FISHEYE: Topology Aware Choice of Peers for Overlay Networks

Dragan Milić, Torsten Braun
Institute of Informatics and Applied Mathematics
University of Bern, Neubrückstrasse 10, 3012 Bern, Switzerland
email: {milic|braun}@iam.unibe.ch phone: +41 31 511 {2633|2631}

***Abstract*—Constructing a topology aware overlay network is an open research topic. In this paper we propose a novel protocol for building overlay networks - a distributed fisheye view. Similar to round trip time (RTT) prediction approaches, we consider the end systems to be embedded in a virtual metric space. Unlike other approaches, we use only the distances (measured RTTs) to build an RTT proximity aware overlay network. Therefore, we are able to construct a fisheye view without performing the embedding. At the same time we are still able to guarantee the geographical diversity of the neighbors. Once built, the fisheye views on the end systems are continuously refined as information about new potential neighbors is available. This makes our overlay network adaptive to changes in the network topology. To evaluate our approach, we compared it with an existing topology aware overlay network construction approach – binning. We based this comparison on RTT measurements obtained using the King RTT measurement method and from the Planet Lab "all site ping" experiment. Our evaluation shows that the overlay network built using our approach outperforms binning in terms of relative RTT stretch. We also show that for increasing number of neighbors the performance of our approach converges towards an optimal solution.**

I. Introduction

Peer-to-peer (p2p) and overlay networks are widely used in the Internet today. Such networks are usually used, when a service is required that is missing in the underlying network, such as multicast or a distributed hash table (DHT). The advantage of using p2p and overlay networks in such a scenario is the ease of the deployment. In contrast to deploying a new service in the underlying network, which implies changing the infrastructure and/or deploying servers, a p2p network is deployed only on end systems interested in using the service. This means that there is no need for the cooperation of the network provider or some third party.

As of today, there are numerous p2p and overlay protocols. Most of them are designed to provide services such as service discovery [10], [26], DHT [6], [12], [17], [19], [22], application layer multicast (ALM) [2], [3], [20]. All of those protocols are designed to be scalable and efficient within the overlay network. For example, the number of hops for a lookup in Chord [22] is guaranteed to grow logarithmically with the size of the network. On the other hand, the same protocol totally disregards the topology of the underlying network. This means that routing from end system A to end system B, which are in the same autonomous system (AS), may be performed via an end system C, which is in another AS. As a result, one query may unnecessarily traverse "long" (in terms of communication delay) links in the underlying network several times before it reaches its destination. This unnecessarily increases the communication delay.

Basically there are two approaches for extracting topology information for the underlying network. The most straightforward method is to consider the underlying network as a white box. This means that the topology information (topology graph) of the underlying network is extracted and used to construct an optimal overlay network. Topology information gathering is usually done either by querying databases provided by the Internet service providers and BGP routing tables or using topology discovery tools [1], [11]. This approach may yield a good result for small overlay networks, because it allows quite precise optimization on the link level. On the other hand, this approach introduces too much overhead and complexity for large overlay networks. The amount of information needed to optimize the overlay structure of a large overlay network is quite huge. In addition, finding the optimal solution for the problem is more or less equivalent to the traveling salesman problem, which is NP-hard. Another issue is that ISPs are rarely willing to provide the information about the internal structure of their network.

In contrast to the white box approach, in the black box approach, the structure of the underlying network is not analyzed. Instead, targeted QoS parameters such as available bandwidth, RTT, jitter etc. are actively or passively measured between end systems. Based on this information the structure of the overlay network can be optimized. Still, just measuring QoS parameters for each pair of end systems scales quadratically $O(n^2)$ relative to the number of end systems. Also, the optimization task is still NP-hard. This makes the optimization of large overlay networks unscalable. To solve this problem, we should not aim for finding the optimal solution. Instead we need a method that provides a solution close to the optimal one, but with much lower cost in terms of number of measurements required and computational overhead.

In order to obtain a usable method to build a topology aware overlay network we need to reduce the number of required measurements and the overhead of overlay optimization. To have scalable measurements, we could reduce the number of measurements performed at the cost of accuracy, by eliminating the need to perform measurements for all end system pairs. Another effective method is to have a system that does not need the RTT information for every pair of end systems immediately. Instead, the system can perform a constant number of measurements per time unit and use this newly obtained information to continuously converge towards a better solution.

Numerous position based mobile ad-hoc network routing protocols [16], [24] are based on a “fisheye” view of the network. The “fisheye” view of the network implies that each end system has a view of its neighborhood similar to how a fisheye sees. The lens of a fisheye has an extreme wide angle. The light is refracted by the fisheye lens such that the center of the view contains very high level of detail. With increasing distance from the center of the view, the detail level rapidly decreases.

In this paper, we propose a black box based approach to build an overlay network. We propose periodically measuring RTTs and using that information to build a fisheye view of the overlay network at each end system. This fisheye view is constantly refined when new information is available. By doing so we ensure constant convergence towards a better overall solution for the overlay network.

