A Construction Scheme for Scale Free DHT-Based Networks

Salma Ktari, Artur Hecker, and Houda Labiod

TELECOM ParisTech, Paris FRANCE

In this paper, we propose PowerDHT, a novel scheme to extend the classic DHT-based overlay to a network with scale free-like properties. PowerDHT has a distributed rewiring method to improve the structure of the overlay network to a power-law-like graph. Our scheme is characterized through minimal, typically local-only, changes. Through simulations, we show that our proposal constructs an overlay network with an extended peer’s neighborhood knowledge and a reduced network diameter at no additional cost and that it supports a more effective flooding e.g. for generic search.

Index Terms— key based routing, super peer, power law degree, scale free graph, DHT, flooding.

I. INTRODUCTION

Scale-free topologies are very common in natural and man-made networks. Examples vary from social contacts between humans to technological networks such as the Internet [1] [2]. Recent research [3, 4] shows that networks of people in a social group or the Internet, organize such that most nodes have few links, while a few number of nodes, so-called hubs, have a large number of links. As shown in [4], such scale-free networks may appear in nature due to preferential attachment, where newcomers tend to prefer connecting to nodes that already have a strong presence characterized by their high degree. [5] also shows that the popular peer-to-peer (P2P) network Gnutella is a scale-free-like network, combining power law and quasi-constant distributions.

Such power-law networks are characterized through the node degree distribution of the form $P(k) = k^{-\alpha}$, where $\alpha > 0$ is the exponent of the distribution. In such networks, few nodes exhibit extremely high connectivity, while the majority is poorly connected. These graphs are known to scale well, i.e. to be capable of maintaining a low diameter with the growing network size. For instance, for $2 < \alpha < 3$, $d \sim \log \log N$ [5]. Unstructured P2P networks, where no specific topology is imposed, exhibit a large diameter and high network traffic. This is inefficient because it requires nodes to opt for a low radius search, which would limit the probability of finding less popular objects in the network. These design tradeoffs result in either increased signaling or in degraded performance. In contrast, structured networks [6, 7] impose predetermined connectivity relationships between nodes in order to offer a guarantee on the diameter, typically $\log N$. However, the resulting structure biases the possible search method. In practice, generic search is not supported by DHTs [8].

To address this challenge, [14] proposes to use broadcast mechanisms and random walks in DHTs and showed that the broadcast in DHTs can locate faster contents but results in higher overhead compared to random walks. [15] proposes hierarchical and hybrid approaches reuniting both principals, KBR routing at the higher level between super peers and flooding at the lower level. However, this approach introduces non uniformity and the resulting super node selection problem is very challenging.

In this paper, we propose PowerDHT, a structured P2P algorithm that constructs low-diameter network by creating a topology of nodes, whose degree distribution follows a power-law, while being fully distributed. The resulting topology allows nodes in the network to ‘naturally’ emerge as special nodes with a degree of connectivity higher than the average. The proposed algorithm uses a variant of key-based routing on top of a structured power-law P2P network. We propose also to use flooding on top of the resulting power law DHT, to allow generic keyword searches as in unstructured P2P overlays. Power DHT search achieves short routes and a high query success rate, at no additional cost.

The rest of this paper is organized as follows. We discuss the motivation for our work in section 2 and then, in section 3, we detail our search model applied to Chord and Pastry DHTs. In section 4, we describe the simulation environment and provide performance results. Finally, we conclude and discuss possible future works in section 5.

II. OBSERVATIONS FROM REAL DHT TOPOLOGIES

A DHT maps object IDs (keys) to the nodes of the network and provides algorithms for reaching the peer responsible for a given key. DHT designs differ mainly through the way they maintain the calculated neighbors and sequential neighbors [9] and perform lookups: there is a fundamental tradeoff between the necessary network state and its diameter. Links in DHTs are unidirectional and only outdegree distribution is well-known [10]. Typically, the node out-degree for an $N$-node network is in $O(\log N)$.

