Deterministic SkipNet

Nicholas J. A. Harvey [a] J. Ian Munro [b]

[a] *Microsoft Research, Redmond, WA, USA.*
nickhar@microsoft.com

[b] *School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada.*
imunro@cs.uwaterloo.ca

Abstract

We present a deterministic scalable overlay network. In contrast, most previous overlay networks use randomness or hashing (pseudo-randomness) to achieve a uniform distribution of data and routing traffic.

Key words: Data Structures, Distributed Computing, Overlay Networks

1 Introduction

Overlay networks are a useful technique for organizing nodes in a distributed system. For most existing overlays, their primary purpose is to form a *distributed hash table*; examples include CAN [6], Chord [8], Pastry [7], and Tapestry [9]. Support for locality in overlay networks has been shown to yield improved efficiency and reliability. SkipNet [3] is an overlay, based on Skip Lists [5], that incorporates locality. SkipNet differs from most other overlays in that, like its Skip List predecessor, it organizes nodes and data objects primarily by their string names, rather than by hashes of those names. The same basic structure underlying SkipNet was independently invented by Aspnes and Shah [2].

While SkipNet avoids the reliance on hashing that is common in most other overlay networks, it is a randomized data structure, as Skip Lists are. Indeed, most existing overlays depend on hashing or randomness for their construction. A natural extension of this existing work is to consider deterministic overlays. This paper describes Deterministic SkipNet, a variant of the SkipNet overlay network that attains absolute bounds on routing performance, node insertion/departure cost, and the number of routing pointers per node.

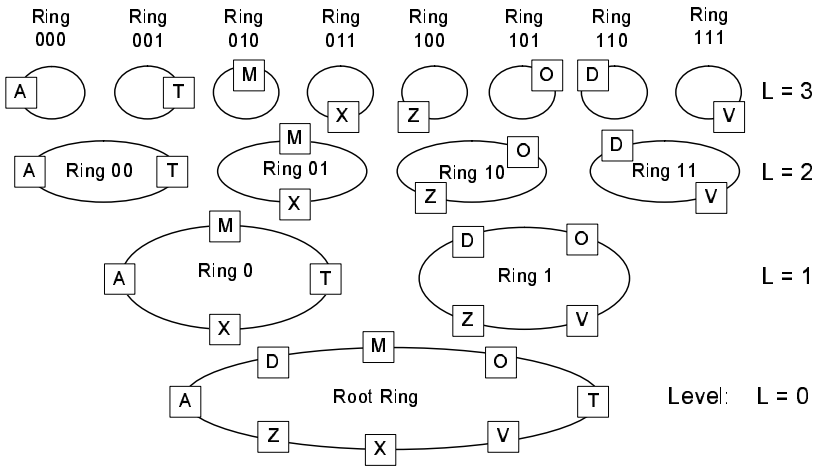

Fig. 1. *A SkipNet structure with $k = 2$.*

The only other deterministic scalable overlay network that we know of is presented in [1]. Our work builds on, but is not an immediate consequence of, the Deterministic Skip Lists due to Munro et al. [4].

2 Skip Lists and SkipNet

A Skip List [5] is an in-memory, sorted linked list in which some nodes are augmented with higher-level pointers that skip over many list elements. Each node chooses a height so that the probability of choosing height h is $1/2^h$. A pointer at level h points to the closest node to the right whose height is at least h, and thus skips over 2^h nodes in expectation. Skip Lists support $O(\log N)$ searches, insertions and deletions with high probability, where N is the number of list elements. Munro et al. [4] designed a deterministic variant of Skip Lists by observing a correspondence between Skip Lists and 2-3 trees.

In SkipNet, every node has two IDs: a string name ID and a sequence of random digits called a numeric ID. Each numeric ID digit is an integer in $[0, k-1]$, where k is a fixed parameter. As shown in Figure 1, all nodes are connected at level 0 by a "root ring" that is sorted by the nodes' name IDs. Additionally, all nodes are members of higher level rings in which the routing pointers skip over many nodes; these rings are also sorted by the nodes' name IDs. All nodes sharing a common numeric ID prefix of length h are members of the same level h ring, which is labeled with their common prefix. Nodes maintain routing pointers to their two neighbours in each ring. All SkipNet nodes have $O(\log N)$ routing pointers with high probability, where N is the total number of nodes. Furthermore searching by name ID, searching by numeric ID, and node insertion all require $O(\log N)$ time, with high probability.

3 Deterministic SkipNet

In SkipNet, nodes' numeric IDs are randomly chosen and are immutable. Since the nodes' name IDs and numeric IDs determine the SkipNet structure, all bounds on SkipNet operations are probabilistic. In contrast, Deterministic SkipNet nodes' numeric IDs are deterministically chosen and adjusted during node insertion/departure, as described in the following sections. Adjustments to the numeric IDs are necessary in order to maintain the balanced properties of the routing structure. Whereas SkipNet allows the numeric ID parameter k to be arbitrary, Deterministic SkipNet fixes $k = 3$. We leave generalization of Deterministic SkipNet to arbitrary values of k for future work.

