# Scalable P2P Overlays of Very Small Constant Degree: an Emerging Security Threat*

Márk Jelasity[1] and Vilmos Bilicki[2]

[1] University of Szeged and Hungarian Academy of Sciences, Hungary, jelasity@inf.u-szeged.hu
[2] University of Szeged, Hungary, bilickiv@inf.u-szeged.hu

**Abstract.** In recent years peer-to-peer (P2P) technology has been adopted by Internet-based malware as a fault tolerant and scalable communication medium for self-organization and survival. It has been shown that malicious P2P networks would be nearly impossible to uncover if they operated in a *stealth mode*, that is, using only a small constant number of fixed overlay connections per node for communication. While overlay networks of a small constant maximal degree are generally considered to be unscalable, we argue in this paper that it is possible to design them to be scalable, efficient and robust. This is an important finding from a security point of view: we show that stealth mode P2P malware that is very difficult to discover with state-of-the-art methods is a plausible threat. In this paper we discuss algorithms and theoretical results that support the scalability of stealth mode overlays, and we present realistic simulations using an event based implementation of a proof-of-concept system. Besides P2P botnets, our results are also applicable in scenarios where relying on a large number of overlay connections per node is not feasible because of cost or the limited number of communication channels available.

## 1 Introduction

In recent years peer-to-peer (P2P) technology has been adopted by botnets as a fault tolerant and scalable communication medium for self-organization and survival [1, 2]. Examples include the Storm botnet [1] and the C variant of the Conficker worm [3].

The detection and filtering of P2P networks presents a considerable challenge [4]. In addition, it has been pointed out by Stern [5] that the potential threat posed by Internet-based malware would be even more challenging if worms and bots operated in a "stealth mode", avoiding excessive traffic and other visible behavior. It has also been shown that state of the art techniques for the detection of P2P networks fail if peers communicate with only a small constant number of neighbors during their lifetime [6].

Fortunately, current infections generate considerable traffic. For example, the Storm worm contacts a huge number of peers, in the range of thousands [2], when joining the network, generating a recognizable communication pattern as well as revealing a large list of botnet members. In general, P2P clients typically contact a large number

* In: Proc. SSS'09, pp 399–412, LNCS 5873, doi:10.1007/978-3-642-05118-0_28 M. Jelasity was supported by the Bolyai Scholarship of the Hungarian Academy of Sciences.

of neighbors due to maintenance traffic, and regular application traffic such as search. There are only a few notable exceptions, such as Symphony and Viceroy [7, 8], which are overlay networks of a constant degree.

It is still an open question whether it is possible to create overlay networks of a very small constant maximal degree that are efficient and scalable. Research activity concerning Symphony or Viceroy has not yet been targeted to the lower end of maximal node degree, potentially as small as 3 or 4. In fact, even negative results are known that indicate the inherent lack of scalability of constant degree networks [9, 10]. In this paper we answer this question in the affirmative and show that it is possible to build a Symphony-inspired overlay network of a very small constant degree, and with the application of a number of simple techniques, this overlay network can be made scalable and robust as well. This result calls for more research into the detection of malicious P2P networks that are potentially of a small maximal degree.

Our contribution is threefold. First, we empirically analyze known as well as new techniques from the point of view of improving the search performance in a Symphony-like overlay network. Second, we present theoretical results indicating that if we add $O(\log N)$ (or, in a certain parameter range, $O(\log \log N)$) backup links for all links (where $N$ is the network size), then a constant degree network becomes fault tolerant even in the limit of infinite network size, while its effective degree remains constant (that is, the network can remain in stealth mode) since the backup links are not used for communication unless they become regular links replacing a failed link. This result is counter intuitive because routing in Symphony requires $O(\log^2 N)$ hops on average. Third, we provide event-based simulation results over dynamic and realistic scenarios with a proof-of-principle implementation of a constant degree network, complete with gossip-based protocols for joining and maintenance.

