NSFA: Nested Scale-Free Architecture for Scalable Publish/Subscribe over P2P Networks

Huanyang Zheng and Jie Wu
Department of Computer and Information Sciences, Temple University, USA
Email: {huanyang.zheng, jiewu}@temple.edu

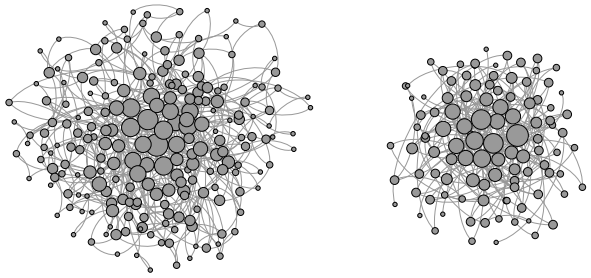
***Abstract*—This paper proposes a scalable publish/subscribe system based on unstructured P2P networks, which are shown to have Nested Scale-Free Architectures (NSFAs). The scale-free architecture is a classic concept, which means that the peer degree distribution follows power-law. 'Nested' indicates that the scale-free architecture is preserved when low-degree peers and their associated connections are removed. We find that NSFA's hierarchy can be distributedly constructed, and has a better bound than classic hierarchies. By leveraging the NSFA's hierarchy, our publish/subscribe system achieves a competitive tradeoff among the event routing efficiency, system robustness, and overhead. For an unstructured P2P network with $|V|$ peers, the number of routing hops for the event deliveries in our system is expected to be $O(\ln \ln |V|)$. For the topological information, each peer only needs to maintain an overhead with a constant size, $O(1)$. Peer arrival, departure, and failure can be handled within a message complexity of $O(\ln \ln |V|)$. Finally, real data-driven experiments demonstrate the efficiency and effectiveness of the NSFA-based publish/subscribe system.**

***Index Terms*—Nested scale-free architecture; network scalability;publish/subscribe; unstructured P2P network**

I. Introduction

Publish/Subscribe (pub/sub) systems are appealing abstractions for the Content Delivery Networks (CDNs). A pub/sub system involves three roles: *subscribers*, *publishers*, and *brokers*. Subscribers express their interests through *subscriptions* with the system, in order to receive *events* matching their subscriptions. Events are issued by publishers and are delivered to subscribers via brokers. Real-world pub/sub system applications include Yahoo Message Broker [5] which is integrated with web applications, Global Data Synchronization Network [4] that exchanges supply chain information among retailers, and SuperMontage [14] that disseminates financial data and orders among traders.

Most existing large-scale pub/sub systems are implemented as overlay networks on the Internet. They require hundreds of servers placed at strategic points across the globe to handle the load, as well as trained professionals to monitor these servers. For example, the infrastructure of Akamai's pub/sub system includes 56,000 dedicated servers among 950 networks in 70 countries [14]. However, such expensive infrastructure costs are not always feasible for many large corporations. Moreover, the scalabilities of these systems are inherently questionable, because all the events are handled on given servers in a centralized manner. As a result, building pub/sub systems on Peer-to-Peer (P2P) networks becomes an alternative option [24]. P2P networks are inexpensive and highly scalable, since all the peers contribute their machines to distributedly increase the computational and storage resources. In P2P-based pub/sub systems, peers are used to not only store events, but also route events to other peers with matching subscriptions.



(a) P2P network with all peers. (b) Top 50% peers.

Fig. 1. NSFA in the Gnutella dataset.

P2P networks can be roughly classified as *unstructured* and *structured*. Unstructured P2P networks, such as Gnutella, Kazaa, and Bitcoin, are formed by peers that "randomly" connect to each other. Since there is not a globally-imposed structure on peers, unstructured P2P-based pub/sub systems have very low construction overheads and are highly robust in terms of frequent churns (peer arrivals and departures). However, routing events from publishers to subscribers becomes extremely inefficient due to the lack of structure. For example, Sub-2-Sub [26] uses a flooding-based routing strategy that leads to a high amount of network traffic. For another example, SIENA [3] cannot provide a bounded routing performance, in terms of the routing hops and overheads. On the other hand, structured P2P networks, such as Chord, P-Grid, and Pastry, organize peers into specific topologies through Distributed Hash Tables (DHTs), and thus enable high-performance routings for pub/sub systems. However, as a tradeoff, structured P2P-based pub/sub systems have considerable construction overheads and are vulnerable to churns, due to the maintenance of large-size DHTs. For example, handling one peer churn in Terpstra [25] and PastryStrings [1] takes logarithmic messages with respect to the number of peers in the system.

Recent advances in network science show that peer connections in unstructured P2P networks are not truly random [2]. We propose that they share a *Nested Scale-Free Architecture* (NSFA). The scale-free architecture is a classic concept [2], meaning that the peer degree distribution follows power-law. In such an architecture, a majority of the periphery peers are

inactive with a small number of connections, while a minority of the core peers are active with a large number of connections. 'Nested' indicates that the scale-free architecture is preserved, when low-degree peers and their associated connections are removed. An example is shown in Fig. 1, where peers with more connections are shown as larger nodes. Fig. 1(a) depicts the largest strongly connected component formed by peers whose IDs are smaller than 500 in the Gnutella dataset [12]. It is scale-free. Then, we iteratively remove the peer with the smallest number of connections, until half of the peers remain. Fig. 1(b) shows the resulting network, which maintains scale-free by the nested property. We find that NSFA's hierarchy can be distributedly constructed, and has a better bound than classic hierarchies [7, 10, 19, 22, 28]. For an unstructured P2P network with $|V|$ peers, the number of hierarchical levels in NSFA can be bounded by $\Theta(\ln|V|)$.

By leveraging the NSFA's hierarchy, this paper proposes a novel distributed pub/sub system for unstructured P2P networks. While our system has very low construction overhead and is highly robust, its event routing efficiency is competitive with those in structured P2P-based pub/sub systems. Our event routing is hierarchical, and has two phases: the publisher first uploads the event to the network core, which in turn downloads the event to the subscriber. For an unstructured P2P network with $|V|$ peers, the number of routing hops can be $O(\ln\ln|V|)$ in our approach. Moreover, instead of using costly DHTs whose sizes scale up with the number of peers, each peer in the proposed system only needs to keep a constant-size overhead as the topological information to handle event routings and peer churns. Peer arrival, departure, and failure can be handled within a message complexity of $O(\ln\ln|V|)$. Our pub/sub system outperforms classic systems, in terms of better bounds of asymptotic performances.

Our main contributions are summarized as follows:

- We address a novel architecture of NSFA in unstructured P2P networks. A distributed labeling scheme is proposed to determine the hierarchical levels in NSFA.
- Based on NSFA's hierarchy, we propose a novel pub/sub system, which has low construction overhead, is highly robust in terms of peer churns, and is efficient for event routings. Our system performance is well-bounded.
- The state-of-the-art pub/sub systems over P2P networks are surveyed in a comparative manner with the proposed work, in terms of the event routing efficiency, system robustness, and overhead.
- Extensive real data-driven experiments on Gnutella and Bitcoin are conducted to evaluate the proposed pub/sub system. Experimental results are shown from different perspectives to provide insightful conclusions.

The remainder of this paper is organized as follows. Section II formulates the problem. Section III presents the NSFA and its hierarchy. Section IV describes the pub/sub system. Section V surveys the related works in a comparative manner with the proposed pub/sub system. Section VI includes experiments. Finally, Section VII concludes the paper.

