A Distributed Algorithm for Constructing a Generalization of de Bruijn Graphs

Nikhil Swamy*, Nikolaos Frangiadakis*, Konstantinos Bitsakos*
{nswamy,ntg,kbits}@cs.umd.edu

*Department of Computer Science
University of Maryland, College Park, MD 20742

Abstract

De Bruijn graphs possess many characteristics that make them a suitable choice for the topology of an overlay network. These include constant degree at each node, logarithmic diameter and a highly-regular topology that permits nodes to make strong assumptions about the global structure of the network.

We propose a distributed protocol that constructs an approximation of a de Bruijn graph in the presence of an arbitrary number of nodes. We show that the degree of each node is constant and that the diameter of the network is no worse than $2logN$, where N is the number of nodes. The cost of the join and the departure procedures are $O(logN)$ in the worst case. To the best of our knowledge, this is the first distributed protocol that provides such deterministic *guarantees.*

I. Introduction

De Bruijn graphs are a class of directed graphs that have been proposed as a near-optimal topology for a DHT [5]. The reasons for this include constant degree at each node of the graph which indicates low control overhead per node; a deep structure that allows each node to make strong assumptions about the behavior of other nodes; and a large bisection width when compared to other constant degree graphs. There have been attempts to construct peer-to-peer overlay networks according to a de Bruijn topology ([4], [3]), but these proposals provide only high probability guarantees of performance. Our principal contribution is a distributed protocol to construct *de Bruijn-like* graphs. Our protocol comes with the following deterministic guarantees: (1) the number of neighbors for each node in the network is no more than a constant; and (2) the diameter of the network is logarithmic in the number of nodes.

Pure de Bruijn graphs are defined over nodes that number d^k for some integers d and k. Each node in such graphs is assigned a unique identifier from the set of possible k-words from an alphabet of size d (typically, a binary alphabet is used; i.e $d = 2$.) The connectivity of a de Bruijn graph is determined by the identifier assigned to each node. Consider a node n with identifier $b_1b_2...b_k,\ b_i \in \{0, 1\}$. n has an out-edge to the nodes with identifier $b_2...b_k0$ and $b_2...b_k1$. This adjacency scheme, based on shifting the identifier strings associated with a node yields a simple prefix based routing policy. The result is a a $log(N)$ bound on the diameter of the graph. By construction, the degree of each node is constant.

The requirement that the number of nodes is precisely 2^k has made it difficult to use de Bruijn graphs in a practical setting. Our protocol, HALO, provides a generalization of the de Bruijn graph to the case where the number of nodes is not a power of two. We achieve this by partitioning the whole graph into regions that locally exhibit precise de Bruijn connectivity. The boundaries of these regions are populated by nodes that emulate the behavior of more than one node. This enables us to "glue together" each of the regions that together constitute the entire graph. The graph constructed retains the degree and diameter bounds as well as the routing scheme of de Bruijn graphs in the presence of an arbitrary number of nodes.

The paper is organized as follows. In Section II we present a high-level summary of our protocol, HALO. In Section III we describe the connectivity structure of the network and show how packets are routed through it. Section IV describes the procedure by which new nodes enter the network. Techniques for maintaining important network invariants are described in Section V. The preservation of these invariants suggests a procedure for permitting nodes to leave the network. This is described in Section VI. In Section VII we state and prove the theorems that provide deterministic guarantees on the behavior of the network. We conclude with a discussion of related work.

II. Overview

Our approach takes the view of the overlay network as a distributed hash table. We consider the space of hash-keys to be all binary strings of length exactly k. Each node in the network is assigned one or more labels from this space. The scheme for assigning labels to nodes maintains the invariant that the labels of all the nodes in the network forms an *universal prefix set*; that is, for every k-length binary string s there exists a *unique* node n such that a label associated with n shares a common prefix with s. We say that the label associated with n *covers* the string s. This invariant suggests a simple prefix-based scheme for routing packets across the network.

We use n and subscripted/superscripted varieties to denote network nodes. We use N to refer to the set of all nodes, or the size of that set depending on the context. The set of labels of a node n is denoted $lab(n)$. Each label is a binary string s of length k. A node n is *addressed* by the prefix of each label in $lab(n)$. That is, a node can have more than one address. Each node n is assigned a *level*, such that $level(n) \leq k$, that specifies the length of the prefix used to address n. For instance, given a node n with $level(n) = \ell$ and $lab(n) = \{b_1 b_2 \dots b_k, b'_1 b'_2 \dots b'_k\}$ for some binary digits b_i and b'_i, then n is addressed by the strings $\{b_1 \dots b_\ell, b'_1 \dots b'_\ell\}$. We write $b_1 b_2 \dots b_\ell \cdot *$ to refer to the set of all binary strings that begin with $b_1 b_2 \dots b_\ell$. Since the set of node labels forms a universal prefix set, we use write $b_1 b_2 \dots b_\ell \cdot *$ to refer to a label associated with a particular node as well as the set of labels in the hash-space *covered* by the label.

If the number of nodes in the network is 2^ℓ then it is straightforward to construct a pure de Bruijn graph by ensuring that the level of all nodes in the network is ℓ. To handle the more general setting of an arbitrary number of nodes we allow the level of nodes in network to vary. We attempt to maintain the invariant that nodes at a similar level remain adjacent to each other. For instance, given nodes n_1, n_2 and n_3 at levels ℓ, $\ell+1$ and $\ell+2$ respectively, we may permit n_1 and n_2 to be adjacent and n_2 and n_3 to be adjacent, but n_1 and n_3 may not be adjacent. Nodes at the same level can also be adjacent. This pattern of connectivity permits the network to be structured into regions of nodes that are of similar level. The local view of the network to a node at level ℓ is as if it were a de Bruijn graph of level ℓ.

