Imbuing Unstructured P2P Systems with Non-intrusive Topology Awareness

Harris Papadakis, Mema Roussopoulos, Paraskevi Fragopoulou[1] and Evangelos P. Markatos

Foundation for Research and Technology-Hellas, Institute of Computer Science
N. Plastira 100, Vassilika Vouton, GR-70013 Heraklion, Crete, Greece
{adanar, mema, fragopou, markatos}@ics.forth.gr

Abstract

The random nature of unstructured P2P overlays imbues them with enhanced self- properties. Most of the algorithms which make searching in unstructured P2P systems scalable, such as dynamic querying and 1-hop replication, rely on the random nature of the overlay to function efficiently. However, they do not take into account the structure of the underlying physical communications network, which is anything but random. Efforts to provide topology awareness to unstructured P2P systems often result to clustered graphs which affect negatively algorithms that rely on random overlays. In this paper, we propose ITA, an algorithm which creates a random overlay of randomly connected neighborhoods providing topology awareness to P2P systems, while at the same time has no negative effect on the self-* properties or the operation of the other P2P algorithms. Using extensive simulations, we demonstrate that ITA reduces communication latency by as much as 50% which is important for P2P users. Furthermore, it reduces by 20% the number of IP network messages which is critical for ISPs carrying the burden of transporting P2P traffic. Finally, ITA is shown to reduce significantly the load imposed on the routers of the IP network layer.*

***Index Terms*—Peer-to-peer, unstructured overlay network, topology awareness, self-* properties, IP network layer, communication latency.**

I. Introduction

Over the last few years P2P systems have experienced an ever expanding use, from file-sharing to content delivery systems to Grid systems. All those applications have a common denominator, which is global-scale deployment. Given the magnitude of deployment of today's P2P systems, it is certain that they cannot afford to be oblivious to the structure of the underlying physical communications network.

Most P2P systems today are based on forming an overlay network atop the underlying IP network layer. The structure and formation procedure of this overlay network is one of the most critical design choices for any P2P system and usually the main goal that drives its formation is the facilitation of the search mechanism. This means that the search algorithm of a P2P system is tightly coupled with the structure of the overlay. This is more evident in the case of structured P2P systems, where the structure of the overlay network is such as to allow for a binary-tree like search to be performed, which amounts to a *O(logN)* search cost in messages.

Unstructured P2P systems do not impose in principle a structure on the overlay, thus most of them form a random graph. This means that each peer selects its neighbors (the peers it connects to) randomly, among all available peers, without any regard to the IP distance. This leads to lack of any correlation between the distance of two peers on the IP layer and on the P2P overlay. The random nature of the graph is essential for the efficient operation of many mechanisms used in unstructured P2P systems today, such as 1-hop replication [22] and dynamic querying [3].

From the above, it is evident that each P2P system has its own agenda on the construction of its overlay. This is the reason most P2P systems in use today fail to take into any consideration the structure of the underlying network, the Internet. The result is a misuse of the IP layer from most of the P2P systems, with adverse impact on their own operation, as well as the operation of the other applications sharing the same medium. Some proposals, which we will present in the following sections, have already been made to rectify this. The main goal of these proposals is to provide a better mapping between the P2P

[1] Paraskevi Fragopoulou is with the Department of Applied Informatics and Multimedia, Technological Educational Institute of Crete, Greece.

overlay and the underlying IP network. However, we argue that this approach may affect some of the fundamental characteristics of P2P systems.

The obliviousness of P2P systems to the underlying network has two main drawbacks. The first is that the average latency between any two neighbors on the P2P overlay is increased since each peer does not actively try to connect to peers which are close-by on the IP layer. The second and most important drawback is that each P2P overlay connection contains a large number of routers. This means that each message may travel around the world many times, passing through many routers before it reaches its destination. Figure 1 illustrates such a simple scenario, where a message from peer A to peer C crosses the Atlantic twice before it reaches peer C on the same continent as peer A. Such an inefficient routing, is one of the main reasons behind the observed domination of P2P traffic in the Internet [24], [25]. An obvious solution to this problem is to make each peer connect to those peers which are closest to itself, in terms of latency. However this would create an overlay with high degree of clustering having a negative impact on the mechanisms employed in unstructured P2P networks.

In this paper, we propose ITA, an algorithm for *Innocuous Topology Aware* construction, which injects topology awareness to unstructured P2P overlays, while at the same time takes into consideration the impact the proposed changes will have on the rest of the mechanisms employed in unstructured P2P systems. It is able to do so by building a random graph of random graphs, therefore preserving the random nature of the overlay, while at the same time allowing for the existence of "neighborhoods", allowing peers to randomly connect to close-by peers. We use a diverse set of metrics to experimentally evaluate out proposal and to give a complete view of its impact in the system's operation. The results we obtain include a 50% reduction in search latency, a 20% reduction in the number of IP messages and a significant reduction on the load of the IP network routers. ITA is shown to have no negative impact whatsoever on the 1-hop replication and the dynamic querying mechanisms.

The remaining of the paper is organized as follows: In Section 2 prior work related to the problem is reviewed. Subsequently, in Section 3 some background knowledge necessary to the understanding of the remaining material is provided. The main result of this paper, the construction of the ITA algorithm and its accompanied searching method, is presented in Section 4 along with some analysis and discussion. Extensive experimental results are demonstrated in Section 5 and, finally, we conclude in Section 6.

