# SELECT: A Distributed Publish/Subscribe Notification System for Online Social Networks

Nuno Apolónia*, Stefanos Antaris‡, Sarunas Girdzijauskas†, George Pallis‡, Marios Dikaiakos‡
*Universitat Politècnica de Catalunya, Spain
†Royal Institute of Technology, Sweden
‡University of Cyprus, Cyprus
apolonia@ac.upc.edu, sarunasg@kth.se, {santar01, gpallis, mdd}@cs.ucy.ac.cy

***Abstract*—Publish/subscribe (pub/sub) mechanisms constitute an attractive communication paradigm in the design of large-scale notification systems for Online Social Networks (OSNs). To accommodate the large-scale workloads of notifications produced by OSNs, pub/sub mechanisms require thousands of servers distributed on different data centers all over the world, incurring large overheads. To eliminate the pub/sub resources used, we propose SELECT - a distributed pub/sub social notification system over peer-to-peer (P2P) networks. SELECT organizes the peers on a ring topology and provides an adaptive P2P connection establishment algorithm where each peer identifies the number of connections required, based on the social structure and user availability. This allows to propagate messages to the social friends of the users using a reduced number of hops. The presented algorithm is an efficient heuristic to an NP-hard problem which maps workload graphs to structured P2P overlays inducing overall, close to theoretical, minimal number of messages. Experiments show that SELECT reduces the number of relay nodes up to 89% versus the state-of-the-art pub/sub notification systems. Additionally, we demonstrate the advantage of SELECT against socially-aware P2P overlay networks and show that the communication between two socially connected peers is reduced on average by at least 64% hops, while achieving 100% communication availability even under high churn.**

***Index Terms*—Publish/Subscribe, P2P Network, Social Networks**

## I. Introduction

One of the fundamental services for Online Social Networks (OSNs) is the real-time delivery of notifications due to users' social interactions. Notifications constitute one of the primary ways social users first learn about the activity of their social friends or their preferable sources (e.g. groups, pages). Twitter users generate on average 8,000 tweets per second[1] which amounts to 500 million of notifications per day from more than 300 million of active users. Thus, large-scale notification systems require to be scalable.

Publish/subscribe (pub/sub) systems are an attractive solution for the design of large-scale social notification systems. A pub/sub system provides a form of interaction among social friends where each social user is subscribed to his/her social friends in order to receive notifications, in effect forming a social graph. In the case of scalable pub/sub systems, such as Google Cloud Pub/Sub[2], typically massive corporate resources are required to accommodate large-scale workloads of social notifications. Likewise, IBM deploys over a thousand servers on geographically distributed data centers [1] to provide a high-quality pub/sub system. Moreover, with the advent of the Internet of Things (IoT) the number of devices is estimated to reach 20 billion[3] by 2020 [2]. Also, the integration of the IoT with social networks [3, 4] will increase the required computational resources, further motivating research for more advanced pub/sub overlay designs.

The above motivations attracted the attention of both academia [5, 6] and industry [7] to decentralized OSNs (DOSNs) [8] and provide pub/sub systems for OSNs [1] using Peer-to-Peer (P2P) networks [9, 10]. In DOSNs, social users are connected in a P2P network and interact with their social friends using the P2P routing mechanism. However, designing a scalable P2P pub/sub notification system for DOSNs requires four main challenges to be addressed, as follows:

- ***Relay Nodes***: A key characteristic of P2P pub/sub systems [5, 11] is that they leverage a generic overlay network (e.g. DHT, tree, full-mesh) without projecting the social graph in the P2P overlay network. Since social users are not always directly connected in the P2P overlay network, the message dissemination in P2P pub/sub systems relies on peers (also known as *relay nodes*) that may or may not be interested for the message.
- ***High Traffic***: Recent pub/sub systems [1] try to simplify the design of the routing tree and focus on the construction of the P2P overlay network in order to improve the efficiency of message dissemination. Each peer has a bounded number of connections that can be maintained, the selection of which is accomplished without leveraging the social graph and the social interactions. Hence, the generated P2P overlay network presents load balancing problems, where a portion of the peers has high traffic overhead against the rest of the peers, due to the high social interactions and the absence of social integration to the design of the overlay.
- ***Dissemination Latency***: Each peer in the P2P overlay network presents different upload and download bandwidth characteristics. Since each peer has a bounded number of connections, retaining a P2P connection with a poor bandwidth rate increases the dissemination latency that affects the overall performance of the P2P pub/sub system.
- ***Dynamic environment***: It is essential for the success of the OSN to provide a failure resilient P2P pub/sub system with minimum disruption to the communication between social

[1] http://www.statista.com/
[2] https://cloud.google.com/pubsub
[3] https://gartner.com/newsroom/id/3598917

1530-2075/18/$31.00 ©2018 IEEE
DOI 10.1109/IPDPS.2018.00106

friends. The design of a churn-resistant P2P pub/sub system has been studied in OMen [6]; however, OMen falls short of identifying the online activity of each social user, which poses an additional latency overhead as the establishment of a P2P connection requires a Multi-Path TCP connection [12].

In synopsis, the major issue that arises from current pub/sub implementations is that they are oblivious to specific workloads which result from specific social structure and interactions, as well as, connectivity restrictions (NATs, firewalls, and others).

**Contributions.** To address the above challenges, we introduce a fully decentralized pub/sub system for DOSNs, called SELECT. The proposed system organizes peers on a ring topology and exploits both the social graph and the online activity of each social user to establish connections between peers. The intuition behind this is the construction of a P2P overlay network that acts as a substrate for the pub/sub system with the minimum number of *relay nodes* and the minimum communication interruption between social users. Therefore, by harnessing the social network graph we are able to build an overlay in which messages propagate towards the subscribers with minimum relay nodes in between (while relay nodes may also be subscribers). SELECT is the first approach which exploits the small-world properties by embedding it into the overlay networks' ID space. Thus, peers are placed in the same area based on their social proximity, establishing a bounded and adaptive number of connections to peers. We consider our approach as solving an NP-hard problem, where a P2P overlay is induced from a workload social graph embedded into the identifier (ID) space. Thus, decentralized greedy routing is not only possible but also very efficient and equivalent to routing in navigable small-world networks [13]. In summary, we define our contributions as:

- A proposal for a full decentralization of a pub/sub system for DOSNs that exploits both social graphs and online activities of the users. We use Locality Sensitive Hashing (LSH) [14] and Cumulative Moving Average (CMA)[4] to identify which connections allow message propagation with minimum hops, as well as, which peers potentially present better online behaviour over time.
- The description and evaluation of an ID re-assignment process which projects the social graph on a P2P overlay network, minimizing the distance on the overlay networks' ID space.
- The SELECT algorithm, which creates a global overlay network that allows for message propagation with minimum number of hops, taking advantage of peers that present better online behaviour over time instead of relying on random peers. SELECT is adaptive to dynamic environments with the use of novel recovery mechanisms, applying re-routing when it is required.
- An evaluation and analysis by means of simulation and experimentation with real-world data sets, in order to understand the value of each step of the proposed approach and the performance gain in the pub/sub system.

