Unstructured Peer-to-Peer Network Architectures

Xing Jin and S.-H. Gary Chan

Abstract With the rapid growth of the Internet, peer-to-peer (P2P) networks have been widely studied and deployed. According to CacheLogic Research, P2P traffic has dominated the Internet traffic in 2006, by accounting for over 72% Internet traffic. In this chapter, we focus on unstructured P2P networks, one key type of P2P networks. We first present several unstructured P2P networks for the file sharing application, and then investigate some advanced issues in the network design. We also study two other important applications, i.e., media streaming and voice over Internet Protocol (VoIP). Finally, we discuss unstructured P2P networks over wireless networks.

1 Introduction

In the recent years, P2P networks have seen enormous successes and rich developments over the Internet. When Napster first emerged in 1999 as a P2P file sharing system, the Internet traffic was dominated by web and ftp (accounting for 65 and 10% total traffic, respectively, according to CacheLogic Research). But in 2006, 50–65% of downstream traffic and 75–90% of upstream traffic was already P2P traffic (according to CacheLogic Research). In total, P2P traffic has accounted for 72% Internet traffic in the year, while that of web and ftp has decreased to 24 and 2%, respectively.

Xing Jin
Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong, e-mail: csvenus@cse.ust.hk

S.-H. Gary Chan
Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong, e-mail: gchan@cse.ust.hk

X. Shen et al. (eds.), *Handbook of Peer-to-Peer Networking,*
DOI 10.1007/978-0-387-09751-0_5,

In P2P systems, cooperative peers self-organize themselves into overlay networks and store or relay data for each other. The major challenge is how to achieve efficient resource search in a large scale distributed-storage network. Currently popular P2P search systems can be classified as *unstructured* and *structured*, depending on the overlay structures. Unstructured systems do not impose any structure on the overlay networks [1–3]. These systems are usually resilient to peer dynamics, and support complex queries with meta information. But they are not efficient for locating unpopular files. Structured systems impose particular structures on the overlay networks, which are commonly referred to as distributed hash tables (DHTs) [38, 42]. In a structured system, any file can be located in a small number of overlay hops, which significantly reduces the search cost as compared to unstructured systems. However, DHT only supports single-keyword exact-match lookups.

In this chapter, we focus on unstructured P2P networks. We first study the file sharing application, which is the original and one of the most important applications for P2P networks. Depending on system decentralization level, we classify unstructured P2P networks into centralized, distributed, hybrid and some other approaches. We select representative examples from each category and analyze their search mechanisms. Then we discuss some advanced issues in system design, i.e., content replication and system security.

We also study other important applications for unstructured P2P networks, including media streaming and VoIP. One of the pioneering P2P streaming softwares, CoolStreaming, has reported to attract more than $25,000$ concurrent peers for one streaming channel [50]. Another streaming software PPLive reported more than $400,000$ concurrent peers for its over 300 channels [25]. As for P2P VoIP, the leading software Skype has shown to attract over 8 million concurrent users in 2008 (from its client software interface), and has reported to accumulate 276 million user accounts at the end of 2007. Clearly, these applications have evolved into influential network applications over the Internet. We hence select CoolStreaming and Skype as two typical examples. We study the challenges in these applications and their corresponding approaches.

Finally, we study the implementation issue of unstructured P2P networks over wireless networks. Wireless networks are going through quick development nowadays. As they share many similar features with P2P networks (such as decentralization and dynamic topology), there has been an increasing interest in integrating them together. We compare P2P networks and wireless networks, highlight the challenges in integrating them, and discuss several state-of-the-art approaches.

The rest of the chapter is organized as follows. In Section 2, we present the general design considerations in unstructured P2P networks. In Section 3, we discuss representative examples of unstructured P2P networks for file sharing. In Section 4, we explore some advanced issues in P2P file sharing. In Section 5, we discuss the media streaming and VoIP applications for unstructured P2P networks. In Section 6, we investigate how to implement unstructured P2P networks over wireless networks. Finally, we conclude in Section 7.

2 Design Considerations

When designing an unstructured P2P network, several issues need to be carefully considered:

- *Search efficiency and replication cost*: In unstructured P2P networks, data are distributedly stored at peers and each peer only holds limited information about the system. Hence, it is important to design efficient search mechanisms. Usually, with more data replications in the network, a data file can be located more quickly. It is hence a tradeoff between storage cost and search time.
- *Scalability*: A P2P system may consist of hundreds of thousands of peers. This often requires a fully distributed system, where peers form a self-organized network and each peer communicates with only a few other peers.
- *Resilience to peer dynamics*: In P2P systems, a peer may arbitrarily join, leave or fail. A good P2P system should be resilient to such peer dynamics.
- *Load balancing*: Peers often have heterogeneous resource (e.g., bandwidth, computational capability, storage space). A good system should be able to achieve balanced loads among peers. This can avoid overloading hot peers, and hence improve system scalability.
- *Security*: In the open environment of the Internet, some participating peers may be selfish and unwilling to upload data to others, or some may launch attacks to disrupt the service. A practical P2P system should be well protected to targeted attacks or free-riders (i.e., peers only downloading data without uploading). Other considerations include peers' privacy and confidentiality.

3 Unstructured P2P Networks for File Sharing

In this chapter, we describe several unstructured P2P networks for the file sharing application. We classified the approaches into four categories: centralized, distributed, hybrid and others. We select one representative example from each category and discuss its system design.

3.1 A Centralized Approach: Napster

In 1999, Napster appeared as the first P2P file sharing system [4]. A Napster system consists of a central directory server and a set of registered users (or peers). The server maintains information of all files in the system, including an index with metadata (such as file name and size) of all files in the system, a list of all registered peers, and a list showing the files that each peer holds and shares.

When a new peer joins the system, it contacts the server and reports a list of files it maintains and shares. When a peer wants to search for a file, it sends a request

to the server. The server will return a list of peers that hold the matching file. The searching peer then contacts the returned peers to download the file.

Figure 1(a) shows the search process in Napster. When peer *A* wants to search for some file, it contacts the central server. The server returns some peers that hold the file, say, peer *B*. Peer *A* then starts to download the file from peer *B*.

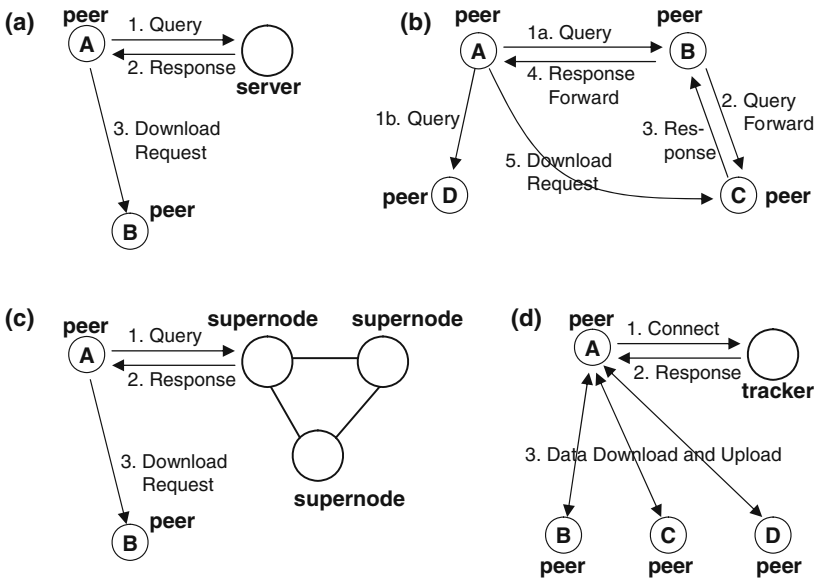

Fig. 1 Search process in unstructured P2P networks. (**a**) Napster. (**b**) Gnutella. (**c**) Kazaa. (**d**) BitTorrent.

The advantage of Napster is its ease of implementation and simplicity of deployment. The system administrator only needs to deploy and maintain a central server. Furthermore, the system is highly adaptive to peer joining and leaving. The major disadvantage is that such a centralized system is not scalable. The server needs to have much resource (such as computational capability and bandwidth) to support a large number of peers. In addition, the server forms a single point of failure. If the server is down, the whole system is broken. It is hence vulnerable to targeted attacks against the server (e.g., DDos).

