Peer-to-peer technologies have proven their strength in large-scale resource sharing and data transfer. Such systems, however, still need to address a variety of issues, including efficient routing, security, quality of service, incentives and reputation. Recent research started leveraging social information to develop new and effective techniques to improve the performance of peer-to-peer systems. However, using social information is a double-edged sword, which can bring benefits as well as new challenges. This survey presents and classifies the types of social information that have been used so far in the design of peer-to-peer systems, how the social fabric has been used to facilitate transactions in the system, and some challenges caused by using social information.

CCS Concepts:
\- Security and privacy → Distributed systems security;
\- Human-centered computing → Social networking sites;
\- Empirical studies in collaborative and social computing;
\- Social and professional topics → Centralization / decentralization;
\- Computing methodologies → Distributed algorithms;
\- Applied computing → Sociology;
\- Information systems → Hierarchical storage management;

Additional Key Words and Phrases: Social networks, peer-to-peer systems, socially aware peer-to-peer systems

ACM Reference Format:
DOI: http://dx.doi.org/10.1145/2894761

1. INTRODUCTION

The peer-to-peer (P2P) design philosophy relies on individual computers’ CPU cycles, communication and storage capacity to enable access to a large pool of resources that self-organize. The P2P architecture hence has the potential to enhance reliability and fault tolerance because of its independence from dedicated servers and centralized control. For example, a P2P Voice-over-IP (VoIP) network can initiate and receive voice calls to or from another peer without relying on centralized servers that can become overloaded.

Because of this autonomy, P2P techniques were deployed in commercial applications such as cloud storage systems (Space Monkey [Monkey 2014; Ramachandran 2013] and Symform [Symform 2014]), Content Delivery Networks (e.g., PeerCDN [PeerCDN 2013]), Voice over IP (Skype [Skype 2015], etc), Video on Demand ([Huang et al. 2007]), file sharing (Kazaa [Leibowitz et al. 2003], etc) and many others. In addition, researchers continue to expand their investigations into the applicability of P2P techniques to other application areas. For instance, P2P-based massive multiplayer online
gaming systems [Yahyavi and Kemme 2013; Chan et al. 2007; Knutsson et al. 2004] are promising due to reduced deployment and maintenance costs when compared to centralized game servers. Most recently, P2P approaches have been introduced into smart phones, driving innovative user experiences in mobile. AllJoyn [AllJoyn 2015; Lioy 2011], as a typical example, provides a framework that enables proximity-based device-to-device communication that connects people in real-time without a server. This framework supports various mobile applications, from media sharing, social networking to entertainment and productivity tools.

Despite the benefits of the P2P architecture, there are several fundamental challenges that restrict adoption and deployment of P2P systems and applications. First, the highly intermittent peer participation (churn) affects resource availability and increases overlay maintenance costs [Stutzbach and Rejaie 2006]. Second, because of the scale of P2P systems and unreliability of individual peers [Daswani et al. 2003], designing efficient resource discovery mechanisms is difficult. Third, due to P2P systems' open and autonomous nature, securing the system against malicious attacks such as denial-of-service (DoS) [Dumitriu et al. 2005] and Sybil attacks [Douceur 2002] is challenging. Fourth, dynamic changing peers and skewed user query patterns may cause load imbalance—the uneven distribution of work loads to nodes—and consequently increases users' response times [Mondal et al. 2003].

To address some of the inherent challenges introduced by decentralization and self-organization, recent research looks at the benefits of exploiting knowledge about the social ties that connect the humans behind the participating nodes in a P2P system. The social ties that connect users are typically declared (such as friendship lists in Facebook, professional connections in LinkedIn, or follower-followed directed relationships in Twitter) or inferred from overlapping interests (e.g., searching for the same files, or declared membership to the same social group). The intuition that supports this research direction comes from the observation that behind compute nodes (or peers, in the P2P terminology) are typically individual users, not institutions. Moreover, through its decentralized, grass-root approach, the activity of a peer reflects directly its user's intentions rather than the policies of a centralized service provider or the behavior of a group. Social information, is posited, is useful for the performance of P2P systems. For instance, in real life people directly and selectively contact their friends when searching for particular information. Intuitively, similar strategies that would directly connect peers whose users are friends might help with resource discovery. Another example is that social communities are formed spontaneously between people. Similar strategies transferred to P2P systems can keep track of a dynamic collection of peers to function coherently, thus addressing the decentralized group management challenge. Yet another example is related to trust and security: social relationships encapsulate some form of trust built out-of-band. Transferred properly to the P2P overlay, these peer-to-peer connections are less subject to manipulation attacks or bootstrapping challenges than in socially-oblivious P2P systems.

We refer to P2P solutions that exploit the social fabrics of their users as socially aware. In such systems, peers are treated as participants in social networks; the formation of P2P overlays is informed by the users' social ties, declared or inferred. The recent availability of knowledge about users' social relationships and interactions, made prevalent by social applications and Online Social Networks (OSNs)—such as Facebook, LinkedIn, and Foursquare—provides the real-life information about the social structure of the user base and thus enables the implementation of many such solutions.

This article surveys the literature in the area of P2P solutions that employ social knowledge or social theories, and classifies the existing work based on the benefits and challenges that such solutions bring to various P2P applications. As such, we center
our discussion on the impact or pitfalls of aspects of social awareness and not on particular classes of P2P applications. Specifically, after general background on relevant social theories and social network characteristics in Section 2 and on P2P systems in Section 3, we discuss benefits such as increased cooperation due to exploiting social incentives (Section 4.1), improved search efficiency due to exploiting homophily (Section 4.2), and defenses against malicious attacks based on social trust (Section 4.3). This paper also discusses challenges introduced by relying on social knowledge as seen in the literature (Section 5). Such challenges include low service availability and imbalanced workloads in friend-only P2P systems and practical security defense policies. Understanding these challenges is key for designing robust, scalable and functional systems. To the best of our knowledge, the pitfalls of leveraging social networks have not been systematically surveyed in the context of socially aware P2P applications.

Given the body of work and the excitement created by online social network applications and services, other survey articles have been previously published in this area. The closest to this survey is [Huang et al. 2012], in which the authors present social network theories and how they apply to P2P applications. While we structure the discussion on benefits of social networks as applied to a cross section of P2P applications, Huang et al. discuss how particular classes of applications can benefit from social networks. Specifically, they discuss the benefits to P2P file downloading applications, to P2P applications based on network coding techniques, and to P2P content-centric networking or named data networking. Because we do not limit our coverage to specific classes of P2P applications, our survey implicitly describes the effect of social awareness on other classes of applications, such as storage sharing and spam avoidance. Other survey articles focus on particular classes of P2P applications and, without specially targeting them, include descriptions of socially-aware solutions. For example, Abboud et al. [Abboud et al. 2011] summarize schemes that enable resilient P2P video streaming systems. In the process, socially aware mechanisms that leverage shared interests and social interactions are discussed. Other surveys focus specifically on socially aware solutions to problems that are not restricted to P2P. For example, Yu [Yu 2011] and Boshman [Boshmaf 2012] survey the literature on social graph-based mechanisms for defending against malicious attacks in open systems.

2. SOCIAL THEORIES AND GRAPH PROPERTIES USED IN P2P SYSTEM DESIGN

There are a variety of social theories in sociology and complex networks, such as homophily [McPherson et al. 2001] and strength of weak ties [Granovetter 1973], and properties of social graphs, such as power law distribution [Newman 2010] and small-world characteristics [Watts and Strogatz 1998]. We summarize below the theories and properties most frequently used in the literature reviewed in this survey.

2.1. Strong vs. Weak Social Ties

Mark Granovetter introduced the concept of tie strength in [Granovetter 1973]. He characterized two types of ties: strong ties are tightly clustered and include close friends and family; weak ties typically connect acquaintances or distant friends.

Strong ties are individually more influential. Their removal disintegrates the local community. Recent studies have shown that people whose social circles overlap are likely connected by strong ties [Gilbert and Karahalios 2009].