To prove the efficiency of SELECT on pub/sub systems we designed and developed a browser-based P2P pub/sub system[5] using the free and open-source W3C standardized protocol, WebRTC[6]. Based on our implementation we emulated the social behaviour using real-world data sets of Facebook [15], Slashdot [16] and Google Plus [16]. To prove the scalability of our proposed system, we also conducted simulations on a large-scale data set with millions of users collected by Twitter [16]. We show experimentally that this social graph exploitation reduces the number of hops required for dissemination over 64% and the number of relay nodes over 89% against state-of-the-art approaches. Moreover, SELECT maintains 100% communication availability by establishing connections on peers that present better online behavior than other peers. Finally, the peers in the proposed overlay network converge to a stable state in 75% fewer iterations than the state-of-the-art approaches.

The remainder of this paper is structured as follows. In Section II we provide a comprehensive analysis of the background in the structured P2P topology construction protocol, the pub/sub mechanisms and the formulation of the problem. The design of SELECT, our proposed distributed pub/sub notification system, is presented in Section III. We present an extensive evaluation of our proposed approach against the state-of-the-art approaches in Section IV and discuss the results in Section V. In Section VI we conduct a review of the related work in the course of topic-based pub/sub services. The last section concludes our work.

## II. Background and Problem Statement

In this section, we provide an analysis of P2P networks and overlay construction, relating to the node relay minimization problem and dissemination of information.

### A. Peer-to-Peer Networks

A P2P overlay network consists of a set of $N$ peers $\mathcal{P}(|\mathcal{P}|=N)$. The identifiers of the peers are assigned from an ID space $\mathcal{I}$, on the unit interval $\mathcal{I}\in[0...1)$, using a uniform mapping function (SHA-1). There exists a distance function $d_{\mathcal{I}}(u,v)$ which indicates the distance between peer $u\in\mathcal{P}$ and peer $v\in\mathcal{P}$ in the ID space $\mathcal{I}$. Each peer $p\in\mathcal{P}$ maintains *short-range* links $\mathcal{R}_p^s\subset\mathcal{P}$, that are peers with the minimum distance $d_{\mathcal{I}}(u,v)$ in the ID space $\mathcal{I}$, and *long-range* links $\mathcal{R}_p^l\subset\mathcal{P}$ have been established with a probability inversely proportional to the distance between the peers. The *short-range* links $\mathcal{R}_p^s$ and *long-range* links $\mathcal{R}_p^l$ of each peer $p$ form its routing table $\mathcal{R}_p=\mathcal{R}_p^s+\mathcal{R}_p^l$, where $|\mathcal{R}_p|\ll N$ (usually $|\mathcal{R}_p|=logN$); this is the case in latest P2P models [17] where the optimization stands in minimizing connections instead of establishing new connections to peers.

The lookup query from a peer $p$ to a peer $u$ is routed in a greedy fashion, i.e. peer $p$ selects the neighbor $w\in\mathcal{R}_p$, that minimizes the distance $d_{\mathcal{I}(w,u)}$ in the ID space $\mathcal{I}$, to forward the query. The lookup process forms an $h$-hop path $p\rightarrow^+u$ with $h=|p\rightarrow^+u|\geq 1$. Based on the selection process of the links $\mathcal{R}_p$, P2P overlay networks can guarantee that the $h$-hop path is bounded in $O(logN)$.

Moreover, peers are heterogeneous in terms of their connectivity characteristics. Different peers present different bandwidth capabilities reflecting in different latency $l$ between

[4]https://en.wikipedia.org/wiki/Moving_average

[5]https://github.com/stefanosantaris/SelectDemo

[6]https://webrtc.org/

peers, and affecting the rate at which a peer can send or receive packets. Therefore, the propagation of messages from peer $p$ to peer $u$ can be given by $l(p,u)=\sum_{i=1}^{h} l_i$.

### B. Publish/Subscribe for OSNs

A pub/sub system for OSNs consists of four basic entities: i) a social graph $G=(\mathcal{V},\mathcal{E})$, where $\mathcal{V}$ is the set of social users and $\mathcal{E}$ is the set of social connections; ii) a *Publisher* set $\mathcal{B} \subseteq \mathcal{V}$ of social users that produces the data; iii) a *Subscriber* set $\mathcal{S} \subseteq \mathcal{V}$ that comprises the publishers' social friends and consumes the data; and iv) an interest function $f : \mathcal{S} \times \mathcal{B} \rightarrow \{true, false\}$. A subscriber $s \in \mathcal{S}$ expresses his interest only on the messages that a user $b \in \mathcal{B}$ produces if $f(b,s) = true \wedge (b,s) \in \mathcal{E}$. When a publisher $b \in \mathcal{B}$ posts a new message, this message needs to be delivered *only* to the interested subscribers $\mathcal{S}_b = \{s \in \mathcal{S} | f(s,b) = true\}$.

A routing tree $RT_b$ is constructed in order to disseminate the message to all the subscribers $\mathcal{S}_b$. Thus, the edges of the routing tree connect social friends' peers that receive and forward the message until all subscribers receive it. However, an edge in the routing tree $RT_b$ does not necessarily correspond to an existing connection in the P2P overlay network. Therefore, the dissemination path latency from a publisher $b$ to all his subscribers $\mathcal{S}_b$ is calculated as follows:

$$l(b,\mathcal{S}_b) = \max_{s \in \mathcal{S}_b} l(b,s) \quad (1)$$

Furthermore, the relay nodes $r \in RT_b$ are viewed as the edges of the routing tree $RT_b$ in the path between a publisher and its subscribers. Therefore, depending on the social graph, relay nodes can be subscribers themselves.

### C. Problem Definition

The focus of our work is to minimize the number of relay nodes used to disseminate messages on pub/sub systems, as well as to reduce the dissemination latency, even under node churn.

Although the routing tree $RT$ guarantees the dissemination of messages to all the social friends, it suffers from high number of relay nodes. This happens because social users that are connected in the routing tree $RT$ are not necessarily directly connected in the P2P overlay network. Hence, each edge in the routing tree consists of $O(logN)$ relay nodes. Replacing the connections of the routing table $\mathcal{R}_p$ with the subscribers $s \in \mathcal{S}_p$ does not provide guarantees that the $h$-hop path would be bounded in $O(logN)$, while also not guaranteeing communication between any two arbitrary peers in the P2P overlay network. We need to ensure that the propagated messages will be reached by all the social user's friends regardless of the structure of the social network.

Therefore, a P2P substrate that minimizes the number of relay nodes in the routing tree $RT$ is required. We define the problem of relay nodes minimization as follows:

*Given a publisher $b$ and a set of subscribers $S_b$, each peer $p \in \mathcal{P}$ aims to establish links in its routing table $\mathcal{R}_p$ such that the routing tree among the publisher $b$ and the subscriber $s \in \mathcal{S}_b$ contains the minimum number of relay nodes $\mathcal{S}_b = \{s \in \mathcal{S} | f(s,b) = false\}$*, granting a near theoretical optima minimal solution.

TABLE I
A PEER'S $p$ LOCAL STATE, LISTING OF LOCAL VARIABLES FOR A GIVEN PEER.

| | |
|---|---|
| $D_p$ | the peer's $p$ identifier |
| $\mathcal{R}_p$ | a set of peers' identifiers that the peer $p$ is connected |
| $\mathcal{C}_p$ | a set of identifiers of the peers that host the peer's $p$ social friends |
| $\mathcal{L}_p$ | a set of connections that the peer $v \in \mathcal{R}_p$ maintains |

## III. THE SELECT SYSTEM

SELECT aims to construct a global P2P overlay network that establishes connections between peers that host social friends. Moreover, SELECT seeks to organize socially-connected peers in close distance in the overlay network, in order to reduce the number of hops required for the routing process. The intuition behind this is to provide a P2P substrate that reduces the number of hops between two socially-connected peers as well as to maintain the minimum number of relay nodes of the routing tree $RT_b$ for each publisher $b \in \mathcal{B}$. Finally, the goal of SELECT is to provide a pub/sub service that has a low latency impact.