We studied the indegree distribution of popular DHTs geometries Chord and Pastry. We measured the number of indegree links of node for network size of 20000 nodes and calculated the number of nodes corresponding to each indegree value. Figure 1 shows the resulting indegree distributions for Chord and Pastry respectively. We observe the fact that, although in conventional DHTs nodes maintain a well known number of outdegree neighbors, the indegree distribution is neither uniform nor normal. From the chart, we can see that it is similar to a power-law (PL) distribution. We have investigated potential factors that may have led to this skew. For instance, we considered whether it resulted from the big size of the identifier space compared to the limited number of nodes. However, simulation
experiments indicate that the skew exists even in the topologies with smaller identifier space sizes. What is the reason for this skew?

Since IDs are randomly placed in the Chord ring, node zones will typically have very different sizes [3]. Then, the larger the size of the zone of responsibility of a node, the larger the probability that a finger falls within this zone and therefore the more inbound fingers this node will have. [11] demonstrated through analytical analysis that this non uniform indegree distribution in Chord is mainly due to the different sizes of the zones between nodes.

For Pastry, in addition to the different sizes of the zones between nodes, we believe that the skew is also due to Pastry’s join and repair mechanisms. In Pastry, a new node picks up routing table entries from other nodes in the system. While this reduces join/repair times and overheads, it makes nodes that joined earlier far more likely to be picked as neighbors as compared to other nodes. This unbalanced in-degree distribution is additionally amplified by the Pastry’s neighbor selection mechanism that tends to prefer nodes with better proximity. This suggests that choosing a certain metric as a criterion produces an unbalanced overlay structure: some nodes are a better choice and therefore have higher in-degrees.

Chord and Pastry indegree distribution is not uniform. In fact, most nodes have about log N neighbors, but some nodes have substantially more “contacts”. Nevertheless, Chord and Pastry DHTs do not use all this additional knowledge for routing, but limit themselves to outdegree neighbors.

In the light of these observations, it seems interesting to construct a DHT-based P2P topology that conforms to a power law for its node degree distribution. Our idea is to extend the DHT graph to use its full neighborhood knowledge for routing and, therefore, to establish a functioning distributed scale-free network.

III. SCALE FREE DHT NETWORK CONSTRUCTION

The main motivation behind Power DHT is to allow nodes in the network to ‘naturally’ emerge as special nodes with a degree of connectivity higher than the average, so that scale-free topology can be formed.

Barabasi-Albert (BA) algorithm [12] is the most popular model that proposes a generation algorithm of network whose degree distribution follows a power law. The BA is characterized through the preferential attachment (PA) paradigm. The network evolves by one node at a time, and this new node is connected to $m$ different existing nodes with probability $p_i = k_i / \sum k_j$, where $k_i$ is the degree of the node $i$.

In order for a power-law topology construction to be practical in distributed P2P overlays, it must allow joining of new nodes by just using locally available information. For this purpose, PowerDHT adopts a heuristic inspired by BA: the preferential attachment is performed on the known sequential neighbors to amplify the effect.

To construct a DHT scale-free topology, the first step is to modify the neighbor selection. Instead of selecting the calculated neighbor according to the respective DHT algorithm, we choose one of the node’s sequential neighbors with probability $p_i$ proportional to their degrees (outdegree plus indegree). This results in a DHT graph with a steeper distribution of node degrees. In the second step, for each neighbor link, we add the reverse link that routes queries between the same two nodes but in the opposite direction. These new links are stored in a new table named reverse table. The resulting topology is a scale-free DHT graph, where each node has a fixed number of routing links but some nodes have substantially more routing links. We use this new topology for routing and object search.

In what follows, we show how the PowerDHT idea can be applied to Chord and Pastry respectively. Note that the general idea is applicable to other structured P2P systems.

A. PowerChord topology construction

Chord [6] structures its identifiers in a circle. The node responsible for a key $\psi$ is its successor. Each Chord node keeps $log N$ neighbors called fingers, whose IDs lie at exponentially increasing fractions of the ID space away from itself. These neighbors are stored in the routing table named finger table (FT). Each node also keeps a successor list of $n_{succ}$ sequential neighbor. In Chord, a lookup for a key terminates at the key’s predecessor and returns its successor as the lookup return value. Periodically, Chord ring runs maintenance algorithms using fix finger messages that detect failures and repair finger tables. This allows requests for a key to be routed correctly to their owner in spite of node churn.

