SN Computer Science (2020) 1:299
https://doi.org/10.1007/s42979-020-00315-8

SURVEY ARTICLE

Peer-to-Peer-Based Social Networks: A Comprehensive Survey

Newton Masinde[1] · **Kalman Graffi**[2]

Received: 31 May 2020 / Accepted: 28 August 2020 / Published online: 11 September 2020

Abstract
The use of online social networks, such as Facebook and Twitter, has grown at a phenomenal rate. These platforms offer services that support interactions via messaging, chatting or audio/video conferencing, and also sharing of content. Most, if not all, of these platforms use centralized computing systems; therefore, the control and management of the systems lies entirely in the hands of one provider, who must be trusted to treat the data and communication traces securely. As a zero-trust alternative, peer-to-peer (P2P) technologies promise to support end-to-end communication, uncompromising access control, anonymity and resilience against censorship and massive data leaks through misused trust. The goals of this survey are threefold. First, the survey elaborates the properties of P2P-based online social networks and defines the requirements for such (zero-trust) platforms. Second, it gives an exposition of the building blocks for P2P frameworks that allow the creation of such sophisticated and demanding applications, such as user/identity management, reliable data storage, secure communication, access control and general-purpose extensibility, which are not properly addressed in other P2P surveys. As a third point, it gives a comprehensive analysis of proposed P2P-based online social network applications, frameworks and architectures by exploring the technical details, inter-dependencies and maturity of these solutions.

Keywords Computer networks · Distributed networks · Peer-to-Peer networks · Online social networks

Introduction

As a means of online user interaction, social networking has experienced unparalleled growth over the last 10 years, a fact that is attested to by the number of providers and the users registered, with the top ten providers shown in Table 1. To date, the most utilized computing model by the popular online social networks (OSNs) is still the centralized model. Several studies OSNs such as [1–5] have enumerated operational and design challenges directly or indirectly related to this computing model. These issues include, but are not limited to, seamless scaling of the network without straining of the available resources (both monetary and physical) and ability of users to control their data and maintain their privacy while using the social networks. While the first issue, challenges due to scaling, have been generally mastered by the providers, the privacy and trust issues have not been extensively considered. In the centralized service provision model, a single operator avails the social networking services and maintains availability of the services. However, the provider has access and ownership rights to all data stored by the "customers". The provider, besides striving to provide appealing services, may in most cases manifest a high interest in monetizing the data of the users, such as profile data, communication traces, all content uploaded and downloaded and all interaction traces. Consequently, the presence of (personalized) ads is an annoyance that users may have to bear with. However, worse cases of misuse of these data can be cited. Examples include the Facebook and Cambridge-Analytica scandal[1], government surveillance that infringes privacy such as the Chinese Social Credit System [6, 7], location tracking[2] [8], social media data mining

✉ Newton Masinde
newton.masinde@hhu.de
https://tsn.hhu.de

Kalman Graffi
kalman.graffi@honda-ri.de
https://www.honda-ri.de

[1] Technology of Social Networks, Heinrich Heine University, Universitätsstr. 1, 40225 Düsseldorf, Germany

[2] Honda Research Institute Europe GmbH, Carl-Legien-Strasse 30, Offenbach am Main 63073, Germany

[1] https://www.cnbc.com/2018/03/21/facebook-cambridge-analytica-scandal-everything-you-need-to-know.html.

[2] https://www.businessinsider.com/three-ways-social-media-is-tracking-you-2015-5?IR=T.

Table 1 Top 10 OSNs as of July 2019 (in millions) from https://www.statista.com

OSN Platforms	Users
Facebook	2375
YouTube	2000
WhatsApp	1600
Facebook Messenger	1300
WeChat	1112
Instagram	1000
QQ	832
QZone	572
Douyin/Tik Tok	500
Sina Weibo	465

Table 2 Top 10 federated OSNs (Nov. 2, 2019)

Project	Nodes	Users	Users per node
Mastodon	2771	2,525,434	912
Diaspora	202	705,662	3494
Pleroma	590	27,770	48
PeerTube	331	20,018	61
Juick	1	18,053	18,053
Friendica	107	14,632	137
Hubzilla	119	7045	60
Write Freely	183	5023	28
PixelFed	101	4644	46
FunkWhale	30	2659	89

for terrorist sentiments [9, 10] which may infringe on free speech, among other effects.

Besides the common centralized data model for OSNs, decentralized federated and peer-to-peer [11, 12] data models have emerged. These promise to have the feasibility of hosting billions of users and shift the data control and ownership to the user. The federated decentralized model is a break away from the centralized models, with no single owner of the network. Instead, users host parts of the OSN and federate for a complete network, becoming the bootstrap nodes for the network, and a list of available bootstrap nodes is availed to new users. There are over 30 federated social network projects listed as being active (https://the-federation.info/[3]). Table 2 gives a list of the top ten federated OSNs based on the number of users active in the network. Although federation tackles scalability concerns, the security and privacy concerns are very similar to the centralized model, as discussed in [5]. It appears that bootstrap node owners are able to access the private information of the users that connect to them, while also having control of the content stored on the servers. Although they might not have access to all data, they still can access and modify the data of tens to hundreds, maybe even thousands, of users. Therefore, here as well, data sovereignty is not in the hand of the user.

To address the shortcomings in both centralized and federated data models for OSNs, we postulate that any proposed solutions must: (a) be financially viable, (b) alleviate the security and privacy concerns of users, and (c) support dynamic system growth seamlessly. We therefore aim at the third data model, peer-to-peer (P2P), to show that it can provide the required properties of scale, security/privacy and organic growth and in addition alleviate the need for monetizing. The P2P model has the advantages of being easily scalable depending on the type of overlay chosen. Peers that join the network bring their own resources, such as a part of their available storage space, flat rate bandwidth and unused computing cycles, leading to an accumulation of "free" resources, that is by far sufficient to host a fully decentralized OSN. In addition, P2P mechanisms are self-organizing and also support zero-trust requirements, with the aim to remain functional under churn (node online dynamics), strong heterogeneity of resources and workload, the presence of malicious nodes in the network and tailored attacks. As all peers are equal, no role bearing higher rights, such as an administrator or operator, exist. In the code, the data sovereignty of the user, security and privacy can be enforced. To build a purely P2P-based OSN seems highly preferable, no operational costs, harnessing of "free" resources and thus more powerful functions, data sovereignty of the user and inability to turn the system down or to censor the content. However, building a purely P2P-based OSN is also highly challenging as key functionalities need to be considered such as routing methods, data storage, distribution and replication mechanisms, messaging-handling techniques, communication schemes and appealing apps, all while ensuring efficient, security and privacy. These requirements are altogether highly challenging to meet within a single system and have thus far not yet been discussed in great detail and with completeness as in this paper.

Identifying the Gaps

Several publications discuss distributed (or decentralized) online social networks (DOSNs), in which P2P-based solutions are also presented. Many of these articles highlight different aspects of the DOSNs such as P2P architectures for DOSNs [13, 14], DOSN design decisions (storage, access control, and interaction and signaling mechanisms) [15], security and privacy in DOSNs [16–19], as well as general surveys on DOSNs [15, 20]. They give insightful information of various aspects of DOSN in general, with P2P-based OSNs forming a small part of the discussion. However, they do not analyze the strong interdependencies stated by the requirements for a secure OSN with the (often limited) P2P

[3] https://the-federation.info/.

Table 3 Types of social media

Type	Description	Example
Social networks	Users can access combination of services such as direct messaging, content sharing (pictures, video/audio clips), audio/video chats, group interactions and so on	Facebook, LinkedIn
Social blogging networks	Users express their thoughts to other readers without worrying about a self-hosted website. Readers of the blogs can also give feedback	Tumblr, WordPress, Blogger, Medium
Microblogs	Are blogs with a limitation on the amount of content users can share in a single post. Post are in the form of text, pictures, links, short videos or other forms of web media	Twitter, Instagram, SnapChat
Media sharing networks	Are generally optimized for sharing videos and photos	YouTube, Vimeo, Pinterest
Social review networks	Users give reviews relating to different locations/experiences. The reviews help others make decision on an activity related to that reviewed location/experience	TripAdvisor, Yelp, Booking.com
Discussion networks	Platform for topical conversations. Users freely ask questions and give their opinions on a specific question	Reddit, Quora

technologies, for example, how to guarantee security in a zero-trust network while ensuring data availability or how to handle social update data, all without violating privacy, among other concerns. Therefore, analyzing parts of the system, unfortunately, does not fully describe how the singular components work in interaction with further required OSN elements.

Also, surveys that cover different component aspects of the P2P networks such as overlays [14, 21], replication [22, 23], searching and indexing [24–26], security [27–29] and so on, as well as several good reference books that cover P2P networks in general such as [30] and [31] are in existence. In our experience, the challenges in building a P2P-based OSN appear when these widely discussed mechanism need to be combined into a single working application that must also be secured. Thus, pure P2P surveys only give a very limited view on applicability of the P2P technologies for the purpose of building P2P-based OSNs.

Our Contributions

In line with the gaps identified in our study, we make the following contributions: (1) We identify key features common in all OSNs, as well the functional and non-functional requirements needed for a working OSN. (2) We give an overview of the concerns raised that have led into research on decentralized solutions, and in particular P2P-based options, for the OSNs. (3) We give a general layout for the technical requirements needed to achieve the functional and non-functional requirements, and use these technical aspects to further define a P2P framework for an OSN. (4) Based on the technical requirements, we give a study on the core P2P network mechanisms that are integral in the design of any P2P application, with considerations into the security aspects for each mechanism. (5) We present a comprehensive survey of a number of the proposed P2P-based OSNs. (6) We provide a roadmap for possible future research into P2P-based OSNs. The structure of the survey is shown in "Table of contents".

Social Networks

A social network in the classical sense involves real people interacting in the real world [32], and a more technical view is as a directed graph structure [33, 34]. However, the term "social network" as it is used today tends to refer to a combination of the real and virtual world via web-based services [35]. Social networking in view of computer-mediated communication is defined as "the use of Internet-based social media platforms to stay connected with friends, family, or peers"[4]. The types of Internet-based social media platforms with some common examples are shown in Table 3.

In this section, we present the various ways of classifying OSNs as identified in the literature. Thereafter, we consider the desired functions in the OSNs that help in meeting the user's online needs. Further, the functional and non-functional SN requirements based on the desired functions will be presented. Lastly, we discuss the principle concerns raised that have necessitated research into decentralized solutions for social networks, and in particular P2P-based solutions.

Social Network Classifications

Social networks can be classified into several ways as shown in Fig. 1 and discussed herein.

[4] https://www.investopedia.com/terms/s/social-networking.asp.

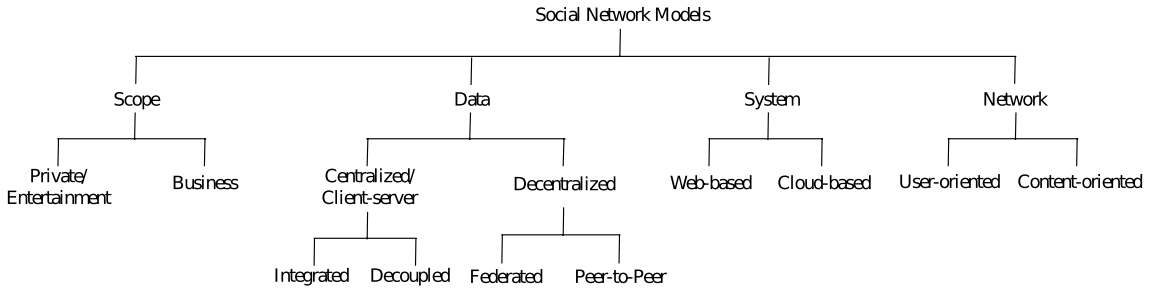


Fig. 1 Social network classifications

The Scope Model

This model considers the core activities which are either, entertainment and business [11].

(a) *Entertainment (or private* [2]) Focuses on the delivery of fun and interactive social experience to users, for example, Facebook, Flickr, MySpace and Hi5.
(b) *Business* Focuses on connecting professionals for purposes of productivity and success, for example, LinkedIn and Xing.

The Data Model

It is also called the *programming paradigm model* [12]. It classifies based on how data management is done [11], hence centralized or decentralized.

(a) *Centralized model* Data management is by a single administrative domain, which is either an integrated client–server or a decoupled client–server.

– *Integrated client–server model* Application developers utilize their own servers to manage and store social relationships and also provide required resources supporting content sharing.
– *Decoupled client–server model* Users manage their own social relationships and only core social services and social relationships which are linked to their accounts are updated centrally.

(b) *Decentralized model* Data management is distributed across multiple administrative domains, and can be either decentralized federated or peer-to-peer.

– *Decentralized federated model* It offers no centralized infrastructure from the application developers but there is reliance on existing decentralized and federated messaging system, such as Extensible Messaging and Presence Protocol (XMPP). Thus, users choose the service provider of their choice so long as they are part of the same federation.
– *Peer-to-peer (P2P) model* It is fully decentralized and users directly connect to their trusted friends and share content. Developers either write their own P2P protocol or rely on existing P2P technologies.

The System Model

This view shifts to manner in which the application servers host content distribute it. Thus, two categories are suggested [11].

(a) *Web-based scheme* Application servers that may be owned by the service provider host the web sites. The tasks of ensuring load balancing, handling failovers and request forwarding to the appropriate application server is handled directly by the service provider. In such cases, most of the services offered to users are usually free.
(b) *Cloud-based schemes* Application servers are hosted by a utility computing infrastructure such as Amazon Elastic Compute Cloud (Amazon EC2), which frees the service provider to only focus on the application only. However, this introduces additional costs to the service provider that may sometimes be passed on directly or indirectly to the users.

The Network Model

In this model, the focus is on the entity upon which relationships are formed in the system [11]. Thus, the network can be user oriented or content oriented.

(a) *User oriented (profile based)* Emphasis is on the social relationships existing between users and how they share their content within a community. Examples of SNs in this category include Facebook, MySpace and LinkedIn.
(b) *Content oriented (content based)* The focus is on the common user interests rather than the social

relationships. Examples in this category include YouTube.

Desired Features for OSNs

For maximum user experience, it is necessary to define the core features that the OSN must include. These features are in essence a summary of real requirements of the OSN based on user expectations for the system. [36] presents a comprehensive discussion on these features which we endeavor to summarize here. They include identity, conversation, sharing, presence, relationships and groups. Further reputation is given, which we discuss separately. A brief summary of these concerns follows.

(a) *Identity* The extent to which users are willing to disclose their personal details hence reveal their identity. This requires keeping the users' personal data private during registration and profile creation. Also, there is a multiplicity of ways in which user's anonymity and privacy can be compromised, particularly via their characteristic online behavior and communication patterns. Therefore, appropriate tools/mechanisms that ensure a proper balance between anonymity and privacy while online must be incorporated to increase accountability among users, prevents cyberbullying and encourages off-topic/-color comments.
(b) *Conversation* This defines how users communicate with each other. Conversations may have a defined format, such as Twitter, or be general, such as in Facebook. Group conversations should also be possible.
(c) *Sharing* Emphasizes how users exchange, distribute and receive content. The ease of sharing content securely, and guaranteeing its availability even when the content owner is offline must be carefully addressed.
(d) *Presence* Focuses on the extent to which users are aware if other users are accessible, that is, knowing where they are (virtual and/or real) and whether they are available. This is possible through provision of statuses, for example, " invisible", "available", "busy" and so on.
(e) *Relationships* Looks at the ability of users to relate to other users leading to conversations, sharing of content and listing one another as friends or fans. This then dictates the what-and-how of information exchange. A general rule is that a social media that esteems the value of identity as low also considers relationships as of low concern and vice versa.
(f) *Groups* This looks at how the users are able to form communities and sub-communities. Dunbar's number [37] is one of the proposed relationship-group metrics, which considers a cognitive limit that bounds the number of stable relationships people can have to 150 people. Two types of groups are considered. In the first group, individuals go through their connections and place their friends, followers or fans into self-created groups. In the second group, the online groups are analogous to clubs in the real world and can be opened to access (public), closed (approval is required) or secret (by invitation only).

While [36] also discusses reputation as an important element, we do not share this view. Reputation is used to emphasize how users identify the standing of other users, as well as themselves, within the social media setting. In reality, such explicit reputation ratings do not take place in OSNs. Only in online business reviews (such as restaurant ratings) are reputations found, but in contrast to social reputations, these are explicitly public, while a social reputation is mainly created for a personal use. Thus, a technical support for reputation is not considered relevant.

Design Requirements for OSNs

The design of OSNs must account for all, or most of the features described in section "Desired Features for OSNs", which are converted into system requirements. In general, two types of requirements are considered, that is, functional and non-functional requirements. We discuss them in detail in this section.

Functional Requirements

These requirements specify what the system must do to meet the core reasons for its existence [38]. They describe a useful capability provided by one or several system components [39], or a system's behavior under specific condition [40]. The following is a brief discussion of the core functionalities required for OSNs as highlighted in [41] and discussed also as service requirements in [42].

(a) *Personal storage space management* Users should control some arbitrary assigned space after creating the account and profile, allowing storage, deletion and manipulation (or editing) of user's content. Most OSNs will usually allow the reporting of the user's action in their personal space to other users that are their social contacts unless this feature is specifically disallowed by the user.
(b) *Social connection management* Users are able to define their relationship with other users by establishing/maintaining/revoking a social connection, for example, via friend lists. Users may also be able to locate and reestablish connections with lost friends and form

new relationships based on common interests such as ideologies and media content.

(c) *Social graph traversal* Also called *social traversal*. By traversing the online social graph and examining friend lists of other users, a search list can be retrieved. Using a traversal policy, traversals may be restricted to a subset of users.

(d) *Means of communication* This ensures the presence of necessary secure channels for users to interact with one another via messages in the form of text, audio, video, photos or other format. The messages can be public or private. The SN should also support both synchronous communication (such as instant messaging) and asynchronous communication.

(e) *Shared storage space interaction* Allows users to interact with each other via walls, forums or commonly shared folders. Having an access-controlled shared storage space, more sophisticated applications can be added that require data-based interaction, such as collaborative cooperation, gaming, digital workplaces and more.

(f) *Search facilities* Gives users the ability to find and connect with new contacts by exploring the social network space.

Non-functional Requirements

These are the qualities that the system must include, resulting in a system that is attractive, usable, fast, reliable or safe [38]. They describe a property/characteristic that the system shows, or a constraint that it should respect [40]. The OSN non-functional requirements that we take into consideration are: privacy requirements, security requirements [42] and metering.

(a) *Privacy* The system must provide confidentiality, ownership privacy, social interaction privacy and activity privacy.

- *Confidentiality* Availability of appropriate access control policies and encryption mechanisms to prevent information leakages.
- *Ownership privacy* Content owner should be able to make a choice of revealing ownership information to other users.
- *Social interaction privacy* User can hide the interaction patterns between him/herself and other users.
- *Activity privacy* Interactions between the user and the application suite are not exposed to the public.

(b) *Security* To provide appropriate security, the system must also include cover channel availability, authentication, data integrity and authenticity, and some also name non-repudiation.

- *Channel availability* Service is available and can be used even under malicious attacks.
- *Channel authentication* There is some form of two-way authentication between an initiator and recipient of the message.
- *Data integrity and authenticity* The system prevents modification of content or messages by any unauthorized users.
- *Non-repudiation* The sender can be traced and the interaction is documented, which might be useful in case of any malicious content/messages. If anonymity has a higher priority, non-repudiation is to be avoided.

(c) *Metering* As an added bonus to the OSN, it would be beneficial to the users as well as the system developers to receive insights into how well the system operates, the general system performance as well as system limitations/overloads if they exist. This way, mitigating solutions may be designed to ensure optimum performance. Thus, a reliable and accurate system monitoring interface can be integrated to collect system data, which can later be analyzed to gauge the health of the system.

Summary With these functional and non-functional requirements, we believe that the desired features discussed in section "Desired Features for OSNs" can be met. In Table 4, we present a feature-to-requirement mapping.

Motivation for Decentralization

The centralized computing model has matured since it was introduced into the market, making it friendly for the development, deployment and administration of network-based applications. It was, therefore, a natural choice for the OSN service providers. However, as the OSNs grew, fueled mostly by newer and better technologies and increased user-base, two key concerns have stood out: accumulated costs for centralized operations, and security and privacy concerns [13]. We discuss each concern in detail.

Accumulated Costs for Centralized Operations

These concerns arise as a direct effect of scalability, with five issues identified [4].

Table 4 OSN desired features to requirement mapping

Desired features	Functional requirements[a]						Non-functional requirements								
	PSSM	SCM	SGT	Comm	SSSI	SF	Privacy[b]				Security[c]				Metering
							Conf	OP	SIP	AP	CAva	CAuth	DIA	NR	
Identity	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Conversation	✓	✓		✓		✓	✓		✓	✓	✓	✓	✓	✓	✓
Sharing	✓	✓		✓	✓		✓		✓	✓	✓	✓	✓	✓	✓
Presence		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			✓
Relationship	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓
Groups	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

[a] *PSSM* personal storage space management, *SCM* social connection management, *SGT* social graph traversal, *Comm* means of communication, *SSSI* shared storage space interaction, *SF* search facilities

[b] *Conf* confidentiality, *OP* ownership privacy, *SIP* social interaction privacy, *AP* activity privacy

[c] *CAva* channel availability, *CAuth* channel authentication, *DIA* data integrity and authenticity, *NR* non-repudiation

(a) *Large number of highly connected users* Managing the growing social graph in real time is an ever-present, growing challenge. It arises from the tight coupling existing among different communities, and the constantly increasing user data generated.

(b) *Infrastructure issues* As the network grows, the equipment used to keep the OSN running needs regular maintenance and upgrades, and eventually newer systems need to be introduced while the older ones are phased out. Hence, ever-increasing operational costs such as replacement of failed equipment, and hiring, training and maintaining skilled staff must be accounted for. High-energy consumption by the servers and the Heating, Ventilation and Air Conditioning (HVAC) system for cooling the servers and networking equipment is also a growing concern.

(c) *Internal network traffic* The numerous interconnections point to high internal traffic, usually due to friend recommendations, real-time notifications, personalized marketing, replication, maintenance and index synchronization, which presents a scaling bottleneck.

(d) *User-generated content management and dissemination* In most of the SNs, the large portion of interactions is related to content creation and sharing. Therefore, handling and disseminating of the user-generated content (UGC) efficiently in a centralized social network create scalability challenges.

(e) *Database scalability* Centralized SNs require some form of reliable database management system to handle huge amounts of data, ensure rapid deployment of data and UGC as well as maintain heterogeneity of content. The traditional relational database management systems (RDBMSs) are unsuitable as horizontal partitioning due to relationships and dependencies among stored data is difficult [43]. Designed to guarantee consistency, RDBMSs have limited scalability and availability especially in case of network partitions [44, 45], and cannot provide required latency and scalability for SNs with clusters replicating over data centers geographically dispersed [44]. Therefore, alternative data management solutions such as Cassandra [44] and Haystack [46] by Facebook, Bigtable [47] and Megastore [48] by Google were developed.

It is evident that the operational costs in centralized system are tremendously high. Most OSN providers thus seek a way to reimburse these costs, and this is mainly achieved by selling of the accumulated user data to product advertisers and other data "analysts". This leads to the second category of concerns.

Security and Privacy Concerns

These concerns are divided into two broad categories [5, 19]:

(a) *User-related threats* Arise from disclosure of private data to other OSN users or even unregistered users. This can be intentional such as by hacking or *unintentional* due to poor/misconfigured privacy settings. Also, although most OSN platforms include options for predefined privacy settings, in most cases having fewer configured privacy settings is better since more tends to reveal a user's preferences [49].

(b) *System provider-related threats* The implicit requirement for self-disclosure during registration assuming the providers can be trusted to handle and protect private information fairly and accurately is a major loophole. By utilizing data mining techniques, implicit data can be extracted that may violate a user's privacy. This

has seen the emergence of personal data markets [50] where these personal data may be used directly by the providers or sold to third parties to be used for personalized advertising, increased band awareness, emotional manipulation [51, 52] as well as online activity surveillance [53–55]. Network-related vulnerabilities such as Sybil attacks [56], social spamming [57] and general cyberattacks on online services [58] may also occur.

These concerns must of necessity be met with workable mitigating solutions. We look at possible solutions discussed in the literature.

Mitigating the Concerns

OSNs are complex systems and it may not be possible to completely eliminate all the concerns. To tackle security and privacy concerns, suggested solutions include the use of anonymization, encryption, fine-grained privacy settings with matching access controls, and encouraging user awareness and change of behavior [19]. To handle infrastructural challenges, the use of more advanced technologies such as use of elastic cloud scaling may offer solutions. However, an all-round solution may entail making concessions on other aspects of the OSN design. It has been suggested that centralized OSNs can be improved in two ways [59]:

– extending the capabilities of provided services, or
– decentralizing the supporting infrastructures.

Extending features and service provided by OSNs has thus far been done to a great degree. However, we see that decentralizing the server-side infrastructure has not been extensively undertaken, thus meriting further exploration. In our view, decentralization will not only solve the privacy and security concerns but also the infrastructural concerns.

Decentralized Online Social Network (DOSN)

This is an OSN designed to run in a distributed environment with minimal or no central control. The distributed nature of the DOSN provides three main benefits for the user in comparison to centralized OSNs:

(a) *The provider's operational costs are ideally reduced to zero, as all resources are provided by the users*. Thus, no monetary requirement is given to sell the users data. DOSNs can be developed as open source solutions and run through the users, thus eliminating the need for a provider at all.
(b) *A better and user-oriented user privacy control can be applied*. No one needs to be trusted, as mathematically trusted access right management mechanisms can be applied and the openly available source code can be verified to correctly implement security mechanisms.
(c) *Innovative development* [60] *is encouraged as resources, in terms of communication and storage options, are widely available*. Thus, depending on the contribution of the users, several gigabytes of storage space can be offered allowing to solve typical use cases such as file synchronization, workspace sharing and messaging large files.

Two main classes of DOSNs are identifiable, web-based and P2P-based DOSNs [61].