This paper is structured as follows. In the next Section, we give an overview of the related work relevant for this paper. In Section III we present how the fisheye view paradigm can be applied to computer networks. In the same Section we show how a fisheye view can be incrementally built for overlay networks even without using an embedding of end systems in a virtual space. Section IV identifies the problems that occur, if we use a naive approach of independently building a fisheye view at each end system. Our solutions to the identified problems are presented in the same Section. Section V presents a distributed communication protocol used to incrementally build and refine the fisheye view of the overlay network on each end system. Section VI compares our approach with other existing topology aware approaches. In the last section we summarize our approach, the experimental results and outline the possible use of such an overlay network.

II. Related Work

When we apply the paradigm of a fisheye view to a position based computer network, each end system should have a fisheye view of the overlay network. This fisheye view is populated with many near neighbors. The density of known neighbors decreases with increasing distance. Theoretically, the density should never reach zero. In the case of an overlay network this is not possible, since both the number of end systems and the diameter of the space populated by the end systems are finite.

The advantages of the fisheye view at each end system of the overlay network are twofold. The fisheye view decreases the amount of data that one end system must store about the network. Since memory at each system is limited, the amount of information stored should be kept constant regardless of the overlay network size. A fisheye view also contains a rather high density of “close” neighbors allowing efficient routing using short distances. At the same time, it contains minimal information about remote neighbors, making the long routing “jumps” possible.

A. Use of Fisheye View in Overlay Networks

The fisheye view paradigm is not only limited to network routing protocols. Many of the existing DHT approaches are based on a fisheye routing table to achieve a balance between the size of the routing table and the number of routing hops. For example, Pastry [19] and other Plaxton-based routing protocols [17] use prefix based routing table, which is nothing else than a fisheye view of the ID space.

B. Topology Aware Overlay Networks

Most of the proposed topology aware overlay networks are used to provide application level multicast (ALM) services.

One of the first proposed topology aware ALM is NARADA [3]. NARADA is designed for relatively small groups of end systems. The approach of NARADA is to build a connected graph of overlay nodes, which the authors termed “mesh”. The mesh is constantly refined according a utility function, which regards the utility gain (reduction of RTT) for the other neighbors. Building multicast trees in NARADA is done using the DVMRP [5] protocol.

Another interesting topology aware overlay protocol is Pastry [19]. Pastry is a general purpose overlay network routing protocol. Each Pastry node has an unique (usually randomly chosen) ID. Routing in Pastry is done using prefix routing table using Plaxton like routing [17]. This is, like many other overlay/p2p routing protocols, a fisheye view of the address space. The feature that makes Pastry different compared to other prefix based routing schemes is the way the routing table is built. If more than one candidate exists for an entry in the prefix table, Pastry chooses the one with smallest RTT. In such a way, Pastry optimizes not only the routing in the overlay, but also in the underlying network.

Meridian [23] is an overlay network for finding near services regarding RTT. Similar to the approach we are proposing here, Meridian keeps a fisheye view of its neighbors. This fisheye of each end system consists of concentric rings with increasing radius. Within each of the rings a constant number of neighbors are kept (k). The geographic diversity of the chosen neighbors within one ring is ensured by maximizing a k-polytope formed by the neighbors. Since Meridian has no guarantee to form a connected graph, it is not suitable for the purpose of unicast or multicast routing. Instead, the purpose of Meridian is to provide an efficient neighbor finding service in terms of communication delay. The number of hops of a query is guaranteed to scale logarithmically $O(\log n)$ relative to the number of end systems participating in the Meridian network.

In our approach we combine the ideas of having a bidirectionally connected graph that is continuously improved as proposed by NARADA with a fisheye view at each end system as proposed by Meridian. We improve Meridian’s method for choosing neighbors by minimizing the forces of gravity instead of having concentric rings and maximizing k-polytopes.

III. Fisheye View

Almost all existing fisheye view approaches [12], [16], [19], [22], [24] are based on an assumption that each end system has a position in a one- or a two-dimensional Euclidean space. When we consider a generic computer network such as the Internet, we are dealing with a connected graph. There is no guarantee that this graph can be embedded into a metric space. Even if there is such an embedding, there is no guarantee that the number of the dimensions used for the embedding is low. To overcome this problem, we could use an approach that does not necessarily embed the distances, but tries to embed distance approximations in order to reduce the overall error. For example, we could use some kind of position based RTT prediction scheme such as GNP [13], [14] or VIVALDI [4] to obtain coordinates of the end systems. The problem of such an approach is that they are per design not accurate – mostly due to the fact that RTTs in the Internet frequently violate one of the basic properties of metric distances: the triangle inequality. Nevertheless, for the clarity of our explanations, we assume that there is an accurate embedding of end systems into a d-dimensional Euclidean space, in such a way that the distance between any end system pair in the metric space is exactly the measured RTT between those end systems. Later, we will show that there is actually no need to perform the embedding, since we can operate only with measured distances, which significantly simplifies our approach.

