ITA: Innocuous Topology Awareness for Unstructured P2P Networks

Harris Papadakis, Mema Roussopoulos, Paraskevi Fragopoulou 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*—One of the most appealing characteristics of unstructured P2P overlays is their enhanced self-* properties, which is due to their loose, random structure. In addition, 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. The underlying communications network (i.e. the Internet), however, is not as randomly constructed. This leads to a mismatch between the distance between two peers on the overlay and the hosts they reside on at the IP layer, which in turn leads to its misuse. The crux of the problem arises from the fact that any effort to provide a better match between the overlay and the IP layer will inevitably lead to a reduction in the random structure of the P2P overlay, with many adverse results. With this in mind, 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, both at the IP router level and autonomous system level, we show that ITA reduces communication latencies by as much as 50%. Furthermore, it not only reduces by 20% the number of IP network messages which is critical for ISPs carrying the burden of transporting P2P traffic, but also distributes the traffic load more fairly 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

The P2P paradigm has received substantial attention from researchers of several fields of the distributed systems, spanning from file-sharing and content delivery to Grid systems. In addition, several P2P-based applications have been in widespread use from the aforementioned file-sharing to IP-telephony. Almost all these uses and applications have one thing in common and that is the global-scale deployment since the P2P paradigm emerged as a design paradigm to provide global scalability, something that the traditional client-server paradigm was unable to achieve. However, as we will describe later, this global deployment magnifies the problem this paper tries to solve.

One of the most critical design aspects of any P2P system is the overlay layer, that is a virtual network of interconnected peers (P2P client-servers) through (and on top of) the underlying IP network. The structure of this overlay network is tightly coupled with the search algorithm, which is usually the main function of a P2P system. This means that the network structure is such as to enable and facilitate this function, which also means that there are rules governing which peers are connected to which peers.This is more apparent 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 requires a logarithmic (*O(logN)*) search cost in the number of messages.

On the other hand, unstructured systems, by definition, do not impose a specific structure on the overlay. Each peer is free to connect to any other (available) peer. Even though this lack of structure denotes a large degree of freedom in the creation of the overlay, we will show that this is misleading. Most mechanisms used widely in unstructured P2P systems today, actually rely on this random selection of neighbors (the peers to connect to), regardless of their distance and position in the IP network. This leads to a complete lack of corellation between the two respective distances (IP network and overlay), wherein lies the problem we aim to rectify.

So, either structured or unstructured, all P2P systems have their own design goals on overlay creation, which do not include taking into consideration the structure of the underlying physical network, the Internet. As a result, most P2P systems make an inneficient use of the IP layer, which has adverse impact not only on their own operation but also on the operation of the other applications, which co-exist on the same medium (the Internet). Some proposals have already been published, which aim to rectify this. Most of them rely on the freedom of peers in unstructured neighbors to connect to any peer they want, in order to create an overlay which better matches the IP layer. However, as we mentioned and will show, this freedom to choose any peer as neighbor is more of a requirement than actual freedom. This means that showing any preference, during neighbourhood selection, on peers depending on their position and distance in the IP network violates this requirement, and thus, we argue, these approaches greatly affect some of the most 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 closer at the IP layer and/or have smaller latency. The second and most important drawback is that the IP path behind each P2P overlay connection contains a large number

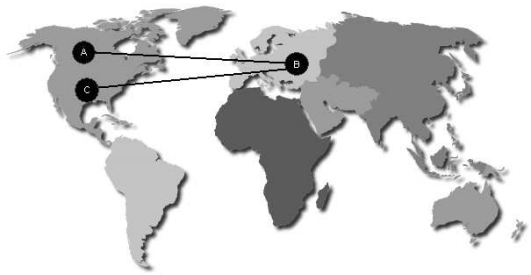


Fig. 1: Illustration of inefficient routing in todayś unstructured P2P systems

of routers. This means that even a single, 1-hop, message between neighbors (at the P2P overlay) may travel through many routers and autonomous systems 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. This 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 have each peer connect to those peers which are closest to itself, in terms of latency, while maintaining a small number of further links to avoid overlay partitioning. However this would create an overlay with a higher degree of structure (clustering), which will have a negative impact on the mechanisms employed in unstructured P2P networks.

In this paper, we aim to solve this canandrum. We propose ITA, an algorithm for *Innocuous Topology Aware* construction, which provides unstructured P2P overlay creation with a large degree of topology awareness, while at the same time taking 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, in a sense, 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 on 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 (approx also 50%) 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, which we describe below.