We first define some notation. Let $D_h(n)$ denote the h^{th} digit of node n's numeric ID. Let $L_h(n)$ denote the distance at level $h-1$ from node n to its left neighbour at level h (i.e., the closest node n' to the left at level $h-1$ with $D_h(n') = D_h(n)$). $R_h(n)$ is defined symmetrically to the right. In SkipNet, $L_h(n)$ and $R_h(n)$ are both $O(k)$ in expectation, for all h and n. In contrast, the fundamental invariant of Deterministic SkipNet is that $2 \leq L_h(n) \leq 5$ and $2 \leq R_h(n) \leq 5$, for all h and n.

Consequently, a Deterministic SkipNet ring at height h has at most $\lfloor N/2^h \rfloor$ nodes, the maximum height of any ring is $\lfloor \log_2 N \rfloor$, and each node has at most $2 \cdot \lfloor \log_2 N \rfloor$ routing pointers. Searching by name ID in Deterministic SkipNet is performed in an identical manner to SkipNet: Follow the pointer that travels closest to the destination without going beyond it. This operation requires $O(\log N)$ time in Deterministic SkipNet.

3.1 Node Insertion

Node insertion consists of two steps: (1) inserting the newcomer into the root ring, and (2) inserting the newcomer into higher level rings. In the first step, the newcomer searches for its own name ID, thereby finding its neighbours in the root ring. Bidirectional pointers are then formed between the newcomer and its neighbours.

For the second step, assume that the newcomer has inserted itself at levels 0 through $h-1$, and must now insert itself at level h, without violating the Deterministic SkipNet invariant. We prove that this operation requires at most two *rotations*, and that exactly one node must recursively insert itself into higher level rings.

A rotation consists of two nodes swapping all of their pointers at levels h and above, and requires $O(\log N)$ time. The destination nodes of all affected

pointers must also adjust their corresponding pointers since pointers are bidirectional. The nodes performing the rotation must also swap their numeric ID suffixes from the h^{th} digit onwards, since numeric IDs correspond to ring membership.

We consider the numeric IDs of the nodes near the newcomer's insertion point in the relevant ring at level $h-1$. Note that these numeric IDs are identical up to the $(h-1)^{th}$ digit but differ in the h^{th} digit. We focus on the h^{th} digit of these nearby nodes in order to choose the newcomer's h^{th} digit and to determine if any rotations are necessary.

Case 1: If there exists a digit α such that none of the three nearest nodes to the left and to the right of the insertion point has their h^{th} digit equal to α then we may choose the newcomer's h^{th} digit to be α. This case may occur if the ring under consideration at level $h-1$ contains only a few nodes.

Case 2: We assume that there is at least one node to the left and to the right of the insertion point, and furthermore that Case 1 did not apply. Let n_{i-1} denote the node immediately to the left of the insertion point at level h, let n_i denote the newcomer, and let n_{i+1} denote the node immediately to the right of the insertion point. Without loss of generality, assume that $D_h(n_{i-1}) = \alpha$ and $D_h(n_{i+1}) = \beta$, where $\alpha, \beta \in [0, 2]$ and $\alpha \neq \beta$. Let $x = R_h(n_{i-1})$ and let $y = L_h(n_{i+1})$.

Inserting the new node n_i causes x and y to increase by 1. If $x < 5$ then inserting n_i does not violate the invariant that $x \leq 5$. Similarly, if $y < 5$ then inserting n_i does not violate the invariant that $y \leq 5$. Thus, if both $x < 5$ and $y < 5$ then we set $D_h(n_i)$ to be γ, the one remaining digit that is neither α nor β, and the invariants on x and y will continue to hold. However, if $x = 5$ or $y = 5$ then rotations are necessary in order to insert n_i while maintaining the invariant. Without loss of generality assume that $x = 5$. The h^{th} digits of the nodes from n_{i-1} to the right must have the following configuration:

$$\ldots \alpha \ \mathbf{?} \ \beta \ \gamma \ \beta \ \gamma \ \alpha \ldots$$

where **?** denotes the as yet undetermined value of $D_h(n_i)$. One may verify that the following operations:

(1) Rotate(n_i,n_{i+1})
(2) Rotate(n_{i+1},n_{i+2})
(3) Set $D_h(n_{i+2})$ to α
(4) Recursively insert n_{i+2} at level $h+1$

achieve the following legal configuration:

$$\ldots \alpha \ \beta \ \gamma \ \alpha \ \beta \ \gamma \ \alpha \ldots$$

The case in which $x = y = 5$ is also handled by this same sequence of operations.