### A. System Model

Our system model consists of a set of peers $\mathcal{P}$ and a set of social users $\mathcal{V}$. Social users join the social network either by invitation or they subscribe independently. Each social user $u \in \mathcal{V}$ is mapped onto only one peer $p \in \mathcal{P}$. We assume that peers communicate with each other over reliable channels (e.g. TCP connections) that bound the number of connections each peer maintains.

Each peer maintains a set of four local variables listed in Table I. The first variable, $D_p$ is the identifier of the peer $p$ and defines the position of the peer $p$ in the ID space $\mathcal{I} \in [0...1)$. SELECT seeks to organize the socially-connected peers in close distance in the overlay network. Thus, each peer $p$ modifies its identifier $D_p$ in order to minimize the distance $d_{\mathcal{I}(p,u)}$ to its most important peer $u$. We measure the importance between two peers and use it as a distance factor between social users, the strength of ties between two users in the social graph, by using the number of common friends that the two nodes share in the social graph. Therefore, we define the social strength between two peers $p$ and $u$ as follows:

$$s(p,u) = \frac{|\mathcal{C}_p \cap \mathcal{C}_u|}{\mathcal{C}_p} where p,u \in \mathcal{V} \quad (2)$$

The second variable, set $\mathcal{R}_p$, constitutes the routing table of the peer $p$. The third variable, set $\mathcal{C}_p$, comprises the identifiers of the peers that host its friends in the social graph. The number of connections that each peer establishes is usually lower than the number of friends that each social user maintains in the social network. This is due to the fact that most of the social friends peers have either equal connections to the same friends or a lower number of friends. Thus, in most cases, $|\mathcal{R}_p| \ll |\mathcal{C}_p|$, given that in social networks the number of friends is much higher than the connections that are required. Also, the set $\mathcal{C}_p$ contains only the identifiers of the peers which enhance the lookup process without establishing direct connections to all the peers of the set $\mathcal{C}_p$. The main goal of SELECT is to establish connections with the maximum number of each social users' neighborhood, while minimizing communication with the rest of the social friends by maintaining a minimum number of hops.

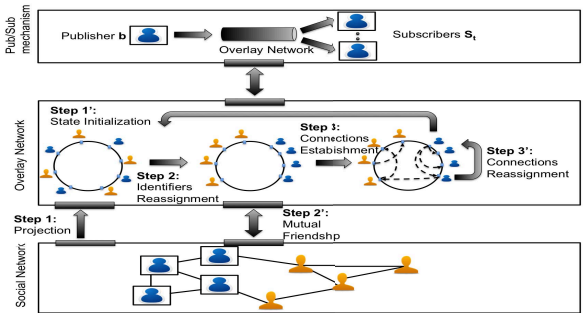


Fig. 1. The three-layer architecture of SELECT.

The fourth variable, set $\mathcal{L}_p$, contains the identifiers of the peers that each peer $u \in \mathcal{R}_p$ maintains. The existence of this variable is similar to the *lookahead* process of Symphony [10]. This lookahead set enhances the routing process as it forwards the message to the neighbor that affirms the connection with the targeted peer.

In our model, each peer establishes its own connections to other peers, according to the social networks' friendship mapping. In overall, all peers will lay in a ring topology shared to all peers in order to gain routing performance across the whole network, by minimizing the relay nodes and establishing social routing instead of plain network routing.

### B. SELECT System Overview

We implement SELECT using a three-layer architecture, as shown in Figure 1, where the bottom layer provides the social network; the middle layer provides a topology construction mechanism, based on each peer's position and its social neighbourhood, that creates the global overlay network; and the top layer refers to the topology construction protocol where pub/sub mechanism is executed.

The SELECT system consists of three main processes:

- ***Projection***: SELECT associates the position of each peer that hosts a social user in the overlay network (**Step 1** in Figure 1). The position of each peer is used to define the distance between two socially connected peers in the ID space $\mathcal{I}$. When a peer joins the overlay network, the local variables of Table I are initialized (**Step 1'**). (cf. Section III-C)
- ***Identifiers Reassignment***: SELECT evaluates the peers position in the ID space and reassigns the identifiers on a round-based basis. Specifically, each peer leverages the social neighborhood information and modifies its identifier in order to reduce its distance in the overlay network with its social friends (**Step 2**). (cf. Section III-C)
- ***Peer Connections Establishment and Reassignment***: While SELECT organizes the socially connected peers in the same area in the ID space $\mathcal{I}$, each peer establishes direct connections to peers that are also connected in the social network (**Step 3**). A link reassignment process is performed (**Step 3'**) to ensure that each social user communicates with the maximum number of his social friends in the minimum required hops, as well as with the minimum dissemination latency. (cf. Section III-D)

Both peers' identifier and connection reassignment processes use a gossip-based peer-sampling methodology to evaluate the topology defined. When the overlay network is constructed, SELECT applies the pub/sub mechanism to construct the routing tree $RT_b$ for the publisher $b \in \mathcal{B}$.

### C. Projection and Identifier Reassignment

The projection process (**Step 1** in Figure 1) determines the peer's initial position that is perceived by the underlying overlay network. Since social users join the social network either by invitation or by subscription, this directly impacts the peer's initial position (**Step 1'** in Figure 1).

As shown in Algorithm 1, the projection of the social user in the overlay network is specified based on the subscription type in the overlay network (line 1). When a social user is subscribed by invitation, his assigned identifier $D_p$ reduces the distance (line 3) between the peer $u$ and the peer $p$ that hosts the invited social user. Otherwise, a random identifier is assigned to the peer $p$ using a uniform hash function (line 5).

As the social network grows, social users create new friendships and the social strength in Equation 2 between two users is modified. SELECT strives to reduce the distance in the ID space $\mathcal{I}$ between social friends. In particular, each peer modifies its identifier in order to minimize the distance in the overlay network, to be near the peer's ID which hosts the social friend that has the highest social strength (**Step 2** in Figure 1).

The new position choice is the centroid of all its social friends position. However, this does not work in social users with high degree, in which the social strength between friends may significantly differ. Thus, social friends can be located in a totally different position in the ID space $\mathcal{I}$. To address this, we use the centroid between the two social friends that maintain the highest social strength value, as presented in Algorithm 2.

The social strength of each user is calculated using a gossip-based peer-sampling protocol, as shown in Algorithm 3 and Algorithm 4. Every peer $p$ periodically (e.g. every 10 seconds) selects a random social friend $u$ and sends its social neighborhood set $\mathcal{C}_p$ (line 3 in Algorithm 3). The peer $u$ compares the received neighborhood set $\mathcal{C}_p$ with its neighborhood set $\mathcal{C}_u$ (line 4 in Algorithm 4) and returns the number of mutual friends to the peer $p$ (line 6 in Algorithm 4).

