LightFlood: an Efficient Flooding Scheme for File Search in Unstructured Peer-to-Peer Systems [*]

Song Jiang, Lei Guo and Xiaodong Zhang
Department of Computer Science
College of William and Mary
Williamsburg, VA 23187, USA
{sjiang, lguo, zhang}@cs.wm.edu

Abstract

"Flooding" is a fundamental operation in unstructured Peer-to-Peer (P2P) file sharing systems, such as Gnutella. Although it is effective in content search, flooding is very inefficient because it results in a great amount of redundant messages. Our study shows that more than 70% of the generated messages are redundant for a flooding with a TTL of 7 in a moderately connected network. Existing efforts to address this problem have been focused on limiting the use of flooding operations. In this paper, we propose LightFlood, an efficient flooding scheme, with the objective of minimizing the number of redundant messages and retaining the same message propagating scope as that of standard flooding. By constructing a tree-like sub-overlay within the existing P2P overlay called FloodNet, the flooding operation in LightFlood is divided into two stages. In the first stage, a message is propogated by using the standard flooding scheme with three or four TTL hops, through which the message can be spread to a sufficiently large scope with a small number of redundant messages. In the second stage, the message propagating is only conducted across the FloodNet, significantly reducing the number of redundant messages.

Our analysis and simulation results show that the LightFlood scheme provides a low overhead broadcasting facility that can be effectively used in P2P searching. Compared with standard flooding used in Gnutella, we show that the LightFlood scheme with an additional 2 to 3 hops can reduce up to more than 69% of flooding messages, and retain the same flooding scope.

1 Introduction

A Peer-to-Peer (P2P) file sharing system is built as an overlay on the existing Internet infrastructure to provide file sharing service to a highly transient population of users (peers). Early systems, such as Napster, use a central server (more precisely, a server cluster) to store indices of participating peers. This centralized design and practice arouse the concerns of performance bottleneck and single point of failure. Researchers and practitioners have studied decentralized approaches in order to provide a scalable file sharing service. Instead of maintaining a huge index in a centralized system, a decentralized system such as Gnutella distributes all searching and locating loads across all the participating peers. Though the decentralized approach addresses the overloading and reliability issues, and is promising to build a highly scalable P2P system, its success is heavily dependent on an efficient mechanism to broadcast messages across a large population of peers. Reaching out to a large scope of peers is a fundamental procedure in an unstructured ad hoc P2P network, because there are no controls and no accurate information on network topologies and locations of desired files. Thus, the system scalability is directly affected by the efficiency of the broadcasting mechanism.

The major existing mechanism for message broadcasting is flooding, in which a peer sends a message to its neighbors, which in turn forward the message to all their neighbors except the message sender. Each message has an unique message ID. A message received by a peer that has the same message ID as the one received previously will be discarded as a redundant message. Flooding is conducted in a hop by hop fashion counted by Time-to-Live (TTL). A message starts off with its initial TTL, which is decreased by one when it travels across one hop. A message comes to its end either when it becomes a redundant message or when its TTL is decreased to 0. Comparatively, during the lifetime of a message, we call the sequence of initial hops of a message's path as *low hops*, and the rest of its hops, i.e. the sequence of final hops of its path, *high hops*. We call this flooding procedure, widely used in P2P systems like Gnutella, *pure flooding*, in the rest of the paper in order to set apart from the flooding scheme we proposed.

Flooding has the following merits: (1) modest latency (or response time), (2) large coverage, and (3) high reliability. A measurement-based study conducted in 2000 and 2001 shows that 95% of peers in Gnutella system could be reached within 7 hops (TTL=7) by pure flooding [2]. This is because more and more peers join to route the message in parallel while the flooding is going on, and the number of peers reached is increased almost exponentially until most peers are covered. Departure or failure of individual peers

[*]This work is supported in part by the U.S. National Science Foundation under grants CCR-0098055 and ACI-0129883.

can hardly have a disruptive impact on the system ability to transmit messages in flooding, because all possible routes in the specified neighborhood are utilized simultaneously. For these merits, flooding is used widely in unstructured P2P systems. However, with the increasing popularity of P2P systems and the rapidly expanding system scales, a serious problem of flooding emerges due to the excessive traffic overheads caused by a large number of redundant message forwardings, particularly in a system with a high connectivity topology. When multiple messages with the same message ID are sent to a peer by its multiple neighbors, all but the first messages are redundant, increasing the bandwdith consumption and peer processing burden without enlarging the propagating scope. It was estimated that the total traffic on a Gnutella system of 50,000 nodes, where flooding search is used, accounted for about 1.7 percent of the total traffic over the U.S. Internet backbone in December 2000 [2]. Considering this volume is only the amount of messages for file search and does not include file transfer traffic, which is out of the Gnutella overlay, flooding search becomes a bottleneck for the scalability of unstructured P2P systems.

Realizing the important role of flooding in ad hoc P2P systems and its problem, we propose an efficient flooding scheme, called *LightFlood*. LightFlood mostly retains the merits of pure flooding. Meanwhile, it can eliminate most of the redundant messages caused by pure flooding, thus greatly enhance the scalability of Gnutella style P2P systems. The design of LightFlood is motivated by an observation that in a pure flooding most redundant messages are generated when messages are flooded within high hops, while the flooding coverage increases in a high rate within low hops. We select a set of links in a P2P overlay to form a sub-overlay, which we call *FloodNet*, to connect all the participating peers. FloodNet is a tree-like network that uses the least number of existing P2P links to organize peers into a small number of low diameter clusters. We let messages on their low hops be flooded by pure flooding in the P2P overlay, then let messages on their high hops be flooded in the FloodNet sub-overlay. The initial pure flooding ensures that a considerable amount of message copies are dispersed across the P2P overlay with a small number of redundant messages. The next stage of flooding in FloodNet ensures that most of redundant messages caused by pure flooding within the rest of hops are eliminated. The integration of these two stages retains the advantages of pure flooding on low latency, high coverage, and high reliability.