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.

II. Related Work

One of the main drawbacks of unstructured P2P systems is the limitation of their scalability due to the large number of messages generated by their search mechanism, called flooding. This is evident in the fact that a large part of the existing literature aims at reducing those messages [20], [27], [9], [5]. However, the vast majority of this work is concerned with reducing the number of 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.

In the case of structured systems, some work has been carried out aiming to address this problem, even though the possibilities are limited since there are specific requirements for the neighbor selection of each peer. Due to the more 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 [6] the authors propose the selection of the closest neighbor whenever there are more than one choices. This approach can be applied in systems like Pastry [23], Kademlia [18], and Tapestry [30]. However, in systems like Chord [28] and CAN [21], each neighbor is uniquely defined.

Our work focuses on unstructured systems, which are not as sensitive to changes in the overlay creation. Topology awareness algorithms that have been proposed for unstructured systems, such as [13], [15], aim at constructing a generic, topologically aware overlay, and thus do not describe any mechanism for efficiently searching on that overlay. In addition, the constructed graph has a high clustering degree, which predicates that the mechanisms already employed in unstructured P2P systems and which depend on a random overlay to function properly, will experience a high loss in efficiency. In particular, the authors of [13] describe an overlay graph creation method, which is based on having each peer connect to those other peers with which it has the longest common domain suffix. Some random links are also maintained in order to avoid the partitioning of the network. In addition to the drawbacks common to all approaches which increase topology awareness by reducing the randomness of the graph this approach has an additional disadvantage. 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 $Time-To-Live(TTL)$ difficult. The same holds for the systems described in [14], [19], 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 and is directly connected not only its scalability but also its operational success. 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 [8], 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 small 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 the increased clustering in the flooding mechanisms. In [15] overlay creation is inspired by the *k*-median algorithm, in order to, again, construct neighborhoods of nearby, latency-wise peers and thus reduce the average latency of any path between any two 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 [15]. 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 [16], [17], so there is no descrption on how to search the overlay. 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 [29]. 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, the method used to construct and the resulting structure of the overlay is tightly coupled with the other mechanisms at work in a P2P system. In existing P2P systems this is especially true for the mechanisms that comprise the search-lookup function. The work we just mentioned does not take into consideration the impact of the proposed methods on those 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. Any increase in topology awareness comes at no-cost. In addition, most of the aforemention work requires that each peer continuously execute the topology-awareness algorithm to adopt to changes in the P2P overlay. This is mainly because most of the aforementioned proposed methods try to connect each peer to its closest possible neighbors. This set however changes dynamically in time, due to the churn in the network. ITA only requires a simple and quick bootstrapping process, after which it can continue to function unaffected by the churn of the system. Furthermore, this continuous operation of most of the aforementioned proposed methods requires each peer to continuously probe the network in case some new, closest peer has joined, imposing additional traffic in the network and burden on each peer.

The most recent related work can be found in [12]. In this work, they describe an algorithm for creating an overlay with constant delay between any two peers in the network. Each peer maintains a number of small random ids. In addition, it samples the network by contacting a number of random peers and initiating a walk from each one, by following peers of decreasing ids, towards the peer with the minimum id in the network. The peers with the lowest latency are chosen as neighbours. We chose this algorithm to compare with ITA latency-wise. Experimental results show that ITA obtains lower latency between peers. In addition, as we shall describe, ITA requires a constant number of samples to create the overlay, whereas Hsiao et al. algorithm requires a number of samples, which is logarithmic to the number of peers in the system. 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 describe those technologies used in unstructured P2P systems today which are both essential for the understanding of the remaining of the paper and also are those more negatively affected by changes in the overlay structure. We will also show both the importance of those mechanisms and why they are so negatively affected. One of the most important, scalability-wise, techniques 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). This information is used during the last hop propagation (at the Ultrapeer level) of a query, by forwarding the request only to those last hop Ultrapeers that their index indicates that they may contain the requested file. One-hop replication reduces only those 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, since the number of messages per increasing TTL, increases exponentially.

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. Proposition 2 shows 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 introduce 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.