To explain our idea for building a fisheye overlay network we assume the following. Our end systems in the overlay network are defined as a set of k nodes $\mathcal{N} := \{n_i | i \in \{0 \ldots k\}\}$. Each of the nodes has a position denoted by $\mathcal{C}_{n_i}$ in a p-dimensional Euclidean space denoted $\mathcal{S}_p$. The positions of the nodes in $\mathcal{S}_p$ are not random – they are chosen in such a way that for every pair of nodes $n_l, n_m \in \mathcal{N}$ the Euclidean distance, defined in (1)

$$\hat{d}(\mathcal{C}_{n_l}, \mathcal{C}_{n_m}) := \sqrt{\sum_{i=1}^{p} (\mathcal{C}_{n_l}^i - \mathcal{C}_{n_m}^i)^2} \tag{1}$$

is equal to the RTT between those two nodes measured through the underlying network $d(n_l, n_m)$. Each node is allowed to store information of at most c neighbors at the same time. The problem we are trying to solve is to assign a fisheye view choice of neighbors to each end system in a distributed manner. This choice should fulfill the properties of the fisheye view, meaning that most of the neighbors should be from the direct vicinity in terms of network distance (RTT). At the same time, the fisheye view should also contain a few "long" links to neighbors that are more distant. Those long links enable efficient routing towards end systems that are far away. Another important property of a fisheye view is the geographical diversity. Geographical diversity means that if we embed the overlay network into a metric space, the chosen neighbors would "surround" the end system. The geographical diversity ensures efficient routing in each "direction" through the overlay network.

A. *Universe Analogy*

In order to explain our approach we use the universe analogy. We consider an overlay network to be a universe. Each end system is one planet located somewhere within this universe. Distances between the planets are RTTs measured between the corresponding end systems. Since the universe has no reference point, each planet has its own "view" of the universe. Each planet's view considers this planet to be in the center of the view. The problem we are trying to solve is to reduce the number of planets within the view to a constant number c. This reduction of the planet's view is equal to choosing the overlay neighbors for one end system. The two most distinguishing properties of the fisheye view are geographical diversity and a degree of detail that decreases with increasing distance to the center of the visual field. Therefore, the reduction we perform should consider those properties. As in the real universe, there is the gravity force in our universe. This gravity force is defined by (2), where r is a distance between two planets and m_1 and m_2 are masses of those two planets.

$$F = G \cdot \frac{m_1 \cdot m_2}{r^2} \tag{2}$$

B. *Geographical Diversity*

Let us assume, we would only try to achieve geographical diversity. As mentioned before, the planet for which we tailor its view is in the center of the view. We denote this planet (end system) with n_c. The other planets surrounding it are not necessarily uniformly distributed through the universe. There may be some clusters of planets, which are located close to each other. Also, there may be some portions of the universe containing no planets at all. This means that the density of planets is not evenly distributed through the universe. The goal of our reduction of planets in the view is to remove the planets, which are close to each other, i.e., to avoid clusters. We could achieve this by iteratively removing planets, one by one. At each iteration we would have to pick one planet that is in the dense part of the universe. This procedure can be continued until we have exactly c planets left. The planets that are still in the view are evenly distributed through the universe and therefore the obtained view is geographically diverse.

The only question that remains is: Which planet should be removed at the iteration? As an answer to this question, we propose removing the planet with the maximum sum of gravity forces, because those planets are in the most dense parts of the universe. We assume that each planet in the universe has a constant mass (for simplicity we assume $m = 1$). For each planet pair n_i, n_j within the universe we could calculate the gravity force using (2). This gravity force increases (more than linearly) with decreasing distance between planets. For that reason, the total sum of gravity forces for planets that are in a cluster of planets is higher than for planets that are located in less dense parts of the universe. Therefore, if we always remove the planet with the maximum sum of gravity forces from the view, we will equalize the density of planets. The equation for computing the gravity force would be (3).

$$F_{n_i^c} = \sum_{j \in \mathcal{N} \setminus \{i,c\}} \frac{1}{(d(n_i, n_j))^2} \tag{3}$$

C. Enforcing Fisheye View

Up to now we assumed that the mass of a planet is equal for every planet. As a result of this, the reduced view of planets is (as far as possible) equally distributed. This equal distribution of the view unfortunately does not have the fisheye property. In order to have the fisheye property, we should favor planets close to the center of the universe (the planet for which we calculate the view). To do so, we propose considering the mass of planets not to be constant, but proportional to the distance to the center of the universe n_c. By doing so, we shift the density of the planets remaining in the view towards the center of the universe to obtain a fisheye view. The final equation for computing the total gravity force for one planet n_i for the center n_c is (4).

$$F_{n_i^c} = \sum_{j \in \mathcal{N} \setminus \{i,c\}} \frac{d(n_i, n_c) \cdot d(n_j, n_c)}{(d(n_i, n_j))^2} \tag{4}$$

As an illustration, Fig. 1 shows the result of our algorithm applied to a “universe” of 300 end systems. The universe is reduced to 25 neighbors. Of course, in a p2p network, one peer never stores the information about the whole overlay network at the same time. Instead, one peer becomes aware of only few new peers at a time. Fortunately, our algorithm is by design incremental and can be applied each time a new peer is known, to determine whether the new peer should replace an existing one in the view. For example, if one peer already has its fisheye view of the network and becomes aware of a new peer, it only needs to add this peer to its “view”, which now contains the old fisheye view and additionally the new peer, and to re-calculate the sum of the gravity forces in it. If the new “planet” has a lower sum of gravity forces than the other planets, then the planet with the largest sum of gravity forces is removed from the fisheye and the new neighbor becomes a new planet in the “view”. By this an end system incrementally refines its fisheye view through its lifetime.

