Random Graphs for Peer-to-Peer Overlays

Peter Mahlmann[1] and Christian Schindelhauer[2]

[1] Heinz Nixdorf Institute and Computer Science Department, University of Paderborn, Germany,
mahlmann@upb.de
[2] Computer Networks and Telematics Institute of Computer Science, University of Freiburg, Germany,
schindel@informatik.uni-freiburg.de

Abstract. Modern peer-to-peer networks consist of several network layers and distributed algorithms providing features like indexing, ressource balancing, entry protocols, security, anonymity, and cryptography. Since peer-to-peer networks are highly dynamic, a fundamental task in the design of these networks is to provide high connectivity. We propose a solution by distributed random link exchange algorithms such that the overlay network can be connected random graphs. Such random graphs are expander graphs with high probability which are robust against node and edge failures. Furthermore these random networks have low diameter and small degree. In this paper we survey recent work.

1 Introduction

Peer-to-Peer networks have become an established design methodology. Until 2000 applications in the Internet were client-server based. Therefore, the reliability of these applications depended on the availability and performance of the servers. If the nodes of a network share the duties of servers and establish a network without hierarchies then we call these nodes peers and the network is called a peer-to-peer network. Peer-to-peer networks belong to the class of so called overlay networks, since they reside in the application layer of the Internet protocol stack. This means, an overlay network is a network which is build on top of a physical network, i.e. the Internet, and uses the physical network for realizing the communication among nodes that are connected in the overlay network.

After nearly a decade starting with the first peer-to-peer-like network Napster in 1999 and the first true peer-to-peer network Gnutella [10] in 2000, peer-to-peer traffic has been grown to an all-time high with more than 60% of the overall Internet traffic in 2005. Only recently the web-based client-server traffic has reclaimed the majority of Internet traffic.

In this short period of time numerous theoretic and practical approaches have been made. Modern peer-to-peer networks consist of several network layers and distributed algorithms providing various features like indexing, ressource balancing, entry protocols, security, anonymity, and cryptography. Such networks have left the grey area of file-sharing and have conquered application areas like group-ware, voice over IP, online chatting and video conferences, e.g. see the Skype network.

Peer-to-peer networks are not merely robust, they have been proven to be practically indestructible: As long as the peers remain connected, the network services can be provided. This leads to a crucial and fundamental problem of connectivity in peer-to-peer networks. If a group of peers loses contact to the rest of the network because the connecting peers leave the network, then the network is partitioned. Without outside information like from the users or from a central server it is impossible rejoin a once broken network. Clearly, increasing the interconnectivity, i.e. the number of

peers a node knows, alleviates the danger. Yet, it comes with the problem to continuously check a large number of neighbors and it is not clear whether the implied graph structure is robust under churn.

In this paper we propose two features for establishing a robust low-degree connectivity layer for peer-to-peer networks: distributed local graph operations and random connected expander graphs.

As an example consider one of the first peer-to-peer networks, the Gnutella network [10]. It relies on a random (uncontrolled) network structure. Here, new peers join the network by connecting to former neighbors. If this fails, e.g. if the Gnutella network is used for the first time, the peer tries out a list of peers which often happen to participate in the network. Specific hosts may cease to exist. If all former members do so, then a peer is not able to connect to the network. Since the list of former members is extended after each session this risk becomes smaller and smaller.

Studies reveal that the network structure of Gnutella is not truly random. It forms a so-called Pareto (or power law) distributed graph [13, 25]. Compared to a random graph, the degree distribution (i.e. the density function of neighbors) is skewed. Also the diameter of the network is larger than in random networks with the same average out-degree. This implies, that the connectivity and network performance of Gnutella can be improved if the peers are connected by a truly random network [18] (Yet, this does not solve the problem of efficient search [26]).

The peer-to-peer network design suite JXTA of Sun Microsystems incorporates random networks. JXTA connects all peers by a random network to provide robustness, i.e. preventing peers or groups of peers from being disconnected. JXTA is a design tool for peer-to-peer network applications like distributed search for web-sites [5]. However, it has never been proved, whether the algorithms used in JXTA actually produce random networks with small diameter or robust connectivity.

Random Networks provide resilient service against failures [15, 16]. These findings are backed by results in graph theory showing that random graphs provide connectivity and expansion even for very small degrees [31]. The expansion property holds if for any partition of nodes the number of edges on the border is proportional to the smaller node set size. For networks such expander graphs are desirable, because they imply logarithmic diameter, large conductance, and short mixing times of random walks [9]. Pandurangan, Rhagavan and Upfal [24] presented a first approach to build low-diameter networks with expansion property. Their approach ensures that the network is connected and has logarithmic diameter with high probability. It is based on a central cache, which holds addresses of a random subset of the nodes and is accessed when nodes join the network.

Random Graphs For the problem of generating random regular graphs with nearly uniform probability distribution Jerrum and Sinclair [12] give a generator for d-regular graphs on n vertices which approximates the uniform probability distribution by a factor of $1+\varepsilon$ and runs polynomial in n and $\log 1/\varepsilon$. This result was improved by Steger and Wormald [28]. It is known that all d-regular graphs are connected and it is easy to use these algorithms to approximate also d-regular connected graphs. However, this methods cannot be applied in a networking concept.

Law and Siu [17] build d-regular random multigraphs in which the set of edges is composed of $d/2$ Hamilton cycles of size n. The probability space produced by their protocol deviates from the uniformly distributed space and may lead to graphs with small or no expansion. Their operation cannot recover from bad graphs.

Expander graphs have a number of advantageous properties, e.g. logarithmic diameter, high vertex connectivity and a small mixing time of random walks.

1. For $S,T \subset V$ denote the set of all edges between S and T by $E(S,T) = \{\{u,v\} | u \in S, v \in T, \{u,v\} \in E\}$.
2. The *edge boundary* of a set $S \subset V$, denoted by ∂S, is $\partial S = E(S,\bar{S})$ with $\bar{S} = V \setminus S$.
3. A graph $G = (V,E)$ provides *expansion* $\beta > 0$, or is a β-*expander*, if for all node sets S with $|S| \leq |V|/2$ $\quad |\partial S| \geq \beta |S|$.

Theorem 1.

1. *For fixed $d \geq 3$ any random d-regular graph is connected asymptotically almost surely, i.e. with probability $1 - o(1)$. [3, 30]*
2. *For $d \in \omega(1)$ a random d-regular graph is a $\Theta(d)$-expander graph. [4]*

Local graph transformations are better suited to establish and uphold random graphs in peer-to-peer networks than centralized operations. Without continuous graph maintenance the graph properties at the start determine the quality of the future networks. Depending on the node insertion procedure long living peers could have more or less neighbors than the average peer which would implant weak spots into the network.

The idea of local graph transformation is to establish a continuous self-healing process that automatically repairs the network without any complicated data structures or central servers. According to [19], a graph transformation for the maintenance of random graphs should fulfill the following requirements to be suitable for the adaption in peer-to-peer networks:

Soundness: No transformation maps to graphs which are not in the domain space, e.g. the connectivity of the graph has to be preserved.

Generality: The random transformation process does not converge to a specific graph. All graphs in a large domain are *reachable* and in the limit all graphs in this domain occur with non-zero probability. This requirement can be tightened to *uniform generality* where in the limit all graphs occur with the same probability.

Feasibility: The graph transformation can be described by a simple (distributed) algorithm. Its implementation in a distributed network should be straightforward.

Convergence rate: Only a small number of transformations is necessary to achieve an approximation of the ultimate distribution of all graphs.

Started from any extreme network a series of local operations should repair the graph until it becomes a truly random graph. Identifying such transformations meeting all these requirements gives an approximate solution to the problem of computing a random probability distribution over all graphs of a kind, as being solved by Steger et al. [28] for the problem of generating all d-regular random graphs.

In [19] and [20] such local graph operations have been presented and investigated. We will revisit these operations, called *Flipper* and *Pointer-Push&Pull*, present recent findings in this areas, and compare them to alternative local random graph operations for overlay networks.

1.1 Notations

A d-regular undirected graph $G = (V,E)$ is defined by a finite node set $V = \{1,2,3,\ldots,n\}$ of size n and the edge set $E := \{\{u,v\} \ : \ u,v \in V, u \neq v\}$ (of size $dn/2$) such that each node is adjacent to exactly d edges. An undirected graph is connected if for each pair of nodes of V there exists a path using edges of E connecting these nodes. For $v \in V$ we denote the set of nodes neighboring v by $N(v)$.