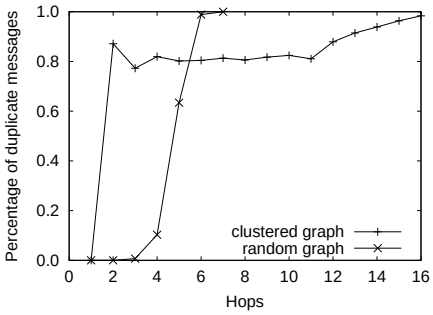


Fig. 2: Percentage of duplicate messages over all messages generated during each distinct flood hop

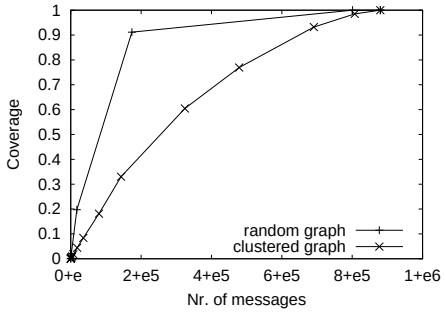


Fig. 3: Coverage of the graph for given number of messages

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 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: $f(i) = n(i-1) * (d-1)$ (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: $\sum_{i=0}^{H} n(i) = r * N$ (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 $\sum_{i=1}^{H+1} f(i) \geq 3 * N$ (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 (1) and (2) 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 rest, 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 before employing 1-hop replication. 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 algorithm whose performance heavily relies on random overlays is dynamic querying. As we mentioned before, the number of messages generated on each TTL increases exponentially. This means that while a flood with a given TTL may reach a small part of the system peers, a flood with the TTL increased by one may well reach all the peers, with a possible forbidding amount of messages. Dynamic querying [10] tries to imbue flooding with a finer granularity, regarding its extent of reach. It relies on the assumption tha a user will be more than happy with enough results to its query, instead of every result present in the system. 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 sequentially to each neighbour, with increasing TTL, until enough results have been obtained or have run out of neighbours. 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. This is because in clustered graphs, neighbouring peers share common neighbours which leads to peers receiving the same message more than once even on the second hop. This leads to duplicate, redundant messages from early on in the flood. This only happens during the last hops in random graphs. 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.

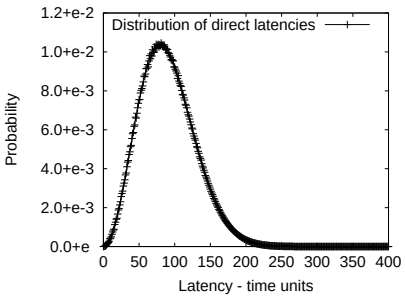


Fig. 4: Distribution of direct latencies between all pairs of peers

IV. ITA Design

This section contains a detailed description of the parts that comprise the design of our ITA algorithm. We then present a discussion and analysis of the advantages which arise from it.

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 definition of the "short" and "long" connections is based on parameter $\alpha \leq 1$ which constitutes the basic and most fundamental 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 peer A calculates a (latency related) 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 a threshold value x so that the number of peers whose latency to A is less than x is $\alpha * N$. In other words, 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. A proof is provided in Proposition 3 below, which shows 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. ■

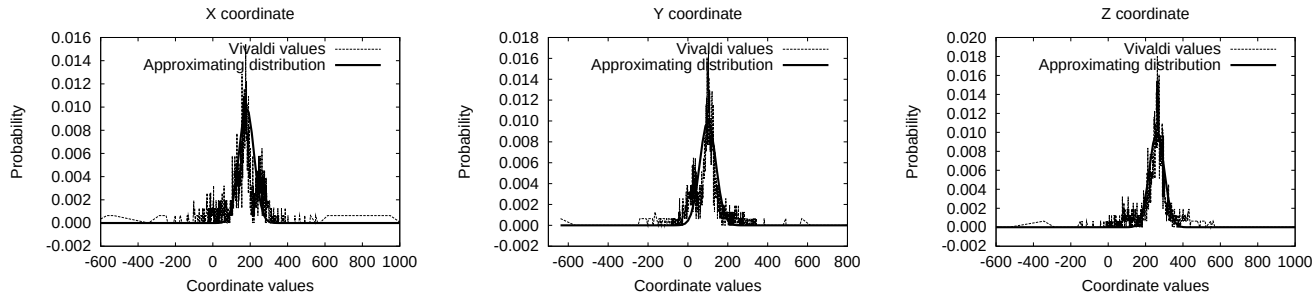
During the sampling measurements, peer A can connect randomly to begin its operation without having to wait for the end of the sampling procedure.

