Graph-Theoretic Analysis of Structured Peer-to-Peer Systems: Routing Distances and Fault Resilience

Dmitri Loguinov, Anuj Kumar, Vivek Rai, Sai Ganesh
Department of Computer Science
Texas A&M University
College Station, TX 77843
dmitri,anujk,vivekr,ssai@cs.tamu.edu

ABSTRACT

This paper examines graph-theoretic properties of existing peer-to-peer architectures and proposes a new infrastructure based on optimal-diameter de Bruijn graphs. Since generalized de Bruijn graphs possess very short average routing distances and high resilience to node failure, they are well suited for structured peer-to-peer networks. Using the example of Chord, CAN, and de Bruijn, we first study routing performance, graph expansion, and clustering properties of each graph. We then examine bisection width, path overlap, and several other properties that affect routing and resilience of peer-to-peer networks. Having confirmed that de Bruijn graphs offer the best diameter and highest connectivity among the existing peer-to-peer structures, we offer a very simple incremental building process that preserves optimal properties of de Bruijn graphs under uniform user joins/departures. We call the combined peer-to-peer architecture ODRI – *Optimal Diameter Routing Infrastructure*.

Categories and Subject Descriptors

C.2.2 [**Communication Networks**]: Network Protocols

General Terms

Algorithms, Performance, Theory

Keywords

Peer-to-peer, Modeling, Graph Theory, DHT, de Bruijn

1. INTRODUCTION

In the last few years, peer-to-peer networks have rapidly evolved and have become an important part of the existing Internet culture. All current peer-to-peer proposals are built using application-layer overlays, each with a set of graph-theoretic properties that determine its routing efficiency and resilience to node failure. Graphs in peer-to-peer networks range from star-like trees (centralized approaches such as Napster) to complex k-node-connected graphs (such as Chord [40], CAN [31] and Pastry [35]). The performance of each peer-to-peer architecture is determined by the properties of these graphs, which typically possess $\Theta(\log N)$ diameter and $\Theta(\log N)$ degree at each node (where N is the number of peers in the system). Until recently [12], [20], [27], understanding whether these bounds were optimal and whether there existed *fixed-degree* graphs with $\log N$ diameter was believed to be the fundamental question of peer-to-peer research [32], [42].

Besides the diameter and degree, a third very important property of a peer-to-peer structure is its resilience to simultaneous node failure. Without fault resilience, the answer to logarithmic routing in fixed-degree networks is obvious and includes a variety of simple tree-like structures. Thus, the main goal of this work is to find a fixed-degree graph with not only *provably minimum* diameter, but also *maximum* connectivity (i.e., fault resilience) among all such graphs.

Another dimension to this work is to provide a unifying analytical framework for understanding the various properties of existing and future peer-to-peer graphs. Many of the existing proposals (e.g., CAN, Chord, Pastry) have not been modeled in ways that can provide a clear quantitative assessment of each graph's resilience to node failure. In addition to these classical approaches, there are proposals based on heuristics (e.g., [14], [26], [41]) with no provably-optimal underlying foundation for choosing one or another graph structure. Our work supplements such proposals with a more fundamental insight into the problem and offers analytical tools for evaluating future peer-to-peer routing architectures.

The paper is organized as follows. We first examine the problem of obtaining logarithmic routing diameter in fixed-degree (sparse) graphs. Our work relies on generalized de Bruijn graphs [19] of fixed degree k and asymptotically optimal diameter $\log_k N$. However, since the diameter itself does not tell the whole story, we also study the *average* distances between all pairs of nodes since this metric (rather than the diameter) determines the expected response time (i.e., number of hops) and the capacity of the peer-to-peer network.

We next examine clustering and small-world properties of each graph and explain how they relate to graph expansion. We derive that de Bruijn graphs have an order of magnitude smaller clustering coefficients than Chord, which explains the differences in expansion, resilience, and diam-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SIGCOMM'03, August 25–29, 2003, Karlsruhe, Germany.
Copyright 2003 ACM 1-58113-735-4/03/0008 ...$5.00.

eter between the two graphs. We then study the resilience of these graphs against node failure, or simply their *connectivity*. In general, connectivity determines the number and location of failures that a graph can tolerate without becoming disconnected. We examine edge bisection width in each graph and demonstrate that de Bruijn graphs are several times more difficult to disconnect than the traditional approaches. We also examine the amount of overlap between alternative (parallel) paths leading to any given destination and investigate how it affects the success of greedy routing under adversarial node failure.

Having confirmed that de Bruijn graphs offer a resilient and diameter-optimal routing structure, we provide an algorithm for building such graphs incrementally as peer nodes join and leave the system. We conclude the paper by showing that under uniform user joins, the diameter of the *peer-to-peer* graph remains asymptotically optimal.

2. BACKGROUND

Many current peer-to-peer networks [31], [35], [40], [43] are based on *distributed hash tables* (DHTs), which provide a decentralized, many-to-one mapping between user objects and peers. This mapping is accomplished by organizing the peers in some virtual coordinate space and hashing each object to these virtual coordinates. The information about each object (such as the IP address of the owner) is kept by the peer to whose coordinates the object hashes. Their distributed structure, excellent scalability, short routing distances, and failure resilience make DHTs highly suitable for peer-to-peer networks.

2.1 Peer-to-Peer DHTs

Many current DHTs [17], [35], [37], [43], [44] rely on the concept of prefix-based routing introduced by Plaxton *et al.* in [29]. Plaxton's framework is extended in *Tapestry* [43] and *Bayeux* [17], [44] to accommodate dynamic join/departure of peers and provide necessary failure-recovery mechanisms. A similar method based on generalized hypercubes (*Pastry*) is shown by Rowstron *et al.* in [35]. Using an alphabet of size b, Pastry builds a k-regular graph with diameter $\log_b N$ and node degree $(b-1)\log_b N, b \geq 2$. Among other approaches, Ratnasamy *et al.* [31] propose a peer-to-peer architecture called *Content-Addressable Network* (CAN) that maps the DHT to a d-dimensional Cartesian space. CAN's diameter is $1/2 dN^{1/d}$ and the degree of each node is $2d$. Stoica *et al.* [40] propose a distributed graph called *Chord*, which uses a 1D modulo field (ring) with the diameter and degree both equal to $\log_2 N$.

Recent proposals start to address the issue of routing in logarithmic time in *fixed-degree* graphs. For example, Considine *et al.* [8] expand on Chord's ring structure by constructing a digraph (directed graph) of fixed degree; however, the proposed structure needs to estimate the number of active nodes to properly build the application-layer graph. Among tree-based structures, Freedman *et al.* [13] propose a DHT based on distributed tries and Tran *et al.* [41] organize peers into a multicast tree of degree $O(k^2)$ and diameter $O(\log_k N)$. Xu *et al.* [42] study diameter-degree tradeoffs of current DHTs and propose a graph based on a modified static butterfly. As of this writing, the details of this framework are still under on-going investigation. Another peer-to-peer architecture based on butterfly networks (*Viceroy*) is shown in [25].

Independently of this work, several recent papers have also proposed de Bruijn graphs for peer-to-peer networks [12], [20], [27]. These developments are complementary to our investigation and provide implementation details and additional analysis not covered in this paper.

2.2 Fault Tolerance of DHTs

Fault tolerance of peer-to-peer networks is an equally important topic. Liben-Nowell *et al.* [23] examine error resilience dynamics of Chord when nodes join/leave the system and derive lower bounds on the degree necessary to maintain a connected graph with high probability. Fiat *et al.* [11] build a *Censorship Resistant Network* that can tolerate massive adversarial node failures and random object deletions. Saia *et al.* [36] create another highly fault-resilient structure with $O(\log^3 N)$ state at each node and $O(\log^3 N)$ per-message routing overhead. Unfortunately, very few studies examine the resilience of existing graphs *in comparison* with each other or attempt to understand whether this resilience can be improved while *preserving* the diameter and routing overhead of the graph. We are aware of only one comparison study, in which Gummadi *et al.* [15] find that ring-based graphs (such as Chord) offer more flexibility with route selection and provide better performance under random node failure compared to several other traditional DHTs.

2.3 Random Graphs

Another direction for building DHTs relies on properties of random graphs. The main thrust in this area is to build logarithmic-time routing structures with constant degree. Pandurangan *et al.* [28] propose a random DHT graph with a constant degree and (almost certainly) logarithmic diameter; however, the paper does not provide an efficient routing algorithm for the proposed structure that can deterministically explore the low diameter of the graph. Aspnes *et al.* [1] examine random graphs of fixed degree $l+1$ and derive upper and lower bounds on the expected routing distance in such graphs. Their results show that both bounds are proportional to $\frac{\ln^2 N}{l \ln \ln N}$. Law *et al.* [21] build random expander graphs based on Hamiltonian cycles with $O(\log N)$ diameter and $O(\log N)$ degree.