Weak ties have the benefit of spreading new information between different communities [Granovetter 1974]. In social graphs, removing weak ties will delete the links that connect different communities, possibly leading to network disconnection.
2.2. Homophily
People with similar characteristics have high probability to form ties with each other, which is known as homophily. Homophily is defined as contacts (or relationships) between similar people that occur at a higher rate than among dissimilar people [McPherson et al. 2001]. In social contexts, people are likely to make friends with like-minded individuals and sometimes even adopt their friends’ behaviors and activities [Turner 1991].

2.3. Community Structure
Graph communities have a structural feature that nodes can be grouped into groups of nodes that are tightly connected inside the same group while loosely connected between groups. Thus, people within the same community have stronger social closeness than people from different communities. For example, communities in a co-authorship network represent different research areas [Girvan and Newman 2002] while within each community authors share similar research interests. To identify communities, various community detection algorithms have been proposed [Fortunato 2010].

2.4. Small World Phenomenon
Watts and Strogatz [Watts and Strogatz 1998] defined a family of networks that exhibit two properties: (1) the average shortest path length is small and (2) the clustering coefficient is higher than the corresponding random network. The former property shows that the average node-to-node distance is small. The latter property indicates that nodes tend to cluster together, forming dense complete triads.

Kleinberg [Kleinberg 2000] proposed a navigable small-world model that is the base of many small-world-based P2P overlays. The model adds long links with linking probability according to the length of the link, turning a 2D mesh network into a small-world network. Recent studies [Mislove et al. 2007] confirmed that the small-world phenomenon also exists in online social networks (not only in real-life social networks). For example, the clustering coefficients of Orkut, Flicker, LiveJournal and YouTube are between 3 and 5 orders of magnitude higher than the clustering coefficients of their corresponding random graphs with the same number of nodes and links. All of these networks have remarkably short diameters.

The small-world property is related to mixing speed that is evaluated by mixing time. Mixing time measures how fast a random walk in a graph reaches the “stationary distribution” [Mohaisen et al. 2010; Spielman 2006] (where nodes are roughly independent from the starting point). Graphs with fast mixing time indicate that any arbitrary destination in the graph is reachable with a probability proportional to the node’s degree. Conversely, a connected graph with a small quotient cut should have a large mixing time [Yu 2011]. A fast mixing graph is necessarily a small world while the opposite is not true [Dellamico and Roudier 2009]. Even if short paths exist in small-world graphs, random walks over such graphs can often remain confined to local clusters, which results in longer mixing time. For a small-world graph to be fast mixing, it has to contain a substantial percentage of bridges that connect different clusters. In this way, random walks have higher probability to move from one cluster to another, thus decreasing mixing time.

3. PEER-TO-PEER SYSTEMS
The P2P architecture has been positioned as an alternative to the traditional client-server architecture. In general P2P systems, each node is managed by an independent entity, and without a central authority, nodes form self-organizing and self-maintaining networks. In most cases, nodes are assumed to have equivalent roles.
This section provides the background needed for contextualizing the work surveyed in this paper. More detailed presentations of P2P systems and their underlying mechanisms are found in literature surveys such as [Hefeeda 2004; Lua et al. 2005; Wallach 2002].

3.1. Overlay Topologies
The P2P overlay, a key component in P2P systems, provides mechanisms for message routing, node membership management, node lookup, etc. Based on the topology of the overlay, P2P systems are typically classified as structured or unstructured. Hybrid topologies that include centralized and decentralized control are better known as super-peer architectures. In such topologies, some nodes (the super peers) exert centralized control over a small number of peers, while being connected among themselves via a structured or unstructured topology.

3.1.1. Structured Peer-to-Peer Overlays. The structured P2P overlay was designed to solve the problem of having a scalable P2P overlay network with no central control, also referred to by the term Distributed Hash Tables (DHTs). The basic idea is the following: objects and nodes have identifiers that can be mapped into the same space via a hash function. Each object is mapped onto the nodes with the closest hashed IDs to its own hashed object ID. The hash function is chosen to guarantee a uniform mapping of objects onto nodes, for good load balance. Looking up an object, therefore, translates into finding the node with the closest hashed ID to the (hashed) key of the object or the next successor node in Chord and ring-based DHTs, which is typically done by routing along a different geometry until the first such node is found. In traditional hash tables, a change in the number of array elements will cause nearly all keys to be remapped. Some of the DHTs use Consistent Hashing [Karger et al. 1997] which is a hashing scheme that does not significantly change the mapping of keys to elements when adding or removing an element. The most common geometries used for routing are ring (Chord [Stoica et al. 2001]), tree (Plaxton [Plaxton et al. 1999], Tapestry [Zhao et al. 2004], Pastry [Rowstron and Druschel 2001a]), hypercube (CAN [Ratnasamy et al. 2001]), and butterfly networks (Viceroy [Malkhi et al. 2002]).

DHT-based P2P overlays provide performance guarantees that unstructured overlays are unable to provide. Specifically, an object stored on a peer is guaranteed to be found. Moreover, in many of these solutions, the routing takes, with high probability, $O(\log(n))$ hops, where $n$ is the number of peers in the network. Another benefit is the small routing state maintained per node. Gupta et al. [Gupta et al. 2003a] and Gupta et al. [Gupta et al. 2004] independently questioned the need to keep per node routing state small and proposed an one-hop routing scheme in which each node in the system maintains complete routing state (membership information), leading to constant lookup paths (one overlay hop) and high lookup success rate.

3.1.2. Unstructured Peer-to-Peer Overlays. Unstructured overlays do not place strong restrictions on the geometry of the topology. Instead, nodes are connected with other nodes in the overlay based on latency, geography, or simply happenstance. This loose organization typically has lower overlay maintenance costs, but provides inefficient searching service because clients need almost blind searches and cover a large number of peers, which is inefficient and generates heavy loads in the network.

The first P2P systems based on unstructured overlays included FreeNet [Clarke et al. 2001] and Gnutella [Gnutella 2013]. Gnutella routed queries by flooding, that is, forwarding queries to randomly selected $k$ neighbors in the overlay, where $k$ was a system parameter initially set to 4. FreeNet improved on Gnutella's blind search by maintaining a routing table to record data placement information at each peer. Thus, each peer's routing table indicated which neighbor was likely to have the inquired file.
Other systems employed unstructured overlays for routing messages between a subset of the nodes in the networks. For example, in KaZaA [Leibowitz et al. 2003], nodes are classified as super-peers or regular peers. Super-peers (described below) are connected via an unstructured overlay, routing messages via flooding among themselves.

Napster [Napster 2001] used a centralized index of file. Upon joining the network, a peer registered its files with the central index. A lookup for a file (identified by filename) was thus a search in the index, which return the IP of the peer(s) that stored a file with the specified name. While the lookup was centralized, the transfer of the content was directly between the interested peers.

3.1.3. Super-Peer Architecture. Pure P2P systems tend to be inefficient [Yang and Garcia-Molina 2003]. By giving each peer equal roles and responsibilities, regardless of capability, bottlenecks are caused by limited resources at some peers and the scale of the network. The super-peer architecture addresses this issue by embedding centralized architecture in a decentralized system: a super-peer represents a group of peers, acting as a local, centralized server [Beverly and Garcia-Molina 2003]. A regular peer can communicate with its super-peers. Super-peers interact with other super-peers for message routing.

Because there are relatively many super-peers in a P2P network, each super-peer handles a small load, thus avoiding to become a bottleneck or a single point of failure for the whole system. Such architecture has the potential for autonomy, load balancing, and robustness to attacks [Saini 2002], and was used with both unstructured [Cholvi et al. 2004] and structured overlays [Zhu et al. 2003]. Deployed systems such as KaZaA [Leibowitz et al. 2003] and Skype [Baset and Schulrinne 2004] are thought to have adopted the super-peer architecture.