A *simple digraph* $G = (V,E)$ is defined by a node set $V = \{1,\ldots,n\}$ and a set of directed edges $E = \{(u,v) : u,v \in V, u \neq v\}$. A digraph is *strongly connected* if for all pairs of nodes there exists a directed path in E and *weakly connected* if there exists a path neglecting the direction of the edges for each pair of nodes. For $v \in V$ we define $N^+(v) = \{w : (v,w) \in E\}$ and $N^-(v) = \{u : (u,v) \in E\}$ to refer to the successor and predecessor nodes of v in G. A digraph is called d-regular if $\forall v \in V : |N^-(v)| = |N^+(v)| = d$. Furthermore, we say a digraph is d-out-regular if $\forall v \in V : |N^+(v)| = d$.

The usage of a multiset for the set of edges in digraphs will allow us to define a more general model of digrahs. Therefore, we give a formal definition of multisets.

A set E and a function $\#_E : E \to \mathbb{N}_0$, specifying the multiplicity of its elements, define a *multiset*. For $e \in E$ we write $e \in_k E$ if $\#_E(e) = k$. This implies $e \in_0 E$ for all $e \notin E$. The cardinality of a multiset E is defined as $|E| = \sum_{e \in E} \#_E(e)$.

A *multi-digraph* $G = (V,E,\#_E)$ is defined by a node set $V = \{1,\ldots,n\}$ and a multiset of directed edges $E = \{(u,v) : \ u,v \in V\}$ with $\#_E$ specifying the multiplicity of the edges. In a multi-digraph self-loops (u,u) and multiple occurrence of edges are explicitly allowed, e.g. an edge (u,v) may occur twice.

Analogous to simple digraphs, a multi-digraph is called *d-regular* if $\forall u \in V : \sum_{v \in V,(u,v) \in E} \#_E((u,v)) = \sum_{v \in V,(v,u) \in E} \#_E((v,u)) = d$ and called *d-out-regular* if $\forall u \in V : \sum_{v \in V,(u,v) \in E} \#_E((u,v)) = d$. Note, that if no slopes occur and the multiplicity of all edges is at most one then a multi-digraph describes a simple digraph. So, simple digraphs form a subset of multi-digraphs. As in case of simple digraphs we define the neighborhood of a node $v \in V$ as $N^+(v) = \{w : (v,w) \in E\}$. Note that $N^+(v)$ is not a multi-set and therefore the $|N^+(v)| < d$ is possible in a d-out-regular multi-digraph.

By $\log n := \log_2 n$ we mean the dual logarithm function. In the following the term "with high probability" (w.h.p.) describes the probability $p \geq 1 - n^{-c}$ and "asymptotically almost surely" (a.a.s) is the probability $p \geq 1 - o(1)$.

2 Distributed Local Graph Transformations

Now we will present some local graph operations suitable for peer-to-peer networks.

2.1 Simple Switching

As a random transformation for d-regular undirected graphs without connectivity we consider the following transformation introduced by McKay [21] and used by [9] page 215 (there called "rewiring").

Definition 1 (Simple Switching). *Choose random edges* $\{v_1,v_2\}$, $\{v_3,v_4\}$ *of the graph* G. *If* $\{v_1,v_3\}$ *and* $\{v_2,v_4\}$ *do not exist in* G *then erase* $\{v_1,v_2\}$, $\{v_3,v_4\}$ *and insert edges* $\{v_1,v_3\}$ *and* $\{v_2,v_4\}$.

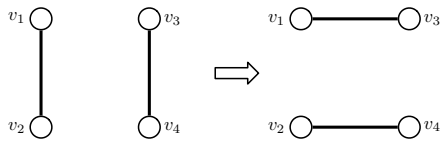

Fig. 1. The Simple-Switching operation.

In [22] Simple Switching is used to generate random d-regular graphs with $d \in O(n^{1/3})$ and it takes an expected time of $O(nd^3)$ per graph to generate a uniform distribution over all d-regular graphs. Simple Switching preserves the degree of each node but does not preserve connectivity. Its convergence speed is polynomial in the number of nodes which follows from the results of [6, 22]. Cooper, Dyer, and Greenhill [7] give an upper bound of $d^{17}n^7\log(dn\varepsilon^{-1})$ for the convergence rate of the switch operation. However, they conjecture that $O(n\log n)$ operations suffice for constant degree.

The Simple Switching operation is feasible if the complete graph is available. However, in a peer-to-peer network this procedure is not feasible, unless there is an operation which can choose two random edges. This can be done by performing a random walk with an appropriate length, i.e. the mixing time of the graph. This technique works only in connected graphs. Since Simple-Switching has the potential of partitioning the graph, without extra network connections the network cannot be rejoined anymore. Then, random walks will stay in their partitions of the graph and will never unify the graph again. On the bottom line Simple Switching is only feasible for connected graphs, a domain which will be left on the run.

2.2 Flipper

For undirected graphs the Flipper (a.k.a. 1-Flipper) [19] establishes a fully decentralized random graph transformation. Like the Simple-Switching operation it switches two edges with the restriction that the two flip edges must be connected by a hub edge.

Definition 2 (1-Flipper). *Consider a d-regular undirected graph $G = (V,E)$ and four distinct nodes $u_1,u_2,u_3,u_4 \in V$ forming a path $P = (u_1,u_2,u_3,u_4)$ in G. Then, if $\{u_1,u_3\}$, $\{u_2,u_4\} \notin E$ the* 1-Flipper *operation F_P^1 transforms graph G to a graph $F_P^1(G) = (V,E')$ with $E' := (E \setminus \{\{u_1,u_2\},\{u_3,u_4\}\}) \cup \{\{u_1,u_3\},\{u_2,u_4\}\}$.*

Figure 2 shows this operation. The edges $\{u_1,u_2\},\{u_3,u_4\} \in E$ are the *flipping edges* and $\{u_2,u_3\} \in E$ is the *hub edge* of the 1-Flipper operation.

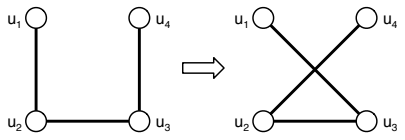

Fig. 2. The 1-Flipper operation F_P^1.

Note that in some graphs for certain hub edges there is no possibility to perform an edge flip at all. In this case all neighbors are connected to both nodes of the hub edge. Then, the random Flipper operation has no effect unless another hub edge is chosen.

Algorithm 1 Random 1-Flipper

Choose random edge $\{u_2,u_3\} \in E$
Choose random node $u_1 \in N(u_2)\backslash\{u_3\}$
Choose random node $u_4 \in N(u_3)\backslash\{u_2\}$
if $\{u_1,u_3\},\{u_2,u_4\} \notin E$ **then**
  $E \leftarrow E \setminus \{\{u_1,u_2\},\{u_3,u_4\}\}$
  $E \leftarrow E \cup \{\{u_1,u_3\},\{u_2,u_4\}\}$

Let $G \xrightarrow{F^1} G'$ denote the predicate that graph G is transformed to G' by a 1-Flipper operation. The following lemma is crucial for understanding the behavior of the Flipper operation.

Lemma 1. *[19]*

1. *For all $d > 2$ there is a connected d-regular graph G such that* $\mathrm{P}[G \xrightarrow{F^1} G] \neq 0$.
2. *For all undirected regular graphs G, G':* $\mathrm{P}[G \xrightarrow{F^1} G'] = \mathrm{P}[G' \xrightarrow{F^1} G]$.

So, every reachable graph will occur with the same probability in the limit. The following lemma shows that all connected regular graphs are reachable.

Lemma 2. *[19] For all pairs G, G' of connected d-regular undirected graphs, with $d \geq 2$ and even, there exists a sequence of Flipper operations transforming G into G'.*

Let $G \xrightarrow{i} G'$ denote the predicate that G' is derived from G by applying i Random 1-Flipper operations. Furthermore let $\mathscr{C}_{n,d}$ denote the set of all connected d-regular graphs with n nodes. The above lemmas imply that the Random 1-Flipper operation provides uniform generality.

Theorem 2. *[19] Let G_0 be a d-regular connected graph with n nodes and $d > 2$. Then in the limit the Random 1-Flipper operation constructs all connected d-regular labeled graphs with the same probability, i.e.*

$$\lim_{t\to\infty} P[G_0 \xrightarrow{t} G] = \frac{1}{|\mathscr{C}_{n,d}|}.$$

In [19] it has been shown that this random operation is sound and feasible, i.e. the Flipper operation preserves connectivity and regularity while only four closely neighbored nodes are involved. For performing a Flipper operation four neighbored nodes are involved. We have seen that it is general for the domain of undirected connected regular graphs. Furthermore, the Flipper, started with any regular connected undirected graph, in the limit constructs all such graphs with the same probability which implies the expander property of the graph by Theorem 1.

