Churn Resistant de Bruijn Networks for Wireless on Demand Systems

Manuel Thiele, Kendy Kutzner and Thomas Fuhrmann
System Architecture Group, Universität Karlsruhe (TH)
76128 Karlsruhe, Germany
eMail: {thiele|kutzner|fuhrmann}@ira.uka.de

***Abstract*— Wireless on demand systems typically need authentication, authorization and accounting (AAA) services. In a peer-to-peer (P2P) environment these AAA-services need to be provided in a fully decentralized manner. This excludes many cryptographic approaches since they need and rely on a central trusted instance. One way to accomplish AAA in a P2P manner are de Bruijn-networks, since there data can be routed over multiple non-overlapping paths, thereby hampering malicious nodes from manipulation that data.**

Originally, de Bruijn-networks required a rather fixed network structure which made them unsuitable for wireless networks. In this paper we generalize de Bruijn-networks to an arbitrary number of nodes while keeping all their desired properties. This is achieved by decoupling link degree and character set of the native de Bruijn graph. Furthermore we describe how this makes the resulting network resistant against node churn.

I. Motivation

Today, typical wireless systems are provider based in the sense that there is one central organization that handles authentication, authorization and accounting (AAA) for all their subscribers. This is obvious with systems like GSM or UMTS where the providers manage everything from the AAA backend to the base stations. But this is also true for home WiFi systems. The difference there is that the subscription relation is pushed from the mobile device to the access point (AP). The owners of the APs can control the access to their devices. Conversely, they are legally responsible for the users. Especially, they have to pay for the bandwidth use. As a consequence such systems typically deny roaming between such privately owned APs, thereby significantly reducing the use of such systems.

In this paper we present one building block for wireless network systems that push the subscription relation back into the end-device while maintaining the current pattern of private AP deployment. It aims at peer-to-peer systems consisting of APs and storing accounts for the users. Credits from such an account can be spent for gaining access while roaming.

Such systems harvest the idle resources available at its participants' APs and provide them on demand to their roaming users. Since in peer-to-peer systems there is no (need for a) central organization the participating nodes have to provide all the required resources and functionality to operate the system. As a consequence, there is no extra cost with doing that. I.e. such an approach enables easy and cheap scalability.

Such systems have been proposed in the literature [1][2]. The key challenge in such networks is the security of the AAA-system to avoid free riding [3] and to motivate participants to contribute resources [4]. This is to ensure that a user employs only resources that are equivalent to the resources he or she provides to the system. Note that 'equivalent' does not mean 'equal' (e. g. in terms of bandwidth). So the system can provide a net gain to its users. E.g., if a person that owns (and pays for) an AP with a flat rate provides access to their AP whenever it is idle; this incurs no extra cost for that person. Conversely, roaming users will highly value being granted access to that AP when they urgently need network access.

We extend that work by presenting a novel approach for one building block of these systems. In our system, messages between AAA-modules of participating peers are routed through a specialized overlay network. This overlay network is formed by the participating APs. Since such an overlay is a virtual structure only, the peers are able to create any network topology. But that topology is crucial for the efficiency, robustness and security of the system: Peer-to-peer systems are systems that lack central instances that could establish relations of trust. They can build trust, however, by combining independent sources of trust to increase the trust in a previously unknown peer. Therefore, the topology has to ensure that seemingly independent peers are in fact with high probability actually independent. One way to achieve this, is an overlay topology that routes messages redundantly over independent paths. Besides the gain in security, this increases the robustness of the system.

In this paper, we present a novel variant of the so-called de Bruijn networks. This class of overlay networks provides for several (depending on a parameter of the network) paths which are non-overlapping with a high probability between any two nodes. So far, such networks have been strongly affected by node churn, i. e. nodes joining and leaving the network, potentially ungracefully and at a high rate. Thus de Bruijn networks were thought unsuitable for ad-hoc and peer-to-peer networks where churn is prevalent due to frequent node joins and leaves[5]. Our proposal keeps all beneficial properties of de Bruijn networks while creating churn resistance with low overhead. Its key idea is to extend the de Bruijn graph construction algorithm from graphs with $n = K^L$ nodes to

graphs with an arbitrary number of nodes.

This paper is structured as follows: Section I-A gives an overview of de Bruijn networks. Section II generalizes their construction to an arbitrary number of nodes and describes how routing works in the resulting networks. Section III gives results from a study using a real-world implementation of the so-modified de Bruijn algorithm. Section IV concludes with an outlook to future work.