II. Problem Statement

The objective of this paper is to build an advanced content-based pub/sub system for unstructured P2P networks. An unstructured P2P network is modeled by a directed graph $G = (V, E)$, where V is a set of peers (nodes), and $E \subseteq V^2$ is a set of directed connections among the peers (directed edges). The numbers of incoming and outgoing connections held by a peer are denoted as its indegree and outdegree, respectively. Each peer may publish events, which are conjunctions of attribute-value pairs. For example, {(temperature, 30), (precipitation, 20)} is an event describing the weather conditions. Each peer has its own interests. Peers express their interests through subscriptions with the system, in order to receive interested events matching their subscriptions. Subscriptions are conjunctions of attribute-operator-value tuples. For example, {(temperature, $<$, 40), (precipitation, $>$, 10)} could be one subscription of a peer. A peer may have multiple subscriptions. A subscription matches an event if all the operators in this subscription are satisfied using the corresponding attributes and values of that event. Peers can also route events to other peers. In other words, each peer can be a publisher, a subscriber, a broker, or an arbitrary combination of these three roles.

To evaluate the system performance, we focus on three metrics: *event routing efficiency*, *robustness*, and *overhead*. Event routing efficiency refers to the number of routing hops in event deliveries. Robustness refers to the number of messages used by the system to deal with peer arrivals, departures, and failures. Overhead refers to each peer's storage consumption on the topological information for event routings, peer churns, and peer failures. Classic unstructured P2P-based pub/sub systems are highly robust and have low overheads, but sacrifice routing efficiency due to the lack of structure. On the contrary, classic structured P2P-based pub/sub systems have a high routing efficiency, but perform poorly in terms of robustness and overhead. This is because they need to maintain large-size DHTs as the topological information. By contrast, our system obtains well-bounded results on all these three metrics by leveraging the NSFA of unstructured P2P networks.

We use a hierarchical approach. We start with the inherent hierarchy in unstructured P2P networks (the NSFA's hierarchy in Section III). The hierarchical level of each peer is determined. Based on the NSFA's hierarchy, our pub/sub system is described (Section IV). Then, a comparative survey of related works is given on their asymptotic performances (Section V). Our system has a better bound than classic approaches.

III. Unstructured P2P Networks

A. Scale-Free Architecture

A P2P network is a distributed application that partitions tasks or work loads between peers. They can be roughly classified as unstructured (Gnutella, Kazaa, and Bitcoin) and structured (Chord, P-Grid, and Pastry), depending on whether particular topologies are specified or not by design. In previous decades, peers in unstructured P2P networks were considered to be "randomly" connected to each other. However, recent

Algorithm 1 Distributed labeling scheme for SFA

Input: The peer v and its neighbors.
Output: The hierarchical level of v in SFA.

1: Initialize v as unlabeled.
2: **repeat** in a round-by-round manner **do**
3: **if** v has the smallest degree (including the tie) among all its unlabeled neighbors **then**
4: Set v's label to be the largest label among its labeled neighbors plus one (in the event that v does not have a labeled neighbor, v's label is one).
5: **return** v's label as its hierarchical level.

advances in network science show that peer connections are not truly random [2]. Bulut et al. [2] showed that unstructured P2P networks have the Scale-Free Architecture (SFA):

***Definition** 1:* A network (i.e., G) satisfies SFA, if its node degree distribution follows power-law.

Since P2P networks are directional, the degree could be either indegree or outdegree. Let P_d denote the fraction of peers with a degree of d, the power-law means that:

$$P_d = \frac{\alpha - 1}{d_{\min}} \cdot \left(\frac{d}{d_{\min}}\right)^{-\alpha} \quad (1)$$

α is a constant power-law exponent ranging from 2 to 3 [2]. $d_{\min}$ is also a constant parameter from where the power-law degree distribution holds. Eq. 1 indicates that majority peers hold only a few connections, while minority peers hold lots of connections. Consequently, SFA can produce the following network hierarchy by degree rankings:

***Definition** 2:* SFA's network hierarchy is defined by ranking hierarchical levels based on peer degrees. Peers with smaller degrees have lower hierarchical levels.

Algorithm 1 is proposed as a distributed labeling scheme that determines peers' hierarchical levels in SFA. It works through synchronous peer iterations (a round-by-round manner in line 2). The label of a peer indicates its hierarchical level. Since unstructured P2P networks are directional, Algorithm 1 has an *indegree version* and an *outdegree version*, depending on the degree used in it. For indegree and outdegree versions, the neighbors in Algorithm 1 (lines 3 and 4) refer to the incoming and outgoing neighbors, respectively.

SFA's hierarchy has been widely acknowledged in many existing works [7, 10, 19, 22, 28]. One of the most famous example includes BubbleRap [10], which presents a routing scheme for delay tolerant networks with social features. A message in BubbleRap is iteratively forwarded to nodes with larger degrees to seek the message destination. However, the number of routing hops in BubbleRap is not well-bounded even if the network satisfies SFA. The number of hierarchical levels in SFA could be $\Theta(|V|)$, where $|V|$ is the number of nodes in the network [28]. As a result, hierarchical routings in SFA may perform poorly under certain scenarios. On the other hand, SFA is not sufficient to describe the ad-hoc nature of unstructured P2P networks, where peers arrive and depart dynamically. Further explorations are conducted.

Algorithm 2 Distributed labeling scheme for NSFA

Input: The peer v and its neighbors.
Output: The hierarchical level of v in NSFA.

Same as Algorithm 1, except the subtle change on the if statement in line 3: **if** v has the smallest *effective degree* (including the tie) among all its unlabeled neighbors

B. Nested Scale-Free Architecture

Differing from structured P2P networks that have specified topologies by design, unstructured P2P networks are formed by peers that arrive and depart dynamically. When new peers arrive, they connect to existing peers as new peripheries of the existing network. Such time-evolving peer dynamics create an *onion-like architecture*, which is defined as the NSFA:

***Definition** 3:* Let G_s denote the set of subgraphs generated by iteratively removing the lowest-degree node and its connections in a network, G. We claim that G satisfies NSFA, if (i) G and all the subgraphs in G_s satisfy SFA, and (ii) the standard deviation of their power-law exponents is $o(1)$.

An intuitive explanation of NSFA is that, G satisfies NSFA, if it satisfies SFA and its power-law exponent (i.e., α in Eq. 1) has almost no variation when the current lowest-degree node in G is *iteratively* removed. Subgraphs in G_s, which are overly small to depict SFA, can be ignored by setting up a threshold. Clearly, NSFA is a "nested" extension of SFA. We claim that unstructured P2P networks satisfy NSFA (verified later). NSFA is our novel contribution, and is the key idea of this paper. NSFA produces the following network hierarchy:

***Definition** 4:* NSFA's network hierarchy is defined by ranking hierarchical levels by iteratively removing the set of peers that have the lowest degrees among their neighbors. Peers that are removed earlier have lower hierarchical levels.

The difference between Definitions 3 and 4 is that NSFA's hierarchy is defined by removing all the *local* lowest-degree peers in one iteration, while NSFA is defined by removing the *global* lowest-degree peer in one iteration. Clearly, through local approximations, NSFA's hierarchy can reveal the structural properties of networks that satisfy NSFA. Algorithm 2 is proposed as a distributed labeling scheme to determine peers' hierarchical levels in NSFA. It involves the following concept: the *effective degree* of a peer is defined as its number of connections to its unlabeled neighbors. Algorithm 2 also works through synchronous peer iterations (round-by-round manner in line 2). The label of a peer indicates its hierarchical level. The only difference between Algorithms 1 and 2 is line 3 (the if statement), which shows the prerequisite for the peer labeling. However, it leads to a fundamental insight difference: Algorithm 2 can reveal the nested property, while Algorithm 1 cannot reveal this property. This is because labeling peers in a round of Algorithm 2 is essentially peeling off the current outermost layer of the onion-like network. Once a peer labels itself, its connections are no longer counted in the effective degrees of unlabeled peers. Peers are iteratively peeled off by the labeling process to reveal the onion-like architecture.