Piecing together different regions of the network that are locally de Bruijn connected requires some care. To achieve this, certain nodes in the network have more than one label. By doing so, these nodes emulate the behavior of as many nodes as they have labels. For instance, a node might have labels $s_1 = b_1 \dots b_\ell 0 \cdot *$ and $s_2 = b_1 \dots b_\ell 1 \cdot *$ and emulates the behavior of two nodes at level $\ell + 1$ that would have labels s_1 and s_2. Note however that in this case s_1 and s_2 share a common prefix of length ℓ. Together s_1 and s_2 cover a set of hash-identifiers that is identical to that covered by the common prefix of s_1 and s_2. We use this insight to place nodes that emulate the behavior of more than one node on the boundary between regions of the network that are at levels that differ by one.

When a request to join the network is received at a node it is typically forwarded to the nearest node that is emulating the behavior of more than one node. Since emulation of a node's behavior is costly, this makes intuitive sense in that the nodes with the heaviest burden are permitted to reduce their load by sharing it with the joining node. It also has the effect of distributing the introduction of new nodes in the network and ensuring that the difference in levels across the entire network is kept small. Forwarding join requests in this manner requires that a node in the network keep track of the state of its neighbors. In the next section, we define precisely the state of a node and the structure of connectivity in the network.

III. States and Adjacencies

The state of a node n in the HALO network is a pair $S(n) = (\texttt{ST}, l)$ and is denoted $\texttt{ST}_l$ where l is the level of the node v. The state identifier ST is taken from the set $\{\texttt{IN}, \texttt{EM}, \texttt{AT}\}$. The state of a node is a function of both its own internal state as well as the state of its immediate neighbors in the network. The states (and their relation to node identifiers) are enumerated below.

1) $\texttt{IN}_0$ A node that wishes to join the HALO network is initially at level 0. It issues a join request to a bootstrap node.
2) $\texttt{EM}_l$ A node n in this state emulates the behavior of two nodes. Specifically, $lab(n) = \{b_1 \dots b_{\ell-1} 0 \cdot *, b_1 \dots b_{\ell-1} 1 \cdot *\}$. Note that the length of both labels in $lab(n)$ is ℓ and both labels share a common prefix of length $\ell - 1$
3) $\texttt{AT}_l$ A node n the state $\texttt{AT}_l$ is said to be *atomic* — it has only a single label. i.e. $lab(n) = \{b_1 \dots b_\ell \cdot *\}$.

The state of a node n is in part determined by the state of nodes n_i that are adjacent to n. The adjacencies themselves are defined purely in terms of the identifiers of a node. We define two types of adjacencies that are maintained at n — $\texttt{SUCC}(n)$ and $\texttt{SIB}(n)$.

$$\texttt{SUCC}(n) \equiv \{n' \in N \mid \bigvee \begin{matrix} b_2 \dots b_\ell 0 \cdot * \in lab(n') \\ b_2 \dots b_\ell 1 \cdot * \in lab(n') \end{matrix} \}$$
$$\text{where} \quad b_1 \dots b_\ell \cdot * \in lab(n)$$

$\texttt{SUCC}(n)$ are the normal de Bruijn successors of a node n at level ℓ. These are the edges that are used to route packets in the network using a prefix-based routing procedure. For instance, the path taken in the network from a node with

label $b_1 \dots b_k$ to a node with label $b'_1 \dots b'_k$ is through the nodes with the labels below[1]:

$$b_2 \dots b_k b'_1, \quad b_3 \dots b_k b'_1 b'_2, \quad \dots, \quad b_k b'_1 \dots b'_{k-1}$$

The other type of edges maintained at a node are used to handle node failures. The structure of the connectivity of "sibling" edges is given below.

$$\begin{aligned} &\texttt{SIB}(n) \equiv \{n' \in N \mid b_1 \dots \bar{b_l} 0{\cdot}* \in lab(n')\} \\ &\text{where} \\ &\quad \bar{0} \equiv 1 \qquad \bar{1} \equiv 0 \\ &\quad lab(n) = \{s\} \Rightarrow s = b_1 \dots b_l{\cdot}* \\ &\quad lab(n) = \{s_1, s_2\} \Rightarrow \bigwedge \begin{array}{l} s_1 = b_1 \dots b_l 0{\cdot}* \\ s_2 = b_1 \dots b_l 1{\cdot}* \end{array} \end{aligned}$$

The manner in which $\texttt{SIB}(n)$ is used to handle node failures will be presented in Section VI. It is worth noting that the SIB() relation is not necessarily symmetric. If n is not at the same level as $\texttt{SIB}(n)$ then $\texttt{SIB}(\texttt{SIB}(n)) \neq n$.

In determining the state of a node we often consider the entire neighborhood of the node. We use $\texttt{NBRS}(n)$ to refer to the set of nodes that n refers to, as well as nodes that refer to n using both $\texttt{SUCC}(n)$ and $\texttt{SIB}(n)$ edges.

IV. The Join Procedure

We make use of join redirections to ensure than a new node joins at an "appropriate" part of the graph. Typically the redirection is limited to a radius of two hops from the site of the original join request, but in the worst case this might require upto $\log N$ hops.

A node $n_{\texttt{IN}}$ that wishes to join the network uses an out-of-band method to contact an arbitrary node n in the network. Initially, $S(n_{\texttt{IN}}) = \texttt{IN}_0$. The node n that receives the request may chose to satisfy the request itself, or it may forward to join request to some other node in its neighborhood. The rules that determine the manner in which a join request is forwarded are presented in Table I.