## 2 Performance of Small Constant Degree Topologies

Our motivation is to understand whether a reasonable routing performance can be achieved in a network that operates in stealth mode; that is, where the maximal node degree is as small as 3 or 4. We will base our discussion on Symphony, a simple constant degree network [7]. We explore several (existing and novel) simple techniques for improving the routing performance of Symphony at the lower extremes of maximal degree. To the best of our knowledge, this problem has not been tackled so far in detail by the research community.

Symphony was proposed by Manku et al. [7] as an application of the work of Kleinberg [11]. Like many other topologies, the Symphony topology is based on an undirected ring that is ordered according to node IDs. Node IDs are drawn uniformly at random from the interval $[0, 1]$ when joining the network. Apart from the two links that belong to the ring, each node draws a constant number of IDs with a probability proportional to $1/d$, where $d$ is the distance from the node's own ID. Subsequently, each node creates undirected long-range links to those peers that have the closest IDs to the IDs drawn. (We note that an implementation needs an approximation of the network size $N$ for normalizing the distribution. A rough, but practically acceptable, approximation exploits the fact that the expected distance of the closest neighbor in the ring is $1/N$.)

Symphony applies a greedy routing algorithm: at each hop the link is chosen that has the numerically closest ID to the target. Due to the undirected ring, the procedure is guaranteed to converge. It can also be proven that routing takes $O(\log^2 N)$ hops on average; the idea of the proof is to show that it takes $O(\log N)$ hops to halve the distance to the target.

In the following we describe techniques for reducing the number of routing hops in small constant degree networks. Subsequently, we systematically analyze these techniques via simulations.

*Lookahead.* Greedy routing can be augmented by a lookahead procedure where nodes store the addresses of the neighbors of their neighbors locally as well, up to a certain distance. This way, route selection is based on the best 2, 3, etc., hop route planned locally as opposed to a 1 hop route. Routing with a single hop lookahead has been studied in detail [12, 13]. Since small constant degree networks have small local neighborhoods that can easily be stored and updated, we study 2 hop lookahead as well.

*Degree balancing.* To enforce a strict small upper bound on node degree, nodes that are already of the maximal degree have to reject new incoming long-range links. To make sure that most joining nodes can create long-range links, we need to introduce balancing techniques. In addition to the usual technique of repeated join attempts, we propose degree balancing: when a node of maximal degree receives a join request, it first checks its closest neighbors in the direction of increasing node ID to see whether they have free slots for a link. This need not require extensive communication as neighbor information is available locally (and, for example, the lookahead mechanism described above also requires local neighborhood information).

*Stratification.* Since each node has only a small constant number of long-range links (1 or 2 in our case), many hops will follow the ring. It is therefore important that neighboring nodes in the ring have different long-range links. We propose a stratified sampling technique that involves dividing the long range links into a logarithmic number of intervals $[e^i/N, e^{i+1}/N]$ $(i = 0, \ldots, [\ln N] - 1)$. All the nodes first choose an interval at random that is not occupied by a long-range link at a neighbor, and then they draw a random ID from that interval with a probability proportional to $1/d$, where $d$ is the distance from the node's own ID.

*Short link avoidance.* Interestingly, if the average route is long, then it might be beneficial to exclude long-range links that are too short. This way we introduce some extra hops at the end of the route, when routing follows the ring only. However, we save hops during the first phases due to the longer long-range links. As we will see later, this technique works well only in very small degree networks where routes are long, but in such cases we can obtain a significant improvement. We implement short link avoidance based on the same intervals defined for stratification above. We introduce a parameter $m$: the number of shortest intervals that should be excluded when selecting long-range links. For example, for $m = 2$, the first possible interval will be $[e^2/N, e^3/N]$.

| | |
|---|---|
| network size | $2^i$, $i = 10, 11, \ldots, 20$ |
| maximal degree ($k$) | 3 or 4 (1 or 2 long-range links) |
| lookahead | 0, 1, or 2 hops |
| stratification | yes or no |
| join attempts | 1, 2, or 4 attempts |
| degree balancing | 1, 5, 10, or 20 neighbors checked |
| short link avoidance ($m$) | 0, 1, 2, 3, or 4 |

