# CONFORMANCE PRESERVING DATA DISSEMINATION FOR LARGE-SCALE PEER TO PEER SYSTEMS

BY

LIPING CHEN

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, 2011

Urbana, Illinois

Doctoral Committee:

Professor Gul Agha, Chair and Director of Research
Professor Roy Campbell
Associate Professor Indranil Gupta
Assistant Professor Feida Zhu, Singapore Management University

# Abstract

In today's applications where dealing with vast stream data becomes a norm rather than an exception, it is in urgent need to design data dissemination systems in large scale. We identify a new pattern of data dissemination based on conformance constraints where data accuracy can be traded for low bandwidth. We formally define the problem of conformance preserving data dissemination and address two types of conformance data dissemination problems that we are interested in: data dissemination based on simple subscriptions and data dissemination based on composite subscriptions.

For simple subscriptions, we propose a Multilevel Cooperative Filter (MCF) overlay. We describe an online greedy algorithm to compute the minimum-size data sequence for dissemination and prove that it gives the optimal approximation ratio to the optimal off-line solution for all deterministic online algorithms. We then show that our multilevel cooperative filter algorithm generates the same dissemination sequence as the online greedy algorithm, thus proving the optimality of our approach. We further prove the NP-hardness of the filter overlay construction and give a $O(\ln n)$-approximation algorithm to minimize the level-wise communication cost.

We extend the model to support a richer and more expressive subscription semantics, allowing the user interest to be specified as arbitrary conformance predicates combined using logical operators on multiple data sources. Through Disjunctive Normal Form (DNF) transformation, arbitrary composite filters are decomposed into conjunctive filters. We then use a hybrid method based on filtering strength to support these conjunctive filters with low communication cost and low latency.

We have built a stock monitoring application using real life stock quotes collected from Yahoo Finance to evaluate the performance. A variety of experiments have been conducted to verify our design choices and deepen our understanding of the impact of various system parameters on both application-level and network-level performances. The simulation results suggest that the approaches are feasible to be deployed in large scale networks.

*To Father and Mother.*

# Acknowledgments

This project would not have been possible without the support of many people. Many thanks to my adviser, Gul Agha, who helped make some sense of the confusion. Also thanks to my committee members, Roy Campbell, Indranil Gupta and Feida Zhu who offered guidance and support. Thanks to the University of Illinois Graduate College for providing me with the financial means to complete this project. And finally, thanks to my family and friends who endured this long process with me, always offering support and love.

# Table of Contents

**List of Tables** . . . . . **vii**

**List of Figures** . . . . . **viii**

**Chapter 1 Introduction** . . . . . **1**

**Chapter 2 Related Work** . . . . . **6**
2.1 Peer-to-Peer Paradigm . . . . . 6
2.1.1 Unstructured P2P Systems . . . . . 6
2.1.2 Structured P2P Systems . . . . . 7
2.1.3 Range Queries on DHT Overlays . . . . . 8
2.2 Content Based Publish/Subscribe Systems . . . . . 9
2.2.1 DHT Based Publish/Subscribe Systems . . . . . 9
2.2.2 Subscription Covering and Merging . . . . . 10
2.2.3 Summarization . . . . . 10
2.2.4 Discussion . . . . . 11
2.3 Approximation Based Data Dissemination . . . . . 11

**Chapter 3 The Overall Framework** . . . . . **13**
3.1 The DHT (Distributed Hash Table) Layer . . . . . 13
3.1.1 Chord Layer . . . . . 14
3.1.2 Multicast in Chord . . . . . 14
3.2 The Dissemination Layer . . . . . 16
3.3 The Stateless Mapping Scheme . . . . . 16

**Chapter 4 Conformance Preserving Data Dissemination Based on Simple Subscriptions** . . . . . **18**
4.1 Introduction . . . . . 18
4.2 Problem Formulation . . . . . 19
4.3 Conformance Preserving Data Dissemination . . . . . 21
4.3.1 OnlineGreedy Is Optimal . . . . . 21
4.3.2 Single-level Filter Method . . . . . 24
4.3.3 Optimization and Difficulties . . . . . 25
4.3.4 Multi-level Filter Decomposition Method . . . . . 26
4.3.5 Multi-level Cooperative Filter Method . . . . . 29
4.4 Performance Evaluation . . . . . 36
4.4.1 Performance Metrics . . . . . 37
4.4.2 Real Data Experiments . . . . . 38
4.4.3 Synthetic Data Experiments . . . . . 41
4.4.4 Discussion on Network Dynamicity . . . . . 44

**Chapter 5 Conformance Preserving Data Dissemination Based on Composite Subscriptions . . . . . 46**
5.1 Introduction . . . . . 46
5.2 Problem Formulation . . . . . 46
5.3 Composite Conformance Preserving Data Dissemination . . . . . 48
5.3.1 Optimization Mechanisms . . . . . 48
5.3.2 The Dissemination Layer . . . . . 53
5.4 Performance Evaluation . . . . . 55
5.4.1 Experiment Setup and Evaluation Metrics . . . . . 55
5.4.2 Experiment Results . . . . . 56

**Chapter 6 Conclusion . . . . . 60**

**References . . . . . 63**

# List of Tables

4.1 Conformance preserving data dissemination for simple subscriptions: models and symbols . . . . . 20

5.1 Conformance preserving data dissemination for composite subscriptions: models and symbols . . . . . 48

# List of Figures


3.1 The system design . . . . . 13
3.2 Responsibility range for each finger node in Chord . . . . . 15

4.1 The dissemination layer of conformance preserving data dissemination . . . . . 22
4.2 The average ratio of delivered data size by OnlineGreedy over GlobalOpt for real stock data . . . . . 24
4.3 Organization: flat vs. hierarchical . . . . . 25
4.4 An example of data delivered to subscribers with different conformance thresholds . . . . . 26
4.5 A stationary state period of two-level filter movement . . . . . 27
4.6 Average subscription cost as the network size increases . . . . . 38
4.7 Average dissemination cost per data generation as the network size increases . . . . . 39
4.8 Average latency between data generation and data delivery as the network size increases . . . . . 40
4.9 Total length of delivered sequence as the percentage of synthetic data increases . . . . . 42
4.10 Average dissemination cost per data generation as the network size increases . . . . . 43
4.11 Average latency between data generation and data delivery as the network size increases . . . . . 44

5.1 The two semantics in composite conformance preserving data dissemination with the shaded area showing the data ranges that should be filtered out . . . . . 47
5.2 Filters of conjunctive or disjunctive semantics . . . . . 49
5.3 An example of filter decomposition . . . . . 50
5.4 Methods for processing subscriptions of size $> 2$ . . . . . 52
5.5 Example subscription processing . . . . . 53
5.6 An Example of data publication . . . . . 54
5.7 Average subscription cost as the network size increases . . . . . 56
5.8 Average dissemination cost as the network size increases . . . . . 57
5.9 Average latency between data generation and data delivery as the network size increases . . . . . 58
5.10 Average maintenance cost as the subscription dimension increases 58
5.11 Average latency as the subscription dimension increases . . . . . 59

# Chapter 1

# Introduction

The advancement of internet technology, pervasive computing, wireless and sensor networks has made vast data streams available. A data stream is a sequence of ordered pairs $(s, \Delta)$ where $s$ denotes a data value and $\Delta$ a time interval. We observe a proliferation of applications, e.g., financial data services, wide-area resource accounting, online-auctions, sensor network monitoring [1, 9, 21, 24, 30, 31, 44], spanning across areas of data engineering, distributed systems, sensor networks where stream-oriented data dissemination plays a crucial part.

A typical data dissemination system involves the participation of three logical components: the data sources continuously generating updates, the users submitting subscriptions to express different interests in the stream data, and the data dissemination network which installs filters based on subscriptions and delivers data to individual subscribers based on the results of the filters upon data updates. Logical divisions do not prevent different components from residing on the same physical node, i.e., a node can be a data provider, a user to other data sources and a part of the data dissemination network all at the same time.

Some common characteristics shared by these applications are: first, data streams are highly dynamic, i.e., the data values change continuously and in most applications, the number of subscribers is huge. Consequently, the communication overhead entailed could be prohibitively high if it is necessary to publish every data item. Second, it is also widely observed that, in many such applications, exact values of data are not required. For instance, in a stock price monitoring system, a trader may be interested in a certain stock only when the price change has reached a certain threshold. The same scenario may exist in a structural health monitoring system where an event is triggered when the changes of sensor readings are beyond some specified bounds.

We term the data dissemination problem in the above scenarios as the *Conformance Preserving Data Dissemination* problem. The subscriber is only interested in an incoming data item whose difference from the last delivered value is greater than the specified threshold. We call this threshold *conformance threshold*. Conformance preserving data dissemination offers a new latitude in handling data to trade data accuracy for low communication overhead.

Most of today's data dissemination systems are *content-based*. Content-based data dissemination systems install "static" filters into the systems in the sense that the data ranges specified by the subscriptions are fixed throughout the life time of these subscriptions in the system. For example, if a housing broker is only interested in houses between $300,000 and $400,000, only those listings that fall within this price range are delivered to the broker. *Conformance preserving data dissemination systems*, on the other hand, deal with *dynamic filters* as they are a function of not only the conformance threshold, but also a local state. The specified data range changes every time a local state changes. Referring to the previous housing broker example, a housing broker may be only interested in receiving a notification about a house when its price change is above a certain threshold. A *local state* of an individual subscriber is the last piece of data delivered to that subscriber, which changes every time when there is a data delivery. This adds another layer of complexity to the data dissemination problem.

Conformance preserving data dissemination is first studied by Shah et al. [42]. Their work focuses on developing techniques to improve data propagation efficiency and network resilience. However, they consider data assignment and client assignment separately and their algorithm is not designed to minimize dissemination cost. A similar problem is addressed by Olston et al. [33] by installing filters at remote data sources to minimize incoming data, which are then sent to a *central stream processor* for query handling. Unfortunately, as in the case with centralized models in general, this approach does not provide resilience to system failure, load balancing and scalability. These issues are even more critical in many of today's stream-data applications. In order to achieve system resilience and scalability, the unprecedented scale of these applications calls for a well-designed stream-data dissemination overlay.

In this thesis, we study the problem of conformance preserving data dissemination from two aspects:

**Single data source** - We propose a Multilevel Cooperative Filter (MCF) algorithm to address conformance based subscriptions to single data sources. Central to MCF are two key ideas: *cooperative filters* and *filter-based overlay*. Cooperative filters make sure that a filter with large conformance threshold can be decomposed into a sequence of smaller ones without compromising conformance preserving property. Filter-based overlay strives to make all such decompositions share as many common smaller filters as possible and thus minimizes the total number of filters to check.

**Multiple data sources** - We extend the model to support a richer and more expressive subscription semantics, allowing the user interest to be specified as constraints on multiple data sources. These constraints are combined using logical operators. Through Disjunctive Normal Form (DNF) transformation, arbitrary composite filters are decomposed into conjunctive filters. We then propose a hybrid method to support these conjunctive filters with low

communication cost and low latency.

## Motivating Scenarios

We highlight three motivating scenarios that may benefit from conformance preserving data dissemination systems below:

### Stock quote services

The advancement in distributed infrastructures has made online personal investment a reality. Millions of users are doing stock trading in front of their computers. Many existing stock monitoring applications track users' favorite stocks and alert them when the stock prices reach a specified high or low. This is equivalent to the notion of content based data dissemination where fixed ranges are specified. We anticipate another useful type of stock price monitoring where the decision making is based on a tolerance bound to the price fluctuation. For instance, an aggressive trader wants to be more in tune with the markets by specifying a stringent conformance threshold such that the price reported should not deviate from the actual value by a few cents. On the other hand, a long term trader may not care about penny losses. They may specify a larger conformance threshold.

More applications can be supported by allowing the user to specify conformance thresholds (filters) on multiple stocks. Multiple filters can be combined using logical operators $\cup$ and $\cap$. $\cup$ denotes an *any* semantics while $\cap$ denotes an *all* semantics. These semantics are useful to support applications to monitor trends, events, or when a decision is made not based on a single stock, but a number of stocks.

### Structural health monitoring through sensor readings

Structural health monitoring involves monitoring a structure over time through periodic readings from a variety of sensors such as light, vibration, temperature, sound, etc to determine the state of the structural health. Sensor readings are generated in large volumes and it is infeasible and unnecessary to deliver all sensor readings to the subscribers.

The degradation of the structures and occurrence of extreme events such as fire or earthquakes are in general reflected in the change of sensor readings rather than the readings themselves. The users may specify conformance thresholds for light, motion, temperature, magnetic fields, etc to monitor different aspects of the structural health. Conformance preserving data dissemination systems would provide a natural infrastructure for such applications.

### Network monitoring applications

Network traffic monitoring is of great interest to Internet Service Providers (ISPs), large corporates and institutions, online service providers for security, infrastructure planning, etc. Network traffics on logins, DNS lookups, visits to individual pages can all be monitored. If not designed carefully, the continuous monitoring might generate a lot of traffic which might be disruptive to the normal service. Hence different aspects of the network traffics can be monitored and for each, different alert levels can be registered to the network based on different tolerance level. Conformance preserving data dissemination provides a natural framework to support such applications.

## Main Contributions

The main contributions of this thesis are:

1. Our work represents the first study of dynamic filters with structured overlays. We formally define the problem of conformance preserving data dissemination and extend the model to support arbitrary composite conformance predicates combined using logical operators.

2. For simple subscriptions, we describe an online greedy algorithm to compute the minimum-size data sequence for dissemination and prove that it gives the optimal approximation ratio to the optimal off-line solution for all deterministic online algorithms. We then show that our multilevel cooperative filter algorithm generates the same dissemination sequence as the online greedy algorithm, thus proving the optimality of our approach. We propose a multilevel overlay of cooperative filters, offering a distributed model for conformance preserving stream-data dissemination.

3. For composite subscriptions, we use Disjunctive Normal Form (DNF) transformation to decompose arbitrary composite filters into conjunctive filters. Semantic optimization is further performed based on filtering strength and we propose a hybrid method to support subscriptions of larger dimentions.

4. We build a stock monitoring application using real time stock quotes and conduct extensive experiments to demonstrate the scalability of our system and show that it is feasible to be deployed in large scale systems.

The rest of the thesis is organized as follows. In Chapter 2, we outline background information and prior research efforts relevant to our research. Chapter

3 introduces the overall framework. We describe our work to support conformance preserving data dissemination in Chapter 4 and 5: Chapter 4 focuses on data dissemination based on simple subscriptions while Chapter 5 supports composite subscriptions based on multiple data sources. Chapter 6 concludes the thesis with a discussion of future research directions.

# Chapter 2

# Related Work

## 2.1 Peer-to-Peer Paradigm

Peer-to-Peer(P2P) systems offer an alternative to traditional client/server systems for large scale distributed applications. Participants within a P2P system share a portion of their resources such as computation power, storage, file content, etc. In contrast to client/server systems where a clear functional distinction exists between clients and servers (dedicated servers provide contents or services while clients request those contents and services provided by the servers), no such clear distinction exists in a P2P system. Participants within a P2P system can function as clients or servers at the same time.

### 2.1.1 Unstructured P2P Systems

In unstructured P2P networks, the placement of data or index of data has no relation with the overlay topology. Each participant in the network has no knowledge on where the data are stored on the other nodes. The actual search is usually through flooding or random-walk.

Napster [32] uses a central server to store the meta-data (index). However, central servers have a single point of failure and they are vulnerable to malicious attacks. Furthermore, the issue of scalability becomes the major concern for large scale applications. On the other end of the spectrum of decentralization is Gnutella [22]. It eliminates the central index server completely and constructs a virtual overlay for routing by each peer maintaining a list of "neighbors". The original Gnutella implementation uses a flooding mechanism to distribute *Query* messages by each node forwarding the messages to all of its neighbors. Scalability issues arise due to the excessive message overhead caused by flooding. Kazaa [26] lies in between the two extremes discussed above. It introduces the notion of *Supernodes* which act as a central server in a small part of the P2P network. The hybrid nature of Kazaa harnesses the advantage of both centralized and decentralized indexing: it reduces the discovery time, the workload and also avoids a single point of failure

### 2.1.2 Structured P2P Systems

In structured P2P systems, the topology of the network is tightly tied to the placement of data or data index. By allowing each node storing a carefully selected set of index and routing information, efficient lookup services could be achieved by employing a globally consistent protocol. Among them, *distributed hash tables* (DHT) are by far the most common types of structured P2P systems.

Distributed hash tables (DHTs) use a consistent hashing scheme to hash data content to keys. Each peer in the network is responsible for maintaining a portion of the keys in such a way that any change in the network will cause minimal interruption. This allows DHTs to scale to large number of participating nodes and to handle network dynamics introduced by constant node joining, leaving and failures. Many DHT schemes have been proposed and in this section, we focus on three representative schemes: Chord, CAN and Pastry.

Chord [17] uses consistent hashing to assign both the data and peers a key in a circular identifier space. In a steady state, each peer maintains routing information for about $O(logN)$ other peers, called fingers, where $N$ is the size of the network. The finger placement algorithm computes exponentially spaced fingers and therefore allows routing in $O(logN)$ hops from one node to any other node in the network. As we use Chord as the underlying DHT overlay in this research, we will postpone the detailed discussion on Chord to Section 3.1.1.

CAN [38] uses a virtual $d$-dimensional Cartesian coordinate space, a $d$-torus in fact. The space is dynamically partitioned among all the nodes in the network. CAN nodes maintain a routing table containing the routing information about nodes that responsible for zones adjacent to their own. Routing between any pair of nodes in CAN can be achieved by Manhattan routing by greedily forwarding data to the neighbor with coordinates closest to the destination coordinates. Therefore, by storing $2d$ neighbors, CAN is able to achieve an average routing path length of $O(N^{1/d})$, where $N$ is the number of zones in the coordinate space or the number of nodes in the network.

Pastry [40] is based on Plaxton graph and is in many ways similar to Chord. The major distinction is in routing and therefore how routing information is maintained in individual nodes. Pastry nodes maintain a routing table with row $i$ storing the routing information about nodes which share a common prefix of length $i$ with them. Prefix routing is used by finding a node in the routing table which shares a longer prefix with the destination address than the peer itself. Pastry improves upon Chord by the use of a neighborhood set. It takes into account the physical closeness on the underlying Internet when selecting which node to forward the data to.

A tradeoff exist in DHTs between the node degree, i.e. the neighbors of any node, and the route length. Intuitively, the higher the node degree, the shorter the route length. However, the higher the degree, the higher the maintenance cost due to network dynamicity caused by constant node join, leave, or failures.

Most of the popular DHTs uses a combination of $O(logN)$ degree with route length $O(logN)$, except that CAN uses $O(d)$ degree with route length $O(N^{1/d})$.

### 2.1.3 Range Queries on DHT Overlays

DHT offers an attractive platform to support applications based on exact matches. The later development calls for systems that support more expressive queries, namely range queries. However, the very feature, namely the randomized hash function, that makes for DHTs' good load balancing works against range queries. To still take advantage of the good load balancing of DHTs, most DHT-based overlay schemes adopt the approach to hash ranges rather than individual values. However, they differ in the underlying DHTs and how ranges are divided and mapped onto the DHT overlays.