Even though random graphs of logarithmic diameter can be built with high probability using random neighbor selection, the design of efficient routing algorithms competitive to those in deterministic graphs is still an open issue.

2.4 Optimal-Diameter Graphs

The problem of designing an optimal-diameter graph of fixed degree has been extensively studied in the past. In one formulation of this problem, assume a graph of fixed degree k and diameter D (the maximum distance between any two nodes in the graph). What is the maximum number of nodes N that can be packed into any such graph? A well-known result is the *Moore bound* [6], [7]:

$$N \leq 1 + k + k^2 + ... + k^D = \frac{k^{D+1} - 1}{k - 1} = N_M. \quad (1)$$

Interestingly, Moore bound N_M is only achievable for trivial values of k and D. In fact, the Moore bound is *provably* not achievable for any non-trivial graph [6]. Directed de Bruijn graphs come close to the Moore bound and can be built with $N = k^D$ nodes [19] or even with $N = k^D + k^{D-1}$ nodes [33]. In general, it is not known how close we can

approach upper bound N_M for non-trivial graphs [7]. In the context of peer-to-peer DHTs, we are concerned with a different formulation of the problem: given N nodes and fixed degree k, what is the minimum diameter in any graph built on top of these N nodes? The answer follows from (1):

$$D \geq \lceil \log_k (N(k-1)+1) \rceil - 1 = D_M. \quad (2)$$

Imase and Itoh [19] construct nearly optimal de Bruijn graphs of diameter $D = \lceil \log_k N \rceil$, which is at most $D_M + 1$; however, for large k, the two diameters become asymptotically equal. In this paper, we use the same basic algorithms [19] even though they can be slightly improved [33].

Another very important metric related to the routing performance of a graph is its average distance μ_d between every pair of nodes (note that we include distances from a node to itself in μ_d while some of the related work does not). The lower bound on μ_d in any k-regular graph is given by the average distance in the corresponding Moore graph and is also not achievable for non-trivial values of N and k [38]:

$$\mu_d \geq D_M - \frac{k(k^{D_M}-1)}{N(k-1)^2} + \frac{D_M}{N(k-1)} \approx D_M - \frac{1}{k-1}. \quad (3)$$

With respect to μ_d, de Bruijn graphs are again asymptotically optimal and converge to the bound in (3) for sufficiently large N and k [38].

3. BASICS OF DE BRUIJN GRAPHS

3.1 Motivation

One of the goals of this work is to build a distributed hash table (DHT) on top of fixed-degree graphs with *provably* optimal routing diameter. Since non-trivial Moore graphs do not exist [6], we use de Bruijn graphs [19] of diameter $\lceil \log_k N \rceil$ and often call them "optimal" since among the class of practically achievable graphs with simple routing rules, they *are* optimal. To illustrate the impressive reduction in diameter compared to the classical DHT structures, assume 1 million nodes and degree k fixed at $\lceil \log_2 N \rceil = 20$. Under these circumstances, Chord offers a graph with diameter D equal to $\lceil \log_2 N \rceil = 20$, while a de Bruijn graph with the same number of neighbors has a diameter four times smaller: $D = \lceil \log_{20} N \rceil = \lceil 4.61 \rceil = 5$. Note that the diameter of the corresponding Moore graph is essentially the same: $D_M = \lceil 4.59 \rceil = 5$.

Throughout the paper, we are concerned with the properties of the underlying graph of each peer-to-peer network. Consequently, we examine the diameter and resilience of these graphs assuming that the hashing function equally spreads users along the DHT space and that all graphs are populated with the maximum number of nodes (this assumption is relaxed in section 7). We further assume for simplicity of notation that the total number of nodes N is a power of node degree and omit ceiling functions whenever appropriate.

3.2 Structure of de Bruijn Graphs

De Bruijn graphs [5], [19], [22], [38] are nearly optimal, fixed-degree digraphs of diameter $\log_k N$, where k is the fixed degree of each node and N is the total number of nodes. Note that de Bruijn graphs are *directed* graphs with k outgoing and k incoming edges at each node, which also holds for many other protocols [35], [40], [43]. Assume that each node x is hashed to a string H_x drawn from some alphabet Σ of size k. For example, if k is two, then H_x is a binary string of 0s and 1s. A directed de Bruijn graph [19] contains $N = k^D$ nodes where D is the diameter of the graph. Each node H_x in the graph is a string $(h_1, \ldots, h_D)$ of length D linked to k other nodes $(h_2, \ldots, h_D, \alpha)$, for all possible $\alpha \in \Sigma$. A classical de Bruijn graph for $k = 2$ and $N = 8$ is shown in Figure 1 (left) [38]. The diameter of the graph in the figure is 3, even though some nodes link to themselves. In fact, de Bruijn graphs contain exactly k nodes $(h, h, \ldots, h)$, for all $h \in \Sigma$, which link to themselves (this makes the graph irregular). It is possible to create a regular de Bruijn graph by linking these nodes to each other for added failure resilience. We discuss this in section 6.

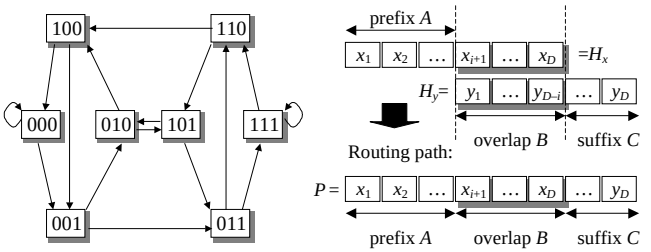

Figure 1: A de Bruijn graph of degree 2 and diameter 3 (left). Optimal routing from H_x to H_y (right).

3.3 Routing

Recall that shortest-path routing between any two nodes in de Bruijn networks follows a greedy procedure executed by individual nodes in a distributed fashion [19], [38]. Assume that node x seeks a shortest path to node y. The choice of the next-hop neighbor follows a simple string-matching algorithm shown in Figure 1 (right). Node x finds the *longest* overlap between the *suffix* of its hash index H_x and the *prefix* of y's hash index H_y. In the figure, the longest overlap, labeled B, contains $(D - i)$ digits, for some i. By merging prefix A, overlap B, and suffix C, node x can generate the entire path P to reach y. Notice that P starts with H_x, ends with H_y, and contains no more than $(D - 1)$ intermediate nodes (each node is a D-character substring of P, read from left to right). As an example, again consider the graph in Figure 1 (left). Suppose node 001 needs to route to node 101 along the shortest path. Using the above procedure, prefix A is 00, overlap B is 1, and suffix C is 01. The resulting shortest path is P = 00101, which translates to 001→010 →101.

3.4 Comparison with Existing Graphs

In this section, we briefly examine diameter-degree tradeoffs of the existing protocols and compare them to those of de Bruijn graphs. We leave a thorough analysis of numerous recently proposed graphs [14], [21], [25], [26], [41], [42] for future work and conduct a detailed study of two classical approaches Chord [40] and CAN [31] in sections 4-6. This section also shows results for Pastry and the static butterfly graph (without detailed analysis) and all together omits Tapestry since its diameter-degree tradeoff is very similar to that of Pastry.[1] Note that our treatment of the butter-

[1]Strictly speaking, Tapestry's routing table size is $b \log_b N$ instead of $(b-1)\log_b N$ [43].

Graph	Degree	Diameter D
de Bruijn	k	$\log_k N$
Trie	k+1	$2\log_k N$
Chord	$\log_2 N$	$\log_2 N$
CAN	$2d$	$\frac{1}{2}dN^{1/d}$
Pastry	$(b-1)\log_b N$	$\log_b N$
Classic butterfly	k	$2\log_k N(1-o(1))$

Table 1: Asymptotic degree-diameter properties of the different graphs.

fly graph follows the traditional definition [22], which is the basis of two recent proposals Viceroy [25] and Ulysses [42]; however, neither of these two graphs exactly implements the classic butterfly. Therefore, as we discuss below, the individual diameter-degree tradeoffs of these approaches are different from that in the classic graph. We further remove all fault-resilient additions of each structure (such as the predecessor pointer and r-element successor list in Chord) and only analyze the "raw" performance of each graph.

As an illustration of fixed-degree tree structures, we also examine k-ary tries as they have been recently proposed for DHTs [13]. A k-ary trie uses prefix-based routing over a tree where each parent maintains k children, one child for each symbol in the alphabet. Consequently, the maximum degree of any node in the trie is $k+1$ and the diameter of the graph is $2\lceil \log_k N \rceil$ (i.e., the distance to the root and back).

