-to-end buffering delay distribution of a single publisher-subscriber path

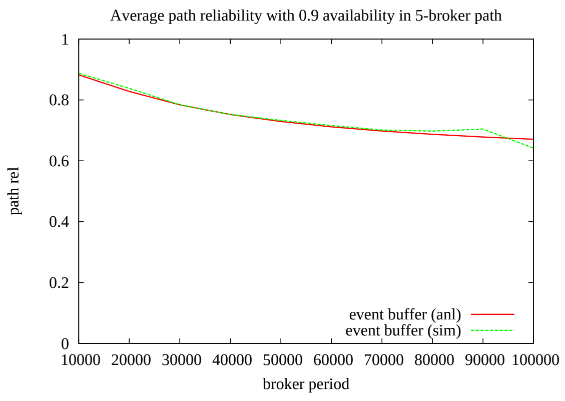
Effect of path length : As shown in Figure 5.6(c), and Figure 5.8(c), increasing path length generally results in lower subscriber reliability. This is obvious as adding a broker into the path will increase the chance that the path will fail. It can be thought as a tradeoff between reliability and scalability (e.g., to increase the system size to support more clients).

Effect of availability : According to the result shown in Figure 5.8(a), Increasing each component's availability will improve the performance of both periodic subscription scheme and event buffering scheme. It also decreases the end-to-end buffering time as shown in Figure 5.6(a).

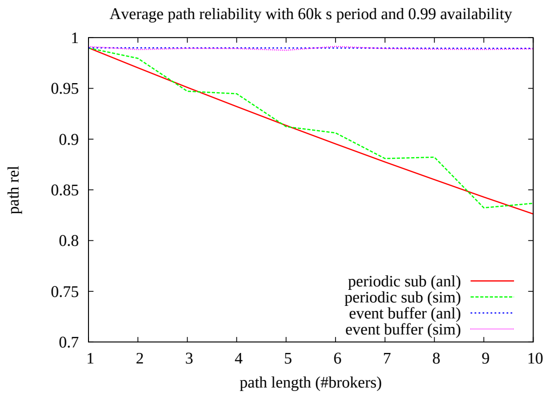
Effect of period length : While the period length does not have the effect of periodic subscription scheme, it affect the performance of event buffering scheme in static tree scenario. As shown in Figure 5.6(b) and Figure 5.7(b), decreasing period length will also decrease the event buffering time in general. This implies that if the two brokers have the same availability level, it is preferable to select a broker that has shorter period, if the event buffering scheme is used.



(a) 60K s period, 5-broker



(b) 0.9 availability, 5-broker



(c) 0.99 availability, 60K s period

Figure 5.8: Average reliability a single publisher-subscriber path

5.7.2.2 Static Random Tree Topology

To study the performance of static-tree reliability schemes (i.e., static tree with periodic subscriptions, and static tree with event buffering), we generate a random broker tree consisting of 10 brokers, 10 publishers, and 500 subscribers. We randomly assign publishers and subscribers to each broker. We assume all publishers and subscribers share the same topic, but each subscriber may subscribe to different content space. Each broker and link has availability value uniformly distributed within [0.9,0.999] range. We perform 10 runs per each simulation setting. We assume perfect pairwise flow estimation between each publisher-subscriber pair. We divide all generated trees into four sets. The first set of trees has each broker availability falling into [0.9,0.95] range and use periodic subscription scheme. The second set of trees has each broker availability falling into [0.9,0.95] range and use event buffering scheme. The third set of trees has each broker availability falling into [0.99,0.999] range and use periodic subscription scheme. The fourth set of trees has each broker availability falling into [0.99,0.999] range and use event buffering scheme. The [0.9,0.95] availability range represents standard, off-the-shelf servers with low-to-moderate maintenance[73]. The [0.99,0.999] availability range represents high-end, commercial servers with high maintenance[72].

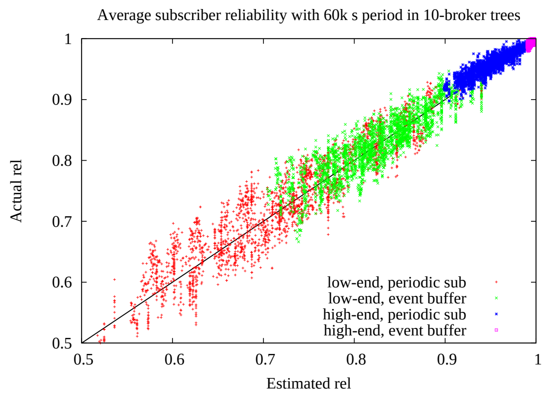


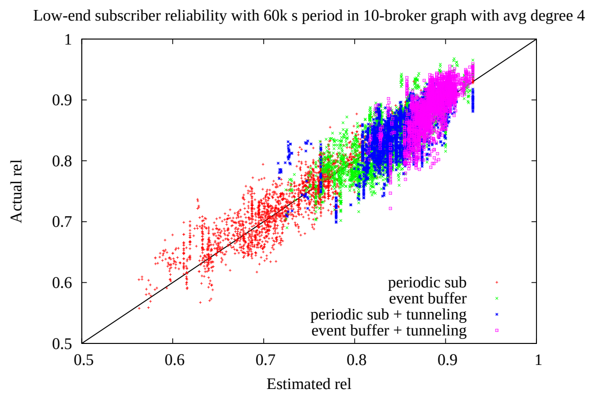
Figure 5.9: Subscriber reliability in static random tree with 10 brokers and 60K s period

As shown in Figure 5.9, there is a clear distinction of reliability value between different groups of tree configuration. The group with lowest reliability is the low-end servers with periodic subscription scheme, followed by the low-end servers with event buffering scheme. Notice that , again, event buffering scheme could achieve high reliability than the periodic subscription scheme, although the performance gain effect may be less, compared to the performance gained from the server's quality.