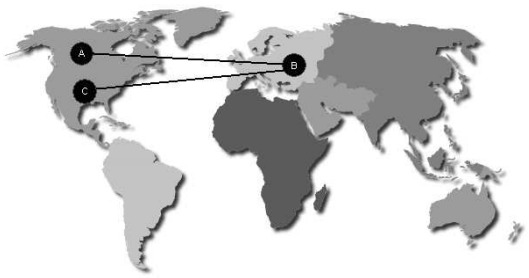

Fig. 1: Illustration of inefficient routing in today's unstructured P2P systems

II. Related Work

One of the main limitation for the scalability of unstructured P2P systems is the large number of messages generated by their flooding mechanism. For this reason, much of the research carried out in the field today aims at reducing those messages [20], [27], [12], [8]. However, the vast majority of the work is concerned with the overlay messages, even though a single overlay message usually translates to several IP messages. This abstraction has been shown to be problematic for the network layer.

Over the last few years, some work has been carried out aiming to address this problem. In the case of structured systems, the possibilities are limited since there are specific requirements for the neighbor selection of each peer. Due to the rigid structure of those systems, one has less freedom on how to rewire the connections in the system to allow for greater topology awareness. In [9] the authors propose the selection of the closest neighbor whenever there is more than one choices. This approach can be applied in systems like Pastry [23], Kademlia [19], and Tapestry [29]. However, in systems like Chord [28] and CAN [21], each neighbor is uniquely defined.

Our work focuses on unstructured systems which offer more versatility. Topology awareness algorithms that have been proposed for unstructured systems, such as [5], [14], focus on how to construct a topologically aware overlay but do not describe how to efficiently search this overlay. In addition, the constructed graph has a high clustering degree, which means that mechanisms employed in unstructured P2P systems and which depend on a random overlay to function properly, will experience a high loss in efficiency. In particular in [5], the overlay graph is formed by having each peer connect to those other peers with which it has the longest common domain suffix. In addition, some random links are inserted to avoid the partitioning of the network. The graph that is formed is comprised of neighborhoods of diverse sizes, since not all domains have the same peer population. This makes the choice for a universal value for the TTL difficult. The

same holds for the systems described in [4], [6], where the neighborhoods are defined by the IP addresses instead of the domain names. In flood-based P2P systems, the TTL value is critical for the efficient operation of the system. A TTL value which is appropriate for some of the neighborhoods can be inefficient for others, leading to either failure to locate content, or to the generation of a large number of duplicate messages. ITA constructs randomly connected "neighborhoods" of roughly equal size, which means that one TTL value "fits all".

In [11], the authors use synthetic coordinates to create neighborhoods of close-by, in terms of latency, peers. Their simulations were performed on a network which comprised 92 IP layer nodes and included 42 overlay peers. This network size makes it difficult to reveal the real benefit of the algorithm. In addition, in experiments of this scale it would be difficult to notice the effect of high clustering in the flooding mechanisms.

In [14], the overlay is constructed using a method inspired by the *k*-median algorithm, in order to construct again neighborhoods and reduce the average latency between peers in the overlay. This theoretical algorithm appears to be computationally expensive since it requires knowledge of the entire overlay topology to function. Furthermore, as the overlay changes from the departure and arrival of peers, the algorithm needs to continuously adjust the overlay in order to maintain its efficiency. The work described in [26] is a follow-up of [14]. The algorithm still needs to be active all the time to preserve the structure of the network. In addition, the main focus of this work is on the construction of an efficient graph for general use, as is the case for the work described in [15], [16]. We focus on how to efficiently construct an overlay with low clustering that maintains the beneficial properties of random graphs and leads to efficient information lookup. Finally, an interesting work is presented in [17]. The method described limits the reorganization of the network to add topology awareness in a 2-hop neighborhood for each peer. ITA constructs the entire overlay from the beginning to allow for the desired topology-awareness.

As we mentioned, previous work does not take into consideration the impact of the proposed methods on the widely deployed mechanisms such as 1-hop replication and dynamic querying. ITA functions without affecting them in any way, which means that there is no trade-off. In addition, the aforemention works require that each peer continuously execute the topology-awareness algorithm to adopt to changes in the P2P overlay. ITA only requires a simple and quick bootstrapping process, after which it can continue to function unaffected by the churn of the system. Furthermore, most of the aforementioned proposed methods try to connect each peer to its closest possible neighbors. This requires each peer to continuously probe the network in case some new, closest peer has joined[3], imposing additional traffic on the network and burden on each peer. Finally, most of the existing literature focuses on reducing the IP latency of queries. We evaluate our work using a variety of metrics including IP latency reduction, IP message reduction, and the traffic load placed on each router in the underling IP network. The latter we believe to be a crucial, often neglected, metric in current widely deployed P2P systems.

III. Background

In this section we provide some information on the technology of unstructured P2P systems today, which is essential for the understanding of the remaining of the paper. In particular, we present those mechanisms that would be most adversely affected by any increase in the clustering of the P2P overlay graph, since they were designed to be deployed on random graphs.

One technique widely used in unstructured P2P systems today, is 1-hop replication [22]. One-hop replication dictates that each peer should send to all of its immediate neighbors an index of its content (usually in the form of a Bloom filter). Using this index during the last hop propagation of a query at the Ultrapeer level, the request is forwarded exclusively to those last hop Ultrapeers that contain the requested file. One-hop replication reduces the number of messages generated during the last hop of flooding. However, as shown below, the traffic generated during that last hop constitutes the overwhelming majority of the traffic generated during the entire flooding.

Below we show the efficiency of flooding using 1-hop replication. We further demonstrate that the efficiency of 1-hop replication relies on a random graph. In Proposition 2 we prove that in order to flood an entire, randomly constructed, network that employs 1-hop replication, one need only reach $3/(d-1)$ of the peers during all hops but the last. Before we proceed to Proposition 2 we need to prove a preliminary result (Proposition 1). In what follows, we assume full network coverage is achieved when flooding has reached 95% of the graph nodes.

Proposition 1: In order to reach 95% of a graph's nodes using naïve flooding we need a minimum of $3 * N$ messages.

