9 The Continuous-Discrete Approach

Next we show how to maintain completely decentralized overlay networks, i.e. now we do not have a supervisor any more. We will assume that peers that want to enter the network already know a peer that is in the network. (In practice, peers contact rendezvous servers available over the world-wide web that will introduce the peer to a peer in the network.)

For the decentralized overlay networks presented in this section we will use an approach called the continuous-discrete approach [3].

9.1 The continuous-discrete approach

The basic idea of the continuous-discrete approach is to define a continuous model of graphs and to apply this continuous model to a finite set of nodes.

Consider some d-dimensional space $U = [0,1)^d$, and suppose that we have a set F of functions $f_i : U \to U$. Then we define E_F as the set of all pairs $(x,y) \in U^2$ with $y = f_i(x)$ for some i. Given any subset $S \subseteq U$, we define

$$\Gamma(S) = \{y \in U \setminus S \mid \exists x \in S : \ (x,y) \in E_F\} \ .$$

The *expansion* α of F is defined as

$$\alpha = \min_{S \subseteq U,\, |S| \leq |U|/2} \frac{|\Gamma(S)|}{|S|} \ .$$

where $|S|$ denotes the volume of a set S. F is called *mixing* if $\alpha > 0$ and *rapidly mixing* if α is a constant. If F does not mix, then there are disconnected areas in U. As a prerequisite for our constructions we require that it is possible to go from any area to any other area in U, and we will therefore assume in the following that F is mixing. Under this assumption, the continuous setting can be transformed into a discrete, connected graph as follows:

Invariant 9.1 *Suppose that we have a set V of n nodes, and suppose that a region $R(v) \subseteq U$ has been assigned to each node $v \in V$ so that the regions cover the entire space U, i.e. $\cup_v R(v) = U$. Then the graph $G_F(V)$ contains an edge (v,w) for every pair of nodes v and w for which there is an edge $(x,y) \in E_F$ with $x \in R(v)$ and $y \in R(w)$.*

The following fact is easy to see.

Fact 9.2 *If F is mixing and $\cup_v R(v) = U$, then $G_F(U)$ is connected.*

Thus, for the invariant to be useful, we need

- a family of functions F on some space U that is mixing, and
- a rule for assigning regions to nodes that cover U.

There are two basic strategies for the second part: cutting U into disjoint regions, or choosing overlapping regions for the nodes.

Disjoint regions

Here, two strategies have been proposed. The proximity approach [3] and the hierarchical decomposition approach [4]. In the proximity approach, a hash function $h : V \to U$ is usually used that assigns random values to the nodes. The region of a node v is defined as

$$R(v) = \{x \in U \mid x \text{ is closest to } h(v) \text{ among all nodes}\}$$

These regions form a so-called Voronoi diagram that can look quite complicated. (Only if $d = 1$, i.e. $U = [0, 1)$, then this approach gives reasonably simple regions.) Therefore, the hierarchical decomposition approach is usually preferred.

In the hierarchical decomposition approach, U is decomposed into a hierarchy of subcubes of U using a binary decomposition tree. The root of this tree represents U. For any node u of the tree representing subcube $U(u)$ that has children v and w, $U(u)$ is cut into two equal-size subcubes along a dimension with the largest side length, and these subcubes are assigned to v and w. The goal of the hierarchical decomposition approach is to assign peers to positions in the decomposition tree so that each peer is responsible for the subcube of the node it is assigned to and the union of these subcubes gives U. This is done in the following way.

The first peer that enters the system will occupy the root position, i.e. it is responsible to the entire set U. Each time a new peer p joins the system, it moves down a random path in the tree until it hits a node u occupied by a peer q. Then p is placed in the left child of u and q is placed in the right child of u. If the regions of the peers previously covered U, then this operation makes sure that the regions of the peers also cover U afterwards. If a peer p wants to leave the system, then it checks whether there is a peer q in its sibling. If so, q is moved upwards to the parent of p and p can leave the system. Otherwise, any peer q placed at a node below the sibling of p is taken to take over the position of p, and the removal of q from its old position is handled like the departure of p above.

