Generic emergent overlays in arbitrary peer identifier spaces

Wojciech Galuba and Karl Aberer

Ecole Polytechnique Fédérale de Lausanne (EPFL),
School of Computer and Communication Sciences,
{wojciech.galuba, karl.aberer}@epfl.ch

Abstract. Unstructured overlay networks are driven by simple protocols that are easy to analyze and implement. The lack of structure, however, leads to weak message delivery guarantees and poor scaling. Structured overlays impose a global overlay topology that is then maintained by all peers in a complex protocol. In contrast to unstructured approaches the structured overlays are efficient and scalable, but leave little flexibility in how their topology can be adapted to the needs of the application.

We propose a generic overlay maintenance and routing algorithm that combines the simplicity of the unstructured overlays and the scalability of the structured approaches, while allowing the application to define its own peer identifier space. The overlay topology is not explicitly defined but emerges in a self-organized way as the result of simple maintenance rules. Independently of the identifier space used, our algorithm exhibits logarithmic scaling of the average routing path length and the average node degree.

The proposed maintenance and routing algorithm is simple and places few constraints on how peers can open their connections. This together with the ability to adjust both the identifier space and the tradeoff between the path length and the node degree makes the overlay customizable in ways that are not possible in the existing approaches.

1 Introduction

Most of the state-of-the-art structured overlay networks [1, 2, 3, 4, 5, 6, 7, 8] follow a similar design paradigm. First, a global network structure is defined and the peers are placed in some identifier space [9]. The structure has properties desirable from the application point of view such as logarithmic routing path length, high routing path redundancy and resilience to failures. Then, this global network structure is expressed as invariants which are maintained by each node locally to ensure the coherence of the global network structure. Structured overlays despite their performance guarantees, have complex distributed protocols and the applications have little flexibility and control over how the overlay topology is formed.

In contrast, in unstructured overlays [10, 11], the design process starts from the local goals and rules without any particular target global structure in mind. The resulting algorithms and protocols are simple and offer much flexibility in forming the overlay topology and routing the messages, however they lack the routing efficiency and scaling guarantees of structured overlays.

In this paper we take an approach to overlay network maintenance that combines the flexibility and ease of implementation of the unstructured overlays with the efficiency and scalability of the structured overlays.

We start out by observing the common characteristics of all structured overlays. Each node is endowed with an identifier and in every structured approach there is some notion of distance defined between the identifiers. Through the distance function each node can know its position relative to other nodes in the topology. In each routing hop the message routed in the overlay is brought closer to its destination in terms of the identifier space distance. To ensure the progress of messages towards the destinations each node maintains a set of connections. The global overlay connection topology guarantees efficient and scalable delivery of messages.

We abstract out the concept of distance between overlay identifiers and allow the application to specify it. We propose a generic overlay routing and maintenance algorithm that relies solely on knowledge of the identifier distance function and does not specify any global topology that needs to be maintained. The global topology emerges in a self-organized way as a result of a simple connection opening rule.

The node identifiers are selected uniformly randomly from the identifier space. In each routing step we greedily route the message to the next hop that is closest to the destination. We compute the rate at which the message approaches its destination, i.e. how much the distance to the destination is shortened during one hop. We require that the rate for each hop be at least γ, a design parameter. If this condition is not satisfied then additional connections are opened by the maintenance algorithm to ensure the minimal rate of message progression towards the destinations.

To verify the claim that the above simple algorithm is indeed able to form an efficient overlay, we evaluate our overlay in simulation. We observe that:

- The resulting overlay has logarithmic scaling properties. Both the average routing path length and the average node degree are logarithmic in terms of the number of nodes in the network.
- The routing path length vs. node degree tradeoff can be controlled by adjusting the single design parameter γ.
- Logarithmic overlay scaling can be achieved for any identifier distance metric by adjusting γ
- Local structures emerge tightly interconnecting nodes in the identifier space on short distances. This common characteristic is shared by all state-of-the-art structured overlays and is crucial for e.g. last hop routing and key replica management in DHTs
- The overlay is robust and has low maintenance overhead even in presence of high churn

The overlay networks generated by our algorithm have characteristics comparable to many of the well-known approaches, while offering a number of additional advantages:

- Up to our knowledge it is the first overlay maintenance algorithm that is driven by the overlay traffic, i.e. connections are created only when they are needed.
- The abstract space of node identifiers and distances between them generalizes over the previous approaches.
- In contrast to other structured approaches there is no pre-defined rigid global structure that has to be maintained, in our case the topology emerges in a self-organized way as a function of the underlying identifier space. The lack of pre-defined topology greatly simplifies the algorithm and minimizes the implementation effort.
- The proposed algorithm leaves plenty of room for adjusting the routing efficiency and the number of connections the nodes need to maintain, this combined with the generalized identifier space gives a level of customizability not available in other overlays.