1) Neighbor selection in Power Chord

To apply the PowerDHT idea, when a node is selected to be a finger, instead of using this node directly, we choose one of its successors with probability $p_i$ proportional to their degree.

In more detail, the finger selection algorithm of Power Chord works as follows. Like in basic Chord, when a node with an identifier $i_{\psi}$ creates its fingers table, it starts by performing a lookup to determine the nodes $n_i$ with an identifier equal to or greater than $ID_i = i_{\psi} + 2^{-i}$. Then, instead of selecting a particular $\psi_{n_i}$ as finger (as it would be done in basic Chord), the finger is picked up from the set of $n_{succ}$ successors of $n_i$ in the interval $[id + 2^{-i}, id + 2^{-i}]$ according to the probability $p_i$. Figure 3 shows the resulting indegree distribution (linear and log scale). From the chart, we estimate that it follows a power law with $\alpha$ estimated at 2.3.

2) Additional data structure

In Power Chord, a node maintains a new reverse table RevT, in addition to the classic finger table. If there is a finger link from one node to its finger, we add a reverse finger link that links the same two nodes, but in the reverse direction. These reverse links connect the node to its finger selectors (FS), i.e. the nodes that selected it as a finger. In addition to the finger selector ID information, the reverse table stores the predecessor’s ID of the corresponding node. To do that, we extend the existing fix finger message with the ID of the finger selector predecessor. Knowing the FS predecessor ID, every node may know the data interval associated to each FS entry. This interval corresponds to the index of the content of the
corresponding node. This knowledge comes at zero communication cost, in the sense that no new exchanges with neighbors are necessary. Besides, we will show later that the in our proposal the extension of the fix finger message does not increase the overall control traffic volume in practice.

3) Routing
To shorten the query search process in Power Chord, we use the reverse table in addition to the finger table for the query routing. Therefore, when receiving a query, the intermediate node first examines its RevT to find the final destination if it exists: the node compares the searched key to the IDs of each finger selector entry and its corresponding predecessor. For a given FS entry, if the key lies between these two IDs, the query is transmitted directly to this finger selector, as it is the owner of the searched key. Thus, the lookup search is resolved. If not, the node forwards the query to its closest preceding finger or finger selector (using both tables).

4) Maintenance
The fix finger message, used in Chord between the local node and its finger to establish the finger link, can be used in Power Chord by both nodes to additionally establish the reverse finger link. Indeed, to maintain its finger list, node S probes its fingers periodically. So, if node S is alive, its fingers will periodically receive S’s probe message. When S fails or leaves, the fingers will no longer receive S’s messages. Furthermore, each node in the reverse table will be attached a timestamp \( t \) (the moment of insertion). When a node (a finger) receives a probing message from the same finger selector, the timestamp is updated. After a period \( \Delta t \), if the timestamp has not been updated, the corresponding entry is deleted from the RevT. Therefore, to maintain the new added table, Power Chord nodes do not require new maintenance messages.

B. PowerPastry topology construction
In order to construct a power-law Pastry graph, we use the same neighbor selection, routing and maintenance methods detailed above but adapted to Pastry DHT.

Pastry [7] nodes and objects have an identifier with a sequence of digits in base \( 2^b \) that determines their position on the tree. For the routing strategy, each Pastry node maintains a routing table and a leaf set of sequential neighbors. The leaf set stores the numerically closest neighbors (\( L/2 \) predecessors and \( L/2 \) successors). The routing table is composed of \( \log_2 N \) rows with \( 2^{b-i} \) neighbor entries each. The \( i^{th} \) entry in the routing table of node \( n_i \) maps to a node that shares a common prefix of length \( i \) with node \( n_j \). To maintain its routing table, a
Pastry node periodically contacts a random entry from each rows \( r_i \) and sends it a row request message [7]. The contacted node responds by sending its row \( r_i \). When receiving the corresponding row, the node tries to merge in its routing table the nodes with better proximity [7]. When a node receives a request for an object, it forwards the request to the node in its routing table or leaf set whose ID shares the longest common prefix with the object.