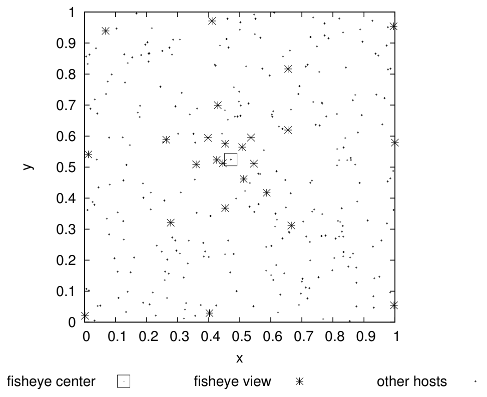

Fig. 1. Example of fisheye view calculated using our gravity force minimization algorithm.

D. Are Positions Required?

If we examine (4), we see that the total gravity force depend only on distances. Since in our case the distances are measured RTTs, we could use our algorithm without embedding the end systems into a virtual space. Instead of embedding, we can directly use the measured distances (RTTs) to construct the fisheye view for each end system. Since our approach does not require the knowledge of the host position, it avoids the error introduced by performing the embedding. This means that our approach should still be effective, even if not all distances fulfill the triangle inequality property. This has been substantiated by experiments performed in [4], [13], [14], [21] that show that those embeddings provide a quite precise RTT prediction service. This means that assuming that the Internet can be embedded in a metric space is not completely false.

IV. Problems of Independent Construction of the Fisheye View

If we simply apply the method we described in Section III, each end system in the overlay network would be able to construct its own fisheye view of the network. At the first glance, this would be a good idea, since there is no requirement for cooperation between the end systems, besides providing RTT measurements. But there are also two major drawbacks to this approach: asymmetry of the connection graph and non uniform connection distribution.

A. Asymmetry of the Connection Graph

We can observe an overlay network as a connected graph. If each end system independently decides which other end systems are known to him, we obtain a directed graph. Such a graph has no guarantee that it will be connected. For example, if we have a portion of the virtual space with high end system density, the chances are growing that at least one of those end systems will not be “chosen” by some other end system to be its neighbor. As a consequence, if one of the end systems is never chosen as a neighbor, there is no edge in the graph representing the overlay network that leads to this end system. This means that the graph is not necessarily connected.
To overcome this problem we propose that the “is a neighbor of” relation of a fisheye view is bidirectional. This means n_1 is a neighbor of n_2 iff n_2 is a neighbor of n_1. If the “is a neighbor of” property holds, the graph is bidirectional and must be connected. This is due to the fact how the overlay network is built. We assume that the existing overlay network is a connected graph. If one node joins the network, the new node must connect to at least one of the end systems of the existing network. If this connection is bi-directional, the graph of the extended overlay network is also connected.

B. Non-Uniform Connection Distribution

Another problem that can occur, if the end systems are not uniformly distributed within the virtual space, is the non-uniform distribution of connections in the overlay network. The problem occur if one end system is isolated and alone in a large portion of the virtual space. Since this end system

is the only end system in a quite large portion of the space, the total sum of the gravitation forces for that end system is probably quite low. As a consequence, this end system will be in the fisheye view of many other end systems. Thus this end system may be forced, depending on what the overlay network is used for, to handle a lot of traffic.
To solve this problem, we split the fisheye view into incoming and outgoing connections. Each end system can have at most $\mathcal{C}^{\text{inc}}$ incoming and $\mathcal{C}^{\text{out}}$ outgoing connections. An outgoing connection is a connection initiated by the local system. Incoming connection means that the connection is initiated by some other end system. Each end system is free to use its outgoing connections to improve its fisheye view by disconnecting from one and connecting to another remote system, if that improves its fisheye. However, an end system is not allowed to disconnect incoming connections. We also add a multiplication factor for the total sum of gravity forces (4) to stimulate connecting to the end systems with lower number of incoming connections: $\frac{1}{\mathcal{C}^{\text{inc}}-\mathcal{E}^{\text{inc}}}$, where $\mathcal{E}^{\text{inc}}$ is the current number of open incoming connections. Thus, the improved function to compute the sum of gravity force in the universe is (5).

$$F_{n_i^c} = \sum_{j\in\mathcal{N}\setminus\{i,c\}} \frac{1}{\mathcal{C}_j^{\text{inc}} - \mathcal{E}_j^{\text{inc}}} \cdot \frac{d(n_i,n_c)\cdot d(n_j,n_c)}{(d(n_i,n_j))^2} \tag{5}$$

V. Distributed Fisheye View

In this Section we propose a distributed algorithm and a communication protocol for building a fisheye overlay network. Our protocol is designed to incrementally converge towards the optimal solution by using only partial measurement information and knowledge of the overlay network.
The goal of the algorithm and the protocol is to build and maintain a fisheye view at each end system. The constructed overlay network represents a bi-directionally connected graph, where each end system has knowledge of only a constant number of other end systems. The protocol is constructed in a way such that it is robust against end node failures. Also it is reactive to topology changes of the underlying network.