Here $S^t(n)$ represents the state of the node that receives the join request at time t [2]; $lab^t(n)$ represents its labels at that time. If the condition in the third column is satisfied for *some* node in the neighborhood of n then the join is forwarded to that node and the state of n remains unchanged. In case the join is not forwarded, the node n computes a fresh pair of identifiers and adjacencies for itself and for $n_{\texttt{IN}}$. At the next time step, the node n then enters into state $S^{t+1}(u)$ and presents $n_{\texttt{IN}}$ with its identifier, adjacencies, and state $S^{t+1}(n_{\texttt{IN}})$.

It is straightforward for n to compute the new identifiers and adjacencies for itself and $n_{\texttt{IN}}$. If $S^t(n) = \texttt{EM}_\ell$ then n already emulates the behavior of two nodes; in this case n simply hands over half of its state to the new node. If $S^t(n) = \texttt{AT}_l$ and $lab^t(n) = \{b_1 b_2 \dots b_l \cdot *\}$ then n gives itself the new identifier $b_1 b_2 \dots b_\ell 0{\cdot}*$ and gives $n_{\texttt{IN}}$ $b_1 b_2 \dots b_l 1{\cdot}*$ thereby splitting the space of hash identifiers exactly in half. We show in Section VII that the resulting $\texttt{SUCC}(\cdot)$ sets for both n and $n_{\texttt{IN}}$ is contained in the original $\texttt{SUCC}(n)$. After the join $\texttt{SIB}(n) = n_{\texttt{IN}}$ and $\texttt{SIB}(n_{\texttt{IN}}) = n$.

V. State Consistency

To be able to guarantee logarithmic diameter with constant degree at each node we require certain state invariants to be maintained. These invariants are listed below.

(A1) If an edge (n_1, n_2) is in the network then $|level(n_1) - level(n_2)| \leq 1$.

(A2) An $\texttt{EM}_\ell$ node has at least one neighbor n such that $S(n) \in \{\texttt{AT}_\ell, \texttt{EM}_{\ell+1}\}$.

The reasons for the exact choice of these invariants will become evident in Section VII. Intuitively, (A1) maintains the property that the local view of the network to any particular node is de Bruijn like by ensuring that only nodes of a similar level are adjacent. Invariant (A2) ensures that nodes do not unnecessarily emulate the behavior of more than one node.

To enforce these invariants, we require a node n to notify all its neighbors each time its state changes. Consider the case where a node n' receives a notification from a node n of a change in the state of n. The node n' responds to this notification in two steps. First, in an attempt to maintain invariant (A1), the state of n' may change due to the change in n's state alone. Next, to maintain (A2) n' may examine the state of all its neighbors and, if necessary, change its state accordingly. If n' state changes it issues a notification to all its neighbors. We show that this chain of state change notifications does not extend beyond a radius of two hops from the node that started the notification.

The rule below indicates the action taken at node n' if it receives a notification from node n.

$$\text{(R1)} \quad \left(\begin{array}{l} S^t(n) = \texttt{AT}_\ell \quad \wedge \quad S^t(n') = \texttt{AT}_{l-1} \quad \wedge \\ lab^t(n') = \{b_1 \dots b_{\ell-1} \cdot *\}) \end{array} \right) \quad \Rightarrow$$

$$\left(\begin{array}{l} S^{t+1}(n) = \texttt{AT}_\ell \quad \wedge \quad (S^{t+1}(n') = \texttt{EM}_\ell \quad \wedge \\ lab^{t+1}(n') = \{ \ b_1 \dots b_{l-1} 0{\cdot}*, b_1 \dots b_{l-1} 1{\cdot}* \ \}) \end{array} \right)$$

For all other combinations of states of n and n' no action is taken at n'. This rule handles the following situation. At time $t-1$, nodes n and n' are at the same level $\ell - 1$. At time t the level of n increases, due to say the arrival

[1] It is possible to optimize the routing scheme to take advantage of similarities between the labels of the source and destination nodes. We exclude this to keep the presentation simple.

[2] For the purpose of describing our protocol an informal treatment of time is sufficient. We adopt a discrete approach, where a single time unit passes when a node receives or sends a control message. With this definition time is a local (i.e. node-specific) property i.e., each node keeps track of its own time.

$S^t(n)$	$lab^t(n)$	Fwd if $\exists n' \in \texttt{NBRS}(n)$ s.t	Else			
			$S^{t+1}(n)$	$lab^{t+1}(n)$	$S^{t+1}(n_{\texttt{IN}})$	$lab^{t+1}(n_{\texttt{IN}})$
$\texttt{AT}_\ell$	$b_1 \dots b_\ell \cdot *$	$S(n') = \texttt{EM}_\ell$	$\texttt{AT}_{l+1}$	$b_1 \dots b_\ell 0 \cdot *$	$\texttt{AT}_{l+1}$	$b_1 \dots b_\ell 1 \cdot *$
$\texttt{EM}_\ell$	$b_1 \dots b_\ell \cdot *$ $b_1 \dots \bar{b_\ell} \cdot *$	$S(n') = \texttt{EM}_{\ell-1}$	$\texttt{AT}_l$	$b_1 \dots b_\ell \cdot *$	$\texttt{AT}_l$	$b_1 \dots \bar{b_\ell} \cdot *$

TABLE I. The Join Procedure

of a new node in the proximity of n. The change in level causes the region of adjacent nodes at level $\ell - 1$ to be split. At this point, the node n' responds by emulating two nodes so as to "glue" the two regions together.