Proof: Let x be the number of messages generated during flooding. We want to compute x so that the flood reaches at least 95% of the graph nodes. This means that at most 5% of the peers will not receive any of the x messages. In a random graph, each time a message is sent, each peer has the same probability $1/N$ of being on the receiving side of that message. The probability that a peer receives neither one of x messages is $(1-1/N)^x \approx e^{-x/N}$. In order to achieve 95% coverage, this probability should

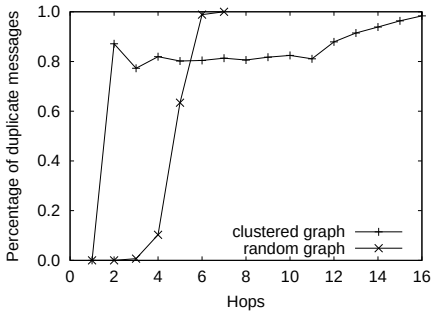

Fig. 2: Percentage of duplicate messages over all messages generated during each distinct flood hop

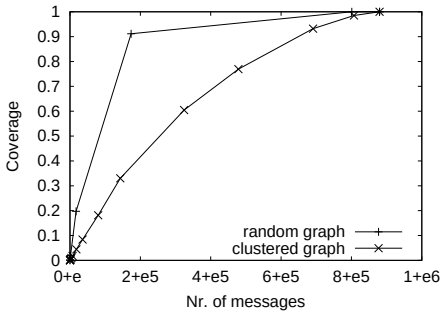

Fig. 3: Coverage of the graph for given number of messages

be less than 0.05:

$$e^{-x/N} \leq 0.05 \Rightarrow ln(e^{-x/N}) \leq ln(0.05) \Rightarrow$$

$$-x/N \leq -3 \Rightarrow x \geq 3 * N$$

Thus, we need at least $3 * N$ messages using flooding to reach 95% of the graph nodes. ■

Proposition 2: In order to reach 95% of a graph's nodes that employs 1-hop replication using flooding, we need to reach $3/(d-1)$ of the graph nodes in all hops except the last one.

Proof: Let n be a function that returns the number of new peers contacted at a given hop. Let f be a function that returns the number of messages generated on a single, given hop. Let d be the average degree of the graph. Initially $f(0) = 0$ and $n(0) = 1$. At each hop i it holds:

$$f(i) = n(i-1) * (d-1) \tag{1}$$

because each one of the nodes that received a message for the first time at hop $i-1$, will send it, at hop i, to all of their neighbors except the one it received the message from, thus to $d-1$ neighbors.
Let H be the hop before the last one. The total number of peers contacted up to hop H is $\sum_{i=0}^{H} n(i)$. Let r be the ratio of peers contacted up to hop H, then:

$$\sum_{i=0}^{H} n(i) = r * N \tag{2}$$

We want to compute ratio r so that after hop $H+1$, we will have reached at least 95% of the graph nodes. We have proven in Proposition 1 that we need a minimum of $3 * N$ messages to reach 95% of a graph's nodes using naive flooding. So

$$\sum_{i=1}^{H+1} f(i) \geq 3 * N \tag{3}$$

If we replace function f from (1) in the above formula:

$$\sum_{i=1}^{H+1} f(i) = \sum_{i=1}^{H+1} [n(i-1) * (d-1)] =$$

$$= (d-1) \sum_{i=1}^{H+1} n(i-1) = (d-1) \sum_{i=0}^{H} n(i)$$

This combined with (2) and (3) gives:

$$(d-1) * r * N \geq 3 * N \Rightarrow r \geq \frac{3}{d-1}$$

Thus the required result. ■

According to Proposition 2, in order to flood an entire randomly constructed network that employs 1-hop replication, one need only reach $3/(d-1)$ of the peers. The last hop peers are reached using 1-hop replication. In todays Gnutella, where the average degree is 30, one would need to reach 10% of the peers and then use 1-hop replication to forward the query to the appropriate last hop peers, in order to reach the entire network. This translates to a big saving in the number of messages. However, this result does not hold for clustered graphs, since the proof is based on the preliminary result in Proposition 1 which is only valid if each peer has equal probability to receive any message. This is the case only in graphs whose edges are constructed randomly. We have thus demonstrated that the efficiency of 1-hop replication heavily relies on a random overlay.

The second algorithms whose performance heavily relies on random overlays is dynamic querying. Dynamic querying [3] tries to imbue flooding with a finer granularity, regarding its extent of reach. The main idea is not to flood all of one peer's neighbors at the same time. Instead, the peer that initiates the flood sends the query message to only one of its neighbors. That neighbor, in turn, floods all its own neighbors with a small TTL. If that flood does not generate enough results, another flood with a higher TTL is sent to the next neighbor of the initiating peer and so forth, until the desired number of results is obtained, or we run out of neighbors. Again,

for this scheme to be efficient, it is required that when forwarding a flood message, the receiving peer has a low probability of having received that same message before, which only is the case in random graphs and not graphs with high degree of clustering. Otherwise, if all of the initiator's peers shared many neighbors, the flood would reach the same peers again and again.

On a clustered graph, the number of new peers contacted on each flood hop is greatly reduced. The reason for this is the distribution of duplicate messages generated on each hop of a flood. Given a flood with a boundless TTL (which will reach all the peers), a large number of the messages generated will be redundant, duplicate messages due to the circles in the graph structure, which will result in each peer receiving the flood message more than once. On a random graph, the majority of these messages will be generated during the last hops of the flood (after most of the graph has already been covered by the flood). On a clustered graph, on the other hand, the duplicate messages are distributed, more or less, evenly among all the hops of the flood (except the first of course). This shows why on a random graph, in the first hops, all flood messages will reach new peers, leading to a larger coverage of the graph using the same number of messages, than in a clustered graph. The above are illustrated in Figure 2, where one can see the difference in duplicate messages distribution between a clustered (small-world) and a random graph. In Figure 3 one can see that with the same number of messages, a larger portion of the random graph is reached. In these figures, we constructed a graph of 80000 nodes with an average degree of 13. The behavior illustrated in those two figures however is the same for any number of nodes or average degree and is only dependant on the degree of clustering of the graph.