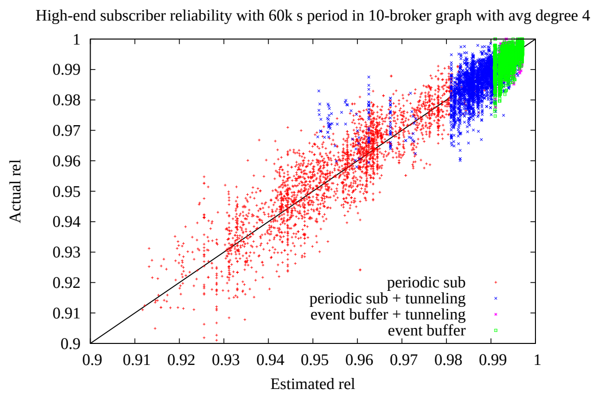
5.7.2.3 Random Graph Topology

We generate a random graph consisting of 10 brokers, 10 publishers, and 500 subscribers. Publishers and subscribers are randomly assigned to each broker. Again, we run the simulations with two broker availability specifications named *low-end* (0.9 - 0.95 availability) and *high-end* (0.99 - 0.999 availability). We compare the performance in terms of subscriber reliability among four publish/subscribe modes discussed in Section 5.4.

Figure 5.10 shows the performance comparison between the four protocols in 10-broker overlay graph. In the low-end broker configuration, the path bypassing scheme with event buffering has the best performance, followed by the path bypassing scheme with periodic subscription, the static tree scheme with event buffering, and the static tree scheme with periodic subscription respectively. However, in the high-end broker configuration, the static tree scheme with event buffering performs best and as well as the path bypassing scheme with event buffering. This finding suggests that one



(a) Low-end brokers



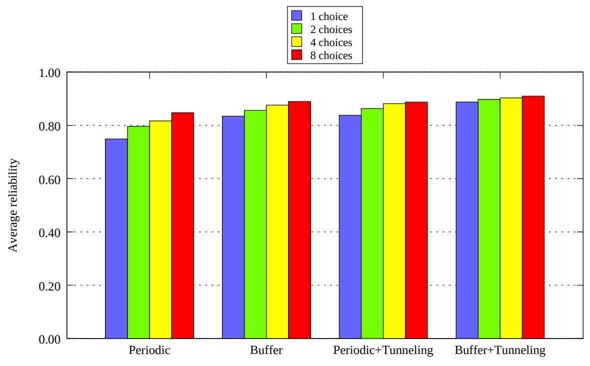
(b) High-end brokers

Figure 5.10: Subscriber reliability in 10-broker overlay graph with average degree 4 and 60K s period

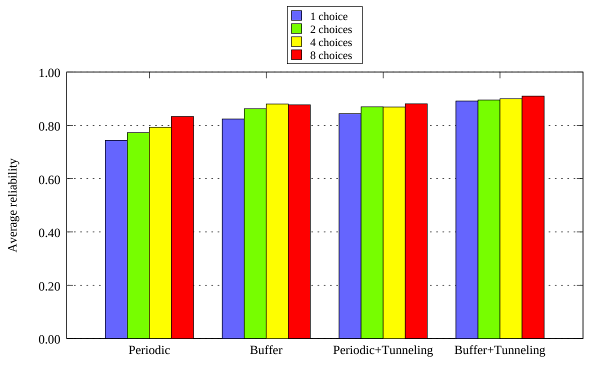
should prefer to use the static tree scheme with event buffering in high-end broker configuration, as it has lower overhead than the path bypassing scheme with event buffering.

5.7.2.4 Home Broker Selection

Figure 5.11 presents the evaluation result of the home broker selection optimization proposed in Section 5.6 with 10-broker overlay graph where each broker has availability between 0.9 and 0.95. Each of the group in the graphs represents each reliability technique used. The broker candidate set for each subscriber is randomly chosen from the total broker set. We vary the number of broker candidates for each subscriber from 1 to 8. Each configuration is simulated for five runs before the average subscriber reliability for that configuration is calculated.



(a) Average broker degree = 4



(b) Average broker degree = 9

Figure 5.11: Average subscriber reliability with home broker selection scheme in 10-broker overlay graph

From Figure 5.11, we can conclude that the more home broker candidates a subscriber has, the more average real-time reliability it will get. The performance gain is significant when the periodic subscription mode or the event buffering mode is used. In contrast, the performance gain is marginal when the overlay graph has higher degree or when the redundant path scheme is used, as each broker has high number of connecting paths already.

5.8 Related Work

5.8.1 Reliable Overlay Networks

Improving reliability, timeliness, and other quality of service metrics in wide-area overlay networks have been a significant topic of researchs for many years[77, 89, 90, 91]. Most approaches in this category deploys a set of application-level routers, which exchange link information with each other and construct end-to-end path quality metrics in order to select the best path between any pair of such routers. However, the approaches in this category are designed for point-to-point routing and do not specifically address decoupling nature between publishers and subscribers in publish/subscribe systems.

5.8.2 Reliable Multicast Systems

Reliable multicast has been used as a tool to disseminate information from one source to several receivers. Over the past few years, several works have proposed reliable multicast schemes on top of overlay networks[92, 93]. The one-to-many communication nature of multicast makes it suitable for topic-based publish/subscribe systems where the set of subscribers for each topic can be identified. In content-based publish/subscribe systems, however, do not have clear notion of static subscriber group. Hence, content-based publish/subscribe systems cannot directly adopt the standard multicast techniques to disseminate events.

5.8.3 Reliable Publish/Subscribe Systems

There also have been several reliable mechanisms designed to make the publish/subscribe systems fault-tolerant under failures[75, 94, 79]. While several works have discussed the proof of correctness and reliability analysis of the reliable mechanisms[27, 75], they did not provide quantitative analysis in timeliness aspect. This thesis has described several of such techniques in Section 5.3 and also proposed the quantitative analytical model to quantify the performance gained by such schemes. There have been a few works that discuss timeliness of event delivery in publish/subscribe systems under component failures[76, 46]. However, they did not provide analytical model of their proposed systems. Our work is, to the best of our knowledge, combine the conventional reliability analysis with timeliness analysis, which is useful for delay-sensitive publish/subscribe applications over wide-area networks.

5.9 Discussion

In this chapter, we proposed the analytical model to estimate the subscriber real-time reliability with the focus on overlay determinism. We first described broker failure and link failure model before discussing several existing reliability-enhancement for content-based publish/subscribe systems. We then proposed the generic analytical model to estimate subscriber reliability when such reliability techniques are used. The evaluation via simulation has proved the accuracy of our predictive model. Our proposed model can then be used as a building block for optimization problems such as subscriber assignment problem or broker network planning problem.