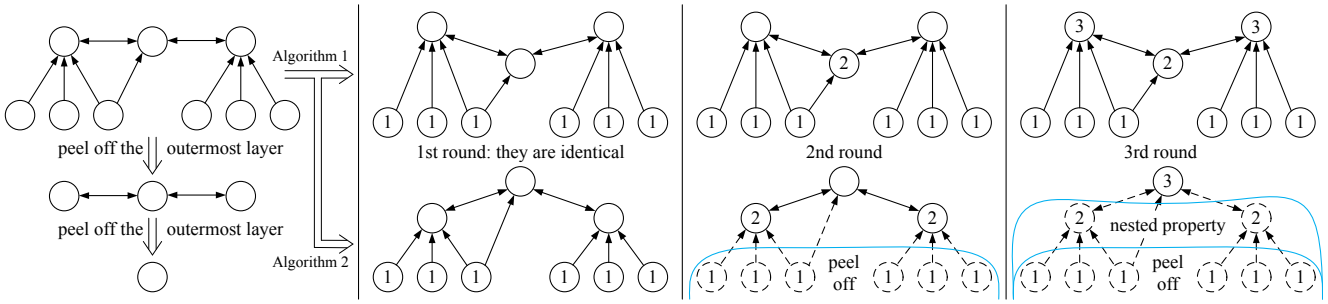


Fig. 2. Difference between Algorithms 1 and 2 (indegree version).

To better illustrate the differences between Algorithms 1 and 2, an example of their indegree versions is shown in Fig. 2. A node represents a peer. The numbers within the nodes are their hierarchical levels. Directed arrows are directed connections among peers. In Fig. 2, the two peers with the largest indegrees have the maximal hierarchical level in Algorithm 1, but they do not have the maximal hierarchical level in Algorithm 2. This is because their connections are peeled off in the first round. Each round of Algorithm 2 peels off the outermost layer of the onion-like network, which is composed of the remaining unlabeled peers. Our next claim is that, as the most important property and the greatest advantage of NSFA, the expected number of rounds for Algorithm 2 to terminate is bounded:

***Theorem** 1:* Suppose an unstructured P2P network has $|V|$ peers and satisfies NSFA. Then, Algorithm 2 is expected to terminate within $\Theta(\ln|V|)$ rounds of synchronous peer iterations. The maximal peer label is also $\Theta(\ln|V|)$.

Proof: The key idea is to show that a constant percentage of unlabeled peers are expected to label themselves in each round of Algorithm 2. We start with the upper bound. Initially, all the peers are unlabeled. Let us consider the probability that an arbitrary peer (say v) labels itself. Suppose v's degree is d_v. Based on Eq. 1, the probability that a neighbor of v has a higher degree than v is:

$$\int_{d_v}^{\infty} \frac{\alpha-1}{d_{\min}} \cdot \left(\frac{d}{d_{\min}}\right)^{-\alpha} \mathrm{d}d = \left(\frac{d_v}{d_{\min}}\right)^{1-\alpha} \tag{2}$$

The node v labels itself when all of its d_v neighbors have larger effective degrees than v. Since all peers are initially unlabeled, the probability that v labels itself is $(d_v/d_{\min})^{(1-\alpha)\times d_v}$. Therefore, the expected percentage of peers that will label themselves in the first round of Algorithm 2 is:

$$\begin{aligned}
&\int_{d_{\min}}^{\infty} \left(\frac{d_v}{d_{\min}}\right)^{(1-\alpha)\times d_v} \cdot \frac{\alpha-1}{d_{\min}} \cdot \left(\frac{d_v}{d_{\min}}\right)^{-\alpha} \mathrm{d}d_v \\
&> \int_{d_{\min}}^{2d_{\min}} \left(\frac{d_v}{d_{\min}}\right)^{(1-\alpha)\times d_v} \cdot \frac{\alpha-1}{d_{\min}} \cdot \left(\frac{d_v}{d_{\min}}\right)^{-\alpha} \mathrm{d}d_v \\
&> \int_{d_{\min}}^{2d_{\min}} \left(\frac{d_v}{d_{\min}}\right)^{(1-\alpha)\times 2d_{\min}} \cdot \frac{\alpha-1}{d_{\min}} \cdot \left(\frac{d_v}{d_{\min}}\right)^{-\alpha} \mathrm{d}d_v \\
&= \frac{1}{2d_{\min}+1}\left[1 - \frac{1}{2^{(\alpha-1)(2d_{\min}+1)}}\right] = c
\end{aligned} \tag{3}$$

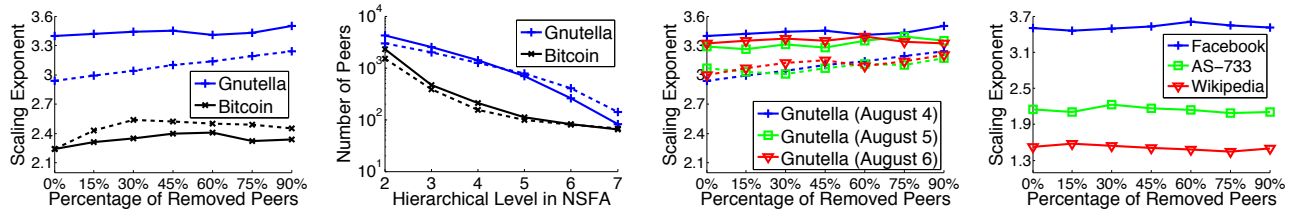
Let c denote the result in Eq. 3. Note that α is a constant that ranges from 2 to 3 [2]. $d_{\min}$ is also a constant parameter. Therefore, c is a positive constant, meaning that more than a constant percentage of unlabeled peers will label themselves in the first round of Algorithm 2. Then, in the second round, these labeled peers are peeled from the remaining network. This is because their connections are not counted in the effective degrees of unlabeled peers. With the nested property, we consider that the effective degree distribution for the unlabeled peers remains the same. This is because peers labeled in one iteration are not adjacent to each other, and the remaining network belongs to G_s by Definition 3. We ignore the limited variation of α. Through the same arguments in Eqs. 2 and 3, more than a constant percentage of unlabeled peers will label themselves in the second round of Algorithm 2. By induction, we conclude that *more than* a constant percentage of unlabeled peers will label themselves in each round of Algorithm 2. Since $|V| \times c^{-\log_c|V|} = 1$, Algorithm 2 is expected to terminate within $-\log_c|V|$ rounds. c is a positive constant that is smaller than one, and thus $-\log_c|V|$ belongs to $O(\ln|V|)$. Therefore, the number of rounds for Algorithm 2 to terminate is expected to be $O(\ln|V|)$.

The proof of the lower bound is similar: *less than* a constant percentage of unlabeled peers will label themselves in each round of Algorithm 2. Similar to Eq. 3, we have:

$$\begin{aligned}
&\int_{d_{\min}}^{\infty} \left(\frac{d_v}{d_{\min}}\right)^{(1-\alpha)\times d_v} \cdot \frac{\alpha-1}{d_{\min}} \cdot \left(\frac{d_v}{d_{\min}}\right)^{-\alpha} \mathrm{d}d_v \\
&< \frac{\alpha-1}{d_{\min}} \int_{d_{\min}}^{\infty} \left(\frac{d_v}{d_{\min}}\right)^{(1-\alpha)\times d_v} \mathrm{d}d_v \\
&= \frac{\alpha-1}{d_{\min}} \int_{d_{\min}}^{\infty} e^{(1-\alpha)\times d_v \times \ln(d_v/d_{\min})} \mathrm{d}d_v \\
&< \frac{\alpha-1}{d_{\min}} \left[\int_{d_{\min}}^{ed_{\min}} \left(\frac{d_v}{d_{\min}}\right)^{(1-\alpha)d_{\min}} \mathrm{d}d_v + \int_{ed_{\min}}^{\infty} e^{(1-\alpha)d_v} \mathrm{d}d_v \right] \\
&= \frac{\alpha-1}{d_{\min}} \left[\frac{e^{(1-\alpha)d_{\min}+1}-1}{(1-\alpha)d_{\min}+1} d_{\min} - \frac{e^{(1-\alpha)ed_{\min}}}{1-\alpha} \right] = c^*
\end{aligned} \tag{4}$$

Let c^* denote the result in Eq. 4. Clearly, c^* is also a constant. Through the same argument, the number of rounds for Algorithm 2 to terminate is expected to be $\Omega(\ln|V|)$.