A. Gossiping and Keep Alive Messages

Each end system participating in the overlay network sends periodically *UPDATE* messages to all of its neighbors. To avoid synchronous "heartbeats" of the network, the retransmission time is randomly (with uniform distribution) chosen from the interval $t_{\text{xmit}} \in \left[\frac{t_{\text{update}}}{2}, t_{\text{update}}\right]$. The *UPDATE* message contains the current fisheye view of the end system and a gossiping list. One purpose of the *UPDATE* message is to confirm the aliveness of an end system to all of its neighbors (keep alive message). At the same time, the *UPDATE* message performs a random gossiping of new end systems that can be used to optimize fisheye views of the neighbors.

B. Opening and Closing Connections

Each end system may at any time open a new outgoing connection to any other end system in the overlay network. Usually one end system contains a list of end systems, which should be used to optimize the overlay network. This list is maintained by combining information from *UPDATE* messages received from the neighbors. Opening the connection is initiated by sending an *OPEN* message to the remote end system. Upon receiving an *OPEN* message, the end system may either reply with an *ACCEPT* or with a *REJECT* message. The response of the end system depends on the number of already established incoming connections. If the number of already established incoming connections reached its maximum, then the response will be *REJECT*. Otherwise the response is *ACCEPT*. With an *OPEN* message, the list of all current neighbors is sent. If the remote system replies with an *ACCEPT* message, it also must perform RTT measurements to all end systems from the list contained in the *OPEN* message and send them back as a part of the *ACCEPT* message. This information is used at the connection initiating end system to calculate a new fisheye view. If the new fisheye view represents a better fisheye view, in terms of the force of gravity minimization described in Section III, then the initiating end system sends a *COMMIT* message to the remote system. Otherwise, a *ROLLBACK* is sent to the remote system, to indicate that the connection has not been established.
The closing of an established outgoing connection is indicated by sending a *CLOSE* message to the other end system. Also, an end system gracefully leaving the overlay network, should send a *CLOSE* message to all of its neighbors, regardless whether the connection is incoming or outgoing.

C. Bootstrapping

To join a fisheye overlay network, an end system must "know" at least one node already joined to the overlay network. To find this node, one of the usual p2p bootstrapping methods can be used, such as having a well known peer, dynamic DNS, etc. [7]. The new node contacts its bootstrapping node by sending a *HELLO* message to it. As a response to a *HELLO* message the bootstrapping node sends the *UPDATE* message, which contains its current neighbors and a gossiping list. The gossiping list contains a random choice of known systems, which are not the direct neighbors. Each end system maintains a list of such potential neighbors. The reason for this is to increase the random choice of neighbors to connect to. Since this is the same message used for maintaining and gossiping in the overlay network, the joining node can use the information contained in it to choose its neighbors for its initial fisheye view and establish a connection as described in Section V-B.

VI. Evaluation

In order to verify the performance of our approach, we have compared the overlay network constructed using our approach with binning, an optimal heuristic and a random overlay construction approach [18]. Since we are focusing on the properties of the constructed overlay network, we avoid using any particular overlay routing protocol. Instead, we always calculate an optimal path in the overlay network (path with the minimal RTT stretch) and compare the lengths of those paths.

A. *Protocol Simulation*

We evaluated our overlay construction approach using an event based simulation [15]. In order to have a realistic network model, we used RTT information measured in the Internet. In our network model, each overlay message was delayed for half of the RTT measured between end systems in the Internet. The RTT distances are obtained by performing RTT measurements for each pair of end systems in the Internet. In our case, we have used one RTT matrix obtained from the Planet Lab "all sites ping" experiment [25] and one obtained using the King [8] method. Used data sets contain a complete RTT distance matrix for 218 Planet Lab end systems and 462 end systems measured with the King method. The nodes in our simulation are started with 0.1 second delay, meaning a high churn at the first 21.8 seconds of simulation in the case of Planet Lab data and 46.2 seconds for the King data set.

B. *Comparison Methodology*

The goal of our evaluation is to determine, whether the overlay network constructed using our fisheye view (*FISHEYE*) approach is better or worse than existing approaches. We compare our approach with the following other approaches:

- *RND:* The random approach is the most common one used in unstructured overlay networks. Each end system randomly chooses S other end systems already participating in the overlay network. The performance of this approach is an upper boundary for a topology aware overlay network, since it totally disregards the topology of the underlying network.
- *OPT-HEUR:* A heuristic method that should approximate the optimal choice of neighbors. This approach is based on choosing $S/2$ nearest neighbors of an end system and $S/2$ of randomly chosen neighbors from the overlay network. According to [18], this approach delivers results that approximate an optimal solution. We used this heuristic method as a lower boundary for the performance of an overlay building strategy, since finding an optimal overlay network is NP-hard. The main drawback of this method is that each end system would need to have a complete knowledge of the overlay network in order to determine the $c/2$ nearest neighbors.
- *BINN:* In a topology aware approach [18]. Each end system is assigned into one of the $L!$ bins relative to L randomly chosen landmarks. A bin is determined by measuring RTT to a set of landmarks and ordering those landmarks according to the relative distance to an end system. End systems in the same bin are considered to be roughly in the same area of the network. Hence, similar to *OPT-HEUR*, this strategy assigns $S/2$ neighbors from the same (or the most similar) bin and $S/2$ neighbors chosen randomly from the overlay network. For our experiments, we have set the number of landmarks $L = 5$, which gives the maximal number of 120 different bins