There are a few possible directions for subscriber reliability analysis over best-effort overlay. First, we assume the component's failure time and repair time are exponentially distributed for analysis feasibility. However, in many cases, the repair time may not be exponentially distributed. For example, a server's repair time may follow a bimodal distribution (i.e., a mixture between short-term reboot and long-term hardware replacement). Such non-Markovian analysis is left as future work.

Another dimension is to validate the proposed model in a larger testbed. Due to resource constraint, we could not evaluate the proposed model in a large-scale system or simulation. Such large-scale validation would be beneficial for further improvement of the proposed analytical model.

CHAPTER 6

PUTTING IT ALL TOGETHER

6.1 Introduction

So far in previous chapters, we have proposed several analytical models to estimate a user-oriented QoS metric called subscriber real-time reliability. In each chapter, we focus on various factors that affect subscriber real-time reliability, ranging from content dynamism, mobility dynamism, to overlay dynamism. In order to focus on each type of dynamism, we made several assumptions in each chapter in order to ignore the effect of other types of dynamism for simplicity of analysis. In fact, many real-world publish/subscribe applications may be sensitive to only one type of dynamism while resilient to other types of dynamism. For example, stock market broker networks with high traffic and complex content filtering may be sensitive to content dynamism but resilient to broker dynamism (because of server maintenance). Mobility dynamism may greatly affect the performance of military or first-responder publish/subscribe operations while having slight effect on civilian publish/subscribe application. Choosing the right analytical model to address the high-impact dynamism while discarding the complexity of low-impact dynamism will result in accurate and efficient performance analysis and prediction.

However, there may be also some scenarios where publish/subscribe systems/applications are moderately affected by several types of dynamism. In such scenarios, it is not sufficient to analyze the publish/subscribe systems in question by focusing on only one type of dynamism, since that will lead to prediction inaccuracy. In such case, it is better to use the integrated model that considers multiple types of dynamism at the same time for the sake of prediction accuracy, but at the cost of complexity in analysis.

In this chapter, we present an integration framework that combines all analytical models presented so far into one unified model for subscriber reliability analysis. This chapter also discusses the scenarios where conflict may arise between each individual model.

6.2 Integration Framework

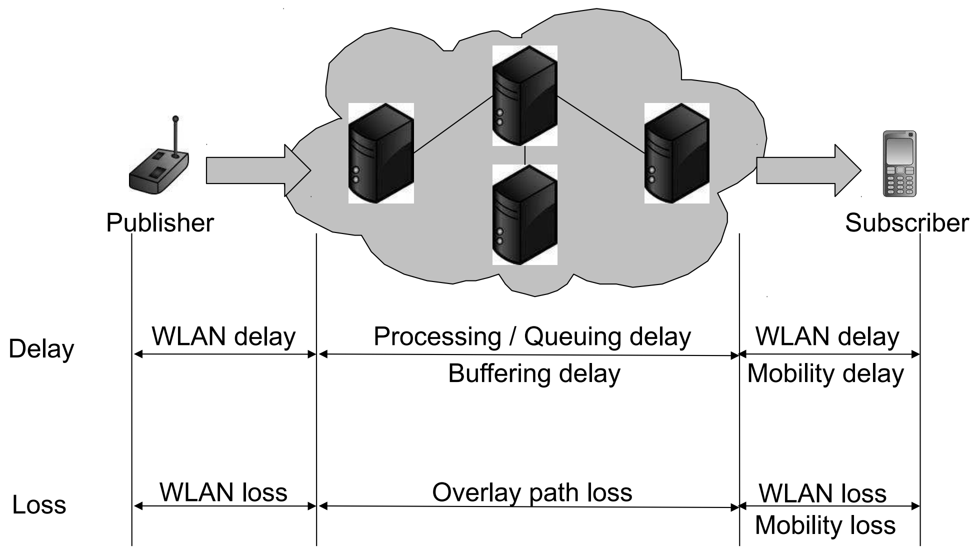


Figure 6.1: End-to-end reliability/timeliness breakdown

As mentioned, in each of previous chapters, we proposed the model to estimate subscriber real-time reliability with the focus on each type of dynamism while assuming no other types of dynamism. On the other hand, the integrated analytical model considers all types of determinism to determine subscriber's real-time reliability. To calculate the subscriber real-time reliability for integrated model, one must consider the definition of subscriber real-time reliability in Section 2.4. That is, there are two necessary conditions for an event to be delivered successfully from a publisher to a subscriber. The first condition is the event is not lost during its transit (end-to-end conventional reliability requirement[1]). The second condition is the event must not be expired when it arrives at the subscriber (end-to-end delay requirement). Thus, to calculate subscriber real-time reliability, it is necessary to calculate publisher-subscriber pairwise end-to-end conventional reliability and delay as shown in Figure 6.1.

From Figure 6.1, we can estimate the integrated subscriber real-time reliability as follows. Let $\grave{r}_{ps}^{E2E}$ denote the end-to-end conventional reliability and d_{ps}^{E2E} denote the end-to-end delay from a publisher p to a subscriber s. We also assume the pairwise traffic flow rate λ_{ps} can be estimated from the publisher's content profile and the subscriber's subscription filter. Thus, assuming each event has constant lifetime D without loss of generality, we can estimate integrated pair-wise real-time

[1]By conventional reliability, we mean the probability that the event is delivered successfully without any real-time constraint

reliability r'^{E2E}_{ps} as

$$\begin{aligned} r'^{E2E}_{ps} &= \text{P[event is not lost].P[event arrives on time]} \\ &= \grave{r}^{E2E}_{ps}.\text{P}[d^{E2E}_{ps} < D] \end{aligned} \tag{6.1}$$

Then we can estimate the integrated subscriber real-time reliability using Equation (5.1).

Note that in order to calculate end-to-end conventional reliability and delivery delay, we use each analytical model proposed in Chapter 3,4, and 5 to calculate each type of conventional reliability and delay. Figure 6.2, 6.3, 6.4 present the component diagram of each analytical model, its input, and its output to be combined into end-to-end metrics for the integrated analytical model.

