

# D2B: a de Bruijn Based Content-Addressable Network<sup>★</sup>

Pierre Fraigniaud<sup>\*,1</sup>

*CNRS*

*Laboratoire de Recherche en Informatique (LRI)  
Université Paris-Sud, 91405 Orsay, France.*

Philippe Gauron<sup>1</sup>

*Laboratoire de Recherche en Informatique (LRI)  
Université Paris-Sud, 91405 Orsay, France.*

---

## Abstract

We show that the de Bruijn graph is appropriate for maintaining *dynamic* connections, e.g., between the members of a P2P application who join and leave the system at their convenience. We describe the content-addressable network D2B, based on an overlay network preserving de Bruijn connections dynamically, and on a distributed hash table (DHT) supporting efficient publish and search procedures. The overlay network has constant expected degree, and any publish or search operation in the DHT takes a logarithmic expected number of steps.

*Key words:* P2P, DHT, de Bruijn graph, overlay network.

---

---

<sup>★</sup> A preliminary version of this paper has been presented as a brief announcement at the 22nd ACM Symp. on Principles of Distributed Computing (PODC), July 13-16, 2003, Boston, Massachusetts (see [10]).

\* Corresponding author.

*Email addresses:* pierre@lri.fr (Pierre Fraigniaud), gauron@lri.fr (Philippe Gauron).

*URLs:* <http://www.lri.fr/~pierre> (Pierre Fraigniaud),  
<http://www.lri.fr/~gauron> (Philippe Gauron).

<sup>1</sup> Both authors are supported by the project “PairAPair” of the ACI “Masse de données”, and by the project “Grand Large” of INRIA.

## 1 Introduction

Let  $G = (V, E)$  be a graph modeling a *physical network*, e.g., the Internet network. The physical network is assumed supporting communication facilities, e.g., the IP protocol, insuring data transfer between any pair of nodes. An *overlay network* is a (di)graph  $H = (U, X)$  where  $U \subseteq V$ , maintaining virtual connections between the nodes in  $U$ . An overlay network provides a *routing* protocol, so that any source node  $s \in U$  can route a message to any target node  $t \in U$ , along a path  $s = u_0, u_1, \dots, u_k = t$ , where the  $u_i$ s belong to  $U$ . The message transfer from  $u_i$  to  $u_{i+1}$  is performed via the communication facilities of the physical network. An overlay network is *dynamic*, in the sense that it allows  $U$  (and therefore  $X$  as well) to evolve with time. More precisely, any node  $u \in U$  can leave the overlay, resulting in a new set of nodes  $U := U \setminus \{u\}$ . Similarly, any node  $v \in V \setminus U$  can join the overlay, resulting in a new set of nodes  $U := U \cup \{v\}$ .

Let  $D$  be a set of resources (e.g., files, data, computation and storage facilities, etc.), shared by a subset of nodes  $U \subseteq V$  of the physical network. A *content-addressable network* enables publishing and searching resources among the nodes in  $U$ , in a distributed manner. A popular way to implement a content-addressable network is by using a *distributed hash table* (DHT). A DHT establishes a correspondence between the resource set and the node set, in the following way. Let  $h : D \rightarrow K$  and  $h' : V \rightarrow K$  be two mapping functions. The set  $K$ , called *key set*, is a metric space, with distance function denoted by  $\text{dist}(\cdot, \cdot)$ . In general,  $h$  and  $h'$  are two hash functions whose role is to evenly distribute the resources and the nodes on the key set. Then,  $d \in D$  is published at node  $u \in U$  if and only if  $u$  is the closest node to  $d$ , that is:

$$\text{dist}(h(d), h'(u)) = \min\{\text{dist}(h(d), h'(v)) \mid v \in U\}.$$

Searching for  $d \in D$  in a DHT uses the overlay network. Roughly, when a node  $u \in U$  looks for  $d \in D$ , a request  $\text{search}(h(d))$  is sent through the overlay network. The overlay network must thus provide facilities so that the request is routed to the node  $v \in U$  where  $d$  has been published. Similarly, when a node  $u \in U$  wants to publish some  $d \in D$ , a request  $\text{publish}(h(d))$  is sent through the overlay network. Again, the overlay network must provide facilities so that this request is routed to the node  $v \in U$  where  $d$  should be published. Clearly, both procedures are based on extensions of the routing protocol in the overlay  $H$ , so that queries can be routed based on keys, and not only on node addresses in  $H$ .

A content-addressable network is an underlying mechanism that can be used to implement fully decentralized *peer-to-peer* (P2P) systems. A user of such a system is a node of the overlay network which has access to every resource stored in the system, via connections to remote nodes. In the Internet, content-addressable networks are good ways to implement distributed publish/search

protocols for finding out IP-addresses of nodes storing requested resources. The design of a content-addressable network consists in defining (1) the overlay network (including the join, leave, and routing procedures), and (2) a DHT based on this overlay. The design of the DHT consists in defining (1) a distributed assignment of keys to resources and nodes, and (2) publish and search procedures (based on the routing procedure in the overlay). This design is subject to the following constraints:

- (1) At any time, all nodes currently in the content-addressable network must be mutually reachable (this constraint is constitutive of a network insuring exhaustive search);
- (2) Any node can leave the content-addressable network at any time, and any node can join the content-addressable network at any time, i.e., a content-addressable network must support a highly dynamic environment;
- (3) At any time, keys must be evenly published among nodes, and any redistribution of keys due to a leave or a join must be fast;
- (4) Publish and search operations must be performed on a key basis, i.e., the route from the consumer of a resource to the node where the resource is published must be determined only by the key assigned to the resource;
- (5) The search time must be small, i.e., the time to reach a node where a resource has been published must be small;
- (6) The update time must be small, i.e., the update of the overlay (connection links, routing tables, key redistribution, etc.) due to a leave or a join must be fast;
- (7) The traffic load incurred by publish and search traffic in the system must be evenly distributed among nodes, to avoid hot spots.

Constraint 5 prevents from connecting the nodes as, e.g., a ring. Constraint 6 prevents from connecting the nodes as, e.g., a complete graph. Constraint 7 prevents from connecting the nodes as, e.g., a binary tree.

### 1.1 Our Results

It was known that the static version of the de Bruijn graph [8] allows to construct large networks of fixed degree and small diameter [5, 6]. We show that one can use the de Bruijn graph to design *dynamic* networks as well. In other words, nodes of a complex dynamic network such as Internet can organize themselves into a constant-degree logarithmic-diameter de Bruijn Graph. More precisely, we describe the content-addressable network D2B, whose overlay network is based on the de Bruijn graph. The most important properties satisfied by D2B are listed below. (We denote by  $n$  the current number of nodes in the overlay network, and an event  $\mathcal{E}$  occurs with high probability (w.h.p.) if  $\text{Prob}(\mathcal{E}) \geq 1 - O(1/n)$ .)

- W.h.p., a search for any key, initiated from any node, reaches the node where the key is published after at most  $O(\log n)$  hops.

- The expected node degree of the D2B overlay network is constant, and there is a constant expected number of control messages that are exchanged during a join or a leave. Hence the expected time of any join or leave is constant. W.h.p., an update takes at most  $O(\log n)$  time.
- The expected number of keys managed by a node in D2B is  $|\mathcal{K}|/n$ , and, with high probability, is at most  $O(|\mathcal{K}| \log n/n)$ , where  $\mathcal{K}$  is the set of keys currently managed by D2B.
- The expected congestion of a node in D2B is  $O((\log n)/n)$ , and the congestion experienced by any node is, w.h.p.,  $O((\log^2 n)/n)$ . Here the congestion of a node is intended to measure the probability that it is involved in a search for a random key, from a random node. It is defined as the expected number of search routes that pass through the node, divided by  $n|\mathcal{K}|$ .

We also define a  $d$ -dimensional version of D2B, for  $d > 2$ . (The basic version of D2B has dimension 2.) The  $d$ -dimensional version of D2B is built upon the de Bruijn graph of dimension  $d$ , and uses the key space  $\mathcal{K} = \{d\text{-ary strings}\}$ . Its expected degree is  $O(d)$ , for an expected diameter  $O(\log n / \log d)$ . This gives a trade-off between the latency for joining or leaving the network, and the latency for a publish/search. Note that a large  $d$  increases the connectivity of the network, and thus its robustness against processor crashes. For  $d = \log n$ , the expected degree is  $O(\log n)$  for an expected diameter  $O(\log n / \log \log n)$ .

## 1.2 Related Works

Graphs augmented with “long range contacts”, as defined in [4], can be used for the design of content-addressable networks. In particular,  $d$ -dimensional toruses augmented with long range contacts chosen at random according to the harmonic distribution yield networks in which routing can be performed on a key basis in  $O(\log^2 n)$  expected number of steps [15]. Symphony [21] has been built on this principle (see also [11, 22]). In [25] is described a dynamic network with constant maximum degree, and  $O(\log n)$  diameter, w.h.p., under a specific probabilistic model of join and leave. However, as already observed by [20], the construction of [25] does not provide a routing scheme, and the intended application is to disseminate queries rather than to route them. In [9], a content-addressable network with logarithmic diameter and fault-tolerant to an adversary deleting up to a constant fraction of the nodes is described. However, the solution is designed for any fixed value of  $n$ , and does not provide for the system to adapt dynamically to a large number of joins or leaves. Moreover, searching generates  $O(\log^2 n)$  messages, and the degree is  $O(\log n)$ . A dynamic version of [9] has been presented in [7].