In order to satisfy the invariant (A2) we require that a node n periodically examine the state of *all its neighbors* and apply the rule below. As previously, if the state of n changes as a result of this rule, n notifies its neighbors of the change.

$$\text{(R2)} \quad \left(\begin{array}{l} S^t(n) = \texttt{EM}_\ell \quad \wedge \\ lab^t(n) = \{b_1 \dots b_\ell \cdot *, b_1 \dots \bar{b_\ell} \cdot *\} \\ \forall n' \in \texttt{NBRS}(n). S^t(n') \in \{\texttt{EM}_\ell, \texttt{AT}_{\ell-1}\} \end{array} \right) \Rightarrow \left(\begin{array}{l} S^{t+1}(n) = \texttt{AT}_{\ell-1} \\ lab^{t+1}(n) = \{b_1 \dots b_{\ell-1} \cdot *\} \end{array} \right)$$

In Section VII we show that this procedure for maintaining state consistency prevents the network from ever entering an erroneous state, and is sufficient to guarantee the two invariants stated at the start of this section.

$\longrightarrow$	$\texttt{EM}_{l-1}$	$\texttt{EM}_l$	$\texttt{EM}_{l+1}$	$\texttt{AT}_{l-1}$	$\texttt{AT}_l$	$\texttt{AT}_{l+1}$
$\texttt{AT}_l$	X	R	S	X	S	X
$\texttt{EM}_l$	R	S	S	S	S	X

TABLE II. The valid states at the two nodes of an edge in the network.

The result of this state maintenance procedures, namely the permitted states of interconnected nodes, are summarized in Table II. For a directed edge (n_1, n_2) in the network, the table shows the possible states of n_1 and n_2. The state of n_1, the starting point of the edge, appears in the left-most column. The state of n_2 appears in the top row. A table entry "X" indicates that such an edge is not permitted in the network; an "R" indicates that (n_1, n_2) can only be a successor or "routing" edge; an "S" indicates that (n_1, n_2) may be either a successor or a sibling edge.

VI. The Departure Procedure

When a node n wishes to leave the network we must find another node n_1 that can take its place. If such a node n_1 is identified we are faced with the obvious problem of finding, in turn, a node to take the place of n_1, since it will fulfill the role of the departed node. The task of the departure protocol then, is to identify a node n_1 that has a sibling n_2 such that n_2 can take over the hash-space assigned to n_1 thus freeing n_1 to take over the hash-space of n, the departing node.

The inductive nature of this problem is captured by the recursive rules that define the node departure algorithm in Table III. The presentation of this algorithm assumes that the node n departs from the network gracefully. In the event of a catastrophic failure the sibling of n detects the failure and executes the departure procedure on behalf of n.

The correctness of this protocol depends on the following partial order defined over the states of nodes.

$$\texttt{EM}_\ell < \texttt{AT}_\ell < \texttt{EM}_{\ell+1}$$

In (case 1a) of Table III the search for the node has arrived at a node n_1 in state $\texttt{AT}_\ell$ which has a sibling n_2 which is also in state $\texttt{AT}_\ell$. $n_1 \sqcap n_2$ is the node that covers the label obtained by computing the maximum common prefix of $lab(n_1)$ and $lab(n_2)$. This operation is well defined since both nodes are in state $\texttt{AT}_\ell$ and thus have only a single identifier each. Node n_3 represents the sibling of the node thus computed. If it can be ensured that n_3 is at the same level as n_1 and n_2, then we can cause n_2 to emulate the behavior of both n_1 and n_2 without introducing any pointers that violate the invariants of Section V. If n_3 happens to be at a deeper level, thus exceeding n_1 in the partial order, we forward the search procedure to n_3.

In cases 1b, 2a, and 2b n_2 already exceeds n_1 in the partial order. In each of these cases we immediately forward the search to node n_2.

Finally, in case 2c, even though the sibling of n_1 precedes n_1 in the partial order, we are guaranteed by invariant (A2) of Section V that a node n_3 must exist in the neighborhood of n_1 that succeeds it in the partial order. We forward the search to any such node.

Note that in each case the search is forwarded to a node that is further along in the partial order. This guarantees the termination of this process.

VII. Properties of HALO

An important property of the HALO network is illustrated in Figure 1. This figure is a schematic illustration of the HALO graph – hence the name. The graph is partitioned such that all edges in the graph are only

$S(n_1)$	$S(n_2 = \texttt{SIB}(n_1))$	Action
$\texttt{AT}_\ell$	$\texttt{AT}_\ell$ (case 1a)	let $n_3 = \texttt{SIB}(n_1 \sqcap n_2)$ if $level(n_3) \leq \ell$ then $id(n_2) := lab(n_1) \cup lab(n_2)$; $S(n_2) = \texttt{EM}_\ell$ found n_1 else forward to n_3 (case 1 or 2)
	$\texttt{EM}_\ell$ (case 1b)	forward to n_2 (case 2)
$\texttt{EM}_\ell$	$\texttt{EM}_{\ell+1}$ (case 2a)	forward to n_2 (case 2)
	$\texttt{AT}_\ell$ (case 2b)	forward to n_2 (case 1)
	$\texttt{EM}_\ell$ (case 2c) $\texttt{AT}_{\ell-1}$	$\exists \texttt{n}_3 \in \texttt{NBRS}(\texttt{n}_1) \mid \texttt{S}(\texttt{n}_3) \in \{\texttt{AT}_\ell, \texttt{EM}_{\ell+1}\}$ forward to $\texttt{n}_3$ (case 1 or 2)
where $n_1 \sqcap n_2$ is a node n with the lab(n) contains the maximum common prefix of $lab(n_1)$ and $lab(n_2)$.		

TABLE III. The Departure Procedure

between nodes of a similar level. The states in the network are maintained such that nodes in the $\texttt{EM}_\ell$ state act as a buffer area between nodes at level ℓ and those at levels $\ell - 1$. The following lemma shows that with sequential joins and departures the difference in levels between the endpoints of an edge is at most 1. Or, that edges in the HALO graph do not extend across adjacent levels in Figure 1.