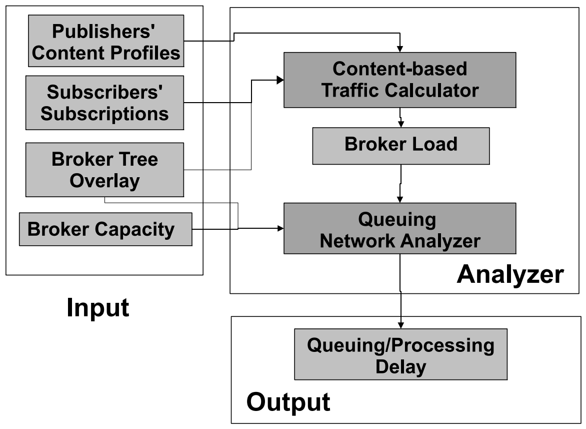


Figure 6.2: Content dynamism component

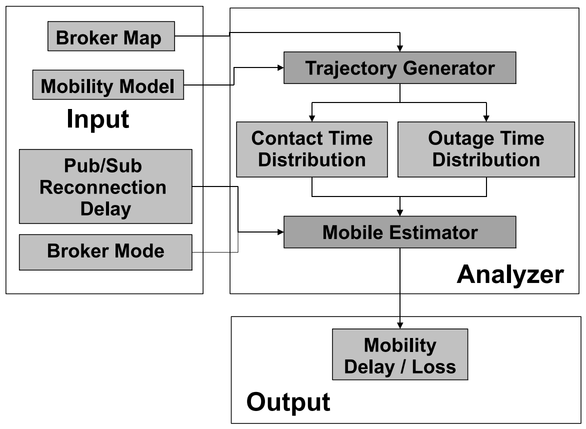


Figure 6.3: Mobility dynamism component

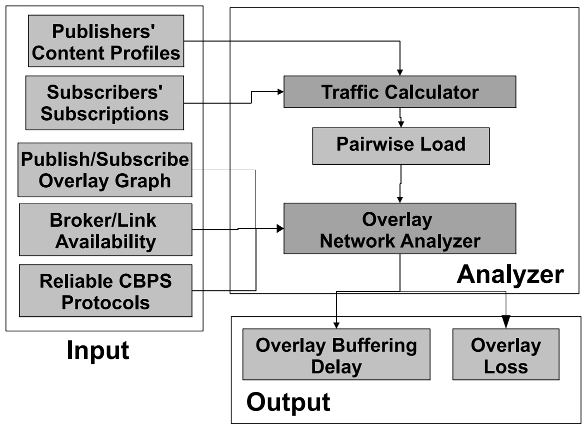


Figure 6.4: Overlay dynamism component

6.3 Discussion

We intentionally do not describe the exact calculation of the integrated model as the exact calculation of the integrated model may vary, depending on the exact implementation of the entire publish/subscribe protocol. However, our proposed integrated framework could be use as a guideline to construct the exact integrated analytical model. Also, hidden dependency among components is possible and may complicate the performance analysis of the protocol. Such dependency must be handled case by case, as it is infeasible to account for every component's parameter space when the system grows larger.

CHAPTER 7

CONCLUDING REMARKS

7.1 Conclusion

We are living in the world of information, where the amount of information and its producers grow exponentially day by day. Such phenomenon affects the communication paradigm, changing it from sender/receiver-based communication to content-based communication. However, content-based communication lacks of predictability, which prevents its full adoption within large-scale distributed systems with best-effort networks. Despite the problems with the content-based communication, time-sensitive content-based publish/subscribe applications have become deployed and started to pose stringent reliability/timeliness requirements to the content-based middleware layer. Such conflict is an important challenge to be solved for content-based paradigm to move forward as the next-generation communication.

This thesis explored the possibility to merge the gap between unreliable, best-effort publish/subscribe middleware and delay/reliability-sensitive content-based applications. To do so, the thesis proposed an analytical model framework to analyze the performance of distributed, content-based publish/subscribe systems under several types of uncertainty (i.e., dynamism). Specifically, the thesis has the following contributions.

- We proposed the use of *subscriber real-time reliability* as a single, client-oriented metric to describe the reliability/timeliness quality of service each subscriber receives from the content-based publish/subscribe systems.
- We proposed an analytical model to estimate subscriber real-time reliability in content-based publish/subscribe systems under *content dynamism*, which represents uncertainty in determining event traffic flow to subscribers. With the use of publisher content profile, the analytical model transforms the publish/subscribe traffic load into pairwise, point-to-point load, which can be used to calculate subscriber real-time reliability with existing queuing theory

techniques.

- We proposed an analytical model to estimate subscriber real-time reliability in content-based publish/subscribe systems under *mobility dynamism*, which represents uncertainty in mobile users' mobility patterns. The proposed analytical model estimates a mobile subscriber's real-time reliability from *contact/outage* duration distributions, protocol hand-off delay, and event lifetime. The proposed model also provides the analysis to derive contact/outage duration distributions from a wide-range of existing mobility model.

- We proposed an analytical model to estimate subscriber real-time reliability in content-based publish/subscribe systems under *overlay dynamism*, which represents failures of components in publish/subscribe routing overlay. The proposed model discussed several existing reliability-enhancement techniques and provided quantitative analysis of such techniques to study their effect to publisher-subscriber pairwise real-time reliability. The proposed model also presented a generic, protocol-independent framework to calculate subscriber real-time reliability, given the protocol-dependent publisher-subscribe pairwise real-time reliability.

- We validated each of the proposed analytical models via simulations with realistic parameters. All results yielded good prediction accuracy (i.e., less than 10% error) of our proposed models. Using the proposed models, we also studied various effects of each dynamism parameter to the performance of the protocol.

- We proposed the integrated analytical framework that combines all individual analytical models altogether in order to account all types of dynamism and predict the subscriber real-time reliability accurately.

We believe that the analytical framework proposed by this thesis work abstracts most existing content-based publish/subscribe systems and their applications. To the best of our knowledge, it is the first work to address reliability/timeliness in content-based publish/subscribe systems under best-effort networks, which represent a huge class of today's Internet applications. The proposed analytical framework can then be used as a building block for many quality of service optimization/adaptation techniques in such scenarios.