**Table 1.** The parameter space of the experiments.

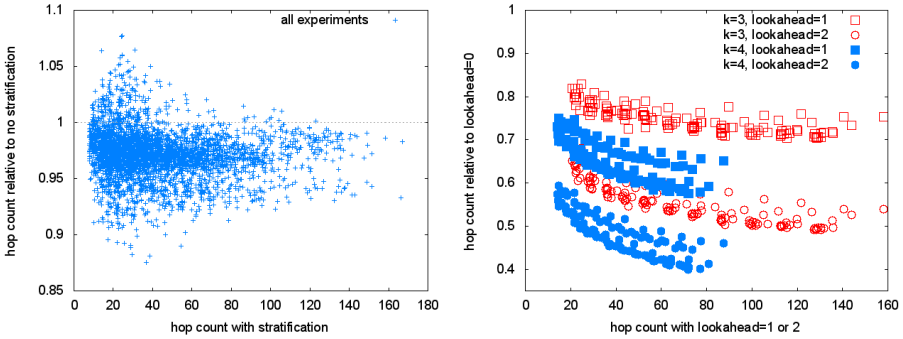


**Fig. 1.** The improvement in hop count as a result of stratification (left) and lookahead (right).

We performed experiments using the parameter space in Table 1. For all parameter combinations, we first constructed the network and subsequently we selected 10,000 random node pairs and recorded the hop count of the routing.

A main methodological tool we apply to evaluate the large parameter space is drawing scatter plots to illustrate the *improvement* in the hop count as a function of a varying parameter. In these plots the points correspond to different combinations of the possible values of a subset of parameters. The remaining free parameters are the ones we are interested in; they are used to calculate the coordinates of the points as follows. The hop count for a specified setting for the free parameters is the horizontal coordinate of a point, whereas the vertical coordinate is the ratio of the horizontal coordinate and the hop count that belongs to another (typically baseline) setting of the same parameters.

The improvement brought about by stratification is illustrated in Figure 1 (left). Clearly, for almost all parameter settings, stratification is a better choice (most values fall below 1). The experiments with values higher than 1 were performed on the smallest networks, with no apparent additional common features. The lack of improvement in these cases is most likely due to the larger noise of random sampling in smaller networks. From now on, we restrict our discussion to experiments with stratification.

The improvement brought about by lookahead is shown in Figure 1 (right). We can see that lookahead helps more if the degree of the network is larger. This is plausible

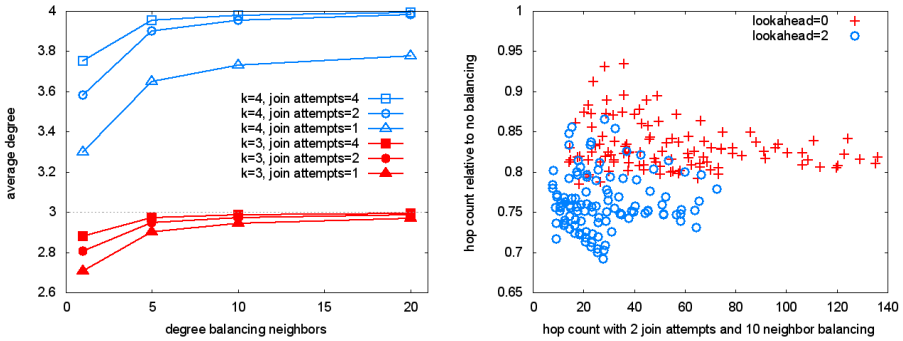


**Fig. 2.** Average degree for $N = 2^{20}$ (left) and the performance improvement achieved by a good balancing strategy (right). The average degree is practically identical with all the other network sizes too (not shown).