For the special case of $U = [0, 1)$, a popular approach to cut U into disjoint regions has been the consistent hashing approach, i.e. each node v is responsible for the region $R(v) = (h(\text{pred}(v)), h(v)]$ where $\text{pred}(v)$ is the node preceding v in $[0, 1)$.

Lemma 9.3 *If $U = [0, 1)$, h is a random hash function, and the nodes v use the consistent hashing scheme to select the regions $R(v)$ in U they are responsible for, then $\mathrm{E}[|R(v)|] = 1/n$ and $|R(v)| = O((\log n)/n)$ with high probability.*

Proof. The expectation follows from symmetry. For the upper bound on $R(v)$, notice that for every node v, the probability that v has no node w with $h(w) \in [h(v), h(v) - (c \ln n)/n)$ is equal to

$$\left(1 - \frac{c \ln n}{n}\right)^{n-1} \leq e^{-\frac{c \ln n}{n} \cdot (n-1)} = e^{-c(1-1/n) \ln n} = \left(\frac{1}{n}\right)^{c(1-1/n)} .$$

Hence, with high probability, $|R(v)| = O((\log n)/n)$ for every node v. □

Overlapping regions

To give an example of how to achieve coverage with overlapping regions, let C_ϵ be the set of all d-dimensional hypercubes in $[0, 1)^d$ of volume ϵ. Suppose again that we map the nodes to points in U

via some function $h : V \to U$. Further, suppose that every node v is given a region $R(v)$ representing a hypercube in C_ϵ with center v. Since the regions are supposed to cover U, ϵ must be at least $1/n$. Fortunately, when mapping the nodes uniformly at random to points in U, ϵ will be quite close to n with high probability if d is small:

Lemma 9.4 *If h is a random hash function, then $\epsilon = \Theta((2^d \log n)/n)$ is sufficient, with high probability, to make sure that the hypercubes of volume ϵ with centers in $h(V)$ cover the entire set U.*

Proof. Consider any point $x \in U$ and let c_x be the d-dimensional hypercube of volume ϵ centered at x. If $\epsilon = (c \log n)/n$ for some constant $c > 0$, then the probability that none of the nodes v has a point $h(v)$ in c_x is equal to

$$\left(1 - \frac{c \ln n}{n}\right)^n \leq e^{-\frac{c \ln n}{n} \cdot n} = e^{-c \ln n} = \left(\frac{1}{n}\right)^c .$$

Hence, if every node chooses a hypercube of volume $2^d\epsilon$ around it, then with high probability there is a node whose hypercube fully covers c_x. Now, U can be partitioned into $1/\epsilon$ cubes of volume ϵ, so there are only $1/\epsilon$ points x to check. Hence, the probability that there is any point $y \in U$ that is not covered by the hypercubes of the nodes of volume $2^d\epsilon$ is at most $(n/(c \ln n)) \cdot n^{-c}$ which is very small if $c > 1$. □

General properties

Next we look at general properties of graphs resulting from the continuous-discrete approach. For any subset $W \subseteq V$, let $R(W) = \cup_{v \in V'} R(v)$. We define the *smoothness* σ of the region assignment as

$$\sigma = \max_{W \subseteq V} \frac{|R(W)|}{|W|} .$$

Using this definition, it holds:

Theorem 9.5 *For every set V of n nodes with $\cup_v R(v) = U$ it holds that $G_F(V)$ has a node expansion of $\Omega(\alpha/\sigma)$ and a diameter of at most $O((\sigma/\alpha) \log n)$.*

The proof is complex and omitted here. Next, we will apply the continuous-discrete approach to transform the hypercube and the deBruijn network into a dynamic, decentralized overlay network. A dynamic variant of the hypercube was first proposed in [5], and dynamic variants of the deBruijn graph were independently proposed in [2, 3].

