Structuring Unstructured Peer-to-Peer Networks

Stefan Schmid and Roger Wattenhofer

Computer Engineering and Networks Laboratory
ETH Zurich
8092 Zurich, Switzerland

Abstract. Flooding is a fundamental building block of unstructured peer-to-peer (P2P) systems. In this paper, we investigate techniques to improve the performance of flooding. In particular, we present *Clustella*, a novel semi-structured P2P architecture with bounded peer degree. Clustella decomposes the network into different clusters, allowing peers to quickly find those neighbors which contribute much to their routing efficiency. By its link selection strategy, Clustella achieves a good performance in static and dynamic environments.

1 Introduction

While *distributed hash tables* (DHT) are well-studied in the research community, most peer-to-peer (P2P) systems in today's Internet are still *unstructured*. Unstructured architectures are attractive because of their simplicity and their high robustness.

In unstructured systems, there is neither a centralized directory nor any control over the network topology or resource placement. When a new peer joins the P2P network, it forms connections with other peers freely, e.g., it selects *arbitrary* peers as neighbors. In order to publish its resources, a peer usually just stores them locally or places them on randomly chosen peers. Generally, unstructured overlays have loose guarantees for resource discovery, and it is possible that a file is not found although it exists in the network.

It is often believed that—due to the absence of topological constraints—such unstructured systems have a better performance and require less maintenance overhead in highly dynamic environments where peers join and leave frequently and concurrently. Usually, these systems also support richer queries than just search by identifier, for example keyword searches with regular expressions, range queries, etc.

The main Achilles heel of unstructured P2P systems are the underdeveloped routing mechanisms. Basically, there are two fundamental routing operations: *flooding* and *random walks*. In flooding, a search packet with a limited *time-to-live* (TTL)—maximally 10 hops in Gnutella for example—is repeatedly forwarded to all neighbors, while in random walks, a packet is only forwarded to one randomly chosen neighboring peer. While a random walk is usually the less costly alternative in terms of the number of messages sent per query, the flooding approach is more robust and has better response times. This paper focuses on the flooding mechanism. However, we believe that our techniques are also useful in systems based on random walks.

The major concern about flooding is the total number of messages caused per query. More severely, in practice many of these messages are of no use and unnecessarily increase the load on the system. The main reason are redundant retransmissions: If a peer's neighbors are likely to be neighbors as well, the peer receives the same packet multiple times.

In this paper, we introduce a measure to evaluate the efficiency of flooding on a given topology. We believe that this criterion captures the essence of flooding well, and also engenders many interesting theoretical questions. We then identify means to structure the topology in order to improve the quality of floods with respect to this criterion. In particular, we present *Clustella*, a novel semi-structured P2P network.

Clustella is a fully decentralized (local) system with undirected connections only and a limited peer degree. Beacons are used to decompose the network into clusters. Peers can orient themselves using the beacon information, and quickly find other peers which—if a link to them is established—significantly increase the number of peers covered by floods. If the found peer already has full degree, local link rotations are applied and—also in this case—the connection request can be satisfied quickly with a good neighbor. Small cycles in the topology—and hence redundant messages—are avoided. Moreover, Clustella is self-stabilizing and maintains its routing performance also in dynamic environments.

The rest of this paper is organized as follows. After reviewing related work in Section 2, the model in general and the flood coverage criterion in particular are introduced in Section 3. The Clustella architecture is described in Section 4. Simulation results are presented in Section 5. After briefly discussing possible extensions in Section 6, the paper is concluded in Section 7.

2 Related Work

Gnutella [17] is probably the most prominent unstructured P2P network. However, albeit its success, there have been concerns about its scalability from the beginning [19,20]. Indeed, when Napster was unplugged in 2001, Gnutella broke down soon afterwards due to the inrush of former Napster users. A lot of interesting solutions have been proposed since then.

