Coordinate-based Routing: Refining NodeIds in Structured Peer-to-Peer Systems

Fabian Hartmann
BrandMaker GmbH
Haid–und–Neu–Straße 7
D–76131 Karlsruhe
Germany
Email: fabian.hartmann@brandmaker.com

Bernhard Heep
Institute of Telematics
Universität Karlsruhe (TH)
Zirkel 2, D–76128 Karlsruhe
Germany
Email: heep@tm.uka.de

***Abstract*—Structured peer-to-peer systems—also known as key-based routing protocols—offer a base for a broad range of applications. In the past, different topology adaptation mechanisms for minimizing key-based routing latencies were proposed and deployed with several today's state-of-the-art protocols. In this paper we introduce coordinate-based routing (CBR), a topology adaptation method, that utilizes landmark-based network coordinate systems and a global a-priori knowledge of node distribution to preserve the uniform distribution of node identifiers. With CBR, a notable decrease of routing latencies in prefix-based KBR-protocols can be achieved, even in combination with other topology adaptation mechanisms enabled. Additionally, CBR allows for a location-based replication strategy in distributed storage applications, which supports the lookup of closer replicas with respect to proximity. Simulation results show a significant decrease of KBR routing latencies and twice as fast *get()*-operations in DHTs.**

***Index Terms*—CBR, peer-to-peer, DHT, network coordinate systems, topology adaptation;**

I. Introduction

Peer-to-peer (P2P) applications have gained major importance during the last years, since they make it possible to provide network services in a distributed instead of a centralized, server-based way. Today's popular and widely deployed Internet applications like BitTorrent and Skype are based on P2P systems. As opposed to other P2P networks types, structured P2P overlays are completely decentralized and self-organized. They are robust and scale with growing node numbers. Data load is shared equally by the participating nodes with well-defined responsibilities. When any node searches for data, the search request is routed to the node that is responsible for holding this data. This is accomplished by each node having a unique identifier, the *nodeId*.

In structured overlays, nodeIds are usually uniformly distributed to provide the equal sharing of data load between all nodes. There is no relationship between a node's nodeId and its position in the underlay topology. Thus, an overlay message which is routed by nodeIds follows a completely random route in the underlay instead of a target-oriented route. This randomness leads to high average latencies for overlay routing. Some structured overlay networks as Pastry and Bamboo bring along topology adaption mechanisms that mainly optimize the nodes' routing tables. This is done by putting closer nodes into the routing tables regarding a proximity metric. As an alternative approach, the manipulation of nodeIds has not been researched thoroughly yet. This way, the routing paths can be optimized without changing the overlay routing protocol. However, this bears the risk that the data load is not shared equally anymore between all participating nodes. This has still to be made sure in order to keep the network scalable. *Network coordinate systems* can provide a basis for the choice of nodeIds, which enables a relationship between the underlay and the overlay structures. This makes a target-oriented routing feasible.

The contribution of this paper is as follows: We utilize network coordinate systems to manipulate nodeIds in KBR protocols without loosing their uniform distribution. This can be achieved by applying a global a-priori knowledge of coordinate spreading. We name this approach *Coordinate-based Routing (CBR)*. Additionally, we propose a location-based replication strategy that directly exploits the manipulated nodeIds to achieve low latencies in distributed storage applications.

The rest of this paper is organized as follows: Section II provides background information for the detailed understanding of CBR, which is described in detail in section III. Implementation details are discussed in section IV and simulation results are evaluated in section V. The paper gives a future outlook in section VI and is concluded in section VII.

II. Background

A requirement for the deployment of CBR is the usage of a recursive, prefix-based key-based routing (KBR) protocol[1] like Pastry[2] [3], Bamboo[4], or Kademlia[5]. Additionally, a landmark-based network coordinate system must be available. These two requirements are explained in this section. An introduction to topology adaption mechanisms is also given, so that CBR can be classified.

A. Prefix-based Key-based Routing

A structured overlay is prefix-based, if it builds up a hypercube overlay structure and uses a routing protocol which works as follows: In each routing step i, the corresponding message is sent from node A_i to B_i. B_i's nodeId shares a prefix with the message's key. This prefix is at least as long as the prefix which B_{i-1}'s nodeId shares with the message's key, thus hops in the ID space get exponentially smaller. When using recursive routing, a message is sent from hop to hop, i.e. $A_i = B_{i-1}$. Iterative routing means that the original node A_1 receives the next hop as intermediate answers and sends the message to the received node in the following iteration, i.e. $A_1 = A_2 = \ldots$ and $A_i \neq B_j$.