Both the aforementioned mechanisms are critical for the scalability of unstructured P2P systems and for this reason we believe that any modifications and new proposals for those systems have to prove they only positively affect their behavior or do not affect it at all.

IV. ITA Design

In this section we shall describe the parts that comprise the design of our algorithm. We then present the advantages which arise from the design.

A. Overlay construction

The ultimate objective of the bootstrapping algorithm is to create for each peer a number of randomly selected *short* connections to *closer* (but not the closest) peers and the same number of randomly selected *long* connections to *distant* peers. The methodology used to select the "short" and "long" connections is based on parameter $\alpha \leq 1$ which constitutes the basic parameter of the algorithm. Let N be the total number of peers in the networks. Each peer A that bootstraps to the network selects its "short" connections randomly among its $\alpha * N$ closer (latency wise) peers, while it selects its "long" connections randomly among the $(1-\alpha) * N$ more distant (latency wise) peers.

To implement this method, each peers A calculates a threshold value x directly related to parameter α. Given the value of parameter $\alpha \leq 1$, each peer A that bootstraps to the network approximates threshold x so that its latency to $\alpha * N$ other peers is below threshold x while its latency to $(1-\alpha) * N$ other peers is above x. If C is the set of all peers P for which it holds that $latency(A, P) \leq x$, peer A calculates its threshold value x so that $|C| = \alpha * N$.

Since the latency from each peer to all other peers cannot be measured, the calculation of the threshold value x is approximated by having each peer A make latency measurements to $30/\alpha$ randomly selected peers. It is proven in Proposition 3 below, that this number of latency samples leads to a good threshold approximation.

Proposition 3: Each peer needs $30/\alpha$ latency measurements to other peers in order to approximate threshold x such that $|C| = \alpha * N$ for given $\alpha \leq 1$, with accuracy 95%.

Proof: A peer belongs to C with probability α. To obtain a good threshold approximation, we will select a peer in C that is among the 0.1*—C— peers whose latency is closer to the threshold value. The number of peers which are closer to the threshold according to our choice is $0.1 * \alpha * |C| = \alpha' * |C|$. The probability that a single randomly selected peer belongs to that space is $\alpha' = 0.1 * \alpha$. The probability that neither one of n randomly selected peers belong to that space is $(1-\alpha')^n \simeq e^{-\alpha' * n}$. To approximate the threshold with accuracy 95% we need,

$$e^{-\alpha' * n} <= 0.05 \Rightarrow ln(-\alpha' * n) \leq ln(0.05) \Rightarrow$$

$$-\alpha' * n \leq -3 \Rightarrow n \geq 3/\alpha' \Rightarrow n \geq 30/\alpha$$

So each peer needs $30/\alpha$ latency measurement samples to approximate the threshold. ■

During the sampling measurements, peer A can connect randomly to begin its operation without having to wait for the end of the sampling procedure.

The Vivaldi coordinate system [10] can be used for the bootstrapping. Vivaldi is a P2P network coordinate system which can assign a 3-dimensional coordinate to a host. The Euclidian distance between two Vivaldi points (corresponding to two hosts) is an approximation of their latency. Thus, each message broadcast by any peer can contain its Vivaldi coordinates. A bootstrapping peer A can monitor incoming traffic, collect $30/\alpha$ Vivaldi coordinates and thus compute the threshold value x.

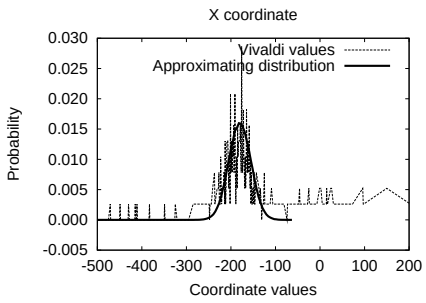

Fig. 4: Actual values and approximation distribution for the x coordinate

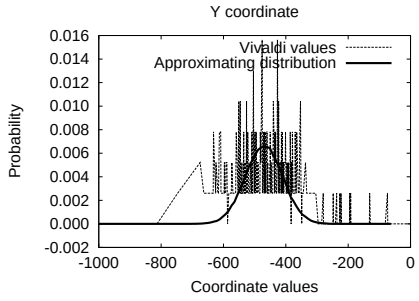

Fig. 5: Actual values and approximation distribution for the y coordinate

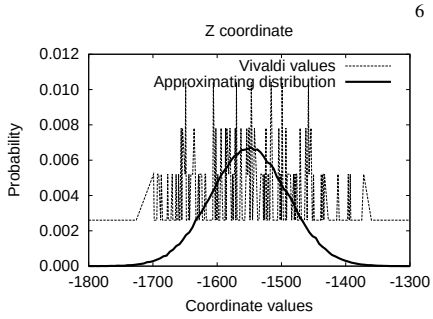

Fig. 6: Actual values and approximation distribution for the z coordinate

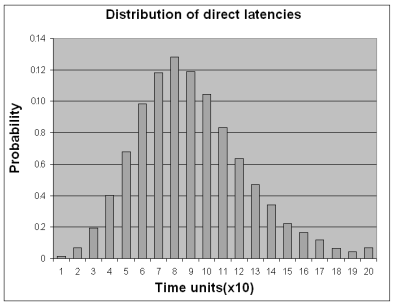

Fig. 7: Distribution of direct latencies between all pairs of peers

It must be noted that, unless the structure and capacity of the network changes significantly, the same threshold value can be used multiple times, and does not need to be recalculated each time the peer joins the overlay. After a threshold value has been obtained, peer A connects to $2/\alpha$ neighbors in the following fashion:

- It connects randomly to $1/\alpha$ peers, all of which belong to C. These links are called *short* links.
- It also connects randomly to $1/\alpha$ other peers, which *do not* belong to C. These are called *long* links.