9.2 Maintaining a dynamic hypercube

For any real $x \in [0, 1)$, let $(\ell_1 \ell_2 \ldots)$ be the binary representation of x, i.e. $x = \sum_{i \geq 1} \ell_i / 2^i$. Define $x(b)$ as $x + (1 - 2\ell_b)/2^b$, which is equivalent to flipping ℓ_b in the binary representation of x. The hypercube has the following geometric representation:

Definition 9.6 *The hypercube of dimension d, $HC(d)$, is an undirected graph $G = (V, E)$ with node set $V = \{j/2^d \mid j \in \{0, \ldots, 2^d - 1\}\}$ and edge set E containing all edges $\{x, y\}$ with $y = x(b)$ for some $b \in \{1, \ldots, d\}$.*

Thus, the way we can extend this to an infinite-dimensional hypercube is to use the space $U = [0, 1)$ and define F_H as the class of all functions f_i, $i \geq 1$, with $f_i(x) = x(i)$.

Using this family of functions, the following theorem is implied by results in [1].

Theorem 9.7 *Let $W \subseteq V$ be the set of nodes currently in the system. If the consistent hashing scheme is used to map the nodes to regions in $[0, 1)$, then the graph $G_F(W)$ resulting from the continuous-discrete approach has a node expansion of $\Omega(1/\log n)$ and a smoothness of $O(\log n)$, with high probability. Furthermore, every node has a degree of at most $O(\log^2 n)$ with high probability.*

Routing in the hypercube

Suppose that Invariant 9.1 is satisfied for our family of hypercubic functions. Then it is fairly easy to route a message from any point $x \in [0, 1)$ to any point $y \in [0, 1)$ along edges in $G_F(W)$.

Consider the path P in the continuous space that results from using the bit adjustment strategy in order to get from x to y. That is, given that $x_1x_2x_3\ldots$ is the bit sequence for x and $y_1y_2y_3\ldots$ is the bit sequence for y, P is the sequence of points $z_0 = x_1x_2x_3\ldots, z_1 = y_1x_2x_3\ldots, z_2 = y_1y_2x_3\ldots, \ldots, y_1y_2y_3\ldots = y$. Of course, P may have an infinite length, but simulating P in $G_F(W)$ only requires traversing a finite sequence of edges:

We start with the region $R(v)$ containing $x = z_0$. Then we move along the edge (v, w) in $G_F(W)$ to the region $R(w)$ containing z_1. This edge must exist because we assume that Invariant 9.1 is satisfied. Then we move along the edge (w, w') simulating (z_1, z_2), and so on, until we reach the node whose region contains y. Using this strategy, it holds:

Lemma 9.8 *Given a node set W with $|W| = n$, it takes at most $O(\log n)$ hops, with high probability, to route in $G_F(W)$ from any point $x \in [0, 1)$ to any point $y \in [0, 1)$.*

Joining and leaving the network

Suppose that a new node v contacts some node already in the system to join the system. Then v's request is first sent to $u = \text{succ}(h(v))$, which only takes $O(\log n)$ hops according to Lemma 9.19. u forwards information about all of its incoming and outgoing edges to v, deletes all edges that it does not need any more, and informs the corresponding endpoints about this. Because $R(v) \subseteq R(u)$ for the old $R(u)$, the edges reported to v are a superset of the edges that it needs to establish. v checks which of the edges are relevant for it, informs the other endpoint for each relevant edge, and removes the others.

If a node v wants to leave the network, it simply forwards all of its incoming and outgoing edges to $\text{succ}(h(v))$. $\text{succ}(h(v))$ will then merge these edges with its existing edges and notifies the endpoints of these edges about the changes.

Combining Theorem 9.7 and Lemma 9.19 we obtain:

Theorem 9.9 *It takes a routing effort of $O(\log n)$ hops and an update work of $O(\log^2 n)$ messages that can be processed in a constant number of communication rounds in order to execute a join or leave operation.*

One can also use the dynamic hypercube for distributed data management by implementing the consistent hash strategy on top of it.