2 Related Work

To address the flooding problem in Gnutella-like P2P systems, researchers and practitioners have proposed many solutions, which can be categorized into three types: (1) adaptive flooding, (2) locality-based flooding reduction, and (3) partially centralized location service.

Unlike pure flooding, which always starts with a fixed TTL from a peer to reach its neighborhood within its radius, adaptive flooding takes more dynamic factors into consideration to reduce the flooding range while maintaining the necessary search quality. For example, in the expanding ring [1] (or iterative deepening [6]), several successive flooding searches are initiated with increasing TTLs until enough responses are received. Though the method might be effective for searching popular files, its performance could be uncertain for less popular files due to the repeated use of floodings. Directed BFS [6] sends query to a neighbor satisfying a specific criterion based on some heuristics, which performs a flooding with the original TTL decremented by one, in order to save the search cost and obtain enough qualified results. However these adaptive flooding algorithms still need to keep flooding as a major component. Regardless of the fluctuating system factors such as file distribution and local connecting conditions, Our LightFlood promises to directly reduce the overhead of flooding. Thus, adaptive floodings can be more effective by integrating LightFlood into themselves.

To reduce the use of flooding and improve search efficiency, interest-based locality [4] enables a peer to create shortcut links with those peers serving it qualified results previously, based on the heuristic that if peer A has a particular piece of content that peer B is interested in, then it is likely that A will have other pieces of content that B is also interested in. In this scheme, common interests are detected through flooding. And flooding on the original overlay is used whenever a searching through shortcut links fails. So a low-cost flooding is also essential to achieve its low overhead goal.

The third prevalent solution adopts super-peers to provide a partially centralized location service [5], like Morpheus and current Gnutella implementation. A super-peer is a node that acts as a centralized server to a subset of clients. It maintains the indices of its client peers and conducts searching and locating on behalf of its clients among super-peers. These super-peers connect to each other forming a pure Gnutella style network. With the expanding scale of the P2P systems, the inefficiency of flooding in super-peer networks remains a grave concern to be addressed.

In summary, lightweight broadcasting is a core technique to improve the efficiency of searching in an ad hoc P2P system. Many schemes aiming at system scalability are expected to benefit from the technique by integrating it into these schemes.

3 Flooding Versus Hops

Recall that flooding is conducted in a hop by hop fashion. With the increment of hops, more and more new peers are reached, and more and more forwarded messages are generated, a large amount of which are redundant messages. In this section we study the regularity of the changes of coverage and the number of redundant messages with the increment of message hops.

We use Gnutella topologies collected during the first six months of 2001 [9] to simulate the flooding behavior. The connectivity degree of these topologies follow a two-segment power-law distribution (see [2] for details), in which we selected three topology traces (see Table 1) to cover a variety of topology sizes and connectivity degrees. In our simulation experiments, we sent a query from each peer in

Topology	Original Name	Average Degree	Number of Peers
T1	graph052701104502.xml	3.40	42822
T2	graph050301100618.xml	4.72	28895
T3	graph183011126.xml	5.43	21781

Table 1. Clip2 topology traces used in our simulation experiments. Original name refers to the trace file name used in [9]

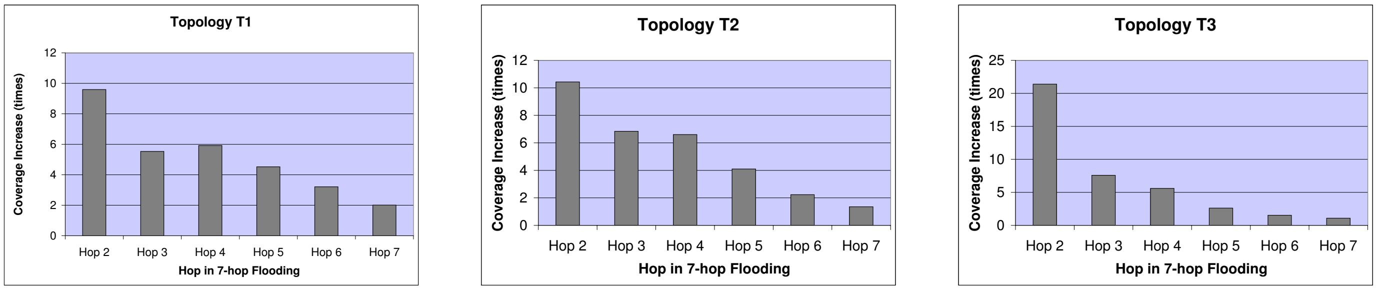


Figure 1. Coverage growth rate of message floodings. The bar with "Hop i" (i=2, 3, ..., 7) represents the ratio of coverages between the initial i hops and $i-1$ hops.

the network with TTL of 7, which is the default value in Gnutella system. Then for each flooding hop, h, of a query, we collected the number of new peers reached, P_h, and the number of forwarded messages generated, M_h. It is noted that it takes at least N messages to reach N new peers. So the redundant message on hop h is $M_h - P_h$. We averaged the P_h and $M_h - P_h$ over all the queries sending from each peer in the topologies for each flooding hop.

To observe the growth rate of message flooding coverage (simplified as coverage in the rest of the paper), we list $\sum_{i=1}^{h} P_i / \sum_{i=1}^{h-1} P_i$ for $h = 2, 3, ..., 7$ in Figure 1 for each topology. From the figure we observed that the coverage growth rate reduces quickly with the increment of hops. The rates for the first several hops are apparently larger than those in the following hops. If every forwarded message had reached a new peer, the coverage would have grown exponentially with the number of hops. However, these forwarded messages are possible to arrive at the peers that have seen a message with the same ID, thus do not contribute to the growth of coverage. Such a possibility would quickly increase when a considerable number of peers have seen the message at the stage of broadcasting within high hops. In contrast, the possibility is much smaller at the stage of broadcasting within low hops. That is why we observed more effective flooding within low hops than that within high hops. It is also observed that a high average connectivity degree in a topology widens the gap between the coverage growth rate within low hops and that within high hops. However, topologies with small average connectivity degrees like topology T1 still have significantly high coverage increase within low hops. This is because there exist some peers with very large connectivity degrees in Gnutella networks, which follow power-law degree distributions, and the chance to reach such peers within low hops is large, boosting the message coverage.