In spite of the large literature about structured *distributed hash tables* [18,21,23,26] (some authors have even proposed to implement unstructured data placement schemes on top of structured systems [2,3]), many researchers have aimed at improving the performance of unstructured systems, acknowledging their simplicity and their predominance in today's Internet. One active thread of research concerns content movement or *replication* [5,6,16,27]. In particular, Cohen and Shenker [5] have shown that search is most efficient if the number of replicas is proportional to the square root of the object's popularity.

Another fruitful field—also for structured networks—is *interest-based locality* [11, 12, 22]. The idea is that instead of (or additionally to) random neighbor connections, peers should establish connections to peers with similar interests. It has been shown that thereby queries can be satisfied with a much smaller flooding radius.

Many recent solutions for unstructured systems use alternatives for the flooding operations, for example random walks (e.g. [4]). But there have also been proposals to improve flooding itself. In [15, 25], the flooding is executed in several successive rounds with increasing TTL, until enough responses are received. While this solution can effectively reduce the message complexity for finding popular files, the response times are worse.

The work which is the closest related to ours is by Jiang et al. [9]. Their scheme aims at minimizing the number of redundant messages by constructing a tree-like sub-overlay on which the packets are propagated. However, in contrast to our work, many connections have to be maintained which are of no use for flooding. This also implies that—compared to the total number of connections in the system—the amount of peers covered by a flooding is low. More severely, the tree-like sub-overlay may result in disconnected components, especially in dynamic environments, reducing the efficiency further. In contrast, in our system, *all* links can be used for flooding, since they have actively been selected in consideration of their quality. Hence, while redundant messages are also rare, the flood coverage is much larger.

Note that our paper is related to literature on *virtual coordinate systems* [7,8]. These systems typically assign coordinates to the different peers in order to estimate distances (in terms of *latency*, rather than number of *hops*) between a node and its (potential) neighbors. However, in this paper, we do not make use of these techniques.

Finally, the idea of structuring unstructured systems is also used by researchers in order to avoid a mismatch of the overlay with the underlying, real network [13].

3 Model

We model the P2P network as an *undirected* graph $G = (V, E)$, where V is the set of peers and E the set of connections between the peers. That is, for $u, v \in V$, $\{u, v\} \in E$ denotes that peers u and v know the IP addresses of each other. The *r-neighborhood* $\Gamma^r(v)$ of a peer $v \in V$ is defined as the set of peers which are at most r hops away from peer $v \in V$, excluding v itself. For v's direct neighbors, we use the short form $\Gamma(v)$ instead of $\Gamma^1(v)$. Moreover, let R be the TTL or flooding radius of the system (e.g., $R \leq 10$ in Gnutella). As will be discussed in Section 4, and unlike some other unstructured systems, in Clustella, every peer has at least δ but no more than Δ neighbors, i.e., for all $v \in V$, $\delta \leq |\Gamma(v)| \leq \Delta$.

In this paper, a pure flooding algorithm is considered where each peer forwards a packet to all its neighbors as long as the packet has a non-zero TTL. In order to maximize the probability of finding a data item or file, a flooding operation should reach—for a given radius or TTL—as many peers as possible. Therefore, $|\Gamma^R(v)|$—the size of the R-neighborhood of a peer v—is a natural criterion to quantify the efficiency of a flooding operation. In the following, we will refer to $|\Gamma^R(v)|$ as v's *flood coverage*. The *flood coverage of a network network* $G = (V, E)$ is defined as the minimal flood coverage of all peers in the network (cf. Definition 3.1).

Definition 3.1. *The* flood coverage $\Xi(G)$ *of a topology* G *for a given flooding radius* R *is defined as*

$$\Xi(G) = \min_{v \in V} |\Gamma^R(v)|.$$

Of course, not every network topology is equally suited for flooding. If a peer has neighbors which are also neighboring, many redundant messages are sent which do not increase the propagation scope. In an optimal topology G, the number of peers reached grows exponentially per hop, and if all peers have degree Δ, it holds that $\Xi(G) = \min\left\{\sum_{i=1}^{i=R} \Delta(\Delta-1)^{i-1}, |V|\right\}$.