An alternative but very similar construction to the dynamic hypercube is known as Chord [5].

9.3 Chord

In the Chord network, the following invariant has to be kept at any time.

Invariant 9.10 *For any set of nodes V currently in the system, it holds for every $v \in V$ that v is connected to*

(a) pred$(h(v))$, succ$(h(v))$, *and*

(b) succ$(h(v) + 1/2^i)$ *for all* $i \geq 1$ *with* succ$(h(v) + 1/2^i) \neq$ succ$(h(v))$.

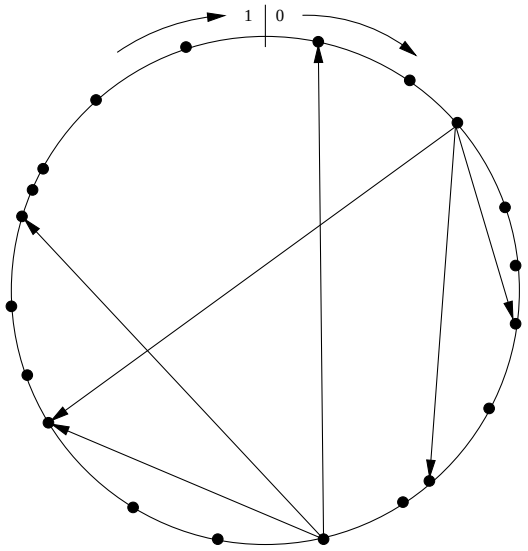

Figure 1: An example of a Chord network (only short-cut pointers for two nodes are given).

Invariant 9.10(a) requires that the nodes are organized in a sorted, doubly linked cycle, the so-called *Chord ring*, and Invariant 9.10(b) makes sure that messages can move quickly from any node to any other node on the cycle. The first type of edges are called *ring edges* and the second type of edges are called *hypercubic edges*, due to their similarity with the edges in Definition 9.6.

Similar to Theorem 9.7, the Chord overlay network has the following properties, where n is the current number of nodes in the system:

Theorem 9.11 *If Invariant 9.10 is true and h assigns random numbers to nodes, then Chord has a maximum outdegree of $O(\log n)$, a maximum indegree of $O(\log^2 n)$, a diameter of $O(\log n)$, and a node expansion of $\Omega(1/\log n)$ with high probability.*

Proof. We start with the maximum outdegree. Certainly, every node only has two ring edges. Thus, it remains to bound the number of hypercubic edges. Consider any fixed node $v \in V$ and let $I_v = [h(v), h(v)+1/n^3]$. For any node $w \in V \setminus \{v\}$, the probability that $h(w) \in I_v$ is equal to $1/n^3$. Hence, the expected number of nodes in $w \in V \setminus \{v\}$ with $h(w) \in I_v$ is at most $(1/n^3) \cdot (n-1) \leq 1/n^2$. Using this in the standard Markov Inequality (i.e. $\Pr[X \geq k] \leq \mathrm{E}[X]/k$) it follows that the probability of having at least one node $w \in V \setminus \{v\}$ with $w \in I_v$ is at most $1/n^2$. Hence, the expected number of

nodes $v \in V$ that have at least one other node in I_v is at most $1/n$, and therefore, again from the Markov Inequality, the probability that there is at least one node $v \in V$ with another node in I_v is at most $1/n$. Thus, with high probability it holds that for all nodes $v \in V$, $\text{succ}(h(v) + 1/2^{3 \log n}) = \text{succ}(h(v))$. Hence, with high probability every node only has $O(\log n)$ hypercubic pointers.