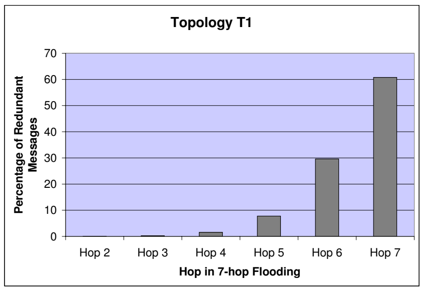
To observe the redundant message distribution across the hops, we listed the percentage of the redundant messages on a specific hop over all the redundant messages within 7 hops, $(M_h - P_h)/\sum_{i=1}^{7}(M_i - P_i)$ for $h = 2, 3, ..., 7$, in Figure 2. Apparently the figure shows that the redundant messages generated within the initial 4 hops of floodings are much less than those within the high hops. For example, the number of redundant messages within 2nd, 3rd, and 4th hop combined is only 1.9%, 2.9%, and 10.7% of all 7-hop redundant messages in topologies T1, T2, and T3, respectively, while most of redundant messages are generated on the last several hops. This is because widely dispersed message copies across the overlay within high hops generate increasingly more redundant messages.

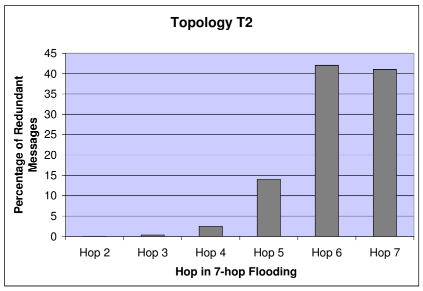
Considering the large coverage growth rate and small overhead in terms of redundant messages within low hops, we found that flooding is only efficient at this stage. The fact that a considerable number of peers that have received the message have been widely dispersed across the network motivates us to conduct flooding on high hops across only part of the links. This subset of links forms the FloodNet that we deliberately maintain for this purpose, rather than flood within all the links by the pure flooding.

4 Description of the LightFlood Scheme

4.1 FloodNet: a Tree-like Sub-overlay

If we have a spanning tree connecting all peers on the existing P2P overlay network, all redundant messages can be avoided. However, broadcasting only along the spanning tree





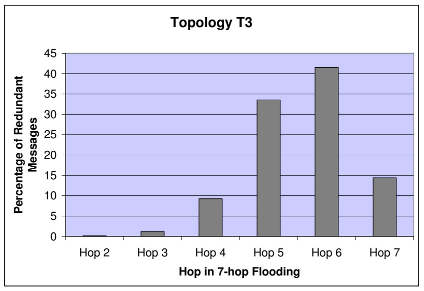


Figure 2. Redundant message distribution. The bar with "Hop i" (i=2, 3, ..., 7) represents the percentage of redundant messages generated on the ith hop in all redundant messages generated in the 7-hop floodings.

of a well-connected network like Gnutella is not desirable because of the concerns of greatly prolonged latency and weakened reliability, though the cost could be minimized. Our simulation has shown that on a typical Gnutella topology in [9] where it takes a pure flooding 7 hops to reach 95% of nodes, it takes more than 30 hops for a flooding to have the same coverage on a randomly constructed spanning tree over the topology! Further, a link or node failure in the tree could disrupt a large portion of networks. However, if a broadcast on the tree is initiated from a large number of nodes simultaneously, the restraints imposed by the tree structure are removed. We have shown that pure flooding within low hops could cover a considerable number of nodes with a small number of redundant messages. These nodes can work as initiators of a tree flooding, keeping the advantages of the pure flooding while reducing the flooding cost.

There are several principles in constructing a tree-like sub-overlay for P2P networks. (1) Because the system is designed to be fully autonomous and highly dynamic, only local information is cheaply available and can be used for construction; (2) To increase the coverage of a flooding with a given TTL, the topology diameter should be low; (3) Because of high transiency of the system, it must be efficiently maintained. Following these principles, we construct the tree-like sub-overlay called *FloodNet* in this way: (1) Each peer notifies its immediate neighbors of its connectivity degree; (2) Once the degrees of all its neighbors are known, a peer selects the neighbor that has the maximum degree as its father [1], and notifies its father peer so that the peer will not again be chosen as father by its father peer. Thus there could be a peer without a father because all its neighbors have chosen it as "father". Note that the results of the construction could be multiple unconnected tree-like components, each of which has at most one circle. However that is not problematic for our purpose because (1) we start flooding on FloodNet only after several hops of pure flooding when peers possessing message copies are widely dispersed across the P2P overlay; (2) Redundant messages caused by the loops are detected and discarded on FloodNet as the pure flooding does. Our simulation has shown that the number of disconnected components is usually small, normally less than 10.

[1]If the neighbor already selects it as its father peer, the peer selects the next maximum degree peer that is not its child as its father.

FloodNet can be constructed with little cost with only local information, and its depth in each cluster is low because the links to high degree peers are utilized. Further, its maintenance cost is minimal: when a peer's immediate neighbors arrive or depart, or their degrees are changed, the peer re-evaluates the neighbor degrees, possibly selects a new father peer, and notifies the affected neighbors. Compared with the maintenance of super-node indices and interest-based locality, FloodNet can be maintained with little overhead.

4.2 LightFlood: a Class of Schemes by Combining of Pure Flooding and FloodNet Flooding

Facilitated with FloodNet, we present our LightFlood broadcasting scheme as follows: A message is flooded for the initial several hops across the original P2P overlay, then for the rest of hops, it is flooded across FloodNet; that is, a received message is only forwarded to the peer's child/father peers along the links in the FloodNet once its TTL drops to a certain value. If a LightFlood policy specifies that a message floods over original overlay for its first M hops and then continues its flooding for the next N hops over FloodNet, we call it (M, N) policy. Specifically, a pure flooding with a given TTL, *ttl*, can be regarded as (*ttl*, 0). We also denote the class of policies with M hops pure flooding at first, and any number of following hops of FloodNet flooding as $(M, *)$.