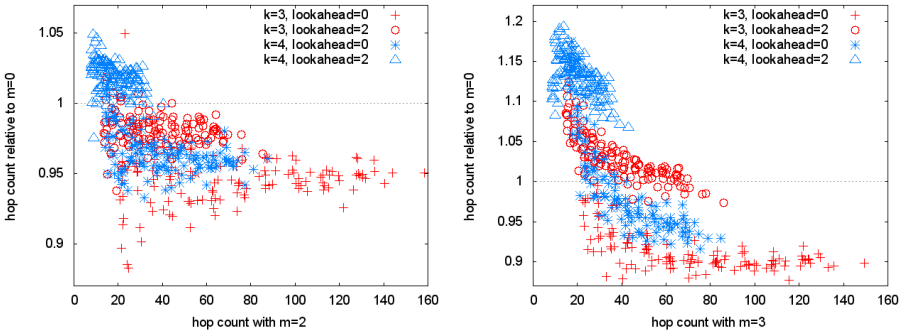


**Fig. 3.** The improvement in hop count as a result of setting $m \neq 0$.

since the local neighborhood is exponentially larger in a network of a larger degree. We also notice that lookahead is more useful in larger networks where the routes are longer.

Let us now have a look at the average degree of the networks (Figure 2). The main observation here is that it is important to approximate the maximal degree because in some cases we can observe a performance improvement of almost 30% relative to the baseline approach (that is, when no balancing efforts have been made), especially if lookahead has been applied as well. The setting of 2 join attempts with degree balancing over 10 neighbors appears to be a good compromise between cost and performance.

Figure 3 illustrates the effects of short link avoidance. When $m = 2$ (left), performance is improved with each parameter setting, except for $k = 4$ and $lookahead = 2$, where routing is so efficient that even for the largest networks there are too few hops, so short link avoidance does not result in a net gain in hop count. For $m = 3$ (right) the same effect is amplified: for parameter settings with a large hop count the relative improvement is larger, but for short routes the relative cost is larger as well. All in all, the effect of this technique depends on the other parameters, but $m = 2$ appears to be

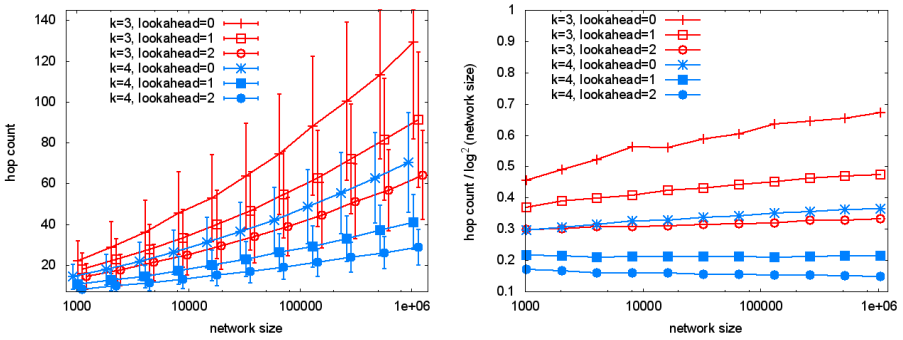


**Fig. 4.** Scalability of routing. Statistics over 10,000 random node pairs are shown. Stratified sampling was applied, along with 2 join attempts with a degree balancing over 10 neighbors, and we set $m = 2$.

rather robust and results in a slight improvement in most settings. We note that $m = 4$ was not the best setting in any of the experiments, so the maximal reasonable value was $m = 3$ in our parameter space.

Lastly, Figure 4 shows hop count as a function of network size. Theory predicts an $O(\log^2 N)$ hop count complexity; to a good approximation we can observe this scaling behavior, especially for $k = 4$. The very large difference between the best and the worst setting is also worth noting. Moving from $k = 3$ to $k = 4$ results in a very significant improvement: 2 long-range links instead of 1 causes the speed of routing to double, as predicted by theory [7].