Finally, recall that the traditional butterfly network contains $N = mk^m$ nodes (where k is again the degree of each node) and has diameter $D = 2m$–1. Notice that D can be expressed in terms of degree k using Lambert's function W [22]:

$$D = 2m - 1 = 2\frac{W(N \ln k)}{\ln k} - 1 = 2\log_k N(1 - o(1)). \quad (4)$$

Even though butterflies are appealing graphs, there are non-trivial difficulties in building them as nodes join and leave the system. In one example, Viceroy [25] implements a binary butterfly with diameter $3\log_2 N$ and degree 7 and further requires estimation of the number of nodes in the system. In another example, Ulysses [42] adds $\log N$ neighbors to each node and is no longer a fixed-degree graph. As we show in section 7, distributed de Bruijn graphs possess no more conceptual complexity than Chord, achieve optimal diameter in the peer-to-peer graph, and can be built with a fixed application-layer degree.

Table 1 shows asymptotic diameter and node degree of de Bruijn graphs and several existing (deterministic) structures. First note that we assume that CAN uses *circular* (toroidal) routing in each of the dimensions, which means that all nodes along the borders maintain $2d$ neighbors and that the CAN graph is regular. Second, Chord allows a generalization to $d\log_2 N$ neighbors and diameter $\log_{1+d} N$ [40]; however, it is most frequently used with the default value of $d = 1$ shown in the table (which is also the version studied in this paper). Finally, the trie maintains its *average* degree over all nodes equal to only 2 (since approximately $\frac{k-1}{k}$ fraction of the nodes are leaves); however, the imbalance in the middle of the tree with nodes of degree $k+1$ creates a rather pessimistic diameter-degree tradeoff.

k	de Bruijn	Trie	Chord	CAN	Pastry	Classic butterfly
2	20	–	–	huge	–	31
3	13	40	–	–	–	20
4	10	26	–	1,000	–	16
10	6	13	–	40	–	10
20	5	10	20	20	20	8
50	4	8	–	–	7	7
100	3	6	–	–	5	5

Table 2: Graph diameter for $N = 10^6$ (cells with a dash indicate that the graph does not support the corresponding node degree).

We next examine the performance of these graphs in a hypothetical peer-to-peer system of $N = 10^6$ nodes. Table 2 shows the diameter of each graph as a function of its degree k. Notice that for low-degree networks ($k \leq 20$), even the trie offers a better diameter than the three classical approaches (i.e., CAN, Chord, and Pastry). In fact, the trie routes in *half* the time compared to Chord or CAN. Also notice that de Bruijn graphs with the same number of neighbors offer diameters at least four times smaller than those in Chord and CAN. Furthermore, de Bruijn graphs can route between any pair of nodes in 20 hops with only 2 neighbors, which is 10 times less than that required by CAN, Chord, or Pastry to achieve the same diameter. Finally, the traditional butterfly offers a diameter approximately 50-60% larger than that in de Bruijn graphs for all values of k.

One interesting observation about CAN points to the fact that selection of the number of dimensions d is an important decision for a given number of nodes N. It is noted in [31] that d is likely to be fixed while N changes; however, as Table 2 shows, many small values of $d \ll \log_2 N$ result in greatly suboptimal diameters. This observation is easy to explain since CAN's diameter $\frac{1}{2}dN^{1/d}$ is a strictly convex function with a unique minimum located at $d = \ln N$ (e peers per dimension). Keeping in mind that each dimension must contain an integer number of peers, the best practical diameter is achieved for $d = \log_3 N$. Thus, for $N = 10^6$, the optimal number of dimensions d is 12 ($k = 24$ neighbors) and the optimal diameter is 19. Also note that when $d = \frac{1}{2}\log_2 N$, CAN's degree and diameter are *both* equal to that of Chord (this is shown in Table 2 for $k = 20$ and also noted in [31], [40]).

Further examining Table 2, notice that Pastry offers a good diameter only for large $b \gg 2$. In fact, to come within one hop of the optimal diameter for $N = 10^6$, Pastry requires *at least* 160 neighbors (not shown in the table). Such large routing tables may be impractical in the real Internet due to high volume of traffic required to maintain peer-level connections and repair broken links when existing neighbors frequently fail. From this angle, bounded-degree graphs are preferable.

On the other hand, Pastry has an advantage over other structures in its ability to employ proximity-based peer selection. In theory, such graph construction is possible in Chord, CAN, and de Bruijn, but it requires a less-transparent implementation. Since the benefits of topologically aware peer-to-peer networks are hard to quantify with respect to other metrics, our current study only focuses on pure graph performance of each approach. Construction of topologi-

cally aware de Bruijn graphs is a possible topic for future investigation.

4. ROUTING ANALYSIS

De Bruijn graphs have desirable properties for peer-to-peer networks that stem from their small diameter. However, the diameter of a graph is simply the largest distance between any pair of nodes and only provides an *upper bound* on the delay (number of hops) experienced by the users. A much more balanced metric is the *average* distance between any pair of nodes since this is the performance a user can expect from the peer-to-peer system when searching for objects. In fact, it is possible to reduce the diameter of a graph and at the same time increase its average routing distance as was recently demonstrated in [42].

Define $d(x,y)$ to be the shortest distance between nodes x and y in a given graph. To better understand how the distribution of $d(x,y)$ is formed and study expansion properties of each graph, the full version of the paper [24] first derives the density (mass) function of $d(x,y)$ and then computes its expectation μ_d. In this paper, we present the most important results from [24] and omit all technical proofs.

4.1 Chord

Stoica *et al.* [40] showed in simulation that the average inter-node distance μ_d in Chord is $\frac{D}{2}$ and offered a simple explanation of why this happens. They further showed the distribution of $d(x,y)$ to be bell-shaped as demonstrated in Figure 2 (left) for N = 1,024. The histogram appears to be Gaussian as illustrated by an almost-perfect fit of a Gaussian distribution in the figure. It has been noted before that certain real-world graphs (such as those describing webpage linkage structure [3]) exhibit Gaussian distributions of $d(x,y)$, but no explanation of why this happens has been offered. Below, we analyze Chord's distribution of shortest distances, understand why it appears to be Gaussian, and provide additional qualitative insight into the structure of the graph using "small-world" terminology.

The following lemma is proved in [24].

LEMMA 1. *Each node in Chord can reach exactly C_n^D nodes at shortest distance n.*

Using symmetry of nodes in Chord and the result of this lemma, the PMF (probability mass function) of $d(x,y)$ is given by a binomial distribution with parameters $p = q = 1/2$ (recall that $N = 2^D$):

$$p(n) = \frac{C_n^D}{N} = \frac{C_n^D}{2^n 2^{D-n}} = C_n^D p^n q^{D-n}, \qquad (5)$$

where $p(n)$ is the PMF of shortest distances $d(x,y)$. Our simulation results confirm that (5) gives the *exact* distribution of shortest path lengths in Chord. The expected value μ_d of a binomial random variable is a well-known result and equals Dp, or simply $\frac{D}{2}$. This provides an alternative derivation of the result previously shown in [40].

The reason why the distribution of shortest distances in Chord appears to be Gaussian is explained by the de Moivre-Laplace theorem, which states that the binomial distribution in (5) asymptotically tends to a Gaussian distribution with mean $Dp = \frac{D}{2}$ and variance $Dpq = \frac{D}{4}$ for sufficiently large D. Even though we have not provided an insight into why certain Internet graphs exhibit Gaussian distributions of shortest paths, we found a clear explanation of this phenomenon in Chord.

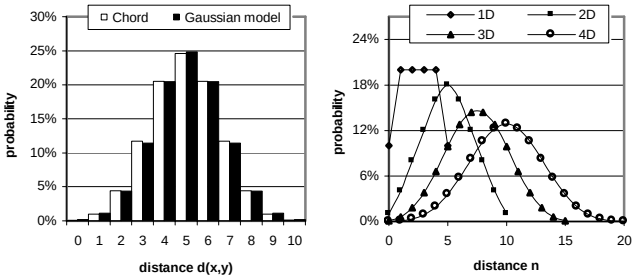

Figure 2: Distribution of shortest paths $d(x,y)$ in Chord for N = 1,024 together with a Gaussian model (left). Shortest path distribution in CAN for N = 10 (1D), 10^2 (2D), 10^3(3D), 10^4 (4D) nodes (right).

There is also a simple intuitive link between the bell shape of the curve in Figure 2 (left) and the expansion properties of the graph. As the distance from any given node x increases, the number of *new* neighbors found by the search slowly saturates and starts declining after half the nodes have been reached. This means that many of the newly found nodes link to some of the previously discovered nodes. This leads to a situation where the new neighbors "know" many of the old neighbors, which is often called the *small-world property* (or *clustering*) of the graph [3], [4]. In graph theory, the growth in the number of new neighbors discovered at a certain distance is related to *node expansion* of the graph. Quickly expanding graphs maintain an exponentially increasing number of new neighbors up to the diameter of the graph, which means that very few of the new neighbors "know" the old ones (and hence their clustering coefficients are virtually zero). We study these phenomena more carefully in section 5, but currently conjecture that we should expect reasonably high clustering and low expansion from Chord.