7.2 Lessons Learned and Future Work

7.2.1 Lessons Learned

During the process of developing the analytical framework, we have gained several insights in publish/subscribe performance prediction as follows.

Trade-off between flexibility and predictability : Since the advancement from point-to-point routing to content-based routing, the important factor that determines traffic and overhead has been shifted from communication endpoints to information content. The more level of flexibility a system becomes, the less predictability one can expect from it. This does not only apply to publish/subscribe systems, but also to other disruptive paradigms such as peer-to-peer systems as well[95, 74]. Up to now, a set of assumptions must be made to simplify the system and achieve predictability. It is still a question whether such trade-off between flexibility and predictability can be quantified or circumvented.

Trade-off between reliability, timeliness, and overhead : In this dissertation, we have revisited many existing reliability techniques that are commonly used in publish/subscribe systems such as retransmission, replication, and multi-path dissemination. In most techniques, there exists a trade-off between reliability, timeliness, and overhead. One must consider this trade-off when designing a publish/subscribe systems for any specific application such that the levels of reliability and timeliness satisfy such application. Moreover, since our application domain focuses on reliability and timeliness, the trade-off becomes more stringent but at the same time clearer to determine.

Trade-off between prediction accuracy and prediction efficiency : Our proposed analytical framework aims to determine fine-grained, user-oriented subscriber reliability for each individual subscriber in the system, which requires higher prediction overhead than other proposed models[31, 56] but also achieves more fine-grained prediction result. One must consider both prediction granularity requirement from the application as well as the system size and computational resource in order to pick the appropriate analytical model for each application and scenario.

No single best solution for all configurations/scenarios : We have analyzed several different publish/subscribe schemes under different settings and scenarios. While some publish/subscribe schemes outperform others in many scenarios, there is no single publish/subscribe scheme that performs best in all scenarios. With the proposed prediction model, one can determine which publish/subscribe configuration to be used for a specific scenario and gives the best performance.

We hope that these findings will be useful for any future research in distributed system analysis, specifically in content-based publish/subscribe systems.

7.2.2 Future Work

Several future research directions of this thesis are possible. Here, we list some of such potential future work

Better publisher content predictor : In Chapter 3, we currently assume simple statistical method such as periodic sampling to estimate the publisher content profile. However, we can employ more sophisticated publisher content prediction to get better traffic rate estimation.

More accurate analysis with less assumptions : The analysis used several assumptions for the purpose of analytical feasibility. Such assumptions include temporal locality in publisher's content, known publisher mobility, and exponential failure/repair times. Although such assumptions are generally held in many system research areas, we seek to improve the accuracy of the model while relaxing such assumptions for better applicability.

Other quality of service metrics : The analytical model proposed in this thesis considers reliability and timeliness as the quality of service metrics. So far, we did not address other metrics such as bandwidth to avoid the complexity in the analysis[95]. However, incorporating several other metrics such as bandwidth, ordered delivery, and jitter into the analytical framework is also one important future research direction.

Optimization techniques based on the predictive model : The proposed predictive model can be a useful tool for many performance optimizations of content-based publish/subscribe systems such as subscriber admission, subscriber-broker allocation, broker capacity planning. Such optimization techniques can be investigated in the future.

Distributed admission control algorithm : We currently assume a centralize algorithm to estimate each subscriber's reliability and admission control. A decentralized estimation and admission control algorithm can be done by coordination among brokers.

Realistic subscription popularity distribution : Most simulation results in this thesis are based on uniform subscription popularity. However, this does not reflect real-world scenarios where subscription popularity follows Zipf distribution[96]. The impact of subscription popularity distribution to subscriber reliability/timeliness is leaft as one possible future research direction.

Large-scale validation : Performance modeling of large-scale distributed systems can be proved useful only if it accurately describes the real behavior of the actual systems. While the validation of the proposed model via simulations with realistic parameters yields accuracy and applicability of the proposed model, further validation of the model via large-scale deployment would be the best way to show the effectiveness of our approach. Hence, such a large-scale validation is one of important future research directions.

REFERENCES

[1] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, "The many faces of publish/subscribe," *ACM Comput. Surv.*, vol. 35, no. 2, pp. 114–131, 2003.

[2] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf, "Design and evaluation of a wide-area event notification service," *ACM Trans. Comput. Syst.*, vol. 19, no. 3, pp. 332–383, 2001.

[3] Y. Zhao, D. Sturman, and S. Bhola, "Subscription propagation in highly-available publish/subscribe middleware," in *Proc. ACM Middleware '04*. New York, NY, USA: Springer-Verlag New York, Inc., 2004, pp. 274–293.

[4] G. Cugola and H.-A. Jacobsen, "Using publish/subscribe middleware for mobile systems," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 6, no. 4, pp. 25–33, 2002.

[5] A. Rowstron, A.-M. Kermarrec, M. Castro, and P. Druschel, "Scribe: The design of a large-scale event notification infrastructure," in *In Networked Group Communication*, 2001, pp. 30–43.

[6] L. F. Cabrera, M. B. Jones, and M. Theimer, "Herald: Achieving a global event notification service," in *Proc. HOTOS '01*, Washington, DC, USA, 2001, p. 87.

[7] S. Q. Zhuang, B. Y. Zhao, A. D. Joseph, R. H. Katz, and J. D. Kubiatowicz, "Bayeux: an architecture for scalable and fault-tolerant wide-area data dissemination," in *Proc. NOSSDAV '01*. ACM, 2001, pp. 11–20.

[8] P. R. Pietzuch and J. Bacon, "Hermes: A distributed event-based middleware architecture," in *Proc. ICDCSW '02*, 2002, pp. 611–618.

[9] O. M. Group, "Corba Event Service," http://www.omg.org/.

[10] M. Ryll and S. Ratchev, "Towards a publish / subscribe control architecture for precision assembly with the data distribution service," 2008, pp. 359–369.

[11] R. Baldoni, L. Querzoni, and A. Virgillito, "Distributed event routing in publish/subscribe communication systems: a survey," Tech. Rep., 2005.