Regardless whether recursive or iterative routing is used, in prefix-based overlays the length of the shared prefix grows with every hop. The nodeId and the destination key are not necessarily compared bitwise, but by *digits*, which are bit sequences with a fix length (usually 1, 2, or 4). The digit length has a direct influence on the routing: The longer a digit is, the less hops are needed to reach the destination node, because more nodeId bits are covered by one hop. However, this increases the probability that no known node has this needed bit sequence on the needed position in its nodeId. A shorter digit length provides a greater pool of nodes for the next hop, but it takes probably more hops until the destination node is reached.

B. Network Coordinate Systems

Network coordinate systems offer a way to estimate network latencies between nodes e.g in an overlay network. As an only precondition, the considered nodes' synthetic coordinates have to be known. These coordinates usually are points in an n-dimensional Euclidean space. The estimation is done by calculating the distance between the given coordinates.

Landmark-based coordinate systems like GNP[6] and NPS[7] rely on particular nodes that span the Euclidean space. These landmarks form the base of the coordinate system by providing the reference coordinates. In a initialization phase, all landmarks pairwise probe each other for measuring the round-trip time (RTT). Then, the landmarks' coordinates are calculated with the goal to find a set of coordinates such that the overall error between measured distances and estimation is minimized. In a second phase, all ordinary nodes probe $d+1$ landmarks to get those RTTs and d-dimensional coordinates. With this data, the nodes' own d-dimensional coordinates are calculated, again by minimizing the sum of errors of distances to the landmarks and measured latencies. While GNP completely relies on landmark nodes, NPS defines different layers of landmarks. Ordinary nodes, that have already calculated their coordinates can also play the role of a landmark node, hence this relieves the landmark nodes leading to a better scalability. A node's *NPS layer* is the maximum layer of its landmarks + 1, where real landmarks are on layer 1. GNP can be considered as NPS with a maximum layer of 1.

Vivaldi[8] and S/Vivaldi[9] take a complete decentralized approach, where landmarks are needless. As these systems do not have a fixed base for calculating network coordinates, they are not suitable for the CBR approach.

C. Topology Adaptation in Structured P2P Overlays

To achieve low latencies in KBR protocols, the overlay has to be aware of the underlying network topology. Without any knowledge, single overlay hops have the average latency of end-to-end routing in the underlay. Therefore, the overlay should adapt the underlaying topology, with the effect, that overlay nodes fill up their overlay routing tables with nodes in their proximity. In [3][10], three kinds of topology adaptation mechanisms are distinguished:

- **Proximity Routing** (PR) is a trade-off between progress in ID-space and routing messages through nearby nodes. For each routing hop, several candidates $(X_1, ..., X_n)$ are available, where the one is chosen, that is close in the physical network as well as close to the destination key in ID-space. To decide which node to choose, the distance of all possible next hops must be determined in advance. So, the main characteristic of proximity routing is *not* to decide due to proximity reasons whether a node is put into the routing tables or not.
- **Proximity Neighbor Selection** (PNS) considers proximity aspects while constructing the routing tables. If a node X is found, that fits in the routing table of A, an existing entry Y is displaced if X is closer than Y to A. If there is space for k entries for this position in the routing tables, the closest k nodes are kept. When routing a message, only progress in ID-space is considered. Similar to proximity routing, the distance of all possible neighbors must be determined before they are put into the routing tables.
- **Topology-based NodeId Assignment** modifies the nodeIds or parts of it to map the overlay's ID-space onto the underlying network topology. This way, progress in ID-space causes routing in the destination node's direction. *Geographic routing* in CAN[11] takes this approach: CAN places nodeIds into zones, depending on their distances to given landmarks. As a major drawback, the number of nodes within two different zones might differ, which results in a non-uniform nodeId distribution. Besides, CAN needs $\mathcal{O}(N)$ routing hops instead of $\mathcal{O}(\log N)$ routing hops needed by Pastry and Bamboo.

The in this paper proposed *Coordinate-based Routing* can be considered as an enhanced version of *Topology-based nodeId Assignment* which maintains uniform nodeId distribution.