Corollary 1. *For $d > 2$ consider any d-regular connected graph G_0 with n nodes. Then in the limit the Random 1-Flipper operation establishes an expander graph after a sufficiently large number of applications a.a.s..*

There is little been known about the convergence rate of this process. There is a recent paper [8] showing that the convergence rate can be bounded by a high degree polynomial, i.e. $O(d^{56}n^{53})$. However, our findings presented in Section 4 point towards an convergence time of $O(dn\log n)$. So, Flipper constitutes a simple local random operation providing robustness and randomness.

2.3 Pointer-Push&Pull

Surprisingly, there is an even simpler operation than Flipper which provides comparable features. First, Flipper involves the active participation of four peers. Second, the maintenance of undirected, or to be more precise bi-directed, graphs is costly and not necessary. In practice digraphs are completely sufficient. Peers do not need to know who is pointing towards them as long as the network is connected and robust. Note, that there are several peer-to-peer networks which use digraphs, see [29, 14, 23, 11] for example. The following operation involves only three nodes, and only two of them participate actively. Its main advantage is simplicity: For a local update operation only two peers need to exchange two messages. Hence, Pointer-Push& is feasible.

Definition 3 (Pointer-Push&Pull Operation).
Let $G = (V, E, \#_E)$ be a d-out-regular multi-digraph and let nodes $v_1, v_2, v_3 \in V$ form a directed path $P = (v_1, v_2, v_3)$ in G. Then, the Pointer-Push&Pull operation PP_P transforms graph G to graph $PP_P(G) = (V, E', \#_{E'})$ with

$$E' = (E \setminus \{(v_1, v_2), (v_2, v_3)\}) \cup \{(v_1, v_3), (v_2, v_1)\} .$$

The Pointer-Push&Pull operation is illustrated in Figure 3. We will later discuss the components of the Pointer-Push&Pull operation: The Pointer-Push and Pointer-Pull operations, performing worse.

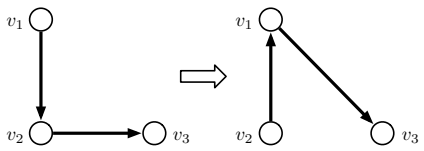

Fig. 3. The Push&Pull operation PP_P.

In [20] it has been noted that this operation is sound. It upholds weak connectivity, when the two neighbored peers do not participate in one Pointer-Push&Pull operation at a time.

There are two randomized versions of this operation providing different forms of generality. Both versions produce all weakly connected d-outregular multi-graphs with same probability. Yet, there are slight differences in the probability distributions, as we will point out shortly. This random Pointer-Push&Pull operation chooses its starting node uniformly at random, then performs a random walk of length two with some probability. Recall that due to multi-edges $|N^+(v)|$ may be less than d in d-out-regular multi-digraphs. Therefore, a random Pointer-Push&Pull operation may cancel with a probability proportional to the number of multi-edges of v_1 and v_2.

Based on the following lemma and on the observation that $O(nd)$ Pointer-Push&Pull operation can transform any weakly connected d-regular multi-graph to any other such graph the generality follows.

Lemma 3. *[20] The random Pointer-Push&Pull operation is symmetric for out-regular multi-digraphs, i.e. for two out-regular multi-digraphs G, G' the probability to transfom G to G' by a random Pointer-Push&Pull operation is the same as to transform G'*

Algorithm 2 Random Pointer-Push&Pull

$v_1 \leftarrow$ random node $\in V$
if random event with $p = \frac{|N^+(v_1)|}{d}$ occurs **then**
  $v_2 \leftarrow$ random node $\in N^+(v_1)$
  if random event with $p = \frac{|N^+(v_2)|}{d}$ occurs **then**
    $v_3 \leftarrow$ random node $\in N^+(v_2)$
    $E \leftarrow (E \setminus \{(v_1,v_2),(v_2,v_3)\}) \cup \{(v_1,v_3),(v_2,v_1)\}$

to G by a random Pointer-Push&Pull operation, i.e.

$$P[G \xrightarrow{PP} G'] = P[G' \xrightarrow{PP} G] \in \left\{0, \frac{1}{nd^2}\right\}$$

where $G \xrightarrow{PP} G'$ denotes that G can be transformed to G' with a single Pointer-Push&Pull operation.

Theorem 3. *[20] Let G' be an arbitrary d-out-regular weakly connected multi-digraph with n nodes. Then, applying random Pointer-Push&Pull operations repeatedly to G' will construct every d-out-regular weakly connected multi-digraph with the same probability in the limit, i.e.*

$$\lim_{t\to\infty} P[G' \xrightarrow{t} G] = \frac{1}{|MDG_{n,d}|},$$

where $MDG_{n,d}$ denotes the set of all d-out-regular weakly connected multi-digraphs with n nodes.

Interestingly, the Pointer-Push&Pull operation cannot used to generate all out-regular simple digraphs with the same probability. With a slight change of the probability distribution for the choice of edges multi-digraphs with simple digraphs occur with higher probability.

The Pointer-Push&Pull operation is modified to work with *d-out-regular edge labeled multi-digraphs*, which are defined by a node set $V = \{1,\ldots,n\}$ and an edge set $E^* = \{(u,v,i) : u,v \in V, i \in \{1,\ldots,d\}\}$, where i specifies the label of an edge. The unique labels are numbered as $1,\ldots,d$, i.e. $\forall u \in V, \forall i \in \{1,\ldots,d\} : \exists v \in V : (u,v,i) \in E^*$. In d-out-regular edge labeled multi-digraphs we use the notation $N^+(v,i)$, $i \in \{1,\ldots,d\}$, $v \in V$ to refer to v's neighbor due to the i-th labeled edge. In addition we use $E^-(v) = \{(w,v,i) \in E^*\}$ to refer to the set of v's ingoing edges. Recall that $N^+(v,i) = N^+(v,j)$, $i \neq j$ is possible in a multi-digraph.

Algorithm 3 shows the random Pointer-Push&Pull operation, modified for edge labeled multi-digraphs.

This operation is designed to provide the symmetric probability property.

Lemma 4. *[20] The labeled-Pointer-Push&Pull operation PP^* is symmetric within the domain of out-regular weakly connected edge labeled multi-digraphs. That is, for two arbitrary graphs G_1^*, G_2^* of this domain*

$$P[G_1^* \xrightarrow{PP^*} G_2^*] = P[G_2^* \xrightarrow{PP^*} G_1^*].$$

Algorithm 3 Random labeled-Pointer-Push&Pull

$v_1 \leftarrow$ random node $\in V$
$i \leftarrow$ random number $\in \{1,...,d\}$
$v_2 \leftarrow N^+(v_1,i)$
$j \leftarrow$ random number $\in \{1,...,d\}$
$v_3 \leftarrow N^+(v_2,j)$
$E^* \leftarrow (E^* \setminus \{(v_1,v_2,i),(v_2,v_3,j)\})$
$\quad \cup \{(v_1,v_3,i),(v_2,v_1,j)\}$

Finally, these results lead to the following theorem showing that the labeled-Pointer-Push&Pull operation provides uniform generality within the domain of out-regular weakly connected edge labeled multi-digraph.

Theorem 4. *[20] Let G_0^* be an d-out-regular weakly connected edge labeled multi-digraph with n nodes. Then, applying random labeled-Pointer-Push&Pull operations repeatedly to the graph will construct every d-out-regular weakly connected edge labeled multi-digraph with the same probability in the limit, i.e.*

$$\lim_{t\to\infty} P[G_0^* \xrightarrow{t} G^*] = \frac{1}{|MDG_{n,d}^*|},$$

where $MDG_{n,d}^*$ denotes the set of all d-out-regular weakly connected edge labeled multi-digraphs.

This probability distribution gives simple di-graphs a higher probability than the unlabeled version. The probability of a node being the source of only simple edges (not multiple edges or self-loops) is at least $1-\frac{d}{n-d-1}$ and suffices practical needs. Still, the fraction of simple digraphs created by the labeled-Pointer-Push&Pull operation is rather small, i.e. decreasing exponentially with the degree.

Since multiple edges are an unnecessary waste of resources one might want to generate simple digraphs only. A straight forward solution is to modify the labeled- or unlabeled-Pointer-Push&Pull operation such that it is only applied if the resulting digraph is simple. Unfortunately, the simple-Pointer-Push&Pull operation is not general for the domain of simple digraphs. To see this consider any symmetric digraph, i.e. a digraph where for each edge (u,v) also the edge (v,u) is in the edge set. In such digraphs no simple-Pointer-Push&Pull operation can be applied, since each operation would either create a multi-edge or a self-loop and therefore leave the domain of simple digraphs. The only way to reach all simple digraphs using Pointer-Push&Pull operations is to allow multiple occurrences of edges, i.e. the use of multi-digraphs.