The content-addressable network CAN described in [28] is based on the  $d$ -dimensional torus topology, and uses the key space  $\mathcal{K} = [0, 1]^d$ . The expected diameter is  $O(dn^{1/d})$ , and the expected degree is  $O(d)$ . Tapestry [32] and Pastry [29] (see also [13]) implement the protocol proposed in [26]. The degree of the induced topology is  $O(d \log_d n)$ , and its expected diameter is  $O(\log_d n)$ ,

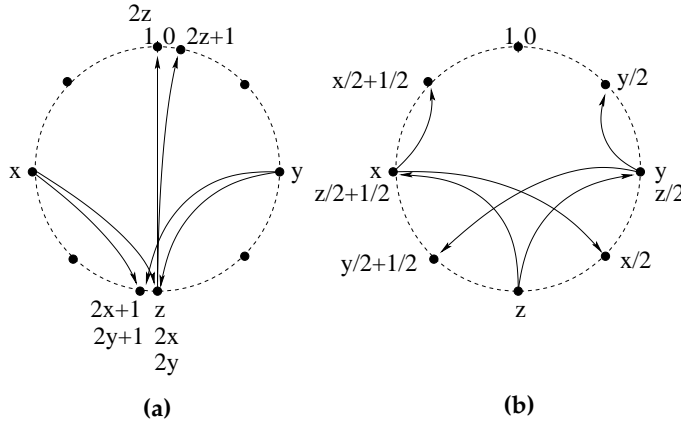


Fig. 1. (a) Koorde connections vs. (b) halving connections

where  $d$  is the base in which the node IDs are encoded. Chord [31] is based on the hypercube topology, and uses the key set  $\mathcal{K} = \{0, \dots, 2^m - 1\}$ . Its expected diameter is  $O(\log n)$ , and the expected degree is  $O(\log n)$ . Viceroy [20] uses the same key set as Chord, but is based on the butterfly graph. The simplified version of Viceroy has expected degree  $O(1)$ , expected diameter  $O(\log n)$  (w.h.p.,  $O(\log^2 n)$ ), and expected congestion  $O((\log n)/n)$  (w.h.p.,  $O((\log^2 n)/n)$ ). An improved version of Viceroy, including a more sophisticated lookup strategy, has a diameter  $O(\log n)$  with high probability. In addition, a “bucket mechanism” allows to fix the maximum degree of the nodes, which is only bounded by  $O(\log n)$ , w.h.p., in the simplified version. Viceroy is the first known constant-degree content-addressable network with logarithmic diameter. Its construction and management are however relatively complex, and require sophisticated procedures which might be difficult to implement in a practical setting. DMBN [7] uses the same set of keys as Chord and Viceroy, and is based on the multi-butterfly graph. It has same diameter and degree as Chord, but is also fault-tolerant to an adversary deleting up to a constant fraction of the nodes.

Some efforts have been made for designing overlay networks that fit with the topology of the physical network (see, e.g., [17, 26, 28, 29, 32]). In particular, LAND [2] is an overlay network that, under some restrictions on the structure of the physical network, has stretch factor (i.e., the ratio between the length of the routes in the overlay network and the length of the routes in the physical network) as close to 1 as desired. The expected degree and diameter of LAND are  $O(\log n)$ . Other efforts have been made for designing overlay networks that fit with the user interests (see, e.g., [30]), but the solutions are currently based on flooding the network, not on routing.

Interestingly, several groups have simultaneously independently discovered the interest of the de Bruijn topology for the design of dynamic networks (cf. [1, 10, 14, 24]). The DHT in [14] uses the same key-space as Chord. A key  $\kappa \in [0, 2^m - 1]$  induces connections to positions  $2\kappa \bmod 2^m$  and  $2\kappa + 1 \bmod 2^m$ , as

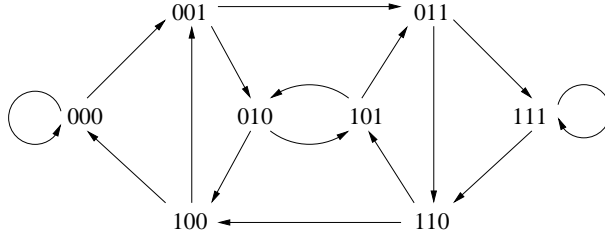


Fig. 2. The de Bruijn graph  $B(2, 3)$

illustrated on Figure 1(a). The distance-halving DHT of [24] uses key-space  $[0, 1)$ . A key  $\kappa$  induces connections to positions  $\frac{\kappa}{2}$  and  $\frac{\kappa}{2} + \frac{1}{2}$ , as illustrated on Figure 1(b). A general methodology for constructing overlay networks based on arbitrary static graphs is described in [1]. Instantiating their method for the de Bruijn graph results in a DHT inheriting the nice properties of this graph. Finally, Broose [12] is a de Bruijn version of Kademlia [23] that was recently proposed to increase the reliability of de Bruijn based DHTs, including D2B.

## 2 The Content-Addressable Network D2B

In this section, we describe D2B, including the join and leave procedures. D2B is parametrized by a parameter  $d \geq 2$ . For the sake of clarity, we describe here the version of D2B for  $d = 2$ . The description of the  $d$ -dimensional D2B network for arbitrary  $d$  is given in Section 4.1.

### 2.1 The de Bruijn Graph

The underlying static topology of the  $d$ -dimensional D2B,  $d \geq 2$ , is the de Bruijn graph  $B(d, k)$ , for  $k \geq 1$ .  $B(d, k)$  is defined from [8]. It is the directed graph whose nodes are all strings of length  $k$  on the alphabet  $\{0, \dots, d-1\}$ , and there is an edge from any node  $x_1x_2 \dots x_k$  to the  $d$  nodes  $x_2 \dots x_k \alpha$ , for  $\alpha = 0, \dots, d-1$ . Figure 2 displays  $B(2, 3)$ . Note that there are loops around all nodes  $\alpha \dots \alpha$ ,  $\alpha \in \{0, \dots, d-1\}$ , and that  $B(d, k)$  is not vertex-transitive. Nevertheless, we will see that this has no impact on the performances of D2B which are uniformly balanced among nodes.  $B(d, 1)$  is the complete graph of  $d$  nodes, with loops.

$B(d, k)$  has  $d^k$  nodes, in-degree and out-degree  $d$ , and diameter  $k$ . Routing from  $x_1 \dots x_k$  to  $y_1 \dots y_k$  is achieved by following the route

$$x_1 \dots x_k \rightarrow x_2 \dots x_k y_1 \rightarrow \dots \rightarrow x_k y_1 \dots y_{k-1} \rightarrow y_1 \dots y_k.$$

A shorter route is obtained by looking for the longest sequence that is suffix of  $x_1 \dots x_k$ , and prefix of  $y_1 \dots y_k$ . If there is such a sequence  $x_i \dots x_k = y_1 \dots y_{k-i+1}$ , then the shortest path from  $x_1 \dots x_k$  to  $y_1 \dots y_k$  is

$$x_1 \dots x_k \rightarrow x_2 \dots x_k y_{k-i+2} \rightarrow \dots \rightarrow x_{i-1} \dots x_k y_{k-i+2} \dots y_{k-1} \rightarrow y_1 \dots y_k.$$

## 2.2 Overall Description of D2B

The 2-dimensional D2B uses the set  $\mathcal{K} = \{0, \dots, 2^m - 1\}$  as key space, also viewed as the set of binary strings of length  $m$ . All participants to D2B use a same function  $h$  that hashes the resource IDs into  $\mathcal{K}$ . Nodes of D2B are given *labels* that are also binary strings, but of length *at most*  $m$ . The function  $h'$  that assign labels to nodes will be precisely described later. Thus there are at most  $2^m$  nodes in D2B. Note that this is not a limitation for  $m = 128$  (or even 256) in practice, to insure a number of keys much larger than the number of IPv6 addresses. The *value* of a node  $u$  labeled  $x_1 \dots x_k$ ,  $x_i \in \{0, 1\}$ , is  $val(u) = 2^{m-k} \cdot \sum_{i=1}^k x_i 2^{k-i}$ .

**Definition 2.1** *A universal prefix set is a set  $S$  of binary words such that, for any infinite word  $\omega \in \{0, 1\}^*$ , there is a unique word in  $S$  which is a prefix of  $\omega$ . The empty set is also a universal prefix set.*

For instance,  $\{0, 100, 1010, 1011, 11\}$  is a universal prefix set. D2B insures that, at any time, the set of labels of all nodes currently in the network is a universal prefix set.

**Key distribution.** Node  $u$  labeled  $x_1 \dots x_k$  is responsible for all keys between  $val(u)$  and  $2^{m-k}(1 + \sum_{i=1}^k x_i 2^{k-i}) - 1$ . More explicitly, the key whose binary representation is  $\kappa_1 \dots \kappa_m$  is managed by node  $x_1 \dots x_k$  currently in the system if and only if  $x_1 \dots x_k$  is a prefix of  $\kappa_1 \dots \kappa_m$ . Hence, a node labeled  $x_1 \dots x_k$  is responsible for  $2^{m-k}$  keys. Conversely, a node responsible for  $2^q$  keys has a label on  $m-q$  bits. All keys are assigned since, by construction, the node-labels form a universal prefix set.

**Routing connections.** At any given time, node labeled  $x_1 \dots x_k$  has either a unique out-neighbor of the form  $x_2 \dots x_j$ ,  $j \leq k$ , or several out-neighbors, of the form  $x_2 \dots x_k y_1 \dots y_\ell$  where  $1 \leq \ell \leq m-k+1$ . In the latter case, the set of sequences  $y_1 \dots y_\ell$  forms a universal prefix set. In particular, if  $x_2 \dots x_k y_1 \dots y_\ell$  is an out-neighbor of  $x_1 \dots x_k$ , then none of the labels  $x_2 \dots x_k y_1 \dots y_i$ ,  $i < \ell$ , is currently used in the network. In the remaining part of the paper, an out-neighbor of a node  $u$  is simply called a *child* of  $u$ . The children of a node labeled  $x_1 \dots x_k$  are displayed on Figure 3(a). In this example, node  $x_1 \dots x_k$  has five children labeled  $x_2 \dots x_k 0$ ,  $x_2 \dots x_k 100$ ,  $x_2 \dots x_k 1010$ ,  $x_2 \dots x_k 1011$ , and  $x_2 \dots x_k 11$ . In the network, there is no node labeled  $x_2 \dots x_k 1$ ,  $x_2 \dots x_k 10$ , or  $x_2 \dots x_k 101$ .