3.2 A Distributed Approach: Gnutella

Gnutella is a fully distributed P2P system for file sharing [1]. It first appeared in 2000 and got quick development in a few years. According to Slyck news report, as of June 2005, Gnutella's population was 1.81 million computers. Gnutella's source

code is publicly available in the Internet. This enables the development of different client softwares by different groups (e.g., LimeWire developed by the LimeWire group, or Gnucleus developed by the Morpheus group).

In the basic Gnutella protocol, when a new peer joins the system, it first connects to some public peers. For example, a list of public peers are available at *http://gnutellahosts.com*. A new peer then sends a PING message to any peer it is connected to. The message announces the existence of the new peer. Upon receiving a PING message, a Gnutella peer returns a PONG message and propagates the PING message to its neighbors. A PONG message contains IP address and port of the responding peer, and information of files being shared by the responding peer. In a dynamic network with frequent peer joining and leaving, a peer periodically sends PING messages its neighbors.

Search in Gnutella is based on flooding, which is broadcasting in the overlay. A search query is propagated to all neighbors from the original requesting peer. The query is replicated and forwarded by each intermediate peer to all its neighbors. Each intermediate peer also examines its local contents and responds to the query source on a match. The query responses are routed back along the opposite path towards the original requesting peer. To reduce the amount of query messages in the network, each query message contains a time-to-live (TTL) field. The TTL value will be decremented by one at each peer. When it reaches zero, the message is dropped.

Figure 1(b) shows the search process in Gnutella. Suppose peer *A* wants to search for some file. It floods its search query to its neighbors, i.e., peers *B* and *D* in the figure. When peer *B* receives the query, it checks whether itself holds the matching file. If not, it forwards the query to its neighbors. As in the example, peer *B* forwards the query to its neighbor *C*. Suppose *C* holds the file that *A* wants. *C* returns a response to the peer that sends it the query, which is *B* in the figure. *B* then continues forwarding the response to the query sender *A*. Finally, *A* contacts *C* to download the file.

Different from Napster, Gnutella is a dynamic, self-organized network. Each peer independently connects to and communicates with a few other peers in the system. The system is hence able to contain an unlimited number of peers, if no constraint on search efficiency. Meanwhile, the system is highly robust to peer dynamics. If a peer leaves the system, its neighbors can connect to other peers through the exchange of PING and PONG messages.

A limitation of Gnutella is its relatively low search efficiency. In flooding search, the number of query messages exponentially increases with the number of overlay hops. Then a query may generate many messages, especially for unpopular files, where a query has to go through many overlay hops and many peers before reaching a matching peer. Given the huge number of peers in the system, the traffic load for queries could be significantly high. The use of TTL can reduce the number of query messages. However, choosing an appropriate TTL is not easy. If the TTL is too high, peers unnecessarily burden the network. If the TTL is too low, a peer might not find the file even though a copy exists somewhere.

In addition, there are many duplicate messages in flooding search, particularly in heavily connected networks. This is because multiple copies of a query may be sent to a peer by its multiple neighbors. These duplicate queries consume extra processing power at peers and network bandwidth. Hence, it is necessary to develop some duplication detection mechanisms. However, even with duplicate suppression, the number of duplicate messages in flooding can be excessive, and the problem gets worse as the TTL increases.

Therefore, many new search methods have been proposed to improve or replace flooding search. Expanding ring is a method that addresses the TTL selection problem in flooding [34]. That is, a peer starts flooding with a small TTL. If the search does not succeed, the peer increases the TTL and starts another flooding. The process repeats until the search succeeds. This method is efficient for locating hot files. Hot files are widely replicated in the network, and a small TTL value is enough to locate them. But this method leads to higher search time due to repeated flooding.

Expanding ring does not address the message duplication problem. Another method uses multiple parallel random walks to address this issue [34]. In standard random walk, when receiving a search query, a peer randomly chooses one neighbor and propagates the query to it. To reduce the search time, the method uses multiple random walks. This method, together with proactive file replication (discussed in Section 4.1), can significantly improve the search performance in terms of search time, per-peer query load, and message traffic.

3.3 A Hybrid Approach: FastTrack/Kazaa

Given the limitations of purely centralized networks and purely distributed networks, there is a third approach which combines these two types of networks. FastTrack is a typical example as a partially centralized P2P protocol. In FastTrack, peers with the fastest Internet connections and the most powerful computers are automatically designated as *supernodes*. A supernode maintains information about some resource as well as connections with other supernodes. When a peer performs a search, it first searches for the closest supernode, which returns immediate results if any and refers the search to other supernodes if needed. Two practical softwares based on FastTrack are Kazaa [2] and Grokster [5]. But the latter closed its service in 2005 due to the copyright issue.

Figure 1(c) shows the search process in Kazaa. When peer *A* wants to search for some file, it sends the search query to the closest supernode. The supernode either returns some matching peers, or forwards the query to other supernodes. Finally, *A* will obtain some matching peers from the supernode (say, peer *B* in the figure) and download the file from these peers.

Therefore, an ordinary peer (e.g., peer *A* in the figure) communicates with a supernode as if communicating with the server in Napster. Then, Gnutella like search is performed in a highly pruned overlay network of supernodes.

As compared with purely distributed networks like Gnutella, Kazaa achieves much lower search time. Search among supernodes is much faster than search among all peers, because the number of supernodes is much smaller than the total number of peers. As supernodes have high bandwidth and large storage space, they can efficiently process a large amount of queries from ordinary peers. The system hence makes good use of peer heterogeneity. In addition, unlike Napster, it does not form a single point of failure. If some supernodes go down, the peers connecting to them can connect to other supernodes.

In view of the success of Kazaa, Gnutella also considers adopting a hybrid structure. Chawathe et al. propose the Gia system to improve Gnutella [12]. It uses a dynamic topology adaptation protocol to put most peers within a short distance of a few supernodes. Supernodes will receive a large proportion of the queries. Together with a few other improvements (e.g., active flow control and biased random walk), Gia increases the system capacity by three to five orders of magnitude. Later, Gnutella version 0.6 formally incorporates the idea of "ultrapeers". In detail, peers entering into the network are kept at the edge of the network as leaves, not responsible for any routing. Peers which are capable of routing messages are promoted to ultrapeers, which will accept leaf connections and route searches and network maintenance messages. Normally, a leaf peer is connected to 3 ultrapeers, and each ultrapeer is connected to more than 32 other ultrapeers. Within this network, the maximum number of hops a query can travel is reduced to around 4. The search efficiency and the whole system scalability are then greatly improved.

3.4 Other Approach: BitTorrent

BitTorrent is a P2P system that does not belong to any of the above categories [3]. As an important P2P file sharing application, it is estimated by CacheLogic to represent 35% of all Internet traffic in 2004.

BitTorrent uses a central location to coordinate data upload and download among peers. To share a file f, a peer first creates a small torrent file, which contains metadata about f, e.g., its length, name and hashing information. Usually, BitTorrent cuts a file into pieces of fixed size, typically between 64 KB and 4 MB each. Each piece has a checksum from the SHA1 hashing algorithm, which is also recorded in the torrent file. Most importantly, the torrent file contains the URL of a tracker, which keeps track of all the peers who have file f (either partially or completely) and the lookup peers.

A peer that wants to download the file first obtains the corresponding torrent file, and then connects to the specified tracker. The tracker responds with a random list of peers which are downloading the same file. The requesting peer then connects to these peers for downloading.

Figure 1(d) shows the search process in BitTorrent. When peer A wants to search for some file, it first needs to obtain the corresponding torrent for the file. From the torrent, A knows the address of the tracker and connects to the tracker. The tracker

then returns a list of peers who are downloading or sharing the file. *A* then exchanges data with these peers.

In BitTorrent systems, torrent files are often published on large websites, which also serve as trackers. Clearly, the centralization of trackers brings some barriers in the system. If a tracker is down, peers will not be able to start their sharing (by uploading their torrents to the tracker), and new incoming peers cannot start their downloading. In order to remove the need of central trackers, the latest BitTorrent clients implement a decentralized tracking mechanism (e.g., μTorrent, BitComet, KTorrent). In the mechanism, every peer acts as a mini-tracker. Peers first join a DHT network, which is inherently implemented in the BitTorrent client. A torrent is then stored at a certain peer according to the DHT storage method. All peers in the DHT network can search for the torrent through DHT search. Therefore, this mechanism eliminates central trackers from the system.