3.2. Advantages and Challenges of Peer-to-Peer Infrastructures

Several reasons account for the attractiveness of the P2P model. First, little or no administrative intervention is needed to maintain the system. Second, P2P networks are resilient in the face of faults and attacks, because no single node is critical to the system's operation, and a small number of node failures do not affect the performance of the entire system. Therefore, to attack or shut down a P2P system, an attacker must simultaneously target a large proportion of nodes. Third, in some scenarios, P2P systems have potential for good load balancing due to their decentralized character, which means that no node will act as a server to serve all requests from other nodes. Fourth, popular P2P systems have an abundance of diverse resources that few organizations are able to afford [Rodrigues and Druschel 2010]. The resources tend to be diverse in terms of their hardware, software, geographic location, and network services. This diversity reduces the system's vulnerability to correlated failures, attacks, and even censorship.

However, P2P technologies also face many challenges. We summarize the main challenges in P2P networks as follows:

Security and privacy: In the absence of a central authority for managing strong user identities, achieving a high level of security is difficult. Malicious users can distort message delivery [Castro et al. 2002], launch Sybil attacks [Douceur 2002] or denial-of-service attacks [Dumitriu et al. 2005].

Incentives for resource contribution and participation: The nature of voluntary resource contribution mechanism in P2P networks requires incentives to stimulate participants to donate their resources for the common good of all peers. In most scenarios, users would not contribute their resources (bandwidth, disk space, data, or CPU cycles) if they can get a service for free [Krishnan et al. 2004]. Moreover, once they get what they want (e.g., download a large file), users/peers leave the network, thus reducing...
the resource pool (e.g., another copy of the file) and increasing churn. Therefore, designing incentive mechanisms to encourage user collaboration and ultimately improve the system’s performance is an inevitable issue for P2P-style systems.

**Quality of service:** A main challenge for P2P systems is the lack of quality of service guarantees. The quality of P2P solutions is prone to the unreliability of individual participants (whether in hardware or software shared resources or their accurate representation [Saroiu et al. 2001]), high churn [Stutzbach and Rejaie 2006], and adequate provisioning for dynamic workloads. While incentives [Habib and Chuang 2004; Chuang 2004], churn-resistance [Qiao and Bustamante 2005] and reputation systems [Kamvar et al. 2003; Gupta et al. 2003b] address part of the problem, the resource provisioning challenge seems to have roots in the very philosophy of the free-sharing, grass roots movement at the base of P2P systems.

**Resource Management and Search:** Managing resources under high churn conditions becomes difficult. For example, search for resources in a high churn environment, thus, whether explicit part of the application features (as in file-sharing or video-streaming systems) or implicit to the functioning of the system becomes a building block. Two performance metrics are particularly important and thus subject to the design improvements described in literature: response time and success rate.

### 3.3. Examples of Peer-to-Peer Applications

As many P2P-style systems will be introduced in the following sections, we only list several typical examples of commercial applications. Our focus on commercial systems, while artificially limiting, allows us to give an overview of representative P2P applications without losing the focus of this paper.

File sharing applications have been widely used to disseminate files on the Internet. For example, BitTorrent is a P2P protocol that distributes large amounts of data over the Internet. eMule [Kulbak and Bickson 2005] is another P2P file sharing application, in which files are directly exchanged between peers and a credit system (as incentives) is used to reward uploaders who frequently upload videos. LimeWire is also a P2P file sharing client program, which uses Gnutella network and BitTorrent protocol to support search, download, upload, and file distribution functions.

P2P streaming video freeware like PPLive and PPLive [Silverston and Fourmaux 2007] combine P2P and Internet TV to broadcast live stream over Internet. In P2P streaming, participants actively contribute their resources by forwarding their contents to other peers. It supports high-volume video transmission traffic: several hundred thousand users can watch a live stream smoothly [Hei et al. 2007].

Voice over IP (VoIP) technologies combine P2P infrastructure with voice to allow direct voice communication among Internet users. Skype is the most successful example. It uses a hybrid, super-peer architecture [Baset and Schulrinne 2004].

P2P storage systems allow users to store copies of their files at other nodes in the network, thus files can be retrieved if the local filesystem fails. This decentralized storage system can be an attractive way to increase the availability of users’ data by replication. Such applications, like Crashplan [Inc. 2014], supply P2P storage services: backup data to users’ other computers, to remote servers, or to computers that belong to users’ friends and family.

P2P Online Social Networks (OSNs) attempt to address two main limitations of centralized OSNs—user privacy and scalability—while providing similar functionalities. The best known commercial system, Diaspora [Diaspora 2014], is a privacy-aware, decentralized online social network where users are given the power of deciding where to store their data. Many such systems have been proposed in the literature, as shown in a recent survey [Paul et al. 2014], without having been adopted at scale. For instance, Safebook [Cutillo et al. 2009] relies on users’ trust relations from social networks to
generate a cooperative environment and build a privacy-preserving mechanism. LifeSocial [Graffi et al. 2011] shifts the load for operating the infrastructure from the service provider to users and distributes services in a P2P fashion to mitigate imbalanced workloads and resource limitations.

Domain Name System (DNS) adopts DHT techniques to address the bottleneck problem and DoS attacks. For example, the Cooperative Domain Name System (CoDoNS) [Ramasubramanian and Sirer 2004b] provides high lookup performance, resilience to DoS attacks and fast propagation of updates. CoDoNS built on Beehive [Ramasubramanian and Sirer 2004a], which replicates DNS mappings in the network to match predicted demand, decreasing the lookup hops to $O(1)$ in DHT. To contain the DoS attack, CoDoNS monitors the changes in the demand for objects and replicates objects across several nodes to balance the load on individual node, reducing hotspots. Consequently, it is difficult for attackers to identify a strategy that can cause the maximum damage to the system when nodes dynamically maintain equal load in the system.

Content delivery networks (CDNs) exploit DHT techniques to lower the cost that is generated by requesting a huge amount of data. In DHT-based CDNs, data are copied automatically and the replicas are organized in a DHT. When a client requests data, the DHT basic find algorithm checks for the existence of the data. If not found, the CDN server directly copies the data from the original server. According to the request frequency, the replica propagates reversely along the routing path. Data with less request frequency finally will be deleted from the caching server to save storage space for more desired data [Zhang et al. 2013; Passarella 2012]. Thus, the popular data have more replicas in the network and the response time is short, while the unpopular data have fewer copies to save storage space and update bandwidth.

One typical DHT-based CDN systems is CoralCDN [Freeman et al. 2004]. It reduces websites’ traffic by aggregating volunteers’ (who run CoralCDN) bandwidth to absorb the majority of traffic in the system. Instead of considering resource publishers, CoralCDN replicates content according to the proportion of contents’ popularity. CoralCDN is built on top of Coral, which is a P2P key/value indexing infrastructure that has no hot-spot congestion. Coral uses DHT routing techniques to locate nearby replicated copies of web content but adapts DHT for web content distribution.

The concept of P2P has also been used for designing virtual monetary systems to supply low fee (e.g., 2-3% lower than credit card processors [Wingfield 2013]), no charge-backs and nearly anonymous transactions. BitCoin [Nakamoto 2008] is the best known virtual currency payment system. Unlike traditional currencies, BitCoin does not rely on any centralized authority (country or state) to control the process of transaction but relies on voluntaries with cryptographic proof. It uses a public-key cryptography for money transfer.

4. BENEFITS OF SOCIALLY AWARE PEER-TO-PEER SYSTEMS

As already mentioned, we call socially aware P2P systems those systems that integrate social knowledge in their design. In some sense, social networks and P2P networks are a natural match [Huang et al. 2012]. On one hand, the typical challenges of P2P systems, such as free riding, inefficient search, and trust management, have built-in solutions in the social fabric among system users. On the other hand, in social information systems, the management of large amount of (user) information can be more efficiently managed by distributed technologies than by centralized solutions. Consequently, leveraging the social fabric in P2P system design can provide benefits to both P2P mechanisms and social information management services.

Figure 1 depicts a general-purpose architecture of decentralized socially informed applications as per the topic of this paper. The architecture consists of two layers. The
Fig. 1. The architecture of socially aware P2P applications. The lowest layer is a P2P-based communication overlay. The social graph as a logical layer captures the relationships among users. These two layers together support various socially aware P2P applications.

This section presents the benefits of integrating social properties in the design of P2P applications as presented in the literature to date.