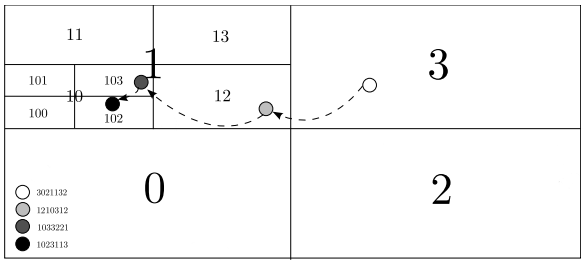


Fig. 1: Routing paths using CBR

III. Coordinate Based Routing

A. Basic idea

The main idea of CBR is to map a node's network coordinates onto a nodeId prefix—comparable to a city's area code.

At first, the underlay topology gets divided into 2^d *main areas*. The main areas themselves get subdivided into *prefix areas*: The finer these subdivisions are made, the smaller the areas and the longer the corresponding prefixes get. This way, a dependency between the nodeId-based overlay structure and the underlay network position is established.

Since the common prefix with the destination node grows with each hop, the first hop leads to the whereabouts of the destination node (in the overlay as well as in the underlay). Next, the following hops get smaller, leading target-oriented to the destination as illustrated in Fig. 1.

It is an essential task of CBR to find the correct mapping depending on the number and positions of the participating

nodes in the network. A node needs to complete this mapping task before it joins the network with the location-prefixed nodeId. It is critical here that all prefixes are given in the same frequency as if a uniform distribution was used i.e when using random nodeIds. This prerequisite must take into account that not all nodes are spread equally within the underlay network, but there are congested and sparsely populated areas. For this reason, the prefixes need to be mapped onto areas of different sizes, depending on the number of participating nodes in each area.

B. Calculation of Network Coordinates

Before a node can create its nodeId and thus can participate in the network, it must gain knowledge of its underlay position. This is achieved by using a landmark-based network coordinate system like GNP and NPS. An essential foundation for a distinct coordinate-to-prefix mapping is a fix coordinate basis. This basis is spanned by at least $d + 1$ reference points that already calculated their own coordinates among themselves. This is exactly what is done in a landmark-based network coordinate system.

The initial definition of the coordinate basis is so important, because latency measurements are the only input data for the coordinate calculation. In GNP and NPS this calculation is done by the simplex downhill method, which solves a global minimization problem by using all pair-wise latency measurements between the landmarks and random start coordinates. The latencies between the landmarks define only their relative distances to each other. Depending on the start coordinates, the resulting patterns after two different runs of the simplex downhill method may be shifted, stretched, rotated or mirrored. As a result, the positions of the joining nodes also differ.

C. Assignment of Network Coordinates to NodeIds

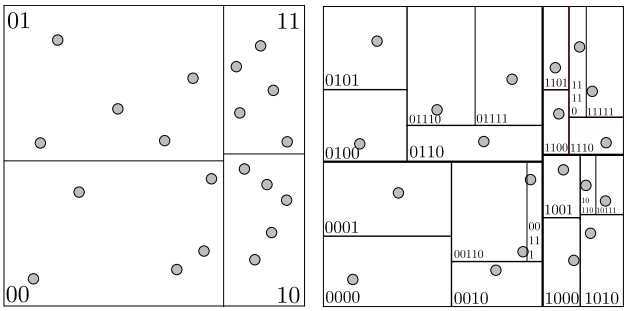
Now that a node can gain its network coordinates, the next step is the determination of its nodeId prefix. So, a function f is needed, which maps the d-dimensional coordinate tuple to a single scalar value:

$$f : \mathbb{R}^d \rightarrow P \quad \text{with } P = \{p_i \in \mathbb{N}_0 \mid p_i < 2^i, 1 \leq i \leq max\}$$

The number of prefix bits i on which the coordinates get mapped may differ, where max is the largest number of bits possible. This mapping function must be known to all nodes.

As mentioned above, instead of a precise mathematical function, CBR uses a partitioning of the underlay into prefix areas. Here, each area has a lower and an upper bound in each dimension. For a two-dimensional coordinate space, the areas are rectangles, in a three-dimensional space they are cuboids. The same idea applies to higher dimensions.