To illustrate, let's assume that parameter α is set to 0.1. This means that C contains approximately $0.1 * N$ nodes. Each peer A will create $1/\alpha = 10$ short links randomly selected among the 10% closer to A peers, and the same number of long links randomly selected from the 90% further peers. The number of sample measurements required for the calculation of the threshold, in this case, is $30/\alpha = 300$. Given that a Gnutella peer receives 50 query messages per second on average [18], the latencies to 300 random peers can be collected in seconds.

B. Search algorithm

Search is conducted in the following fashion:

- The Initiator peer floods its long links with $TTL = 1$.
- Each of the peers that receives the flood over a long link (and the Initiator peer) initiates a flood with a given $TTL = ttl$ (system parameter) over their short links only.

The long link peers which initiate the localized floods (over their short links) use 1-hop replication as well as dynamic querying the same fashion it is used in Gnutella today. Since short links are randomly connected the efficiency of dynamic querying and 1-hop replication is guaranteed.

C. Analysis

The constructed graph, in conjunction with the described search method, has the following advantages:

- Both the long link-based, system-wide graph and the short link-based, local graphs are random, since each peer selects peers (outside and inside C respectively) randomly for neighbors. This enables both 1-hop replication and dynamic querying to operate as if they were executed on a random graph.
- Since any peer in C can serve as short link (instead of opting for the closest ones), the bootstrapping procedure is very fast and lightweight. The same holds for the long links. This means that each peer need only set up its neighbors once, regardless of arrivals and departures elsewhere in the overlay, making ITA as little affected by churn as Gnutella (i.e. a peer only needs to act when a neighbor leaves the system by simply replacing it with another one, as in Gnutella). This simplicity helps preserve almost intact the unstructured nature of the overlay. If we tried to connect to the closest possible peers, this would require each peer to be on the constant lookout for some closer peer connecting to the network. In addition, the threshold value is only affected by changes in the structure of the underlying IP network (which are not very frequent) and not by changes in the P2P overlay, which are rather frequent.
- $(1 - \alpha) * N$ peers (furthest away on the IP layer)

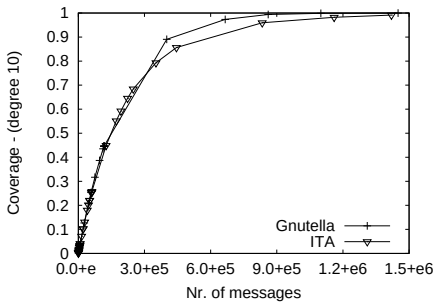

Fig. 8: Flood reach for given number of messages. Average degree = 10

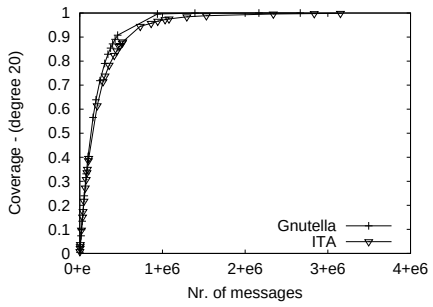

Fig. 9: Flood reach for given number of messages. Average degree = 20

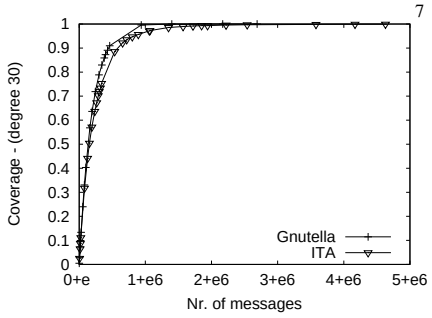

Fig. 10: Flood reach for given number of messages. Average degree = 30

are excluded from becoming short links, which means the proposed system is quite aware of the underlying physical network topology. Increased awareness in the form of a very small α (i.e. trying to connect to the closest possible peers) would help us gain little but lose much, since the small size of the local neighborhoods would lead to high clustering.

- All local clusters/neighbourhoods have the same size, enabling the use of a single, system-wide $TTL = ttl$ for flooding the short links.

We have conducted experiments using three distinct values for α, namely 0.1, 0.05 and 0.033. These values correspond to a number of 10, 20 and 30 long and the same number of short links. The above discussion justifies the reason for not using smaller values. Values in this range are sufficient for excluding most of the peers from the local "neigborhood" set C of each peer, while being at the same time large enough to allow large enough neighborhoods for quick and simple bootstrapping procedure (i.e. being able to quickly locate short-link neighbors). The value of α also dictates the number of the long links, since there are $N/|C| = 1/\alpha$ "neighborhoods". In addition, the use of $1/\alpha$ long links is due to the fact that the use of long links should only take place on the first hop, to avoid extra delays in the flood process. Finally, it is important to note that there is no 1-hop replication between peers connected by long links, so there is no index information exchange. Thus, the maintenance overhead for the additional $1/\alpha$ long links very low.

V. Experimental results

In order to verify the arguments made in the previous section, we performed several experiments comparing our system with Gnutella, the most widely used unstructured P2P network (approximately 2 million users) [7]. We performed the comparison with Gnutella 0.6, which employs a 2-tier architecture [13], focusing on the Ultrapeer layer where flooding occurs. The metrics upon which our comparison was based were selected to capture the design goals of ITA, namely to satisfy users by allowing them to get the same number of search query results faster by reducing query response time, and to satisfy ISPs by reducing the load imposed on their routers.