[12] X. Guo, H. Zhong, J. Wei, and D. Han, "A new approach for overload management in content-based publish/subscribe," *ICSEA'07*, vol. 0, p. 32, 2007.

[13] R. S. S. Filho and D. F. Redmiles, "A survey of versatility for publish/subscribe infrastructures," University of California, Irvine, Tech. Rep. UCI-ISR-05-8, 2005.

[14] Y. Liu and B. Plale, “Survey of publish subscribe event systems,” Indiana University, Tech. Rep. TR574, May 2003.

[15] F. Araújo and L. Rodrigues, “On qos-aware publish-subscribe,” in *ICDCSW ’02: Proceedings of the 22nd International Conference on Distributed Computing Systems.* Washington, DC, USA: IEEE Computer Society, 2002, pp. 511–515.

[16] S. Behnel, L. Fiege, and G. Muhl, “On quality-of-service and publish-subscribe,” in *ICDCSW ’06: Proceedings of the 26th IEEE International ConferenceWorkshops on Distributed Computing Systems.* Washington, DC, USA: IEEE Computer Society, 2006, p. 20.

[17] S. P. Mahambre, M. K. SD, and U. Bellur, “A taxonomy of qos-aware, adaptive event-dissemination middleware,” *IEEE Internet Computing*, vol. 11, no. 4, pp. 35–44, 2007.

[18] X. Lu, T. Yang, Z. Liao, X. Li, Y. Wang, W. Liu, and H. Wang, “A novel qos-enable real-time publish-subscribe service,” in *ISPA.* IEEE, 2008, pp. 19–26.

[19] G. Deng, M. Xiong, A. Gokhale, and G. Edwards, “Evaluating real-time publish/subscribe service integration approaches in qos-enabled component middleware,” in *ISORC ’07: Proceedings of the 10th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing.* Washington, DC, USA: IEEE Computer Society, 2007, pp. 222–227.

[20] R. Rajkumar, M. Gagliardi, and L. Sha, “The real-time publisher/subscriber inter-process communication model for distributed real-time systems: Design and implementation,” in *Design and Implementation, in First IEEE Real-time Technology and Applications Symposium*, 1997, pp. 66–75.

[21] J. Wang, J. Cao, J. Li, and J. Wu, “Achieving bounded delay on message delivery in publish/subscribe systems,” in *ICPP ’06: Proceedings of the 2006 International Conference on Parallel Processing.* Washington, DC, USA: IEEE Computer Society, 2006, pp. 407–416.

[22] B. Zieba, “Overview of the reliability aspects in the publish/subscribe middleware,” in *OTM Workshops (2)*, ser. Lecture Notes in Computer Science, R. Meersman, Z. Tari, and P. Herrero, Eds., vol. 4806. Springer, 2007, pp. 1091–1100.

[23] P. C. Matteo, P. Costa, M. Migliavacca, G. P. Picco, and G. Cugola, “Introducing reliability in content-based publish-subscribe through epidemic algorithms,” in *In: Proceedings of the 2nd International Workshop on Distributed Event-Based Systems.* ACM Press, 2003, pp. 1–8.

[24] A. Gaddah, “A pro-active mobility management scheme for publish/subscribe middleware systems,” Ph.D. dissertation, Carleton University, 2009.

[25] S. Hu, V. Muthusamy, G. Li, and H.-A. Jacobsen, “Transactional mobility in distributed content-based publish/subscribe systems,” in *ICDCS ’09: Proceedings of the 2009 29th IEEE International Conference on Distributed Computing Systems.* Washington, DC, USA: IEEE Computer Society, 2009, pp. 101–110.

[26] G. Muhl, “Large-scale content-based publish/subscribe systems,” Ph.D. dissertation, University of Technology Darmstadt, 2002.

[27] M. A. Jaeger, "Self-managing publish/subscribe systems," Ph.D. dissertation, Technische Universität Berlin, 2007.

[28] R. Baldoni, R. Beraldi, S. Tucci Piergiovanni, and A. Virgillito, "On the modeling of publish/subscribe communication systems: Research articles," *Concurr. Comput. : Pract. Exper.*, vol. 17, no. 12, pp. 1471–1495, 2005.

[29] F. He, L. Baresi, C. Ghezzi, and P. Spoletini, "Formal analysis of publish-subscribe systems by probabilistic timed automata," in *27th IFIP WG 6.1 International Conference on Formal Techniques forNetworked and Distributed Systems (FORTE 2007), Tallinn, Estonia, June 27-29*, ser. Lecture Notes in Computer Science, vol. 4574. Springer, 2007, pp. 247–262.

[30] L. Baresi, C. Ghezzi, and L. Mottola, "On accurate automatic verification of publish-subscribe architectures," in *ICSE '07: Proceedings of the 29th international conference on Software Engineering*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 199–208.

[31] A. Schröter, "Modeling and optimizing content-based publish/subscribe systems," in *Proc MDS '09*. New York, NY, USA: ACM, 2009, pp. 1–6.

[32] A. Schröter, G. Mühl, S. Kounev, H. Parzyjegla, and J. Richling, "Stochastic performance analysis and capacity planning of publish/subscribe systems," in *DEBS '10: Proceedings of the Fourth ACM International Conference on Distributed Event-Based Systems*. New York, NY, USA: ACM, 2010, pp. 258–269.

[33] T. Pongthawornkamol, K. Nahrstedt, and G. Wang, "Probabilistic qos modeling for reliability/timeliness prediction in distributed content-based publish/subscribe systems over best-effort networks," in *ICAC '10: Proceeding of the 7th international conference on Autonomic computing*. New York, NY, USA: ACM, 2010, pp. 185–194.

[34] T. Pongthawornkamol and K. Nahrstedt, "Towards timeliness and reliability analysis of distributed content-based publish/subscribe systems over best-effort networks," University of Illinois at Urbana-Champaign, Tech. Rep. http://hdl.handle.net/2142/14415, November 2009.

[35] "Google finance," http://www.google.com/finance.

[36] "Gary-chicago-milwaukee corridor transportation information," http://www.gcmtravel.com/.

[37] "National climate data center: Online climate data directory," http://lwf.ncdc.noaa.gov/oa/climate/climatedata.html.

[38] "Rdf site summary (rss) 1.0," http://web.resource.org/rss/1.0/spec.

