Alleviating the Topology Mismatch Problem in Distributed Overlay Networks

Vassilis Moustakas, Hüseyin Akcan, Mema Roussopoulos, and Alex Delis, *Member, IEEE*

***Abstract*—Peer-to-peer (*P2P*) systems have enjoyed immense attention and have been widely deployed on the Internet for well over a decade. They are often implemented via an overlay network abstraction atop the Internet's best-effort IP infrastructure. *P2P* systems support a plethora of desirable features to distributed applications including anonymity, high availability, robustness, load balancing, quality of service and scalability to name just a few. Unfortunately, inherent weaknesses of early deployments of *P2P* systems, prevented applications from leveraging the full potential of the paradigm. One major weakness, identified early on, is the *topology mismatch problem* between the overlay network and the underlying IP topology. This mismatch can impose an extraordinary amount of unnecessary stress on network resources and can adversely affect both the scalability and efficiency of the operating applications. In this paper, we survey over a decade's worth of research efforts aimed at alleviating the topology mismatch problem in both *structured* and *unstructured* *P2P* systems. We provide a fine-grained categorization of the suggested solutions by discussing their novelty, advantages and weaknesses. Finally, we offer an analysis as well as pictorial comparisons of the reviewed approaches since we aim to offer a comprehensive reference for developers, system architects and researchers in the field.**

***Index Terms*—Distributed systems, peer-to-peer, overlay network, topology mismatch, topology awareness**

I. Introduction

PEER-TO-PEER (*P2P*) networks are self-organizing, distributed systems where participating nodes, called *peers*, act as both resource providers and resource consumers in contrast to the conventional *client-server* model where nodes undertake specific roles. For over a decade, *P2P* networks have been widely deployed and have enjoyed immense popularity from Internet communities, primarily because of the great number of features they offer to distributed applications built atop them. Such diverse features include: high availability and robustness, load-balancing, quality of service, scalability, decentralized administration, and anonymity.

The peer-to-peer paradigm gave impetus to two "killer" applications: file-sharing and Internet telephony. The *Napster* file-sharing system was widely acknowledged as the "fastest growing Internet application ever" in 2001 when it topped 26 million users sharing over 80 million songs. Other file-sharing applications followed suit, including *Gnutella* and *Limewire* enjoying 3 million concurrently-connected peers, as well as *BitTorrent* connecting over 150 million monthly users by January 2012 [1]. *P2P* telephony saw explosive growth with the advent of *Skype*. Since its introduction in 2003, *Skype* has become extremely popular with more than 650 million users in 2011 [2] and an astonishing 50+ million concurrently-online users [3]. Moreover, it has consistently eroded the traffic handled by traditional telephony carriers by slicing away a staggering 214 billion minutes of pertinent traffic in 2013 alone [4].

Apart from the above, a startling number of diverse and successful applications have been built based on *P2P* architectures. Some of them include: distributed search engines [5], distributed data-storage systems [6], [7], [8], [9], [10], [11], [12], [13], Web caches, archives and publishing systems [14], [15], [16], [17], messaging and dissemination applications [18], [19], event-notification infrastructures [20], [21], [13], naming services [22], censor-resistant stores [23] and lately, even cloud-based platforms [24].

P2P systems are implemented using *overlay networks*. An overlay network, is a virtual system of nodes featuring logical interconnects (or links) created above an existing network; overlays provide an abstraction that enables the implementation of efficient, fully distributed, application-layer services such as routing messages to destinations that are not known in advance or offering QoS guarantees (i.e., in content-distribution) over best effort infrastructures. Overlay nodes communicate through *virtual connections* each of which may correspond to a path of possibly many physical links in the underlying network. Figure 1 illustrates a simple four-node overlay constructed over a wide-area network.

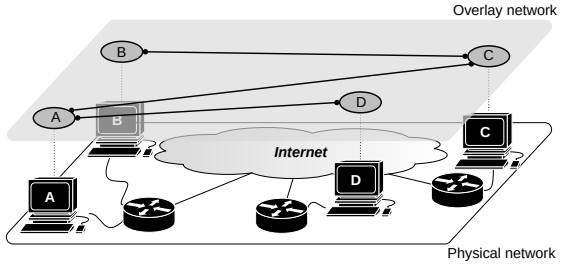


Fig. 1. An example overlay network.

The single key issue that determines the efficiency of an overlay network, is how well the overlay maps to the underlying network topology on which it "rests". Consider two nodes[1]

Vassilis Moustakas, Mema Roussopoulos and Alex Delis are with the Department of Informatics and Telecommunications, *University of Athens*, GR15784, Athens, Greece, e-mails: {*vsmoustakas,mema,ad*}*@di.uoa.gr*

Hüseyin Akcan is with the Department of Software Engineering, *Izmir University of Economics*, 35330 Balcova, Izmir, Turkey, email: *huseyin.akcan@ieu.edu.tr*

Manuscript received April 30, 2015; revised April 30, 2015.

[1]In *P2P* networks, the participating nodes are typically user-PCs operating at the edge of the Internet.

that are connected with each other via a path of overlay links. If the application running on the nodes, generates heavy traffic along the overlay path, it would be beneficial to construct the overlay topology in a way that the number of underlying IP links between these two nodes is minimized. Should the overlay network be constructed so that it does not match the underlying topology well, the inherent *topology mismatch* gives two major problems. First, the performance of the application per se, can be adversely affected since traffic must flow over a larger, redundant, number of physical hops resulting in poor user experience entailing noticeable latencies or jittering. Second, other applications running on the underlying network infrastructure may be adversely affected as well. Studies have shown that highly popular *P2P* applications contribute the largest portion of the overall Internet traffic [25], [26], [27], with some reporting that more than 60% of this traffic to be *P2P*-related [28], [29]; it was also projected that this traffic would reach 7 *Exabytes*-per-month by 2018 [30]! Evidently, this constitutes a major burden for Internet Service Providers (ISPs) who must route all of this traffic to destinations at the edge of the Internet. If the *P2P* overlay topology is poorly designed, the demand on the Internet's backbone infrastructure may substantially increase as traffic might have to flow "back and forth" several times between two neighboring ISPs while trying to travel from the source to its destination node in the overlay. Hence, it is critical that *P2P* networks be laid out in ways that their topology matches, as closely as possible, the underlying IP topology.

For over a decade, researchers have extensively investigated various aspects of the topology mismatch problem. The main objective of this paper is to offer a comprehensive survey of the work carried out in the area and provide a taxonomy of the proposed solutions. We point out synergies, as well as similarities and differences in the published approaches. Ultimately, our goal is to help readers sift through the voluminous literature, to help them understand the advantages and disadvantages of each work, and to provide them with enough perspective so that when the need arises, they are able to select, amongst the different approaches, the one that is most suitable for their particular application. The rest of the paper is organized as follows: Section II provides background on overlay architectures including centralized, decentralized-unstructured and decentralized-structured *P2P* systems. We also formally define the problem of topology mismatch and offer the rationale behind our evaluation of the techniques discussed in this paper. Sections III and IV respectively outline the surveyed research efforts for decentralized-unstructured and decentralized-structured *P2P* systems. Conclusions can be found in Section V.

II. Background

In this section, we provide some basic background information on the types of *P2P* architectures that have been implemented and deployed. We also describe and motivate more formally the topology mismatch problem whose various aspects have been tackled over the past several years.

A. Peer-to-Peer (P2P) Overlay Architectures

Circa 2000, the first file-sharing *P2P*-system introduced the *servent* concept [31], a *portmanteau* that blends the notions of server and client to denote the dual role of a node participating in the *P2P*-network. Such server-less systems have proven to be able to achieve outstanding aggregate resource capacities as more and more participants join the system without requiring additional expenditure for infrastructure. While certain undesirable features such as free-riding [32], [33], [34], distribution of illegal content and other *socio-technical* issues [35] have arisen, *P2P* systems have continued to gain popularity throughout the years primarily due to their successful file-sharing applications amongst *vast* numbers of participating users. For over a decade, excitement over the immense potential of *P2P* systems has generated a flurry of research and development, resulting in a wide range of popular protocols, networks, and applications. As *P2P* systems evolved, three main architectures have emerged, namely: *i*) *centralized*, *ii*) *decentralized unstructured*, and *iii*) *decentralized structured*.

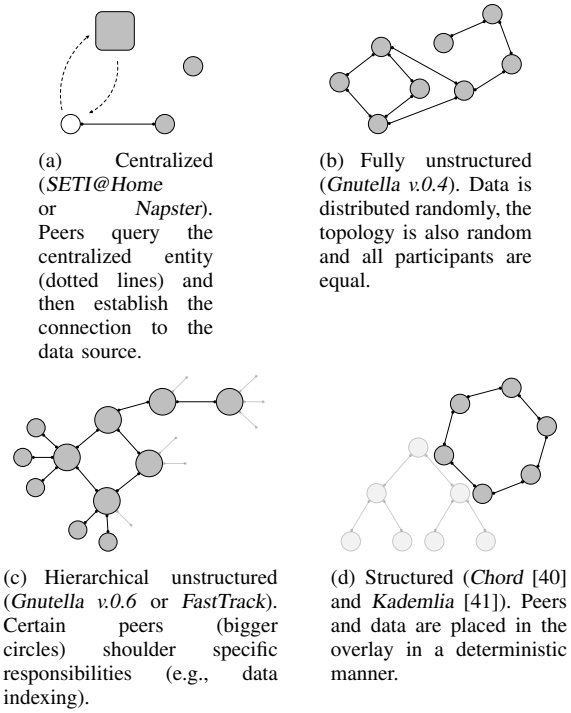
Designers of *centralized* architectures (Figure 2(a)) were the first to observe that requests for resources (e.g., CPU cycles or popular content) need not be sent to a dedicated server. Instead, such requests could be handled by many hosts that already possessed the resources in question. The downside of this approach was that it required a centralized search directory that inevitably became a single point of failure, a scalability bottleneck or a target of malicious behavior such as *DoS* attacks.

Napster was the most successful incarnation of the centralized approach in file-sharing [36]. It maintained a *central index server* based on file-lists that were peer-provided. Users would query the index server that would ultimately yield pointers to the actual content. Thus, by centralizing search while distributing downloads, *Napster* achieved a highly functional design that, at its height, attracted 26 million users sharing approximately 80 million songs[37]. The central indexing component played a vital role in the demise of *Napter*, as the *Recording Industry Association of America* successfully argued in court that this software entity was enabling copyright infringement of the artists' songs exchanged through the system.

Architectures that distributed both search and resource provision capabilities (Figure 2(b)) soon followed. In these *decentralized* approaches, resource placement within the overlay topology is random [38]; for this reason, such architectures are referred to as *unstructured*. Key properties of such *P2P* systems are that they support inherent heterogeneity of peers, are highly resilient to peer failures, and incur low maintenance overhead at handling the dynamics of peer participation [39]. Unstructured systems are also known as *broadcast-based* systems for they use message flooding among peers to propagate search queries. In its first realization, *Gnutella*(*v.0.4*) became a well-known example of the fully decentralized unstructured approach.

A key disadvantage of the unstructured approach is that message flooding burdens the network since queries travel

within the network randomly, visiting nodes that do not have the sought resource and thus wasting bandwidth. To save on bandwidth, unstructured networks typically limit how long a query travels within the overlay via the *time-to-live* parameter (*TTL*). When a query is received by a node, it decreases its *TTL* by one unit. If the *TTL* has not reached zero, the node forwards the query to its neighbors. Otherwise, it drops the query. While this approach prevents queries from visiting the entire network, it does not guarantee that the resource of interest, (e.g., a rare recording) will, ultimately, be found.



(a) Centralized (*SETI@Home* or *Napster*). Peers query the centralized entity (dotted lines) and then establish the connection to the data source.

(b) Fully unstructured (*Gnutella v.0.4*). Data is distributed randomly, the topology is also random and all participants are equal.

(c) Hierarchical unstructured (*Gnutella v.0.6* or *FastTrack*). Certain peers (bigger circles) shoulder specific responsibilities (e.g., data indexing).

(d) Structured (*Chord* [40] and *Kademlia* [41]). Peers and data are placed in the overlay in a deterministic manner.

Fig. 2. The various *P2P* architectures.

To shorten search times, improve scalability, and lighten network load, unstructured configurations evolved into *hierarchical* systems that dynamically assigned indexing functions to special peers, called *ultra-* or *super-peers* (Figure 2(c)). Peers with reliable network connections and/or high compute power became *ultra/super-peers* facilitating less-powerful *leaf-nodes* to find resources of interest. Examples of such hierarchical unstructured systems include *Gnutella*(*v.0.6*)/*Limewire* [31], *KaZaA* [42] and *Skype* [43]. Although the hierarchical approach relieves network pressure as search queries were flooded over the much smaller subset of ultra/super-peers, it fails to address the reduced search scope and inability to locate rare resources.

Decentralized structured schemes were introduced as an answer to the above problems of decentralized unstructured *P2P*–approaches (Figure 2(d)). Here, the main objective was to offer a self-organizing infrastructure for large-scale *P2P* applications [44], [40], [45], [46], [41], [47]. To achieve that, they implement a *Distributed Hash Table (DHT)* that maps objects to nodes through a deterministic mechanism. *DHT*s provide a guaranteed bound on the number of overlay routing hops that have to be "traveled" to locate a resource, even in the case when only a *single* copy exists within the system. This requires $O(log(n))$ hops, compared to the unstructured networks that require $O(n)$ to reliably locate any object. Unfortunately this efficiency does not come without cost. Structured overlays can only support exact-match queries (i.e., queries that identify resources by name) as opposed to content-based retrieval provided by unstructured overlays. This means that unstructured overlays can support more versatile resource location queries such as keyword and partial matching searches. Several popular file-sharing *P2P* applications have been based on *DHT*s including *Overnet*/*eDonkey* [48], *Kademlia*/*Kad Network* [41], and some flavors of *BitTorrent* [49]. Moreover, *DHT*s sparked a flurry of research for many, diverse, applications ranging from distributed search engines to event-notification and cloud-based platforms [6], [20], [24].

B. The Topology Mismatch Problem

One of the major issues that defines the efficiency of an overlay network is its mapping to the underlying physical infrastructure. Recall, in Figure 1 nodes A and B are on the same local network, while C and D are in different networks. The top layer represents the overlay interconnection formed by these four nodes at a higher level. Link connections there, can change as needed by the running environment, without any particular constraint by the underlying physical topology. For instance, assume nodes A, B, C and D being connected via the physical network shown in Figure 3 in which costs are depicted in milliseconds. When peer A sends a message

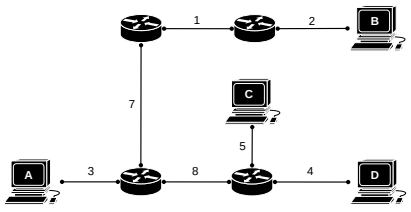
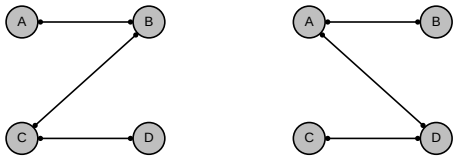


Fig. 3. Interconnection of nodes in the physical level.

its D counterpart and depending on the overlay configuration, nodes will experience different performance. For example, in the overlay depicted in Figure 4(a), the message will traverse the following sequence of links in the physical layer (marked with their costs): $3 \rightarrow 7 \rightarrow 1 \rightarrow 2 \rightarrow 2 \rightarrow 1 \rightarrow 7 \rightarrow 8 \rightarrow 5 \rightarrow 5 \rightarrow 4$. In the alternative overlay of Figure 4(b), the path will be: $3 \rightarrow 8 \rightarrow 4$. The respective costs are $45ms$ and $15ms$. Evidently, the second overlay is more congruent with the underlying physical network in comparison to the first and, thus, is more efficient.

Ideally, an overlay should be constructed to achieve an *optimal* mapping with its underlying network links, to avoid inefficient states where redundant physical resources are utilized for the operation of the same virtual link in the overlay. The ratio of the actual *IP* network distance a message travels to reach an object of interest (via overlay routing) to the minimal *IP* network distance to that object (i.e. through *IP*) is known as *stretch* or *Relative Delay Penalty (RDP)* [50].



(a) Nodes B and C have direct overlay connection.

(b) Nodes A and D have direct overlay connection.

Fig. 4. Two different overlay connection configuration.

The problem of constructing an optimal overlay is referred to as the *topology mismatch problem*, and is formally defined as follows:

Definition 1: Let V=$\{v_1, \ldots, v_n\}$ be a set of points denoting the network nodes, $\{v_i, v_j\} \in E$ be the set of unicast distances between nodes v_i and v_j, $G = (V, E)$ be a complete distance graph over V. The topology mismatch problem is to construct a minimal spanning tree, where node degree is restricted to a constant ($k \geq 2$) by the bandwidth of each node v_i.

Early overlay protocol implementations did not pay much attention to map optimally to the underlying network topology. *Gnutella*, for example, is considered far from scalable [51] for every peer chooses its neighbors without any knowledge of the underlying network, resulting in mismatches. Queries are flooded over multiple paths in an effort to reach the node(s) having the sought content. In addition, pairs of nodes may simultaneously forward the same query to each other in a superfluous mode of operation in this opportunistic and without coordination environment. [52] showed that, even when 95% of any two nodes are less than 7 hops away from each other, a *flooding-based Gnutella*-like algorithm can generate 330TB/month traffic in a $50,000$ node network. A topology mismatch can thus have a serious effect on Internet's routing performance.

Similar problems are observed in decentralized structured schemes as well. Typically, node IDs are assigned *randomly*, resulting in excellent load balancing, scalability, and robustness for the overlay. Unfortunately, this randomness has a negative impact on the *routing locality* of the network. This means that even though the target node can be reached with a logarithmic number of overlay hops, the distance traveled in the underlying network, during the overlay routing process, can be far from optimal.

C. Motivation and Goal for this Survey

A topology–unaware overlay network is able to control the sequence of peers a message traverses before reaching its destination, but it completely ignores how the actual packets are switched at the underlying infrastructure along the overlay path. In particular, a single logical point-to-point link on the overlay typically corresponds to multiple physical links in the underlying layer. Moreover, a link in the physical network often lies on several overlay paths inadvertently increasing the traffic on that link; this is known as the link's *stress* [53]. There is also the stochastic phenomenon during which peers depart and others arrive in the network, known as *churn* [39]. Even when an approach offers the best possible location when a node joins in, the churn is set to continuously undermine the effort to maintain the best possible position of the peer in the network.

The topology mismatch problem is exacerbated by the extremely large amounts of traffic generated by highly popular *P2P* applications [54]. Addressing this problem remains a fundamental challenge in contributing to the overall efficiency of the Internet. Unfortunately, topology mismatch is known to be an *NP-Hard* problem [55], [56]. There is also additional complexity due to the fact that end-to-end latencies demonstrate *triangle inequality violations (TIVs)* which are a consequence of the Internet's structure and routing policies; as such, they will remain a property of the Internet for the foreseeable future [57]. *TIV*s affect both network coordinate [58], [59] and positioning [60] systems and make the construction of *IP*-topology-aware overlays difficult.

Over the past several years, researchers have investigated an extensive array of heuristic solutions to the topology mismatch problem. The purpose of this survey is to gather and organize this extensive body of produced knowledge. We aspire to assist the reader in better understanding the different approaches suggested and to accomplish this, we present methods for both structured and unstructured distributed overlay networks. We outline the respective algorithmic aspects and we categorize the surveyed material so that pros and cons of each method can become apparent in their context.

We employ both tabular and pictorial means to articulate features of the proposed methodologies and outline both strengths and weaknesses. To ascertain the effectiveness of each proposal, we predominantly use three criteria that capture the behavior of the algorithms discussed: *efficiency*, *overhead* and *scalability*. For each of these criteria, we use a coarse-grained, three-level, gradation to denote how well the algorithms do, namely, *low*, *medium* and *high*. Each criterion has its own interpretation of the above literal values. *Efficiency* means how well the algorithm performs its search and topology adaptation functions. Our assessment also considers any negative effects that might come along with the application of the algorithm on intrinsic characteristics of the system (i.e., search scope coverage, download performance etc.).

For the *overhead* criterion, we evaluate the cost of computing and communicating peer join, message forwarding, and overlay topology maintenance. For instance, the overhead that an algorithm, which needs global knowledge of the overlay topology, imposes to the system is greater compared to a protocol that only relies on peers exchanging routing tables within ~~a~~ a 2-hop radius from each other.

For the *scalability* criterion, we consider system behavior with respect to the number of participating peers it can reliably support. We characterize protocols as being lowly, mediumly or highly scalable if the are proven to effectively support hundreds, tenths of thousands or millions of nodes, respectively.

III. Unstructured *P2P* Networks

In this section, we present algorithms that tackle the topology mismatch problem in unstructured *P2P* networks. We

classify them based on their use of the overlay structure, their message forwarding scheme for peer communication and the techniques they use for detecting proximity.

A. Algorithms for Unstructured Architectures

To improve over *Gnutella*'s "blind flooding" approach, [38] proposed a practical and easy to implement solution weaved around three different message forwarding methods, namely *iterative deepening*, *directed BFS* and *local indices*.

In *iterative deepening*, the search is performed on a BFS tree with multiple preset depths. The depth limit is iteratively increased by the source node for each query, based on the quality of the results. The source node may issue a new request with increased depth limit that will trigger the nodes at the last visited depth level to resume the search thus avoiding restart of the entire search process and consequently reducing the load on the nodes of the upper levels of the tree. Its major drawback is the delay between successive iterations, as the source-node has to examine the results at each attempt before deciding to either quit or "unfreeze" the query.