#### Space Filling Curves

Adrzezak and Xu [3] are perhaps the first to support range query processing over DHT-based P2P systems. They use CAN as the underlying DHT and range queries are supported by mapping intervals to zones according to two properties: first, close-by intervals should be mapped to two close-by zones; second, if an interval is partitioned into two sub-intervals, the zones of two sub-intervals must partition that of the original interval. Such mapping can be achieved by using space filling curves and they use Hilbert curve in their work.

While [3] supports range queries on 1-dimensional space by mapping a 1-dimensional space to a d-dimensional CAN zones using space filling curves, space filling curves can also be used in a reverse manner by encoding the d-dimensional keyword space to a 1-dimensional index space, as done in Squid [41]. A linear order is therefore imposed on the partitions of $d$-dimensional keyword space so that they can be mapped to Chord. Such a construction makes *multi-attribute range queries* a possibility.

#### Tree Based

A variety of tree-based over-DHT indexing schemes are proposed in the literature. Common to these approaches are a tree structure to support range queries and a mapping scheme to map the tree nodes to the underlying DHTs. They differ in the types of trees selected and the mapping scheme.

*Prefix hash tree (PHT)* is used in [39]. The root of the tree is labeled with the attribute name and all downstream vertices are labeled recursively this way: given a vertex $l$, its left and right child are labeled as $l0$ and $l1$ respectively. Data are stored only at leaf nodes and internal nodes are for routing purposes only. The prefixes of the leaf nodes in the PHT form a universal prefix set such that for any infinite binary sequence, there is exactly one element in the set which is a prefix of that sequence. The mapping between the tree structure and the

underlying DHT is achieved by hashing the prefix labels of PHT vertices over the DHT node identifier space.

*P-tree* is introduced in [15, 16] as a distributed version of B+-tree. The key idea behind the P-tree is to maintain parts of semi-independent B+-trees at each peer. Furthermore, each peer stores and maintains only the left-most root-to-leaf path of its corresponding B+-tree. Therefore, P-tree provides search performance similar to B+-tree while at the same time, only responsible for maintaining the consistency of a limited number of nodes, i.e., nodes along its leftmost root-to-leaf path.

Distributed quad-tree is used in [45] to superimpose a quad-tree over Chord. Multi-dimensional data space is recursively decomposed into four congruent square blocks and each quad-tree block can be uniquely identified by its centroid. A linear order is imposed on these centroids by hashing such that the responsibility for a quad-tree block is associated with a Chord peer.

A general framework is proposed in [13] to map tree-structured logical data onto a DHT-based physical node space. Three mapping schemes are compared with respect to their performance characteristics under different scenarios, each of which uses a different replication mechanism to reduce range search time and to achieve load balance.

## 2.2 Content Based Publish/Subscribe Systems

In a publish/subscribe or data dissemination system (we use these two terms interchangeably in this thesis), as subscribers and publishers are distributed in the network, a publisher does not know who are interested in its data, and vice versa, a subscriber does not know where in the network its data of interest are available. So the fundamental problem in a publish/subscribe systems is to design a mechanism to deliver data updates to interested users quickly. The early connection between data dissemination and DHT-based P2P systems are through topic based publish/subscribe systems. Scribe [11] and Bayeux [53] are two earlier such attempts based on Pastry [40] and Tapestry[52] respectively. As DHT overlays efficiently route subscriptions and data updates to their corresponding nodes that are determined based on the hashing function, they provide a natural infrastructure to support data dissemination systems.

The later development in DHT overlays to support range queries also opens a door for its adoption to support content based publish/subscribe systems. Based on their focuses, we classify existing research efforts into three categories:

### 2.2.1 DHT Based Publish/Subscribe Systems

Research efforts in this category focus on understanding different DHT overlays and utilizing them to provide efficient and effective data delivery. These approaches differ in the choice of underlying overlays and how subscriptions are

mapped to overlay addresses. In particular, Meghdoot [23] supports subscriptions of range constraints by converting an n-dimensional range to a point in 2n-dimensional space and deriving a proper scheme to map logical partitions to CAN [38] nodes. [37] builds over Pastry [40]. By choosing the right combination from forward and reverse path of subscription and advertisement, it supports both subject-based and content-based subscriptions. Both [46] and [5] use Chord [17] as the underlying overlay. [46] proposes an order preserving Chord where each attribute domain occupies a continuous segment of the Chord ring, while [5] introduces a general architecture that adopts an abstract stateless mapping.

### 2.2.2 Subscription Covering and Merging

Research efforts in this category further exploit the relation among individual subscriptions through subscription covering and merging [10, 28, 35]. Subscriptions in [10] are represented in a partially ordered set, where the partial order is defined by the covering relations. The routing information is pushed out from a subscriber to all potential publishers according to the subscription covering relation stored at interfaces of participating nodes . Therefore forwarding states are set up to deliver messages of interest to the subscriber.

Subscriptions in [28] are represented using modified binary decision diagrams (MBDs) so that publication routing, subscription covering and merging can be supported in a unified manner. A novel subscription covering algorithm is proposed based on MBDs. Subscription merging, especially imperfect merging, is also explored.

The approach proposed in [35] uses a "Monte Carlo" type probabilistic algorithm for subscription set reduction. Instead of exploring set covering relation among single subscriptions, it checks the coverage relationship between a subscription and a set of subscriptions. Hence it reduces the set of active subscriptions in the overall system and provides gains in terms of publication matching because the tested subscription set is reduced.

Covering relations are exploited in the aforementioned systems due to the existence of *Inclusion Property* defined in Section 4.3.3. Conformance preserving data dissemination, however, deals with conformance thresholds, which are *dynamic filters*. They may not exhibit the Inclusion Property and hence a different approach should be adopted.

### 2.2.3 Summarization

Subscription summarization introduces a new paradigm for publish/subscribe systems [46]. Instead of exploring relations among subscriptions, it compacts the subscription information per broker. Each subscription is dissolved into its attribute-value pairs, which are in turn merged into their corresponding summary structures. In this paradigm there are no subscription entities, only sub-

scription summaries. New corresponding matching algorithms are developed. Furthermore, multi-broker subscription summaries are also explored and distributed algorithms for efficiently propagating multi-broker summaries among brokers are also developed.

### 2.2.4 Discussion

Content based data dissemination systems deal with *static filters*, where data delivery is based on absolute value ranges rather than a conformance bound. Conformance preserving data dissemination systems, on the other hand, deal with *dynamic filters*. Since dynamic filters are filters moving with the current state of each subscriber, a naive way to support dynamic filters in content-based data dissemination systems is to continuously de-register and re-register the filters as the underlying state changes. Such repeated registration would incur prohibitively high communication cost when the number of subscriptions is huge, which is common in a large scale data dissemination systems. Therefore, we are seeking a new approach to deal with dynamic filters.

## 2.3 Approximation Based Data Dissemination

The idea of using numeric bounds or precision constraints in persistent queries are studied in [6, 33, 43, 49] in the database community. Their focus is on maintaining numeric bounds on *aggregated values* from multiple sources. However, in this thesis, we are not interested in approximation on *aggregated values*, but the data streams themselves.

Negotiation among data sources in a distributed manner is proposed in [6, 43, 49] with the goal of reducing stream rates with varying numerical conformance constraints. However, the various algorithms proposed are not designed for the purpose of minimizing dissemination cost. The approach presented in [33] uses a centralized processor to install filters at remote data sources to minimize stream rates while guaranteeing precision constraints of persistent queries. As in the case with centralized models in general, this approach does not provide resilience to system failure, load balancing and scalability.

The most close research work to ours is [42]. They propose a network of corporative repositories to deliver data with precision guarantees for a large population of remote subscribing users. The data delivery is through a dynamic data dissemination tree where the data source is the root of the tree. The dissemination tree is formed in such a way that repositories with stringent coherence requirements are placed closer to the root. However, their algorithm is not designed to minimize the dissemination cost. Furthermore, they do not use DHT as the underlying structure, hence resilience, scalability issues have to be dealt with separately.

The problem of maximizing data precision given the available communication resources is addressed in [34], which is an inverse problem to ours.

We observe the counterparts of dynamic filters in other research contexts, with different concerns and solutions. In the context of replicated network service, algorithms are proposed in [51] to efficiently bound absolute error among replicated services using only local information. However, they are not concerned with data dissemination or other network issues such as self organization, scalability. Conformance maintenance is studied in [8] and [50] in the context of web caching. In web caching, the staleness of the data object is measured by time, while in our context, it is measured by value. Maintaining conformance based on time is considered a simpler problem since time only monotonically increases while values could fluctuate in both directions.

We study conformance preserving data dissemination systems based on simple subscriptions in [12]. This thesis extends the model to support both simple and composite subscriptions. Composite queries are studied in [27] based on *and* and *or* operators. However, their problem setting is different from ours as they are to support group based aggregation queries.

# Chapter 3

# The Overall Framework

Data dissemination systems present an important paradigm for asynchronous communication in a distributed environment. We propose a general framework for large-scale data dissemination. The algorithms developed for small-scale applications may not be suitable in this context. To this end, we leverage the strength of a distributed model to support massive data dissemination.

Our design principle is to harness the benefit of DHT while at the same time, build over-DHT overlays to counteract the randomness introduced by DHT. Figure 3.1 shows the overall design diagram. It consists of three components: the DHT layer, the dissemination layer and a stateless mapping scheme to map dissemination layer to the DHT layer. We introduce each of them below.

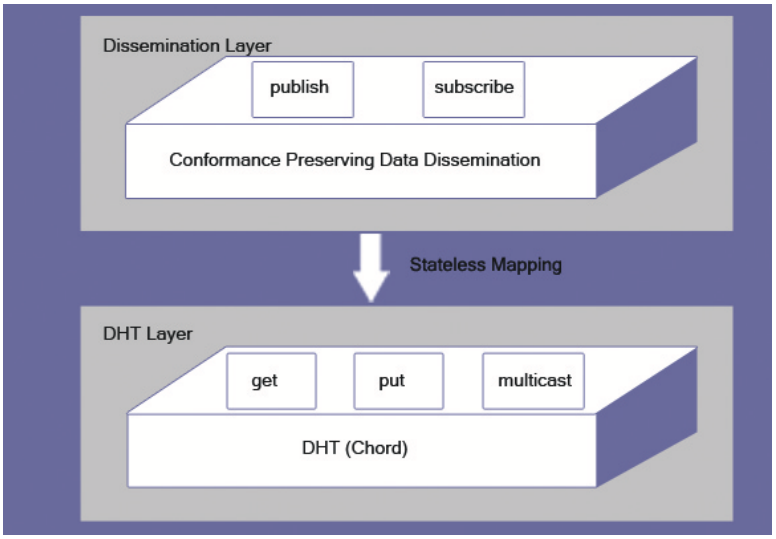



Figure 3.1: The system design

## 3.1 The DHT (Distributed Hash Table) Layer

In the literature, we observe two branches of research efforts to support distributed data dissemination: the DHT-based overlay schemes which build an add-on layer on top of DHT to address complex queries that are not supported

directly by DHT; direct distributed indexing schemes such as skip graph [4, 19] or tree-based structures [25, 18], which remove the extra layer of DHT and use distributed data structures to support range-based queries. The DHT-based overlays are less efficient than the latter since they introduce another layer of indirection. However, direct distributed indexing schemes lack the inherent benefits of load balancing, self-organization that come with DHT. They usually need to deal with those issues separately.

In this research, we focus on a DHT-based overlay scheme, Chord specifically, due to the following reasons: first, we want to support different types of data dissemination and a common DHT layer would maximize the reusable components of the system. Second, DHTs could save us the trouble of addressing the scalability, load balancing and robustness issues in a data dissemination system so that we could focus more on designing the right dissemination layer. Third, DHT-based schemes can be easily deployed on existing DHT networks, which is important from a practical point of view.

The idea of DHTs is to "provide a conceptually global, but physically distributed directory"[7]. DHTs provide a generic *put/get* interface for storage and retrieval, which is based on an abstract key space. $put(k, data)$ computes a key based on the data and stores the data on a network node in charge of the key using the underlying routing mechanism. $get(k)$ retrieves the data from the node storing the key.

### 3.1.1 Chord Layer

Chord[17] orders the identifiers in a ring modulo $2^m$, where $2^m$ is the size of the identifier space. Both the key space and the node address space (i.e., IP addresses) are mapped to this same identifier space. A given key, $k$, is assigned to the first node whose identifier is equal to or immediately follows $k$ in this circular space. That node is called the key's *successor*.

For a network overlay of $N$ nodes, Chord maintains two sets of neighbors for each node for routing purposes. The first set consists of two nodes: the *successor* and the *predecessor*. Routing correctness is achieved with this set. The second set stores the finger table of $O(logN)$ nodes spaced exponentially from the current node in the key space. Routing is achieved by forwarding the key to the closest node in the finger table that precedes the key. Finger table achieves routing efficiency by ensuring that the lookup cost is $O(logN)$.

### 3.1.2 Multicast in Chord

Large scale data dissemination often involves delivering the same set of data to multiple destinations. Hence a basic multicast mechanism is needed. As Chord is mainly designed as a point-to-point protocol, it lacks a basic multicast mechanism which is indispensable in a data dissemination network. We add an additional multicast() interface in the DHT layer, as shown in Figure 3.1.

Since in Chord a linear order is imposed on all the participating nodes to form the ring, this characteristic could be used to construct an efficient multicast primitive.

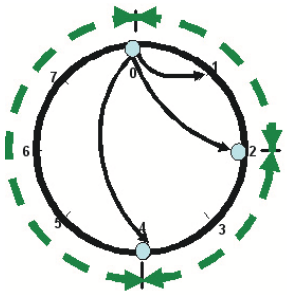



Figure 3.2: Responsibility range for each finger node in Chord

If we view *broadcast* in Chord as forming a spanning tree rooted at the node which initiates the action, multicast in Chord effectively prunes the spanning tree by exploring only those branches which lead to required destinations. Given a list of destinations *dlist* and a finger table, the finger node at entry $i$ is only responsible for forwarding messages to those destinations whose *ChordID*s fall with the range of $[f[i], f[i+1])$. Figure 3.2 shows the responsibility range of each finger table node using the arrowed lines. Algorithm 1 gives the detailed multicast algorithm. Algorithm 2 describes a helper function used by Algorithm 1 to extract a set of destinations from the destination list *dlist* that falls within the clockwise interval of $[n_1, n_2)$. PROCESS(d) is a generic function indicating the processing of data once they reach the destination.

**Algorithm 1** Multicast on node $n$

**Input:** message $\langle$**multicast**, $\omega$, *dlist*$\rangle$

1: **if** $n \in dlist$ **then**
2: $dlist = dlist \setminus n$
3: n.PROCESS($\omega$)
4: **end if**
5: **if** $n.successor \in dlist$ **then**
6: $dlist = dlist \setminus n.successor$
7: send $\langle$**multicast**, $\omega$, $\{n.successor\}\rangle$ to $n.successor$
8: **end if**
9: **for** $i = 1$ to $m$ **do**
10: d = EXTRACTDESTS(*dlist*, $n.f[i]$, $n.f[(i+1) \mod m]$
11: **if** $d \neq \emptyset$ **then**
12: send $\langle$**multicast**, $\omega, d\rangle$ to $n.f[i]$
13: **end if**
14: **end for**

**Algorithm 2** ExtractDests on node $n$

**Input:** $\langle dlist, n_1, n_2 \rangle$

```
if n1 < n2 then
  return {k ∈ dlist | k ∈ [n1, n2)};
else
  return {k ∈ dlist | k ∈ [n1, 2^m] ∪ [0, n2)};
end if
```

## 3.2 The Dissemination Layer

The dissemination layer shown in Figure 3.1 provides two functional interfaces: publish() and subscribe(). Users use the subscribe() interface to register data constraints in the system while data sources use publish() to deliver data to interested users.

Since data sources and users are distributed across the network, the dissemination layer should provide some form of directory service. A centralized directory service is used in earlier data dissemination systems. However, as data dissemination expands to a much larger scale, a centralized solution is no longer feasible. The directory has to be partitioned and stored at a collection of network nodes. Requirements for doing such partitioning and storage are twofold: first, both publishers and users should be able to locate the relevant directory service easily without global knowledge; second, no single node is a performance bottleneck.

In the next two chapters, we will discuss the detailed designs of dissemination layer to support different types of conformance preserving data dissemination, namely data dissemination based on simple subscriptions and data dissemination based on composite subscriptions.

## 3.3 The Stateless Mapping Scheme

The dissemination layer that we have proposed to support conformance preserving data dissemination needs to be mapped to the Chord layer, which essentially enforces a linear order on the network nodes. A stateless mapping scheme excels in a distributed environment as frequent node joins or leaves will only affect a limited number of network nodes. No global knowledge is required; such knowledge is usually not available in a distributed system.

We define a stateless mapping scheme to map the index structure to the Chord space as a hash function: $h(x, \delta) = h_1(x) + h_2(\delta)$, where $x$ is the data source. $\delta$ may be defined differently for different index schemes and it will become apparent what $\delta$ represents when we discuss different index schemes in the next two chapters. Hash function $h(x)$ consists of two portions: the randomized portion $h_1$ and the order preserving portion $h_2$. The randomized portion is to randomly distribute within the Chord space subscriptions to different data sources. The order preserving portion is to store subscriptions to the same data

source on a contiguous Chord space, a ring segment essentially. We use SHA-1 as $h_1$ to map data source $x$ to a number $k$ in Chord space, which marks the beginning of the ring segment that subscriptions to $x$ occupy. Depending on the $\delta$ value, $h_2$ determines which node within the ring segment is used to store the subscription. The range of $h_2$ is bounded by $L$, the length of the ring segment. $L$ is determined by the number of subscriptions to a particular data source $x$ and the node capacity. We use a simple linear function as $h_2$ where $h_2(\delta) = a\delta$ and $a = L/max(\delta)$.

# Chapter 4

# Conformance Preserving Data Dissemination Based on Simple Subscriptions

## 4.1 Introduction

The advancement of the P2P paradigm has proved it to be a feasible means for data dissemination. Most of today's data dissemination systems are content-based. These systems deal with *static filters*: filters specifying a fixed data range allowed to be delivered to subscribers. With the proliferation of streaming data such as news feed, stock quotes or sensor readings, we see an opportunity for a new pattern of data dissemination. With the data generated continuously in large quantities, it is almost infeasible and unnecessary to deliver every piece of data generated. The subscribers are no longer interested in exact values of the data stream, but a quantitative approximation guarantee of the data delivered.

We consider an environment where a subscriber is only interested in an incoming data item whose difference from the last delivered value is greater than the specified threshold. We call this threshold *conformance threshold* and this type of data dissemination *conformance preserving data dissemination*.

