An Incremental Approach to Maintaining Up-to-Date Global Statistics in Peer-to-Peer Networks

Nabeel Ahmed
University of Waterloo
200 University Ave.
Waterloo, ON N2L 3G1
n3ahmed@uwaterloo.ca

David Hadaller
University of Waterloo
200 University Ave.
Waterloo, ON N2L 3G1
dthadaller@uwaterloo.ca

ABSTRACT

A significant paradigm shift is taking place as we move from traditional centralized servers to highly decentralized large-scale systems, such as Peer-to-Peer networks. The highly volatile nature of such networks precludes the use of traditional distributed systems algorithms. Global summaries provide useful, if not indispensable, information about the network, and ensuring that these summaries are up-to-date is a significant challenge. In this paper, we study the problem of maintaining such global summaries in the network. Our first contribution to this new area is a fundamental classification of the dimensions of the problem. Within this classification, we focus on epidemic routing protocols and show that traditional techniques are computationally inefficient and provide only moderate accuracy for maintaining up-to-date global statistics. We propose and analyze incremental algorithmic extensions to flooding and randomized gossip, as well as our new random-walk-based scheme. We show that these incremental approaches are able to reduce convergence times by as much as 50% relative to traditional schemes.

1. INTRODUCTION

The Internet continues to grow at a phenomenal rate. With this growth comes the desire to collaborate and share information. Exchanging data using Peer-to-Peer (P2P) systems such as Gnutella [1] and Kazaa [2] has become increasingly popular; it is currently used by millions of users sharing petabytes of data.

The large pool of resources in P2P systems provides incredible potential. However, coordinating machines at such massive scales in a decentralized fashion is a non-trivial task. For example, retrieval of data continues to be one of the fundamental challenges in P2P data sharing systems [13]. In particular, typical database queries involving the computation of a summary (or aggregate) over the entire data set are extremely challenging in a decentralized P2P system.

However, computing such global statistics in a data sharing system is an important, if not necessary, part of the system. For example, query optimization in decentralized database systems relies on global statistics, such as table or index statistics, to estimate query selectivity and query cost. Additionally, pre-computed global summaries of data, such as materialized views, provide significant cost savings for queries involving aggregates of large amounts of data. Computing such aggregates in a decentralized system is a desirable, yet challenging, task.

To the best of our knowledge, proposed solutions for computing global statistics have not addressed efficient ways of maintaining such up-to-date statistics. In a P2P system, nodes may join or leave the system, or their data might change. This creates a situation where the global statistic becomes stale (out-of-date) at some nodes. Keeping global statistics up-to-date is the focus of our work and it is just as important as computing them, since out-of-date statistics could lead to erroneous results.

Having local access to up-to-date global information is critical for certain applications. Consider the problem of information retrieval in a P2P system [28]. Ranking the results of a full-text keyword search on a large set of documents involves computing a relevance score for each matching document. This score is often computed using a term weighting scheme which assigns more weight to matching terms that are not common across all documents. The Okapi term weighting scheme [26] was one of the top performers in the TREC-8 evaluation [3], and is the term weighting scheme used by eSearch [28], a full-text P2P keyword search system. This system, as well as Okapi, assumes the existence of global statistics such as the number of documents that contain a specific term and the average document length. Without local access to accurate global statistics, the system would not be able to rank documents correctly. Thus, having a scheme for maintaining up-to-date global statistics is crucial for the proper functioning of the system. Our work directly addresses this need.

Computing global aggregates in decentralized systems has been addressed in recent work [18, 23, 9, 30]. Rather than using the typical query-response paradigm, the computation of aggregates can be done as a background process. This process is analogous to a continuous query in which all nodes cooperate to compute the global aggregate. Once all of the nodes have a local copy of the global aggregate, the computation has converged.

Techniques for maintaining freshness of global statistics is the focus of this work. The intuition for our approach stems from the following three types of systems.

1. **Stable Networks.** Consider a network in which no data changes occur. In this situation, once the computation of the global statistic has converged, it will never change. Current techniques proposed in the literature indefinitely send messages between nodes, thus wasting network resources. In such systems, we propose not devoting any additional resources to computing the statistic once it has converged. Being sensitive to the *amount of change* in the system is a key part of maintaining freshness of global statistics.

2. **Semi-stable Networks.** Consider a network which is moderately stable; data changes and node joins/leaves occur infrequently. In this case, there will exist time periods where the global statistic will have converged. When a change in the network occurs, current techniques maintain up-to-date global statistic in one of two ways. Either the scheme runs continuously and the global statistic is constantly kept up-to-date, or the statistic is dropped and re-computed on a periodic basis. However, as we show, these schemes are computationally inefficient and achieve only moderate accuracy. We propose that, in such a system, when a change occurs, it is more efficient to run an incremental update protocol rather than performing a complete re-computation. Being aware of changes in the system is key for efficiently updating global statistics.

3. **Chaotic Networks.** In the extreme case of a chaotic system where data and node changes are frequent, convergence of the global statistic will be rare. In such systems, continually computing the global statistic will be more efficient than attempting to manage updates using an update protocol. We show that there exists a point after which it will be more efficient to re-compute the global statistic rather than running an update protocol. Being aware of this point is important for efficiently maintaining freshness of global statistics.

Contributions In this paper, we examine the problem of reducing staleness of global statistics. We begin by classifying the problem domain into its constituent components, an important task that has yet to be done in the literature. We then perform experimental analysis in one particular dimension while keeping the others constant. Specifically, we evaluate the use of different routing protocols for maintaining freshness of global statistics. We compare traditional protocols such as flooding, gossip, and random walk. We then introduce new incremental techniques, which outperform traditional techniques in stable and semi-stable networks both in terms of reduced convergence time and protocol cost.

The remainder of this paper is organized as follows. In section 2 we briefly discuss the background of the problem, followed by a complete classification of the different dimensions of the problem domain in 3. Section 4 outlines our approach to addressing the problem of maintaining global statistics and we present our experimental evaluation in section 6. Finally, section 8 ends with some directions for future work.

2. BACKGROUND

P2P systems present new challenges not present in traditional distributed systems. First, P2P systems operate on a *massive scale*, often connecting millions of nodes. Second, nodes are free to join and leave the system at any time and cannot be assumed to be reliable or available. The frequency of nodes joining, leaving, or failing is referred to as the *churn rate* of the system.

Problem Components

1. Components of Epidemic Protocols
   (a) Network Topology
      i. Unstructured (e.g. Random Graph)
      ii. Structured (e.g. Tree)
   (b) Aggregation Mechanism
      i. Order and Duplicate Sensitive
      ii. Order and Duplicate Insensitive
   (c) Routing Protocol
      i. Unstructured (e.g. Epidemic)
      ii. Structured
2. Query Type
   (a) Extremal (e.g. Min, Max)
   (b) Non-Extremal (e.g. Sum, Average)
   (c) Hybrid Queries (e.g. Top-K)
3. Change Model
   (a) Node Join/Leave
   (b) Node or Link Failure
   (c) Data Change

Figure 1: Classification of the problem domain

Epidemic protocols have emerged as a method for computing global statistics in such systems, as traditional centralized and distributed systems approaches are not suitable. In an epidemic protocol, at every time step, each node selects one or a few nodes to exchange data with. The dynamics of how information spreads through the network resembles the spread of an epidemic [5], which provides increased fault-tolerance [10] against high network churn.

Although our work focuses on P2P systems, our techniques can also be applied to other domains, such as sensor networks [20, 30] and distributed databases [24].

3. PROBLEM DOMAIN

Techniques for dealing with staleness can be broadly classified into two areas: *proactive* and *reactive*. A proactive approach is one in which staleness is prevented through some means, such as an advanced warning or prediction. For example, a node could notify the network of its intended future departure and the global statistics could be adjusted accordingly. In contrast, reactive techniques involve updating the global statistic only after a change occurs. In this paper we consider only reactive techniques.

The problem of maintaining up-to-date global statistics has many dimensions. Each of these dimensions can affect the performance of our proposed incremental algorithm. Before presenting the specifics of our solution, we discuss these dimensions in greater detail. These dimensions are classified into (1) the components of the epidemic protocols, (2) the types of aggregate queries we can compute, and (3) the change model of the system, as summarized in Figure 1.