Observe that our definition of a network's flood coverage is somehow related to the important criterion of *graph expansion* [24]. However, there are two crucial differences: First, we are not concerned with the expansion of all *subsets* of peers, but of single peers only (subsets of size one). And second, for these peers the entire R-neighborhood is considered (instead of just their immediate neighbors).

An ideal topology achieving maximal flood coverage must have a large *girth*, i.e., large minimal cycles. While finding such graphs is an interesting research area on its own (cf. [10] for an explicit construction), we do not follow these theoretical considerations further but go on and describe our semi-structured P2P system Clustella which strives—in a decentralized manner—for creating topologies with large flood coverage.

4 Clustella

In this section, we introduce the basic techniques used in Clustella for creating topologies with large flood coverage. As described in Section 3, a peer should connect to peers of different areas of the network, such that the shortest path between two neighbors $\pi', \pi'' \in \Gamma(\pi)$ of a peer π—*except* for the one via π itself—is long.

However, in unstructured P2P systems, if a peer π learns about another peer π', π has a priori no information about π''s location in the network, and thus also not about the hop distance between π and π'. Therefore, π can not decide whether it is useful to connect to π', or whether its flood coverage using the existing neighbors is better.

4.1 Clustering Topology and Beacons

A natural way to get rough topological location information is to decompose the network into *clusters* or zones. The idea is that if the neighbors are chosen from different clusters, then they are likely to be distant from each other. In order to be applicable in realistic and dynamic environments, our system must fulfill three properties: (1) The clustering mechanism should not require the peers to perform global operations or gather large amounts of additional topological information. (2) The clustering should be established quickly and in a decentralized manner. (3) The solution should be fair in the sense that all peers incur more or less the same amount of work.

Such a clustering can be achieved by having some peers assuming the role of *beacons*. Concretely, in Clustella, if a peer has no beacon in its R_d-neighborhood, it considers itself a beacon. Furthermore, a protocol ensures that each peer knows the beacons in its R_b-neighborhood. Both R_d and R_b are system parameters which are explained later in more detail. For now, assume that $R_d < R_b$ and $R_b \approx R$. In order to find out whether it is worth establishing a connection to each other, two peers π and π' exchange the information about their beacons. If their neighborhoods look very different, the distance between the peers must be large and thus the connection $\{\pi, \pi'\}$ of good quality.

In Clustella, a beacon peer appends its identifier (IP address) to the packets passing through it, together with a TTL initialized to R_b. The other peers then forward this packet as usual, decrementing the TTL in every step. Since the packet may have travelled several hops before a beacon appends its identifier, and since the beacon parameter R_b can be larger than the packet flooding radius, the packet's TTL can expire before the beacon information is propagated R_b hops far. Therefore, if a peer receives a packet with TTL=0, it buffers the beacon identifiers with non-zero hop count, and piggybacks them on future packets. The propagation of the beacon information becomes independent of the packets' TTLs, and beacons are indeed known in their R_b-neighborhoods. Moreover, note that due to the piggybacking, Clustella itself does not require the transmission of any messages at all.[1] This solution also fulfills the fairness property (Property (3)), as the work is equally divided between both beacon and non-beacon peers. The basic ideas of the clustering topology are illustrated in Figure 1.

4.2 Neighbor Selection

In this section, we first present Clustella's neighbor selection strategy. Afterwards, the issue of replacing existing connections by better ones is addressed.

New Neighbors. Consider a peer π which—for example during the joining process—is given a set of candidate peers from which it has to choose the best neighbor. For each such candidate π', π knows—due to a preceding information exchange—the identifer of π''s closest beacon, and which beacons it has in common with π'. If π itself does not see the closest beacon of π', the hop distance between π and π' must be larger than $R_b - R_d$. In addition to this deterministic guarantee, the amount of beacons which are in *both* π's and π''s R_b-neighborhood

[1] Of course, however, the *size* of the messages becomes larger as they include beacon information.