Contrast to content-based data dissemination, conformance preserving data dissemination systems deal with varied data ranges. Only data that deviate beyond a specified conformance threshold need to be delivered. The last piece of data delivered to an individual subscriber represents that subscriber's *local state*. As the local states change over time, the allowed data ranges shift. Thus, a conformance based filter is a function of two variables: the conformance threshold and a local state. We call such filters *dynamic filters*.

Because of their prohibitive dissemination cost, large scale conformance preserving data dissemination systems present special challenges, as well as opportunities for performance optimization. Subscriptions may potentially come from any node in the network and a mixture of fine and coarse filters may coexist in the system. A naive approach is to apply to the stream a distinct filter for each given conformance threshold; however, this approach incurs significant communication cost as each data item needs to be sent to every filter to check for conformance.

We use a sequence of small conformance thresholds in the system to simulate larger ones. This is motivated by the following observation: values in the data stream are not random but represent relatively small changes from prior values.

We call this property of streams *temporal locality*. We exploit temporal locality to reduce unnecessary checks: by organizing the filters in multi-level hierarchy, we need not check the data against relatively coarse-grained filters when more fine-grained filters are not satisfied. As a result much fewer network nodes are multicast for each level.

The main contributions of this work are as follows:

1. To the best of our knowledge, the work presented in this chapter represents the first study of dynamic filters with structured overlays.

2. We describe an online greedy algorithm to compute the minimum-size data sequence for dissemination and prove that it gives the optimal approximation ratio to the optimal off-line solution for all deterministic online algorithms. We then show that our multilevel cooperative filter algorithm generates the same dissemination sequence as the online greedy algorithm, thus proving the optimality of our approach.

3. We propose a multilevel overlay of cooperative filters, offering a distributed model for conformance preserving stream-data dissemination. In particular, we give the constraints for the decomposition of any single filter based on a conformance threshold into multilevel filters and prove the NP-hardness of the filter overlay construction for a collection of conformance thresholds. We give a greedy algorithm which is a $O(\ln n)$-approximation for the optimal solution to minimize the level-wise communication cost.

4. Extensive experiments are conducted with three system parameters: the network size, the data streams variability and the number of subscriptions. Compared to two approaches, the baseline Single-level Filter algorithm and the more refined Multilevel Filter Decomposition algorithm, our approach Multi-level Cooperative Filter(MCF) algorithm gives the best application-level and network-level performance. MCF is scalable: a linear increase in network sizes only produces a logarithmic increase in resource usage.

## 4.2 Problem Formulation

The stream data in our study are confined to numeric values. We let $\mathbb{X}$ denote the set of all data sources and let $x$ range over $\mathbb{X}$. Denote as $X = x_1, x_2, \ldots, x_T$ the entire sequence of data values generated by the data source $x$ at time interval 1, 2 until T and $c_s$ be the conformance threshold specified by a user subscription $s$. Denote as $\underline{X} = x_{i_1}, x_{i_2}, \ldots, x_{i_M}$ the sequence of data values delivered to the subscriber. *Conformance preserving property* (**P1**) requires $\underline{X}$ to be a *subsequence* of $X$, and furthermore, for all $x_{i_m} \in \underline{X}$ where $m \in [1, M]$ and $i_m \in [1, T]$, $|x_t - x_{i_m}| \leq c_s$, where $t \in [i_m, i_{m+1})$.

A subscription $s$ is a vector of four components: a unique subscription ID, data source $x$, conformance threshold $c_s$ and the subscriber.

We denote a filter based on a conformance threshold $c_s$ as $f_{c_s}$. It is constructed for each subscription with the last delivered data $\hat{x}_s$, the local state, as its center of alignment. The range of a filter $f$, denoted as $R(f)$, is defined as $[\hat{x}_s - c_s, \hat{x}_s + c_s]$. Dynamic filters differ from static ones in that, as the local state varies with every data delivery, a dynamic filter each time specifies a different range, i.e., it moves with each data delivery. After each time a new value is delivered, the state of the subscriber is refreshed and the filter is re-centered.

In general, filters can be used in two ways in data delivery: One is to demand any data that fall *inside* the range to be delivered to the subscribers. The other is to demand only data values that fall *outside* the range to be delivered. The two situations can be transformed into one another with easy adaptation, since to require a data item to fall outside a range $[c_1, c_2]$ is equivalent to requiring it to fall inside either of the two ranges $(-\infty, c_1)$ and $(c_2, \infty)$. In this paper, we would focus on the latter case, as users are interested in incoming data items that fall outside the specified bound.

**Definition 1.** ***Conformance Preserving Data Dissemination*** *is defined as the problem of designing a filter overlay for data dissemination which satisfies the property of (**P1**) for each subscriber, given a set of incoming data sources $\mathbb{X}$ and a set of subscribers each with some specified conformance thresholds.*

We also make the following basic assumptions when building our system. First, each subscriber or data source knows at least one network node, the *bootstrap* node. Second, without loss of generality, data values are converted to integers. All data items and hence the conformance thresholds are bounded.

For easy reference, Table 4.1 summarizes the symbols that we have discussed so far and are used throughout the rest of the chapter.

| Symbol | Meaning |
|---|---|
| $\mathbb{X}$ | the set of all data sources, we let $x$ range over $\mathbb{X}$ |
| $X$ | the entire sequence of data values generated by $x$ |
| $\underline{X}$ | the sequence of data values delivered to the subscriber |
| $s$ | the subscription |
| $c_s$ | the conformance threshold |
| $f_{c_s}$ | the filter based on the conformance threshold $c_s$ |
| $\hat{x}_s$ | the local state based on subscription $s$ |
| $R(f)$ | conformance range $[\hat{x}_s - c_s, \hat{x}_s + c_s]$ |

Table 4.1: Conformance preserving data dissemination for simple subscriptions: models and symbols

## 4.3 Conformance Preserving Data Dissemination

In this section, we discuss a series of attempts to design the index structure to gain a better understanding on how our final approach is derived and a number of interesting results we achieved along the way.

The task of conformance preserving data dissemination is to deliver the necessary data to the user with minimum dissemination cost. Due to the capacity limitation of each node, a collection of nodes are needed to store and process the subscriptions for each data source. We call these nodes *processors*. Other nodes simply forward messages until the destination is reached. We call them *forwarders*. Based on this understanding, we further divide the dissemination cost into the *maintenance cost* and the *delivery cost*. The maintenance cost is the communication cost for building and routing within the processors, i.e., the index structure of the dissemination network. The delivery cost is the communication cost for multicasting the data to individual subscribers once they pass through the index structure. For a fixed number of subscriptions and network topology at any given time, the delivery cost is minimized when $|\underline{X}|$ is minimized.

Hence, conformance preserving data dissemination consists of two aspects: First, given a conformance threshold and a data stream, how to minimize the size of subsequences delivered to individual subscribers; Second, given a collection of conformance thresholds, how to construct an overlay which minimizes the communication cost among the participating nodes while preserving the optimality of delivered subsequences. In the rest of this section, we will address these two aspects. Section 4.3.1 studies the problem of minimizing $|\underline{X}|$ and provides an optimal OnlineGreedy algorithm. Section 4.3.2, 4.3.3, 4.3.4, and 4.3.5 detail a series of our attempts to extend the result of Section 4.3.1 to a collection of nodes, namely, the Single-level Filter (SF), the Multilevel Filter Decomposition (MFD) and the final Multilevel Cooperative Filter (MCF) algorithm.

Figure 4.1 illustrates the dissemination layer of our conformance preserving data dissemination system. Different organization schemes, namely, the Single-level Filter (SF), Multiple Filter Decomposition (MFD) and Multilevel Corporative Filter (MCF) method, are treated as pluggable modules in the layer, each of which has to provide the basic publication and subscription functionality.

### 4.3.1 OnlineGreedy Is Optimal

In this section, we address the problem of minimizing the size of subsequences delivered to individual subscribers given a conformance threshold.

Each time a data source $x$ refreshes its data, the *processors* need to decide whether the newly published data value should be delivered to the subscribers. The processors use *the OnlineGreedy algorithm, which delivers a data item to*

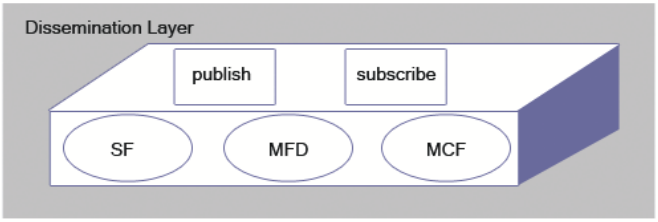



Figure 4.1: The dissemination layer of conformance preserving data dissemination

*a subscriber if and only if it differs greater than $c_s$ from the last delivered data value of that subscriber.*

Despite the simplicity of the *OnlineGreedy* algorithm, we prove in the rest of this section an interesting result: the *OnlineGreedy* algorithm actually gives the optimal approximation to the optimal off-line solution.

In order to measure how well the *OnlineGreedy* algorithm performs, we are seeking an optimal solution, called *GlobalOpt*, which produces the minimum-size subsequence $X_{opt}$ of $X$ while preserving conformance requirement. *GlobalOpt* is an off-line algorithm, which assumes the entire data stream $X$ is known in advance. Algorithm 3 gives the algorithm. For two data items $x_i, x_j \in X$ with $i \leq j$, we say $x_i$ *dominates* $x_j$ if $|x_i - x_j| \leq c_s$.

**Algorithm 3** GlobalOpt

**Input:** The stream data $X$, the conformance threshold $c_s$
**Output:** $X_{opt}$

1: $M \leftarrow X$
2: $X_{opt} \leftarrow \emptyset$
3: **while** $M \neq \emptyset$ **do**
4: Find the smallest $i$ such that $x_i$ dominates all $x_j, j \geq i$, where $x_i$ and all $x_j \in M$
5: $X_{opt} \leftarrow \{x_i\} \cup X_{opt}$
6: Delete from $M$ all $x_j, j \geq i$
7: **end while**

Unfortunately, an online stream data environment makes it impossible to know the entire data stream in advance as we can only process each data item as it comes in. In this case, what is the optimal performance one can achieve?

**Theorem 1.** *Given a user-specified conformance threshold $c_s$, no deterministic online algorithm can achieve an approximation ratio better than $\theta(n)$ for the optimal off-line solution, where $n$ is the length of the data stream.*

*Proof.* we use an adversary argument [14] to prove the theorem. Let the deterministic online algorithm be $A$. Suppose the data stream is revealed one by one by an adversary who attempts to force $A$ to deliver as much data as possible. The adversary keeps in hand two data streams each of length $n$. $X_1 =$

$\{4, 8, 9, 12, 6, 12, 6, \ldots, 12, 6\}$, i.e., after the first three values $4, 8, 9$, the stream toggles between 12 and 6 for the rest $n-3$ items. $X_2 = \{4, 8, 9, 5, 9, 5, \ldots, 9, 5\}$, i.e., after the first two values $4, 8$, the stream toggles between 9 and 5 for the rest $n-2$ items. The conformance threshold is set to 3. After the third data item 9 is revealed to the algorithm $A$, depending on the deterministic action taken by $A$, the adversary chooses one of the two data streams to reveal for the rest of process. Recall that we require that the sequence of data to be delivered is a subsequence of the source stream. This ensures that each decision made by $A$ is irreversible.

1. If $A$ decides to deliver this item 9, the adversary chooses to reveal $X_2$. It is clear that the optimal off-line solution for $X_2$ is to deliver $\{4, 8\}$, a sequence of length 2. However, now that $A$ has chosen to deliver 9, it has to deliver every data item thereafter to preserve conformance. In total, it would deliver all $n$ data. The approximation ratio is thus $n/2$.

2. If $A$ decides not to deliver this item 9, the adversary chooses to reveal $X_1$. It is clear that the optimal off-line solution for $X_1$ is to deliver $\{4, 8, 9\}$, a sequence of length 3. However, now that $A$ has chosen not to deliver 9, it has to deliver every data item thereafter to preserve conformance. In total, it would deliver $n-1$ data. The approximation ratio is thus $(n-1)/3$.

In conclusion, no deterministic online algorithm can achieve an approximation ratio better than $\theta(n)$ for the optimal off-line solution. □

It's trivial to see that OnlineGreedy gives an $n$-approximation for the optimal off-line solution. Hence the following corollary follows from the theorem.

**Corollary 1.** *Given a user-specified conformance threshold $c_s$, OnlineGreedy is optimal for all deterministic online algorithms.*

Although the performance of the best online algorithm is poor in the worst case, fortunately, as shown by our experiments in Figure 4.2, the OnlineGreedy algorithm provides a good approximation to the optimal off-line solution for real stock stream data. The dark solid line in Figure 4.2 shows the average ratios between the size of the delivered data computed by OnlineGreedy over that by GlobalOpt for different conformance thresholds from 1 to 63. Note that each point in the line is an average of ratios computed based on all 5000 stock stream data in our data set. Figure 4.2 also shows ratios for four individual stocks, namely, Yahoo, Ebay, Intel and Microsoft. For most conformance threshold values, the OnlineGreedy achieves 3/2-approximation for the optimal off-line solution.

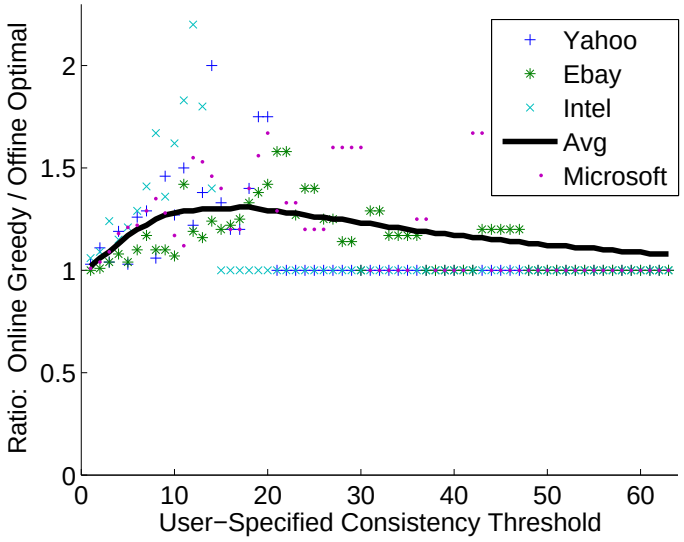



Figure 4.2: The average ratio of delivered data size by OnlineGreedy over GlobalOpt for real stock data

### 4.3.2 Single-level Filter Method

*OnlineGreedy* gives the algorithm for a single processor to process the incoming data. From this section onwards, we extend the result to a *collection of processors*; our goal is to construct an overlay to minimize the communication cost among the participating nodes while preserving the optimality of delivered subsequences.

The Single-level Filter (SF) method is a simple scheme where we have a flat organization of processors, each of which stores a portion of the directory and functions independently from each other. The mapping is done as follows: We replace $\delta$ by conformance threshold $c_s$ in hash function $h$ introduced in Section 3.3.

**Subscription.** Given a subscription $s$, the subscription layer uses the underlying Chord layer to route $s$ to the target node, whose ChordID immediately succeeds $h(x, c_s)$. The subscriber is assigned an initial state for subscription $s$ based on the most recent value of $x$ at the source.

**Publication.** Each time data source $x$ generates data, this data value has to be sent to the entire collection of processors for $x$, namely, a Chord ring segment of length $L$, starting at $h_1(x)$. Individual processor uses OnlineGreedy to determine whether the data value should be delivered to respective subscribers and the actual data delivery is done through the multicast primitive described in Section 3.1.2.

### 4.3.3 Optimization and Difficulties

Single-level Filter (SF) method proposes a flat organization of processors. Each time a data source generates a new data value, it has to be sent to the entire collection of processors, as illustrated in Figure 4.3(a). Considering the high data generation rate and large number of processors (due to the huge number of subscriptions), the communication cost is high.

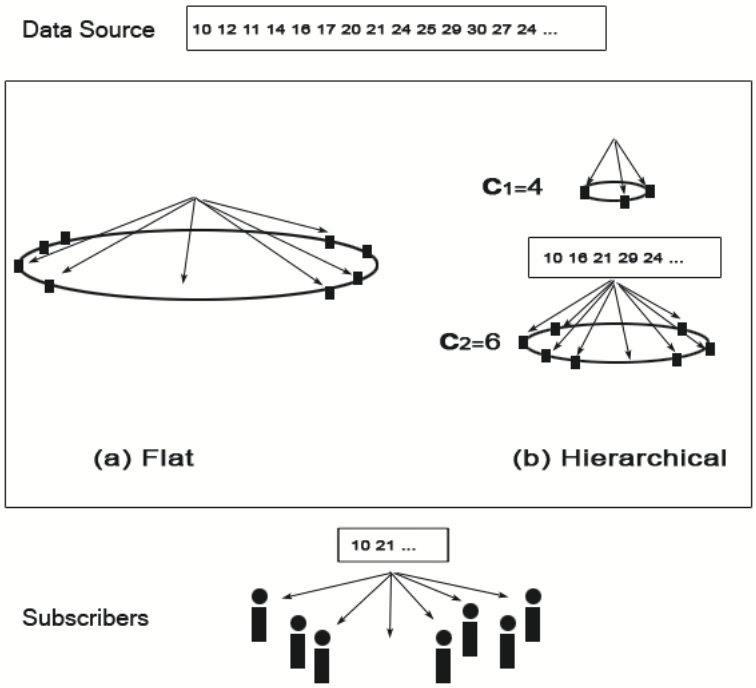



Figure 4.3: Organization: flat vs. hierarchical

In a large scale conformance preserving data dissemination system, subscriptions may potentially come from any node and a mixture of fine and coarse filters may co-exist in the network. Optimization could be achieved by using filters of smaller conformance thresholds to save unnecessary checking for larger ones. This idea is illustrated in Figure 4.3(b) where processors are organized into a hierarchy and only the data passing through the higher level filters are passed on to lower level filters.

Take a simple example for static filters. If we only deliver data falling *outside* a user-specified range, any data that can not pass a finer filter $[3, 4]$ will not be able to pass a coarser filter $[2, 5]$.

In general, in order for the pre-screening of finer filters to work for coarser filters, the *Inclusion Property* must hold. If we let $\mathbb{F}$ denote the universe of filters and $\underline{X}_i$ the data sequences filtered by the filters $f_i$, the Inclusion Property can

be formally stated as:

$$\forall f_1, f_2 \in \mathbb{F}, f_1 \subseteq f_2 \ \Rightarrow \underline{X}_1 \subseteq \underline{X}_2$$

Filters $f_i$ are intervals. We use $\subseteq$ to denote interval inclusion. The Inclusion Property says that if $f_1$ is a sub-interval of $f_2$, the data filtered by the filter $f_1$ is a subsequence of those by $f_2$. This property can be utilized for early pruning. As we have seen in our example, static filters usually exhibit the Inclusion Property and therefore admit effective optimization. However, things get tricky in the case of dynamic filters. The following example illustrates the difficulties.

| | |
|---|---|
| Data stream: | 10 16 **21** 19 32 25 19 13 **8** $\cdots$ |
| Data Delivered to $s_1(c_1 = 5)$: | 10 16 32 25 19 13 $\cdots$ |
| Data Delivered to $s_2(c_2 = 10)$: | 10 **21** 32 19 **8** $\cdots$ |

Figure 4.4: An example of data delivered to subscribers with different conformance thresholds

Figure 4.4 demonstrates a sample data stream and two subscribers with different conformance thresholds of 5 and 10 respectively. The first row shows the source data stream while the rest rows show the actual data sequences delivered to individual subscribers. The anomaly occurs when trying to deliver data value 21 and 8. Although they do not pass through the filter of 5, they pass through the filter of 10. Consider a naive delivery scheme where the data values are not checked against a coarser filter if they do not pass through a finer one, value 21 and 8 would not be delivered to subscribers of conformance thresholds of 10, where indeed they should.