1) Neighbor selection in PowerPastry

To construct a power-law topology with Pastry, similarly to Chord, when receiving a row request message, a PowerPastry node forwards the message to the leaf \( i \) with probability \( p_i \) proportional to the leaf degree. The chosen leaf responds then by sending the requested row. When receiving the row, the node merges its routing table with the received data.

2) Additional data structure

PowerPastry also maintains a reverse table that stores the set of reverse links. In addition to the neighbor selector ID information, the reverse table stores the smallest and the biggest leaf IDs of the corresponding neighbor selector.

3) Routing

PowerPastry uses its reverse table in addition to the routing table for routing. When a node receives a query, it first checks its leaf set to find the final destination if it exists. If not, the node examines its RevT: the node compares the searched key to the IDs of each smallest and biggest leaf corresponding to a RevT entry. For a given RevT entry, if the key lies between these two IDs, the query is transmitted directly to this neighbor selector, as the key lies in its leaf set. Thus, the lookup search is resolved. Otherwise, the node forwards the query to its neighbor or neighbor selector that share the longest prefix with the searched key.

4) Maintenance

The maintenance process is done as in PowerChord. PowerPastry uses the same Pastry row request message to maintain the reverse link.

C. Flooding search in PowerDHT

We propose to perform flooding in the resulting scale free DHT topology. In search by flooding, the source node sends the query to all its neighbors in the routing table and reverse table (e.g. the fingers and the finger selectors in PowerChord). If the neighbors do not have the requested item, they send on to their respective neighbors excluding the source node. This process is repeated TTL times, with TTL fixed to \( \log N/2 \), the average path length of the DHT. In this structured broadcast, complex queries can be used since searches are not anymore limited to simple key lookups. PowerDHT overlays can thus provide broadcast functionality, a missing feature in DHTs.

D. Conclusion

PowerChord and PowerPastry add bidirectional links to enlarge the neighborhood knowledge especially for topologically important nodes. The resulting reverse table helps to perform the search in the entire neighborhood of the intermediate node. By integrating the available local neighborhood knowledge, but without changing routing algorithm, we increase the routing efficiency. In fact, PowerChord and PowerPastry principally increase the usage of the highly connected nodes for routing. Since the routing tables include the most connected nodes, forwarded messages are mainly transmitted by these nodes. At each intermediate node, the search region is broader and consequently routes are even shorter. Moreover, this mechanism does not require nodes to keep any additional information since nodes only use their sequential neighbors.

IV. SIMULATION RESULTS

To evaluate our proposals, we have performed numerous experiments based on simulation. We run simulations using Oversim, a flexible overlay network simulation framework based on OMNeT++ [21]. We compare the performance of our proposed search models named PowerChord and PowerPastry to the classic Chord and Pastry DHTs. We use the following list of criteria for our evaluation.

- Request success rate: The total fraction of requests for which the source node receives an answer.
- Total network load: the total number of messages sent per second in the network.
- KBR route length: successful KBR look up hop count.
- Flood path length: number of hops for flooding search.
- Next hop location: the frequency of use of routing table, reverse table and sequential neighbor to find the next hop.

We measured the request success rate as a function of node lifetime. We simulated a network of \( 2^{13} \) nodes and varied the mean lifetime. When a node is created, its lifetime will be drawn randomly from the Poisson probability distribution. When this time is reached, the node is removed. This metric reflects the protocol’s ability to deliver packets and to resolve multiple-hop queries in face of churn, i.e. the protocol’s resilience to failures. Figure 3 (a,b) plots the request success rate as a function of node’s mean lifetime. We note that PowerChord and PowerPastry outperform Chord and Pastry respectively. PowerPastry achieves the best result with a success ration over 95% for all churn values. For short lifetime values, PowerChord success rate decreases but still remains better than Chord. PowerChord and PowerPastry feature over 85% success for all lifetime values greater than 5000 sec, while for Chord and Pastry, the success ratio degrades to less than 70% and 75% respectively. Since in PowerChord and PowerPastry, a node has a larger routing table and so a larger neighborhood, there are more alternative next hop nodes and consequently node failure affects less the routing process.