Symmetrically, at any given time, node labeled  $x_1 \dots x_k$  has in-neighbors, simply called *parents*, either of the form  $\alpha x_1 \dots x_j$  with  $\alpha \in \{0, 1\}$  and  $j \leq k$ , or of the form  $\beta x_1 \dots x_k y_1 \dots y_\ell$ , where  $\beta \in \{0, 1\}$  and  $1 \leq \ell \leq m-k-1$ . In the latter case, the set of sequences  $y_1 \dots y_\ell$  forms a universal prefix set. Note that the two forms may coexist simultaneously, but then  $\alpha \neq \beta$ .

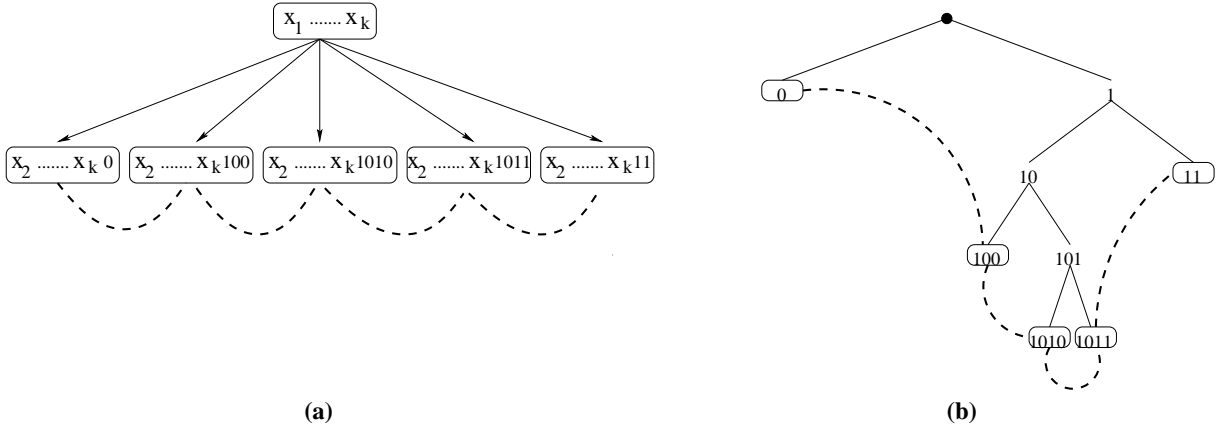


Fig. 3. Children (plain arrows) and sibling (dotted lines) connections

**Remark.** Because of the loops around nodes  $0 \dots 0$ , and  $1 \dots 1$ , the child and parent-connections are slightly different for these two nodes. A node  $u$  labeled  $\alpha \alpha \dots \alpha$ ,  $\alpha \in \{0, 1\}$ , has children the nodes labeled  $\alpha \dots \alpha y_1 \dots y_\ell$ ,  $\ell \geq 1$ , where  $y_1 = \bar{\alpha}$ . The set of labels  $y_2 \dots y_\ell$  is a universal prefix set. The parents of node  $u$  are labeled either  $\bar{\alpha} \alpha \alpha \dots \alpha$  with  $j \leq k$  symbols  $\alpha$ , or  $\bar{\alpha} \alpha \alpha \dots \alpha y_1 \dots y_\ell$ , with  $k$  symbols  $\alpha$ , and  $\ell \geq 1$ . In the latter case, the set of sequences  $y_1 \dots y_\ell$  forms a universal prefix set.

**Sibling connections.** In addition to the child and parent-connections, children of any node  $u$  are linked together by *sibling connections* as follows (see Figure 3(a)). If  $v$  is a child of  $u$  in D2B, then there is an up-sibling connection from  $v$  to  $w$  where  $w$  is the child of  $u$  with the smallest value  $val(w)$  larger that  $val(v)$ . (If  $v$  has the largest value among all children of  $u$ , then  $v$  has no up-sibling.) Similarly, there is a down-sibling connection from  $v$  to  $w$  where  $w$  is the child of  $u$  with the largest value smaller that  $val(v)$ . (If  $v$  has the smallest value among all children of  $u$ , then  $v$  has no down-sibling.) The sibling connections are used for the purpose of key redistribution, and not for routing.

**Routing protocol.** Routing in D2B performs roughly the same as in de Bruijn graph. More precisely, let  $x_1 \dots x_k$  be the label of a node  $u$  in D2B, and let  $\kappa = \kappa_1 \dots \kappa_m$  be any key. Let  $S$  be the longest binary string that is a suffix of  $x_1 \dots x_k$  and a prefix of  $\kappa_1 \dots \kappa_m$ , possibly  $S = \emptyset$ . If  $S = x_1 \dots x_k$ , then  $u$  holds  $\kappa$ . Otherwise, if  $u$  has a unique child  $v$  labeled  $x_2 \dots x_j$ , then the search for key  $\kappa$  is forwarded to this child. If  $u$  has several children, then the search for key  $\kappa$  is forwarded to the child  $v$  labeled  $x_2 \dots x_k y_1 \dots y_\ell$  such that  $S y_1 \dots y_\ell$  is a prefix of  $\kappa_1 \dots \kappa_m$ . By the universal prefix set property, such a children exists, and is uniquely defined.

**Publication of the keys.** A node  $u$  of the system aiming to publish a resource computes the corresponding key  $\kappa \in \mathcal{K}$  by applying the hash function  $h$ . Then it sends a publish message through the network. The format of such

a message is  $\langle \textit{publish}, @_u, \kappa \rangle$ , where  $@_u$  is the physical address of  $u$  (e.g., its IP-address). It is routed like a search message, based on the binary representation of  $\kappa$ . When the node responsible for  $\kappa$  receives a publish message,  $\langle \textit{publish}, @_u, \kappa \rangle$ , it places  $@_u$  in the entry  $\kappa$  of its lookup table, i.e., the table storing the correspondences between resource physical locations and resource keys.

### 2.3 The Join Procedure

As for most of content-addressable networks (see, e.g., [28]), we assume that the physical addresses of some nodes currently in the network are (at least partially) publicly available (e.g., from a web site). Hence, we assume that a node aiming to join the network knows some contact nodes already in the network, called *entry points*. The joining procedure has mainly three stages:

- (1) Getting a D2B label;
- (2) Redistribution of the keys;
- (3) Updating the connections (including routing, and sibling connections).

Let  $u$  be a node joining the system, and let  $v$  be its entry point in D2B, i.e.,  $u$  knows the physical address  $@_v$  of  $v$ .

#### 2.3.1 Getting a D2B label

To join,  $u$  chooses uniformly at random a *temporary* label which is an  $m$ -bit string  $s_1 \dots s_m$ . Then,  $u$  contacts  $v$ , and a join message is sent from  $v$  through the network. The format of such a message is  $\langle \textit{join}, @_u, s_1 \dots s_m \rangle$ , where  $@_u$  is the physical address of  $u$ . This message is routed as a search message, where  $s_1 \dots s_m$  plays the role of the key. Hence, the join message eventually reaches a node  $w$ , with label  $x_1 \dots x_k$ , and responsible for the key  $s_1 \dots s_m$ . I.e.,  $x_1 \dots x_k$  is a prefix of  $s_1 \dots s_m$ . If  $k = m$ , i.e.,  $x_1 \dots x_k = s_1 \dots s_m$ , then the join fails, and  $u$  must choose another temporary label. Such a failure occurs with probability at most  $n/2^m$ , which is virtually null even for one billion nodes, for  $m = 128$  or  $256$ . Hence, assume  $k < m$  (in practice  $k$  is much smaller than  $m$ ). Node  $u$  gets the label  $x_1 \dots x_k 1$ . Node  $w$  extends its label to  $x_1 \dots x_k 0$ . This operation is called *label extension*.

At this point, only  $w$  knows about  $u$ . To preserve consistency,  $w$  carries on acting as  $x_1 \dots x_k$  until the end of the join procedure.

#### 2.3.2 Key redistribution

In the lookup table stored by  $w$ , all keys that have  $x_1 \dots x_k 1$  as prefix are transferred from  $w$  to  $u$ . Actually, the keys that are transferred to  $u$  are only those corresponding to physical addresses of nodes holding published resources. Hence, the volume of the transfer is much smaller than  $2^{m-k-1}$  which is the range of keys managed by  $u$ . Again, to preserve consistency, node  $w$  keeps a copy of the lookup table corresponding to the transferred keys, until the end of the

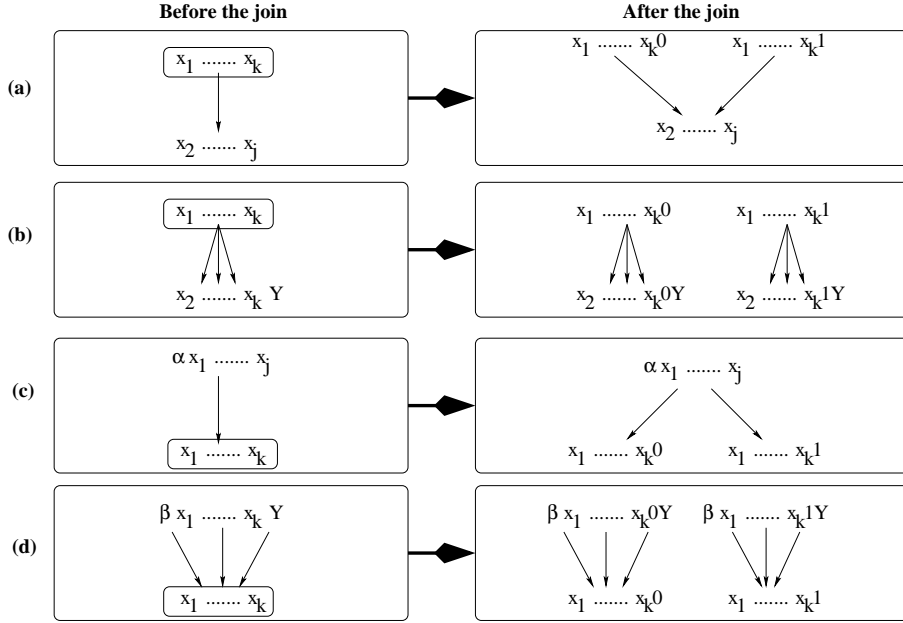


Fig. 4. Updating the connections after a join

join procedure.