To compare the efficiency of the paths in the overlay network, we compare the relative length of the path through the overlay network in terms of RTT between each pair of end systems in the overlay network. We define the relative path between two overlay systems as:

$$\frac{\text{RTT of the optimal path through the overlay network}}{\text{RTT of the optimal path through the underlying network}}$$

The relative path length of 1.0 means that the communication delay through the overlay network is optimal (as good as the best route through the underlying network). A relative path length value larger than 1.0 indicates the "penalty" (additional overhead) for routing using the overlay network. Values smaller than 1.0 are not possible.

C. *Results*

Since the performance of all approaches varies depending on the initial random seed, we performed 10 runs of each experiment with different seeds. Figures 2 and 3 show the median of relative paths for the overlay networks using the Planet Lab and King data sets for all four approaches depending on the number of neighbors (S) that we varied between 9 and 21. As the Figures show, the median of all relative paths

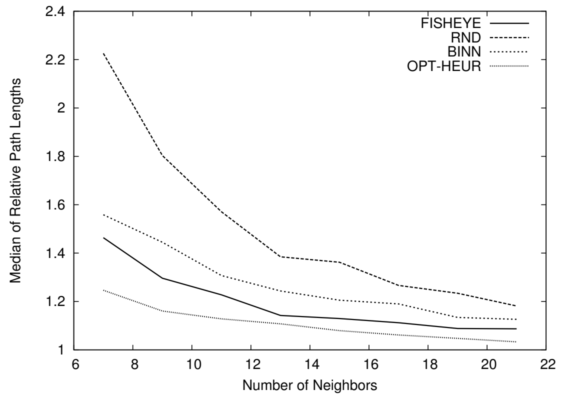

Fig. 2. Median of relative path length in the overlay network in dependence of number of neighbors for the Planet Lab data set.

in our approach outperforms both RND and BINN overlay construction strategies and approaches the performance of *OPT-HEUR* for increasing S.

In order to keep Figures 2 and 3 readable, we omitted confidence intervals. To have a better overview of the performance, we also show the cumulative distribution function (CDF) of optimal paths for 9 and 21 neighbors in Figures 4 and 6 for Planet Lab and in Figures 5 and 7 for the King data set.

The CDF data for 7 neighbors show that *FISHEYE* and *BINN* do not significantly differ. With increasing number of neighbors, performance of *FISHEYE* improves and for 21 neighbors outperforms the *HEUR-OPT* strategy in the King data set. Both median and CDF comparisons show that the FISHEYE approach clearly outperforms both RND and BINN overlay

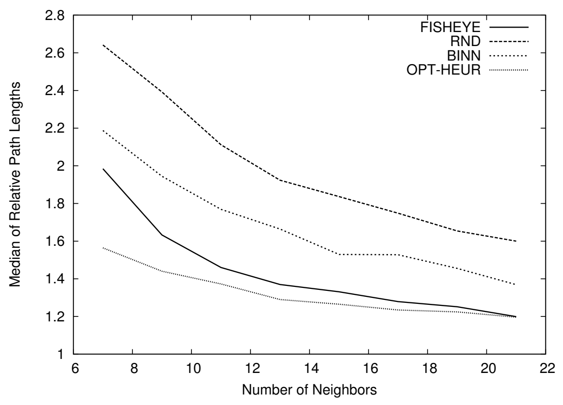

Fig. 3. Median of relative path length in the overlay network in dependence of number of neighbors for the King data set.

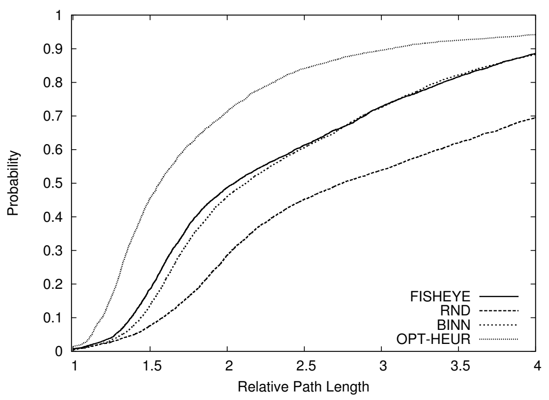

Fig. 5. CDF of relative path lengths in the overlay network with 7 neighbors for the King data set.

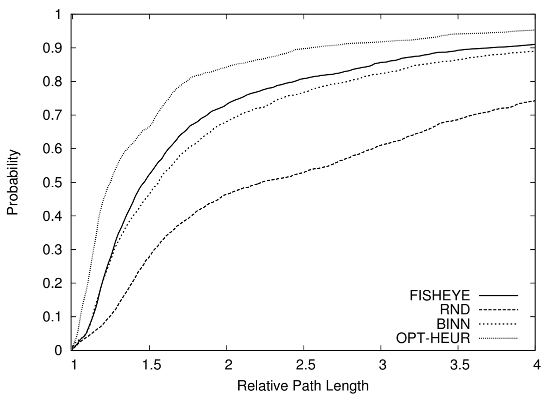

Fig. 4. CDF of relative path lengths in the overlay network with 7 neighbors for the Planet Lab data set.