5 Performance Evaluation

The overhead of flooding can be quantified by the number of its generated redundant messages that are discarded once detected. In the ideal case where all redundant messages are eliminated, it takes only N messages to reach N peers from a message initiator. There are several questions we are particularly interested in:

1. With a given TTL, how does the increase of M affect the performance of pure flooding?

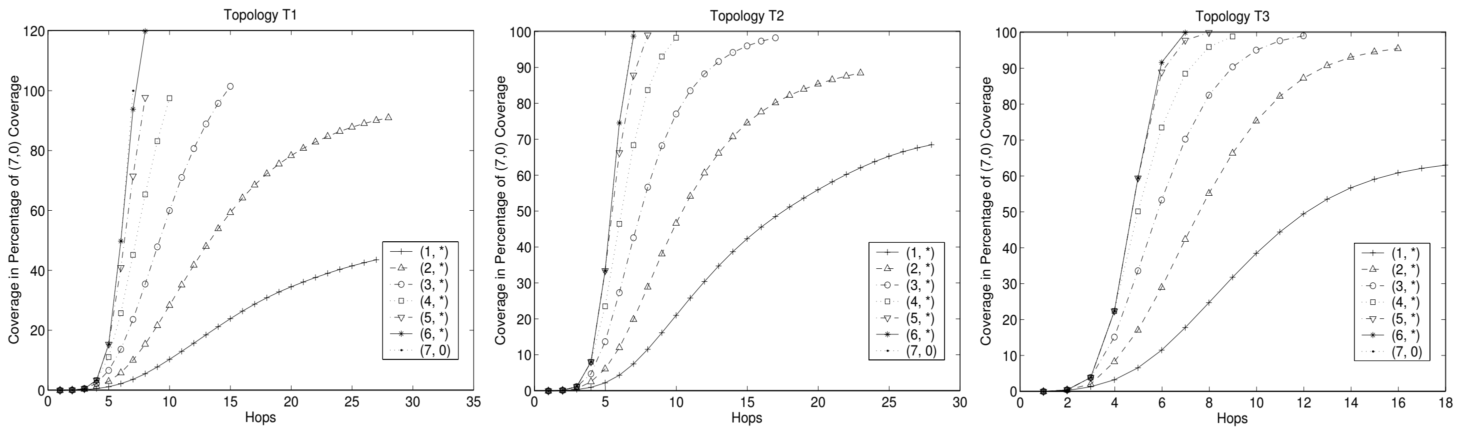


Figure 3. The coverages of various flooding policies in the three Gnutella topologies. Note (7,0) represents the 7-hop pure flooding. All the coverages are normalized to the (7,0) coverage. There are two conditions for a flooding to complete: (1) Its coverage exceeds 97% of the coverage of (7,0); or (2) the flooding has experienced 7 hops and the growth of the coverage within one hop is less than 1% of the coverage of (7,0).

2. How does the connectivity of a topology affect the performance of pure flooding and LightFlood?

3. With a coverage comparable to that of a pure flooding, how much is the reduction of redundant messages of LightFlood, and how many additional hops does LightFlood have to travel to achieve the comparable coverage?

4. How does the departure or failure of peers affect the performance of pure flooding and LightFlood?

5. How well do the existing flooding improvement techniques, such as expanding ring, benefit from LightFlood being integrated in these schemes?

We run a series of simulations to answer the above questions. We use the three topology graphs listed in Section 3. For each experiment, we broadcasted a message from each peer in a topology. Statistics are collected from the broadcastings. We report their average values in the following subsections.

5.1 How does M of LightFlood (M, N) Affect its Latency for a Given Coverage?

We normalized the coverage (the number of peers reached) of all the broadcasting policies to that of 7-hop pure flooding (7, 0) for convenience of comparisons. Figure 3 lists the coverage growths with the increase of hops for different policies in the three topologies. There are two conditions for us to stop a broadcasting: (1) Its coverage exceeds 97% of coverage of (7,0); or (2) the flooding has experienced 7 hops and the growth of coverage within one hop is less than 1% of the coverage of (7,0). From the figure, we can see that in the set of policies $(M, *)$ (M=1, 2, ..., 6), only a small M, the number of initial pure flooding hops, causes a significant prolonged TTL time to reach the similar coverage as that of (7, 0). It takes a reasonable time to reach that coverage once M has a little increase. For example, It takes 16 hops to reach the similar coverage for policy (2, *), while it takes only 9 hops for policies (4,*) in topology T3, which is very close to the 7 hops for pure flooding. Another observation is that (1, *) can be unable to reach that coverage because of the existence of multiple isolated components in FloodNet. Without dispersing messages among a sufficient number of peers using original overlay by pure flooding in the initial several hops to make the preparation for the following FloodNet flooding, $(M, *)$ with a very small M could cover only some of FloodNet components, which could seriously limit their broadcasting coverages. For example, (1, *) covers only 42% of the coverage of (7,0) in topology T1. The lack of preparation of enough peers before FloodNet flooding also hinders the speed of broadcasting, because fewer messages travel simultaneously across the FloodNet. But this only happens with very small initial pure flooding hops. Actually, as we have shown in Section 3, several hops of pure flooding can prepare a considerable number of peers with the copies of a broadcasted message with a minor number of redundant messages. For example, 4-hop pure flooding will reach 1067, 2222, and 4938 peers for the following FloodNet flooding in topologies T1, T2, and T3 respectively, while their redundant messages are only 1.85%, 2.87%, and 10.53% of those in corresponding (7,0) pure floodings. Accordingly, policies (4, *) takes only additional 3, 3, and 2 hops to reach a similar coverage like that of (7,0) in T1, T2, and T3, respectively. The third observation is that the TTL time of policy $(M, *)$ $(M > 2)$ for the given coverage in the topologies with various average degrees are close. This is because we use (7,0) coverage in each corresponding topology as a target for other policies to reach. A low average connectivity degree means a low coverage target for other policies to reach. Thus it takes similar TTL times in topologies with different connectivities.