### 2.3.3 Updating the connections

**a) Child-connections.** Node  $u$  gets from  $w$  the physical addresses of all the children of  $w$ . We consider two cases, depending on whether  $w$  has a loop around it, i.e., whether or not  $w$  is labeled either  $00\dots 0$  or  $11\dots 1$ .

- (1) General case: there is a pair of indexes  $i \neq j$  such that  $x_i \neq x_j$ . We consider the two exclusive cases:
  - (a) If  $w$  has a unique child labeled  $x_2\dots x_j$ ,  $j \leq k$ , then this child becomes the unique child of  $u$ , and remains child of  $w$  (see Figure 4(a)).
  - (b) If  $w$  has several children, with labels of the form  $x_2\dots x_k y_1\dots y_\ell$ ,  $\ell \geq 1$  (see Figure 4(b)), then those satisfying  $y_1 = 1$  become the children of  $u$ . They are informed by  $w$  that  $w$  is no more their parent, and must be replaced by  $u$ . Children of  $w$  with  $y_1 = 0$  remain children of  $w$ .
- (2) Specific case:  $w$  is labeled  $\alpha\alpha\dots\alpha$ ,  $\alpha \in \{0, 1\}$ . Then it has children of the form  $\alpha\dots\alpha y_1\dots y_\ell$ ,  $\ell \geq 1$ , with  $y_1 = \bar{\alpha}$ . By label-extension, either  $w$  or  $u$  takes label  $\alpha\alpha\dots\alpha\alpha$ , while the other takes label  $\alpha\alpha\dots\alpha\bar{\alpha}$ . Node labeled  $\alpha\alpha\dots\alpha\alpha$  takes  $\alpha\alpha\dots\alpha\bar{\alpha}$  as unique child, while node  $\alpha\alpha\dots\alpha\bar{\alpha}$  takes all nodes  $\alpha\dots\alpha y_1\dots y_\ell = \alpha\dots\alpha\bar{\alpha}y_2\dots y_\ell$  as children.

**b) Parent-connections.** Every parent  $w'$  of  $w$  is informed by  $w$  of the existence of a new node  $u$  labeled  $x_1\dots x_k 1$ , of the physical address  $@_u$  of  $u$ , and of the new label  $x_1\dots x_k 0$  of  $w$ . For every parent  $w'$ , D2B proceeds as follows:

- (1) General case: there is a pair of indexes  $i \neq j$  such that  $x_i \neq x_j$ . We

consider the two following sub-cases:

- (a) If  $w'$  is labeled  $\alpha x_1 \dots x_j$  with  $j \leq k$  (see Figure 4(c)), then  $w'$  takes  $u$  as one of its child, and modifies the label of  $w$  in its connection table. Hence  $w'$  has one more child, and its degree increases by one.
  - (b) If  $w'$  is labeled  $\beta x_1 \dots x_k y_1 \dots y_\ell$  with  $\ell \geq 1$  (see Figure 4(d)), then  $w'$  keeps  $w$  as child if  $y_1 = 0$ , or replaces  $w$  by  $u$  if  $y_1 = 1$ .
- (2) Specific case:  $w$  is labeled  $\alpha \dots \alpha$ ,  $\alpha \in \{0, 1\}$ . Again, by label-extension, either  $w$  or  $u$  takes label  $\alpha \dots \alpha \alpha$ , while the other takes label  $\alpha \dots \alpha \bar{\alpha}$ . There are two exclusive sub-cases:
- (a)  $w$  has a parent of the form  $\bar{\alpha} \alpha \dots \alpha$  with  $j \leq k$   $\alpha$ 's. Then node labeled  $\alpha \dots \alpha \alpha$  takes this node as its parents, while node labeled  $\alpha \dots \alpha \bar{\alpha}$  takes both  $\alpha \dots \alpha \alpha$  and  $\bar{\alpha} \alpha \dots \alpha$  as parents.
  - (b)  $w$  has parents of the form  $\bar{\alpha} \alpha \dots \alpha y_1 \dots y_\ell$ , with  $k$   $\alpha$ 's and  $\ell \geq 1$ . Then node labeled  $\alpha \dots \alpha \alpha$  takes those with  $y_1 = \alpha$  as parents, while  $\alpha \dots \alpha \bar{\alpha}$  takes those with  $y_1 = \bar{\alpha}$ , together with node  $\alpha \dots \alpha \alpha$ , as parents.

c) Sibling connections. Node  $u$  gets from  $w$  the physical addresses of its up-sibling, which is just the former up-sibling of  $w$ . This node is informed that its down sibling is no more  $w$  but  $u$ . The new up-sibling of  $w$  is simply  $u$ , and the down-sibling of  $u$  is  $w$ .

## 2.4 The Leave Procedure

The leave procedure performs in three stages:

- (1) Finding a substitute for the leaving node;
- (2) Redistribution of the keys;
- (3) Updating the connections.

Obviously, if a node crashes, the leave procedure may not be entirely executed, possibly it would not be executed at all. The case of a crash is in fact very different from the case of a leave, and thus will be considered later in the text (cf. Section 4.2.1). In the current setting, we consider a node  $u$  labeled  $x_1 \dots x_k$  leaving carefully the system.

### 2.4.1 Node substitution

If a node  $v$  labeled  $x_1 \dots x_{k-1} \bar{x}_k$  is in the network (as up or down sibling of  $u$ ), then the lookup tables managed by  $u$  and  $v$  are merged and stored entirely by  $v$ , which is relabeled in  $x_1 \dots x_{k-1}$ . If  $x_1 \dots x_{k-1} \bar{x}_k$  is not a node label in the network, then the node-substitution procedure is slightly more complex. For instance,  $x_1 \dots x_{k-1} \bar{x}_k$  may have been extended in  $x_1 \dots x_{k-1} \bar{x}_k 0$  and  $x_1 \dots x_{k-1} \bar{x}_k 1$ . Possibly, one of these two latter labels (possibly both) has then been extended, and so on. Such label-extensions create a virtual binary tree rooted at  $x_1 \dots x_{k-1} \bar{x}_k$ , whose leaves are nodes currently in the system (see Figure 3(b)). In this tree, the children of an internal vertex  $x_1 \dots x_{k-1} \bar{x}_k y_1 \dots y_p$

are vertices  $x_1 \dots x_{k-1} \overline{x_k} y_1 \dots y_p 0$  and  $x_1 \dots x_{k-1} \overline{x_k} y_1 \dots y_p 1$ . Since the depth of this virtual binary tree is finite (it is at most  $m - k$ ), there exists at least one pair of leaves whose labels differ only at the rightmost bit-position. Let us call *critical pair* such a pair of leaves. In Figure 3(b), there is a critical pair  $\{x_2 \dots x_{k-1} \overline{x_k} 1010, x_2 \dots x_{k-1} \overline{x_k} 1011\}$ .

The sibling connections allow to find a critical pair for every node  $u$  labeled  $x_1 \dots x_k$  leaving the system, as follows. If  $x_k = 0$ , a critical-pair message is sent to the up-sibling  $u'$  of  $u$ . This message has format  $\langle \text{leave}, @_u \rangle$ . If  $u'$  has label  $x_1 \dots x_{k-1} 1$ , then  $\{u, u'\}$  is a critical pair. Otherwise,  $u'$  forwards the message to its up-sibling  $u''$ . If the labels of  $u'$  and  $u''$  differ only at the rightmost bit-position, then  $\{u', u''\}$  is a critical pair. And so on. Since the sibling chain is bounded, a critical pair will eventually be found. In case  $x_k = 1$ , one proceeds the same using down-sibling connections instead of up-sibling connections.

Informally, one node of the critical pair will be the substitute for  $u$ , and the other will be the substitute for the two nodes of the critical pair. This is detailed in the next section.

#### 2.4.2 Updating the network

There are two cases depending whether the leaving node  $u$  belongs to the identified critical pair  $\{v, v'\}$ .

If  $u \in \{v, v'\}$ , then node  $u'$  labeled  $x_1 \dots x_{k-1} \overline{x_k}$  belongs to  $\{v, v'\}$  as well, and becomes the substitute for  $u$  and  $u'$ . Hence,  $u'$  receives from  $u$  all information about the keys managed by  $u$ . It also receives from  $u$  all the information about the sibling, parents, and children connections of  $u$ . Node  $u'$  is relabeled in  $x_1 \dots x_{k-1}$ . (This operation is called *label-contraction*.) Then  $u'$  informs its parents that its label was contracted, and  $u$  informs its parents that it leaves the network. Node  $u$  can then leave the network. Node  $u'$  stores in its routing tables the physical addresses and labels of the former children of  $u$ , which now become children of  $u'$ . Finally,  $u'$  informs the former parents of  $u$  that it is now their child, with label  $x_1 \dots x_{k-1}$ .