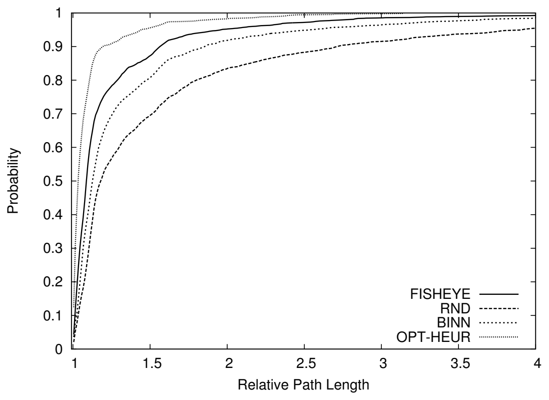

Fig. 6. CDF of relative path lengths in the overlay network with 21 neighbors for the Planet Lab data set.

construction strategies for a number of neighbors larger than 7. Another interesting result is that with the increasing number of neighbors, the *FISHEYE* approach converges towards the performance of the OPT-HEUR approach.

D. Convergence

We also examined the speed of convergence of our approach at high rates of joins and leaves. For this, we simulated building a *FISHEYE* overlay network with $S = 19$ using the King data set for 500 seconds. Every 10 seconds of the simulation, we calculated optimal paths in the constructed overlay network. The median of this data is shown in Fig. 8. Since we used the King data set for this experiment, all end systems have joined the overlay network after 46.2 seconds of simulation time. The system converged towards a stable state at 210 seconds after the start of the simulation. This means that the overlay protocol reaches a stable state after roughly 3.5 times the duration of a high churn period.

E. Comparison With Pastry

As another illustration of efficiency of our overlay building protocol, we compare it with the open source implementation of the Pastry protocol [9]. We compared the relative lengths of optimal paths in the fisheye overlay network and the path lengths using the Pastry routing. For this comparison, we used the Planet Lab data set. For our approach, we used $S = 18$ (18 neighbors). Although Pastry is topology aware and has much larger routing tables (potentially more than 100 neighbors), Fig. 9 shows that after the stabilization phase, our approach outperforms the Pastry overlay by a factor of 7.

VII. Conclusions

In this paper we have presented a fully distributed method for topology aware selection of neighbors in order to construct an overlay network. This method is based on providing a fisheye view of the overlay network to each end system participating in it. The basic principle of the proposed method is to incrementally refine the fisheye view of each end system. The constructed overlay network is guaranteed to be a fully connected bidirectional graph and has a constant upper limit of known neighbors (limited maximal fan out). Those guarantees make such an overlay network a perfect candidate for numerous overlay applications such as topology aware multicast service or as the overlay network for an unstructured P2P network. In the evaluation we showed that the overlay network constructed using our method outperforms the one constructed by an already existing binning overlay building strategy in terms of relative path length of routes.

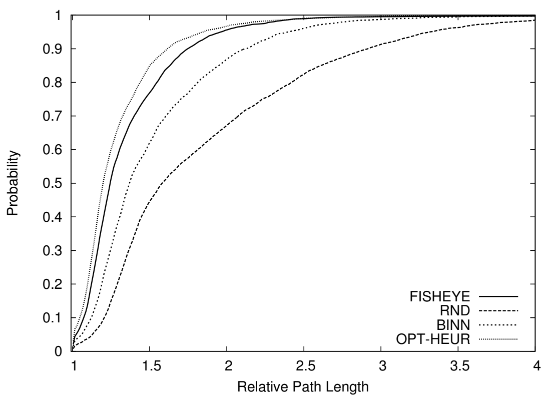

Fig. 7. CDF of relative path lengths in the overlay network with 21 neighbors for the King data set.

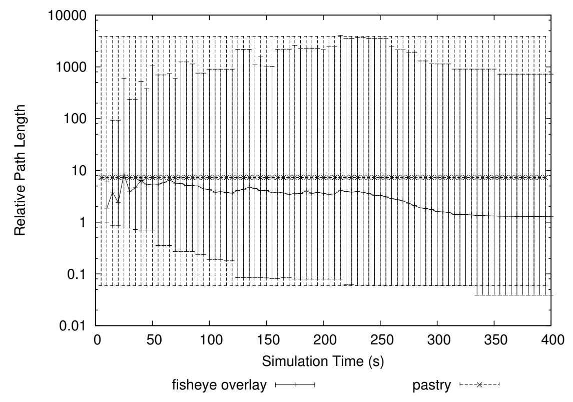

Fig. 9. Relative path lengths of FISHEYE overlay network and Pastry

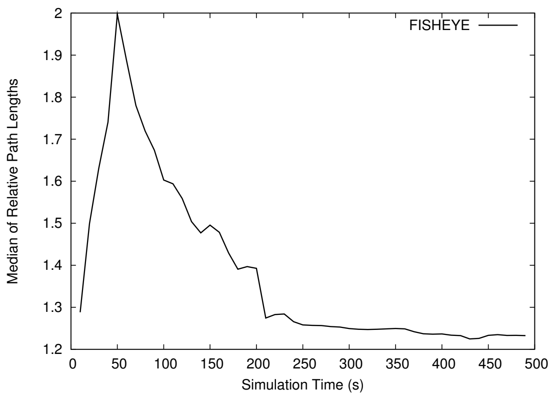

Fig. 8. Convergence at high churn of the FISHEYE overlay network with 19 neighbors using the King data set.