Node insertion requires $O(1)$ time per level if no rotations are required, and requires $O(\log N)$ time per level if rotations are required. In total, node insertion requires $O(\log^2 N)$ time.

3.2 Node Departure

Node departure involves iteratively removing a node from all of its rings, from the highest level down to level 0. As with node insertions, we must ensure that this node's departure does not cause violations of the Deterministic SkipNet invariant at other nodes. We prove that at most two rotations per level suffice to maintain the invariant during node departure.

Consider the departure of a node from level $h-1$. By examining the h^{th} digits of the nearby nodes we can determine what rotations, if any, are needed to ensure that the invariant continues to hold. We first define the following notation: let n_i be the departing node, and let n_{i-1} and n_{i+1} be its immediate left and right neighbours respectively at level $h-1$. Also, let $\alpha = D_h(n_i)$, let $x = L_h(n_i)$ and $y = R_h(n_i)$.

Case 1: Suppose that $D_h(n_{i-1}) = D_h(n_{i+1}) = \beta \neq \alpha$. If γ is the remaining digit that is neither α nor β then we must have either $D_h(n_{i-2}) = \gamma$ or $D_h(n_{i+2}) = \gamma$. Without loss of generality, assume $D_h(n_{i+2}) = \gamma$. Our configuration is then

$$\ldots \beta\ \alpha\ \beta\ \gamma \ldots$$

where α corresponds to the departing node n_i. If $R_h(n_{i+1}) < 5$, one may verify that Rotation(n_i, n_{i+1}) followed by departure of n_i is sufficient to maintain the invariant. If $R_h(n_{i+1}) = 5$, the invariant is maintained by performing Rotation(n_{i+1}, n_{i+2}) followed by departure of n_i.

Case 2: Suppose that $D_h(n_{i-1}) = \beta \neq \alpha$, and $D_h(n_{i+1}) = \gamma, \alpha \neq \gamma \neq \beta$. If $x + y \leq 6$ then node n_i may depart without any rotations. Otherwise, assume without loss of generality that $x > 3$. To maintain the invariant, we perform Rotation(n_{i-1}, n_i), and if $R_h(n_{i-1}) = 5$ we additionally perform Rotation(n_i, n_{i+1}). These rotations are followed by departure of n_i.

In both cases, the node departure algorithm proceeds by iteratively removing node n_i from level $h-2$ and performing any necessary rotations, until $h-2 < 0$. As was the case with node insertion, node departure requires $O(\log N)$ time per level if rotations are required and $O(1)$ otherwise. In total, node departure requires $O(\log^2 N)$ time.

4 Conclusions

Deterministic SkipNet is a scalable overlay network that is both deterministic and incorporates locality. Abraham et al. [1] also present a mechanism for deterministic overlay construction but it is not clear whether their approach can yield overlays that incorporate locality. Deterministic SkipNet supports node insertions in $O(\log^2 N)$ time, node departure in $O(\log^2 N)$ time, searches in $O(\log N)$ time, and $O(\log N)$ routing pointers per node. In future work, we plan to investigate a generalization of Deterministic SkipNet where the numeric ID parameter k may take arbitrary values.

References

[1] I. Abraham, B. Awerbuch, Y. Azar, Y. Bartal, D. Malkhi, E. Pavlov, A generic scheme for building overlay networks in adversarial scenarios, in: International Parallel and Distributed Processing Symposium (IPDPS), 2003.

[2] J. Aspnes, G. Shah, Skip Graphs, in: 14th ACM-SIAM Symposium on Discrete Algorithms (SODA), 2003, pp. 384–393.

[3] N. J. A. Harvey, M. B. Jones, S. Saroiu, M. Theimer, A. Wolman, SkipNet: A Scalable Overlay Network with Practical Locality Properties, in: USENIX Symposium on Internet Technologies and Systems (USITS), 2003, pp. 113–126.

[4] J. I. Munro, T. Papadakis, R. Sedgewick, Deterministic Skip Lists, in: 3rd ACM-SIAM Symposium on Discrete Algorithms (SODA), 1992, pp. 367–375.

[5] W. Pugh, Skip Lists: A Probabilistic Alternative to Balanced Trees, in: Workshop on Algorithms and Data Structures (WADS), 1989, pp. 437–449.

[6] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Shenker, A Scalable Content-Addressable Network, in: Proceedings of ACM SIGCOMM, 2001, pp. 161–172.

[7] A. Rowstron, P. Druschel, Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems, in: IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), 2001, pp. 329–350.

[8] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, H. Balakrishnan, Chord: A scalable Peer-To-Peer lookup service for internet applications, in: Proceedings of ACM SIGCOMM, 2001, pp. 149–160.

[9] B. Y. Zhao, J. D. Kubiatowicz, A. D. Joseph, Tapestry: An Infrastructure for Fault-Resilient Wide-area Location and Routing, Tech. Rep. UCB//CSD-01-1141, U. C. Berkeley (Apr. 2001).