In addition, the Vivaldi coordinate system [7] can be used to facilitate and speed-up the bootstrapping process. 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. Ultrapeers today are reached by at least fifty query messages per second, making the threshold calulation this way a matter of seconds.

It should also be noted that, unless the structure and capacity of the network changes significantly, the threshold value remains unchanged, and so 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 (i.e.: any peers with a latency lower than the threshold value). 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 of all the nodes (peers) in the system. Note that the set C is, of course, different for each peer. Each peer A will create $1/\alpha = 10$ short links randomly selected among the 10% closer to A peers (i.e. among the peers in $A's$ C set), and the same number of long links randomly selected from the 90% further peers. The number of sample measurements required for the



(a) x coordinate (b) y coordinate (c) z coordinate

Fig. 5: Actual values and approximation distribution for the three coordinates

calculation of the threshold, in this case, is $30/\alpha = 300$. Not only it take a few seconds to perform this number of RTT measurements, it only takes place once, and not every time a peer (re-)connects in the system.

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.

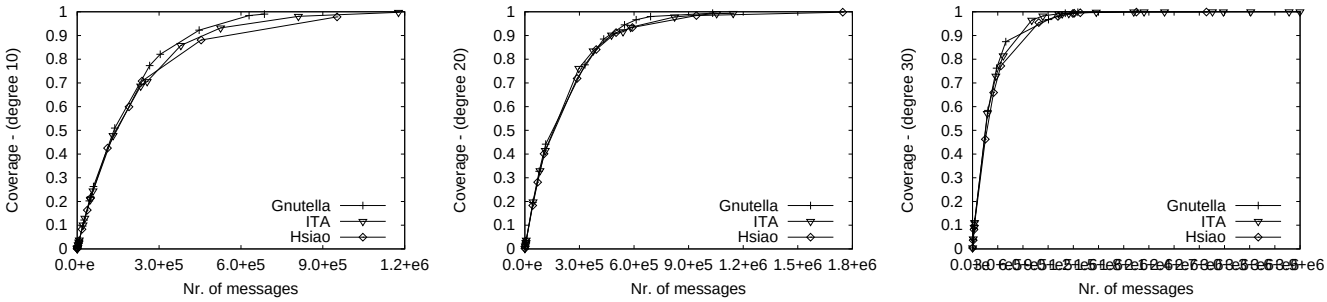
The long link peers which initiate the localized floods (over their short links) use 1-hop replication as well as dynamic querying the same fashion it is used in Gnutella today. Since short links are randomly connected the efficiency of dynamic querying and 1-hop replication is guaranteed. Alternatively, Dynamic Querying can be used on the long links level by sequenatialy sending a new flood with increasing TTL, to each long link neighbour.

C. Analysis

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

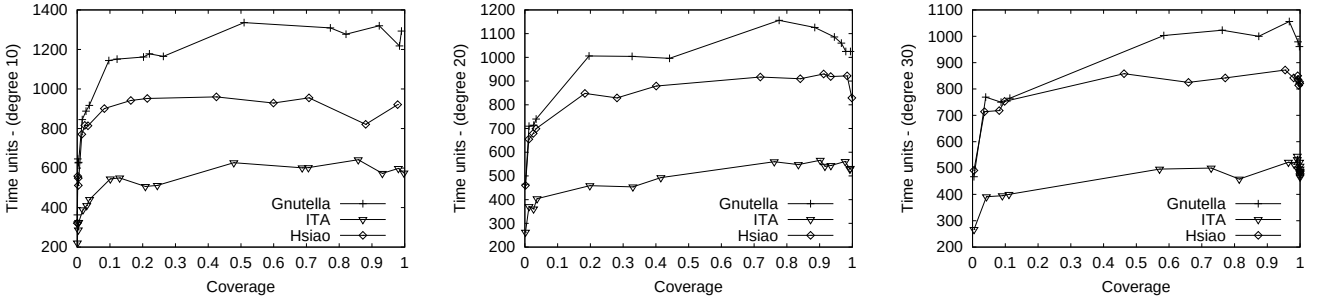
- Both the long link-based, system-wide graph and the short link-based, local graphs are random, since each peer selects peers (outside and inside C respectively) randomly for neighbors (i.e. each peer, for instance, in C has the same probability of becoming a short link peer of the same peer A). This 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. As a result 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 and the simplicity of construction of the overlay. What is more, 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 (anywhere) to the network. This constant probing would increase both the traffic in the network and the computational load of the system. 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. So the value is calculated only once and not each time the peer (re-)joins the network.
- $(1 - \alpha) * N$ peers (furthest away at the IP layer) 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.
- Finally, 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.



