Proactive Uniform Data Replication by Density Estimation in Apollonian P2P Networks

Nicolas Bonnel, Gildas Ménier, Pierre-François Marteau

Proactive Uniform Data Replication by Density Estimation in Apollonian P2P Networks

Nicolas Bonnel, Gildas Ménier and Pierre-Francois Marteau

Valoria, European University of Brittany, 56000 Vannes, France

Abstract. We propose a data replication scheme on a random apollonian P2P overlay that benefits from the small world and scale free properties. The proposed algorithm features a replica density estimation and a space filling mechanism designed to avoid redundant messages. Not only it provides uniform replication of the data stored into the network but it also improves on classical flooding approaches by removing any redundancy. This last property is obtained at the cost of maintaining a random apollonian overlay. Thanks to the small world and scale free properties of the random apollonian P2P overlay, the search efficiency of the space filling tree algorithm we propose has comparable performances with the classical flooding algorithm on a random network.

1 Introduction

Although costly (bandwidth wise and/or memory side), data replication remains a crucial operation for distributed systems to ensure scalability and fault tolerance. Optimizing the data replication is needed to harness the expensive cost of creating replicas : there must be as few replicas as possible and these replicates should be wisely chosen and distributed. In unstructured P2P network [5, 9, 16], most of the search strategies involves a blind query propagation leading to a cost proportional to the number of random nodes to visit, called the Expected Search Size (ESS) [6].

In uniform replication, since data are evenly replicated, the ESS is minimized for high quantity of unsolvable queries [6]. This strategy can be achieved as follows: each time a new piece of data is inserted into the network, a fixed number of replicas is created and spread accross the network. This strategy is efficient in structured P2P networks [11, 15, 10, 7] because routing protocols allow to easily query for the presence of the data into the network. On the other hand, this approach is very costly to implement in unstructured P2P network since data presence can be answered only by querying the whole network, which is very costly.

A replication strategy that is lineary proportional to popularity is used in most unstructured or centralized P2P architecture designed to exchange files as it lowers the access cost to popular data. For low number of unsolvable queries, the best performance is reached when using the square root of popularity [6]. It is defined as follows: given two pieces of data A and B in the network, the

ratio between the number of copy R_A and R_B of these data is the square root of the ratio of their popularity P_A and P_B : $\frac{R_A}{R_B} = \sqrt{\frac{P_A}{P_B}}$. Freenet [4] achieves this replication mechanism thru path replication as stated in [6].

In most P2P structured architectures, data can be replicated when a node leaves the network : the missing data is detected because of the direct relationship used to bind data to a node-location. This reactive replication only uses bandwidth when necessary thus leading to accidental surges of bandwidth use that can be a problem for overlying applications.

Proactive replication scheme anticipates this problem by wisely replicating data to balance the load over time [14] however at a cost of a slightly higher network traffic that can be tuned to a bandwidth budget (see also path replication scheme in Freenet [4]).

We present in this paper a uniform replication strategy that uses a local replica density estimation. Because performances of the algorithm are linked to the exploration strategy, we also present an unstructured random apollonian P2P overlay that features non redondant exploration scheme. Next section introduces data replication by density estimation, section 3 presents random apollonian P2P networks, section 4 shows and discusses experiments and section 5 concludes this papers with suggestions for future work.

2 data replication by density estimation

Since the relationship between data position and node position in unstructured network is not defined as in structured P2P, the query language capabilities are not limited or constrained by any assumptions. In this study, we focus on languages with XPath-like [12] or XQuery-like [3] features. The more specific queries are, the least answers are returned: this also increases the number of unsolvable queries. As stated previously, uniform replication seems to be one of the best approaches when the number of unsolvable queries is high because it minimizes the expected search size. Because it may be difficult to detect missing data after a node departure, we focus here on proactive replication.

In this paper, we propose a uniform proactive replication scheme based on a local estimation of replica density for unstructured P2P networks where the average number of replica remains proportional to the (unknown) network size. This proactive strategy can be tuned to the amount of available network resources: nodes with higher bandwidth can replicate data more often than the ones featuring low bandwidth.