If  $u \notin \{v, v'\}$ , then assume, w.l.o.g., that  $v$  is labeled  $x_1 \dots x_{k-1} \overline{x_k} y_1 \dots y_p 0$  and  $v'$  is labeled  $x_1 \dots x_{k-1} \overline{x_k} y_1 \dots y_p 1$ . Node  $v'$  is the substitute for  $v$  and  $v'$ , while node  $v$  is the substitute for  $u$ . Hence  $v'$  perform the same procedure for  $v$  and  $v'$ , as  $u'$  performed for  $u$  and  $u'$  in the previous case. In particular, the label of  $v'$  is contracted into  $x_1 \dots x_{k-1} \overline{x_k} y_1 \dots y_p$ . Node  $v$  takes the label of  $u$ , and retrieves from  $u$  its lookup and routing tables. As soon as  $v$  has retrieved all information from  $u$ , node  $u$  leaves the system.

#### 2.5 Example

An example of the behavior of D2B is presented in the Appendix.

### 3 Main properties of D2B

This section is entirely dedicated to the proof of Theorem 1. We prove correctness under the assumption that joins and leaves do not overlap. Nevertheless, one can statistically relax this restrictive assumption by using the techniques presented in [18, 19]. For more detail on procedures supporting simultaneous joins and leaves, see [16].

**Theorem 1** *The D2B network of key-set  $\mathcal{K} = \{m\text{-bit binary strings}\}$  is provably correct, and satisfies the following:*

- *The expected number of keys managed by a node of an  $n$ -node D2B network is  $|\mathcal{K}|/n$ , and is, w.h.p., at most  $O(|\mathcal{K}| \log n/n)$ .*
- *A search/publish for a key  $\kappa$  initiated from any node labeled  $x_1 \dots x_k$  is routed correctly, and, w.h.p., reaches the node responsible for the key  $\kappa$  in at most  $O(\log n)$  hops. With probability  $1 - o(1)$ , the longest route followed by a lookup message is at most  $O(\log n)$  hops.*
- *At every intermediate node, the routing decision takes  $O(\log \log n)$  comparisons of words on  $O(\log n)$  bits. The expected congestion of any node is  $O((\log n)/n)$ , which is optimal among all networks of constant degree. W.h.p., the congestion of a node is at most  $O((\log^2 n)/n)$ .*
- *During a join or a leave, the key redistribution involves only two nodes for a join, and at most three nodes for a leave. The expected number of link modifications due to a join or a leave is  $O(1)$ , and is, w.h.p., at most  $O(\log n)$ .*

The fact that, during a join or a leave, the key-redistribution involves only two nodes for a join, and at most three nodes for a leave, is straightforward by construction. All the other properties are consequences of the following lemmas.

**Lemma 1** *At any given time, we have :*

- (1) *For any  $\kappa \in \{0, 1\}^m$ , there is a unique node in the D2B network whose label is a prefix of  $\kappa$ .*
- (2) *Let  $u$  be a node of D2B labeled  $x_1 \dots x_k$ , with at least two children. If there are  $i \neq j$  such that  $x_i \neq x_j$ , then the children of  $u$  are of the form  $x_2 \dots x_k y_1 \dots y_\ell$ ,  $\ell \geq 1$ , and the set of sequences  $y_1 \dots y_\ell$  of all the children of  $u$  is a universal prefix set. If  $x_1 = \dots = x_k = \alpha$ , then the children of  $u$  are of the form  $x_2 \dots x_k y_1 \dots y_\ell$ ,  $\ell \geq 2$ ,  $y_1 = \bar{\alpha}$ , and the set of sequences  $y_2 \dots y_\ell$  of all the children of  $u$  is a universal prefix set.*

**Proof.** Initially, there is a unique node in the network, labeled by the empty string  $\emptyset$ . This label is the prefix of any string in  $\{0, 1\}^*$ . Thus (1) holds initially. Node with label  $\emptyset$  has no parent, nor child. So (2) holds as well. We show that these two properties are preserved after a join or a leave.

*The case of a join.* Assume that the network currently satisfies (1) and (2), and that a new node  $u$  joins the network. Let  $s_1 \dots s_m$  be the temporary label of  $u$ , and let  $x_1 \dots x_k$  be the label of node  $v$  currently responsible for key  $s_1 \dots s_m$ . The joining node  $u$  is given label  $x_1 \dots x_k 1$ , while  $v$  extends its label to  $x_1 \dots x_k 0$ . The key-redistribution protocol described in Section 2.3.2 clearly insures that (1) is satisfied after the join since all keys with prefix  $x_1 \dots x_k 1$  are moved from  $v$  to  $u$ . For (2), we consider separately the children- and parent-connections.

If node  $v$  had at least two children before the join, then, by (2), these children have labels of the form  $x_2 \dots x_k y_1 \dots y_\ell$ . By construction (cf. Section 2.3.3), if there are two indexes  $i \neq j$  such that  $x_i \neq x_j$ , then children with labels  $x_2 \dots x_k 1 y_2 \dots y_\ell$  become children of  $u$ , while children with labels of the form  $x_2 \dots x_k 0 y_2 \dots y_\ell$  remain children of  $v$ . Since the initial set of sequences  $y_1 \dots y_\ell$  is a universal prefix set, the same holds for the two sets of sequences  $y_2 \dots y_\ell$  corresponding to  $u$  and  $v$ . (Note that these sequences may be empty, but an empty string is a universal prefix set, and anyway the lemma considers only nodes with at least two children.) Therefore, (2) remains satisfied for both  $u$  and  $v$ . If  $x_1 = \dots = x_k = \alpha$ , then node  $x_1 \dots x_k \alpha$  has a unique child, and node  $x_1 \dots x_k \bar{\alpha}$  has children all the initial children of  $v$ . By (2), these children were labeled  $\alpha \dots \alpha y_1 \dots y_\ell$  with  $y_1 = \bar{\alpha}$ ,  $\ell \geq 2$ , and the set of sequences  $y_2 \dots y_\ell$  is a universal prefix set. Therefore, (2) remains satisfied for  $u$  and  $v$ .

If node  $v$  had a parent with label of the form  $\alpha x_1 \dots x_j$ ,  $j \leq k$ , before the join, then, after the join, this parent has replaced its child  $x_1 \dots x_k$  by two children labeled  $x_1 \dots x_k 0$  and  $x_1 \dots x_k 1$ , and therefore (2) holds. If node  $v$  had parents with label of the form  $\beta x_1 \dots x_k y_1 \dots y_\ell$  before the join, then, after the join, parents of the form  $\beta x_1 \dots x_k 0 y_2 \dots y_\ell$  have  $x_1 \dots x_k 0$  as unique child, and those of the form  $\beta x_1 \dots x_k 1 y_2 \dots y_\ell$  have  $x_1 \dots x_k 1$  as unique child. Therefore (2) holds after the join.

*The case of a leave.* Assume now that the network satisfies (1) and (2), and that node  $u$  labeled  $x_1 \dots x_k$  leaves the network. From the description of the procedure in Section 2.4.2, we assume first, for the sake of simplicity, that  $u$  belongs to the critical pair, i.e., there is a node  $v$  labeled  $x_1 \dots x_{k-1} \bar{x}_k$  currently in the network. By construction, node  $v$  relabels itself in  $x_1 \dots x_{k-1}$ , and takes care of all keys previously managed by  $u$ . Hence, (1) remains satisfied after the leave.

If  $x_1 \dots x_k$  had a unique child  $x_2 \dots x_j$ ,  $j < k$ , before the leave, then  $x_1 \dots x_{k-1} \bar{x}_k$  had also  $x_2 \dots x_j$  as unique child. After the leave, node  $x_2 \dots x_j$  becomes the unique child of  $x_1 \dots x_{k-1}$ . Hence, (2) remains satisfied after the leave. If  $x_1 \dots x_k$  had a unique child  $x_2 \dots x_k$ , before the leave, and  $x_1 \dots x_{k-1} \bar{x}_k$  had children with labels of the form  $x_2 \dots x_{k-1} \bar{x}_k y_1 \dots y_\ell$  before the leave, where the sequences  $y_1 \dots y_\ell$  form a universal prefix set (possibly empty), then, after the

leave,  $x_1 \dots x_{k-1}$  has children  $x_2 \dots x_k$  and all the  $x_2 \dots x_{k-1} \overline{x_k} y_1 \dots y_\ell$ . Hence, (2) remains satisfied after the leave since  $\{x_k\} \cup \{\overline{x_k} y_1 \dots y_\ell\}$  is a universal prefix set. Finally, if node  $x_1 \dots x_k$  had children with labels  $x_2 \dots x_k y_1 \dots y_p$  before the leave, while  $x_1 \dots x_{k-1} \overline{x_k}$  had children with labels  $x_2 \dots x_{k-1} \overline{x_k} z_1 \dots z_q$ , then, after the leave, node labeled  $x_1 \dots x_{k-1}$  has children nodes labeled

$$x_2 \dots x_k y_1 \dots y_p \text{ and } x_2 \dots x_{k-1} \overline{x_k} z_1 \dots z_q.$$

Property (2) remains satisfied after the leave since both the  $y_1 \dots y_p$ 's and the  $z_1 \dots z_q$ 's are universal prefix sets.