3.1 Components of Epidemic Protocols

3.1.1 Network Topology

Network topology can have a significant impact on the performance of epidemic protocols or incremental schemes. Here, we briefly outline some topologies and discuss their characteristics in regards to the collection of global statistics. These topologies do not typically exist as part of the underlying network but however can easily be realized in the form of *network overlays*. We further classify each of the different topologies into structured and unstructured topologies.

Unstructured Topologies

- **Clique:** The simplest topology that can be realized in a P2P network is a fully-connected clique (or mesh). The fully connected nature of a clique allows nodes to exchange local state very quickly since the diameter of the network is one. Although not very realistic, such topologies provide a good platform to evaluate properties of epidemic protocols since they preclude protocol dependencies on the underlying topology. For this reason, we use a clique topology in our experimental evaluation.

- **Power-Law Random Graphs (PLRG):** It is a well-known fact that the topology of the Internet follows a power-law heavy-tailed distribution. As well, recently Jain et al. [15] have shown that some naturally evolving P2P systems also exhibit such properties (e.g. early versions of Gnutella). Such networks are typically characterized by a very small percentage of very high-degree nodes and a much larger percentage of low degree nodes. Such networks are more representative of realistic topologies of peer-to-peer networks.

Structured Topologies

- **Tree:** Using tree topologies has become popular for computing global statistics in previous work [20, 30, 9]. In such topologies, the querying node (i.e. node requesting the statistic) acts as the root (or *sink*) of the tree and collects statistics from other nodes by having them propagate the information along the edges of the tree. The query proceeds in two phases, the *distribution phase*, and the *collection phase*. In the first phase, the node floods a query out to all nodes in order to organize them into a tree structure. In the second phase, the values of the leaf nodes are sent to their parents, which are then aggregated and sent on to their parents. This process repeats until the root receives all of the values. This approach is highly scalable for large numbers of nodes, however, it does not provide robustness in the face of failures. Failure of nodes higher up in the tree can render an entire branch of statistics inaccessible during the query process. Considine et al. [9] have discussed a variety of ways to provide added robustness to trees using techniques such as multi-path routing. Others [20, 30] have used alternate techniques (e.g. soft-state consistency) to provide maintenance of the tree topology. However, such schemes introduce issues of double-counting as we discuss section 3.1.2.

- **Hypercube:** The hypercube topology is based on a d-dimensional torus structure. In such a structure, the nodes are organized in a d-dimensional cartesian coordinate space. Due to the complexity of their structures, hypercubes are typically only constructed once and maintained throughout the life-time of the network. Balke et al. [6] discuss the use of hypercubes for continuous processing of *ice-berg* (i.e. Top-K) queries, which are discussed in section 3.2. Hypercubes have gained popularity due to the concept of *super-peers*. Super-peers are *privileged* peers that are delegated greater responsibility and form a hypercube structure among themselves. Super-peers are expected to have much longer uptimes than *regular* peers and thus are considered more reliable. The robustness of this topology is directly related to the dimensionality of the structure. A larger number of dimensions leads to greater robustness but requires greater maintenance overhead and vice versa.

- **Rings:** A rings topology is based on the concept of node broadcasting. Rings of different diameters are placed concentrically relative to each other and the centre of the rings forms the position of the querying node. All other nodes are placed on the edges of the rings. The collection of statistics in the rings topology is analogous to the tree topology, where the query processing occurs in two phases. However, the main difference between this topology and a tree is that the rings topology is much more robust. This added robustness is due to the fact that each node in the outermost ring can choose to route to any one of the nodes (in the next-level ring) within its *range*[1]. Assuming the distribution of nodes across each of the rings is uniform, there are many alternate paths that may be chosen to route around failures. Nath et al. [23] discuss the use of such a topology for collection of statistics over a sensor network.

- **Butterfly:** A traditional butterfly network is typically organized as a multi-level hierarchy where a ring is maintained at each level. Butterfly networks are very attractive topologies for P2P networks since they minimize the amount of information that needs to be maintained for the topology while at the same time provide a moderate degree of robustness. Malkhi et al. [21] discuss the use of such topologies for DHT-based lookup services. Since the maintenance overhead of such networks is very low, they respond very well to high-degrees of volatility. For further details on the specifics of butterfly networks, refer to the survey by Sung et al. [27].

3.1.2 *Aggregation Mechanism*

Aggregation refers to a mechanism whereby components of a distributed system collect global information about different statistics of the system, e.g. network size, average load, etc. Epidemic protocols naturally support such aggregation techniques and a number of challenges need to be addressed when computing such aggregates. Depending on the query that is issued, the topology used, and the routing protocol employed, a number of issues may arise in the computation of such aggregates. For AVG and COUNT

[1]Rings topologies are mostly applicable for wireless sensor networks that have a notion of range. Since we do not specifically consider such networks as part of our study, we do not consider exploring this topology in greater depth.

queries, discussed in section 3.2, *double-counting* problems may occur where nodes may contribute to aggregates more then once, causing distortion in the overall result. The order in which the aggregation takes place is also important since some queries may be sensitive to the ordering process as well. Here, we classify each of the techniques into *Order and Duplicate Sensitive* (ODS) and *Order and Duplicate Insensitive* (ODI) techniques.

- **Order and Duplicate Sensitive:** Most literature on structured topologies for computing aggregates does not apply any special mechanisms to avoid double counting. Techniques to resolve these problems are not required, since the routing protocol and the topology are specifically constructed to avoid such problems. Therefore, such techniques typically *avoid* the double counting problem. However, in unstructured topologies, since there is no such explicit structure that can be exploited, mechanisms are needed to prevent double counting.

  Typically, the problem of double counting is solved by maintaining additional state at each participating node such that whenever an aggregate arrives at a particular node, it is able to identify whether it has contributed to the aggregate or not. We may also need to know how much contribution the node has made to the aggregate. The latter typically requires maintaining more state than the former. However, in general, the greater the information, the higher the accuracy. In our proposed schemes, we follow the latter approach and therefore are able to achieve *exact* results. We discuss the details of our aggregation technique in section 4.

- **Order and Duplicate Insensitive:** Order and Duplicate Insensitive techniques are a more recent class of aggregation mechanisms. Although a number of such techniques have been proposed in the literature, we only briefly discuss prominent work in the area. Since ODI approaches avoid double counting problems, they inherently provide a greater degree of flexibility than ODS. For example, they provide a greater degree of flexibility in the selection of routes.

  – *ODI Synopses/Sketches:* Nath et al. [23] first introduced the order and duplicate insensitive techniques for computing aggregates. They use the approximate FM counting algorithm, originally pioneered by Flajolet and Martin [11]. The key idea in their approach is to minimize the message overhead of the aggregation technique by maintaining small message sizes and a small amount of state at each of the nodes. Each node maintains a summary of its local state (or *synopsis*) and propagates that to other nodes in the network. A fusing of multiple synopses results in an aggregate synopsis being generated using smaller individual synopses. Each synopsis is maintained as a bit vector that represents the contributions that have been made to it. This allows the technique to maintain the synopsis of an entire network of n nodes in a bit vector of size approximately $\log n$. FM can also be regarded as a lossy compression algorithm that provides approximate results. Considine et al. [9] also adopt a similar approach (called *sketches*). However, the FM algorithm approach produces only approximate answers that could potentially be off by as much as 50%, which may be acceptable in some scenarios.

  – *Push Synopses:* Another ODI technique is the push synopsis protocol proposed by Kempe et al. [18]. This technique represents each node's local state as a portion of the complete *mass* of the network. Following the property of *mass conservation*, each node divides its local synopsis in to multiple shares and *pushes* (in a unidirectional manner) the shares to each of its neighbours. In this way, no duplicates are created in the network and double counting is avoided. Kempe et al. [18] have shown that under certain conditions, this aggregation technique in combination with uniform gossip is able to achieve exponentially fast convergence times. Jelasity et al. [17] have extended this basic epidemic protocol to a *push-pull* approach where bidirectional information exchange occurs during each iteration of the protocol. This technique avoids problems in the unidirectional approach where more central nodes in the network may potentially attract more "weight" towards themselves than others.

3.1.3 Routing Protocol

Separate from the aggregation technique is the mechanism used to disseminate information throughout the network. Here we discuss different routing protocols used in current work.

Unstructured

- **Flooding**: A naive approach to propagating data in the network is flooding. Flooding involves sending data in an epidemic fashion, whereby the source node sends to all of its neighbours, who then send to all of their neighbours. This process repeats until either all nodes in the network have received the data or a certain distance from the initiating point (flood depth) is reached.

