9 Decentralized overlay networks II

So far, we saw how to construct completely decentralized peer-to-peer systems with good topological properties if the peers are assigned to random locations in the $[0,1)$-interval. However, there are several scenarios in which it would be much better if the peers are organized according to their real, user-defined names instead of just random names.

For example, suppose that we want to implement a distributed name service such as the well-known domain name service (DNS). Then we would like to organize the peers in a peer-to-peer system so that a peer with a given name can be found quickly. If the names were well-spread in the name space so that we could interpret them as well-spread numbers in the $[0,1)$-interval, then we could use the dynamic de Bruijn network to implement such a service. However, we cannot guarantee that the names will be well-spread, and therefore we need a different overlay network design.

As another example, consider the situation that we want to design a peer-to-peer system in which we can take locality issues into account. Locality is an important issue in the Internet. Using the dynamic de Bruijn network can mean that a message is sent $\log n$ times across the world before it reaches its destination. Instead, imagine that we knew the geographic location of every peer. One possible way of specifying such a location could be

North_America.USA.MD.Baltimore.Johns_Hopkins_Univesity.Computer_Science

If such information is available, we could organize the peers in an overlay network sorted according to this location information so that now messages will only be sent once across the world in the worst case. Instead of a geographical location, one could also use a hierarchically specified Internet location, starting with the backbone ISP, the local ISP, and so on (which may be determined via traceroute, for example).

In the following, we present overlay network designs that allow peers to be ordered according to arbitrary user-defined names. We first present (random) skip graphs [1, 4], and then we present deterministic skip graphs which are also known as hyperrings [3].

9.1 Skip graphs

Given an infinite bit string $b = x_1 x_2 x_3 \ldots$, we define $\text{prefix}_0(b) = \epsilon$ (the empty word) and $\text{prefix}_i(b) = x_1 x_2 \ldots x_i$ for every $i \geq 1$. Suppose that we have a (pseudo-)random hash function h assigning to each node an ID representing an infinite bit string. Given a set of nodes V, we define for every $v \in V$ and $i \geq 0$:

- $\text{succ}_i(v) = \text{argmin}\{w \in V \mid \text{Name}(w) > \text{Name}(v)$ and $\text{prefix}_i(h(v)) = \text{prefix}_i(h(w))\}$, i.e. $\text{succ}_i(v)$ is the node w whose name is the closest successor of v's name (with respect to lexicographical ordering) with the same i first bits in $h(w)$ as $h(v)$, and
- $\text{pred}_i(v) = \text{argmax}\{w \in V \mid \text{Name}(w) < \text{Name}(v)$ and $\text{prefix}_i(h(v)) = \text{prefix}_i(h(w))\}$.

Notice that we view the name space as a ring here. This means for $\text{succ}_i(v)$ that if there is no node w with $\text{Name}(w) > \text{Name}(v)$ that fulfills the prefix condition, then we associate $\text{succ}_i(v)$ with the node w with smallest name so that $\text{prefix}_i(h(v)) = \text{prefix}_i(h(w))$. If there is no other node w in the network with that property, then we set $\text{succ}_i(v) = v$. In skip graphs, the following invariants have to be kept at any time.

Invariant 9.1 *For any set of nodes V currently in the system, it holds for every $v \in V$ that v is connected to* $\mathrm{succ}_i(v)$ *and* $\mathrm{pred}_i(v)$ *for all* $i \geq 0$.

Invariant 9.1 requires that the nodes are organized in a hierarchy of doubly linked cycles, where the node names have to be sorted in every cycle, and every node participates in exactly one cycle for every $i \geq 0$. A cycle at level i is called i-cycle or i-ring, and an edge in an i-ring is called an i-edge. Skip graphs have the following properties, where n is the current number of nodes in the network:

Theorem 9.2 *If Invariant 9.1 is true and h assigns random bit strings to nodes, then the skip graph has a maximum degree of $O(\log n)$, a diameter of $O(\log n)$, and a node expansion of $\Omega(1)$, with high probability.*