Next we consider the indegree. We only sketch the proof. For any node $v \in V$, let $I'_v = [h(v) - (\epsilon \ln n)/n, h(v)]$ for some constant $\epsilon \in (0, 1)$. The probability that for some fixed node v there is no other node w with $w \in I'_v$ is equal to

$$(1 - |I'_v|)^{n-1} = \left(1 - \frac{\epsilon \ln n}{n}\right)^{n-1} \geq e^{-(n-1) \cdot \frac{\epsilon \ln n}{n - \epsilon \log n}} \geq e^{-2\epsilon \ln n} = n^{-2\epsilon}$$

because it is known that $(1 - 1/k)^{k-1} \geq 1/e$ for all $k > 1$. Hence, the expected number of nodes v with no other node in I'_v is at least $n \cdot n^{-2\epsilon} = n^{1-2\epsilon}$. Using this, one can also show that with high probability, there will be at least one node v with no other node in I'_v, if $\epsilon \leq 1/3$. For such a node v, all nodes in the intervals $[h(v) - (\epsilon \log n)/n - 1/2^i, h(v) - 1/2^i)$ would have to have a hypercubic edge to v, and there are on expectation, and with high probability, $\Theta(\log^2 n)$ of these nodes. So there will be a node with indegree $\Theta(\log^2 n)$ with high probability.

Next we bound the diameter. Consider any two nodes v and w, and let $\delta_{v,w} = h(w) - h(v)$ if $h(w) \geq h(v)$ and otherwise $\delta_{v,w} = 1 + h(w) - h(v)$, i.e. $\delta_{v,w}$ is the distance between v and w when walking clockwise along the Chord ring. If we take a hypercubic edge to $\text{succ}(h(v) + 1/2^i)$ where i is the largest value so that $h(v) + 1/2^i \leq h(w)$ (resp. $h(v) + 1/2^i \leq 1 + h(w)$ if $h(w) < h(v)$), then we end up at a node u with $\delta_{u,w} \leq \frac{1}{2}\delta_{v,w}$. Hence, hypercubic edges always allow us to cut the distance a message has to travel by at least a factor of 2. Hence, in $O(\log n)$ hops a message can be sent between any pair of nodes, proving the diameter bound.

The proof for the expansion bound is quite involved and will therefore not be discussed here. See [1] for details. □

In order to maintain Invariant 9.10, we need proper rules for joining and leaving the network. As a subroutine for the JOIN method, we need a routing mechanism to send a message to any peer in the network.

Routing in Chord

Consider the following routing strategy:

Suppose that node u is the current location of a message with destination $x \in [0, 1)$. As long as $x \notin (h(\text{pred}(h(u))), h(u)]$ (in which case the message has not yet reached $\text{succ}(x)$), u sends the request to the node $\text{succ}(h(u) + 1/2^i)$ with maximum i so that $h(u) + 1/2^i \leq x$ (treating $[0, 1)$ here as a ring).

Using the diameter proof in Theorem 9.11, we obtain the following result:

Lemma 9.12 *For any node $v \in V$ and any real number $x \in [0, 1)$, it takes at most $O(\log n)$ hops to send a message from v to* $\text{succ}(x)$.

Joining and leaving the network

Suppose that a new node v contacts node $w \in V$ to join the system. Then w will forward v's request to $\text{succ}(h(v))$ using the Chord routing strategy with $x = h(v)$. $\text{succ}(h(v))$ will then integrate v between

succ$(h(v))$ and pred$(h(\text{succ}(v)))$. Afterwards, succ$(h(v))$ will send the endpoints of all of its outgoing hypercubic edges over to v. v then sends requests to these endpoints, using the Chord routing strategy, to establish its own outgoing hypercubic edges. Furthermore, succ$(h(v))$ will move all of the incoming hypercubic edges relevant for v according to Invariant 9.10 to v and notify the origins of these edges about that. Once v has received information about all of its outgoing and incoming hypercubic edges, the join operation terminates.