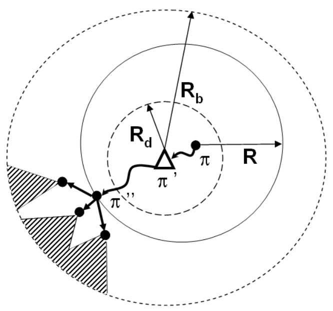

Fig. 1. When the query flooded by peer π arrives at the beacon π', π' appends its identifier. The packet is forwarded until its TTL becomes zero at peer π''. Peer π'' extracts the beacon's identifier and piggybacks it to other packets. Hence, the information about π' is propagated in the entire R_b-neighborhood of π'.

correlates roughly with the length of the shortest path between the two peers. In Clustella, the following algorithm is applied to select the best candidate: From all candidates of which π does not see the closest beacon (if possible), π chooses the one with which it has the least beacons in common.[2]

In our system, connections are undirected, and there is an upper bound Δ on the amount of links a peer can have. This poses an interesting problem: What happens if a peer wants to connect to another peer which already has Δ neighbors? To find alternative candidates can be time-consuming, or even impossible if all existing peers have full degree. In the following, we describe Clustella's joining procedure which—in spite of this problem—quickly establishes new connections of good quality (i.e., between formerly distant peers).

First recall that a peer is allowed to have between δ and Δ neighbors. When a peer joins the Clustella network (or if some of its neighbors have crashed), it only looks for new neighbors as long as its degree is smaller than δ. However, a peer π always accepts a connection request from another peer π' if it has less than Δ neighbors. If π already has Δ neighbors when it receives the connection request from π', it checks whether there exists a neighbor $\pi'' \in \Gamma(\pi)$ with degree smaller than Δ. If this is the case, the connection $\{\pi', \pi''\}$ is built. Since π'' is adjacent to π, the quality of the link $\{\pi', \pi''\}$ is similar to the one of $\{\pi', \pi\}$. If on the other hand the degrees of all neighbors are also Δ, π drops a connection with an arbitrary existing neighbor π'' and accepts π''s joining request. While this would already be a good solution, Clustella exploits the situation further and additionally establishes a connection between π' and π''. Note that since π' had a degree of at most $\delta - 1 < \Delta - 2$ before issuing the join request, this operation is legal. More importantly, the link $\{\pi', \pi''\}$ must be of good quality (similar to the one of $\{\pi, \pi''\}$ before it was broken). Finally, this trick also speeds up the joining process. We have the following result.

Theorem 4.1 *In Clustella, by choosing $\delta < \Delta$, a connection request at a peer π can always be satisfied by π itself or by a neighboring peer $\pi' \in \Gamma(\pi)$.*

[2] Note that the distances to the beacons are not considered.

Also observe that since the links established by Clustella are between distant peers only, a new link does not degrade the quality of existing links.

Our system only requires that $\delta < \Delta$, and hence choosing $\delta = \Delta - 1$ is fine. However, it may be beneficial to have a larger range of allowed degrees. With this increased flexibility, small local link rotations could be performed in order to increase the flood coverage further during joins. Such mechanisms are planned for future versions of Clustella.

Neighbor Replacement. A peer can always try to replace its current neighbors by better ones. However, the evaluation of existing neighbors of a peer π poses a problem: Since they are already adjacent to π, they must as well have almost the same set of beacons in their neighborhood.

One simple solution would be to ignore this issue and just change the neighbors once in a while. As a slight improvement, the neighbors could be assigned a score depending on the beacons they had *before* the link has been established; then, depending on these scores, sporadically the worst neighbor is replaced.

In Clustella, a more sophisticated approach is used: Each beacon information record also contains its flooding path, i.e., a peer adds its identifier (at most R_b entries) before forwarding the packet. Thereby, the peers are able to compute via which neighbors they know about a given beacon. In order to evaluate an existing neighbor π', a peer π asks for π''s beacons without the ones known through the link $\{\pi, \pi'\}$. If π' and π still have one or more beacons in common, then the connection has to be replaced.

4.3 Dynamic Failures

Most P2P systems currently in use are very transient and have a high peer turnover rate. Therefore, it is important that Clustella can efficiently handle peers which leave or crash. The creation of beacons—and the propagation of the corresponding information—is a fully decentralized process and therefore our system adapts quickly to changing environments on its own.