2 The model

In this section we present the basic assumptions followed by the formulation of the proposed overlay routing and maintenance algorithm.

2.1 Graph embedded in a metric space

Let the network be represented by a graph $G(V, E)$, where V is the set of overlay nodes and E is the set of overlay connections between them. Let $id : V \rightarrow I$ be a function that assigns an identifier from the set I to each of the nodes in V. Let $d : I \times I \rightarrow \mathbb{R}$ be the distance function. The id function embeds the nodes in the metric space defined by d, hence d must satisfy the four properties of the distance function in a metric space: non-negativity, symmetry, identity and the triangle inequality. The pair (I,d) is the *identifier space* and the *mapping function id* maps the overlay nodes into that space.

The communication in the network proceeds by sending messages. The messages can only be sent along connections. Once a connection is established between two nodes it can be used to send messages in both directions.

2.2 Routing

Assume a source node $m.source$ wants to send a message m to a destination node $m.destination$ ($m.f$ denotes the field f in message m). The routing proceeds in the standard hop-by-hop way. The next hop is selected by greedily minimizing the identifier space distance to $m.destination$ and at the same time avoiding previously visited nodes.

Let $m.visited$ be the set of nodes through which m has already been routed. Let v_c be the node that currently holds m and needs to forward it. Let $neigh(v_c)$ be the set of neighbors of v_c. The node v_c selects the next hop v_{nh} based on the following rule: take the set $neigh(v_c) \setminus m.visited$, and from it select node v_{nh} for which the value $d(id(v_{nh}), id(v_d))$ is the lowest. After selecting the next hop, v_c adds v_{nh} to $m.visited$ and forwards m to v_{nh}.

The following special cases occur:

- **NHimp** - next hop impossible - a message reaches a dead end, $neigh(v_c) \setminus m.visited$ is an empty set, the message is dropped
- **TTL0** - TTL zero - each message has a time-to-live counter decremented with every hop, when it reaches zero the message is dropped

2.3 Overlay maintenance

The maintenance in our overlay is driven by routing. To ensure eventual delivery, each routing hop should advance messages closer towards to the destination in the space defined by d. What is more, this advancement should occur at a certain minimal rate of progression to provide efficient overlay message delivery, otherwise new connections have to be created to ensure that this happens. We base our overlay maintenance algorithm on this simple maintenance rule.

For a given next hop v_{nh} from the current node v_c towards the destination $m.destination$ we define the *routing convergence rate* as $cvg(v_c, v_{nh}, m.destination) = \frac{d(id(v_c), id(p.destination))}{d(id(v_{nh}), id(m.destination))}$. Let γ be the minimum required routing convergence rate. Routing convergence rate is the measure of how much the next hop shortens the distance to the destination. If a hop $(v_c \rightarrow v_{nh}, m.destination)$ does not satisfy the condition $cvg(v_c, v_{nh}, m.destination) \geq \gamma$ then that hop is *weak*, otherwise it is *strong*.

When a weak hop is encountered while routing a message m, the maintenance protocol sends a connection request $cr = (v_c, p.destination)$ with the destination set to $m.destination$ indicating the origin of the request as v_c, the current node. The connection request is routed towards the destination normally as other messages in the greedy self-avoiding way. When some node v_{resp} receives a connection request $cr = (v_o, dest)$ and if the hop $(v_o \rightarrow v_{resp}, dest)$ is not weak then v_{resp} responds to v_o with connection acknowledgement and the connection between v_o and v_{resp} is established and the routing of cr stops. Otherwise if $(v_o \rightarrow v_{resp}, dest)$ is weak, the cr continues to be routed.

When a timeout happens while sending on one of the connections then the sender closes that connection and removes the recipient from its neighbor set.

2.4 Maintenance suppression

Every node keeps track of the connection requests that it has sent until either the corresponding connection response arrives or a timeout occurs. This request-response tracking has the following purpose. Consider the time between two events: (1) the sending of a connection request $c(v_c, dest)$ by node v_c and (2) the receipt of the corresponding connection response. Assume additionally that there are no connection requests being sent or responses arriving during that time. However, there may be many messages with the same destination $dest$ arriving at v_c. According to the proposed maintenance algorithm each of these messages takes a weak next hop and triggers a connection request, which is identical to the one already sent earlier. This would lead to the generation of many unnecessary connection requests.

To prevent this from happening, whenever some $cr = (v_c, dest)$ is about to be sent the list RL of requests currently awaiting their responses is checked. If any of the potential responders to the connection requests on RL is also a valid responder to cr, then cr is not sent. Let $cr' = (v_c, dest') \in RL$ be some connection request awaiting its response. Let v_r be the potential responder to cr', v_r must satisfy the condition (1) $cvg(v_c, v_r, dest') \geq \gamma$. If v_r is also a valid responder for cr, then it also satisfies (2) $cvg(v_c, v_r, dest) \geq \gamma$. We also know that the three identifiers of the three nodes $(v_r, dest, dest')$ must satisfy the (3) triangle inequality. Combining (1), (2) and (3) we obtain (4) $\gamma d(id(dest), id(dest')) < d(id(v_c), id(dest)) + d(id(v_c), id(dest'))$. If there is any $cr' = (v_c, dest') \in RL$ for which (4) is true then $cr = (v_c, dest)$ is not sent.