The parents of  $x_1 \dots x_k$  and  $x_1 \dots x_{k-1} \overline{x_k}$  are respectively of the form

$$\alpha x_1 \dots x_k y_1 \dots y_p \text{ and } \beta x_1 \dots x_{k-1} \overline{x_k} z_1 \dots z_q.$$

They have  $x_1 \dots x_{k-1}$  as unique child after the leave. Hence (2) is satisfied. A parent of  $x_1 \dots x_k$  and  $x_1 \dots x_{k-1} \overline{x_k}$  with label of the form  $\alpha x_1 \dots x_j$ ,  $j < k$ , has child  $x_1 \dots x_{k-1}$  after the leaves. Therefore, if  $\alpha x_1 \dots x_j$  satisfied (2) before the leave, then it still satisfies (2) after the leave.  $\square$

**Lemma 2** *A search initiated by any node labeled  $x_1 \dots x_k$  reaches its destination in at most  $k$  hops. The same result holds for publishing.*

**Proof.** Let us consider a node  $u$  labeled  $x_1 \dots x_k$  searching for key  $\kappa = \kappa_1 \dots \kappa_m$ . The lookup is performed by routing  $\kappa_1 \dots \kappa_m$  toward the node  $v$  responsible for that key.

*Claim.* The label of any node  $w \neq v$  along the route from  $u$  to  $v$  is of the form  $x_i \dots x_j S$  where  $j \geq i$ ,  $S$  is a binary string, possibly empty, and the length of the longest binary string that is a suffix of  $x_i \dots x_j S$  and a prefix of  $\kappa$  is of length at least  $|S|$ .

*Proof.* At  $u$ ,  $i = 1$ ,  $j = k$ , and  $S = \emptyset$ , so the claim holds for the first node of the route. Let  $x_i \dots x_j s_1 \dots s_\sigma$  be the label of the current node  $w \neq v$ , and assume that the length of the longest binary string  $T$  that is a suffix of  $x_i \dots x_j s_1 \dots s_\sigma$  and a prefix of  $\kappa$  is of length at least  $\sigma$ . Assume first that  $w$  is not labeled  $\alpha \alpha \dots \alpha$ . If  $w$  has more than one child, then, from Lemma 1, there is a child  $w'$  labeled  $L = x_{i+1} \dots x_j s_1 \dots s_\sigma y_1 \dots y_\ell$  such that  $T y_1 \dots y_\ell$  is a binary string that is a suffix of  $L$  and a prefix of  $\kappa$ . From the choice of  $y_1 \dots y_\ell$  in the routing protocol, the next node on the route from  $u$  to  $v$  is the node  $w'$  labeled  $L$ . This label is of the form  $x_{i'} \dots x_{j'} S'$  and satisfies the property of the claim. If  $w$  has a unique child, then it is either of the form  $x_{i+1} \dots x_{j'}$  where  $i+1 \leq j' \leq j$ , or of the form  $x_i \dots x_{j'} s_1 \dots s_{\sigma'}$  where  $1 \leq \sigma' \leq \sigma$ . In both cases, the label of the child satisfies the hypothesis of the claim. The case where  $w$  is labeled  $\alpha \alpha \dots \alpha$  (which can actually occur only for  $w = u$ ) is treated similarly, again by application of Lemma 1. This completes the proof of the claim.  $\diamond$

From the claim, if  $x_i \dots x_j S$  is the label of the current node along the route from  $u$  to  $v$ , then the label of next node is of the form  $x_{i+1} \dots x_{j'} S'$ , where  $S$  and  $S'$  satisfy the hypotheses of the claim. Therefore, either, after  $i - 1$  hops from node labeled  $x_1 \dots x_k$ , one reaches a node labeled  $x_i \dots x_j S$  where  $x_i \dots x_j S$  is a prefix of  $\kappa$ , or, after  $i - 1$  hops, one reaches a node labeled  $x_i S$  where  $S$  is a prefix of  $\kappa$ . In the former case, we are done. In the latter, the next node of the route is the destination. Thus, in both cases, one reaches the node  $v$  responsible for the key  $\kappa$ , and the number of hops along the route from  $u$  to  $v$  is at most  $(i - 1) + 1 \leq k$ .  $\square$

**Lemma 3** *Assume that nodes joins and leaves uniformly at random. Then, w.h.p., the label  $x_1 \dots x_k$  of any node of an  $n$ -node D2B network satisfies*

$$\log n - \log \log n - O(1) \leq k \leq O(\log n).$$

*Also, with probability  $1 - o(1)$ , the longest label  $x_1 \dots x_k$  satisfies  $k = O(\log n)$ .*

**Proof.** Let us consider a node  $u$  with label  $x_1 \dots x_k$  in D2B. Since nodes independently join and leave uniformly at random, the set of labels in an  $n$ -node D2B network are those that would be obtained by choosing  $n$  integers independently and uniformly at random in  $[0, 2^m)$ . Let  $I$  be an interval of  $[0, 2^m)$  starting at  $val(x)$ , and containing  $c2^m \log n/n$  integers, for any constant  $c > 3$ . The probability that an integer is chosen in  $I$  is  $c \log n/n$ . Let  $X$  be the random variable counting the number of integers chosen in  $I$ . From Chernoff bound,  $\text{Prob}(|X - c \log n| > \sqrt{3c} \log n) < 2/n$ . Therefore, w.h.p., at least one integer is chosen in  $I$ , and thus  $u$  is responsible for less than  $c2^m \log n/n$  keys. Hence, since a node responsible for at most  $2^q$  keys has a label on at least  $m - q$  bits, we have  $k \geq \log n - \log \log n - \log c$ . Also, w.h.p., no more than  $O(\log n)$  integers are chosen in  $I$ , and thus  $u$  is responsible for at least  $\frac{|I|}{2^{O(\log n)}} = \frac{c2^m \log n}{n2^{O(\log n)}}$  keys. Hence,  $k \leq O(\log n)$ .

Let us split  $[0, 2^m)$  into  $\Theta(n/\log n)$  intervals of size  $2^m \log n/n$ , and consider  $n$  integers independently and uniformly chosen at random in  $[0, 2^m)$ . We apply the following result by Raab and Steger [27], on the ‘‘balls into bins’’ game. Assume that we throw  $n$  balls independently and uniformly at random into  $b$  bins, where  $n = c b \log b$  for some constant  $c$ . Let  $X$  be the random variable counting the maximum number of balls in any bin. Then  $\text{Prob}(X > d \log n) = o(1)$  where  $d$  is a constant depending on  $c$ . Applying directly this result to our setting yields that the probability that the maximum number of integers chosen in any interval exceeds  $O(\log n)$  is  $o(1)$ . Therefore, with probability  $1 - o(1)$ , the minimum number of keys managed by any node of a D2B network is at least  $2^m \log n/n2^{O(\log n)}$ , and thus the maximum length of all labels is at most  $O(\log n)$ .  $\square$

The following is a direct consequence of Lemma 3.

**Corollary 1** *The number of keys managed by a node of an  $n$ -node D2B net-*

work is, w.h.p., at most  $O(2^m \log n/n)$ .

The following is a direct consequence of Lemmas 2 and 3.

**Corollary 2** *The number of hops experienced by a publish/search to reach its destination in an  $n$ -node D2B network is, w.h.p., at most  $O(\log n)$ .*

**Lemma 4** *The expected number of link-modifications due to a join or a leave is constant, and is, w.h.p., at most  $O(\log n)$ .*

**Proof.** Let  $x_1 \dots x_k$  be the label of a node  $u$ . If  $u$  has more than a single child, then these children are labeled with string of the form  $x_2 \dots x_k y_1 \dots y_\ell$  where  $\ell \geq 1$ . The range of keys covered by the children of  $u$  goes from  $x_2 \dots x_k 0 \dots 0$  with  $m - k + 1$  zeros, to  $x_2 \dots x_k 1 \dots 1$  with  $m - k + 1$  ones. From Lemma 3, w.h.p.,  $k \geq \log n - \log \log n - O(1)$ . Therefore the number of keys managed by all children of  $u$  together is at most  $2^{m - \log n + \log \log n + O(1)} = O(2^m \log n/n)$ . From Chernoff bound, this range of keys is, w.h.p., covered by at most  $O(\log n)$  nodes. Therefore, the out-degree of  $u$  is, w.h.p.,  $O(\log n)$ . The same argument applies for the in-degree of node  $u$  by considering separately parents of the form  $0x_1 \dots x_k y_1 \dots y_\ell$ , and those of the form  $1x_1 \dots x_k y_1 \dots y_\ell$ . Hence the degree of  $u$  is, w.h.p.,  $O(\log n)$ .  $\square$

**Lemma 5** *The expected congestion of a node is  $O((\log n)/n)$ , and is, w.h.p.,  $O((\log^2 n)/n)$ .*

**Proof.** Let  $u$  be any node currently in D2B, and let  $x_1 \dots x_k$  be its label. We compute an upper bound on the load of  $u$ , i.e., on the number of search routes that pass through  $u$ , or ends at  $u$ . The expected size of the lookup table stored by  $u$  is  $O(2^m/n)$ . Therefore, since there are  $n - 1$  possible sources, the expected load induced by search for keys published at  $u$  is  $O(2^m)$ . The search routes which traverse  $u$  have a specific format. For a source labeled  $y_1 \dots y_\ell x_1 \dots x_i$ ,  $i \geq 1$ , the searched keys must be of the form  $x_j \dots x_k \kappa_1 \dots \kappa_{m-k+j-1}$  where  $j \leq i + 1$ . The expected number of nodes with a label ending with the sequence  $x_1 \dots x_i$  is  $n/2^i$ . The number of keys of the form  $x_j \dots x_k \kappa_1 \dots \kappa_{m-k+j-1}$  with  $j \leq i + 1$  is at most  $2^{m-k+i}$ , and thus in average at most  $2^{m+i}/n$ . Therefore, for a given  $i$ , the expected contribution to the load is at most  $2^m$ . Since there are  $O(\log n)$  possible values for  $i$ , the expected total load is  $O(2^m \log n)$ , and thus the expected congestion is at most  $O(\log n/n)$ .

