11 Overlay networks for searching

In this section we will present two overlay networks useful for data management where peers can join and leave without the aid of a supervisor. Certainly, a supervisor may always have to exist because otherwise new peers are not able to find out who is currently in the system. However, the role of the supervisor would simply reduce to providing a pointer (such as the IP address) to some peer in the system. Everything else will be handled by the peers themselves.

An overlay network has to offer the following operations to be suitable for data management:

- p.JOIN(q): peer p is contacted by peer q in order to join the system.
- p.LEAVE(): peer p wants to leave the system.
- p.SEARCH(key): peer p searches for the peer responsible for key

We will describe two overlay networks that provide this functionality in an efficient way: Chord [3] and Koorde [1, 2].

11.1 Chord

Suppose that we have a (pseudo-)random hash function h assigning to each node an ID representing a real number in $[0, 1)$. Given a set of nodes V and a real number $x \in [0, 1)$, we define:

- succ$(x) = \mathsf{argmin}\{v \in V \mid h(v) > x\}$, i.e. succ$(x)$ is the node whose hash value is the closest successor of x, and
- pred$(x) = \mathsf{argmax}\{v \in V \mid h(v) < x\}$, i.e. pred$(x)$ is the node whose hash value is the closest predecessor of x.

Notice that we view $[0, 1)$ as a ring here. This means for succ(x) that if there is no node v with $h(v) > x$, then we associate succ(x) with the node with smallest $h(v)$ among all nodes in V. In the Chord network, the following invariants have to be kept at any time.

Invariant 11.1 *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$(h(v))$.

Invariant 11.1(a) requires that the nodes are organized in a sorted, doubly linked cycle, the so-called *Chord ring*, and Invariant 11.1(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 *shortcut edges*. These shortcut edges actually form a hypercubic structure. To see this, recall the definition of the hypercube:

Definition 11.2 (Hypercube) *The* d-dimensional hypercube $H(d)$ *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 *and* w *differ in exactly one bit.*

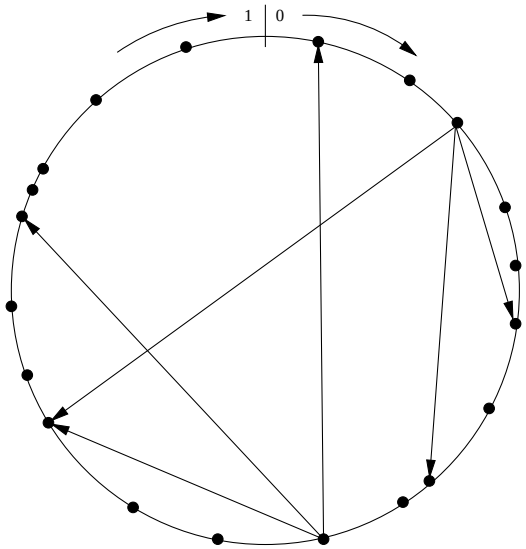

Figure 1: An example of a Chord network (only short-cut pointers for two nodes are given).

Now, view any node $v = (v_{d-1}, \ldots, v_0)$ as a real number $x_v = \sum_{i=1}^{d} v_i 2^{-d-i}$. Obviously, $x_v \in [0, 1)$, so it is a legal number in our domain. Using this encoding, it holds for every node w that differs from v only in the ith largest bit that $x_w = x_v + 1/2^{-i}$ or $x_w = x_v - 1/2^{-i}$ (depending on v_{d-i}). This is very similar to the requirement for our shortcut edges in Invariant 11.1.

The Chord overlay network has the following properties, where n is the current number of nodes in the system:

Theorem 11.3 *If Invariant 11.1 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 $\Omega(1/logn)$ 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 shortcut 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 $v \in V$, $\mathrm{succ}(h(v) + 1/2^{3 \log n}) = \mathrm{succ}(h(v))$. Thus, with high probability every node only has $O(\log n)$ shortcut 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 shortcut pointer 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 shortcut pointer to 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, shortcut pointers 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. □

In order to maintain Invariant 11.1, 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 succ(x)), u sends the request to the node 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 11.3, we obtain the following result:

Lemma 11.4 *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* 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 succ(v) using the Chord routing strategy with $x = h(v)$. succ(v) will then integrate v between succ(v) and pred$(h(\text{succ}(v)))$. Afterwards, succ(v) will send all of its outgoing shortcut pointers over to v. v then sends requests to these pointers, using the Chord routing strategy, to establish its own outgoing shortcut pointers. Furthermore, succ(v) will move all of the incoming shortcut pointers relevant for v according to Invariant 11.1 to v and notify the origins of these pointers about that. Once v has received information about all of its outgoing and incoming shortcut pointers, the join operation terminates.