We measured the total network load during simulation time. From Figure 3 (c, d), we note that for all algorithms, the number of sent messages is increasing with the network size. For larger networks, however, the network load generated by PowerChord and PowerPastry is considerably smaller. Since the latter produce shorter routes, the number of forwarded queries is smaller, and consequently, the total load is reduced. Note that compared to this reduction, the slight extension of the signaling messages does not fall into consideration.
FIG. 3 SIMULATION RESULTS
We also measured the path length as a function of network size $N$. We simulated a network with $N = 2^k$ nodes and varied $k$ from 9 to 14. During every experiment, each node picked up a random set of keys to look up using Chord, PowerChord, Pastry and PowerPastry algorithms respectively. Figure 3 (e, f) plots the average query path length for all algorithms.

We note that PowerDHT versions outperform standard DHTs. The enhancement is more important for Pastry DHT. In fact, for the largest network, PowerPastry and PowerChord achieve 23% and 20% less hops respectively. The fact that the query traverses essentially highly connected nodes expands the search region to considerably more than $\log N$ neighbors, and accelerate the query process accordingly.

We also measured the path length when performing flooding (FL) instead of KBR routing. Due to lack of space we only present results for Chord and PowerChord but Pastry behaves similarly. From Figure 3 (g), we notice that the route length in P-Chord is env. $\log \log N$ like in scale free graphs and is significantly shorter than flooding path in Chord.

We finally measured the frequency of use of the new added table for PowerPastry algorithm. At each intermediate node, if the destination is found, the received query it sent to the destination next hop found in the leaf set or the reverse table. Otherwise, the query is forwarded to the closest node found in the routing table or the reverse table. For each table, we calculate the number of times the next hop or destination is found in it. From Figure 3 (h), we note that the reverse table is used at least 30% of the time to find either the next hop or the final destination. Moreover, when the network size increases, the frequency of use of the RevT to find the destination node increases, while it decreases for the leaf set. Globally, the usage frequency for RevT increases with the network size.

Finally, one may argue that the comparison of PowerDHT versions to the original proposals is not conclusive. Indeed, a lot of improvements to the standard DHTs have been proposed since their introduction. However, our proposal does not feature any additional signaling messages or complex global changes. The change comes at no communication cost within the standard DHT. In essence, we only use the knowledge already available in the standard DHTs and perform some local rewiring. Therefore, the comparison seems reasonable.

V. CONCLUSION

In this paper, we presented PowerDHT, a distributed algorithm that creates a scale free topology on top of an existing DHT. To the knowledge of the authors, PowerDHT is the first contribution that reunits structured P2P networks and scale-free topologies. By rewiring the overlay connections, but without changing DHT’s routing algorithm per se, our distributed algorithm finds topologically important nodes and assigns these nodes with considerably more routing tasks.

Simulation results show at the examples of two distinct DHTs that the proposed changes result in a more efficient search and achieve improvements over classic DHTs. PowerDHT versions achieve the shortest routes and the lowest network load at a reduced cumulative maintenance/query routing cost in a practical network setup with reasonable query frequencies. We therefore empower the classic DHTs. Moreover, as a super peer network, PowerDHTs effectively support search methods other than exact match queries. In fact, when flooding queries, the average path length is shortened to $\log \log N$.

The resulting power-law structure amplifies the non-uniformity of node usage and may be inappropriate for highly homogeneous networks composed of weak nodes exclusively. In this case, fairness-oriented approaches such as [12] seem more appropriate. Yet, given the node heterogeneity of real-world networks and the popularity of super-peer systems, it is interesting to use the more resourceful nodes more extensively. Besides, it remains possible for a node to reject assignment. The incentive for such an assignment is better connectivity, better network knowledge, and faster search. This issue is not treated within this paper. An additional proposal/agreement protocol is subject of a future work.

REFERENCES