Now, from Lemma 3, w.h.p.,  $k \geq \log n - \log \log n - O(1)$ . Therefore, the size of the lookup table stored by  $u$  is, w.h.p., at most  $O(2^m \log n/n)$ . Let  $i_0 = \log n - \log \log n$ , and let  $i \leq i_0$ . Applying Chernoff bound, the number of nodes whose labels end with the sequence  $x_1 \dots x_i$  is at most  $O(n/2^i)$  with probability at least  $1 - O(\frac{1}{n \log n})$ . Therefore, the contribution of such nodes to the load of  $u$  is at most  $O(2^{m-k+i} n/2^i) \leq O(2^m \log n)$  with probability at

least  $1 - O(\frac{1}{n \log n})$ . Therefore, nodes with label containing a sequence  $x_1 \dots x_i$  as suffix for some  $i \leq i_0$  contribute of  $O(2^m \log^2 n)$  to the load of  $u$ , with high probability.

Let  $i > i_0$ , and let us compute the contribution to the load of nodes with labels containing a sequence  $x_1 \dots x_i$  as suffix. By Chernoff bound, there are, w.h.p., at most  $O(\log n)$  nodes with labels ending with the sequence  $x_{i-i_0} \dots x_i$ . Therefore there are at most  $O(\log n)$  nodes with labels ending with the sequence  $x_1 \dots x_i$ . The contribution of these nodes to the load is at most  $O((\log n)2^{m-k+i})$ . Summing up these contributions for all  $i$ 's,  $i_0 < i \leq k$ , the resulting contribution to the load is  $O(2^m \log n)$ .

Therefore, the total load of  $u$  is, w.h.p.,  $O(2^m \log^2 n)$ . Thus its congestion is, w.h.p.,  $O((\log^2 n)/n)$ .  $\square$

**Remark.** An expected congestion of  $O((\log n)/n)$  is optimal for an  $n$ -node network of constant degree with  $|\mathcal{K}|/n$  keys per node. Indeed, let us consider a directed graph with maximum in- and out-degree  $\Delta$ . The number of nodes at distance  $\leq d$  from any node  $u$  is at most  $\sum_{i=0}^d \Delta^i$ . Therefore, there are at most  $O(\sqrt{n}/\Delta)$  nodes at distance  $\leq \frac{1}{2} \log_{\Delta} n$ , and thus there are  $\Theta(n)$  nodes at distance  $\Omega(\log n)$ . Therefore, each node contributes of  $\Omega(|\mathcal{K}| \log n)$  to the load, resulting in a global load of  $\Omega(n|\mathcal{K}| \log n)$ . To have  $n - o(n)$  nodes with load  $O(|\mathcal{K}| \log n)$ , the global load must be balanced among nodes. Thus  $n - o(n)$  nodes have load  $\Omega(|\mathcal{K}| \log n)$ , and thus a congestion  $\Omega((\log n)/n)$ .

## 4 Variants of the Construction

In this section, we present several variants of D2B, including (1) the  $d$ -dimensional version of the network, (2) an attempt to match the logical network to the physical one, (3) a discussion about the robustness of D2B, and (4) a simple strategy to decrease the degree of the nodes.

### 4.1 The $d$ -dimensional D2B network

The  $d$ -dimensional D2B,  $d \geq 2$ , uses the set of keys  $\mathcal{K} = \{d\text{-ary strings}\}$ , i.e., the set of words of length  $m$  on an alphabet of  $d$  letters  $0, 1, \dots, d-1$ . The underlying topology of D2B is  $B(d, k)$ . More precisely, a node of the  $d$ -dimensional D2B is labeled by a pair  $\langle x_1 \dots x_k, [a, b] \rangle$  where  $x_i \in \{0, \dots, d-1\}$ , and  $0 \leq a \leq b \leq d-1$ .

Node labeled  $\langle x_1 \dots x_k, [a, b] \rangle$  is responsible for the key  $\kappa \in \{0, \dots, d-1\}^m$  if and only if  $x_1 \dots x_k \alpha$  is a prefix of  $\kappa$  for some  $\alpha \in [a, b]$ . A universal prefix property, defined similarly to the case  $d = 2$ , insures that all keys are assigned. During a join, if the temporary label of a node  $u$  is managed by node  $w$  of label  $\langle x_1 \dots x_k, [a, b] \rangle$ , then  $v$  extends its label in the following way. If  $a < b$ , then  $v$  changes its label to  $\langle x_1 \dots x_k, [a, a + \lfloor \frac{b-a}{2} \rfloor] \rangle$  while  $u$  takes label  $\langle x_1 \dots x_k, [a +$

$\lfloor \frac{b-a}{2} \rfloor + 1, b \rangle$ . If  $a = b$ , then  $v$  changes its label to  $\langle x_1 \dots x_k a, [0, \lfloor \frac{d-1}{2} \rfloor] \rangle$  while  $u$  takes label  $\langle x_1 \dots x_k a, [\lfloor \frac{d-1}{2} \rfloor + 1, d-1] \rangle$ .

The children of node  $\langle x_1 \dots x_k, [a, b] \rangle$  are either of the form  $\langle x_2 \dots x_j, [\alpha, \beta] \rangle$ ,  $j \leq k$ , or of the form  $\langle x_2 \dots x_k y_1 \dots y_\ell, [\alpha, \beta] \rangle$ ,  $\ell \geq 1$ . Routing performs as in the 2-dimensional case, by looking for the longest prefix of the requested key among the suffixes of the labels of the children. The sibling connections are defined in a way similar to the 2-dimensional case, and the leave procedure also performs the same by looking for a critical pair among the sibling nodes.

One can easily check that the expected length  $k$  of a label  $\langle x_1 \dots x_k, [a, b] \rangle$  is  $O(\log_d n)$ , yielding an expected degree of  $O(d)$  and a diameter  $O(\log_d n)$ . Hence, the  $d$ -dimensional D2B allows a trade-off between the lookup latency, and the time required to update the connections after a join or a leave. Also, the  $d$ -dimensional D2B provides a network more robust against processor crash, as discussed in the next section.

## 4.2 Improving the performances of D2B

### 4.2.1 Robustness

A peer-to-peer system must be able to support a certain number of processor crashes, and the brute disconnection of users not respecting the leave procedure. As in most of the content-addressable networks proposed in the literature, nodes of D2B must control each other by periodical exchanges of pings between neighbors. When the failure of a node is detected, its neighbors act as for a leave of this node. The local lookup table of the faulty node is however lost. Nodes republish their keys periodically so that lost lookup tables can be reconstructed (see [13] for more detail). If a node loses all its neighbors, i.e., if all neighbors of a node quit brutally the network (without executing the leave procedure), then this node must recontact an entry point (one of those nodes in the network whose addresses are public), and simulate the leave procedure executed by its neighbors. To avoid an excessive use of the entry points, it is desirable that the disconnection of nodes be unlikely. Hence, it is desirable that the degree of a node be sufficiently large. The  $d$ -dimensional D2B has expected degree  $\Theta(d)$ . Hence, one can take  $d$  large enough so that a node has little chance to lose all its neighbors simultaneously. If one prefers to use the 2-dimensional D2B (say, for a sake of simplicity), an appropriate solution consists to systematically connect every node  $x_1 \dots x_k$  to at least  $\log n$  descendants of the form  $x_i \dots x_k y_1 \dots y_\ell$ , for  $i \geq 1$ . As a side effect, this solution provides shorter routes to the search/publish messages.

### 4.2.2 Optimized choice of node label

The maximum degree of D2B is determined by a ‘‘balls into bins’’ game. Given an interval  $I$  of  $[0, 2^m)$  of length  $2^m \log n/n$ , we have seen that the Chernoff bound insures that at most  $O(\log n)$  nodes have values in  $I$ , w.h.p., and hence the degree of any given node is  $O(\log n)$ , w.h.p. Using the result in [27], we

have seen that the maximum, taken over all intervals  $I$ , of the number of nodes having values in  $I$ , is  $O(\log n)$ , with probability  $1 - o(1)$ . This follows from the fact that throwing  $n$  balls at random into  $b$  bins, with  $n \simeq b \log b$ , results in a maximum number of balls in any bin of  $O(\log n)$  with probability  $1 - o(1)$ .

Now, in their seminal paper, Azar *et al.* [3] considered the following process: balls are thrown one by one;  $d$  bins are selected at random for each ball; the ball chooses the bin containing currently the least number of balls among the  $d$  selected bins. It is shown in [3] that, as  $n$  goes to infinity, the number of balls in the fullest box is  $\Theta(n/b + \ln \ln n / \ln d)$ , with probability  $1 - o(1)$ . Hence the deviation to the mean is exponentially less than if no choice is given to the balls, even for  $d = 2$ . This suggests to give a choice among  $d \geq 2$  different labels for each node that joins the network. Each joining node  $u$  chooses  $d$  temporary labels. For each temporary label  $L$ ,  $u$  computes how many keys would be assigned to it if choosing  $L$  as label. Node  $u$  chooses the label that maximizes the number of keys that will be under its responsibility. In this way, one expects the keys to be better balanced among nodes.

#### 4.2.3 Matching with the physical network

Connections between neighbors in the overlay network may not respect the locality constraints of the physical network (geographical, technological, etc.). Tapestry and some other content-addressable networks offer several alternative routes between any two nodes, and routing aims to select the best one (whose quality is often estimated as the round-trip time of a ping). However, the setting of the network itself is not optimized, and the routes are computed *a posteriori*. If a joining node selects several temporary labels, it may select the “best” labels among them. The selection could be performed according to the physical address of the neighbors, giving a preference to the label with neighbors that are close physically. An alternative choice could be based, as for Tapestry, on pings addressed to the neighbors.