Dynamic filters don't satisfy the Inclusion Property. This is due to the fact that local states are involved in constructing the intervals and local states may change over time. Hence special care has to be taken when organizing filters.

In the next two sections, we will discuss two hierarchical schemes to organize processors as an improvement on SF: the Multi-level Filter Decomposition (MFD) Method in Section 4.3.4 and the Multi-level Cooperative Filter (MCF) Method in Section 4.3.5.

### 4.3.4 Multi-level Filter Decomposition Method

In Multi-level Filter Decomposition (MFD) Method, larger filters are decomposed into smaller ones and filters are organized into a multi-level hierarchy.

Let's check the example given in Figure 4.5 to see how decomposition works in a two-level hierarchy. Suppose the user-specified conformance threshold is $c_s = 11$. We decompose $c_s$ into two conformance thresholds $c_s^1 = 3$ and $c_s^2 = 8$. We will explain why $c_s$ is decomposed in this way shortly. Assume $x_1$ is the

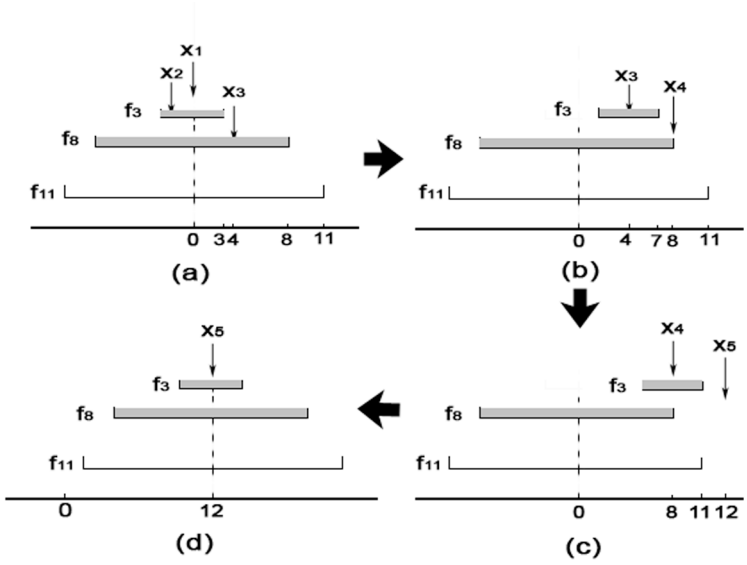



Figure 4.5: A stationary state period of two-level filter movement

last delivered data value to the user. At this moment, both filters $f_{c_s^1} = f_3$ and $f_{c_s^2} = f_8$ are aligned with $x_1$ being the center, as well as $f_{c_s} = f_{11}$. As another data item comes, e.g. $x_2$, if it falls within the range of $f_3$, it won't pass through the first level filter, as illustrated in (a). We move $f_3$ only if the incoming data item falls out of $f_3$'s range, e.g. $x_3$, at which point we would move $f_3$ to align with $x_3$, as in (b). Having passed through the first level filter, $x_3$ would reach the second level filter $f_8$. As it falls within $f_8$, we don't move $f_8$. If an incoming data item falls out of the ranges of both $f_3$ and $f_8$ like $x_5$ in (c), both filters would be moved to align with $x_5$, and $x_5$ will be delivered to the user, as shown in (d). Hence $x_5$ becomes the last delivered data item and the configuration of the filters is restored to a condition same as that in (a). We call the duration between (a) and (d) a *stationary state period*.

Now we show how we should decompose $c_s$ into $c_s^1$ and $c_s^2$ to preserve conformance. We have the following observations.

1. To preserve conformance, we have to ensure that $c_s^1 + c_s^2 \leq c_s$. Otherwise, in (c) for example, we would have the right boundary of $c_s^1$ lying outside of the right boundary of $c_s$. As a result, incoming data which pass through the original filter $f_{c_s}$ are not able to pass through $f_{c_s^1}$ now, and therefore will not be delivered to the user, violating the conformance requirement.

2. To prevent redundant data from being delivered, we have to ensure that $c_s^1 + c_s^2 \geq c_s$. Otherwise, in (c) for example, we would have the right boundary of $c_s^1$ lying inside of the right boundary of $c_s$. As a result, incoming data that do not pass through the original filter $f_{c_s}$ has a chance to pass through both $f_{c_s^1}$ and $f_{c_s^2}$ now, and therefore will be delivered to

the user.

It follows that

$$c_s^1 + c_s^2 = c_s \tag{4.1}$$

It is also easy to see that in order to use $f_{c_s^2}$, we also have

$$c_s^1 < c_s^2 \tag{4.2}$$

We may extend the constraints of constructing two-level filters to those of constructing multi-level ones. Let $k$ be the number of levels, then for a given conformance threshold $c_s$,

$$c_s = \sum_{i=1}^{k} c_s^i \tag{4.3}$$

$$c_s^i > \sum_{j=1}^{i-1} c_s^j \text{ for } 1 < i \leq k \tag{4.4}$$

In the rest of this section, we will describe a simple scheme that we adopt to satisfy constraints (4.3) and (4.4).

### Multi-level Hierarchy

We need to derive a multi-level hierarchy to fit in all conformance thresholds in $[1, max(c_s)]$ that satisfy the decomposition constraints (4.3) and (4.4). One easy way of decomposition is to represent the conformance thresholds as a number in base $b = \lceil \sqrt[k]{max(c_s) + 1} \rceil$,where $k$ is the number of levels.

Let's use an example to illustrate. Given 63 as $max(c_s)$ and a three-level hierarchy, we are to use base 4 encoding. We could easily determine the decomposition of any conformance threshold. For instance, $21_{(10)} = 111_{(4)} = 001 + 010 + 100_{(4)}$, $34_{(10)} = 202_{(4)} = 002 + 200_{(4)}$.

The encoding also tells which node within the logical structure that the subscription $s$ with conformance threshold $c_s$ should be stored. First, the number of non-0s within the encoding decides the number of composing sub-filters and hence the level at which $s$ is stored. For instance, a subscription $s$ with $c_s$ value of $21_{(10)} = 111_{(4)}$ should be stored at level 3 while $34_{(10)} = 202_{(4)}$ stored at level 2. After deciding on the level, the leading non-0 digit decides on which node at that level $s$ resides.

### Mapping Scheme

A linear order is imposed on the nodes of multi-level filter hierarchy from left to right and higher level to lower level, starting from 0. We call this the *index* of the logical node. Given the index number $\iota$, we apply $h(x, \iota) = h_1(x) + h_2(\iota)$, where

$x$ is data source, $h_2(\iota) = a\iota$ and $a = L/max(\iota)$, $L$ is the length of the Chord ring segment. The Chord node whose ChordID immediately succeeds $h(x, \iota)$ stores the logical node $\iota$. It's possible that several logical nodes are mapped to a single physical node.

#### Subscription

Given a subscription $s$, we convert $c_s$ to base $b$ encoding. Based on the encoding, we determine the index $\iota$ of the logical node and use $h(x, \iota)$ to decide the target physical node for storing the subscription.

However, as opposed to SF, the registration of a single subscription may involve registration at multiple logical nodes at different levels at initial stage. This is due to the fact that in MFD, the filters at lower level can only be achieved by passing through a path from level 1. For instance, in order to establish a filter of conformance threshold $21_{(10)} = 111_{(4)}$, we need to establish filters of $011_{(4)}$ and $001_{(4)}$ if they do not already exist. We call these filters *proxy filters*. We represent proxy filters as $\rho(x, c_s)$. Note that this representation shows that although multiple subscriptions to data source $x$ with the same conformance threshold $c_s$ may coexist, only one proxy filter exist for a $(x, c_s)$ pair. The sole purpose of the proxy filters is to achieve filters at lower levels. Hence proxy filters should be replaced by real subscriptions with the same $(s, c_s)$ pair if such subscriptions are issued later.

#### Publication

The filter hierarchy is built in such a way that the output of level $i$ is the input of level $i + 1$. If no output is generated at level $i$, no further forwarding is necessary. Algorithm 4 shows the detailed BasicMove algorithm.

**Algorithm 4** BasicMove for node u

**Input:** message $\langle$**pub**, $x_t$, $c_s^i\rangle$

**if** $x_t$ falls outside of the range of $f_{c_s^i}$ **then**
  Align $f_{c_s^i}$ with $x_t$
  **if** $f_{c_s^i}$ is not the bottom level filter **then**
    Find index $\iota$ of $c_s^{i+1}$
    Compute destination node $d = h(x, \iota)$
    Send $\langle$**pub**, $x_t$, $c_s^{i+1}\rangle$ to $d$ (follow the Chord routing)
  **else**
    Deliver $x_t$ to the subscriber
  **end if**
**end if**

### 4.3.5 Multi-level Cooperative Filter Method

MFD improves upon SF by building a filter hierarchy such that filters of smaller conformance thresholds can be utilized to save unnecessary checking for larger

ones. However, we observe two aspects which can be further optimized: first, there is no cooperation between filters at different levels. Although MFD preserves the conformance requirement, it may produce sub-optimal delivered subsequences, hence violating the optimality preserving property; second, MFD constructs the filter overlay in a convenient rather than efficient way. The goal that we are aiming for is to minimize small filters participating in higher levels. Please note that all these improvements are done in the logical layer. The mapping of our proposed Multi-level Cooperative Filter (MCF) method from logical layer to physical layer is identical to that of MFD. In the rest of this section, we are to address these two aspects.

### Cooperative Filtering

In this section, we first look at the fixed data change case, in which the difference between any two successive incoming data items is a fixed unit, i.e., $|x_t - x_{t+1}| = \sigma$, for all $1 \leq t \leq T-1$ where $\sigma$ is a positive constant. We show the formula to decompose $c_s$ in this case and prove that the BasicMove algorithm used in MFD gives the optimal delivery cost. Then we would study the varied data change case, in which the data difference requirement is dropped, and build upon the solution for the fixed data change case by a slight modification to the decomposition and the algorithm.

**Fixed Data Change.** Without loss of generality, we can always normalize the fixed change $\sigma$ to unit length. In this case, an incoming data item can only fall outside the current first level filter $f_{c_s^1}$ by a difference of one unit. Accordingly, the filter moves exactly $c_s^1 + 1$ units each time it is moved to a new alignment. Therefore, we obtain the second equation for the decomposition

$$c_s^2 = a \cdot (c_s^1 + 1) \tag{4.5}$$

for an integer $a \geq 1$. This equation guarantees that whenever $f_{c_s^1}$ moves outside of $f_{c_s^2}$, the right boundary of $f_{c_s^1}$ aligns exactly with the right boundary of the original filter $f_{c_s}$, preserving the optimality. Combining equations (4.1) and (4.5), we get the formula for the decomposition

$$c_s^1 = \frac{c_s - a}{a + 1} \tag{4.6}$$

For example, when $c_s = 11$, setting $a = 2$, we get $c_s^1 = 3$ and $c_s^2 = 8$. Notice that for a given $c_s$, there might be more than one choice of decompositions. Each of them would satisfy the conformance and optimality preserving requirements. The choice of decomposition is a result of the global optimization of communication cost across all user-specified consistencies, i.e., the filter overlay design, which will be discussed later in the section.

**Theorem 2.** *For stream data with fixed data change, we can compute a two-level cooperative filter such that it achieves optimal delivery cost and conformance*

*preserving with BasicMove algorithm.*

*Proof.* First, observe that it is sufficient to prove the theorem for each *stationary state period* as defined in Section 4.3.4 since the filters are restored to the same configuration at the beginning of each period. Each stationary state period starts with the two filters $f_{c_s^1}$, $f_{c_s^2}$ aligned with the last delivered data, and ends when $f_{c_s^2}$ moves. We first prove conformance preservation: Any data item whose difference from the last delivered one is greater than the specified conformance threshold $c_s$ will be delivered to the user, i.e., any data item $x_t$ falling outside of the original filter $f_{c_s}$ should not be caught by $f_{c_s^2}$ if it remains still. Before the movement of $f_{c_s^2}$, the incoming data must be caught by either $f_{c_s^1}$ or $f_{c_s^2}$. Observe that it takes $f_{c_s^1}$ two straight moves outside of the right boundary of $f_{c_s^2}$ to be able to catch $x_t$. Then before the last move of $f_{c_s^1}$, there must be a time when an incoming data item falls outside of both filters. As this would result in a move of $f_{c_s^2}$, which contradicts the fact that $f_{c_s^2}$ remains still throughout a stationary state period.

We now prove optimality preservation, i.e., no data item whose difference from the last delivered one is not greater than $c_s$ will be delivered to the user. Recall that a data item is delivered to the user when it falls outside $f_{c_s^2}$. As the data item comes with a fixed data change, there must be a time when it falls exactly on the boundary of $f_{c_s^2}$, which would prompt a move of $f_{c_s^1}$ to cover the entire gap between the boundaries of $f_{c_s^2}$ and $f_{c_s}$. This would prevent any data item falling in the range of $f_{c_s}$ to pass through both filters. □

**Varied Data Change.** The varied data change case allows wild data fluctuation and models the behavior of actual financial stream data. How does it affect the design of the two-level cooperative filter? We shall now discuss the necessary changes in the decomposition of conformance thresholds and filter movement algorithm so that we can still guarantee conformance and optimality.

A closer examination of the filter interaction shows that the difference from the fixed data change case lies in the situation when $f_{c_s^1}$ is still completely inside the range of $f_{c_s^2}$ and the next incoming data item "jumps" outside of $f_{c_s^2}$. In other words, if $f_{c_s^1}$ is not able to anticipate the range in which the next incoming data item will fall, it will not be able to cover the gap between $f_{c_s^2}$ and $f_{c_s}$ before any data item falls in it. In fact, we can still equip $f_{c_s^1}$ with this ability if we impose two additional rules on the conformance threshold decomposition. Let $\xi_x$ be an upper bound on the difference between any two consecutive incoming data items, i.e. $\xi_x = \max_{i=1}^{T-1}\{|x_t^s - x_{t+1}^s|\}$. We have

$$c_s^1 > \xi_x \tag{4.7}$$

$$c_s^2 \geq c_s^1 + \xi_x \tag{4.8}$$

This ensures that as long as the right boundary of $f_{c_s^1}$ is at least $c_s^1$ away from the right boundary $f_{c_s^2}$, the next incoming data item will not fall outside of $f_{c_s^2}$,

preventing unnecessary data from being delivered to the user in these cases. If we could also manage to prevent unnecessary data delivery when the right boundary of $f_{c_s^1}$ is within $c_s^1$ away from the right boundary $f_{c_s^2}$, we would achieve optimality for varied data change case. The idea is to move $f_{c_s^1}$ to cover the gap *in one step* whenever it finds its boundary to be within $c_s^1$ from that of $f_{c_s^2}$. The filters follow the *CooperativeMove* algorithm:

**Algorithm 5** CooperativeMove for node u

**Input:** message $\langle$**pub**, $x_t$, $c_s^i\rangle$

1: **if** $f_{c_s^i}$ is the bottom level filter **then**
2:  **if** $x_t$ falls outside of the range of $f_{c_s^i}$ **then**
3:  Align $f_{c_s^i}$ with $x_t$
4:  Deliver $x_t$ to the user
5:  **end if**
6: **else**
7:  **if** $x_t$ falls within $c_s^i$ from a boundary of $f_{c_s^{i+1}}$ **then**
8:  Align $f_{c_s^i}$ with that boundary of $f_{c_s^{i+1}}$
9:  **else**
10:  **if** $x_t$ falls outside of the range of $f_{c_s^i}$ **then**
11:  Align $f_{c_s^i}$ with $x_t$
12:  Find index $\iota$ of $c_s^{i+1}$
13:  Compute destination node $d = h(x, \iota)$
14:  Send $\langle$**pub**, $x_t$, $c_s^{i+1}\rangle$ to $d$ follow the Chord routing
15:  **end if**
16:  **end if**
17: **end if**

**Theorem 3.** *For stream data with varied data change, we can compute a two-level cooperative filter such that it achieves optimal delivery cost and conformance preserving with CooperativeMove algorithm.*

*Proof.* We prove both conformance preserving and optimality for each stationary state period as Theorem 2. We first prove conformance preserving. Each stationary state period starts with the two filters $f_{c_s^1}$, $f_{c_s^2}$ aligned with the last delivered data, and ends when $f_{c_s^2}$ moves. If an incoming data item $x_t$ falls more than $c_s$ far from the last delivered data, then it must fall more than $c_s^1$ from the boundary of $f_{c_s^2}$. Since we have $c_s^1 > \xi_x$, there must be a preceding data which falls outside of $f_{c_s^2}$ but within $c_s^1$ from its boundary. It follows that at this point $f_{c_s^1}$ will be aligned with the boundary of $f_{c_s^2}$. After this configuration, according to the CooperativeMove algorithm, for any data $x_t$ falls more than $c_s$ far from the last delivered data, it will be passed to $f_{c_s^2}$. Since it also falls outside of $f_{c_s^2}$, this will initiate a movement of $f_{c_s^2}$ and hence a new stationary state period.

We then prove optimality preserving. As we have $c_s^1 > \xi_x$ and $c_s^2 \geq c_s^1 + \xi_x$, before a data item $x_t$ falls outside of $f_{c_s^2}$, there must be one preceding data item falling within $c_s^1$ from the boundary of $f_{c_s^2}$. As a result, $f_{c_s^1}$ would align with the boundary of $f_{c_s^2}$. After this movement, any data passing through both filters must fall at least $c_s^1 + c_s^2 = c_s$ far from the last delivered data. Hence the

optimality. □

**Multilevel filter**

To extend the two-level filter to multilevel, we have the following constraints of conformance threshold decomposition. Let $k$ be the number of levels, then for a given conformance threshold $c_s$,

$$c_s = \sum_{i=1}^{k} c_s^i \tag{4.9}$$

$$c_s^1 > \xi_x \tag{4.10}$$

$$c_s^i \geq \sum_{j=1}^{i-1} c_s^j + \xi_x, \text{ for } 1 < i \leq k \tag{4.11}$$

For a given $c_s$, we decompose $c_s$ as follows. We use the formula $c_s^1 = \frac{c_s - a}{a+1}$ subject to the set of constraints to decompose $c_s$ into a two-level filter with conformance thresholds $c_s^1$ and $c_s^2$. If there exists at least one solution, then for each solution, we recursively decompose the bottom level. We call this decomposition *Multi-level Decomposition* (MuLeD). As we mentioned earlier, there could be more than one way to decompose a given $c_s$.