We may conclude that routing in networks of a very small constant degree is feasible if certain techniques are applied. We found that the most effective technique is lookahead based on locally available information about the neighborhood of the nodes. In addition, degree balancing is very important as well. Further techniques such as short link avoidance and stratification also result in an additional 5-10% improvement, depending on the parameters. With these techniques we can route in around 30 hops in a network of size $N = 2^{20} \approx 1,000,000$ with a maximal node degree of only 4.

## 3 Scalability of Fault Tolerance

In the above sections we discussed several aspects of scalability. Our focus in this section will be whether the *fault tolerance* of the network diminishes as the network grows. This is crucial from the point of view of P2P networks (in particular, botnets), which have to tolerate a considerable node churn, as well as other types of failures.

We first touch on some important issues regarding the scalability of constant degree topologies, and then we propose the simple technique of using backup links to increase their fault tolerance. We present theoretical results to show that the proposed technique indeed turns constant degree networks scalable in a well defined sense in the presence of node failures.

We consider properties of networks of size $N$ as $N \to \infty$. This means that the results presented here are mainly of theoretical interest, since in practice an upper bound on network size can easily be given, and the algorithm designer can set protocol parameters according to the upper bound even if the algorithm is not scalable in the present sense. Still, our results are somewhat counter intuitive, and as such increase our insight into the behavior of constant degree networks.

The Achilles' heel of constant degree networks is fault tolerance and not performance. Performance is not a problem if the network is reliable. It is well-known that a constant number of neighbors is sufficient to build a connected structure. Not only the trivial constant degree topologies such as the ring or a tree are connected, but also there exist random topologies of constant degree such as the random $k$-out graphs. In such graphs each node is connected to $k$ random other nodes. It has been shown that for $k \geq 4$ a k-out graph is connected with high probability [14]. It is also well-known that a constant degree is sufficient for an efficient routing algorithm. In the Symphony network routing takes $O(\log^2 N)$ hops while in Viceroy, the optimal $O(\log N)$ hop-count is achieved [7, 8].

Unfortunately, constant degree networks do not tolerate node failure very well. We will examine the case when each node is removed with a fixed constant probability $q$ (that is, the expected number of nodes remaining in the network is $(1-q)N$). For example, the 4-out random graph is no longer connected with high probability in this model. In fact, in order to get a connected random topology in spite of node failures, one needs to maintain $O(\log N)$ neighbors at all nodes [10]. Similarly, it has been shown by Kong et al. [9] that—in this failure model—DHT routing is not scalable in Symphony, while it is scalable on other topologies that are able to find more alternative routes via maintaining $O(\log N)$ links at all the nodes. In the following, we summarize the results of Kong et al. for completeness and extend them to show how to achieve an effectively constant degree, yet scalable, topology.

Kong et al. examined the *success probability* of routing $p(h, q)$, the probability that in a DHT a node $h$ hops away from a starting node will be reached by the routing algorithm under a uniform node failure probability $q$ [9]. Their criterion for scalability is

$$\lim_{N\to\infty} p(h,q) = \lim_{h\to\infty} p(h,q) > 0, \quad 0 < q < 1 - \epsilon, \tag{1}$$

where $\epsilon > 0$, and $h$ is the average routing distance in the topology under study ($h = O(\log^2 N)$ for Symphony). This expresses the requirement that increasing network size should not increase sensitivity to failure without limit. Given this criterion, the proposed methodology consists of finding the exact formula or a lower bound for $p(h, q)$ for a topology of interest, and then calculating the limit to see whether it is positive. To calculate $p(h, q)$, one can create a Markov chain model of the routing process under failure, and determine the probability of reaching the failure state.

Kong et al. proved that Symphony is not scalable. They showed that for each step the probability of failure is a constant ($C$), so

$$\lim_{h\to\infty} p(h,q) = \lim_{h\to\infty} (1-C)^h = 0. \tag{2}$$

However, if we assume that there are *backup* links for each link in Symphony, then the situation changes dramatically. We do not go into detail here about how to collect the backup links; Section 4 discusses an actual algorithm. From our point of view here the important fact is that the backup links are such that if a link is not accessible, then the first backup is the best candidate to replace it. If the first backup is down as well, then the second backup is the best replacement, and so on.