2.1 Principle

Our architecture is designed as follows: each user (node) n has two storage spaces. The first one is its home space E_n in which all data explicitely downloaded by n or shared with other users are stored. The second storage space is a cache C_n whose size is controlled by the users. C_n is used by our architecture to store data replicas. In order to distributively maintain the correct amount of replica,

a score is periodically computed for each replica. According to their score, data are then replicated, kept or removed.

2.2 Replica management

Let each piece of data have a unique identifier d. Each node n in the network maintains a list L_n containing all pieces of data that are candidates for replication. Elements in L_n are submited by other peers in the network. More precisely, L_n contains pairs (d, A_m), with d a data identifier and A_m the address of the node m that submited the data to n and owns a local copy of this data. We make no assumption on the size of L_n.

Score measurement Let S_d be the measure of the score for a piece of data d. S_d is related to the density estimation of the replicas for d. This estimation is made by exploring the neighborhood of the node and counting the number of replicas of d encountered during this exploration. Therefore, most replicated data should have the highest scores.

Proactive uniform replication In the following part of the paper, we do not take into account heterogeneity in data size. The replication algorithm features several stages that are performed periodically (and asynchronously) on every node of the network. For each node n, a data candidate d is selected for replication/removal according to a score S_d. As described below, S_d is related to the local replica density which is estimated by a local exploration of the neighbourhood of the node n. This exploration quantifies the number of d on each node n.

For each piece of data d that could be potentially replicated, if its score S_d is lower than the average score of all data in the cache of node n, then the piece of data with the highest score in C_n is deleted and d is dowloaded into C_n. When this is finished, L_n is cleared.

The node n then selects k pieces of data in $C_n \cup E_n$ having the lowest scores, contacts k nodes selected at random (with a random walk) and submits to each node a piece of data for replication. The algorithm 1 describes the whole replication process. The frequency of the execution of this algorithm can be tuned for each node, for instance according to the resources available at the node.

3 Apollonian P2P Networks

3.1 Apollonian Networks

2-dimensional Apollonian Networks (ANs) [1] can be produced as illustrated on figure 1: the initial network is reduced to a triangle and at each generation step, a node is added into each triangle and connected to the 3 nodes that compose the triangle. More precisely, we call these networks Deterministic ANs (DANs)

Algorithm 1: Replication algorithm performed periodically on each node n.

Data: k : number of data submited per node for replication
while *true* **do**
  `// Performs local exploration for computating the score`
  `// ` S_d `for all data` d `in` $C_n \cup E_n \cup L_n$
  $\overline{S} \leftarrow \frac{\sum_{d \in C_n} S_d}{|C_n|}$
  forall $d \in L_n$ **do**
    if $S_d < \overline{S}$ **then**
      $C_n \leftarrow C_n \cup \{d\}$
      $C_n \leftarrow C_n \setminus \{d_{max} \in C_n / \forall d_x \in C_n, S_{d_{max}} \geq S_{d_x}\}$
  end
  $L_n \leftarrow \emptyset$
  $[d_1..d_k] \leftarrow k$ pieces of data that have the lowest score in $C_n \cup E_n$
  $[m_1..m_k] \leftarrow k$ nodes in the network randomly contacted
  forall $i \in [1..k]$ **do**
    $L_{m_i} \leftarrow L_{m_i} \cup (d_i, A_n)$
  end
  `// Waits for the next iteration`
end

because of their building algorithm. Random Apollonian Networks (RANs) have the same properties that DANs (scale-free, small-world, Euclidean and space filling) [19] but are slightly different in their construction. 2-dimensional RANs are incrementaly built by inserting at each step a node in a triangle chosen at random. This node is then connected to the three other nodes of the triangle, an example of such network obtained is shown on figure 1(d). Both deterministic and random ANs have also been studied in higher dimension [17, 18] by replacing triangles with simplexes.