(i) *Web-based DOSNs* They heavily rely on a distributed web server infrastructure. Thus, configuring and setting up the system is only possible by experts. In addition, there is the need for a reliable web space, else user profiles become inaccessible.
(ii) *P2P-based DOSN* This is a major step in distributed computing as participants (peers) simply install a program and cooperate with each other to realize a desired service. The P2P DOSN can be used by novice users and thus provides a broader acceptance. However, building a reliable, secure and appealing P2P OSN is only possible after addressing some challenges in providing essential services to OSN [59].

Summary Many users of OSNs have been attracted to them because of the perceived benefits. However, of late, concerns have been raised, not only by the users (social concerns) but also by the providers themselves (technical concerns). To this end, one proposed solution worth exploring to tackle these concerns is decentralization of the infrastructure. The available options include web-based and P2P-based DOSNs, with P2P-based solutions being a more viable choice. In the next section, we discuss how the essential services needed for a functional OSN are achieved by defining technical requirements for a P2P framework for an OSN.

Technical Requirements for a P2P Framework for Social Networks

To match expected user experience in a P2P-based online social network, a set of P2P component is required which provides a reliable basis for the social networking operations. The aim is to ensure overall quality of the P2P-based OSN is equivalent to, or surpasses that of, popular OSNs in terms of ability to scale, application richness, response time, security and privacy. Therefore, essential infrastructural elements need to be conceptualized and realized. DOSN

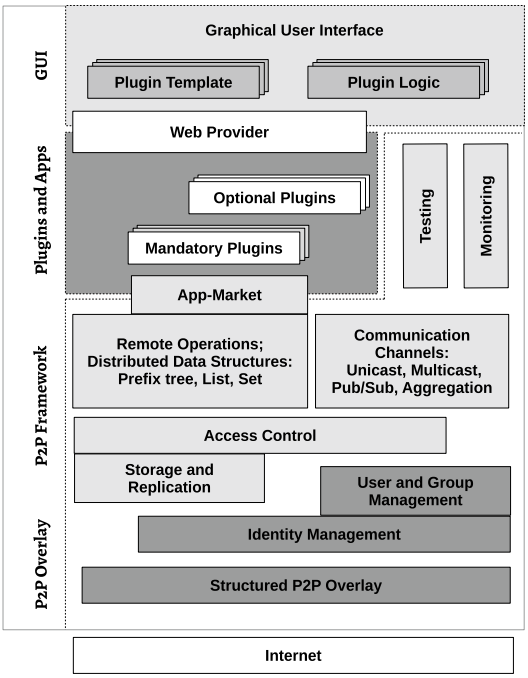


Fig. 2 A P2P architectural model for OSNs

architectural designs for P2P networks abound, such as [15, 42, 59] and [62], and in general, a fourfold architecture is to be applied, delineated as

- overlay,
- storage and communication framework,
- social networking elements, and
- graphical user interface.

These are portrayed in Fig. 2 as the P2P architectural model for OSNs. These functionality blocks are responsible for interconnecting the nodes reliably, providing rich P2P-based interaction functionality and building high-quality social networking functions with an appealing look and feel. In the following, we elaborate in detail the technical requirements defined by these functional elements.

Overlay Network

This is first layer of the framework. It defines how nodes are to be addressed and connected, to achieve the primary purpose of the social network. An overlay network requires the nodes to have a node identifier, a routing table and a list of open connections to other nodes in the network. Therefore, services that are offered at this layer include handling the addressing scheme, joining and leaving protocols, routing and maintenance protocols and ensuring that peers remain connected, operational and optimized even under strong network dynamics. We discuss each briefly here but an in-depth discussion is given in section "Overlay Structures".

Addressing of Users, Nodes and Data

Typical P2P networks are designed for file sharing. These have no concept of user identifiers as nodes simply share files with one another within the network; hence, only the data location is relevant. OSNs, on the other hand, require reliable addressing of users, data and nodes in the network, for example, to send a message to a user, to retrieve a data item by its ID or to send a data item to a specific node based on its ID. Technically, the addresses of users, data and nodes have to be consolidated and a search and/or lookup for these IDs must be supported. There is the possibility of the peers changing their physical address over several interactive sessions and with this also the location of the data they carry. Therefore, an appropriate mechanism for addressing the nodes to support searching of friends and data, as well as discovering new friends, is needed. This ensures that for existing connections, the trust links are maintained.

Routing

A core functionality of the overlay is routing of messages to users, nodes and nodes carrying specific data by their ID. This allows building of applications such as chatting apps in which messages are sent to specific user IDs, building of functionality such as replication where data is stored on a specific node determined by its nodes ID, and sending of messages to a node by a given data ID, so that the responsible node in the structured P2P overlay can implement the task described in the message on the corresponding data. The response time for the resolution of a lookup operation is crucial to ensure users do not wait for long periods for query results. For file-sharing application, a slightly longer response time is tolerated in perspective of a long download time for any file.

Security

Confidentiality, integrity and authentication during communications are essential in OSNs. Ideally, these security goals are to be integrated deep in the P2P-based OSN, thus in the overlay. Nodes should be able to engage in a secure communication directly with their first message to each other without exchanging further security related messages upfront and without using services outside of the P2P-network.

Having peers, data and users in the P2P system with individual identifiers raise the question on how to map the users to peers and how to handle device changes by users as they join the P2P network. Therefore, the network should support the use of multiple devices by a user (although not

simultaneously), and ideally also should not require any physical property (such as private key) to authenticate the users. The authentication of the user should be purely based on his/her knowledge.

Derived Requirements

These requirements are not explicitly outlined in the set of requirements but are needed to satisfy one or more of them. The derived requirements at the overlay focus mainly on the aspect of system robustness. Two points are worth consideration.

(a) *Robustness to Churn* Churn is the rate at which peers join and leave the network. It leads to broken links and invalid entries in the routing tables. Thus, when a connection is needed, nodes might discover missing contacts that need replacement, inducing a delay. Sometimes the delay and the impact on the routing table is so high that a correct state cannot be reached anymore resulting in a partitioned network. Thus, maintenance protocols are needed to keep the routing table consistent and operational.
(b) *Robustness to attacks* While the security measures guarantee authenticated and secure communication, there might be nodes with adapted protocols that aim at compromising the network and causing harm. Such attacks may also cause network partitioning, hinder correct routing hence lookup failure, or attract routing and maintenance traffic to spy on node behavior. The overlay should withstand the presence of malicious nodes aimed at sabotaging the network functions.

P2P Framework

This is the second layer. It hosts advanced functions that support the building of a P2P-based OSN which include user and group management, data storage and replication, single and group communication, monitoring and quality control, and testing. With a set of mature and reliable mechanisms, independent of the later use cases, it is possible to build rich applications that are unaware of the underlying P2P functions. We discuss these functions in greater detail in the following.

Single Data Storage

Unstructured overlays do not have a defined data storage location and, any active node is utilized. Thus, data retrieval in the network typically accomplished by flooding, which often generated high traffic, has the risk of missing data even though it is in the network. On the other hand, structured overlays provide a routing mechanism based on data ID and the node responsible for that data ID responds to a request. Thus, data are always found and retrieved if they are in the network. The consistency of the data in the network is essential so that nodes interested in a data item receive the most recently updated copy. The single data storage and retrieval functionality is one of the basic building blocks for P2P-based OSNs and is discussed in section "Storage Techniques and Redundancy".

Reliable Data Redundancy

Data availability guarantee requires efficient data redundancy mechanisms such as replication and erasure codes. Erasure codes are more suitable for static data such as in file-sharing applications, and replication is more suited for OSN as it allows for regular data updates as the data changes. Data replication must be done in a reliable and consistent manner to prevent data loss due to churn. Protocols should be in place to monitor the number of network copies of a data item and new replicas created in case of fewer copies. The item replicas should be high enough to guarantee data availability, and few enough to ensure faster replica updates and minimal loading on the nodes due to requests. Replication mechanisms should incorporate and maintain the security requirements on the data and its replicas. Optional requirements for data locality and trust-based replication might be considered. The data redundancy mechanism are discussed in section "Storage Techniques and Redundancy".

Data Access Control for Users and Groups

When data are created, the author of the data stores that in the network, and modifications should be allowed only for the initial author of the data. Read only access may be granted to a list of privileged users for the data and their replicas while all other users, including the replica nodes, should not access the data. These rights apply even when the data owner is offline. The set of users with privileged access is not explicitly required to be the list of "friends" in an OSN. The reason is that the list of "friends" is typically not identical with the list of trusted nodes and it may also not be the intent of the users to provide access rights for each data item to only this set of users. The P2P framework must, therefore, support definitions for a set of read-enabled users for each data item. To scale the security and access control protocols, it is reasonable to introduce groups or roles. Single users can be added to groups, where the members have the same access rights. This makes it easy to assign the read or write access to a specific group, such as thousands of users at once, instead of assigning them the access rights individually.

Search for Data and Users

The method used for locating data is based on the overlay structure. Unstructured overlays search for data using flooding techniques. Structured overlays, however, support lookup, which assumes that the user and data ID are known. The OSN, on the other side, interconnects users and allows them to create and access data items. Users and data items, with corresponding access permissions, should be searchable based on keywords. Thus, users could be searched based on their location and interests, and data could be searched based on its content and tags. Generally, the overlay topology provides for either an indexing support or for better flooding approaches, but not both simultaneously. In special cases, additional indexing structures may be considered as the topology may not support advanced searching methods, such as multidimensional queries. An overview on resource lookup and search approaches is discussed extensively in section "Overlay Function: Search and Lookup Mechanisms".

Direct Communication, Multicast and Publish/Subscribe

One core function of an OSN is the 1-to-1 or direct communication between nodes, either asynchronously, when typically the storage is used, or synchronously, when direct messages are sent between peers. This supports implementing direct chatting and messaging. Multicast communication makes it possible for multiple parties to communicate at once. This can be either one-to-many (1-to-M), many-to-many (N-to-M) or many-to-one (N-to-1) communication. Multicast communication has made it possible to implement multicast event notification systems called publish/subscribe systems. In all communication, security considerations must be put in place. These communication options are discussed in section "Communication: Unicast, Multicast andPublish/Subscribe".

Optional Derived Requirements

Two additional optional requirements are identified, which may prove useful.

(a) *Secure distributed data structures* The storage option thus far presented is suitable for storing simple data items such as profile data and small files only. However, this finds its limitations when storing more complex data structures such as a list of albums containing hundreds of photos, which would overload a single peer. In such cases, it is advisable to distribute the data by storing the various connected data items on different nodes which distributes the load and allows for parallelized data retrieval. Distributed data structures (DDSs), such as proposed in [63–65], provide a means of creating and accessing the data items tailored for OSNs. Typical use cases are unordered sets (such as for photos in an album), ordered lists (such as for list of comments on the user's wall) as well as tree structures (such as for folder structures for a collaboration space). DDSs should include fine-grained access options such as adding entries to a set or list or rearranging the tree structure without replacing the whole data structure. While single data items are important, the use of DDSs in the P2P network is more convenient as it suits the needs of an OSN. In section "Advanced Storage: Distributed Data Structures" we discuss the options to implement distributed data structures.

(b) *Distributed quality monitoring and control loop* Information on the performance of the OSN can help identify situations of bad performance and give direction on how to control the performance. A monitoring module integrated into the system may be key in obtaining information on performance of the P2P network and OSN in form of aggregated statistics. Example of such statistics include total, average, minimum and maximum retrieval times, network traffic, storage load and also give a standard deviation on this. The availability of such complete, aggregated and timely information, allows users, potential operators and the system itself keep track of the performance of the P2P networks and OSN. With the monitoring information a distributed control loop can be implemented in the P2P network, which can be auto-tuned to ensure that performance goals are reached and maintained. The discussion on monitoring is done in section "Services: Monitoring and Management".

OSN Plugins and the Graphical User Interface

The overlay and framework layers discussed thus far are quite general. They are not limited to the use case of P2P-based OSNs as they do not provide dedicated functions such as profile handling, friend lists or photo albums. For the implementation of OSN specific functionality, ideally, a modular system is used which utilizes the functions offered by the P2P framework and P2P overlay and allows the programmer to add new social plugins easily.

Mandatory OSN Plugins

OSNs at a minimum require a personal storage space (profile), and social contacts (friends). In combination with the opportunity for social graph traversal, nodes can present themselves, connect to friends and browse these friends' profiles. In addition to this, users expect means of communication (messaging, a wall for conversation) as well as a

basic set of shared storage space, where they can present photo albums and link each other. The main interaction will take place in direct messaging communication, through comments and updates on the wall of the users as well as through photo uploads and comments. With the P2P framework underneath the plugin layer a standard quality is provided and a foundational security concept established, with convenient storage and communication functions. But this also defines a specific set of requirements that all plugins must follow. Having an encrypted profile at first glance hinders it from being recommended by specific plugins.

Optional OSN Plugins

While the behavior of mandatory plugins is quite fixed and predictable, additional functionality can be added through the optional plugins, such as a profile recommendation service. This allows for opt-in services available only for those that opted in (that is, by installing the optional plugin) such as profile recommendation services. The downside with inclusion of such plugins is the risk of opening services to untrusted third-parties, which may expose all other plugins to the privacy problem. The trend nowadays is towards inclusion of machine learning-driven plugins into OSN. From a user's perspective, the only data that should be breached are those marked as publicly accessible, thus of low value. From a data analyst point of view, data provided in an OSN are not to be trusted, especially if the OSN aims for anonymity and privacy. Websites with such characteristics, such as 4chan.org[5], even claim that the information on them is not to be trusted. Thus, the value of data obtained from such OSNs is to be evaluated. For the P2P environment, the peers' choice of enabling a third party application should not affect other peers. Using the functions of the P2P framework no new plugins should have the ability to circumvent the security features of the P2P framework as all security and performance related information are handled by the P2P framework, while the OSN plugins only utilize the framework and are bound to its policies.

Graphical User Interface (GUI)

Most popular OSNs are accessible via browsers. Consequently, it is desirable for the GUI to be browser-based, supporting the newest features of HTML5. It should be capable of providing an overview on the OSN functions offered, and easily integrate OSN apps added later. The OSN apps added should be restricted from tampering with the presentation of other apps.

[5] https://4chan.org/.

Having elaborated the technical requirements for a P2P framework for social networks leads us to the analysis of what solutions are already available in P2P literature. In the next section, we give a overview on the core P2P building blocks.

Peer-to-Peer Networks

A P2P network (or system) is a virtual, self-organized network formed over the existing physical network by introduction of specialized protocols that allow heterogeneous nodes to autonomously interact and share resources. General characteristics of P2P networks are resource sharing, interconnection between peers, decentralization, self-organization, stability and fault resilience, scalability, anonymity and shared cost of ownership [66]. To realize these characteristics, P2P network must incorporate certain key essential services within the architecture. These essential services then provide a foundation for designing suitable application such as OSNs. In this section, we look at the core services that P2P networks should at a minimum incorporate, with a focus on services needed to implement an P2P-based OSN. The structure of the study is shown in Fig. 3.

Overlay Structures

The overlay network is a collection of logical links that connect nodes to the application layer [67], and is in essence a set of protocols that run by mimicking the physical network's behavior [68]. The logical links may involve one or more actual physical links between participating nodes and the construction of the overlay itself is generally not dependent on the underlying layers although the information from these layers (such as network delays and physical proximity) is used in the operation of the P2P network [67]. Different ways of classifying P2P networks exist and they depend on level of broadness adopted by an author [69] which is dictated by the evolutionary process of P2P architecture [70]. Traditionally, classification was based on routing management; hence, structured and unstructured P2P overlays. However, this method is limiting in its view because it does not take into account many recent changes observed in overlay designs. In our view, the classification given by [21, 67] covers more ground, which classifies them as single-overlay, multi-overlay or bio-inspired P2P networks, and takes into consideration other factors in affecting the overlays.

Single-Overlay P2P Networks

This class of P2P networks form a single overlay, and hence is the traditional form of P2P networks. It is further divided based on two levels: type of index and type of

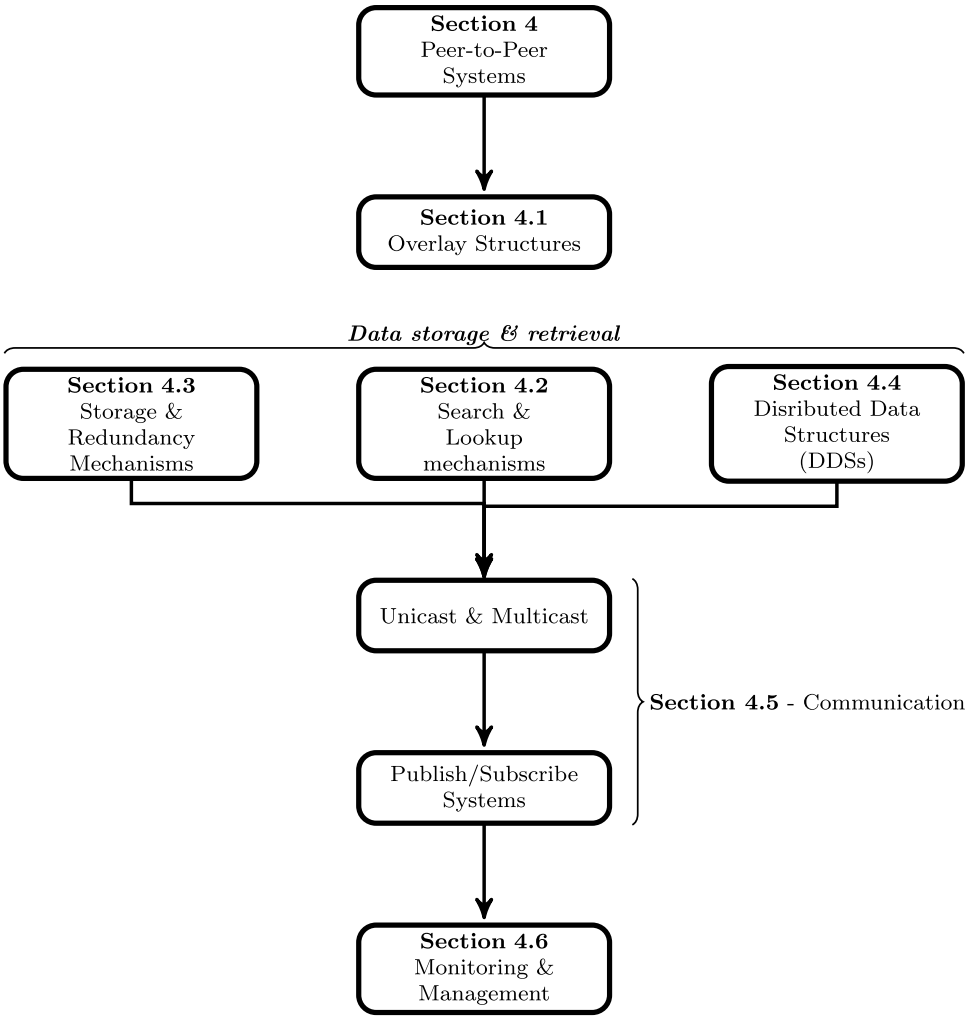


Fig. 3 Structure of study on P2P technology

Table 5 Single-layer overlays

Indexing mechanism	Network structure	Example(s)
Centralized	Structured	None
	Unstructured	Napster, BitTorrent [85]
	Structured + unstructured	Trackerless BitTorrent
Distributed	Structured	DHT-based P2P networks, e.g. Pastry [86], Tapestry [87], Chord [88], Kademlia [89]
	Unstructured	Gnutella v0.4, Freenet [90, 91]
	Structured + unstructured	None
Hybrid	Structured	P2PSIP [92]
	Unstructured	JXTA [93], Gnutella v0.6, FastTrack/KaZaA [94, 95]
	Structured + unstructured	Skype

structure. Table 5 gives examples of P2P networks in this group.

Type of index Indexing mechanism are useful for locating nodes or sharing resources. The mechanisms may be *centralized*, *distributed* or *hybrid*. An in-depth discussion on the indexing mechanism is given in section "Overlay Function: Search and Lookup Mechanisms".

Type of structure The P2P network may be unstructured, structured or a combination of the two. Each type of index

Table 6 Multi-layer overlays

Classification	Example(s)
Vertical	NICE [96], HIERAS [97],
Horizontal	Structella [98], Cyclone [74]

mechanisms can further be broken down into these three groups.

(a) *Unstructured overlays* The overlay has flexible node relationships and lookup operations [21], and nodes rely only on adjacent nodes for message delivery [68]. The overlays supports churn with a degree of flexibility, and node failure does not adversely affect searching [25]. Search is based on broadcasting (flooding) schemes, and implementing a Time-To-Live (TTL) value for query messages helps reduce network load [71]. These overlays are unsuitable for exact match queries, but are quite efficient for replicated data while also supporting keyword searches but at a high cost to bandwidth.

(b) *Structured overlays* These have a tightly controlled topology maintained via a network graph, with resources placed in a deterministic fashion using distributed hash tables (DHTs) [21], and nodes cooperatively maintain routing information about how to reach all nodes in the overlay [68]. Thus, structured overlays support key-based routing protocols, and can only handle exact match queries with high precision but are not designed for keyword searches.

(c) *Combination of structured and unstructured* Such P2P systems rely on hybrid indexing schemes, hence have super-nodes [67]. These supernodes are connected in a structured network formation and the communication existing between the regular nodes is unstructured.

Multi-overlay P2P Networks

The overlay is constituted of several interconnected overlays forming a single functional entity [67]. This may provide a means to solve the totality of issues regarding pervasive networks [72]. The need for multiple P2P overlays is probably a consequence of perceived benefits realized due to its use in virtualization [73], which enables the utilization of the same physical resources by many different applications [21]. Classification of the multi-overlay schemes is determined by considerations towards temporal synergies and dynamicity, as well as communication, state and service interactions [21], hence are vertical and horizontal. Examples of multi-layer overlays are in Table 6.

Table 7 Bio-inspired overlays

Inspiration	Example(s)
Ant colony optimization	AntCAN [99], P2PSI [100], BlatAnt [101], Self-chord [102], AntOM [103], Self-CAN [104]
Bee foraging	Antares [105], P2PBA [106],
Neurons	SCAN [107]
Fungus	Myconet [108]

(a) *Vertical multi-layer overlay* Several overlays are clustered one on top of the other, with each layer being independent structured P2P overlay network, and a higher level overlay exploits functionalities of the lower level overlay [21]. In most cases, these layers are usually DHT based. Communication occurs via *gateway nodes* responsible for message routing between two vertically adjacent nodes.

(b) *Horizontal multi-layer overlays* Here the parallel operations of overlays are the focus [21]. Multiple overlay networks, each referred to as a leaf, are joined to form a single DHT-based P2P network, with possible connections existing between the leaf overlays. The overall function of the resultant DHT-based P2P network is to optimize routing and maintain the conceptual hierarchy of the leaf overlays [74]. Unlike vertical multi-overlay networks, there are no gateway nodes, but the leaf overlays connect by carefully selected links to ensure a limited number of total links per node.

Bio-inspired P2P Networks

These overlays are a result constructing P2P overlays networks using algorithms and techniques that are inspired by naturally occurring biological phenomena. These bio-inspired solutions are characterized as being highly adaptive and reactive, having support of heterogeneity, distributed operations, resilience to component failure and can self-organize [75]. Therefore, bio-inspired approaches have been taunted as a possible alternative for managing P2P overlay networks having been proven as an effective solution in the computer network domain [76]. Existing solutions are mostly based on the collective behavior of ant colonies or bees called swarm intelligence, but other approaches have also been studied such as biological neurons and fungi growth. Examples of proposed overlays are listed in Table 7.

Other Overlay Considerations

Several theories have been taken into consideration for the systematic development of overlays with desired properties such as locality awareness, anonymity, mobility and other

features [77]. Examples include Geodemlia [78] and LobSter [79] which are location-aware overlays, FRoDO [80] which supports anonymous communications.

Security Discussion: Overlay

P2P networks are embedded into the TCP/IP protocol suite and, therefore, security issues affecting networks can also be found in P2P networks. Threats that affect the overlay have mainly to do with denial of essential services. These include denial-of-service (DoS) and distributed denial-of-service (DDoS), man-in-the-middle attacks and routing attacks such as eclipse attack [81], wrong routing forwards (attrition attacks [82]), identity theft [83] and churn attacks [84]. To mitigate these types of attacks, the solutions must, therefore, consider securing the communication. These solutions are discussed in depth in section "Communication: Unicast, Multicast andPublish/Subscribe" where provision of communication channels is considered in depth.

Overlays for P2P social networks In SNs, each individual data item is relevant and should be easily retrievable in the shortest time possible. Thus, only single-layer overlay, structured networks seem to be suitable. It is more important to be able to retrieve rare data items in at most $O(logN)$, while tolerating an expensive keyword-based search which may require development of other mechanisms to support it, than to have a cost-efficient searching, as availed by unstructured networks, but high costs in locating and retrieving the profile data of connected friends. Another essential aspect is the ability to change data, such as profiles. The network should support updating of all copies of such data. In structured overlays, only the node responsible for the data's identifier needs to be contacted, which is feasible, in contrast to searching through the entire network for copies as in an unstructured overlay network. Although multi-layer overlay and bio-inspired overlay P2P networks may seem promising as solutions for social networks, it may require much more effort to implement needed mechanisms for social network services to match the centralized SNs.

Overlay Function: Search and Lookup Mechanisms

At the core of solving the P2P search/lookup challenge is the development of appropriate and agile indexing mechanisms and querying mechanisms for efficient information retrieval. This makes the search techniques dependable and adaptable to the changing network. P2P indexes can be classified as follows:

(a) *Local indexes* Each peer maintains an index for its own data or objects only, as seen in the first Gnutella. They support rich queries along with simple key lookups, and query flooding is used for global data search. However, the use of local indexes in a large and growing network becomes inefficient.

(b) *Centralized indexes* They depend on a single server to maintain data references to the many peers, such as Napster[6]. However, a centralized index for a P2P network reintroduces the problems of centralized systems, and hence is discouraged for a fully decentralized application system.

(c) *Distributed indexes* Nodes maintain information for a part of the identifier space and a systematic routing table to reach nodes responsible for other parts of the identifier space. The index can be semantic or semantic-free indexes.