3.5 Comparison and Discussion

The main characteristic of unstructured P2P networks is that the storage of files is completely unrelated to overlay topology. As a result, file search mechanism in such networks essentially amounts to random search. As compared to structured P2P networks, unstructured P2P networks have the following advantages:

- *Resilient to peer dynamics*: Because there is no specified requirement on the overlay structure, unstructured P2P networks are often formed in a random way. In case of peer joining and leaving, an unstructured P2P network can easily reconstruct the overlay. As a comparison, overlay reconstruction in DHT networks is much more complex and expensive, especially in a highly dynamic network with frequent peer joining and leaving.
- *Supporting complex search*: In unstructured P2P networks, peers eventually check their local resource to answer a query. Hence, unstructured P2P networks inherently support complex search based on file meta-data. On the contrary, in DHT networks, each file has a single keyword. File storage and search are all based on this keyword. DHT hence only supports single-keyword search.

Unstructured P2P networks have their own limitations. Its major problem is the low search efficiency, especially for unpopular files. Unpopular files have few copies in the system. Search for such a file may lead to large-scale flooding. Different from it, DHT networks guarantee a certain number of overlay hops for any search, which is very helpful for unpopular file search. In view of the advantages of structured and unstructured networks, some researchers have taken effort to integrate them together [11]. Clearly, there are many challenges in the integration due to their fundamental difference on network structures.

We now give a quick comparison of the above unstructured P2P networks. The centralized approach requires a central server for file management and search, and

the supernode-based approach relies on elected or pre-deployed supernodes. Different from them, the fully distributed approach relies on peers for file storage and search, which does not require additional facilities. BitTorrent, a second generation P2P system, requires central trackers to initiate the file sharing process.

According to the system architecture, the approaches have different search methods. In the centralized approach, a query is directly sent to the server. In the supernode-based approach, a query is first sent to a supernode, which forwards the query to other supernodes if needed. In the fully distributed approach, blind or informed flooding is used to answer a query, which may consume much network bandwidth. In BitTorrent, a peer can directly obtain a list of peers sharing the file from the tracker. It hence eliminates the search process for matching peers as in other systems. This is a fundamental difference between BitTorrent and other unstructured P2P systems.

In practice, the supernode-based approach and BitTorrent are the most successful applications. Both the centralized and fully distributed approaches have serious limitations. The centralized approach has poor scalability. It cannot accommodate a large number of peers. But the huge number of peers, and hence huge resource in total, is exactly an attractive aspect of P2P systems. The fully distributed approach does not need any central component, however, it is not highly scalable due to its inefficiency and high bandwidth consumption in search.

As a comparison, the supernode-based approach has shown high scalability and high search efficiency given enough supernodes. It makes practical use of peer heterogeneity. BitTorrent requires central trackers. But the responsibility of a tracker is much lighter than the server in a centralized approach. A tracker only needs to track the peers sharing and downloading the specific file. And torrent files could be put at different trackers. Hence, the scalability of BitTorrent is not an issue in practice. In addition, the newly proposed decentralized tracking mechanism completely eliminates central trackers from the system.

4 Advanced Issues in File Sharing

In this chapter, we discuss two advanced issues in unstructured P2P file sharing systems, i.e,, content replication and system security.

4.1 Content Replication

Search in unstructured P2P networks is essentially random search. Content replication is hence a fundamental issue for search performance. Intuitively, with more replications in the network, it is easier (or faster) to locate a file. On the other side of the coin, more replications take up more storage space. We hence need to explore

efficient replication mechanisms to achieve good tradeoff between search efficiency and storage cost.

In the original Gnutella, peers requesting a file make copies of the file. Other systems like FastTrack allow for more proactive replications of files, where a file may be replicated at a peer even though the peer has not requested the file.

Cohen et al. have proposed a network model for proactive replication in unstructured P2P networks [13]. Suppose there are in total m files and n peers in an unstructured P2P network. Each file i ($1 \le i \le m$) is replicated at r_i ($1 \le r_i \le n$) random distinct peers. Clearly, if there is not any limitation on storage space, a straightforward strategy would be to replicate everything everywhere, and search becomes trivial. Hence, we assume that the total amount of storage space over all peers is fixed. If we further assume that each file has a unit size, the total amount of storage space can be computed as $R = \sum_{i=1}^{m} r_i$.

Suppose that file i is requested with the query rate q_i. Here q_i is normalized so that $\sum_{i=1}^{m} q_i = 1$. We consider search of randomly probing peers until the specific file is found. Then, the probability that file i is found on the k'th probe is given by

$$Pr_i(k) = \frac{r_i}{n}\left(1 - \frac{r_i}{n}\right)^{k-1}.$$

Define search size of a query as the number of probes to locate the matching file for the query. For a certain file i, its average search size A_i is simply n/r_i. Hence, the average search size over all files is

$$A = \sum_{i=1}^{m} q_i A_i = n \sum_{i=1}^{m} \frac{q_i}{r_i}. \tag{1}$$

Given the above network model, we can analyze the performance of several different replication mechanisms.

- *Uniform replication*: This replication mechanism equally replicates all files regardless of their popularities. In other words, $r_i = R/m,\ \forall i \in [1,m]$. Hence, the average search size $A_{uniform}$ is given by

$$A_{uniform} = n \sum_{i=1}^{m} \left(q_i \frac{m}{R}\right) = \frac{nm}{R}.$$

  Clearly, $A_{uniform}$ is independent of the query distribution.
- *Proportional replication*: This replication mechanism replicates more copies for more popular files. In other words, $r_i = Rq_i,\ \forall i \in [1,m]$. Hence, the average search size $A_{proportional}$ is given by

$$A_{proportional} = n \sum_{i=1}^{m} \frac{q_i}{Rq_i} = \frac{nm}{R}.$$

  Therefore, the proportional and uniform replication mechanisms yield the same average search size, and that average search size is independent of the query

distribution. On the other side, the distribution of average search size for a certain file is different for these two mechanisms. In uniform replication, all files have the same average search size (which is nm/R). In proportional replication, $A_i = n/(Rq_i)$. Hence, a popular file with a high query rate will have a small average search size, since it has many replicated copies in the network.

- *Square-root replication*: Given the formula of A as in Equation (1), Cohen et al. prove in [13] that A is minimized when

$$r_i = \frac{R\sqrt{q_i}}{\sum_{j=1}^{m}\sqrt{q_j}}.$$

The resulting average search size is

$$A_{optimal} = \frac{n}{R}\left(\sum_{i=1}^{m}\sqrt{q_i}\right)^2.$$

This result is later confirmed by Lv et al. through simulations [34].

Given the above replication mechanisms, an immediate question is how to achieve them via a distributed protocol in a decentralized unstructured P2P network. This is easy for uniform and proportional replications. For uniform replication, the system creates a fixed number of copies when the file first enters the system. For proportional replication, the system creates a fixed number of copies every time the file is queried. But the case for square-root replication is much more difficult. The major challenge is that no individual peer sees enough queries to estimate the query rate for a certain file.

Cohen et al. study several ways to achieve square-root replication [13]. Generally, when a query succeeds, the requesting peer creates some number of copies (denoted the number as C) at randomly selected peers. There is also some deletion mechanism to guarantee that in the steady state the creation rate equals the deletion rate. There are various ways to determine C. The first one is called path replication, which sets C to the search size (i.e., the number of peers probed). Another method further improves path replication by requiring a peer to record the value C with each copy. This is called replication with sibling-number memory. A third method is called replication with probe memory. Each peer records the number and the combined search size of probes it sees for each file. It then determines C by collecting this information from a certain number of peers. Clearly, this method needs extra inter-host communication. For more details of these methods and their comparison, please refer to [13].