Once a conformance threshold is decomposed, the multilevel filter would follow the CooperativeMove algorithm for movements. We prove that the MCF thus constructed is able to preserve conformance and optimality. Before we prove the theorem, we would like to make the following observations concerning MCF.

**Observation 1.** A stationary state period of MCF ends if and only if the bottom level filter moves.

**Observation 2.** The filter at level $i$ moves if and only if the incoming data value $x_t$ falls away from the center of the filter for a distance greater than $\sum_{j=1}^{i} c_s^j$.

**Observation 3.** For any two levels $i$ and $i+1$ in the MCF, $R(f_{c_s^i}) \cap R(f_{c_s^{i+1}}) \neq \emptyset$.

Each of these observations can be proved either by induction of the number of levels or by straightforward reasoning from the CooperativeMove algorithm. Their proofs are omitted due to limitation of space.

**Theorem 4.** *Given a user specified conformance threshold $c_s$ and the number of levels $k$, we can compute a MCF $f_{c_s^1}, \ldots, f_{c_s^k}, k \geq 1$ such that it achieves optimal delivery cost and conformance preserving with CooperativeMove algorithm.*

*Proof.* We prove by induction on $l$, the number of levels of the MCF. For $l = 1$, the MCF is a single filter $f_{c_s}$. It's trivial to see that it preserves conformance and optimality. Suppose when $l = k, k \geq 1$, the MCF preserves conformance and optimality. For $l = k + 1$, we decompose the bottom level filter $f_{c_s^k}$ into two filters $f_{c_s^k}$ and $f_{c_s^{k+1}}$ by MuLeD. We prove conformance preserving and optimality for each stationary state period.

Consider an incoming data item $x_t$ falls further than $c_s$ away from the last delivered data. Due to Observation 3 and the fact that the bottom level filter $f_{c_s^{k+1}}$ remains still, the union of all the filter ranges covers a distance at most $c_s$ from the last delivered data, i.e., $|\bigcup_{i=1}^{k+1} R(f_{c_s^i})| \leq c_s$. As such, $x_t$ will not fall in any filter's range, in particular that of the bottom level, and will therefore be delivered to the user. Hence conformance preserving holds. For the optimality preserving, first observe that any data item which falls within $c_s^k + c_s^{k+1}$ from the last delivered data will not pass through. This is because we have proved in the two-level filter case that the last two-level decomposition operation in MuLeD for $c_s^k$ preserves optimality. We then left to consider a data item $x_t$ falling more than $c_s^k + c_s^{k+1}$ from the last delivered data. Since by supposition, a $k$-level MCF preserves optimality. Therefore, $x_t$ will not pass the $k$-level MCF . To do so, there must be a moment when all filters $f_{c_s^i}, i < k$ aligned with the boundary of the old filter $f_{c_s^k}$. We only have to show that we will reach the same configuration in the new $k+1$-level MCF. Observe that since $c_s^i > \xi_x$ for all $i$, there must be a preceding data $x'$ which falls between $c_s^{k+1}$ and $c_s^k + c_s^{k+1}$ from the last delivered data. This will initiate all filters $f_{c_s^i}, i < k$ to align with the boundary of $f_{c_s^k}$, which is itself aligned with the boundary of $f_{c_s^{k+1}}$ then. We thus reaching the same configuration as that in a $k$-level MCF.

We therefore proved that when $l = k + 1$, the MCF still preserves conformance and optimality. Since our MCF preserves conformance and optimality, it delivers the same sequence of data items as OnlineGreedy. Therefore, MCF also gives the optimal approximation ratio. □

**Filter Overlay**

In this section, we show how to choose a particular conformance threshold decomposition, and accordingly an MCF, for each user-specified conformance threshold and how to organize all the MCFs into a data dissemination overlay network such that the maintenance cost is minimized.

Let $C = \{c_1, c_2, \ldots, c_{|C|}\}$ be the set of all conformance thresholds specified by the users for the system for all data sources. For each $c_i \in C$, let $\Gamma_i = \{F_1^i, F_2^i, \ldots, F_m^i\}$ be the set of all feasible MCFs to decompose $c_i$ for a given maximum decomposition level $L$. Each $F_j^i \in \Gamma_i$ is an ordered sequence of conformance thresholds such that $F_j^i[l]$, the $l$-th value, is the conformance threshold for the $l$-th level filter in $F_j^i$. For example, suppose we pick $c_1 = 11 \in C$ and $(2, 9)$, $(3, 8)$ and $(5, 6)$ are three feasible MCF decompositions for $c_1$, $\Gamma_1 = \{F_1^1, F_2^1, F_3^1\}$ where $F_1^1 = (2, 9)$, $F_2^1 = (3, 8)$ and $F_3^1 = (5, 6)$. To minimize the maintenance cost of the data dissemination network, we first need to minimize number of conformance thresholds assigned to the first level. This is because for the filters at the first level, we need to multicast to each of them every item published from the data source. Only those data items passing through the first level filter will be multicasted to the filters at the second level, and so on. We assign

conformance thresholds level by level starting from the first level.

Let's look at the assignment for the first level with more details to illustrate the idea. Let $U_1 = \bigcup_{1 \leq i \leq |C|, F_j^i \in \Gamma_i} \{F_j^i[1]\}$, i.e., $U_1$ is the set of all first-level conformance thresholds for all feasible MCFs for all $c_i \in C$. If we choose a conformance threshold $c'$ and assign it to the first level, then for each $c_i \in C$, if there exists $F_j^i \in \Gamma_i$ such that $F_j^i[1] = c'$, this means $c_i$ can be decomposed with $c'$ as the conformance threshold for the first level filter. In such case, we say $c_i$ is *covered* by $c'$. Denote as $L_i$ the set of conformance thresholds assigned to level $i$. Given $\Gamma_i$ for each $c_i \in C$, how should $L_i$ be determined for each level $i$? To minimize the maintenance cost, we should select a $L_i$ as small as possible while ensuring that every $c_i \in C$ is covered. Suppose now we have fixed $L_1$, then each $c_i$ could be covered by more than one values in $L_1$. To minimize the data item passing through the first level and reaching the second level, we should pick the largest conformance threshold covering $c_i$. This is because a filter with a larger conformance threshold allows fewer data to pass. After we have chosen a particular $c'$ to cover $c_i$, we delete all MCF $F_j^i$ in $\Gamma_i$ such that $F_j^i[1] \neq c'$. This leaves a smaller $\Gamma_i$ for each $c_i$, and we go on to assign the second level.

Formally, we define the problem of determining $L_i$ for each level $i$ as

**Definition 2** (Minimum Filter Assignment(MFA))**.** *Given a set of conformance threshold $C$, and all $\Gamma_i$ for each $c_i \in C$ and the level $l$, let $U_l = \bigcup_{1 \leq i \leq |C|, F_j^i \in \Gamma_i} \{F_j^i[l]\}$, find a minimum-size $L_l \subseteq U_l$ such that every $c_i \in C$ is covered.*

It is easy to see at this point that MFA is closely related to the *Set Covering* problem. An instance $(U, S)$ of the *Set Covering* problem consists of a finite universe set $U$ and a family $S$ of subsets of $U$, such that every element of $U$ belongs to at least one subset in $S$. A subset in $S$ is said to cover its elements. The problem is to find a minimum-size subset $X \subseteq S$ whose members cover all elements in $U$.

**Theorem 5.** *MFA is NP-hard.*

*Proof.* We reduce the Set Covering problem to MFA. Given an instance $(U, S)$ of the Set Covering problem, we transform it to an instance of MFA as follows. Set $C = U$, $l = 1$; For each $s_i \in S$ and each $u \in s_i$, add into $\Gamma_u$ one $F^u$ such that $F^u[1] = i$. Thus constructed, there is a one-to-one correspondence between each $s_i \in S$ and $i \in U_1$, such that $u \in s_i$ if and only if $F^u[1] = i, F^u \in \Gamma_u$. It is easy to see that a solution for MFA is precisely a solution for the corresponding Set Covering problem. Since the Set Covering problem is NP-hard [20], and the reduction can be performed in polynomial time, we conclude that MFA is NP-hard. □

Having established the hardness of MFA, we use the greedy algorithm in Algorithm 6 to compute $L_l$, the set of conformance thresholds assigned to a given level $l$.

**Algorithm 6** Greedy-MFA

**Input:** The set of conformance thresholds $C$, the level $l$, the feasible MCF set $\Gamma_i$ for each $c_i \in C$

**Output:** $L_l$

1: $U \leftarrow \bigcup_{1 \leq i \leq |C|, F^i \in \Gamma_i} \{F^i[l]\}$
2: $L_l \leftarrow \emptyset$
3: **while** $C \neq \emptyset$ **do**
4: Select a $c' \in U$ which covers the most $c_i \in C$
5: Remove from $C$ all the $c_i$ covered by $c'$
6: $L_l \leftarrow L_l \cup \{c'\}$
7: **end while**

**Theorem 6.** *Greedy-MFA is a polynomial-time $H_n$ factor approximation algorithm for the MFA problem, where $n = |C|$ and $H_n = 1 + 1/2 + \cdots + 1/n \leq \ln n + 1$.*

By a similar technique as used in the proof of Theorem 5, it can be shown that we can also reduce MFA to the Set Covering problem. This correspondence between the instances of the two problems means that a polynomial-time $H_n$ factor approximation algorithm for the Set Covering problem is also a polynomial-time $H_n$ factor approximation algorithm for the MFA problem. Greedy-MFA is the exact counterpart of the greedy algorithm for the Set Covering problem. The proof of Theorem 6 is essentially the same as that of the greedy algorithm for the Set Covering problem. The readers are referred to [47] for proof details.

## 4.4 Performance Evaluation

Through the previous few sections, we have presented a series of attempts which lead to our final approach - Multilevel Cooperative Filter (MCF) Method. In this evaluation, we are to verify the superiority of MCF in large scale stream data dissemination networks. A further understanding is also expected to be obtained through the comparison with the other two schemes, namely the Single-level Filter (SF) and the Multi-level Filter Decomposition (MFD) method.

To this end, we build a stock monitoring application using real life stock quotes collected from Yahoo Finance [48]. The original dataset consists of 30 different stocks, collected in 1 minute interval from Nov. 11th, 2002 to Sep. 12th, 2003. The total data size is about 108MB. Since our evaluation is to simulate one-day events, We tailor the dataset such that each stock within a single day is treated as a separate data source. Hence we create a large pool of 5453 data sources. These data sources are of different changing rate. We classify these generated data sources into 10 groups based on the standard deviation and data sources are picked from these groups uniformly at random. Conformance thresholds range from 1 to 63, as scaled per data source.

### 4.4.1 Performance Metrics

To capture both the application level and network performance, we use the following performance metrics:

#### Network Performance Metrics

The metrics in this category measures the communication cost using the number of messages exchanged (or hop counts) on the Chord layer. For a data dissemination network, the communication cost consists of two components: *subscription cost* and *dissemination cost*. *Subscription cost* specifies the number of messages to register a subscription to the network. *Dissemination cost* can be further divided into the *maintenance cost* and the *delivery cost*. The maintenance cost is the communication cost for building and routing within the logical structure of the dissemination network. The delivery cost is the communication cost for multicasting the data to individual subscribers. For a fixed number of subscriptions at any given time, the delivery cost is minimized when the size of the delivered data sequence is minimized.

Sometimes, we may use the length of *delivered sequence* to reflect the delivery cost since the underlying multicast mechanism is the same.

For a data dissemination network, dissemination cost is dominant as subscriptions are relatively static while data are generated and pushed to individual subscribers continuously.

#### Application Level Performance Metric - Latency

This metric measures the end-to-end delay in terms of hop counts from the data source to the subscribers who need to update their local states. This metric measures the performance as seen by individual subscribers. The reason why time is not used is that it heavily depends on hop delays, and irregular hop delays caused by irrelevant factors may bias our evaluation.

We want to emphasize that latency and resource usage are two different metrics. A low latency implementation in our context does not necessarily lead to low resource usage. For instance, broadcast could easily achieve low latency since the maximum latency for individual subscribers is determined by the network diameter. However, the resource usage is prohibitively high. In fact, in a network, parallelism might help to reduce individual latency. To reduce the resource usage, however, we may want to merge as many messages as possible, which may inhibit the parallelism. Hence a good implementation should balance between the usage of parallelism and message merges.

### 4.4.2 Real Data Experiments

The first set of experiments examines the performance of our proposed MCF as compared to SF and MFD when network size increases. We vary the participating peers from 100 to 1600, doubling each time. We use the real stock quotes described at the beginning of this section as data sources. Note that we use log as the x-axis in all figures of this chapter and the next chapter, hence a line trend implies a logarithmic behavior.

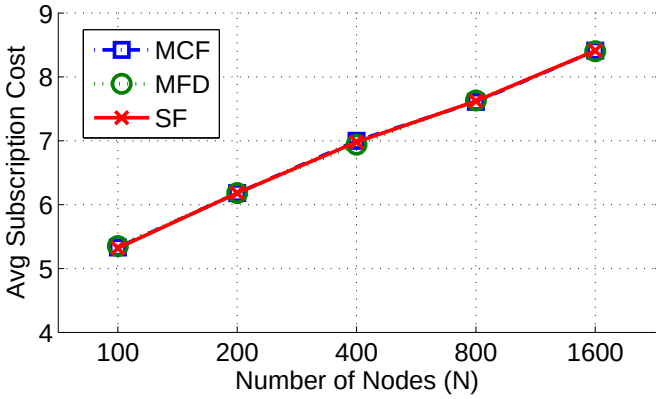
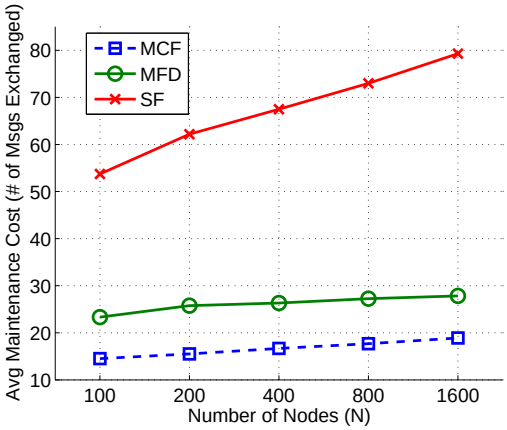



Figure 4.6: Average subscription cost as the network size increases

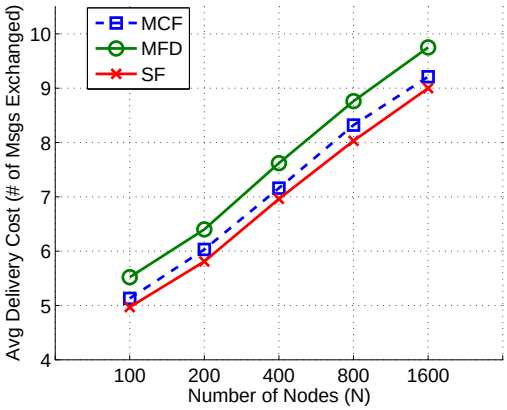
Figure 4.6 shows the average subscription cost of the three schemes as the network size increases. For Single-level Filter (SF) method, this is the number of messages exchanged when routing the subscription to any of the nodes within the chord ring segment. For both Multi-level Filter Decomposition (MFD) and our proposed Multi-level Cooperating Filter (MCF) method, this is the cost to route the subscription to the respective level according to the conformance threshold. As we can see from the graph, not much difference is exhibited among the three methods.

Figure 4.7a, 4.7b, 4.7c show the average dissemination cost of the three schemes per data generation. Conceptually the filter structure proposed in three schemes can be viewed as a single big filter to sift the incoming data stream. The maintenance cost measures how efficient this big filter behaves while the delivery cost measures the effectiveness. Figure 4.7a shows the average maintenance cost. SF exhibits a very high maintenance cost since each generated data $x$ has to be broadcast to the entire filter structure. Our proposed MCF outperforms MFD due to the optimal organization of the filter overlay so that less communication is exchanged among overlay nodes.

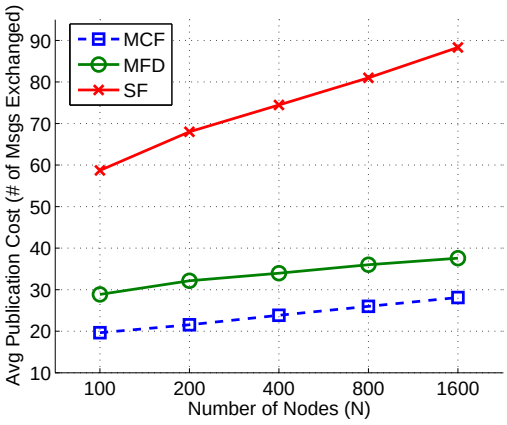
Figure 4.7b presents the average delivery cost. SF is the most effective in filtering out the data streams as individual filters use OnlineGreedy algorithm to filter out data values, which conforms to our discussion in Section 4.3.1. As shown in Figure 4.2, the OnlineGreedy algorithm achieves an average of 1.5-




a) Average maintenance cost




b) Average delivery cost




c) Average dissemination cost

Figure 4.7: Average dissemination cost per data generation as the network size increases

approximation for the optimal off-line solution in our stock applications. Since the underlying multicast mechanism is the same and the subscribers are the same, SF gives a delivery cost which is around 1.5-approximation to the optimal solution. We found in our experiments that some conformance thresholds can only be decomposed into two-level MCFs if we strictly follow the decomposition formulas. As a result, there could be many conformance thresholds crowding at the second level. The maintenance cost thus increased could offset the benefits offered by guaranteeing optimal delivery cost. We therefore adopt alternative filter decomposition in the experiments for better overall performance. This is why there is a slight gap between the optimal delivery cost produced by SF and that of ours. Although our proposed MCF trades effectiveness for efficiency, it still gives a performance pretty close to SF. MFD is the worst in filter effectiveness due to the lack of cooperation between multi-level filters. In the next section, we will have a clearer view on how high delivery-cost paralyzes the overall performance when number of subscriptions goes up.

Dissemination cost is an aggregate of the maintenance cost and the delivery cost. Figure 4.7c shows the average dissemination cost. Our proposed MCF significantly improves upon MFD by an average of 30% and SF by 70%.

Figure 4.8 shows the average latency (end-to-end delay) as observed by the individual subscriber. Without a proper design, while striving for a low dissemination cost, the performance on end-to-end delay might be compromised. Figure 4.8 clearly demonstrates the idea. MFD improves on SF with a lower dissemination cost, but with degradation on user experience. This might become a concern in applications where timely response is critical. Our proposed MCF gives the lowest average latency, which is appealing from the application point of view.