Proof. The probability that some fixed node pair v and w fulfills $\mathrm{prefix}_i(h(v)) = \mathrm{prefix}_i(h(w))$ is equal to $1/2^i$. Hence, for $i \geq 3\log n$, it holds that

$$\begin{aligned}
&\Pr[\text{there is a node pair } v,w \text{ with } \mathrm{prefix}_i(h(v)) = \mathrm{prefix}_i(h(w))] \\
&\quad\leq \sum_{v,w} \Pr[\text{node pair } v,w \text{ fulfills } \mathrm{prefix}_i(h(v)) = \mathrm{prefix}_i(h(w))] \\
&\quad= \sum_{v,w} \frac{1}{2^{3\log n}} \leq n^2 \cdot \frac{1}{n^3} = \frac{1}{n} \,.
\end{aligned}$$

Hence, with high probability there is no ring of level $3\log n$ or higher. Thus, every node has a degree of at most $2(3\log n + 1)$.

Next, we bound the diameter. Consider any pair of nodes $v, w \in V$. Our aim is to move from v to w by adapting to the bits in $h(w) = x_1x_2x_3\ldots$ one by one. To do this, we move from $u_0 = v$ to the closest successor u_1 on the 0-ring with $\mathrm{prefix}_1(u_1) = x_1$, and in general from node u_i to the closest successor u_{i+1} on the current i-ring with $\mathrm{prefix}_{i+1}(u_{i+1}) = x_1x_2\ldots x_{i+1}$. For any $i \geq 0$, it holds:

$$\Pr[\text{the distance from } u_i \text{ to } u_{i+1} \text{ on the } i\text{-ring is } \delta] = \left(\frac{1}{2}\right)^{\delta+1} .$$

Hence,

$$\begin{aligned}
\mathrm{E}[\text{distance to } u_{i+1}] &= \sum_{\delta\geq 0} \delta \cdot \left(\frac{1}{2}\right)^{\delta+1} = \sum_{\delta\geq 0} (\delta+1) \cdot \left(\frac{1}{2}\right)^{\delta+2} \\
&= \frac{1}{4}\left(\sum_{\delta\geq 0}\left(\frac{1}{2}\right)^{\delta}\right)^2 = \frac{1}{4}\cdot 4 = 1 \,.
\end{aligned}$$

Since we know from the degree proof that there are at most $3\log n$ levels with high probability, the expected number of hops we need to perform to get from v to w is $O(\log n)$, and this can also be shown to hold with high probability. Thus, the diameter is $O(\log n)$, with high probability.

The expansion proof is involved and will not be shown here. See [2] for details. □

Routing in skip graphs

Consider the following routing strategy:

Suppose that node u is the current location of a message with destination Name. As long as $\mathrm{Name} \notin [\mathrm{Name}(u), \mathrm{Name}(\mathrm{succ}_0(u)))$ (i.e. the message has not yet reached a node u that is the closest predecessor of Name), u sends the message to the node $\mathrm{succ}_i(u)$ with maximum i so that $\mathrm{Name}(\mathrm{succ}_i(u)) \leq \mathrm{Name}$ (treating the name space as a ring).

One can show the following result:

Lemma 9.3 *For any node $v \in V$ and any name Name, it takes at most $O(\log n)$ hops, with high probability, to send a message from v to the node whose name is the closest successor to Name.*

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 $\mathrm{pred}_0(v)$ using the routing strategy above with $\mathrm{Name} = \mathrm{Name}(v)$. $\mathrm{pred}_0(v)$ will then integrate v between $\mathrm{pred}_0(v)$ and $\mathrm{succ}_0(v)$. Afterwards, v sends out two requests along the 0-ring to find $\mathrm{pred}_1(v)$ and $\mathrm{succ}_1(v)$. Once they are found, v integrates itself into its 1-ring. v then uses the 1-ring to find $\mathrm{pred}_2(v)$ and $\mathrm{succ}_2(v)$, and then integrates itself into the 2-ring. This continues until v has integrated itself into the highest possible ring containing at least 2 nodes.