– *Semantic indexes* Found in most unstructured P2P networks. They utilize human readable indexes hence *semantic*. Further discussion is in section "Semantic Mechanisms for Searching". However, they do not support persistent object references and prevent contention free references [109].
– *Semantic free* Indexing supports content-based referencing, as DHTs are used to enable object location using persistent keys in a high-churn network [110]. Detailed discussion is in section "Semantic-Free Mechanism for Lookup".

(d) *Hybrid indexes* They utilize the best of both worlds, that is, they combine two or more types of single-layer overlays to achieve effective indexing. In most cases, the structured network consists of nodes that perform the indexing called the super-nodes, while other nodes are maintained in an unstructured format. A result of these hybrid indexes is the development of multidimensional indexing mechanisms to tackle the problem of rich text and multidimensional data searching.

Figure 4 is an overview of the indexing mechanisms used in resource discovery. In the following, we describe the semantic-free, semantic and multidimensional indexing mechanisms.

Semantic-Free Mechanism for Lookup

Mechanisms that support some form of key-based routing (KBR) will generally offer semantic-free indexing. These methods cover the use of DHTs, tree-based mechanism and skip lists/skip graphs.

[6] Originally available on http://www.napster.com.

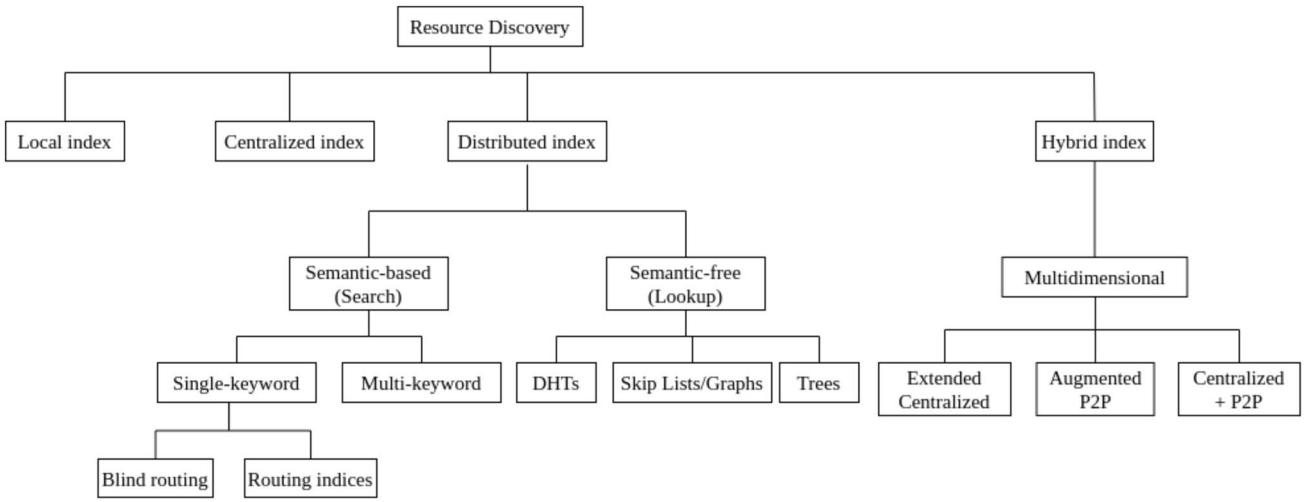


Fig. 4 Resource discovery using indexes

(a) *DHTs* Traditional DHTs such as Chord [88], Pastry [86] or Tapestry [111] use a routing table of the size of $\Theta(logN)$ neighbors which ensures routing to the responsible peers within $\Theta(logN)$ hops. However, hashing as used in the DHT based indexes destroys the locality of data, particularly data item's closeness users (content locality) and prevents global routing if querying and answering nodes are within the same locality (path locality) [112].

(b) *Skip list & skip graphs* The Skip List [113]-based overlays, Skip graphs [114], SkipNet [112] and HSkip+ [115] can be used instead of DHT-based overlays to alleviate the problem of locality. Skip Lists ensure balance through probabilistic balancing during insert and delete operations with every node having averagely $\Theta(logN)$ neighbors. Unlike DHTs, Skip graphs support prefix searches, proximity searches [116] as well as location-sensitive name searches [24].

(c) *Tree-based overlays* They also offer logarithmic key-dependent searching such as BATON [117]. In BATON, each of the peers in the network maintains a node of the tree. Node links to other nodes may be parent links, children links, adjacent links or neighbor links. Each node (leaf or internal) is assigned and manages a range of values that should be greater than those of the left adjacent node but smaller than the right adjacent node. Searching is then performed based on the range in which the value falls either towards the right of the tree if value is greater or towards the left if the value is smaller.

Semantic Mechanisms for Searching

Semantic-free mechanisms guarantee that a key can be found if it exists in the network because they rely on the structure of the overlay network but they do not show the relationships existing among objects. However, semantic indexes do reveal these relationships, but they do not guarantee locating rare items as they rely on heuristics. Semantic indexing is mainly the domain of unstructured P2P networks. For the unstructured networks, performing exact searches can only be realized through an exhaustive all node contacting flood, as a global index cannot be constructed [118]. However, keyword searches can be undertaken quite easily as this involves querying for a single keyword or several keywords. The keyword search mechanisms can be broken down into single- and multi-keyword searches.

(a) *Single-keyword searches* There are basically two ways that this can be realized, blind routing and routing indices [118].

– *Blind routing* techniques do not take into account resource distribution thus are likely to get wider coverage. However, they generate high network traffic loads. These searches are achieved using flooding techniques. The flooding techniques can be pure flooding, flooding across hops, TTL limit-based flooding such as expanding ring search, and probabilistic limit-based flooding such as random walks [119].

– *Routing indices approaches* build indices to help guide forwarding queries. The indices can be peer-content based such as naive routing [120], or peer-query based.

(b) *Multi-keyword search* This is a great motivation for making searching in the systems much easier and faster. However, [118] suggests three ways that can be looked into to solve the problem of multi-keyword searches by introducing modifications to the single-keyword search methods. These are

– performing single-keyword search and then merging the results;
– assumption that the multiple keywords are a single query; or
– viewing the problem from a query routing process perspective.

Multidimensional Indexing Mechanism

In addition to supporting simple searching or key-based lookup queries, a highly desirable quality in P2P systems is support for complex rich text queries [121] and multi-dimensional data [26, 122]. Multidimensional indexing (MI) allows users to perform querying efficiently in cases of multi-dimensional data such as Geo-spatial data. This can be achieved using multidimensional indexing structures such as Skip Lists and Skip Graphs [122]. In [26], three classes of MI are discussed:

– P2P-based MI methods that are extensions of centralized MIs that have been decentralized,
– P2P-based systems that have been augmented to achieve MI, and
– combining of centralized MI and P2P-based systems.

This way, it is possible to run more advanced query types such as aggregation queries, multi-attribute queries, join queries, *k*-nearest neighbor query and range query.

Security Discussion: Search and Lookup Corruption

One of the main problems in searching and lookup is holes arising from attacks that affect data availability causing inconsistencies in routing, storage or resource lookup. The following are some of the common ones:

– *Content availability depletion* [123] Arise from attacks targeting content availability which make finding a needed resource difficult. They are accomplished by poisoning or pollution of the replicated resources, lowering the relative availability of usable content in the network.
– *Random decoy injection* [124] This is the process of poisoning by the insertion/injection of large quantities of decoys in the network resulting in poor ranking of usable files from the search results.
– *Replicated decoy injection* [124] Occurs when numerous replicas of the same decoy are injected into the network, resulting in higher rankings for the injected decoy in the search results. This attack, however, can be easily detected.
– *Replication transient decoy injection* [124] This is an alternative that overcomes detection, for example, by a reputation system, by frequently replacing the replicated decoys injected in the network
– *Routing table poisoning attacks* This affects structured networks, because of the reliance on the routing tables to perform lookups.

Resource discovery in P2P-based OSN For an OSN capable of finding other resources, the preferred option is incorporating lookup (semantic-free indexing) mechanisms as they offer efficient search and retrieval. However, in cases where the P2P network is to be designed for file sharing applications where there is need to find the closest neighbor that is available to share a file, then searching (semantic indexing) mechanisms would suffice.

Storage Techniques and Redundancy

Designing a reliable storage mechanism is aimed at ensuring data availability and in P2P systems and several proposal exist. DHT-based mechanisms include PAST [125] on Pastry, Cooperative File System (CFS) [126] on Chord and OceanStore [127] on Tapestry [87]. Most unstructured P2P networks are essentially storage networks as they were designed for file sharing purposes such as Freenet, FastTrack and BitTorrent. Irrespective of the overlay upon which the storage technique has been designed on, the most important consideration is guarantee on data availability. In most distributed systems, this is achieved through the inclusion of data redundancy mechanisms which utilize replication and/or erasure codes [128]. We discuss how this is achieved.

Data Availability Through Replication

In replication, copies (full or partial) of the data are distributed to chosen peers in the network to guarantee fault tolerance within a distributed system. The term replica refers to copies of the replicated objects. Data replication in a distributes system ensures high availability, reliability and fault tolerance, scalability, increased performance and presence of "fail-safe" infrastructures [23]. Caching is very similar to replication and is aimed at releasing loads experienced at particular hot spots and decreasing file query and retrieval

latency. Caching is usually performed near the file owners or the file requestors or along a query path from a requestor to an owner [129]. However, caching is done opportunistically and is uncoordinated, leaving no information about where caches exist in case there arises a need to update the cached data items. In replication the storage points of the data copies are and remain well known so that *all* copies of the data item can be found and addressed, for example, to update them. When utilizing replication, the replica control mechanisms and data replication techniques have to be considered. These are discussed next.

Replica control mechanisms To maintain file consistency during replication, update management using suitable replica control mechanisms is necessary. These mechanisms are classified based on replication point, update propagation method or replica distribution [23, 130, 131].

(a) *Replication point* Protocols are either single-master or multi-master with the option to perform push-based or pull-based updates to the slaves.
(b) *Update propagation* The mechanisms can be either synchronous or asynchronous. Asynchronous propagation mechanisms are either pessimistic or optimistic.

– *Pessimistic propagation* Ensure single-copy consistency and prevent replica access unless it is up to date. This suffices for small network but fails in a globally distributed networks, such as the Internet which is generally slow and unreliable; hence, pessimistic algorithms would scale poorly in such contexts, coupled with the fact that some online human activities necessitate asynchronous data sharing [132].
– *Optimistic propagation* Allows sharing of data efficiently in wide area and mobile environments hence is preferred for globally distributed networks. However, optimistic replication faces challenges due to divergent replicas and concurrent update conflicts, thus is unsuitable for systems rarely experiencing conflicts and having high tolerance to data inconsistencies [23], such as most P2P applications.

(c) *Replica distribution* This can be performed as full replication, where each site stores a copy of the shared objects, or partial replication, in which case sites only store a subset of the shared objects, thus sites store different replica objects which save space overall.

Data replication techniques These are divided into three groups: site selection techniques, file granularity techniques and replica distribution techniques.

(a) *Site selection techniques* In unstructured networks, solutions include owner replication, path replication and random replication which are discussed in [133] as well as HighlyUpFirst replication and HighlyAvailableFirst replication which are discussed in [134]. Structured networks site selection techniques include successor replication, multiple hash functions, correlated hashing and symmetric replication as discussed in [135].
(b) *File granularity techniques* Include full file replication, block level replication and erasure code replication [136, 137].
(c) *Replica distribution techniques* These include uniform replication, proportional replication and square-root replication discussed in [133], Pull-then-Push replication [138] and optimal content replication [139].

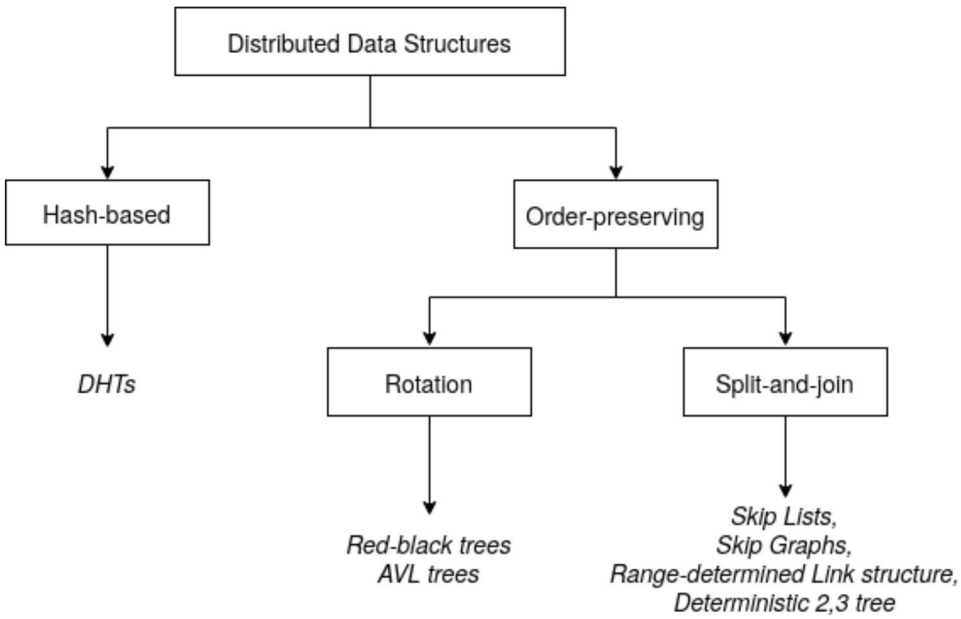
Most P2P systems utilize one or more of these replication strategies in combination to achieve some form of reliable replication. The second aspect of achieving data availability is based on the use of erasure codes which we discuss next.

Data Availability Through Erasure Coding

Systems that rely solely on replication generally achieve high availability only with high space overhead [140]. Error Correcting Codes (ECCs) [141] have been used to prevent information loss experienced during transmission of a data stream. Erasure codes [142] are a special class of ECCs which are used if a system can differentiate in advance the missing or corrupted encoded data segments. Generally, a data block b of size S_b is first broken into m equal-sized fragments of size $S_f = S_b/m$ which are then coded into n blocks by adding r redundancy blocks in a way that it is possible to reconstruct b from any subset of m blocks among the $m + r$ $(= n)$ fragments. The original blocks are referred to as data blocks and the coded blocks as check blocks. The ratio n/m is called the stretch factor and m/n is the useful space. The main idea behind the erasure codes is that given any choice for m blocks, it is possible to reconstruct the original data. Replication can be seen as a special case of erasure codes where $m = 1$ [140]. The most commonly used erasure coding techniques include Reed-Solomon codes [143], Regenerating codes [144], and Hierarchical codes [145].

In addition to selecting an appropriate coding technique, another challenge is maintaining a minimum number of data fragments in the network for durable long-term storage in spite of failures by ensuring proper fragment placement. It has been shown that the choice of fragment placement has an impact on system performance [128]. Therefore, not only is the coding technique important, but also the replica placement policy. Examples of placement policies include

Fig. 5 Classification of DDSs



global & random policy, chain policy and Buddy (or RAID) policy [128].

Replication vs. erasure coding When retrieving a replicated data item, it is sufficient to contact one single peer which has the replicated data item. This can be done through one lookup. On the other hand, with erasure codes, at least m nodes are required so as to retrieve the complete data item which takes more time and generated more network traffic. Much more severe is the case when data updating is to be performed. With replication, there is typically only a handful of nodes to be contacted, that is, those which are the replica holders, to update the data item. In the case of erasure codes, all m nodes have to be contacted, which is typically a much higher number. Thus, replication is faster in terms of the replication, update and retrieval process, but requires drastically more space. Hence, as noted in [146], the use of replication is preferred for OSNs as the data are frequently updated, while the use of erasure codes is a better choice in systems handling large static data for archival/backup purposes despite the fact that they are very space efficient.

Advanced Storage: Distributed Data Structures

A distributed data structure (DDS) is a data structure that has been designed to work in a distributed environment as a self managing storage layer. The DDS consists of

– a *data organization scheme* specifying a collection of local data structures acting as stores of data items by copying them to various sites in the network, and
– a *set of distributed access protocols* supporting the processors in issuing of modification and query instructions to the network and getting appropriate responses [147].

DDSs have strictly defined consistency models (operations on its elements are atomic), support a single, logical data item view for clients despite replication (one-copy equivalence), and they use two-phase commits for replication coherence [148]. They can be grouped into two classes, hash-based and order-preserving as shown in Fig. 5. This is based on the fact that insertions and retrievals in the DDS is either based on hashing or keys.

Hash-Based Schemes

The basis of the DDSs in this class is the hash table, a data structure used to map keys to values, store these $\langle key, value \rangle$ pairs, and retrieve the values using the provided keys. A hash table consists of two parts, an array which is a table that stores data to be searched and a mapping function known as the hash function that maps data item keys onto the integer space that defines the indices of the array. The most commonly used hash-based scheme in DDSs is the distributed hash table.

A Distributed Hash Table (DHT) provides a decentralized mechanism for associating hashed key values to some stored data item, hence supporting hash table functions. As DHTs are utilized in structured overlay networks, the interconnection that exists between the nodes supports the efficient delivery of the key lookup and insertion requests from the requestor to the node storing the key. Generally, replication

of stored items as well as maintenance of the $\langle key, value \rangle$ pairs within the overlay network bolsters robustness against node churn.

Advantages of DHTs DHTs offer several advantages such as support decentralization of operations, scalability, load balancing, system churning and fast and efficient routing as well as data retrieval [149]. With few exceptions, hash-based schemes require $\Theta(logN)$ links per node and $O(logN)$ hops to perform routing. Viceroy and Koorde gives $O(1)$ links per node while maintaining the $O(logN)$ hops [150].

Disadvantages of DHTs There are several shortcomings of hash-based schemes. They have the tendency of destroying the key ordering, resulting is scattering of the data in the array due to the hash function. This prevents performing range queries and possibly stored data items may be located far from its frequent users [151]. They include high cost stabilization, maintenance and recovery protocols that operate in the background to mitigate against failures hence achieving system state consistency. Finally, hash-based structures are not able to self-organize.

Examples Popular DHT implementations include Chord [88], Tapestry [87], Pastry [86], CAN [152] and Viceroy [153]. An alternative approach observed in literature is layering of range query schemes over the DHT systems. In these systems, the DHT is the routing substrate and the upper layer handles the order-based queries. Examples of such include P-Tree [154] which uses a B^+-tree layer on top of Chord, and Squid [155] which uses Hilbert space filling curves on top of Chord, which then support multidimensional indexing mechanism (section "Overlay Function: Search and Lookup Mechanisms").

Order-Based Schemes

These were proposed to offer solutions to some of the problems associated with DHT implementations. The P2P systems are then constructed based on tries or other types of search trees that provide distributed search tree (DST) capabilities, which supports order-based searches [151]. They therefore rely on order-preserving structures, in specific, balanced search trees, hence directly supporting search queries that depend on the key order, for example, range searches. The DST schemes are distinguished into two groups based on the balancing technique the trees applies: *rotation based* and *split-and-join based* schemes [151].

(a) *Rotation order-based schemes* Utilize the red-black tree and AVL tree structures to maintain order. When a node joins or leaves the system, a restructuring operation called a rotation is performed which will usually affect several nodes in the tree structure. This introduces a non-local operation as rotations cascade. Therefore concurrent insertions and deletion of nodes may cause inconsistencies to arise. A solution may be to implement and utilize some form of mutual exclusion mechanism which may impair the scalability of the entire system. Examples of P2P systems that have been implemented using these schemes include BATON [117] which is based on the AVL tree.

(b) *Split-and-join order-based schemes* are based on the B-tree and its derivatives such as the 2, 3 tree, skip-tree and skip lists. They maintain order in the system by performing restructuring through a split and join operation which tend to be local as opposed rotation-based schemes. Therefore, only a very minimal set of nodes in the tree are accessed and balancing is achieved via randomization. Consequently, these schemes are highly scalable as they do not need the mutual exclusion mechanism. Example implementations include Skip Graph [114] and Skip Tree Graph [151] based on the Skip List, Skip web [156] based on a range-determined Link structure, and Hyperring [157] based on a deterministic 2,3 tree.

Advantages of order-based scheme The order-based schemes have some itinerant advantages. They show high resilience to node failures even in cases where adversaries target specific node sets. An advantage they have over hash-based schemes is the provision of content and path locality. In addition to this, the ordering of the data based on their keys allows the use of range queries as well as other types of multidimensional queries for searching data items. Moreover, these schemes have in-built structural repair mechanisms that ensure the order is maintained, which in some cases may simply involve rearranging of the pointers.

Disadvantages of order-based scheme However, on the downside, the schemes are strongly affected by some security faults. One such security issue is the Byzantine fault problem, which may need some Byzantine fault models to be applied to these designs. Also, they are vulnerable to security breaches such as DDoS attacks and can also be used as a platform for launching the DDoS attacks.

Security Discussion: Storage and Resource Lookup

Security concerns that affect storage also affect the distributed data structures, as they are used to store objects, and the resource lookup mechanisms, which are used to locate the stored objects. Information integrity in the P2P network may be compromised through the introduction of low quality (degraded) or by otherwise misrepresenting the content identity (false labeling) [158]. The main security threats that target the content stored therefore focus at corruption

or erasure of stored data in the system. Some of the threats include worm propagation, the rational attacks, storage and retrieval attacks, index poisoning attacks, pollution attacks and query flooding attacks [27].

Most of these challenges can be solved easily by incorporating a trust model within the system such as a reputation system. It has also been shown that a trust model can mitigate worm propagation [159]. Reputation systems are useful in the detection of selfish peers, thus are good for mitigating against free-riders, but they fail in the detection of Byzantine peers and malicious peers. Byzantine peers are peers who behave randomly, that is, they misbehave, but not necessarily following a pattern to maximize their benefit or to disrupt the system while malicious peers perform actions based on a target that is either detrimental or beneficial to the system (or both) [160]. To handle this, *micropayment systems* (MPS) can be used. MPSs are indirect incentive systems in which virtual or real currency, such as Bitcoin [161], is used to create a form of indirection between the contribution of a service and the request of similar contribution from another peer [160, 162]. The MPS architecture includes a Broker that issues the currency and certifies its value. Additionally, the MPSs usually incorporate a significant amount of cryptographic verification and hence are mostly used in static content distribution systems. However, solutions that require trusted third parties are to be avoided.

Communication: Unicast, Multicast and Publish/Subscribe

In general, communication systems support three types of communication models, that is, *unicast* (one-to-one), *multicast* (many-to-many) and *broadcast* (one-to-all). In P2P systems, the use of broadcast algorithms is highly discouraged and when used there is a need to setup a limitation on the number of hops to prevent network flooding and consequently slow down the entire network. Thus, it is preferable for P2P systems to utilize unicast and multicast algorithms so as to optimally and efficiently realize effective communications. We therefore discuss the different communication options that are preferred for P2P systems.

Unicast and Multicast Communications

Unicast communications This allows users to make use of application features such as direct messaging, video/audio chatting, file sharing among others.

Multicast communications This allows sending of packets to a group of recipients that may be scattered throughout the network and users can choose whether to participate in a multicast group or not. Therefore, because the packets travel only to subscribed users, there is reduced network load and end-to-end-delay, in comparison to broadcasting systems [163]. Multicasting can be [164]:

- *1-to-M multicasting* Useful in application that offer scheduled audio/video distributions, push media, file distribution and caching, announcements and monitoring of real-time information.
- *N-to-M multicasting* Utilized in multimedia conferencing, synchronized resources such as databases, concurrent processing (in particular, distributed parallel processing), collaboration (such as shared document editing), distance learning, chat groups, distributed interactive simulations, multi-player games and jam sessions.
- *N-to-1 multicasting* Seen in resource discovery, data collection, auction systems, polling, jukebox systems and accounting.

Publish/Subscribe Systems

One particular use of all the features that multicasting provides is seen in the development of publish/subscribe (pub/sub) system, event-driven distributed system composed of three types of processes, inter alia, publishers, subscribers and brokers [165]. They support distribution of information/data from the publishers (data/event producers) to the subscribers (data/event consumers). The publisher sends out a notification of an application event, and any user who subscribes to that application event becomes a target for the notification. Brokers are essentially routing algorithms that match the event notifications against the subscriber requirements and deliver the notifications to the target subscribers. Pub/sub-systems models can be based on: subscription, event routing and overlay topology [166, 167].

(a) *Subscription models* Subscribers have the ability to precisely matching their interests. The model determines the overall specification of events and also has an effect on how the events are routed within the event channel. Subscription models can be topic-based, content-based and type-based models.

- *Topic based* Events have locally or globally unique IDs that are usually identifiable character strings. Topics also represent logical connection channels between publishers and interested subscribers with network multicasts and diffusion trees being utilized for event distribution. Because they take only coarse-grained subscriptions, they give limited expressiveness and choices for subscriptions. Examples include Scribe [168] and Bayeux [169].
- *Content based* Notifications are composed of value-attribute pair sets. A subscription can be any randomly

chosen number of attribute names with filtering based on their values. The advantage of content-based models over topic-based models is that the subscription selectivity is increased due to increased dimension of choices. Events that meet the subscription criteria are then delivered to the subscriber. However, the disadvantage with these systems is in developing matching algorithms that are scalable and remain efficient.

– *Type based* Events are objects of a specific type group which can also encapsulate attributes and methods. In this model, declaring of a desired type becomes the distinguishing attribute. They take a middle ground between the previous two subscription models, giving a coarse-grained structure on events (topic based) on which fine-grained constraints can be expressed over attributes (content based).

(b) *Routing models* These models take into consideration the problem of event dispatching. They ensure that matched events are properly routed to the relevant subscribers. The events-to-subscription matchings are done using an appropriate filter. Then the routing algorithm forwards the events to the subscriber either directly or indirectly via elements close to the subscriber. The routing algorithms are classified as follows [167]:

– *Selective filtering* Subscriptions are filtered somewhere along the notification channel, thus presenting the need for a subscription or a routing table.
– *Gossiping* Utilizes a probabilistic neighbor forwarding strategy.
– *Flooding* Events are broadcasted through the notification channel.
– *Rendezvous* One node acts as a routing point for a given class of events.