However, it is beneficial to employ additional mechanisms to cope with churn. If a neighbor of a peer crashes, a good substitute has to be found—a costly operation if done from scratch. Therefore, in Clustella, a peer stores for each neighbor π some of π's neighbors. When π crashes, a connection to one of π's former neighbors can be established immediately. The quality of this alternative connection is similar to the one of the old connection.

5 Simulation

We have analyzed the flood coverage of the topologies created by Clustella by simulation for up to 1 million peers. Our tests mainly focused on the parameter space $\Delta \in \{4, ..., 7\}$ (each with $\delta = \Delta - 1$) and $R \in \{5, ..., 10\}$.

Although choosing R_d smaller than R_b gives a deterministic guarantee for the minimal girth, this feature can not be exploited fully since it is very expensive:

The amount of beacon information per peer grows quickly as longer shortest paths are enforced this way. However, such a hard guarantee seems not to be necessary, as already $R_d = R_b - 2$ yields very good results: Since $R_d < R_b$, a peer has enough beacons in its neighborhood for orientation, but also not too many even in case of a large beacon radius R_b. Generally, R_b should roughly equal—or be slightly larger than—R, i.e., $R_b \approx R$.

In our simulations, the following neighbor discovery algorithm has been used: In order to find an additional neighbor, a peer π in the Clustella network sends an exploration packet of only a small constant TTL (e.g. 10 hops). This packet contains information about the beacons in the neighborhood of π, plus a section where already visited peers on the path can be stored. A peer π' which receives this packet forwards it to its neighbor π'' which—without the connection $\{\pi', \pi''\}$—shares the least beacons with π. Neighbors which are already contained in the path are avoided, and if several neighbors are equally well-suited, a random one is chosen.

We have compared Clustella to two other strategies: a *Gnutella-like strategy* and a *random walk strategy*. In the Gnutella-like strategy, in order to find new peers it can connect to, a peer asks its neighbors for their neighbors. This procedure is repeated recursively, until the peer reaches its desired degree. In the random walk strategy, a peer sends a discovery packet with the same TTL as Clustella. However, unlike in Clustella, this packet is always forwarded to a random neighbor, and does not benefit from the beacon information for orientation.

In all our simulations, the performance of the Gnutella-like strategy was of course poor: The flood coverage was up to one hundred times worse than the coverage of the other two strategies. The resulting topologies were highly clustered, and the neighbors' neighbors were often adjacent. While the random walk strategy had a much better flood coverage than the Gnutella-like strategy, it was outperformed by Clustella where peers quickly reached much more distant neighbors.

Finally, since existing P2P systems often use longer random walks in order to find new neighbors, we have also studied a different scenario. Thereby, both Clustella and the random walk strategy chose neighbor candidates uniformly and at random *from the entire network*. Such a *uniform sampling* can be achieved by sufficiently long random walks. Clustella outperformed this random graph, albeit only by up to slightly more than 10%. (Of course, for very large graphs where only a small fraction of peers can be reached by a flooding, the difference of the coverage of the two graphs diminishes. Similarly, for very small graphs where all peers can be reached, the flood coverage is the same.) However, we believe that this uniform sampling scenario is not realistic in practice, as the *mixing times* [14]—and thus the length—of the random walks are large and also difficult—or even impossible in dynamic environments!—to compute. (Note that this computation requires a good estimation of the total number of peers in the system.) Furthermore, the probability that an exploration packet is lost is high for long walks, and it is necessary to send several redundant packets in parallel.

Therefore, Clustella does not apply this neighbor discovery strategy but only sends packets with small TTLs, as described above.

In conclusion, our first *in vitro* evaluation results are promising both with respect to the search efficiency and with respect to the messages' sizes. Of course, we are aware that many issues such as the dynamics of the system remain to be analyzed in detail in future work. Moreover, there is a wide variety of other alternative approaches to which have not compared Clustella. Although we plan to perform such comparisons, our focus here is rather on the introduction of novel ideas to improve flooding than on proposing a complete and ready-to-use system; in fact, Clustella can be enhanced by adding many existing heuristics, for example by introducing some form of replication, or by the extensions discussed in the next section.