4.2 Security and Reputation System

Most P2P systems work on the assumption of truthful cooperation among peers. However, in the open environment of the Internet, some participating peers may not

cooperate as desired. They may be selfish and unwilling to upload data to others, or they may have abnormal actions such as frequent rebooting which adversely affect their neighbors. More seriously, some peers may launch attacks to disrupt the service or distribute viruses in the overlay network. We call these uncooperative, abnormal or attacking behavior *malicious actions* and the associated peers *malicious peers*.

To detect malicious peers or reward well-behaved ones, a reputation system is often used. In a typical reputation system, each peer is assigned a reputation value according to its history performance. Differentiated services are then provided to peers according to their reputation. We now study two key issues in P2P reputation systems, namely, reputation computing and storage.

4.2.1 Reputation Computing

There are mainly three reputation computing techniques in current P2P networks. We elaborate them as follows.

- *Social Networks* In this approach, all feedbacks available in the network are aggregated to compute peer reputation. It can be further classified into two categories: *separated reputation model* and *correlated reputation model*. In a separated reputation model, only the direct transaction partners (e.g., resource provider/downloader or streaming neighbor) of a peer can express their opinion on the reputation of the peer [14, 16, 21, 27, 28, 35, 41]. A practical example is eBay reputation system (although eBay is not a P2P network) [6]. After each transaction at eBay, the buyer and the seller rate each other with a positive, negative and neutral feedback. The reputation is calculated at a central server by assigning 1 point for each positive feedback, 0 point for each neutral feedback and -1 point for each negative feedback. The reputation of a participant is computed as the sum of its points over a certain period. Considering that peers may lie in their feedbacks, Mekouar et al. propose to monitor suspicious feedbacks [35]. The more suspicious feedbacks a peer generates, the smaller weight in reputation computing its feedback has. Xiong et al. develop a general reputation model, which considers, for example, feedbacks from other peers, credibility factor for the feedback sources, and transaction context factor for discriminating the importance of transactions [46]. In fact, almost all the separated reputation models can be expressed by this generalized model.
  In a correlated reputation model, the reputation of a peer is computed based on the opinion of its direct transaction partners as well as some third-party peers [29, 40]. In this model, a peer A who wishes to know the reputation of another peer B, can ask some peers (e.g., its neighbors) to provide their opinion on B (although some of the peers may not have conducted any transaction with B). A then combines the opinion from the peers to calculate B's reputation. Clearly, this model is more like our real social networks, where third-party peers besides transaction partners can express their opinion on a peer. But it takes more cost to collect and aggregate third-party opinion.

- *Probabilistic Estimation* This approach uses sampling of the globally available feedbacks to compute peer reputation. It often has some assumptions on peer behavior. For instance, it may assume that a peer is trustworthy with a certain, but unknown probability. And when sharing its own experience with others, a peer may lie with some, again unknown, probability [17]. It then uses well known probabilistic estimation techniques to estimate all unknown parameters.
  Many estimation methods may be used. Despotovic et al. use maximum likelihood estimation [17]. However, it assumes that peers do not collude, which may not be practical in real networks. Mui et al. use Bayesian estimation to assess the future performance of peers based on their history performance, but it uses only direct interaction among peers and does not use third-party opinion [36]. Buchegger et al. take into account third-party opinion, but the approach is empirical rather than theoretically solid [10].
  By using a small portion of the globally available feedbacks, the probabilistic model spends a lower cost in feedback collection than the social network approach. On the other hand, the social network approach can use a complicated reputation model, and is robust to a wide range of malicious actions. But the probabilistic model can be applied to only simple reputation models (due to the difficulties in probabilistic estimation) and is effective to only a few malicious actions. The performance of the two models has been compared in [18]. It has been shown that the probabilistic model performs better for small malicious population, while the social network approach is better when most peers are malicious.
- *Game-Theoretic Model* Different from the above two approaches, the game-theoretic model assumes that peers have rational behavior and uses game theory to build a reputation system. Rational behavior implies that there is an underlying economic model in which utilities are associated with various choices of the peers and that peers act so as to maximize their utilities. Li et al. present a game-theoretic framework for analyzing reputation [30], and Fudenberg et al. offers certain characterizations of the equilibria payoffs in the presence of reputation effects [22].

4.3 Reputation Storage and Retrieval

A basic principle in reputation storage is that the reputation of a peer cannot be locally stored at the peer. Because this has no protection against dishonest peers. A dishonest peer may misreport its reputation value in order to gain rewards or avoid punishments. We summarize several techniques for reputation storage and retrieval in unstructured P2P networks as follows.

- *Centralized* This method uses a powerful server to keep the reputation of all peers. For example, eBay uses a central server to collect and keep all users' reputation [6]. Feedbacks from users are sent to and stored at the server. A query of

a user's reputation is also sent to and replied by the server. Similar approaches have been used in [21, 27, 28].

This approach is easy to implement and deploy. Security of a central server is much easier to achieve than that of distributed components in a distributed approach. Furthermore, centralization makes reputation management independent of peer joining and leaving, which greatly simplifies reputation retrieval. However, as discussed in Section 3.1, a centralized approach is not scalable to large P2P networks. And the server forms a single point of failure, making the system vulnerable.

- *Supernode-Based* Mekouar et al. propose a malicious detector algorithm to detect malicious peers in Kazaa-like systems [35]. They assume that supernodes are all trustworthy and maintain reputation information for ordinary peers. Each peer is attached to a unique supernode. All the evaluation results about a peer are maintained at its attached supernode. Supernodes can then enforce differentiated services according to peers' reputation. Note that supernodes in Kazaa are elected according to peers' computational power and edge bandwidth. A supernode may not always be trustworthy. A possible improvement is to deploy some proxies (e.g., a content distribution network) to replace unauthenticated supernodes.

  The security and trustworthiness of pre-deployed proxies are much better than self-elected supernodes.

  The supernode-based approach is an extension of the centralized approach. In the approach, a set of supernodes instead of a single server serve peers. However, to serve a large P2P network, a large number of supernodes are needed, which leads to high implementation and maintenance costs. In addition, the search and load balancing mechanisms among supernodes need to carefully designed.
- *Unstructured Overlay* XREP uses a polling algorithm to help peers choose reliable resource in Gnutella-like networks [14, 16]. It consists of four operations: resource searching, vote polling, vote evaluation and resource downloading (as shown in Figure 2). The first operation is similar to searching in Gnutella. A peer broadcasts to all its neighbors a *Query* message. If a peer receiving a *Query* message has the matching file, it responds with a *QueryHit* message (as shown in Figure 2(a)). In the next operation, upon receiving *QueryHit* messages, the original searching peer selects the best matching resource among all possible choices. It then polls other peers using an encrypted *Poll* message to enquire their opinion on the resource or the resource provider. To achieve that, each XREP peer maintains information for its own experience on resource and other peers. Upon receiving a *Poll* message, each peer checks its experience data. If there is any information about the resource or the provider indicated by the *Poll* message, the peer sends its vote to the polling peer with an encrypted *PollReply* message (as shown in Figure 2(b)).

  In the third operation, the polling peer collects a set of votes and evaluates the votes. It first decrypts the votes and discards corrupt ones. Then it analyzes voters' IPs and detects cliques of dummy or controlled votes. After that, it randomly selects a set of votes and directly contacts them with a *TrustVote* message. Each contacted voter is required to send a *VoteReply* message for vote confirmation.