**Complexity Analysis :** For each peer $p \in \mathcal{P}$, the initial position in the overlay network is calculated in $O(1)$, since the identifier is assigned either uniformly or based on the invited peer's identifier. Thus, the initial projection of the social graph in the P2P topology requires $O(N)$ complexity. The reassignment of the peers' identifiers based on the peer-sampling protocol requires a $O(|\mathcal{C}_p|)$ complexity for each peer, where $|\mathcal{C}_p| \ll N$. In modern social networks usually $|\mathcal{C}_p|$ is on the range of hundreds of social friends, while the size of the network $N$ is billions of users [16]. The total complexity of the Projection and Identifier Reassignment algorithm is

$$O(N \cdot |\mathcal{C}_p|) \quad (3)$$

### D. Peer Connections Establishment and Reassignment

SELECT utilizes a gossip-based peer-sampling service to construct the topology in the overlay network. Each peer periodically acquires its social neighbor's connections in

**Algorithm 1:** Peer Identifier Assignment

**Input:** $v \in \mathcal{V}$ newly registered social user
**Output:** $D_p$ peer's identifier
1 **if** $\mathcal{C}_v \neq \emptyset$ **then**
2 | u ← the peer of the social user that invited $v$;
3 | $D_p \leftarrow \min_D d_{\mathcal{I}}(u,v)$
4 **else**
5 | $D_p \leftarrow$ uniformHash($v$);
6 **end**
7 **return** $D_p$;

**Algorithm 2:** Peer Identifier Reassignment

1 **Procedure** *evaluatePosition()*
2 $u \leftarrow$ peer with the highest social strength in $\mathcal{C}_p$;
3 $v \leftarrow$ peer with the second highest social strength in $\mathcal{C}_p$;
4 $D_p \leftarrow \frac{|d_{\mathcal{I}(u,v)}|}{2}$;
5 **end procedure**

**Algorithm 3:** Peer-sampling - Active Thread

1 **Procedure** *ExchangeRT*
2 socialFriend ← getRandomSocialFriendPeer();
3 **Send** $<\mathcal{C}_p,\mathcal{R}_p>$ **to** socialFriend;
4 **Receive** $<nMutual,M>$ **from** socialFriend;
5 $socialFriend$.nMutual = nMutual;
6 $socialFriend$.M = M;
7 $D_p \leftarrow$ evaluatePosition();
8 $\mathcal{R}_p \leftarrow$ createLinks();
9 **end procedure**

**Algorithm 4:** Peer-sampling - Passive Thread

1 **Procedure** *ResponseExchangeRT*
2 **Receive** $<\mathcal{C}_u,\mathcal{R}_u>$ **from** socialFriend;
3 nMutual ← $|\mathcal{C}_u$.merge($\mathcal{C}_p)|$;
4 $socialFriend$.nMutual ← nMutual;
5 $M \leftarrow \mathcal{C}_u$.constructFriendshipBitmap($\mathcal{R}_p$);
6 **Send** $<nMutual,M>$ **to** socialFriend;
7 $M' \leftarrow \mathcal{C}_p$.constructFriendshipBitmap($\mathcal{R}_u$);
8 $socialFriend$.bitMap = $M'$;
9 $D_p \leftarrow$ evaluatePosition();
10 $\mathcal{R}_p \leftarrow$ createLinks();
11 **end procedure**

the overlay network and evaluate its current established connections (**Step 3** in Figure 1). Specifically, each peer $p$ seeks to establish direct connections with the maximum number of its social neighborhood $\mathcal{C}_p$.

Each peer is allowed to accept only $K$ incoming links, while maintaining two *short range* outgoing links $\mathcal{R}_p^s$ with his successor and predecessor in the overlay network in order to create a ring topology and $K$ *long range* outgoing links $\mathcal{R}_p^l$ with its social friends. The intuition behind the $K$ incoming links is to avoid having peers that have too many connections, because other peers seek to connect to them, and consequently present more traffic than others. When the $K$ incoming links are established, the peer accepts a new incoming connection if the new connection has better bandwidth capability than the already existing connections. The $K$ outgoing *long range* links are selected by applying the Locality Sensitive Hashing (LSH) technique to the social neighbor's connections retrieved from the peer-sampling service (lines 3-6 and 2-8 in Algorithms 3 and 4, respectively). The LSH technique is used to choose the long range links from different zones of the overlay and avoid link overlap in the overlay network. We consider that the LSH family technique to be reliable in maintaining long range connectivity for the overlay network.

The connection establishment mechanism is shown in Algorithm 5. We begin by indexing the bitmaps of the social neighborhood in $\mathcal{H}$ buckets in the LSH index (lines 2 - 4). In our algorithm, we consider that the number of buckets is equal to the number of *long range* links defined ($|\mathcal{H}| = K$). The reason for selecting $|\mathcal{H}| = K$ buckets in the LSH index is to simplify the selection process of the direct connections. Peers, whose connections are similar, will be indexed in the same bucket. This results in selecting only one peer in each bucket, establishing at most $K$ long range links.

The bitmap of $u \in \mathcal{C}_p$ is an array of size $|\mathcal{C}_p|$, the values of which define the link existence in $\mathcal{R}_u$ between two socially connected peers $u \in \mathcal{C}_p$ and $v \in \mathcal{C}_p$, where $u \neq v$, as follows:

$$bitmap(u,v) = \begin{cases} 1 & \text{if } (u,v) \in \mathcal{R}_u \\ 0 & \text{if } (u,v) \notin \mathcal{R}_u \end{cases}$$

While the bitmaps are indexed in the $\mathcal{H}$ buckets of LSH, in lines 5 - 18, we aim to select one peer of each bucket $h \in \mathcal{H}$ to establish connection. However, not all buckets may contain only one peer, as social friends tend to converge to similar connections. In order to establish connections with the maximum number of the peer's $p$ social neighborhood $\mathcal{C}_p$, but also the minimum dissemination latency, we select the peer that attempts to establish a connection using a picker (line 8), as shown in Algorithm 6. In doing so, we select the peer that achieves the maximum number of social connections and presents better bandwidth capability in order to propagate the message with higher rate. Moreover, we drop an already established connection $(p,u) \in \mathcal{R}_p$ with a peer $u$ that presents similar connections with newly established connection $(p,v) \in \mathcal{R}_p$ (lines 12-16) in Algorithm 5.

**Algorithm 5:** Peer Links Reassignment

1 **Procedure** *createLinks()*
2 **for** $u \in C_p$ **do**
3 | LSHIndex($u.bitMap$);
4 **end**
5 **for** $h \in \mathcal{H}$ **do**
6 | $\mathcal{P}_h \leftarrow$ peers assigned in the same bucket $h$;
7 | **if** $\mathcal{P}_h \neq \emptyset$ **then**
8 | | $u \leftarrow$ picker($\mathcal{P}_h$);
9 | | **if** $(p,u) \notin \mathcal{R}_p$ **then**
10 | | | $\mathcal{R}_p \leftarrow (p,u)$
11 | | **end**
12 | | **for** $v \in \mathcal{P}_h, v \neq u$ **do**
13 | | | **if** $(v,p) \in \mathcal{R}_p$ **then**
14 | | | | $\mathcal{R}_p$.remove(v);
15 | | | **end**
16 | | **end**
17 | **end**
18 **end**
19 **end procedure**

**Complexity Analysis :** The complexity analysis of the *Connections Establishment and Reassignment* algorithm is analogous to the number of social friends $|\mathcal{C}_p|$ that each user maintains and the number of buckets $|\mathcal{H}|$ assigned on the LSH index.