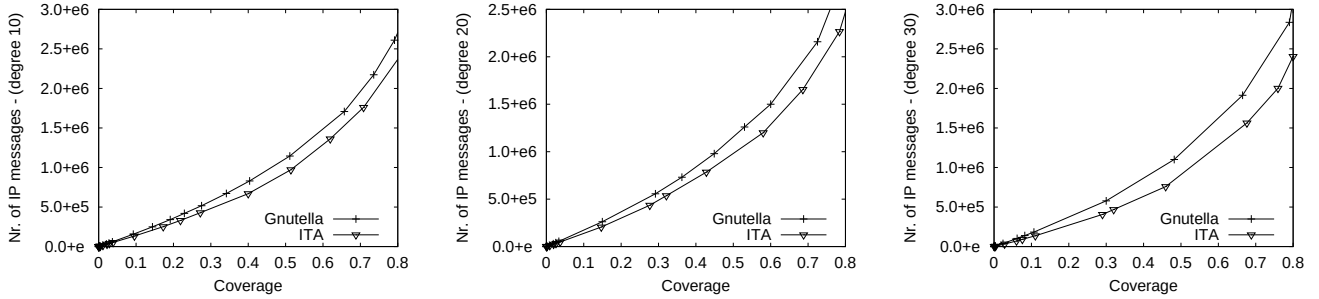
(a) Average degree = 10 (b) Average degree = 20 (c) Average degree = 30

Fig. 6: Flood reach for given number of messages.



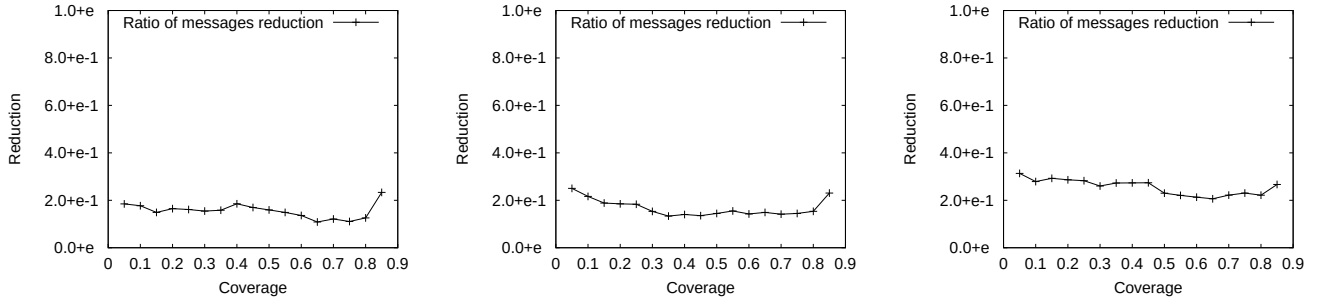
(a) Average degree = 10 (b) Average degree = 20 (c) Average degree = 30

Fig. 7: Time required by a flood versus the percentage of nodes reached.



(a) Average degree = 10 (b) Average degree = 20 (c) Average degree = 30

Fig. 9: IP messages generated by a flood versus the percentage of nodes reached. Router-level



(a) Average degree = 10 (b) Average degree = 20 (c) Average degree = 30

Fig. 10: Ratio of IP message reduction. Router-level

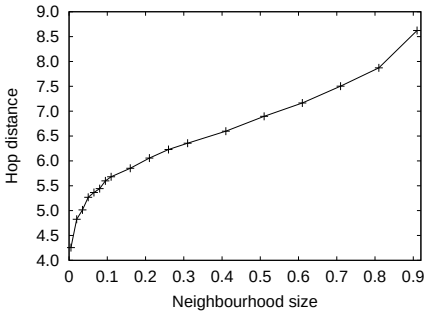


Fig. 8: Diameter (in hops) of different neighborhood sizes

D. Duplicate messages experiments

V. Experimental results

In order to verify the arguments made in the previous section, we performed several experiments comparing our system with Gnutella, at its peak usage population (approximately 2 million users) [4]. We performed the comparison with Gnutella 0.6, which employs a 2-tier architecture [11], 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.