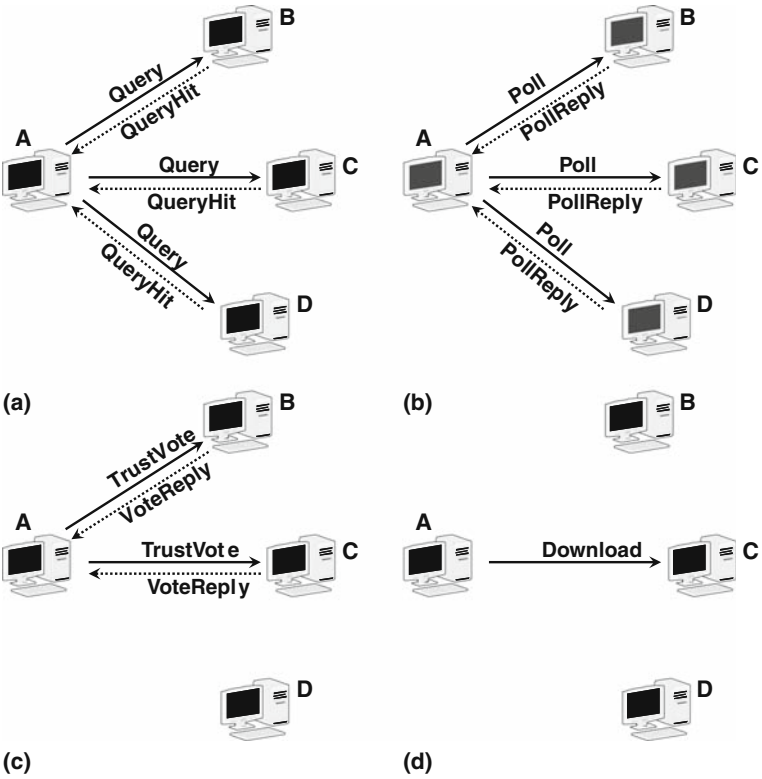

Fig. 2 Operations in XREP. (**a**) Resource searching. (**b**) Vote polling. (**c**) Vote evaluation. (**d**) Resource downloading.

This forces potential attackers to pay the cost of using real IPs as false witness (e.g., shilling attack). After this checking process, the polling peer can obtain the reputation of the resource or the provider, and finally decides to download it (as shown in Figure 2(d)). If the polling peer decides not to download from the current provider, it can repeat the voting process on another resource.

More examples of using unstructured overlays include NICE reputation model [40] and TrustMe [41]. All the approaches based on unstructured overlays have the security concern. Messages may be intercepted or blocked during transmission, and voting is vulnerable to collusion among peers. Therefore, no secure reputation computing or delivery can be guaranteed. Furthermore, searching or voting on an unstructured overlay is based on flooding, which incurs heavy traffic in the network. For example, in XREP, *Poll* messages are broadcast throughout the network each time a peer needs to find out the reputation of a resource or a provider. This in turn affects the scalability of the system because an increase in the number of peers can potentially lead to an exponential increase in the number of *Poll* messages and responses.

5 Other Applications of Unstructured P2P Networks

Besides file sharing, unstructured P2P networks have been widely used in other network applications. In this chapter, we discuss two example applications based on unstructured P2P networks: media streaming and VoIP.

5.1 Media Streaming: CoolStreaming

With the popularity of broadband Internet access and P2P technologies, media streaming has gone through rapid growth. Typical services include on-demand video streaming that allows users to choose and watch favorite movies anytime, Internet Protocol television (IPTV) and live streaming that provide live TV service.

In a P2P streaming system, one or multiple supplying peers who have all or part of the requested media can forward the data to the requesting peers. In turn, the requesting peers can become supplying peers for other requesting peers. Because each peer contributes its own resource (storage and network bandwidth) to the system, the whole system's capacity is vastly amplified compared to the traditional client-server architecture. The major challenges in P2P streaming include [47]:

- *Peer dynamics*: In P2P networks, peers do not always stay online in the system. Supplying peers might unexpectedly crash or leave. In this case, the requesting peers need to find new supplying peers to replace the failed ones. Therefore, the system should be highly robust to withstand such peer dynamics.
- *Limited and dynamic peer bandwidth*: Unlike powerful video servers, peers have limited bandwidth capacities. Each supplying peer might only be able to support a few requesting peers (or multiple supplying peers are required to support one requesting peer). Also, the available bandwidth of supplying peers might fluctuate. Hence, the system should be able to adaptively adjust each supplying peer's sending rate to keep the streaming quality at requesting peers unaffected.

CoolStreaming is one of the popular softwares that provide live streaming service through P2P networks [32, 51]. As the first P2P-based streaming system that attracts a remarkable amount of users, CoolStreaming has several notable features: (1) Intelligent scheduling algorithm that copes well with the bandwidth heterogeneity of peers; (2) Swarm-style architecture that builds a gossip overlay to distribute contents.

In CoolStreaming, a peer needs to search for some other peers called partners, with which the peer collaborates to download streaming contents. To construct the overlay network among partners, the system employs the Scalable Gossip Membership protocol (SCAM) [23]. The SCAM protocol is fully distributed and scalable, and can provide a uniform partial view of the whole system at each peer. Based on it, CoolStreaming forms an unstructured overlay among partners, which achieves excellent resilience against random failure and enables decentralized operation. Such an overlay is similar to the BitTorrent network [37].

In detail, a newly joined peer first contacts the boot-strap server, which responds with a randomized list of the currently active peers. The newly joined peer stores this list in its cache and randomly selects a few peers from the cache to establish TCP connections, i.e., partnership. Once the partnership is established between a pair of peers, they exchange and update their cache contents. The maximum number of peers in the cache is usually on the order of $\log N$, where N is the total number of peers in the system.

Content delivery in CoolStreaming is achieved as follows. The video stream is divided into segments of uniform length, and the availability of the segments in a peer's buffer is represented by a buffer map (BM). Note that the aforementioned cache is used to store system management data and the buffer here is used to store media content. Each peer periodically exchanges its BM with its partners. Upon receiving the BM from a peer x, a peer y chooses the data segments it does not possess and sends a request indicating the demanded data segments to x. Then, x delivers the requested data segments to y. Clearly, if a peer has multiple partners, the peer has to select one partner for each of its missing segments. This selection problem is called packet scheduling. Interested readers may refer to [48, 51] for more details.

Figure 3 shows an example of the CoolStreaming network. The system consists of a video source, a boot-strap server and four peers (labeled from A to D). The video source provides the complete video content for the system. In the figure, it delivers video content to peers B and D. The boot-strap server helps peers join the system. The real lines with double-ended arrows show the partnership between peers. For example, peer C has three partners A, B and D.

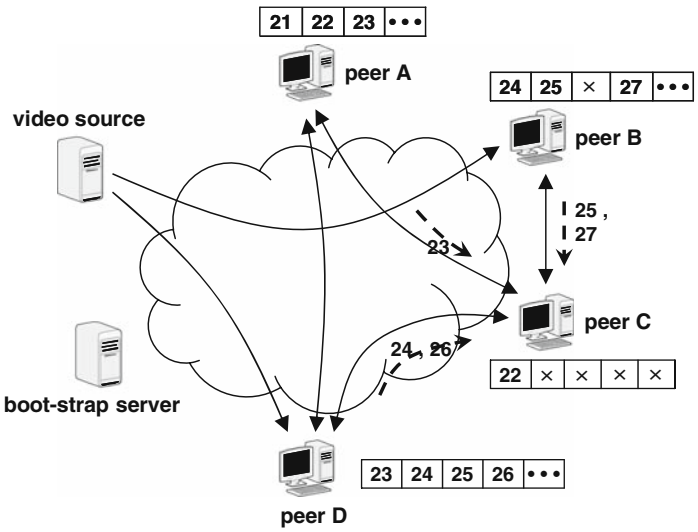

Fig. 3 An example of the CoolStreaming network

The square table along a peer shows part of the peer's BM. For example, peer *B* possesses segments 24, 25 and 27 in its buffer. But it does not possess segment 26. Missing of a segment is denoted as "×" in the BM. The subsequent segments after the 27th are not shown in *B*'s BM, which is denoted as "...". Note that peers have different starting segments in their BMs. This is because they have different play points (due to, for example, different end-to-end delay, or different downloading bandwidth).

The figure also shows an example of packet scheduling at peer *C*. Suppose *C* has known the BMs of its partners. It then uses the packet scheduling algorithm to decide to fetch which missing segment from which partner. In the figure, it requests segment 23 from peer *A*, segments 25 and 27 from peer *B*, and segments 24 and 26 from peer *D*, respectively. This scheduling imposes similar uploading load on its partners.

In summary, the advantages of CoolStreaming include:

- *Fully distributed and scalable*: Each peer distributedly joins the system and selects partners. During content delivery, a peer exchanges information with only a few partners. The whole system is hence highly scalable. This allows CoolStreaming to accommodate a large number of peers. As reported in [31, 45], it recorded over 80,000 concurrent users with an average bit rate of 400 Kbps.
- *Highly resilient to peer dynamics*: The use of multiple partners and the corresponding multiple path delivery at peers provide high system resilience. If some partners of a peer unexpectedly leave, the peer can still retrieve data from other partners.