For the question of convergence speed we conjecture $O(dn\log n)$ parallel operations. Despite numerous simulations, this question is open.

2.4 Pointer-Push

The Pointer-Push operation is an example of a feasible and sound random local operation for multi-graphs that preserves connectivity, yet completely lacks generality. It should not considered for the purpose of generating robust graphs. However, we present it as a show piece for unwanted behavior of local transformations.

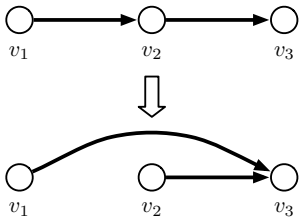

Fig. 4. The Pointer-Push operation.

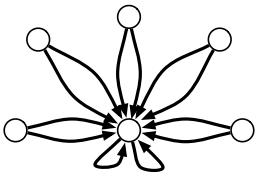

Fig. 5. The star-graph as the terminal graph of the Pointer-Push operation.

The *Pointer-Push* performs a random walk v_1, v_2, v_3 in G and replaces (v_1, v_2) with (v_1, v_3) (see Fig. 4).

Figure 4 illustrates the Pointer-Push operation. A randomized version of the Pointer-Push operation is given by Algorithm 4.

Algorithm 4 Random Pointer-Push

Choose random node $v_1 \in V$
$v_2 \leftarrow$ random node $\in N^+(v_1)$
$v_3 \leftarrow$ random node $\in N^+(v_2)$
$E \leftarrow (E \setminus (v_1, v_2)) \cup (v_1, v_3)$

The lack of generality becomes apparent, when a long series of random Pointer-Push operation are performed in a weakly connected multi-graph. Intuitively, nodes with higher indegree are prone to receive even higher indegree.

Theorem 5. *[20] A series of random Pointer-Push operations will transform any connected multi-digraph into a connected set of stars in the limit with probability* 1.

Clearly, the star-graph (Fig. 5) is not a robust structure for a peer-to-peer network because it constitutes a client-server-network with the vulnerability of the central node. Furthermore, experiments indicate fast convergence to this undesirable graph.

2.5 Pointer-Pull

The Pointer-Pull operation is an example of a feasible random local operation that is not sound. The *Pointer-Pull* performs a random walk v_1, v_2, v_3, v_4 in G and replaces (v_3, v_4) with (v_3, v_1) (see Figure 6).

A randomized version is given by Algorithm 5.

Algorithm 5 Random Pointer-Pull

Choose random node $v_1 \in V$
$v_2 \leftarrow$ random node $\in N^+(v_1)$
$v_3 \leftarrow$ random node $\in N^+(v_2)$
$v_4 \leftarrow$ random node $\in N^+(v_3)$
$E \leftarrow (E \setminus (v_3, v_4)) \cup (v_3, v_1)$

As in case of the Pointer-Push operation, the Pointer-Pull operation does not change the outdegree of any node. The intuition, in contrast to the Pointer-Push operation, is that applying random Pointer-Pull operations may balance the indegree of the nodes. This is because the starting node v_1 of each operation will increase its indegree by one and v_1 is chosen uniformly at random. Furthermore, nodes with high indegree have a higher probability to be endpoint of a Pointer-Pull operation and therefore, higher probability to get their indegree decreased. Even if this is the case, the following theorem shows a major drawback of the Pointer-Pull operation.

Theorem 6. *[20] Starting with an arbitrary multi-digraph G with n nodes, random Pointer-Pull operations disconnect G into n components of single nodes with slopes in the limit.*

The Pointer-Pull operation is not able to preserve connectivity in a graph and therefore is not sound. To see this, note that there is a non-zero probability for a node to create self-loops. Once all edges of a node are pointing to itself this node is disconnected from the rest of the graph. Without global knowledge such a situation is irrevocable and therefore the graph will consist only of disconnected nodes in the long run.

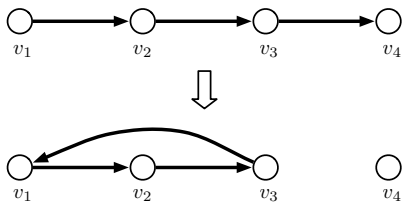

Fig. 6. The Pointer-Pull operation.

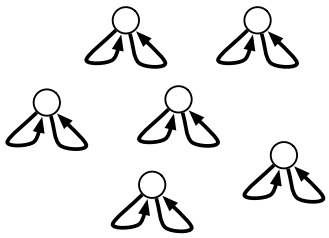

Fig. 7. This graph of islands is the terminal graph of the Pointer-Pull operation.

While the mathematical analysis is straight-forward, the convergence to this terminal graph is rather slow as experiments show. For multi-graphs with large degree, large number of nodes and a random start graph empirical tests do not show any graph partitions during the test time. One might argue that this shows the superiority of mathematical proofs. Yet, it could also be interpreted the operation is sound enough for practical use if the degree is not too small.

Finally, note that a Pointer-Push&Pull can be seen as a combination of Pointer-Push and Pointer-Pull combining the balancing effect of the Pull operation with the connectivity of Push.

2.6 k-Flipper

In [19] the Flipper operation has been generalized by replacing the hub edge with a random walk of length k. It is a feasible and sound random graph transformation. The convergence to expander graphs is provable fast, if one sacrifices the locality of the operation by long random walks.

Unlike as the Flipper, the k-Flipper can disconnect a graph. Figure 9 shows a k-Flipper operation which uses the flipping edge $\{u_1, u_2\}$ twice such that the resulting

graph is partitioned into disconnected components. In order to preserve connectivity we have to ensure that there is a hub path between nodes u_2 and u_{k+2} of a k-Flipper operation without using the flipping edges. On the other hand we do not want to bias the random walk by forbidding to use the first edge or the a priori unknown last edge.

Fortunately, this problem is easy to handle. A simple solution is to truncate the path $P = (u_1, \ldots, u_{k+3})$ to a path $P' = (u_\ell, \ldots, u_{r+1})$ with $1 \leq \ell < r < k+3$ such that $\{u_\ell, u_{\ell+1}\} = \{u_1, u_2\}$, $\{u_r, u_{r+1}\} = \{u_{k+2}, u_{k+3}\}$ and $\{u_1, u_2\}, \{u_{k+2}, u_{k+3}\}$ do occur only once in P' (see Figure 8). This observation leads to the following algorithm.

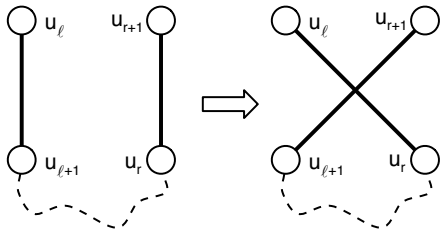

Fig. 8. The k-Flipper with truncated hub path.

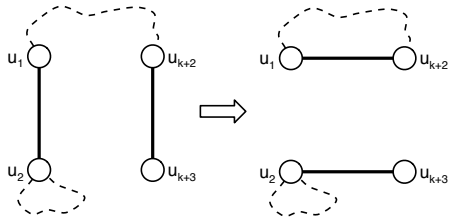

Fig. 9. A k-Flipper can disconnect a graph.

Algorithm 6 Random k-Flipper

Choose random node $u_1 \in V$
for $i \leftarrow 1$ **to** $k+2$ **do**
  Choose random node $u_{i+1} \in N(u_i)$
for $i \leftarrow k+2$ **downto** 2 **do**
  if $\{u_i, u_{i+1}\} = \{u_{k+2}, u_{k+3}\}$ **then** $r \leftarrow i$
for $i \leftarrow 1$ **to** r **do**
  if $\{u_i, u_{i+1}\} = \{u_1, u_2\}$ **then** $\ell \leftarrow i$
if $r \geq \ell + 2$ and $\{u_\ell, u_r\}, \{u_{\ell+1}, u_{r+1}\} \notin E$ **then**
  $E \leftarrow E \setminus \{\{u_\ell, u_{\ell+1}\}, \{u_r, u_{r+1}\}\}$
  $E \leftarrow E \cup \{\{u_\ell, u_r\}, \{u_{\ell+1}, u_{r+1}\}\}$

Therefore, it is not clear if the Random k-Flipper provides uniform generality. Nevertheless, it establishes an expander graph in a polynomial number of rounds for large random walks.