The *directed BFS* tries to avoid the aforementioned delay by forwarding query messages only to a selected subset of available neighbors. The selection criteria varies, from the number of results received or the distance (hops) traveled to locate these results, to the bandwidth, or the query load of the neighbor. That way, fewer nodes are visited but quality of query responses is maintained, to a large degree, on the premise that the selected heuristics can direct the search to the right path. In *local indices*, each node indexes data hosted by neighboring nodes within a radius of r hops and uses this local index to answer queries on behalf of other nodes. That way, local indices, greatly reduce the aggregate bandwidth usage of the network and improve query efficiency. However, updating such indices in the presence of frequent node joins/departures may introduce significant overhead should the radius is kept broad. Below, we outline how the techniques in [38] fare against the ~~3~~ major performance criteria of Section II.

	Efficiency	*Overhead*	*Scalability*
Iterative Deepening	low	low	medium
Directed BFS	medium	medium	medium
Local Indices	high	medium	medium

A *delay–aware P2P* system termed *DAPS* was introduced in [61]. *DAPS* seeks to attain reduced look-up times by dividing peer routing tables into several sectors of increasing delay. The source node that issues the query designates the delay boundary it may tolerate, providing so a *pruning factor*. In this context, user requests are forwarded only to nodes whose expected delay is less than or equal to the indicated boundary. *DAPS* primarily focuses on user "experience" and deploys an end-to-end delay monitoring mechanism that may enable the clustering of routing tables. In ~~an~~ dynamic environment, the formation of very accurate such routing tables might be however an elusive goal. In terms of the 3 performance criteria, *DAPS* stands as follows:

Efficiency	*Overhead*	*Scalability*
low	low	medium

The key objective of the *Gia* system [62] is to help alleviate the scalability omnipresent in unstructured *P2P* file-sharing systems. At first, *Gia* replaced *Gnutella*'s blind flooding with random walks. Although this adoption was a step in the right direction [63], issuing a single copy of the query within the network effectively reduces the search scope and may negatively affect the success rate of the query in question. To overcome this limitation, *Gia* introduces a token-based flow control mechanism that gradually redirects queries to nodes which are more likely to answer. This flow control mechanism also helps prevent node overloading as each peer "announces" the number of query requests it can handle, in terms of tokens, to its neighbors; to this end, peers only forward query requests to nodes that they previously received tokens from. Further refinement to the search mechanism is the support for *one–hop replication* of pointers to content. *Gia* also acknowledges the heterogeneity in peer bandwidth, processing power, disk speed, etc., of *P2P* nodes and uses this information when connecting nodes to each other. By using a topology adaptation algorithm, *Gia* places low capacity nodes within short proximity to peers with high performance features. This topology adaptation algorithm is based on the metric each node maintains about its satisfaction –ranging between 0 and 1– for the neighbors it finds itself associated with. Through message exchange, a peer can establish new connections or drop superseded ones in order to improve its satisfaction. Despite the fact that *Gia*'s topology adaptation algorithm that primarily focuses on the satisfaction of peers improves the network's scalability, it falls short in addressing the mismatch problem as considerations for the underlying network are not handled in explicit terms [64]. Moreover it reduces the search scope, exhibits poor performance in the worst case when matching data is not found quickly [65], [66] and can potentially build disconnected topologies [67]. In terms of the 3 stated criteria, *Gia*'s approach is as follows:

Efficiency	*Overhead*	*Scalability*
medium	low	medium

Another problem with topology unaware systems is the duplication of messages due to cycles in the network graph. Interestingly, such cycles appear even along the correct forwarding path. [68] focuses on this exact deficiency of overlay networks and introduces the *Distributed Cycle Minimization Protocol (DCMP)*, a dynamic, fully distributed method that removes cycles; this is accomplished without sacrificing overlay connectivity, resilience and other key properties of unstructured *P2P* architectures and by avoiding a hierarchical organization of peers. Once a cycle is detected in *DCMP*, the most powerful node in that cycle is elected as the *Gate Peer* and the cycle is then broken in that place that it will result in the minimization of the distance between the *Gate Peer* and all other nodes that are currently part of the cycle. This process is managed by using two specialized message types namely, *Information Collection Message (ICM)* and *Cut Message (CM)*. *DCMP* bases its operation on messages whose travel is limited by an imposed *TTL* value (max set to 7 for most cases). This inherently limits the protocol in detecting cycles that span for more than *TTL* nodes. Even though cycle elimination does improve network performance, it cannot directly contribute to the solution of the topology mismatch problem, while its

performance in a high traffic and churn environments and its ability to retain important characteristics of a *Gnutella*-like topology (i.e., like degree distribution, average peer distance, diameter e.t.c.) has been doubted [69]. On the other hand, even though some of the overhead saved by the duplicate message reduction is spent to control the cycle cutting mechanism it is reported that still stays for example one to two orders of magnitude less than LTM due to its "lazy" broadcasting of control messages (as opposed to LTM's periodic approach) [68]. The expected *DCMP* behavior as far as our 3 criteria is:

Efficiency	*Overhead*	*Scalability*
medium	low	high

In [70] and [63], replication is used as a way to improve inefficient blind search. The observation is that as the number of replicas increases in the network, it would be relatively easy to locate items even if search is random.

The work discusses 3 approaches for replication allocation namely *uniform*, *proportional*, and *square-root replication*. In the uniform model, all objects are replicated without taking into account the query distribution, while in the proportional, objects are replicated analogously to their query rate, so that frequently queried items can be found more often. Experimental outcomes indicate that the uniform allocation minimizes the maximum search length and so it can reduce the time spent on unsuccessful searches. The proportional strategy effectively decreases the search time for popular items, but suffers when needing to locate the rare entities. When it comes to expected successful search size, it is the same for for both uniform and proportional models and any approach in between to always behave much better. Finally, the square-root model is an attempt to find the golden ratio between uniform and proportional allocation models. It suggests setting the number of object replicas to the square root of the searching rate for an object[2]. Theory in [70] suggest that square-root minimizes the expected search size on successful queries, claim that is supported by simulations [63].

In [63], a *k-walker* query algorithm is presented and shown that using it along side square-root allocation results in performance similar to *Gnutella*'s query flooding method but incurring up to two orders of magnitude less network traffic. Also checks 3 strategies for replica placement, namely *owner replication* where an object is replicated at the requesting node, *path replication* that caches the object along the path from the requesting to the providing node and *random replication* where the replication is taking place randomly on visited nodes. Observations in this work are not directly applicable to structured DHTs, because it is assumed that the lookup time for an object depends only on the number of replicas and not the placement strategy [72].

The table below outlines how the 3 allocation approaches fare:

	Efficiency	*Overhead*	*Scalability*
Uniform Replication	low	low	medium
Proportional Replication	low	medium	low
Square Root Replication Allocation	medium	medium	medium

Narada [50], [73], [53] is a generic protocol for creating self-adapting overlay networks capable of application-layer multicast communications without requiring IP multicast infrastructure at the network layer. Although IP–multicast would present a choice in implementing *Narada*, it is in general considered that it violates the stateless design of the current Internet. Despite the fact the *Narada* was not designed as *P2P* system per se, it was the first (along with *Scattercast* [55]) to consider the feasibility of overlay-based, application-layer services over the Internet while taking into account bandwidth and latency properties of the underlying physical infrastructure. In the context of this work, the inefficiency caused by the topology mismatch ~~and~~ attempted to be tackled by building an highly connected graph termed *mesh* with each source featuring its own minimum spanning tree. A gossip-protocol was deployed for the creation of these spanning trees [74]. Both graph and trees are dynamically updated as nodes keep joining or departing the network. The protocol aims to ease the physical link stress, the overall resource usage as well as the relative delay among end-systems. Unfortunately, the main limitation for *Narada* is that although it works reasonably well for small groups, it does not scale well for larger networks. Hence, it is not a suitable choice for potentially large *P2P* networks:

Efficiency	*Overhead*	*Scalability*
medium	high	low

Along with *Narada*, the *Adaptive Overlay Topology Optimization (AOTO)* [75] is one of the first attempts to address the topology mismatch problem. *AOTO* is a distributed algorithm that seeks to optimize the usage of the underlying physical resources and operates in 2 phases: *Selective Flooding* and *Active Topology*. In *Selective Flooding*, a minimum spanning tree is built for each peer and its immediate neighbors so that queries do not flood the entire network and at the same time preventing their search scope from shrinking. This way some neighbors become non–flooding. During *Active Topology*, each peer replaces independently such non–flooding neighbors, with closer nodes as an attempt to revise the overlay links so that they can reflect more closely the physical network topology. Picking a replacement out of these non-flooding neighbors follows a random policy called *Randomized AT* algorithm. To accomplish the above actions, a peer has to keep track of its communication costs with all its neighbors (e.g., network delays) as well as the costs between any pair of neighbors. The *randomized AT* algorithm is applied by a source peer every time its neighbor list is updated or an updated neighbor cost table is received. *AOTO*'s performance regarding the 3 criteria is as follows:

Efficiency	*Overhead*	*Scalability*
medium	high	medium

The *Adaptive Connection Establishment (ACE)* approach [76] uses the network delay as a metric to estimate the costs between nodes. Each peer computes the costs to its logical one-hop neighbors and forms a *neighbor cost table*

[2]An interesting application of the square-root principle has also been explored in the context of topology reconstruction, by ensuring node degree to be proportional to the square-root of its content popularity [71].

(NCT) using a special routing message type. Any pair of neighboring peers exchange their *NCT*s and so a minimal overlay topology can be formed. Based on the obtained *NCT*s a minimum spanning tree for each peer and its one-hop neighbors is created. Finally, neighbors located physically far away are replaced by physically-closer counterparts. In particular, a peer S iteratively probes the distance d between itself and every of its non-flooding neighbor nodes G as well as the distance between S and G's neighbors designated as H. If $d_{SG} > d_{SH}$, then link SG is replaced by link SH. If, on the other hand, $d_{SG} < d_{SH}$ then also d_{GH} and then we have the following options. Either $d_{SH} < d_{GH}$ in which case link SH is kept or $d_{SH} > d_{GH}$ in which case S will keep probing other G's direct neighbors. The above optimization is conducted within 1-neighbor closure using as base, a peer and checking all its direct neighbors. Evidently the scope of such an operation could be extended. Should a larger scope be used, a better topology matching can be obtained at a greater computational overhead.

Efficiency	*Overhead*	*Scalability*
medium	medium	medium

Location-aware Topology Matching (LTM) [77] seeks to optimize an overlay *P2P* structure based on the physical topology. To achieve this, peers issue special messages called *TTL-detector*s whose *TTL* values are 0 or 1; in this regard, peers discover 1- and 2-hop neighbor sets, designated as N and N^2 respectively, and proceed to compute communication costs. Time-stamps are used to derive network latency measurements that are then used to improve the overlay network without sacrificing the search scope. Each node compares the latency information received from its direct neighbors; peers with longer latencies are placed on a *will-cut* list where they remain for a certain period of time after which they are finally eliminated and are placed on the peer's *cut-list*. Thus, low-productivity connections are dropped and replaced by more efficient ones, reducing the latency on the overall network. Although *LTM* improves the overall efficiency of the *P2P* network, it is unable to offer any warranty for effectively addressing the mismatch problem. *LTM* fares as follows regarding our 3 criteria:

Efficiency	*Overhead*	*Scalability*
medium	medium	medium

The *Scalable Bipartite Overlay (SBO)* [78] reduces the overhead of creating and maintaining a minimum spanning tree by randomly dividing the nodes into two groups –*red*s and *white*s– and assigning different tasks to the two groups. When a peer joins the network, it is randomly assigned with an initial color (red or white). Then the network bootstrap host node furnishes the joining peer with a list of active peers along with their color information. The joining peer uses this list to establish connections to differently colored peers. In this regard, all peers form ~~a~~ a bipartite overlay. Using the network delay as a metric, white-peers ~~peers~~ measure distances from red counterparts and report the encountered red neighbors. Having information on all their 2-hop neighbors (N^2) red-peers create minimum spanning trees for the neighbors in question and assign efficient 2-hop forwarding paths. This process can render a white peer E non–forwarding neighbor of the red peer P. Direct neighbor replacement is a process conducted by E in order to replace P with a $P' \in N^2(P)$ as its new neighbor. This adaptation tackles the topology mismatch problem by reducing message duplication and shorten response times caused by the problem identified as *Revisit Not known (RN)*. RN is the situation when a node is visited some times as a relay stop before it is visited as an overlay peer, thus creating duplicate unnecessary messages to the network and delaying query responses. Even though this attempt is based on a simple and elegant solution it is reported to not guarantee performance in terms of average communication delay or search scope [79]. In regards to the 3 stated criteria, *SBO* behaves as follows:

Efficiency	*Overhead*	*Scalability*
high	medium	medium

The distributed heuristic termed *Two-Hop-Away Neighbor Comparison and Selection (THANCS)* [80], [81] attempts to minimize overlay hop costs. *THANCS* is essentially a *local search method* as it aims to find a locally optimum solution by exploiting knowledge within a 2-hop radius. The algorithm consists of two main components: *piggybacking neighbor distance on queries* and *neighbor comparison and selection*.

The *piggybacking component* requires peers to probe immediate neighbors using delay distance measurements and store this information locally. The algorithm introduces a special query message type, the *Piggy Message (PM)* which includes information about the neighbor identification (*IP*) and measured distance (d). PMs are piggybacked to normal *Gnutella* messages. When a node P receives a query from Q, it constructs a PM_{PQ}, piggybacks it to the query and forwards it to P's neighbors. As soon as a neighbor detaches a received PM_{PQ}, records the distance d_{PQ} and processes the query. Selection of which incoming queries should be piggybacked with a *PM* is determined using either a *pure probability-based (PPB)* or a *new neighbor triggered (NNT)* policy.

The *neighbor comparison and selection* component defines that a peer S probes all his 2-hops away neighbors (a set denoted as $N^2(S)$) not probed thus far. Let P be a 1-hop distance neighbor of S, and Q be a probed peer by S. When S receives a message piggybacked with message PM_{PQ}, the following 2 cases are identified:

1) If Q is already a direct neighbor of S then we check the distances d_{SQ}, d_{SP} and d_{PQ}. If either d_{SQ} or d_{SP} is the longest, then the longest link will be placed in a *will-cut* list[3]. If d_{PQ} is the longest, then nothing is done by S; being fully distributed, neighbor comparison and selection process, will have the opportunity to handle PQ link when initiated by peers P or Q.
2) If Q is strictly a 2-hop neighbor of S and have never probed Q in the past, stores distance d_{SQ} and checks distances d_{SQ}, d_{SP} and d_{PQ}. If d_{SQ} is the longest, S will not create link SQ. If d_{SP} is the longest, SQ will be created and SP will be put into the will-cut list. If

[3]A peer will not send or forward queries to connections in its will-cut list but will preserve them for some time in order not to effect ongoing response messages travelling the inverse path.

PQ is the longest, links SP and SQ will be preserved expecting that P or Q will disconnect PQ later.

As discussed above, *THANCS* is fully distributed and needs only minimum knowledge of, at most, a 2-hop distant peers. This features renders the method a good candidate for large overlay networks. Also its topology adaptation helps construct a well fit overlay with respect to the underlying network–at least with respect to network distances. In addition, will-cut list and distance caches (which store already probed peers) minimize the unnecessary messages for the network maintenance. In [82], the simulation-derived results of *THANCS* as far average broadcast delay and overlay improvement guarantees are discussed; overall, *THANCS* fares as follows regarding the 3 stated criteria:

Efficiency	*Overhead*	*Scalability*
medium	low	high

The *Hops Adaptive Neighbor Discovery (HAND)* algorithm [83] uses a fully-distributed triple–hop adjustment strategy, applied to a network graph G in order to create an optimal graph G^*, free of the implications of topology mismatch. This optimality is attained if all peer hop sequences $(v_1, v_2, \ldots, v_k)$ of G exist in G^* and are in the same order. The latter indicates that in practice triple sequences (v_1, v_2, v_3) are used. A mismatch is detected as follows: should we want to verify a sequence, say $v_2 - v_1 - v_3$, two probing messages are dispatched from v_1 to v_2 and v_3 and yield delays of x and z respectively. When the probing message arrives at v_2, it gets forwarded directly to v_3. Similarly, the message reaching v_3 is directly forwarded to v_2. The above two forwarding actions seek to quantify the corresponding delays of (v_2, v_3) and (v_3, v_2) for the physical path y. If $y = z - x \pm \varepsilon$, the sequence $v_2 - v_1 - v_3$ is mismatched and should be adjusted to $v_1 - v_2 - v_3$ by deleting edge (v_1, v_3) and then adding a new (v_2, v_3). If $y = x - z \pm \varepsilon$, sequence $v_2 - v_1 - v_3$ is mismatched and should be adjusted to $v_1 - v_3 - v_2$ by deleting edge (v_1, v_2) and adding a new (v_3, v_2). The ε is a small positive real number denoting additional delays caused by possible forwarding and jitter delays. The advantages of *HAND* are that it *i*) does not need any clock synchronization, *ii*) is a fully distributed algorithm. *iii*) involves low traffic overhead, *iv*) can be used in dynamic *P2P* environments, and *v*) furnishes low query response times. In terms of the 3 stated criteria, *HAND* fares as follows:

Efficiency	*Overhead*	*Scalability*
medium	low	high

The *Adaptive Peer Selection (APS)* approach uses machine learning techniques to form peer selection strategies based on past experience [84]. A decision tree is used to rate peers based on information collected for the characteristics of established connections. Such features include connection link load, bandwidth, and past uploading experience. Subsequently, a Markov decision process is used to shape the policy for switching among the peers. The success of approach depends on how fast the introduced peer selection functions. Admittedly, *APS* is slow due its on-line learning process and inherent complexity [85]. *APS*' behavior regarding the 3 criteria has as follows:

Efficiency	*Overhead*	*Scalability*
high	high	low

The *Innocuous Topology Aware Overlay Construction (ITA)* approach [86] seeks to offer both an overlay formation approach and an effective way to carry out searching. When constructing the overlay, *ITA* exploits the notion of *short* and *long* connections. Should N be the number of network peers, $\alpha \leq 1$ a system-wide magic number and x an α-related latency threshold, the bootstrapping peer randomly selects $\alpha * N$ "close" (latency below x) and $(1 - \alpha) * N$ distant (latency above x) nodes as its neighbors. Of course, the bootstrapping peer is not able check all the peers in the network to find the global best for the above sets. Authors prove that latency measurements against $30/\alpha$ randomly selected peers result to a good estimation. Searching occurs in 2 phases: in the initial, the querying node floods its distant neighborhood with *TTL=1*. Subsequently, peers receiving a query over a "long link" commence a local flood with *TTL=ttl* where *ttl* is system-defined. The main objective of the overlay construction phase is to yield a network that exhibits low clustering. In turn, this is a beneficial characteristic for random graphs as it can lead to a larger coverage of peers and at the same time help reduce duplication of messages travelling around the network. This combination offers efficient lookups that pose minimal or no negative impact to other mechanisms possibly employed by the *P2P* systems (e.g., 1-hop replication or dynamic querying mechanisms). Experimental results in the same paper suggest a 50% reduction in communication latency among peers by cutting off up to 25% of the *IP*-level traffic generated. Regarding the 3 criteria, *ITA* is placed as follows:

Efficiency	*Overhead*	*Scalability*
medium	low	medium

EGOIST [87] is a set of algorithms used to construct and manage overlay networks. *EGOIST* utilizes a selfish approach in the sense that every participating peer continuously updates its neighbors so as to minimize the sum of distances to all destinations under shortest-path routing. A newly arriving peer, randomly connects to an already participating node through a bootstrap server. Once connected, the peer starts receiving information throughout the link-state mechanism and thus, after some time, it has a complete picture of the overlay graph. Then, the peer estimates the delay to all other nodes to determine its potential neighbors and ultimately connects to the overlay using some policy; such a policy might be for example the minimization of the average delay to all its neighbors. *EGOIST* requires extensive resource usage to continuously update the overlay connections for all nodes in the system. This needs $O(n^2)$ measurements. Fortunately, each node does not need to do these measurements for maintenance (monitoring and updating) with all other participating nodes, but just with a number of $k < n$ nodes that it chooses to establish links with. The latter yields a reduced $O(kn)$ time complexity for maintenance while complexity $O(n^2)$ is required less frequently and only when nodes reevaluate their connections. *EGOIST* fares as follows:

Efficiency	*Overhead*	*Scalability*
high	high	low

The main objective of the *Biased Neighbor Selection (BNS)* approach [88] is to strengthen *BitTorrent*'s [49] locality by

carefully choosing most of the neighbors of a peer to come from the same *ISP*, while simultaneously adhering to the near-optimal download performance of the protocol. By and large, *BitTorrent* deploys mechanisms that have proved aggressive to *ISP* networking and accounting as they function in a "without-borders" approach. The default *BitTorrent* specification designates 35 connections for each peer regardless if such connections are placed within or outside the borders of the *ISP* that hosts the node. *BNS* proposes that only k out of these 35 connections be established with nodes beyond the *ISP*-borders; the k connections offer an extended and "global" view of the network at large and seek to strike a balance between the load imposed inside and outside the *ISP*. The latter refrains peers from exchanging traffic through expensive network links should there exist alternative local connections offering the required services. *BNS* is realized through 1) the modification of the tracker so that *ISP*-local resources are rapidly identified 2) the use of *P2P*–traffic–shaping devices on *ISP* edge–routers. In [88], a combination of biased neighbor selection along with bandwidth throttling and caching techniques for near-optimal lookup results is proposed. The effectiveness of *BNS* is debated in [89] where experimental results suggest that most peers in a swarm reside in different autonomous systems (*AS*s). Consequently, clients frequently do not have enough peers to leverage intra-*AS* connections [89]. Moreover, *BNS* deployment requires knowledge of *ISP* network mappings and awareness of possible changes that occur in the infrastructure limiting its widespread adoption; the following table qualitatively depicts the expected behavior of *BNS*:

Efficiency	*Overhead*	*Scalability*
high	low	medium

Ono [90] is a protocol that helps contain *BitTorrent* traffic within individual *ISP*s and enhances the downloading rates by favoring connections within the borders of a single autonomous system. Contrary to *BNS* [88], *Oro* leads to improved performance without requiring any explicit cooperation between *ISP* subscribers or any additional infrastructure and network topology information. *Oro*'s selection approach is landmark-based and leverages existing *CDN*-infrastructure for peer distance estimation. *CDN*s already use both static (i.e., geographical, *IP*-clustering) and dynamic (i.e., network delay measurement) information for their replica selection. Thus, the algorithm leverages the observation that peers that are redirected to a specific *CDN* replica does not only mean that these are close by this replica but to each other as well. The redirection behavior is modeled in terms of *ratio map*, a vector of $<$*replica-server,ratio*$>$ tuples, where *ratio* is the percentage of times the *CDN* redirects the peer to the specific *replica-server*. The protocol's bootstrapping phase consists of the incoming peer performing *DNS* lookups to *CDN*-names to build its redirection information. Initially, the *DNS* polling interval is set to 30 seconds and increases by 1 minute every time redirection information to the *CDN* remains unchanged after the lookup. The *DNS* polling interval is halved if pertinent redirection information has changed. To avoid the bootstrapping phase when the *ratio map* is sufficiently fresh, *Oro* persists the map after the end of a *BitTorrent* session. Experimentation with *Ono* yields minimization of inter-*ISP* traffic costs and achieves near optimal performance of *BitTorrent* download [88]. The *ISP*-friendly design of *Ono* however has been questioned when it comes to Internet-scale [91], [92], [93]; its qualitative rating stands as follows:

Efficiency	*Overhead*	*Scalability*
medium	medium	low

In [92], three techniques that "inject" locality awareness in *BitTorrent*-like applications are discussed. The first *macroscopic-level* technique focuses on improving the neighbor selection process. When a peer asks a tracker to join, the later sorts its swarm peers according to their hop-count distance from the requesting peer send out the top-k list of nodes (e.g., k=50). The second technique, applied in an *intermediate-level*, manipulates *BitTorrent*'s chocking/unchocking mechanism. A peer unchockes its 4 closest, in terms of autonomous system hop-count, interested neighbors. The same applies also when the peer turns to the seeding state; this *intermediate-level* approach favors least distance and is in contrast to the original *BitTorrent* implementation which favors uploading speed. The last technique operates at a *microscopic-level* by choosing to pick the next chunk to download based on a locality-first policy; this picks first the chunk that is closer, as opposed to *BitTorrent*'s rare-first chunk picking policy. The distance value of a chunk is computed as the mean value of the distances of the peers that posses it. Through experimentation on a real-world Internet topology simulated in PlanetLab, the locality-based approach achieves traffic optimization. Nevertheless, it does not do that well when compared with the random approach of the standard *BitTorrent* protocol.

All 3 techniques in [92] seek to harness the locality of the basic functions of the *BitTorrent* protocol. Even though they do not restructure the network to map better to the underlying infrastructure, they favor the intra-*AS* over inter-*AS* communications thus, reducing the overall communication cost. Again, there is a trade-off between reducing inter-domain traffic and fairness among peers in terms of the data the peers upload [92]; as far as the 3 criteria, the overall approach fares as follows:

Efficiency	*Overhead*	*Scalability*
medium	low	medium

The key objective of *TopBT* [89] is to enhance proximity awareness in *BitTorrent* without the need of additional infrastructure. *TopBT* is built on the conjecture that a good peer selection metric should take into account both the downloading speed and the network topology. The metric proposed is defined as the ratio of download to upload rate d/u divided by either the link-level l or the number of routing hops required a. This metric should guarantee selection of peers with high download rate, low reciprocal upload demand and decreased routing hops. The above *TopBT*-metric is applied in various aspects of the original *BitTorrent* protocol to "inject" topology awareness for improved peer selection. More concretely, *TopBT* modifies: 1) the peer list the tracker returns when someone first tries to connect to the swarm, 2) the initial connection establishment, and 3) the unchoking mechanism of *BitTorrent*. *TopBT*'s key features are that it

does not rely on any Internet infrastructures (ISPs or CDNs) whatsoever and it cooperates with already deployed BitTorrent clients. It also supports both link-hop and *AS*-hop metrics to identify proximity in different levels in order to reduce overall traffic while preserving downloading performance. Through experimentation on hundreds of *PlanetLab* and residential hosts, *TopBT* offers more than 25% traffic reduction and 15% faster downloads if compared with popular *BitTorrent* implementations. In reference to the stated 3 criteria, *TopBT* fares as follows:

Efficiency	*Overhead*	*Scalability*
medium	low	high

The *Underlying Topology-Aware Peer Selection (UTAPS)* [94] presents an enhanced peer selection strategy. *UTAPS* operates in two stages: in the first, it collects information to develop a picture of the underlying topology and in the second, it proceeds to make the peer selection based on this knowledge. In the first stage, *UTAPS* employs network tomography, a technique that probes from a large network's endpoints to infer its internal characteristics [95]. Upon arrival of a new peer, the tracker trace-routes it to obtain some basic knowledge including *IP* address, routers involved, *RTT* and number of hops traversed. The more the peers in the swarm the better picture a tracker has for the underlying infrastructure. The tracker provides this information to the newcomer in the form of the bootstrapping peer list. These newly joined peers traceroute the tracker peer list themselves and return back to the tracker in order to further enhance the tracker's view of the infrastructure. In the second stage, *UTAPS* employs a set of heuristics which target those peers that expose low *RTT* and are found within certain hop-counts away from the routers observed during the initial traceroute. Peers running *UTAPS* instead of the random peer selection, achieve better downloading rates and at the same time offer reduced burden on the underlying *ISP*s [94]. *UTAPS* stands as follows in regard to the 3 criteria:

Efficiency	*Overhead*	*Scalability*
medium	medium	medium

A clustering approach that differentiates peers encountered in a swarm in those that are local, intra-*ISP* and inter-*ISP* is presented in [96]. The classification is carried with the help of the core routers used while peers communicate with the tracker. It is assumed that even though the network may be highly volatile, these core routers are ~~are~~ usually more stable. For this, a newly arriving *BitTorrent* peer, trace-routes the tracker and stores vectors containing information like the *IP* address and the hop number of a traced router, as well as the link delay of the traced router and the previous hop along the trace-route path. These vectors are reported to the tracker which uses this information to classify the router through a k-means classification algorithm, with $k = 3$. For peer selection, [96] proposes a biased approach where the tracker returns c close peers and $d = N - c$ distant peers, where N is the length of the returned list. In choosing c, the tracker employs an iterative search process first in the peer's local neighborhood. If not sufficient number of peers are available there, the tracker goes to intra-*ISP* and subsequently, if needed, to the inter-*ISP* cluster. Experimentation in a controlled environment indicates up to 5% faster downloading times as well as up to 22% reduction of the total cross-*ISP* traffic. This clustering approach fares as follows in regard to the 3 criteria:

Efficiency	*Overhead*	*Scalability*
medium	medium	medium

In [97], two *Peer-exchange Routing Optimization Protocols (PROPs)* are introduced to help adjust the neighborhood graph of an overlay network while aiming at reducing the network's overall link latency. *PROPs* are weaved around the concept of "exchange" of neighbors among peers; this is carried out so that participant node can collectively benefit from the attained reduction in network delays. This "collaboration", is what differentiates this approach from others by allowing two peers to optimize their neighborhood environment, than simply letting each node to "selfishly" choose its own strategy. The *PROPs* follow two phases: during the *warm-up*, a node u probes its neighbors to collect distance information. Subsequently, u contacts, at a fixed time rate *timer*, a random node v, *nhops* away. New distance information is calculated independently by nodes u and v, now for the hypothetical state when the potential exchange would occur. If the benefit gained, in terms of reducing the average distance among the peers involved, is above some predefined threshold then the exchange does occur. Otherwise, no further action is taken. The second *maintenance* phase differs from the warm-up in that the probability of peers to be probed and the *timer* interval that this happens now depends on past peer exchange results.

In the generic form of the *PROP-G* protocol, a peer can exchange all of its neighbors with those of another peer. This essentially results in the two nodes exchanging positions so the topology and connectivity of the overlay is not affected by the operation of the algorithm. The optimized version *PROP-O* permits peers to exchange selected sets of their neighbors. It is not allowed to exchange neighbors along the path of the nodes that perform the exchange in order to ensure that in the end of the process the peers will remain connected. Additionally, the algorithm needs to preserve the characteristics of the network (i.e., the graph remains connected, preserve the degree of the nodes), so the exchange always involves equal number of neighbors. The following table depicts the *PROP*s behavior regarding the stated criteria:

	Efficiency	*Overhead*	*Scalability*
PROP-G	medium	low	medium
PROP-O	medium	medium	medium

The *Distributed Domain Name Order (DDNO)* approach [98] uses domain names to detect topologically-close nodes. The fundamental assumption of the approach is the nodes found in the same domain are also topologically close. *DDNO*'s outcome is a flat overlay topology which, with some adjustments, can be utilized in super-peer architectures as well. According to the algorithm, half of the possible connections of a node are used to connect to local peers (called *sibling* connections) and rest are used to randomly connect to the peers anywhere on the network (called *random* connections). The former ensure the reduction of long distance traveling for messages, while the latter secure the connectivity of the

structure and prevent network partitioning.

Upon arrival, a peer n asks a *host-cache* to establish its random connections. To identify its siblings, n multicasts a specialized message l to locate the right candidate(s). Initially, the behavior of this message, is modeled as a random walker, until l reaches some peer d capable to guide it through its next step. This capability is enabled through a *ZoneCache*, a data structure that contains information for nodes accessible in an r-hop radius from d. Ultimately l will reach a node m that is candidate to become n's sibling. Then, m issues a broadcast message to its own siblings and then all in the group will inform the initiating peer n of who is willing to accept new connections. The population as well as the maintenance of *ZoneCaches* across the network are key concerns when it comes to the efficient construction of the overlay. While seeking to address the mismatch problem, *DDNO* is a heuristic approach that uses *Split-Hash* and *dnMatch* algorithms to effectively encode domain names and determine domain locality respectively. In terms of our 3 criteria, *DDNO* fares are follows:

Efficiency	*Overhead*	*Scalability*
medium	low	medium

The *Critical Topology-Aware Grouping (CTAG)* approach [99] advocates a grouping strategy of peers based on the *Internet Assigned Numbers Authority (IANA)* and the respective *Regional Internet Registry (RIR)*'s *IP* assignments. Based on these assignments, nodes that belong in the same organization are always addressed from the same block of *IP* addresses. *CTAG*, exploits this observation to construct topology-aware unstructured overlays. On top of this, *Adjacency Measurement (AM)* is proposed, as a technique which uses the longest matching *IP* segment criterion to compute node proximity. *CTAG* focuses on both the construction of the overlay as well as its constant and dynamic adaptation. In its first phase, called *bootstrapping grouping*, the *Gnutella* web caching mechanism is modified so that a newcomer may choose the closest, in terms of *AM*, cache to retrieve its bootstrapping list. Similarly, during its second stage, called *dynamic revision*, the standard connection establishment mechanism is also altered so that nodes with the highest *AM* metric are chosen. When a node reaches its maximum number of neighbor connections, it disconnects those neighbors with the lowest *AM*. *CTAG* behaves as follows regarding the 3 criteria:

Efficiency	*Overhead*	*Scalability*
low	low	medium

Landmark Binning [100] partitions close-by nodes into bins based on their distance from well known anchor nodes across the Internet.To detect locality, peers mainly use network latency (i.e., *RTT*). Despite the fact that such delays are not always accurate, they are used in [100] for they are non-intrusive, transparent and easy to apply. For the binning to work, a few anchor servers with known physical locations need to be installed in strategic positions across the Internet. It is conjectured [100] that 12 such servers can prove sufficient for the task. An arriving node measures its distance from these landmarks and unilaterally decides to join a specific bin based on its measurements. Specifically, the node measures its round-trip time to each of the landmarks and orders the landmarks in decreasing order of those *RTT*-values. Each permutation of the set of landmarks represents a specific bin. Should there m landmarks be adopted, $m!$ different bins potentially exist. Peers that end up with the same ordering, belong to the same bin, in the sense that if they experience similar overall latencies from fixed network points; it is also very likely that the peers in question are close to each other.

As the operation of the method is independent of the model incorporated by the overlay network, the approach can be applied with no significant changes to either structured or unstructured *P2P* systems. The rather obvious disadvantage of the approach lies in the landmark servers that must be installed and maintained throughout the Internet. Provided that a *P2P*-system often may have a few million nodes connected at any given point in time, landmark servers do inherently play a pivotal role in the operation of the network. The approach has received a fair amount of criticism. For instance, [101] points out that end-hosts may bear the cost of latency measurements and so, they may become bottlenecks. [102], [103], [104] argue that uneven distribution of nodes ultimately lead to load imbalances and [105] indicates that its coarse-grained nature fails to effectively distinguish among relatively close nodes. Finally, [106] underscores the potentially increased maintenance cost(s). The landmark binning behaves as follows:

Efficiency	*Overhead*	*Scalability*
medium	low	low

The *mOverlay* [104] approach addresses scalability issues that might arise when static landmark servers are in use. To this end, the use of dynamic landmarks is proposed. *mOverlay*'s founding notion is that of a *group* that designates a set of peers found in close proximity. This proximity is user-defined in the protocol and may involve metrics including *RTT*s and network latencies. By and large a clustering approach, *mOverlay* seeks to recreate small-world-like properties by producing a two-level hierarchical structure: at the top level, there are only connections among groups while at the bottom, only *intra-group* connections occur among peers.

Clearly, identifying groups and accurately finding the closest group to a peer is a fundamental concern in the creation of the overlay. Nodes are grouped based on their distance to the groups already in the network, rendering the latter be the *dynamic landmarks* in the process. For a incoming peer Q, the *grouping criterion* designates that when the distance of Q and some group As neighbor groups is the same as the distance between group A and group As neighbor groups, then host Q should belong to group A. During network initialization or when the grouping criterion is not met, new groups are created. A peer may reach its group by expending at most $O(logN)$ messages. Finally, *mOverlay* maintains stability and constant overheads when a host either fails or departs the network; this is achieved through periodic cache updates and group leader selections, should a node either leaves or dies. In terms of the stated 3 criteria, *mOverlay* fares as follows:

Efficiency	*Overhead*	*Scalability*
medium	medium	medium

B. Discussion on the Algorithms for Unstructured Architectures

While surveying efforts to overcome the mismatch problem in the area of unstructured *P2P* systems, we came to identify four key methodologies utilized by the discussed approaches; they are:

i) topology adaptation
ii) forwarding optimization
iii) caching and replication, and
iv) landmarking.

A key characteristic shared by many of the presented techniques is that they do not strictly adhere to a single methodology. As approaches attempt to address the challenging problem of topology mismatch, they resort to heuristics and so, they do not often furnish "pure" approaches that exclusively belong to one of the 4 aforementioned methodologies. For instance, *mOverlay* combines topology adaptation with landmark binning while *Gia* weaves together 3 methodologies: topology adaptation, forwarding optimization and caching. This combination inevitably leads to feature/performance trade-offs: for example, the effectiveness of a caching/replication component can be undermined by a continuously adapting overlay that removes important links between peers. If a technique is to be evaluated for a specific application domain, its overall approach as well as its underlying used methodologies have to be considered.

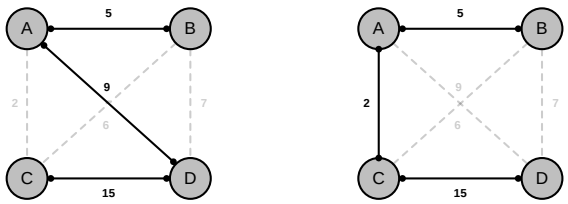
In what follows, we outline how the surveyed protocols use elements of the above four methodologies. We then offer a summary qualitative comparison for all surveyed approaches applicable for *unstructured P2P systems*. Specifically, Table I summarizes, for each effort, the methodologies it incorporates, its special highlights as well as a rough estimation of the pros and cons in its implementation. Last but not least, Figures 9, 10 and 11 pictorially compare the surveyed techniques in terms of *efficiency*, *overhead* and *scalability*; we defined these three criteria in Subsection II-C and we collectively used them as a yardstick to qualitatively evaluate the presented approaches.

1) Topology Adaptation Methodology: Protocols based on topology adaptation modify the topology of the *P2P* network using various schemes. The two most commonly used create *spanning tree*s using connection graphs as well as *clusters* of physically close nodes.

Narada and subsequent algorithms including *AOTO, LTM, SBO* try to solve the problem by building "a richer connected graph" and by forming minimum spanning trees over this graph that can efficiently route messages among peers. *AOTO, LTM* and *SBO* try to overcome this limitation with ingenious schemes like forming minimum spanning trees for the 2-hop away neighbors for each node, separating participant nodes into groups, sometimes with different responsibilities and tasks at hand (e.g., *SBO*). The advantage of building minimum spanning trees is that they maintain the connectivity on the network in an efficient manner while still preserving the overall search scope. However, their construction and their update costs, especially in dynamic and high-churn environments, may cause large traffic overheads on the underlying network [50], [73], [53].

The cluster-based approaches, on the other hand, link physically-close nodes to each other. *T2MC*, for example, uses trace-route logs and *DDNO* exploits domain names to cluster nodes in proximity at peer join. Further enhancements may include dynamic local restructuring of the overlay graph through neighbor exchange like in *PROP* or cycle-cut like in *DCMP* to achieve continuous adaptation throughout a peer's life-cycle. Unfortunately, commonly used methods for proximity detection across Internet do not always return reliable results and therefore, mapping accuracy is not guaranteed. Specifically for traceroute, its overhead is not negligible and routers or firewalls in a network may have already been configured to disable traceroute response from the start. The most problematic aspect of clustering, though, is its nature per se. Limited connectivity among the various local domains can significantly shrink the search scope, negatively affecting the query response time that the *P2P* user experiences. Among others, *DDNO* tries to balance the efficiency of the clustering approach with enhanced node connectivity by forcing half of each node's connections to be with other, randomly selected, nodes.

As a methodology, topology adaptation, ultimately aims to reduce the average path traversing cost from one node to another. In doing so, the methodology re-arranges the overlay network so that it becomes a better fit for the underlying *IP* network as Figure 5 depicts. Despite the fact that the above is a plausible proposition, it is not always advantageous though. For example, a topology adaptive algorithm can exchange a slow edge in the overlay with a faster one thus reducing the average latency of the network communication. The pitfall here is that this new virtual link can traverse a fast *AS*–to–*AS* link meaning that even though message round-trip-time is reduced, it has actually additional cost in terms of inter-*ISP* communication accounting and management.



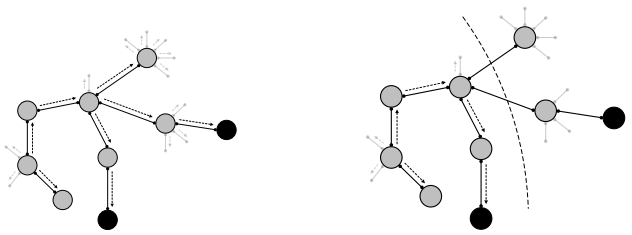
(a) Inefficient overlay topology averaging $\simeq$ 16 delay units.

(b) Efficient overlay topology after adaptation averaging $\simeq$ 11 delay units.

Fig. 5. A simplified example on how overlay topology adaptation may improve matters.

2) Forwarding Optimization Methodology: In unstructured systems, there is no way to know where data are actually located. This is why early works like Gnutella [107] used blind flooding, a *BFS*–approach with depth d and a system-maximum *TTL* value (in terms of hops) for messages. This is inefficient since it generates a large number of messages, a lot of which are duplicates. Approaches based on *forwarding-optimization* propose intelligent search mechanisms in an attempt to make searching more efficient.

Figure 6 shows a simple example of how forwarding optimization may work. Fig. 6(a) shows a protocol that simply



(a) Typical blind flooding.

(b) A node can decide where to forward the messages.

Fig. 6. Forwarding optimization in constraining message flooding.