4.2 CAN

Recall that CAN organizes its nodes into a d-dimensional Cartesian space. We first examine the average distance in this graph and then show that for the same degree, CAN's distribution of routing distances becomes identical to that in Chord. The following result in proved in [24] and is mentioned for the even values of N in [31].

LEMMA 2. *The expected distance between any pair of nodes in CAN is $\frac{D}{2}$ for even N and $\frac{2D+d}{4} - o(1)$ for odd N.*

Our next lemma shows that as the number of dimensions d increases, CAN's distribution of shortest paths becomes Gaussian as well [24].

LEMMA 3. *For large d, CAN's distribution of shortest distances $p(n)$ is Gaussian.*

This lemma is illustrated in Figure 2 (right) for four different values of d. As the figure shows, starting with d = 4, the PMF function $p(n)$ becomes Gaussian with very high accuracy (the Gaussian model is not shown in the figure).

As noted in section 3.4, when $d = 1/2 \log_2 N$, CAN's degree and diameter are both $\log_2 N$, or those of Chord. We

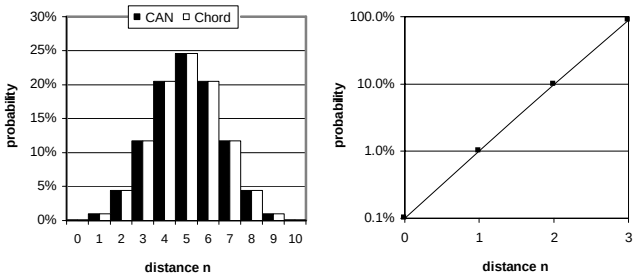

Figure 3: Comparison between Chord's and CAN's shortest path distributions for $N = 1{,}024$ and $d = 5$ (left). Distribution of shortest distances in de Bruijn for $N = 1{,}000$ and $k = 10$ (right).

call such CAN "logarithmic" and note that the size of its dimensions is $N^{1/d} = 4$ peers.

LEMMA 4. *The distribution of shortest distances in logarithmic CAN ($d = \frac{1}{2}\log_2 N$) is binomial and identical to that in Chord.*

The result of this lemma is illustrated in Figure 3 (left) for N= 1,024, $d = 5$, and $D = 10$, which shows a perfect match between the two graphs (the distributions also match numerically). Note that the only problem that prevents CAN from "becoming" Chord is the difficulty of dynamically adjusting the number of dimensions d. Even though CAN offers more flexibility with the choice of d, the choice itself is very difficult as it must be made a-priori and can only rely on the *expected* number of nodes in the system. Small (fixed) values of d prevent the system from scaling to large N, while large values of d are inefficient in settings where N happens to be small. Therefore, it appears that in practical networks of non-trivial size, the ability to scale its degree *dynamically* provides Chord with a clear advantage over CAN.

4.3 De Bruijn

From the diameter perspective, de Bruijn graphs offer significantly smaller end-to-end upper bounds on routing time. However, the improvement by a factor of 4 over Chord for $N = 10^6$ no longer holds when we examine the *average* shortest distances in each graph. Nevertheless, the improvement in μ_d is still substantial, but drops down to about a factor of two faster than Chord as we demonstrate below.

In general, the distribution of de Bruijn's distances $d(x, y)$ is very complicated and there is no known closed-form expression for its PMF $p(n)$ [38]. In the full version of the paper [24], we derive a simple formula for $p(n)$ that is *exact* for all graphs of diameter $D \leq 3$ and is very close to the real $p(n)$ for the rest of the graphs.

LEMMA 5. *The asymptotic distribution of shortest distances in de Bruijn graphs is given by:*

$$p(n) \approx \frac{k^n}{N} - \frac{k^{2n-1}}{N^2} \geq \frac{k^n - k^{n-1}}{N}. \quad (6)$$

It immediately follows from the lemma that de Bruijn graphs expand *exponentially* and that the majority of nodes are reachable at shortest distance D from each node v. This is demonstrated in Figure 3 (right), which shows de Bruijn's $p(n)$ for N = 1,000 and $k = 10$ (note the log scale of the y-axis). Intuitively, it is clear that the average distance in de Bruijn graphs must be very close to diameter D and that the local structure of the graph at each node looks like a tree (i.e., very few short cycles and low clustering). We examine the cyclic structure of each graph in section 5 and in the meantime, focus on de Bruijn's average distance μ_d.

k	Moore graph	de Bruijn	Chord	CAN	Classic butterfly
2	17.9	18.3	–	huge	22.4
3	11.7	11.9	–	–	14.7
4	9.4	9.5	–	500	11.8
10	5.8	5.9	–	19.8	7.3
20	4.5	4.6	10.0	10.0	5.7
50	3.5	3.5	–	–	4.3
100	2.98	2.98	–	–	3.65

Table 3: The average distance in each graph ($N = 10^6$ nodes).

LEMMA 6. *The average distance in de Bruijn graphs is asymptotically:*

$$\mu_d \approx D - \frac{1}{k-1}. \quad (7)$$

As expected, the average distance in de Bruijn graphs cannot shrink much beyond its already very small diameter D. In fact, for large values of N, the average distance μ_d in (7) asymptotically tends to D as k becomes large.

4.4 Butterfly

The final graph we examine in this section is the classic butterfly. Even though its diameter and average distance are close to optimal, they are always higher than those in (non-trivial) de Bruijn graphs. Recall that the average distance in the butterfly graph is given by the following [18]:

$$\mu_d = \frac{3m-1}{2} - \frac{1}{k-1} + \frac{m}{N(k-1)} \approx \frac{3\log_k N}{2}, \quad (8)$$

which, for large N and k, is asymptotically 50% larger than the same metric in de Bruijn graphs.

4.5 Discussion

The results of this section indicate that de Bruijn graphs offer not only provably-optimal diameter D, but also smaller average routing times compared to Chord, CAN, and the static butterfly. As shown in Table 3 for $N = 10^6$, the average distance in de Bruijn graphs is still smaller than half of that in Chord and CAN for the same number of neighbors and 22% smaller than that in the butterfly. Also notice that for large k, μ_d in de Bruijn graphs converges to the best possible average distance of Moore graphs shown in the first column of Table 3.

This result has several practical implications. First, μ_d determines the expected delay in the graph and represents a measure of responsiveness of the system to user searches. Second, the average distance determines the *capacity* of a peer-to-peer network, where the capacity is a term widely used in interconnection and wireless networks to define the throughput available to each node under random communication patterns within the network. Since each peer must forward requests for other peers, the expected useful capacity of a node is determined by the inverse of μ_d (i.e., for each

useful request a node makes, it must forward on average μ_d other requests).

Assuming fixed transmission bandwidth and discounting interference effects, the average capacity $c(G)$ of wireless ad-hoc networks is $O(1/\sqrt{N})$ due to spatial restrictions on connectivity [16], while both Chord and logarithmic CAN maintain an average capacity of $\frac{1}{\mu_d} = \frac{2}{\log_2 N}$. Compared to wireless networks, this is a much better bound; however, it is still several times lower than that in de Bruijn graphs. Even assuming a worst-case average distance $\mu_d = D$ in de Bruijn graphs, their average capacity with $\log_2 N$ neighbors is superior to Chord's for all $N > 16$:

$$c(G) = \frac{1}{\log_k N} = \frac{\log_2 \log_2 N}{\log_2 N}. \quad (9)$$

In fact, the ratio of these two capacities grows infinitely large (albeit very slowly) for large N. Compared to the static butterfly, de Bruijn graphs offer asymptotically 50% more capacity and at least 22% more capacity in graphs of practical size examined in this work.

In the current Internet, each search request typically carries a small amount of information and it is not clear at this point whether future peer-to-peer systems will be utilized to the point of their ultimate capacity. Nevertheless, we believe that it is desirable to design the underlying structure of the application-layer graph to be able to carry as many concurrent requests as possible. Thus, we must conclude that de Bruijn graphs offer clear benefits in terms of expected capacity and routing distances (delay) over the existing approaches.

There exists, however, one drawback of fixed-degree graphs based on the static butterfly and de Bruijn graphs. In both cases, *edge* load distribution is not uniform and results in some edges carrying more load (on average) than others. In addition, de Bruijn graphs demonstrate non-uniform *node* load distribution. This is a well-studied area in interconnection networks and there are many improvements to the routing protocols that can distribute the load in de Bruijn and butterfly networks more evenly [18], [38], [42]. We leave exploration of this direction for future work.

Next, we investigate clustering and then resilience features of de Bruijn graphs before addressing their practical use in peer-to-peer networks.

5. CLUSTERING AND EXPANSION

Following significant research effort to model the structure of the current Internet, it was discovered that many of the existing topology generators did not accurately match the "small-world" (clustering) properties of the Internet graph [3], [4]. Clustering is a very interesting concept that is found in many natural phenomena and that determines how tightly neighbors of any given node link to each other. In what follows, we examine clustering in Chord, CAN, and de Bruijn graphs, study graph-theoretic semantics behind the clustering coefficient, and show why metrics related to clustering are important concepts for peer-to-peer systems.