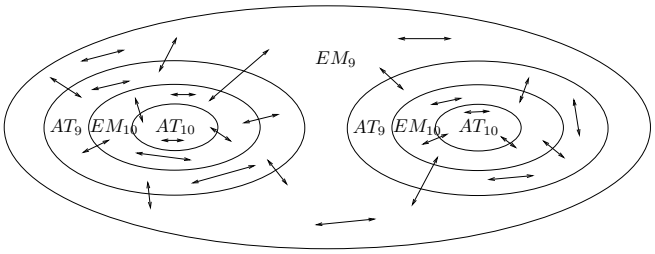

Fig. 1. A schematic illustration of a HALO network. Nodes of various levels are arranged in "concentric" rings. This figure shows two local regions with $\texttt{AT}_{10}$ nodes surrounded by $\texttt{EM}_{10}$ and $\texttt{AT}_9$ nodes. All these regions occur within a region of $\texttt{EM}_9$ nodes. The arrowed line segments indicate edges between nodes that reside in each of the regions.

Lemma 7.1 (No Steep Edges): If (n_1, n_2) is an edge in the HALO graph, then either $level(n_1) = level(n_2)$, or $S(n_1) = \texttt{EM}_\ell$ and $S(n_2) \in \{\texttt{AT}_{\ell-1}, \texttt{EM}_{l+1}, \texttt{EM}_{l-1}\}$.

Proof: The proof is by induction on the number of vertices in the graph. Clearly with a single node in the graph there is no edge that violates the above constraint. Let us assume the property holds for a graph with N nodes. From this state the graph can go to a new state only if a new node joins the network or an existing node leaves it. We examine the two cases separately.

Case 1: A node $n_{\texttt{new}}$ joins the network

When a node $n_{\texttt{new}}$ joins the graph its request is received by an existing node of the network. The request might be forwarded to other nodes, until it at time t reaches node $n \in N$ which processes it.

1) $\texttt{S}^t(\texttt{n}) = \texttt{EM}_\ell$: If n had a neighbor n' in state $\texttt{EM}_{\ell-1}$ the join request would be forwarded to n'. Thus we are guaranteed that all EM neighbors of u are at a level no less than ℓ.
   Let $lab^t(n) = \{b_1 \dots b_{\ell-1}0 \cdot *, b_1 \dots b_{\ell-1}1 \cdot *\}$. After the join $lab^{t+1}(n) = \{b_1 \dots b_{\ell-1}0 \cdot *\}$ and $lab^{t+1}(n_{\texttt{new}}) = \{b_1 \dots b_{l-1}1 \cdot *\}$. Since no new ids have been introduced by the join, we trivially have that *the set of the successor edges of n before the join is equal to the union of the sets of the successor edges of n and $n_{\texttt{new}}$ after the join.*
   After the join $S^{t+1}(n) = S^{t+1}(n_{\texttt{new}}) = \texttt{AT}_\ell$. It remains to be shown that none of the edges introduced violate the property. We handle each of the cases for nodes n' adjacent to n and $n_{\texttt{new}}$ using $\texttt{SUCC}(\cdot)$ edges only — i.e $\{(n', n), (n', n_{\texttt{new}})\} \subseteq \texttt{SUCC}(n')$ or $\{(n, n'), (n_{\texttt{new}}, n')\} \subseteq \texttt{SUCC}(n) \cup \texttt{SUCC}(n_{\texttt{new}})$.
   a) $\texttt{S}^{t+1}(\texttt{n}') = \texttt{AT}_\ell$: Since $S^{t+1}(n) = S^{t+1}(n_{\texttt{new}}) = S^{t+1}(n')$ the property is satisfied.
   b) $\texttt{S}^{t+1}(\texttt{n}') = \texttt{EM}_\ell$ Since the level of n' is the same as the level of n and $n_{\texttt{new}}$ the property is satisfied.
   c) $\texttt{S}^{t+1}(\texttt{n}') = \texttt{EM}_{\ell+1}$ The level of n' is only one greater than the level of n and $n_{\texttt{new}}$ and n' is in the state EM; thus the property is satisfied.
   d) $\texttt{S}^{t+1}(\texttt{n}') = \texttt{AT}_{\ell-1}$ The node n' receives a notification from either n or $n_{\texttt{new}}$ and, according to rule (R1), transitions to state $S^{t+2\prime}(n') = \texttt{EM}_\ell$ thus confining the edge to a single level.
2) $\texttt{S}^t(\texttt{n}) = \texttt{AT}_\ell$: If $\exists n' \in \texttt{NBRS}(n)$ s.t. $S(n') = \texttt{EM}_\ell$ then the join request is forwarded to n'. Thus we need only consider the case when all neighbors of n are either $\texttt{AT}_\ell$ or $\texttt{EM}_{\ell+1}$.
   If $lab(n) = \{b_1 \dots b_l \cdot *\}$ after the join we have $lab^{t+1}(n) = \{b_1 \dots b_l 0 *\}$ and $(* n_{\texttt{new}}) = \{b_1 \dots b_l 1 \cdot *\}$. The neighbors of n according to the $\texttt{SUCC}(\cdot)$ relation at time t are nodes with labels in

$$L = \{ \begin{array}{ll} b_2 \ldots b_l 0{\cdot}*, & b_2 \ldots b_l 1{\cdot}*, \\ 0 b_1 \ldots b_{l-1}{\cdot}*, & 1 b_1 \ldots b_{l-1}{\cdot}* \end{array} \}$$

As in the previous case, after the join, the neighbors of n and $n_{\texttt{new}}$ according to the $\texttt{SUCC}(\cdot)$ relation after join are those with labels belonging to the set L above. Thus, we need only consider nodes that were in the original neighborhood of n.
$S^{t+1}(n_{\texttt{new}}) = S^{t+1}(n) = \texttt{AT}_{\ell+1}$. All the nodes $n' \in \texttt{NBRS}()t+1n \cup \texttt{NBRS}()t+1(n_{\texttt{new}})$ must have $S(n') \in \{\texttt{AT}_\ell, \texttt{EM}_{\ell+1}\}$. For nodes in state $\texttt{EM}_{l+1}$ the edges to n and $n_{\texttt{new}}$ no longer cross a level boundary. Nodes in state n' such that$S^t(n') = \texttt{AT}_\ell$ receive a notification that by rule (R1) causes a transition to $S^{t+1}(\texttt{EM}_{l+1})$ which then causes the edge to be contained within a level.