Recall that the Symphony topology consists of a ring and a constant number of shortcuts. For the ring, the notion of backup should be clear. A shortcut link is defined by a randomly generated ID: we need to find the numerically closest node in the network to that ID. The first backup in that case is the second closest node in the network, and so on. This notion can be extended to all routing geometries as well.

The backup links do not increase the *effective* degree of an overlay node: a DHT can use the original links if they are available, even if some of the backups were closer to the target. In fact, backup links are *never used* for communication, not even during maintenance or any other function, except when they become regular links after replacing a failed regular link. In addition, as we explain in Section 4, backup links can be collected and updated during regular DHT maintenance without any extra messages.

It seems clear that backup links can turn Symphony scalable. However, the question is how many of them do we need? In the following we show that $O(\log N)$ backup links are sufficient, and under some circumstance even $O(\log \log N)$ links will do.

**Lemma 1.** *If in a DHT routing network all the links have $f(N)$ backup links then $p(h,q) \geq (1-q^{f(N)})^h$.*

*Proof.* The probability of being able to use the best link in the original overlay is $1-q$. Considering the backups this probability becomes $1-q^{f(N)+1} > 1-q^{f(N)}$. Now, if we follow only the optimal link in each step then the probability of success is not smaller than $(1-q^{f(N)})^h$. Clearly, $p(h,q)$ is no less than this value since it accounts for methods for routing around failed links as well.

**Lemma 2.** $\lim_{N\to\infty}(1-q^{\log N})^{\log^k N} > 0$ *if* $0 \leq q < 1-\epsilon$ *and* $k \in \mathbb{R}$.

*Proof.* For $k <= 0$ the lemma is trivial. For $k > 0$, according to Theorem 1 in [9] we need to prove that

$$\lim_{N\to\infty} q^{\log N} \log^k N < \infty$$

and the lemma follows. The convergence of the above expression can be proven by applying the l'Hospital rule on $(\log^k N)/q^{-\log N}$ a suitable number of times.

**Lemma 3.** $\lim_{N\to\infty}(1-q^{\log\log N})^{\log^k N} > 0$ *if* $0 \leq q < \min(e^{-k}, 1-\epsilon)$.

*Proof.* We again need to prove that

$$\lim_{N\to\infty} q^{\log\log N} \log^k N < \infty.$$

Substituting $x = \log N$ we get

$$q^{\log x} x^k = x^{\frac{1}{\log_q e}} x^k = x^{\frac{1}{\log_q e}+k}$$

This means that we need $\frac{1}{\log_q e} + k \leq 0$ for convergence. Elementary transformations complete the proof.

**Theorem 1.** *The Symphony topology is scalable, that is,* $\lim_{h\to\infty} p(h,q) > 0$*, if (i) all the links have* $O(\log N)$ *backup links, or if (ii) all the links have* $O(\log\log N)$ *backup links and* $q \leq e^{-2} \approx 0.135$*.*

*Proof.* Straightforward application of the previous lemmas for Symphony where $k = 2$, that is, $h = O(\log^2 N)$.

To sum up, we have shown that Symphony-like topologies can be made scalable by adding only $O(\log N)$ backup links for all the links, and under moderate failure rates even $O(\log\log N)$ suffices. This is rather counter-intuitive given that routing still takes $O(\log^2 N)$ steps. It is also promising, because these results suggest that collecting good quality backup links can dramatically improve scalability at a low cost.

## 4 Experimental Results

In this section we present proof-of-principle experiments with a simple implementation of a small constant degree network in realistic churn scenarios. Our goal is not to present a complete optimized implementation but rather to show that it is indeed possible to achieve acceptable fault tolerance and performance in realistic environments.

We performed the experiments using the PeerSim event-based simulator [15]. In our system model nodes can send messages to each other based on a node address. Nodes have access to a local clock, but these clocks are not synchronized. Messages can be delayed and nodes can leave or join the system at any time. The statistical model of node churn is based on measurement data [16], as we describe later.