The operation of our overlay routing and maintenance algorithm has been summarized in Algorithm 1. Note that for clarity the handling of timeouts, TTL0 and NHimp events has been omitted.

3 Simulation results

In this section we examine the behavior of our routing and overlay maintenance algorithms experimentally.

3.1 Experimental setup

We use a custom-developed event-driven simulator. Each event occurs at a specific moment of the virtual simulated time and all references to time in this section pertain to this simulated time not the real time.

Each node in the simulated network generates messages in a Poisson process. The generation rates are identical across all the nodes. A node v_i generates a message m with the destination $v_j \neq v_i$ selected uniformly randomly.

Simulation execution is divided into measurement epochs. At the beginning of each measurement epoch all the measured variables are reset. At the end of a measurement epoch the accumulated measurements are logged. Each measurement epoch by default lasts 30s. The average message generation at all nodes is set to one message every second. All messages, connection requests and connection responses are delivered with a latency uniformly distributed between 100ms and 200ms. The time-to-live for overlay messages is set to 100 hops. We do

Algorithm 1: Overlay routing and maintenance algorithm for arbitrary peer identifier spaces

```
initialize
    $RL \leftarrow \emptyset$
    $neigh \leftarrow$ initialize with peers via bootstrap mechanism

// application calls this function to send messages
function sendMessage(payload,dest)
    forwardMessage(Message(payload,self,dest,$\emptyset$))

function forwardMessage(Message(payload,src,dest,visited))
    $next\_hop = argmin_{x \in neigh \setminus visited} d(id(x), id(dest))$
    send Message(payload,src,dest,visited $\cup$ self) to next_hop
    if payload is ConnectionRequest then
        return
    end
    // maintenance is triggered by weak hops
    // only for payload that is not a ConnectionRequest
    if $\frac{d(id(self),id(dest))}{d(id(next\_hop),id(dest))} < \gamma$ then
        // check the maintenance suppression condition
        if $\neg\exists_{cr'(self,dest') \in RL} \gamma d(id(dest), id(dest')) <$
        $d(id(self), id(dest)) + d(id(self), id(dest'))$ then
            $RL \leftarrow RL \cup cr(self, dest)$
            // route the connection request as a new message
            forwardMessage(Message(ConnectionRequest(self,dest),self,dest,$\emptyset$))
        end
    end

receive Message(payload,src,dest,visited)
    if payload is ConnectionRequest(origin,dest) then
        // check if can accept the connection request
        if $\frac{d(id(origin),id(dest))}{d(id(self),id(dest))} \geq \gamma$ then
            send ConnectionResponse(self,dest) to origin
            return
        end
    else if dest=self then
        deliver Message to application
        return
    end
    forwardMessage(Message(payload,src,dest,visited))

receive ConnectionResponse(responder,dest)
    $RL \leftarrow RL \setminus cr(self, dest)$
    $neigh \leftarrow neigh \cup responder$
```

not explicitly simulate the underlying network topology. This simple network model is sufficient for verifying the correctness of our algorithm and the structural properties of the overlay, performance testing in a more realistic setting is left as future work (Section 4).

The bootstrap process is as follows. Each simulation begins with a set of 30 nodes interconnected uniformly randomly with an average degree of 5. The size of this initial network is large enough to ensure that it remains a connected graph even if some of the initial nodes depart at the beginning of the simulation. Each new joining node connects to 5 uniformly randomly selected neighbors from the network. These bootstrap connection requests and responses are delivered directly and are not routed in the overlay. In a concrete implementation the peers would be bootstrapping from a known host list (e.g. downloaded from the Web), we do not simulate this process in detail since our overlay is not sensitive to the choice of initial neighbors for the peer.

To demonstrate that the results are independent of the chosen identifier space we select five representative spaces for the experiments:

- 1D - one dimensional ring, as in Chord[1], $I = [0, 1)$, $d(a, b) = min_{k\in\{-1,0,1\}}|a - b + k|$
- 2D - two dimensional spherical coordinates, the identifiers are placed on the sphere $I = [0, 2\pi)^2$ and the shortest distance is measured along the great circle crossing the two identifiers
- 3D - three dimensional Euclidean space with wraparound (surface of a 4D hypertorus).
- PFX - prefix routing as in Pastry[4] with the identifier space of 128bit vectors, assume we are computing $d(a, b)$, bits in a and b are compared from the highest order bit to the lowest order bit, if i is the index of the first bit which differs between a and b, then $d(a, b) = 2^i$
- XOR - XOR distance metric as in Kademlia [5], an XOR of two identifiers is computed and the result is taken as an integer distance value with 160 bits

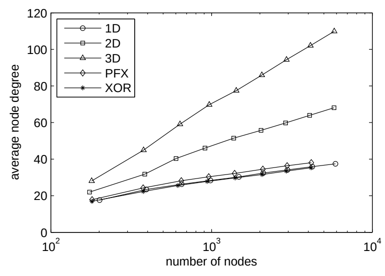
For all the identifier spaces the nodes select their identifier uniformly randomly out of the set of all possible identifiers.