Combining the upper and lower bounds, we conclude that Algorithm 2 is expected to terminate within $\Theta(\ln|V|)$ rounds of synchronous peer iterations. In each round, the maximal peer label increases by one (line 4 in Algorithm 1, which is also used by Algorithm 2). Therefore, the expected maximal peer label in NSFA is also $\Theta(\ln|V|)$. ■



(a) Power-law scaling exponent. (b) Peer distribution in NSFA. (c) Peer arrival and departure. (d) Other NSFA networks.

Fig. 3. NSFA verifications in unstructured P2P networks.

Theorem 1 shows that *the number of hierarchical levels in NSFA can be logarithmically bounded*. By contrast, classic hierarchies ranked by the node connectivity [19, 28], social centrality [10, 22], or organizational roles [7] cannot guarantee such a bound. This is because they do not require structural similarities among different hierarchical layers [6]. Moreover, NSFA's hierarchy can be constructed distributedly with low overheads, since each node only takes local neighborhood information. Such great advantages of NSFA enable bounded high-performance routings. Furthermore, the number of routing hops in NSFA can be even asymptotically smaller than $\Theta(\ln |V|)$, since some hierarchical levels can be skipped. We will discuss this property later in Section IV (Theorem 2).

C. Verify the Existence of NSFA

To verify the existence of NSFA, real data-driven experiments on Gnutella [12] and Bitcoin [18] are conducted. The Gnutella dataset has 10,876 peers with 39,994 connections on August 4, 2002 (peer arrivals and departures are recorded for the following days). The Bitcoin dataset has 4,579 peers with 18,667 connections (partial dataset for comparisons with Gnutella). Verification results are shown in Fig. 3, where solid and dashed lines are the results of indegree and outdegree versions, respectively. Fig. 3(a) shows the variation of the power-law exponent (i.e., α), when the current lowest-degree peer is iteratively removed. α has a limited variation, even if a large percentage of peers are removed. Based on Definition 3, Gnutella and Bitcoin satisfy NSFA. Then, Fig. 3(b) shows the distribution of peers in NSFA's hierarchy. The number of peers decreases exponentially with respect to the hierarchical level. As a result, the number of hierarchical levels in NSFA is logarithmically bounded. Fig. 3(c) shows the scenario of peer arrivals and departures in Gnutella (three days from August 4 to 6). It can be seen that NSFA naturally holds when peers arrive and depart. The nested architecture actually results from peer arrivals and departures, in which SFA is satisfied. This phenomenon reveals that peer connections are not truly random [2]. Real data-driven experiments validate that unstructured P2P networks satisfy NSFA. Moreover, NSFA is not unique for unstructured P2P networks, i.e., NSFA exists in other types of networks. Experiments are conducted in three undirected networks [12]: Facebook (online social networks), AS-733 (autonomous systems), and Wikipedia dataset (website networks). Fig. 3(d) shows that these networks also satisfy NSFA, i.e., the variation of the power-law exponent is limited when the lowest-degree node is iteratively removed.

TABLE I
COMPARE HIERARCHIES

Dataset	SFA indegree hierarchy		NSFA indegree hierarchy	
	# of levels	# of LMPs	# of levels	# of LMPs
Gnutella	20	1,715	10	120
Bitcoin	41	204	19	76
Dataset	**SFA outdegree hierarchy**		**NSFA outdegree hierarchy**	
	# of levels	# of LMPs	# of levels	# of LMPs
Gnutella	19	1,430	10	676
Bitcoin	36	167	17	43

D. Compare Hierarchies

This subsection experimentally compares SFA's hierarchy and NSFA's hierarchy (Algorithms 1 and 2) in unstructured P2P networks. We start with the following definition:

***Definition** 5:* If a peer has a higher hierarchical level than all of its neighbors, it is a Local Maximum Peer (LMP).

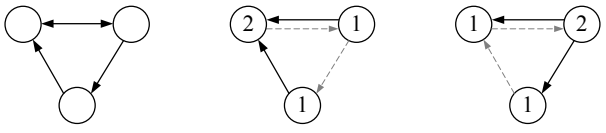
We have two claims: (i) NSFA's hierarchy has fewer hierarchical levels than SFA's hierarchy, and (ii) NSFA's hierarchy in has fewer LMPs than SFA's hierarchy. The first claim means that the maximal peer label in Algorithm 2 is smaller than that in Algorithm 1. Algorithm 2 terminates faster than Algorithm 1. The second claim means that NSFA's hierarchy is a more concentrated. To verify our claims, experiments on Gnutella [12] and Bitcoin [18] are conducted, based on the same settings as in the previous subsection. The result is shown in Table I, where the NSFA's hierarchy has significantly fewer levels than SFA's hierarchy (basically half in both datasets). NSFA's hierarchy has many fewer LMPs than does SFA's hierarchy, especially in the indegree version (fewer than 10% for Gnutella). The above results demonstrate our claims.

NSFA's hierarchy has key advantages over SFA's hierarchy in terms of hierarchical event deliveries in pub/sub systems. This event delivery has two phases. In the first phase, a peer uploads its event to the network core, which is formed by the peers with the highest hierarchical levels. In the second phase, peers in the network core download the event to the subscribed peers. Since NSFA's hierarchy has fewer levels, events can be uploaded to the network core with fewer routing hops. Meanwhile, NSFA's hierarchy has fewer LMPs, meaning that there exist fewer local extrema for the event uploads and downloads. More details are described in the next section.

IV. PUBLISH/SUBSCRIBE IN NSFA

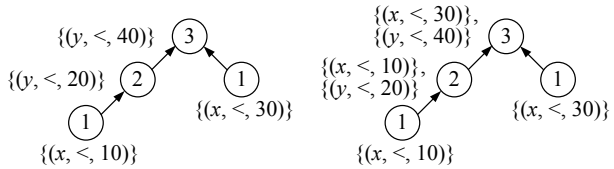
A. System Design Overview

In a general pub/sub system, each peer may publish events and receive its own interested events. Each peer maintains an event filter to determine whether an event should be received



(a) Network. (b) Indegree. (c) Outdegree.

Fig. 4. Examples of NSFA-based forests (indegree and outdegree versions).



(a) Peer interests. (b) Subscriptions.

Fig. 5. An example of peer interests and the corresponding subscriptions.

or not. Each peer also needs some overhead to maintain the topological information for event routings, peer churns, and peer failures. In the proposed pub/sub system, such topological overheads of a peer only include its hierarchical level in both the indegree and outdegree versions of NSFA. Consequently, the storage consumption of a peer is a constant that does not scale up with the network size. In other words, the size of the overhead per peer is $O(1)$.