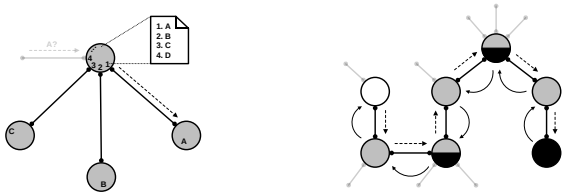
floods the entire network in search of object found in the nodes colored in black. Alternatively, a node can decide to forward its messages not to every output link but to a specific or subset of its outgoing links. In Fig. 6(b), the node in the middle, does not flood its neighbors as instead in only picks one of them. The dashed line on the right, shows that, for this routing process, the protocol has rendered two potential forwarding paths as inefficient (e.g., the target nodes show overloading signs).

Many alternative schemes have been proposed for this category, some of which we have surveyed; iterative deepening, modified *BFS*, local indices [38], k-walker random walks [63] or heterogeneity exploitation and token-based flow control mechanisms [62]. More sophisticated alternatives also exist, taking into account application–level requirements or even exploiting machine learning techniques to adjust overlay network routing [84].

Forwarding-optimization schemes in unstructured *P2P* systems can be classified as *DFS*- or *BFS*-based. Routing indices [108] for example is a *DFS*-based technique while all the aforementioned are *BFS*. Also, the schemes in question can be classified as deterministic or probabilistic (probabilistic, random or ranking-based query forwarding). Iterative deepening and local indices [38] are examples of deterministic approaches. Moreover, in the literature algorithms are also taxonomised as blind or informed depending on whether nodes keep some metadata to facilitate the search. For example, k-walker random walks or iterative deepening are considered blind searches while local indices informed.

On one hand, forward-optimization approaches have the advantage of enhancing the search responsiveness and reducing the aggregate resource usage of the physical network. On the other, they suffer from drastic reduction of the search scope (in Fig. 6(b) the object on the far right is not reached) thus, limiting the scalability of the whole network. Forward-optimization suggestions address the problem of the mismatch in a limited manner since they do not provide any guarantees that overlay and underlying topologies are aligned with each other. To this end, forward-optimization techniques are commonly applied in conjunction with other methodologies to improve the overall quality of *P2P* systems.

3) Caching and Replication Methodology: Caching is widely used to exploit locality and minimize redundant transfer of data. Caching has with much success been successfully adopted by web– and file–server application environments. Since peers in a *P2P* system also operate as servers, it is intuitively expected that *P2P* file–sharing systems can also benefit from caching in improving performance and reducing overall resource usage. However, the design of caches in this context is non-trivial compared to the web-based caching. Due to the fact nodes play the double role of server and client, two important issues have to be considered at design time. First, the lifetime of a query is short, as the nodes join and leave frequently. Second, the result of a single query string is not always the same, as this depends on the source of the query, the *TTL* value set for the messages, the current interconnection of peers and the high volatility of the environment. Thus, to develop a successful caching system for a *P2P* architecture, these parameters also have to be carefully considered. *P2P* caching/replication can be applied at two different levels, namely caching indices or pointers to data (Fig. 7(a)) or caching the data itself (Fig. 7(b)). Successful implementations



(a) Indexing can reduce the cost of the last hop.

(b) Data replication along the path of a successful query.

Fig. 7. Index and data replication strategies.

have already been developed in some commercial *P2P* systems, like *KaZaA* or less popular ones including *Gia* and *BNS*. Even though the state of the art in *P2P* protocols using caching methods helps reduce the burden of network resources, their contribution in addressing the actual mismatch between the overlay and ~~and~~ the underlying networks remains limited.

4) Landmarking: In landmark-based algorithms, nodes use network delay (e.g., *RTT*) as a distance measurement method to position themselves with respect to "a priori" known servers on the Internet, like *Ono* which uses the *CDN*-infrastructure for this purpose. For example, in Figure 8, the newly arriving peer, measures its distance to an array of landmark points denoted as L_n. Then it creates an ordered list of the landmarks based on its measured distances uses that information to select its neighbors. Specifically, the landmark–servers are used by nodes to estimate their positions based on the intuitive assumption that peers with similar distances to a set of landmarks, are physically close to each other, as well, over the network.

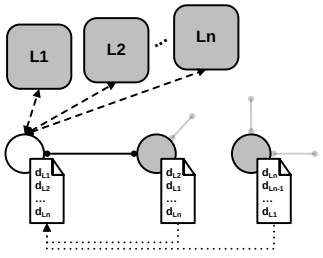


Fig. 8. Landmark binning during node bootstrap.

TABLE I: Summary table for unstructured algorithms.

Algorithm / Paper	Topology Adaptation	Forwarding Optimization	Caching / Replication	Landmarking	Highlights	Pros / Cons
[38]	☐	☑	☑	☐	Iterative Deepening (ID). Directed BFS (DBFS). Local Indices (LI).	+ ID reduces the messages especially in upper levels of the tree. + LI reduces aggregate bandwidth usage and improves query efficiency. – ID needs evaluation time between iterations. – DBFS uses heuristics so it depends on their efficient choice. – LI add index update overhead which might be heavy process especially in high-churn systems.
DAPS [61]	☐	☐	☐	☐	Clustered routing tables based on delay. Pruning flood, an iterative deepening and multiple BFS approach with a pruning boundary.	+ It is a system between structured and unstructured.
Gia [62]	☑	☑	☑	☐	Random Walks (RW).	+ RWs issue one copy of the query thus not flooding the whole network. – RWs can reduce search scope.
DCMP [68]	☑	☐	☐	☐	Cycle detection.	+ Drastically reduces duplicate messages. – Cannot detect cycles in distance bigger than the TTL value of the IC message.
[70] & [63]	☐	☑	☑	☐	Uniform replication. Proportional replication. Square root replication allocation. *k*-walker query scheme.	+ Uniform replication reduces time spend on unsuccessful searches. + Reduces search time for frequent queries. *k*-walker quering scheme can reduce network traffic up to two orders of magnitude. – Proportional replication struggles in locating rare objects.
Narada [50]	☐	☑	☐	☐	Mess creation. Minimum spanning trees.	+ Mess and trees are kept up to date in high churn environments. – Works well only for small groups of peers.
AOTO [75]	☑	☑	☐	☐	Minimum spanning trees. Peer proximity heuristic for removing costly links.	+ Spanning trees only to immediate neighbors so no flooding and at the same time no shrinked search scope. + Selective flooding effectiveness is detached from physical or overlay topologies. + The more logical neighbors, the more effective selective flooding becomes – High recalculation costs. – No sophisticated selection policy for candidate non-flooding peers.

Continued on next page

TABLE I – *Continued from previous page*

Algorithm / Paper	Topology Adaptation	Forwarding Optimization	Caching / Replication	Landmarking	Highlights	Pros / Cons
ACE [76]	☑	☑	☐	☐	Minimum spanning trees. 1-hop proximity heuristic.	+ No flooding. + Less overhead compared to AOTO since computation is done within a certain diameter from the source peer. – Slow convergence speed. – Enhanced topology optimization comes to the expence of higher communication/computation overhead.
LTM [77]	☑	☐	☐	☐	TTL detector (2-hop distance). Delayed low productive connection cutting.	+ Compared to AOTO, ACE and SBO achieves faster convergence speed. – Creates more overhead than AOTO, ACE and SBO. – Needs synchronization of peer clocks. – Does not consider shortcuts created by powerful peers when choosing to disable connections (only uses delay metric).
SBO [78]	☑	☑	☐	☐	Red/white bipartite overlay.	+ Efficient in both static and dynamic environments. + Compared to AOTO incurs half the overhead. – Needs almost double the steps of LTM to converge (static or dynamic environments).
THANCS [80]	☑	☑	☐	☐	Local optimum heuristic Piggybacking neighbor distance in queries	+ Completely distributed approach. + Presents trivial overhead compared to the query cost savings. + Convergent speed faster among AOTO, LTM, SBO. + Does not shrink the search scope. – Design cannot be extended to support non-flooding-based systems.
HAND [83]	☑	☐	☐	☐	Triple-hop adjustment.	+ No need for clock sync. + Fully distributed. + Low overhead for the triple hop adjustment. + Applicable to both static and dynamic environments. + Low query response time. – Compared to LTM has lower traffic reduction and query response rates.
APS [84]	☑	☐	☐	☐	machine learning adaptive mechanism.	+ Fully dynamic switching decision policy. - Low convergence due to the learning process.
ITA [86]	☑	☑	☐	☐	Short/long connections. Local flooding.	+ Low clustering. + Large peer coverage. + Reduced duplication. + Low or no impact to other mechanisms of unstructured p2p networks (e.g. 1-hop replication, dynamic querying).

Continued on next page

TABLE I – *Continued from previous page*

Algorithm / Paper	Topology Adaptation	Forwarding Optimization	Caching / Replication	Landmarking	Highlights	Pros / Cons
EGOIST [87]	☑	☐	☐	☐	Selfish shortest path routing	– constructs a global view of the network
BNS [88]	☑	☐	☑	☐	ISP clustering (tracker-side or ISP-side detection). Bandwidth throttling. Caching.	+ Localizes traffic within an ISP. + Preserves the efficiency of BitTorrent protocol. – Needs ISPs to either provide information or infrastructure changes. – Locality-based approaches do not treat fair all peers.
Ono [90]	☑	☐	☐	☑	ISP clustering. Landmarking based on existing CDN infrastructure (CDN redirection measurements).	+ Needs no ISP cooperation. + Needs no network topology information. – Deepends on feeds from major internet infrastructures and large deeployment of its client. – ISP-friendly approaches do not seem to have a great impact on Internet-scale basis.
[92]	☑	☐	☐	☐	AS hop count minimization on neighbor selection, on chocking/unchocking mechanisms and on next-chunk picking.	+ Optimization of the inter-AS traffic. – Locality based approaches do not treat fair all peers.
TopBT [89]	☑	☐	☐	☐	Peer selection metric that takes both downloading speed and network topology into account. Applied in multiple places of the BitTorrent protocol (bootstrap, connection establishment/replacement, unchocking).	+ No need for additional infrastructure. + Enhances both traffic and downloading. – Needs off-line processing of BGP dumps.
UTAPS [94]	☑	☐	☐	☑	Network tomography to construct a picture for the underlying network.	+ Reduced ISP burden. + Better downloading speeds. – Small, laboratory-scale evaluation setup.
[96]	☑	☐	☐	☑	Cluster peers in a swarm into local, intra- and inter-ISP.	+ Reduced ISP burden. + Better downloading speeds. – Similarly to UTAPS, the evaluation setup was small scale.
PROP [97]	☑	☐	☐	☐	Neighbor exchange between peers.	+ Cooperation between peers. + Guarantees the connectivity of the network between exchanges.
DDNO [98]	☑	☐	☐	☐	Domain name topology detection (Split-Hash and dnMatch).	+ Can be applied to both fully unstructured and super-peer based architectures. + Secures connectivity of the network. + Reduces cost of message exchange.
CTAG [99]	☑	☐	☐	☐	Clustering based on longest matching IP segment.	+ Focuses on both construction and adaptation.

Continued on next page

TABLE I – *Continued from previous page*

Algorithm / Paper	Topology Adaptation	Forwarding Optimization	Caching / Replication	Landmarking	Highlights	Pros / Cons
Landmark Binning [100]	☑	☐	☐	☑	Landmark binning.	+ It is independent of the overlay model. + The technique can be considered scalable. – Uses not so reliable network latency metric (this can lead to load imbalance etc). – The use of landmark servers renders the technique not fully distributed. – Excessive traffic flow towards the landmark servers is possible. – Fixed points in a network are inherently more exposed to malicious attacks. – Coarse-grained scheme.
mOverlay [104]	☑	☐	☐	☑	Dynamic landmarks.	+ Fully distributed. + It is independent of the overlay model. + Load balanced. – Coarse-grained scheme.

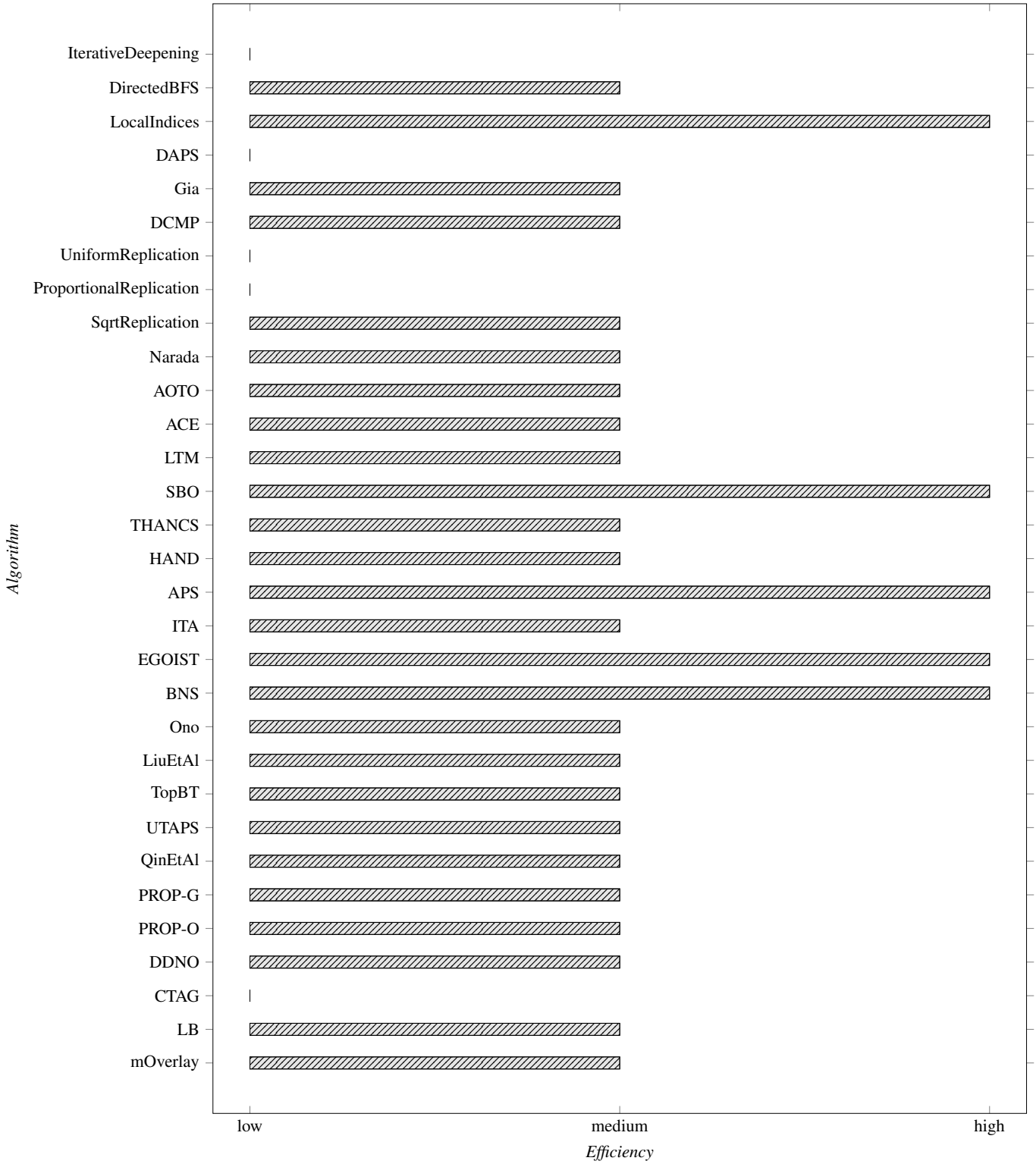


Fig. 9. *Efficiency* Pictorial Comparison of Unstructured Approaches.

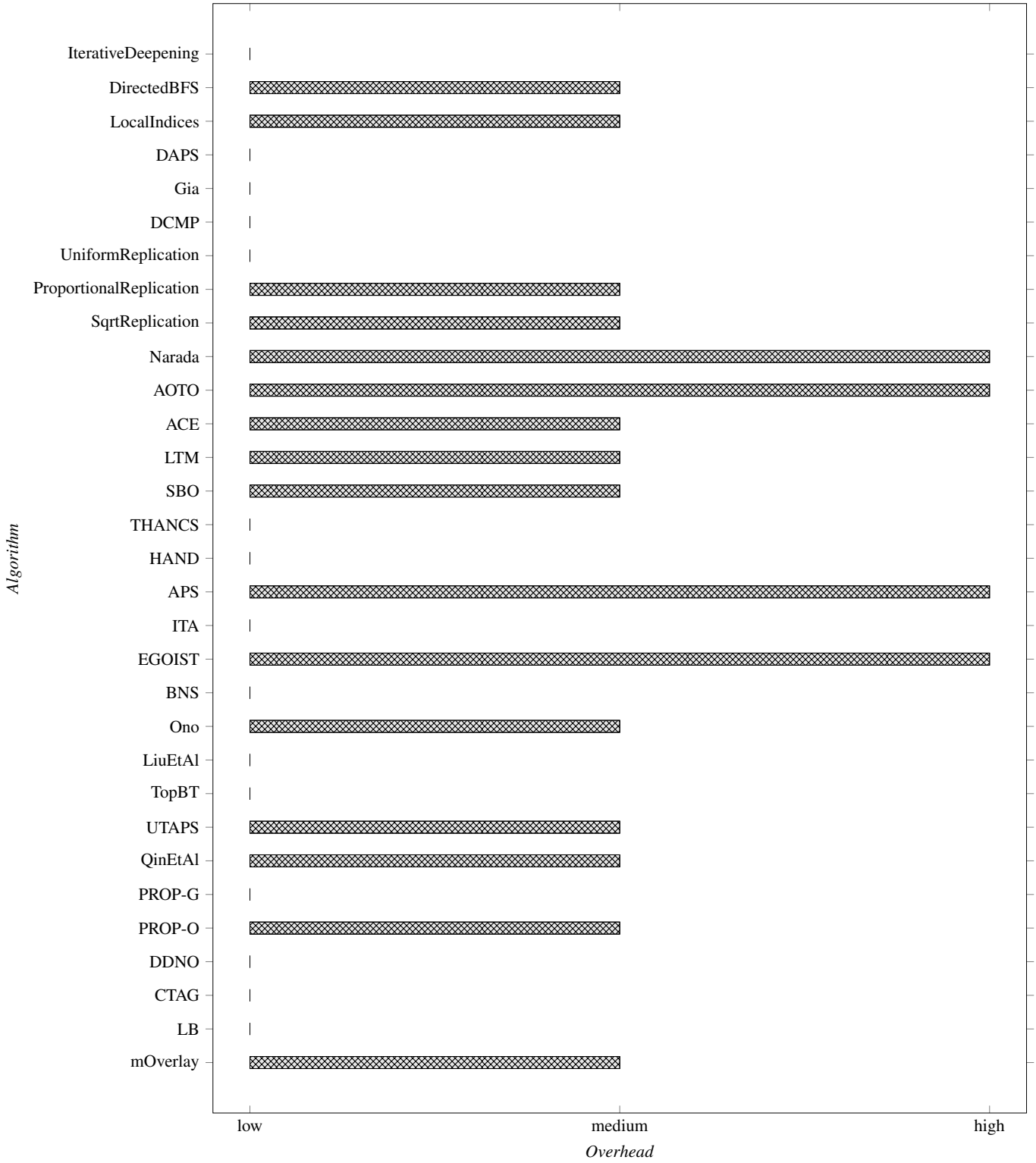


Fig. 10. *Overhead* Pictorial Comparison of Unstructured Approaches.

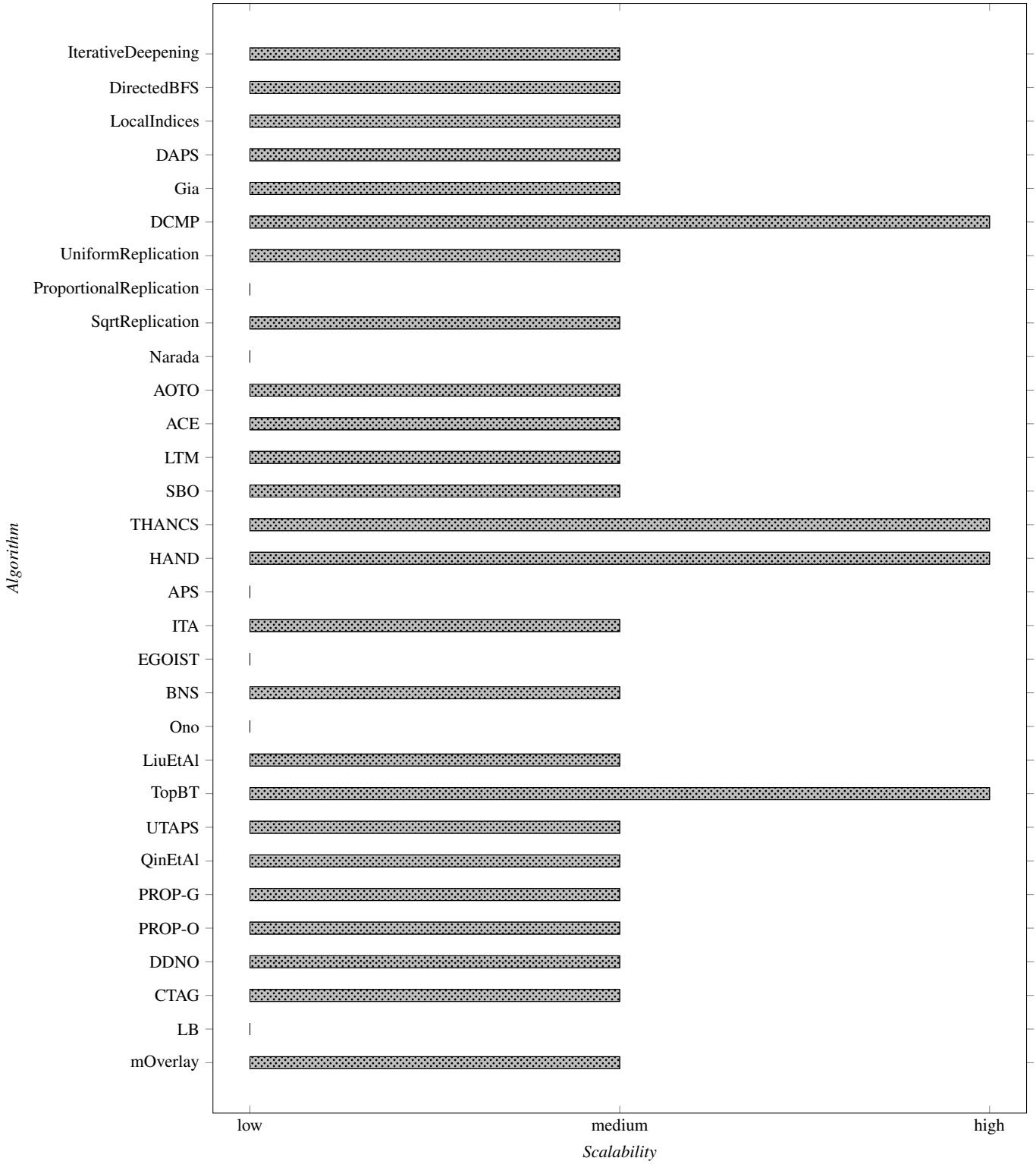


Fig. 11. *Scalability* Pictorial Comparison of Unstructured Approaches.