Using a probabilistic analysis, one can show the following result:

Theorem 9.4 *Inserting a new node requires $O(\log n)$ time and work with high probability.*

If a node wants to leave the system, it does this by simply connecting $\mathrm{pred}_i(v)$ with $\mathrm{succ}_i(v)$ for every $i \geq 0$. This gives the following result:

Theorem 9.5 *Deleting a node requires $O(\log n)$ time and work with high probability.*

Searching

When a node v searches for a node with name Name, then it simply uses the routing strategy described above. Once a node w with $\mathrm{Name}(w) = \mathrm{Name}$ has been found, w reports its IP address back to v. This strategy has the following performance:

Theorem 9.6 *Any search operation requires $O(\log n)$ time and work with high probability.*

9.2 The hyperring

Next we consider the hyperring. Like the skip graph, also the hyperring consists of a hierarchy of rings. However, here we are much more strict about how the rings are maintained.

Suppose that we have a hyperring with n nodes. Then it consists of approximately $\log n$ levels of rings, starting with level 0. Each level $i \geq 0$ consists of approximately 2^i directed cycles of approximately $n/2^i$ nodes, which we call *rings*. All rings have the same orientation, and we require the nodes in every ring to be ordered according to their names. For every ring R at level i, two rings of level $i+1$ share its nodes in an intertwined fashion. As before, a ring at level i will be called an i-*ring*, and a level i edge will be called an *i-edge*. Consider some i-ring R and let (u, v, w, x) be four consecutive nodes on R. We say that (u, v, w, x) form an *i-bridge* (or simply a *bridge* if i is clear from the context) if there is an $(i+1)$-edge from u to x and an $(i+1)$-edge from v to w. An $(i+1)$-edge is called *perfect* if it bridges exactly two i-edges.

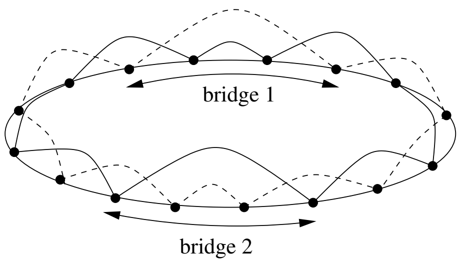

Figure 1: An example of a hyperring. The bridges have a distance of 5 from each other.

It is possible to maintain a hyperring with at most one bridge in every ring. However, in this case we would create too much update work for JOIN or LEAVE operations. Instead, we only demand that i-bridges are sufficiently far apart from each other. A hyperring is called *k-separated* if in every i-ring R the i-bridges on R are at least k nodes apart from each other, which means that there are at least $k-1$ nodes between the quadruples of nodes forming a bridge. We start with a few properties of hyperrings which are easy to prove.

Lemma 9.7 *For every $k \geq 0$, the k-separated hyperring has a maximum degree of at most $2(1 + 2/(k+1)) \log n$ and a diameter of at most $3 \log n$.*

Proof. First, we bound the maximum degree. Consider some i-ring R. In order to minimize the size of an $i+1$-ring R' on top of R without violating k-perfectness, the best one can do is using a repetitive sequence of $\lceil k/2 \rceil + 1$ edges, where one edge bridges three edges in R and the remaining $\lceil k/2 \rceil$ edges bridge two edges in R. Hence,

$$\begin{aligned} |R'| &\geq \frac{|R|}{3 + 2\lceil k/2 \rceil} \cdot (1 + \lceil k/2 \rceil) = \left(\frac{1}{2} - \frac{1}{4(\lceil k/2 \rceil + 1) + 2} \right) \cdot |R| \\ &\geq \frac{1}{2} \left(1 - \frac{1}{k+3} \right) \cdot |R| \end{aligned}$$

This also implies that $|R'|$ can be at most $\frac{1}{2}(1 + \frac{1}{k+3})|R|$. Hence, an i-ring R can have a size of at most

$$\frac{1}{2^i} \left(1 + \frac{1}{k+3} \right)^i \leq \frac{1}{2^i} \cdot e^{i/(k+3)} \leq 2^{-i(1-2/(k+3))} .$$

This is at most 1 if $i \geq (\log n)/(1 - 2/(k+3))$. Since each node has 2 edges in each level in which it participates, the maximum node degree is $2(\log n)/(1 - 2/(k+3)) = 2(1 + 2/(k+1)) \log n$.