ACKNOWLEDGMENTS

This work was carried out within the Swiss National Science Foundation project "Efficient and Robust Overlay Networks (ERON)" (SNF No. 200021-109270/1).

REFERENCES

[1] D. G. Andersen, N. Feamster, S. Bauer, and H. Balakrishnan. Topology inference from bgp routing dynamics. In *IMW '02: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, pages 243–248, New York, NY, USA, 2002. ACM.

[2] S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable application layer multicast. In *SIGCOMM '02*, volume 32, pages 205–217, New York, NY, USA, October 2002. ACM Press.

[3] Y.-H. Chu, S. G. Rao, and H. Zhang. A case for end system multicast (keynote address). *SIGMETRICS Perform. Eval. Rev.*, 28(1):1–12, 2000.

[4] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: a decentralized network coordinate system. In *SIGCOMM '04*, pages 15–26, New York, NY, USA, 2004. ACM Press.

[5] S. E. Deering. Multicast routing in internetworks and extended lans. *SIGCOMM Comput. Commun. Rev.*, 18(4):55–64, 1988.

[6] S. R. et al. A scalable content-addressable network. In *Proceedings of ACM SIGCOMM*, 2001.

[7] P. Francis. Yoid : Extending the Multicast Internet Architecture. April 1999. White paper, http://www.aciri.org/yoid.

[8] K. P. Gummadi, S. Saroiu, and S. D. Gribble. King: estimating latency between arbitrary internet end hosts. *SIGCOMM Comput. Commun. Rev.*, 32(3):11–11, 2002.

[9] J. Hoye. Freepastry avail. online: http://freepastry.org/.

[10] P. Kirk. Gnutella - a protocol for a revolution url: http://rfc-gnutella.sourceforge.net/, 2003.

[11] Z. M. Mao, J. Rexford, J. Wang, and R. H. Katz. Towards an accurate as-level traceroute tool. In *SIGCOMM '03*, pages 365–378, New York, NY, USA, 2003. ACM.

[12] P. Maymounkov and D. Mazieres. Kademlia: A peer-to-peer information system based on the XOR metric. In *Proc. 1st Internat. Workshop on Peer-to-Peer Systems (IPTPS'02)*, pages 53–65. Springer, 2002.

[13] T. S. E. Ng and H. Zhang. Predicting internet network distance with coordiantes-based approaches. In *IEEE Infocom02*, New York / USA, June 23-27 2002.

[14] T. S. E. Ng and H. Zhang. A network positioning system for the internet. In *USENIX 2004*, pages 141–154, Boston MA, USA, June 2004.

[15] OMNET++, avail. online:http://www.omnetpp.org, 2007.

[16] G. Pei, M. Gerla, and T.-W. Chen. Fisheye state routing in mobile ad hoc networks. In *Proceedings of ICDCS Workshop on Wireless Networks and Mobile Computing, April 2000, Taipei, Taiwan*, pages D71–D78. ICDCS, April 2000.

[17] C. Plaxton, R. Rajaraman, and A. W. Richa. Accessing nearby copies of replicated objects in a distributed environment. In *Proceedings of the ACM SPAA*, pages 311–320, Newport, Rhode Island, June 1997.

[18] S. Ratnasamy, M. H, R. Karp, and S. Shenker. Topologically-aware overlay construction and server selection. In *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, 2002.

[19] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for largescale peer-to-peer systems. IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), Nov. 2001.

[20] A. Rowstron, A.-M. Kermarrec, M. Castro, and P. Druschel. Scribe: The design of a large-scale event notification infrastructure. In J. Crowcroft and M. Hofmann, editors, *Networked Group Communication, Third International COST264 Workshop (NGC'2001)*, volume 2233 of *Lecture Notes in Computer Science*, pages 30–43, Nov. 2001.

[21] Y. Shavitt and T. Tankel. Big-bang simulation for embedding network distances in euclidean space. *IEEE/ACM Trans. Netw.*, 12(6):993–1006, 2004.

[22] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: a scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Trans. Netw.*, 11(1):17–32, 2003.

[23] B. Wong, A. Slivkins, and E. G. Sirer. Meridian: a lightweight network location service without virtual coordinates. In *SIGCOMM '05*, pages 85–96, New York, NY, USA, 2005. ACM.

[24] C.-C. Yang and L.-P. Tseng. Fisheye zone routing protocol: A multi-level zone routing protocol for mobile ad hoc networks. *Comput. Commun.*, 30(2):261–268, 2007.

[25] C. Yoshikawa. Planetlab all-sites-pings experiment, 2006.

[26] Napster webpage: http://napster.com/.