- **Gossip**: Gossip protocols are a scalable form of flooding used to exchange information between all nodes in the network rather than from a single source. In gossip, time is divided into a series of rounds. During each round, each node selects one or more neighbours to exchange information with. Information propagates through the network in an epidemic or rumor-spreading fashion [10]. A commonly used form of gossip is uniform/random gossip, where each node chooses one neighbour at random to exchange information with during each time step.

  Gossip protocols are simple, low-overhead approaches that are highly robust in the face of failure and do not require error recovery mechanisms [10]. Thus, they are well suited to computing global statistics.

- **Random Walk**: Random walks are becoming a popular approach to performing querying in P2P networks. They are popular because have the desirable property that they reach nodes uniformly at random rather than biased to high degree nodes. Gkantsidis et al. [12] make the observation that in connectivity graphs which are highly clustered or when multiple random walks are used, the walks reaches a more random sample of the nodes in the network than flooding.

We propose a variant of random walk in the context of computing global statistics. Random walk protocols can be thought of as a form a gossip protocol. To initiate the protocol, a random walk is started at a random node in the network. One random walk can be perceived as a package of information moving through the network. At each time step, the node currently holding the package performs three actions: (1) it updates its local statistic based on information contained in the package, (2) it adds its local contribution to the package, and (3) it selects a random neighbour to forward the package to. At the next time step, the new node with the package does the same thing, and the process repeats. Eventually, all nodes will both contribute to the package and the package will have reached all nodes in the network.

Using only one random walk to disseminate information may take many iterations to achieve convergence. In order to speed convergence, multiple random walks can concurrently exist in the system. In this case, the random walks can take advantage of cooperation by exchanging information between packages. That is, when a package A arrives at a node, the node processes the package and forwards it, and when a package B arrives at the same node, sometime in the future, the node can add information contained in package A to package B. This technique yields a significant improvement in convergence time compared to the case where all random walks operate independently, as shown in Section 6.2.

Structured

Many structured approaches to computing global aggregates have been proposed in the literature [20, 30, 9, 6, 23, 21]. Structured routing protocols typically involve performing a flood in a network with an assumed structure. Section 3.1.1 discusses this process for the tree or the ring topology in more detail.

Structured topologies have many advantages for computing aggregates such as fast convergence times and no double-counting. However, structured approaches have the major drawback that they are not robust to failure, which makes them difficult to deploy in a P2P network.

3.2 Query Types

In this section, we present a classification of the different kinds of aggregation queries that may be computed over a peer-to-peer network. An aggregation query is defined as a function computes a summary of some global information about the network. In order to better understand the concept of an aggregation query, the query can be thought of as being computed over a multi-set that is spread across the entire network, where each node maintains an individual set that forms part of the *global multi-set.* We discuss aggregation queries using this particular analogy.

Extremal Queries: Extremal queries are queries whose results lie at either extremes of the multi-set on which the query is posed.

MIN/MAX Queries. MIN and MAX are examples of extremal queries. As their names imply, MIN returns the smallest item over the entire multi-set whereas MAX returns the largest item in the set. The computation of these queries is semantically different from other queries. In particular, double counting of values does not affect the correctness of their results. Although the nodes may contribute multiple times to the synopsis, the largest value in the network will always be correctly returned.

The effect of churn on MIN/MAX is also different from other query types. Given that only a very small subset of nodes actually maintain the maximum/minimum that is desired for the query, any packet loss or topology change may only minimally introduce errors in the computation. However, the latency for such queries may be higher than for example computing the average as is discussed later. Therefore, these queries are fairly robust to dynamic changes in the topology. However, it is important to note that network reachability is a requirement for such queries since network partitions can lead to incorrect results.

Non-Extremal Queries: Many useful queries are those that do not compute extremal values. In general, there are only a few queries that are used as basic building blocks for more sophisticated queries. We briefly discuss some of these queries further.

SUM/COUNT Queries. The two most basic queries are SUM and COUNT. As their names imply, SUM is used to compute the sum of all the values over the entire multi-set in the network. COUNT is a special case of SUM where the set at each of the nodes essentially contains 1 (i.e. we count the number of nodes in the network). These queries are duplicate sensitive since double counting can cause an incorrect final result. Different aggregation techniques uses different methods to compute these aggregates. Some compute approximate values (e.g. Sketches) while others compute exact values (our schemes). Network churn also has a large impact on the results of such queries. Since the loss of any one value can completely alter the final result being computed, these queries are highly sensitive to churn rate. In addition, the latency of computing such queries is generally higher for more accurate results. Due to these challenges, we explore the behaviour of such queries in greater depth. In our approach, we use an ODS mechanism that is able to compute the exact values for COUNT/SUM queries.

AVG Queries. Another popular query is AVG. AVG computes the average across all the values of the global multi-set. It is worth noting that average can be computed in many different ways. In particular, Keshav [19] discusses a technique where a histogram can be constructed and used for this purpose. Each portion of the histogram represents a small range of values that are maintained in a bin of values within that range. Using this technique, we can provide more accurate computations of average then simply summing all the values and dividing by the number of nodes. Using the histogram approach, other statistics such as quantiles [18], variance, and other moments [17] may be also be computed. The AVG query is fairly robust given that most of the values are uniformly close to the real average across the entire network. If that is the case, very accurate results may still be returned even during very high churn rates.

Other non-extremal queries such as PRODUCT can also be implemented by extending any of the queries discussed here.

Hybrid Queries Another important class of queries has emerged that combines aspects extremal and non-extremal techniques for query computation.

Top-K Queries. Top-K or *Iceberg* queries typically return the Top-K values from the multi-set that constitute the top values given a particular property or attribute under consideration (e.g. the top K most replicated music titles). These queries can be implemented using previously discussed query types.

Top-K queries can be computed using the histogram approach previously discussed. In particular, the Top-K entries can be the computed by considering the bins that contain the K highest entries and returning the values contained in those bins. The query could also be adapted to find all values that occur above a certain threshold T by considering only those bins that contain values higher than T.

3.3 Change Model

A variety of events can lead to the staleness of global statistics. These events can be classified into one of the following.

- *Node Join*: When a new node joins the network, its presence or data may change the global statistic in the network.
- *Node Leave*: When a node leaves the network, its departure will remove data from the network, which may affect the global statistic.
- *Node Failure*: A node failing has the same impact on the network as a node leaving, however a failure must be detected, whereas node leaves could be announced.
- *Link Failure*: Links failing in the network could disrupt the computation of the global statistic, or in the worst case, cause a network partition which isolates two sets of nodes from each other. In this case, the global statistic will need to maintained as accurately as possible.
- *Data Change*: A node's data may change, causing the global statistic to become inaccurate.

4. INCREMENTAL APPROACH

In this section, we outline our approach to maintaining up-to-date global statistics. We introduce incremental versions of flood, gossip and random walk and propose an update algorithm to decide when to send updates.

4.1 Incremental Routing Protocols

In this section, we propose three new versions of existing routing protocols which are well suited to performing updates of global statistics. We first discuss the general differences between traditional approaches and our proposed incremental approaches.

- **Maintaining Global Statistics using Existing Techniques**:

  Techniques in the literature for computing global statistics either do not address the issue of staleness or deal with it in one of two ways. The first approach relies on eventual convergence; if a change occurs in the system, the protocols continues to execute and eventually the update is propagated throughout the system.

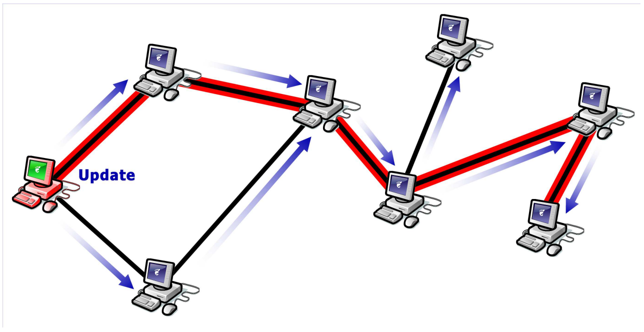

Figure 2: An incremental update message is propagated throughout the network in an epidemic fashion.

  This approach of indefinitely executing the protocol, as pointed out in the introductory section, is completely oblivious to the rate of updates in the system, and could potentially waste resources. The second approach to dealing with staleness of global statistics is to completely drop the statistic at each node and perform a complete re-computation [17]. This approach is necessary in schemes like ODI [23], sketches [9], and push-synopsis [18] due to the inability of a node to determine if it has contributed to a synopsis or not.