On the other side of the coin, CoolStreaming has some limitations due to its design.

- *High control overhead*: CoolStreaming has high control overhead for the gossip mesh maintenance and data distribution. A peer has to frequently communicate with its current and potential partners to keep a highly refreshed overlay. Otherwise, a peer cannot quickly find new partners in case of partner leaving. Furthermore, a peer has to periodically exchange BM with its partners. This further increases the control overhead.
- *High end-to-end delay*: Peers in CoolStreaming often encounter high end-to-end delay. This is sometimes fatal to the quality of service, for example, for people gambling during a live soccer play. The reason is two-fold. Firstly, the gossip mesh is randomly formed without considering peer locality. The mesh often contains long connections between faraway peers. Secondly, the procedure of BM exchange increases the delay. Before a peer can download a segment, it has to first obtain some valid BMs and then send a transmission request to the selected partner.

There has been much effort to improve CoolStreaming and study new streaming systems. For example, Ren et al. explore how to build a low-delay overlay mesh among peers by considering peer locality [39]. Zhang et al. propose new data delivery mechanism to reduce delay due to BM exchange [49]. Meanwhile, many other

P2P streaming systems are proposed and evaluated, for example, AnySee [33] and GridMedia [43] for live streaming, P2VoD [20] and oStream [15] for video-on-demand.

5.2 VoIP: Skype

VoIP service (also referred to as IP telephony, Internet telephony, or voice over broadband) allows for the transmission of voice through the Internet. Traditional telephone lines use the Public Switched Telephone Network (PSTN), which works on circuit switching and connects callers to receivers through electrical circuits. VoIP is based on packet switching, where data packets are carried across the Internet, from one computer to either another computer or a PSTN telephone. There have been many VoIP softwares in the market, for example, Skype [7], Google talk and AOL Instant Messenger. VoIP operates in different forms. Here are a few examples.

- *Computer to computer*: This is the most frequently used way of VoIP. Each computer should be equipped with a sound card, a headset consisting of earphones and microphone, and some VoIP software. Most VoIP softwares provide free service for one computer to connect to any other computer running the same software.
- *Computer to phone*: Some VoIP softwares allow users to call regular telephone landlines and mobile phones from a computer. This service is usually not free, but its cost is often lower than traditional telephone charges.
- *Phone to phone*: There are two ways to make a phone-to-phone connection: (1) Use a regular phone plugged into an Analog Terminal Adaptor (ATA), which in turn connects to the Internet. (2) Use a VoIP phone that connects to the Internet.

Figure 4 shows the above three forms of VoIP systems. Current VoIP softwares also provide many other features such as instant messaging, file transfer and video conferencing.

We now study Skype as an example VoIP system. Skype launched its service in 2003 and experienced rapid growth after that. In October 2005, it was purchased by eBay. According to eBay quarterly report, as of December 31, 2007, Skype had accumulated 276 million user accounts.

Skype uses a proprietary and closed-source protocol. Numerous attempts have been undertaken to study and reverse engineer the protocol [9, 24, 44]. It is believed that Skype uses a Kazaa-like P2P network. Both companies were founded by the same individuals and much of the technology in Skype was originally developed for Kazaa. This is further confirmed by comparing Skype and Kazaa traffic on the packet level [9].

The Skype system consists of three main entities: supernodes, ordinary peers and login servers. Supernodes are elected from ordinary peers which have high bandwidth, adequate processing power and no firewall. There are also a number of

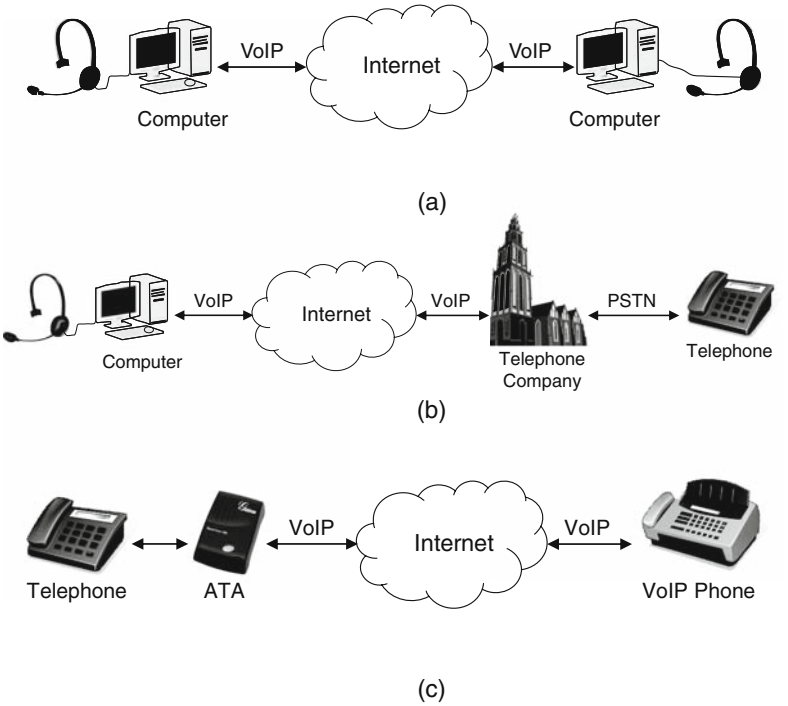

Fig. 4 Different forms of VoIP systems. (**a**) Computer-to-computer. (**b**) Computer-to-phone. (**c**) Phone-to-phone.

pre-deployed supernodes in the system, which keep staying online.Supernodes maintain an overlay network among themselves.

When login, a peer first connects to the Skype network. This is achieved by establishing a TCP connection and exchanging information with a supernode. To do that, the new peer contacts a default supernode to obtain a list of supernodes. The peer then caches the supernode list and regularly refreshes it. If the TCP connection fails, the peer tries to connect to some bootstrap IP addresses hard-coded in the client software (in version 1.2), or simply generates a login failure report (in version 0.97) [9]. After connecting to the Skype network, the peer authenticates the username and password with the Skype login server. An obfuscated list of servers has been hardcoded in the client software. Then the peer advertises its presence to other peers, determines the type of network address translator (NAT) and firewall it is behind and discovers peers that have public IP addresses.

When a peer *A* wants to call another peer *B*, *A* first queries some supernodes for *B*'s address. Through a search among supernodes, *A* can obtain *B*'s address. If both *A* and *B* are publicly reachable, *A* sets up a connection to *B* and directly exchanges voice traffic with *B*. If any participating peer is behind firewall or NAT, it sets up a connection to a supernode. The transmission of voice traffic is then relayed by the

supernode. Sometimes Skype also routes calls through ordinary peers to ease the crossing of Symmetric NATs and firewalls.

The use of supernodes for communication has many advantages. Firstly, peers behind NATs or firewalls are not publicly reachable. Through public supernodes, such peers can be reached and called. Secondly, supernodes can manage multi-user sessions such as conferencing. They can store messages from different users and accordingly forward them. On the other hand, this network structure puts heavy burden on supernodes, leading to unfair work loads among peers.

6 Mobile Unstructured P2P Networks

With the advance of mobile devices and technologies, data distribution among handhelds has become a reality. Wireless networks share many similar features with P2P networks, e.g., distributed network structure and dynamic network topology. In this chapter, we discuss how to implement unstructured P2P networks over wireless networks.

6.1 Characteristics of Mobile Wireless Networks

There are different types of wireless networks. We list a few as examples.

- *Mobile* ad-hoc *networks (MANETs)*: MANETs do not have any infrastructure support (such as base stations, access points or remote servers). All network functions are performed by the nodes forming the network, which often have high mobility and low processing power. The resulting topology is then dynamic and unstable. An example of MANET is a vehicular ad-hoc network, where wireless devices in vehicles interact with each other while moving at high speed.
- *Wireless sensor networks (WSNs)*: A WSN uses spatially distributed autonomous sensors to monitor physical or environmental conditions (such as temperature, pressure and pollutants). The sensors are low-profile devices with very limited processing power, memory and battery. The primary concern of a sensor network is its lifetime, therefore protocols for sensor nodes often focus on power conservation.
- *Wireless mesh networks (WMNs)*: A WMN consists of gateways, mesh points and wireless end-users. Gateways provide access to the Internet. Mesh points are small devices with limited processing power and memory, which are often mounted on lamp posts or rooftops. Each mesh point is associated with a certain gateway, and forwards Internet-bound traffic from associated users to the gateway. Mesh points hence extend the network coverage of gateways. While end-users are mobile and may switch their associated mesh points at any time, gateways and mesh points are generally stationary. A WMN is reliable and offers redundancy. If some mesh points fail, the rest mesh points can self-form a new mesh to continue communications.