4.1. Social Incentives Encourage Cooperation

P2P systems rely on voluntary resources contributed by individual peers and the cooperation among users. In these systems, users have implicit disincentives to cooperate because cooperation consumes their own resources (e.g., bandwidth and disk space) which may degrade their own performance. In the absence of appropriate incentives, P2P systems are plagued with free-riding [Hughes et al. 2005] and churn [Stutzbach and Rejaie 2006] at levels higher than socially informed P2P systems [Nguyen et al. 2011; Li and Dabek 2006]. For example, an early Gnutella study [Krishnan et al. 2004] shows that almost 70% of users are free-riders, and the top 1% contributing peers return 50% of all search results. If a contributing peer leaves, others may be negatively affected by poor quality of service, increased overhead, or end-to-end latency.

At the same time, studies have confirmed the real-life fact that people are more generous toward friends [Axelrod 1985] and towards the closest people in their social circle [Branas-Garza and Alejos 2006]. This inherent generosity towards existing so-
cial contacts has been tapped for sustainable cooperation in distributed systems. For example, declared friendships on Orkut [Tran et al. 2008], or shared photo interests in Flickr [Zwol 2007] have been utilized as social incentive mechanisms. In the following, we describe some P2P systems that leverage social incentives to improve the performance of systems.

F2F [Li and Dabek 2006] gives peers the choice of storing data on their friends’ computers. This approach exploits users’ embedded incentives to help their friends and results in a more stable system, which consequently reduces the cost of data maintenance. There are two types of social incentives exploited in this solution: first, users are likely to keep their systems online when they know they may be of use to their friends, which reduces churn. Second, distinguishing between permanent failures and transient failures can be done more accurately, with out-of-band social relationships, which reduces significantly the costs of unnecessary replication. Experimental results show that a F2F system needs 5.6 times less storage and bandwidth compared to a non-F2F solution for maintaining the same level of quality of service.

Stir [Nguyen et al. 2011] is a video streaming P2P system that provides the means for users to interact with each other. The solution exploits the spontaneously formed social relations between peers for incentives. Users need to socialize by offering an in-session communication channel, where social relations among peers are formed spontaneously. Because such social relations are formed in the P2P network, more peers could have similar video interests compared to the imported social relations from other separate social networks (e.g., Facebook). As a result, a user could have enough qualified connections, and the more social connections a user, the longer he or she stays in a session to interact with them, helping to establish durable connections that are useful for reducing peer churn.

4.2. Homophily Improves Search Efficiency

In P2P networks, peers lack global knowledge about the network, thus they need to interact with each others to find the required resources. Search thus becomes a basic component with significant impact on the quality of the services provided.

Several systems, described below, have been designed to exploit homophily to improve search performance in large P2P networks. Homophily supports the principle that people with similar tastes and characteristics tend to connect socially with each other [McPherson et al. 2001]. The intuition used in these systems is that directing the search to closer social contacts first can find a solution much faster than the system-based routing used in the previous solutions.

For example, Anwar et al. [Anwar et al. 2005] leverage users’ shared interests from an online social network, Orkut, as the basis to form a routing mechanism. The routing begins from searching a user’s directly connected friends with a breadth-first search, then friends of friends, and so on, looking for a match in (declared) interests. To evaluate the strength of this search solution, four different kinds of interests (e.g., music, sports, movie and TV series) are tested. Experimental results indicate that, on average, it took only three hops to find similar interests with a probability of 97%. Based on this evaluation results, the authors further compare Orkut-based routing with two non-social routing protocols, ESM [Chu et al. 2002] and NICE [Banerjee et al. 2002], and find that social-based routing gives not only lower delays, but also steady performance, even when increasing the size of user friends’ groups that are formed by users with similar declared interests on Orkut. In contrast, ESM presents a large fluctuation of delays as user group sizes increase, and NICE shows higher delays than Orkut-based routing.

SPROUT [Marti et al. 2004a] also employs social links from OSNs in the design of its routing algorithm. When routing to a node with ID \( k \), SPROUT locates the node's
Table I. Summary of socially informed P2P search systems.

<table>
<thead>
<tr>
<th>System</th>
<th>Social Knowledge</th>
<th>Benefits</th>
<th>Topology</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tribler [Pouwelse et al. 2008]</td>
<td>Users' similar tastes of files</td>
<td>Fast speed on content discovery</td>
<td>Structured</td>
</tr>
<tr>
<td>Anwar's Search Algorithm [Anwar et al. 2005]</td>
<td>Searching only through friends</td>
<td>Defend against misroute; Reduce query times</td>
<td>Structured</td>
</tr>
<tr>
<td>SPROUT [Marti et al. 2004b]</td>
<td>Routing along trusted social links</td>
<td>Defend against misroute; Reduce query times</td>
<td>Structured</td>
</tr>
<tr>
<td>Intelligent Social Search [Yang et al. 2009]</td>
<td>Group users with similar social interests</td>
<td>Shorten search routes</td>
<td>Unstructured</td>
</tr>
<tr>
<td>socP2P [Farahbakhsh et al. 2012]</td>
<td>Social commons</td>
<td>High searching success rate; Low message overhead</td>
<td>Unstructured</td>
</tr>
<tr>
<td>Interest-sharing graph [Iamnitchi et al. 2011]</td>
<td>Users' similar data requests</td>
<td>High hit rate</td>
<td>Unstructured</td>
</tr>
<tr>
<td>Social-P2P [Li and Shen 2012]</td>
<td>Users' common interests</td>
<td>Efficient file search; High hit rate</td>
<td>Structured</td>
</tr>
<tr>
<td>NetTube [Cheng and Liu 2012]</td>
<td>Users' similar interests of videos</td>
<td>Quick video search; High hit rate</td>
<td>Structured</td>
</tr>
</tbody>
</table>

friend node with the closest ID but smaller or equal to $k$. If such a node exists, it will forward the message to it. The friend node repeats the same procedure. If no friend node has its ID smaller than $k$, the regular Chord routing algorithm is used to continue forwarding the message to the destination. SPROUT prominently improves the success of query results and reduces query delays: simulations show that the average routing path in SPROUT is more than 1.5 times as likely to succeed compared to regular Chord with augmented links that are comparable to the social links used in SPROUT.

Tribler [Pouwelse et al. 2008] is a social-based P2P file-sharing system. Its approach is based on the fact that homophily—in this case, similar taste in content—encourages altruistic behavior. Relying on the “Files I Like” module in Tribler, each peer indicates its file preferences and the system utilizes this function to connect people with similar tastes on files. At the same time, social network creation in Tribler is facilitated by directly importing contacts from existing social networks, such as MSN or GMail. In this way, groups are formed either with similar file preferences or with original groups in OSNs. Within the same group, by introducing social group incentives, altruistic peers contribute their bandwidth by joining a swarm even if they are not interested in the content, without requesting anything in return. Thus, more file seeds are available in the system. Simultaneously, file downloaders dynamically choose the best file seeds from their available social circles. All these reasons together are the key to improve the entire system's download performance. Experiments with a swarm of about 1,900 peers prove Tribler's resilience to free-riding and an improvement of up to 6 fold in file downloading performance.

Yang et al. [Yang et al. 2009] implement a P2P search system that clusters peers into social groups according to criteria like shared preferences or background. The solution is based on the fact that social closeness may help people find information more effective [Mislove et al. 2006; Evans and Chi 2008]. Each group has a super-peer to handle its peers' requests, and peers and their super-peers are "friends". Search begins within the group. If the content cannot be found within the group, queries will be forwarded to neighboring super-peers, which in turn search in their own groups. The solution establishes and maintains an overlay network of super-peers on top of an unstructured P2P overlay. To evaluate the performance of the super-peer-based social search system, three different routing schemes are compared: flooding, random grouping of peers, and interest-based clustering of peers. Simulation results show that social-based super-peer search system can significantly shorten search routes and enhance performance. The system's average hit path length is 1.8, which is the shortest compared to other three strategies (3.3 for flooding, 2.5 for random super-peer, and
1.9 for interest cluster). In addition, social-based super-peer and interest cluster approaches can find about 80% of relevant resources while the other two only reach 25%.