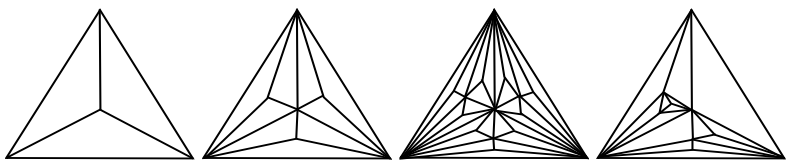

Fig. 1. Three 2D DANs with one (a), two (b) and three (c) generations of nodes, and a 2D RAN (d).

In a fully distributed P2P environment, peers do not have a global knowledge, it is therefore impossible to distributively build a DAN. However it is possible in

such environment to build a RAN by contacting a random node and obtaining a random simplex from this node, that is why we focus our study on RANs.

3.2 Efficient exploration in Apollonian P2P Networks

We propose an efficient way to perform not redundant exploration in P2P RANs [2]. While our approach needs a few more hops than flooding or Lightflood for the same coverage, it does not generate redundant messages, and do not use cache on nodes to avoid the redundancy. Furthermore, it guarantees completeness as long as the overlay contains only one connected component. Contrary to flooding (or LightFlood [8]) filling trees on RANs are exhaustive and do not need a cache on nodes to store previous seen queries, as there is no discarding process.

Principle We assume the network is stable during the flooding process. We design walkers able to incrementally build a spanning tree to explore a network area. They keep the sequence of the visited nodes, recording a kind-of Ariadne sequence composed with successive visited nodes. This sequence ensures that a node is visited once and only once. On a planar graph, this sequence builds incrementally an uncrossable fence for the walker. When encountering this limit, the walker splits, and each new walker inherits the previously recorded path. The sequences of paths taken by the walkers is a tree that fills the neighborhood of the source node P_s, ensuring that all nodes are scanned at least and at most one time.

When the request carried out by the walker is fulfilled, it returns to the initial node using the inverse path it has recorded. The walker terminates if all neighbors have already been visited or if its TTL (Time To Live) reaches 0.

Cloning mechanism Let $N(n)$ be the 1-hop neighborhood of node n. When a walker visits n, it clusters the unvisited nodes of $N(n)$. Thanks to the triangular mesh, all nodes in $N(n)$ describe a ring. By removing from this ring already visited nodes and their connections, there are between 0 and $x = \lfloor \frac{|N(n)|}{2} \rfloor$ remaining connected components. A clone of the walker is spawned inside each of these remaining connected component to fullfill the exploration.

A node is selected for propagation in each connected component according to different heuristics, for instance the node with the highest or smallest neighborhood, a random node, etc.

4 Experiments

We evaluate in this section several exploration strategies for estimating the density of replica: flooding, biased random walk (walkers cannot visit twice the same node) and filling trees. For this last exploration strategy, we study different heuristics for propagation. All simulated networks contains 10000 nodes.

Flooding and biased random-walk are performed on a random graph while filling trees are used on a random apollonian network.

In order to deal with heterogeneity among node resources [13], we attribute to each node a capacity level according to the empirical distribution given in figure 2. This capacity level is used to compute the maximum degree and amount of data nodes can store. In order to simulate heterogeneity among data popularity, we give each piece of data a popularity level according to a uniform distribution. Nodes then receive an amount of data proportionnal to their capacity level, data pieces being chosen with a biased probability proportional to its popularity.

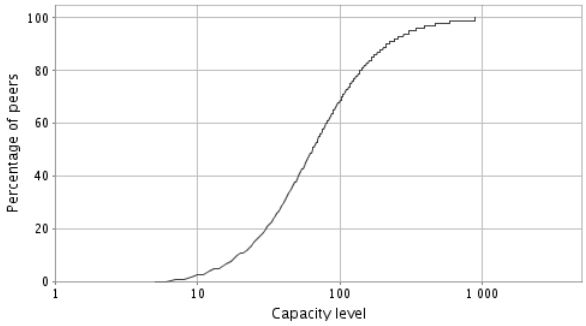

Fig. 2. Cumulative distribution of peers capacity.