We also compared our system latency-wise with the most recent algorithm we could find in the literature, proposed in [12] by Hsiao et al. Each peer in the proposed algorithm also samples the network, in a different function, to locate peers with lower latency in order to connect to. That number of samples is however relative to the natural logarithm of the total number of peers in the system. In our algorithm, the number of samples is constant, regardless of the size of the network.

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 [3], the company that 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 (short and long). We performed a large number of floods, in each experiment, with varying TTL values, resulting in a range of the ratio of the peers reached by the flood. For each TTL value we performed 100 floods and averaged the results. 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. First though, we prove that the injection of topology awareness in the overlay construction has not affected the "randomness" of the system.

The comparison with the algorithm proposed in [12] is made in terms of latency and the degree each algorithm affects the "randomness" of the system. In the context of that algorithm, we used a γ value of 2, the same value used in the paper for degrees around 15 or more.

The random nature of the constructed overlay is indicated by the extent of the reach of a flood for given number of messages. This is because on a clustered graph, as shown in Figures 2 and 3, duplicate messages appear even from the second hop of the flood. Since duplicate messages, by definition, arrive at a peer which has already received another flood message, they do not add to the reach of the entire flood. Figures 6(a), 6(b) and 6(c) show the similarity between the Gnutella overlay (random graph), the overlay constructed by ITA an the algorithm of Hsiao et al, with respect to flooding. The close fit of ITA and Hsiao et al. curves with the Gnutella curve on the two graphs shows that the flood reach is the same in all three graphs using the same number of messages. This means that ITA (and Hsiao et al.) 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. It should be noted here that latencies between neighbours in this experiment were modeled the same way as in the Latency experiments described next.

A. Latency experiments

In order to model the 200,000 by 200,000 latencies between our simulation peers, we obtained approx. 1000 real-world Vivaldi coordinates. Those 3D coordinates were produced by the Vivaldi project experiments on PlanetLab [7]. 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 5(a), 5(b), 5(c) show the values of the original Vivaldi coordinates as well as the distributions generated from our approximation distribution. 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 4 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.

Figures 7(a), 7(b) and 7(c) provide the experiment results for the first metric. They show the time it takes to flood the network, for 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 most 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 expires. On the other hand, even though the Hsiao et al. algorithm does reduce the time for a flood, compared to Gnutella, it still requires more time than ITA.

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. What is more important however, is the fact that it is common for a flood to still be active and being propagated in the network, even though no new results are (and are going to be) 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 and divide traffic load more evenly among routers (as we will show in the next section), but also reduces the build-up of queues in the router buffers.

B. IP layer experiments

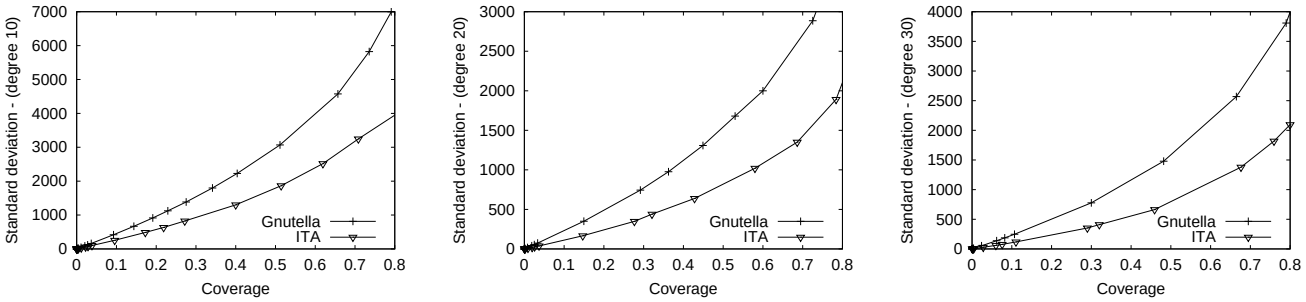
In this section we focus on the impact the ITA algorithm has on the IP layer. In order to perform simulations including the IP layer we obtained the latest trace of the router-level topology of the Internet from CAIDA [1]. This trace was publicly released in 2010 and it contais a much larger number of routers than the previous one. This trace initially contained approximately 33 million routers. However, we decided to remove the 1-degree routers (leaf routers) for two reasons. The first is the fact that performing simulations with this number of routers was time (and probably memory) prohibiting. In addition, the existance of leaf routers in the IP topology would not add to the accuracy of the simulation results. By pruning those routers, we ended up with the much more managable dataset of 1.2 million routers. This dataset, in addition to making simulations feasible, still retains the structure of the Internet intact. In addition, it still is about six times larger than the previous CAIDA dataset and hundrends times larger than most of the router graphs used in similar simulations in the literature we have described.