Theorem 11.5 *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 pointers 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 11.6 *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 very similar to the nearest neighbor strategy presented in Section 8: Also data items are hashed to random values in $[0,1)$, and any data item d is placed at the node v with $v = \text{succ}(h(d))$.

This strategy shares all the properties of the nearest neighbor strategy. That is, it is faithful, space-efficient, time-efficient, and 2-competitive concerning adaptivity. Hence, 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 11.4.

Theorem 11.7 *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 11.8 *If nodes have random IDs, 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. □

11.2 Koorde

The basic approach of Koorde [1] is similar Chord. Also here, every node is given a random value in $[0, 1)$ via some hash function h. However, instead of using a hypercubic structure for interconnecting the nodes, Koorde uses a DeBruijn-like structure. More precisely, the following invariant has to be maintained at any time.

Invariant 11.9 *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$(h(v))$,

(b) succ$(h(v)/2)$, *and* succ$((1 + h(v))/2)$.

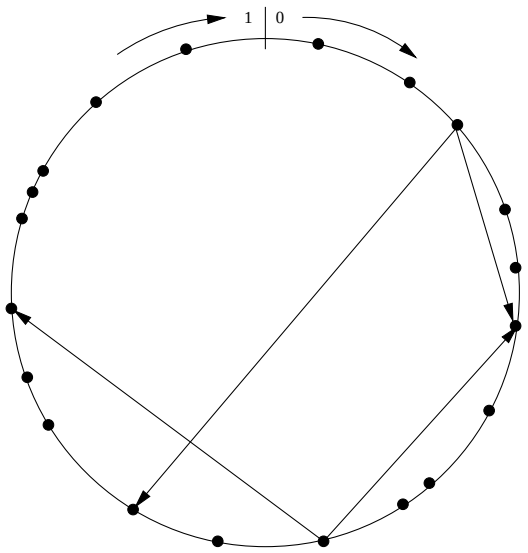

Figure 2: An example of a Koorde network (only short-cut pointers for two nodes are given).

To see that this indeed forms a DeBruijn-like structure, recall the definition of the DeBruijn graph:

Definition 11.10 (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 $w \in \{(x, v_{d-1}, \ldots, v_1) : x \in \{0, 1\}\}$.*

Now, view any node $v = (v_{d-1}, \ldots, v_0)$ as a real number $x_v = \sum_{i=1}^{d} v_i 2^{-d-i}$. Obviously, $x_v \in [0, 1)$, so it is a legal number in our domain. Using this encoding, $(0, v_{d-1}, \ldots, v_1)$ is approximately equal to $x_v/2$ and $(1, v_{d-1}, \ldots, v_1)$ is approximately equal to $(1 + x_v)/2$, so a DeBruijn-like structure can be observed in Invariant 11.9.

The Koorde overlay network has the following properties, where n is the current number of nodes in the system:

Theorem 11.11 *If Invariant 11.9 is true and h assigns random numbers to nodes, then Koorde has a maximum outdegree of 4, a maximum indegree of $O(\log n)$, a diameter of $O(\log n)$, and a node expansion of $\Omega(1/\log n)$ with high probability.*

Proof. The bound on the maximum outdegree is obvious. For the maximum indegree, notice that there will be a node v in the system with high probability where the interval $I_v = [h(v) - (\epsilon \log n)/n, h(v)]$ is not occupied by any other node if the constant $\epsilon \in (0, 1)$ is sufficiently small (see the proof of Theorem 11.3). Hence, all nodes in the intervals $[2(h(v) - (\epsilon \log n)/n), 2h(v))$ and $[2(h(v) - (\epsilon \log n)/n) - 1, 2h(v) - 1)$ would have to have a shortcut pointer to v, and there are on expectation, and with high probability, $\Theta(\log n)$ of these nodes. So there will be a node with indegree $\Theta(\log n)$ with high probability.