  Neither of these two approaches is desirable for updating global statistics. Dropping the entire statistics not only incurs high overhead to perform a re-computation, but also creates a situation where the global statistic at each node is in a constant state of flux. If the global statistic isn't available when it is needed, this approach defeats the purpose of computing the statistic. Similarly, if an update occurs in a relatively stable system, it is more efficient to run an update protocol rather than rely on eventual convergence by performing the global statistic calculation indefinitely, as we show in section 6.2.

- **Maintaining Global Statistics using Incremental Techniques**:

  We propose an incremental approach to updating global statistics. Rather than relying on eventual convergence of the update or dropping the statistic and re-computing, we propose to use a protocol specifically designed to propagate an update throughout the system, when it occurs. Figure 2 illustrates an example of this process. As discussed in the introductory section, our approach is update-aware and performs better for maintaining the global statistic than existing techniques, in stable and semi-stable networks.

  We propose the following incremental techniques.

- **Incremental Flooding**: In order to compute a global aggregate at all the nodes, traditional flooding works by having every node flood over the entire network. We propose an incremental version, where, if an update occurs at a node, only that node floods an update message throughout the network. Much like traditional flooding, this is also a naive approach to solving the problem, but provides perspective and insight into the problem, as discussed in section 6.
- **Incremental Gossip**: Our incremental approach to gossip maintains the simplicity of traditional gossip,

where each node chooses one random neighbour to send data to at each time step. In the incremental version, the statistic is never dropped, and gossip is only performed when an update occurs. That is, once the global statistic has converged, communication is stopped. When an update occurs at any node, gossip is initiated at all of the nodes and is run until the update has propagated throughout the network.

- **Incremental Random Walk**: We also propose an incremental version of our proposed random walk scheme. In this scheme, when an update occurs, instead of starting random walks at random nodes in the network, all of the random walks are initiated from the update point, and contain update-specific contents. This is equivalent to the updated node "sending out" packages to its neighbours containing its update. The neighbours receiving the packages then perform the non-incremental random walk protocol as described above, until the network has converged. When multiple updates occur simultaneously, multiple random walks will be initiated from multiple points in the network.

4.2 Update Algorithm

Separate from using an incremental routing protocol to update global statistics, is the update algorithm used to decide when to send an update. In some situations, when an event in the system causes the global statistic to become stale, it may not be desirable to initiate the update protocol. Consider the case of computing a global average across the network Suppose a node which contains a value very close to the average, leaves the network. In this case, the node's departure will have much less impact on the global statistic than if it were to contain an extreme value.

An update algorithm refers to the decision of whether or not to send an update when an event has occurred which could potentially change the global statistic. In the above example, an update algorithm could use a significance threshold to decide if an update should be sent. For example, if the difference between the node's value and the global average is below a threshold, the change is regarded as insignificant and no update is sent. If the value is above the threshold, then an update is sent. This threshold can be a pre-specified value or can be adapted over time. Pseudo-code for this combined with our incremental update protocol is shown in Figure 3.

```
Incremental Random Walk Update Algorithm

i) At each Updated Node: (At any given time $t$)
    1. Recompute global statistic $G_t$, based on local update.
    2. Compute update difference $U_t = G_t - G_{t-1}$
    3. if $U_t > D$, where $D$ is the significance threshold, initiate update routing protocol, else do nothing.

ii) At each Non-Updated Node: (At any given time $t$)
    1. Upon receipt of a package $A$, check to see if already contributed to package. If so, proceed to step 3.
    2. Update package $A$ with local contribution.
    3. Check to see if local state contains other more up-to-date information than package $A$. If not, proceed to step 5.
    4. Update package with more up-to-date local state.
    5. Use routing protocol to re-send package $A$ to neighbour(s).
```

Figure 3: Pseudo-code for updates using our incremental random walk protocol with our update algorithm

5. ASSUMPTIONS

5.1 Problem Assumptions

We wish to study how different routing protocols can be used to maintain global statistics. We do this by keeping all other aspects of the problem constant while varying only the routing protocol. We have chosen is methodology in order to minimize the potential impact of such dimensions of the problem on our results. These choices can be stated as the assumptions we make in our approach to address the problem of maintaining up-to-date global statistics.

- **Network Topology: Clique**. A clique topology sufficiently captures the behaviour of the routing protocol without introducing a bias in the results due to the characteristics of the topology. In addition to using a clique topology, we also perform a sensitivity analysis using Power-Law Random Graphs.

- **Aggregation Mechanism: Order and duplicate sensitive with exact results**. Although computing exact aggregates requires more state than approximate approaches, computing exact aggregates eliminates the unpredictability associated with using approximate techniques such as ODI or push-synopsis.

- **Query Type: SUM**. We consider only the SUM aggregate as it is a foundation on which many other queries can be built. This query is more challenging to compute, since it is much more sensitive to the dynamics of the network, as discussed in section 3.2.

- **Update Algorithm: Always send updates**. To improve efficiency of updating global statistics, we use the threshold mechanism to decide when to send updates. In this work, we choose to always send updates, which is equivalent to maintaining a significance threshold of zero in our algorithm presented in Figure 3.

- **Change Model: Data Change**. Modeling data change at nodes is sufficient for our analysis of accuracy, cost, and convergence time (discussed in section 6.1). In future work which will also analyze robustness, it will be necessary to model node and link failures as well.

5.2 Environmental Assumptions

We also briefly list some assumptions we make about the operating environment in order to simplify the construction and analysis of our schemes. These assumptions are discussed further:

1. **Peer-to-Peer File-sharing Environment:** We analyze the performance of our incremental scheme in the context of a peer-to-peer file sharing environment. Therefore, we do not consider power-constraints, bandwidth, minimal storage and other limitations inherent in other scenarios. Instead, our study explores the performance of our scheme in isolation and provides groundwork for extension to more sophisticated scenarios.

2. **Non-malicious Peers:** We assume that peers cooperate with each other and do not behave maliciously. Existing research has already addressed methods of dealing with malicious peers in the context of computing global statistics [16].

3. **Uniform Data Distribution:** We assume that data changes in the peer-to-peer network are uniformly distributed across the entire system. Therefore, for simplicity in analysis, we do not consider *hotspot* changes within the network. This provides a worst-case for our analysis as multiple updates can be treated as one using our incremental approaches. However, it has been observed in existing systems that the majority of data changes occur at a small percentage of peers, due to the free-rider problem [4].

4. **Batched Data Changes:** Data changes in our study are modeled *sequentially.* That is, data changes and the running of the incremental update scheme do not happen concurrently. This is a requirement for obtaining bounds on protocol convergence time. Kempe et al. [18] show that the Push-Synopsis protocol converges exponentially fast in the face of failures, only if these failures do not occur concurrently with the execution of the protocol. We also maintain this assumption in order to simplify our analysis.

5. **Globally Synchronized Protocol:** For simulation purposes, all peers are globally synchronized and our protocol proceeds in the form of *rounds.* In each round, peers exchange information with and perform local aggregations before the end of the round. This process continues in a globally synchronized fashion. This assumption is only made for analytic simplicity and does not affect the correctness of our proposed techniques.

6. **Identical Peer Configurations:** We assume that the peers maintain identical configurations and are identical in every way. We also assume that peers possess enough resources to compute and store information as necessary for the computation of global statistics. This is a reasonable since the amount of data and processing power required extremely small relative to the capacities that individual nodes in P2P system typically have on the Internet today.

7. **Uniform Network Characteristics:** Finally, we also assume that the network itself has stable and uniform characteristics throughout. This assumption allows us to concentrate on the performance of the proposed incremental schemes in isolation, as we do not model network-level inconsistencies such as packet loss and congestion.

6. SIMULATIONS

In this section, we study the performance of our proposed incremental algorithms as well as existing naive drop and recompute schemes presented in the literature. Section 6.1 discusses the simulation methodology we use for our comparisons along with the metrics that are used in order to gauge the performance of each of the schemes. Section 6.2 presents our simulation results for different scenarios.

6.1 Simulation Metrics

We have implemented a compact simulator in C that allows us to experiment with many different network topologies, aggregation schemes, and routing protocols. Our focus is on the routing protocol component, within which we compare our incremental techniques with traditional epidemic protocols. We use the following performance metrics for our comparison.