The event routing is accomplished through NSFA's hierarchy. This hierarchy abstracts *a forest of rooted trees* from the unstructured P2P network. The roots of these trees are LMPs. For a non-LMP peer, its parent in the tree is the neighbor that has the maximal hierarchical level. Since Algorithm 2 has two versions, the resultant forest has both indegree and outdegree versions. Fig. 4 shows such an example. Fig. 4(a) shows a network. The corresponding forests of indegree and outdegree versions are shown in Figs. 4(b) and 4(c), respectively. The numbers within the nodes represent their hierarchical levels in NSFA. Connections, which are not in the forest, are gray and dashed. We claim that the maximal tree depth in an NSFA-based forest can be bounded:

***Theorem** 2:* The maximal tree depth in an NSFA-based forest is expected to be $O(\ln\ln|V|)$.

Proof: Let $l_{\max}$ denote the maximal peer label in NSFA. We start with an arbitrary peer (say v) in an arbitrary tree of the NSFA-based forest. Let l_v denote the label of this peer. Suppose the labels of v's neighbors are uniform-randomly distributed from 1 to $l_{\max}$. Then, for v's neighbors whose labels are larger than l_v, they are expected to have labels of $l_v + (l_{\max} - l_v)/2$. Since v's parent has the maximal label among v's neighbors, its label is expected to be larger than $l_v + (l_{\max} - l_v)/2$. In other words, the label difference between the current label and the maximal label is expected to be at least halved, when we move from an arbitrary peer to its parent. Since $2^{\log_2 l_{\max}} = l_{\max}$, it takes $O(\ln l_{\max})$ steps to move from a leaf to a root in a tree. Since Theorem 1 has stated that $l_{\max} \in \Theta(\ln|V|)$, the maximal tree depth in an NSFA-based forest is expected to be $O(\ln\ln|V|)$. ■

The insight of Theorem 2 is that a peer may have a parent with a hierarchical level that is much higher than its own, leading to a double logarithmic tree depth. Theorems 1 and 2 show that connections in unstructured P2P networks are not truly random, in terms of NSFA. The maximal tree depth of an NSFA-based forest has a better bound than classic forests [17, 20]. Moreover, our NSFA-based forest can be constructed in a distributed manner. Our key idea is to utilize the NSFA-based forest as the network backbone to deliver events among peers with a bounded performance and a limited overhead.

B. Subscription

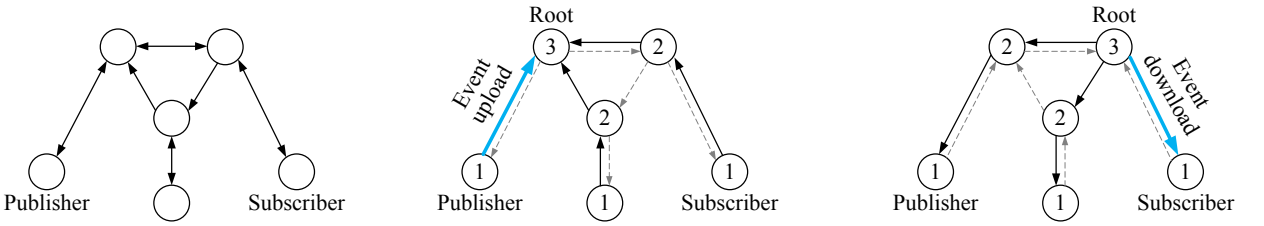
Peers express their interests through subscriptions with the system. Such subscriptions are captured by the event filter, which is maintained by each peer to determine whether an event should be received or not. The NSFA-based forest of the indegree version is used to collect these subscriptions. A peer may not only receive its own interests, but also may receive other peers' interests for the purpose of event deliveries. In our approach, a peer collects all the events interesting to itself, along with the interests of its descendants in the NSFA-based forest. Hence, a peer notifies all of its precedents in terms of its interests. An example is shown in Fig. 5, where x and y are two attributes. Fig. 5(a) shows the interests of each peer (attribute-operator-value tuples). Fig. 5(b) shows the resulting subscriptions. The root will receive all the events matching that x is smaller than 30 or y is smaller than 40.

A peer subscription takes $O(\ln\ln|V|)$ messages. This is because this peer needs to notify all of its precedents, while the maximal tree depth is bounded by Theorem 2. The unsubscription process is similar and also takes $O(\ln\ln|V|)$. Note that peers with higher hierarchical levels tend to have more descendants, and thus they tend to collect more events. Hence, their event matchings are more time-consuming and can be accelerated by some existing works [13, 16].

C. Event Delivery

The proposed pub/sub system uses a two-phase hierarchical event delivery scheme. The first phase is the upload phase, which utilizes the NSFA-based forest of the indegree version. Once a peer wants to publish an event, it will upload this event to the root of its tree, through recursively forwarding this event to the parent. Once the root receives an event in the upload phase, the second phase, or download phase, begins. This phase utilizes the NSFA-based forest of the outdegree version. Once a peer receives a matched event, it will forward this event to all of its children in the tree. This is because a peer collects all the events interesting to itself and its descendants through subscriptions. If the received event is not matched, the peer will drop it. In summary, the publisher first uploads the event to the network core (peers with highest hierarchical levels), which in turn downloads the event to the subscriber.

A potential problem is that multiple roots may exist in the NSFA-based forest. Meanwhile, the roots in the NSFA-based forest of the indegree version are not necessarily roots in the NSFA-based forest of the outdegree version. Hence, roots need to connect with each other before the event deliveries. Roots can register their addresses at an extra registration server.



(a) Pub/Sub over a P2P network. (b) NSFA-based forest (indegree version). (c) NSFA-based forest (outdegree version).

Fig. 6. An illustration for the two-phase hierarchical event delivery in the proposed pub/sub system.

Since Table I shows that roots are few, the registration cost is ignorable. Through the registration server, roots can set up special connections to each other before the event deliveries. Similar techniques have been used in existing works [17, 20].

Fig. 6 illustrates an example for the event deliveries. The network topology is shown in Fig. 6(a). The publisher and subscriber are the peers who publish and subscribe to the event, respectively. Fig. 6(b) shows the upload event delivery phase in the NSFA-based forest of the indegree version. For simplicity, this forest includes only one tree. The numbers within the nodes (peers) are their hierarchical levels. Connections without the forest are gray and dashed. The event is uploaded to the root, by recursively being forwarded to the parent in the tree. Once the root receives the event, the upload phase terminates. The root sends this event to the other roots through special connections that are set up in advance of the event deliveries. Then, the download event delivery phase begins at the root in the NSFA-based forest of the outdegree version. As shown in Fig. 6(c), the peer will forward matched events to all of their children, but drop non-matched events.

Since Theorem 2 claims that the tree depth is $O(\ln\ln|V|)$, the number of routing hops for the event deliveries is also $O(\ln\ln|V|)$. The upload phase takes $O(\ln\ln|V|)$ to deliver the event from the publisher to the root, while the download phase also takes $O(\ln\ln|V|)$ to deliver the event from the root to the subscriber. The proposed upload-and-download routing scheme is based on classic hierarchical routing schemes [10, 17, 20, 27]. Our novel contribution is the NSFA. NSFA's hierarchy can be distributedly constructed, and has a better bound than classic hierarchies. The outstanding performance of our event delivery comes from the bound of NSFA's hierarchy. Our work is asymptotically compared with classic approaches in Table II (later in Section V).