The arrivals and departures of the nodes (churn) are simulated. Arrivals are a Poisson process with a default average rate of 0.002 nodes per second. The lifetime of the nodes is power-law distributed [12] with the minimal lifetime of 10s and the exponent of -1.2.

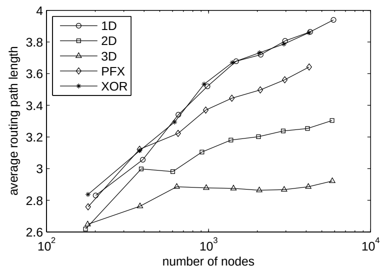
During the simulation we track the number of TTL0 and NHimp events (Section 2.2). The parameters for most experiments are adjusted in such a way that the number of routing failures is negligible. We devote section 3.3 to the study of routing failures under high churn conditions.

3.2 Scaling

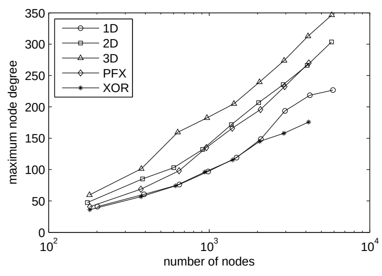
To test the fundamental scaling properties of the overlay we let it increase in size over time by setting the minimal node lifetime to a higher value of 50s. For different network sizes and identifier spaces we measure the average routing path length and the average and maximum degrees. The average path length is measured over all the messages that have reached their destinations. This excludes the connection requests and responses. The node degrees are measured on the snapshot of the overlay topology at the end of a measurement epoch.



(a)



(b)



(c)

Fig. 1. The scaling of the average path length and the node degree. Each data point is an average of 20 measurements. Standard deviation bars are omitted for clarity. Standard deviation does not exceed 10% for all the points.

The measurements are plotted on Figure 1. Both the average path length and the average degree scale logarithmically in terms of the number of nodes as in the state-of-the-art overlays. In addition the logarithmic scaling of the maximum degree is evidence for a balanced degree distribution, which is crucial for balancing the message forwarding load among the peers.

The case of the 3D identifier space is an outlier in our scaling experiments. In contrast to other identifier spaces, the average degree is rising much more rapidly. This is caused by the "curse of dimensionality" problem, which we discuss in Sect. 3.5.

3.3 Maintenance overhead and failures in extreme churn conditions

Apart from scaling, another important characteristic of an overlay is its maintenance overhead and resilience to node departures or failures. To measure these characteristics we switch from the power-law node lifetime model to Poisson arrivals and departures, which helps avoid the central core of long-lived nodes that typically forms under power-law lifetime model. This allows us to have more control over the number of nodes in the network and creates a more extreme case of churn. Both the rate of arrivals and the rate of departures are gradually increased. The two rates are adjusted such that the network size oscillates around 1000 nodes. To stress test the overlay we set the churn rates to values that are considerably higher than those typically seen in peer-to-peer network deployments.

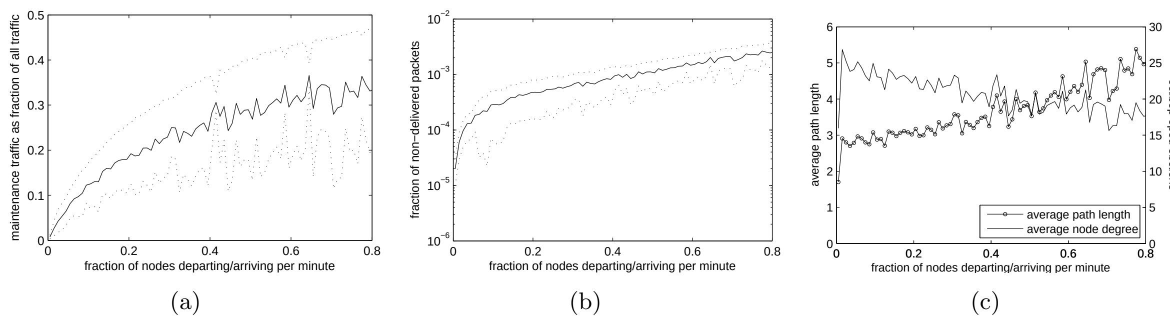


Fig. 2. Maintenance cost and failure rates under extreme churn conditions. The plots aggregate results from 20 independent experiments. Standard deviations are marked with a dotted line. Churn rate is measured as the number of arrivals or departures that happened during a measurement epoch as a fraction of the size of the network. Since the arrival and departure rates are the same, a churn rate of 0.8 means that 40% of the nodes have been replaced by new ones in one minute. Non-delivered messages are those that have never reached their destination due to a TTL0 or an NHimp event or due to a forwarder departure between receiving the message and forwarding it further. Traffic is measured as the number of forwarded or sent messages.