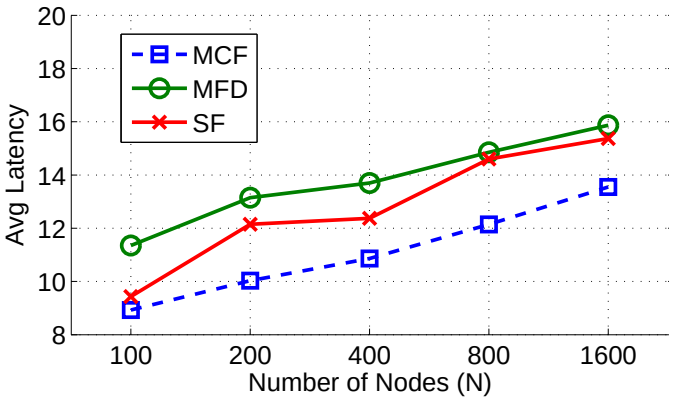



Figure 4.8: Average latency between data generation and data delivery as the network size increases

To summarize, the logarithmic behavior of our proposed MCF in all performance metrics tested on real data demonstrates good scalability. Furthermore,

it significantly reduces the dissemination cost as compared to SF and MFD by 70% and 30% respectively. MCF achieves the lowest latency which is an appealing feature as seen by the subscribers.

### 4.4.3 Synthetic Data Experiments

In the previous section, we have discussed the impact of network sizes. In this section, we focus on the impact of data streams and subscriptions. Multi-level filter structure trades effectiveness for efficiency, which leads to more local state updates and higher delivery cost. This doesn't seem to present a problem in the previous set of experiments with real data since the overall dissemination cost of a multi-level filter method (either our proposed MCF or MFD) is better than a single level filter method. But will this "deficiency" start to hurt the performance when we vary the data sources or the number of subscriptions? In order to establish a more controlled environment for experiments, we create synthetic data streams such that it is targeted at the possible vulnerable zone of both multi-level filter methods ( the value range around the boundary of decomposed filters). We want to see how our proposed MCF performs in regard to such attacks as compared against SF and MFD.

This is how we generate the synthetic data: given a conformance threshold $c$ and a decomposition of $c = c_1 + c_2$, if the upper level filter $c_1$ has not been moved over the boundary of second level filter $c_2$ in time, the effect of $f_{c_1}$ is equivalent to only $f_{c_2}$. Based on this understanding, the synthetic data are generated in the following way: for a particular $c$, we find $a$ and $b$ such that $b = c \mod \mathbb{G}$ and $c = a\mathbb{G} + b$ where $\mathbb{G}$ is the maximum pair-wise difference allowed within a data stream. We start with the initial value $x_0$, we execute the following block of actions: monotonically increase or decrease $x_0$ by $\mathbb{G}$ $a$ times and followed by $b$. We are guaranteed that before $a+1$ time steps away, we will hit a boundary case. Once the boundary case is hit, we repeat by re-execute the block of actions. To make the synthetic data look more like a stock data, we randomly mix increments with decrements when executing the block of actions. We create a pool of synthetic data and by varying the percentage of synthetic data included in the data sources, we vary percentage of increase in the length of the delivered sequence.

We fix the network size and the number of subscriptions per data stream, and vary the percentage of included synthetic data from 10% to 40% with 10% increments. Figure 4.9 shows the result using the total length of all delivered sequence to measure the effectiveness of our filter structure. SF is most effective in filtering and used as our baseline. Our proposed MCF method approaches SF, which shows good effectiveness. The MDF curve starts to diverge from the baseline. This implies a larger increment in delivery cost and eventually in dissemination resource usage. The narrow gap between the optimal online delivery cost and ours is due to the reason as mentioned in the discussion for

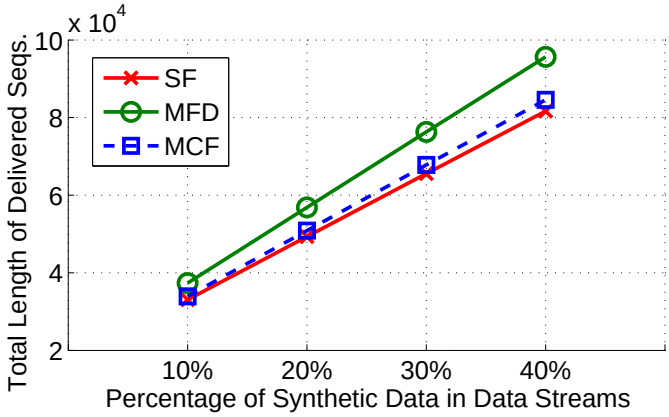



Figure 4.9: Total length of delivered sequence as the percentage of synthetic data increases

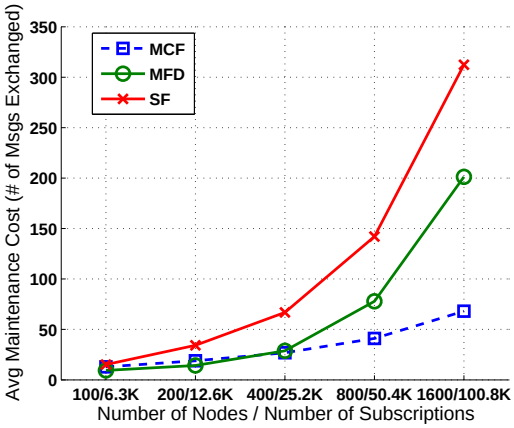
Figure 4.7b.

In order to understand the impact of synthetic data to the network level performance, we fix a moderate synthetic data percentage, say 20% and vary the network size. In this set of experiments, we consider another possible scenario where network size increase causes a proportional increase in users and subscriptions, which is not considered in previous experiments. With a fixed capacity of each node, this implies that a linear increase in the number of nodes storing the subscriptions and participating in the filter structure. This has an impact on the dissemination cost.

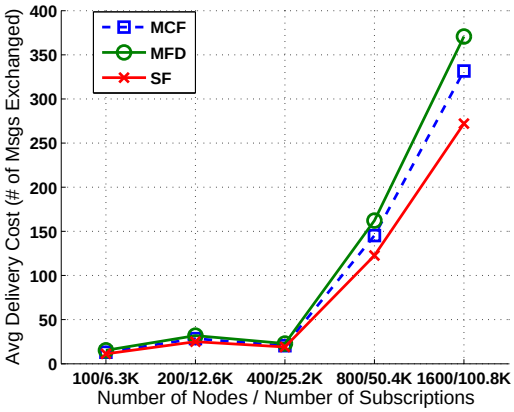
The average subscription cost in this experiment setup produces very similar result as Figure 4.6: all three schemes show the same average subscription cost as the network size (and also subscriptions) increases.

Figure 4.10a, 4.10b, 4.10c show the average dissemination cost of the three schemes per data generation. In all figures, the x-axis indicates both the network size and the number of subscriptions. All three curves exhibit an exponential trend on a log x-axis, which imply a linear increase as the number of the subscriptions increase. From experiments of the previous section, we know that this linear increase is not caused by network size increase, but the subscription increase. A linear increase in subscriptions per data source implies a potential linear increase in the total length of all delivered sequences (the delivery cost) and a potential linear increase in the number of nodes participating in the filter structure (maintenance cost).

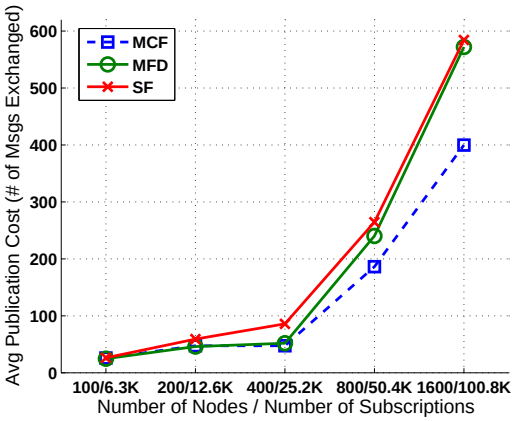
Figure 4.10a shows the average maintenance cost. SF still shows the highest maintenance cost while our proposed MCF shows the lowest. In fact, MCF shows a very slow growth curve which is almost logarithmic when the number of subscriptions is below 50K. Figure 4.10b presents the average delivery cost. As expected, MFD has the highest delivery cost while our proposed MCF exhibit a delivery cost close to SF. Figure 4.10c shows an interesting phenomenon which




a) Average maintenance cost




b) Average delivery cost




c) Average dissemination cost

Figure 4.10: Average dissemination cost per data generation as the network size increases

is not present in Figure 4.7c. The dissemination cost of MFD catches up with SF when the number of subscriptions increase to 100K and network size to 1600 nodes. This is where the high delivery cost of multi-level filter methods hurts the performance. However, our MCF does not exhibit such behavior. In fact, the dissemination cost growth rate of MCF is much lower than SF, which implies a much better overall performance.

Figure 4.11 shows the average end-to-end delay as observed by individual subscribers. Unlike Figure 4.8 where MFD gives the largest latency, SF gives the largest latency as the number of subscriptions increases to 50K. This is due to the fact that more subscriptions are affected by the high maintenance cost of SF and hence the average latency increases.

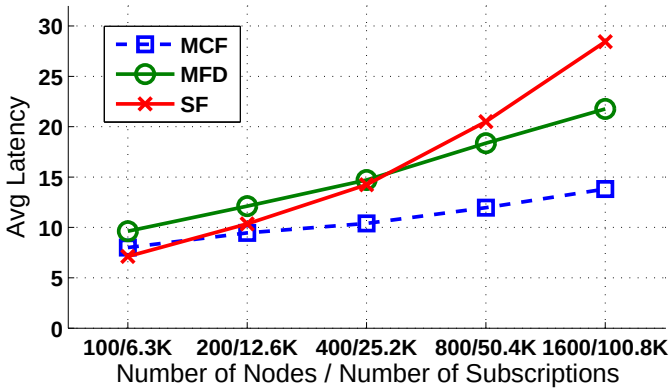



Figure 4.11: Average latency between data generation and data delivery as the network size increases

To summarize, the linear behaviors of all schemes in this set of experiments are caused by the linear increase of subscriptions, not the network size. With a higher delivery cost, MFD may hurt the performance when the network size and the number of subscriptions reach a certain level. However, our proposed MCF still maintains the best overall network level and application level performance.

### 4.4.4 Discussion on Network Dynamicity

Our proposed data dissemination system can be easily deployed in a controlled environment such as Gryphon [2] where network nodes are dedicated servers scattered around the world. If we consider an open network where network nodes are peers which can join and leave the system at any time, another dimension of complexity is added. This is what we call the *network dynamicity*. The stateless mapping within the Chord layer is the first attempt to handle the node join and leave gracefully. The mapping from our proposed Multi-level Cooperating Filter structure to the Chord ring segment is stateless, which is an obvious advantage when dealing with network dynamicity. However, when a node $n_i$ leaves the network, we need to move the subscriptions stored at $n_i$ to the next node $n_{i+1}$

or next few nodes in the Chord ring depending on the capacity of each node. If a node joins a network, the next node $n_{i+1}$ in the Chord ring will be informed by the underlying Chord layer and part of the subscriptions stored at node $n_{i+1}$ might be moved to the newly joined node to reduce the load on node $n_{i+1}$.

# Chapter 5

# Conformance Preserving Data Dissemination Based on Composite Subscriptions

## 5.1 Introduction

We propose an approximation based data dissemination model, conformance preserving data dissemination, as an initial step towards dealing with vast data streams where data accuracy can be traded for bandwidth. We envision that such an approximation model would be extremely useful with today's data proliferation.

We have discussed conformance preserving data dissemination based on a single data source in Chapter 4. As we observe the development in the pub/sub system, the earlier topic based pub/sub systems make pub/sub system a popular paradigm and later user needs call for a system supporting richer semantics, i.e., the content-based pub/sub system. We envision a parallel development in conformance preserving data dissemination. The user may specify composite subscriptions based on conformance thresholds to multiple data sources. For instance, in a structural health monitoring system, an alert is triggered only when the change of both the vibration and acceleration are above certain thresholds.

The main contributions of this work are as follows:

1. We extend the model to support a richer and more expressive subscription semantics, allowing the user interest to be specified as constraints on multiple data sources. Through Disjunctive Normal Form (DNF) transformation, arbitrary composite filters are decomposed into conjunctive filters.

2. We perform semantic optimization based on filtering strength and propose a hybrid method to support subscriptions of larger sizes.

3. Experiments are conducted to study the effects of two system parameters: network size and subscription dimension. Our system is scalable and the proposed hybrid method incurs low communication cost and low latency.

## 5.2 Problem Formulation

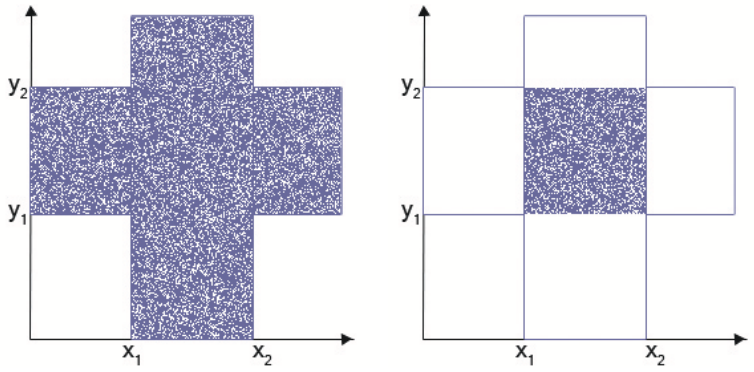
In Section 4.2, we have formulated the problem of conformance preserving data dissemination based on a single data source. In this section, we describe the

problem of conformation preserving data dissemination based on multiple data sources combined using conjunctive or disjunctive operators.

Each data source generates a sequence of data values at time interval 1, 2 until $T$. A subscription $s$ is a vector of three components: (unique subscription ID, *conformance predicate*, subscriber). A simple *conformance predicate* is a pair $(x,\ c_x)$ where $x$ is the data source and $c_x$ is the conformance threshold for this data source. We call this pair a *conformance predicate* as it specifies a condition for data delivery: a data values $x_t$ is filtered out by the system if $|x_t - \hat{x}| \leq c_x$, where $\hat{x}$ is the local state of subscription $s$ to data source $x$ . In another word, a data value $x_t$ is delivered to the subscriber if $|x_t - \hat{x}| > c_x$. The entire Chapter 4 is dedicated to addresses data dissemination based on simple conformance predicates.

A composite conformance predicate provides a richer semantics by combining several simple conformance predicates through conjunctive $\cap$ or disjunctive $\cup$ operator . The semantics of these two operators merits some discussion in the context of conformance threshold.

A simple conformance predicate $(x, c_x)$ defines a filter $f_{c_x}$ based on the conformance threshold $c_x$. A conjunction of two simple conformance predicates, $(x, c_x) \cap (y, c_y)$ for instance, denotes an *all* semantics: a data update is delivered to the user if the changes of values in all data sources are above the specified thresholds. A disjunction of two filters, on the other hand, denotes an *any* semantics: a data update should be delivered to the user if the change of values in any data source is above the specified threshold.




a) Conjunctive semantics b) Disjunctive semantics

Figure 5.1: The two semantics in composite conformance preserving data dissemination with the shaded area showing the data ranges that should be filtered out

This is better explained using a diagram. Figure 5.1 shows an example with two data sources $x$ and $y$. We install filters for each data source, i.e. a filter with conformance range of $[x_1, x_2]$ for data source $x$ and another one with conformance range of $[y_1, y_2]$ for $y$. In conjunctive semantics, $(x, c_x) \cap (y, c_y)$

specifies a new filter which filters out data values as long as one value is filtered out by either $f_{c_x}$ or $f_{c_y}$. The shaded area on Figure 5.1a shows the region to be filtered out by our system. In disjunctive semantics, $(x, c_x) \cup (y, c_y)$ specifies a new filter which filters out data values only when both $x_t$ and $y_t$ are filtered out by the respective filter $f_{c_x}$ and $f_{c_y}$, as shown in Figure 5.1b.

As illustrated in Figure 5.1, a conjunctive semantics defines a larger filter area, i.e. less data values would be delivered to the subscriber while a disjunctive semantics defines a smaller filter area.

For easy reference, Table 5.1 summarizes the symbols that we have discussed so far and are used throughout the rest of the chapter.

| Symbol | Meaning |
|---|---|
| $(x, c_x)$ | a simple conformance predicate which defines a filter $f_{c_x}$ |
| $f_{c_x} \cap f_{c_y}$ | conjunctive semantics, data values have to pass both filters $f_{c_x}$ and $f_{c_y}$ in order to be delivered |
| $f_{c_x} \cup f_{c_y}$ | disjunctive semantics, data values have to pass either filter $f_{c_x}$ or $f_{c_y}$ in order to be delivered |
| $\rho_{f_{c_x}}$ | filtering strength of $f_{c_x}$ |
| $Seg_x$ | ring segment in the Chord space to store all subscriptions to data source $x$ |

Table 5.1: Conformance preserving data dissemination for composite subscriptions: models and symbols

## 5.3 Composite Conformance Preserving Data Dissemination

Consider the following composite conformance predicates: $((a, 3) \cup (b, 2)) \cap ((c, 4) \cup (d, 10))$. A naive way to deliver data for such subscription would be to route data from all four data sources a, b, c and d to where subscription is stored. However, as data sources have no knowledge of where the subscription would be registered, this would imply a potential flooding to the entire network from all data sources.

Our goal is to construct a dissemination layer to optimize the communication cost for both data publishers and subscribers. In the rest of this section, we will discuss the optimization mechanisms that are employed in our conformance preserving data dissemination system based on composite subscriptions.

### 5.3.1 Optimization Mechanisms

A composite conformance predicate combines several simple conformance predicates using conjunctive and disjunctive operators. Figure 5.2 shows a conjunction and a disjunction of two filters as defined in Section 5.2. $(a, c_a) \cup (b, c_b)$ specifies a combined filter which is equivalent to two filters, $(a, c_a)$ and $(b, c_b)$,

functioning independently of each other. $(a, c_a) \cap (b, c_b)$, on the other hand, specifies a filter whose effect is equivalent to two filters chained together. Whether to activate the second filter is dependent on the result of the first filter. The chaining can be in either order $f_{c_a} \rightarrow f_{c_b}$ or $f_{c_b} \rightarrow f_{c_a}$.

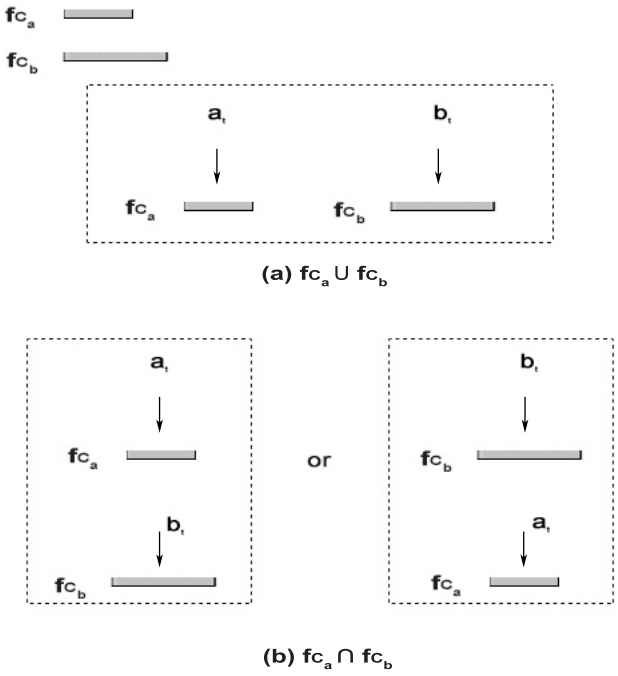



Figure 5.2: Filters of conjunctive or disjunctive semantics

Based on this understanding, a series of optimization mechanisms are employed to minimize the communication cost, namely filter decomposition, semantic optimization and duplicates removal.