A compromise has to be reached of how many bits or digits should be manipulated by CBR, i.e. how long the nodeId's prefix and suffix should be. A too short prefix leads to too less target-oriented hops, thus the effect of CBR on routing latencies is low. Here, the prefix' size directly depends on the total number of nodes in the network and the applied protocol. The suffix should be long enough to prevent nodeId collisions in each area. Therefore, simulations with varying prefix sizes are performed in Sect. V.



(a) First recursion (b) Full recursion

Fig. 2: Partitioning

We will now describe how the prefix region partitioning is created, distributed to the nodes and handled by these. A very important aspect which must be regarded during this creation is the load balancing.

1) Load Balancing: Because of the non-uniform distribution of nodes in the d-dimensional topology, the area sizes must be chosen depending on the node density: The sparser a topology region is populated, the larger the prefix area must be. This may sound like a contradiction at first, but it is not the actual size of an area which is relevant, but the length of its prefix and the number of nodes it contains. This allows a uniform distribution of the nodeIds in a non-uniformly populated space. This is shown in figure Fig. 2(a) where the division of a two-dimensional space is shown after one step in each dimension. It is essential not to draw the borders for each dimension simultaneously, which may result in differently populated prefix areas – instead each dimension must be divided after another. In order to determine the right position of the border, a well-known global distribution of nodes (*Global Knowledge*) is needed and then the border is drawn in the middle of the these nodes, biparting them into two halves. This procedure can be repeated until there is only one more node left in each prefix area, as shown in figure Fig. 2(b).

2) Gaining Global Knowledge: As described in the last paragraph, the so-called Global Knowledge is essential to create load-balanced prefix areas. This Global Knowledge is basically a representative overview on the topology of participating nodes: The borders in Fig. 2 are drawn relative to the nodes' network coordinates, which got calculated before. It is assumed that the used nodes' positions are representative for the whole network using these borders.

Transferring this simple example to a real situation, a phase of data harvesting is required before CBR can be used. As in the example, the network deployers need a representative idea about the topology of the nodes that are going participate. This topology may scale from a relatively small region (like a federal state) to the whole world. The intended use of the network and its ranges must be at least roughly known before deployment. During the data harvesting phase, a picture of the underlying topology is created. Network coordinates of as many participating nodes as possible get calculated. This is accomplished by latency monitoring nodes, preferably the landmarks used for the GNP/NPS system that are needed for CBR. A representative set of nodes might be an overlay, which

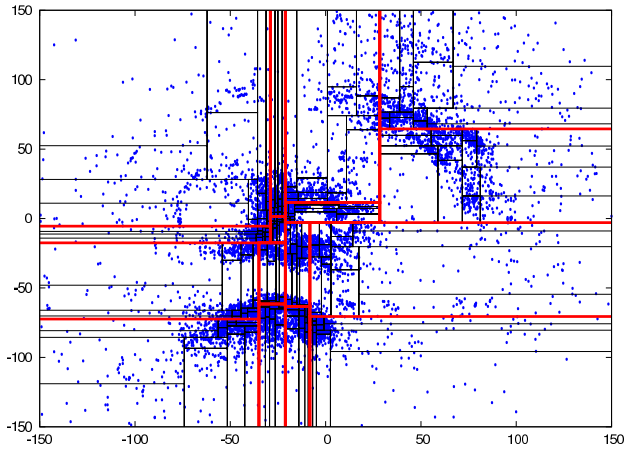


Fig. 3: Global Knowledge

is already established and should be improved by using CBR.

As soon as sufficient latency data throughout the topology got collected, the Global Knowledge can be created by performing network coordinates calculations in the wanted dimensionality, drawing the borders accordingly and attaching prefixes to the resulting regions. The overlay network deployer is responsible for providing the network coordinate system as well as for the distribution of the calculated Global Knowledge. As long as its coordinate basis does not change, new nodes can look up their correct nodeId prefix in the distributed Global Knowledge using their network coordinates.

An example for a Global Knowledge for the deployment of a world-wide overlay network is shown in Fig. 3, where Skitter measurements[12] were utilized as latency data. Skitter data was also used for the simulation in Sect. V to achieve realistic latencies in the evaluation.