Figure 2 summarizes the results. Our overlay maintenance algorithm keeps the routing failures under 0.2% even when 40% of the nodes are replaced with new ones every minute. For small churn rates the maintenance traffic increases faster with the increasing churn, when the churn is higher the massive parallelism in connection request sending lowers the number of connection requests a newly joined node needs to open as it is more likely to receive connection requests from other newly joined nodes. As the churn rate increases the average degree decreases, there are more missing connections in the topology caused by churn. Even though some poorly connected nodes might lay on the routing path, its average length increases only slightly in high churn conditions.

The overall robustness of our overlay is high, the overlay does not loose connectivity, high routing efficiency and low failure rates are maintained.

3.4 Varying the γ parameter

The routing convergence parameter γ is crucial in our algorithm. We explore experimentally how the changes of this parameter influence the performance of the overlay. For *gamma* values varying from 0 to 4 and for the different identifier spaces we grow the network until it reaches 1000 nodes. For the resulting network we measure the average number of hops and the average node degree.

The results show (Figure 3) that adjusting the γ parameter allows for precise control of the path length vs. degree tradeoff. Distinct operational regimes can be defined:

– **low degree** - $\gamma \ll 1.0$ - most of the hops are strong and only a few new connections are opened, message routing relies more on the self-avoidance property of the routing algorithm, many nodes need to be visited as the routing gradually and mostly randomly converges towards the destination, messages are frequently dropped due to the TTL0 and NHimp events (Section 2.2)
– **high degree** - $\gamma \gg 1.0$ - most of the hops are weak and a large number of connections needs to be opened to form strong hops to the different areas of the identifier space, the convergence of a message is guaranteed by the high γ value, the distance to the destination exponentially decreases (at least by the γ factor in each hop), self-avoidance rarely has to be used and the average path length is small
– **balanced** - $\gamma \approx 1.0$ - in this regime the scaling of both the average degree and the average number of hops are logarithmic in terms of the number of nodes as demonstrated in section 3.2.

In the next section we discuss the high degree regime further and provide a way for selecting the γ parameter such that the overlay operates in the balanced regime.

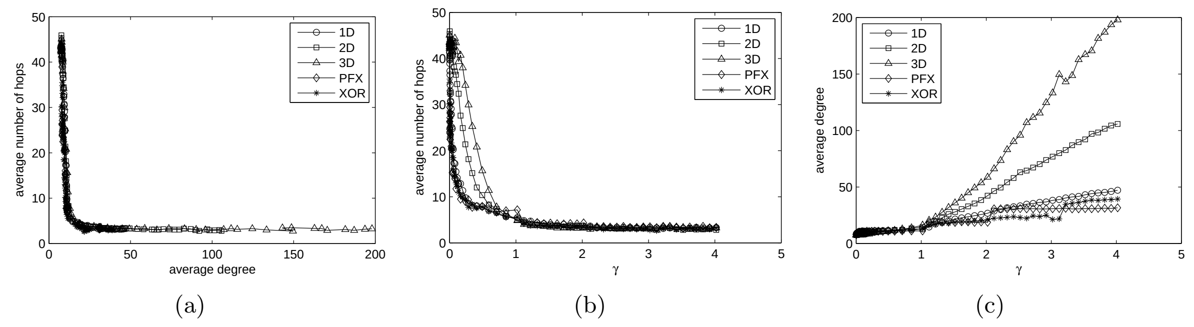


Fig. 3. The influence of varying the γ parameter on the performance of the overlay. Each point represents a measurement on a separate overlay that was independently run.

3.5 Routability

For a given identifier space only for some γ values the overlay is in the balanced regime and we clearly need a way of determining these values. To achieve this we define the concept of routability. The *routability* $R(\gamma, I, d)$ of the identifier space (I, d) under the convergence rate γ is the expected probability of finding a strong next hop taken over all possible sources and destinations. Let X be the random variable describing the distance between two random identifiers in the (I, d) space, then $R(\gamma, I, d)$ is $\int_0^{x_{max}} \mathbb{P}(X = x)\mathbb{P}(0 < X < \frac{x}{\gamma})dx$. The values of R are computed numerically for the different spaces and values of γ (Figure 4). Low routability values indicate that the network will operate in the high degree regime and vice versa high routability is a good predictor of the low degree regime (compare these results to Figure 3).