**Algorithm 6:** Picker Peer Connection

```
Procedure picker(P_h)
PS_h ← sortPeers(P_h);
if (|PS_h| > 0) && (PS_h(0).bw < PS_h(1).bw) then
    RETURN PS_h(1)
end
RETURN PS_h(0)
end procedure
```

The peer-sampling protocol aggregates the bitmaps in $O(|\mathcal{C}_p|)$ complexity. The index of the bitmaps in $|\mathcal{H}| = K$ buckets requires $O(|\mathcal{C}_p| \cdot log(|\mathcal{C}_p|) \cdot K)$ complexity. The selection of the $K$ *long range* links using the LSH index is performed in $O(K)$ cost. Summarizing, the total complexity of the *Connections Establishment and Reassignment* algorithm for each peer $p \in \mathcal{P}$ is

$$O(|\mathcal{C}_p|^2 \cdot log(|\mathcal{C}_p|) \cdot K^2) \quad (4)$$

### E. Pub/Sub system

The pub/sub system utilizes the generated overlay network to create the routing tree $RT_b$ for each social user $b \in \mathcal{B}$ and guarantees the delivery of the published messages to all of his social friends $\mathcal{S}_b$.

Following the lookahead technique of [10], each peer $p$ maintains a lookahead set $\mathcal{L}_p$ of connections that each peer $u \in \mathcal{C}_p$ maintains. Peer $p$ uses the lookahead set $\mathcal{L}_p$ to create the routing tree $RT_b$ and forward the message during the routing process. Each peer monitors its routing table $RT_b$ and the lookahead set $\mathcal{L}_p$ and forwards the message to the peer that guarantees the delivery of the message within 1 or 2 hops. If the peer $u \in \mathcal{S}_b$ is not included in the routing table $\mathcal{R}_p$ and the lookahead set $\mathcal{L}_p$, the peer $v \in \mathcal{R}_p$ that minimizes the distance $d_{\mathcal{I}}(v,u)$ is selected.

### F. Recovery Mechanism

Peers join and depart the overlay network at unexpected rate (churn). Also, to maintain the pub/sub reliability property for message delivery, we need to manage a routing table that efficiently recovers from peers departure. In doing so, peers periodically request each social friend of their routing table for their state. The availability of each peer is recorded and their online behavior is calculated using the Cumulative Moving Average (CMA). The intuition of using CMA is to identify the average online behavior of a social user during the period of the last few days and ensure that the social user is a good candidate for establishing a connection. Thus, the peer identifies if a connection is unresponsive because a social user is mostly offline or if it is a temporal connection failure. In doing so, a peer $p$ decides to keep an unresponsive connection to the peer $u$ in order not to create a chain of connections reassignment to the peers that are connected to the peer $p$. In contrast, when a peer is unresponsive and its CMA value is low, we replace the unresponsive peer with another peer from the same bucket of the LSH index (see Section III-D). Using this approach, SELECT maintains direct connections with peers that host social friends and publish a message on non-relay nodes while also being adaptive to dynamic environments.

## IV. Evaluation

For our evaluation, we considered two types of experiments, one as a simulation and another as a realistic environment. For the simulation experiments, we used the Gelly Graph API[7] which runs over the Apache Flink[8] distributed data processing framework. We ran our experiments on a Flink cluster with 20 nodes in order to provide a distributed discrete event simulator suitable to conduct large-scale experiments with millions of peers. For the realistic experiments, we used WebRTC to create the peers as browser-dependent and deployed on a cloud infrastructure several VMs (in total 18 VMs are used). The VMs contained several peers spread among each, hosting all the users in each data set. The communication between peers was done through the network interface, which allowed to emulate the latency between nodes and achieve a realistic environment.

TABLE II
FOUR REAL-WORLD DATA SETS OF SOCIAL NETWORKS THAT INCLUDE USERS INFORMATION, SUCH AS SOCIAL CONNECTIONS AND AVERAGE DEGREE.

| Data Set | Users | Connections | Average Degree |
|---|---|---|---|
| Facebook | 63,731 | 817,090 | 25.642 |
| Twitter | 3,990,418 | 294,865,207 | 73.89 |
| Slashdot | 82,168 | 948,463 | 11.543 |
| GooglePlus | 107,614 | 13,673,453 | 127 |

The implementation of SELECT is performed using the vertex-centric iterative model [18]. Specifically, in synchronized iteration steps, each peer produces messages to other peers and updates their identifiers and their connections in the overlay network using the SELECT algorithms.

Experiments are performed in evolving networks, where users join the overlay network at different phases. We initiate our experiments by selecting a social user $u \in \mathcal{V}$ from the data set at random. Thereafter, we insert into the social network a portion of the user $u$'s social friends, following the model of [19]. Based on [19], social users establish friendship connections at high rate in the beginning of the join process, and this rate decreases exponentially over time. Therefore, at each iteration step, we select a registered social user and insert into the social graph a number of her social friends that preserves the exponentially decreasing rate of the model.

Additionally, we introduced the churn rate of each peer in the overlay network, following the model of [20]. Specifically, at each iteration step, we select a number of peers based on a log-normal distribution to be excluded from the overlay network. When the iteration step is completed, the removed peers are recovered in the overlay network.

When the overlay network is constructed, we perform simulations of the pub/sub mechanism to measure the number of relay nodes that exist on each routing tree. In order to realistically simulate a real-time notification system in the social network, each publisher posts messages at exponential rate following the model of [21].

The realistic experiments follow the same pattern as the simulation experiments, however since each node has different bandwidth capabilities, different latency is applied for each node and accounted in the analysis. Also, in the pub/sub system, packets of 1.2MB are sent from the publishers to the subscribers.

[7]https://ci.apache.org/projects/flink/flink-docs-master/dev/libs/gelly/index.html

[8]http://flink.apache.org/

### A. Data sets

Our evaluation is performed with four real-world data sets, listed in Table II. These data sets cover a wide range of social graph features, from less connected graphs (Slashdot [16], Facebook [15]) to highly connected graphs (Twitter, Google Plus [16]), that enhance the evaluation of our proposed approach on several graph types. Moreover, we conduct experiments on the large-scale data set of Twitter in order to demonstrate the scalability of our algorithm. The characteristics of the data sets are presented in Table II.

### B. Metrics

In order to measure the efficiency of SELECT, we use the following metrics:

- **Number of Hops:** The average number of overlay hops within the path between two peers.
- **Number of relay nodes:** The average number of relay nodes that exists in the pub/sub routing tree.
- **Number of iterations:** The average number of iterations required to organize the peers in the overlay network.
- **Percentage of messages:** The percentage of messages that each peer forwards in the dissemination tree.
- **Latency:** The average latency of communication between peers in the overlay network, counting the latency between intermediate peers in a given path. Only used for the realistic experiments, since simulations do not account with latency.

To validate our analysis, for each metric we report the average result out of 100 independent trials to decrease the risk of statistical error. We consider these metrics to be important to understand the behaviour of SELECT and the pub/sub system. Thus, be able to compare the end results with other works while also giving feedback on the use of SELECT for the domain of pub/sub systems.

### C. Simulation Experiments

We compared SELECT with several existing pub/sub systems of different categories: i) a pub/sub system over the *Symphony* P2P overlay network without any further modification on the P2P topology; ii) *Bayeux*, a pub/sub system that organizes peers into a DHT in a P2P overlay and builds a spanning tree for each topic to propagate the messages; iii) *Vitis*, a gossip-based pub/sub system that organizes the subscribers into clusters; and iv) *OMen*, that constructs TCOs to disseminate information on each topic.