Given graph $G = (V, E)$, node $v \in V$, and its neighborhood $\Gamma(v) = \{u\colon (v,u) \in E\}$, clustering coefficient $\gamma(v)$ is defined as the ratio of the number of links $L(\Gamma(v))$ that are entirely contained in $\Gamma(v)$ to the maximum possible number of such links (if the graph is undirected, each link in $L(\Gamma(v))$ is counted twice):

$$\gamma(v) = \frac{L(\Gamma(v))}{|\Gamma(v)|(|\Gamma(v)| - 1)}. \quad (10)$$

Graph clustering $\gamma(G)$ is the average of $\gamma(v)$ for all vertices v with degree at least 2. The main questions that we study in this section are: *what exactly does clustering mean and how does it affect the properties desirable in peer-to-peer networks?*

5.1 Clustering Coefficients

We first present the values of clustering coefficients of all three graphs (for proofs, see [24]) and then explain the meaning of our results.

LEMMA 7. *Chord's clustering coefficient is* $\frac{1}{\log_2 N}$.

LEMMA 8. *De Bruijn's clustering coefficient is* $\frac{k-1}{N}$.

Notice that de Bruijn's $\gamma(G)$ decays to zero much quicker than Chord's confirming our earlier conjecture based on the distribution of shortest paths in section 4. The derivation of $\gamma(G)$ for CAN is much simpler as one can easily notice that *none* of the nodes in any neighborhood link to each other. Hence, CAN's $\gamma(G)$ is zero. This is somewhat counter-intuitive since CAN's number of new neighbors becomes saturated at $\frac{D}{2}$ just as in Chord and therefore its clustering properties should be similar to Chord's. We next examine the reasons behind this phenomenon and generalize clustering to become a global metric.

5.2 Cycles

There are two ways to better understand what clustering means and assess its importance for peer-to-peer networks. The first insight is based on cycles. Given a k-regular *undirected* graph G, it is easy to notice that the number of 3-cycles per node determines the clustering coefficient of the graph. Recall that an n-cycle is a path that starts and ends in the same node and contains exactly n edges[2]. Hence, any 3-cycle must involve two direct neighbors of node v, which results in clustering.

Since one goal in peer-to-peer networks is to reach as many nodes as possible within a certain number of hops, cycles that lead back to the original node where the request started are not very helpful. Another goal of peer-to-peer networks is to provide a fault-resilient environment where a simultaneous collapse of several nodes does not separate the graph into disjoint components. Short cycles mean that paths from any node x through different neighbors leading to any destination y must *overlap* with each other. This is not desirable since multiple parallel paths to y may be compromised when nodes in the neighborhood fail. This is shown in Figure 4 (left) where failure of node 1 leaves x with no path leading outside of its neighborhood. In fact, when node 1 fails, nodes 2 and 3 are also disconnected from the rest of the graph since all of their outgoing (as well incoming) edges are locally clustered.

Now we come back to the issue of why CAN has zero clustering, but identical shortest-path properties to those found in Chord. The absence of 3-cycles in CAN is explained by

[2]Usually, these paths are required to be edge and/or node disjoint, but this always holds for 3-cycles.

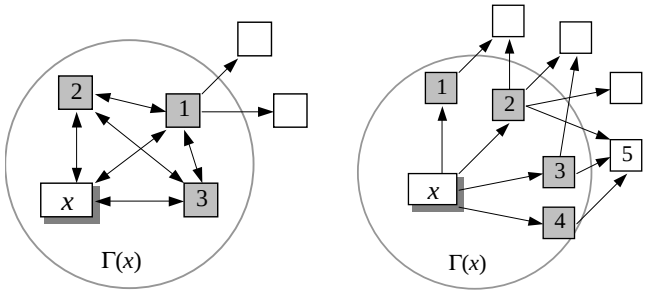

Figure 4: High clustering leads to weak connections outside a neighborhood (left). A more generic definition of clustering (right).

the fact that it has no *odd* cycles whatsoever, but it does have plenty of *even* cycles. In fact, the number of 4-cycles in CAN is roughly the same as in undirected Chord with the same number of neighbors. Consequently, *local* properties captured by the clustering coefficient do not necessarily mean much for graphs like CAN where only "friends of friends" have common acquaintances while direct friends of node x never know each other. This is illustrated in Figure 4 (right), where clustering coefficient $\gamma(x)$ is zero, but nodes 2, 3, 4 all link to the same "friend of a friend" node 5.

The concept of n-cycles applies to *directed* graphs as well; however, it does not directly produce the clustering coefficient because of a stricter nature of directed cycles. These difficulties lead us to generalize the framework of clustering using expansion analysis below.

5.3 Graph Expansion

Examine Figure 4 (right), which shows how undirected 4-cycles contribute to a graph's global clustering properties. *Global clustering* is a concept of "friends knowing each other" generalized to "friends knowing each others' friends". Although the previous discussion of cycles allows one to account for these cases, we seek a more generic and useful definition of clustering that goes beyond n-cycles ($n \geq 3$) and has a simple closed-form analytical expression for all three graphs.

We next study graph *expansion*, which determines how quickly the graph finds "unknown" nodes. Consider graph $G = (V, E)$ and select some of its nodes into set $S \subset V$. Define the set of all edges between S and the rest of the graph $V \backslash S$ to be $\partial S = \{(u, v)$: $(u, v) \in E$, $u \in S$, $v \in V \backslash S\}$. ∂S is called the *edge boundary* of S. Edge expansion $i(S)$ is defined as the ratio of the size of ∂S to the size of S:

$$i(S) = \frac{|\partial S|}{|S|}. \tag{11}$$

It is easy to see the relationship of $i(S)$ to clustering. Select S to be the neighborhood $\Gamma(v)$ of some node v. Therefore, $|S| = k$ and the number of edges contained within S is $k^2 - |\partial S|$, generically assuming a k-regular graph. Then the clustering coefficient of v is given by:

$$\gamma(v) = \frac{k^2 - |\partial S|}{k(k-1)} = \frac{k^2 - i(\Gamma(v))k}{k(k-1)} = \frac{k - i(\Gamma(v))}{k-1}. \tag{12}$$

Edge expansion determines the strength of the graph in the presence of edge failure. Clearly, a larger clustering coefficient in a k-regular graph implies smaller $i(\Gamma(v))$ as seen in (12) and generally leads to weaker graphs.

Definition 1. *Graph edge expansion (sometimes called the isoperimetric number of the graph) $i(G)$ is the minimum of $i(S)$ for all non-empty sets $S \subset V, |S| \leq \frac{|V|}{2}$.*

Notice that by examining $i(G)$, we no longer focus on *local* clustering, but rather on global properties of the graph and its resilience to edge failure over *all possible* sets S. Edge expansion tells us how many edges link outside any set S; however, it does *not* tell us if the outgoing edges link to the same node multiple times. For example, in Figure 4 (right), there are 8 edges leaving neighborhood $\Gamma(x)$, but they link to only 4 unique nodes, which indicates a good amount of path overlap. Edge expansion tells us the size of the edge cut between $\Gamma(x)$ and the rest of the graph, which is a useful analysis tool for studying a graph's resilience when *edges* are expected to fail (i.e., 8 edges in the cut are better than 4). In peer-to-peer systems, *node* failure is much more common than edge failure, in which case regardless of how many edges cross the cut, the strength of the neighborhood is determined by the number of nodes on the other side of ∂S. Hence, from the resilience perspective of peer-to-peer networks, it makes more sense to examine *node* expansion of the graph as we define below.

Definition 2. *Consider graph $G = (V, E)$ and some subset of nodes $S \subset V$. Define the node boundary of S to be $\partial S = \{v : (u, v) \in E, u \in S, v \in V \backslash S\}$. Node expansion $h(G)$ of the graph is given by:*

$$h(G) = \min_{\{S:|S| \leq |V|/2\}} \frac{|\partial S|}{|S|}. \tag{13}$$

Both $i(G)$ and $h(G)$ are related to edge and node bisection widths of the graph and generally are NP-complete problems. Furthermore, even after many years of research, the exact expression of these metrics for de Bruijn graphs remains unknown. Below, we limit our analysis to sets S that are neighborhoods of a given node (i.e., balls centered at the node) and study graph expansion that explains how well each ball is connected to the rest of the graph. Note that these balls do not necessarily represent the weakest sets S of each graph and do not, in general, achieve the minimum bound in (13). Derivation of better bounds on $h(G)$ is the topic of on-going research.