In order to be more thorough in the evaluation of our algorithm, we also performed the same number of experiments at the AS layer. We also obtained an AS-level graph from CAIDA. This dataset contained approximately 30.000 Autonomous Systems. By obtaining the number of subnets for each AS from the Internet Assigned Numbers Authority (IANA) [2], we were able to extract an AS population distribution, which we used to assign peers to each AS in our simulation.

In the IP layer experiments, both on the router and AS level, we used again 200,000 peers, each of which was randomly assigned to a router in the router-level graph, or AS in the AS-level graph. Since the CAIDA datasets do not contain latencies, we approximated the latencies with the number of IP hops between any two peers. Thus, each peer tries to form short links with those other peers whose routers are close to its own at 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 messages 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 8, 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.

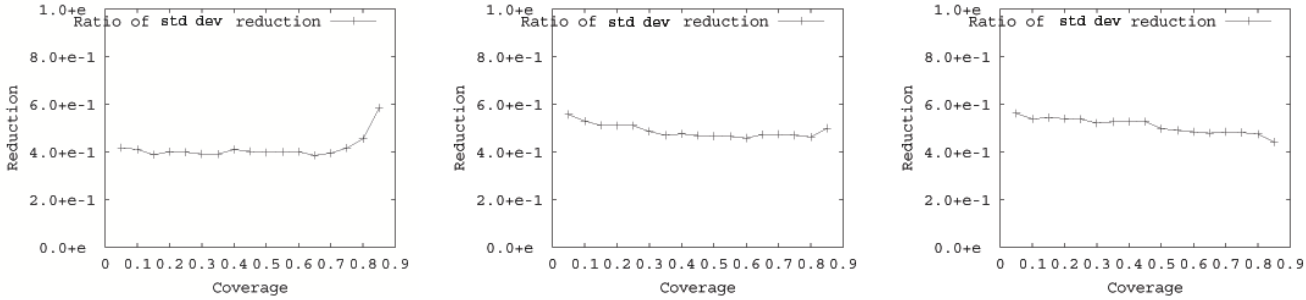
After running the simulations, which included performing several floods from random peers, with several TTL values, we obtained the following results on the router level: Figures 9(a), 9(b) and 9(c) illustrate the reduction in the number of IP messages for floods of various lengths. As one can see, the expected reduction that is observed is in the range of 15% to 25%. Figures 10(a), 10(b) and 10(c) show the reduction of the IP messages generated by ITA, compared to Gnutella. They show that, on average, 20% of the Gnutella IP messages, on the router graph experiments, are absent from the ITA experiments. Figures 13(a), 13(b) and 13(c) display the results of the similar experiments, albeit conducted in the AS level. The similarity of the results provide a good argument for their consistency.

Another important metric for the efficiency of any topology-aware overlay construction algorithm is the traffic load distribution across the routers in the system. ANy reduction in the total number of IP messages is of little use if the number of messages forwarded by a small number of (possibly core) routers remains unchanged. For this reason, we plotted the standard deviation in the traffic load of all routers, again for floods of different sizes. Figures 11(a), 11(b) and 11(c) show that ITA reduces the standard deviation approximately by 40% to 50% on the router level graph. Similarly, figures 12(a), 12(b) and 12(c) show the relative reduction in the standard deviation of router loads. This means that there is a reduction in the effect of bottle-necks in the network. The