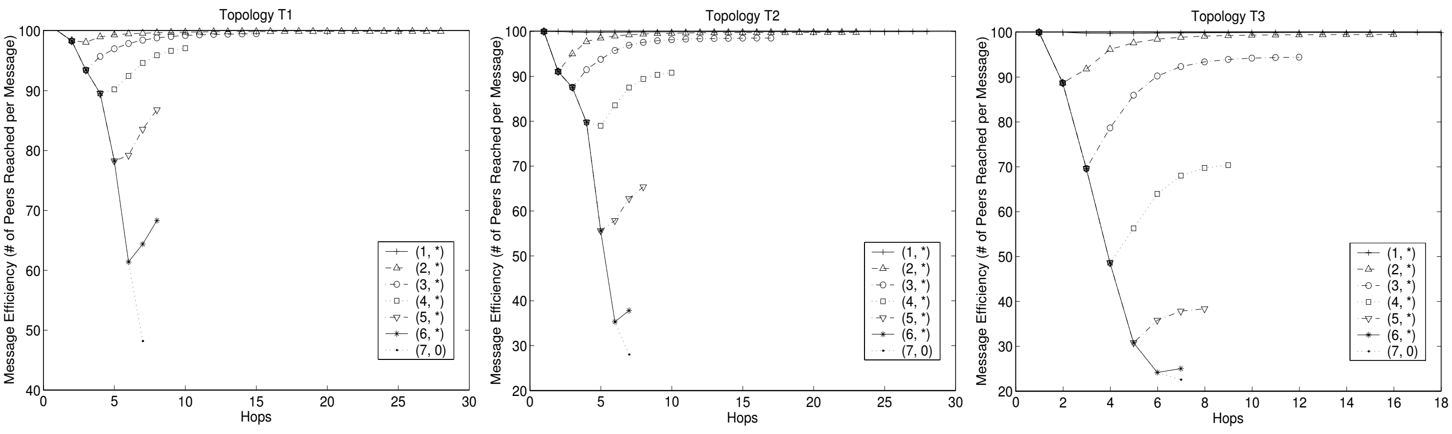


Figure 4. The message efficiencies of various floodings in the three Gnutella topologies. The flooding completion conditions are the same as described in Figure 3. The message efficiency is the ratio of the message coverage and number of forwarded messages, reflecting the overhead caused by redundant messages in a flooding. For example, with policy (4,*) in topology T1, in average, the broadcasted messages reached 1066.9 peers with 1193.7 forwarded messages within the initial 4 hops, thus the efficiency shown in the figure for (4, *) for the 4th hop is $1066.9/1193.7 \times 100\% = 89.4\%$.

5.2 How does M in LightFlood (M, N) Affect its Efficiency?

Figure 4 shows the message efficiency at different stages of floodings with various policies. The message efficiency for a specific hop, h, is defined as the ratio of the number of peers reached and the number of the messages forwarded in the initial h hops. The message efficiency for the last hop is the policy efficiency. The ideal efficiency is 1 if there are no redundant messages. In these experiments, we have the same conditions to stop a broadcasting as we did in Section 5.1. From the figure we can see that though (7,0) used least hops to reach its coverage, its message efficiency deteriorates sharply with the increase of hops used. This is consistent with what we observed in Section 3, where more redundant messages are generated with low coverage growth rate within high hops than those within low hops. But once pure floodings switch to floodings over FloodNet, the efficiency curves immediately stop dropping and start to rise. This results in a large gap of message efficiency for the final hop between (7, 0) and $(M, *)$ for $(M \leq 5)$. The smaller the value of M is, the larger of the efficiency improvement is. For example, for $M = 1, 2, 3$, their final efficiencies are close to 100%. At the same time, the increase of TTL time for the similar coverage is modest for $M \geq 3$. Generally, policies (3, *), (4, *), and (5, *) strike a good balance between efficiency and latency. For example, the efficiency of pure flooding (7,0) is improved from 48.2% to 97.1% by policy (4, 6) in topology T1, while (4,6) only increases TTL by 3. Another observation is that the efficiency becomes worse with the increase of average degree of topologies. For example, the (7,0) efficiency for T1 with an average degree 3.40 is 48.2%, while the efficiency for T2 with average degree 4.72 and T3 with average degree 5.43 are 28.1%, 22.6% respectively. The efficiencies are improved to 97.1%, 90.8%, and 70.4% by (4,6) in T1 and T2, and (4,5) in T3, respectively. This is because higher connectivities cause more redundant connections among links, thus generating more redundant messages. This makes LightFlood more attractive and more necessary for systems with high topology connectivities, which is evidenced by the 50.2%, 69.1%, and 67.9% flooding message reduction for T1, T2, and T3, respectively, when we use policy (4, *) as an example.

In summary, even with a large variety of topology connectivities, policies (3, *), (4, *), and (5, *) provide 1.7-4.2 times message efficiency improvement, which means 41.2%-76.2% flooding cost reduction, while they require a modest TTL time increase, namely 1 to 10 for a comparable coverage as that of pure flooding. Comparatively, (3, *) is beneficial to flooding efficiency in the cost of reduced coverage, while (5, *) is beneficial to flooding coverage in the cost of lowered efficiency. A low connectivity of topologies is more beneficial to flooding efficiency, while a high connectivity of topologies is more beneficial to flooding coverage. With various topology connectivities in consideration, (3,*) can be used in topologies with high connectivity like T3, while (5,*) can be used in topologies with low connectivity like T1. There is a spectrum of policies (M, *) with pure flooding (7,*) and pure tree broadcast (1,*) on the two extreme sides, respectively. According to the observation in [2], the changes of the average connectivity are small over a long period of time. A policy (M, *) with reasonably chosen M values (3, 4, and 5) apparently performs better than pure flooding in terms of search efficiency, and strikes a good balance between system-wide traffic consumption and user-perceived latency. Because LightFlood is intended to be a substitute of pure flooding and its search coverage is compared with the corresponding pure flooding coverage, the actual P2P network size does not affect the selection of M. We tested (4,*) on all 48 Gnutella topologies in [9] with their average connectivity degrees ranging from 2.37 to 6.73, and found it consistently performs well in terms of efficiency and coverage compared with (7,0). So (4, *) is an optimal chioce of the family of policies regardless of system connectivity and network size in terms of traffic and latency. We refer to (4, *) when we mention LightFlood scheme later. In the following experiments examining various aspects of flooding policies, (4,*) is used.