Theorem 7. *[19] If we choose $d \in \Omega(\log n)$ applying $O(dn)$ Random $\Theta(d^2 n^2 \log 1/\varepsilon)$-Flipper operations transforms any given d-regular connected graph into a connected d-regular graph with expansion $\Theta(d)$ with high probability.*

The Random k-Flipper is not symmetric, i.e. the transition probability from graph G to G' can differ from the transition probability from G' to G. It is unknown whetther the Random k-Flipper with $k > 1$ provides uniform generality. Slightly modified versions can provide symmetry. For this, the random walk needs to avoid to traverse the flipping edges more than once. The Random k-Flipper is allowed to traverse these edges and a sub-path of the random walk will be chosen. This choice causes the break of symmetry. If we avoid visiting nodes more than once we have the *Node Disjoint Random k-Flipper*

and if we avoid visiting edges more than once we get the *Edge Disjoint Random k-Flipper*. Both operations have symmetric transition probabilities. However, when using these operations long random walks are not possible, especially not paths of length $\Theta(d^2n^2 \log 1/\varepsilon)$. So, the proof of Theorem 7 cannot be applied since the random walk used in these modified versions is biased.

However, as soon as the expansion property is established one can reduce the length of the random walk of the Random k-Flipper to a polylogarithmic term. Possibly, such short random walks are always sufficient. Or as conjectured above, the 1-Flipper operation is already the solution.

3 Applying Operations in Peer-to-Peer-Networks

We have seen that Flipper and Pointer-Push&Pull represent local graph transformation suited for upholding connectivity and optimizing the network structure by randomizing the network. Furthermore the network connections are validated automatically by the ongoing local operations. These operations are started distributedly by every peer from time to time. Each peer regularly chooses to initiate a random process constituting a local random graph operation. Since we are now considering computer networks and not merely graphs, problems like deadlocks, peer failures and peer arrival need to be investigated. We will discuss these problems for some of the above introduced operations.

3.1 Node Arrival

For joining a peer-to-peer network a candidate has to know at least one node of the network. In Gnutella this problem has been solved by hard-wiring addresses of peers, who usually attend the network, into the program code. Other networks use external ways of communicating entry nodes, like web-pages, forum entries, e-mails, etc. We start our considerations from the point where a joining peer knows one peer in the network.

In directed multi-graphs the joining process is very simple. The new peer initiates all edges as pointers to the known peer in the network. The local random graph transformation, e.g. Pointer-Push&Pull, will automatically replace these edges later on. If the network arrival rate is higher than the graph transformations can handle it is helpful to perform d random walks from the the first known peer and connect to these points. Tests show that this increases the stability of peer-to-peer networks dramatically.

In regular graphs the main problem is to uphold the degree bound. If the degree d is an even number, the following algorithm works very well [19]. Randomly choose $d/2$ connections defined by d distinct nodes, erase these connections and connect to each of the d nodes: “The peer places itself in the middle of these connections”, see Fig. 10. The distinctness of the d peers of these connections is crucial, since otherwise the new peer would either create multiple connections to some peers or reduce the degree of some peers violating the d-regularity. For this, we start from the known peer v in the network and send a control message on a random walk. At each node visited u selects a random neighbor v' of v and creates a lock on the connection $\{v, v'\}$ preventing other nodes from selecting this edge. These locks prevent graph transformation operations

from choosing these edges. According to [1] the length of the random walk can be chosen as $O(d^2 \log d)$. In the expectation this number of hops is sufficient to detect d distinct nodes in any graph by a random walk. If after $O(d^2 \log d)$ hops not enough random connections could be found the random walk is canceled. This process can fail if either the network is too small or if the network structure is bad (which is unlikely, yet possible). A new peer which could not find enough neighbors can retry finding new neighbors by the same procedure after some time. If the control message has successfully finished the random walk reporting enough neighbors, u is contacted by enough peers of the network and can places itself in the middle of each of the collected connections.

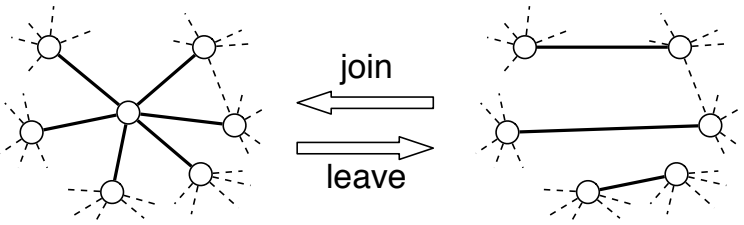

Fig. 10. Peer arrival and departure in a regular graph

3.2 Edge Failure

Edges can fail because they cannot be established in the first place. In the Internet there is no guarantee if node u can connect with node v and node v can connect with node w that then node u can connect with w. Such problems are often caused by Network Address Translation (NAT) and Firewalls. Another reason for edge failure without node failure can be high traffic on the edge. Then, random packets will be dropped by the router and there is a chance that a peer-to-peer connection appears to have failed, because its packets happen to be erased.

Edge failures endanger the network connectivity. Edges should be dropped reluctantly, since the random operations do not perform a connectivity test. They guarantee connectivity only in the absence of failures. Yet, if an expander has been established, the partition of the network by a single edge failure occures with extremely small probability.

As a repair mechanism in directed multi-graphs the peer with the failed edge simply chooses another edge and duplicates the entry. The self-repair mechanism of the local random graph transformation, e.g. Pointer-Push&Pull, will replace this edge later on. The positive side-effect of this mechanism is that an edge failure does not induce additional traffic. Especially, if the edge has been failed because of network overload this may help the network to recover, or at least does not worsen the situation.

For regular undirected graphs an edge failure violates the degree property at two possibly otherwise distant nodes of the graph. It is impossible to reconnect these nodes without an extra data structure or without an extensive search in the graph. Yet, the benefit of having exactly d neighbors is rather small if d is a large constant like 16. An easy solution to this problem is to allow nodes to have either d or $d-1$ neighbors

leading to so-called $\{d, d-1\}$-regular networks. Then, only if the edge failure occurs at a $d-1$ degree node u then this node can perform a random walk using the other edges to find a non-adjacent edge $\{v,w\}$ in the network. Then, this edge $\{v,w\}$ is replaced by two edges $\{u,v\}$, $\{u,v\}$.

3.3 Node Failure

The network experiences a node failure by detecting the corresponding edge failures. Of course, this endangers connectivity more than a failure of single edge. Yet, for sufficient large degree the chances of partitioning the graph is extremely small, if the graph is an expander. It is advisable to increase the update frequency of the local graph transformations for peer-to-peer networks with high churn rate.

If the node announces its departure from the network this can be used to guarantee d-regularity in undirected graphs by using the somewhat inverse operation of a network join. As well as for undirected graphs as for multi-digraphs no local operation is known to guarantee the connectivity of the network even in this benign case. Other solutions presented so far involve the existence of a central entity or additional global data structures.

3.4 Concurrency

Concurrent local graph operations can disconnect the graph. As an example, consider the random k-Flipper operation. If the hub paths of two k-Flippers use the flip edge of the concurrent process, then the network can be disconnect as shown in Fig. 11.

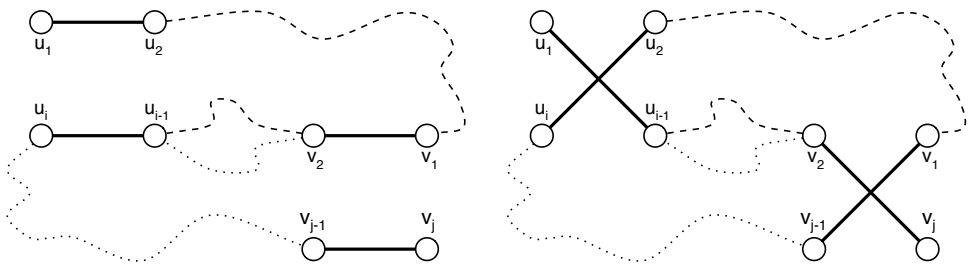

Fig. 11. Concurrent Random k-Flipper operations can disconnect a graph. The graph at the top shows random walks performed by two Random k-Flipper operations U (dashed) and V (dotted), which result in the disconnected graph shown beneath.

This problem can be solved by introducing soft locks (which disappear after a certain time) on all involved nodes. Then concurrent local graph operations are canceled as soon as they interfere with existing operations. Since the whole operation is canceled deadlocks are prevented. Because of the randomization also livelocks are prevented. Note that this locking mechanism reduces the parallelism in the operation, and thus small local algorithms like Flipper locking four nodes and Pointer-Push&Pull locking two nodes perform much better than operations like k-Flipper. Especially, this is the case if $k \in \Theta(d^2 n^2 \log 1/\varepsilon)$ is chosen as in Theorem 7 then the whole network is swamped with locks for one operation.

4 Convergence Rate

Except for the convergence rate of the k-Flipper operation with extraordinary large k little is known for the better local graph transformations Flipper and Pointer-Push&Pull. For the convergence rate of Pointer-Push&Pull we conjecture $O(dn\log n)$ parallel operations. At the moment, not even a polynomial bound for the convergence is known. This may due to the fact that this operation use multi-digraphs which complicates the analysis compared to the undirected graphs of the Flipper operation.