In both cases we need to also consider sibling edges. After the join procedure we have $\texttt{SIB}()t+1n = \texttt{SIB}()t+1n_{\texttt{new}}$ and $level(n) = level(n_{\texttt{new}})$; so the newly introduced sibling edges do not violate the hypothesis. A node $n' \in \texttt{NBRS}()t+1n \mid n = \texttt{SIB}()t+1n'$ (recall that the $\texttt{SIB}(\cdot)$ relation is not always symmetric) receives a state change notification just as any other node in $\texttt{NBRS}()t+1n$ and will make the necessary state transition using rule (R1). Finally, it is worth noting that the state change notifications are confined to a radius of 2 from the location of the join. This can be seen by noting that a notification from an $\texttt{EM}_\ell$ node causes no state change in the recipient of the message.

Case 2: A node n_d departs the network

When a node n_d leaves the network, the departure protocol discovers at time t a node n that will take the place of n_d in the network. The state of n at time t, $S^t(n)$ is always $\texttt{AT}_\ell$, and n has a sibling n' that is also in state $\texttt{AT}_\ell$. On completion of the departure procedure, the state of n' is $S^{t+1}(n') = \texttt{EM}_\ell$. We consider the newly introduced successor and sibling edges separately. Since n takes the place of n_d, it adjacencies at the end of the departure protocol are the same as those of n_d prior to the departure. Hence all adjacencies of n after the departure protocol is complete are legal.

After the departure is complete, n' takes on all adjacencies of n that were present prior to the departure of n_d. The set of non-sibling adjacencies of n' after the departure are the non-sibling edges in $\texttt{NBRS}()t(n) \cup \texttt{NBRS}()t(n')$. Since the level of n' did not change no new edge violates the property.

For sibling edges, the new sibling of n', $n'' = \texttt{SIB}()t+1(n')$ is precisely the node computed by $\texttt{SIB}(()u^v)$, and its level is is checked to ensure that it is no greater than ℓ. $level(n'')$ can in fact be no less than $\ell - 1$, since otherwise $labn''$ would contain a label that with a prefix of length $\ell - 1$ that was also a prefix of a label in $lab(n')$. ■

Corollary 7.2 (Degree Bound): The degree of a node is, in the worst case, 8.

Proof: Clearly, nodes n in state $\texttt{EM}_\ell$ are the only nodes with degree greater than 4. We proceed by induction on the number of nodes.

If the node n makes the state transition $EXT_{\ell-1} \rightarrow \texttt{EM}_\ell$, then there must have been a join of the node $n_{\texttt{new}}$ at $n' \in NBRS(n)$. After the join n is at worst adjacent to both n' and $n_{\texttt{new}}$, thus at most increasing its degree to 5.

If, however, n is already in the state $\texttt{EM}_\ell$ and a join occurs at $n' \in NBRS(n)$. The node n' must be in state $S(n') = \texttt{EM}_\ell$ or $\texttt{EM}_{\ell-1}$. n' cannot be in state $\texttt{AT}_\ell$ since otherwise the join would have been forwarded to n, nor can n' be in state $\texttt{EM}_{\ell+1}$ since once again the join would have been forwarded to n.

By the hypothesis, the degree of n is at most 8, implying $lab(n) = \{b_0 \ldots b_{\ell-1}0{\cdot}*, b_0 \ldots b_{\ell-1}1{\cdot}*\}$. Thus,

$$\texttt{NBRS}(n) = \{ \begin{array}{ll} b_1 \ldots b_{\ell-1}00{\cdot}*, & b_1 \ldots b_{\ell-1}01{\cdot}*, \\ b_1 \ldots b_{\ell-1}10{\cdot}*, & b_1 \ldots b_{\ell-1}11{\cdot}*, \\ 0 b_1 \ldots b_{\ell-1}0{\cdot}*, & 0 b_1 \ldots b_{\ell-1}1{\cdot}*, \\ 1 b_1 \ldots b_{\ell-1}0{\cdot}*, & 1 b_1 \ldots b_{\ell-1}1{\cdot}* \end{array} \}$$

But since $n' \in \texttt{NBRS}(()n)$ and $S(n') = \texttt{EM}_\ell$ then $id(n')$ is a prefix cover of at least 2 elements of $\texttt{NBRS}(()n)$. Thus the degree of n can be no more than 7. After the join n can at worst become adjacent to both n' and $n_{\texttt{new}}$, further increasing its degree only by 1, to 8.

In other words, in the worst case, a node emulates no more than two nodes. ■

Lemma 7.3 (Level Inequality): Let ℓ_{min} and ℓ_{max} denote the lowest and highest level of a node in the graph. $\ell_{min} - 1 \leq logN \leq \ell_{max}$, where N is the number of nodes. *Proof:* We first note that the $\ell_{min} \leq logN$. This can be observed as follows: Let λ denote the maximum length of a identifier. Since the set of nodes in the HALO graph form a universal prefix set we require that the entire space of identifiers 2^λ be covered by the identifiers of all the nodes in the graph. A node at level ℓ covers $2^{\lambda-l}$ elements of the identifier space. Therefore, a conservative upper bound [3] on the proportion of the space covered by a single node is $2^{\lambda-\ell_{min}+1}$. Thus we require $N \cdot 2^{\lambda-\ell_{min}+1} = 2^\lambda$ to satisfy the universal prefix property. Or, $\ell_{min} \leq logN + 1$.