D. Reallocation of Replicas in Distributed Storage

In structured overlays, fail-safe redundancy is usually realized by replicating data on nodes that have neighboring nodeIds to the responsible node. When using CBR, this means that all replicas would gather in a single underlay region. A network outage in this region would lead to a loss of all replicas, also this geographic aggregation of specific data may be unwanted for political reasons. For these reasons, CBR must also provide a new replication mechanism which leads to wide-spread replicas all over the network.

A key-value pair sent to n nodes, that are responsible for one of the keys $K_n \in C$, with the given set of hashes $C = \{K|K = H^i(V), 0 < i \leq m\}$ using a given hash function H and a value V. m is also called the number of *spreaded replicas*. The destination key for *get()*-messages is chosen out of all these n candidates K_n. Here, a single *get()*-message is sent to the closest key K_n, optionally with respect to the proximity information taken from the global knowledge. Fig. 4 illustrates this strategy: A node placed in the area with the prefix "9" choses K_n as destination key placed in area "B", as its center is closest to the initiator node. If the request failed—and no ordinary replicas are available—the next closest key in area "8" is tried as destination key.

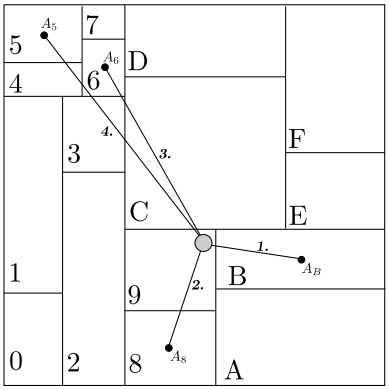


Fig. 4: Replication strategy based on CBR

E. Step-by-step: Preparation of CBR

As a concise summery, the following 4 steps have to be taken to provide CBR for structured P2P systems:

1) **Landmark Initialization**: Setting up landmark nodes, calculation of landmark coordinates as fixed basis
2) **Data Harvesting Phase**: Gathering coordinate data to get a representative picture of participating nodes
3) **Creation of Global Knowledge**: Partitioning of collected data to provide global knowledge
4) **Utilization**: Distribution of global Knowledge, building up prefix-based KBR overlay with CBR-based nodeIds

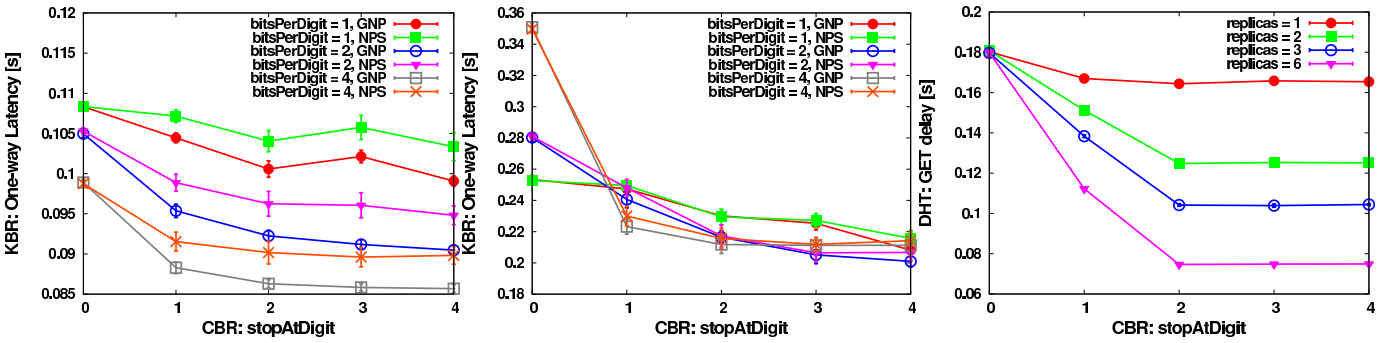
IV. Implementation Details

CBR is implemented using the *OverSim* framework[13][14]. So, the implementation can make use of the KBR protocols already provided there. As CBR does not change the protocols' behavior, but only manipulates the nodeIds of nodes before they join the overlay, just marginal changes have to be done in the overlay base classes. NPS/GNP is integrated as a submodule of the *Neighbor Cache* all nodes in OverSim maintain. The global knowledge is provides as a *Global Function* in OverSim. The partitioning and mapping of collected coordinates onto nodeId prefixes is implemented as a Ruby script, that is run in advance before the simulations start.