A. Introduction to de Bruijn Networks

In a de Bruijn-graph each node has a unique identifier consisting of letters of a given alphabet and having a certain length L. The size of the alphabet is called the de Bruijn-basis K and equals the out degree of each node. The length of the identifier is also known as the level of the node or graph.

Each possible combination of the letters of the alphabet with the given length is present in the graph. Because of this, there only exist de Bruijn-graphs with K^L nodes.

In such a graph every node is connected to all those nodes whose identifier can be created by deleting one character on one side and adding a character on the other side. Thus the network functions like a shift register.

On this graph greedy routing can be used to exploit the low diameter. This is done by determining the overlap of the lower end of the current node identifier with the high end of the destination address. Then routing is performed by shifting in the first character of the destination address, after the overlapping area.

Such a network has a constant in- and out-degree of K and a near optimal diameter of $\log_k(N)$ (The optimum [6] is at: $\lceil \log_k(N*(k-1)+1) \rceil - 1$). Additionally there are K different paths from any source node to any target node. An alternative path is selected by routing to the wrong next hop during the first routing step. These paths all have a comparable length and a high chance that they are non-overlapping, i. e. have no more nodes in common than the source and the destination. For example a de Bruijn network with 1000 nodes and K=10 has a path overlap of 3.7% ([6]). The probability of paths overlapping further decreases proportionally to the product of K and the diameter of the graph. This path redundancy is one of the main reasons to use de Bruijn graphs for security relevant systems: An attacker has to subvert all paths to alter a message.

B. Related Work

The idea of de Bruijn networks is older than most recent peer-to-peer networks. It has been used for interconnection of processors in parallel computing, for example [7]. With regard to the usability in peer-to-peer networks, [6] compared many different network graphs. The de Bruijn graph was one of the most favorable, because of its low diameter and constant degree. The resulting network design was called ODRI (for optimal diameter routing infrastructure). At at the same time de Bruijn graphs have been proposed for peer-to-peer networks [8], [5], [9]. Koorde [9] is an embedding of the de Bruijn graph into a Chord [10] network, it therefore inherits many properties from Chord. To gain resistance against node failures, either a number of backup links are introduced or the alphabet size K is increased up to $O(\log n)$. Since the number of possible de Bruijn nodes is larger than the number of nodes present in the network, each Koorde node is responsible for a large number of de Bruijn identifiers. In [8] the de Bruijn topology is used to establish a distributed hash table which strives for probabilistic guarantees for lookups. Since the lookup guarantees are loosened, the routing table maintenance overhead is reduced too. Furthermore, lookup hot spots can be alleviated by means of caching, and key collisions (which may occur e. g. in file sharing networks) can be tolerated at the cost of more lookup steps.

II. Generalization of de Bruijn Networks

As described in section I-A a de Bruijn-graph is only defined for K^L nodes. This would be a tough restriction, because a peer-to-peer-network normally has an arbitrary and permanently changing size. We therefore alter the building rules of the graph slightly, so that it can hold any number of nodes.

The restriction of the graph that all nodes have the same level L (length of the node identifier) is loosened. Without further restrictions, such generalized de Bruijn-networks tend to degenerate under churn, i.e. to accumulate level differences of neighboring nodes. By this, de Bruijn-networks loose their good properties (constant degree, low diameter, high probability of non-overlapping paths). This paper describes a means to circumvent this degeneration.

To prevent this corruption, the maximum level difference between neighboring nodes is set to one. This is a greater relaxation than is theoretically required, because only a level difference of one level in the whole graph is necessary. But the latter rule can only be enforced with global knowledge, which is hard to obtain in the envisioned distributed system. It will be described in the following how it is possible to keep the property of a limited level difference between neighboring nodes in a environment with churn.

In the following sections we will describe the rules for building up, maintaining and routing in a graph with level differences in time and space. The goal of these rules is first of all to avoid unnecessary rebuilds of the graph as changes happen in time. Second, the properties of the graph should be altered as little as possible when nodes of different levels neighbor each other.

A. Linkage of Neighboring Nodes

To achieve these goals, the ID of a node is extended to the right if the level increases. The node is split into K new nodes, one for every possible extension (see figure 2 compare figure 1).