Theorem 9.13 *Inserting a new node requires $O(\log n)$ time and message transmissions on expectation and $O(\log^2 n)$ time and message transmissions with high probability.*

Proof. We only sketch the proof. Due to symmetry reasons, the expected fraction of the $[0,1)$ ring owned by any particular node v is $1/n$ (i.e. all edges with an endpoint in that area will be assigned to v). Furthermore, the largest possible interval in the $[0,1)$ ring without a node is at most $O((\log n)/n)$ with high probability, because the probability that no node chooses a value in some fixed interval of size $(c \ln n)/n$ is equal to

$$\left(1 - \frac{c \ln n}{n}\right)^n \leq \mathrm{e}^{-\frac{c \ln n}{n} \cdot n} = n^{-c} \; .$$

Hence, the maximum fraction of the $[0,1)$ ring owned by a node is $O((\log n)/n)$ with high probability.

If a new node receives an area of size k, then on expectation and with high probability, $O(\log n)$ new edges from v have to be created and $O(k \log n)$ edges to v have to be changed, resulting in the theorem. □

If a node v wants to leave the system, it first notifies all nodes having edges to it to point to v's successor, and then it removes itself from the doubly linked cycle by connecting its predecessor and successor. Also here we get:

Theorem 9.14 *Deleting a node requires $O(\log n)$ time and message transmissions on expectation and $O(\log^2 n)$ time and message transmissions with high probability.*

Data management

The data management works exactly like the consistent hashing strategy presented in Section 5: Data items are hashed to random values in $[0,1)$, and any data item d is placed at the node v with $v =$ succ$(h(d))$.

Using this strategy, data will on expectation be evenly distributed among the nodes, and on expectation, at most a factor of 2 more data than necessary has to be replaced if a node joins or leaves.

Searching

When a new data item has to be inserted, or when a data item has to be located, we can use the Chord routing strategy above to learn about the node responsible for it and then contact it directly to insert or retrieve the data item. This strategy achieves the following result, which follows immediately from Lemma 9.12.

Theorem 9.15 *Any search operation can be executed in $O(\log n)$ hops, with high probability.*

Fault tolerance

In order to achieve a high fault tolerance, every node maintains pointers to its $O(\log n)$ successors on the doubly linked cycle. In this case it holds:

Claim 9.16 *If the nodes have random values in $[0,1)$, then even if ϵn nodes leave at the same time for some constant $\epsilon < 1$, the probability that a node loses all of its successors is polynomially small in n.*

Proof. If ϵn nodes leave the system, the probability that all $c \log n$ successors of a node are among them is

$$\frac{\epsilon n}{n} \cdot \frac{\epsilon n - 1}{n-1} \cdot \ldots \cdot \frac{\epsilon n - (c \log n - 1)}{n - (c \log n - 1)} \leq \epsilon^{c \log n} = n^{-c \log(1/\epsilon)} .$$

This is polynomially small if c is sufficiently large compared to ϵ. □

9.4 Maintaining a dynamic deBruijn graph

Next we show how to dynamically maintain a deBruijn graph. Recall the definition of a deBruijn graph.

Definition 9.17 (deBruijn) *The d-*dimensional DeBruijn graph $DB(d)$ *is an undirected graph $G = (V, E)$ with node set $V = [2]^d$ and edge set E that contains all edges $\{v, w\}$ with the property that $v = (v_{d-1}, \ldots, v_0)$ and $w \in \{(x, v_{d-1}, \ldots, v_1) : x \in \{0,1\}\}$.*

Viewing the binary label of a node as a point in $[0,1)$, it turns out that x is connected to y if $y = x/2$ or $y = (1+x)/2$ (after removing the least significant bit in x). Hence, it is natural to formulate the following continuous version of the deBruijn graph:

Let $U = [0,1)$ and F consist of two functions, f_0 and f_1, where $f_i(x) = (i+x)/2$ for each $i \in \{0,1\}$.