SocP2P [Farahbakhsh et al. 2012] relies on social commons (shared interests, background and experiences) among peers to locate content. In SocP2P, nodes improve search by using the social knowledge they gained during their stay in the network, such as peers’ common interests, friendships, and previous interactions. The P2P links are established based on users’ interest similarities instead of random associations. New links are created when an interest similarity is found between two users, for example when a node responds to another node’s request. This search mechanism achieves a higher success rate and lower message overhead compared to Random Walk [Bisnik and Abouzeid 2005].

Iamnitchi et al. [Iamnitchi et al. 2011] introduce an interest-sharing graph that connects users who request the same resources during a certain period. They show that the interest-sharing graph can then be used as an unstructured P2P overlay that can be partitioned into clusters of shared interests. A search mechanism on top of this overlay is proposed, in which local search history is disseminated within each cluster, and requests are forwarded from cluster to cluster when answers are not found locally. Experiments based on real traces show that this approach leads to 70% of the requests to be satisfied locally.

Social-P2P [Li and Shen 2012] uses peers’ social closeness and common interests to enable trustworthy file sharing. It groups common-multi-interest peers into interest-based clusters where nodes are connected based on their social links, and trust between peers is measured by interaction frequency. Nodes forward their queries to neighbors with higher trust. Each cluster selects a stable node as an ambassador, and ambassadors form a DHT used for inter-cluster file sharing. Inside each cluster, random walk is used to forward messages along trustworthy paths. When files cannot be found within a cluster, inter-cluster routing is needed. In this case, the ambassador in each cluster, acting as a super-peer, is in charge of forwarding queries to ambassadors in other clusters via DHT routing. This solution also relies on social incentives to ensure the quality of the answers.

NetTube [Cheng and Liu 2012] is a social-based, P2P assisted, short-video streaming system that creates a bi-layer overlay. The first overlay is among videos that are related based on users’ expressed interests: if a user watched two videos, the two videos are linked in this overlay. Each peer caches all the videos it played. The upper-layer overlay connects peers based on their common interest in a particular video, as in a typical BitTorrent swarm: if two peers played (and thus are caching) the same video, they are linked in this overlay. This approach connects a peer with many more neighbors than peers in the non-social Peer-Assisted-Video-on-Demand (PA-VoD) system [Huang et al. 2007], which is known for relieving server stress of MSN videos without client-side caching and sharing. This bi-layer overlay is mainly used for video search. By exploring peers’ video interest correlations, a peer in NetTube can correctly and quickly locate the potential suppliers for the video that enables a smooth video playing transition. If the videos are independent, peers can only find less than 20% sources. When considering video interest correlation, the accuracy reaches 55%.

Table I summarizes the search algorithms presented in this section.

4.3. Social Trust Prevents Privacy Disclosure and Defends Against Malicious Attacks

The popularity of P2P networks has resulted in huge amounts of services and data being distributed on a global-scale over Internet. The strength of the P2P paradigm stems from its decentralized nature—no central authority manages the entire network. However, the downside is that users generally lack accountability and have no control about whom they are interacting with, which causes private sensitive informa-
tion disclosure and security concerns. Large-scale P2P networks like BitTorrent and Skype are vulnerable because of lacking appropriate policies to protect users’ privacy. For example, in BitTorrent, users’ behaviors are easily monitored by anyone who cares to look [Isdal et al. 2010]. Similarly, eavesdropping on Skype Internet traffic to collect data about who is talking to whom and how long the conversation lasts is easily doable [Suvanto 2005]. Moreover, a small set of adversaries can control a large portion of the network, and increase their participation in routing in order to suppress correct content delivery to other users (e.g., drop, forward, or damage messages) or to endlessly delay responses. The privacy and security issues are significantly challenging in the P2P context due to the lack of global servers. Hence, this research area has triggered a great number of follow-up studies. Recent studies have suggested that integrating social knowledge into the design of P2P systems is one way to provide secure communication which further protects user privacy and thwarts adversaries [Mittal 2012].

4.3.1. Social Ties Prevent Privacy Disclosure. The main idea is to leverage trust relationships from social networks and limit information exposure only to such trusted peers, thus preventing eavesdroppers or malicious peers from having access to information. Social trust provides the guarantee that friends will not garble or drop messages deliberately. Likewise, friends will relay messages through their friends, continuing the forwarding process until messages reach their destination. Such systems are presented in the following section.

Pythia [Nilizadeh et al. 2011] provides a privacy-enhanced social search solution that preserves the privacy of askers and responders in a social search system. A peer’s expertise areas are not related to its identity information beyond a certain threshold probability, and the same should hold true for an asker’s queries. Such anonymity prevents attackers from observing which node is answering or asking questions. Pythia groups nodes in social networks into anonymizing communities according to their social relationships by using the distributed community-forming algorithm proposed by Ramaswamy et al. [Ramaswamy et al. 2005]. This algorithm identifies a representative for that community who will be the only one communicating with other communities, via their representatives. Requests are sent to the representative of the local community, who sends it to all peers in its community. Responses are sent back to the representative, who forwards them to the requester. If no responses are found in the local community, the request is forwarded to the representatives of other communities, who repeat the process.

X-Vine [Mittal et al. 2012] is a protection mechanism for DHTs, which prevents participants from security and privacy vulnerabilities by leveraging social links for pseudonymous communication. The architecture is composed of three layers: the top layer is a communication overlay, the intermediate layer is the social network topology, and the bottom layer is the IP layer. X-Vine constructs a social overlay on top of an online social network where nodes maintain direct links with friends and a Chord-like overlay with non-friends. These non-friend nodes are selected in a Chord manner: each node is assigned an identifier from a ring namespace, and overlay links are formed to nodes that are successors and fingers. Using a Chord-like routing scheme, nodes can route to any other node in the namespace. However, unlike in Chord, a node is not allowed to directly communicate with its DHT neighbors, but can only communicate with its social contacts. In this way, X-Vine prevents users social contacts and their IP addresses from leaking outside their social circles.

Turtle [Popescu et al. 2004] relies only on a social graph to build an overlay on top of users’ pre-existent trust relationships to supply safe sharing of sensitive data (e.g., private pictures and confidential documents). Each user creates a shared secret and
Table II. Representative socially aware P2P applications.

<table>
<thead>
<tr>
<th>Application</th>
<th>Solution Examples</th>
<th>Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>Social P2P File Sharing</td>
<td>Social-P2P [Li and Shen 2012], Tribler [Pouwelse et al. 2008]</td>
<td>Sharing files among socially close friends with access control</td>
</tr>
<tr>
<td>P2P Storage</td>
<td>F2F [Li and Dabek 2006], Friendstore [Tran et al. 2008], FriendBox [Moreno-Martinez et al. 2012]</td>
<td>Backing up resources with trusted friends</td>
</tr>
</tbody>
</table>

establishes a cryptographically secure connection with the nodes in their friend set by out-of-band means. Once established, the secure links can be used for any type of communication in which both the requester and the responder are only known by their own friends. This allows the requester and the responder anonymity, even protects each query path of the intermediate data. All direct interactions take place between peers with trusted and respected ‘friends’. As a consequence, Turtle withstands most of the denial of service (e.g., malicious routing, bogus query hits and aborted transfers) and Sybil attacks.

Another method for limiting the power of malicious users is presented in Prometheus [Kourtellis et al. 2010]. Prometheus is a decentralized social data management system built on top of a P2P architecture. Its basic assumption is that socially-connected users will choose to store their social data and security credentials on the computing node (peer) contributed by one of their friends. Thus, groups of socially-connected users will share the same peer in the P2P network. This assumption implicitly leads to an increased resilience to malicious routing, since access to social contacts close in the social network will be mostly local, thus reducing the chance of a malicious peer intervening in the routing process. Experimental results show that, in comparison with a random mapping of the social graph onto the P2P overlay, the social mapping limits the influence of malicious nodes by up to 40%.

Table II summarizes representative socially aware P2P applications discussed by far in this section.