Landmark–based protocols have four important drawbacks: first, the network delay is not a reliable distance estimation method. For example, based on the load on the network, the delay to certain nodes or networks can significantly change over time; this eventually affects the distance measurements and yields wrong measurements that ultimately lead to wrong estimated positions for the nodes or incorrect and non–optimal clusterings of nodes. Second, it can be characterized as a rather coarse-grained approximation, therefore not particularly well suited for identifying the correct positions of nodes within close distance to each other. Third, relying on predefined nodes makes the whole paradigm not fully distributed and the landmark system prone to single point of failure. Lastly, landmark infrastructure requires costly installation and maintenance across the Internet and the different autonomous system domains. As popular *P2P* file–sharing applications usually have millions of peers connected to at any time, the network costs of maintaining these landmark servers can be quite high. A possible solution to the scalability problem of the static landmark servers is to use ordinary nodes as dynamic landmarks in an approach reminiscent to that of *mOverlay*. Even though this scales better, the accuracy of measurements may affect the overall performance of the system, especially in an environment with high churn.

IV. Structured *P2P* Networks

This section offers description and analysis of algorithms proposed to address the topology mismatch problem in structured *P2P* networks. The categorization of the approaches is based on how different levels of proximity information is gathered as well as the way peer routing tables are maintained and used in an effort to make overlays as efficient as possible in terms of hops traversed. Our categorization is based on previous work found in [109], [102], [110].

A. Algorithms for Structured Architectures

Global Soft-State [106] builds a global view to help nodes choose shorter routing paths. To generate the proximity information, it combines the landmark binning method with small scale distance probes to reveal the properties of the underlying network.

This global view of the network state is used by organizing it into "maps" for regions of the overlay, that are strategically placed across the network and made available to any node. The notion of a region is a well defined property for structured *P2P* networks and can be part of the Cartesian space in overlays like *eCAN* [111] or a set of nodes sharing a particular prefix in overlays such as *Pastry* [45]. Moreover, a node may appear in a maximum of $log(N)$ such maps.

Direct access to this information, also coined *soft states*, helps nodes find the best way to route their messages, while a publish/subscribe mechanism allows them to get notified on dynamic network changes.

Although this approach does reduce the routing latency to far nodes, it can become expensive as it is possible to end up with a very large number of regions each of which has to maintain a map. The fact that a peer may appear in multiple such maps points into additional traffic for probing and notifications and so renders the approach not scalable [105]. Moreover, to refine measurements additional inner-bin measurements are needed [112]. Also, improvements within a region can be minimal as there exists limited knowledge about neighboring zones. In terms of our 3 stated criteria, the approach fares as follows:

Efficiency	*Overhead*	*Scalability*
medium	high	low

Mithos [113] ensures that neighboring nodes in the overlay are close also in terms of the underlying layer by integrating proximity layout and proximity routing overlay optimizations.

During bootstrap, an incoming peer J obtains from a bootstrap node a set of candidate peers to establish links with. Subsequently, the node works iteratively to identify peers that minimize a given metric (i.e., network delay). The process stops when no additional peers complying with the above metric can be found. To tackle the local minima problem that such a gradient descent method suffers from, Mithos probes nodes within a 2-hop radius from the current node that helps minimize the distance before concluding the process.

J is then assigned a synthetic *ID* that represents the state of minimum distance from its close-by 2-d neighborhood as this was discovered during the aforementioned phase. The benefit of adopting synthetic coordinates is that distance computations among nodes can be done in the *ID* space without requiring physical measurements. Mithos proceeds then to designate J's closest neighbors as previous steps do not reveal with certainty which these nodes are. Using Mithos *quadrant-based* mechanism, J divides the 2–d space of neighbors in four regions and subsequently, establishes a link to the closest neighbor in each quadrant of the plane. It does so, by starting from a known node in the neighborhood and scanning towards all quadrant borders [114], [115].

A noted Mithos limitation is that the protocol cannot effectively handle dynamic arrivals and departures from the network [105]. [116] points out extensive probing for Mithos to determine distances and less accurate proximity results compared to other virtual coordinate systems. The last issue has been also attributed by [59] to the iterative neighbor selection process that is reportedly prone to premature termination at local minima. [58] argues that elements of the protocol's synthetic ID generation require a centralized implementation affecting Mithos' scalability. In regard to the 3 criteria, *Mithos* stands as follows:

Efficiency	*Overhead*	*Scalability*
medium	medium	low

LAPTOP [117] organizes the overlay into a tree-based hierarchy so that hops required during message routing be reduced and maintenance overhead be minimized. A caching scheme is also used so as to further lower routing table update costs. It is theoretically shown that *LAPTOP* routing path length is bounded by $O(log_d(N))$ and node joining and leaving in the overlay network is bounded by $O(d * log_d(N))$ hops in a balanced overlay tree with N representing the number of nodes and d the maximum degree of each node (i.e., number of links). *LAPTOP* adopts the geographical layout approach and

constructs its layout in a self-organizing and efficient fashion. The physical distances are estimated with round trip times to a few existing nodes in the overlay network. Each node is assigned a dot-formatted address (e.g., 1.3.4) with each octet ranging from 1 to d. The assignment process is done by appending a unique octet to the address of the parent of each node, while the root node is assigned address 1.

LAPTOP's routing scheme is similar to the longest-prefix *IP* matching scheme. At each forwarding hop, a message travels up the tree until the first common ancestor of both source and destination nodes is reached and then, the message starts descending towards its target. During tree traversals, special entries in the routing tables are maintained to increase routing efficiency and achieve finer–load balance. These special entries constitute the *routing cache* which enables a node to forward a message to a better longest-prefix match than that of its direct ancestor; this yields a quicker and more cost–effective step through the overlay towards the destination. To improve scalability, the number of child nodes and the size of the routing cache are limited. To contain maintenance costs, *LAPTOP* employs a simple *heartbeat*-based regime in which a parent is responsible for the monitor of its own children. *LAPTOP*'s behavior for the 3 criteria stands as follows:

Efficiency	*Overhead*	*Scalability*
medium	low	high

In order to optimize the underlying network use, [118] introduces proximity controls over the *vanilla Kademlia* [41] protocol's random *ID* assignment. In its quest to improve routing performance, the overlay uses a cost "yardstick" (i.e., metric) provided by the underlying network. Such cost information can be acquired by: *i*) measuring previous lookups, *ii*) exchanging or jointly-computing measurements with others, or *iii*) using a local database to look–up information.

[118] exploits two routing techniques: *proximity routing selection (PRS)* and *proximity neighbor selection (PNS)*. In *PRS*, the goal is to choose the best next hop while routing a message. *Kademlia* always selects the closest–node with respect to the *XOR* metric [41]. With *PRS* in place however, *Kademlia* may also choose the node that exhibits the smallest routing cost; in this way, it balances a trade-off between the logical overlay and actual underlying distances. In *PNS*, on the other hand, the goal is to keep peers with the least routing cost in the routing table.

Kademlia proximity injection improves locality of connections among peers. To enhance traffic locality, the use of *MaxMind GeoIP* database[4] is suggested to form clusters of nearby peers. To this end, Vivaldi coordinates [58] could be also used as an alternative for morphing locality. [118] however suggests that the clustering approach when combined with *PRS*/*PNS* helps contain 40% of messages within an *ISP* and allows only a 6% to go through the backbone. The behavior of [118] with respect to the stated criteria is deemed as follows:

Efficiency	*Overhead*	*Scalability*
medium	medium	medium

CHOP6 [119] is a Chord-variant DHT that exploits *IPv6* address format as well as available *RTT* information to enable proximity estimation among peers during message routing. Similar to *Chord* [40], *CHOP6* uses a finger table with each entry now holding $k>1$ candidate nodes. The next hop for a message is selected to be the one closest to the destination among those k candidate peers. This proximity is determined either by exploiting RTT measurements or, if there is no such information yet (i.e., a new peer appears), by estimating distance on the overlay's ID space.

When joining the network, each node receives a 64-bit *ID*. It comprises of a random part and the IPv6 prefix which is obtained by hashing the node's IP address and port number; this combination is deployed in an effort to preserve the uniformity of node assignment. The prefix in question is based on IPv6's global routing prefix. As IANA[5] assigns address blocks to regions (RIRs) which are further distributed at national level (NIRs), the first 32-bits of the address can estimate the region or the country the node is connected in. Thus, CHOP6's IPv6 prefix ID is used for proximity estimation if no RTT information is available during routing. With respect to the 3 qualitative criteria, *CHOP6* stands as follows:

Efficiency	*Overhead*	*Scalability*
low	low	medium

T2MC's [120] objective is to diminish the redundant multiple messages that may unnecessarily cross autonomous system (*AS*) boundaries. The latter represent links that are the costliest to ISPs and hence, minimizing topology mismatches greatly affects pertinent operational expenditure. To achieve that, each node exploits the stable structure of the Internet routers to help solve a clustering problem. The algorithm uses *traceroute*-logs to detect inter-*AS* boundaries and clusters peers to either nearby or far-away groups by applying a customized k-mean classification ($k = 2$). When joining the network, the peer randomly picks an IP and uses traceroute and selects the routers with minimum and maximum latencies in the above *traceroute* path as initial centroids for the "nearby" and "far-away" sets respectively. Following an iterative procedure, the node comes up with the two sets demonstrating minimum intra-set latency variance. At this stage, routers in the "far-away" set are not considered critical. In contrast, the router from the "near" set having the maximum number of hops from the just-joining peer is key as it denotes an edge–gateway. Peers that share the same edge–gateway or other members in the "nearby" router cluster are considered intra-*AS* neighbors.

Even though *traceroute* provides detailed information about the network structure, its extensive use may create overhead on the overall network structure. In this respect, administrators frequently disable *traceroute* support on their routers this undoubtly affects *T2MC*'s operation. Although overall experimental results have been encouraging, *T2MC* is considered rather coarse-grained as it only classifies neighbors into two groups [96]. *T2MC* fares as follows:

Efficiency	*Overhead*	*Scalability*
high	high	low

[4]http://www.maxmind.com

[5]http://www.iana.org/

PChord [121] is based on the *Chord DHT* and adds proximity into its routing mechanism. To achieve that, the standard finger table is augmented with a *proximity list* that contains nodes in the network found close-by in terms of RTT. In this respect, the next hop during routing can be decided based on both the direction towards the search key and minimizing the cost of that hop.

When a peer joins the network, its proximity list is empty. This list is maintained in a heuristic fashion during the node's lifetime. While a node attempts routing, the next hop is selected as the closest node to the target key from *both* its finger table and proximity list. An initial choice for the next hop made out of the finger list can be substituted by a shorter one from the proximity list if it is both closer to the key and in less physical distance. This yields an equal or even more efficient routing than traditional Chord.

Unfortunately the overlay maintenance cost is somewhat higher than *Chord* as additional heart-beat messages are required for connection verification for the entries of the proximity list. Fortunately, such entries represent nodes already close-by so the additional cost is kept to a minimum. However, *PChord* has been reported to exhibit slow convergence speed and inefficiency in high-churn environments [122]. *PChord* qualitatively behaves as follows:

Efficiency	*Overhead*	*Scalability*
medium	medium	medium

AChord [122] exploits IPv6's *anycast* mechanism to relieve the protocol from complex joining procedures, harness network proximity and achieve high routing efficiency. When sending a message to an IPv6 anycast group, the originator accomplishes this delivery by interacting with the physically closest node in that group [123]. Thus, *AChord* organizes all nodes participating in the overlay network into a single anycast group. An arriving peer is automatically forwarded to the physically nearest node for bootstrapping; this curtails the need to explicitly maintain participants and the effort of locating the physically nearest.

The *ID* of an incoming node is computed based on both the identifier of the bootstrap node and the bootstrap's predecessor; this is done in a way that the *ID* of the incoming peer is positioned *between* the two identifiers mentioned above. After joining, a *finger* table is created ala *Chord*. Moreover, an *adjacency table* maintains information about the k closest–known peers. The routing decision takes place in a way reminiscent to *Chord* and uses both adjacency and finger tables. *AChord* behavior has as follows:

Efficiency	*Overhead*	*Scalability*
medium	medium	medium

Chord6 [124] is another *Chord*-variant that uses IPv6's hierarchical features to create a substrate that reduces inter-domain traffic among service providers. The key difference between *Chord6* and *Chord* lies on the identifier definition. The adopted approach is deemed as easily portable to other *DHT*s including *CAN*, *Pastry* and *Tapestry*. *Chord6*'s identifier contains two parts: the higher bits are obtained by hashing the node's IPv6 address prefix (of specific length), while the remaining lower bits of the identifier is the hash-value of the remaining of the IPv6 address. As a result, nodes in a domain are mapped onto a continuous key space on the overlay network; the latter avoids unnecessary message–forwarding across different service providers and helps minimize overall routing cost.

Chord6 does reduce the average length of hops traversed to be logarithmically proportional to the number of available domains instead of the number of participating peers. Due however to the large number of Internet domains and the possibility of occasionally having only a small number of peers within each domain, [122] challenges the approach's scalability and efficiency. Overall, the approach fares as follows:

Efficiency	*Overhead*	*Scalability*
low	low	low

A proximity neighbor selection scheme functioning atop the *Chord DHT* is introduced in [125]. Its key objective was to group physically-close nodes as neighbors in the *DHT* table while exploiting proximity information. To detect proximity, virtual network coordinates for peers are used with the help of the Vivaldi protocol [58]. The virtual coordinates are then used to help map nodes to the identifier–space of the *DHT*. The space in question is partitioned using a *concentric circle clustering scheme* where successive cycles of radiuses ρ, 2ρ, 3ρ and so on, are constructed. The formed annuluses are then divided into $2\chi - 1$ *sectors*, where χ denotes the sequence number of the annulus starting from $\chi = 1$ for the center cycle. In this way, [125] prove that each sector This characteristic favors a more load–balanced clustering operation should we assume uniform node distribution. Every sector in this $2d$ space is mapped to a unique *region* in the *DHT* space forming a multi-layer node identifier space. Consequently, nodes from the same sector are mapped to the same region as well, preserving their proximity relationship.

Individual elements are uniquely mapped to the identifier–space, allowing logarithmic lookup operations with high probability on *Chord*. To this end, a node can obtain its *DHT* key and its region key, in a fully distributed manner, just by applying a consistent hash function. The node can then communicate with the region's master node called *Cluster Node (CN)* which is responsible for clustering the nodes belonging in the same sector or region with that of the node at hand. The node registers to its corresponding region and publishes its information, through the special peer *CN*; the peer can also ask *CN* for other nodes that have previously joined the region to enhance the peer's neighbor set for future routing table optimization. If no other nodes are found within the region, the search is expanded to neighboring regions. Even though scaling appeared promising, only limited simulations were carried out with configurations involving up to 2,048 nodes. Moreover the approach assumes uniform node distribution which is not always the case, especially in a synthetic coordinate system. *Chord DHT* fares as follows in reference to the stated 3 criteria:

Efficiency	*Overhead*	*Scalability*
medium	medium	medium

[126] proposes *Quasi-Chord*, a *Chord*-based network that constructs its overlay network while attempting to match the

underlying network topology. The overlay creation follows 3 stages: first, each host acquires its coordinates in a geometric space using the *Global Network Position (GNP)* protocol [60] and a set of landmarks. Second, using the *Cantor* space–filling curve, the 2-dimensional is converted to 1-dimensional space. The latter is employed in the third and final stage to build the *Quasi-Chord* circle according to the host's *Cantor* value. Physical proximity is thus denoted by *Cantor* value proximity. The *Chord* cycle is constructed by sorting nodes in ascending order. *Quasi-Chord* qualitative behavior has as follows:

Efficiency	*Overhead*	*Scalability*
medium	high	medium

Proximity neighbor selection algorithms often use probing to detect proximity. However, such methods suffer by imprecisions often due to network overload. IP-based clustering or *IPBC* [127] uses IP address prefixes (16 bits for IPv4) to detect proximity. Such proximity information can be easily stored in the DHT itself.

Each peer in the overlay has a unique *ID* and ~~and~~ an IP address to characterize it. To publish itself, the peer hashes a fixed-length prefix of its IP to obtain a key and stores the resulting *ID* and IP in the DHT. Newly joining nodes with the same *IP* prefix can, thus, query the *DHT* and readily identify all neighbors with the same prefix. Nodes periodically update their entries in the *DHT* by removing those who leave or have timed-out.

The overhead of the algorithm is very low as proximity detection requires only a comparison of prefixes. On the other hand, the efficiency of the proximity is caught into a vicious cycle with the storage space needed for *DHT* object publishing. Moreover, its static nature assumes high correlation of common IP prefix length of participating peers and latency (with no other measurements), which might not always be the case, especially in cases of high-churn environments. [128]. The suggested *IPBC* approach fares as follows:

Efficiency	*Overhead*	*Scalability*
low	low	medium

DynaMO [112] attempts to create an overlay that does not only consider physical proximity amongst peers but also takes into account mobility attributes of nodes. To adapt to mobile ad-hoc networks, *DynaMO* strives to maintain an evenly–distributed overlay *ID*-space so that hot-spots are avoided. Although based on *Pastry* and its well understood built-in locality heuristics [129], *DynaMO*'s own mechanisms are DHT-independent and so, they can be applied to other overlays. *DynaMO*'s approach may be viewed as bottom-up.

An incoming node computes physical information about its neighborhood and uses the outcome to unilaterally assign itself an *ID*. The approach suggests that for every new overlay routing-hop the physical distance between nodes is likely to increase resulting in a dominating last routing step [45], [129]. To identify this dominating step, two techniques are used: *Random LandMarking (RLM)* and *Closest Neighbor Prefix Assignment (CNPA)*. In *RLM*, a set of peers are chosen to be responsible for a set of landmark keys. These keys have to be chosen in a way that they equally divide the overlay space (i.e., in a 16–base an appropriate split would be 000..0, 100..0, 200..00, ... , F00..00).

A new-coming peer then computes distances from those landmarks, sorts them, and assigns itself an *ID* on this landmark ordering as follows: the peer adopts sets the prefix of its *ID* to be equal to that of its closest landmark. The length of this prefix is $l_{prefix} = \lfloor log_b k \rfloor$, where b is the *ID*–base and k the number of landmarks. The remaining of bits of the peer-*ID* are assigned either randomly or computed with additional proximity optimizations. In this way, two desirable features are attained: *i*) landmark assignment is dynamic, and *ii*) the set of a node leaf may reference peers that are physically close by. The latter in particular helps reduce average routing cost(s) for the leaf-set of a node is used to efficiently determine the last routing step.

CNPA is proposed for network setups for which a lightweight bootstrap phase is required. It delegates the task of identifying the physically close–node to *Pastry*; this node is used by the incoming peer to get bootstrapped. The newcomer takes the bootstrapping nodes's *ID* prefix and fills the remaining bits of its key using the *RLM* technique. *RLM* gets evaluated as presenting good proximity in a load-balanced environments. Moreover, it is adaptable to high-churn settings. Unfortunately, *RLM* bootstrap overhead is substantial and can effect the set of landmarks [112]. *CNPA* on the other hand, incurs less overhead but this come at the expense of being more coarse-grained as it is based on prefixes to determine proximity during. *DynoMO*'s elements fare as follows regarding our 3 criteria:

	Efficiency	*Overhead*	*Scalability*
RLM	medium	medium	high
CNPA	medium	low	medium

3D Overlay (3DO) [130] seeks to address the network mismatch problem for peers operating in mobile networks. As its name suggests, the network logical space is a 3–d space and the used "transient" (continually changing) logical *ID*s (*LID*s) consist of 3 axes, x, y and z in the form of a tuple; each such axis value is represented by an M-bit identifier. A newly-arriving peer broadcasts a join request using a *TTL*-based expanding ring search helping locate peers that are physically adjacent. Responding peers send in their *LID*s along with a list of their neighbors and their corresponding distances from those neighbors. Once a joining peer receives the above responses, it stores requisite information in its own routing table and so, it generates an undirected connected graph of itself with both its 1- and 2-hop neighbors. Starting off this graph and with the help of hop distances obtained from the underlying MANET, the incoming peer builds a minimum spanning tree (MST) with itself as the root node; in this MST, the nodes adjacent to the root become the direct neighbors of the peer. Moreover, the entries of the initial routing table that do not correspond to above direct neighbors are eliminated.