### Filter Decomposition

A composite conformance predicate could come in a form of an arbitrary combination of simple conformance predicates. Optimization through filter decomposition tries to introduce structure into subscription composition.

We observe that for independent filters, $(a, c_a) \cup (b, c_b)$ for instance, we could split the subscription to two independent subscriptions $(a, c_a)$ and $(b, c_b)$ and inject them into the dissemination network independently of each other. Data values that passing through any one of the filters should be delivered to the user.

For chained filters, $(a, c_a) \cap (b, c_b)$, we use a rendezvous node to consolidate results from multiple filters. The subscriber itself could be the rendezvous node. This implies data values that pass through only one of the filters would still be

routed through the dissemination network until the subscriber consolidates the result and decides to abandon the values. Unnecessary bandwidth is wasted for delivering such data values. To save the bandwidth, the decision needs to be made earlier, this requires that a rendezvous node is installed as close to the data source as possible. This is described in the next section.

Based on the observation of conjunctive and disjunctive conformance predicates, we transform arbitrary composite conformance predicates into a Disjunctive Normal Form (DNF). Figure 5.3 revisits the previous example $(A \cup B) \cap (C \cup D)$, where $A$ represents a simple conformance predicate of $(a, c_a)$. A DNF is a two level expression where the simple conformance predicates in the first level are combined through $\cap$ operators and these composite predicates are further combined through $\cup$ operators in the second level. Hence expression $(A \cup B) \cap (C \cup D)$ can be transformed into a DNF $(A \cap C) \cup (B \cap C) \cup (A \cap D) \cup (B \cap D)$. We establish the filter equivalence: $f_{(A \cup B) \cap (C \cup D)} = f_{(A \cap C) \cup (B \cap C) \cup (A \cap D) \cup (B \cap D)}$.

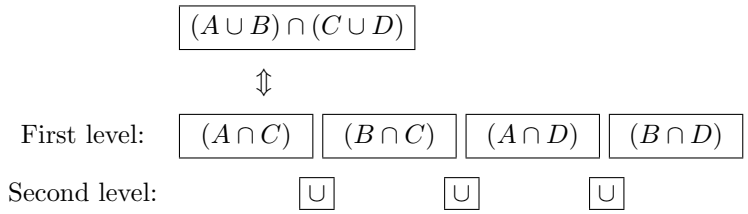



Figure 5.3: An example of filter decomposition

Transforming the composite conformance predicates into a DNF provides a method for filter decomposition. A complex filter can be achieved by decomposing it into a few simpler filters where each of them functions independently of each other to achieve the filtering required by the complex filter. After the filter decomposition, we are left with only conjunctive conformance predicates in the network. In the next section we will focus on mechanisms to optimize such predicates.

### Semantic Optimization

In this section, we discuss semantic optimization for conjunctive conformance predicates.

Recall that given a conjunctive conformance predicate $(a, c_a) \cap (b, c_b)$, the result of this complex filter is equivalent to chaining the two simple filters $f_{c_a}$ and $f_{c_b}$, as shown in Figure 5.2(b). The chaining can be done in any order. However, to save bandwidth, we want to choose a chaining order which filters out more data values as early as possible.

In order to achieve this, we want to install a filter of higher *filtering strength* earlier in the chaining. Filtering strength is determined by the conformance threshold as a larger threshold indicates a larger filtering range, and therefore more data values are filtered out. However, a conformance threshold itself is

insufficient to determine filtering strength. Consider the scenario were a data source $a$ reports sound frequencies while $b$ reports temperatures. These two data sources produce data with very different fluctuation rates: the source $a$ has a much higher data changing rate. So even if we have a conformance predicate $(a, c_a) \cap (b, c_b)$ where $c_a$ is larger than $c_b$, we may still want to install $f_{c_b}$ earlier in the chaining as the fast changing rate of $a$ might offset the larger tolerance level of $c_a$.

Based on the above observation, the filtering strength, denoted by $\rho$, is determined by two factors: the conformance threshold $c_x$ and the data changing rate $\Delta_x$ of data source $x$. We use the following formula to compute the filtering strength:

$$\rho_{f_{c_x}} = \frac{c_x}{\Delta_x} \tag{5.1}$$

It is obvious that the filtering strength is proportional to the conformance threshold required for the data source, and inversely proportional to the data changing rate of the data source. Nodes with higher filtering strength should be installed earlier in the filter chain.

Data changing rate $\Delta$ is a rather steady characteristic of a data source and its value is broadcast to the entire network at the time the data source joins the network. Further updates on the rate are broadcast to the entire network. However, this should be considered rather rarely and should not occupy much of the network bandwidth.

So far, we have discussed data dissemination when subscription dimension is 2, that is, a subscription of the form $S_1 \cap S_2$. We define the *subscription dimension* as the number of simple conformance predicates that a composite subscription is composed of. In the rest of this section, we discuss the approach that we adopt when the subscription dimension is larger than 2 (subscriptions of the form $\cap_i S_i$ where $i \in \{1, \ldots, n\}$).

On one end of the spectrum of parallelism, we have the rest of the simple conformance predicates chained sequentially and only when data passes the filters earlier in the chain do we send data request messages $REQUEST$ to data sources participating in later filters. We call such method *chaining* as shown in Figure 5.4(a). On the other end of spectrum, we send multiple $REQUEST$ messages simultaneously to all other data sources participating in the rest of the subscription and consolidate the final result when data values are returned back. Figure 5.4(c) illustrates this approach. Intuitively, the chaining method minimizes the message exchanged as only when data passes the early filters do we send messages to request for data values of later filters. However, as the resolving of predicates are chained, the turnaround time might be long. On the other hand, the *fanning* method sends all messages simultaneously, each of them goes through routing independently of each other, the turn around time is

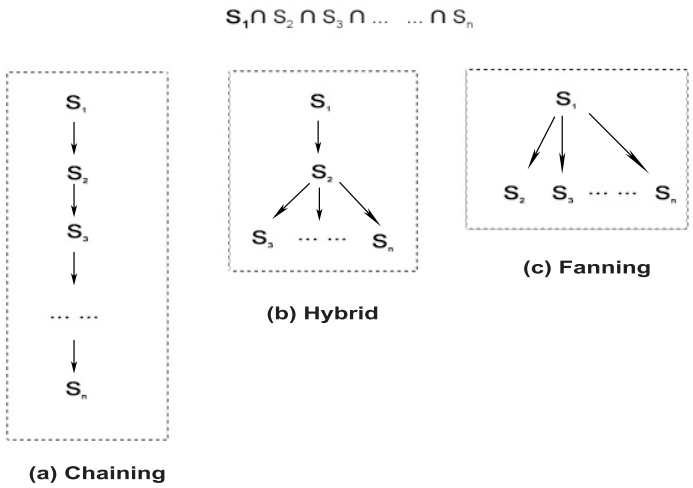



Figure 5.4: Methods for processing subscriptions of size $> 2$

faster as it is the maximum turnaround time of a single $REQUEST$ message. However, the number of messages exchanged is high as $REQUEST$ messages are sent regardless of whether they are needed.

We adopt a hybrid method which tries to harness the power of both the chaining and the fanning method, as shown in Figure 5.4(b). We use filtering strength $\rho$ to decide whether to chain or to fan. For a subscription $S_1 \cap S_2 \cap ... S_n$, the filters are sorted according to $\rho$. Assuming $\rho_1 > \rho_2 ... > \rho_n$, for all filters with a filtering strength $\rho_i > \theta_\rho$, we use chaining. For the rest of the filters, we use fanning. Since filtering strength $\rho$ is the ratio between conformance threshold and data changing rate, we choose $\theta_\rho$ to be 1 as $\rho > 1$ indicates that the tolerance for changing is larger than the changing rate. Hence it is very likely that the data would be filtered out by the filter, making it a good candidate for chaining. $\rho < 1$ indicates that the changing rate is larger than the tolerance bound. This implies that it is very likely that the data would pass the filter, and hence fanning these filters would be a good way to reduce the turnaround time.

### Duplicates Removal

After DNF transformation, the same simple conformance predicate may appear in multiple conjunctive predicates. For instance, after $(A \cup B) \cap (C \cup D)$ is transformed into $(A \cap C) \cup (B \cap C) \cup (A \cap D) \cup (B \cap D)$, $A$ appears in two conjunctive predicates $(A \cap C)$ and $(A \cap D)$. Since data values that pass the two filters $f_{A \cap C}$ and $f_{A \cap D}$ are delivered independently of each other, data values from $A$ are delivered twice.

To remove such duplicates, each node that participates in the filtering process for data source $x$ would keep a list of subscribers that data values are delivered to in the current interval. These nodes would suppress the duplicate delivery if the data values from the same data source have been sent to the same subscriber.

### 5.3.2 The Dissemination Layer

So far, we have discussed various optimization mechanisms. In this section, we describe how dissemination layer is constructed based on these optimization mechanisms.

We observe that after DNF transformation, the subscriptions are decomposed into sub-subscriptions which consist of either a singleton conformance predicate or conjunctive conformance predicates. As we described earlier in Section 3.3, we use SHA-1 to hash the data source $x$ to $k = h_1(x)$ in the Chord space, which marks the beginning of a contiguous Chord space that all subscriptions to $x$ occupy. We denote such Chord segment as $Seg_x$. All sub-subscriptions with singleton conformance predicates to data source $x$ are organized into multi-level cooperative filtering structure as described in Chapter 4. We use the first part of $Seg_x$ to store these filters and use the mechanisms detailed in Chapter 4 for data dissemination. We focus on data dissemination for composite conjunctive conformance predicates in this section. Specifically, we describe how the two functional interfaces are implemented: subscribe() and publish(). Although we are only using the second part of $Seg_x$ for storing such filters, we use $Seg_x$ in this section for simplicity.

**subscribe()**

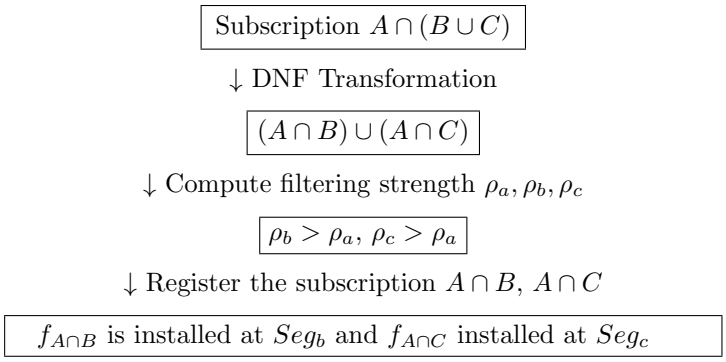



Figure 5.5: Example subscription processing

Figure 5.5 uses an example to demonstrate the process of subscription registration. Consider a subscription $A \cap (B \cup C)$. First the subscription is transformed into a DNF and two sub-subscriptions are created based on this transformation: $(A \cap B)$ and $(A \cap C)$.

To register subscription $(A \cap B)$, we compute the filtering strength $\rho_a$ and $\rho_b$ using Formula 5.1. As $\rho_b > \rho_a$, we want to install $f_{c_b}$ earlier in the filter chain. Hence the subscription is stored in the ring segment $Seg_b$. This is achieved by routing the subscription towards $h_1(b)$ and storing the subscription at the first node that it encounters in $Seg_b$. Similarly, as $\rho_c > \rho_a$, subscription $A \cap C$ is stored in the ring segment $Seg_c$.

**publish()**

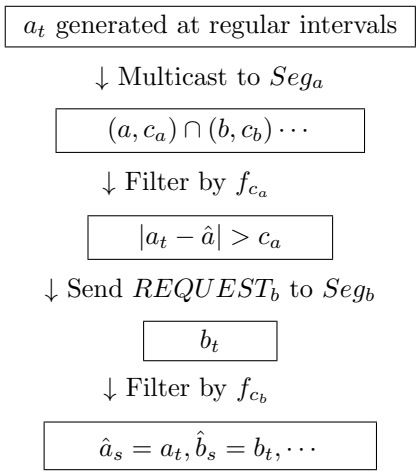



Figure 5.6: An Example of data publication

Each node in $Seg_x$ stores local states for each subscriptions according to the data sources involved. For instance, if we have a subscription $A \cap B$ stored in $Seg_a$, we store local states $(\hat{a}, \hat{b})$ for data source $a$ and $b$.

Data sources generate data values at regular intervals. Figure 5.6 illustrates an example of data publication. At each interval, the data value $a_t$ is multicast to $Seg_a$. Each node in $Seg_a$ processes all subscriptions that it stores in the following manner: consider a subscription $(a, c_a) \cap (b, c_b)$. $a_t$ is compared against the local states of $\hat{a}_s$ first. Only when $a_t$ passes the filter $f_{c_a}$ does the use of second filter in the composite conformance predicates is activated. As $Seg_a$ is only populated with values from data source $a$, upon activating the second filter, it will send a value request message $REQUEST_b$ to a random node in $Seg_b$. Once it receives $b_t$, the second filter is used to determine whether $b_t$ passes the filter. The steps of sending $REQUREST$ messages to data sources and filtering received values are repeated if the subscription dimension is greater than 2. As described in the previous section on semantic optimization, depending on the filtering strength, we decide whether we want to perform these steps in sequence or in parallel. Only when values pass through all filters $f_{c_a}, f_{c_b}, \cdots$, the data values should be delivered to the subscriber through multicast. At the same time, the local states $\hat{a}_s, \hat{b}_s, \cdots$ should be updated.

*REQUEST* messages for the same data source are only sent once to further save the bandwidth.

**Design Justification**

We have described dissemination layer for conjunctive conformance predicates. Compared with Chapter 4, although we used filter chaining to save bandwidth, we didn't choose to further improve the performance by building a filter hierarchy such that the finer filters can be utilized to save the checking for coarser ones. This is due to the following two reasons:

- As pointed out in Section 4.3.3, *inclusion property* doesn't hold for conformance thresholds. This situation is worsen by the presence of composite conformance predicates. Consider a filter $f_{c_a}$. When only single data sources are involved, $f_{c_a}$ would function the same as long as the same value is specified for $c_a$, say 5. However, in the case of composite filters, $f_{c_a}$ would function differently depending on which other filters it is combined with through $\cap$ operator. For example, consider the subscriptions $s_1 = (a, 3) \cap (b, 4)$ and $s_2 = (a, 3) \cap (b, 5)$, suppose both the subscriptions have the same local states $\hat{a} = 10, \hat{b} = 12$, given newly generated data values $a_t = 14, b_t = 17$, the local states for $s_1$ would be updated with the new values as both data values of $a$ and $b$ pass the filters while the local states for $s_2$ are unchanged. This observation indicates that a filter hierarchy should be built based on composite filters rather than individual filters.
- Composite filters are exponentially larger than single filters in number. The cost of building an effective multi-level filter hierarchy, if possible, based on composite filters would outweigh the possible performance gain by building such systems.

## 5.4 Performance Evaluation

In this section, we evaluate the effectiveness and efficiency of our proposed data dissemination network for composite conformance thresholds using the stock monitoring application that we have introduced earlier in Section 4.4. The same underlying DHT layer is utilized which provides the basic routing and multicast functionalities. In the dissemination layer, we extend the functionality of publish() and subscribe() to handle composite conformance thresholds as described in Section 5.3.2.

### 5.4.1 Experiment Setup and Evaluation Metrics

The simulation environment is built with a Chord simulator that we have implemented using a discrete event-driven simulator called PARSEC developed at

UCLA. The simulation is for up to 1600 nodes. As described in Section 4.4, data sources are of different changing rate and we classify them into 10 groups based on the standard deviation. Data sources belonging to the same group are of similar changing rates.

We use the same performance metrics introduced in Section 4.4.1 to measure the application level and network level performance, namely subscription cost, dissemination cost and latency. Dissemination cost is further divided into maintenance cost and delivery cost where the maintenance cost measures the efficiency of our in-network filtering structure while the delivery cost measures the effectiveness of the filters.

### 5.4.2 Experiment Results

As described in Section 5.3.1, arbitrary composite conformance predicates can be transformed into a DNF and after filter decomposition, we are left with only conjunctive conformance predicates in the network, $S_1 \cap S_2 \cap ... S_n$, where $n$ is the subscription dimension. Hence it suffices to experiment with these predicates to understand the performance of arbitrary composite conformance predicates.

We vary the participating nodes from 100 to 1600, double each time. The data values are published from each data source in a one-minute interval. At any time interval, we have around 6000 subscriptions registered in the network, potentially from any node in the network. Each $S_i$ in subscriptions $S_1 \cap S_2 \cap ... S_n$ is generated by picking a data source from the 10 groups uniformly at random. The conformance thresholds range from 1 to 63, as scaled per data source.

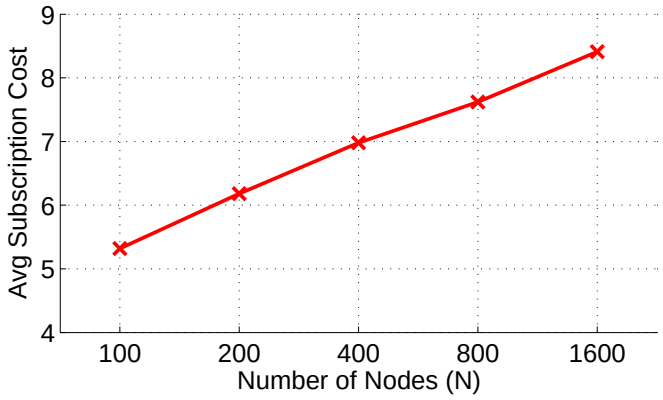



Figure 5.7: Average subscription cost as the network size increases

Figure 5.7 shows the average subscription cost of registering a subscription as the network size increases. As we described earlier in Section 5.3.2, a subscription $s$ is stored at one node in $Seg_x$, where $x$ is the data source which has the highest filtering strength that subscription $s$ subscribes to. Hence the

cost of subscription is simply the number of messages exchanged when routing the subscription to any of the nodes in $Seg_x$. The average cost shows a logarithmic growth as the network size increases, conforming to the Chord routing characteristic.

In a data dissemination network, the dissemination cost and latency determine the performance of the system. We are interested in two factors that may affect them: the network size and the subscription dimension. In the rest of this section, we will study the impact of these two factors.

### The Impact of Network Size

The first set of experiments are performed to evaluate the impact of network size on the dissemination cost and latency. We used a fixed subscription dimension of 2, i.e., subscriptions that are of the form $S_1 \cap S_2$.