D. Peer Arrival, Departure, and Failure

Our pub/sub system can handle the peer arrival, departure, and failure. As shown in Fig. 3(c), NSFA can naturally hold when peer arrives and departs. NSFA also holds if the failure is uniformly distributed. Therefore, we only need to adjust the hierarchical levels of peers. (i) When a new peer arrives at the pub/sub system, its hierarchical level is set to the lowest hierarchical level among all of its neighbors (or one in the case of no neighbor). Then, this peer will express its interests through subscriptions, taking $O(\ln\ln|V|)$ messages. (ii) If a non-root peer wants to depart, it will unsubscribe from the system. Its connections with its children can be transferred to its parent. If a root peer wants to depart, its neighbor with the highest hierarchical level will be selected as the new root. The connections and subscriptions of this root will be transferred to the new root. The new root will register its address within the network to set up connections with the other roots. (iii) If a peer finds a failing child, it examines the subscriptions of its children and then unsubscribe the failed child. If a peer finds a parent failure, it re-selects a parent to re-subscribe. Since Theorem 2 claims that the tree depth is $O(\ln\ln|V|)$, the peer arrival, departure, and failure take $O(\ln\ln|V|)$ messages. Our pub/sub system can also run Algorithm 2 periodically to reset NSFA's hierarchy. Moreover, the following theorem shows that, when a peer's interests are popular, the number of messages for its arrival, departure, and failure can be reduced:

***Theorem** 3:* If a peer has the same interests as $\Omega(\frac{1}{\ln\ln|V|})$ fraction of peers in the system, then its arrival, departure, and failure are expected to take $O(\sqrt{\ln\ln|V|})$ messages.

Proof: Suppose a peer, v, has the same interests as a fraction, f, of all peers in the system. Note that a peer collects all the events interesting to itself, along with the interests of its descendants in the NSFA-based forest. Hence, the subscriptions of peers with larger labels are more likely to include v's interests than those with smaller labels. For peers with labels of one, f fraction of their subscriptions include v's interests. These peers (a fraction, c, of total peers by Eq. 3) notify their parents of their subscriptions. For peers whose labels are larger than one, the fraction of their subscriptions that include v's interests is $1-(1-f)\times(1-\frac{c}{1-c}\times f)\approx(1+c)f$, assuming $f\ll 1$ and $c\ll 1$. Then, let us remove the peers with labels of one, and look at peers with larger labels. By induction, for peers with labels of $i+1$, the fraction of their subscriptions that include v's interests is $(1+ic)f$.

We start with the peer arrival. Once a new peer (say u) arrives, it needs to notify all of its precedents to express its interests. However, such a notification can terminate earlier, if the subscription of u's precedent already includes u's interests. Suppose u has a distance of L to the root. Then, the expected number of notification messages is:

$$\sum_{i=1}^{L}\Big\{\prod_{j=0}^{i-1}\big[1-(1+jc)f\big]\Big\}\leq\sum_{i=1}^{L}\big[1-(1+\frac{i-1}{2}c)f\big]^i \qquad (5)$$

In Eq. 5, $\prod_{j=0}^{i-1}\big[1-(1+jc)f\big]$ is the upper bound of the probability that the i-th message in the notification process is sent. When $f=0$, L messages are sent for the subscription.

TABLE II
COMPARISONS AMONG EXISTING SYSTEMS AND OUR NSFA-BASED SYSTEM.

Metrics	Structured P2P-based Pub/Sub			Unstructured P2P-based Pub/Sub			
	Terpstra [25]	Meghdoot [9]	PastryStrings [1]	Sub-2-Sub [26]	Vitis [17]	Poldercast [20]	NSFA-based
Event routing	$O(\ln \lvert V\rvert)$	$O(\tau \lvert V\rvert^{\frac{1}{\tau}})$	$O(\log_\mu \lvert V\rvert)$	$O(\lvert V\rvert)$	$O(\ln^2 \lvert V\rvert)$	$O(\ln \lvert V\rvert)$	$O(\ln \ln \lvert V\rvert)$
System robustness	$O(\ln \lvert V\rvert)$	$O(\tau \lvert V\rvert^{\frac{1}{\tau}})$	$O(\log_\mu \lvert V\rvert)$	$O(1)$	N/A	$O(\ln \lvert V\rvert)$	$O(\ln \ln \lvert V\rvert)$
Overhead	$O(\ln \lvert V\rvert)$	$O(\tau)$	$O(\mu \log_\mu \lvert V\rvert)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$

When $f \in \Theta(\frac{1}{L})$ and $i \in \Theta(\sqrt{L})$, $[1-(1+\frac{i-1}{2}c)f]^i$ becomes a constant according to the definition of the Euler's number. When $f \in \Theta(\frac{1}{L})$, $\sum_{i=1}^{L}\left[1-(1+\frac{i-1}{2}c)f\right]^i \in O(\sqrt{L})$. This is because the terms with $i \in \Omega(\sqrt{L})$ can be ignored in the summation. Since Theorem 2 claims that $L \in O(\ln \ln |V|)$, the proof completes the part for peer arrivals.

The proofs for peer departures and failures are similar and are omitted due to the page limitation. The key idea is that, when a peer has the same interests as many existing peers in the system, its arrival, departure, and failure can be handled more locally instead of notifying all the precedents. ■

V. RELATED WORKS AND DISCUSSIONS

The pub/sub system is a well-known paradigm for CDNs with the objective of providing efficient event deliveries from the publishers to the subscribers [11]. Classic pub/sub systems are usually implemented as overlay networks on the Internet, using hundreds of servers to deliver content to clients [14]. Since traditional pub/sub systems are infrastructure-dependent, infrastructure-free P2P-based pub/sub systems become another option [25]. Early pub/sub systems were built on unstructured P2P networks due to their low construction overheads and inherent robustness. However, event routing is inefficient [26], since peers are "randomly" connected to each other. Therefore, state-of-the-art pub/sub systems are built on structured P2P networks to improve the routing performance, through using DHTs to organize peers into specific topologies [23]. Nevertheless, the maintenance of DHTs lead to high construction overheads, and degrades the system robustness. We focus on an asymptotical approach. Therefore, some classic pub/sub systems (e.g., SIENA [3], PADRES [8], and Hermes [15]) are not compared, since their performances are not bounded.

Table II compares the existing systems and the NSFA-based pub/sub system, in terms of the event routing efficiency (number of routing hops for event deliveries), system robustness (message complexity for peer churns and failures), and overhead (storage consumption per peer). $|V|$ is the number of peers in the pub/sub system. Terpstra [25], Meghdoot [9] and PastryStrings [1] are structured P2P-based pub/sub systems. Terpstra organizes peers as a Chord [21]. The topology of Meghdoot is a Cartesian space with a dimensionality of τ. Compared with Terpstra, Meghdoot has a smaller overhead at the cost of a worse event routing efficiency and a worse system robustness. PastryStrings has a prefix-based routing through string trees, where μ is a pre-specified digit base. It sacrifices the overhead to improve the event routing efficiency. Sub-2-Sub [26], Vitis [17], and Poldercast [20] are unstructured P2P-based pub/sub systems. Sub-2-Sub considers that connections among peers are random, and uses broadcasts to deliver events. Sub-2-Sub has a very poor routing performance, but is highly robust. Vitis and Poldercast also consider that peer connections are random. While Vitis uses a clustering-based event delivery protocol, Poldercast chains all the peers to a ring network with shortcuts for efficient event routings (based on small worlds).

By contrast, our pub/sub system has the best event routing efficiency by leveraging the inherent NSFA of unstructured P2P networks, where peer connections are not truly random [2]. NSFA's hierarchy can be distributedly constructed, and has a better bound than classic hierarchies [7, 10, 19, 22, 28]. The overhead of the NSFA-based pub/sub system is asymptotically optimal (a constant size per peer), since each peer only needs to maintain its hierarchical level as the topological information. The system robustness is competitive, and can be further reduced for scenarios in which peers share the same interests (Theorem 3). A notable point for the proposed system is that peers with higher hierarchical levels tend to have larger loads than peers with lower hierarchical levels (the load is not balanced). We argue that peers with more connections could have more computational and storage resources to share, i.e., "with great power comes great responsibility."