This leads to the rule that the character with the highest significance is always deleted from one node to the next. Additionally on the right as many characters as needed are added. This can be zero, if the level decreases, one, if the level stays the same or two, if the level increases. Because

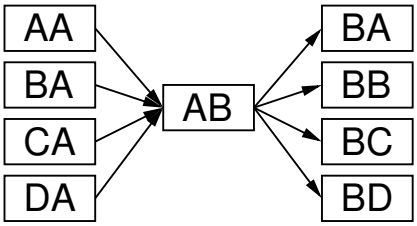

Fig. 1. Part of a Level 2 de Bruijn-Graph with $K = 4$

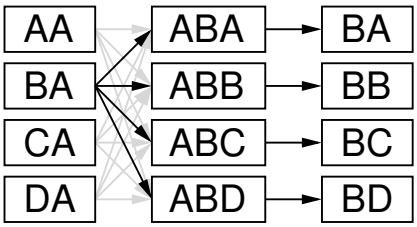

Fig. 2. The ID is extended to the right

of this, the changes of the graph are only local and as many connections as possible are reused.

With these rules the in-degree of a node always stays the same, but the out-degree changes between one and K^2 (reduction of the changes see II-A.1). If the out-degree is of importance (for example for security reasons, there should be always a maximum number of paths to every possible destination) it can be maintained by holding the connections to the former neighbors as irregular neighbors. These are only needed for routing as the first step, when one needs multi-path-routing.

1) Decouple Linking Degree and Character Set: In this scenario K strongly influences the behavior of the network. It sets the connection degree (G) and the number of possible characters (Z).

$$Z = K = G$$

The number of possible characters again defines the number of new nodes, which are formed out of one, when the level of a node changes. The number of new nodes once more causes the decrease or increase of out-degree as shown in the formula.

$$\begin{aligned} &\text{all neighbors have lower level: out-degree} = G/Z = 1 \\ &\text{all neighbors have higher level: out-degree} = G \cdot Z = G^2 \end{aligned}$$

Because Z and G are the same, the out-degree vacillates so strongly. But this strong correlation is not necessary. A much weaker correlation is also possible:

$$G = Z^M$$

being M a arbitrary positive integer.

This replaces only one old character through M new characters, the rest stays the same.

But there is no need to add M characters, if the level of a node is increased. Instead, just one character may be added.

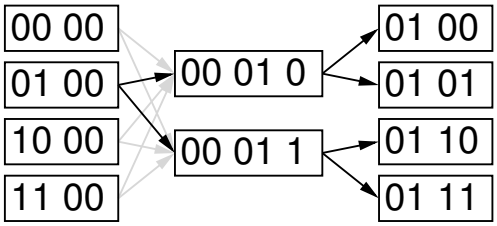

Fig. 3. Decoupling of alphabet and node level.

By this the vacillation is decreased from square to linear. To achieve the best possible gain, Z should be the smallest possible number. The optimum is when $Z = 2$.

Figure 3 shows the idea. The letters of the alphabet are replaced as follows: A=00, B=01, C=10 and D=11.

B. Creating Churn Resistance

The border for level differences of one is hard to obtain in such a network and requires a lot of maintenance effort. This again would cause a low churn resistance.

Secondly for the safety of the data there has to be a certain redundancy of the accounting data through out the network. For these nodes keeping redundant data communication is hard, because they have to use the network, which needs on average $\log_K(n)$ steps.

To overcome these problems, all of the instances which hold the same data for safety reasons only (not those kept apart for security reasons) are brought together in one node (the number of them shall be C). From now on we distinguish between nodes (vertices of the de Bruijn graph) and members of nodes or instances (APs running the implementation). A node can have many members. All these members of one node have the same look of the de Bruijn network and are totally interchangeable. They are connected through a second network for synchronization purposes. This network can vary greatly, it can be a (skip) ring, a (hyper) cube or fully connected. Let the diameter of this network be Innerdim(C,I) depending on the maximum node size and the numbers of connections used.

Such a node has a minimal and a maximal number of members. If the maximum is surpassed the node is split into Z new nodes with a level that is one higher than the old one. If the minimum is reached the split is undone and the Z nodes are joined again into one node.

The number of steps needed for communication between the members of a node is constant because of this. Additionally the numbers of changes in the de Bruijn-network is greatly (depending on the number of node members) reduced. Depending on the threshold values for node splits and joins, the expectation value for a node disappearing at all can be negligible.