Our goal was to design a protocol to construct and maintain a Symphony topology in a fault tolerant way, with backup links (see Section 3). To this end, we applied T-Man, a generic protocol for constructing a wide range of overlay topologies [17]. Here we briefly outline the protocol and the specific details of the present implementation. The reader is kindly requested to consult [17] for more information.

In our experiments each node has a single long-range link; that is, the maximal effective degree is 3. Each node has three local caches: long-range backups, ring backups and random samples. We set a maximal size of 80 for both the long-range and ring backup caches, and 100 for random samples. These values were chosen in an ad hoc way and were not optimized.

The caches contain node descriptors that include the ID and the address of a node. Each node periodically sends the contents of all its caches to all its neighbors. The period of this communication is called the *gossip cycle*, and was set to 1 minute in our experiments.

As described in Section 3, the two backup caches should ideally contain those nodes from the entire network whose IDs are closest to the node's own ID (for the ring neighbors), and the ID of the long-range link, respectively. When receiving a message containing node descriptors, a node updates its own local caches. It also updates the random sample cache, using a stratified sampling approach: the ID space is divided into

100 equal intervals, and each cache entry is selected from one of these intervals. If the random sample cache can be improved using any of the incoming node descriptors, the cache is updated.

In addition, if a node receives a message from a node it should not receive messages from (for example, because the sender has inaccurate knowledge about the topology) the node sends its caches to the sender of the misdirected message as well, so that it can improve its backup caches.

The join procedure starts by generating the node's own ID at random, as well as the ID for the long-range link. The caches need to be initialized as well, using a set of known peers; we applied 50 fixed descriptors for the initialization. Once the caches contain at least one link, the gossip protocol sketched above can start, and all the caches will fill and improve gradually.

When handling a routing request, a node applies greedy routing using the three links: the two ring links and the long-range link. However, before using the currently active ring links or long-range link, the node always checks the best candidate in the backup caches for availability (sending a ping message). Note that we do not check the best candidate for the message to be routed; we check the best candidate for the given link slot (ring or long-range). This way, it is guaranteed that the right links are used based on the current state of the network at any given time.

The scenario we experimented with involves node churn. Applying appropriate models of churn is of crucial importance from a methodological point of view. Researchers have often applied an exponential distribution to model uptime distribution, which corresponds to a failure probability independent of uptime. Measurements of a wide range of P2P networks in [16] suggest that a Weibull distribution of uptime is more realistic, with a shape parameter around $k = 0.5$. In this case the failure rate decreases, that is, the more time a node spends online, the less likely it is to fail. This favors longer sessions, but the Weibull distribution is nevertheless not heavy-tailed.

We applied the Weibull distribution with $k = 0.5$ to model uptime, and scaled the distribution, so that around 30% of the nodes live longer than 30 minutes [16]. The downtime distribution was modeled by a uniform random distribution, with an average downtime of 2 minutes. This average is very short; however, longer downtimes result in a relative increase in the proportion of nodes in the network that have long session lengths. Paradoxically, if the downtime is long, then the network is almost completely stable in the time range we are interested in (around 30 minutes).

As noted in [16], the lengths of the online sessions of a node correlate: there are nodes that tend to be available and nodes that are not. We assigned each node a fixed session length from the distribution above, that remained fixed during the experiment.

We applied a 1 minute gossip cycle. Each experiment lasted for 40 cycles. The network gradually grew to its final size during the first 10 cycles, when we added each node at a random time. During the remaining 30 cycles churn was applied. The network sizes we tested were $N = 2^i$, $i = 10, \ldots, 14$. Parameter $m$ (which controls short link avoidance) was set to $m = 0$ or $m = 3$. Other features such as lookahead, stratification, and degree balancing were not implemented at the time of writing.