5.3 Message Coverage from Individual Peers

In the results we reported above, the average efficiency and coverage combined characterize the cost on the overlay, and average coverage reflects the scope that an average peer can reach, which implies its quality of service – the number of results returned. However, the specific coverage size of an individual peer could be sacrificed even though the average size is satisfactory. So people may worry that the use of FloodNet could shrink flooding coverage from certain peers, even though the harm to these peers can not be reflected in the average statistics, which could include some large coverages offsetting the shrunk coverages. If this were the case, it could discourage these users from staying in the systems.

To investigate the issue, we compared the distributions of coverages of all peers in the topologies between (4, 6) and (7, 0) in T1 and T2, (4,5) and (7,0) in T3. Figure 5 gives their Cumulative Distribution Function (CDF) curves of coverage distributions, which shows the percentage of peers from which flooding coverage is below a certain coverage in percentage of the total number of peers. ¿From the figure we can see the impact of connectivity of topologies on peer coverage distributions. The larger the average degree is, the fewer peers with small coverage will be. For example, for pure flooding (7,0), there are 15% peers whose coverages are less than 50% of total peers in topology T1, while there are almost no such peers in T2 and T3 because of their relatively high connectivity degrees. The CDF curves for (7,0) and (4,*) are very close in the three topologies. Though policies (4,*) have more low coverage peers, the differences are marginal. Thus LightFlood not only keeps the coverage of pure flooding with small additional TTL time collectively, but also performs as well as pure flooding individually.

5.4 Impact of Peers' Departure on Performance Degradation

When there are a considerable number of peers leaving the system or failing due to malicious attacks simultaneously, the coverage of a flooding message can be reduced, because each peer also serves as a router forwarding messages. In LightFlood, when a peer leaves, its child peers would select another available neighbor with the highest degree as its father to take place of the leaving peer. It has been shown that Gnutella is highly resilient in the face of random breakdown, but is highly vulnerable in the face of removal of best connected peers, which could happen in a well-orchestrated, targeted attacks [3].

To test the impact of the situation on LightFlood compared with that on pure flooding, we selected (4, 6) for topologies T1 and T2, and (4, 5) for topology T2 to show their coverage changes when randomly chosen peers are removed, and when the best connected peers are removed. The experiment results are shown in Figure 6. Our results confirmed that coverage has a graceful degradation with random removals, but the network could become mostly unconnected with removals of a small percentage of best connected peers. Though our LightFlood can not improve the worst case with removal of high degree peers because FloodNet does not build additional links among peers beyond the originally existed P2P links, it does behave almost the same as pure flooding with graceful degradation in face of random removal.

5.5 How Much Does Expanding Ring Benefit from LightFlood?

The expanding ring scheme has been shown to be an effective approach to achieve a very small stopping TTL [2] and to eliminate most of the messages in flooding when widely duplicated files are searched [1]. For example, it takes only one or two hops to find one result when the files are duplicated at over 10% of the peers. In such a case, there is no difference when expanding ring uses either pure flooding or LightFlood, because LightFlood also uses pure flooding for its initial hops. The concern with expanding ring is on its search for less popular files. Regarding this case, paper [1] has shown that the stopping TTL could be much enlarged and the number of messages used could be significantly increased. Our measurement-based study on Gnutella in [7] has shown that more than 20% of queries can only find less than 10 results in a 7-hop flooding with more than 50,000 peers in the system. For these queries requesting unpopular files, the expanding ring has to considerably increase its stopping TTLs, which could adversely deteriorate the broadcasting efficiency, thus inflict heavier burden on the systems than pure flooding does.

To investigate the benefits expanding ring could obtain by using LightFlood instead of pure flooding in its repeated broadcastings, we ran the simulations to compute the average number of forwarded messages used and stopping TTLs when the numbers of satisfactory results are 10 and 20, respectively. We set the starting TTL 1 and increase TTL by 2 each time as suggested in [1]. We use a range of duplication ratios, the one between the number of peers with results and all the peers in a topology, from 0.1% - 5%. The LightFlood policy used is (4, *). Figure 7 shows the traffic generated in both policies with various duplication ratios. We see that the traffic is extremely heavy, even exceeds the traffic of pure flooding (7,0) when duplication ratios are low in all three topologies, especially when 20 results are required, though the traffic is reduced sharply with the increase of duplication ratios. Considering the prolonged TTL time caused by repeated broadcastings, practitioners would be discouraged from implementing expanding ring due to its probably worse scenarios. However, the expanding ring scheme significantly reduces its traffic to well below (7,0) traffic in all the cases when it is built on LightFlood, especially with low duplication ratios (see Figure 7). At the same time, the increase of TTL time is trivial compared with the time spent

[2]Stopping TTL is the TTL used in the last flooding of its multiple consecutive floodings in expanding ring scheme.