To bound ℓ_{max}, we note that $2^{\lambda-\ell_{max}}$ is a lower bound on the number of identifiers covered by a node. Since there is no redundancy in the universal prefix set, we require $N \cdot 2^{\lambda-\ell_{max}} \leq 2^\lambda$. Thus $logN \leq \ell_{max}$. This argument also shows that $\ell_{max} = logN$ implies $\ell_{max} = l_{min}$. ■

[3] Nodes in state $\texttt{EM}_\ell$ cover q$2 \cdot 2^{\lambda-l}$ identifiers

Lemma 7.4 (Tree Edges): The out-degree of all but two internal nodes in the shortest path tree [4] is at least 2. *Proof:* Let the root of the shortest path tree be n_r and $r_1 \dots r_{\ell_1} * \in lab(n_r)$. Consider a node n with $b_1 \dots b_{\ell_2} * \in lab(n)$. First of all, there can be no path from r to n of length less than $|\ell_1 - \ell_2|$. This follows directly from the No Steep Edge lemma. Without loss of generality, assume the successors of n are n_1 and n_2 with labels that cover $b_2 \dots b_{\ell_2} 0 \cdot *$ and $b_2 \dots b_{\ell_2} 1 \cdot *$, respectively. The node n_1 need not be at the same level as n_2 but by the No Steep Edges lemma, and since both n_1 and n_2 are adjacent to n, the levels of n_1 and n_2 may differ by at most 2. We will demonstrate that either both edges $\{(n, n_1), (n, n_2)\}$ are tree edges, or neither edge is a tree edge. ■

We define $overlap(s_1, s_2)$ be the length of the longest prefix of of the string s_1 that is also a suffix of the string s_2. For instance, if $s_1 = 111000$ and $s_2 = 000111$, then $overlap(s_1, s_2) = 3$. Let l_r, l, l_1, l_2 denote the labels of r, n, n_1 and n_2 respectively. Since l_1 and l_2 are identical up to bit position ℓ_2, and the level of n_1 is at most two greater than the length of n_2, $i = overlap(l_1, l)$ and $j = overlap(l_2, l)$ differ by at most 3. The edge (n, n_1) is a tree edge iff $b_1 = b_{\ell_1 - i}$ [5]. Similarly for (n, n_2).

Without loss of generality let's assume that $i \geq j$. If we want only one edge in the set $\{(n, n_1), (n, n_2)\}$ to be a tree edge, we require the following:

(T1) $r_{l_1 - i} \neq r_{l_q - j}$. This is necessary since we want b_1 to match only one of these values, and hence contribute only a single edge to the tree.

(T2) $overlap(l_2, l_r) = \ell_2 - 1$. Since n_1 and n_2 share the first $\ell_2 - 1$ bits, for condition (T1) to be satisfied at least $\ell_2 - 1$ bits must be in the overlap for both l_1 and l_2 with l_r.

(T3) $r_{\ell_1 - j} \dots r_{\ell_1}$ to be a prefix of $r_{\ell_1 - i} \dots r_{\ell_1}$. This is required since the l_1 and l_2 differ only in bit position ℓ_2, therefore the matched suffixes of l_r must also be prefixes of each other. This condition also implies that the last i bits of l_r are identical.

For these conditions to be true simultaneously, the suffix of l_r that is $i + 1$ bits long must either be $100 \dots$ or $011 \dots$. For any value of i, there is exactly one possible pair of nodes n_1 and n_2 that satisfy the remainder of the properties. In particular, for some i, pick $i \geq j \geq i - 2$; then $l_2 = 0^j 1$ and l_1 is any one element from $L = \{0^j 0, 0^j 00, 0^j 01, 0^j 000, 0^j 001, 0^j 010, 0^j 011\}$, where 0^j represents a string of j zeroes). A similar case exists for $l^2 = 1^j 0$. This is true by the No Steep Edges lemma which guarantees that nodes two hops from each other can at most be two levels apart; and by the universal prefix property which disallows l_2 to be a prefix of l_1. Furthermore, if there exists more than one node with labels from L adjacent to n then by the above argument it is guaranteed that both those edges will be tree edges. Thus, in the entire graph, there exists only a single pair of nodes that have out-degree of 1 in the shortest path tree. All other internal nodes have out-degree 2.

Lemma 7.5 (Tree Depth): Let r be a node at the deepest level ℓ_{max}, with $r_1 \dots r_{\ell_{max}} \cdot * \in lab(r)$. The maximum depth of the shortest path tree rooted at r is $O(logN)$, where N is the number of nodes. *Proof:* First, if $\ell_{max} = logN$, by Lemma 7.3 we trivially have that there is only one level, and the HALO graph is a pure De Bruijn graph. The diameter of a De Bruijn graph is $logN$ and the depth of the shortest path tree is bounded by the diameter. We consider below the case where $\ell_{max} > logN$.

Consider a node n with $b_1 \dots b_\ell \cdot * \in lab(n)$ with successors n_1 and n_2 with labels $b_2 \dots b_\ell 0 *$ and $b_2 \dots b_\ell 1 *$, respectively. By the Lemma 7.4 we have that either (n, n_1) and (n, n_2) are both tree edges, or neither. We refine the conditions under which (n, n_1) and (n, n_2) are tree edges.