4.3.2. The Topological Characteristics of Social Graphs Defend against Sybil Attacks. Sybil attacks refer to malicious users joining the system under multiple identities [Douceur 2002] in order to control the system or collect information. For example, in P2P systems, Sybil users can collude to break reputation mechanisms or misroute messages. Sybil attacks have been discovered in systems such as Maze [Lian et al. 2007] and applications implemented based on Kad routing protocol [Steiner et al. 2007]. The obvious solutions against Sybil attacks have proved inefficient: validating the uniqueness of a node’s IP address [Rowaihy et al. 2007] can be subject to spoofing [Herberlein and Bishop 1996]; employing CAPTCHA puzzles [Ahn et al. 2003] to distinguish between humans and robots is counterbalanced by crowdsourcing [Doan et al. 2011].
This section gives a quick review of typical systems that leverage social graph attributes to mitigate Sybil attacks. More in-depth discussion can be found in [Yu 2011; Boshmaf 2012]. We include the latest work on this topic in this section.

Solutions that identify Sybil users typically exploit the intuition that (1) regardless of how many Sybil identities a malicious user creates, it can only establish limited number of edges between honest nodes and Sybil nodes, and (2) if malicious users create too many identities and given that the number of edges between honest nodes and Sybil nodes remains fixed, the resulting graph structure is suspiciously different from an “honest” graph. Such solutions typically make the following assumptions:

1. The Sybil and the honest region are loosely connected compared to the non-Sybil node circles [Boshmaf 2012]: even if an attacker can create arbitrary Sybil accounts, he cannot establish an arbitrary number of social ties (called attack edges in Figure 2) with non-Sybil nodes in social networks.
2. Social graph’s honest region is fast mixing. The mixing time is no longer than $t(n)$ where $t(n)$ is a graph routing time function of the size $n$ of the honest region [Yu 2011].
3. At least one honest node is known.

Based on these three assumptions, various techniques have been proposed to distinguish Sybil from non-Sybil nodes in social graphs.

Decentralized or hybrid schemes require each node to be aware only of its immediate neighbors in the network, without information of the full graph. For example, SybilGuard [Yu et al. 2006] and SybilLimit [Yu et al. 2008] are two decentralized techniques that detect Sybil nodes via random walks in the social graph to identify “abnormal” mixing times. SybilGuard guarantees that it only admits $O(\sqrt{n \log n})$ Sybil users for each attack edge ($n$ is the number of users in the network). SybilLimit further limits the amount of admitted Sybil nodes for each attack edge to $O(\log n)$ by allowing multiple random walks to occur on the graph ($n$ is the number of honest users in the network).

GateKeeper [Tran et al. 2011] further improves this bound to $O(\log k)$ where $k$ is the number of attack edges the adversary has, by using a distributed ticket mechanism to admit most legitimate users but limiting Sybil nodes. In GateKeeper, each node only needs knowledge of its immediate neighborhood rather than complete knowledge of the graph. The design assumption is that a randomly chosen ticket source is relatively far away from most attack edges because an attacker only controls a small number of attack edges. As a result, few tickets spread along attack edges. Each ticket source propagates tickets to its immediate neighbors in a breadth-first search manner after consuming one ticket for itself, until no tickets are left. Ultimately, a node is labeled as
“honest” as long as it is reachable by a certain number of ticket sources \( f \times m \) where \( f \) is a parameter with a small value (e.g., \( f = 0.2 \)), and \( m \) is the number of ticket sources.

Whanau [Lesniewski-Laas and Kaashoek 2010] applies random walks and fast mixing on an expander graph to resist Sybil attacks in the DHT layer. It depends on the sparse cut assumption to distinguish legitimate from Sybil regions: random walks on legitimate regions are fast mixing and the end nodes of random walks have high probability to be legitimate nodes. Based on this theory, Whanau uses social relations to build routing tables and designs a strawman protocol to resist Sybil attacks. More specifically, each node stores \( k \) keys at most, and the strawman protocol means that each node initiates a certain length of random walks on the social graph and records a random key from the final node of each walk in a local table. To perform a lookup, a node first checks its local record table. If the key is not in the table, the node broadcasts the key to \( O(\log \sqrt{km}) \) nodes in its table, where \( k \) is the number of keys stored per node, and \( m \) is the number of honest edges. Simulations using social networks from Flickr, LiveJournal, YouTube and DBLP demonstrate that with the number of attack edges smaller than 10% of the number of honest nodes, most lookups in Whanau return correct values within a few messages. Even if the number of attack edges increases, successful lookup rates are still stable.

Cao et al. propose SybilRank [Cao et al. 2012] to rank users based on their likelihood of being Sybils. In a social graph, when a random walk is sufficiently long, its convergence property [Behrends 2000] leads to a uniform degree-normalized probability of landing at any node. However, a short random walk that begins from a legitimate (non-Sybil) node tends to stay within the non-Sybil region because the walk has low probability to traverse any of the relatively few attack edges. SybilRank ranks nodes according to the degree-normalized probability of a short random walk that starts from a non-Sybil node to land on them. The low-ranked nodes are classified as potential Sybil users. SybilRank achieves a significantly higher detection accuracy than other methods (including GateKeeper and SybilLimit) at a low computational cost.

Centralized strategies for defending against Sybil attacks require global knowledge of the social graph. Usually, a peer has to load the entire graph information to implement the Sybil detection or resistance technologies.

Ostra [Mislove et al. 2008] leverages social links to thwart unwanted communications from malicious to honest users based on the fact that a user has to take extra efforts to create many arbitrarily trust relationships, which is difficult (Assumption 1). Users classify communication they receive as relevant or irrelevant. The basic idea is that for two nodes to be able to communicate, there must be a path between them on the social graph, otherwise the communication will be blocked. Each link between users is assigned a quota. If a user is reported as sending spam (irrelevant communication), then all his links’ quotas will be decreased to limit the user’s misbehaviors. Thus, if an attacker keeps sending irrelevant messages to honest users, the quotas on all links from the malicious user will finally drop to zero. Then, the adversary will run out of paths to honest users, which means he cannot send messages anymore. By using relationships declared on YouTube, Ostra significantly limits the number of unwanted messages: on average, every legitimate user receives 0.22 malicious messages per week when 1% users are attackers who send more than two million unwanted messages.

Viswanath et al. [Viswanath et al. 2010] show that off-the-shelf community detection techniques are an effective tool to detect Sybil nodes in social graphs without any other knowledge. Community detection algorithms work because of two facts. First, nodes within a community are highly connected. Second, when ranking nodes based on how well they connect to a trusted node, nodes in the local community that are close to trusted nodes are ranked higher than the rest of the nodes. That is, community de-
A Survey of Socially Aware Peer-to-Peer Systems

Table III. Summary of Defending Sybil Attacks via Using Social Knowledge.

<table>
<thead>
<tr>
<th>System</th>
<th>Assumptions</th>
<th>Algorithm</th>
<th>Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>SybilGuard [Yu et al. 2006]</td>
<td>1, 2, 3</td>
<td>Random walk</td>
<td>Decentralized</td>
</tr>
<tr>
<td>SybilLimit [Yu et al. 2008]</td>
<td>1, 2, 3</td>
<td>Random walk</td>
<td>Decentralized</td>
</tr>
<tr>
<td>GateKeeper [Tran et al. 2011]</td>
<td>1, 2</td>
<td>BFS random walk</td>
<td>Decentralized</td>
</tr>
<tr>
<td>SybilRank [Cao et al. 2012]</td>
<td>1, 2, 3</td>
<td>Random walk</td>
<td>Decentralized</td>
</tr>
<tr>
<td>Ostra [Mislove et al. 2008]</td>
<td>1</td>
<td>Random walk</td>
<td>Centralized</td>
</tr>
<tr>
<td>Community-based detection [Viswanath et al. 2010]</td>
<td>1, 2, 3</td>
<td>Community detection algorithms</td>
<td>Centralized</td>
</tr>
</tbody>
</table>

dection algorithms can classify Sybils and non-Sybils into two loosely connected groups. The key point is to specify an adequate threshold parameter to control the number of communities generated by community detection algorithms. The experimental results prove that community detection algorithms can achieve similar false positives or negatives as SybilLimit.