Accuracy. We use the mean error in the local estimate of the global aggregate to quantify the accuracy of each of the schemes. The mean error is defined as $e_t = \frac{1}{N}\frac{1}{m_t}\sum|\hat{m_t} - m_t|$, where $\hat{m_t}$ is the local estimate of the aggregate, m_t is the true value of the aggregate, and N is the total number of nodes in the network. The closer this value is to zero, the closer the local value is to the actual value in the network. The two factors that affect accuracy are the aggregation technique and, for the incremental schemes, the update algorithm.

Protocol Cost. The cost of a protocol can be determined by its message overhead. Message overhead is the number of messages expended by the protocol to converge to the correct value. Cost can also include the size of the message and the amount of state that needs to be maintained at each node in the network. However, since the message size and state are components of the aggregation mechanism, we do not consider them in our cost function, as we are focusing exclusively on routing protocol. The factors that primarily affect cost are the aggregation technique and the routing protocol.

Convergence Time. The convergence time of the protocol is a measure of how fast the protocol is able to converge to the correct value of the aggregate. Convergence time is measured in terms of the number of rounds until convergence, where each round represents a fixed time interval in which a node or set of nodes gossip amongst each other[2]. More abstractly, the convergence time itself is defined as an *epoch* of time which varies for different protocols. We define the relevance of epochs further as we present our results in section 6.2. The factors that affect the time for convergence include the aggregation technique, routing protocol, and topology.

Robustness. The robustness of each of the schemes is characterized by the protocol's resilience to node/link failures and high-churn rates in the network. Since our change model only considers changes to the data at individual peers and does not consider node/link failures or node joins, we defer simulating such changes as part of future work. The robustness of the protocol depends fundamentally on the routing protocol and the underlying routing topology. Although we do not directly address robustness in our study, we present a sensitivity analysis which shows that our tech-

[2]The actual length of the round depends on the underlying transmission medium and is therefore not explicitly assigned in our simulations

niques apply to larger and more sophisticated topologies as well as simple cliques.

In each of our simulations, we report the average measure over 500 epochs to minimize statistical variation in our results. In order to analyze the affect of updates on the system, we begin our simulations with a *stable* network, i.e. the global statistic has converged. Once the network has converged, we induce updates at a specific number of nodes and observe the behaviour of the epidemic protocol. Unless otherwise indicated, our network size includes 512 peers that participate in the protocol; and for incremental random walk schemes, we use a total of 512 random walks.

6.2 Simulation Results

We have chronologically divided our results into separate sections in order to distinguish between the importance of each result. Specifically, we have categorized our results into the following: (1) incremental techniques vs. full re-computation approaches, (2) incremental routing protocols, (3) multiple random walk analysis, and (4) protocol sensitivity analysis. Each result logically follows from the previous result.

In our simulations, each peer maintains a single value that it contributes to the entire global statistic of the network.

6.2.1 *Incremental vs. Re-computation*

Here we show that incremental techniques are better than re-computation techniques for semi-stable networks.

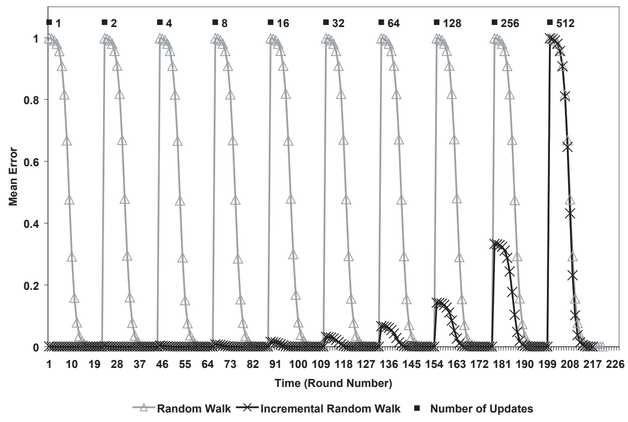

Figure 4: Accuracy of random walk protocols as updates are made in the system; the points at the top of the graph indicate the number of updates injected into the system at that point in time.

Accuracy. We compare our incremental techniques with re-computation schemes and present our results for accuracy in Figures 4 and 5. Here we compare the accuracy over time of each scheme. An epoch in this graph is the time between injecting updates into the system and the time it takes for the error of both schemes to drop to zero. Figure 4 indicates that the incremental random walk protocol has almost negligible error for very small amounts of updates when compared to the re-computation version of random walk which continually jumps to almost 100% when data changes are introduced in the system. Gossip-based and flood techniques (Figure 5) exhibit similar trends. In general, we see that as the number of updates increases in the system, the effectiveness of the incremental approach decreases. However, for a

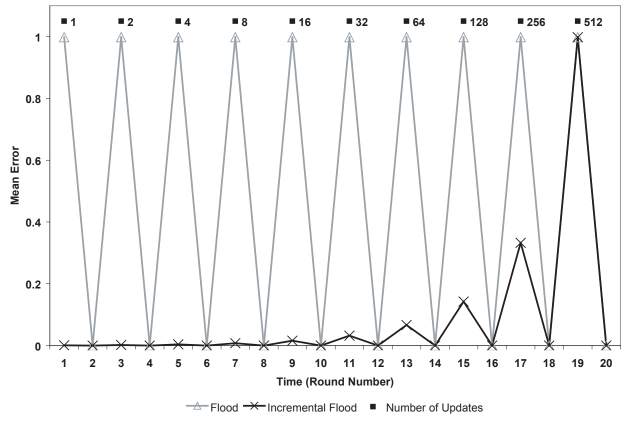

Figure 5: Accuracy of flooding protocols as updates are made in the system.

moderate number of updates (e.g. 256 updates/epoch), incremental techniques remain superior in terms of accuracy.

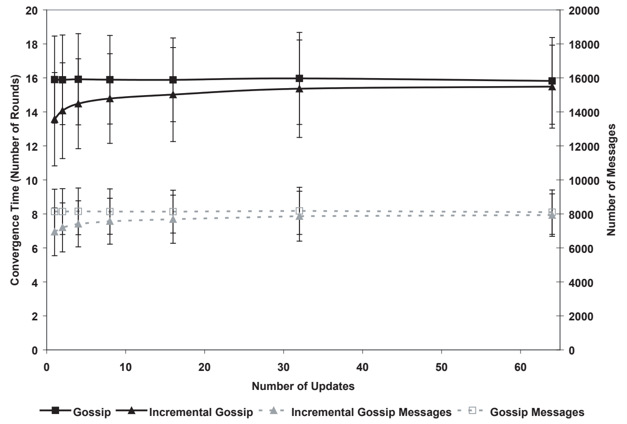

Figure 6: Impact of updates on performance of gossip protocols.

Convergence Time. From the left axis in Figure 6, we see that our incremental gossip scheme reduces the convergence time by as much as 20% over traditional gossip for small numbers of updates. In the re-computation scheme, since peers drop and recompute the summary, the re-computation scheme takes longer to converge. The relative effectiveness of incremental schemes drops off as the number of updates increases. This is due to the observation that as the number of updates is increased (e.g. to point where every node is updated), a full re-computation is unavoidable. Random walk results (Figure 7) follow similar trends; we achieve a 2.9 times improvement in convergence time for a small number of updates, and a decrease of 68% even for large numbers of updates. Flood maintains the fastest convergence times since updates are instantaneously sent to all nodes in the clique. Incremental flood and flood perform identically.

Protocol Cost. The right axis in Figure 6 represents the total number of messages transmitted throughout the running of the protocol until convergence. We observe that incremental gossip has less message overhead than traditional gossip since it requires additional rounds to converge, due to re-transmitting previously dropped statistics. However, for larger number of updates, since almost all the data in

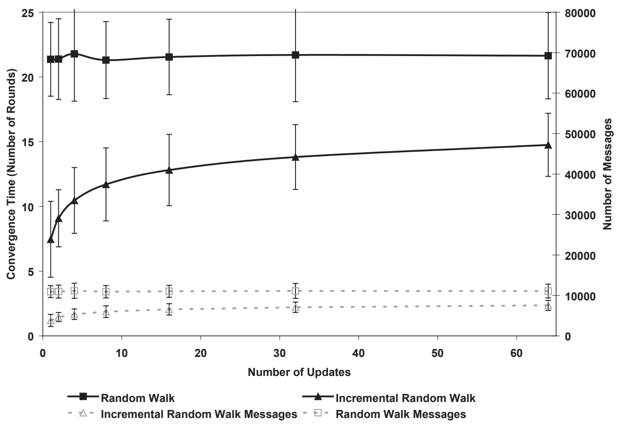

Figure 7: Impact of updates on performance of random walk protocols.