Next, we bound the diameter. Consider any two nodes v and w on a ring R, and let R_0 and R_1 be the two intertwined rings on top of R. Furthermore, let v_0 be the node in R_0 nearest to v and w_0 be the node in R_0 nearest to w. Define v_1 and w_1 in the same way for R_1. First of all, $d_R(v, v_0)$, $d_R(v, v_1)$, $d_R(w, w_0)$, and $d_R(w, w_1)$ are all at most 1. Hence, $d_R(v_0, w_0) \leq d_R(v, w) + 2$

and $d_R(v_1, w_1) \leq d_R(v, w) + 2$. Since the nodes used by R_0 and R_1 are disjoint, it must hold that either $d_{R_0}(v_0, w_0) \leq d_R(v, w)/2 + 1$ or $d_{R_1}(v_1, w_1) \leq d_R(v, w)/2 + 1$. Hence, if we always take the ring of lower distance in each layer, then for each layer i we obtain the recursion $d_{i+1} \leq d_i/2 + 3$ with $d_0 = d_R(v, w)$. Therefore, the total number of edges used is at most $3 \log n$. □

Unfortunately, hyperrings with constant separation can have a bad expansion.

Theorem 9.8 *For every $k \geq 0$, the k-separated hyperring has, in the worst case, an edge expansion of*

$$O(1/n^{1/(2(3(k+4))^2)}) .$$

The proof of this theorem is quite involved and can be found in [3]. Unfortunately, Theorem 9.8 implies that no k-separated hyperring with $k = O((\log n)^{1/2-\epsilon})$ for some constant $\epsilon > 0$ can guarantee an expansion of $\Omega(1/\log^c n)$ for some constant c depending on ϵ. Hence, in order to have a good expansion, we need $k = \Omega(\sqrt{\log n})$. However, notice that when k depends on the size of the hyperring, node insertions and deletions that have been performed in the past might have used a k that significantly differs from the k used by current insertions and deletions. Hence, parts of the hyperring may be out of date. So the question is whether it is necessary to revisit these parts in order to bring the hyperring up to date. Fortunately, as one of the main results in [3], it was shown that *this is not necessary*. One can simply use as the current k the degree of any node currently in the system when executing a JOIN or LEAVE operation, and old JOIN or LEAVE operations never have to be revisited, to show the following result. ($|R|$ denotes the number of nodes in a ring R, and $|e|$ denotes the number of node on the 0-ring bridged by edge e.)

Proposition 9.9 *At any time it holds:*

1. *the* ring distortion is low, *i.e. for every i-ring R, $|R| \in [\frac{1}{2} \cdot n/2^i - 1, 2 \cdot n/2^i + 1]$ and*
2. *the* edge distortion is low, *i.e. for every i-edge e, $|e| \leq 4 \cdot 2^i$.*

The proof for this is quite complicated and can be found in [3]. For simplicity, we assume for the rest of this section that k is fixed. We start with describing how to route in the hyperring.

Routing in the hyperring

We use the same routing strategy as for skip graphs:

Suppose that node u is the current location of a message with destination Name. As long as $\mathrm{Name} \notin [\mathrm{Name}(u), \mathrm{Name}(\mathrm{succ}_0(u)))$ (i.e. the message has not yet reached a node u that is the closest predecessor of Name), u sends the message to the node $\mathrm{succ}_i(u)$ with maximum i so that $\mathrm{Name}(\mathrm{succ}_i(u)) \leq \mathrm{Name}$ (treating the name space as a ring).

Since this routing strategy prefers edges of higher level and every $i + 1$-edge bridges at most 3 i-edges for every i, we obtain the following fact.

Fact 9.10 *Any message moves along a sequence of edges of non-increasing level and uses at most two edges in each level.*

Combining this with Lemma 9.7, which says that there are at most $(1 + 2/(k + 1)) \log n$ levels, we achieve the following result.