V. Evaluation

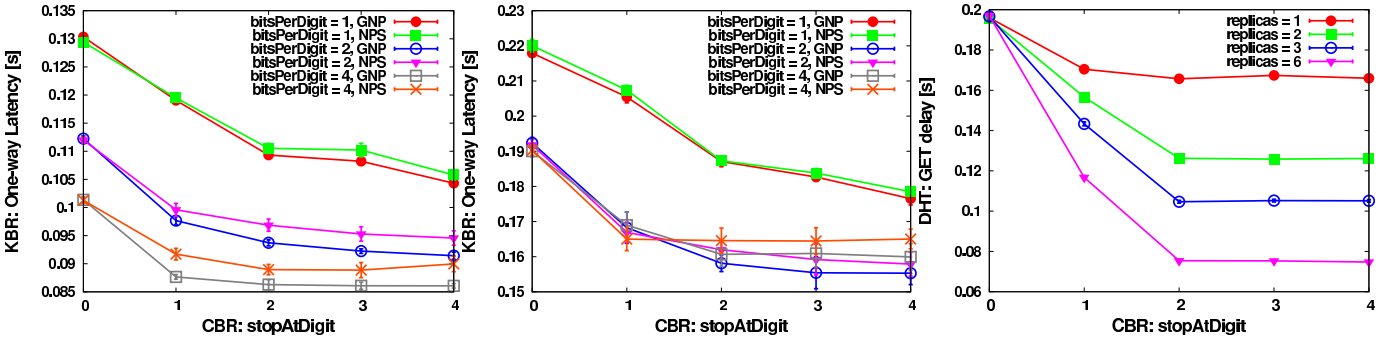
A. Underlay Abstraction

For the evaluation of CBR, OverSim's *Simple* underlay model is used in this paper. Here, data packets are sent directly from one overlay node to another by using a global routing table. Packets between overlay nodes get delayed by a delay calculated from the nodes' distance. For this, each node is placed into a two-dimensional Euclidean space. The coordinates used for the following simulations were calculated from the Skitter dataset[12] as a representative selection of Internet nodes. This leads to a reasonable model of node population. In addition, the node is assigned to a logical access network characterized by inbound and outbound bandwidth, access delay and packet loss, so that heterogeneous access networks and queuing delays can be simulated. Due to the low simulation overhead of these techniques, this model leads to a high level of accuracy and the ability to simulate networks with a large number of nodes.



(a) KBR, No churn, 4,500 nodes (b) KBR, Churn, 2,500 nodes (c) DHT, no churn, 2,500 nodes

Fig. 5: Simulation results of Pastry



(a) KBR, No churn, 4,500 nodes (b) KBR, Churn, 2,500 nodes (c) DHT, no churn, 2,500 nodes

Fig. 6: Simulation results of Bamboo

To assure the invalidity of the triangle inequality—as it is violated by Internet latencies—all simulated packet delays are forged with a synthetic error, based on the results in [9], where the error is constant between pairs of nodes. This way, network coordinate systems simulated by using this underlay have similar accuracy as they would have in real networks like the Internet.

B. Simulation Parameters

For the evaluation of CBR, simulation runs with 4,500 overlay nodes are performed. Churn and DHT scenarios comprise of 2,500 nodes. All nodes have a 10 MBit Ethernet network access. As KBR protocols, Pastry and Bamboo using semi-recursive routing mode are chosen, where Pastry uses a leaf set of 16 nodes, while Bamboo only keeps 8 nodes in his leafs set. Bamboo's *local tuning* is performed every 10s, *global tuning* every 20s, and *leaf set maintenance* every 4s. For both protocols, per-hop acknowledgements are activated.

In the simulation runs illustrated below, the following parameters are varied using the here listed values:

- Bits per digit {1, 2, 4}
- Churn {no churn, moderate churn}
- Network Coordinate System {GNP, NPS}
- CBR stop digit {noCBR, 1, 2, 3, 4}
- DHT replicas {1, 2, 3, 6}

All parameter combinations are evaluated by 20 runs with different random seeds. Churn is modeled as identified in [15][16]. GNP and NPS use 2 dimensions, the maximum layer for NPS is set to 3.

The simulated underlay latencies are based on the Skitter data mentioned in Sect. V-A with information about 15,000 nodes and corresponding landmarks. The global knowledge is directly calculated from this information, thus replaces the initialization phase for the following simulations.