Each time a node enters the network, its cache is freed. The size of the node cache is equal to the amount of data the node received at the begining of the experiment. We can then tune the average number of replica for each data by altering data diversity.

We use the standard deviation estimation to measure the uniformity of the replication. More precisely, we use a relative standard deviation in order to compare the different approaches we evaluate.

4.1 Static network

In this experiment, the starting average replication rate is 0.6% (the target replication rate is 1.2%). We set the TTL for the different approaches so that we obtain approximatively the same network coverage, near 20%. Results are shown in the table 1. At each step, nodes submit $k = 2$ pieces of data having the lowest score for replication. This allows the algorithm to converge faster than when k is set to 1, while preventing nodes copying many pieces of data simultaneously.

Figure 3 shows the evolution of relative standard deviation for replica quantity. Random walk and filling trees with either a random or smaller degree neighbor selection ffer the best performances. Results obtained with flooding are acceptable while filling trees with the selection of the neighbor that has the

Strategy	TTL	Network coverage
Flooding	4	19, 65%
Random walk	2000	19.98%
FT-2hopNeighboor	13	18.07%
FT-random	120	19.86%
FT-smallestNeighborhood	290	20.67%

Table 1. Network coverage for different exploration strategies.

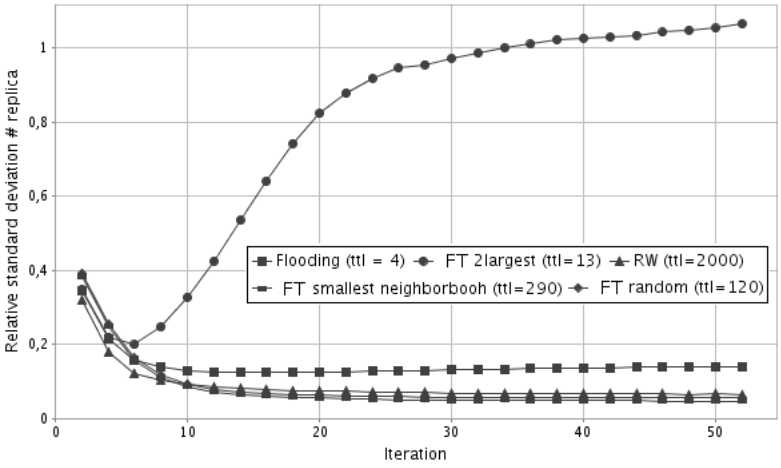

Fig. 3. Evolution of relative standard deviation for replica quantity.

highest degree offers very poor performances: while the relative standard deviation decreases in the few first steps, it then highly increases. We believe this phenomenon is because nodes with the lowest degree are never visited and some data may have a lot of replicas located on these nodes.

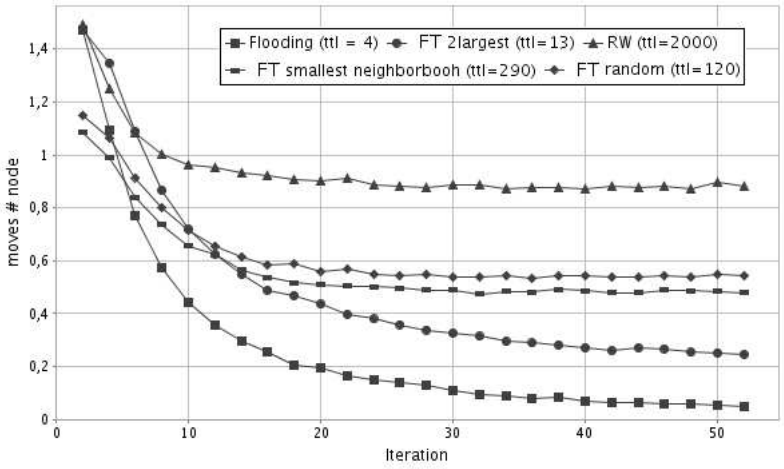

Fig. 4. Number of replicas created per nodes each step.