Also for the Flipper little is known. Recently in [8] is was shown that the convergence rate can be bounded by a high degree polynomial, i.e. $O(d^{56}n^{53})$. Now we present the simulation results for the Flipper operation which indicate that this bound is far apart from the true bound.

There is no method known at the moment which can test the true randomness of a graph. Therefore we concentrate on the expansion of the graph, which is known to be *NP*-hard. We approximate a lower bound as shown in [2]. Consider the adjacency matrix of a regular graph G. Since the matrix A is symmetric there are n real eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$. These eigenvalues of A(G) are called the Spectrum (denoted as $\sigma(G)$) of the graph G. For the first eigenvalue we have $\lambda_1 = d$ and the corresponding eigenvector $v_1 = (n^{-1/2}, n^{-1/2}, \ldots, n^{-1/2})$.

For a d-regular graph G with spectrum $\lambda_1 \geq \ldots \geq \lambda_n$ we have for the expansion $\beta = h(G)$ of the graph G the following equation.

$$\frac{d-\lambda_2}{2} \;\leq\; h(G) \;\leq\; \sqrt{2d(d-\lambda_2)}$$

The term $d-\lambda_2$ is also known as the *Spectral Gap*. A large spectral gap implies a high expansion. The second eigenvalue of G is at last $2\sqrt{d-1}-o(1)$ (Alon-Boppana 1986).

It follows that $d-\lambda_2 \leq d-2\sqrt{d-1}$ which reduces the interval for the spectral gap from $[0,2d]$ to $[0,d-2\sqrt{d-1}]$ which is a better estimation. Hence, we define a functions $\rho_1(d) = \frac{d-2\sqrt{d-1}}{2}$ which approximates the lower bound for the expansion ratio of a graph from above. We apply this function to the series of transformed graphs to check when a graph is a good expander.

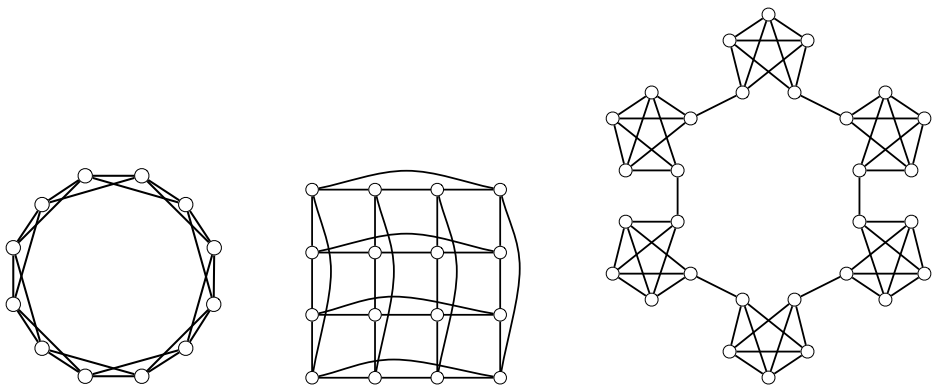

Fig. 12. For the tests the start graphs ring, torus and ring of cliques are used

The calculation of the spectral gap $d-\lambda_2$ of a graph G uses the computation of the second largest eigenvalue of $A = A(G)$. Due to the fact that A is symmetric (since G is undirected) and positive definite the Vector-Iteration Algorithm by F. L. Bauer is an appropriate method for this calculation [27]. Our simulations show that the Flipper operations seem to have been converged when we observe the following property.

Definition 4. *[2] We assess a d-regular graph G as an expander graph when the lower-bound* $(d-\lambda_2)/2$ *of* $h(G')$ *satisfies*

$$\frac{d-\lambda_2}{2} \geq 0.98\, \rho_1(d)\ .$$

The convergence behavior of this expansion estimator depends on the start graph, see Fig. 13. All start graphs have 10,000 nodes and degree 4. The torus-graph is a two-dimensional torus of 100×100 nodes. The ring of 10,000 nodes is enhance with edges to nodes with distance 2 on the ring. The ring of cliques consists of 200 cliques on a ring of size 5 where one edge is removed and the adjacent nodes now have edges to the left and right cliques on the ring, see Fig. 12. Every curve results describes the average of 10 tests.

In our experiments the ring of cliques had the worst performance. Fig. 14 exemplifies the convergence for a single test instance starting with the 10,000 nodes 4-regular ring of cliques. We observe the following phases:

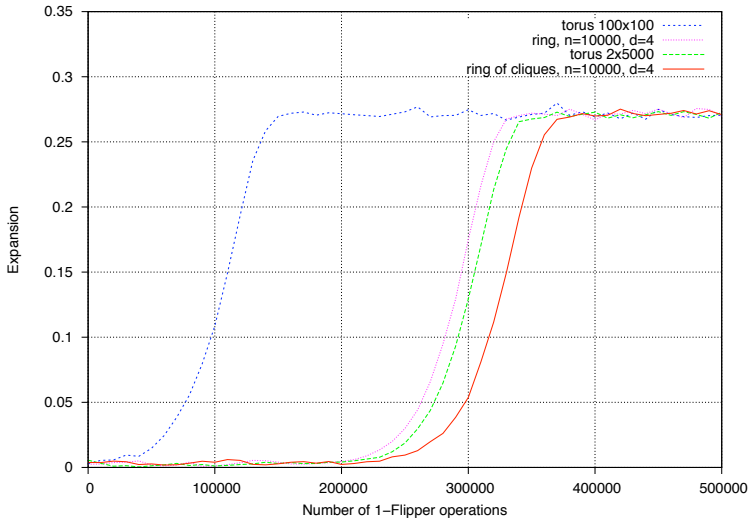

Fig. 13. The start graph influences the convergence speed of the Flipper operation

1. Initial phase (first 25,000 Flipper operations): steady small expansion
2. Expansion phase (Flipper operations 25,0000-35,000): the expansion grows dramatically
3. Stable terminal phase (after 35,000 Flipper operations): the maximal expansion has been reached

The Flipper operation cannot be applied when there is a triangle with a hub edge and a flip edge as two of the edges. Fig. 15 shows that during the initial phase the graph is already modified. Then, the number of triangles and the diameter of the graph

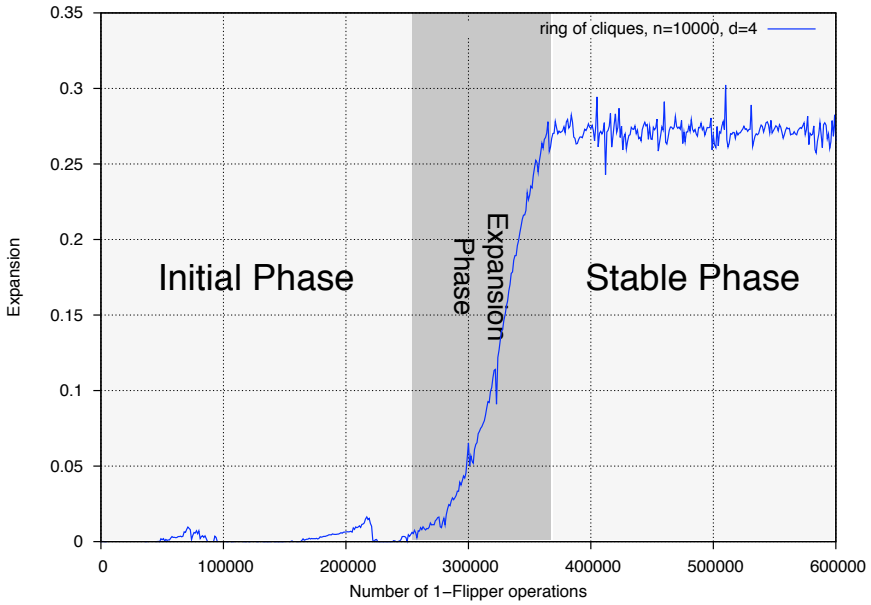

Fig. 14. Convergence phases of Flipper operations started on 10,000 node 4-regular ring of cliques

is reduced dramatically. This seems to be a pre-requisite for the expansion phase. The test has been performed for a 1,000 node 4-regular ring of cliques and is averaged over 10 test runs.