C. Impact on KBR latencies with CBR

At first, CBR is evaluated in no churn scenarios with 4,500 nodes. Here, all nodes send probe messages periodically (every 60 seconds) to random nodeIds. For all results presented here, it has to be considered that due to the use of PNS, Pastry and Bamboo achieve low routing latencies already without CBR.

Fig. 5(a) shows the results with Pastry. Here, CBR decreases routing latencies by up to 10% with $bitsPerDigit = 1$. The usage of NPS has a significant influence on the decrease, as it leads to faulty coordinates and therefore, not the optimal routing path is taken. With $bitsPerDigit = 2$ a decrease of 13% can be achieved. Again, NPS comes with higher latencies: A decrease of only 9% can be observed. Using $bitsPerDigit = 4$ leads to similar results, a lowering of 13% is possible. Except for $bitsPerDigit = 1$—where a small increase of latencies is observable between 2 an 3 manipulated digits—all latencies are steadily falling with an increasing number of manipulated digits in the nodeId's prefix.

Fig. 6(a) shows Bamboo's results with CBR activated and no churn: Using $bitsPerDigit = 1$, for GNP and NPS similar

latencies are measured, decreasing up to 20% with increasing stop digit. A reduction of 18% is feasible when using 2 $bitsPerDigit$. Like in Fig. 5(a), a significant difference between GNP and NPS is noticeable. For $bitsPerDigit = 4$ Bamboo achieves similar results to Pastry. Latencies go down up to 15% depending on the used coordinate system.

Fig. 5(b) and Fig. 6(b) show Pastry's and Bamboo's results under churn. The KBR delivery ratio is not affected by using CBR and is nearly at 100% for both protocols. Up to $stopAtDigit = 2$, an immense decrease of routing latencies can be recognized. Pastry's one-way latencies go down from 350ms to 220ms with $bitsPerDigit = 4$, which equates a decrease of about 37%. Bamboo is not as unstable under churn, thus latencies are low overall. With CBR they lower by up to 16%. Higher values than 2 for the prefix' size do not have any remarkable effect.

D. Proximity aware choice of DHT replicas

For the evaluation of the CBR-based replication strategy in DHTs, a simple DHT module is deployed using Pastry's and Bamboo's KBR interface. A DHT test application periodically sends *put()* and *get()*-requests (RPCs), where the corresponding key-value pairs (K, V) are additionally stored in or retrieved from a global map, respectively. Put messages—delivering the key-value pair—are sent directly to the responsible nodes after performing KBR lookups on the destination keys K_n with $0 < n \leq replicas$. *get()*-messages are routed directly in a recursive manner to keep delivery latencies small.

Fig. 5(c) and Fig. 6(c) show the results for DHT *get()*-delays in a 2,500 nodes scenario without churn. Here, 4 $bitsPerDigit$ and GNP are used. Responses to *get()*-requests are significantly faster when using a higher number of *spreaded replicas*. A decrease of latencies of up to 61% when using 6 replicas can be observed. A higher value than 2 for $stopAtDigit$ does not lead to lower latencies anymore.

VI. Future Work

While CBR already achieves satisfying results, we plan to utilize other partitioning schemas or clustering algorithms for the determination of nodeIds. Also, other KBR protocols like Kademlia are interesting to use with CBR. More exhaustive simulations have to be done for CBR-replication in DHTs, combined with standard replication strategies for better reliability under churn. Another important issue is the validation of the generated simulation results using other well-established simulation models for the underlaying network as well as the deployment of CBR in real-world networks or testbeds like PlanetLab[17] or G-Lab[18].

VII. Conclusion

In this paper we presented Coordinate-based Routing (CBR), a method of manipulating nodeIds in structured peer-to-peer systems to achieve low routing latencies. CBR relies on landmark-based network coordinate systems and works with prefix-based KBR-protocols like Pastry and Bamboo.

With CBR, routing latencies can significantly be decreased, while preserving the uniform distribution of nodeIds. Additionally, we proposed a CBR-based replication strategy in DHTs, which leads to an immense speed-up of *get()*-operations, even though sending a single request. Unlike most other topology adaptation mechanisms, CBR does not change the behavior of the protocols. Instead, it solves the problem of high key-based routing latencies by manipulating nodeIds while still maintaining load balance among the nodes, which was a major drawback of earlier mechanisms targeting the nodeIds. It is a fresh approach to topology-based nodeId assignment which can be combined with well-established topology adaptation mechanisms like PNS.