The studies introduced above focus on applying social networks to defend against Sybil attacks by distinguishing between Sybil and honest nodes. Table III summarizes these solutions.

5. CHALLENGES OF SOCIALLY AWARE PEER-TO-PEER SYSTEMS

Much of the promise of socially aware P2P systems stems from their decentralized design and the exploiting of social (graph) properties. However, these very properties also expose such systems to challenges not faced by other types of systems. An overview of the challenges that socially aware P2P systems face is presented below. Some issues have been addressed to varying degrees in the literature, while others remain open.

5.1. Problems Caused by Small Friendsets

Many of the socially aware P2P solutions presented above, such as Friendstore [Tran et al. 2008], FriendBox [Moreno-Martínez et al. 2012], and F2F [Li and Dabek 2006], fit in the Friend-to-Friend category, in which direct connections are made only between peers corresponding to users who are socially connected. Such solutions, however, were shown to suffer from several significant limitations. First, users with a small set of friends are penalized by lack of available services for their needs [Li and Dabek 2006]. Second, friends are typically in close geographical proximity, and thus their online times are synchronized, exacerbating the problem of low service availability [Raúl et al. 2012] (discussed in Section 5.1.1). Third, as a consequence of the first two issues, the system suffers from overall imbalanced workloads that lead to low resource utilization (addressed in Section 5.1.2).

5.1.1. Low Service Availability. In F2F-like systems, service availability (e.g., data availability in storage systems) strongly correlates to the number of social contacts and the overlap of online time between users and their friends. Due to geographical colocation, it is possible that a user's friends are all simultaneously offline when the user desires to obtain services from friends' peers [Sharma et al. 2011]. This occurs with high probability especially when users have few social connections. Thus, generally speaking, F2F systems have lower service availability than generic systems with random service providers [Li and Dabek 2006].

Redundancy is the default method to improve service availability. For example, in storage systems, replicas on more than one of the user's friends' machines is the go-to solution, yet it may bring higher costs for unclear benefits.
Sharma et al. [Sharma et al. 2011] address this problem via a greedy data placement heuristic to optimize the trade-off between the number of replicas and the covered data availability in F2F storage systems. The purpose of the algorithm is to find the minimum number of friends that can supply maximum coverage. The greedy algorithm defines critical time slot as that period when only one neighbor is online; and the specific neighbor is called critical neighbor because it is critical in providing such coverage. To get maximum coverage with minimum number of friends, a node $n$ first picks all critical neighbors from its friends. If the critical neighbors cannot cover all critical time slots that node $n$ searches, then it chooses a non-critical friend node that can cover the largest number of time slots not yet covered. This node-selection process continues until no more nodes are left, or no nodes left that can cover uncovered time slots. The authors observe that when using a greedy data placement heuristic in real-world users’ online/offline traces from an instant message server, roughly 50% of nodes have more than 70% of their time covered with three replicas. For comparison, if data is randomly replicated to friends or only with critical nodes (with the same replication), then 45% (data placed on friends randomly) and 40% (data placed on critical nodes) of nodes both have 70% coverage. In an extreme scenario, when data are replicated on all of a node’s friends, 50% of nodes have more than 90% of coverage. But such backup scheme results in high costs for storing data with multiple replicas, especially for users with many friends. Because it treads critical and non-critical nodes equally, the greedy heuristic works well even if the number of available friends is not large.

Instead of optimizing service placement algorithms, another approach is to expand the resource set while preserving social incentives. Zuo et al. [Zuo et al. 2016; 2014a; 2014b] propose a social strength metric $SS_n$ (where $n \geq 2$) to expand the set of service resources available to node $i$ to neighbors located $n$ hops away in the social graph who are still socially closer than $i$’s weakest direct social contact. The definition of the social strength metric is based on the following observations: (1) Social interactions among users were shown to represent more meaningful relations than just declared relationships on OSNs [Wilson et al. 2009]; (2) Social strength between two users is limited by the strength of the weakest tie on that path. Furthermore, the strength decreases with the length of the shortest path between the two individuals, as noted in [Friedkin 1983]; (3) Multiple types of social interactions result into a stronger (direct) relationship than only one type of interaction [Wellman and Wortley 1990]; (4) Social ties between individuals are asymmetrically reciprocal [Wellman 1988]. By using real-world datasets and both simulated and real-life users’ online/offline behaviors, this methodology improved data availability in some cases from 35% to 55%, thus by more than 50%.

5.1.2. Imbalanced Workload. In F2F systems, the loads placed on nodes are directly related to their degree, i.e., nodes with high degree are the ones most often chosen as a storage location while resources from low degree nodes are almost never requested. The capacity of F2F systems is limited by the resources contributed by its peers, such as bandwidth, disk space, or the number of files. However, the available resources are limited and determined by the number of a peer’s friends. As a result, the distribution of workloads in the entire system is imbalanced—the majority of workloads are allocated to sociable users while a few move to low-degree users, which results in low resource utilization and network traffic jams. For example, Friendstore has lower utilization (87%) [Tran et al. 2008] compared to storage systems such as DHash [Dabek et al. 2001], Pastry [Rowstron and Druschel 2001b] and OpenDHT [Rhea et al. 2005], which can achieve perfect utilization. The main reason behind such low resource utilizations is that most unused resources reside on nodes with few social connections [Tran et al. 2012]. Therefore, how to balance the workloads between low-degree and high-degree
users while still keeping incentives and trust based on social knowledge is an open problem.

Because its most intuitive advantage is an increase in the number of resource sharing candidates, the social strength metric $SS_n$, discussed in Section 5.1.1 is a promising solution for mitigating imbalanced workloads in F2F systems. Although the candidate set of high-degree users expands more than that of low-degree users, adequate load allocation protocols could even the workload distribution. One straightforward protocol is giving high priority to low-degree users to be selected as service providers. In this way, low-degree users are intentionally involved in more resources sharing activities than without friendsets expansion, which could transfer some workload from high-degree users.

5.2. Meaningful Quantification of Social Closeness

Social closeness is used to measure the trust among resource-sharing peers, that in turn is used for stimulating incentives for cooperation, preventing privacy disclosure, and thwarting malicious users in self-organized, open environments. The proliferation of OSNs provide researchers a rich amount of data about users’ computer-mediated interactions, and with it new opportunities to examine and quantify social closeness. A straightforward proxy to evaluate users’ social closeness is based on declared relationships, such as “friends”.

However, with better understanding of the use of OSNs, it becomes evident that declared relationships do not accurately translate into meaningful social closeness due to two facts. First, declared relationships in OSNs often reduce relationships to simple binary relations. In real life, the strengths of social relationships between people are unequal [Hangal et al. 2010] and asymmetrical [Wellman 1988]. Second, people often do not decline friendship requests even when bogus. For example, Boshmaf et al. [Boshmaf et al. 2011] prove that in practice many Facebook users blindly accept friendship requests made by social bots.

To give a fine-grained measure for the strength of a social tie, studies have employed users’ interaction frequency [Wilson et al. 2009], overlapping of social circles [Onnela et al. 2007] and duration of communications [Gilbert and Karahalios 2009], which cannot be directly seen from binary, declared relationships. Unfortunately, validating such metrics is difficult, given that user studies based on surveys are often unreliable [Kilworth et al. 1990].

A more accurate measure of the strength of social ties will probably need to capture interactions across multiple communication channels, as proposed in [Iamnitchi et al. 2012], such as over email, instant messaging, mobile phone, various OSN platforms (both generic, as Facebook, and activity-specific, such as Steam Community), etc. In this situation, however, significant privacy issues will emerge due to the aggregation of information from different social spheres (e.g., professional and personal) [Kayes and Iamnitchi 2013]. Moreover, the digital recording of our social interactions will still be a limited representation of our social life: face-to-face communication will hopefully still be the de facto for close relationships, such as family. In addition, interactions that occurred long time ago are difficult to value [Viswanath et al. 2009].