Lemma 9.11 *For any node $v \in V$ and any name* Name*, it takes at most $O(\log n)$ hops to send a message from v to the node whose name is the closest successor to* Name.

Joining and leaving the network

First, we introduce some notation. Let $\mathrm{succ}_i(v)$ be the successor of v in its i-ring and $\mathrm{pred}_i(v)$ be the predecessor of v in its i-ring. For every node v on R, its $> i$-endpoints represent all endpoints of edges in v with level more than i. Notice that each node has two endpoints in each level. By "moving" the i-endpoints from u to v, we mean that we replace the i-edges $(\mathrm{pred}_i(u), u)$ and $(u, \mathrm{succ}_i(u))$ by the i-edges $(\mathrm{pred}_i(u), v)$ and $(v, \mathrm{succ}_i(u))$. By "permuting" the i-endpoints of u and v, we mean that we move the i-endpoints of u to v and the i-endpoints of v to u.

Suppose now that a new node u contacts some node $v \in V$ to join the system. Then v will forward u's request to $\mathrm{pred}_0(u)$ using the routing strategy above with $\mathrm{Name} = \mathrm{Name}(u)$. $\mathrm{pred}_0(u)$ will then integrate u between $\mathrm{pred}_0(u)$ and $\mathrm{succ}_0(u)$. Afterwards, u is integrated into the hyperring level by level, starting with level 0. In each level i, we integrate the node by either removing an already existing bridge in its $k+2$-neighborhood or by creating a new bridge. A bridge is removed by first dragging it over to u by permuting $> i$-endpoints (see Figure 3). Then case (b) or (c) in Figure 2 is applied. Otherwise, we just apply case (a). JOIN terminates once we reach a ring of size in $\{4, \ldots, 7\}$ (for larger rings, two new subrings are created).

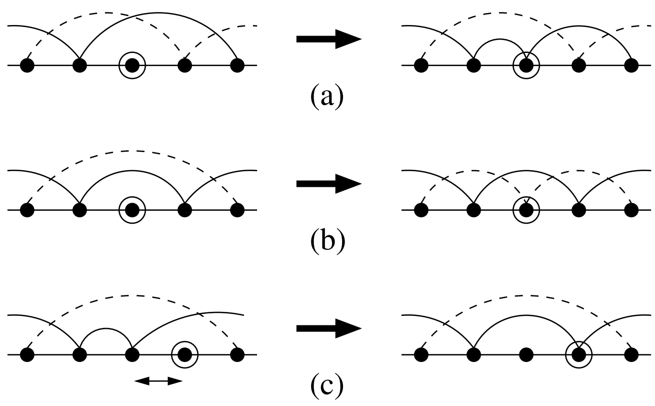

Figure 2: The three cases when adding a node. Case (c) reduces to case (b).

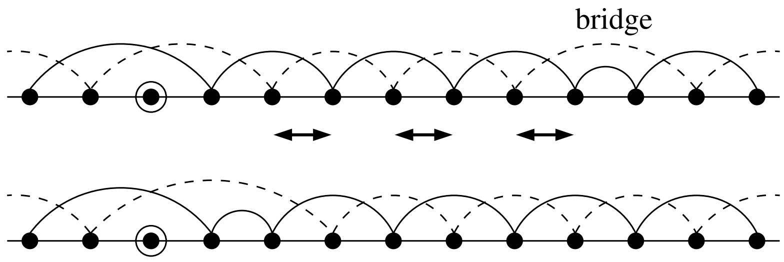

Figure 3: Permuting $> i$-endpoints drags the bridge over to obtain, e.g., case (c) in Figure 2.

Theorem 9.12 JOIN *locally preserves the* k*-separation of the hyperring and requires* $O(k \log^2 n)$ *work and* $O(\log k \cdot \log n)$ *time.*

Proof. JOIN locally preserves the k-separation property because it only creates a bridge if there is no other bridge in the $k+2$-neighborhood. Otherwise, it removes a bridge. Thus, it remains to prove the work and time bounds.