The above procedure ascertains that close peers on the overlay are also adjacent at the physical layer. At this stage, a peer can generate its *LID* that automatically places it in the 3-d network space. Updates in the peer's connectivity information reflecting continual changes in the mobile network, trigger updates on the composition of *LID* itself.

3DO stores objects in the same 3-d space. For each file f, a key k of the form of an x, y and z tuple is hashed out of the file's unique *ID* (e.g., filename). The node whose *LID* is closer to k stores the file itself. To retrieve the file, the requesting node calculates f's k and issues a query message. The message is ~~a~~ associated with a *TTL* value. Each peer forwards that query to the node, in its routing table, that is closest to k until either the peer that stores the file is located or the query's lifetime expires.

3DO focuses on the creation of a 3-d space whose stated objective is to yield improved load balancing and exploit fine–grained information to ensure that each hop on the overlay during object location, is also closer physically to the sought object. However, the use of *TTL* in query messages renders the network not–a–pure DHT as it abolishes the zero false–negative queries that traditional DHTs maintain. [130] also concedes that the approach does not rapidly adapt to network changes and so its evaluation was solely based on low mobility scenarios. For the above reasons, *3DO* exhibits the following behavior in conjunction with our three stated criteria:

Efficiency	*Overhead*	*Scalability*
low	low	medium

SAT-Match [105] defines an iterative process that adaptively changes the overlay network to ease the topology mismatch problem. It consists of a *probing phase* in which a peer queries nearby nodes for distances and a *jumping phase* in which the peer decides to change its neighborhood in order to reduce average neighbor distances. This process is performed iteratively until no additional optimization is required; yet, the methods achieves optimization within a sufficiently large scope. The notion of *stretch* is used as the means for quantifying the *topology match degree* of the constructed overlay; stretch is defined as: $S = \bar{L_l}/\bar{L_p}$ where $\bar{L_l}$ is the average logical link latency and $\bar{L_p}$ is the average physical link latency. When a peer joins the *DHT*, it starts its *probing phase* by contacting the neighborhood with messages having small *TTL=k* [131]. Recipients return information to the source and forward the message if not expired. After discovering its *TTL-*k neighborhood, the source measures *RTT*-times, sorts them and picks the two nodes with the smallest *RTT*s.

During the *jumping phase*, the source node calculates the stretch change to all its direct neighbors and to the direct neighbors of the first peer that was selected in the probing phase, as if the jump has been performed. Using heuristics, the node determines if the stretch reduction is sufficient to perform the actual jump; otherwise, it selects the second peer from the probing phase and repeats the computation. If again, the threshold is not met, then no jump is ultimately carried out.

Experimentation shows that *SAT-Match* attains a 27% stretch reduction over a landmark-enhanced *CAN*-alternative. This result shows that *SAT-Match* may perform better than landmark binning in terms of the matching degree of the logical to the physical networks. *SAT-Match* behaves as follows with respect to the stated 3 criteria:

Efficiency	*Overhead*	*Scalability*
high	medium	medium

B. Discussion on the Algorithms for Structured Architectures

In this section, we have discussed research efforts trying to address the issue of inefficient network resource utilization manifested by network–topology mismatches in *structured P2P*-systems. Based on the salient feature of outlined techniques [109], [102], [110], we classify furnished solutions based on the following aspects:

i) geographic-layout,
ii) proximity routing, and
iii) proximity neighbor selection,

Similarly to unstructured approaches, most of the methods reviewed in this section entail features from multiple of the above categories. For example, [118] uses both proximity routing and proximity neighbor selection to achieve topology adaptation, while *LAPTOP* combines geographic–layout with replication strategies to refine its results. The heuristic nature of the suggested solutions opens them up to trade-offs which, in most cases, are highly affected by the application domain and the implementation itself.

In what follows, we place the surveyed *structured P2P*-methods in the three above categories formulated based on utilized proximity and geographic aspects. Table II offers a comparative view of the algorithms in a way reminiscent to that used in Table I. Table II outlines key properties for all surveyed approach that helps differentiate and better position competing techniques. Clearly, we do not further elaborate topics that we have included and discussed at length in Section III; these aspects entail topology adaptation, landmarking, and caching/replication. Finally, Figures 13, 14 and 15 help with the pictorial comparison of the algorithms' bahavior with respect to the three effectiveness criteria we defined in subsection II-C.

1) Geographic-layout: This category consists of methods that try to map the overlay id space on to the physical one, as depicted in Figure 12. Routing tables of involved peers, are adjusted and maintained by exploiting proximity information that describes the overall (geographic) positioning of the participants. This comes in contrast to proximity routing and proximity neighbor selection (discussed later on) where decisions are taken based on and effecting a peer's nearby environment.

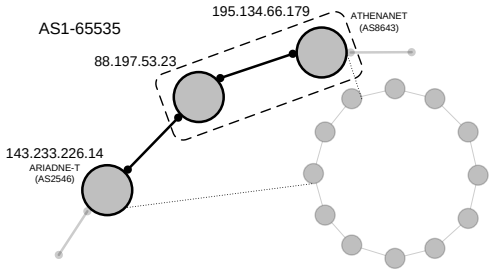


Fig. 12. Successive nodes in the overlay circle can be chosen to be from the same or near autonomous systems (*AS*).

A number of the reviewed algorithms including *Global Soft-State*, *Mithos* and *LAPTOP* resort to clustering close-by nodes in order to reduce the network's average delay.

TABLE II: Summary table for structured algorithms.

Algorithm / Paper	Topology Adaptation	Landmarking	Proximity Routing	Proximity Neighbor Selection	Geographic Layout	Caching / Replication	Highlights	Pros / Cons
Global Soft-State [106]	☑	☑	☐	☐	☑	☐	Hybrid landmark binning and probing scheme for proximity detection. Strategic injection of proximity information across the network. Subscription-notification system to dynamically adapt to network changes.	+ Greatly reduces routing latency to far away nodes. – Potential maintenance costs are high. – Little knowledge about neigboring regions. – Unable to identify nodes that are close to routers/gateways. – Static nature due to the use of landmarks.
Mithos [113]	☐	☐	☑	☐	☑	☐	Directed incremental probing. Synthetic coordinates.	+ Distance measurement is done on the overlay level. – Does not effectively manage dynamic peer arrival and leave. – Centralized implementation of the its spring relaxation technique. implementation
LAPTOP [117]	☑	☐	☐	☐	☑	☑	Tree-based hierarchy.	+ Reduces hops during message routing. + Minimal maintenance overhead. – Heartbeat approach incurs overhead even when not needed.
[118]	☑	☐	☑	☑	☐	☐	Replacing XOR metric with a function that minimizes the underlying cost. Clustering (MaxMind geolocation technology).	+ Proximity routing works in Kademlia to improve connection locality. – Using virtual coordinates renders the approach rigid in high churn environments. ~~environments.~~
CHOP6 [119]	☐	☐	☑	☐	☐	☐	Ipv6 format exploitation. Additional RTT information.	+ Relatively straightforward integration of IPv6 global routing prefix into ID space. – Coarse-grained. – IPv6 is still not widely applied and supported.
T2MC [120]	☐	☑	☑	☐	☐	☐	Clustering using trace-route logs.	+ Prioritize interaction of peers and edge gateways. – Coarse-grained. – Trace-route extends overhead and is sometimes disabled by ISPs.
PChord [121]	☐	☐	☑	☐	☐	☐	Maintenance of a proximity list.	+ Can constrain costly jumps in and out of network partitions. + Maintenance cost kept to a minimal (node join/leave heartbeat for lists). – Rigid in high churn environments.
AChord [122]	☐	☐	☑	☑	☐	☐	Exploits IPv6's any-cast mechanism.	+ Delegates proximity calculation during bootstrap to the anycast mechanism. + Easy to implement (just the additional neighborhood table). – No adaptation after node bootstrap.

Continued on next page

TABLE II – *Continued from previous page*

Algorithm / Paper	Topology Adaptation	Landmarking	Proximity Routing	Proximity Neighbor Selection	Geographic Layout	Caching / Replication	Highlights	Pros / Cons
Chord6 [124]	☐	☐	☑	☐	☐	☐	Exploits IPv6's hierarchical features.	+ Reduces inter-domain traffic between ISPs. + Easily portable to other DHTs. – Cannot distinguish between nearby domains. – Not adaptable after id assignment
DHT-PNS [125]	☐	☐	☐	☑	☐	☐	Grouping through synthetic coordinates. Partitioning using a concentric cycle clustering scheme.	+ Good convergence speed and optimizing performance. – It assumes uniform node distribution which is not always the case.
Quasi-Chord [126]	☐	☑	☐	☑	☐	☐	Geometric space coordinates (GNP) Transformation of the coordinate space into 1-d Cantor space for easier mapping to the Chord hierarchy Two finger tables (clockwise, anti-clockwise)	– Not fully distributed (GNP is landmark based). – Makes an indirect assumption of a maximum number of allowable hosts. – Doubles the required routing information which needs to be created and maintained.
IPBC [127]	☐	☐	☐	☑	☐	☐	IP address prefixes (16 bit for IPv4) to detect proximity.	+ Prefix is stored in the DHT so the proximity identification becomes as easy as to query the prefix. + DHT maintenance mechanisms for both voluntary or ungraceful departures of peers. – Performance/accuracy trade-off in choosing the prefix length to be used.
DynaMo [112]	☐	☑	☐	☑	☑	☐	Random Landmarking (RLM). Closest Neighbor Prefix Assignment (CPNA). Making last hop as local as possible.	+ Developed with mobile, ad-hoc networks in mind. + Evenly distributed IDs. + Dynamic landmarking is a fully distributed approach. – RLM may impose extra overhead to landmark nodes. – CPNA is a coarse-grained proximity approach.
3DO [130]	☑	☐	☑	☑	☐	☑	3 dimentional space.	+ Fine granularity of peer and object embedding (eight octants). + Load balance. – DHT with false negative queries. – Slow adaptation to network changes.

Continued on next page

TABLE II – *Continued from previous page*

Algorithm / Paper	Topology Adaptation	Landmarking	Proximity Routing	Proximity Neighbor Selection	Geographic Layout	Caching / Replication	Highlights	Pros / Cons
SAT-Match [105]	☑	☐	☐	☑	☐	☐	Selective jumps to adjust peer positioning in the DHT. Stretch reduction scheme.	+ Continuously adaptive mechanism. + It can be considered lightweight. + Can coexist with other approaches (like landmarking). + Works sufficiently for large scopes as well as environments with high churn. + Compared to Mithos, scales much better. – Needs synchronization within zones for every probing/jump for each node. – Contention situation in selective jump phase. – Probing phase incurs unnecessary traffic.

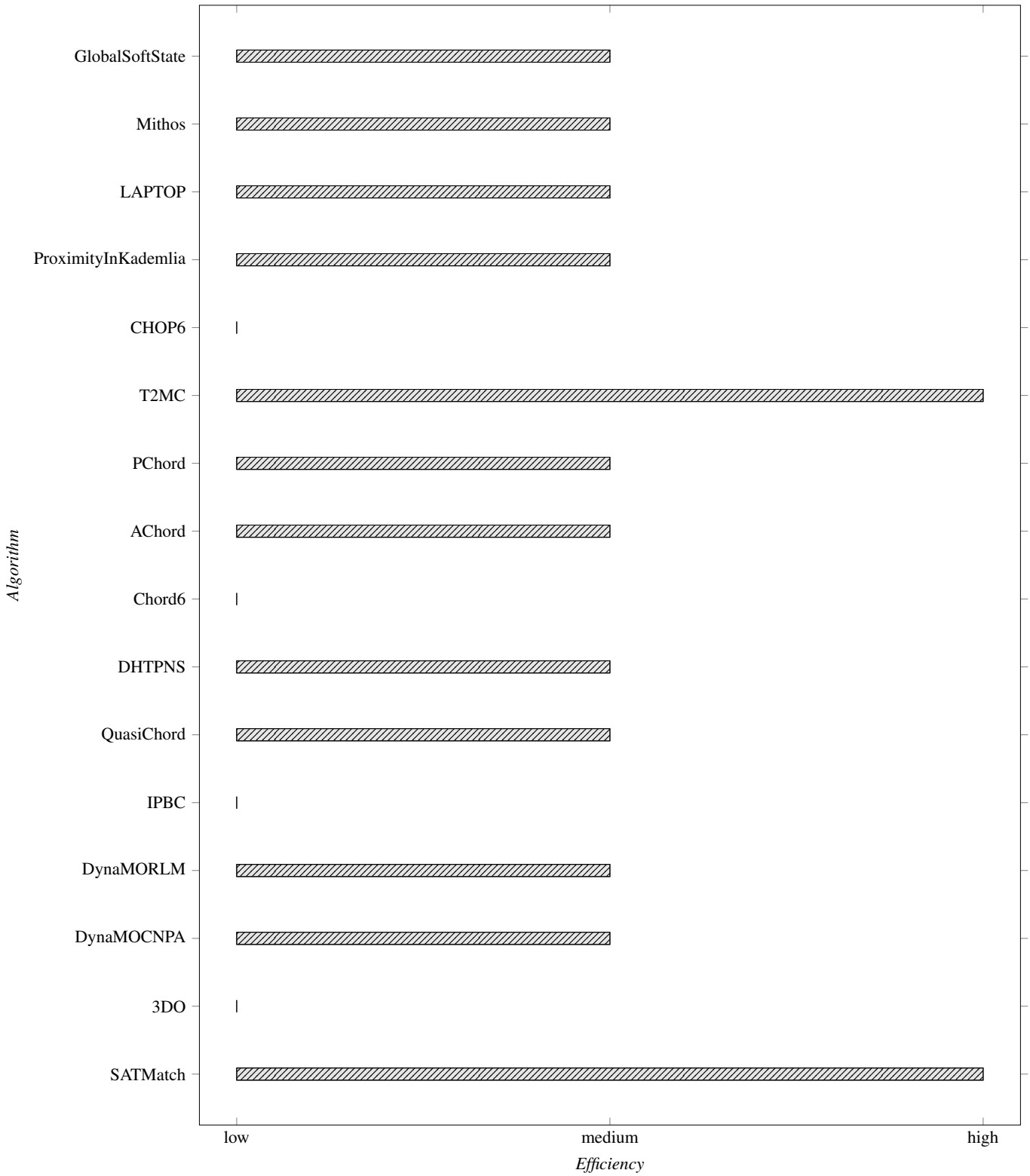


Fig. 13. *Efficiency* Pictorial Comparison of Structured Approaches.

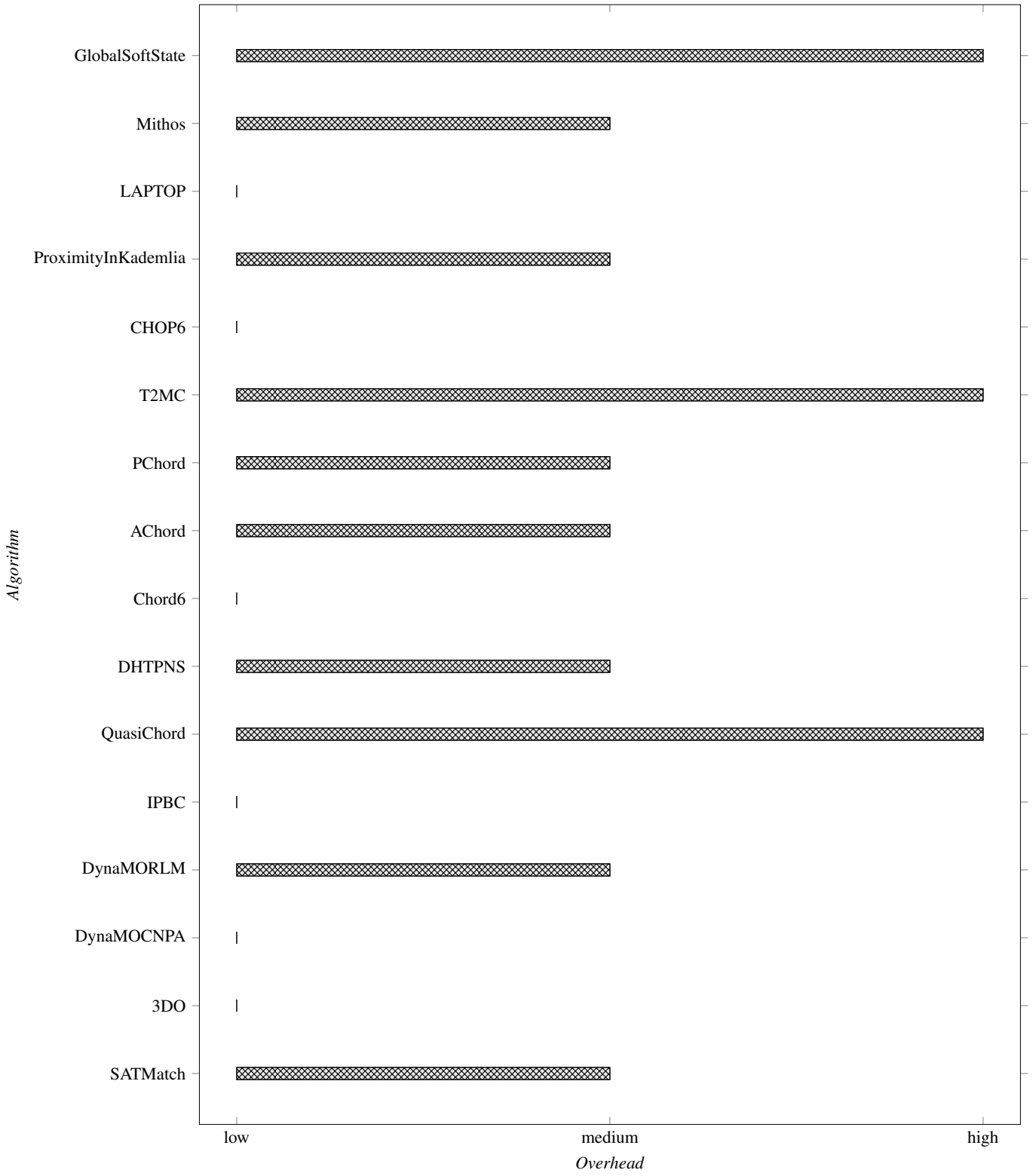


Fig. 14. *Overhead* Pictorial Comparison of Structured Approaches.

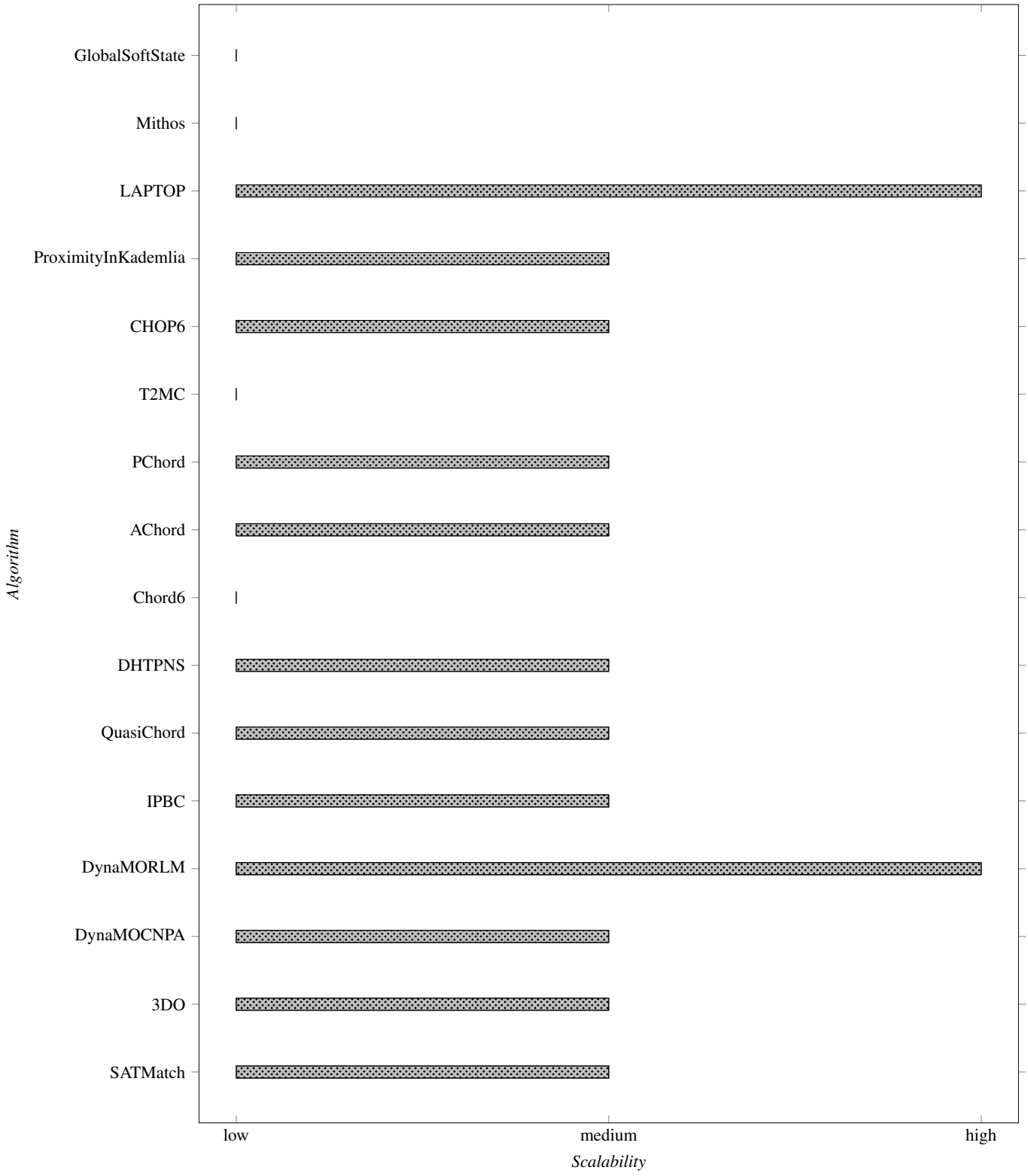


Fig. 15. *Scalability* Pictorial Comparison of Structured Approaches.