Recall that ball $B(v,n)$ of radius n centered at node v contains all nodes reachable from v in no more than n hops. In other words: $B(v,n)=\{u$: $d(v,u) \leq n\}$. It is easy to notice that the boundary of a ball is simply $\partial B(v,n)=\{u$: $d(v,u) = n+1\}$ and that our derivations in section 4 can be applied to study expansion (and global clustering) of each graph. Both logarithmic CAN and Chord have the same expansion properties since their distributions of $d(x,y)$ are identical. Hence, from now on, we only consider Chord.

Lemma 9. *Chord's ball expansion $h_B(G)$ is asymptotically $\Theta(1/\sqrt{\log_2 N})$.*

This function slowly decays from 0.45 for $N = 2{,}048$ to 0.27 for $N = 10^9$. Contrast this result with that in de Bruijn graphs below, which maintain constant connectivity $h_B(G)$ for all ball sizes and all values of N.

Lemma 10. *De Bruijn's ball expansion $h_B(G)$ is no less than $k - 1$.*

De Bruijn graphs expand so quickly, they actually approach the maximum possible bound on $h_B(G)$ and keep all balls $B(v,n)$ connected to the rest of the graph through at least $(k-1)|B(v,n)|$ external nodes. In fact, this not only explains the low diameter of de Bruijn graphs, but also leads to two important results. First, clustering in de Bruijn graphs is minimal at both local *and* global levels since exponential neighborhood expansion is preserved for all balls smaller than the graph itself. Second, path overlap in the graph is virtually non-existent due to little global clustering. This means that *shortest* parallel paths towards any given destination are expected to be node-disjoint with high probability.

In the next section, we study fault resilience of these graphs and examine whether k-node-connectivity of each graph can always be exploited using greedy routing.

6. RESILIENCE

6.1 Generic Methods

Classical failure analysis in peer-to-peer networks (e.g., [23], [40]) focused on analyzing the probability that a given node x becomes disconnected under a p-percent node failure. This amounts to computing the probability that all k neighbors of x fail simultaneously and leads to very small individual failure probabilities p^k for most practical networks. Also note that results derived using this method hold for any k-regular graph, regardless of its internal structure. Clearly, this analysis is insufficient to distinguish between all k-regular graphs since some of them may contain "weak" parts that can partition the graph into several disjoint components while no *single* node is completely disconnected from its component.

Another approach often used in classical fault resilience analysis is to examine k-node-connectivity of the graph in question. Given our graph structures, we show below that this metric does not lead to any significant insight either.

DEFINITION 3. *A k-regular graph is k-node-connected if there are k node-disjoint paths between any pair of nodes.*

This implies that a k-node-connected graph can tolerate the failure of any $k-1$ nodes without becoming disconnected and that the diameter of the graph after any $k-1$ nodes have failed is at most $D+1$. Both CAN and Chord are k-node-connected[3], while de Bruijn graphs are not due to several "weak" nodes with self-loops. This classical form of de Bruijn graphs has been shown to be $(k-1)$-node connected [39]; however, we seek to achieve maximum fault tolerance, which leads us to removing the loops and linking these k "weak" nodes to each other. Consider node $(h,h,\ldots,h)$, $h \in \Sigma$, with a self-loop. A *chain-linked* de Bruijn graph has directed links $(h,h,\ldots,h) \rightarrow (g,g,\ldots,g)$, for all $h \in \Sigma$ and $g = (h+1) \bmod k$. Recent development in consecutive-d graphs [9] also studied chain-linked de Bruijn graphs and proved that they are k-node connected.

What we know so far from classical peer-to-peer network analysis and maximum fault-tolerance metrics is that all three graphs are similar in their resilience. Hence, we seek additional methods that can distinguish between the fault tolerance offered by each graph. One such metric is *bisection width* [22], which is defined as the smallest number of (possibly directed) edges between any two equal-size partitions of the graph. Graph bisection width determines the difficulty of splitting the graph into giant components by failing individual edges. We next examine this metric in all three graphs.

[3]This can be shown for CAN by generating all possible orders of traversing d-dimensional paths between any pair of nodes. Chord's connectivity is easily derived from the well-known properties of hypercubes.

6.2 Bisection Width

Note that besides determining resilience, bisection width of a graph often provides tight upper bounds on the *achievable* capacity of the graph. Assume that each node sends messages to random destinations at a certain fixed rate. This communication pattern generates N messages per time unit. Each message is replicated μ_d times (on average) and each edge is expected to carry $\frac{N\mu_d}{Nk} = \frac{\mu_d}{k}$ messages per time unit. Note, however, that this analysis assumes that the combined load is *equally* distributed between all edges. There may be bottlenecks in which the load is significantly higher than the average and the resulting throughput capacity of the graph may be lower than the expected (mean) value.

Recall that approximately half of all communication in the graph is expected to cross the bisection cut. Thus, if this part of the graph is narrow (contains only a few edges), it will lead to congestion and inability of the graph to carry its expected load. One example of graphs with unacceptably small bisection width are trees, which are susceptible to both easy disconnects and severe congestion near the root.

LEMMA 11. *Chord's bisection width $bw(G)$ is N.*

Note that this value is double the bisection width of binary hypercubes since Chord uses *directed* links while hypercubes are undirected.

LEMMA 12. *Assuming the size of each dimension is even, CAN's bisection width $bw(G)$ is $2N^{(d-1)/d}$.*

Applying this result to logarithmic CAN ($d = \frac{1}{2}\log_2 N$), notice that its bisection width is $\frac{N}{2}$. Furthermore, if we view undirected links of logarithmic CAN as being composed of two directed edges, its bisection width matches that of Chord. Also note that CAN achieves its maximum $bw(G)$ when $d = \log_3 N$ and that all *sub-logarithmic* values of $d \ll \log_2 N$ result in "weaker" graphs. This is another way of showing that CAN with small fixed values of d may not be competitive to Chord in practical settings.

The bisection width of the butterfly is $\frac{kN}{2m}$ [22], where m is given by Lambert's function W in (4). Asymptotically, this bisection becomes $\frac{kN}{2\log_k N}$, although for small N it is slightly better. Finally, the exact value of $bw(G)$ of de Bruijn graphs is unknown and the best available upper and lower bounds differ by a factor of four [34]:

$$\frac{kN}{2\log_k N}(1-o(1)) \leq bw(G) \leq \frac{2kN}{\log_k N}(1+o(1)). \qquad (14)$$

Using the lower bound in (14), the bisection width of de Bruijn graphs for $k = \log_2 N$ is larger than that in Chord or CAN by a factor of $\frac{1}{2}\log_2\log_2 N$ (which is 2.2 for $N = 10^6$) and is generally no worse than that in the butterfly. It is further conjectured that the actual bisection width of de Bruijn graphs is at least 40% higher than the pessimistic lower bound used in the above comparison [10].

In summary, larger values of bisection width in de Bruijn graphs point towards higher resilience against graph partitioning and lower congestion in the bisection cut in addition to their optimal routing established earlier. Below, we examine several other resilience metrics that may lead to an even better understanding of de Bruijn's fault tolerance.

6.3 Path Overlap

So far, we have been omitting a great deal of simulation results since most of the derived formulas in this paper were exact. In this section, we examine several *empirical* metrics and back our analysis of these metrics with simulations.

Define set $P(x,y)$ to contain all vertices along *some* path from x to y. Denote by $Q(x,y)$ the set of all vertices in $P(x,y)$ except x and y: $Q(x,y) = P(x,y)\backslash\{x \cup y\}$. A graph's k-node-connectivity means that for every pair of nodes (x,y), there are exactly k pair-wise non-overlapping paths $P_1(x,y)$, $\ldots$, $P_k(x,y)$: $Q_i(x,y) \cap Q_j(x,y) = \emptyset$ for all $i \neq j$. Node-disjoint paths are very attractive to peer-to-peer networks as they provide *independent* backup routing options when the main shortest path fails.

Now notice that even though all three graphs under study have k node-disjoint paths for each pair (x,y), not all of these paths can be found using *greedy* routing at each node. In fact, in order to deterministically find all k non-overlapping paths between x and y in a *generic* graph, one needs to flood the entire graph using breadth-first search or similar techniques. Below we examine how well the routing rules in each graph are able to find non-overlapping paths and what happens to the diameter of the graph when nodes along the best path are failed.

For any pair of nodes (x,y), define $P_i(x,y)$ to be the *shortest* (according to the greedy routing rules of the corresponding graph) path to y through x's neighbor i. We are interested in the structure of these shortest paths, because when the best neighbor towards y fails, the graph routes through the second-best neighbor also trying to achieve the shortest path to y. Hence, if these paths overlap and nodes common to multiple parallel paths fail, both the diameter and connectivity may significantly deteriorate.