The random nature of the constructed overlay is indicated by the extent of the reach of a flood for given number of messages. As shown in Figure 3, a flood on a clustered graph will have a much smaller reach than on a random graph, using the same number of messages. Figures 8, 9 and 10 show the similarity between the Gnutella overlay (random graph) and the overlay constructed by ITA with respect to flooding. The close fit of both curves on the two graphs shows that the flood reach is the same in both graphs using the same number of messages. This means that ITA can provide reduced latency and reduced router load benefits (see below) without affecting 1-hop replication, dynamic querying, and the self-* properties on which Gnutella-like systems depend for their performance.

Subsequently, we compare ITA and Gnutella using three different metrics. The first metric is the latency of the connections of the peers, which affects the duration of a flood. We measure the average time it takes for a flood to complete, for different TTL values. The second metric is the number of IP messages generated during a single flood. We measure the average number of IP messages generated during floods of increasing $TTLs$. Finally, the last metric is the standard deviation of the message load imposed on the routers that comprise the IP layer of the Internet. We argue that a reduction in the total number of IP messages in the whole network is of little use if there exist a small number of bottleneck routers whose traffic load remains the same as before. As we mentioned above, the key goals of the ITA algorithm is to benefit *both* the P2P application *and* other applications sharing the same medium, the Internet.

We simulated a network of 200,000 peers, which is a realistic number for the size of the Ultrapeer overlay in Gnutella according to LimeWire [2], the company that

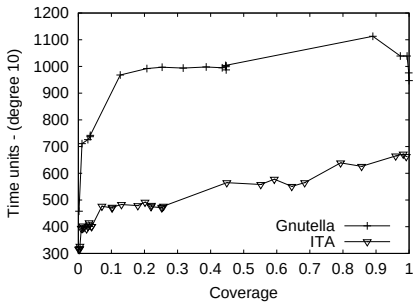

Fig. 11: Time required by a flood versus the percentage of nodes reached. Average degree = 10

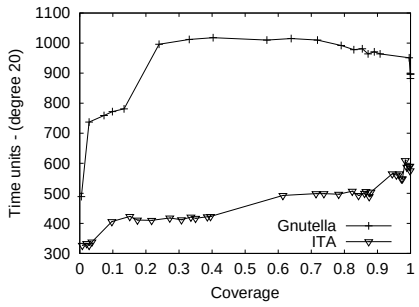

Fig. 12: Time required by a flood versus the percentage of nodes reached. Average degree = 20

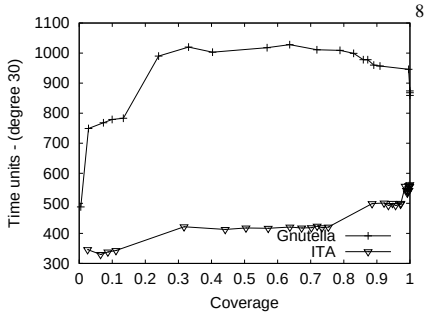

Fig. 13: Time required by a flood versus the percentage of nodes reached. Average degree = 30

developed the most popular Gnutella client today. We also used three average degree values for the overlay, namely 30 (which is the average number of connections in a Gnutella Ultrapeer today), 20 and 10. These three numbers correspond to the number of connections per peer in the Gnutella simulations and the number of short and long links in the simulations of the ITA algorithm. Note that since the long links are only used during the first hop of flooding, whereas the short links are used during the second and the remaining hops, the outbound degree during any flood hop is the same both in Gnutella and ITA, even though our algorithm uses double the number of links.

In order to model the 200,000 by 200,000 latencies, we obtained 250 real Vivaldi coordinates. Those 3D coordinates were produced by the Vivaldi project experiments on PlanetLab [10]. We then calculated a distribution which best fits the values observed in those coordinates and we generated 200,000 Vivaldi coordinates using this distribution, thus being able to model the latency between any pair of the 200,000 peers. Figures 4, 5 and 6 show the values of the original Vivaldi coordinates as well as the distributions generated for each of the three Vivaldi subcoordinates (X, Y, Z). The close fit is an assurance that our randomly generated coordinates closely reflect real-world Vivaldi coordinates. Given the 200,000 x 200,000 latency matrix we generated, Figure 7 shows the distribution of the latency for an optimal full mesh graph where each peer has a direct overlay connection to each other peer. The figure shows that the average latency between any two peers is 90 time units.

A. Latency experiments

Figures 11, 12 and 13 show the time it takes to flood the network, for a given node coverage. We can see that for any desired coverage, the time it takes for our system to reach that number of peers is, on average, at least half the time for Gnutella flooding. Note that the measured time reflects the time from the beginning of the flood until even the last message generated by that particular flood reaches its destination.

There are two reasons for measuring flood duration rather than average response time for a search query. First, a reduction by half in flood duration implies a similar reduction in average query response time. Most importantly however, it is common that a flood is still active and being propagated in the network, even though no new results are provided to the user, so minimizing flood duration when possible is important. Given a constant rate by which new queries enter the network, by measuring the time it takes for a single flood to complete to the last message, we show that ITA doubles the exit rate of floods from the network. This means that ITA doesn't only reduce the number of IP messages per flood (as we will show in the next section) but reduces also the build-up of queues in the router buffers.

B. IP layer experiments

In this section we focus on the impact the ITA algorithm has on the IP layer. In order to perform simulations including the IP layer we obtained the latest trace of the router-level topology of the Internet from CAIDA [1]. We opted to use a router-level graph instead of an Autonomous System graph for two reasons. Firstly, a router-level graph is more detailed and will provide a better approximation of the actual number of messages generated on the IP layer. Secondly, many ASs contain routers even on different continents. An AS-level graph will hide the latencies between those routers.