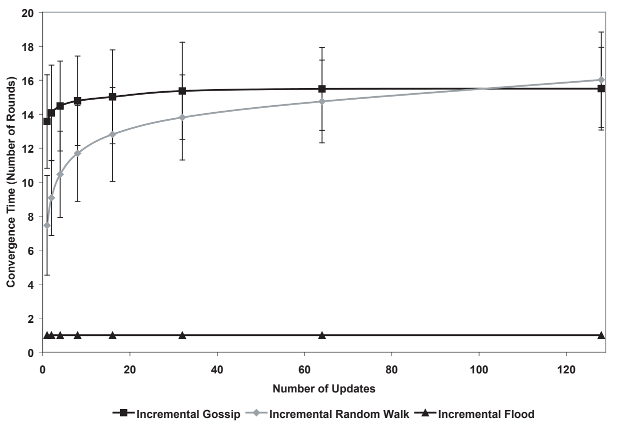

Figure 8: Impact of updates on convergence speed of incremental protocols.

the network is being modified, dropping and re-computing the entire global statistic requires approximately the same number of messages than the incremental schemes.

Random walk results (Figure 7) exhibit much larger difference in the number of messages. The primary reason for this variation is due to the way in which our incremental scheme behaves. Since our scheme explicitly propagates updates from the update point instead of from random points in the network, the number of messages required to carry the update to all the nodes in the network is substantially reduced. Therefore, for random walk, our incremental random walk scheme provides significant cost savings.

For flooding schemes, the number of message is deterministic. In our incremental flood scheme, since updates always propagate from the update point, and since we assume an underlying clique topology, the message overhead for incremental flood is equal to the number of peers in the network N. However, in traditional flood since each peer floods to all of its neighbours, the message overhead is N^2. Therefore, incremental flood is more cost effective than traditional flood.

As our results indicate, all our proposed incremental schemes outperform the corresponding traditional re-computation schemes in all dimensions when the number of updates are moderate. We present similar results on more complex topologies in section 6.2.4.

6.2.2 Comparing Incremental Routing Protocols

We provide a comparison of the different incremental routing protocols, proposed in Section 4.1. Since we are using order and duplicate sensitive techniques to achieve exact results, the accuracy is the same for each of the incremental routing protocols. Therefore, it is unnecessary to compare routing protocols using this metric.

Convergence Time. Figure 8 presents a comparison of the convergence time of incremental gossip, random walk, and flood. Since the underlying topology is a clique, the convergence time for flood is one since each update is flooded to all of the nodes in one time step. Therefore, flood provides the lowest convergence time in comparison to the other approaches, but comes at a significant cost, as show next.

Incremental random walk performs the next best for a moderate number of updates, outperforming incremental gossip by up to 50%. This improvement is due to the effective update-specific propagation mechanism used in incremental random walk. Since update messages are initiated by the updated nodes, the update *spreads* more effectively through the network than in incremental gossip. In incremental gossip, since communication is not update-aware, many message exchanges do not convey useful information. However, for a larger number of updates, since the updates are spread out uniformly across the nodes, the benefit of an update-aware propagation mechanism diminishes and incremental gossip begins to perform better. This is evident from the point at which the two cross on the graph. Therefore, when the number of updates is high, such as in a chaotic system, it is more desirable to use incremental gossip.

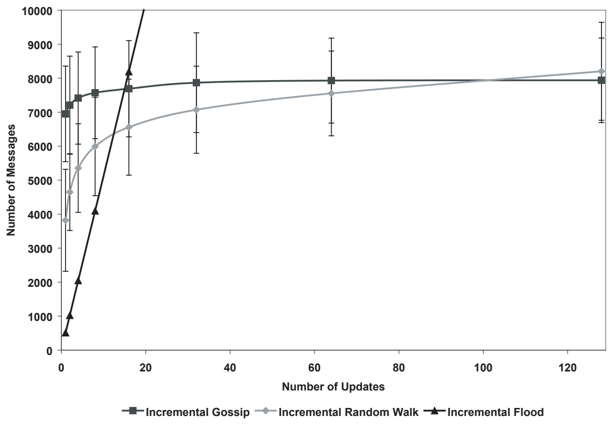

Figure 9: Impact of updates on cost of incremental protocols.

Protocol Cost. Figure 9 illustrates the message overhead of our three incremental techniques. We see that the flooding scheme performs poorly since each individual update is flooded out to all of the nodes. Therefore, the overhead for flooding is approximately NM, where N is the number of peers in the network and M is the number of updates that have been introduced in the system in a given epoch.

Since incremental gossip is not update-aware, the message overhead is approximately independent of the number of updates. The message overhead in incremental random walk increases in the same fashion as the convergence time, as more messages are required to propagate more updates. The point at which the random walk curve crosses the incremental gossip curve is the same point as on the convergence time graph. Therefore, we see that the message overhead of the protocol is directly related to the convergence time.

We observe that for a moderate number of updates, incremental random walk outperforms incremental gossip both in convergence time and message overhead. However, as the number of updates increase (i.e. as the system becomes more chaotic), incremental gossip performs better. This leads us to believe that a hybrid scheme may be more suitable to achieve better overall performance. We discuss possibilities for such an approach as part of future work.

6.2.3 Effect of Multiple Random Walks

In this section, we analyze the affect of multiple random walks on the performance of incremental random walk. We provide these results as an extension of our study on incremental random walks. As discussed in the previous section, we do not consider accuracy for these set of results. Based on previous results, we have chosen to fix the number of updates to 16. We then compare the performance of each scheme with different network sizes. This allows us to gauge how well the scheme is able to scale up to larger networks.

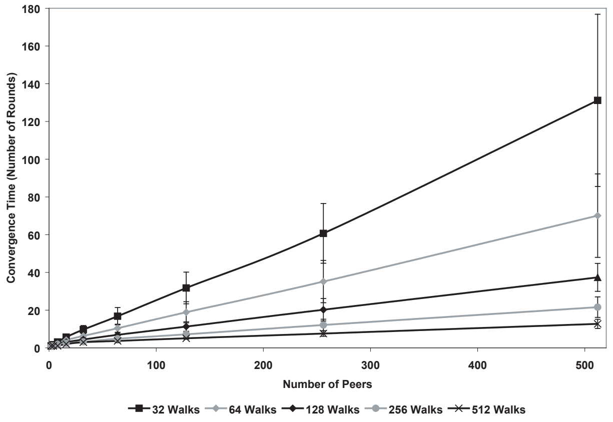

Figure 10: Effect of increasing number of random walks on convergence time

Convergence Time. Here we analyze how fast the scheme is able to converge with different numbers of walks initiated from the update points. Figure 10 illustrates our results. We observe that for large network sizes and a small number of walks (e.g. 32), doubling the number of walks results in approximately a 50% decrease in the convergence time. However, we see diminishing returns as we continue to double the number of random walks for a fixed network size. We hypothesize that the initial increases in number of walks cause a large decrease in convergence time since a larger number of walks can be used to propagate the updates through the network. However, after a point, over-provisioning the number of walks does not provide significant benefit since the network has become saturated by the random walks. Nevertheless, we see a significant decrease in the overall convergence time as the number of random walks increases.

Protocol Cost. In Figure 11, we see that increasing the number of random walks does not significantly increase the message overhead. We observe that for a 16 fold increase in the number of random walks (e.g. using 512 vs. 32), we see only a 55% increase in the message overhead. This is an un-

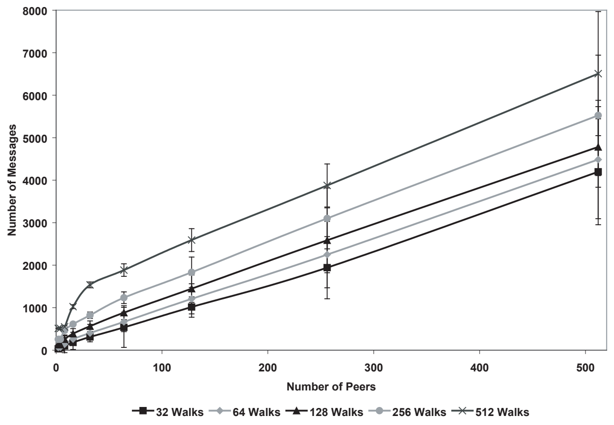

Figure 11: Effect of increasing number of random walks on message overhead

expected result since increasing the number of walks would seem to increase the message overhead. However, since our scheme is *co-operative*, where multiple random walks are able to carry updates for each other, resulting in reduced the convergence time and thus reduced message overhead.