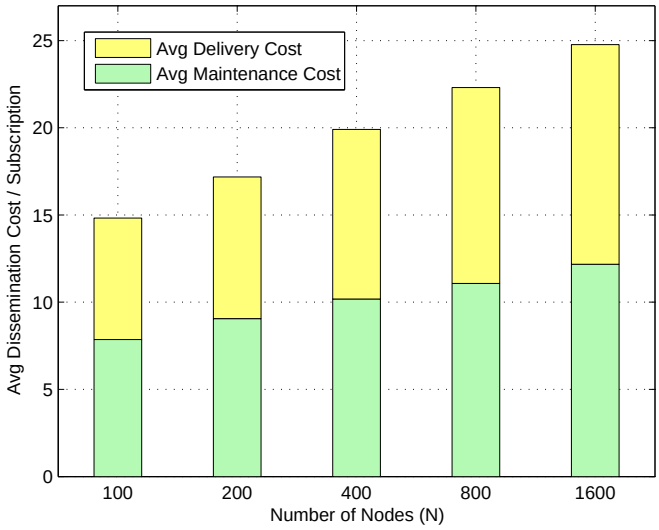



Figure 5.8: Average dissemination cost as the network size increases

Figure 5.8 shows the average dissemination cost per subscription. Dissemination cost consists of two portions: the maintenance cost and delivery cost. The bottom green bar shows the maintenance cost while the top yellow bar shows the delivery cost. As the network size increases, we observe a logarithmic growth in maintenance cost, delivery cost and the overall dissemination cost. This demonstrates good scalability of our system. Figure 5.9 shows the average latency (end-to-end delay) as observed by the individual subscriber. As we discussed in Section 4.4, while striving for a low dissemination cost, the performance on end-to-end delay might be compromised. Figure 5.9 clearly demonstrates that our proposed system gives an average latency only increased logarithmically with the network size increase, which is appealing from the ap-

plication point of view.

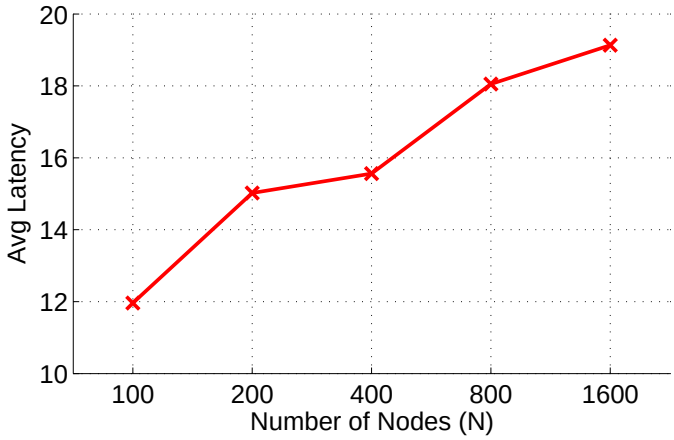



Figure 5.9: Average latency between data generation and data delivery as the network size increases

### The Impact of Subscription Dimension

In this set of experiments, we evaluate the impact of subscription dimension. A subscription is of the form $S_1 \cap S_2 \cap ...S_n$. To this end, we fixed the network size to be 1600 and vary the subscription dimension from 2 to 5.

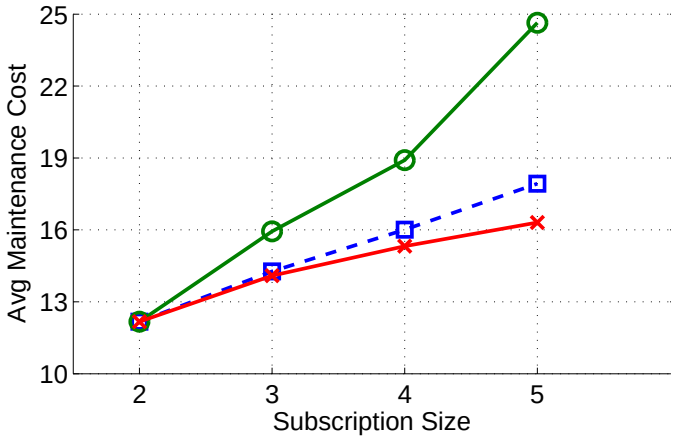



Figure 5.10: Average maintenance cost as the subscription dimension increases

We compare our hybrid approach with the chaining and fanning method. The three methods only differ in how filters are activated, not the results produced. Hence the delivery costs are the same for all three methods. We exclude the delivery cost and use only maintenance cost and latency for comparison.

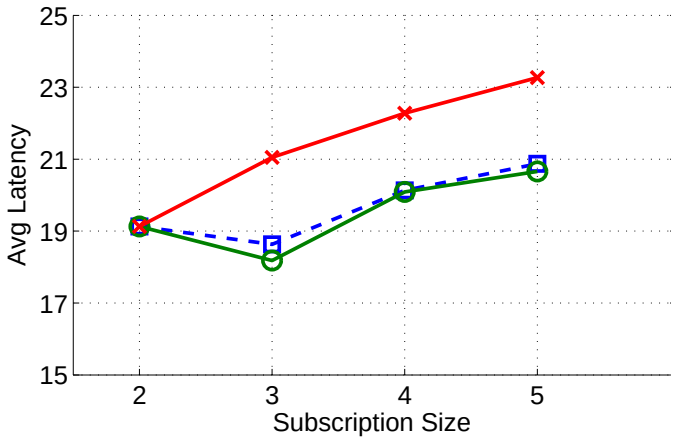



Figure 5.11: Average latency as the subscription dimension increases

Figure 5.10 shows the maintenance cost and Figure 5.11 shows the latency. The chaining method gives the lowest maintenance cost while fanning method gives the highest. However, the fanning method demonstrates lowest latency while the chaining method the highest. Our hybrid method harnesses the power of both approaches and achieves a performance close to the best performance of the two approaches in both maintenance cost and latency.

# Chapter 6

# Conclusion

In today's applications where dealing with vast stream data becomes a norm rather than an exception, it is in urgent need to design data dissemination systems in large scale. We identified a new pattern of data dissemination based on conformance constraints where data accuracy can be traded for low bandwidth. We formally defined the problem of conformance preserving data dissemination and addressed two types of conformance data dissemination problems that we are interested in: data dissemination based on simple subscriptions and data dissemination based on composite subscriptions.

For simple subscriptions, we proposed a Multilevel Cooperative Filter (MCF) overlay. Central to MCF are two key ideas: *cooperative filters* and *filter-based overlay*. Cooperative filters make sure that a filter with large conformance threshold can be decomposed into a sequence of smaller ones without compromising conformance preserving property. Filter-based overlay strives to make all such decompositions share as many common smaller filters as possible and thus minimizes the total number of filters to check. We described an online greedy algorithm to compute the minimum-size data sequence for dissemination and proved that it gives the optimal approximation ratio to the optimal off-line solution for all deterministic online algorithms. We then showed that our multilevel cooperative filter algorithm generates the same dissemination sequence as the online greedy algorithm, thus proving the optimality of our approach. We further proved the NP-hardness of the filter overlay construction and give a $O(\ln n)$-approximation algorithm to minimize the level-wise communication cost.

We extended the model to support a richer and more expressive subscription semantics, allowing the user interest to be specified as arbitrary conformance predicates combined using logical operators on multiple data sources. Through Disjunctive Normal Form (DNF) transformation, arbitrary composite filters are decomposed into conjunctive filters. We then used a hybrid method based on filtering strength to support these conjunctive filters with low communication cost and low latency.

We have built a stock monitoring application using real life stock quotes collected from Yahoo Finance to evaluate the performance. A variety of experiments have been conducted to verify our design choices and deepen our

understanding of the impact of various system parameters on both application-level and network-level performances. The simulation results suggest that the approaches are feasible to be deployed in large scale networks.

# Future Work

We see two important directions for future research. First, the work in this research on quantitative data can be extended to qualitative data, namely text streams, so that efficient data dissemination could be achieved based on changes of text content. Second, a more general form of dynamic filters based on monotonically increasing or decreasing functions can be further explored.

## Quantitative Data vs Qualitative Data

Micro-blogging, as exemplified by the phenomenal success of Twitter, has become a new form of social communication to express opinions, broadcast news, or communicate with friends. People often comment on events in real time, with several hundred micro-blogs (tweets) posted each second for significant events.

Text streams generated by micro-blogging services share some common characteristics with our quantitative data streams:

- Text streams are generated on a continuous basis and in a large volume. Upon event bursts, many messages are redundant in the sense that the same text sources are rephrased or "retweeted" when circulating within the system.
- Stream filtering is needed to remove redundant information and deliver interesting updates to the users.

Twitter, and many other existing micro-blogging services, are centralized systems. Although this thesis targets at quantitative data streams only, if we could model the content change of text streams in a quantitative way, the framework that we have proposed would offer a distributed alternative to the existing infrastructure.

New event detection, also known as first story detection, is an important task in topic detection and tracking in the area of information retrieval to detect topic shift or change based on the similarities among documents. New event detection based on social media such as Twitter streams starts to attract attentions such as [29, 36]. These efforts provide a basis to model the content change of social media streams in a quantitative way.

## General Form of Dynamic Filters

In this thesis, we identify one useful type of dynamic filters, namely the conformance threshold. More general form of dynamic filters, such as those based

on monotonically increasing or decreasing functions of values, remain to be explored. One rather common case is likely to be changes in terms of percentages. For example, consider the requirement that a stock quote be delivered to a user only when it differs from the local state by at least 50%. Instead of dealing with conformance thresholds, we are dealing with percentage of change. The decomposition proposed in our approach might still help when dealing with multiple filters of different percentage values. However, special care has to be taken: when dealing with conformance thresholds, the decomposition follows the addition rule: $f_{50}$ would be decomposed as $f_{20}$ and $f_{30}$. However, in the case of percentage of change, the decomposition follows the multiplication rule. Note that if we use logarithmic arithmetic, we might still be able to use some form of addition rule. However, the decomposition is no longer symmetric: the left and right intervals are not of the same magnitude on the log scale.

# References


[1] D. J. Abadi, D. Carney, U. Cetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, and S. Zdonik. Aurora: A new model and architecture for data stream management. *The VLDB Journal*, 12(2):120–139, 2003.

[2] M. K. Aguilera, R. E. Strom, D. C. Sturman, M. Astley, and T. D. Chandra. Matching events in a content-based subscription system. In *Proc. PODC*, Atlanta, GA, USA, 1999. ACM Press.

[3] Artur Andrzejak and Zhichen Xu. Scalable, efficient range queries for grid information services. In *Proceedings of the 2nd IEEE P2P*, pages 33–40, 2002.

[4] James Aspnes and Gauri Shah. Skip graphs. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 384–393, 2003.

[5] R. Baldoni, C. Marchetti, A. Virgillito, and R. Vitenberg. Content-based publish-subscribe over structured overlay networks. In *Proc. IEEE ICDCS*, 2005.

[6] D. Barbara and H. Garcia-Molina. The demarcation protocol: a technique for maintaining linear arithmetic constraints in distributed database systems. In *Proc. EDBT*, London, UK, 1992. Springer-Verlag.

[7] M. Bender, S. Michel, S. Parkitny, and G. Weikum. A comparative study of pub/sub methods in structured p2p networks. *Databases, Information Systems, and Peer-to-Peer Computing*, pages 385–396, 2007.

[8] P. Cao and C. Liu. Maintaining strong cache consistency in the world wide web. *IEEE Transactions on Computers*, 47(4), 1998.

[9] D. Carney., U. Cetintemel, M. Cherniack, C. Convey, S. Lee, G. Seidman, M. Stonebraker, N. Tatbul, and S. Zdonik. Monitoring streams - a new class of data management applications. *The VLDB Journal*, 12(2):120–139, 2003.

[10] A. Carzaniga, M. J. Rutherford, and A. L. Wolf. A routing scheme for content-based networking. In *Proc. IEEE INFOCOM*, 2004.

[11] M. Castro, P. Druschel, A. Kermarrec, and A. Rowstron. SCRIBE: A large-scale and decentralized application-level multicast infrastructure. *IEEE Journal on Selected Areas in communications (JSAC)*, 20(8), 2002.

[12] L. Chen and G. Agha. State aware data dissemination over structured overlays. In *The Sixth IEEE International Conference on Peer-to-Peer Computing.*

[13] L. Chen, K. S. Candan, D. Agrawal, D. C., and J. Tatemura. On overlay schemes to support point-in-range queries for scalable grid resource discovery. In *The Fifth IEEE International Conference on Peer-to-Peer Computing*, Aug 2005.

[14] J. G. Erickson. *Lower Bounds for Fundamental Geometric Problems*. PhD thesis, 1996. Chair-Raimund Seidel.

[15] Adina Crainiceanu et al. P-ring: An index structure for peer-to-peer systems. Cornell Univ., Comp. and Info., Scieence TR2004-1946, 2004.

[16] Adina Crainiceanu et al. Querying peer-to-peer networks using p-trees. In *WebDB'2004*, 2004.

[17] Ion Stoica et al. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM*, pages 149–160, 2001.

[18] K. Aberer et al. Advanced peer-to-peer networking: The P-Grid System and its Applications. *PIK Journal, Special Issue on P2P Systems*, 2003.

[19] N. Harvey et al. Skipnet: A scalable overlay network with practical locality properties. In *USITS03*, 2003.

[20] M. Garey and D. Johnson. *Computers and Intractability*. W.H.Freeman and Company, New York, NY, 1979.

[21] P. B. Gibbons, B. Karp, Y. Ke, S. Nath, and S. Seshan. Irisnet: An architecture for a worldwide sensor web. *IEEE Pervasive Computing*, 02(4):22–33, 2003.

[22] Gnutella. http://gnutella.wego.com/.

[23] A. Gupta, O. D. Sahin, D. Agrawal, and A. E. Abbadi. Meghdoot: content-based publish/subscribe over p2p networks. In *Proceedings of ACM Middleware*, 2004.

[24] R. Huebsch, J. M. Hellerstein, N. L. Boon, T. Loo, S. Shenker, and I. Stoica. Querying the internet with pier. In *Proc. VLDB*, 2002.

[25] H. V. Jagadish, B. C. Ooi, and Q. H. Vu. Baton: A balanced tree structure for peer-to-peer networks. In *VLDB '05: Proceedings of the 31st international conference on Very large data bases*, pages 661–672, 2005.

[26] KaZaA. http://www.kazaa.com/.

[27] S. Y. Ko, P. Yalagandula, I. Gupta, V. Talwar, D. Milojicic, and S. Iyer. Moara: flexible and scalable group-based querying system. In *Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware*, pages 408–428, New York, NY, USA, 2008.

[28] G. Li, S. Hou, and H. Jacobsen. A unified approach to routing, covering and merging in publish/subscribe systems based on modified binary decision diagrams. In *Proc. IEEE ICDCS*, pages 447–457, Columbus, OH, USA, 2005. IEEE Computer Society.

[29] G. Luo, C. Tang, and P. S. Yu. Resource-adaptive real-time new event detection. In *Proceedings of ACM SIGMOD*, pages 497–508, New York, NY, USA, 2007.

[30] S. Madden and M. J. Franklin. Fjording the stream: An architecture for queries over streaming sensor data. In *Proc. ICDE*, Washington, DC, USA, 2002. IEEE Computer Society.

[31] R. Motwani, J. Widom, A. Arasu, B. Babcock, S. Babu, M. Datar, G. Manku, C. Olston, J. Rosenstein, and R. Varma. Query processing, resource management, and approximation in a data stream management system. In *Proc. CIDR*, 2003.

[32] Napster. http://www.napster.com/.

[33] C. Olston, J. Jiang, and J. Widom. Adaptive filters for continuous queries over distributed data streams. In *Proc. ACM SIGMOD*, 2003.

[34] C. Olston and J. Widom. Best-effort cache synchronization with source cooperation. In *Proc. SIGMOD*, New York, NY, USA, 2002. ACM Press.

[35] A. Ouksel, O. Jurca, I. Podnar, and K. Aberer. Efficient probabilistic subsumption checking for content-based publish/subscribe systems. In *ACM Middleware*, 2006.

[36] S. Petrovic, M. Osborne, and V. Lavrenko. Streaming first story detection with application to twitter. In *The 2010 Annual Conference of the North American Chapter of the ACL*, pages 181–189, Los Angeles, California, 2010.

[37] P. R. Pietzuch and J. Bacon. Peer-to-peer overlay broker networks in an event-based middleware. In *Proceedings of DEBS*, 2003.

[38] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable content-addressable network. In *Proceedings of the 2001 conference on applications, technologies, architectures, and protocols for computer communications*, pages 161–172. ACM Press, 2001.

[39] Sylvia Ratnasamy, Scott Shenker, and Ion Stoica. Routing algorithms for DHTs: Some open questions. In *Proceedings of the first International Workshop on Peer-to-Peer Systems*, 2002.

[40] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Proc. ACM Middleware*, 2001.

[41] C. Schmidt and M. Parashar. Enabling flexible queries with guarantees in p2p systems. *IEEE Internet Computing*, 8(3):19–26, 2004.

[42] S. Shah, K. Ramamritham, and P. Shenoy. Resilient and coherence preserving dissemination of dynamic data using cooperating peers. *IEEE TKDE*, 16(7), 2004.

[43] N. Soparkar and A. Silberschatz. Data-value partitioning and virtual messages. In *Proc. PODS*, Austin, TX, USA, 1990. University of Texas at Austin.

[44] E. Souto, G. Guimarães, G. Vasconcelos, M. Vieira, N. Rosa, C. Ferraz, and J. Kelner. Mires: A publish/subscribe middleware for sensor networks. *Personal and Ubiquitous Computing*, 10(1):37–44, 2005.

[45] E. Tanin, A. Harwood, and H. Samet. Using a distributed quadtree index in peer-to-peer networks. *The VLDB Journal*, 16(2):165–178, 2007.

[46] P. Triantafillou and A. Economides. Subscription summarization: A new paradigm for efficient publish/subscribe systems. In *Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'04)*, pages 562–571, 2004.

[47] V. Vazirani. *Approximation Algorithms*. Springer, 2003.

[48] H. Wu, B. Salzberg, and D. Zhang. Online event-driven subsequence matching over financial data streams. In *Proc. ACM SIGMOD*, 2004.

[49] T. Yamashita. Dynamic Replica Control Based on Fairly Assigned Variation of Data with Weak Consistency for Loosely Coupled Distributed Systems. 2002.

[50] J. Yin, L. Alvisi, M. Dahlin, and A. Iyengar. Engineering server-driven consistency for large scale dynamic web services. In *World Wide Web*, 2001.

[51] H. Yu and A. Vahdat. Efficient numerical error bounding for replicated network services. In *The VLDB Journal*, 2000.

[52] B. Zhao, L. Huang, J. Stribling, S. Rhea, A. Joseph, and J. Kubiatowicz. Tapestry: A resilient global-scale overlay for service deployment. *IEEE Journal on Selected Areas in Communications*, 2003.

[53] 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. of NOSSDAV*, June 2001.