In this simulation scenario, we used again 200,000 peers, each of which was randomly assigned to a router in the router-level graph. Since the router-level trace did not contain latencies, we approximated the latencies with the number of IP hops between any two peers. Thus, each

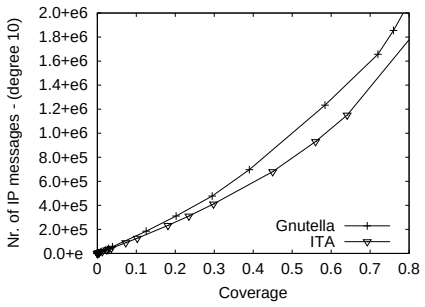

Fig. 14: IP messages generated by a flood versus the percentage of nodes reached. Average degree = 10

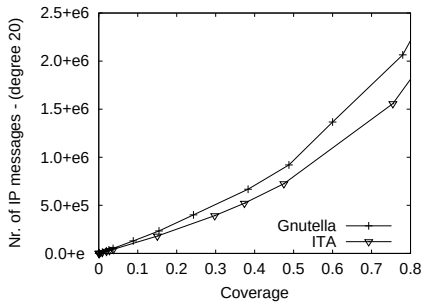

Fig. 15: IP messages generated by a flood versus the percentage of nodes reached. Average degree = 20

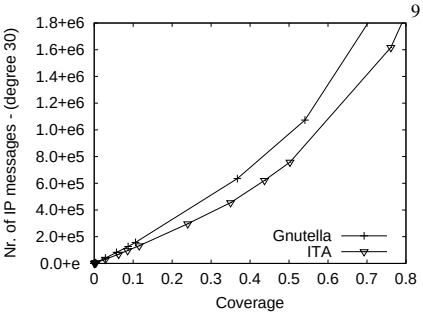

Fig. 16: IP messages generated by a flood versus the percentage of nodes reached. Average degree = 30

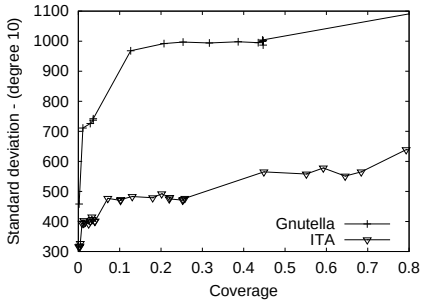

Fig. 17: Standard deviation of router traffic loads versus the percentage of nodes reached. Average degree = 10

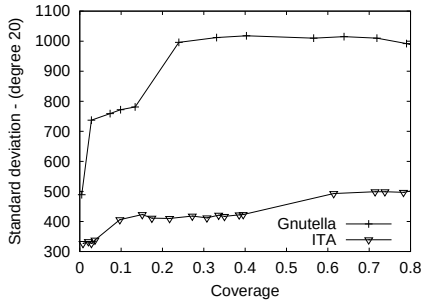

Fig. 18: Standard deviation of router traffic loads versus the percentage of nodes reached. Average degree = 20

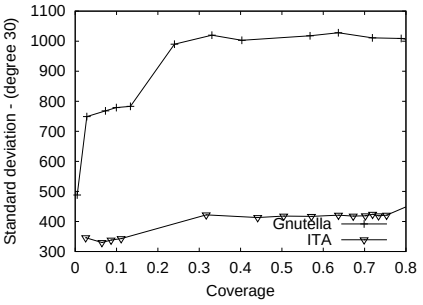

Fig. 19: Standard deviation of router traffic loads versus the percentage of nodes reached. Average degree = 30

peer tries to form short links with those other peers whose routers are close to its own on the IP layer. Again, we do that by obtaining the $\alpha * 100\%$ of all routers which are closest to our own router. Long links are again formed randomly, as are short links in a given neighborhood. Some measurements on the formed overlay show that the average number of routers in a Gnutella direct link between two peers is 6.9. In contrast, the same number for ITA's long links is 7 and for the short links it is 5.5. As we shall see below, we can expect a reduction of message on the order of 15% to 25%% ($\simeq (7 - 5.5)/7$). Given the percentage of the routers which can be reached for a single TTL value, which is shown in Figure 20, the average values we mentioned make a lot of sense. This figure shows the ratio of all peers that can be reached for a given hop distance. This shows that the vast majority of routers need to traverse a chain of at least 3 hops before they start encountering more than one per hop routers. This means that 4 is, more or less, a minimum value for a short link, imposing a lower bound on the reduction of IP messages that we can accomplish.

After running the simulations, which included performing several floods from random peers, with several TTL values, we obtained the following results: Figures 14, 15 and 16 illustrate the reduction in the number of IP messages for floods of various lengths. As one can see, the expected reduction that is observed is in the range of 15% to 25%.

Another important metric for the efficiency of any topology-aware overlay construction algorithm is the traffic load distribution across the routers in the system. For this reason, we plotted the standard deviation in the traffic load of all routers, again for floods of different sizes. Figures 17, 18 and 19 show that ITA reduces the standard deviation approximately by 40% to 50%. Finally, we measured the traffic load for the most heavily used router, which ITA also cuts down by half.

VI. Conclusions

We presented ITA algorithm, a novel approach for injecting topology awareness into unstructured Gnutella-like P2P systems, while maintaining the self-* properties of the overlay topologies that are highly desirable in these systems. ITA reduces to half the time required for a search query to achieve a particular network coverage compared to the latest version of the widely deployed Gnutella.