As the number of direct connections increases, we observe a substantial reduction, over 90%, on the average number of hops required for the communication between two socially-connected peers. However, as the number of links used overcomes the logarithmic number of peers in the overlay network, no further improvement is performed. Based on the above observation, for the rest of the experiments, we assign $log_2 N$ direct connections on each peer in order to construct a P2P topology.

Figure 2 presents the average number of hops required for a publisher to propagate information to each one of his subscribers. As the network grows, the average number of hops increases logarithmically. However, SELECT performs with 76%, 83%, 75% and 85% fewer hops compared to the

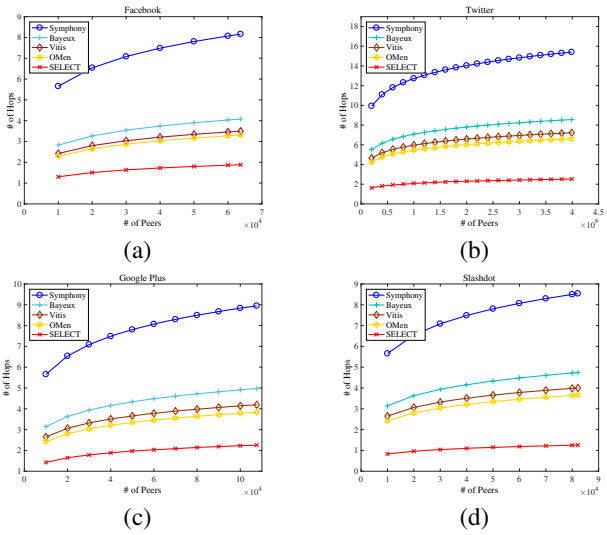


Fig. 2. Number of hops per social lookup for the (a) Facebook, (b) Twitter, (c) Google Plus and (d) Slashdot data sets.

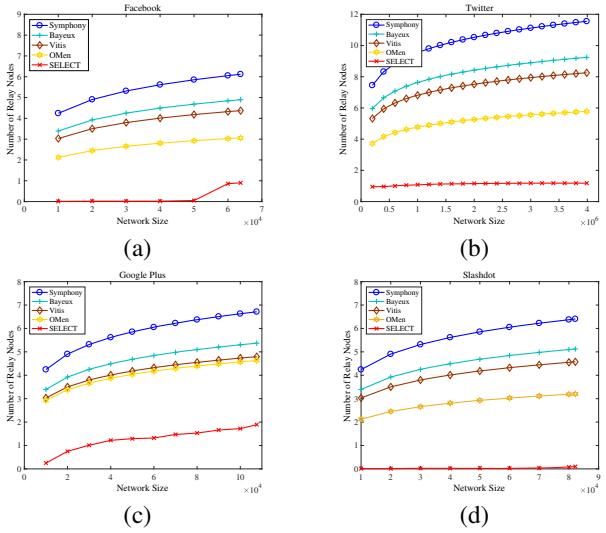


Fig. 3. Number of relay nodes per pub/sub routing path for the (a) Facebook, (b) Twitter, (c) Google Plus and (d) Slashdot data sets.

pub/sub mechanism built over the Symphony overlay network and for the Facebook, Twitter, Google Plus and Slashdot data sets, respectively. This occurs due to the fact that Symphony's construction of long range links is completely oblivious to the social graph and the publication workload. In contrast, SELECT establishes connections between socially-connected peers, and as such subscribers are 1 or 2 hops away from the publisher. Compared to the state-of-the-art pub/sub approaches, SELECT achieves more than 43%, 61%, 41% and 65% reduction for the Facebook, Twitter, Google Plus and Slashdot data sets, respectively. This happens because peer identifiers on SELECT are mutable and socially-connected peers are clustered in the same region in the ID space. Hence, a small-world network is accomplished on SELECT, in contrast to the presented approaches where an immutable identifier policy is applied.

Figure 3 presents the impact of SELECT on the number

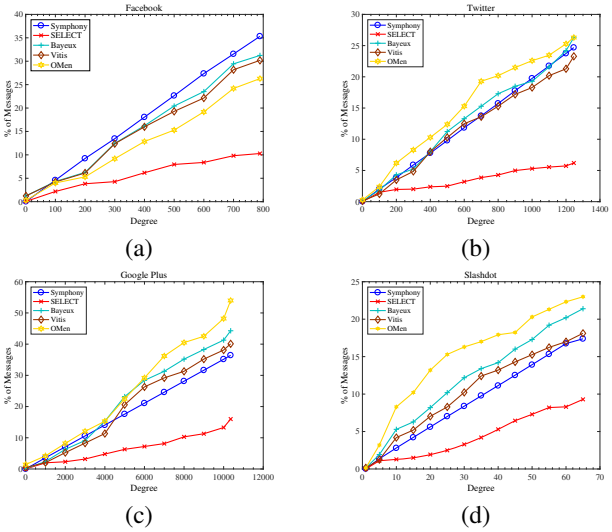


Fig. 4. Messages forwarded per social degree in a pub/sub routing tree for the (a) Facebook, (b) Twitter, (c) Google Plus and (d) Slashdot data sets.

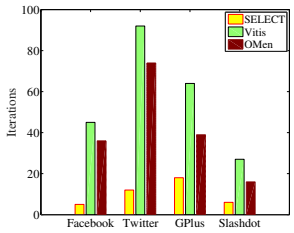


Fig. 5. Number of iterations required to construct the overlay. Symphony and Bayeux approaches are excluded as they provide no iterative connection establishment process.

of relay nodes that exist in the routing path between publisher and subscriber. SELECT presents over 98% reduction on the number of relay nodes for all data sets, in comparison to the Symphony, Bayeux, Vitis and OMen approaches. This happens because in Symphony, Bayeux, Vitis and OMen the probability of two socially-connected users to be also connected in the overlay network is extremely low. In contrast, SELECT leverages the social graph and establishes connections between socially-connected peers that reduces the number of relay nodes in the routing path between publisher and subscriber.

In Figure 4, we investigate the balance of the load that each peer presents, by measuring the percentage of messages that each peer forwards in the routing tree against the degree of the peer. Figure 4 indicates that SELECT provides better load balancing than Symphony, Bayeux, Vitis and OMen approaches. This happens because Symphony and Bayeux are agnostic to the social network dynamics, and thus information propagation converges to the peers that present high social degree. In contrast, Vitis and OMen leverage the social network dynamics but the peer connection strategy that they follow emphasize on connecting peers with high social degree. SELECT presents more than 60%, 73%, 56% and 46% improvement against Symphony, Bayeux, Vitis and OMen approaches for the Facebook, Twitter, Google Plus and Slashdot data sets, respectively.

The total number of iterations required to establish the connections between peers, are presented in Figure 5. Symphony and Bayeux are excluded from this set of experiments as they provide no iterative algorithms. Based on the reporting results in Figure 5 we observe that SELECT converges in significantly lower number of iterations than Vitis and OMen. This observation is due to the fact that Vitis and OMen initially organize the peers following a standard DHT-based overlay network and optimise the connections when the overlay network is formed. Thus, connections are established between non socially-connected peers and the gossip algorithm applied requires more iterations in order to identify the socially-connected peers. In contrast, SELECT establishes immediately the connections between peers that are socially-connected and thereafter optimises the connections in order to improve the information propagation. This results in a lower number of iterations to organize the peers since most of the peers' connections are already to a socially-connected peer.