(c) *Overlay topology* Pub/sub-systems may be classified based on the event channel's architectural realization or topology organization [170, 171]. Thus, the main classifications are

– *Centralized fixed topology* One broker acts as a centralized server, storing all subscriptions, performing event to subscription mappings and undertaking event delivery to matched subscribers. However, these present a single point of failure and are not highly scalable and reliable as is required for distributed applications. Examples include Elvin [172] and S-ToPPS [173].
– *Distributed static topology* Are also referred to as hybrid P2P or partially decentralized. The static topology points to existence of a graph-like distribution of brokers that are predictable and not expected to significantly change over time. The main topologies include, hierarchical (such as tree-based such as JEDI [174]), acyclic where event-flow via brokers is not permitted to form cycles (such as REBECA [175]), cyclic where event-flows result in a general graph, or a combination of the above (such as SIENA [176, 177]).
– *Distributed dynamic topology* Also referred to as pure P2P or fully decentralized. The broker overlay relies on a secondary real P2P overlay or is actually part of the overlay itself. Examples of include Scribe [168], Bayeux [169], NICE [96], Meghdoot [178] and LightPS [179].

Security Discussion: Communication and Publish/Subscribe Systems

The very first need for users in communication ensuring security of data during communication data. Using appropriate cryptographic services (encipherment), which can be either symmetric (secret key) and asymmetric (public key infrastructure) depending on system needs. The public key infrastructure (PKI) incorporates services such as node certification, node revocation, certificate storage and certificate retrieval, which ensure that there is secure assignment of NodeIDs, as well as provide for authentication. It also provides necessary security controls such as availability, resiliency, unforgettability, proactive security and secure communications, while also supporting efficient scalability, distribution of functionality and tolerance to churn [180].

To better identify nodes in the network, the public key infrastructure (PKI) can be used, as it links a user's identity to the cryptographic key. The solutions proposed for node identification include use a set of trusted certification authorities (CA) to assign the NodeIDs to principals as well as to sign the NodeID certificates that bind a random NodeID to the public key of the principal and its IP address [181], as well as using a distributed PKI [180]. The CAs, however, are a single point of failure as they are vulnerable to both legal and technical attacks while the distributed PKIs are easily affected by Sybil attacks [181]. Alternatively nodes may be required to solve crypto puzzles to obtain a NodeID [182] which has been shown to mitigate against Sybil attacks [183, 184]. PKIs, however, face the limitation that either trust among the users or trust in a third party is assumed, which typically is not given in fully decentralized networks.

In case symmetric encryption is used, an appropriate key exchange mechanisms is essential, such as Diffie–Hellman key exchange protocol. Asymmetric keys can also serve as digital signatures to enforce non-repudiation of data. In combination with the cryptographic schemes, an appropriate access control method is needed to authenticate the identity of a user or information about a user. To complete the

requirements for secure communications, secure routing is necessary. This must support unique and secure ID assignment to prevent abuse of the illegal IDs by malicious peers. With well designed security mechanics, P2P networks which are normally designed to harbour semi-trusted and untrusted peers can remain robust and secure.

The pub/sub systems of the P2P networks are not free from security concerns. The main concern in pub/sub systems is ascertaining the confidentiality of exchanged information without limiting the decoupling of the paradigm [185]. Security threats that affect pub/sub-systems include identity attacks, network communication attacks, network protocol attacks, passing illegal data, stored data attacks, remote information inference, loss of accountability and uncontrolled operations [167]. To secure pub/sub systems, they should incorporate trust management, information flow control, ubiquitous security self-adaptation, decentralized security, plugins and dynamic security reconfiguration and combination of static and dynamic solution features [167]. This ensures publication confidentiality so that the content of events cannot be known by the broker or any unauthorized third party, and subscription confidentiality so that filter details are hidden from brokers and unauthorized third parties.

Efficient P2P data transfer In real-time applications such as online social networks, reliable and efficient the data transmission is an enormous requirement. In centralized systems, solutions to this challenge are dependent on the finances available to the service provider in ensuring a consistent broadband connection or in utilization of cloud-based solutions. On the other hand, for P2P networks, the upload bandwidth is provided by the users (typically about 10 MB/s). With parallel chunk-based downloads from different replica holders, increasing the available upload bandwidth is possible but at the cost of additional communication overhead to contact those replica holders. In social networks, images account for a large percentage of the transferred data, otherwise mostly smaller data (less than 10 kB), such as messages, are transferred. For larger data sizes, such as videos, concepts such as BitTorrent can be used to share unique files or Chunked Swarm [186] to benefit from p2p-supported live video streaming and reduce the upload requirements of the stream provider by up to 95%. Thus although the single peer upload bandwidth is small, the presence of redundancy help significantly.

Services: Monitoring and Management

Using of the functionality blocks thus far discussed, building a decent and rich P2P-based social network application is now feasible. However, for quality-focused, fully decentralized P2P application, an essential requirement is monitoring and quality management. Once the system is operational, it is now possible to measure the performance and ascertain its quality based on the capacities of network nodes, current workload and initial system configuration. Hence, equipping the system with monitoring capabilities will provide a timely and precise view on the performance of the P2P network, and in conjunction with management capabilities, allow for automated system configuration changes at the nodes to improve performance hence overall quality.

Network Monitoring

Goal of monitoring The monitoring component retrieves an exhaustive statistical view on a wide set of metrics on all peers in the network and to disseminate it to all peers in the network. The set of metrics is an extendable list common to all network nodes. The metrics are based on local measurements at each peer, such as the bandwidth consumption or observed lookup delays or a peer. The statistical view on the metrics (average, minimum, maximum and so on) is taken over the local measurements in the network and local measurements are gathered to obtain a global view on the system statistics. The global statistics are then disseminated to all network peers to ensure they have a global view of the entire network.

Uses of monitoring information Monitoring information can be used for various purposes, for example,

- To foresee low availability of replicating nodes and nodes with relevant duties and thus to counteract by selecting further nodes for replication [187].
- Using information on message priorities, differentiated services can be provided through adaptive strategies to forward messages in P2P overlays, as presented in [188].
- Optimized routing in P2P-based social networks based on monitoring information including social interaction patterns is discussed in [189].
- Self-stabilization, a property that allows converging from any given connected topology to a desired topology, such as Chord as described in [190]. This approach can benefit from global monitoring statistics as the self-stabilization process can be accelerated.

Distributed Monitoring Approaches

Centralized monitoring approaches such as the simple network management protocol (SNMP) [191] or network/transport layer-focused approaches [192] are not suitable for a distributed environment, necessitating a decentralized approach. Integrating the monitoring functionality into the used overlay, such as it is done in DASIS [193] or Willow

[194] is one option. Here the prefix-based routing tables of the corresponding overlays are extended to maintain monitoring data as well and a corresponding data exchange protocol is included in the routing table update communication. P2P-Diet [195] and HilbertChord [196] are further variants of integrated monitoring solutions in existing P2P overlays. However, when combining the monitoring and routing functionality, there is the problem that the two functionalities cannot be independently improved and optimized.

Decentralized P2P monitoring solutions are broken into unstructured or structured approaches.

(a) *Unstructured approaches* They simply use the contacts that are available in the overlays routing table and apply a gossip based information exchange, where each peer exchanges periodically its knowledge with neighbors. Examples for this category is gossiping [197], T-MAN [198] and push-sum [199]. While all nodes can directly start monitoring and the monitoring topology is robust against churn, information spreads slowly and redundancy occurs, leading to outdated monitoring results.

(b) *Structured approaches* They build new topologies, typically a tree structure, on top of the used P2P overlay for dedicated monitoring data flows. The monitoring information is gathered, cyclic free towards the root, aggregated on its way towards the root, and then spread to all participating nodes in the tree again. Examples are SkyEye [200, 201], CONE [202, 203], or SOMO [204]. SkyEye, as an advanced example, uses a tree-based approach which allows for efficient aggregation and dissemination of information, up and down the tree respectively. The tree height defines the freshness of the aggregated statistics, which can be obtained without any redundant information transfer, thus highly optimized. Also the costs are bound by the fixed node degree of the nodes. Regardless of the position in the tree, each node encounters the same load. Lastly, one may note that trees in the first place are vulnerable to churn. Through the creation of multiple trees, such as in [205], a dynamic set of parallel monitoring topologies are created that is used to highly reduce the failure of an individual node on the monitoring tree. Through the expected similarity of the monitoring results in the parallel topologies, errors and outliers in the monitoring data can be identified and corrected.

Network Management

Goal of management The management component takes the current monitoring statistics of the network and using a mechanism for distributed analysis, makes it possible to assess the situation and plan changes to the configuration of all nodes, to effect improvement in overall performance of the network. The monitoring information thus gives insights on the quality and weaknesses of the network. This can then be used to implement a distributed control-loop for P2P systems, as suggested in [206, 207]. The monitoring information is obtained through a distributed approach, analyzed and parameter changes are decided which are then communicated and executed throughout the network. The network is now capable of identifying and resolving its own weaknesses, and then lead the system back to a threshold state.

For example, averagely long lookup delays will lead to long data access and retrieval times in the network. Through monitoring, nodes realize a high hop-count during routing hence lookup delay and decide, via a distributed mechanism, to increase the size of their routing tables with the perspective to have the better contacts in the routing table resulting in a lower hop count. This decision is diverged to all nodes and takes effect once all extended routing tables are sufficiently filled with more nodes. Thereafter nodes can again evaluate if the lookup delay is solved or whether further adaptations are needed.

Summary Thus, several improvements become possible through a monitoring approach. We further tend to believe that such a monitoring and management mechanism with an integrated quality monitoring and control loop is essential for the operation of a fully distributed P2P-based social network.

Having reviewed the essential building blocks for advanced P2P applications, next we present and discuss the proposed solutions of P2P-based social networks.

P2P-Based Social Networks

Peer-to-peer-based SNs have been put forward as a possible alternative for addressing the security and scalability challenges associated with centralized approached for SNs. To this end, various proposals for possible solutions have been made, as is evident from literature. Each proposal tries to achieve a fully functional P2P-based OSN using various combinations for P2P components, and may sometimes propose new methods to achieve a critical aspect of OSN within the P2P environment. In discussing the proposals studied, we take the direction of classifying them based on the structure of the overlay as illustrated in Fig. 6. In addition, we also consider whether the SN is a microblog or a full SN (which we simply refer to as SN). In the analysis, the trend is to give a general overview of the design goals, the architecture, and a brief discussion of possible flaws noted with the proposal.

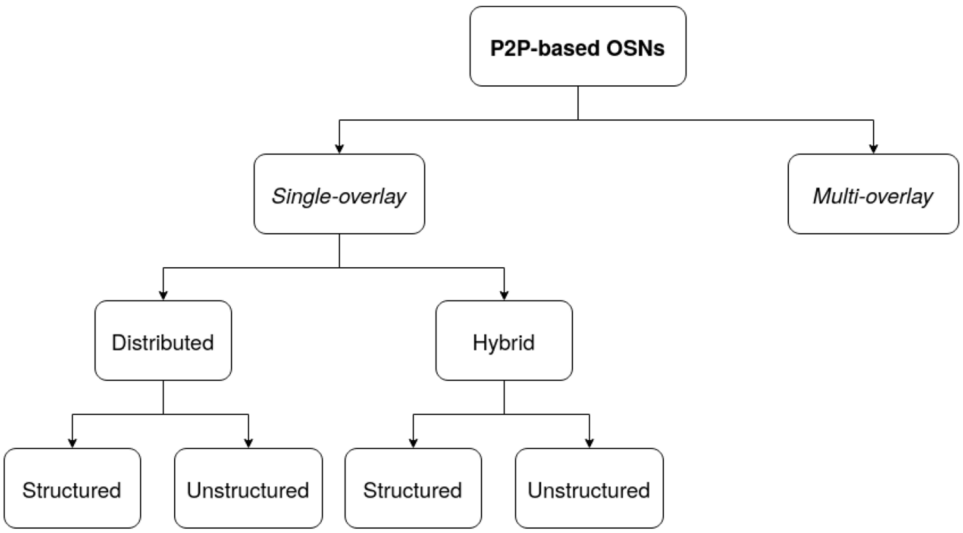


Fig. 6 Classification of the P2P-based OSNs

Single-Overlay Distributed Social Networks

As highlighted in section "Single-Overlay P2P Networks", these social networks are designed on a single overlay (structured or unstructured) and, therefore, all routing and storage procedures are handled by the overlay itself. We discuss the proposals in this category.

LifeSocial.KOM/LibreSocial

LifeSocial.KOM [62, 208, 209]/LibreSocial [210] was first proposed in 2008. Due to name conflicts, the initial name of LifeSocial.KOM has been changed to LibreSocial (https://libresocial.com). It is a plugin-based and extendible P2P based OSN build on a composition of various essential P2P functionalities within a P2P framework.

Architectural design It utilizes an OSGi-based framework which is highly modular, with FreePastry for the overlay network and PAST to provide reliable storage with data replication mechanisms. Both FreePastry and PAST have been heavily modified to increase robustness, fault-tolerance and security. PAST has been extended to support data updates, such as profile data updates, and to ensure secure data storage that incorporates access control functions. The stored data is replicated with support for WRITE, READ, UPDATE, DELETE and APPEND operations. The replication mechanisms maintain availability of the stored data and ensure load balancing, overload avoidance and the support for weak nodes in the network. It further includes secure, access controlled and replicated distributed data structures, such as distributed sets, linked lists and prefix hash trees. This supports a variety of data forms, such as (comment) lists, (photo and friend) sets or (forum thread) trees. As the underlying P2P framework and secure data structures are very general, it is easy to add new application functionalities in LibreSocial. One unique function in LibreSocial is the availability of a message inbox which can be read by the corresponding user only but filled with entries from various users. This feature is cryptographically enforced to ensure that crucial messages, intended only for the eyes of the recipient user must not be read by his/her friends. This is guaranteed in LibreSocial. A mechanism for keyword-based and range-queries is integrated to support the searching for searchable items. Currently, it is used to support the search for users based on the profile information that explicitly has been marked as public by its owner. The OSN offers end-to-end secured unicast, multicast and publish/subscribe functions. As FreePastry is a DHT that offers reliable key-based routing. To quicken the process the routing protocol has been adapted to become iterative such as in Kademlia instead of the previous recursive approach. In joining the network, weak and strong nodes are treated differently so as to support weak nodes in the network, mainly as clients, without involving them in the burden of carrying load, or routing.

Security This is enforced via the registration and login mechanisms, and using access control for data stored as described in [211]. An asymmetric key mechanism without the use of a server, certificate authorities or even the trust in other nodes is used for for the creation of cryptographic keys, that are used throughout the system. The public key is also the node ID, which allows for direct encrypted communication and authentication. Access control is managed by encrypting the data for the various access enabled users in the network. One outstanding feature of LibreSocial is that the friends in the network do not play any privileged role in terms of security and trust in them is not required.

OSN features All the common OSN functionalities such as user profiles, friends lists, user groups, photo albums, chatting and status updates are provided. It also includes the forums and data spaces for collaboration, messaging and calendars for coordination, as well as text-, audio-based and video-based real-time communication. In addition to the rich set of applications, it comes with an integrated decentralized app repository that any user can host, and which allows to created and share application plugins with other users. Also it comes with a fully decentralized monitoring that allows to observe and evaluate the performance of the P2P network.

Porkut/My3

Porkut [212] and My3 [213] are similar proposals by the same authors to provide privacy-preserving data access. The OSN was designed to achieve three goals: the elimination of a single administrative control; privacy preservation of individual's privacy content, giving users complete control of their profile and its content; and the exploitation of trust relationships among network users for improvement of content availability and storage performance.

Architectural design The application has three key features: a *DHT* such as OpenDHT [214] that is used to store meta information of the user based on a *user-to-TPS* mapping to form the $\langle key, value \rangle$ pair; an *online time graph* that contains all the user's friends as vertices and edges are only existent if there is an overlap in online times between two trusted pairs; and a storage layer which is a construction of the trusted proxy set (TPS) for a user u. The TPS is a set of self-defined nodes in which a user's profile is hosted. Using an appropriate algorithm, the set TPS for a particular user is constructed from his social graph in which users are characterized as having two parameters:a geographical location that determines the time zone of the user, and an online time period which is the time the user is online in the social network. The proposed criteria possible for selecting the proper set of members into TPS from all possible trusted friends are low access and consistency costs and high data availability.

Security A privacy-preserving index of the social network contents is constructed such that it is possible to perform privacy-aware searching which enables content discovery among friends in the OSN and allowing new discovery of new friends and the establishment of new social connections. The index mechanism uses k-anonymization techniques so that a list of keys are mapped to a list of values. This helps achieve content and owner privacy, so that, with the indexing scheme, strangers can contact each other based on interested content.

OSN features The main aspects of the system that are emphasized, in addition to the basic requirements for an OSN, are storage layer formed by the TPS construction, profile accessibility through an available mount point of a given user, update propagation as a user's profile is replicated to other mount points, and eventual consistency since concurrent updates ensure mount points are up to date.

This system assumes that users have friends they trust and that online times of these trusted friends overlap. However, it is important to note that the existence of (online) friends as well as the requirement to trust these friends is an assumption that is not always existent in reality. Without friends or trust in them, then data availability, confidentiality and integrity is not guaranteed.

Megaphone

Megaphone [215] is a P2P microblogging application designed with the aim of overcoming the problem of single point of failure due to reliance on web-based services in centralized systems.

Architectural design It utilizes Pastry and Scribe to perform message and group routing by organizing the social graph of users into a multicast trees on top of Pastry using Scribe in which the "poster" node is the root of the tree with "followers" being child nodes. Therefore, the poster creates the tree and performs the task of managing the joins, lists of followers, storage of the public keys of child nodes and sending of messages. Followers can send response messages to the posters. As the overlay utilized is Pastry, the network guarantees logarithmic routing from source to destination.

Security features The messages can be signed and encrypted using public key infrastructure based on RSA [216] algorithm. Posters generate session keys encrypted with their private key and further encrypted using the followers' public keys. All nodes in the multicast tree cache this session key but it is readable only by authorized nodes. Authentication may be done by self-generated and self-signed certificates or generated by a certification authority. Node IDs are based on concatenated hashes of the username and public key to guarantee uniqueness. All members of a multicast tree, hence followers, have knowledge of the originator of any post and all the followers which means that the privacy of the users is not fully guaranteed.

eXO

eXO [217] is a completely decentralized, scalable system that is designed to offer key social networking services, while relying on a P2P platform. The system has two main goal, namely, foster the idea of highly distributed SN

functionality, that is, autonomy, and support full user control even when sharing content.

Architectural design The underlay consists of a large number of nodes, with each node running a routing protocol for a structured overlay DHT such as Pastry [86] and Chord [88]. A node can have any of three roles. It can be a request solver, serving user requests and dispatching the requests; a network storage interface for contents and profile replicas; or act as a catalogue node that stores indexing data structures. Content item are indexed by representing them by a set of terms, or keywords that describe them, called the content profile. This is useful in performing "top-*k* similar content items" queries. A user is also described by a set of user-defined terms which is then called the user profile. These two profiles make the use of tags possible. Tags are terms contributed by a user describing a specific content item or user, and they help in achieving better quality query result by leveraging community wisdom. *eXO* supports public networks and personal SNs. The public network is composed of the DHT, and the content and user profiles stored at the DHT nodes can be indexed and accessed via the DHT. The DHT structure makes it possible to perform queries on the user profiles as desired. Through the process of searching for interesting user profiles, a user can identify and add these user profiles to their personal social network.

Security features To support autonomy and privacy, content shared by a user (primarily images, audio and video content and secondarily text) is kept only at the user's node while replication is done only on nodes adjacent to the user node in the *ID* space at the owner's request or in case the content's availability is needed. Content as well as user profiles can be either public which is indexed and available to all, or private which is not indexed.

The main limitation in *eXO* is that data can be either fully public or hidden. Through the lack of a foundation for security mechanisms, confidentiality, access control and data integrity is not given. Thus, the platform can be used for only very simple applications.

PAC'nPOST

PAC'nPOST [218] is a framework for a microblogging social network. The two goals of the systems are, first, to enable the users the retrieve blogs of other users that are being followed and, secondly, to allow a user to perform keyword-based searches.

Architectural design It is implemented on an unstructured P2P network. Its search and retrieval mechanism is based on a probably approximately correct (PAC) search architecture, where a query is sent to a fixed number of random nodes in the network and the probability of attaining a certain accuracy being a function of nodes queried (fixed) and the documents' replication rate. As the nodes do not push their blog entries, but rather they must be searched, chances are high that blog entries are missed or that over time, in case that the number of participants rises, the network will be overloaded with search messages.

This OSN is quite limited in terms of features it offers. Further, there are no security considerations presented or suggested. In addition, in unstructured overlay networks, a high search precision necessarily requires an adequately high traffic load, thus the network functions do not scale. Therefore, its actual usage may be very limited.

DECENT

DECENT [219] is a proposed decentralized OSN architecture, whose goal is to: utilize object-oriented design (OOD) to enhance flexible data management; ensure efficient access revocation and fine-grained data policies using suitable cryptographic methods; and combine confidentiality, integrity, and availability through use of DHT functionalities.

Architectural design DECENT has modular design. The data objects, cryptographic mechanisms and the DHT are three separate components that interact through interfaces. This allows the OSN to use any type of DHT or cryptographic mechanism. The participants in the OSN are organized into a DHT, such as FreePastry or Kademlia, which provides for a scalable $\langle key, value \rangle$ pair store having an efficient lookup mechanism for the location of objects stored in the nodes. Every object in the OSN has three access policies (read, write and append) associated with it which can be attribute based (AB), identity based (IB) or a combination of both. These policies are defined by the user at object creation and stored in the object's metadata. Objects are stored in the DHT with the object ID as the key and replicated for redundancy. Each object has a version number as part of its metadata for freshness guarantees that is authenticated by a write-policy signature to prevent modifications by a malicious node. Storage nodes do not know the write-policy signature public key (SPK) as it is part of the object reference. Hence, the storage node cannot differentiate between a legitimate and a malicious update. This is solved by adding an unencrypted metadata field to an object containing a public key, which is used to authenticate write requests (write authentication public key, or WAPK). Thus, write/delete request must be signed by the corresponding secret key. If not, the storage node refuses the request.

Security features DECENT implements a modified attribute-based encryption (ABE) algorithm that includes support for immediate revocation using an extended EASiER scheme.

With ABE the message is encrypted using a randomly chosen symmetric encryption key, then encrypted with ABE. The ABE encrypted symmetric key is part of the object reference and not included in the object itself. This keeps the policy hidden from untrusted storage nodes. Objects can have different read policies associated if there exists several references for an object. The EASiER scheme uses a proxy in every decryption to ensure the revoked contact is no longer able access data that requires the revoked attribute(s). There are two extensions to the EASiER scheme. First, the proxy functionality is divided among several randomly selected nodes using threshold secret sharing. Assuming majority of nodes are not actively malicious, this ensures the security of the proxy. Second, is an extension to support attribute delegation. To enforce authenticity, the write policy public key must be part of the object reference, rather than the object itself. Also, as the append policy is authenticated by the write-policy signature, it is included as part of the object metadata.

DECENT is similar to Cachet in providing security, sharing the same shortcomings. First, the origin or the user credentials are unclear and second, through the focus on secure storage, the communication options are neglected.

PESCA

This proposal by [220] was designed with the aim of achieving privacy in the social communication and as well as social data availability.

Architectural design PESCA assumes a structured P2P overlay based on DHTs for an efficient lookup service. It monitors the users's online patterns as well as the devices they use, taking into consideration the time of the day and the days of the week which are stored into the user online table (UOT) as small non-overlapping time slots which also indicate the device used. The replica placement strategy that is utilized take into account the online direct/indirect friends and the data audiences. The strategy determines the best replica matrix corresponding to a user's data in a greedy fashion. The replica candidate list includes the user's friends who are online and have enough storage space. Each candidate is scored based on the number of overlaps between its uptime and the audience's uptime, and candidates with scores of zero are eliminated, and those with the highest scores are chosen. In case of similar scores available storage space is considered.

Security features Data confidentiality and access control are achieved using broadcast encryption (BE) scheme [221]. Users have a uniquely selected global identity (GID) based on a hash value of the user's email address. A user also generates a virtual identity (VID), an ambiguous index, and a BE secret key for every social contact added as a friend. The user also allocates space, called space budget (SB) that can be spared for the friend. The user then sends the VID and the BE secret key via a secure channel along with the GID, the current UOT and assigned SB via a public channel.

The designers of PESCA do not provide any information about which OSN services the proposal offers outside of guaranteeing secure social communication and social data availability. This opens the proposals to questions on the practicality of its use in the real world.

WebP2P

This work is a break from the norm in the sense that it aims at providing a DOSN without the need to install any new software. Disterhöft et al [222] propose a solution and prototype for a Web-based P2P framework that supports social networking. Specifically, it supports secure buddy lists, data storage and personal text, audio and video communication.

Architectural design It is implemented on a browser-based implementation of the Chord Protocol that relies on WebRTC (https://webrtc.org/). WebRTC allows browser instances to establish connections to each other. Based on this, a DHT is implemented based on OpenDHT and has been extended to support secure user identities, secure communication and a simple secure storage. The Web-based DHT overlay provides the basic functions that support buddy list management, communications via text-based chat, and device-independent, distributed storage of contacts and chat history. Thus, the functions are focused on chatting and maintaining a buddy list. Larger data items cannot be stored as WebRTC is limited to use only 5MB of storage space to hinder potential attacks. Data are stored and replicated in the DHT created among the browsers and encrypted by the data owner. Besides the overlay-based storage and communication, WebRTC channels are also used to establish video and audio calls between the participants.

Security features Authentication relies on public keys generated using an asymmetric Elliptic Curve Cryptography mechanism. The public keys also function as node IDs and user IDs. Each user initially has to use these cumbersome lengthy 160 bit IDs as user identifiers until, through a dialogue the user can identify each other and assign in the GUI and buddy list an alternative name for this user ID. Thus, fully decentralized authentication is provided. As the user IDs are public keys, the communication can be encrypted with the public key of the receiver and signed by the sender, providing confidentiality and integrity. A distributed identity-based access control mechanism is further used that uses self-signed certificates.

The WebP2P-framework is self-contained, secure and provides for the purpose of secure chatting in a web-based P2P-DOSN with focus on chatting, video and audio.

Single-Overlay Hybrid Social Networks

These SNs implement a hybrid structure in which a single overlay (structured or unstructured) is utilized and some degree of centralization is incorporated.