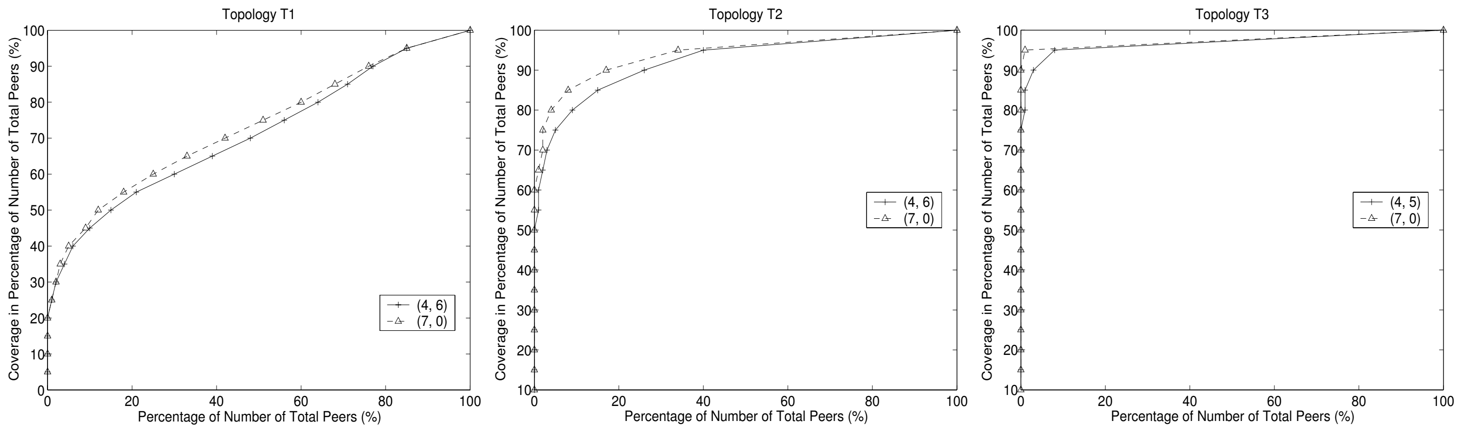


Figure 5. CDF curves of coverage distributions in selected LightFlood and pure flooding policies, which show the percentage of peers from which flooding coverage is below a certain coverage in percentage of total peers. For example, with policy (4,6) in topology T1, there are about 20% of total peers whose (4,6) coverage is less than 55% of all peers.

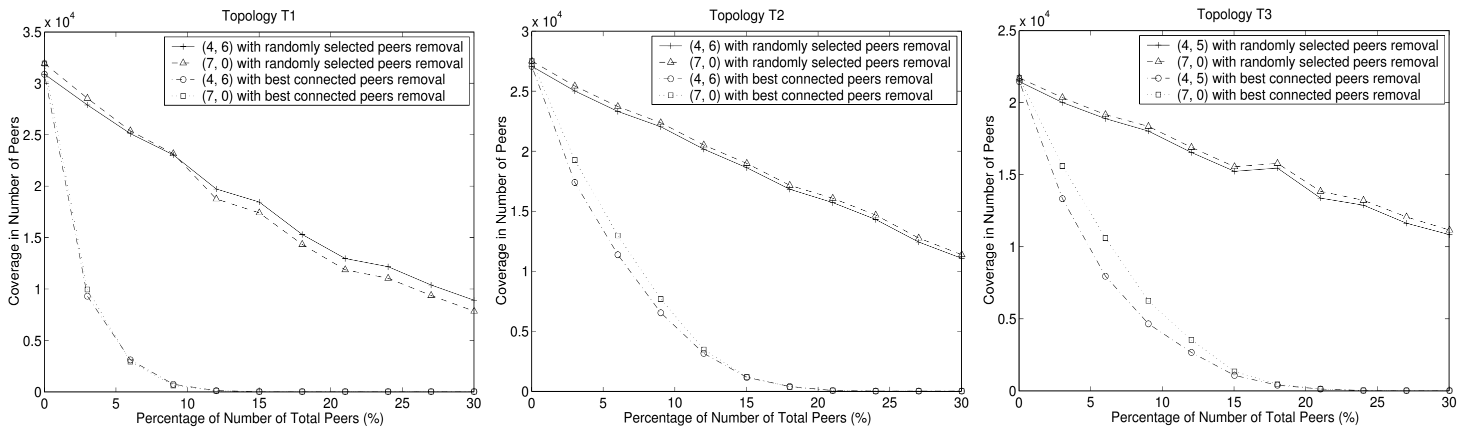


Figure 6. Changes of coverage size with the number of removed peers in percentage of total peers in the three topologies for selected LightFlood and pure flooding policies. There are two options to select removed peers: (1) randomly chosen peers or (2) the best connected peers in the topologies.

on sequential floodings and the waiting time between them in the expanding ring (see Figure 8). For example, the number of forwarded messages is reduced from 117% to 43% of (7,0) traffic by policy (4,6), while its stopping TTL is only increased from 6.9 to 8.3 on topology T2 when the number of satisfiable results is 20 and the duplication ratio is 0.1%. In summary, though expanding ring or iterative flooding are promising searching techniques to replace pure flooding, they only become practical when they are built on a low overhead flooding technique like LightFlood.

6 Conclusion

While flooding is an essential operation in an unstructured ad hoc P2P network, its overhead imposed on the underlying infrastructure significantly limits the system scalability. Our LightFlood, represented by its (4,*) policy, provides a simple scheme to perform broadcast in a cost-effective way in unstructured P2P overlays. It combines the advantage of low latency and high reliability merits of pure flooding and low traffic overhead merit of broadcasting on tree structure by using a tree-like sub-overlay, FloodNet. The construction and maintenance of FloodNet rely on only local knowledge and are of low cost. Not only it is a general solution for efficient broadcast in P2P networks, LightFlood can also greatly improve the performance of existing schemes such as expanding ring, directed BFS, super nodes, and others. We believe that the LightFlood scheme can be widely used as a core mechanism for efficiently broadcasting messages in P2P systems.

Acknowledgement: This work is a part of an independent research project sponsored by the US National Science Foundation for program directors and visiting scientists. We thank Bill Bynum for reading the paper and his comments.