[39] "Real time sportscast - rts," http://www.rtsportscast.com/.

[40] M. J. Fischer, N. A. Lynch, and M. S. Paterson, "Impossibility of distributed consensus with one faulty process," in *PODS '83: Proceedings of the 2nd ACM SIGACT-SIGMOD symposium on Principles of database systems*. New York, NY, USA: ACM, 1983, pp. 1–7.

[41] A. Gupta, O. D. Sahin, D. Agrawal, and A. E. Abbadi, "Meghdoot: content-based publish/subscribe over p2p networks," in *Proc Middleware '04*, 2004, pp. 254–273.

[42] G. Cugola, D. Frey, A. L. Murphy, and G. P. Picco, "Minimizing the reconfiguration overhead in content-based publish-subscribe," in *SAC*, H. Haddad, A. Omicini, R. L. Wainwright, and L. M. Liebrock, Eds. ACM, 2004, pp. 1134–1140.

[43] G. Cugola, A. Margara, and M. Migliavacca, "Context-aware publish-subscribe: Model, implementation, and evaluation," in *ISCC*. IEEE, 2009, pp. 875–881.

[44] K. Jayaram, C. Jayalath, and P. Eugster, "Parametric subscriptions for content-based publish/subscribe networks," in *Proc Middleware '10*, 2010.

[45] G. Li, S. Hou, and H.-A. Jacobsen, "A unified approach to routing, covering and merging in publish/subscribe systems based on modified binary decision diagrams," in *Proc. ICDCS '05*, Washington, DC, USA, 2005, pp. 447–457.

[46] Z. Jerzak and C. Fetzer, "Soft state in publish/subscribe," in *DEBS '09: Proceedings of the Third ACM International Conference on Distributed Event-Based Systems*. New York, NY, USA: ACM, 2009, pp. 1–12.

[47] W. Whitt, "The Queueing Network Analyzer," *Bell System Technical Journal*, vol. 62, no. 9, pp. 2779–2815, November 1983.

[48] M. A. Marsan, C. Casetti, G. Mardente, and M. Mellia, "A framework for admission control and path allocation in diffserv networks," *Comput. Netw.*, vol. 51, no. 10, pp. 2738–2752, 2007.

[49] S. Even, A. Itai, and A. Shamir, "On the complexity of time table and multi-commodity flow problems," in *SFCS '75: Proceedings of the 16th Annual Symposium on Foundations of Computer Science*. Washington, DC, USA: IEEE Computer Society, 1975, pp. 184–193.

[50] "The network simulator - ns-2," http://www.isi.edu/nsnam/ns/.

[51] M. Ripeanu, I. T. Foster, A. Iamnitchi, and A. Rogers, "A dynamically adaptive, unstructured multicast overlay," in *Service Management and Self-Organization in IP-based Networks*, 2005.

[52] D. Surendran, "Visualizing connection bandwidths and delays in planetlab," http://people.cs.uchicago.edu/ dinoj/vis/planetlab/.

[53] H. Liu and H.-A. Jacobsen, "Modeling Uncertainties in Publish/Subscribe Systems," in *Proc ICDE'04*, March-2 April 2004, pp. 510–521.

[54] X. Guo, J. Wei, and D. Han, "Efficient Event Matching in Publish/Subscribe: Based on Routing Destination and Matching History," *Proc NAS'08*, vol. 0, pp. 129–136, 2008.

[55] R. Baldoni, R. Beraldi, S. T. Piergiovanni, and A. Virgillito, "On the modelling of publish/subscribe communication systems," *Concurrency - Practice and Experience*, vol. 17, no. 12, pp. 1471–1495, 2005.

[56] S. Kounev, K. Sachs, J. Bacon, and A. Buchmann, "A methodology for performance modeling of distributed event-based systems," in *Proc ISORC '08*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 13–22.

[57] Z. Jerzak and C. Fetzer, "Handling overload in publish/subscribe systems," in *ICDCSW '06: Proceedings of the 26th IEEE International ConferenceWorkshops on Distributed Computing Systems*. Washington, DC, USA: IEEE Computer Society, 2006, p. 32.

[58] S. Arianfar, "Optimizing publish/subscribe systems with congestion handling," M.S. thesis, Helsinki University of Technology, 2008.

[59] M. Caporuscio, A. Carzaniga, and A. L. Wolf, "Design and evaluation of a support service for mobile, wireless publish/subscribe applications," *IEEE Trans. Software Eng.*, vol. 29, no. 12, pp. 1059–1071, 2003.

[60] S. Tarkoma and J. Kangasharju, "On the cost and safety of handoffs in content-based routing systems," *Computer Networks*, vol. 51, no. 6, pp. 1459–1482, 2007.

[61] J. Wang, J. Cao, J. Li, and J. Wu, "Mhh: A novel protocol for mobility management in publish/subscribe systems," in *ICPP*. IEEE Computer Society, 2007, p. 54.

[62] A. Zeidler and L. Fiege, "Mobility support with rebeca," in *ICDCS Workshops*. IEEE Computer Society, 2003, pp. 354–.

[63] G. Cugola and H.-A. Jacobsen, "Using publish/subscribe middleware for mobile systems," *Mobile Computing and Communications Review*, vol. 6, no. 4, pp. 25–33, 2002.

[64] V. Muthusamy, M. Petrovic, D. Gao, and H.-A. Jacobsen, "Publisher mobility in distributed publish/subscribe systems," in *ICDCSW '05: Proceedings of the Fourth International Workshop on Distributed Event-Based Systems (DEBS) (ICDCSW'05)*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 421–427.

[65] C. M. Grinstead and J. L. Snell, *Introduction to Probability*, 2nd ed. American Mathematical Society, July 1997.

[66] M. M. Zonoozi and P. Dassanayake, "User mobility modeling and characterization of mobility patterns," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 7, pp. 1239–1252, 1997.

[67] V. Bychkovsky, B. Hull, A. Miu, H. Balakrishnan, and S. Madden, "A measurement study of vehicular internet access using in situ wi-fi networks," in *MOBICOM*, M. Gerla, C. Petrioli, and R. Ramjee, Eds. ACM, 2006, pp. 50–61.