In each level, only a $O(k)$-neighborhood is investigated. In the worst case, a bridge has to be moved to u (resp. to the node to be integrated into that level in place of u). This requires $O(k \log n)$ message transmissions. Since the hyperring has $O(\log n)$ levels, the total work is $O(k \log^2 n)$.

When using edges in higher levels, we can investigate the $O(k)$-neighborhood of a node in $O(\log k)$ steps. Thus, in $O(\log k)$ steps we can update the endpoints necessary to proceed with the next higher level. Since there are $O(\log n)$ levels, this results in $O(\log k \cdot \log n)$ time. □

We also remove a node u from the hyperring level by level, starting with level 0. In each level, we remove the node by either removing an already existing bridge in its $k+2$-neighborhood or by creating a new bridge. A bridge is removed by first dragging it over (Figure 3) and then applying case (b) or (c) in Figure 4. Otherwise, we just apply case (a). LEAVE terminates once we reach a ring of size in $\{4, \ldots, 7\}$ (rings smaller than 4 are removed).

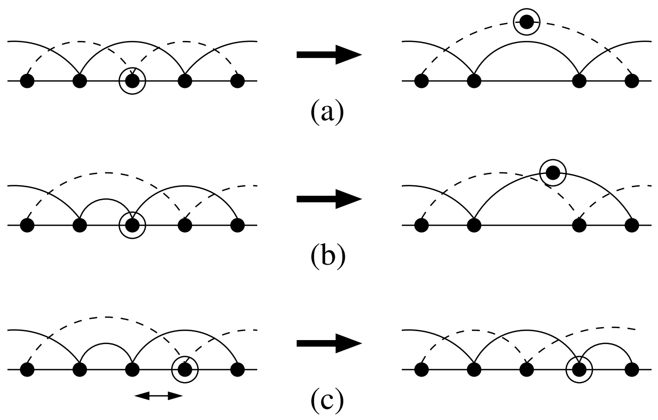

Figure 4: The three cases when removing a node. Case (c) reduces to case (b).

Theorem 9.13 LEAVE *locally preserves the k-separation of the hyperring and requires $O(k \log^2 n)$ work and $O(\log k \cdot \log n)$ time.*

The proof is similar to the proof of Theorem 9.12.

Searching

When a node v searches for a node with name Name, then it simply uses the routing strategy described above. Once a node w with $\mathrm{Name}(w) = \mathrm{Name}$ has been found, w reports its IP address back to v. This strategy has the following performance:

Theorem 9.14 *Any search operation requires $O(\log n)$ time and work with high probability.*

Furthermore, we can show the following result, demonstrating that not only the dilation but also the congestion of search requests can be kept low in the hyperring.

Theorem 9.15 *The congestion caused by n* SEARCH *requests, one per node, with random destinations is $O(\log n)$, with high probability.*

Proof. Fact 9.10 implies that every i-ring R can only receive requests from rings on top of it. Thus, it can only receive requests from its own nodes. Consider now an arbitrary node v in R. It is easy to check that only those requests will be sent to v whose destination is bridged by the i-edge e leaving v in R. From Proposition 9.9 we know that e bridges at most $4 \cdot 2^i$ nodes and that R consists of at most $3 \cdot n/2^i$ nodes. Since every node is the starting point of one request and every request has a random destination, the expected number of requests that want to reach v in R is at most $(4 \cdot 2^i/n) \cdot (3 \cdot n/2^i) = 12$. Combining this with the fact that every request only uses at most 2 edges in R (see Fact 9.10), the expected number of requests that traverse v in R is at most 24. Because every node participates in at most $\log n + O(1)$ levels, the overall expected number of search requests passing through v is $O(\log n)$. Using the fact that every request picks a random destination independently from other requests, one can also show that the congestion caused by SEARCH is $O(\log n)$ with high probability. □

9.3 Robustness against adversarial join-leave behavior

Finally, we consider the problem of protecting the skip graph or hyperring against adversarial join-leave behavior.