Once again, let l_r, l, l_1, l_2 denote the labels of r, n, n_1, n_2 respectively. If the $i = overlap(l, l_r) \geq 2$ then $l_2 = r_1 \dots b_1 b_2 \dots u_i \cdot *$. Thus $overlap(l_1, l_r)$ and $overlap(l_2, l_r)$ are both precisely $i - 1$ [6] Both edges (n, n_1) and (n, n_2) are tree edges since l has a greater overlap with l_r, and b_1 matches $r_{\ell_{max} - i}$, the necessary bit position in l_r. We also have that the distance from r to u is $\ell_{max} - i$. Since $logN < \ell_{max}$, we have all nodes within $(logN - 1)$ distance from the root r are internal nodes in the tree.

Since the degree of internal nodes in the tree is 2, and with $logN - 1$ levels of the tree as internal nodes, the shallowest leaf may only be at distance $logN$. Since we have at least $\frac{N}{2}$ internal nodes, this bounds the depth of the deepest leaf to be less than $logN + 3$. ■

Lemma 7.6 (Level Bound): The number of levels in the HALO graph is $O(logN)$ in the worst case, where N is number of nodes in the graph. *Proof:* We have from the Tree Depth lemma that the depth of the shortest path tree rooted at the deepest node is $O(logN)$. From the No Steep Edges lemma, even if we assume conservatively that every tree edge crosses a level boundary, $\ell_{max} - \ell_{min} = O(logN)$. ■

Theorem 7.7 (Diameter Bound): The diameter of the HALO graph with N nodes is $O(logN)$. *Proof:* We trivially have that the diameter of the network is at most ℓ_{max}. This can be observed by recalling the procedure used for routing in the De Bruijn network – it is always

[4] We only consider successor edges in the shortest path tree

[5] This statement follows directly from the de Bruijn routing algorithm. For instance, if the label of the root node $l_r = 000111 \cdot *$, and the label of n, $l = 111000 \cdot *$, then for (n, n_1) to be a tree edge, we must have $l_1 = 011100$

[6] There are exactly two nodes in the graph for which this is not true. When r *ends* with the sequence 101010 or 01010, and u, v *begin* with the same sequence. By an argument identical to that in the *Tree Edge* lemma these cases are rendered irrelevant.

possible to reach n_1 from n_2 by following edges that shift in $lab(n_1)$ one bit at a time. Since we never have to shift more that ℓ_{max} bits the diameter of the graph is less than ℓ_{max}.

By the Level Inequality lemma we have $\ell_{min} - 1 \leq logN \leq \ell_{max}$, and by the Level Bound we have $\ell_{max} - \ell_{min} = O(logN)$. Thus the maximum diameter $l_{max} = O(logN)$. ■

VIII. Related Work

Since their conceptualization in mid-40's [1], de Bruijn graphs have attracted significant attention in graph theory and related areas such as communication networks ([8],[9]). Especially their elegant routing schema has been widely studied ([6], [2], [7], [9]). With the introduction of Peer-to-Peer computing in the last decade and the on going research on Distributed Hash Tables (DHTs), interest in network architectures and their characteristics has been rekindled. To the best of our knowledge, two de Bruijn-based schemas for building DHTs have been proposed, *Koorde* ([4]) and *D2B* ([3]).

Koorde is an adaptation of the Chord protocol ([10]) for use with de Bruijn graphs. The de Bruijn edges are only used for routing, and for the remaining operations (namely node joins and departures) the Chord algorithms are used. Furthermore, only a probabilistic bound for logarithmic data lookup ($O(logN)$) is provided. *D2B*, on the other hand, attempts to organize its nodes such that they form a de Bruijn network. As a result, D2B can guarantee only with high probability that the out degree is $O(1)$. D2B, as well as our protocol, modifies the node identifiers by redirecting joins appropriately, while Koorde doesn't. In general, we believe that our approach is closer to D2B than Koorde. Nevertheless, our schema provides deterministic guarantees of $O(1)$ edges per node and $O(logN)$ diameter.

References

[1] N. De Bruijn. A combinatorial problem. In *Proc.* Koninklijke Nederlandse Akademie van Wetenschapper, volume 49, pages 758–764, 1946.

[2] Abdol-Hossein Esfahanian and S. Louis Hakimi. Fault-tolerant routing in debruijn communication networks. In *IEEE Transactions on Computers 34*, volume 9, pages 777–788, 1985.

[3] P. Fraigniaud and P. Gauron. The content-addressable network d2b. Technical Report 1349, CNRS Universie de Paris Sud, January 2003.

[4] M. Frans Kaashoek and David R. Karger. Koorde: A simple degree-optimal distributed hash table. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03)*, 2003.

[5] Dmitri Loguinov, Anuj Kumar, Vivek Rai, and Sai Ganesh. Graph-theoretic analysis of structured peer-to-peer systems: routing distances and fault resilience. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 395–406. ACM Press, 2003.

[6] S. Perennes. Broadcasting and gossiping on de bruijn shuffle exchange and similar networks. Technical Report 93-53, INRIA, Sophia-Antipolis, October 1993.

[7] M. R. Samatham and D. K. Pradhan. The de bruijn multiprocessor network: A versatile parallel processing and sorting network for vlsi. *IEEE Trans. Comput.*, 38(4):567–581, 1989.

[8] M. L. Schlumberger. *De bruijn communications networks.* PhD thesis, Stanford University, 1974.

[9] Kumar N. Sivarajan and Rajiv Ramaswami. Lightwave networks based on de bruijn graphs. *IEEE/ACM Trans. Netw.*, 2(1):70–79, 1994.

[10] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001 conference on applications, technologies, architectures, and protocols for computer communications*, pages 149–160. ACM Press, 2001.