Landmark servers and *RTT* measurements are two popular methods, which can also be used in conjunction, to discover peers in physical proximity. However, as already discussed, these methods do not always yield reliable estimates for node positions over the Internet. Landmark servers are not self-organizing and present maintenance overhead. Moreover, to serve a structured *P2P*–network with potentially tens of thousands or more peers, multiple such servers are required to be distributed over the world, which is a difficult proposition per se. *RTT*s can measure the delay between peers, but their use constitutes a greedy method that can result in sub-optimal overlay topologies; the latter is especially true if nearby nodes happen to be connected through low bandwidth connections.

On one hand, approaches based on *geographic layout* improve the query efficiency of the system in general and are especially effective when applied to *CAN*-like implementations. On the other hand, they inherently undermine the uniform distribution of peer identifiers. This is an important downside as it creates hot-spots that hurt system scalability. Moreover, it adversely affects system availability as nearby nodes are more likely to suffer collective failures [132].

2) Proximity Routing (PR): This category of approaches use physical proximity knowledge to decide the best next hop during routing. As shown in Figure 16, maintenance of routing tables themselves is a separate process based only on the system's DHT construction mechanism and overlay ID distribution.

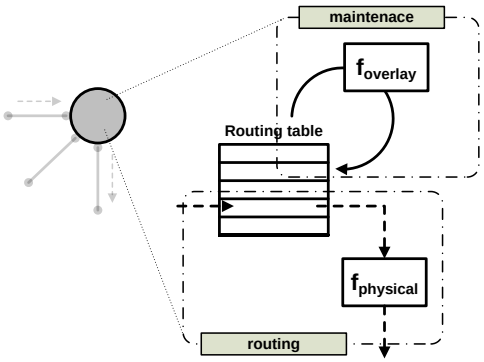


Fig. 16. Proximity routing takes proximity information into account during the routing process.

We reviewed an array of methodologies that strictly adhere to the tenets of this category such as *Mithos* and *PChord* as well as others that function along with proximity neighbor selection such as *DynaMO*. When *PR*–based algorithms route messages, they try to balance between choosing the node that will further progress the routing towards the destination and picking the closest entry in the routing table in terms of network proximity. In the degenerate case, a system could use a greedy approach to select only the lowest latency candidate peers during forwarding. That would to longer, sub–optimal, paths to the destinations; thus imposing extra stress to network resources. PR is also less effective than geographical layout when applied to CAN–like implementations.

3) Proximity Neighbor Selection (PNS): Algorithms in this category use proximity data during routing table construction and maintenance along the lines of Figure 17. The proximity

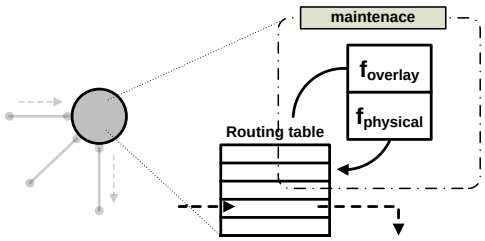


Fig. 17. Proximity neighbor selection dictates that proximity is taken into account during routing table maintenance.

information used is different from that of landmark-based systems described in Section IV-B1; here, *RTT* values between nodes (i.e., *SAT-Match*) or *IP* address prefixes (i.e., *DynaMO*) are predominantly used to detect proximity. [133] states that 97% of prefixes larger than 24 bits belong to a single geographical location. However, using a smaller number of bits creates less precise results and a larger number of bits may increase the burden in the network and simultaneously reduce the possible number of neighbors. Moreover, [121] argue that prefix–based approaches can destroy the uniform distribution in the key space and do not work in one-dimensional key space where the mapping is restricted to operate at the overlay.

Therefore, a careful selection is required in terms of performance/accuracy trade-off. *Tapestry*, *Pastry*, and *CAN* successfully incorporated proximity neighbor selection into their methods. The routing protocol in *Pastry* is based on longest node *ID* prefix matching, while *CAN* uses *RTT* values to detect nearby nodes. [129] reports that proximity neighbor selection is an effective proximity based method. In general, proximity neighbor selection is considered superior to proximity routing, however, simultaneous use of both are also possible.

V. Conclusion

Research and development in applications and systems that follow the *P2P* architectural paradigm have flourished for more than a decade now. By and large, such artifacts exploit the loose-coupling and self-organization of participating nodes to shape substrates upon which diverse applications can run; the latter offer services including, but not limited to, distributed naming, Internet telephony and video conferencing, storage and indexing, content and file sharing, web crawling and caching, event notification and subscribing/publishing. These services are provided atop the best-effort infrastructure of today's Internet and they have to effectively deal with an array of key quality attributes that they may have to offer, such as node availability, robustness, scalability, load-sharing, quality-of-service and user anonymity. The difficulty to handle such features effectively, emanates from the fact that peer nodes operate at the edge of the Internet and no adequate attention is paid to the structure of the physical network during the network formation at the application-level. The deviation of the distributed logically-formed overlay from the physical network yields suboptimal use of the underlying infrastructure and is known as the topology mismatch problem.

Development of *P2P*–systems and applications typically involve juggling a set of tradeoffs (e.g., choosing higher accuracy at the cost of increased system overhead). In surveying more than a decade's worth of research efforts aimed at solving the topology mismatch problem in both unstructured and structured *P2P* networks, we find this tussle-of-tradeoffs property to hold. With regards to the three criteria we used –efficiency, overhead, and scalability– we find that none of the proposed solutions is superior to the others on all three fronts. This is not surprising. Nonetheless, by *i*) presenting an analysis of each of the solutions, including what distinguishes it from others as well as its advantages and disadvantages and *ii*) offering a pictorial comparison of how each solution fares in regards to others as far as efficiency, overhead, and scalability are concerned, we hope to have provided *P2P* developers and researchers with enough insight and perspective; as they face specific problems, they will be readily able to draw the best possible design decisions for their own application environments.

REFERENCES

[1] BitTorrent, Inc., "*BitTorrent* and *µTorrent* Software Surpass 150 Million User Milestone; Announce New Consumer Electronics Partnerships," http://www.bittorrent.com/intl/es/company/about/ces_2012_150m_users, January 2012, online; accessed April 24th, 2015.

[2] TELECOMPAPER BV., "*Skype* grows FY Revenues 20%, reaches 663 mln Users," http://www.telecompaper.com/news/skype-grows-fy-revenues-20-reaches-663-mln-users, March 2011, online; accessed March 18th, 2015.

[3] J. Mercier, "50 Million Concurrent Users Online!" skypenumerology.blogspot.se/2013/01/50-million-concurrent-users-online.html, January 2013, online; accessed on April, 19th 2015.

[4] I. TeleGeography/PriMetrica, "Skype traffic continues to thrive," https://www.telegeography.com/products/commsupdate/articles/2014/01/15/skype-traffic-continues-to-thrive/, January 2014, online; accessed on April, 20th 2015.

[5] YaCy Project, http://yacyweb.de, 2012, online; accessed on March 14th, 2015. [Online]. Available: http://yacyweb.de

[6] J. Kubiatowicz, D. Bindel, Y. Chen, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao, "OceanStore: An Architecture for Global-scale Persistent Storage," in *Proc. of the ACM ASPLOS-IX*. Cambridge, MA: ACM, November 2000.

[7] W. J. Bolosky, J. R. Douceur, D. Ely, and M. Theimer, "Feasibility of a Serverless Distributed File System deployed on an Existing Set of Desktop PCs," *Performance Evaluation Review*, vol. 28, no. 1, pp. 34–43, June 2000, aCM Sigmetrics.

[8] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica, "Wide-area Cooperative Storage with CFS," in *Proc. of the 18th ACM Symposium on Operating Systems Principles (SOSP)*. Banff, Canada: ACM, 2001, pp. 202–215.

[9] P. Druschel and A. Rowstron, "PAST: A Large-Scale, Persistent Peer-to-Peer Storage Utility ," in *Proc. of 8th Workshop on Hot Topics in Operating Systems (HotOS)*. Elmau/Oberbayern, Germany: IEEE Computer Society, May 2001, pp. 75–80.

[10] A. Adya, W. Bolosky, M. Castro, R. Chaiken, G. Cermak, J. Douceur, J. Howell, J. Lorch, M. Theimer, and R. Wattenhofer, "FARSITE: Federated, Available, and Reliable Storage for an Incompletely Trusted Environment," in *Proc. of the 5th Symp. on Operating Systems Design and Implementation (OSDI)*. Boston, MA: USENIX, 2002.

[11] A. Muthitacharoen, R. Morris, T. M. Gil, and B. Chen, "Ivy: A Read/Write Peer-to-Peer File System," in *Proc. of 5th Symposium on Operating Systems Design and Implementation (OSDI)*. Boston, MA: USENIX Association, December 2002.

[12] B. Grönvall, A. Westerlund, and J. Danielsson, "Arla Project," http://www.stacken.kth.se/project/arla, 2010, online; accessed on April 20th, 2015.

[13] L. O. Alima, A. Ghodsi, S. El-Ansary, P. Brand, and S. Haridi, "Multicast in DKS(N, k, f) Overlay Networks," in *Proc. of the 3rd Int. Conf. on Peer-to-Peer Computing (P2P)*. Linkoping, Sweden: IEEE, 2003.

[14] S. Iyer, A. Rowstron, and P. Druschel, "Squirrel: A Decentralized Peer-to-Peer Web Cache," in *Proc. of 21st Annual Symposium on Principles of Distributed Computing (PODC)*. Monterey, CA: ACM, July 2002.

[15] R. J. Bayardo Jr., R. Agrawal, D. Gruhl, and A. Somani, "YouServ: a Web-hosting and Content Sharing Tool for the Masses," in *Proc. of the 11th Int. Conf. on World Wide Web (WWW)*. Honolulu, HI: W3C, 2002.

[16] M. Waldman, A. Rubin, and L. Cranor, "Publius: A Robust, Tamper-evident, Censorship-Resistant Web Publishing System," in *Proc. 9th USENIX Security Symposium (SSYM)*. Berkeley, CA: USENIX Association, 2000, pp. 59–72.

[17] M. Waldman and D. Mazières, "Tangler: a Censorship-Resistant Publishing System Based on Document Entanglements," in *Proc. of 8th ACM Conf. on Computer and Communications Security (CCS)*. Philadelphia, PA: ACM, 2001, pp. 126–135. [Online]. Available: http://dx.doi.org/10.1145/501983.502002

[18] Microsoft, "Three Degrees," http://web.archive.org/web/20040103063458/http://threedegrees.com/, 2012, archive; accessed on March 14th, 2015. [Online]. Available: http://web.archive.org/web/20040103063458/http://threedegrees.com/

[19] A. Papadimitriou and A. Delis, "Data Dissemination in Unstructured Peer-to-Peer Networks," in *Proceedings of the 37th Intrnational Conference on Parallel Processing (ICPP'08)*. Portland, OR: IEEE, September 2008.

[20] A. I. T. Rowstron, A. M. Kermarrec, M. Castro, and P. Druschel, "SCRIBE: The Design of a Large-Scale Event Notification Infrastructure," in *Proc. of the 3rd Int. Workshop in Networked Group Communications (NGC)*. London, UK: LCCS 2233, Springer, November 2001, pp. 30–43.

[21] M. Castro, P. Druschel, A. M. Kermarrec, and A. I. T. Rowstron, "Scribe: a Large-Scale and Decentralized Application-level Multicast Infrastructure," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 8, pp. 1489–1499, October 2002.

[22] R. Cox, A. Muthitacharoen, and R. Morris, "Serving DNS Using a Peer-to-Peer Lookup Service," in *Peer-to-Peer Systems*, ser. Lecture Notes in Computer Science, P. Druschel, F. Kaashoek, and A. Rowstron, Eds. Berlin, Heidelberg: Springer Berlin / Heidelberg, Oct. 2002, vol. 2429, ch. 15, pp. 155–165. [Online]. Available: http://dx.doi.org/10.1007/3-540-45748-8_15

[23] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong, "Freenet: A Distributed Anonymous Information Storage and Retrieval System," in *Workshop on Design Issues in Anonymity and Unobservability*. Berkeley, CA: Springer, 2000, pp. 46–66.

[24] R. Mondéjar, P. García-López, C. Pairot, and L. Pamies-Juarez, "CloudSNAP : A Transparent Infrastructure for Decentralized Web Deployment using Distributed Interception," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 370–380, January 2013.

[25] S. Saroiu, K. P. Gummadi, R. J. Dunn, S. D. Gribble, and H. M. Levy, "An Analysis of Internet Content Delivery Systems," in *Proc. of 5th Symp. on Operating Systems Design and Implementation (OSDI)*. Boston, MA: ACM, December 2002, pp. 315–327.

[26] S. Sen and J. Wang, "Analyzing Peer-to-Peer Traffic Across Large Networks," *IEEE/ACM Transactions on Networking*, vol. 12, no. 2, pp. 219–232, April 2004.

[27] T. Karagiannis, P. Rodriguez, and K. Papagiannaki, "Should Internet Service Providers Fear Peer-assisted Content Distribution?" in *Proc. of the 5th ACM SIGCOMM Conf. on Internet Measurement (ICM)*. Berkeley, CA: USENIX Association, 2005, p. 6. [Online]. Available: http://portal.acm.org/citation.cfm?id=1251092

[28] D. Ferguson, "P2P File Sharing - The Evolving Distribution Chain," http://retroshare.sourceforge.net, June 2006. [Online]. Available: http://retroshare.sourceforge.net

[29] H. Schulze and K. Mochalski, "Ipoque: Internet study 2008/2009," https://www.christopher-parsons.com/Main/wp-content/uploads/2009/04/ipoque-internet-study-08-09.pdf, 2009. [Online]. Available: https://www.christopher-parsons.com/Main/wp-content/uploads/2009/04/ipoque-internet-study-08-09.pdf

[30] Cisco, "White paper: Cisco visual networking index: Forecast and methodology, 2013 - 2018," http://www.cisco.com/c/en/us/solutions/collateral/service-provider/ip-ngn-ip-next-generation-network/white_paper_c11-481360.html, Tech. Rep., June 2014. [Online]. Available:

http://www.cisco.com/c/en/us/solutions/collateral/service-provider/ip-ngn-ip-next-generation-network/white_paper_c11-481360.pdf

[31] Gnutella, "The Gnutella Protocol Specification v0.6," http://rfc-gnutella.sourceforge.net/, 2009, online; accessed August 15th, 2014.

[32] S. Saroiu, P. K. Gummadi, and S. D. Gribble, "A Measurement Study of Peer-to-Peer File Sharing Systems," in *Proc. of SPIE Multimedia Computing and Networking (MMCN)*. San Jose, CA: SPIE, Volume 4673, January 2002. [Online]. Available: citeseer.nj.nec.com/saroiu02measurement.html

[33] E. Adar and B. A. Huberman, "Free Riding on Gnutella," *First Monday*, vol. 5, no. 10, p. 2000, 2000.

[34] D. Hughes, G. Coulson, and J. Walkerdine, "Free Riding on Gnutella Revisited: The Bell Tolls?" *IEEE Distributed Systems Online*, vol. 6, no. 6, pp. 1–13, 2005.

[35] D. Hughes, J. Walkerdine, and K. Lee, "Socio-Technical Issues in P2P File Sharing Systems," http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.62.1347, 2008, online; accessed in January 2015.

[36] *Napster*, http://en.wikipedia.org/wiki/Napster, 2007, online; accessed on April 20th, 2015.

[37] JupiterMediaMetrix, "Global Napster Usage Plummets, But New File-Sharing Alternatives Gaining Ground, Press Release," http://web.archive.org/web/20080413104420/http://www.comscore.com/press/release.asp?id=249, July 2001, online; accessed October 9th, 2014. [Online]. Available: http://web.archive.org/web/20080413104420/http://www.comscore.com/press/release.asp?id=249

[38] B. Yang and H. Garcia-Molina, "Improving Search in Peer-to-Peer Networks," in *Proc. of the 22d Int. Conf. on Distributed Computing Systems (ICDCS)*. Vienna, Austria: IEEE Computer Society, July 2002, pp. 5–14.

[39] D. Stutzbach and R. Rejaie, "Understanding Churn in Peer-to-Peer Networks," in *Proc. of the 6th ACM SIGCOMM Conf. on Internet Measurement (IMC)*. Rio de Janeiro, Brazil: ACM, 2006, pp. 189–202.

[40] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications," in *Proc. of 2001 ACM SIGCOMM Conf.* San Diego, CA: ACM, 2001, pp. 149–160.

[41] P. Maymounkov and D. Mazières, "Kademlia: A Peer-to-Peer Information System Based on the XOR Metric," in *Proc. of the 1st Int. Workshop on Peer-to-Peer Systems (IPTPS)*. London, UK: Springer-Verlag, 2002, pp. 53–65.

[42] Kazaa, "http://www.kazaa.com/," 2009. [Online]. Available: http://www.kazaa.com/

[43] *Skype*, http://www.skype.com, 2012, online; accessed on April 24th, 2015.

[44] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker, "A Scalable Content-Addressable Network," in *Proc. of the 2001 ACM SIGCOMM Conf.* San Diego, CA: ACM, 2001, pp. 161–172.

[45] A. I. T. Rowstron and P. Druschel, "Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems," in *Proc. of the IFIP/ACM Int. Conf on Distributed Systems Platforms (Middleware)*. Heidelberg, Germany: Springer-Verlag, November 2001, pp. 329–350.

[46] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph, "Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing," University of California–Berkeley, Berkeley, CA, Tech. Rep., April 2001, tR No. UCB/CSD-01-1141.

[47] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz, "Handling Churn in a DHT," in *Proc. of the USENIX Annual Technical Conf. (ATC)*. Boston, MA: USENIX Association, July 2004. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.10.9928

[48] Overnet, "http://www.overnet.org/," 2009. [Online]. Available: http://www.overnet.org/

[49] B. Cohen, "Incentives Build Robustness in BitTorrent," in *Workshop on Economics of Peer-to-Peer Systems*. Berkeley, CA: UC-Berkeley, June 2003.

[50] Y. Chu, S. G. Rao, and H. Zhang, "A Case for End System Multicast," in *Proc. of the ACM SIGMETRICS Int. Conf.* Santa Clara, CA: ACM, 2000, pp. 1–12.

[51] J. Ritter, "Why Gnutella Can't Scale. No, Really." http://www.darkridge.com/~jpr5/doc/gnutella.html, 2001, online; accessed on November 30th, 2014.

[52] M. Ripeanu, A. Iamnitchi, and I. Foster, "Mapping the Gnutella Network," *IEEE Internet Computing*, vol. 6, no. 1, pp. 50–57, January 2002.

[53] Y. Chu, S. Rao, S. Seshan, and H. Zhang, "A Case for End System Multicast," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 8, pp. 1456–1471, October 2002.

[54] J. F. Buford, H. Yu, and E. K. Lua, *P2P Networking and Applications*. Burlington, MA: Morgan Kaufmann, March 2009.

[55] Y. Chawathe, "An Architecture for Internet Content Distribution as an Infrastructure Service," Ph.D. dissertation, Univ. of California, Berkeley, CA, May 2000, http://research.chawathe.com/people/yatin/publications/docs/thesis-single.ps.gz.

[56] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY: W. H. Freeman and Co., 1979, series of Books in Mathematical Sciences.

[57] H. Zheng, E. K. Lua, M. Pias, and T. G. Griffin, *Internet Routing Policies and Round-Trip-Times*. Berlin/Heidelberg, Germany: Springer, LNCS 3431/2005, March 2005, pp. 236–250.

[58] R. Cox, F. Dabek, F. Kaashoek, J. Li, and R. Morris, "Practical, Distributed Network Coordinates," *ACM SIGCOMM Computer Communications Review*, vol. 34, no. 1, pp. 113–118, January 2004.

[59] B. Wong, A. Slivkins, and E. G. Sirer, "Meridian: a Lightweight Network Location Service Without Virtual Coordinates," in *Proc. of the 2005 SIGCOMM Conf.* Philadelphia, PA: ACM, 2005, pp. 85–96.