6 Extensions

The basic system as described in Section 4 can be extended in several ways. In this section, we briefly discuss two possible enhancements.

The first enhancement concerns the clustering. So far, there is only one level of beacons. It might be beneficial to organize the beacons in a *hierarchy* of several levels with increasing radius of responsibility. If a peer looks for a distant peer to connect to, it can choose the candidate with which it has only high-level beacons in common. A small beacon hierarchy (three or four levels) might already do the job: Since floods have a small constant radius, the optimal flood coverage can also be achieved with peers which are not very far away.

There are several challenges. In particular, the hierarchy must be easily maintainable when peers (and thus also beacon peers) join and leave. Moreover, it should be no disadvantage to be a high-level beacon (fairness property), and information about all beacons must be propagated efficiently.

The second enhancement concerns the size of the transmitted messages. Besides general compression mechanisms, the use of *Bloom filters* [1] is appealing: Sending only the Bloom array instead of the entire beacon identifiers can reduce the burden on the system's resources (memory and bandwidth) while still yielding acceptable probabilistic guarantees.

7 Conclusion

This paper has embarked on identifying techniques to make flooding on unstructured P2P topologies much more efficient. Unlike other systems which only combat the symptoms, we strive for avoiding bad connections from the beginning. Moreover, in contrast to many structured P2P systems, Clustella can be started from arbitrary network topologies, i.e., from any connected graph, and develops towards better network structures in a *self-stabilizing manner.* Our first evaluations are promising. Moreover, many heuristics such as smart data replication are orthogonal to our approach and could be integrated to further improve search performance.

An interesting feature of our techniques is that they can *co-exist* in networks with normal clients (e.g., Gnutella clients); it is not necessary for the existing clients to know about our new clients. What is more, the entire network will benefit from our neighbor selection of—possibly a small number of—new clients, as many existing clients will experience a larger fan-out as well.

We plan to investigate efficient flooding topologies further, addressing for instance Clustella's dynamics: Does the system require measures in order to stabilize quickly, and if yes, which mechanisms are best? The ultimate goal is to have a running Clustella client which collaborates seamlessly with other unstructured P2P clients. Finally, we believe that our clustering approach may be interesting in other areas of distributed computing as well.

Acknowledgments

The authors would like to thank the participants of the DCG Mediterranean Seminar in Toscany, Italy for fruitful discussions. This research has been supported in part by the Swiss National Science Foundation (SNF).

References

1. Bloom, B.H.: Space/Time Trade-offs in Hash Coding with Allowable Errors. Commun. ACM 13(7), 422–426 (1970)
2. Castro, M., Costa, M., Rowstron, A.: Should We Build Gnutella on a Structured Overlay? In: Proc. 2nd Workshop on Hot Topics in Networks (HotNets) (2003)
3. Castro, M., Costa, M., Rowstron, A.: Peer-to-Peer Overlays: Structured, Unstructured, or Both? Technical Report MSR-TR-2004-73, Microsoft Research, Cambridge, UK (2004)
4. Chawathe, Y., Ratnasamy, S., Breslau, L., Lanham, N., Shenker, S.: Making Gnutella-like P2P Systems Scalable. In: Proc. Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM) (2003)
5. Cohen, E., Shenker, S.: Replication Strategies in Unstructured Peer-to-Peer Networks. In: Proc. ACM SIGCOMM Conference (2002)
6. Cooper, B.F.: Quickly Routing Searches Without Having to Move Content. In: Castro, M., van Renesse, R. (eds.) IPTPS 2005. LNCS, vol. 3640, pp. 163–172. Springer, Heidelberg (2005)
7. Dabek, F., Cox, R., Kaashoek, F., Morris, R.: Vivaldi: A Decentralized Network Coordinate System. In: Proc. Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM), pp. 15–26 (2004)
8. Francis, P., Jamin, S., Jin, C., Jin, Y., Raz, D., Shavitt, Y., Zhang, L.: IDMaps: A Global Internet Host Distance Estimation Service. IEEE/ACM Trans. Netw. 9(5), 525–540 (2001)
9. Jiang, S., Guo, L., Zhang, X.: LightFlood: An Efficient Flooding Scheme for File Search in Unstructured Peer-to-Peer Systems. In: Proc. Itl. Conf. on Parallel Processing (ICPP) (2003)
10. Lazebnik, F., Ustimenko, V.A.: Explicit Construction of Graphs with Arbitrary Large Girth and of Large Size. Discrete Applied Math. 60, 275–284 (1997)