[68] M. J. Pitkänen, T. Kärkkäinen, and J. Ott, "Opportunistic web access via wlan hotspots," in *PerCom*. IEEE Computer Society, 2010, pp. 20–30.

[69] E. Zola and F. Barcelo-Arroyo, "Impact of mobility models on the cell residence time in wlan networks," in *SARNOFF'09: Proceedings of the 32nd international conference on Sarnoff symposium*. Piscataway, NJ, USA: IEEE Press, 2009, pp. 14–18.

[70] Z. Zheng, Z. Lu, P. Sinha, and S. Kumar, "Maximizing the contact opportunity for vehicular internet access," in *In Proc. of IEEE INFOCOM*, 2010.

[71] Z. Zheng, P. Sinha, and S. Kumar, "Alpha coverage: Bounding the interconnection gap for vehicular internet access," in *INFOCOM*. IEEE, 2009, pp. 2831–2835.

[72] "Site5 uptime reports for all servers," http://www.site5.com/support/uptime/.

[73] "Planetlab - all pairs pings," http://pdos.csail.mit.edu/ strib/pl_app/.

[74] Z. Yao and D. Loguinov, "Link lifetimes and randomized neighbor selection in dhts," in *INFOCOM*. IEEE, 2008, pp. 146–150.

[75] C. Esposito, D. Cotroneo, and A. S. Gokhale, "Reliable publish/subscribe middleware for time-sensitive internet-scale applications," in *DEBS*, A. S. Gokhale and D. C. Schmidt, Eds. ACM, 2009.

[76] C. Esposito, D. Cotroneo, and S. Russo, "Reliable event dissemination over wide-area networks without severe performance fluctuations," in *ISORC '10: Proceedings of the 2010 13th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing*. Washington, DC, USA: IEEE Computer Society, 2010, pp. 97–101.

[77] D. G. Andersen, "Improving end-to-end availability using overlay networks," Ph.D., Massachusetts Institute of Technology, Feb. 2005.

[78] P. B. Godfrey, "Repository of availability traces," http://www.cs.uiuc.edu/homes/pbg/availability/.

[79] R. Chand and P. Felber, "Xnet: A reliable content-based publish/subscribe system," in *SRDS '04: Proceedings of the 23rd IEEE International Symposium on Reliable Distributed Systems*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 264–273.

[80] R. S. Kazemzadeh and H.-A. Jacobsen, "Reliable and highly available distributed publish/subscribe service," in *SRDS '09: Proceedings of the 2009 28th IEEE International Symposium on Reliable Distributed Systems*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 41–50.

[81] A. U. Shankar and S. S. Lam, "Time-dependent distributed systems: Proving safety, liveness and real-timeproperties," Austin, TX, USA, Tech. Rep., 1985.

[82] "Computing tcp's retransmission timer," http://tools.ietf.org/html/rfc2988.

[83] M. K. SD and U. Bellur, "Availability models for underlay aware overlay networks," in *DEBS*, ser. ACM International Conference Proceeding Series, R. Baldoni, Ed., vol. 332. ACM, 2008, pp. 169–180.

[84] P. Yalagandula, S. Nath, H. Yu, P. B. Gibbons, and S. Seshan, "Beyond availability: Towards a deeper understanding of machine failure characteristics in large distributed systems," in *In Proc. of USENIX Workshop on Real, Large Distributed Systems (WORLDS*, 2004.

[85] G. Latouche and V. Ramaswami, *Introduction to Matrix Analytic Methods in Stochastic Modelling*, 1st ed. ASA SIAM, 1999, ch. 2: PH Distributions.

[86] C. Moler and C. V. Loan, "Nineteen dubious ways to compute the exponential of a matrix," *SIAM Review*, pp. 801–836, 1978.

[87] L. G. Valiant, "The complexity of enumeration and reliability problems," *SIAM Journal on Computing*, vol. 8, no. 3, pp. 410–421, 1979. [Online]. Available: http://scitation.aip.org/getabs/servlet/GetabsServlet?prog=normal\&id=SMJCAT000008000003000410000001\&idtype=cvips\&gifs=yes

[88] M. Dahlin, B. B. V. Chandra, L. Gao, and A. Nayate, "End-to-end wan service availability," *IEEE/ACM Trans. Netw.*, vol. 11, no. 2, pp. 300–313, 2003.

[89] Z. Duan, Z.-L. Zhang, and Y. T. Hou, "Service overlay networks: Slas, qos, and bandwidth provisioning," *IEEE/ACM Trans. Netw.*, vol. 11, no. 6, pp. 870–883, 2003.

[90] Y. Amir and C. Danilov, "Reliable communication in overlay networks," in *DSN.* IEEE Computer Society, 2003, pp. 511–520.

[91] L. Subramanian, I. Stoica, H. Balakrishnan, and R. H. Katz, "Overqos: an overlay based architecture for enhancing internet qos," in *NSDI'04: Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation.* Berkeley, CA, USA: USENIX Association, 2004, pp. 6–6.

[92] M. Charikar, J. S. Naor, and B. Schieber, "Resource optimization in qos multicast routing of real-time multimedia," *IEEE/ACM Trans. Netw.*, vol. 12, no. 2, pp. 340–348, 2004.

[93] G. Tan and S. A. Jarvis, "Improving the fault resilience of overlay multicast for media streaming," *IEEE Trans. Parallel Distrib. Syst.*, vol. 18, no. 6, pp. 721–734, 2007.

[94] G. Cugola, G. P. Picco, and A. L. Murphy, "Towards dynamic reconfiguration of distributed publish-subscribe middleware," in *SEM*, ser. Lecture Notes in Computer Science, A. Coen-Porisini and A. van der Hoek, Eds., vol. 2596. Springer, 2002, pp. 187–202.

[95] G. Tan and S. A. Jarvis, "Stochastic analysis and improvement of the reliability of dht-based multicast," in *INFOCOM.* IEEE, 2007, pp. 2198–2206.

[96] Y.-M. Wang, L. Qiu, D. Achlioptas, G. Das, P. Larson, and H. J. Wang, "Subscription partitioning and routing in content-based publish/subscribe systems," in *16th International Symposium on DIStributed Computing (DISC'02).*