Fig. 6. The impact of churn in the data availability during the information propagation. Dash line represents the node churn and continuous line the availability.

Finally, in Figure 6, we present the impact of the unexpected join and leave of peers in the relay nodes between two socially-connected peers. In this set of experiments we ran a simulation for over ten hours, where in each second a random number of peers depart or join the network. The total number of peers that are available in the P2P overlay network cannot be less than half of the overall social network. Based on this experiment, we observe that each peer efficiently replaces the unresponsive connection with another peer that presents similar connections based on the LSH index. Thus, the routing process maintains 100% data availability in all data sets.

### *D. Realistic Experiments*

In the realistic experiments, we perform the comparison with other pub/sub systems, as Symphony, Bayeux, Vitis and Omen, as described in the previous experiments.

Towards understanding the behaviour of our algorithm when latency is applied, we start by introducing an initial experiment on simultaneous connectivity. The peers join a network and connect to a central peer, without applying any selection algorithm. Thus, the central peer is connected to all others. Afterwards, the central peer creates a data fragment of 1.2MB (average image size) and sends to all its connections simultaneously. In our findings when increasing the number of connections there is a linear increase in the total time for transfers. Therefore, we can establish that an issue is not the number of connections to be established, but the simultaneous transfers to peers.

Figure 7 presents the latency for message dissemination between the publishers and their subscribers. At first, without

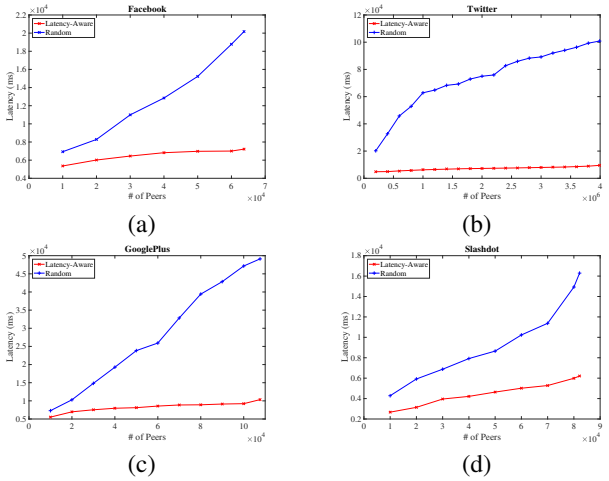


Fig. 7. Average latency of data dissemination in the pub/sub routing tree for the (a) Facebook, (b) Twitter, (c) Google Plus and (d) Slashdot data sets.

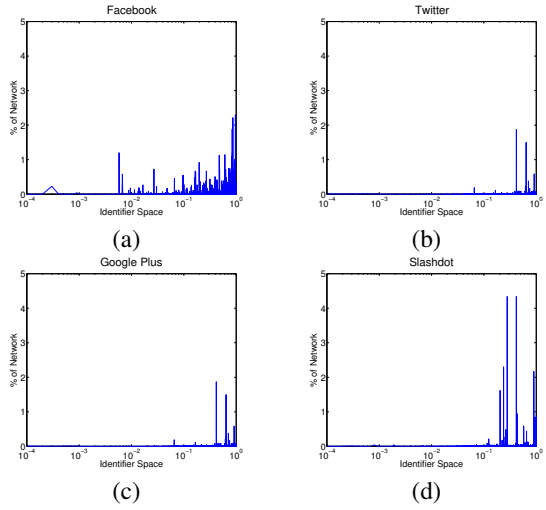


Fig. 8. Identifiers distribution among the network for the (a) Facebook, (b) Twitter, (c) Google Plus and (d) Slashdot data sets.

selection algorithm (random), for each of the data sets we find that the peers connectivity can grow exponentially making the dissemination process costly in terms of timing. When applying SELECT, the overlay becomes latency aware and therefore the dissemination latency has a small *linear growth* accommodating more peers in the overlay without sacrificing dissemination time.

Figure 8 presents the distribution of the identifiers after applying SELECT, for each of the data sets. We determine that SELECT rearranges the overlay in such a way that the nodes distances are maintained as low as possible while still being able to reach all of the network. In fact, we can observe that small groups of nodes are within the same regions, which aggregate the socially-connected nodes without losing connectivity between regions.

## V. Discussion

Our approach to disseminate data in pub/sub systems relies on the social network connections. Due to the fact that state-of-the-art approaches rely on different aspects of the network for their optimization, it is hard to provide a fair cost comparison between them and the SELECT algorithm. We clearly see from our experimental results that the actual costs come from the added social information which is necessary to create the friendship graph in order to augment the global overlay.

We assure the correctness of our approach by grounding it in a ring topology, since it gives us the ability to continue sending messages to all peers and guarantee that all nodes are able to receive them. Other topologies, such as mesh, tend to create isolated communities of nodes. Therefore, the use of other topologies may not guarantee the same results when applying different social networks.

We also show that the issue of simultaneous data transfers may degrade the performance of a peer when disseminating concurrent messages. This issue can be optimized by having more than one paths to the subscribers in order to guarantee the transmission; however, it is unlikely to find paths of the same length and latency stability.

Finally, we can observe that SELECT achieves its uni-dimensional network construction in real world environments very successfully and without compromising any of the required large-scale pub/sub properties. This proves that SELECT is fully applicable on OSNs in real world settings, although a geographically distribution study would augment our findings.

## VI. Related Work

A significant body of research has considered the construction of a P2P pub/sub system so that the number of relay nodes is minimized. The proposed approaches are divided in two main categories: i) the design of a routing tree, the construction of which relies on the routing process of the underlying P2P overlay network [11]; and ii) the construction of a P2P topology such that the paths in the routing tree contain the minimum number of relay nodes [1, 5, 22].

In the first category, Bayeux [11] organized peers into a DHT, where each peer maintains $O(logN)$ connections. Then a routing tree is built for each topic with a rendezvous node at the root, which delivers the events to the peers that join the tree. This approach, however, forces many nodes to relay the messages for which they have not subscribed. Consequently, Bayeux-based systems suffer from high traffic overhead as they fail to minimize the number of relay nodes.

Rahimian et al. [5, 23] proposed a gossip-based hybrid P2P overlay for pub/sub systems, called Vitis. Peers in Vitis are organized in a ring structure and run a gossip-based peer sampling algorithm to identify the subscription and establish connections so that peers that are interested on similar topics are organized in clusters. Although Vitis manages to reduce the number of relay nodes, peers with high social degree present high traffic overhead since the rest of the peers aim to connect with the social users that maintain the most social friends in common.

The theoretical formulation of pub/sub overlay design with the minimum number of relay nodes originated in [22]. Chockler et al. [22] investigated the problem of constructing a routing tree with minimum edges. They presented a Greedy Merge (GM) algorithm that achieves a logarithmic approximation ratio for the average peer degree. The GM algorithm produces

a routing tree for each topic with unbalanced peer degrees by effectively creating hotspot peers with a high node degree.