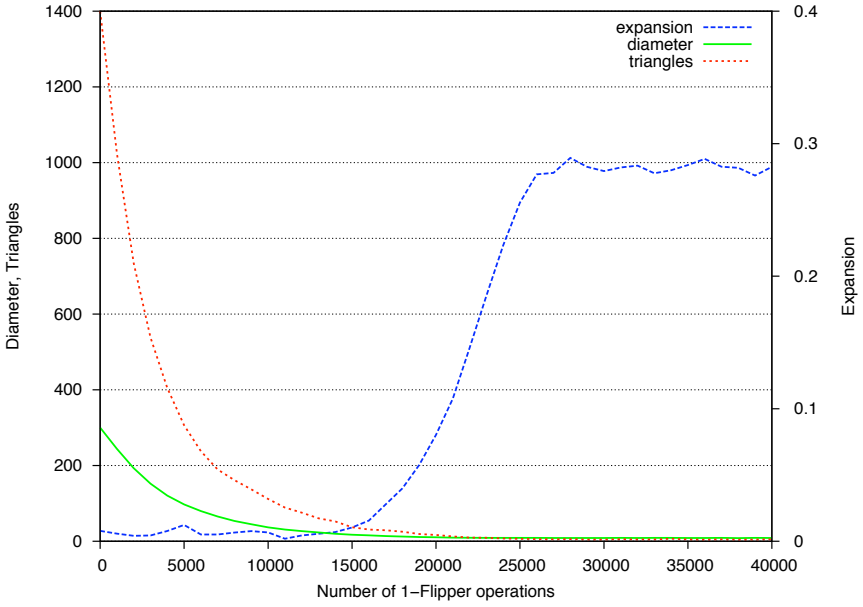

Fig. 15. Expansion, diameter and number of triangles for Flipper operation started on a 1,000 node 4-regular ring of cliques

The convergence rate depends on the degree of the graph. Since $dn/2$ edges need to be changed one expects at least a linear growth of the convergence speed depending on the degree. In fact the growth seems to be linear as Fig. 16 indicates. Starting from a d-regular ring of cliques with 10,000 nodes the number of operations to reach maximal

expansion is shown. The figure shows the average over 10 measurements with distance showing the standard deviation.

We conjecture a convergence time of $O(dn\log n)$ for the number of Flipper operations. The graph in Fig. 17 shows the number of operations necessary to reach the stable phase for degrees 4, 8, and 12 for growing number of nodes. The start graph is a ring of cliques with 10,000 nodes. As a comparison the graphs of the functions $4n\log n$ and $8n\log n$ are added to the diagram. Again the values have been averaged over 10 test runs.

As a generalization of the Flipper we have presented the k-Flipper. Fig. 18 shose that with increasing hub-path length k it provides faster convergence. The start graph is a ring of cliques of 10,000 nodes. 10 test runs have been averaged. There is a difference between the 1-Flipper and the Flipper when it comes to the success rate of an operation. Hence, the 1-Flipper performs by a factor of 2 worse than the Flipper operations.

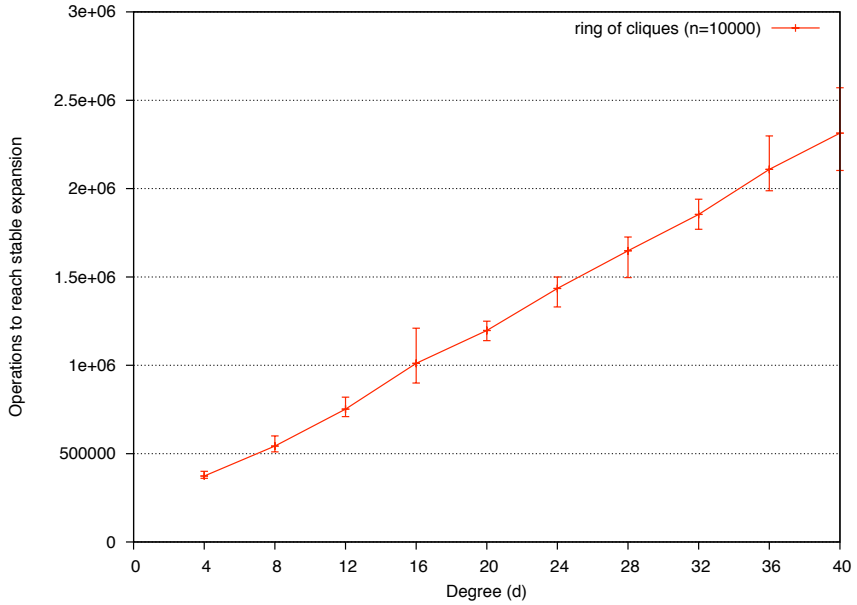

Fig. 16. Number of Flipper operations to reach stable expansion for variable degree d starting from a 10,000 node ring of cliques

This last Fig. 19 shows the growth of expansion for various k-flippers with respect to the number of operations. The curves are averaged over 10 test runs each. The start graphs were 4-regular 10,000 nodes ring of cliques. Again for $k = 1$ we have chosen the 1-Flipper operation instead of the Flipper operation.

5 Conclusion and Open Questions

We have presented a variety of random, local, distributed graph transformation and discussed how they can be applied to peer-to-peer networks to uphold a random, robust, stable, and self-healing network.

The best choices seem to be the Pointer-Push&Pull and Flipper operations because of their simplicity, the small number of involved nodes, and the fast (conjectured) convergence rate. For both operation we conjecture that they converge in time $O(dn\log n)$

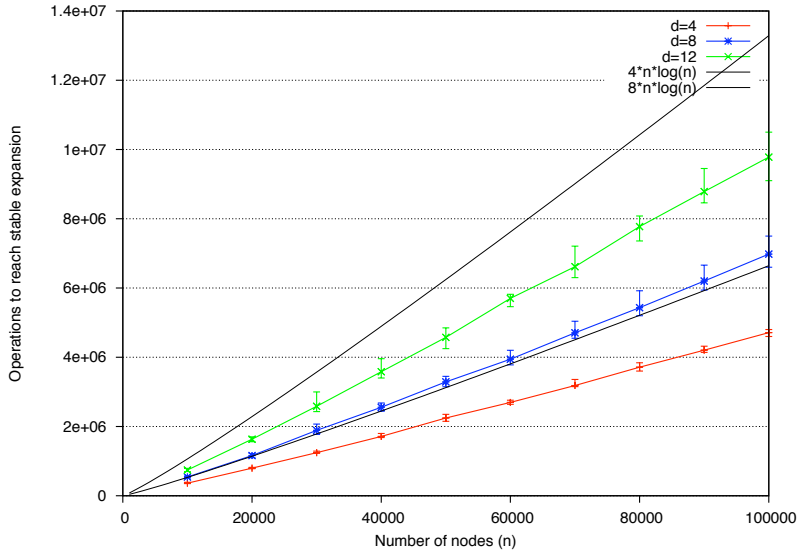

Fig. 17. Number of Flipper operations to reach stable expansion for variable degree d and variabe number of nodes starting from node ring of cliques

to a truly random graph which provides a good expansion. For the Flipper our empirical tests presented in Section 4 strengthen our conjecture.

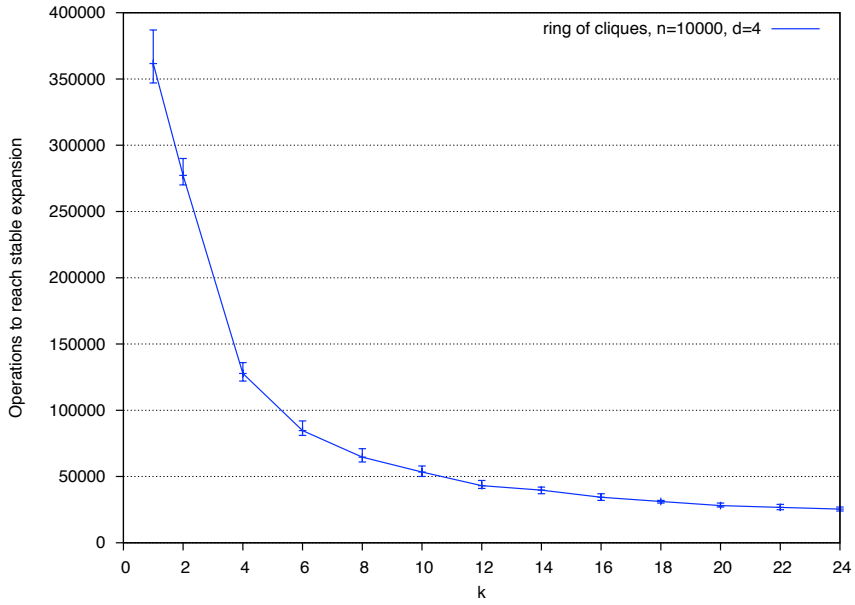

Fig. 18. Number of k-Flipper operations to reach stable expansion for 10,000 nodes and 4-regular ring of cliques

It turned out that k-Flipper operations converge even faster. Yet, parallel random k-Flipper operations may block each other or in the best case are performed sequentially. For the networking point a small choice for k is highly desirable. However, little is known about the convergence rate except for the case of expander graphs. In an expander graph $k \in O(\log n)$ can be used. The above presented tests underline the thesis that the parallel use of Random 1-Flipper converges as fast as the sequential use of Random k-Flipper operations.