By this, the maintenance of the network is much simplified, requires few resources. These are basic requirements for churn resistance.

Because the node size has a (relatively low and constant) upper limit, the effort for the inner node network is almost neg-

ligible (constant). This can justify full connectivity between the members.

As mentioned above, an instance now holds some connections to other de Bruijn nodes and some connections to instances of the same node. Let the upper limit of intra-node connections be I. Therefore an instance still has a constant (maximum) out-degree, but there is a total of $K+I$ connections required. So I and K compete with each other. The total diameter of the network is $\log_K(\frac{n}{C})+\text{Innerdim}(C,I)$ This tern has to be optimized by balancing K and I and maybe C with regard to the churn resistance and the relation of the traffic amount for synchronization purposes and the amount of node to node communication. For example, the target value for I (and thereby node members assuming full connectivity among them) can be set to $O(\log n)$ to create probabilistic survival even if $\frac{1}{2}$ of all instances fail simultaneously.

C. Optimizing Network Properties

In such a network the degree of overlap of paths depends on the level of the node with the lowest level. The diameter again can be estimated by the node with the highest level.

To optimize these and other properties, the level differences should be as small as possible. For this task the fact can be exploited that a node consists of several members. This can be done by not only balancing the node levels but also the node sizes among nodes in the network.

One can use two means:

1) by carefully placing new members
2) by letting members rejoin when differences above a threshold are detected.

In the event of a network with a growing or stable size, the placement of new members through for example random multi point sampling or random walks (see [11]) will be sufficient for balancing the network. Otherwise the number of leaving computers is greater than the one of new computers. Because of this there may be additional balancing needed. For this the neighbors of a node are monitored. If the size of a neighbor (with accounting for the level) is more than one smaller than the own, local members re-execute the placement procedure for new members with special attention to this neighbor. This point of departure is superior to just sending a member to the neighbor because a more global balancing is done and security mechanisms built in this placement procedure can not be circumvented.

The maximum level difference between the node with the highest and the lowest level is

$$\frac{\text{network diameter}}{\text{minimum size of a node}}$$

and kept below this fraction by the eventual node change of individual computers.

III. Implementing and Evaluation

This network design can be easily transformed into a layered architecture shown in figure 4. The lowest layer combines computers or instances to de Bruijn nodes (NMP, Node Management Protocol). If a computer leaves gracelessly, it is detected on this layer.

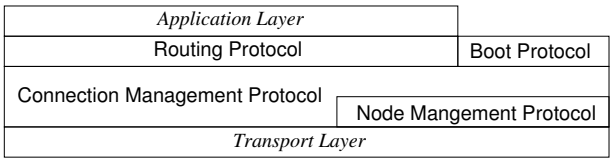

Fig. 4. Protocol stack architecture

TABLE I
Performance of the accounting system (averaged)

maximum number of instances	2186
minimum number of instances	114
lost packets	$4.7\cdot10^{-4}$
missing synchronizations	$2.6\cdot10^{-4}$
half opened accounts	$3.2\cdot10^{-3}$

A second layer assembles nodes to the de Bruijn-network, by handling the neighbors and their knowledge of each other. Additionally, it handles the balancing of the network by detecting differences between neighbors. For all this to work the CMP (Connection Management Protocol) uses the NMP. The routing of messages from an arbitrary node to any other is done by a third layer (RP, Routing Protocol), by using the connections created by the CMP.

The application itself, i. e. the accounting system, also uses the routing protocol to transfer messages between instances of the accounting protocol. It utilizes the offered multi-path-routing for security related data exchange.

At last, the choosing of the integration place for a new computer is done by a fourth protocol (BP, Boot Protocol). This needs to use the CMP, but may also use any of the other protocols.

Such a network, including the accounting system, was implemented in POSIX-C using TCP/IP for the connections[4]. The full source code is available from the authors upon request. Networks with up to two thousend instances of this implementation were tested with respect to functionality and stability under churn. The number of nodes per second, which where disconnected from their neighbors because all of them left before the network could react appropriately, was chosen as metric for the stability of the network. In such extreme cases, the instances execute the boot procedure to re-join the network. The examination of the results confirmed that aggregation of instances to nodes greatly increased the churn resistance of the overall network. Also as expected, the longer members stay in the network, the more stable the network became. The tests showed that with a per instance failure rate of $0.8\cdot10^{-3}$ per second a minimum node size of 5 leads to a stable network. For failure rates of $3.2\cdot10^{-3}$ per second the node size had to be increased to 8 in a network of 300 instances.