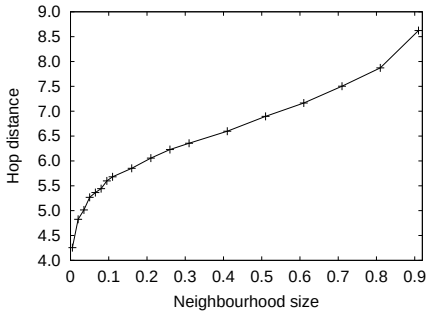

Fig. 20: Diameter (in hops) of different neighborhood sizes

Moreover, ITA reduces the number of IP messages generated during a search query flood by as much as 25%, which is a significant reduction for ISPs who care about the load imposed on their routers and its effect on the performance of other applications.

VII. Acknowledgments

This research work was carried out partially under the FP6 NoE CoreGRID funded by the EC (IST-2002-004265). We would like to thank the people behind the Vivaldi and Pyxida projects for their help in obtaining the Vivaldi coordinates. Finally, we would also like to thank Prof. Michalis Faloutsos for his helpfull suggestions.

References

[1] Cooperative association for internet data analysis, http://www.caida.org/home.

[2] Limewire inc, http://www.limewire.com.

[3] A. Author, Fisk. Gnutella ultrapeer query routing, v. 0.1. 2003.

[4] B. Krishnamurthy Author and J. Wang. Topology modeling via cluster graphs. *SIGCOMM Internet Measurement Workshop*, 2001.

[5] V. Author, Kalogeraki, D. Gunopulos, and D. Zeinalipour-Yazti. A local search mechanism for peer-to-peer networks. *Proceedings of the 11th international conference on Information and knowledge management, (CIKM02)*, page 300307, 2002.

[6] V.N. Padmanabhan Author and L. Subramanian. An investigation of geographic mapping techniques for internet hosts. *ACM SIGCOMM*, 2001.

[7] Eric Bangeman. Study: Bittorrent sees big growth, limewire still nr.1 p2p app. *ars technica*, 2008.

[8] M. Mihail C. Gkantsidis and A. Saberi. Hybrid search schemes for unstructured peer-to-peer networks. *INFOCOM*, 2005.

[9] M. Castro, P. Druschel, Y. C. Hu, and A. Rowstron. Proximity neighbor selection in tree-based structured peer-to-peer overlays. Technical Report MSR-TR-2003-52, Harvard, 2003.

[10] Frank Dabek, Russ Cox, Frans Kaashoek, and Robert Morris. Vivaldi: A decentralized network coordinate system. *SIGCOMM*, 2004.

[11] André Dufour and Ljiljana Trajković. Improving gnutella network performance using synthetic coordinates. In *QShine '06: Proceedings of the 3rd international conference on Quality of service in heterogeneous wired/wireless networks*, page 31, New York, NY, USA, 2006. ACM.

[12] V. Cholvi P. Felber and E. Biersack. Efficient search in unstructured peer-to-peer networks. *Proc. 16th ACM Symposium on Parallelism in Algorithms and Architectures*, 2004.

[13] The Gnutella Developer Forum. Gnutella 0.6 protocol specification.

[14] Nikolaos Laoutaris, Georgios Smaragdakis, Azer Bestavros, and John Byers. Implications of selfish neighbor selection in overlay networks. *IEEE INFOCOM*, 2007.

[15] Z. Li and P. Mohapatra. Impact of topology on overlay routing service. *IEEE INFOCOM*, 2004.

[16] Y. Liu, H. Zhang, W. Gong, and D. F. Towsley. On the interaction between overlay routing and underlay routing. *IEEE INFOCOM*, 2005.

[17] Yunhao Liu, Li Xiao, Xiaomei Liu, L. M. Ni, and Xiaodong Zhang. Location awareness in unstructured peer-to-peer systems. *Parallel and Distributed Systems, IEEE Transactions on*, 16(2):163–174, 2005.

[18] Evangelos P. Markatos. Tracing a large-scale peer to peer system: an hour in the life of gnutella. In *CCGrid 2002: the second IEEE International Symposium on Cluster Computing and the Grid*, pages 65–74, 2002.

[19] P. Maymounkov and D. Mazieres. Kademlia: A peer-to-peer information system based on the xor metric. *IPTPS02*, 2002.

[20] Harris Papadakis, Paraskevi Fragopoulou, Marios Dikaiakos, Alexandros Labrinidis, and Evangelos Markatos. Divide et impera: Partitioning unstructured peer-to-peer systems to improve resource location. *Achievements in European Research on Grid Systems CoreGRID Integration Workshop 2006 (Selected Papers)*, 2007.

[21] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Schenker. A scalable content-addressable network. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 161–172, New York, NY, USA, 2001. ACM.

[22] Christopher Rohrs. Query routing for the gnutella network. 2001.

[23] Antony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, 2001.

[24] S. Saroiu, K. P.Gummadi, R. J. Dunn, S. D. Gribble, and H. M. Levy. An analysis of internet content delivery systems. *5th Symposium on Operating Systems Design and Implementation*, 2002.

[25] S. Sen and J. Wang. Analyzing peer-to-peer traffic across large networks. *ACM SIGCOMM Internet Measurement Workshop*, 2002.

[26] Georgios Smaragdakis, Nikolaos Laoutaris, Azer Bestavros, John W. Byers, and Mema Roussopoulos. Egoist: Overlay routing using selfish neighbor selection. *BUCS-TR-2007-013*, 2007.

[27] K. Sripanidkulchai, B. Maggs, and H. Zhang. Efficient content location using interest-based locality in peer-to-peer systems. *INFOCOM*, 2003.

[28] Ion Stoica, Robert Morris, David Karger, Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable Peer-To-Peer lookup service for internet applications. *Proceedings of the 2001 ACM SIGCOMM Conference*, pages 149–160, 2001.

[29] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, UC Berkeley, 2001.