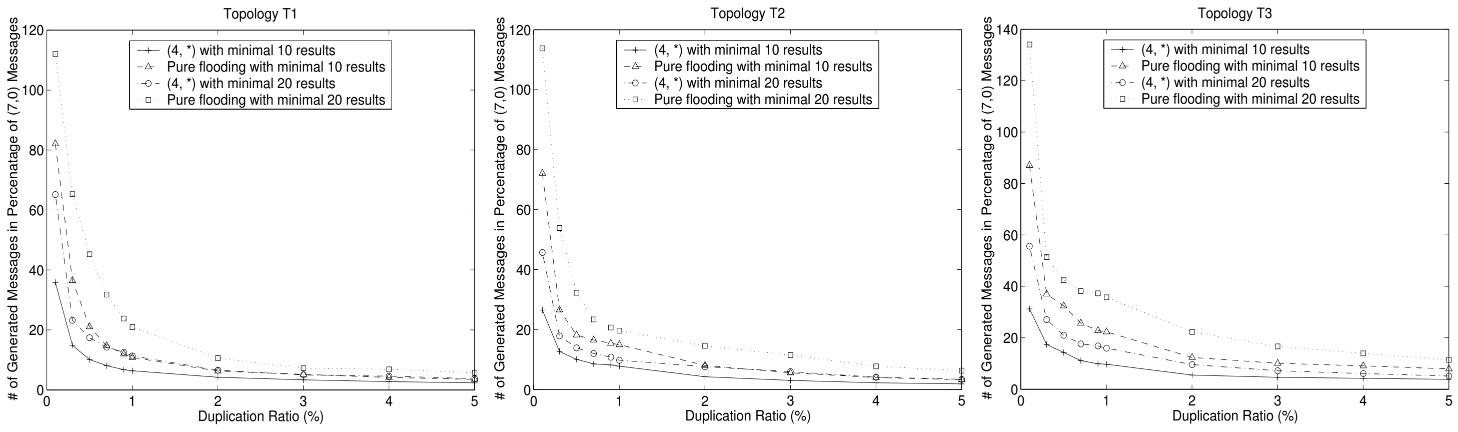


Figure 7. Traffic (number of generated messages) in percentage of (7,0) traffic used in two kinds of expanding ring (LightFlood policy (4,*) and pure flooding) for at least 10 or 20 satisfactory results with various duplication ratios.

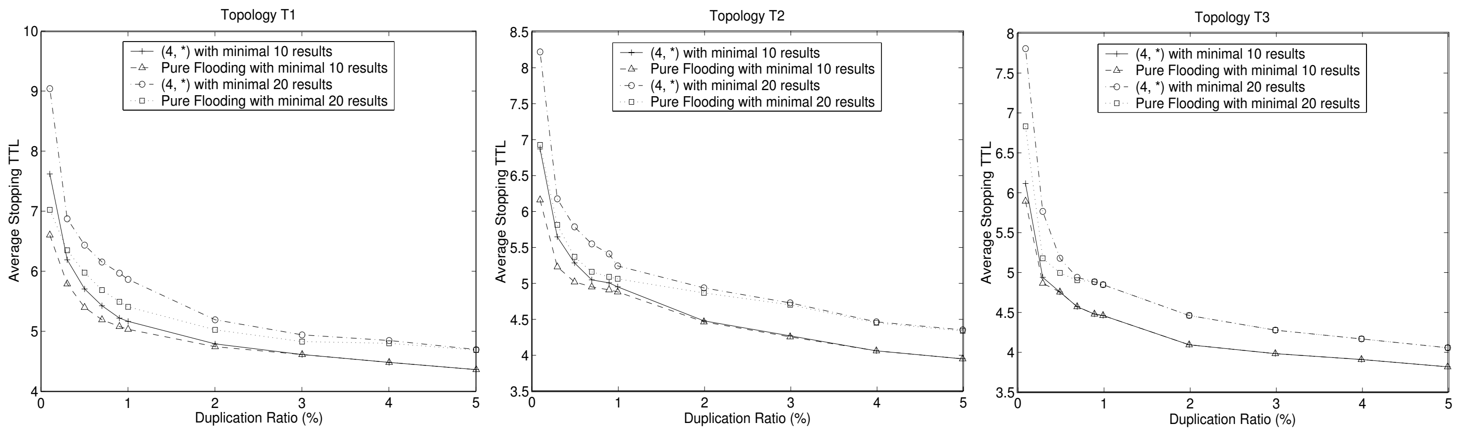


Figure 8. Stopping TTLs in two kinds of expanding ring (LightFlood policy (4,*) and pure flooding) for at least 10 or 20 satisfactory results with various duplication ratios

Comments from the annonymous referees are constructive and helpful.

References

[1] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, "Search and Replication in Unstructured Peer-to-Peer networks", *Proceedings of the 16th ACM International Conference on Supercomputing*, ACM Press, June 2002, pp. 84-95.

[2] M. Ripeanu and I. Foster, "Mapping Gnutella Network", *IEEE Internet Computing*, January/February 2002, pp. 50-57.

[3] S. Saroiu, P. Gummadi, and S. Gribble, "A Measurement Study of Peer-Peer file Sharing Systems", *Proceedings of ACM Multimedia Computing and Networking*, January 2002, pp. 156-170.

[4] K. Scipanidkulchai, B. Maggs, and H. Zhang, "Efficient Content Location Using Interest-Based Locality in Peer-to-Peer Systems", *Proceedings of IEEE INFOCOM 2003.*

[5] B. Yang, H. Garcia-Molina, "Designing a Super-peer Network", *Proceedings of the 19th International Conference on Data Engineering*, March 2003.

[6] B. Yang, H. Garcia-Molina,"Improving Search in Peer-to-Peer Systems", *Proceedings of the 22nd International Conference on Distributed Computing Systems*, July 2002, pp. 5-14.

[7] L.Guo, L. Xiao, S. Jiang, and X. Zhang, "Low Traffic and Low Latency Search Protocols in P2P Networks", Technical Report, Computer Science Department, College of William and Mary, January, 2003.

[8] http://www.limewire.com

[9] Clip2.com, "Clip2 Gnutella crawl files"