Using this family, one can show the following result:

Theorem 9.18 *Let $W \subseteq V$ be the set of nodes currently in the system. If the consistent hashing scheme is used to map the nodes to regions in $[0,1)$, then the graph $G_F(W)$ resulting from the continuous-discrete approach has a node expansion of $\Omega(1/\log n)$ and a smoothness of $O(\log n)$, with high probability. Furthermore, every node has a degree of at most $O(\log n)$ with high probability.*

Routing in the deBruijn graph

Suppose that Invariant 9.1 is satisfied for our family of hypercubic functions. Then we use the following trick to route a message from any point $x \in [0,1)$ to any point $y \in [0,1)$ along edges in $G_F(W)$.

Let z be a randomly chosen point in $[0,1)$. Let $x_1 x_2 x_3 \ldots$ be the binary representation of x, $y_1 y_2 y_3 \ldots$ be the binary representation of y, and $z_1 z_2 z_3$ be the binary representation of z. Let P be the path along the points $x = x_1 x_2 x_3 \ldots, z_1 x_1 x_2 \ldots, z_2 z_1 x_1 \ldots, \ldots$ and P' be the path along the points $y = y_1 y_2 y_3 \ldots, z_1 y_1 y_2 \ldots, z_2 z_1 y_1 \ldots, \ldots$ Then we simulate moving along the points in P by moving along the corresponding edges in $G_F(W)$ until we hit a node $w \in W$ with the property that $R(w)$ contains a point in P and a point in P'. At that point, we follow the points in P' backwards until we arrive at the node $w' \in W$ that contains y in $R(w')$. Using this strategy, it holds:

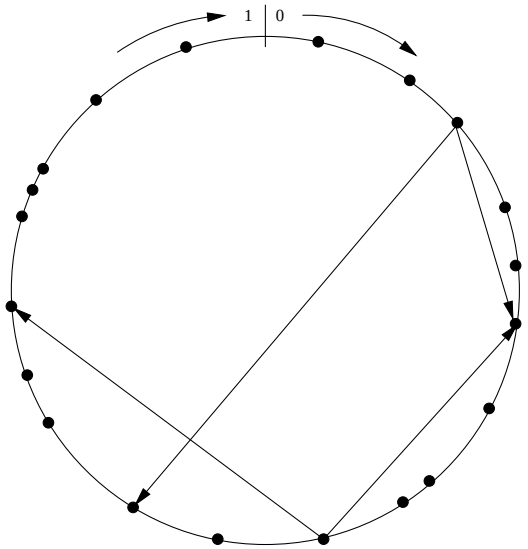

Figure 2: An example of a dynamic deBruijn network (only some short-cut pointers for two nodes are given).

Lemma 9.19 *Given a node set W with $|W| = n$, it takes at most $O(\log n)$ hops, with high probability, to route in $G_F(W)$ from any point $x \in [0, 1)$ to any point $y \in [0, 1)$.*

Joining and leaving the network

Joining and leaving the network is done in basically the same way as in the hypercube, giving the following result:

Theorem 9.20 *It takes a routing effort of $O(\log n)$ hops and an update work of $O(\log n)$ messages that can be processed in a constant number of communication rounds in order to execute a join or leave operation in the dynamic deBruijn graph.*

One can also use the dynamic deBruijn graph for distributed data management by implementing the consistent hash strategy on top of it.

References

[1] B. Awerbuch and C. Scheideler. Chord++: Low-congestion routing in chord. Unpubished manuscript, Johns Hopkins University, June 2003.

[2] M. F. Kaashoek and D. R. Karger. Koorde: A simple degree-optimal distributed hash table. In *2nd Intl. Workshop on Peer-to-Peer Systems (IPTPS)*, 2003.

[3] M. Naor and U. Wieder. Novel architectures for P2P applications: the continuous-discrete approach. In *Proc. of the 15th ACM Symp. on Parallel Algorithms and Architectures (SPAA)*, pages 50–59, 2003.

[4] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *SIGCOMM '01*, 2001.

[5] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for Internet applications. In *SIGCOMM '01*, pages 149–160, 2001.