Finally, OMen [6] is one of the most recent approaches that emphasizes on the design of the P2P overlay network in order to provide a P2P pub/sub system. OMen [6] incorporated the design of a Topic connected Overlay (TCO) [24], which is an approximation of the GM algorithm [22]. Forming a P2P small world overlay network of [1], each peer in the OMen pub/sub system maintains a *shadow set*, which is a subset of backup peers that maintains the information to repair the TCO when churn occurs. Furthermore, Omen utilizes social network connectivity to generate the pub/sub workloads and propagates messages through the established overlay. Although OMen provides a fast recovery mechanism, while maintaining low number of relay nodes, no monitoring on the peers' online activity is performed, thus presenting high traffic overhead to the peers that establish connection to peers with extremely low online behavior.

Recent works such as SpiderCast [25] and PolderCast [26] extend the mentioned works however we do not compete directly with them. Furthermore, such works build upon the same frame of reference works, not fully exploiting the underlying social structure for efficient routing, which can result in heavy relay costs or different handling of the social and P2P graphs.

Our work differs in designing an overlay network that leverages the social graph topology and interactions to organize both the peers in the overlay network and their established connections. Specifically, in our work, the peers are placed in the same area in the overlay network based on their social proximity and establish a bounded and adaptive number of connections with peers that are also connected in the social graph. Thus, we avoid congestion in peers with high social degree, spreading the connections with other social users that retain better behavior. Hence, we simplify the routing tree construction and reduce the number of relay nodes.

## VII. Conclusions

In this paper, we address the problem of relay nodes in a pub/sub system for social notifications and propose SELECT - a distributed pub/sub system. We design a P2P overlay network that exploits the social graph to organize the peers in an overlay network and establish connections between socially-connected peers. Using a gossip-based peer sampling service, SELECT reduces the number of hops required to communicate two socially-connected peers. Additionally, the constructed routing trees in the pub/sub system exhibit the minimum number of relay nodes. We evaluate SELECT in simulated and realistic environments using four real-world data sets and highlight the performance of SELECT against state-of-the-art approaches. Modern social networks, such as Facebook, Twitter and Spotify, have to propagate a vast amount of notifications. Consequently, to account for the fact that such notification systems need to offload processing from their dedicated resources it is worth to consider the implementation of SELECT that reduces the number of relay nodes, while maintaining 100% communication availability.

## Acknowledgement

This work was partially supported by the iSocial EU Marie Curie ITN project (FP7-PEOPLE-2012-ITN).

## References

[1] C. Chen and Y. Tock, "Design of Routing Protocols and Overlay Topologies for Topic-based Publish/Subscribe on Small-World Networks," in *Proceedings of the Industry Track of the 16th ACM/IFIP/USENIX Middleware conference (Middleware Industry 2015)*, Dec 2015.

[2] R. Want, B. N. Schilit, and S. Jenson, "Enabling the internet of things," *IEEE Computer*, no. 1, pp. 28–35, 2015.

[3] M. Nitti, R. Girau, and L. Atzori, "Trustworthiness management in the social internet of things," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 5, pp. 1253–1266, May 2014.

[4] A. Ortiz, D. Hussein, S. Park, S. Han, and N. Crespi, "The cluster between internet of things and social networks: Review and research challenges," *IEEE Internet of Things Journal*, vol. 1, no. 3, pp. 206–215, June 2014.

[5] F. Rahimian *et al.*, "Vitis: A gossip-based hybrid overlay for internet-scale publish/subscribe enabling rendezvous routing in unstructured overlay networks," in *IEEE International Conference on Parallel Distributed Processing Symposium (IPDPS)*, 2011.

[6] C. Chen, R. Vitenberg, and H.-A. Jacobsen, "OMen: Overlay Mending for Topic-based Publish/Subscribe Systems Under Churn," in *Proceedings of the 10th ACM International Conference on Distributed and Event-Based Systems (DEBS 2016)*, June 2016.

[7] V. Setty *et al.*, "The hidden pub/sub of spotify: (industry article)," in *Proceedings of the ACM International Conference on Distributed Event-based Systems*, 2013.

[8] A. Datta, S. Buchegger, L.-H. Vu, T. Strufe, and K. Rzadca, *Decentralized Online Social Networks*, 2010, pp. 349–378.

[9] M. Jelasity, A. Montresor, and O. Babaoglu, "T-man: Gossip-based fast overlay topology construction," *Computer Networks*, vol. 53, no. 13, pp. 2321–2339, Aug. 2009.

[10] G. S. Manku, M. Bawa, and P. Raghavan, "Symphony: Distributed hashing in a small world," in *Proceedings of the Conference on USENIX Symposium on Internet Technologies and Systems*, 2003.

[11] S. Q. Zhuang *et al.*, "Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination," in *Proceedings of the 11th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, 2001.

[12] B. Hesmans and O. Bonaventure, "Tracing multipath tcp connections," in *Proceedings of the 2014 ACM Conference on SIGCOMM*, 2014.

[13] S. Girdzijauskas, "Designing peer-to-peer overlays: a small-world perspective," *EPFL thesis no. 4327, advisor: Karl Aberer*, vol. 154, 2009.

[14] A. Gionis, P. Indyk, R. Motwani *et al.*, "Similarity search in high dimensions via hashing," in *VLDB*, 1999.

[15] B. Viswanath *et al.*, "On the evolution of user interaction in facebook," in *Proceedings of the ACM Workshop on Online Social Networks*, 2009.

[16] J. Leskovec and A. Krevl, "SNAP Datasets: Stanford large network dataset collection," http://snap.stanford.edu/data, Jun. 2014.

[17] K. Dhara, Y. Guo, M. Kolberg, and X. Wu, *Overview of Structured Peer-to-Peer Overlay Algorithms*. Boston, MA: Springer US, 2010, pp. 223–256. [Online]. Available: http://dx.doi.org/10.1007/978-0-387-09751-0_9

[18] R. R. McCune, T. Weninger, and G. Madey, "Thinking like a vertex: A survey of vertex-centric frameworks for large-scale distributed graph processing," *ACM Comput. Surv.*, vol. 48, no. 2, 2015.

[19] K. Zhu, W. Li, and X. Fu, "Modeling population growth in online social networks," *Complex Adaptive Systems Modeling*, vol. 1, no. 1, 2013.

[20] A. Berta, V. Bilicki, and M. Jelasity, "Defining and understanding smartphone churn over the internet: A measurement study," in *IEEE International Conference on Peer-to-Peer Computing*, 2014.

[21] J. Jiang *et al.*, "Understanding latent interactions in online social networks," in *Proceedings of the ACM SIGCOMM Conference on Internet Measurement*, 2010.

[22] G. Chockler, R. Melamed, Y. Tock, and R. Vitenberg, "Constructing scalable overlays for pub-sub with many topics," in *Proceedings of the Annual ACM Symposium on Principles of Distributed Computing*, 2007.

[23] F. Rahimian, T. Le Nguyen Huu, and S. Girdzijauskas, *Locality-Awareness in a Peer-to-Peer Publish/Subscribe Network*, 2012.

[24] C. Chen, H.-A. Jacobsen, and R. Vitenberg, "Algorithms based on Divide and Conquer for Topic-based Publish/Subscribe Overlay Design," *ACM/IEEE Transactions on Networking*, 2014.

[25] G. V. Chockler, R. Melamed, Y. Tock, and R. Vitenberg, "Spidercast: a scalable interest-aware overlay for topic-based pub/sub communication," in *DEBS*, 2007.

[26] 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*, P. Narasimhan and P. Triantafillou, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 271–291.