[60] T. S. E. Ng and H. Zhang, "Towards Global Network Positioning," in *Proc. of the 1st ACM–SIGCOMM Workshop on Internet Measurement (IMW)*. San Francisco, CA: ACM, November 2001, pp. 25–29.

[61] D. Zhang and C. Lin, "Efficient Delay Aware Peer-to-Peer Overlay Network," in *Advances in Web-Age Information Management*. Berlin/Heidelberg, Germany: Springer, 2005, pp. 682–687, lNCS 3739.

[62] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker, "Making Gnutella-like P2P Systems Scalable," in *Proc. of the 2003 SIGCOM Conf.* Karlsruhe, Germany: ACM, 2003, pp. 407–418.

[63] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, "Search and Replication in Unstructured Peer-to-peer Networks," in *Proc. of the 16th Int. Conf. on Supercomputing (ICS)*, ser. ICS '02. New York, NY: ACM, 2002, pp. 84–95. [Online]. Available: http://doi.acm.org/10.1145/514191.514206

[64] Y. Liu, L. Xiao, X. Liu, L. M. Ni, and X. Zhang, "Location Awareness in Unstructured Peer-to-Peer Systems," *IEEE Trans. on Parallel and Distributed Systems*, vol. 16, no. 2, pp. 163–174, Feb. 2005. [Online]. Available: http://dx.doi.org/10.1109/TPDS.2005.21

[65] Y. J. Pyun and D. S. Reeves, "Constructing a Balanced, (log(N)/loglog(N))-Diameter Super-Peer Topology for Scalable P2P Systems," in *Proc. of the IEEE Int. Conf. on Peer-to-Peer Computing*. Zurich, Zwitzerland: IEEE Computer Society, 2004, pp. 210–218.

[66] H. Cai and J. Wang, "Foreseer: a Novel, Locality-aware peer-to-peer System Architecture for Keyword Searches," in *Proc. of the 5th ACM/IFIP/USENIX Int. Conf. on Middleware*, ser. Middleware '04. Toronto, Canada: Springer-Verlag Inc., 2004, pp. 38–58. [Online]. Available: http://dl.acm.org/citation.cfm?id=1045658.1045663

[67] M. Srivatsa, B. Gedik, and L. Liu, "Large Scaling Unstructured Peer-to-Peer Networks with Heterogeneity-Aware Topology and Routing," *IEEE Trans. on Parallel and Distributed Systems*, vol. 17, no. 11, pp. 1277–1293, Nov. 2006. [Online]. Available: http://dx.doi.org/10.1109/TPDS.2006.158

[68] Z. Zhu, P. Kalnis, and S. Bakiras, "Dcmp: A distributed cycle minimization protocol for peer-to-peer networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 3, pp. 363–377, Mar. 2008.

[69] J. Chandra, S. Shaw, and N. Ganguly, "HPC5: An Efficient Topology Generation Mechanism for Gnutella Networks," *Computer Networks*, vol. 54, no. 9, pp. 1440–1459, Jun. 2010. [Online]. Available: http://dx.doi.org/10.1016/j.comnet.2009.11.017

[70] E. Cohen and S. Shenker, "Replication Strategies in Unstructured Peer-to-Peer Networks," in *Proc. of the ACM SIGCOMM Conference*. Pittsburgh, PA: ACM, August 2002, pp. 177–190.

[71] B. F. Cooper, "Quickly Routing Searches Without Having to Move Content," in *Proc. of the 4th Int. Conf. on Peer-to-Peer Systems*, ser. IPTPS'05. Ithaca, NY: Springer-Verlag, 2005, pp. 163–172. [Online]. Available: http://dx.doi.org/10.1007/11558989_15

[72] V. Ramasubramanian and E. G. Sirer, "Beehive: O(1)lookup performance for power-law query distributions in peer-to-peer overlays," in *Proc. of the 1st Symposium on Networked Systems Design and Implementation - Volume 1*, ser. NSDI'04. San Francisco, CA: USENIX Association, 2004, pp. 8–8. [Online]. Available: http://dl.acm.org/citation.cfm?id=1251175.1251183

[73] Y. Chu, S. Rao, S. Seshan, and H. Zhang, "Enabling Conferencing Applications on the Internet using an Overlay Muilticast Architecture,"

ACM SIGCOMM Computer Communications Review, vol. 31, no. 4, pp. 55–67, 2001.

[74] L. Xing-feng, Y. Bao-ping, and L. Wan-ming, "Overlay multicast network optimization and simulation based on narada protocol," *Advanced Communication Technology, 2008. ICACT 2008. 10th International Conference on*, vol. 3, pp. 2215–2220, Feb. 2008.

[75] Y. Liu, Z. Zhuang, L. Xiao, and L. Ni, "AOTO: Adaptive Overlay Topology Optimization in Unstructured P2P Systems," in *Proc. of the 2003 IEEE GLOBECOM Conf.* San Fransisco, CA: IEEE, December 2003, pp. 4186–4190.

[76] Y. Liu, Z. Zhuang, L. Xiao, and L. M. Ni, "A Distributed Approach to Solving Overlay Mismatching Problem," in *Proc. of the 24th Int. Conf. on Distributed Computing Systems (ICDCS)*. Tokyo, Japan: IEEE Computer Society, 2004, pp. 132–139.

[77] Y. Liu, X. Liu, L. Xiao, L. Ni, and X. Zhang, "Location-Aware Topology Matching in P2P Systems," in *Proc. of the 23rd IEEE INFOCOM Conf.*, vol. 4. Hong Kong, China: IEEE, March 2004, pp. 2220–2230.

[78] Y. Liu, L. Xiao, and L. Ni, "Building a Scalable Bipartite P2P Overlay Network," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 9, pp. 1296–1306, September 2007.

[79] H.-C. Hsiao, H. Liao, and C.-C. Huang, "Resolving the Topology Mismatch Problem in Unstructured Peer-to-Peer Networks," *IEEE Trans. on Parallel and Distributed Systems*, vol. 20, no. 11, pp. 1668–1681, November 2009.

[80] Y. Liu, L. M. Ni, L. Xiao, and A.-H. Esfahanian, "Approaching Optimal Peer-to-Peer Overlays," in *Proc. of the 13th IEEE Int. Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*. Atlanta, GA: IEEE Computer Society, September 2005, pp. 407–414.

[81] Y. Liu, "A Two-Hop Solution to Solving Topology Mismatch," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 11, pp. 1591–1600, November 2008.

[82] H.-C. Hsiao, H. Liao, and P.-S. Yeh, "A Near-Optimal Algorithm Attacking the Topology Mismatch Problem in Unstructured Peer-to-Peer Networks," *IEEE Trans. on Parallel and Distributed Systems*, vol. 21, no. 7, pp. 983–997, 2010.

[83] X. Chen, Z. Li, Y. Zhuang, J. Han, and L. Chen, *HAND: An Overlay Optimization Algorithm in Peer-to-Peer Systems*, ser. Lecture Notes in Computer Science. Berlin, Germany: Springer, September 2006, vol. 4208/2006, pp. 290–299.

[84] D. Bernstein, Z. Feng, B. N. Levine, and S. Zilberstein, "Adaptive Peer Selection," in *Proc. of the 2nd Int. Workshop on Peer-to-Peer Systems (IPTPS '03)*. Berkeley, CA: NSF, February 2003.

[85] W. Zeng, Y. Zhu, H. Lu, and S. Zhuang, "Path-Diversity P2P Overlay Retransmission for Reliable IP-Multicast," *IEEE Trans. on Multimedia*, vol. 11, no. 5, pp. 960–971, 2009.

[86] H. Papadakis, P. Fragopoulou, E. Markatos, and M. Roussopoulos, "ITA: Innocuous Topology Awareness for Unstructured P2P Networks," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 24, no. 8, pp. 1589–1601, Aug 2013.

[87] G. Smaragdakis, V. Lekakis, N. Laoutaris, A. Bestavros, J. W. Byers, and M. Roussopoulos, "The EGOIST Overlay Routing System," in *Proc. of ACM CoNEXT Conference*. Madrid, Spain: ACM, December 2008.

[88] R. Bindal, P. Cao, W. Chan, J. Medved, G. Suwala, T. Bates, and A. Zhang, "Improving Traffic Locality in BitTorrent via Biased Neighbor Selection," in *Proc. of the 26th IEEE Int. Conf. on Distributed Computing Systems (ICDCS)*. Lisbon, Portugal: IEEE, 2006.

[89] S. Ren, E. Tan, T. Luo, S. Chen, L. Guo, and X. Zhang, "TopBT: A Topology-Aware and Infrastructure-Independent BitTorrent Client," in *Proc. of 2010 IEEE INFOCOM*. San Diego, CA: IEEE, March 2010.

[90] D. R. Choffnes and F. E. Bustamante, "Taming the Torrent: a Practical Approach to Reducing Cross-ISP Traffic in Peer-to-Peer Systems," *ACM SIGCOMM Computer Communications Review*, vol. 38, no. 4, pp. 363–374, August 2008.

[91] M. Piatek, H. V. Madhyastha, J. John, A. Krishnamurthy, and T. Anderson, "Pitfalls for ISP-friendly P2P Design," in *Proc. of the 8th ACM Workshop on Hot Topics in Networks (HotNets-VIII)*. New York, NY: ACM, October 2009.

[92] B. Liu, Y. Cui, Y. Lu, and Y. Xue, "Locality-Awareness in BitTorrent-like P2P Applications," *IEEE Transactions on Multimedia*, vol. 11, no. 3, pp. 361–171, April 2009.

[93] R. Cuevas, N. Laoutaris, X. Yang, G. Siganos, and P. Rodriguez, "Deep Diving into BitTorrent Locality," *CoRR*, vol. abs/0907.3874, pp. 1–16, 2009.

[94] W. Li, S. Chen, and T. Yu, "UTAPS: An Underlying Topology-Aware Peer Selection Algorithm in BitTorrent," in *Proc. of the 22nd Int. Conf. on Advanced Information Networking and Applications (AINA)*. Okinawa, Japan: IEEE Computer Society, March 2008, pp. 539–545. [Online]. Available: http://dx.doi.org/10.1109/AINA.2008.92

[95] M. Coates, A. Hero, R. Nowak, and B. Yu, "Internet Tomography," *IEEE Signal Processing Magazine*, vol. 19, no. 3, pp. 47–65, May 2002.

[96] F. Qin, J. Liu, L. Zheng, and L. Ge, "An Effective Network-Aware Peer Selection Algorithm in BitTorrent," in *Proc. of 5th Int. Conf. on Intelligent Information Hiding and Multimedia Signal Processing*. Kyoto, Japan: IEEE Computer Society, September 2009, pp. 418–421. [Online]. Available: http://dx.doi.org/10.1109/IIH-MSP.2009.90

[97] T. Qiu, G. Chen, M. Ye, E. Chan, and B. Y. Zhao, "Towards Location-aware Topology in both Unstructured and Structured P2P Systems," in *Proc. of the 2007 Int. Conf. on Parallel Processing (ICPP)*. Xian, China: IEEE Computer Society, September 2007.

[98] D. Zeinalipour-Yazti and V. Kalogeraki, "Structuring topologically aware overlay networks using domain names," *Comput. Netw.*, vol. 50, no. 16, pp. 3064–3082, 2006.

[99] J. Zhao and J. Lu, "Solving overlay mismatching of unstructured p2p networks using physical locality information," in *P2P '06: Proceedings of the Sixth IEEE International Conference on Peer-to-Peer Computing*. Washington, DC: IEEE Computer Society, 2006, pp. 75–76.

[100] S. Ratnasamy, M. Handley, , R. Karp, and S. Shenker, "Topologically-Aware Overlay Construction and Server Selection," in *Proc. of the 21st IEEE INFOCOM*, vol. 3. New York, NY: IEEE, June 2002, pp. 1190 – 1199.

[101] K. Gummadi, S. Saroiu, and S. D. Gribble, "King: Estimating Latency Between Arbitrary Internet end Hosts," in *Proc. of the 2nd ACM SIGCOMM Workshop on Internet Measurment*, ser. IMW '02. Marseille, France: ACM, 2002, pp. 5–18. [Online]. Available: http://doi.acm.org/10.1145/637201.637203

[102] M. Castro, P. Druschel, Y. C. Hu, and A. Rowstron, "Topology-aware Routing in Structured Peer-to-Peer Overlay Networks," in *Proc. of Future Directions in Distributed Computing (FuDiCo)*. Bertinoro, Italy: Springer-Verlag, 2003.

[103] G. Manku, "Routing Networks for Distributed Hash Tables," in *Proc. of the 22nd ACM Symposium on Principles of Distributed Computing*, ser. PODC '03. Boston, MA: ACM, 2003, pp. 133–142. [Online]. Available: http://doi.acm.org/10.1145/872035.872054

[104] X. Y. Zhang, Q. Zhang, Z. Zhang, G. Song, and W. Zhu, "A construction of locality-aware overlay network: moverlay and its performance," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 1, pp. 18–28, Jan. 2004.

[105] S. Ren, L. Guo, S. Jiang, and X. Zhang, "SAT-Match: A Self-Adaptive Topology Matching Method to Achieve Low Lookup Latency in Structured P2P Overlay Networks," in *Proc. of IEEE Parallel and Distributed Processing Symposium (IPDPS)*, vol. 1. Santa Fe, NM: IEEE Computer Society, April 2004, pp. 83–92.

[106] Z. Xu, C. Tang, and Z. Zhang, "Building topology-aware overlays using global soft-state," in *ICDCS '03: Proceedings of the 23rd International Conference on Distributed Computing Systems*. Washington, DC: IEEE Computer Society, 2003, p. 500.

[107] Gnutella, "The Gnutella Protocol Specification v0.4," http://www9.limewire.com/developer/gnutella_protocol0.4.pdf, 2000.

[108] A. Crespo and H. Garcia-Molina, "Routing Indices For Peer-to-Peer Systems," in *Proc. of the 22nd Int. Conf. on Distributed Computing Systems (ICDCS'02)*, ser. ICDCS '02. Vianna, Austria: IEEE Computer Society, 2002, pp. 23–33. [Online]. Available: http://dl.acm.org/citation.cfm?id=850928.851858

[109] M. Castro, P. Druschel, Y. C. Hu, and A. Rowstron, "Exploiting Network Proximity in Distributed Hash Tables," in *Proc. of Int. Workshop on Future Directions in Distributed Computing (FuDiCo)*. Bertinoro, Italy: Springer-Verlag, June 2002, pp. 52–55.

[110] S. Ratnasamy, I. Stoica, and S. Shenker, "Routing Algorithms for DHTs: Some Open Questions," in *Proc. of 1st Int. Workshop on Peer-to-Peer Systems (IPTPS)*. Cambridge, MA: Springer-Verlag, 2002, pp. 45–52, lNCS-2429.

[111] Z. Xu and Z. Zhang, "Building low-maintenance expressways for p2p systems," HP Laboratories, Internet Systems and Storage Laboratory, Tech. Rep. HP Laboratories Palo Alto, Mar. 2002.

[112] R. Winter, T. Zahn, and J. Schiller, "DynaMO: A Topology-Aware P2P Overlay Network for Dynamic, Mobile Ad-Hoc Environments," *Telecommunication Systems*, vol. 27, no. 2-4, pp. 321–345, October 2004.

[113] M. Waldvogel and R. Rinaldi, "Efficient topology-aware overlay network," *ACM SIGCOMM Computer Communications Review*, vol. 33, no. 1, pp. 101–106, January 2002, hot Topics in Networks (HotNets-I).
[114] B. Karp and H. T. Kung, "GPSR: greedy perimeter stateless routing for wireless networks," in *Proceedings of the 6th annual international conference on Mobile computing and networking*, ser. MobiCom '00. New York, NY, USA: ACM, 2000, pp. 243–254. [Online]. Available: http://dx.doi.org/10.1145/345910.345953
[115] R. Rinaldi, "Routing and data location in overlay peer-to-peer networks," 2002, institut Eurécom and Università degli Studi di Milano. Diploma thesis. Also available as IBM Research Report RZ-3433.
[116] M. Costa, M. Castro, A. Rowstron, and P. Key, "PIC: Practical Internet Coordinates for Distance Estimation," in *Proc. of 24th Int. Conf. on Distributed Computing Systems (ICDCS'04)*, ser. ICDCS '04. Washington, DC: IEEE Computer Society, 2004, pp. 178–187. [Online]. Available: http://dl.acm.org/citation.cfm?id=977400.977998
[117] C.-J. Wu, D.-K. Liu, and R.-H. Hwang, "A location-aware peer-to-peer overlay network," *Int. J. Commun. Syst.*, vol. 20, no. 1, pp. 83–102, 2007.
[118] S. Kaune, T. Lauinger, A. Kovacevic, and K. Pussep, "Embracing the Peer Next Door: Proximity in Kademlia," in *Proc. of the 8th Int. Conf. on Peer-to-Peer Computing (P2PC)*. Aachen, Germany: IEEE, September 2008, pp. 343–350.
[119] S. Morimoto and F. Teraoka, "Chop6: A dht routing mechanism considering proximity," in *Proc. of the 2007 Int. Symposium on Applications and the Internet Workshops (SAINT-W)*. Hiroshima, Japan: IEEE Computer Society, January 2007, p. 59.
[120] G. Shi, Y. Long, J. Chen, H. Gong, and H. Zhang, "T2MC: A Peer-to-Peer Mismatch Reduction Technique by Traceroute and 2-Means Classification Algorithm," in *Proc. of the 2008 Int. Conf. on NETWORKING: Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet*. Singapore: IFIP Springer, LNCS 4982, May 2008, pp. 366–374.
[121] F. Hong, M. Li, J. Yu, and Y. Wang, "PChord: Improvement on Chord to Achieve Better Routing Efficiency by Exploiting Proximity," in *Proc. of the 1st Int. Workshop on Mobility in Peer-to-Peer Systems (MPPS)*. Columbus, OH: IEEE Computer Society, June 2005, pp. 806–811.
[122] L. H. Dao and J. Kim, "AChord: Topology-Aware Chord in Anycast-Enabled Networks," in *Proc. of Int. Conf. on Hybrid Information Technology (ICHIT)*. Jeju Island, Korea: IEEE Computer Society, November 2006, pp. 334–341.
[123] C. Metz, "IP Anycast: Point-to-(Any) Point Communication," *IEEE Internet Computing*, vol. 6, no. 2, pp. 94–98, Mar. 2002.
[124] J. Xiong, Y. Zhang, P. Hong, and J. Li, "Chord6: IPv6 Based Topology-Aware Chord," in *Proc. the Joint Int. Conf. on Autonomic and Autonomous Systems and Int. Conf. on Networking and Services (ICAS/ICNS)*. Papete, Tahiti: IEEE Computer Society, October 2005, p. 4.
[125] H. Duan, X. Lu, H. Tang, X. Zhou, and Z. Zhao, "Proximity Neighbor Selection in Structured P2P Network," in *Proc. of the 6th IEEE Int. Conf. on Computer and Information Technology (CIT)*. Seoul, Korea: IEEE Computer Society, September 2006, p. 52.
[126] M. Sun and Z. Zhang, "Quasi-Chord: Physical Topology Aware Structured P2P Network," in *Proc. of the 11th Joint Conf. on Information Sciences (JCIS-2008)*, ser. Advances in Intelligent Systems Research. Shenzhen, China: Atlantis Press, December 2008.
[127] P. Karwaczynski and J. Mocnik, "IP-based Clustering for Peer-to-Peer Overlays," *Journal of Software*, vol. 2, no. 2, pp. 30–37, 2007.
[128] H. Eom, D. Wolinsky, and R. J. Figueiredo, "SOLARE: Self-Organizing Latency-Aware Resource Ensemble," in *Proc. of the 13th IEEE Int. Conf. on High Performance Computing and Communications (HPCC)*. Banff, Canada: IEEE, September 2011, pp. 229–236.
[129] M. Castro, P. Druschel, Y. Charlie, and H. A. Rowstron, "Exploiting Network Proximity in Peer-to-Peer Overlay Networks," Microsoft Research, Tech. Rep., 2002, mSR-TR-2002-82.
[130] S. Abid, M. Othman, and N. Shah, "3D P2P overlay over MANETs," *Computer Networks*, vol. 64, no. 0, p. 89111, May 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128614000486
[131] S. Jiang, L. Guo, X. Zhang, and H. Wang, "LightFlood: Minimizing Redundant Messages and Maximizing Scope of Peer-to-Peer Search," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 5, pp. 601–614, 2008.
[132] W. Huijin and L. Yongting, "Cone: A Topology-Aware Structured P2P System with Proximity Neighbor Selection," in *Proc. of Future Generation Communication and Networking (FGCN)*. Jeju-Island, Korea: IEEE Computer Society, 2007, pp. 43–49.
[133] M. J. Freedman, M. Vutukuru, N. Feamster, and H. Balakrishnan, "Geographic Locality of IP Prefixes," in *Proc. of the 5th SIGCOMM Conf. on Internet Measurement (IMC)*. Berkeley, CA: USENIX, 2005, pp. 13–13.

Vassilis Moustakas is a researcher with the University of Athens in Athens, Greece.

Hüseyin Akcan is an Associate Professor of Computer Science at Izmir University of Economics in Izmir, Turkey.

Mema Roussopoulos is an Associate Professor of Computer Science at the Univ. of Athens in Athens, Greece.

Alex Delis is a Professor of Computer Science at the Univ. of Athens in Athens, Greece.