For large-scale applications with a fixed landmark set available, CBR can be an alternative to meet low latency demands. A decentralized *Domain Name System* (DNS), other directory services, or gaming overlays based on CBR are imaginable.

References

[1] F. Dabek, B. Zhao, P. Druschel, J. Kubiatowicz, and I. Stoica, "Towards a Common API for Structured Peer-to-Peer Overlays," in *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03)*, vol. 2735/2003, 2003, pp. 33–44.

[2] A. Rowstron and P. Druschel, "Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems," in *Middleware 2001 : IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg, Germany, November 12-16, 2001. Proceedings*, vol. 2218/2001, Nov 2001, pp. 329+.

[3] M. Castro, P. Druschel, and Y. C. Hu and Antony Rowstron, "Topology-Aware Routing in Structured Peer-to-Peer Overlay Networks," Microsoft Research, Tech. Rep. MSR-TR-2002-82, 2002.

[4] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz, "Handling churn in a DHT," in *ATEC '04: Proceedings of the annual conference on USENIX Annual Technical Conference*. Berkeley, CA, USA: USENIX Association, 2004, pp. 10–10.

[5] P. Maymounkov and D. Mazières, "Kademlia: A Peer-to-Peer Information System Based on the XOR Metric," in *Peer-to-Peer Systems: First International Workshop, IPTPS 2002 Cambridge, MA, USA, March 7-8, 2002. Revised Papers*, vol. 2429/2002, 2002, pp. 53–65.

[6] T. S. Eugene Ng and H. Zhang, "Predicting internet network distance with coordinates-based approaches," in *In INFOCOM*, 2001, pp. 170–179.

[7] T. S. Eugene Ng and H. Zhang, "A network positioning system for the internet," in *ATEC '04: Proceedings of the annual conference on USENIX Annual Technical Conference*. Berkeley, CA, USA: USENIX Association, 2004, pp. 11–11.

[8] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, "Vivaldi: a decentralized network coordinate system," in *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM Press, 2004, pp. 15–26.

[9] J. Ledlie, P. Gardner, and M. Seltzer, "Network coordinates in the wild," in *In Proceeding of USENIX NSDI'07*, 2007.

[10] M. Castro, P. Druschel, Y. C. Hu, and A. Rowstron, "Exploiting network proximity in distributed hash tables," in *International Workshop on Future Directions in Distributed Computing (FuDiCo)*, 2002, pp. 52–55.

[11] S. Ratnasamy, P. Francis, S. Shenker, and M. Handley, "A Scalable Content-Addressable Network," in *In Proceedings of ACM SIGCOMM*, 2001, pp. 161–172.

[12] B. Huffak, D. Plummer, Daniel, D. Moore, and K. Claffy, "Topology discovery by active probing," in *SAINT-W '02: Proceedings of the 2002 Symposium on Applications and the Internet (SAINT) Workshops*. Washington, DC, USA: IEEE Computer Society, 2002, p. 90.

[13] I. Baumgart, B. Heep, and S. Krause, "OverSim: A flexible overlay network simulation framework," in *Proceedings of 10th IEEE Global Internet Symposium (GI '07) in conjunction with IEEE INFOCOM 2007, Anchorage, AK, USA*, May 2007, pp. 79–84.

[14] I. Baumgart, B. Heep, and S. Krause, "A P2PSIP Demonstrator Powered by OverSim," in *Proceedings of 7th IEEE International Conference on Peer-to-Peer Computing (P2P2007), Galway, Ireland*, Sep 2007, pp. 243–244.

[15] D. Stutzbach and R. Rejaie, "Understanding churn in peer-to-peer networks," in *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*. New York, NY, USA: ACM, 2006, pp. 189–202.

[16] M. Steiner, T. En-Najjary, and E. Biersack, "Long Term Study of Peer Behavior in the KAD DHT," *IEEE/ACM Transactions on Networking*, vol. 17, no. 6, 2009.

[17] "Planet-Lab Website." [Online]. Available: http://www.planet-lab.org/

[18] "G-Lab Website." [Online]. Available: http://www.german-lab.de/