To provide an extreme case we have included a highly dimensional space EUC50, the surface of 51-dimensional hypertorus with Euclidean distance metric. The value of routability for EUC50 at $\gamma = 2.0$ is very close to 0.0 and for $\gamma = 1.1$ it is 0.17 which we have verified experimentally to be enough to provide logarithmic scaling. This result also demonstrates that highly dimensional spaces are not good a good choice for routing due to their "curse of dimensionality" [13]. Only when the node degree is very high can efficient routing be achieved.

The routability concept can be conveniently used to find the ranges of γ values and identifier spaces for which the network operates in the balanced regime, i.e. with routability values in the mid-range, close to 0.5. It has to be noted that this is a necessary condition for good scaling of the network, not a sufficient one.

Routability depends on the variable X which among other factors depends on the distribution of the identifiers in the identifier space, which thus far was assumed to be uniform. If

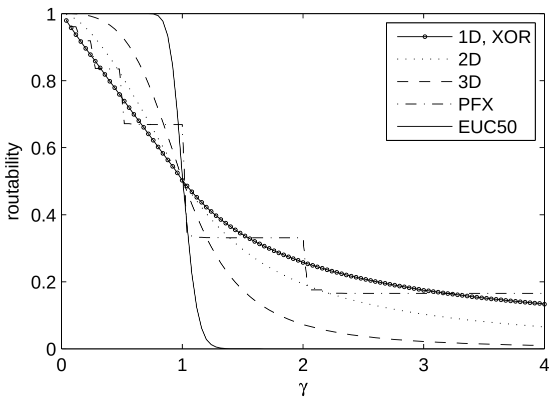


Fig. 4. Routability for the different identifier spaces and values of γ.

that distribution is skewed, this may greatly influence the routability value. The routability formula should also factor in the actual message traffic distribution which we assumed to be uniformly distributed over the set of all source-destination pairs. Exploring these dependencies is beyond the scope of the paper and is left as future work.

3.6 The emergence of local structures

Some overlays maintain a completely deterministic set of neighbors, others create random links [8]. However, all of the current structured approaches maintain at least one deterministic connection, usually to its closest neighbor. Those connections are eagerly maintained and are crucial to reliable routing, especially at the very last hop. In our routing and maintenance algorithm, the creation of this type of connections is not explicitly a part of the algorithm. However, nodes following the simple local rule for sending the connection requests create a dense network of short-range links.

To measure this effect we define the value of *locality* for each node. With a vertex $v_i \in V$ we associate the series of vertices $l_{i1}, l_{i2}, ...l_{i(n-1)}$ where $n = |V|$, and $\forall_{i,j,k} d(id(v_i), id(l_{ij})) < d(id(v_i), id(l_{ik})) \Rightarrow j < k$. Then locality of v_i is equal to m if $\forall_{k=1..m} l_{ik} \in neigh(v_i)$. Informally, the locality of v_i is the number of nodes closest to v_i in the identifier space such that all of these closest nodes are connected to v_i.

We take the topology snapshots of the overlays constructed in different identifier spaces with 1000 nodes. We measure the fraction of nodes with the given value of locality (Figure 5). The topology snapshot is taken while the overlay is churning and thus some of the nodes may

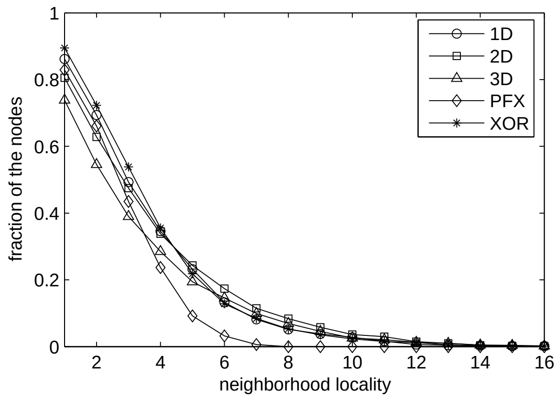


Fig. 5. The fraction of nodes with a given locality. Each point is an average of 5 independent measurements. Standard deviations are negligible.

have just joined the overlay and have not opened a sufficient number of connections or some of the nodes may have lost a connection due to a neighbor's departure and have not replaced this missing connection with another one. Churn decreases the measured fractions of nodes. Despite churn, 80-90% of nodes are connected to their closest neighbor, and 70-80% of the nodes are connected to two of their closest neighbors. In a 1D ring identifier space if all of the nodes have locality 2 then there exists a global ring spanning all of the nodes, it is possible to visit all the nodes by hopping along that ring in one of the two directions. In overlays based on the 1D ring identifier space (e.g. Chord[1]) this global spanning ring is explicitly maintained. In the case of our algorithm, the ring emerges as a result of the simple connection opening rule in the maintenance algorithm. This rule is independent of the identifier space used and similar local structures are universally created in identifier spaces other than the one-dimensional ring.