Figure 4 shows the average number of replicas created on each node at each step. We can see that the algorithm converges on a static network with a flooding exploration while there is no convergence at all with random walk. Results obtained with filling trees are between random walk and flooding. We believe these results are due to the randomness of the exploration strategy chosen: filing trees with the selection of the higest degree neighbor features less randomness that the selection of the smallest degree neighbor.

It seems that it is very important to visit the same nodes during each exploration for the algorithm to converge on a static network, and performances could be improved for filling trees. However we demonstrate in the next experiment this criterion is not as important in a dynamic environment.

4.2 Impact of the density estimation

We can see on Figure 4 that the amount of created replica becomes steady within 20 steps. We measure the efficiency of uniform replication by estimating replica density while running our replication algorithm for 20 steps. Each 20 run, the average amount of data replica is updated so that we can evaluate the average number of replicas seen per exploration. We compare in this study flooding (best convergence on a static network) and filling trees with the selection of the smallest degree neighbor (most uniform replication on a static network).

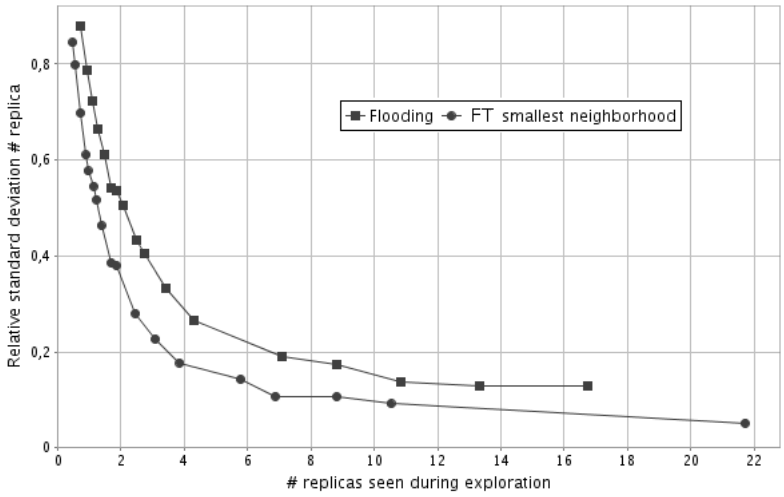

Fig. 5. Relative statard deviation within 20 steps of replication algorithm according to the average number of node having a replica seen during exploration.

Figure 5 shows the relative standard deviation obtained after 20 iterations. When the number of visited node is too small or when the data is loosely replicated, the number of replicas encountered during the exploration is too small and the algorithm does not converge, and even worse, it diverges (the initial relative standard deviation is bound between 0.3 and 0.4 as illustrated on figure 3).

Using flooding exploration, at least 7 replicas are needed to obtain a relative standard deviation bellow 0.2 whereas an exploration with filling tree with the choice of the lowest degree neighbor for propagation requires only 4 replicas. Two hypothesis could explain this difference. The first one being the partial randomness of filling tree exploration whereas flooding is fully deterministic. The second one could be the heterogeneity of the size of the 4-hop neighborhood in a random topology: a more homogoneous exploration space size could lead to better performances.

Moreover, this experiment shows that uniform replication with density estimation is adpated to environment featuring high replication rates.

4.3 Dynamic network

We have simulated a dynamic network by removing and adding new nodes at each iteration step, so that the size of the network remains on average constant in time. When new nodes are added, they receive an amount of data proportional to their capacity, data being taken at random as described at the begining of this section. Caches of newly added nodes are empty and their sizes are equal to

the number of data the node has. The replication algorithm is performed when 10% of nodes have been renewed.

Parameters in this experiment are as follows: the initial replication rate is 0.6% (the target replication rate is 1.2%). Each node submits only one ($k = 1$) piece of data for replication to another node at each replication step: this reduces the number of replicas created at each step but the algorithm takes more time to converge.