Next we estimated the maximal join and leave speed

our implementation could tolerate without disturbance at the de Bruijn level. We found that with a join rate below two instances per node per second the invariant of maximal level difference between neighboring nodes of one was never violated. Since a rebalance is necessary after instance departures where two nodes are involved, the maximum handled departures rate was one instance per node per second. When these rates are increased, short periods of inconsistencies can occur.

On top of the established de Bruijn routing protocol we implemented an accounting as system designed in [4]. Table I shows the performance tests of the system. Even under the imposed churn (the network shrank by a factor of 20), the system worked as designed. This is due to the stability and churn resistance of the underlaying de Bruijn network. When instances get killed during the reception of a message, this message is lost. In the implemented accounting protocol this leads to missing synchronizations between different account holders and half opened accounts. Both problems can be solved very easily with the repetition of the lost message. In the shown tests these messages were deliberatly not re-sent to show that even without it the system is very stable.

IV. Conclusion and Outlook

In this paper we showed that it is possible to extend the de Bruijn graph from K^L to an arbitrary number of nodes. A network based on such a graph can be an important building block for AAA-systems in the wireless domain, because the resulting network has many desirable properties for security relevant systems. First, the network scales well and has a small diameter, resulting in few hops to, for example, account holders. Second, the structure of the network allows multiple independent paths between any two nodes, which reduces the necessary trust in intermediate nodes. The implementation of the proposal is described together with mechanisms to make the system resistant against node churn.

Further work should explore ways to make the system more resilient against non-arbitrary node failures. Also the ability of the network to recover from net splits or re-joins can be improved.

References

[1] E. C. Efstathiou and G. C. Polyzos, "A peer-to-peer approach to wireless lan roaming", in *WMASH '03: Proceedings of the 1st ACM international workshop on Wireless mobile applications and services on WLAN hotspots*, New York, NY, USA, 2003, pp. 10–18, ACM Press.

[2] D. Barkai, "Technologies for sharing and collaborating on the net.", in *Peer-to-Peer Computing*. 2001, pp. 13–28, IEEE Computer Society.

[3] D. Hausheer and B. Stiller, "Peermint: Decentralized and secure accounting for peer-to-peer applications.", in *NETWORKING*, R. Boutaba, K. C. Almeroth, R. Puigjaner, S. X. Shen, and J. P. Black, Eds. 2005, vol. 3462 of *Lecture Notes in Computer Science*, pp. 40–52, Springer.

[4] M. Thiele, "Entwicklung eines sicheren Protokolls für ein Peer-to-Peer-Hotspot–Accounting-System", Feb. 15 2005.

[5] A. Datta, S. Girdzijauskas, and K. Aberer, "On de bruijn routing in distributed hash tables: There and back again.", in *Peer-to-Peer Computing*, 2004, pp. 159–166.

[6] D. Loguinov, A. Kumar, V. Rai, and S. Ganesh, "Graph-theoretic analysis of structured peer-to-peer systems: routing distances and fault resilience", in *Proceedings of the SIGCOMM2003 conference*. 2003, pp. 395–406, ACM Press.

[7] M. L. Schlumberger, *De bruijn communications networks.*, PhD thesis, 1974.

[8] A.-T. Gai and L. Viennot, "Broose: A practical distributed hashtable based on the de-bruijn topology", in *Peer-to-Peer Computing*. 2004, pp. 167–164, IEEE Computer Society.

[9] M. F. Kaashoek and D. R. Karger, "Koorde: A simple degree-optimal distributed hash table", in *Proceedings of the 2nd International Workshop on Peer-to-Peer System (IPTPS '03)*, Berkeley, CA, USA, Feb. 2003.

[10] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications", in *Proceedings of the SIGCOMM 2001 conference*. 2001, pp. 149–160, ACM Press.

[11] X. Wang, Y. Zhang, X. Li, and D. Loguinov, "On zone-balancing of peer-to-peer networks: analysis of random node join", in *SIGMETRICS 2004/PERFORMANCE 2004: Proceedings of the joint international conference on Measurement and modeling of computer systems*, New York, NY, USA, 2004, pp. 211–222, ACM Press.