These results are encouraging and indicate that increased performance is achievable by utilizing multiple random walks. Although we don't specifically present results for robustness, we hypothesize that multiple random walks are more robust then a single random walk. Since multiple random walks allow multiple nodes to receive updates concurrently, a failure of a fraction of peers in the system can still allow the update to be propagated successfully, given that the network has not been partitioned. This justifies our use of multiple random walks in the results presented earlier.

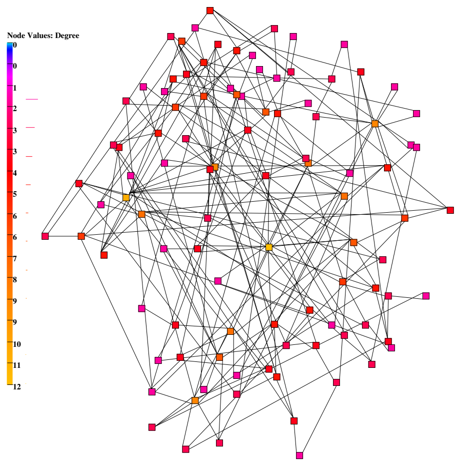

Figure 12: Sample 100 node PLRG topology, generated using BRITE. The line graph on the left represents the frequency of node degrees.

6.2.4 Effect of Topology

We now perform a sensitivity analysis of our schemes by executing our schemes on more complex topologies. For such topologies, we have employed power-law random graphs (PLRGs), discussed in section 3.1.1. We used the BRITE [22] topology generator to construct such topologies. An example of a PLRG generated by BRITE for a network of size 100 is presented in Figure 12. Due to space limitations, we omit results for flooding protocols.

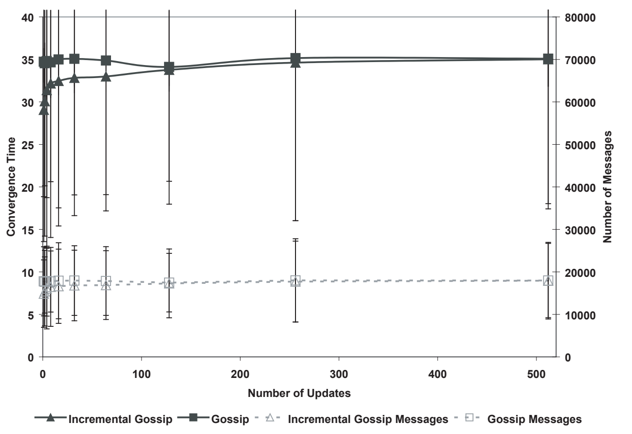

Figure 13: Performance of gossip protocols in a network with complex topology.

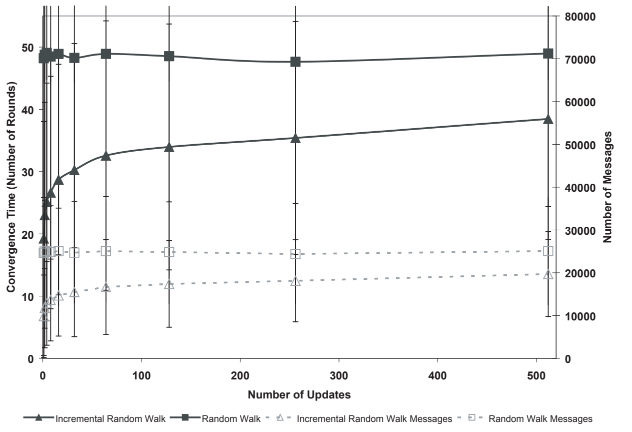

Figure 14: Performance of random walk protocols in a network with complex topology.

In Figures 13 and 14, we see that the relative performance of incremental gossip and incremental random walk is virtually the same as the results presented using a clique. Our incremental schemes outperform the re-computation schemes in all cases. However, the statistical variation in the results is higher for these experiments than they are for the clique topology. Due to the heavy-tailed distribution of node degrees, the average degree a node is low. We hypothesize that if, in the early rounds of the protocol, very high-degree nodes are encountered, the convergence time decreases sharply since the node is able to disseminate information to other nodes much faster. However, since the dissemination is inherently random, the protocol may not encounter high-degree nodes later in the execution of the protocol, and it will take much longer to converge, resulting in long convergence times and high message overhead. Nevertheless, in the average case, we see that the trends from the clique topology carry over to PLRG-like topologies.

7. RELATED WORK

The work presented in this paper focuses on supporting *consistency* of global statistics for P2P networks. To the best of our knowledge, no other study has been done that specifically addresses this problem for global statistics. However, since this work overlaps with a number of different fields, including global state maintenance and gossip-based dissemination and aggregation, we provide an overview of the most relevant work in these areas.

Global state maintenance in traditional centralized or small-scale distributed systems is a well-studied problem. The most well-known technique is the Chandy-Lamport Snapshot protocol [8]. Although this protocol has some very interesting theoretical properties, it is not suitable for large scale distributed environments such as P2P systems since it does not support intrinsic characteristics of such systems such as massive distributions and high-volatility. Therefore, its applicability is limited in such scenarios and other techniques are required.

There has been a recent interest in using gossip-based protocols for computing global statistics in P2P systems [23, 18, 9]. Many techniques have been proposed that support such computation for both traditional peer-to-peer networks operating across the Internet as well as recent P2P wireless ad hoc networks that operate under more constrained conditions. In our work, we focus on the former and assume the use of a P2P file-sharing application[3] (as discussed by Zaharia et al. [29]).

Many gossip-based systems have been proposed that assume a structure/hierarchy (e.g. Tree or DAG) on the nodes in the network in order to allow easy aggregation of statistics [20, 30]. These techniques have been shown to scale well, however, they do not meet the robustness requirement of P2P systems and require excessively high amounts of maintenance overhead during at even moderate volatility rates. Recently, unstructured techniques, such as random gossip, have been extensively studied to combat these problems and our work follows such trends.

Flooding techniques have been investigated in order to meet the robustness requirement. However, studies have shown that flooding does not provide fast convergence times in grid-like networks [18]. Other *intelligent* flooding approaches like *directed diffusion* [14] have also been proposed that are able to *direct* specific kinds of information between different nodes. However, even for these techniques, no theoretical bounds on convergence times have been shown. Recently, probabilistic propagation mechanisms have been studied that aim to improve on the drawbacks of flooding. Kempe et al. [18] have proposed a *linear-synopsis approach* that uses uniform/randomized gossip (*push-synopsis*) for information dissemination to compute global statistics. They provide theoretical results showing that such protocols are able to converge exponentially fast. Boyd et al. [7] have also shown that a Semi-Definite Program mechanism can be used to construct optimal randomized gossip protocols. Jelasity et al. [17] extend the push-synopsis approach and propose a *push-pull* technique that supports a weaker network connectivity model. They also provide empirical results supporting the performance of their technique. However, neither of these works address consistency of the computed statistics and simply advocate re-computing them at fixed intervals (*automatic restarting*). Zaharia et al. [29] also propose a similar approach in their study. We have shown in this work the benefits of an incremental approach and present empir-

[3]We discuss such constraints and limitations as a part of future work

ical results quantifying the advantages of such a technique over a complete re-computation.

Random walks have also been explored for information dissemination in peer-to-peer networks. Gkantsidis et al. [12] present both theoretical as well as empirical results to show that such techniques are highly effective for searching and network overlay construction. Zhong et al. [31] also present bounds on convergence times for random-walks in non-uniform membership peer-to-peer systems. Our work complements this area since we show that random-walks are also very effective for incremental information dissemination of data updates/changes in the network.

Many aggregation mechanisms have also been proposed in the literature that support the construction of synopses (sketches) for both order and duplicate sensitive and insensitive techniques for information aggregation (to prevent issues like *double-counting*). Nath et al. [23] are the first to introduce the Order and Duplicate Insensitive (ODI) synopsis approach that uses a clever FM counting algorithm for representing synopses. The resulting synopsis is in the form of a compact bit vector that represents the aggregate statistic. In parallel, Cosidine et al. [9] have also proposed a very similar sketches technique. Additionally, Kempe et al. [18] have also proposed an ODI push-synopsis approach that disseminates information by dividing each local synopsis into individual shares and sending these shares to neighbouring nodes. On the other end of the spectrum, Order and Duplicate Sensitive (ODS) techniques have mostly been used in the structure-based gossip protocols indicated above. In this preliminary study, we also adopt an ODS approach and leave extensions to ODI as part of future work.