Next we bound the diameter. Consider any node v. If we take a shortcut pointer to succ$((1 + h(v))/2)$, then we arrive at a node w with $h(w) \geq 1/2 + h(v)/2$. Hence, no matter at which node v we start, after t hops to succ$((1 + h(v))/2)$, we arrive at a node w with $h(w) \geq \sum_{i=1}^{t} 1/2^t$. Once we arrive at a node w with smallest $h(w)$ (i.e. we passed the 0/1 border), we stop. According to the proof of Theorem 11.3, this takes only $O(\log n)$ hops. Hence, there must be a path from any node to any other node of length $O(\log n)$, which proves the diameter bound.

Again, the proof of the expansion bound is quite involved and left out here. □

Thus, compared to Chord, Koorde reduces the degree bounds by a logarithmic factor while preserving diameter and expansion. In order to maintain Invariant 11.9, we use the following rules when joining or leaving the network.

Routing in Koorde

Consider the following routing strategy for sending a message from a node v with $h(v) = 0.x_1x_2x_3\ldots$ in binary notation (i.e. $h(v) = \sum_{i \geq 1} x_i/2^{-i}$) to the virtual destination $z = 0.z_1z_2z_3\ldots$ in binary notation:

Forward the message along the nodes succ$(0.y_1x_1x_2x_3\ldots)$, succ$(0.y_2y_1x_1x_2\ldots)$, succ$(0.y_3y_2y_1x_1\ldots)$, etc., where the bits $y_1, y_2, y_3, \ldots$ are chosen at random, until a node succ$(0.y_k\ldots y_1x_1x_2\ldots)$ is reached with

$$\text{succ}(0.y_k\ldots y_1x_1x_2\ldots) = \text{succ}(0.y_k\ldots y_1z_1z_2\ldots)\ .$$

At that point, forward the message along the nodes succ$(0.y_{k-1}\ldots y_1z_1z_2\ldots)$, succ$(0.y_{k-2}\ldots y_1z_1z_2\ldots)$, etc., until node succ$(z_1z_2z_3\ldots)$ is reached.

Notice that each virtual hop above may require more than just moving along a shortcut edge. Also ring edges may have to be used once in a while to adapt the position. However, on expectation, only $O(1)$ ring edges have to be used for each virtual hop above. The reason why this routing strategy is more complicated than the Chord strategy is that routing in the DeBruijn graph requires knowledge

about the size of the graph to accurately simulate the rotation operations defining the edges. Since a node may not necessarily be aware of the size of the network, this more complicated strategy, that proceeds towards a random intermediate destination until the two trajectories from the source and destination meet, has to be used.

The Koorde routing strategy achieves the following result (which will not be shown here):

Lemma 11.12 *For any node $v \in V$ and any real number $z \in [0, 1)$, it takes at most $O(\log n)$ hops, with high probability, to send a message from v to* succ(z).

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 succ(v), using the Koorde routing strategy above. succ(v) will then integrate v between succ(v) and pred(succ(v)). Afterwards, succ(v) will send all of its outgoing shortcut pointers over to v. v then sends requests to these pointers, using the Koorde routing strategy, to establish its own outgoing shortcut pointers. Furthermore, succ(v) will move all of the incoming shortcut pointers relevant for v according to Invariant 11.9 to v and notify the origins of these pointers about that. Once v has received information about all of its outgoing and incoming shortcut pointers, the join operation terminates.

Theorem 11.13 *Inserting a new node requires $O(\log n)$ time and message transmissions with high probability.*

If a node v wants to leave the system, it first notifies all nodes having pointers 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 11.14 *Deleting a node requires $O(\log n)$ time and message transmissions with high probability.*

Data management and searching

Data and search requests are handled in the same way as in Chord, with the only difference that search requests will use the Koorde routing strategy to reach the node responsible for a data item.

References

[1] 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.

[2] 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.

[3] 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.