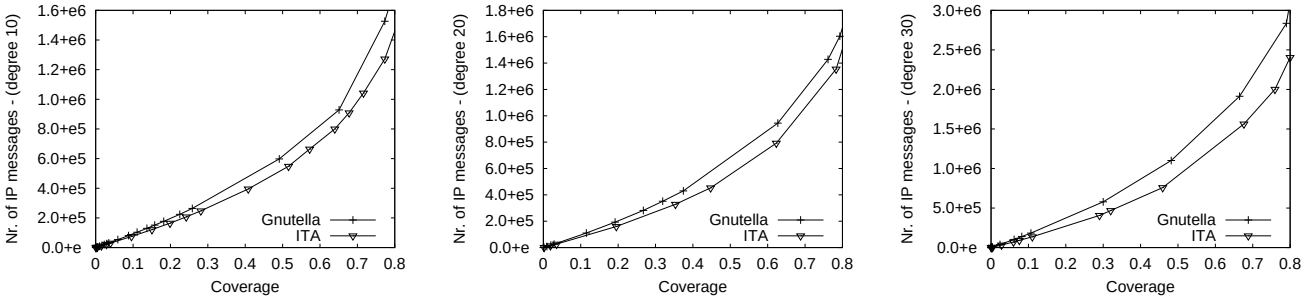
(a) Average degree = 10 (b) Average degree = 20 (c) Average degree = 30

Fig. 11: Standard deviation of router traffic loads versus the percentage of nodes reached. Router-level



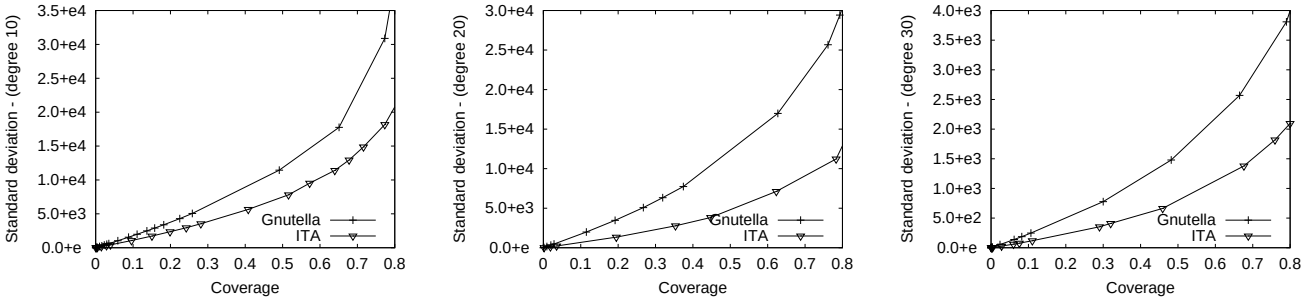
(a) Average degree = 10 (b) Average degree = 20 (c) Average degree = 30

Fig. 12: Ratio of standard deviation reduction. Router-level



(a) Average degree = 10 (b) Average degree = 20 (c) Average degree = 30

Fig. 13: IP messages generated by a flood versus the percentage of nodes reached. AS-level



(a) Average degree = 10 (b) Average degree = 20 (c) Average degree = 30

Fig. 14: Standard deviation of router traffic loads versus the percentage of nodes reached. AS-level

same experimental results on the AS level are presented in figures 14(a), 14(b) and 14(c). 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. 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. Finally, there is an additional reduction by approximately by half on the standard deviation of router loads.

References

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

[2] The internet assigned numbers authority (iana), http://www.iana.org/.

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

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

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

[6] 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.

[7] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: A decentralized network coordinate system. *SIGCOMM*, 2004.

[8] A. Dufour and L. 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.

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

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

[11] T. G. D. Forum. Gnutella 0.6 protocol specification.

[12] H.-C. Hsiao, H. Liao, and C.-C. Huang. Resolving the topology mismatch problem in unstructured peer-to-peer networks. *IEEE Transactions on Parallel and Distributed Systems*, 20:1668–1681, 2009.

[13] V. 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.

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

[15] N. Laoutaris, G. Smaragdakis, A. Bestavros, and J. Byers. Implications of selfish neighbor selection in overlay networks. *IEEE INFOCOM*, 2007.

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

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

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

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

[20] H. Papadakis, P. Fragopoulou, M. Dikaiakos, A. Labrinidis, and E. 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] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. 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] C. Rohrs. Query routing for the gnutella network. 2001.

[23] A. Rowstron and P. 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] G. Smaragdakis, N. Laoutaris, A. Bestavros, J. W. Byers, and M. 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] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A scalable Peer-To-Peer lookup service for internet applications. *Proceedings of the 2001 ACM SIGCOMM Conference*, pages 149–160, 2001.

[29] L. Yunhao, X. Li, L. Xiaomei, N. L. M., and Z. Xiaodong. Location awareness in unstructured peer-to-peer systems. *Parallel and Distributed Systems, IEEE Transactions on*, 16(2):163–174, 2005.

[30] 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.