We have measured the time until the Random Flipper constructs an expander graph. It is not clear, whether this implies that the terminal probability distribution has been reached then. Anyhow, the establishment of an expander graph is the main motivation for this operation.

In this paper we have only discussed regular graphs and multi-digraphs, but no digraphs. It turns out that the Pointer-Push&Pull operation is unable to maintain simple digraphs. It is not clear if there exists a similar operation for simple digraphs suitable for peer-to-peer networks.

The research in this field of local random graph transformation is still in its infancy. At the moment empirical tests are the best method at hand to evaluate the convergence of these operations. For connectivity or the terminal distributions rigorous analysis has been proven to be very effective. Here, empirical analysis may lead to false claims, as we have seen for the case of Pull operations.

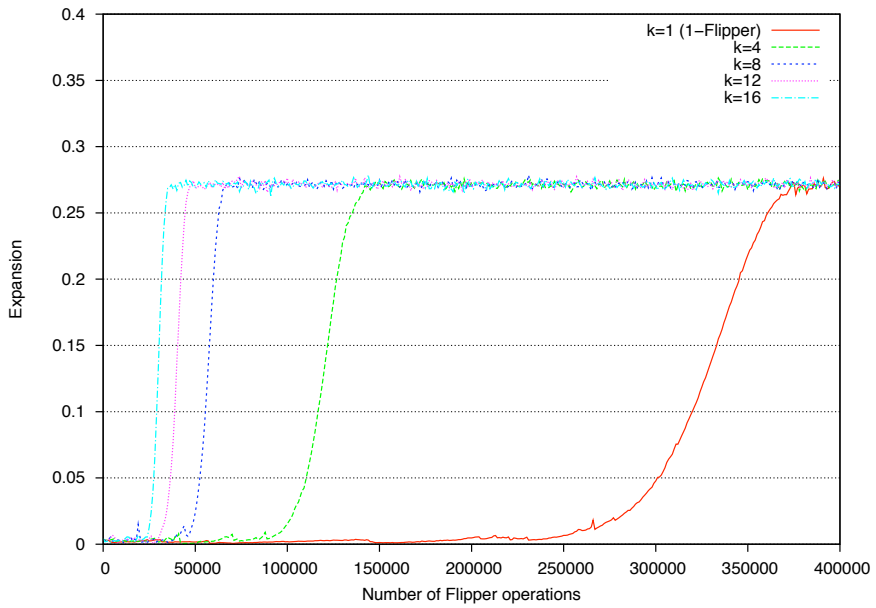

Fig. 19. Convergence speed of k-Flipper operations with the 10,000 4-regular ring of cliques as start graph

The open questions in this area is to find better operations and to improve the analytic tools to get a mathematical understanding of these operations. From a practical point of view there is no reason why stable backbones should not be provided by random networks upholded by Flipper or Pointer-Push&Pull operations. This technology developed within the DELIS project enhances peer-to-peer overlays with a robust connection network.

References

1. G. Barnes and U. Feige. Short random walks on graphs. In *STOC '93: Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 728–737. ACM Press, 1993.
2. D. Baldin, Analysing the Graph Expansion of Flipper Maintained Random Graphs Bachelor Thesis at the University of Paderborn, Department of Computer Science, 2007
3. B. Bollobas. Random graphs. *Combinatorics*, 52:80–102, 1981.
4. B. Bollobas. The isoperimetric number of a random graph. *European Journal of Combinatorics*, 9:241–244, 1988.

5. S. M. Botros and S. R. Waterhouse. Search in jxta and other distributed networks. In *Proceedings of the 1st International Conference on Peer-to-Peer Computing (P2P 2001)*, pages 30–35, 2001.
6. C. Cooper, M. Dyer, and C. Greenhill. On markov chains for random regular graphs, 2005. Unpublished Draft.
7. C. Cooper, M. Dyer, and C. Greenhill. Sampling regular graphs and a peer-to-peer network. In *SODA '05: Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 980–988, Philadelphia, PA, USA, 2005. Society for Industrial and Applied Mathematics.
8. T. Feder, A. Guetz, M. Mihail, and A. Saberi. The flip markov chain and peer-to-peer networks. http://www.stanford.edu/∼saberi/switch.pdf, 2006.
9. C. Gkantsidis, M. Mihail, and A. Saberi. Random walks in peer-to-peer networks. In *IEEE INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, Mar. 2004.
10. Gnutella. The gnutella protocol specification v0.4.
11. K. Hildrum, J. D. Kubiatowicz, S. Rao, and B. Y. Zhao. Distribted object location in a dynamic network. In *Proceedings of the fourteenth Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA-02)*, pages 41–52, New York, Aug. 10–13 2002. ACM Press.
12. M. Jerrum and A. Sinclair. Fast uniform generation of regular graphs. *Theoretical Computer Science*, 73(1):91–100, 1990.
13. M. A. Jovanovic, F. S. Annexstein, and K. A. Berman. Scalability issues in large peer-to-peer networks — a case study of Gnutella. Technical report, University of Cincinnati, 2001.
14. 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 Systems (IPTPS'03)*, pages 98–107, 2003.
15. M. Kim and M. Medard. Robustness in large-scale random networks. In *IEEE INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, Mar. 2004.
16. M. Kim, M. Medard, and E. A. M. Torres. On reliability of large-scale random networks. http://web.mit.edu/minkyu/www/doc/ ToNsubmitted.pdf, 2005.
17. C. Law and K.-Y. Siu. Distributed construction of random expander networks. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 2133–2143, Mar. 2003.
18. Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. In *ICS '02: Proceedings of the 16th international conference on Supercomputing*, pages 84–95, New York, NY, USA, 2002. ACM Press.
19. P. Mahlmann and C. Schindelhauer. Peer-to-peer networks based on random transformations of connected regular undirected graphs. In *SPAA'05: Proceedings of the seventeenth annual ACM Symposium on Parallelism in Algorithms and Architectures*, pages 155–164, New York, NY, USA, 2005. ACM Press.
20. P. Mahlmann and C. Schindelhauer. Peer-to-peer networks based on random transformations of connected regular undirected graphs. In *SPAA'06: Proceedings of the eighteenth annual ACM Symposium on Parallelism in Algorithms and Architectures*, pages 308–317, Cambridge, Massachusetts, USA, 2006. ACM Press.
21. B. D. McKay. Subgraphs of random graphs with specified degrees. *Congressus Numerantium*, 33:213–223, 1981.
22. B. D. McKay and N. C. Wormald. Uniform generation of random regular graphs of moderate degree. *Journal of Algorithms*, 11:52–67, 1990.
23. M. Naor and U. Wieder. Novel architectures for p2p applications: the continuous-discrete approach. In *SPAA '03: Proceedings of the fifteenth annual ACM Symposium on Parallel Algorithms and Architectures*, pages 50–59, New York, NY, USA, 2003. ACM Press.
24. G. Pandurangan, P. Raghavan, and E. Upfal. Building low-diameter p2p networks. In *FOCS '01: Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science*, page 492, Washington, DC, USA, 2001. IEEE Computer Society.
25. M. Ripeanu. Peer-to-peer architecture case study: Gnutella network. In *Proceedings of the 1st International Conference on Peer-to-Peer Computing (P2P 2001)*, pages 99–100, 2001.
26. J. Ritter. Why gnutella can't scale. http://www.darkridge.com/∼jpr5/doc/gnutella.html, 2001.
27. H. Rutishauser. Computational aspects of F. L. Bauer's simultaneous iteration method In *Numerische Mathematik*, 13(1):4–13, 1969.
28. A. Steger and N. C. Wormald. Generating random regular graphs quickly. *Combinatorics, Probability and Computing*, 8:377–396, 1999.
29. I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A scalable Peer-To-Peer lookup service for internet applications. In R. Guerin, editor, *Proceedings of the ACM SIGCOMM 2001 Conference (SIGCOMM-01)*, volume 31, 4 of *Computer Communication Review*, pages 149–160, New York, Aug. 27–31 2001. ACM Press.
30. N. C. Wormald. The asymptotic connectivity of labelled regular graphs. *Journal of Combinatorial Theory, Series B*, 31(2):156–167, 1981.
31. N. C. Wormald. Models of random regular graphs. In *Surveys in Combinatorics, 1993, Walker (Ed.), London Mathematical Society Lecture Note Series 187, Cambridge University Press*. 1999.