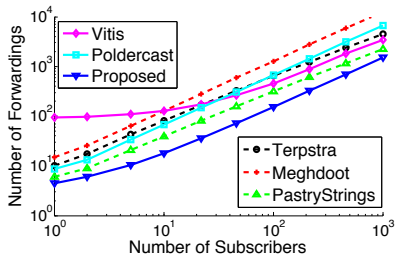
VI. EXPERIMENTS

A. Real Data-driven Experiments

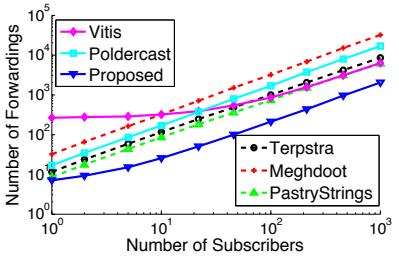
This section conducts real data-driven experiments based on Gnutella [12] and Bitcoin [18]. To reveal the asymptotic performance gap, we use the complete Bitcoin dataset, which includes 6,336,769 peers and 37,450,461 connections. Our pub/sub system on Gnutella can facilitate large-size content deliveries, such as high-definition movie sharing services and high-volume enterprise data distributions. Our pub/sub system on Bitcoin can facilitate financial order disseminations among traders and payroll transactions among workers.

B. Pub/Sub System Evaluations

The proposed pub/sub system is evaluated through comparisons with Terpstra, Meghdoot, PastryStrings, Vitis, and Poldercast. These systems have been introduced in Section V. Sub-2-Sub is not included, since it has an extreme design with the worst event routing efficiency, the best system robustness, and the smallest overhead. While unstructured P2P-based pub/sub systems are tested directly on the Gnutella and Bitcoin datasets, structured P2P-based pub/sub systems are tested with their own topologies that have the same number of peers with the datasets. In Meghdoot, we set $\tau = 8$ to minimize $\tau|V|^{\frac{1}{\tau}}$ for $|V| = 10,876$ in the Gnutella dataset. In PastryStrings, we use $\mu = 4$ as its digit base to encode event routings.

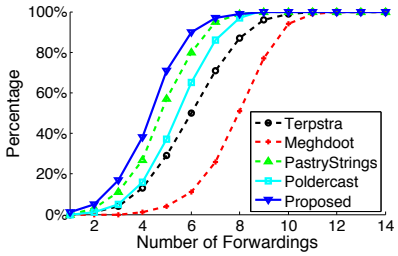


(a) Gnutella.

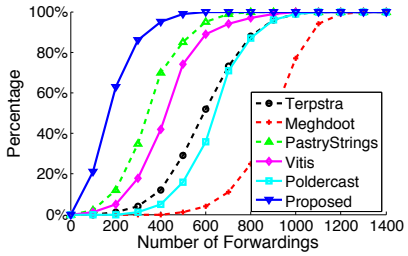


(b) Bitcoin.

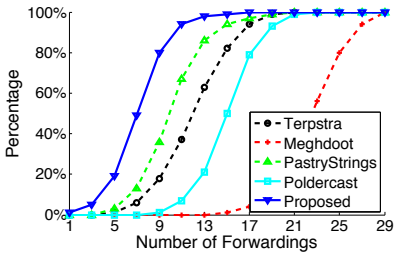
Fig. 7. Event routing efficiency.



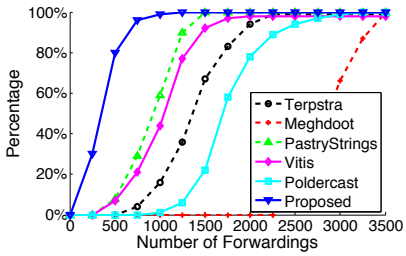
(a) Gnutella with one subscriber.



(b) Gnutella with one hundred subscribers.



(c) Bitcoin with one subscriber.



(d) Bitcoin with one hundred subscribers.

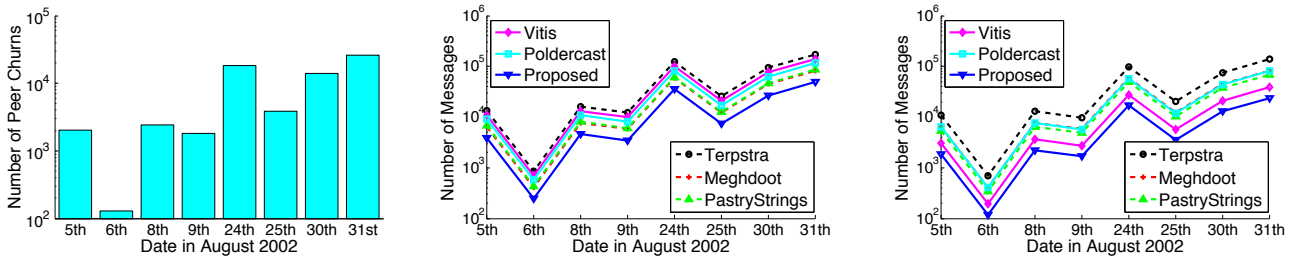
Fig. 8. The CDF for the number of event forwardings.

To evaluate the event routing efficiency, one publisher and several subscribers are uniform-randomly selected. The result of the total number of forwardings from the publisher to all subscribers in an event delivery is shown in Fig. 7. For smoothness, it is averaged over 100,000 times. The NSFA-based pub/sub system outperforms the others, since it has the best asymptotical bound of $O(\ln \ln |V|)$. Vitis has the worst performance when subscribers are few, since it has a bad asymptotical bound of $O(\ln^2 |V|)$. Meghdoot performs poorly due to the same reason. A notable point is that the performance gap between the proposed system and the other system is larger in Bitcoin than that in Gnutella. This is because Bitcoin has many more peers than Gnutella. The Bitcoin dataset has 6,336,769 peers and 37,450,461 connections. The asymptotical performance gap is revealed, when the network scales up.

Fig. 8 shows the Cumulative Distribution Function (CDF) with respect to the total number of forwardings in Fig. 7. Left and right subfigures are the results with one subscriber and one hundred subscribers, respectively. Vitis is not included in Figs. 8(a) and 8(c), since it needs a much larger number of forwardings when subscribers are few. Figs. 8(a) and 8(b) show that, when there are more subscribers, the proposed system has a more significant advantage over the other systems due to the lower asymptotical bound. Fig. 8(a) also shows that the NSFA-based system is not likely to deliver the event from a publisher to a subscriber within 3 forwardings, due to the event upload phase. On the other hand, most event deliveries in our system can be finished within 7 forwardings (Gnutella with one subscriber). When we have 100 subscribers, as shown in Fig. 8(b), most event deliveries can be finished within 500 forwardings, which is less than $7 \times 100 = 700$ forwardings. This is because the upload phases of different subscribers are shared. Figs. 8(c) and 8(d) present similar results for Bitcoin. The proposed pub/sub system performs significantly better than the other systems in Bitcoin, due to the asymptotical performance advantage in large-scale networks. In Fig. 8(d), the number of forwardings used by our system is basically one third of PastryStrings and Vitis.