Define $T(x,y)$ to be the total number of vertices in all shortest paths $P_i(x,y)$ from x to y and $U(x,y)$ to be the number of *unique* vertices in all such paths: $T(x,y) = \Sigma|P_i(x,y)|$ and $U(x,y) = |\cup P_i(x,y)|$. Further define the average percentage $U(G)$ of unique nodes in all parallel paths:

$$U(G) = \frac{\sum_x \sum_y U(x,y)}{\sum_x \sum_y T(x,y)}. \quad (15)$$

Finally, define *path overlap* $J(G)$ to be $1 - U(G)$. We demonstrate the performance of the graphs using the same example of $N = 1{,}024$ for Chord and $N = 1{,}000$, $k = 10$ for de Bruijn since the two graphs are almost identical in their size and node degree. In these graphs, path overlap $J(G)$ is 36% for Chord and only 3.7% for de Bruijn. In fact, for larger k, de Bruijn's overlap $J(G)$ monotonically decays to zero inverse proportionally to the product of k and μ_d (whereas in Chord, overlap $J(G)$ actually *increases* for higher k and N). This decrease in de Bruijn graphs is easy to explain – each set of k parallel paths between any pair of nodes (x,y) contains $k\mu_d$ nodes on average, out of which only one vertex is repeated in more than one path. The difference in $J(G)$ between Chord and de Bruijn is significant, but not

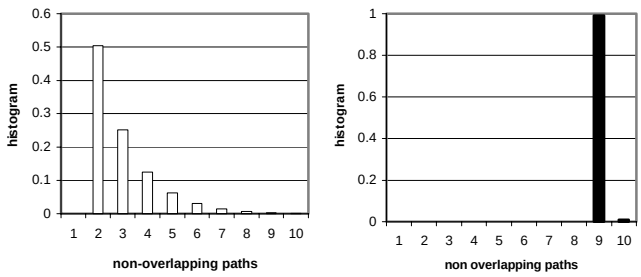

Figure 5: Distribution of the number of non-overlapping shortest paths in Chord for N = 1,024 (left). The same distribution in de Bruijn for N = 1,000, k = 10 (right).

quite unexpected given our previous discussion of expansion properties of each graph. This result means that de Bruijn (automatically) selects backup paths that do not overlap with the best shortest path *or* with each other.

Next examine Figure 5, which shows the number of pair-wise non-overlapping paths between any pair of nodes in the corresponding graph (we exclude pairs (x,x) and direct neighbors from the figure). Interestingly, 50% of pairs (x,y) in Chord have only *two* non-overlapping shortest paths $Q_i(x,y)$ and $Q_j(x,y)$. Hence, when nodes along these paths fail, many alternative paths are likely to be affected. Further observe in the figure that the number of node-disjoint paths in Chord is given by a right-shifted geometric distribution with $p = q = 1/2$ and (asymptotic) mean $1 + \frac{1}{p} = 3$. Thus, no matter how many neighbors k Chord has, its average number of node-disjoint paths Q_i is no more than 3. Thus, for $N = 32$, the mean of the distribution is 2.61 and for $N = 8{,}192$ it is 2.99. As N tends to infinity, the expected number of node-disjoint paths tends to 3.

The right side of the same figure shows that de Bruijn graphs have *at least* 9 non-overlapping paths between any pair of nodes. This means that when nodes fail and packets get re-routed along the optimal paths of each neighbor, they have very little likelihood of encountering the already-failed nodes. Qualitatively, this difference leads to better fault-resilience of de Bruijn graphs and smaller diameter *under node failure*.

In our next experiment, we introduced adversarial failures into the network. We failed all nodes along the shortest path from x to y and routed traffic through the second-best neighbor (i.e., the neighbor that is expected to have the shortest distance to y among the remaining neighbors). Then we failed all nodes along the second-best path and examined the third-best path, and so on. The distribution of average path lengths in the graph is shown in Figure 6. As demonstrated by the figure, the average distance in Chord rises to as high as 17.6 hops when routed through some of the "suboptimal" neighbors. De Bruijn graphs, on the other hand, maintain the same low diameter and the average distance rises only by one hop. Note that we plotted the average distances according to *path rank* (from the best to the worst), which does not necessarily represent the order in which Chord or de Bruijn would typically choose the next-best neighbor. However, as the figure shows, all *backup* neighbors of de Bruijn graphs are approximately equivalent and achieve the same suboptimal average distance. This

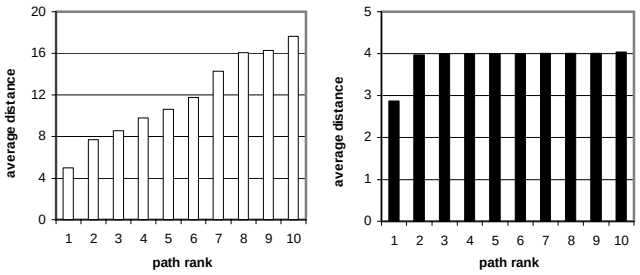

Figure 6: Distribution of shortest-path distances in Chord for N = 1,024 (left) and de Bruijn for N = 1,000, k = 10 (right) under adversarial failures.

cannot be said about Chord, which has certain neighbors that show tendency to construct shortest paths that always overlap with the already-failed ones.

7. ODRI

We have accumulated sufficient evidence that shows that de Bruijn graphs possess both short routing distances and high fault tolerance. In this section, we discuss ODRI, *Optimal Diameter Routing Infrastructure*, which builds de Bruijn graphs incrementally and preserves their nice properties at the application layer. Fortunately, de Bruijn graphs are very simple to build incrementally and many of the details (some of which we skip) are almost identical to those in Chord. We also feel that the algorithmic structure of ODRI is much simpler than that of other recently proposed fixed-degree graphs [25], [42].

Let N_{max} be the maximum possible number of nodes in the system (such as 10^{10}). Organize the space of all possible nodes between $[0, N_{max} - 1]$ into a modulo-N_{max} number field. Now notice that each node x in de Bruijn graphs is a base-k integer H_x and that its neighboring rules can be expressed as:

$$H_x \rightarrow (kH_x + i) \bmod N_{max}, i = 0, 1, ..., k - 1, \qquad (16)$$

since a shift left by one digit is equivalent to multiplication of H_x by k. In ODRI, each existing peer holds a consecutive stretch of the number space, which can be denoted by $[z_1, z_2]$, for some $z_1, z_2 \in [0, N_{max} - 1]$. To join the network, a node routes to the area of the circle where its hash index H_x is located and asks the previous owner of the zone to split it in half. Notice that building the routing table for a newly joined node requires only $O(1)$ message complexity as it can be copied from the previous owner of the zone. Notification of existing neighbors has another $O(k)$ message overhead.

Peer-to-peer linking rules are also straightforward. Consider node x that owns zone $[z_1, z_2]$. Each of the integer values in $[z_1, z_2]$ corresponds to the underlying de Bruijn graph of size N_{max}. Hence, to preserve de Bruijn linkage at the application layer, x must connect to all peers holding the other end of each edge originating in $[z_1, z_2]$. This means that there is an *application-layer* edge (x, y) if and only if there is an edge (u, v) in the underlying de Bruijn graph such that $u \in Z_x$ and $v \in Z_y$, where Z_x and Z_y are the corresponding zones held by x and y.

We next present several useful results about ODRI. We first address the issue of whether the application-layer graph maintains fixed degree and optimal diameter under the condition of equal-size zones. We then extend this analysis to random zones created by a uniform hashing function.

7.1 Equal-Size Zones

LEMMA 13. *If all zones have the same fixed size, ODRI maintains the application-layer degree equal to k.*

Given the assumptions of the previous lemma, notice that the application-layer graph in ODRI is a scaled-down version of the underlying de Bruijn graph. Thus, the diameter of the peer-to-peer graph under these conditions must remain optimal as we show in the next lemma.

LEMMA 14. *If all zones have the same fixed size, ODRI builds an N-node application-layer de Bruijn graph with diameter $\lceil \log_k N \rceil$.*

7.2 Random Zones

Achieving constant-size zones using distributed join and leave processes is a non-trivial, but well-studied problem [2], [27], [30]. Equal zone sizes are desirable as they maintain a fixed out-degree at the application layer and provide better balancing of user objects between the peers. Assuming uniform random hashing, it can be shown [27], [30] that after a sequence of N random joins, the maximum zone held by a peer is larger than average by a factor of $O(\ln N)$ with high probability (note that the same bound applies to the maximum *out-degree* of each peer). The following result about the application-layer *in-degree* is less obvious.

LEMMA 15. *Under a uniform hashing function, ODRI's in-degree at each peer is no less than k with high probability.*

Our next result shows that the imbalance in zone sizes has very little impact on the diameter of the peer-to-peer graph.

LEMMA 16. *Under a uniform hashing function, ODRI constructs a peer-level graph with diameter $\lceil \log_k N \rceil (1 + o(1))$ with high probability.*

This lemma further implies that the average distance in the application-layer graph is also asymptotically optimal.