In most of these cases, centralization is used to provide a solution for indexing while the P2P overlay is used to handle routing. We discuss some of the SNs proposed in this category in the following.

P2P Social Networking (PeerSoN)

PeerSoN [223] was an advancement of the ideas put forward in [60]. It is built to address privacy concerns raised over OSNs as well as look at how to ensure availability. In the proposed prototype, the privacy problem is addressed by integrating encryption and access control to implement a user login procedure. Availability is made possible by including novel file sharing procedures.

Architectural design PeerSoN has a two-tier architecture that ensures users' contents are decoupled from the control mechanisms. The first tier, the lower level, consists of the users and the content, which allows the users to exchange content directly with each other. The second level is the DHT, specifically OpenDHT, which provides lookup services, stores updates for a user in case a user goes offline and the user's meta-data.

Security features Security and privacy concerns in the system are addressed using identity management. PeerSoN assumes the availability of a public-key infrastructure (PKI) with the possibility of revocation of keys and encryption using public keys of intended audience.

OpenDHT is a centrally managed deployment of the Bamboo DHT on PlanetLab [7]. The use of a centralized component in the OSN is not desired when the overall goal is to realize full decentralization.

Safebook

Safebook [224–226] is a decentralized SN designed to achieve two goals. First, it utilizes a P2P architecture to avoid user data and user behavior control by a single entity such as a service provider. Second, it aims at providing privacy and trust management for user data and communication in the system through trust relations existing in the social network.

Architectural design Safebook implements a three-tier architecture with a direct mapping of layers to the OSN level. The layers are a user-centered SN layer which forms the SN level of the OSN, a P2P substrate which implements the application services (AS), and the Internet which represents the communication and transport (CT) level. Participants in the OSN are viewed as a host node on the Internet, a peer node in the P2P overlay and a member of the SN layer. Participating nodes form two types of overlays: a set of *Matryoshkas* which are concentric structures in the SN level that provide distributed data storage with privacy and end-to-end confidentiality by assuming that users trust their friends in handling their data carefully; and a P2P substrate such as a DHT that supports data lookup services.

Security features It utilizes a *trusted identifier service* (TIS) to give nodes a unique pseudonym and identifier for the SN level, and related certificates. The TIS is not involved in data management and hence does not violate the goal of privacy preservation. This guarantees protection against attacks such as Sybil and impersonation attacks. After obtaining an identity, a new user begins the process of creating their own Matryoshka. This is done by first sending requests to friends or trusted peers. Secure communication between users is made possible by encryption and decryption of the pseudonyms using private and public keys, hence message integrity and confidentiality.

Disadvantageous in this approach is the need to have friends to be able to store data reliable, to trust these friends to not tamper with the data as well as the need for the TIS as centralized component. In Safebook all data are to be assumed to be shared with the "friends", thus confidentiality is not given. In addition, users without friends or with friends that are often online cannot maintain the availability of their data.

Cuckoo

Cuckoo [227, 228] is a socio-aware online microblogging system that is proposed and built to be compatible to the Twitter architecture.

Architectural design It is designed to utilize the Twitter servers to conserve bandwidth and storage resources while also taking advantage of the P2P technologies for scaling and reliable microblogging services. Cuckoo uses Pastry [86] as its underlying overlay. Typical microblogging services allow for the following social relations to exist between users: friend, neighbor, follower, and following. Along with the Pastry routing table, users maintain these four user lists. So

[7] https://www.planet-lab.org.

as to perform searching to perform status updates, Cuckoo uses a hybrid search method, that is, flooding to fetch statuses from influentials and the DHT to fetch statuses from normal users. Newly published micro-content is disseminated to users using push method rather than pull as it is more efficient. Micro-content is also replicated among followers thus followers can provide the content to each other in a cache-and-relay style in case the original publisher is unavailable.

It has to be noted that without a cryptographic foundation any communication and data stored can be tampered and read. Cuckoo is insecure and highly requires the existence friends, neighbors, followers and followed users.

Litter

Litter [229, 230] is a lightweight microblogging service that leverages P2P virtual private network (P2PVPN) technologies, such as Hamachi[8] and SocialVPN[9]. P2PVPNs utilize P2P technologies for direct IP traffic tunneling among peers. They also provide a trusted platform in which peers can communicate and collaborate at the IP layer with each other. Thus the trusted connections between the social peers form a social overlay P2P network. The P2PVPN also supports IP multicasting by tunneling the multicast packets to each friend with whom a node has an encrypted P2P connection.

Architectural design Litter's microblogging service is composed of two basic IP layer mechanisms. The first IP layer is IP multicasting that is used to propagate messages to two-hops neighbors within the social graph. The second IP layer is UDP datagrams that are necessary for traversing the social graph for update dissemination to social distant peers. The use of these services is based on the assumption that users are running a P2PVPN. The messages in the system are distributed in a push/pull format in followers in four ways namely, multicast push to followers, multicast pull by followers, random-walk push to distant followers and random-walk pull by followers.

Security features To achieve peer discovery and cryptographic key distribution, there are two proposed solutions. In the first solution there is exchange of endpoint information and public keys via some trusted, out-of-band communication path by the users. This model is referred to as the Freenet darknet model. The second solution achieves peer discovery and key exchange through a reliance on the XMPP federation as the trusted medium. Message privacy is done through the use of a permission flag that controls who can share the posts as well as a time-to-live (TTL) to control the scope of the updates. Message verification and integrity is done through the use of signatures. Followers are responsible for acquiring the publisher's public key via a trusted out-of-band system.

A disadvantage in such VPN networks is that the entry into the network is unclear. Nodes without friends, just joining the network, do not find contact points to connect to. Also, unfortunately, both security and trust initialization approaches, either through out-of-band channels or though centralized external XMPP servers, are not fully suitable for a fully decentralized OSN. Additionally, as only messaging is in the focus and routing is based on random-walks, the essential reliable data storage and retrieval is not provided.

SuperNova

SuperNova [146] is a distributed OSN designed to provide flexibility in terms of storage, that is, users have the choice of where to store their content and whose content they want to store.

Architectural design The architecture relies on a super-peer-based network of volunteer agents. Data availability in case a user is offline, is solved by users replicating their contents to a list of users called *storekeepers*. The super-peer nodes provide services in the system and particularly to new nodes. They take part in the formation of the network's control infrastructure, and are the basic building block of the entire architecture as they provide lookup services, storage services, book-keeping services, recommendation services (such as for new friends or for storekeepers) among other services.

Security features The users choose three different level of access to other users: public (accessible to all), private (only accessible by owner), and protected (accessible to a chosen subset of friends). This means the system guarantees full content ownership.

While the design addresses several requirements stated for a P2P-based OSN, the root of trust is undecided. Users must trust the Storekeepers to treat their data properly. Also, incentives for this cumbersome task are not given.

HorNet

HorNet [231] is a proposed microblogging service for contributory social networks that is built on a structured P2P overlay network. A contributory social network is a SN whereby the SN's resources, such as CPU, network and storage, are voluntarily contributed by the participating network members. The service is developed with the focus on availability, decentralization and performance.

[8] https://www.vpn.net/.

[9] http://ipop-project.org

Architectural design HorNet is split into three layers: the communication layer, a middle layer and the upper layer. The communication layer serves the function of connecting all the nodes that join and leave the system. It is implemented on FreePastry which offers key-based routing (KBR) which supports scalability and churn-tolerance. The middle layer guarantees availability of application logic's data and components by providing a file-based storage service for persistence of messages and other data, and a freely available middleware called CoDeS [232]. CoDeS has several functions. It forms a platform for service deployment by aggregation of a set of non-dedicated, global and heterogeneous computers, which guarantees required services are always available, self-managed and decentralized. It also provides redundancy for failure-tolerance and uses weak consistency replication of its internal information so as increase performance while ensuring almost accurate information provision. Finally, it allows users to manage which resources to share with the network. The upper layer of HorNet is a set of small services. Each user has an instance of the services in this layer deployed using CoDeS, and through CoDeS, the instances are kept available. The overall design of the HorNet system is thus divided into two main parts. The first is client providing the HorNet features to the users which is deployed as a web container. The second aspect of the design is a component set that is deployed as CoDeS services which enforces HorNet's core functionalities in a decentralized manner.

Security features Authentication is done using a PKI and it assumes that users hold a public-key certificate from a trusted CA upon registration for the services. All messages sent in the system are signed to protect authenticity and integrity.

HorNet uses the CA and the CoDeS servers as centralized components, as long as they are available HorNet networks might operate. As soon as one element is switched off, the network fails. For a fully decentralized OSN, servers should be avoided.

LotusNet

LotusNet [42] is a framework for the development of P2P-based social network services. Its goal is to provide support for strong user authentication and offers a solution for the trade-off between security, privacy and services within a DOSN. It achieves this through provision of users options for tuning privacy settings via a very flexible and fine-grained access control system. It includes a suite of high-level services which support custom application development and mash ups.

Architectural design The LotusNet architecture is based on a DHT called Likir [233], a customized version of Kademlia. The DHT offers distributed storage supporting implementation of social widgets to share and collect data. Likir unlike other DHTs requires users to fulfill a preliminary user registration procedure before receiving a certified identifier for their DHT node. Likir includes of a centralized Certification Service (CS) for this purpose. Directly on top of LotusNet's P2P layer is a custom suite of widgets that interact by exchanging objects through the DHT that provides the essential social network services. The provision of these services is directly based on the API of the overlay node which includes the identity management and authentication features. The communication between widgets is not restricted to be only via the DHT, but the widgets can establish direct connections if needed. As identity management is at the overlay level, all data published by the same node are marked with the same user identity with no consideration on the nature of the widget that generated the content. Therefore integration becomes easy since every widget is able to collect and aggregate content from the different widgets owned by the known social contacts by a simple method invocation. Thus every application is potentially able to cooperate with other modules, that is, it guarantees maximum interoperability.

Security features Likir includes two properties that are key to securing the overlay. Firstly, the overlay communications are two-way authenticated. Authentication as well as binding of the user's identity to a fixed but random Kademlia ID effectively counteracts threats such as Sybil attacks, resulting in a more robust P2P layer. Secondly, it offers verifiable content ownership by attaching owner-signed certificates to content published on the DHT. This allows for secure identity-based resource retrievals, resulting in a filtering facility that performs very sharp resource retrievals. In cases direct communication of, the distributed storage can be used for preliminary Diffie–Hellman exchange for the setting up of a secure out-of-band connection. The resulting new secure channel is also authenticated and encrypted as the key agreement protocol is done on a fully authenticated layer. Privacy of the shared information is guaranteed using signed grants. As the grants are linked to social contacts rather than to shared resources, their numbers do not grow with respect to the resources owned or with the number of privacy policy rules. A Discretionary Access Control Module (DACM) layered directly in the Likir node is used to manage the individual social connections and to set privacy policies in assigning grants. To further reduce risk of privacy violation for sensitive information, the privacy level of widgets is tuned by storing the sensitive data at a set of trusted contacts specified by the user.

While LotusNet gains essential elements of its security through Likir, the use of a centralized Certification Service in Likir is disadvantageous. A further limitation is that the access control is enforced through policies that are selected

rather than individual settings for each data object. Thus it is impossible to eventually fine tune the access rights on each document, or for example every picture, that is uploaded.

Vegas

Vegas [234] was designed as a secure OSN that limits the access to a user's social graph to the ego network only. It ensures that users have full control over who can access their personal profile and published content by enforcing strong trust relationships that are mapped to the real world. It also aims at offering mobility support while guaranteeing profile availability in cases when the user is offline.

Architectural design Vegas is designed on an unstructured P2P network. Communication between P2P devices occurs through exchangers, secure asynchronous communication channels that support delay-tolerant information exchange. To ensure profile availability despite P2P anomalies like churn, Vegas utilizes datastores, which enforces a write-one read-all storage policy, where only the owner can write. The datastores can be a web resource such as FTP or a cloud storage service such as Amazon S3[10], or Dropbox[11]. Users can operate several datastores through which the friends can access the profile and shared content. The OSN supports emailing, short messaging (SMS), instant messaging (XMPP) and microblogging (Twitter).

Security features Messages are secured using a public key pair K^-/K^+ called the link-specific key pair. For a given friend X of user Y, if Y is sending a message to X, then Y applies X's public key K_X^+ to the message to encrypt it, K_X^- to sign it, and adds the fingerprint of K_X^+ then sends it to an accessible exchanger. The identity of A can only be verified by X since only X knows the mapping of the fingerprint. To ensure access control for profile data during profile synchronization, a user applies a symmetric key to any given profile attribute, applies K^+ for each of his/her friends to encrypt it, and then updates all the corresponding datastores. Compromise on the link-specific key occurs if the mapping between the public part and the key issuer is disclosed. This is solved by a simple key refresh protocol that allows for in-band key negotiation and exchange. To enforce strong trust relationships, Vegas does not support social graph traversals to seek for others. Instead, friendships are formed by out-of-band invitations or by coupling in which a user introduces two other users to each other.

The downside with Vegas is that it takes into account the desired security and privacy aspects as proposed by the authors at the expense of functionality. In addition, it utilizes XMPP for instant messaging and Twitter for microblogging services which reintroduces the problems of centralized OSN to the entire setup.

Decentralized OSN Using P2P Technology

Tran et al. [235] propose a social network based on P2P architecture that supports social computing services in a distributed environment. The goals of the proposed P2P based SN are to achieve scalability in architecture, reliability in content distribution and autonomy in administration. It is also aimed at solving the problem of heterogeneity using certain peers (called super peers) with more resources, such as storage, processing power and bandwidth, to support other peers with complex operations.

Architectural design The architecture relies on a super peer network based on the Gnutella protocol. It implements two authentication and posting services to reduce reliance on centralized servers, and increase and encourage group communication in the social network. The authentication service gives users access to network services. By performing more social activities users can find more super peers. After authentication is complete, users can utilize the posting services to post messages to and request messages from individual users or user groups. The messages contain user statuses, user profiles and discussion updates. Group communication is supported for updating discussions. Group communication is implemented by defining the user group information based on user profiles. The users then select to send messages to either the whole group, a set of users or only one user. The posting service improves data privacy and search capability by allowing the users to keep personal data on peers and super peers. Each peer has a MySQL database to store user data, peer data and messages.

Security features The network includes and publishes several registration servers for user registration. Registration servers maintain a database storing details of the super peers and registered users, while the super peers maintain a database which stores registered users synchronized by the registration server. After registration, a list of super peers in the network is forwarded to the user and the super peers also receive an update on the registered user. Henceforth, the users simply authenticate themselves on the super peers during future logins. In case the super peers are offline, users can authenticate with the registration servers and obtain an updated list of super peers.

Disadvantageous in this approach is the strong reliance on the registration server which has to maintain the overview on all super peers and all nodes. Also secure communications is not addressed properly. Another stark limitation of the

[10] https://aws.amazon.com/s3.

[11] https://www.dropbox.com.

approach is that only messaging is supported, that is, nodes can push and pull data to/from each other, but there is not reliable data storage available. Consequently, the option in building an appealing OSN on top are limited.

HPOSN

HPOSN [236] is an optimized hybrid OSN model based in P2P rather than a fully distributed OSN. It is designed to solve Local Service Fault Partition (LSFP), a situation in which the network is inaccessible because of fault partitioning caused by equipment/link failure or network attacks in centralized OSNs, which completely prevents users from logging into the network.

Architectural design The OSN leverages P2P technologies along with the centralized servers to offer a solution to the LSFP problem. The application operates in centralized mode in normal conditions and incorporates a supplemental P2P mode when the LSFP problem arises. Only data that is considered important is stored in the local terminal, and the rest is stored in the servers. An index pointing to the unimportant data stored at the server is maintained by the nodes.

Security features Unlike centralized OSNs, HPOSN supports direct communication between nodes by leveraging SocialVPN to establish direct communications. The system adopts Onion Routing [237] to guarantee anonymous communications. Data stored at the terminals and the servers is encrypted using asymmetric encryption, with the private key of the user being used to encrypt the data.

This proposal, because of its strong reliance on the use of centralized servers, does not fully guarantee all aspects of privacy. This is evident from the fact that the system providers still have some access to the private data and may employ data mining algorithms to find out more information about the users. Also, because of the use of the servers as the main storage of the network, the scalability of the entire system is in question.

Multi-overlay Social Networks

This class of P2P-based social networks utilize an architecture that implements two or more overlays, such as structured overlay based on a DHT and a social overlay, so as to realize efficient indexing and storage. They are discussed hereafter.

Cachet

Cachet [238] was designed with the main focus being on protection of the confidentiality, integrity and availability of the user content, while also ensuring privacy of the user relationships.

Architectural design Cachet utilizes a hybrid structured-unstructured overlay format, that augments a DHT with social links between users. The DHT enables decentralization. Data is stored as an object in a DHT, such as Pastry [86] or Kademlia [89], using a random object identifier (*objID*) as the DHT key. The use of the DHTs covers certain desired features such as lookup and prevention of lookup attacks, availability replication, and prevention of malicious data overwrites using the write-policy verification. Due to the basic construction of the base architecture, Cachet proposed a gossip-based social caching algorithm used in combination with an underlying DHT. This is used to leverage on the social trust relationships to improve the performance and reliability when downloading and reconstructing a social contact's wall or an aggregated newsfeed, a process which would otherwise be a lengthy process because of the decryption process. It supports user profiles and wall features such as status updates, wall posts from the other social contacts, post comments as well as newsfeeds. Policies are given through user identities or attributes. The identity-based policies are set to define user-specific access while attribute-based policies define group-access of social contacts that share common features. There are basically three types of policies defined on objects, namely, read, write and append policies, defined by the owner of the object at creation time and stored in the object metadata.

Security features Cryptographic techniques are used to enforce data confidentiality and objects to represent data. It uses policies to enforce security, and these are given based on the user identities or attributes. The identity-based policies are set to define user-specific access while attribute-based policies define group-access of social contacts that share common features. Three types of policies, read, write and append, are defined on objects by the owner of the object at creation time and stored in the object metadata. The access policies are enforced cryptographically through a hybrid scheme utilizing traditional public keys and attribute-based encryption (ABE). With ABE, an object is encrypted using an AB policy. The ABE scheme used for Cachet is an extended version of EASiER [239] that provides supports for efficient revocation for Ciphertext Policy Attribute-based Encryption [240] with the help of a minimally trusted proxy. For the hybrid mode, that is, when utilizing the social links, message encryption is by a randomly chosen symmetric encryption key, further encrypted with ABE. The Read policy is placed in the object reference instead of the object itself, enforcing policy privacy from storage nodes. Hence, the social graph is hidden from the storage nodes as authorization does not reveal identities of users. Therefore

the storage nodes are unaware of the identities of the users that store and retrieve data from it. The Write and Append policies are enforced through access control of the corresponding signature keys. Encryption for the Write policy key is by the object owner and Append policy with an AB policy. The storage nodes verify the Write Policy on objects.

While it is to highlight that Cachet fulfills the requirements we state for secure storage in decentralized OSNs, it does not specify the root of trust for the security mechanisms. Through the use of social caching, data is spread but cannot be located reliably for an update or deletion. Further, the functionality is limited as the focus is on secure data handling but does not cover communication options such as unicast, multicast or publish/subscribe.

Twister

Twister [241, 242] is a microblogging architecture that leverages P2P technologies. The goal of the design is to foster scalability, resiliency to failures and attacks, independence from central authority for user registration and provision of easily usable encrypted private communications and public posts.

Architectural design The proposed system is made up of three mostly independent overlay networks. The first overlay is based on the Bitcoin protocol [161]. The second P2P network is a structured DHT overlay network based on Kademlia [89]. It provides $\langle key, value \rangle$ storage for user resources and tracker location for the third network. This DHT overlay allows for arbitrary resource storage and user retrievals, which includes profiles, avatars and posts. The resource-to-peer mapping is based on a one-way hash function that ensures deterministic resource location while ensuring even content distribution across the network. The third network is a collection of possibly disjoint "swarms" of followers. This swarm mechanism is used for distributing new posts and it solves the problem of efficient notification delivery of new posts to users thus sparing the followers of the need to poll on a certain address of the DHT network to check for updates. The swarm is a modified BitTorrent P2P unstructured overlay network.

Security features The use of the Bitcoin protocol provides decentralized and secure user registration through the use of the Blockchain mechanism thus avoiding the need for a central authority. To enforce privacy and prevent compromise, the one-way hash function used for the resource-to-peer mapping is performed on the user's IP address and port number instead of on the user's username only.

One drawback of BitTorrent and variants is that while it is optimized for a fast delivery, it does not support data availability. If no node is available in the swarm, the file is not available. Also, the update or deletion of content is not considered, as it is not needed in a file sharing scenario, but essential in a social networking scenario.

DiDuSoNet

DiDuSoNet [189] is a DOSN that is developed on a P2P overlay network with the aim of taking advantage of trust relationships to enforce certain services such as trustness, information diffusion and data availability.

Architectural design DiDuSoNet designers present a two-tier level system. The first level is a Dunbar-based P2P social overlay and the second level is the DHT. In the, Dunbar-based social overlay, connections between the nodes are akin to the social relations of the ego networks of the users which were first identified by the psychologist Robin Dunbar. In social overlays (SOs), nodes of a P2P system only connect to one another if their owners are friends. DiDuSoNet leverages a social aspect called Dunbar approach [37] in the SOs, which considers the fact that a user stably maintains approximately 150 friend connections at any given time. This number is referred to as the Dunbar number. An ego network [243] is a network consisting of an actor (ego) and other actors that he is connected to (alters), and an ego network can be quite large. By reducing an ego network using the Dunbar number, the result is a Dunbar-based ego network. The DHT makes lookup of other nodes easier and makes the system robust to churn. The system used Pastry [86] as the underlying overlay. Atop the DHT, a data availability service is implemented, which autonomously selects two nodes in an ego network to store each published profile. To search for profiles, a Point of Storage (PoS) list called the PoS table stores the Overlay IDs of all PoSs for a given ID (*SocialID*). Data stored inside the Dunbar-based Social Overlays is private and only visible to friends. Private data is stored at the owner's node.

Security features To hide the overlay IDs of their PoSs to prevent spying by undesired nodes, the authors suggest having the nodes use an attribute-based encryption (ABE) scheme or a ciphertext-policy attribute-based encryption (CP-ABE) [240] scheme which mitigates the access of the PoSs list to selected friends only. Also to prevent unauthorized access of data, the authors suggest the use of asymmetric keys.

However, in this work only the secure data storage is considered, further elements of a DOSN are left out, such as communication, applications and a real implementation. Also, as previous examples, this solution requires that users share their data with their friends, which requires abandoning confidentiality for the sake of availability.

SEDOSN

SEDOSN [244] was designed to provide a secure decentralized OSN framework based on P2P technology.

Architectural features The application consists of three layers: an overlay network layer, function layer and the user interface layer. The network layer is designed using TomP2P[12], an open source DHT library, and a P2P network that connects the peers is built upon the physical network, making the peers independent of the physical network. The function layer consists of four modules: a User Relationship Module made using SQLite (a lightweight database) for managing the metadata of the relationships of the users; the Attribute Encryption Module that utilizes a modified RW's [245] attribute-based encryption (ABE) algorithms; the BitTorrent Module for efficient transfer of shared file; and the Storage Service Module to store and get objects in the P2P network. The User Interface layer is designed using the JavaFX technique which supports creating and delivery of rich internet applications.

Security features The modified RW's ABE algorithms incorporates discretionary authorization to enable fine-grained access control on users' data. Users generate a public key and a secret master key. A secret decryption key for each of the friends based on the friends attributes is generated using the secret master key. Messages are encrypted using the access policy, and decrypted using the appropriate secret decryption key provided by the file owner.

The system focuses on ensuring secure file sharing without compromising the privacy setting of the users. However, the system seems to lack advanced functionalities such as chatting and messaging which are standard in most OSNs, while offering only file sharing services.

Blogracy

Blogracy [246] is a microblogging social networking system that is focused on achieving anonymity and resilience of censorship, content authentication and activity stream semantic interoperability.

Architectural design It has a modular architecture built on two components: an underlying BitTorrent module for basic file sharing and an OpenSocial application programming interface (API). The BitTorrent module provides four key services to the OpenSocial container: StoreService for new key-value pairs storage request handling in the DHT; LookupService for searching values associated with a requested key in the DHT; SeedService for seeding newly shared file; and DownloadService for alerting users of the availability of a requested file. The P2P file sharing mechanisms utilizes two logically separated DHTs. The first DHT maps the user's identifier, including a reference, to his activity stream which is represented in a standard format that is encoded in a JSON file. This JSON file contains a reference to the user's profile and references to user generated content, which are in the form of Magnet-URIs. The references are the keys to the second DHT, which are resolved as actual files. The OpenSocial module implements the social aspects of Blogracy via a web application and as indicated, relies on the services that the BitTorrent module provides. The OpenSocial containers design is based on the Model-View-Controller architecture. The controller's function is to distribute responsibilities for various operations in key classes. The system is also built to support core functionality extensions by use of autonomous agents thus providing recommendations on users and content, personalized results and trust negotiation mechanisms. Semantic interoperability is also possible as it uses activity streams and weak semantic data formats for contacts and profiles, hence can be integrated into existing social platforms such as Twitter and RSS-based content streams, as either data source or a data sink.

Security features Blogracy strives to offer anonymity and pseudonymity while ensuring content is verifiable authentic and has integrity using two methods. The first method uses the public key as the user's identifier. Then the user signs his/her messages and indexes so that verification of authenticity and integrity are easily done by receivers. The second method is the use of a cryptographic hash of the public key, and for Blogracy, the hash function corresponds to the one used by the DHTs. The proposed system also utilizes attribute-based encryption to prevent unauthorized data access (posts, contacts, communications and activities), based on the Cyphertext-Policy Attribute-Based Encryption (CP-ABE) protocol [240]. Different levels of confidentiality for each individual social activity are allowed. The content creator releases the content with parameterized attribute credentials directly to acknowledged followers by encrypting the content using the public key of the followers. To support anonymity, Blogracy is implemented on I2P [247], which is an anonymizing P2P overlay network that implements a protocol resembling Tor [248]. Tor (the Onion Router) is a networking technology that is developed with the aim of guaranteeing some level of anonymity for the users by hiding their real network location.