While these wireless networks have different forms and usage, they have some common characteristics, which differentiate them from the wired Internet.

- *Wireless transmission*: Packet transmission on wireless channels is based on broadcast. And the wireless communication medium is accessible to any entity with the appropriate equipment and adequate resource. As a comparison, Internet transmission is mainly based on unicast, and sometimes multicast. In addition, wireless channels have limited bandwidth and each hop has a certain transmission range.
- *Lack of routing infrastructure*: In the wired network, routing is readily available. But in wireless networks, routing is a non-trivial issue. Two important issues in wireless routing are high maintenance overhead of routes and inefficient bandwidth utility due to long routes.
- *Low processing capability, memory capacity and energy power*: Wireless devices are often small handheld devices. They are normally low in processing capability and limited in memory capacity and energy power.

6.2 Approaches for Mobile Unstructured P2P Networks

We select MANET as an example of wireless networks. A MANET is similar to a P2P network in the following aspects.

- Both systems are distributed. In MANETs, nodes usually have low local resource and cannot serve as servers. A scalable P2P network should also be fully distributed in order to accommodate a large number of peers.
- Their topologies frequently change because of peer on/off or mobility.
- Nodes or peers in the systems have similar functionalities. They cooperate to route queries and rely messages. In both systems, flooding or broadcasting is employed to some extent for data exchange or routing among peers/nodes.

There are also some differences between P2P and MANET. For example, P2P works on the application layer in the protocol stack, while MANET focuses on the network and lower layers. As mentioned above, peers in MANET are mobile and constrained by limited energy, bandwidth and computational power, which is not a big concern in P2P systems over the Internet. And MANET uses physical broadcast but P2P uses physical unicast.

When deploying P2P networks over MANETs, the major problem is how to quickly find the requested data in spite of the mobility and the scarcity of power and bandwidth in the underlying MANET. Ding et al. propose several ways for such purpose [19]. A straightforward approach is to simply implement the P2P flooding mechanism over MANET on-demand routing protocols. That is, a query message is flooded to every virtual neighboring peer in the P2P overlay. As two virtual neighboring peers may be multiple hops away in the MANET, we need to obtain the underlying route between them. Then the network routing request is also broadcast at the network layer.

We show an example in Figure 5(a). Circles in the figure represent mobile nodes in a MANET. Two mobile nodes connected by a real line are within the transmission range of each other in the MANET. Shadowed rectangles (labeled as *A*, *E* and *F*) represent peers in an unstructured P2P network. Two peers connected by a dashed line are neighboring peers in the P2P network. From the figure, *A* and *E* are neighboring peers in the P2P network. So are *E* and *F*. In this example, if *E* wants to search for some file, it floods its query to its neighboring peers in the P2P network, i.e., *A* and *F*. Different from the wired Internet, the route from *E* to *A* (or to *F*) is not readily available and needs to be discovered through the MANET routing protocol. A broadcast for routing on the network layer is then necessary.

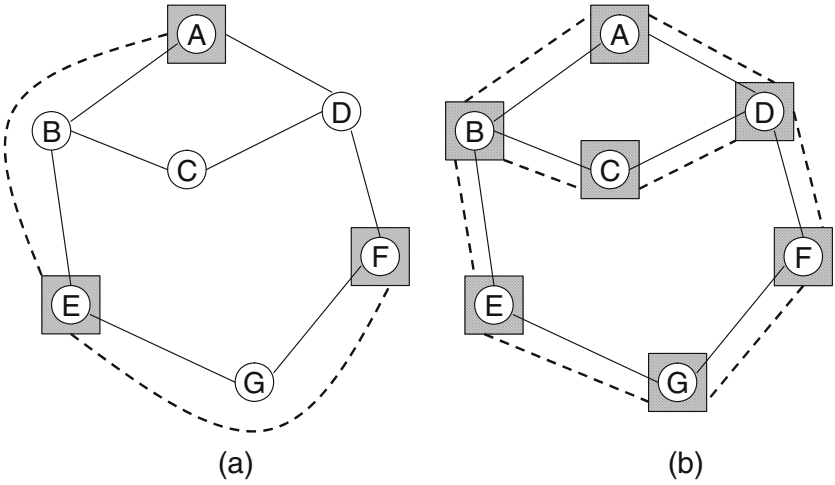

Fig. 5 Two examples of unstructured P2P network over MANET. Circles represent mobile nodes in a MANET. Two mobile nodes connected by a real line are within the transmission range of each other in the MANET. Shadowed rectangles represent peers in an unstructured P2P network. Two peers connected by a dashed line are neighboring peers in the P2P network

This approach is easy to implement. But it is not scalable due to the double-layer broadcasts. Two neighboring peers in the P2P network might be physically far away from each other. Hence, flooding in the P2P network might be expensive and inefficient. Therefore, this approach is only applicable to small MANETs.

Another approach is to map the MANET network to a P2P overlay network. Each MANET mobile node can be regarded as a peer in the P2P network. A pair of neighboring nodes in MANETs (within the transmission range of each other) correspond to a pair of neighboring peers in P2P. As wireless networks always employ broadcast to transmit data, the MANET routing protocol and the P2P flooding protocol can be implemented by one-pass broadcast.

For example, in Figure 5(b), if *E* wants to search for some file, it broadcasts a query to its neighboring mobile nodes *B* and *G*, which are also its neighboring peers

in the P2P network. Upon receiving the query message, *B* and *G* check their local file resource and continue broadcasting the message if needed.

Clearly, this method is more efficient than the first one. It directly finds the shortest path between the file source and the original requester. But the whole network is still flooded by query messages, which imposes heavy burden on communication bandwidth and power supply for mobile nodes. So it still cannot work for large MANETs.

There are many other approaches for building P2P networks over wireless networks. For example, Hora et al. study how to reduce energy consumption and delay in mobile P2P networks [26]. Akon et al. propose a novel gossip protocol to build and maintain a P2P network over a mobile wireless network [8]. Interested readers can refer to these papers for more details.

7 Conclusion

We discuss in this chapter unstructured P2P networks, one type of widely used P2P networks. In an unstructured P2P network, peers form an overlay (often in a random way) to exchange or relay data. Different from structured P2P networks, unstructured P2P networks do not impose any structure on the overlays. As a result, data storage is unrelated to the overlay, and file search essentially amounts to random search.

We discuss several applications of unstructured P2P networks, including file sharing, media streaming and VoIP. For the first one, we classify existing approaches into four categories and explore representative examples from each category. We also discuss two advanced issues in file sharing, i.e., content replication and reputation system. For the other two applications, we study their key challenges. We select one state-of-the-art approach for each application and analyze its system design.

We also investigate how to implement unstructured P2P networks over wireless networks. The characteristics of wireless networks (such as limited bandwidth and transmission range) impose new challenges for building P2P networks. We study the state-of-the-art approaches and discuss their advantages and limitations.