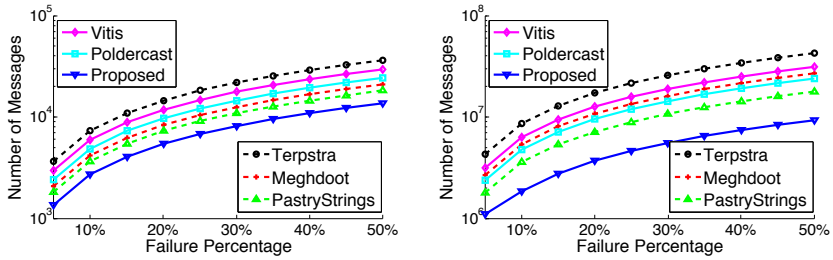
To evaluate the system robustness, the Gnutella dataset is further explored, since it has the peer churn records. However, the Bitcoin dataset does not include the peer churn records. Fig. 9(a) shows the available peer churn statistics in Gnutella on August 2002. August 24 and 31 have the largest numbers of peer churns. Note that the number of peer churns cannot be ignored with respect to the total number of peers in Gnutella. As shown in Fig. 3(c), NSFA holds when peer churns. Fig. 9(b) shows the number of messages dealing with peer churns, when peers do not share an interest. The number of messages scales up with the number of peer churns. Our NSFA-based pub/sub system uses the smallest number of messages to deal with peer churns. By contrast, the number of messages used by Terpstra is nearly quadrupled. To study the message complexity when peers have common interests, we set up a special scenario where the interest of each peer is uniform-randomly chosen from ten given interests. The result is shown in Fig. 9(c). Our system and Vitis have lower message complexities since they can handle peer churns more locally.

The number of messages used for peer failures is shown in Fig. 10, where a given percentage of peers are randomly chosen to fail. Our NSFA-based system uses the smallest number of messages to handle the peer failures. By contrast, the number of messages used by Terpstra is almost tripled. Experiments are not conducted with respect to the overhead, since the overhead of our system is asymptotical optimal, i.e., $O(1)$. Instead, we study the forwarding load distribution of root peers for our system in Gnutella and Bitcoin. 10,000 publisher-subscriber pairs are uniform-randomly selected for event deliveries. Fig 11 shows the cumulative distribution function of root peer forwardings. None of the root peers have more than 700 forwardings within 10,000 event deliveries.



(a) Peer churn statistics. (b) Message complexity (no shared interest). (c) Message complexity (shared interests).

Fig. 9. System performance with respect to the peer churns in Gnutella (the dataset only includes eight nonconsecutive days).



(a) Gnutella. (b) Bitcoin.

Fig. 10. System performance with respect to the peer failures.

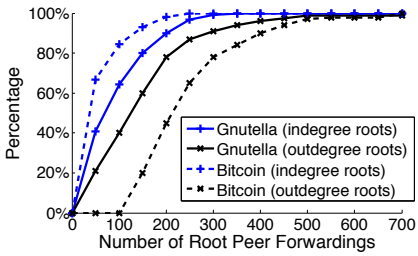


Fig. 11. The CDF of root peer forwardings.

VII. Conclusion

This paper proposes a scalable pub/sub system based on unstructured P2P networks, which are shown to have NSFAs. NSFA's hierarchy can be distributedly constructed, and has a better bound than classic hierarchies. By leveraging NSFA's hierarchy, our system achieves a competitive tradeoff among the event routing efficiency, system robustness, and overhead. The number of event routing hops is $O(\ln \ln |V|)$. Each peer only maintains an overhead of a constant size . Peer arrival, departure, and failure can be handled within a message complexity of $O(\ln \ln |V|)$. Experiments demonstrate the outstanding performance of the proposed pub/sub system.

Acknowledgement

This research was supported in part by NSF grants CNS 1449860, CNS 1461932, CNS 1460971, CNS 1439672, CNS 1301774, and ECCS 1231461.

References

[1] I. Aekaterinidis and P. Triantafillou. PastryStrings: A comprehensive content-based publish/subscribe DHT network. In *ICDCS 2006*.
[2] E. Bulut and B. K. Szymanski. Constructing limited scale-free topologies over peer-to-peer networks. *TPDS*, 2014.
[3] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf. Design and evaluation of a wide-area event notification service. *TOCS*, 2001.
[4] C. Chen, R. Vitenberg, and H.-A. Jacobsen. A generalized algorithm for publish/subscribe overlay design and its fast implementation. *Distributed Computing*, 2012.
[5] B. F. Cooper, R. Ramakrishnan, U. Srivastava, A. Silberstein, P. Bohannon, H.-A. Jacobsen, N. Puz, D. Weaver, and R. Yerneni. PNUTS: Yahoo!'s hosted data serving platform. *VLDB Endowment*, 2008.
[6] B. Corominas-Murtra, J. Goñi, R. V. Solé, and C. Rodríguez-Caso. On the origins of hierarchy in complex networks. *PNAS*, 2013.
[7] R. Daft. *Organization theory and design*. Cengage learning, 2012.
[8] E. Fidler, H.-A. Jacobsen, G. Li, and S. Mankovski. The PADRES distributed publish/subscribe system. In *FIW 2005*.
[9] A. Gupta, O. D. Sahin, D. Agrawal, and A. E. Abbadi. Meghdoot: content-based publish/subscribe over P2P networks. In *Middleware 2004*.
[10] P. Hui, J. Crowcroft, and E. Yoneki. Bubble rap: Social-based forwarding in delay-tolerant networks. *TMC*, 2011.
[11] K. Jayaram, P. Eugster, and C. Jayalath. Parametric content-based publish/subscribe. *TOCS*, 2013.
[12] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graph evolution: Densification and shrinking diameters. *TKDD*, 2007.
[13] A. Margara and G. Cugola. High-performance publish-subscribe matching using parallel hardware. *TPDS*, 2014.
[14] V. Muthusamy and H.-A. Jacobsen. Infrastructure-free content-based publish/subscribe. *ToN*, 2014.
[15] P. R. Pietzuch and J. M. Bacon. Hermes: A distributed event-based middleware architecture. In *ICDCS 2002*.
[16] S. Qian, J. Cao, Y. Zhu, and M. Li. REIN: A fast event matching approach for content-based publish/subscribe systems. In *INFOCOM 2014*.
[17] F. Rahimian, S. Girdzijauskas, A. H. Payberah, and S. Haridi. Vitis: A gossip-based hybrid overlay for internet-scale publish/subscribe enabling rendezvous routing in unstructured overlay networks. In *IPDPS 2011*.
[18] F. Reid and M. Harrigan. An analysis of anonymity in the bitcoin system. *Security and Privacy in Social Networks*, 2013.
[19] J. F. Rodrigues Jr, H. Tong, J.-Y. Pan, A. J. Traina, C. Traina, and C. Faloutsos. Large graph analysis in the gmine system. *TKDD*, 2013.
[20] V. Setty, M. Van Steen, R. Vitenberg, and S. Voulgaris. Poldercast: Fast, robust, and scalable architecture for P2P topic-based pub/sub. In *Middleware 2012*.
[21] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. *ACM SIGCOMM Computer Communication Review*, 2001.
[22] M. Takaffoli, O. R. Zaïane, et al. Social network analysis and mining to support the assessment of on-line student participation. *ACM SIGKDD Explorations Newsletter*, 2012.
[23] M. A. Tariq, B. Koldehofe, and K. Rothermel. Efficient content-based routing with network topology inference. In *DEBS 2013*.
[24] M. A. Tariq, B. Koldehofe, and K. Rothermel. Securing broker-less publish/subscribe systems using identity-based encryption. *TPDS*, 2014.
[25] W. W. Terpstra, S. Behnel, L. Fiege, A. Zeidler, and A. P. Buchmann. A peer-to-peer approach to content-based publish/subscribe. In *DEBS 2003*.
[26] S. Voulgaris, E. Rivière, A.-M. Kermarrec, and M. Van Steen. Sub-2-Sub: Self-organizing content-based publish and subscribe for dynamic and large scale collaborative networks. In *IPTPS 2006*.
[27] X. Xu, R. Ansari, A. Khokhar, and A. V. Vasilakos. Hierarchical data aggregation using compressive sensing (hdacs) in wsns. *TOSN*, 2015.
[28] S. Yang, Q. Sun, S. Ji, P. Wonka, I. Davidson, and J. Ye. Structural graphical lasso for learning mouse brain connectivity. In *KDD 2015*.