Blogracy thus pushes the content to the followers, which is typical to microblogging, but different to social networks, where data is pulled and browsed. This requires a reliable data storage, which is not guaranteed in this case.

[12] https://tomp2p.net.

Comparative Analysis and Summary

In this section, we consider the various aspects of the systems, taking into consideration the functional requirements and non-functional requirements. We systematically assess each of the proposals and compare the contributions made and milestones met by each in realizing a fully distributed, secure and scalable OSN. This comparison will take into consideration two aspects that will be evident, that is, the overlay (single-overlay distributed/hybrid and multi-overlay) and the services offered (mixed services and microblogging). The term mixed services is used to denote an OSN that offers more than one type of service to the consumers such as chatting, messaging, audio-visual communications, (micro)blogging and so on. Microblogging systems in this case are only limited to offering a platform for microblogs to the consumers. We also briefly consider the developmental timelines of the proposed solutions, inherent trends that may not be directly visible from, as well as the status of the proposal.

OSN Requirements and System Status

The requirements for OSNs have been defined in section "Design Requirements for OSNs". Accordingly, each P2P-based OSN has been objectively analyzed to show what requirements are met by the implemented features and the analysis is presented in Table 8. Two important aspects are taken into account during the analysis which must be mentioned for clarity. The first aspect is on the requirements presented. Based on the literature available for any proposal, in cases where a suggestion is made to use a particular solution to realize any desired functionality, the assumption was made that the proposed feature was not implemented and consequently any affected requirements were not met. The second aspect is the system status. Although it may immediately be assumed that the OSN may have actually been implemented, our analysis took into account the presence of irrefutable evidence of the existence of a prototype or system deployment. The discussion that follows takes into consideration the type of overlay.

Single-Overlay Distributed OSNs

These OSNs are designed on a single overlay (structured or unstructured) and rely on a distributed indexing mechanism for resource location. We consider the solutions based on the type of structure individually.

(a) *Structured overlays* The indexing mechanism are based on keys and hence most of the solutions proposed are DHT-based. In this group, there were seven (7) OSNs that were identified, of which only Megaphone was a microblogs. The rest, LibreSocial, Porkut/My3, eXO, DECENT, PESCA and WebP2P all offer mixed services. In terms of achieving the functional requirements, with the exception of Megaphone and WebP2P, all the remaining five proposals met all the requirements. However, the numbers of the proposals that meet all the non-functional requirements is drastically different, with only LibreSocial standing out in this category. Complete privacy is not achieved by Porkut/My3, Megaphone, and eXO, and complete security is not met by Porkut/My3 and eXO. Metering is only implemented in LibreSocial. Of these OSNs, only LibreSocial and WebP2P have prototypes.

(b) *Unstructured overlays* There was only one proposal in this category, that is PAC'nPOST which is a microblog. It did not meet all the six functional requirements, lacking shared storage space interaction. It also fails to meet all the non-functional requirements, in addition to the fact that the system status is not known.

Single-Overlay Hybrid OSNs

The OSNs in this category are also designed on a single overlay (structured or unstructured), while the indexing mechanisms relies on a hybrid of distributed and centralized mechanisms. Because of the incorporation of centralized solutions in these OSNs, there is a general tendency not to meet all the requirements as the centralized mechanisms reintroduce some of the challenges faced in centralized OSNs. The solutions are discussed based on the base overlay.

(a) *Structured overlays* Five proposals are discussed here, PeerSoN, Safebook and LotusNet being OSNs with mixed services, while Cuckoo and HorNet are microblogs. LotusNet meets all the defined functional and non-functional requirements. Both HorNet and Cuckoo meets all functional requirement but not the security and privacy requirements. Safebook guarantees privacy and security although it does not meet some functional requirements. However, none of the OSNs offers metering as a non-functional requirement. We note that the only proposal with a prototype is Cuckoo.

(b) *Unstructured overlays* The proposals in this classification are SuperNova, HPOSN and the proposal by Tran et al. [235] offering mixed services, and Litter and Vegas as the microblogs, which incidentally are the only proposals that have prototypes. Three of these proposals, Litter, SuperNova and Tran et al. [235] meet all functional requirements. Two proposals Vegas and HPOSN achieve all privacy requirements as well as security requirements. Litter achieves all security

Table 8 System status, functional and non-functional requirements

Overlay structure	Type	Proposal	Services	System status	Requirements													
					Functional[a]						Non-functional							
					PSM	SCM	SGT	Com	SSI	SF	Privacy[b]				Security[c]			Metering
											Cf	OP	SIP	AP	CCA	DIA	NR	
Single-overlay distributed	Structured	LifeSocial. KOM/ LibreSocial	Mixed services	Prototype	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
		Porkut/My3	Mixed services	–	✓	✓	✓	✓	✓	✓		✓	✓	✓		✓	✓	
		Megaphone	Microblogging	–	✓	✓	✓	✓		✓	✓				✓	✓	✓	
		eXO	Mixed services	–	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓		✓	
		DECENT	Mixed services	–	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
		PESCA	Mixed services	–	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
		WebP2P	Mixed services	Prototype	✓	✓		✓		✓	✓	✓	✓	✓	✓	✓	✓	
	Unstructured	PAC'nPOST	Microblogging	–	✓	✓	✓	✓		✓								
Single-overlay hybrid	Structured	PeerSoN	Mixed services	–	✓	✓		✓	✓	✓	✓				✓			
		Safebook	Mixed services	–	✓	✓		✓		✓	✓	✓	✓	✓	✓	✓	✓	
		Cuckoo	Microblogging	Prototype	✓	✓	✓	✓	✓	✓	✓					✓	✓	
		HorNet	Microblogging	–	✓	✓	✓	✓	✓	✓	✓				✓	✓	✓	
		LotusNet	Mixed services	–	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
	Unstructured	Litter	Microblogging	Prototype	✓	✓	✓	✓	✓	✓	✓				✓	✓	✓	
		SuperNova	Mixed services	–	✓	✓	✓	✓	✓	✓								
		Vegas	Microblogging	Prototype	✓	✓		✓		✓	✓	✓	✓	✓	✓	✓	✓	
		Tran et al. [235]	Mixed services	–	✓	✓	✓	✓	✓	✓								
		HPOSN	Mixed services	–	✓	✓		✓		✓	✓	✓	✓	✓	✓	✓	✓	
Multi-overlay		Cachet	Mixed services	–	✓	✓		✓		✓	✓	✓	✓	✓	✓	✓	✓	
		Twister	Microblogging	Deployed	✓	✓		✓		✓	✓	✓			✓	✓	✓	
		DiDuSoNet	Mixed services	–	✓	✓	✓	✓	✓	✓								
		SEDOSN	Mixed services	Prototype	✓	✓		✓		✓	✓	✓	✓	✓	✓	✓	✓	
		Blogracy	Microblogging	Prototype	✓	✓		✓		✓	✓	✓	✓	✓	✓	✓	✓	

[a]Functional requirements: (*PSM* personal storage management, *SCM* social connection management, *SGT* social graph traversal, *Com* means of communication, *SSI* shared storage space interaction, *SF* search facilities)

[b]Non-functional requirement: privacy (*Cf* confidentiality, *OP* ownership privacy, *SIP* social interaction privacy, *AP* activity privacy)

[c]Non-functional requirement: security (*CCA* cover channel authentication, *DIA* data integrity and authenticity, *NR* non-repudiation)

Table 9 Developmental timeline of the P2P OSN proposals

Year	Proposal	Services	Institution
2008	LifeSocial.KOM / LibreSocial	Mixed services	TU Darmstadt
2009	PeerSoN	Mixed services	TU Berlin/EPFL/NTU Singapore
	Safebook	Mixed services	TU Darmstadt
2010	Cuckoo	Microblogging	Univ. of Göttingen/Nanjing Univ./Fudan Univ.
	Megaphone	Microblogging	California State University Long Beach
	Porkut	Mixed services	EPFL, Switzerland
2011	eXO	Mixed services	Univ. of Patras/Univ. of Ioannina
	My3	Mixed services	EPFL, Switzerland
	Litter	Microblogging	Univ. of Florida
2012	Vegas	Microblogging	Ludwig-Maximilians-University Munich
	Cachet	Mixed services	Univ. of Illinois
	DECENT	Mixed services	Univ. of Illinois
	HorNet	Microblogging	Univ. Oberta de Catalunya
	LotusNet	Mixed services	Univ. degli Studi di Torino
	PAC" nPOST	Microblogging	Univ. College London
	SuperNova	Mixed services	NTU Singapore
2013	twister	Microblogging	PUC-Rio
2015	PESCA	Mixed services	Isfahan Uni. of Tech./Foulad Inst. of Tech./ Ryerson Univ.
	Tran et al. [235]	Mixed services	Intl. Univ., Ho Chi Minh
	DiDuSoNet	Mixed services	Univ. of Pisa/IIT-CNR Pisa/Univ. of Düsseldorf
	WebP2P	Mixed Services	Univ. of Düsseldorf
	SEDOSN	Mixed Services	Peking University
	HPOSN	Mixed Services	Shandong Normal University
2016	Blogracy	Microblogging	Univ. of Parma

requirements although it does not meet the privacy requirements. Metering requirement is not met by any of the proposals.

Multi-Overlay OSNs

This group of OSNs is interesting because the solutions are designed to utilize more than one overlay so achieve functionality. The solutions seek to combine the advantages offered by different overlays in combination to overcome the disadvantages seen in each individual overlay. The overlays utilized may be structured only, unstructured only, or a combination of structured and unstructured overlays. Five proposals are analysed that fall in this category, that is, Cachet, DiDuSoNet and SEDOSN that offer mixed services, and Twister and Blogracy which are microblogs. SEDOSN and Blogracy have prototype implementations while Twister[13] is the only P2P OSN that is in active deployment. DiDuSoNet is the only proposal that meets all the functional requirements, but is the only proposal in this category that fails in meeting all the non-functional requirements. From the remaining four proposals, only Twister fails to meet all privacy requirements. No proposal includes the ability for metering for measurements of system health.

One interesting aspect that is visible from a cursory look at the functional requirements is that at a minimum, all proposed solutions met four functional requirements, that is, personal storage management, social connection management, communication and search facilities.

Developmental Progression

In [11], the authors analyzed selected OSNs from 1997 to 2011, indicating a boom between 2003 and 2006. Table 9 is a summary of the P2P OSNs highlighting the timelines and current status of the proposed implementation over the period of the analysis (2008–2016). The boom of the P2P OSN platforms is seen to be clustered between 2011 and 2013, with the majority being in the year 2012. Although the P2P OSNs may seem to be a decade too late, it is most probable that their development may be a direct result of concerns observed in centralized OSNs during the previous decade, in particular, privacy, security and scalability. As has been shown, the different P2P OSNs aim at meeting one or more of these concerns using various techniques. It

[13] http://twister.net.co.

Table 10 Components realised in the P2P-based OSNs

Structure	Type	Proposal	Infrustructural Aspects				
			Overlay	Storage mechanism	Lookup/search mechanism	Data redundancy mechanism	Pub/sub mechanism
Single-overlay distributed	Structured	LifeSocial. KOM/ LibreSocial	Pastry	Standard—PAST; Advanced - Sets, linked lists, prefix hash trees	Semantic-free	Replication & caching	✓
		Porkut/My3	OpenDHT	Trusted Proxy Set (TPS)	Semantic-free	Replication to TPS	✓
		Megaphone	Pastry	Local storage	Semantic-free	Replication	✓
		eXO	Pastry/Chord	Local storage	Semantic free	Replication to adjacent nodes	
		DECENT	Pastry/Kademlia	DHT for object storage	Semantic-free	Replication & versioning	✓
		PESCA	DHT-based	Local storage	Semantic-free	Replication based on friends' user online times	✗
		WebP2P	Chord	Local storage	Semantic-free	Replication	✗
	Unstructured	PAC'nPOST	Unstructured	Local storage	Semantic (probabilistic search)	Replication	✓
Single-overlay hybrid	Structured	PeerSoN	OpenDHT	Local storage	Semantic-free	Replication	✗
		Safebook	KAD	Nodes in concentric "Matryoshka"-like circles	Semantic-free	Replication	✗
		Cuckoo	Pastry	Local and server cloud	Semantic (flooding) for influentials else semantic-free	Replication to servers	✓
		HorNet	Pastry	Storage service API based on DHT	Semantic-free	Replication	✓
		LotusNet	Likir	Local and set of trusted contacts	Semantic-free	Replication on trusted contacts	✓
	Unstructured	Litter	Social overlay	Local storage	Semantic (pseudo-random walk)	Replication to one-hop peers	✓
		SuperNova	Super-peers	List of users	Search queries sent to super peers	Replication	✗
		Vegas	Unstructured	Web-based datastores	Search queries to subset of friends	Replication of datastores	✓
		Tran et al. [235]	Gnutella	MySQL database at local nodes	Semantic (flooding with TTL)	Replication	✓
		HPOSN	Social overlay	Local storage and cloud servers	Queries to server based on stored index	Replication on servers	✗

Table 10 (continued)

Structure	Type	Proposal	Infrastructural Aspects				
			Overlay	Storage mechanism	Lookup/search mechanism	Data redundancy mechanism	Pub/sub mechanism
Multi-overlay		Cachet	Pastry/Kademlia & Social overlay	Local storage	Semantic-free	Replication	✓
		Twister	Bitcoin, Kademlia, BitTorrent swarm	Local storage, BitTorrent network	Semantic-free	Replicaion	✓
		DiDuSoNet	Pastry & social overlay	Local storage	Semantic-free	Replication to ego network	✓
		SEDOSN	TomP2P & BitTorrent	Local storage, SQLite database for metadata	BitTorrent protocol (Client/Tracker)	–	✗
		Blogracy	Two BitTorrent DHTs	Local storage	Semantic-free	Replication	✓

is also evident that most of these systems were developed in academic environments.

Essential P2P Components

In section "Peer-to-Peer Networks", a general overview was given in consideration of key components that must be considered during the design of any P2P-based OSN. In line with that, in Table 10, we present a summary of the components realised in the surveyed P2P OSNs. The key components that were considered were the overlay, storage mechanisms implemented, lookup/search mechanisms, data redundancy mechanisms utilized and inclusion of a publish/subscribe mechanism.

Single-Overlay Distributed OSNs

We consider the structured and unstructured overlays separately and discuss them herein.

(a) *Structured* Most of the proposals were based on common DHT-based overlays, such as Pastry, Chord and Kademlia, to form the network. Therefore, the lookup mechanics were based on key-based routing, hence semantic free. The proposals all implemented replication in differing ways so as to guarantee data availability, as the local nodes replicated the local data to other nodes in the network using appropriate algorithms. Only LibreSocial included advanced storage features, that is, distributed sets, linked lists and prefix hash trees. Only WebP2P and eXO did not include the publish/subscribe mechanisms as part of the features.

(b) *Unstructured* The only solution here, PAC'nPOST is based on a purely distributed overlay, hence pure P2P (unstructured). Therefore, semantic searching using probabilistic search methods is performed. Each node handles its own data but replicates it to other nodes to for data availability guarantees, in addition to implementing a pub/sub mechanism.

Single-Overlay Hybrid OSNs

The component discussion here also considers the base overlay and follows.

(a) *Structured* These OSNs had differing methods in how the storage is handled, but all use replication to support data availability. PeerSoN uses local storage with replication to other nodes based on OpenDHT's algorithm. Safebook utilizes nodes directly connected to the local node and arranged in a concentric circle as the replicating nodes, while Cuckoo uses the server cloud to support data replication. HorNet implements a DHT that relies on a storage service API and LotusNet relies on the local DHT and replicates to trusted contacts. PeerSoN, Safebook, HorNet and LotusNet rely on semantic-free lookup mechanisms for data location, with Cuckoo relying on a combination of semantic-free lookup and semantic-based search. Finally, Cuckoo, HorNet and LotusNet incorporate a pub/sub mechanism.

(b) *Unstructured* The OSNs in this category implement different techniques to handle storage, and no two OSNs have the same method. This is expected in unstructured overlays as the network does not offer any distributed data structures such as DHTs, but allows the designers to develop novel techniques to handle data management. Data availability is guaranteed

Table 11 Security features of the P2P-based OSNs

Overlay structure	Structure type	Proposal	Security aspects					
			Identity creation	Identity verification	Access Control	Confidentiality	Integrity	Anonymity
Single-overlay distributed	Structured	LifeSocial. KOM/ LibreSocial	Public key as Unique ID	Public key	User-centric settings + Access Control Lists	Symmetric encryption	Digital signatures	✓
		Porkut/My3	No information	No information	✗	✗	✗	✓
		Megaphone	Concatenated hash of username and public key	Public key	Session keys + Asymmetric encryption	Asymmetric encryption	Digital signatures	✗
		eXO	Hash on user-specific detail	User ID	User-centric configurations	✗	✗	✓
		DECENT	Random ID as User ID	User ID	Attribute-based policies	Asymmetric attribute-based encryption	Digital signatures	✓
		PESCA	Hash of user's email address	Global ID	Virtual ID + symmetric key	Broadcast encryption	Digital signatures	✗
		WebP2P	Asymmetric key from username and passwordA	Public key as Chord ID	Identity-based access control	Asymmetric encryption	Digital signatures	✗
	Unstructured	PAC'nPOST	No information	No information	✗	✗	✗	✗
Single-overlay hybrid	Structured	PeerSoN	Hash of user's email	Globally unique ID (GUID)	✗	Asymmetric encryption	✗	✗
		Safebook	TIS-generated ID and pseudonym	Node ID and pseudonym	Attribute-based policies	Asymmetric encryption	Digital signatures	✓
		Cuckoo	Server generated ID	Not required	✗	✗	✗	✗
		HorNet	Public key	Public key certificate	Access control list	✗	Digital signatures	✗
		LotusNet	Public key and OpenID to obtain certificate	Likir ID	Access control grants	Nonce-based two-way authentication	Digital signatures	✓

Table 11 (continued)

Overlay structure	Structure type	Proposal	Security aspects					
			Identity creation	Identity verification	Access Control	Confidentiality	Integrity	Anonymity
	Unstructured	Litter	No information	User ID (UID)	✗	Encrypted IP tunnels	Digital signatures	✗
		SuperNova	Username, location and interests	Userlist at superpeers	✗	Threshold-based secret sharing [249]	✗	✗
		Vegas	No information	No information	Asymmetric encryption	Symmetric + asymmetric encryption	Digital signatures	✓
		Tran et al. [235]	Registration servers	Super peers	super peer controlled	✗	✗	✗
		HPOSN	Servers	Globally unique ID	Asymmetric encryption	Asymmetric encryption (Onion routing)	✗	✗
Multi-overlay		Cachet	User ID	User ID	Attribute-based policies	Asymmetric attribute-based encryption	Digital signatures	✓
		Twister	Unique user ID	Username and password	✗	Elliptic Curve Integrated Encryption Scheme	Digital signatures	✗
		DiDuSoNet	Unique Social ID	Social ID	✗	✗	✗	✗
		SEDOSN	User's email address	Global ID	Attribute-based encryption	Symmetric encryption	Digital signatures	✓
		Blogracy	Public key and username	Public key	Ciphertext-policy attribute-based encryption	Asymmetric encryption	Digital signatures	✓

by replication in all cases, and with the exception of SuperNova and HPOSN, the remaining OSNs integrate a pub/sub mechanism.

Multi-Overlay OSNs

The proposed OSNs in this group all incorporate an additional data storage mechanisms in addition to the local storage that relies on the DHT mechanisms. Twister incorporates the BitTorrent network and SEDOSN incorporates an SQLite database to handle metadata, while the rest rely solely on the local storage. All solutions with the exception of SEDOSN, for which no information was provided, ensure data availability via replication, and similarly with the exception of SEDOSN, all include a pub/sub mechanism.

Security Considerations

Any discussion about OSNs without paying special attention to the aspect of security management is incomplete. In Table 11, a summary of key security features identified in the analyzed P2P OSNs is shown. At the minimum, it is desired that they provide some form of identity creation and verification, include an access control mechanism, guarantee confidentiality and ensure data integrity, while ensuring user anonymity. In the survey of the P2P OSNs, it is seen that only LibreSocial, DECENT, LotusNet, Safebook, Cachet, SEDOSN and Blogracy incorporate all the required security mechanisms to ensure secure communications and guarantee user privacy. However, it is important to note that

the microblogs generally have a tendency to not implement all the security requirement because, as an unwritten rule, microblogs do not guarantee privacy, as users are able to access all the messages in the network, and in many cases, view the profiles of other users whether known directly/indirectly or not known and as well as follow/unfollow other users.

Lessons Learned

Building social networks that are designed to operate in a fully distributed environment is not a new idea and has been studied quite extensively. In particular, using P2P networks as a platform for building decentralized online social networks (DOSNs) has been taunted as a solution to the problems due to the accumulated costs for centralized operations [4, 13] and security and privacy concerns [5, 19]. However, as has been show, any functional P2P-based OSN must at least achieve desired functional requirements for OSNs and at best the non-functional requirements so as to effectively address the concerns raised (see section "Design Requirements for OSNs") while ensuring users enjoy the best services. During the course of undertaking this study and compiling this survey, some lessons were learned that are worth considering.

Need for agreeable P2P protocol standards Different proposed solutions achieve the functional and non-functional requirements using different combinations of P2P mechanisms. This is an indication of the need for adoption of a standard for P2P technologies. An attempt has been made to create a standard by the Internet Architecture Board (IAB) that was published as RFC 5694 [250] in November 2009. However, this standard was based on the early P2P technologies which have since experienced a considerable metamorphosis. Therefore, a newer standard must be tabled within the research community and eventually adopted. That said, the fact that there is much that has been done is an indication of the diversity of solutions for any given problem in P2P technology, and the ease in which it can be adapted to achieve a desired functionality. In general, most applications, and in particular OSNs, designed on the P2P platform, seem to follow a general format for the architecture: *an overlay*, required services on top of the overlay (here referred to as *the framework*) and *the application*. Similar layouts have been suggested in [15] This may well be a precursor to an adoptable standard for P2P applications. In line with this need to achieve a standard, we have undertaken a general survey of individual P2P component solutions that have been proposed in literature that we see meet the basic technical requirements for the P2P architecture (defined in section "Technical Requirements for a P2PFramework for Social Networks").

From research systems to active systems Research on P2P-based OSN has matured and many of the proposals have to a large degree aimed at solving one or more challenge experienced in the centralized OSNs in a unique manner. As yet, with few exceptions, the viability and behavior of these proposed systems in the real world is yet to be seen. To compete with the centralized OSNs, the designers of the P2P-based OSNs must strive to ensure that the product they offer gets to the users and find ways to inform users of their presence. Although in many cases the focus has been on a secure and private solution, there is need for finding a balance between security/privacy and system usability. This may be achievable by having the proposal deployed and monitoring the behavior of the users against the behavior of the system and then making appropriate adjustments.

Motivating user communities Even though the proposed systems are brought online, it is another thing to get users to start using them. DOSNs have been around for some time yet the user communities have not grown at the same rate as centralized OSNs. DOSNs founded on federated solutions such as Diaspora[14], Friendica[15] and Mastodon[16] have been able to get user communities but the numbers are far much smaller than those of the centralized OSNs. Generally, DOSNs promise to offer many features such as more security and privacy control, but the greatest uphill task faced is convincing the users of centralized OSN users to migrate and use them, as the centralized OSNs have large, established user bases, are easily accessible worldwide, and boast of a mature infrastructure [5]. The ability to monetize the user's data by the providers of the OSNs stands out as a major reason for the continued growth and establishment of centralized OSNs. Thus far, it appears to the users that the accrued benefits of using centralized OSNs are way better than what DOSNs offer.

Pure vs hybrid In the early days of P2P technology, most proposed P2P-based OSNs tended towards two directions: either pure P2P (structured, unstructured or combinations) or P2P augmented with centralized technology (hybrid). A major drawback experienced in pure P2P applications is the need for an always online bootstrap node, which may not always be achievable, to ensure that the network formed is kept alive which in turn affects profile and content availability as well as content distribution [4]. On the other hand,

[14] https://diasporafoundation.org/.

[15] https://friendi.ca/.

[16] https://joinmastodon.org/.

hybrid P2P systems, despite overcoming these challenges faced in pure P2P systems, reintroduce the shortfalls of centralized systems that affect the OSNs. Therefore, solutions proposed in either line must weigh the pros and cons associated, and what system designers are willing to make compromises vis-à-vis what the users are willing to tolerate.

Conclusion

This survey was divided into three major sections. The first section laid a foundation for the P2P-based OSNs by introducing social networks in general, providing a clear road map for the entire study. The second section of the study was a comprehensive breakdown and discussion of the key enabling features of P2P networks that support implementation of applications, and in particular, implementation of online social networks. However, the technology is fast evolving and there are many changes that have since been observed. This section highlighted the various developments and proposals that have been presented in literature, considering each key feature. The final section was an analysis of selected proposed P2P-based OSNs, highlighting key P2P features implement.

From the analysis done on the P2P-based OSNs, there are some positives that can be highlighted. It is important to note that the P2P-based OSNs, have been as a result of seeking to bring to face a solution that meets the shortfalls seen in the centralized OSNs, by providing a decentralized platform that was not only privacy-preserving, secure and scalable but also achievable. The P2P platform has shown capabilities of meeting all these goals, but only after the implementation of novel solutions to meet application specific challenges that guarantee robustness, storage, data availability, reliable communication as well as security. A brief summary of some of the solutions to achieve this have been discussed.

From the analysis of the P2P-based solutions, it is seen that some of these proposals fail in meeting all the minimum requirements to guarantee maximum user experience. The analysis considered if the defined requirements (functional and non functional) are achieved. Further, the P2P components implemented in the proposals were compared. Additionally, the security features were analyzed as a key ingredient in the OSNs, and in particular because the P2P-based OSNs are designed with a goal of being able to provide security and privacy. A major concern observable with all the P2P-based OSNs, excluding Twister, is that all of them are not online, or indeed have never been online. This may probably be due to several factors: (a) most P2P-based OSNs have been research projects so that beyond the initial work no one undertakes further development, (b) the number of critical network users in the P2P-based OSNs is not easily achieved (chicken–egg problem), and (c) lack of monetization of the P2P-based OSNs hence no motivation for further system development due to the absence of any meaningful financial gain. This means that although in theory, these systems are better than the current centralized implementations, they are not seen to make an impact in terms of user communities that utilize them.