References

1. Gnutella. URL `http://gnutella.wego.com`
2. Kazaa. URL `http://www.kazaa.com`
3. BitTorrent. URL `http://www.bittorrent.com`
4. Napster. URL `http://www.napster.com`
5. Grokster. URL `http://www.grokster.com`
6. eBay. URL `http://www.ebay.com`
7. Skype. URL `http://www.skype.com/`

8. Akon, M., Shen, X., Naik, S., Singh, A., Zhang, Q.: An inexpensive unstructured platform for wireless mobile peer-to-peer networks. Peer-to-Peer Networking and Applications **1**(1), 75–90 (2008)
9. Baset, S.A., Schulzrinne, H.: An analysis of the Skype peer-to-peer Internet telephony protocol. In: Proc. IEEE INFOCOM'06 (2006)
10. Buchegger, S., Boudec, J.Y.L.: A robust reputation system for P2P and mobile ad-hoc networks. In: Proc. P2PEcon'04 (2004)
11. Castro, M., Costa, M., Rowstron, A.: Should we build Gnutella on a structured overlay? SIGCOMM Computer Communication Review **34**(1), 131–136 (2004)
12. Chawathe, Y., Ratnasamy, S., Breslau, L., Lanham, N., Shenker, S.: Making Gnutella-like P2P systems scalable. In: Proc. ACM SIGCOMM'03, pp. 407–418 (2003)
13. Cohen, E., Shenker, S.: Replication strategies in unstructured peer-to-peer networks. In: Proc. ACM SIGCOMM'02, pp. 177–190 (2002)
14. Cornelli, F., Damiani, E., Vimercati, S., Paraboschi, S., Samarati, P.: Choosing reputable servents in a P2P network. In: Proc. ACM WWW'02, pp. 376–386 (2002)
15. Cui, Y., Li, B., Nahrstedt, K.: oStream: Asynchronous streaming multicast in application-layer overlay networks. IEEE Journal on Selected Areas in Communications **22**(1), 91–106 (2004)
16. Damiani, E., Vimercati, S., Paraboschi, S., Samarati, P., Violante, F.: A reputation-based approach for choosing reliable resources in peer-to-peer networks. In: Proc. ACM CCS'02, pp. 207–216 (2002)
17. Despotovic, Z., Aberer, K.: Maximum likelihood estimation of peers performance in P2P networks. In: Proc. P2PEcon'04 (2004)
18. Despotovic, Z., Aberer, K.: P2P reputation management: Probabilistic estimation vs. social networks. Computer Networks **50**(4), 485–500 (2006)
19. Ding, G., Bhargava, B.: Peer-to-peer file-sharing over mobile ad hoc networks. In: Proc. IEEE PercomW'04, pp. 104–108 (2004)
20. Do, T., Hua, K.A., Tantaoui, M.: P2VoD: Providing fault tolerant video-on-demand streaming in peer-to-peer environment. In: Proc. IEEE ICC'04, pp. 1467–1472 (2004)
21. Dragovic, B., Kotsovinos, E., Hand, S., Pietzuch, P.: XenoTrust: Event-based distributed trust management. In: Proc. DEXA'03, pp. 410–414 (2003)
22. Fudenberg, D., Levine, D.: Reputation and equilibrium selection in games with a patient player. Econometrica **57**(4), 759–778 (1989)
23. Ganesh, A.J., Kermarrec, A.M., Massoulie, L.: SCAMP: Peer-to-peer lightweight membership service for large-scale group communication. In: Proc. NGC'01 (2001)
24. Guha, S., Daswani, N., Jain, R.: An experimental study of the Skype peer-to-peer VoIP system. In: Proc. IPTPS'06 (2006)
25. Hei, X., Liang, C., Liang, J., Liu, Y., Ross, K.W.: A measurement study of a large-scale P2P IPTV system. IEEE Transactions on Multimedia **9**(8), 1672–1687 (2007)
26. da Hora, D.N., Macedo, D.F., Nogueira, J.M.S., Pujolle, G.: Optimizing peer-to-peer content discovery over wireless mobile ad hoc networks. In: Proc. IFIP/IEEE MWCN'07, pp. 86–90 (2007)
27. Jin, X., Chan, S.H.G., Yiu, W.P.K., Xiong, Y., Zhang, Q.: Detecting malicious hosts in the presence of lying hosts in peer-to-peer streaming. In: Proc. IEEE ICME'06, pp. 1537–1540 (2006)
28. Jun, S., Ahamad, M., Xu, J.: Robust information dissemination in uncooperative environments. In: Proc. IEEE ICDCS'05, pp. 293–302 (2005)
29. Kamvar, S.D., Schlosser, M.T., Garcia-Molina, H.: The EigenTrust algorithm for reputation management in P2P networks. In: Proc. WWW'03, pp. 640–651 (2003)
30. Kreps, D., Wilson, R.: Reputation and imperfect information. Journal of Economic Theory **27**(2), 253–279 (1982)
31. Li, B., Xie, S., Keung, G., Liu, J., Stoica, I., Zhang, H., Zhang, X.: An empirical study of the Coolstreaming+ system. IEEE Journal on Selected Areas in Communications **25**(9), 1627–1639 (2007)

32. Li, B., Xie, S., Qu, Y., Keung, G., Lin, C., Liu, J., Zhang, X.: Inside the new Coolstreaming: Principles, measurements and performance implications. In: Proc. IEEE INFOCOM'08, pp. 1031–1039 (2008)
33. Liao, X., Jin, H., Liu, Y., Ni, L.M., Deng, D.: Anysee: Peer-to-peer live streaming. In: Proc. IEEE INFOCOM'06 (2006)
34. Lv, Q., Cao, P., Cohen, E., Li, K., Shenker, S.: Search and replication in unstructured peer-to-peer networks. In: Proc. ICS '02, pp. 84–95 (2002)
35. Mekouar, L., Iraqi, Y., Boutaba, R.: Peer-to-peer's most wanted: Malicious peers. Computer Networks **50**(4), 545–562 (2006)
36. Mui, L., Mohtashemi, M., Halberstadt, A.: A computational model of trust and reputation. In: Proc. HICSS'02, pp. 2431–2439 (2002)
37. Qiu, D., Srikant, R.: Modeling and performance analysis of BitTorrent-like peer-to-peer networks. In: Proc. ACM SIGCOMM'04, pp. 367–378 (2004)
38. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Schenker, S.: A scalable content-addressable network. In: Proc. ACM SIGCOMM'01, pp. 161–172 (2001)
39. Ren, D.N., Li, Y.T.H., Chan, S.H.G.: On reducing mesh delay for peer-to-peer live streaming. In: Proc. IEEE INFOCOM'08, pp. 1058–1066 (2008)
40. Sherwood, R., Lee, S., Bhattacharjee, B.: Cooperative peer groups in NICE. Computer Networks **50**(4), 523–544 (2006)
41. Singh, A., Liu, L.: TrustMe: Anonymous management of trust relationships in decentralized P2P systems. In: Proc. IEEE P2P'03, pp. 142–149 (2003)
42. Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for Internet applications. In: Proc. ACM SIGCOMM'01, pp. 149–160 (2001)
43. Tang, Y., Luo, J.G., Zhang, Q., Zhang, M., Yang, S.Q.: Deploying P2P networks for large-scale live video-streaming service. IEEE Communications Magazine **45**(6), 100–106 (2007)
44. Xie, H., Yang, Y.R.: A measurement-based study of the Skype peer-to-peer VoIP performance. In: Proc. IPTPS'07 (2007)
45. Xie, S., Li, B., Keung, G., Zhang, X.: Coolstreaming: Design, theory, and practice. IEEE Transactions on Multimedia **9**(8), 1661–1671 (2007)
46. Xiong, L., Liu, L.: PeerTrust: Supporting reputation-based trust for peer-to-peer electronic communities. IEEE Transactions on Knowledge and Data Engineering **16**(7), 843–857 (2004)
47. Yiu, W.P.K., Jin, X., Chan, S.H.G.: Challenges and approaches in large-scale P2P media streaming. IEEE Multimedia **14**(2), 50–59 (2007)
48. Zhang, M., Xiong, Y., Zhang, Q., Yang, S.: On the optimal scheduling for media streaming in data-driven overlay networks. In: Proc. IEEE Globecom'06 (2006)
49. Zhang, M., Zhang, Q., Sun, L., Yang, S.: Understanding the power of pull-based streaming protocol: Can we do better? IEEE Journal on Selected Areas in Communications **25**(9), 1678–1694 (2007)
50. Zhang, X., Liu, J., Li, B.: On large scale peer-to-peer video streaming: Experiments and empirical studies. In: Proc. IEEE MMSP'05 (2005)
51. Zhang, X., Liu, J., Li, B., Yum, T.S.P.: CoolStreaming/DONet: A data-driven overlay network for peer-to-peer live media streaming. In: Proc. IEEE INFOCOM'05, pp. 2102–2111 (2005)