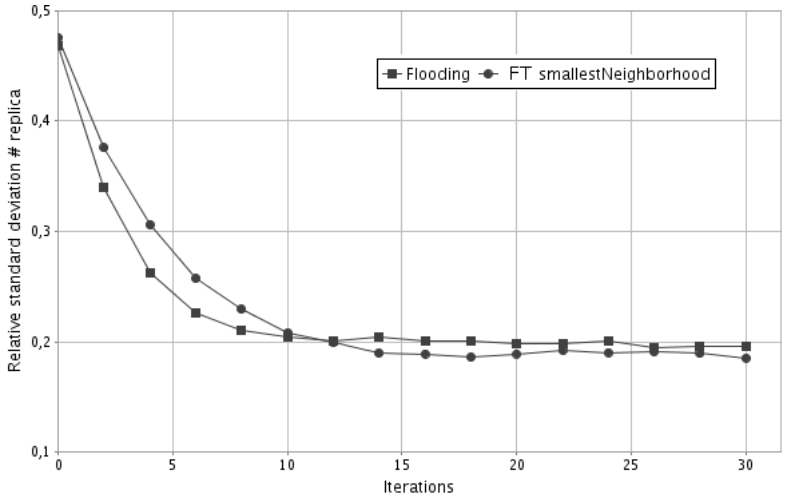

Fig. 6. Relative standard deviation of replicas amount.

Figure 6 shows the evolution of the relative standard deviation of replicas amount. At the begining the amount of data adapts to the network, and then there is a steady state with an average number of replica that remains nearly constant. An exploration with flooding allows to go a little bit faster to the steady state and both exploration schemes produce a replication similarly uniform. The relative standard deviation is higher comparatively to a static network mainly because new added nodes have empty caches.

Figure 7 shows the average number of replicas created per node at each iteration step. We can see that the copy rate is lower with an exploration strategy using filling trees comparatively to flooding. Performing the replication algorithm when 10% of nodes have been renewed could seem to be important but for the record, half of the nodes are renewed within one hours in operationnal P2P networks [13]. We have not tried a configuration for which the algorithm is performed when 50% of nodes have been renewed, since we believe that high delay between two replication steps could lead to data loss.

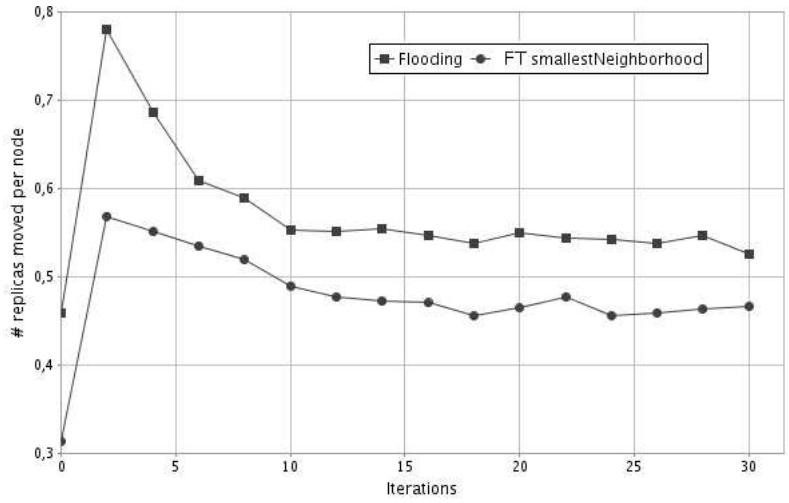

Fig. 7. Average number of replicas created per node each iteration step.

5 Conclusion and future work

We have presented a uniform replication strategy based on a local density estimation scheme. This approach is mainly designed for unstructured P2P architectures supporting complex query languages with high rejection capability (ie many requests cannot be fulfilled).

The local replica density estimation is performed by an efficient local exploration: we proposed a random apollonian P2P small world and scale free architecture that pairs with a non redundant exploration scheme and that does not use any cache on nodes. This architecture manages the heterogeneity among peers capacities and deals very well in transient or dynamic environments.