Funding Open Access funding provided by Projekt DEAL.

Compliance with Ethical Standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Ajami R, Ramadan N, Mohamed N, Al-Jaroodi J. Security challenges and approaches in online social networks: a survey. Int J Comput Sci Netw Secur (IJCSNS). 2011;11(8):1–12.
2. Heidemann J, Klier M, Probst F. Online social networks: a survey of a global phenomenon. Comput Netw. 2012;56(18):3866–78.
3. Caviglione L, Coccoli M, Merlo A. A taxonomy-based model of security and privacy in online social networks. Int J Comput Sci Eng. 2014;9(4):325–38.
4. Maqsood T, Khalid O, Irfan R, Madani SA, Khan SU. Scalability issues in online social networks. ACM Comput Surv. 2016;49(2):40:1–42.
5. Kayes I, Iamnitchi A. Privacy and security in online social networks: a survey. Online Soc Netw Media. 2017;3–4:1–21.
6. Chen Y, Cheung ASY. The transparent self under big data profiling: privacy and chinese legislation on the social credit system. J Comparat Law. 2017;12(2):326–78 (University of Hong Kong Faculty of Law Research Paper No. 2017/011).
7. Liang F, Das V, Kostyuk N, Hussain MM. Constructing a data-driven society: China's social credit system as a state surveillance infrastructure. Policy Internet. 2018;10(4):415–53.
8. Li M, Zhu H, Gao Z, Chen S, Yu L, Hu S, Ren K. All your location are belong to us: breaking mobile social networks for automated user location tracking. In: Proceedings of the 15th ACM international symposium on mobile ad hoc networking and computing, MobiHoc '14; 2014, pp. 43–52.
9. Ahmad S, Asghar MZ, Alotaibi FM, Awan I. Detection and classification of social media-based extremist affiliations using sentiment analysis techniques. Hum-Centric Comput Inf Sci. 2019;9(1):24.

10. Almoqbel M, Xu S. Computational mining of social media to Curb terrorism. ACM Comput Surv. 2019;52(5):87:1–25.
11. Pallis G, Zeinalipour-Yazti D, Dikaiakos MD. Online social networks: status and trends. New Direct Web Data Manag. 2011;1:213–34.
12. Juste PS. A peer-to-peer architecture for social networking applications. In: Ph.D. thesis, University of Florida, Herbert Wertheim College of Engineering, Department of Electrical and Computer Engineering; 2014.
13. Guidi B, Conti M, Ricci L: P2P architectures for distributed online social networks. In: International conference on high performance computing & simulation, HPCS 2013, Helsinki, Finland; 2013, pp. 678–681.
14. Korzun D, Gurtov A. Hierarchical architectures in structured peer-to-peer overlay networks. Peer-to-Peer Netw Appl. 2014;7(4):359–95.
15. Paul T, Famulari A, Strufe T. A survey on decentralized Online Social Networks. Comput Netw. 2014;75:437–52.
16. Greschbach B, Kreitz G, Buchegger S: The devil is in the metadata—new privacy challenges in Decentralised Online Social Networks. In: 2012 IEEE international conference on pervasive computing and communications workshops; 2012, pp. 333–339.
17. Taheri-Boshrooyeh S, Küpçü A, Özkasap Ö: Security and privacy of distributed online social networks. In: 2015 IEEE 35th international conference on distributed computing systems workshops; 2015, pp. 112–119.
18. De Salve A, Mori P, Ricci L. A survey on privacy in decentralized online social networks. Comput Sci Rev. 2018;27:154–76.
19. Oukemeni S, Rifà-Pous H, Puig JMM. Privacy analysis on microblogging online social networks: a survey. ACM Comput Surv. 2019;52(3):60:1–36.
20. Chowdhury SR, Roy AR, Shaikh M, Daudjee K. A taxonomy of decentralized online social networks. Peer-to-Peer Netw Appl. 2015;8(3):367–83.
21. Malatras A. State-of-the-art survey on P2P overlay networks in pervasive computing environments. J Netw Comput Appl. 2015;55:1–23.
22. Spaho E, Barolli L, Xhafa F. Data replication strategies in P2P systems: a survey. In: 2014 17th international conference on network-based information systems; 2014. pp. 302–309.
23. Spaho E, Barolli A, Xhafa F, Barolli L. P2P data replication: techniques and applications. In: Xhafa F, Barolli L, Barolli A, Papajorgji P, editors. Modeling and processing for next-generation big-data technologies: with applications and case studies. Cham: Springer International Publishing; 2015. p. 145–66.
24. Risson J, Moors T. Survey of research towards robust peer-to-peer networks: search methods. Comput Netw. 2006;50(17):3485–521.
25. Kang C. Survey of search and optimization of P2P networks. Peer-to-Peer Netw Appl. 2011;4(3):211–8.
26. Zhang C, Xiao W, Tang D, Tang J. P2P-based multidimensional indexing methods: a survey. J Syst Softw. 2011;84(12):2348–62.
27. Trifa Z, Khemakhem M. Taxonomy of structured P2P overlay networks security attacks. Int J Comput Electr Autom Control Inf Eng. 2012;6(4):470–6.
28. Trifa Z, Khemakhem M. Mitigation of sybil attacks in structured P2P overlay networks. In: 2012 Eighth international conference on semantics, knowledge and grids; 2012. pp. 245–248.
29. Germanus D, Roos S, Strufe T, Suri N. Mitigating eclipse attacks in peer-to-peer networks. In: 2014 IEEE conference on communications and network security; 2014. pp. 400–408.
30. Shen X, Yu H, Buford J, Akon M, editors. Handbook of Peer-to-Peer networking. Boston: Springer; 2010.
31. Kwok YKR. Peer-to-Peer ccomputing: applications, architecture, protocols, and challenges. Boca Raton: CRC Press; 2012.
32. Garton L, Haythornthwaite C, Wellman B. Studying online social networks. J Comput-Mediat Commun. 1997;3:75–105.
33. Marin A, Wellman B. Social network analysis: an introduction. In: Scott J, Carrington PJ (eds) The SAGE handbook of social network analysis, chap. 2, pp. 11–25. SAGE Publications; 2011.
34. Aggarwal CC. An introduction to social network data analytics. In: Aggarwal CC, editor. Social network data analytics. Boston: Springer; 2011. p. 1–15.
35. Boyd D, Ellison NB. Social network sites: definition, history, and scholarship. J Comput-Mediat Commun. 2007;13(1):210–30.
36. Kietzmann JH, Hermkens K, McCarthy IP, Silvestre BS. Social media? Get serious! Understanding the functional building blocks of social media. Bus Horiz. 2011;54(3):241–51 (Special issue: social media).
37. Dunbar RIM. The social brain hypothesis and its implications for social evolution. Ann Hum Biol. 2009;36(5):562–72.
38. Robertson S, Robertson J. Mastering the requirements process: getting requirements right. 3rd ed. Boston: Addison-Wesley; 2012.
39. Young RR. The requirements engineering handbook. Norwood: Artech House; 2004.
40. Wiegers K, Beatty J. Software requirements. 3rd ed. Washington, DC: Microsoft Press; 2013.
41. Zhang C, Sun J, Zhu X, Fang Y. Privacy and security for online social networks: challenges and opportunities. Netw IEEE. 2010;24(4):13–8.
42. Aiello LM, Ruffo G. Lotusnet: tunable privacy for distributed online social network services. Comput Commun. 2012;35(1):75–88.
43. Agrawal D, El Abbadi A, Das S, Elmore AJ. Database scalability, elasticity, and autonomy in the cloud. In: Yu JX, Kim MH, Unland R, editors. Database systems for advanced applications. Berlin, Heidelberg: Springer; 2011. p. 2–15.
44. Lakshman A, Malik P. Cassandra: a decentralized structured storage system. SIGOPS Oper Syst Rev. 2010;44(2):35–40.
45. Lloyd W, Freedman MJ, Kaminsky M, Andersen DG. Don't settle for eventual consistency. Commun ACM. 2014;57(5):61–8.
46. Beaver D, Kumar S, Li HC, Sobel J, Vajgel P. Finding a Needle in Haystack: Facebook's Photo Storage. In: Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation, OSDI'10, pp. 47–60. USENIX Association, Berkeley, CA, USA 2010.
47. Chang F, Dean J, Ghemawat S, Hsieh WC, Wallach DA, Burrows M, Chandra T, Fikes A, Gruber RE. Bigtable: a distributed storage system for structured data. ACM Trans Comput Syst. 2008;26(2):4:1–26.
48. Baker J, Bond C, Corbett JC, Furman J, Khorlin A, Larson J, Leon JM, Li Y, Lloyd A, Yushprakh V. Megastore: Providing scalable, highly available storage for interactive services. In: Proceedings of the Conference on Innovative Data system Research (CIDR). Google; 2011. pp. 223–234.
49. Brandimarte L, Acquisti A, Loewenstein G. Misplaced confidences: privacy and the control paradox. Soc Psychol Personal Sci. 2013;4(3):340–7.
50. Spiekermann S, Acquisti A, Böhme R, Hui KL. The challenges of personal data markets and privacy. Electron Markets. 2015;25(2):161–7.
51. Kramer ADI, Guillory JE, Hancock JT. Experimental evidence of massive-scale emotional contagion through social networks. Proc Nat Acad Sci. 2014;111(24):8788–90.
52. Bakir V, McStay A. Fake news and the economy of emotions. Digit Journal. 2018;6(2):154–75.
53. Bekkers V, Edwards A, de Kool D. Social media monitoring: responsive governance in the shadow of surveillance? Govern Inf Q. 2013;30(4):335–42.

54. Stoycheff E. Under surveillance: examining Facebook?s spiral of silence effects in the Wake of NSA internet monitoring. Journal Mass Commun Q. 2016;93(2):296–311.
55. Roy R, Gupta N. Digital Capitalism and Surveillance on Social Networking Sites: A Study of Digital Labour, Security and Privacy for Social Media Users. In: Kar AK, Sinha S, Gupta MP, editors. Digital India: reflections and practice. Berlin: Springer International Publishing; 2018. p. 67–81.
56. Al-Qurishi M, Al-Rakhami M, Alamri A, Alrubaian M, Rahman SMM, Hossain MS. Sybil defense techniques in online social networks: a survey. IEEE Access. 2017;5:1200–19.
57. Gao H, Hu J, Huang T, Wang J, Chen Y. Security issues in online social networks. IEEE Internet Comput. 2011;15(4):56–63.
58. NaliniPriya G, Asswini M. A survey on vulnerable attacks in online social networks. In: International confernce on innovation information in computing technologies; 2015, pp. 1–6.
59. Datta A, Buchegger S, Vu LH, Strufe T, Rzadca K. Decentralized online social networks. In: Furht B, editor. Handbook of social network technologies and applications. Boston: Springer; 2010. p. 349–78.
60. Buchegger S, Datta A. A case for P2P infrastructure for social networks-opportunities & challenges. In: Proceedings of the Sixth International Conference on Wireless On-Demand Network Systems and Services, 2009. WONS 2009; 2009. pp. 161–168.
61. Paul T, Buchegger S, Strufe T. Decentralizing social networking services. In: International tyrrhenian workshop on digital communications; 2010. pp. 1–10.
62. Graffi K, Podrajanski S, Mukherjee P, Kovacevic A, Steinmetz R. A distributed platform for multimedia communities. In: Proceedings of the IEEE international symposium on multimedia (ISM'08), pp. 208–213. IEEE; 2008.
63. Janiuk J, Mäcker A, Graffi K. Secure distributed data structures for peer-to-peer-based social networks. In: 2014 International Conference on Collaboration Technologies and Systems, CTS 2014, Minneapolis, MN, USA; 2014, pp. 396–405.
64. Al-Aaridhi R, Graffi K. Sets, lists and trees: distributed data structures on distributed hash tables. In: 35th IEEE international performance computing and communications conference, IPCCC 2016, Las Vegas, NV, USA; 2016. pp. 1–8.
65. Al-Aaridhi R, Yuksektepe A, Amft T, Graffi K. Distributed data structures improvement for collective retrieval time. In: 19th international symposium on wireless personal multimedia communications, WPMC 2016, Shenzhen, China, pp. 85–90. IEEE; 2016.
66. Drozd O, Fabian B. Sharing operational costs in business peer-to-peer systems with a centralized clarke-groves service. 2017. https://ssrn.com/abstract=3005370 or https://doi.org/10.2139/ssrn.3005370.
67. Koskela T, Kassinen O, Harjula E, Ylianttila M. P2P group management systems: a conceptual analysis. ACM Comput Surv. 2013;45(2):20:1–25.
68. Buford JF, Yu H. Peer-to-peer networking and applications: synopsis and research directions. In: Shen X, Yu H, Buford J, Akon M (eds.) Handbook of peer-to-peer networking, chap. 1, pp. 3–45. Springer; 2010.
69. Androutsellis-Theotokis S, Spinellis D. A survey of peer-to-peer content distribution technologies. ACM Comput Surv (CSUR). 2004;36(4):335–71.
70. Harjula E, Ylianttila M, Ala-Kurikka J, Riekki J, Sauvola J. Plug-and-play application platform: towards mobile peer-to-peer. In: Proceedings of the 3rd international conference on mobile and ubiquitous multimedia, MUM '04; 2004. pp. 63–69.
71. Nhat BM. Searching in P2P networks: a survey. in: Tech. Rep. TKK T-110. 5190, Department of Computer Science and Engineering, School of Science and Technology, Aalto University; 2009.
72. Cooper BF. Trading off Resources Between Overlapping Overlays. In: Steen Mv, Henning M (eds) Middleware 2006, ACM/IFIP/USENIX 7th International Middleware Conference, Melbourne, Australia, Proceedings, lecture notes in computer science, vol. 4290; 2006. pp. 101–120.
73. Mao Y, Loo BT, Ives Z, Smith JM. MOSAIC: declarative platform for dynamic overlay composition. Comput Netw. 2012;56(1):64–84.
74. Artigas MS, Lopez PG, Ahullo JP, Skarmeta AFG: Cyclone: a novel design schema for hierarchical dhts. In: Fifth IEEE international conference on peer-to-peer computing (P2P'05); 2005. pp. 49–56.
75. Babaoglu O, Canright G, Deutsch A, Caro GAD, Ducatelle F, Gambardella LM, Ganguly N, Jelasity M, Montemanni R, Montresor A, Urnes T. Design patterns from biology for distributed computing. ACM Trans Auton Adapt Syst. 2006;1(1):26–66.
76. Balasubramaniam S, Leibnitz K, Lio P, Botvich D, Murata M. Biological principles for future internet architecture design. IEEE Commun Mag. 2011;49(7):44–52.
77. Amft T. The impact of resource sharing on coexisting P2P overlays and stacked overlay modules. In: Ph.D. thesis, Department of Computer Science, Faculty of Mathematics and Natural Sciences; 2017.
78. Gross C, Stingl D, Richerzhagen B, Hemel A, Steinmetz R, Hausheer D. Geodemlia: a robust peer-to-peer overlay supporting location-based search. In: 2012 IEEE 12th international conference on peer-to-peer computing (P2P); 2012, pp. 25–36.
79. Amft T, Graffi K. Moving peers in distributed, location-based peer-to-peer overlays. In: 2017 international conference on computing, networking and communications (ICNC); 2017. pp. 906–911.
80. Amft T, Guidi B, Graffi K, Ricci L. FRoDO: Friendly routing over dunbar-based overlays. In: 2015 IEEE 40th conference on local computer networks (LCN); 2015. pp. 356–364.
81. Singh A, Ngan TW, Druschel P, Wallach DS. Eclipse attacks on overlay networks: threats and defenses. In: INFOCOM 2006. 25th ieee international conference on computer communications. Proceedings. IEEE, Barcelona, Spain; 2006.
82. Giuli TJ, Maniatis P, Baker M, Rosenthal DS, Roussopoulos M. Attrition defenses for a peer-to-peer digital preservation system. In: Proceedings of the 2005 USENIX Annual Technical Conference, Anaheim, CA, USA, pp. 163–178. USENIX Association; 2005.
83. Puttaswamy KP, Zheng H, Zhao BY. Securing structured overlays against identity attacks. IEEE Trans Parallel Distrib Syst. 2009;20(10):1487–98.
84. Stutzbach D, Rejaie R. Understanding Churn in Peer-to-peer Networks. In: Proceedings of the 6th ACM SIGCOMM conference on internet measurement, IMC '06; 2006, pp. 189–202.
85. Cohen B. The bittorrent protocol specification; 2008.
86. Rowstron A. Druschel P. Pastry: scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In: Middleware 2001, IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg, Germany, Proceedings, Lecture Notes in Computer Science, vol. 2218; 2001. pp. 329–350.
87. Zhao BY, Kubiatowicz J, Joseph AD. Tapestry: an infrastructure for fault-tolerant wide-area location and routing. In: Technical report, University of California at Berkeley, Berkeley, CA, USA; 2001.

88. Stoica I, Morris R, Karger D, Kaashoek MF, Balakrishnan H. Chord: a scalable peer-to-peer lookup service for internet applications. In: Proceedings of the ACM SIGCOMM 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, San Diego, CA, USA; 2001.
89. Maymounkov P, Mazieres D. Kademlia: A peer-to-peer information system based on the xor metric. In: Druschel P, Kaashoek MF, Rowstron AIT (eds) Peer-to-peer systems, first international workshop, IPTPS 2002, Cambridge, MA, USA, revised papers, lecture notes in computer science, vol. 2429; 2002. pp. 53–65.
90. Clarke I. A distributed decentralised information storage and retrieval system. In: Master's thesis, Division of Informatics, University of Edinburgh; 1999.
91. Clarke I, Sandberg O, Willey B, Hong TW. Freenet: a distributed anonymous information storage and retrieval system. In: Federrath H, editor. Designing privacy enhancing technologies: international workshop on design issues in anonymity and unobservability Berkeley, CA, USA, July 25–26, 2000 Proceedings. Berlin, Heidelberg: Springer; 2001. p. 46–66.
92. Jennings C, Lowekamp B, Rescorla E, Baset S, Schulzrinne H. REsource LOcation And Discovery (RELOAD) base protocol. RFC 6940. 2014. https://doi.org/10.17487/RFC6940. https://rfc-editor.org/rfc/rfc6940.txt
93. Traversat B, Arora A, Abdelaziz M, Duigou M, Haywood C, Hugly JC, Pouyoul E, Yeager B. Project JXTA 2.0 super-peer virtual network. Technical report, Sun Microsystems, Inc.; 2003.
94. Leibowitz N, Ripeanu M, Wierzbicki A. Deconstructing the Kazaa network. In: Internet Applications. WIAPP 2003. Proceedings. The Third IEEE Workshop on, pp. 112–120. IEEE; 2003.
95. Liang J, Kumar R, Ross KW. The FastTrack overlay: a measurement study. Comput Netw. 2006;50(6):842–58.
96. Banerjee S, Bhattacharjee B, Kommareddy C. Scalable application layer multicast. SIGCOMM Comput Commun Rev. 2002;32(4):205–17.
97. Xu Z, Min R, Hu Y. HIERAS: a DHT based hierarchical P2P routing algorithm. In: 2003 International conference on parallel processing, 2003. Proceedings; 2003. pp. 187–194.
98. Castro M, Costa M, Rowstron A. Should we build Gnutella on a structured overlay? SIGCOMM Comput Commun Rev. 2004;34(1):131–6.
99. Apel S, Buchmann E. Biology-inspired optimizations of Peer-to-Peer overlay networks. Praxis der Informationsverarbeitung und Kommunikation. 2005;28(4):199–205.
100. Hoh CC, Hwang RH. P2P File sharing system over MANET based on swarm intelligence: a cross-layer design. In: 2007 IEEE wireless communications and networking conference; 2007. pp. 2674–2679.
101. Brocco A, Malatras A, Hirsbrunner B. Enabling efficient information discovery in a self-structured grid. Future Gener Comput Syst. 2010;26(6):838–46.
102. Forestiero A, Leonardi E, Mastroianni C, Meo M. Self-Chord: a bio-inspired P2P framework for self-organizing distributed systems. IEEE/ACM Trans Netw (TON). 2010;18(5):1651–64.
103. Peng F, Malatras A, Hirsbrunner B, Courant M. AntOM: Constructing multi-layer overlays for pervasive environments. In: 2012 IEEE international conference on pervasive computing and communications workshops; 2012. pp. 649–654.
104. Giordanelli R, Mastroianni C, Meo M. Bio-inspired P2P systems: the case of multidimensional overlay. ACM Trans Auton Adapt Syst. 2012;7(4):35:1–28.
105. Forestiero A, Mastroianni C. A Swarm algorithm for a self-structured P2P information system. IEEE Trans Evol Comput. 2009;13(4):681–94.
106. Dhurandher SK, Misra S, Pruthi P, Singhal S, Aggarwal S, Woungang I. Using bee algorithm for peer-to-peer file searching in mobile ad hoc networks. Dependable multimedia communications: systems, services, and applications. J Netw Comput Appl. 2011;34(5):1498–508.
107. Ghanea-Hercock RA, Wang F, Sun Y. Self-organizing and adaptive Peer-to-Peer network. IEEE Trans Syst Man Cybern Part B (Cybern). 2006;36(6):1230–6.
108. Snyder PL, Greenstadt R, Valetto G. Myconet: a fungi-inspired model for superpeer-based peer-to-peer overlay topologies. In: 2009 Third IEEE international conference on self-adaptive and self-organizing systems; 2009. pp. 40–50.
109. Walfish M, Balakrishnan H, Shenker S. Untangling the Web from DNS. In: 1st symposium on networked systems design and implementation (NSDI 2004), San Francisco, California, USA, Proceedings, vol. 4, pp. 225–238. USENIX Association; 2004.
110. Hellerstein JM. Toward network data independence. ACM SIGMOD Record. 2003;32(3):34–40.
111. Zhao BY, Huang L, Stribling J, Rhea SC, Joseph AD, Kubiatowicz JD. Tapestry: a resilient global-scale overlay for service deployment. IEEE J Sel Areas Commun. 2004;22(1):41–53.
112. Harvey NJA, Jones MB, Saroiu S, Theimer M, Wolman A: Skipnet: a scalable overlay network with practical locality properties. In: 4th USENIX symposium on internet technologies and systems, USITS'03, Seattle, Washington, US, vol. 34. USENIX Association; 2003.
113. Pugh W. Skip lists: a probabilistic alternative to balanced trees. In: Dehne FKHA, Sack J, Santoro N (eds) Algorithms and data structures: workshop WADS '89 Ottawa, Canada, Proceedings, lecture notes in computer science, vol. 382; 1989. pp. 437–449.
114. Aspnes J, Shah G. Skip graphs. ACM Trans Algorithms (TALG). 2007;3(4):37.
115. Feldotto M, Scheideler C, Graffi K. HSkip+: a self-stabilizing overlay network for nodes with heterogeneous bandwidths. In: 14th IEEE international conference on peer-to-peer computing; 2014. pp. 1–10.
116. González-Beltrán A, Milligan P, Sage P. Range queries over skip tree graphs. Comput Commun. 2008;31(2):358–74.
117. Jagadish HV, Ooi, BC, Vu QH. BATON: a balanced tree structure for peer-to-peer networks. In: Proceedings of the 31st international conference on very large databases, Trondheim, Norway, pp. 661–672. VLDB Endowment; 2005.
118. Han D, Yu Y. Keyword search in unstructured Peer-to-Peer networks. In: Shen X, Yu H, Buford J, Akon M, editors. Handbook of Peer-to-Peer networking. Boston: Springer; 2010. p. 405–26.
119. Barjini H, Othman M, Ibrahim H, Udzir NI. Shortcoming, problems and analytical comparison for flooding-based search techniques in unstructured P2P networks. Peer-to-Peer Netw Appl. 2012;5(1):1–13.
120. Crespo A, Garcia-Molina H. Routing indices for peer-to-peer systems. In: Proceedings of the 22nd international conference on distributed computing systems (ICDCS'02), Vienna, Austria, pp. 23–32. IEEE Computer Society; 2002.
121. Harren M, Hellerstein JM, Huebsch R, Loo BT, Shenker S, Stoica I. Complex queries in DHT-based peer-to-peer networks. In: Druschel P, Kaashoek MF, Rowstron AIT (eds) Peer-to-peer systems, first international workshop, IPTPS 2002, Cambridge, MA, USA, Revised Papers, lecture notes in computer science, vol. 2429; 2002. pp. 242–250.
122. Bongers E, Pouwelse J. A survey of P2P multidimensional indexing structures. CoRR **abs/1507.05501**. 2015. arXiv:1507.05501.
123. Ferdous MS, Chowdhury F, Moniruzzaman M. A taxonomy of attack methods on peer-to-peer network. In: Proceedings of the 1st Indian Conference on computational intelligence and information security (ICCIIS, 07); 2007. pp. 132–138.