11. Le Blond, S., Guillaume, J.-L., Latapy, M.: Clustering in P2P Exchanges and Consequences on Performances. In: Castro, M., van Renesse, R. (eds.) IPTPS 2005. LNCS, vol. 3640, Springer, Heidelberg (2005)
12. Le Fessant, F., Handurukande, S., Kermarrec, A.-M., Massoulié., L.: Clustering in Peer-to-Peer File Sharing Workloads. In: Voelker, G.M., Shenker, S. (eds.) IPTPS 2004. LNCS, vol. 3279, Springer, Heidelberg (2005)
13. Liu, X., Xiao, L., Liu, Y., Ni, L.M., Zhang, X.: Location Awareness in Unstructured Peer-to-Peer Systems. IEEE Trans. Parallel Distrib. Syst. 16(2), 163–174 (2005)
14. Lovász, L.: Random Walks on Graphs: A Survey. Combinatorics 2 (1993)
15. Lv, Q., Cao, P., Cohen, E., Li, K., Shenker, S.: Search and Replication in Unstructured Peer-to-Peer Networks. In: Proc. 16th ACM Itl. Conf. on Supercomputing (ICS) (2002)
16. Morselli, R., Bhattacharjee, B., Srinivasan, A., Marsh, M.A.: Efficient Lookup on Unstructured Topologies. In: Proc. 24th Annual Symposium on Principles of Distributed Computing (PODC), pp. 77–86 (2005)
17. Open Source Community. Gnutella (2001), `http://gnutella.wego.com/`
18. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S.: A Scalable Content Addressable Network. In: Proc. of ACM SIGCOMM 2001 (2001)
19. Ripeanu, M., Foster, I.: Mapping Gnutella Networks. IEEE Internet Computing , 50–57 (2002)
20. Ritter, J.: Why Gnutella Can't Scale. No, Really (2001)
21. Rowstron, A., Druschel, P.: Pastry: Scalable, Decentralized Object Location and Routing for Large-Scale Peer-to-Peer Systems. In: Proc. 18th IFIP/ACM Int. Conference on Distributed Systems Platforms (Middleware), pp. 329–350 (2001)
22. Sripanidkulchai, K., Maggs, B., Zhang, H.: Efficient Content Location Using Interest-Based Locality in Peer-to-Peer Systems. In: Proc. 22nd IEEE Conf. on Computer Communications (INFOCOM) (2003)
23. Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H.: Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In: Proc., A.S. (ed.) Proc. ACM SIGCOMM Conference (2001)
24. Walters, I.: The Ever Expanding Expander Coefficients. Bull. Inst. Combin. Appl. 97 (1996)
25. Yang, B., Garcia-Molina, H.: Improving Search in Peer-to-Peer Systems. In: Proc. 22nd Itl. Conf. on Distributed Computing Systems (ICDCS) (2002)
26. Zhao, B.Y., Huang, L., Stribling, J., Joseph, A.D., Kubiatowicz, J.D.: Tapestry: A Resilient Global-scale Overlay for Service Deployment. IEEE Journal on Selected Areas in Communications 22(1) (2004)
27. Zhong, M., Shen, K.: Popularity-Biased Random Walks for Peer-to-Peer Search under the Square-Root Principle. In: Proc. 5th Intl. Workshop on Peer-to-Peer Systems (IPTPS) (2006)