8. FUTURE WORK

In this section, we briefly outline some avenues that we consider as part of future work.

Sophisticated Topologies. In our study, we have considered only unstructured topologies. We would also like to investigate other topologies that can be used for computing tighter bounds on the convergence of the routing protocols (e.g. TAG, Hypercubes, etc). In addition, we would also like to investigate the robustness of the protocols on such topologies. In this regard, we would like to follow the approach proposed by Kempe et al. [18], who adopt a multipath routing technique for added robustness and use FM to avoid problems of double counting.

Other Aggregation Mechanisms. We are also looking into investigating other aggregation mechanisms that can be used to generate aggregate statistics and extend them as part of our incremental approach. As is discussed by Keshav [19], we are especially interested in exploring techniques to enhance the FM counting algorithm proposed by Nath et al. [23] since it has some nice properties in terms of performance and cost of the aggregation technique (i.e. compact message sizes). Since this approach is inherent lossy, it becomes hard to determine the contributions that have been made to each of the bit vectors. We propose a delete vector approach that can be used to support incremental updates to the synopsis. We leave the details for future work.

Enhanced Change Model. This work only considers modifications to the data values which is not fully representative of the dynamics of peer-to-peer systems. We plan to study the behaviour of the incremental schemes under a more realistic change set that incorporates all of the aspects that we initially proposed in our change model. In doing so, we hope not only to gain further insight into the performance of the schemes, but also to enhance them in order to deal with the different dimensions of volatility in the system.

Thresholding Schemes. In this work, we only consider the version of the update algorithm that *activates* itself whenever any update occurs in the system. Other possibilities include utilizing a thresholding mechanism that decides when to initiate updates. The threshold can be fixed ahead of time or can be adapted to changes over time based on the dynamics of the system (i.e. the degree of churn in the system). We do not argue the merits of either approach in this work and are currently exploring these techniques in greater depth.

Hybrid Routing. Our simulation results indicate that a combined routing scheme appears to be more efficient for maintaining up-to-date global statistics. Specifically, for low churn rates, we advocate using an incremental random walk propagation approach. When the churn rate reaches a point at which the incremental gossip scheme outperforms the incremental random walk scheme, we switch to the gossip scheme instead. This allows our work to be easily extended to chaotic networks discussed in the introduction section of the paper.

Proactive Updates. As outlined in section 3, maintaining up-to-date global statistics can be classified into proactive and reactive schemes. In this work, we have only concentrated on reactive schemes that detect staleness and react to it. Proactive schemes are more difficult to construct since they require *a priori* information about the future state of the network and peers. If such information is inaccurate, the resulting scheme may potentially degrade the accuracy and performance of global statistics computations. Therefore, these class of schemes must provide a high degree of accuracy in *predicting* the future behaviour of the network. Although this may seem to be an insurmountable task, our preliminary survey of the area has provided some interesting avenues for further exploration [25], which we plan to investigate.

9. CONCLUSION

This work has addressed the problem of maintaining up-to-date global statistics. We first provide a fundamental classification of the problem domain. We discuss how existing techniques fit into this classification. Using our classification, we introduce incremental approaches to maintaining up-to-date global statistics. We also present an evaluation showing that existing techniques, which use a naive approach, perform poorly in comparison to our incremental schemes. Using our approach, we are able to reduce convergence times by as much as 50%.

We also show that there exists a point after which it will be more efficient to use incremental gossip to compute the global statistic rather than using incremental random walk. Being aware of this point is important for efficiently maintaining freshness of such statistics, and we propose to investigate a hybrid routing scheme in the future.

To the best of our knowledge, this is the first serious treatment of incremental approaches to maintaining freshness of global statistics in P2P systems. Our ongoing efforts include evaluating the other aspects of the problem as well as our proposed hybrid solution.

Acknowledgements

The ideas in this work owe much to ongoing discussions with Dr. Srinivasan Keshav.

10. REFERENCES

[1] Gnutella. http://www.gnutelliums.com/.

[2] Kazaa. http://www.kazaa.com.

[3] Text Retrieval Conference (TREC). http://trec.nist.gov.

[4] E. Adar and B. Huberman. Free riding on gnutella, 2000.

[5] N. Bailey. *The Mathematical Theory of Infecious Diseases and its Applications*. Hafner Press, 1975.

[6] W. Balke, W. Neidi, W. Siberski, and U. Thaden. Progressive distributed top-k retrieval in peer-to-peer networks. In *Proceeding of 21st International Conference on Data Engineering (ICDE)*, April 2005.

[7] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Gossip algorithms: Design, analysis and applications. In *Proceedings of INFOCOMM 2005*, March 2005.

[8] K. M. Chandy and L. Lamport. Distributed snapshots: determining global states of distributed systems. *ACM Trans. Comput. Syst.*, 3(1):63–75, 1985.

[9] J. Considine, F. Li, G. Kollios, and J. Byers. Approximate aggregation techniques for sensor databases.

[10] A. Demers, D. Greene, C. Houser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database maintenance. *SIGOPS Oper. Syst. Rev.*, 22(1):8–32, 1988.

[11] P. Flajolet and G. N. Martin. Probabilistic counting algorithms for data base applications. *J. Comput. Syst. Sci.*, 31(2):182–209, 1985.

[12] C. Gkantsidis, M. Mihail, and A. Saberi. Random walks in peer-to-peer networks. In *Proceedings of INFOCOMM 2004*, March 2004.

[13] S. Gribble, A. Halevy, Z. Ives, M. Rodrig, and D. Suiu. What can databases do for peer-to-peer? In *WebDB 2001*.

[14] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proceedings of ACM/IEEE International Conference on Mobile Computing and Networking*, Aug. 2000.

[15] S. Jain, R. Mahajan, and B. Niswonger. Self-organizing overlays. Technical report, Washington University, 2000.

[16] M. Jelasity, A. Montresor, and O. Babaoglu. Towards secure epidemics: Detection and removal of malicious peers in epidemic-style protocols. Technical Report UBLCS-2003-14, University of Bologna, Nov. 2003.

[17] M. Jelasity, A. Montresor, and O. Babaoglu. Gossip-based aggregation in large dynamic networks. *ACM Journal*, 15(5), November 2004.

[18] D. Kempe, A. Dobra, and J. Gehrke. Gossip-based computation of aggregate information.

[19] S. Keshav. Personal communications.

[20] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. Tag: A tiny aggregation service for ad hoc sensor networks. In *Proceedings of USENIX OSDI*, December 2002.

[21] D. Malkhi, M. Noar, and D. Ratajczak. Viceroy: A scalable emulation of the butterfly. In *Proceedings of the PODC*, July 2002.

[22] A. Medina, A. Lakhina, I. Matta, and J. Byers. BRITE: an approach to universal toplogy generation. In *Proceedings of MASCOTS*, Aug. 2001.

[23] S. Nath, P. Gibbons, S. Seshan, and Z. Anderson. Synopsis diffusion for robust aggregation in sensor networks. In *Proceedings of SenSys '04*, November 2004.

[24] M. T. Ozsu and P. Valduriez. *Principles of distributed database systems*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1991.

[25] S. Saroui, K. Gummadi, and S. Gribble. A measurement study of peer-to-peer file sharing systems. In *Proceedings of the SPIE/ACM Conference on Multimedia Computing and Networking (MMCN) 2002*, January 2002.

[26] S.E.Robertson and S.Walker. Okapi/Keenbow at TREC-8. In *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*, pages 151–162, 1999.

[27] A. Sung, N. Ahmed, D. Hadaller, R. Blanco, H. Li, and M. Soliman. A survey of data management in peer-to-peer systems, April 2005.

[28] C. Tang and S. Dwarkadas. Hybrid global-local indexing for efficient peer-to-peer information retrieval. In *Proceedings of the Symposium on Networked Systems Design and Implementation (NSDI)*, June 2004.

[29] M. Zaharia and S. Keshav. Adaptive peer-to-peer search. In *Submitted for Publication*, January 2005.

[30] J. Zhao, R. Govindan, and D. Estrin. Computing aggregates for monitoring wireless sensor networks.

[31] M. Zhong, K. Shen, and J. Seiferas. Non-uniform random membership management in peer-to-peer networks. In *Proceedings of IEEE INFOCOM*, March 2005.