4 Discussion of results and future work

We have shown how to use the concept of routability to find the range of γ parameters that ensure that the network stays in the balanced operating regime for a given identifier space. We verified experimentally that this regime exhibits optimal overlay scaling properties, however one might ask a question why do the values of γ close to 1.0 still produce networks with logarithmically increasing routing path length, despite the fact that each hop is not required to shorten the distance to the destination. This is explained by the fact that each node has a non-negligible probability of opening a long range link which acts as an effective shortcut in the identifier space. Though there are few of those long range links in the case when γ is close to 1.0 they significantly lower the average path length. We plan to complement our extensive simulations with an analytical treatment of the routing and maintenance algorithms to explore this phenomenon in detail and relate it to other work in small-world networks, such as the results of Kleinberg [14].

Throughout the paper we have assumed that γ is a system-wide constant. However, it may vary in the following ways:

– per node - each node might locally decide at what routing convergence rate it forwards the messages, indirectly this gives the node a way of controlling its degree
– over time - γ can change to adapt to changing network conditions, e.g. churn
– per message - some types of traffic may be prioritized, e.g. traffic with a specific destination might have a higher routing convergence rate associated with it

Fine-grained control over γ gives a considerable degree of flexibility in shaping the overlay, adjusting the length of the routing paths for different types of traffic and deciding which nodes the traffic traverses through. This may be particularly useful in cases when the source-destination distribution of the message traffic is skewed.

In our maintenance algorithm the connection requests are triggered by messages and are normally sent immediately after the message itself towards the same destination as the message. It is very likely that both the connection request and the message that triggered it follow the same routing path. This can be exploited to piggyback the connection requests on the actual application messages. This can be easily implemented since each message already contains the visited list, the connection request can then be a single bit flag attached to a node in the visited list. We plan to implement connection request piggybacking and investigate how much maintenance bandwidth can be saved in this way.

On the other hand, our protocol adds additional overhead to each of the messages by storing the visited list. The size of this list equals to the number of hops and scales logarithmically in network size. The list is used to prevent self looping while routing. However, once the overlay is stable there are no loops and the visited lists are not used. For $\gamma > 1$ Each routing loop has at least one weak hop, this suggests that it may be sufficient to store visited nodes only when the next hop is weak. Moreover, instead of explicitly storing the whole list of visited nodes, Bloom filters on node identifiers could be used to drastically decrease the space needed for storing the list. These optimizations await experimental investigation.

If our overlay is used as the routing substrate for the distributed hash tables, the emergent local structures (Section 3.6) can be used to manage the replicas of the key-value pairs. Furthermore, the node identifier space can be selected such that it better suits the needs of a particular DHT key space and the application that uses the DHT. We plan to implement a DHT on top of our overlay and investigate the advantages brought by flexible identifier spaces.

5 Related work

Our approach is most similar to Freenet [10], which follows simple rules for opening new connections. Moreover, in Freenet, just as in our approach, routing is loop avoiding. However, in Freenet loop avoidance is implemented by keeping the routing state at the nodes, while in our algorithm we keep it in the message, which is a stronger form of loop avoidance. Another major difference is that in Freenet new connections are opened by performing a random walk when the node joins a network. This leads to a scale-free topology with a power-law node degree distribution with a small number of nodes having a very high degree. In our case the maximum degree increases only logarithmically with the overlay size.

We have shown how adjusting γ can be used to trade off between the node degree and the routing path length. In a real network implementation this corresponds to the maintenance bandwidth vs. routing latency tradeoff. This problem has been studied in the context of Accordion [15], where given a bandwidth budget the protocol adjust the number of maintained connections to the current churn levels. The obtained tradeoff curves are similar to the ones in figure 3. A simplified version of this algorithm can be implemented in our overlay by making γ a function of the current churn level and the given bandwidth budget.

A widely studied topic in the domain of complex networks are the small-world graphs. They are commonly defined [16, 17] as graphs having both a small diameter and high clustering coefficient. Our overlay satisfies both of these properties, except to quantify the clustering we employed our own measure - the node locality (Section 3.6). Our overlay can be viewed as a dynamic model for small-world growth. The first dynamic model capable of generating networks having small-world properties was proposed by Watts and Strogatz [18] in 1998. In their approach they start with a regular graph and then modify its structure through random rewiring. The probability of rewiring can be controlled. As the probability increases, the network goes through three topologically distinct stages. First, the topology is highly regular with high average path length, then it is small world and finally completely random with short path lengths but low clustering coefficient. Kleinberg [14] places the nodes on regular multi-dimensional lattices and addresses the problem of connection length distribution that would ensure efficient routing. We generalize on this work further by considering a wider family of spaces in which nodes are embedded and proposing a concrete routing and network growth algorithm that is able to achieve logarithmic routing path scaling. We performed measurements of the distance distribution between connected nodes in topologies generated by our overlay maintenance algorithm. The distributions exhibit very consistent power-law characteristics identical to the ones observed by Kleinberg. However, we do not include these results as they lay outside the scope of the paper.