5.3. Building Practical Social Graph-Based Security Solutions

A large body of graph-based Sybil defense strategies discussed in previous studies are based on the three assumptions mentioned before: (1) the adversary cannot arbitrarily establish many connections with honest identities; (2) the subgraph, including honest users, is fast mixing; (3) detectors know at least one honest identity in the graph.

Recent research, however, shows that real-world social and information systems do not necessarily conform to the first two assumptions. For example, Leskovec et
al. [Leskovec et al. 2009] show that large information networks (including social networks) have many small communities that seldom connect to tightly-connected communities to form a big one, and large communities gradually connect to the remaining of the network and embed them into the network core. Likewise, Mohaisen et al. [Mohaisen et al. 2010] prove that many graphs are not fast mixing. Moreover, by building socialbots on Facebook, Boshmaf et al. [Boshmaf et al. 2011] show that malicious users can infiltrate a target OSNs at scale via befriending an arbitrary number of regular users.

It is unclear how the graph-based Sybil (GBS) detection algorithms (reviewed in Section 4.3.2) perform when these assumptions do not hold. To systematically evaluate GBS algorithms in practice, Boshmaf et al. [Boshmaf et al. 2013] extend the work from Viswanath et al. [Viswanath et al. 2010] and propose a unified framework to implement and evaluate GBS algorithms empirically. In general, GBS algorithms include two types of graph community detection methods: global and local. However, two drawbacks weaken the applicability of global community detection. First, the detection has to be implemented on the whole graph, which is time consuming. Second, if attackers alter Sybil regions into two disconnected components that are only connected to an honest region by attack edges, for example, then the binary detector will misclassify one of the Sybil region into the honest region. Therefore, instead of finding a community structure around known legitimate nodes, Boshmaf’s framework suggests to find local community structure around a group of known honest nodes, regardless of the global community structure in the graph. Additionally, the authors empirically show that regular Sybil detection algorithms should be run to capture the evolution of the graph before Sybils are well-embedded in the network. However, after Sybils are fully embedded in the network, it is more challenging to identify Sybil nodes, especially in the real-world problems where the assumptions do not hold.

Another practical solution is maintaining directed, weighted, labeled multi-graphs as proposed in Prometheus [Kourtellis et al. 2010] instead of simple undirected graphs. The edge weights could be aggregated from user interactions, conversation duration, or shared common interests; edges are labeled according to the type of social interaction; and edge directions indicate directed interactions. The principle here is that adversaries have to maintain significant levels of interactions with legitimate users over a period of time for getting appropriate edge weights and proper edge labels, otherwise adversaries’ influence in the network is limited. Quantitative studies [Blackburn et al. 2011; Kourtellis et al. 2013] prove that a directed, weighted, labeled multi-graph is more resilient to attacks caused by manipulating the underlying social graph. While building meaningful multi-graphs is not always possible in many practical scenarios, such solution is restricted by obtaining (or collecting) rich information of graph nodes. All in all, how to design a simple yet practical system for Sybil detection is still an open question.

5.4. Private Data Disclosure

In Section 4.3, we showed how social relations are integrated in the design of P2P systems to prevent data disclosure during search, communication, and sharing. However, socially informed P2P systems also have limitations. As each peer is in his/her social circles and connected peers often share common attributes, it is possible for an attacker to use a peer’s attributes to infer his/her friends’ attributes indirectly. Specifically, the principle of homophily suggests that people with similar views are likely to be connected to each other in social networks. Using social ties to guide the design of P2P systems acts as yet another platform for implicitly exposing related users. For example, in a totalitarian political regime, it would be risky for users with dissident political views to organize their resources in a socially aware P2P systems, as they
would thus provide extra information to a potential attacker. This risk is supported by various studies. He et al. [He et al. 2006] showed that privacy information (e.g., gender, location and interests) could be indirectly released via social relations by mapping Bayesian networks to social networks. The inference accuracy is closely related to the social strength between friends. Similarly, Jurgens [Jurgens 2013] showed that even if the locations of individuals are often unknown, only a small number of initial locations can help estimating the locations of a large amount of users in the social network.

As a consequence, in socially informed P2P systems, after compromising one or a group of peers’ social information (e.g., profiles, interests or locations), knowledge-able attackers might be able to infer other users’ attributes via social relations. After gaining enough information, attackers are able to estimate searching paths, group membership information, or even rebuild the social graph to collect even more private information. To protect against such privacy disclosure, socially aware overlays may choose to include some randomization (that might limit the implicit trust) or selectively hide friendship relations by taking social strength of relations into consideration.

6. SUMMARY

Social computing supplies valuable information about users, such as personal characteristics, interactions, and declared relationships. This social information can be mined to better understand users’ future needs and actions, that can in turn be used in the design of distributed infrastructures that respond faster and more efficiently to user requests. This intuition motivated researchers to look for solutions that combine social information and peer-to-peer systems for a variety of applications and services, including distributed search, decentralized storage, security management, and decentralized online social networks.

This article provides a literature survey of the most representative approaches for using social information in the design of P2P systems in an effort to alleviate some of the known challenges, such as providing incentives for cooperation, building trust, and protecting users from malicious attacks. It lists the benefits of including social information in P2P design as proven by the literature, spanning across various applications, from typical P2P applications (such as search, distributed storage and video streaming) to decentralized online social networks.

This article identified three main benefits of social knowledge as exploited for the design of P2P solutions in the literature. First, social incentives were exploited to decrease churn in distributed, user-contributed storage, in file-sharing applications, and in video-streaming solutions. There are two basic ingredients in these systems. Users’ altruistic inclination to share resources with people with similar interests reduces churn and free riding. The inclination to socialize, when encouraged via in-application features, leads also to reduced churn and increased resources.

Second, the literature shows that search in a distributed setting can be improved by designing overlays informed by shared interests or social ties. These solutions build routing strategies based on two types of intuitions: First, a user’s friends might have similar interests, thus might have the resource the user is looking for. Here, friends may mean ties in a declared social network or users with similar declared interests in a user profile. Second, by keeping a history of previous requests, a new request can be routed towards the peers that made that request before, with the underlying assumption that the answer is still remembered.

And finally, the literature provides numerous solutions that analyze the social graph topology to infer trusted relationships and protect against malicious attacks. One line of work attempts to reduce privacy leaks by using a routing path of trusted socially connected users. Another line of work distinguishes between a normal social graph
topology and an abnormal one to identify malicious users who try to infiltrate in a social network.

At the same time, the literature shows that some of the promises of social awareness failed to deliver. The assumed trust between users connected in an online social network can be exploited via clever infiltration attacks that lead to a deceivingly "normal" topological characteristics of the social graph. Thus, many of the solutions built around specific attack strategies are defeated in practice. Because users in a social group are less guarded against each other's, breaking into one account opens a trove of data about the members of the group. And finally, the variability in social popularity leads to solutions that may work for users with an average number of connections, but not for the ones very popular or very lonely.

This article also discusses some unique challenges introduced by the use of social information and our still limited understanding of the dynamics of social networks. In the process, it highlights some open research problems, such as accurately quantifying the strength of social ties and resisting security attacks based on social graph characteristics.

REFERENCES
Qiang Cao, Michael Sirivianos, Xiaowei Yang, and Tiago Pregueiro. 2012. Aiding the detection of fake accounts in large scale social online services. In Proc. of NSDI.

ACM Computing Surveys, Vol. 9, No. 4, Article 1, Publication date: March 2016.
A Survey of Socially Aware Peer-to-Peer Systems


A Survey of Socially Aware Peer-to-Peer Systems


Alan Mislove, Krishna P Gummadi, and Peter Druschel. 2006. Exploiting Social Networks for Internet Search. In In Proceedings of the 5th Workshop on Hot Topics in Networks (HotNets'06).


ACM Computing Surveys, Vol. 9, No. 4, Article 1, Publication date: March 2016.


Antony Rowstron and Peter Druschel. 2001b. Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility. In *In Proceedings of the Eighteenth ACM Symposium on Operating Systems Principles (SOSP)*.


Received May 2014; revised September 2015; accepted Feburary 2016