## References

- [1] I. Abraham, B. Awerbuch, Y. Azar, Y. Bartal, D. Malkhi, and E. Pavlov. A generic scheme for building overlay networks in adversarial scenarios. In *17th Int. Parallel and Distributed Processing Symposium (IPDPS)*, 2003.
- [2] I. Abraham, D. Malkhi, and O. Dobzinski. LAND: Stretch  $(1 + \epsilon)$  locality-aware networks for DHTs. In *15th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, 2004.
- [3] Y. Azar, A. Broder, A. Karlin, and E. Upfal. Balanced allocations. *SIAM Journal on Computing*, 29(1):180–200, 1999.
- [4] L. Barrière, P. Fraigniaud, E. Kranakis, and D. Krizanc. Efficient routing in networks with long range contacts. In *15th Int. Symp. on Distributed Computing (DISC)*, volume 2180 of *LNCS*, pages 270–284, 2001.

- [5] J.-C. Bermond and P. Fraigniaud. Broadcasting and gossiping in de Bruijn networks. *SIAM Journal on Computing*, 23(1):212–225, 1994.
- [6] J.-C. Bermond and C. Peyrat. De Bruijn and Kautz networks: a competitor for the hypercube. In *Hypercube and distributed computing*, pages 279–293. Elsevier, 1989.
- [7] M. Datar. Butterflies and peer-to-peer networks. In *10th European Symp. on Algorithms (ESA)*, number 2461 in LNCS, pages 310–322. Springer, 2002.
- [8] N. de Bruijn. A combinatorial problem. *Koninklijke Nederlandse Academie van Wetenschappen Proc.*, 49:758–764, 1946.
- [9] A. Fiat and J. Saia. Censorship resistant peer-to-peer content-addressable networks. In *ACM/SIAM Symp. on Discrete Algorithms (SODA)*, pages 94–103, 2002.
- [10] P. Fraigniaud and P. Gauron. An overview of the content-addressable network D2B (brief announcement). In *22nd ACM Symp. on Principles of Distributed Computing (PODC)*, page 151, 2003.
- [11] P. Fraigniaud, C. Gavoille, and C. Paul. Eclecticism shrinks even small worlds. In *23rd ACM Symp. on Principles of Distributed Computing (PODC)*, pages 169–178, 2004.
- [12] A.-T. gai and L. Viennot. Broose: a practical distributed hashtable based on the de Bruijn topology. Technical Report 5238, INRIA, 2004.
- [13] K. Hildrum, J. Kubiawicz, S. Rao, and B. Zhao. Distributed object location in a dynamic network. In *14th ACM Symp. on Parallel Algorithms and Architecture (SPAA)*, pages 41–52, 2002.
- [14] F. Kaashoek and D. Karger. Koorde: a simple degree-optimal distributed hash table. In *2nd Int. Workshop on Peer-to-Peer Systems (IPTPS)*, 2003.
- [15] J. Kleinberg. The small-world phenomenon: an algorithmic perspective. In *32nd ACM Symp. on Theory of Computing (STOC)*, pages 163–170, 2000.
- [16] X. Li, J. Mista, and C. Plaxton. Active and concurrent topology maintenance. In *18th Int. Conference on Distributed Computing (DISC)*, volume 3274 of LNCS, pages 321–334. Springer, 2004.
- [17] X. Li and C. Plaxton. On name resolution in peer-to-peer networks. In *2nd ACM Workshop on Principles of Mobile Computing (POMC)*, pages 82–89, 2002.
- [18] N. Lynch, D. Malkhi, and D. Ratajczak. Atomic data access in content-addressable networks: A position paper. In *1st Int. Workshop on Peer-to-Peer Systems (IPTPS)*, 2002.
- [19] N. Lynch, D. Malkhi, and D. Ratajczak. Atomic data access in distributed hash tables. In *Peer-to-Peer Systems*, volume 2429 of LNCS Hot series, page 295. Springer-Verlag, 2002.
- [20] D. Malkhi, M. Naor, and D. Ratajczak. Viceroy: a scalable and dynamic lookup network. In *21st ACM Symp. on Principles of Distributed Computing (PODC)*, pages 183–192, 2002.

- [21] G. Manku, M. Bawa, and P. Raghavan. Symphony: distributed hashing in a small world. In *4th USENIX Symp. on Internet Technologies and Systems*, pages 127–140, 2003.
- [22] G. Manku, M. Naor, and U. Wieder. Know thy neighbor’s neighbor: The power of lookahead in randomized P2P networks. In *36th ACM Symp. on Theory of Computing (STOC)*, 2004.
- [23] P. Maymounkov and D. Mazieres. Kademlia: A peer-to-peer information system based on the XOR metric. In *1st International Workshop on Peer-to-Peer Systems (IPTPS)*, 2002.
- [24] M. Naor and U. Wieder. Novel architectures for P2P applications: the continuous-discrete approach. In *15th ACM Symp. on Parallelism in Algorithms and Architectures (SPAA)*, pages 50–59, 2003.
- [25] G. Pandurangan, P. Raghavan, and E. Upfal. Building low-diameter P2P networks. In *42th IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 492–499, 2001.
- [26] C. Plaxton, R. Rajaraman, and A. Richa. Accessing nearby copies of replicated objects in a distributed environment. In *9th ACM Symp. on Parallel Algorithms and Architectures (SPAA)*, pages 311–320, 1997.
- [27] M. Raab and A. Steger. Balls into bins – a simple and tight analysis. In *2nd Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*, volume 1518 of *LNCS*, pages 159–170. Springer, 1998.
- [28] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *ACM SIGCOMM*, pages 161–172, 2001.
- [29] A. Rowstron and P. Druschel. Pastry: scalable, decentralized object location and routing for large-scale peer-to-peer systems. In *18th IFIP/ACM Int. Conf. on Distributed Systems Platforms (Middleware)*, 2001.
- [30] K. Sripanidkulchai, B. Maggs, and H. Zhang. Efficient content location using interest-based locality in peer-to-peer systems. In *INFOCOM*, 2003.
- [31] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan. Chord: a scalable peer-to-peer lookup service for internet applications. In *ACM SIGCOMM*, 2001.
- [32] B. Zhao, J. Kubiatowicz, and A. Joseph. Tapestry: an infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, UC Berkeley, 2001.

# Appendix

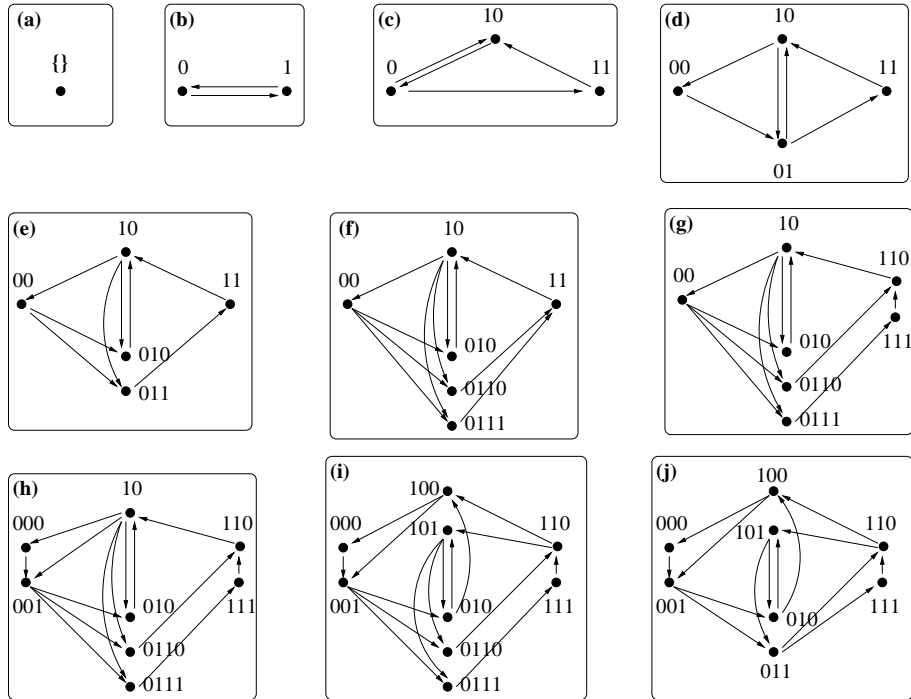


Fig. 5. An example of the behavior of D2B

On Figure 5, the first node entering the network (see (a)) takes the empty string  $\emptyset$  as label. When a second node joins (see (b)), this label is extended to 0 while the new node takes label 1. Then a new node joins (see (c)). Assuming that it chooses a temporary label  $1\dots$ , node labeled 1 extends its label to 10 while the new node takes label 11. A fourth node joins (see (d)). Assuming that it chooses a temporary label  $0\dots$ , node 0 extends its label to 00 while the new node takes label 01. The resulting network is the graph  $B(2, 2)$ . In (e), a new node joins with temporary label  $01\dots$ . In (f), a new node joins with temporary label  $011\dots$ . In (g), a new node joins with temporary label  $11\dots$ . In (h), a new node joins with temporary label  $00\dots$ . Note the out-degree 5 of node 10 in (h). In (i), a new node joins with temporary label  $10\dots$ . Node 100 and 101 have roughly half the degree of node 10 in (h). Finally, in (j), the node with label 0110 leaves the network, and thus node 0111 contracts its label to 011. The resulting network is the graph  $B(2, 3)$ , already depicted on Figure 2. The fact that steps (d) and (j) result in de Bruijn graphs is a coincidence, and in general the D2B topology is not isomorphic to the topology of a de Bruijn graph. However, the “expected topology” of the D2B overlay network is close to a de Bruijn graph for values of  $n$  that are close to powers of 2.