6 Conclusions

In Gnutella [11], requests are flooded through the network until they reach their destination. The nodes are not embedded in any space. What gives the structured overlays their structure is the space the nodes are embedded in. Once the space is added the flow of messages is given directionality and they no longer have to move in all directions simultaneously to find their destinations but only towards directions that decrease their distance to the destinations. A form of this greedy routing is used in all state-of-the-art structured overlays. Routing decisions are made solely based on local measurements of gradients in the identifier space. The knowledge of the properties of the whole space is not necessary. We have proposed a generic algorithm that for any metric identifier space is able to maintain the overlay and route messages based only on local information and simple rules.

Each overlay runs a maintenance algorithm that opens connections to ensure that for every node every forwarded message is brought closer to the destination in terms of the space distance. Our overlay maintenance algorithm generalizes this rule. The proposed algorithm is parameterized by γ which allows for precise control of the path length vs. degree tradeoff while generating relatively small maintenance traffic even under high churn.

Normally, the overlay maintenance algorithm keeps the network prepared for efficient traffic routing from any source to any destination. In our algorithm maintenance is tied to routing and connections are created on demand and only cover the set of destinations that actually appear in the traffic without opening unnecessary connections. This may be controlled at a finer granularity by associating different γ values with different forwarding nodes and different types of routed traffic.

The main insight of our work is that the identifier space is flexible and the properties of the individual identifier spaces used in structured overlays are not significant as long as the spaces provide a consistent local gradient for the greedy routing algorithm to follow. Moreover, the topology of the overlay does not have to be an eagerly maintained pre-defined rigid structure but can emerge in a self-organized way from simple local maintenance rules that adapt the topology to the identifier space. This departure from the structural rigidity of the overlays opens new possibilities of handling skewed overlay traffic distributions, prioritizing traffic, adapting to inhomogeneous allocation of node identifiers and customizing the identifier space to the needs of the application.

References

[1] Stoica, I., Morris, R., Karger, D., Kaashoek, F., Balakrishnan, H.: Chord: A scalable Peer-To-Peer lookup service for internet applications. 149–160

[2] Aberer, K., Cudré-Mauroux, P., Datta, A., Despotovic, Z., Hauswirth, M., Punceva, M., Schmidt, R.: P-Grid: a self-organizing structured P2P system. SIGMOD Record **32**(3) (2003) 29–33

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

[4] Rowstron, A., Druschel, P.: Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. Lecture Notes in Computer Science **2218** (2001) 329–337

[5] Maymounkov, P., Mazieres, D.: Kademlia: A peer-to-peer information system based on the xor metric (2002)

[6] Bharambe, A.R., Agrawal, M., Seshan, S.: Mercury: supporting scalable multi-attribute range queries. In: SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications, New York, NY, USA, ACM Press (2004) 353–366

[7] Ratnasamy, S., Francis, P., Handley, M., Karp, R., Schenker, S.: A scalable content-addressable network. In: Proceedings of the 2001 conference on applications, technologies, architectures, and protocols for computer communications, ACM Press (2001) 161–172

[8] Manku, G., Bawa, M., Raghavan, P.: Symphony: Distributed hashing in a small world (2003)

[9] Aberer, K., Alima, L.O., Ghodsi, A., Girdzijauskas, S., Hauswirth, M., Haridi, S.: The essence of P2P: A reference architecture for overlay networks. In: P2P2005, The 5th IEEE International Conference on Peer-to-Peer Computing. (2005)

[10] Clarke, I., Sandberg, O., Wiley, B., Hong, T.W.: Freenet: A distributed anonymous information storage and retrieval system. Lecture Notes in Computer Science **2009** (2001) 46–53

[11] Ripeanu, M.: Peer-to-peer architecture case study: Gnutella network (2001)

[12] Bustamante, F., Qiao, Y.: Friendships that last: Peer lifespan and its role in p2p protocols (2003)

[13] : http://en.wikipedia.org/wiki/curse_of_dimensionality

[14] Kleinberg, J.: The Small-World Phenomenon: An Algorithmic Perspective. In: Proceedings of the 32nd ACM Symposium on Theory of Computing. (2000)

[15] Li, J., Stribling, J., Morris, R., Kaashoek, M.F.: Bandwidth-efficient management of DHT routing tables. In: Proceedings of the 2nd USENIX Symposium on Networked Systems Design and Implementation (NSDI '05), Boston, Massachusetts (May 2005)

[16] Newman, M.: The structure and function of complex networks (2003)

[17] Albert, R., Barabási, A.: Statistical mechanics of complex networks

[18] Watts, D.J., Strogatz, S.H.: Collective dynamics of "small-world" networks. Nature **393** (1998) 440–442