7.3 Balancing Zones

To overcome imbalance in zone sizes in a highly dynamic environment, ODRI implements a variation of the "power of two choices" algorithm [2], [12], [27] during peer joins *and* departures. To join an existing ODRI network, a node x performs a biased walk through the graph starting in a random location and searching for the largest node to split. The walk is biased towards large nodes since they are more likely to "know" other large nodes. During departure, node x does the same biased walk looking for the smallest node to take over its zone Z_x. The details of this framework are currently under investigation and will be presented in future work.

8. CONCLUSION

At this stage of peer-to-peer research with an overwhelming number of recent proposals, it is hard to assess the benefits of one peer-to-peer network over another without a unifying analytical framework that can capture graph-theoretic properties of each proposal. In this paper, we studied the

diameter-degree tradeoff question of DHT research and conducted an extensive graph-theoretic comparison of several existing methods in terms of their routing performance and fault resilience. We then proposed a distributed architecture based on de Bruijn graphs and demonstrated that it offers an optimal diameter for a given fixed degree, optimal resilience (k-node connectivity), large bisection width, and good node expansion that guarantees very little overlap between parallel paths to any destination. Combining all these findings with incremental construction of ODRI, we conclude that de Bruijn graphs are viable and appealing structures for peer-to-peer networks.

9. REFERENCES

[1] J. Aspnes, Z. Diamadi, and G. Shah, "Fault-Tolerant Routing in Peer-to-Peer Systems," *ACM PODC*, July 2002.

[2] Y. Azar, A. Broder, A. Karlin, and E. Upfal, "Balanced Allocations," *SIAM J. on Computing*, vol. 29, no. 1, 1999.

[3] A.-L. Barabasi, R. Albert, and H. Jeong, "Scale-free Characteristics of Random Networks: The Topology of the World Wide Web," *Physica* A **281**, 2000.

[4] T. Bu and D. Towsley, "On Distinguishing between Internet Power Law Topology Generators," *IEEE INFOCOM*, 2002.

[5] C. Baransel, W. Doboseiwicz, and P. Gburzynski, "Routing in Multi-hop Packet Switching Networks: Gbps Challenge," *IEEE Network Magazine*, 1995.

[6] W.G. Bridges and S. Toueg, "On the Impossibility of Directed Moore Graphs," *Journal of Combinatorial Theory*, series B29, no. 3, 1980.

[7] F. Chung, "Diameters of Communication Networks," *Mathematics of Information Processing*, 1984.

[8] J. Considine and T.A. Florio, "Scalable Peer-to-Peer Indexing with Constant State," *Boston U. Technical Report 2002-026*, August 2002.

[9] D.-Z. Du, D.F. Hsu, H.Q. Ngo, and G.W. Peck, "On Connectivity of Consecutive-d Digraphs," *Discrete Mathematics*, vol. 257, no. 2-3, 2002.

[10] R. Feldmann, B. Monien, P. Mysliwietz, and S. Tschoke, "A Better Upper Bound on the Bisection Width of de Bruijn Networks," *Symposium on Theoretical Aspects of Computer Science* (STACS), 1997.

[11] A. Fiat and J. Saia, "Censorship Resistant Peer-to-Peer Content Addressable Networks," *Symposium on Discrete Algorithms*, 2002.

[12] P. Fraigniaud and P. Gauron, "An Overview of the Content-Addressable Network D2B," *ACM PODC*, 2003.

[13] M.J. Freedman and R. Vingralek, "Efficient Peer-To-Peer Lookup Based on a Distributed Trie," *IPTPS*, March 2002.

[14] P. Ganesan, Q. Sun, and H. Garcia-Molina, "YAPPERS: A Peer-to-Peer Lookup Service over Arbitrary Topology," *IEEE INFOCOM*, March 2003.

[15] K.P. Gummadi, R. Gummadi, S.D. Gribble, S. Ratnasamy, S. Shenker, and I. Stoica, "The Impact of DHT Routing Geometry on Resilience and Proximity," *ACM SIGCOMM*, August 2003.

[16] P. Gupta and P.R. Kumar, "The Capacity of Wireless Networks," *IEEE Trans. on Information Theory*, March 2000.

[17] K. Hildrum, J. Kubiatowicz, S. Rao, and B.Y. Zhao, "Distributed Object Location in a Dynamic Network," *ACM SPAA*, August 2002.

[18] M.G. Hluchyj and M.J. Karol, "Shufflenet: An Application of Generalized Perfect Shuffles to Multihop Lightwave Networks," *IEEE INFOCOM*, 1988.

[19] M. Imase and M. Itoh, "Design to Minimize Diameter on Building-Block Network," *IEEE Trans. on Computers*, vol. 30, 1981.

[20] F. Kaashoek and D.R. Karger, "Koorde: A Simple Degree-optimal Hash Table," *IPTPS*, February 2003.

[21] C. Law and K.-Y. Siu, "Distributed Construction of Random Expander Graphs," *IEEE INFOCOM*, 2003.

[22] F.T. Leighton, "Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes," *Academic Press / Morgan Kaufmann*, 1991.

[23] D. Liben-Nowell, H. Balakrishnan, and D. Karger, "Analysis of the Evolution of Peer-to-Peer Networks," *ACM PODC*, 2002.

[24] D. Loguinov, A. Kumar, V. Rai, and S. Ganesh, "Graph-Theoretic Analysis of Structured Peer-to-Peer Systems: Routing Distances and Fault Resilience," *Texas A&M Technical Report*, 2003.

[25] D. Malkhi, M. Naor, D. Ratajczak, "Viceroy: A Scalable and Dynamic Emulation of the Butterfly," *ACM PODC*, 2002.

[26] G.S. Manku, M. Bawa, and P. Raghavan, "Symphony: Distributed Hashing in a Small World," *USITS*, 2003.

[27] M. Naor and U. Wieder, "Novel Architectures for P2P Applications: the Continuous-Discrete Approach," *ACM SPAA*, June 2003.

[28] G. Pandurangan, P. Raghavan, and E. Upfal, "Building Low-Diameter P2P Networks," *IEEE Symposium on Foundations in Comp. Sci.*, 2001.

[29] C.G. Plaxton, R. Rajaraman, A.W. Richa, "Accessing Nearby Copies of Replicated Objects in a Distributed Environment," *ACM SPAA*, 1997.

[30] M. Raab and A. Steger, "Balls into Bins – A Simple and Tight Analysis," *RANDOM*, 1998.

[31] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A Scalable Content-Addressable Network," *ACM SIGCOMM*, August 2001.

[32] S. Ratnasamy, S. Shenker, I. Stoica, "Routing Algorithms for DHTs: Some Open Questions," *IPTPS*, 2002.

[33] S.M. Reddy, J.G. Kuhl, S.H. Hosseini, and H. Lee, "On Digraph with Minimum Diameter and Maximum Connectivity," *Proceedings of Allerton Conf. on Communications, Control and Computers*, 1982.

[34] J. Rolim, P. Tvrdik, J. Trdlicka, and I. Vrto, "Bisecting de Bruijn and Kautz Graphs," *Discrete Applied Math*, vol. 85, no. 1, June 1998.

[35] A. Rowstron and P. Druschel, "Pastry: Scalable, Decentralized Object Location and Routing for Large-Scale Peer-to-Peer Systems," *IFIP/ACM International Conference on Distributed Systems Platforms*, November 2001.

[36] J. Saia, A. Fiat, S. Gribble, A.R. Karlin, and S. Saroiu, "Dynamically Fault-Tolerant Content Addressable Networks," *IPTPS*, March 2002.

[37] M. Schlosser, M. Sintek, S. Decker, and W. Nejdl, "HyperCuP – Hypercubes, Ontologies and Efficient Search on P2P Networks," *Workshop on Agents and P2P Computing*, 2002.

[38] K.N. Sivarajan and R. Ramaswami, "Lightwave Networks Based on de Bruijn Graphs," *IEEE/ACM Trans. on Networking*, vol. 2, no. 1, 1994.

[39] M.A. Sridhar and C.S. Raghavendra, "Fault-tolerant Networks Based on the de Bruijn Graph," *IEEE Trans. on Computers*, vol. 40, 1991.

[40] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications," *ACM SIGCOMM*, August 2001.

[41] D.A. Tran, K.A. Hua, and T.T. Do, "ZIGZAG: An Efficient Peer-to-Peer Scheme for Media Streaming," *IEEE INFOCOM*, 2003.

[42] J. Xu, A. Kumar, and X. Yu, "On the Fundamental Tradeoffs between Routing Table Size and Network Diameter in Peer-to-Peer Networks," *To Appear in IEEE JSAC*, Nov. 2003.

[43] B.Y. Zhao, J.D. Kubiatowicz, and A. Joseph, "Tapestry: An Infrastructure for Fault-Tolerant Wide-Area Location and Routing," *UC Berkeley Technical Report*, April 2001.

[44] S.Q. Zhuang, B.Y. Zhao, and A.D. Joseph, "Bayeux: An Architecture for Scalable and Fault-Tolerant Wide-Area Data Dissemination," *ACM NOSSDAV*, June 2001.