We have shown that random walk is not well suited for density estimation, while flooding produces quite good results - but also introduces a lot of redundant messages. We deduced from our experiments that two exploration strategies are best adapted to the replication scheme: LightFlood on Random P2P networks and filling trees on random apollonian networks: experimental results show similar performances with both approaches.

While apollonian networks do not use node caching to eliminate redundancy, the LightFlood overlay is less costly to maintain. Filling trees offer several heuristics for query propagation. The smallest neighbourhood heuristic helps the exploration space size remain constant given a fixed TTL : this seems to reduce significantly network traffic in dynamic environment.

This study is still preliminary: data size is uniform and the replica coherency problem remains to be addressed. Moreover, candidate nodes to the reception of data replica are randomly chosen, which could be improved by the use of heuristics.

References

1. J. Andrade, J.H. Herrmann, R.F.S Andrade, and L.R. da Silva. Apollonian networks: Simultaneously scale-free, small world, euclidean, space filling, and with matching graphs. *Phys. Rev. Lett.*, 94(1):018702, Jan 2005.
2. N. Bonnel, G. Ménier, and P.-F. Marteau. Search in p2p triangular mesh by space filling trees. In *16th IEEE Int. Conf. on Networks (ICON'08)*, New Delhi, India, December 2008. IEEE.
3. D. Chamberlin. Xquery: An xml query language. *IBM Syst. J.*, 41(4):597–615, 2002.
4. I. Clarke, O. Sandberg, B. Wiley, and T.W. Hong. Freenet: A distributed anonymous information storage and retrieval system. *Lecture Notes in Computer Science*, 2009, 2001.
5. Clip2. The gnutella protocol specification v0.4, 2002.
6. E. Cohen and S. Shenker. Replication strategies in unstructured peer-to-peer networks, aug 2002.
7. H.V. Jagadish, B.C. Ooi, and Q.H. Vu. Baton: A balanced tree structure for peer-to-peer networks. In *In VLDB*, pages 661–672, 2005.
8. S. Jiang, L. Guo, X. Zhang, and H. Wang. Lightflood: Minimizing redundant messages and maximizing scope of peer-to-peer search. *IEEE Transactions on Parallel and Distributed Systems*, 19(5):601–614, 2008.
9. J. Liang, R. Kumar, and K. Ross. The kazaa overlay: A measurement study. 2004.
10. P. Maymounkov and D. Mazieres. Kademlia: A peer-to-peer information system based on the xor metric, 2002.
11. S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content addressable network. Technical Report TR-00-010, Berkeley, CA, 2000.
12. J. Robie, M.F. Fernández, S. Boag, D. Chamberlin, A. Berglund, M. Kay, and J. Siméon. XML path language (XPath) 2.0. W3C proposed reccommendation, W3C, nov 2006. http://www.w3.org/TR/2006/PR-xpath20-20061121/.
13. S. Saroiu, K. Gummadi, and S. Gribble. Measuring and analyzing the characteristics of napster and gnutella hosts, 2003.
14. E. Sit, A. Haeberlen, F. Dabek, B. Chun, H. Weatherspoon, R. Morris, M. Kaashoek, and J. Kubiatowicz. Proactive replication for data durability, 2006.
15. I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 149–160, New York, NY, USA, 2001. ACM.
16. W.W. Terpstra, J. Kangasharju, C. Leng, and A.P. Buchmann. Bubblestorm: resilient, probabilistic, and exhaustive peer-to-peer search. In *SIGCOMM '07: Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 49–60, New York, NY, USA, 2007. ACM.
17. Z. Zhang, F. Comellas, G. Fertin, and L. Rong. High dimensional apollonian networks, 2005.
18. Z. Zhang, L. Rong, and F. Comellas. High dimensional random apollonian networks, 2005.
19. T. Zhou, G. Yan, P.-L. Zhou, Z.-Q. Fu, and B.-H. Wang. Random apollonian networks, 2004.