124. Christin N, Weigend AS, Chuang J. content availability, pollution and poisoning in file sharing peer-to-peer networks. In: Proceedings of the 6th ACM conference on electronic commerce, EC '05; 2005. pp. 68–77.
125. Druschel P, Rowstron A. PAST: A large-scale, persistent peer-to-peer storage utility. In: Hot topics in operating systems, 2001. Proceedings of the Eighth Workshop on; 2001. pp. 75–80.
126. Dabek F. A Cooperative File System. Master's thesis, Massachusetts Institute of Technology; 2001.
127. Kubiatowicz J, Bindel D, Chen Y, Czerwinski S, Eaton P, Geels D, Gummadi R, Rhea S, Weatherspoon H, Weimer W, Wells C, Zhao B. OceanStore: an architecture for global-scale persistent storage. SIGARCH Comput Archit News. 2000;28(5):190–201.
128. Caron S, Giroire F, Mazauric D, Monteiro J, Pérennes S. P2P storage systems: study of different placement policies. Peer-to-Peer Netw Appl. 2014;7(4):427–43.
129. Shalini K, Surekha Y. Effective file replication and consistency maintenance mechanism in P2P systems. In: Global journal of computer science and technology; 2011.
130. Gray J, Helland P, O'Neil P, Shasha D. The dangers of replication and a solution. In: Proceedings of the 1996 ACM SIGMOD international conference on management of data, SIGMOD '96; 1996. pp. 173–182.
131. Martins V, Pacciti E, Valduriez P. Survey of data replication in P2P systems. In: Tech. Rep. RR-6083, Institut National de Recherche en Informatique et en Automatique; 2006.
132. Saito Y, Shapiro M. Optimistic Repl. ACM Comput Surv (CSUR). 2005;37(1):42–81.
133. Lv Q, Cao P, Cohen E, Cohen E, Li K, Shenker S. Search and replication in unstructured peer-to-peer networks. In: Proceedings of the 16th international conference on supercomputing, ICS '02; 2002. pp. 84–95
134. On G, Schmitt J, Steinmetz R. The effectiveness of realistic replication strategies on quality of availability for peer-to-peer systems. In: Proceedings third international conference on peer-to-peer computing (P2P2003); 2003. pp. 57–64
135. Ktari S, Zoubert M, Hecker A, Labiod H. Performance evaluation of replication strategies in DHTs Under Churn. In: Proceedings of the 6th international conference on mobile and ubiquitous multimedia, MUM '07; 2007. pp. 90–97.
136. Bhagwan R, Moore D, Savage S, Voelker GM. Replication strategies for highly available peer-to-peer storage. In: Schiper A, Shvartsman AA, Weatherspoon H, Zhao BY, editors. Future directions in distributed computing: research and position Papers. Berlin: Springer; 2003. p. 153–8.
137. Goel S, Buyya R. Data Replication Strategies in Wide-Area Distributed Systems, chap. 9. Hershey; 2019.
138. Leontiadis E, Dimakopoulos V, Pitoura, E. Creating and maintaining replicas in unstructured peer-to-peer systems. In: Euro-Par 2006 parallel processing, lecture notes on computer science, vol. 4128, pp. 1015–1025. Springer; 2006.
139. Kangasharju J, Ross KW, Antipolls SATD. Optimal content replication in P2P communities. In: Tech. rep., TU Darmstadt; 2002.
140. Monteiro J. Modeling and analysis of reliable peer-to-peer storage systems. In: Ph.D. thesis, Université Nice Sophia Antipolis; 2010.
141. Hamming RW. Error detecting and error correcting codes. Bell Syst Tech J. 1950;29(2):147–60.
142. Rabin MO. Efficient dispersal of information for security, load balancing, and fault tolerance. J ACM. 1989;36(2):335–48.
143. Reed IS, Solomon G. Polynomial codes over certain finite fields. J Soc Ind Appl Math. 1960;8(2):300–4.
144. Dimakis AG, Godfrey PB, Wu Y, Wainwright MJ, Ramchandran K. Network coding for distributed storage systems. IEEE Trans Inf Theory. 2010;56(9):4539–51.
145. Duminuco A, Biersack E. Hierarchical codes: how to make erasure codes attractive for peer-to-peer storage systems. In: Peer-to-Peer Computing, 2008. P2P'08. Eighth International Conference on; 2008. pp. 89–98.
146. Sharma R, Datta A. SuperNova: super-peers based architecture for decentralized online social networks. In: 2012 fourth international conference on communication systems and networks (COMSNETS 2012); 2012. pp. 1–10.
147. Di Pasquale A, Nardelli E. Scalable distributed data structures: a survey. In: Proceedings in Informatics, 3rd International Workshop on Distributed Data and Structures (WDAS'00), vol. 9, pp. 87–111. Carleton-Scientific, L'Aquila, Italy; 2000.
148. Gribble SD, Brewer EA, Hellerstein JM, Culler D. Scalable, distributed data structures for internet service construction. In: Proceedings of the 4th conference on symposium on operating system design & Implementation—Vol. 4, OSDI'00. USENIX Association, Berkeley, CA, USA; 2000.
149. Tanner T. Distributed hash tables in p2p systems-a literary survey. In: Tech. Rep. T-110.551, Helsinki University of Technology; 2005.
150. Alaei S, Toossi M, Ghodsi M. SkipTree: A Scalable Range-Queryable Distributed Data Structure for Multidimensional Data. In: Deng X, Du DZ (eds) Algorithms and computation: 16th International Symposium, ISAAC 2005, Sanya, Hainan, China, Proceedings; 2005. pp. 298–307.
151. Beltran AG, Sage P, Miligan P. Skip tree graph: a distributed and Balanced Search Tree for Peer-to-Peer Networks. In: 2007 IEEE international conference on communications; 2007. pp. 1881–1886.
152. Ratnasamy S, Francis P, Handley M, Karp R, Shenker S. A scalable content-addressable network. In: Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, SIGCOMM '01, pp. 161–172. ACM, New York, NY, USA; 2001.
153. Malkhi D, Naor M, Ratajczak D. Viceroy: a scalable and dynamic emulation of the butterfly. In: Proceedings of the twenty-first annual symposium on principles of distributed computing, PODC '02, pp. 183–192. ACM, Monterey, California; 2002.
154. Crainiceanu A, Linga P, Gehrke J, Shanmugasundaram J. P-tree: a P2P index for resource discovery applications. In: Proceedings of the 13th international world wide web conference on alternate track papers & Posters, WWW Alt. '04, pp. 390–391. ACM; 2004.
155. Schmidt C, Parashar M. Flexible information discovery in decentralized distributed systems. In: High performance distributed computing, 2003. Proceedings. 12th IEEE International Symposium on, pp. 226–235. IEEE; 2003.
156. Arge L, Eppstein D, Goodrich MT. Skip-webs: efficient distributed data structures for multi-dimensional data sets. In: Proceedings of the twenty-fourth annual ACM symposium on principles of distributed computing, PODC '05; 2005. pp. 69–76.
157. Awerbuch B, Scheideler C. The hyperring: a low-congestion deterministic data structure for distributed environments. In: Proceedings of the fifteenth annual ACM-SIAM symposium on discrete algorithms, SODA '04, pp. 318–327. SIAM, Philadelphia, PA, USA; 2004.
158. Palomar E, Estevez-Tapiador JM, Hernandez-Castro JC, Ribagorda A. Security in P2P networks: survey and research directions. In: Zhou X, Sokolsky O, Yan L, Jung ES, Shao Z, Mu Y, Lee DC, Kim DY, Jeong YS, Xu CZ (eds) Emerging directions in embedded and ubiquitous computing; 2006. pp. 183–192.
159. Rguibi MA, Moussa N. Hybrid trust model for Worm mitigation in P2P networks. J Inf Secur Appl. 2018;43:21–36.
160. Ciccarelli G, Cigno RL. Collusion in peer-to-peer systems. Comput Netw. 2011;55(15):3517–32.

161. Nakamoto S. Bitcoin: a peer-to-peer electronic cash system bitcoin: a peer-to-peer electronic cash system; 2008.
162. Cigno RL, Ciccarelli G. Collusion in peer-to-peer systems. In: Tech. rep., University of Trento; 2009.
163. Popescu A, Constantinescu D, Erman D, Ilie D. A survey of reliable multicast communication; 2007.
164. Quinn B, Almeroth K. IP multicast applications: challenges and solutions. RFC 3170. 2001. https://doi.org/10.17487/RFC3170. https://www.rfc-editor.org/rfc/rfc3170.txt.
165. Nakayama H, Duolikun D, Enokido T, Takizawa M. A P2P model of publish/subscribe systems. In: 2014 Ninth international conference on broadband and wireless computing, communication and applications; 2014. pp. 383–388.
166. Shen H. Content-based publish/subscribe systems. In: Shen X, Yu H, Buford J, Akon M (eds) Handbook of peer-to-peer networking, pp. 1333–1366. Springer; 2010.
167. Uzunov AV. A survey of security solutions for distributed publish/subscribe systems. Comput Secur. 2016;61:94–129.
168. Rowstron A, Kermarrec AM, Druschel P, Castro M. Scribe: the design of a large-scale event notification infrastructure. In: Crowcroft J, editor. Networked group communication: third international COST264 Workshop, NGC 2001 London, UK, November 7–9, 2001 Proceedings. Berlin, Heidelberg: Springer; 2001. p. 30–43.
169. Zhuang SQ, Zhao BY, Joseph AD, Katz RH, Kubiatowicz J.D. 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, pp. 11–20. ACM; 2001.
170. Mayer TR, Brunie L, Coquil D, Kosch H. On reliability in publish/subscribe systems: a survey. Int J Parallel Emergent Distrib Syst. 2012;27(5):369–86.
171. Bellavista P, Corradi A, Reale A. Quality of service in wide scale publish—subscribe systems. IEEE Commun Surv Tutor. 2014;16(3):1591–616.
172. Segall B, Arnold D. Elvin has left the building: a publish/subscribe notification service with quenching. In: Proceedings of the 1997 Australian UNLX Users Group (AUUG'1997), pp. 243–255. Queensland, Australia; 1997.
173. Petrovic M, Burcea I, Jacobsen HA. S-ToPSS: Semantic Toronto Publish/Subscribe System. In: Proceedings of the 29th international conference on very large data bases—Volume 29, VLDB '03, pp. 1101–1104. VLDB Endowment, Berlin, Germany; 2003.
174. Cugola G, Di Nitto E, Fuggetta A. The JEDI event-based infrastructure and its application to the development of the OPSS WFMS. IEEE Trans Softw Eng. 2001;27(9):827–50.
175. Parzyjegla H, Graff D, Schröter A, Richling J, Mühl G. Design and implementation of the Rebeca Publish/Subscribe Middleware. In: Sachs K, Petrov I, Guerrero P, editors. From active data management to event-based systems and more: papers in Honor of Alejandro Buchmann on the occasion of His 60th birthday. Berlin, Heidelberg: Springer; 2010. p. 124–40.
176. Carzaniga A, Rosenblum DS, Wolf AL. Achieving scalability and expressiveness in an internet-scale event notification service. In: Proceedings of the Nineteenth Annual ACM Symposium on Principles of Distributed Computing, pp. 219–227. ACM; 2000.
177. Carzaniga A, Rosenblum DS, Wolf AL. Design and evaluation of a wide-area event notification service. ACM Trans Comput Syst (TOCS). 2001;19(3):332–83.
178. Gupta A, Sahin OD, Agrawal D, Abbadi AE. Meghdoot: Content-Based Publish/Subscribe over P2P Networks. In: Proceedings of the 5th ACM/IFIP/USENIX international conference on middleware, pp. 254–273; 2004.
179. Ahulló JP, López PG, Gómez-Skarmeta AF. LightPS: lightweight content-based publish/subscribe for Peer-to-Peer systems. In: Xhafa F, Barolli L, editors. Second international conference on complex, intelligent and software intensive systems (CISIS-2008), March 4th-7th, 2008. Barcelona: Technical University of Catalonia; 2008. p. 342–7.
180. Avramidis A, Kotzanikolaou P, Douligeris C, Burmester M. Chord-PKI: a distributed trust infrastructure based on P2P networks. Comput Netw. 2012;56(1):378–98.
181. Castro M, Druschel P, Ganesh A, Rowstron A, Wallach DS. Secure routing for structured peer-to-peer overlay networks. ACM SIGOPS Oper Syst Rev. 2002;36(SI):299–314.
182. Merkle RC. Secure communications over insecure channels. Commun ACM. 1978;21(4):294–9.
183. Borisov, N. Computational puzzles as sybil defenses. In: Sixth IEEE international conference on peer-to-peer computing (P2P'06); 2006. pp. 171–176.
184. Rowaihy H, Enck W, McDaniel P, La Porta T. Limiting Sybil Attacks in Structured P2P Networks. In: IEEE INFOCOM 2007—26th IEEE international conference on computer communications; 2007. pp. 2596–2600.
185. Ion M, Russello G, Crispo B. Supporting publication and subscription confidentiality in Pub/Sub networks. In: Jajodia S, Zhou J, editors. Security and privacy in communication networks. Berlin, Heidelberg: Springer; 2010. p. 272–89.
186. Probst C, Disterhöft A, Graffi K. Chunked-Swarm: divide and conquer for real-time bounds in video streaming. In: Balandin S, Andreev S, Koucheryavy Y, editors. Internet of things, smart spaces, and next generation networks and systems. Cham: Springer; 2015. p. 198–210.
187. Graffi K, Stingl D, Gross C, Nguyen H, Kovacevic A, Steinmetz R. Towards a P2P cloud: reliable resource reservations in unreliable P2P systems. In: 16th IEEE international conference on parallel and distributed systems, ICPADS 2010, Shanghai, China; 2010. pp. 27–34.
188. Graffi K, Pussep K, Kaune S, Kovacevic A, Liebau N, Steinmetz R. Overlay bandwidth management: scheduling and active queue management of overlay flows. In: 32nd Annual IEEE conference on local computer networks (LCN 2007), 15-18 October 2007, Clontarf Castle, Dublin, Ireland, Proceedings; 2007. pp. 334–342.
189. Guidi B, Amft T, De Salve A, Graffi K, Ricci L. Didusonet: a P2P architecture for distributed dunbar-based social networks. Peer-to-Peer Netw Appl. 2016;9(6):1177–94.
190. Benter M, Divband M, Kniesburges S, Koutsopoulos A, Graffi K. Ca-Re-Chord: a churn resistant self-stabilizing chord overlay network. In: 2013 conference on networked systems, NetSys 2013, Stuttgart, Germany; 2013. pp. 27–34.
191. Harrington D, Presuhn R, Wijnen B. An architecture for describing simple network management protocol (SNMP) management frameworks. 2002. https://doi.org/10.17487/RFC3411.
192. von Bochmann G, Hafid A. Some principles for quality of service management. Distrib Sys Eng. 1997;4(1):16–27.
193. Albrecht K, Arnold R, Gähwiler M, Wattenhofer R. Aggregating information in peer-to-peer systems for improved join and leave. In: IEEE P2P '04: proceedings of the international conference on peer-to-peer computing, pp. 227–234. IEEE; 2004.
194. van Renesse R, Bozdog A. Willow: DHT, aggregation, and publish/subscribe in one protocol. In: Peer-to-Peer Systems III, third international workshop, IPTPS 2004, La Jolla, CA, USA, revised selected papers, lecture notes in computer science (LNCS), vol. 3279, pp. 173–183. Springer. 2004.
195. Idreos S, Koubarakis M, Tryfonopoulos C. P2P-DIET: an extensible P2P service that unifies ad-hoc and continuous querying in super-peer networks. In: Weikum G, König CA, Deßloch S (eds) Proceedings of the ACM SIGMOD international conference on management of data, Paris, France, pp. 933–934. ACM; 2004.

196. Shen D, Shao Y, Nie T, Kou Y, Wang Z, Yu G. HilbertChord: a P2P framework for service resources management. In: Wu S, Yang LT, Xu TL (eds) Advances in grid and pervasive computing, third international conference, GPC 2008, Kunming, China, Proceedings; 2008. pp. 331–342.
197. Kempe D, Dobra A, Gehrke J. Gossip-based computation of aggregate information. In: 44th symposium on foundations of computer science (FOCS 2003), Cambridge, MA, USA, Proceedings; 2003. pp. 482–491.
198. Man T. Gossip-based fast overlay topology construction. Gossiping in distributed systems. Comput Netw. 2009;53(13):2321–39.
199. Blasa F, Cafiero S, Fortino G, Di Fatta G. Symmetric push-sum protocol for decentralised aggregation. In: Liotta A, Antonopoulus N, Di FG, Hara T, Vu QH (eds) Proceedings of the third international conference on advances in P2P Systems (AP2PS) 2011), pp. 27–32. International Academy, Research, and Industry Association (IARIA). 2011.
200. SkyEye: A tree-based peer-to-peer monitoring approach. Pervasive Mobile Comput 2017;40:593–610.
201. Graffi K, Kovacevic A, Xiao S, Steinmetz R. SkyEye.KOM: an information management over-overlay for getting the oracle view on structured P2P systems. In: 2008 14th IEEE international conference on parallel and distributed systems; 2008. pp. 279–286.
202. Bhagwan R, Varghese G, Voelker GM. CONE: augmenting DHTs to support distributed resource discovery. In: Technical report CS2003-0755, University of California, San Diego; 2003.
203. Li J, Lim DY. A robust aggregation tree on distributed hash tables. In: Proceedings of the 4th Annual student oxygen workshop held at warren conference center & Inn in Ashland, MA. MIT Oxygen Alliance. 2004.
204. Zhang Z, Shi SM, Zhu J. SOMO: self-organized metadata overlay for resource management in P2P DHT. In: Kaashoek MF, Stoica I, editors. Peer-to-Peer systems II. Berlin, Heidelberg: Springer; 2003. p. 170–82.
205. Disterhöft A, Sandkuhler P, Ippisch A, Graffi K. Mr.Tree: multiple realities in tree-based monitoring overlays for peer-to-peer networks. In: 2018 international conference on computing, networking and communications, ICNC 2018, Maui, HI, USA; 2018. pp. 354–360.
206. Graffi K, Stingl D, Rueckert J, Kovacevic A, Steinmetz R. Monitoring and management of structured peer-to-peer systems. In: Schulzrinne H, Aberer K, Datta A (eds) Proceedings P2P 2009, Ninth International Conference on Peer-to-Peer Computing, USA: Seattle, Washington; 2009. p. 311–20.
207. Graffi KG. Monitoring and management of peer-to-peer systems. PhD, Darmstadt University of Technology; 2010.
208. Graffi K, Gross C, Mukherjee P, Kovacevic A, Steinmetz R. LifeSocial.KOM: a P2P-based Platform for secure online social networks. In: Proceedings of the IEEE international conference on peer-to-peer computing (P2P'10), pp. 1–2; 2010.
209. Graffi K, Gross C, Stingl D, Hartung D, Kovacevic A, Steinmetz R. LifeSocial. KOM: a secure and P2P-based solution for online social networks. In: Proceedings of the 2011 IEEE consumer communications and networking conference (CCNC); 2011. pp. 554–558.
210. Graffi K, Masinde N. LibreSocial: a peer-to-peer framework for online social networks. CoRR. 2020. arXiv:2001.02962.
211. Graffi K, Mukherjee P, Menges B, Hartung D, Kovacevic A, Steinmetz R. Practical security in P2P-based social networks. In: Proceedings of the IEEE 34th conference on local computer networks, 2009. LCN 2009; 2009. pp. 269–272.
212. Narendula R, Papaioannou TG, Aberer K. Privacy-aware and highly-available OSN profiles. In: 2010 19th IEEE international workshops on enabling technologies: infrastructures for collaborative enterprises; 2010. pp. 211–216.
213. Narendula R, Papaioannou TG, Aberer K. My3: a highly-available P2P-based online social network. In: Peer-to-Peer Computing (P2P), 2011 IEEE International Conference on, pp. 166–167. IEEE; 2011.
214. Rhea S, Godfrey B, Karp B, Kubiatowicz J, Ratnasamy S, Shenker S, Stoica I, Yu H. OpenDHT: a public DHT service and its uses. ACM SIGCOMM Comput Commun Rev. 2005;35:73–84.
215. Perfitt, T, Englert B. Megaphone: fault tolerant, scalable, and trustworthy P2P Microblogging. In: 2010 Fifth international conference on internet and web applications and services; 2010. pp. 469–477.
216. Rivest RL, Shamir A, Adleman L. A method for obtaining digital signatures and public-key cryptosystems. Commun ACM. 1978;21(2):120–6.
217. Loupasakis A, Ntarmos N, Triantafillou P. eXO: Decentralized autonomous scalable social networking. In: CIDR 2011, fifth biennial conference on innovative data systems research, Asilomar, CA, USA, Online Proceedings; 2011. pp. 85–95.
218. Asthana H, Cox IJ. PAC'nPost: A framework for a micro-blogging social network in an unstructured P2P Network. In: Proceedings of the 21st international conference on world wide web, WWW '12 Companion, pp. 455–456. Lyon, France; 2012.
219. Jahid S, Nilizadeh S, Mittal P, Borisov N, Kapadia A. DECENT: a decentralized architecture for enforcing privacy in online social networks. In: Pervasive computing and communications workshops (PERCOM Workshops), 2012 IEEE international conference on, pp. 326–332. IEEE; 2012.
220. Raji F, Jazi MD, Miri A. PESCA: a peer-to-peer social network architecture with privacy-enabled social communication and data availability. IET Inf Secur. 2015;9(1):73–80.
221. Malek B, Miri A. Adaptively secure broadcast encryption with short ciphertexts. Int J Netw Secur. 2012;14(2):71–9.
222. Disterhöft A, Graffi K. Protected chords in the web: secure p2p framework for decentralized online social networks. In: 2015 IEEE international conference on peer-to-peer computing (P2P), pp. 1–5. IEEE; 2015.
223. Buchegger S, Schiöberg D, Vu LH, Datta A. PeerSoN: P2P social networking: early experiences and insights. In: Proceedings of the second ACM EuroSys workshop on social network systems, SNS '09; 2015. pp. 46–52.
224. Cutillo LA, Molva R, Strufe T. Safebook: Feasibility of transitive cooperation for privacy on a decentralized social network. In: 2009 IEEE international symposium on a world of wireless, mobile and multimedia networks workshops; 2009. pp. 1–6.
225. Cutillo LA, Molva R, Strufe T. Safebook: a privacy-preserving online social network leveraging on real-life trust. Commun Mag IEEE. 2009;47(12):94–101.
226. Cutillo LA, Molva R, Önen M. Safebook: privacy preserving online social network. In: The IAB workshop on Internet Privacy, jointly organized with the W3C, ISOC, and MIT CSAIL, hosted by MIT on 8-9 December 2010, pp. 1–2. Internet Architecture Board (IAB); 2010.
227. Xu T, Chen Y, Zhao J, Fu X. Cuckoo: towards decentralized, socio-aware online microblogging services and data measurements. In: Proceedings of the 2nd ACM international workshop on hot topics in planet-scale measurement, no. 4 in HotPlanet '10, pp. 4:1–4:6. San Francisco, California; 2010.
228. Xu T, Chen Y, Fu X, Hui P. Twittering by cuckoo: decentralized and socio-aware online microblogging services. In: Proceedings of the ACM SIGCOMM 2010 conference on applications, technologies, architectures, and protocols for computer communications, New Delhi, India; 2010. pp. 473–474.
229. Juste PS, Wolinsky D, Boykin PO, Figueiredo RJ. Litter: a lightweight peer-to-peer microblogging service. In: Privacy, security, risk and trust (PASSAT) and 2011 IEEE third inernational

conference on social computing (SocialCom), 2011 IEEE third international conference on IEEE, pp. 900–903. 2011.
230. Juste PS, Eom H, Lee K, Figueiredo RJ. Enabling decentralized microblogging through P2pvpns. In: 2013 IEEE 10th consumer communications and networking conference (CCNC); 2013. pp. 323–328.
231. Iglesias DL, Marques JM, Cabrera G, Rifa-Pous H, Montane A. HorNet: microblogging for a contributory social network. IEEE Internet Comput. 2012;16(3):37–45.
232. Iglesias DL. A middleware for service deployment in contributory computing systems. In: PhD., Universitat Oberta de Catalunya, Barcelona, Spain; 2011.
233. Aiello LM, Milanesio M, Ruffo G, Schifanella R. tempering kademlia with a robust identity based system. In: 2008 eighth international conference on peer-to-peer computing; 2008. pp. 30–39.
234. Dürr M, Maier M, Dorfmeister F. Vegas—a secure and privacy-preserving peer-to-peer online social network. In: 2012 international conference on privacy, security, risk and trust and 2012 international conference on social computing; 2012. pp. 868–874.
235. Tran HM, Nguyen VS, Ha SVU. Decentralized online social network using Peer-to-Peer technology. REV J Electron Commun. 2015;5:1–2.
236. Wang J, Liu F, Li X, Liu H, Zhao X. HPOSN: a novel online social network model based on hybrid P2P. In: 2015 international conference on cloud computing and big data (CCBD); 2015. pp. 342–349.
237. Goldschlag D, Reed M, Syverson P. Onion routing. Commun ACM. 1999;42(2):39–41.
238. Nilizadeh S, Jahid S, Mittal P, Borisov N, Kapadia A. Cachet: a decentralized architecture for privacy preserving social networking with caching. In: Proceedings of the 8th international conference on emerging networking experiments and technologies, CoNEXT '12; 2012. pp. 337–348.
239. Jahid S, Mittal P, Borisov N. EASiER: encryption-based access control in social networks with efficient revocation. In: Proceedings of the 6th ACM symposium on information, computer and communications security, ASIACCS '11; 2011. pp. 411–415.
240. Bethencourt J, Sahai A, Waters B. Ciphertext-policy attribute-based encryption. In: Security and privacy, 2007. SP'07. IEEE Symposium on IEEE, pp. 321–334. 2007.
241. Freitas M. twister—a P2P microblogging platform. CoRR **abs/1312.7152**. 2013.
242. Freitas M. Twister: the development of a peer-to-peer microblogging platform. Int J Parall Emergent Distrib Syst. 2016;31(1):20–33.
243. Everett M, Borgatti SP. Ego network betweenness. Soc Netw. 2005;27(1):31–8.
244. Fang Y, Wen Z, Shen Q, Yang Y, Wu Z. SEDOSN: a secure decentralized online social networking framework. In: Zhang X, Wu Z, Sha X, editors. Embedded system technology. Singapore: Springer; 2015. p. 68–74.
245. Rouselakis Y, Waters B. Practical constructions and new proof methods for large universe attribute-based encryption. In: Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, CCS '13; 2013. pp. 463–474.
246. Franchi E, Poggi A, Tomaiuolo M. Blogracy: a peer-to-peer social network. Int J Distrib Syst Technol (IJDST). 2016;7(2):37–56.
247. zzz (Pseudonym), Schimmer L. Peer profiling and selection in the I2P anonymous network. In: Proceedings of PET-CON 2009.1; 2013. pp. 59–70.
248. Dingledine R, Mathewson N, Syverson P. Tor: the second-generation onion router. In: Tech. rep., Naval Research Lab Washington DC; 2004.
249. Vu LH, Aberer K, Buchegger S, Datta A. Enabling secure secret sharing in distributed online social networks. In: 2009 annual computer security applications conference, 2009:419–428.
250. Camarillo G. Peer-to-Peer (P2P) Architecture: definition, taxonomies, examples, and applicability. RFC 2009:5694 2009. https://doi.org/10.17487/RFC5694. https://rfc-editor.org/rfc/rfc5694.txt