Consider the following game: There are n white pebbles (the honest peers) and ϵn black pebbles (the adversarial peers) for some fixed constant $\epsilon < 1$. Initially, all of the white pebbles are laid down in a ring, and the adversary has all of the black pebbles in its bag. In each round, the adversary can look at the entire ring and can select to add a black pebble to the ring (if its bag is not empty) or to take any black pebble from the ring and put it back into its bag (i.e. we consider *adaptive* adversaries). However, the adversary cannot place a black pebble into any position it likes. This is handled by a join strategy to be specified by the system. The goal is to find an *oblivious* join strategy, i.e. a strategy that cannot distinguish between the white and black pebbles in the ring, that integrates the black pebbles into this ring and may do some further rearrangements so that for a polynomial number of rounds the adversary will not manage to include its black pebbles into the ring so that there is a sequence of $s = \Theta(\log n)$ consecutive pebbles in which at least half of the pebbles are black. If this is achieved by the join strategy, it wins. Otherwise, the adversary wins.

We propose the *k-rotation* strategy in order to randomly perturb the pebbles. The k-rotation strategy works as follows: Initially, the new black pebble is declared a homeless pebble. For $k-1$ rounds, place the currently homeless pebble into a random position of the ring and declare the pebble previously placed at that position the new homeless pebble. Afterwards, create a new position at a random place in the ring and place the homeless pebble there.

It turns out that $k \leq 2$ is not sufficient but $k \geq 3$ is sufficient for the system to win with high probability. Interestingly, the adversary has a good chance of winning for $k = 2$ even if it has only $O(\log n)$ pebbles, whereas the adversary has only a negligible chance of winning for $k = 3$, even when having $n/4$ pebbles. Thus, a sharp threshold can be identified for the system to win or lose. The results are summarized in the following theorem [5].

Theorem 9.16 *Let n and $s = O(\log n)$ be sufficiently large. When using the k-rotation strategy, it holds:*

- *If $k = 1$, then the adversary only needs $s/2$ pebbles to win within $O(n)$ join attempts, with high probability.*
- *If $k = 2$, then the adversary only needs s pebbles to win within $O(n \log s)$ join attempts on expectation and within $O((n \log s) \log n)$ join attempts, with high probability.*
- *If $k \geq 3$, then the adversary loses with high probability as long as it has $\leq \epsilon n$ nodes for some constant $\epsilon < 1 - 2/k$, and this result is tight.*

In fact, the k-rotation rule ensures that for any $k \geq 3$, the fraction of black pebbles in a sequence of s consecutive pebbles is at most

$$(1+\delta)\left(\frac{1+k\epsilon}{k+k\epsilon}\right)$$

with high probability, where $\delta > 0$ can be an arbitrarily small constant depending on s.

Thus, as k increases, ϵ can get arbitrarily close to 1. Note that ϵ must be smaller than 1 because otherwise there is certainly no chance for the system to win.

Using the insight above together with the rule that every peer connects to its closest $\Theta(\log n)$ predecessors and successors on the ring, adversarial behavior can be washed out by majority decision so that topologies based on the skip graph and hyperring concept can be preserved even under adversarial behavior.

References

[1] J. Aspnes and G. Shah. Skip graphs. In *Proc. of the 14th ACM/SIAM Symp. on Discrete Algorithms (SODA)*, pages 384–393, 2003.

[2] J. Aspnes and U. Wieder. The expansion and mixing time of skip graphs with applications. In *Proc. of the 17th ACM Symp. on Parallel Algorithms and Architectures (SPAA)*, pages 126–134, 2005.

[3] B. Awerbuch and C. Scheideler. The Hyperring: A low-congestion deterministic data structure for distributed environments. In *Proc. of the 15th ACM/SIAM Symp. on Discrete Algorithms (SODA)*, 2004.

[4] N. J. Harvey, M. B. Jones, S. Saroiu, M. Theimer, and A. Wolman. Skipnet: A scalable overlay network with practical locality properties. In *4th USENIX Symposium on Internet Technologies and Systems*, 2003.

[5] C. Scheideler. How to spread adversarial nodes? Rotate! In *Proc. of the 37th ACM Symp. on Theory of Computing (STOC)*, pages 704–713, 2005.