Figure 5 illustrates the speed at which the ring topology is being formed despite of the continuous churn. The improvement of the backup links (80 links per node) for the

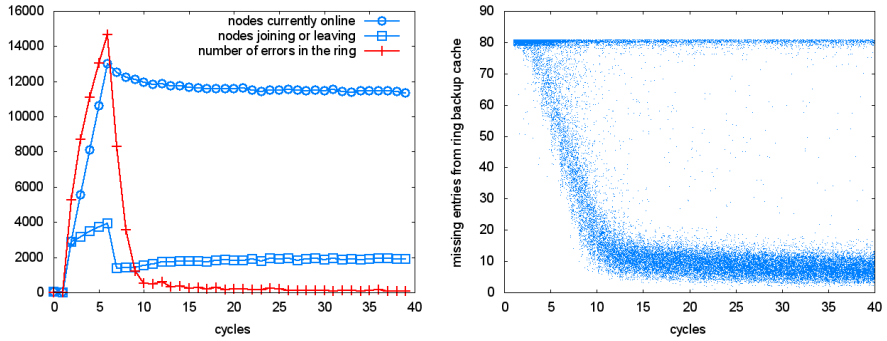


**Fig. 5.** The evolution of the topology and the backup links for $N = 2^{14}$. In the figure on the right points belong to individual nodes and are randomly shifted so as to visualize the density.

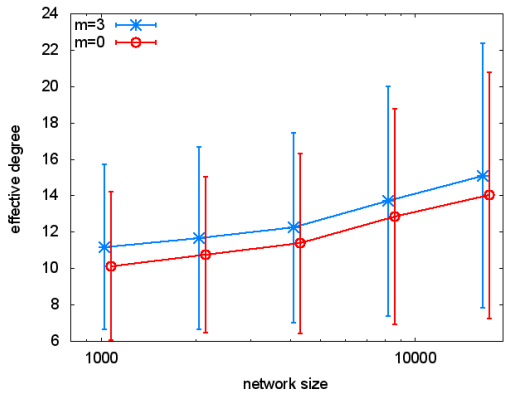


**Fig. 6.** The observed effective degree by the end of the experiments.

ring links is also illustrated. It can be seen that most nodes collect good quality backups, but some of them seem to have no usable backups at all; these nodes have a very short session time and spend very little time in the network.

One of our main goals was to show that the effective degree—the number of nodes an average node actually communicates with—can be kept low. Figure 6 shows that indeed this can be accomplished. Despite heavy churn, which results in a constant fluctuation of the ring neighbors, the effective degree is small and seems to scale well. Recall that, for example, the Storm worm has been observed to communicate with thousands of neighbors [2].

Finally, let us examine the reliability and the efficiency of routing (see Figure 7). Recall that we work with a baseline implementation with no lookahead, stratification, or any other techniques. Only short link avoidance is implemented. When testing routing we pick IDs and not nodes as targets, and consider routing successful if the closest node

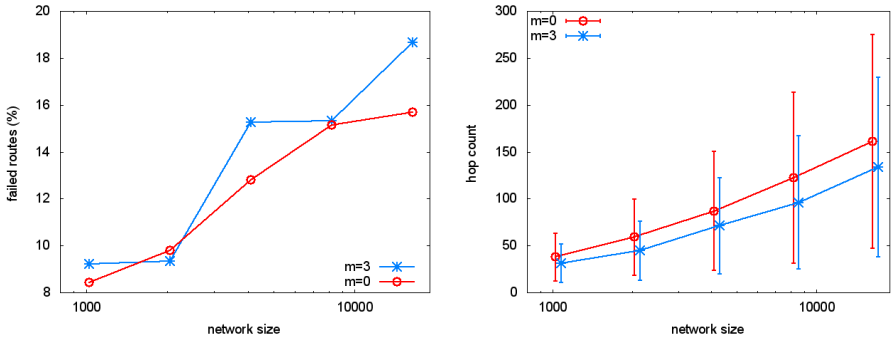


**Fig. 7.** Routing performance. The figure on the right corresponds to successful routes.

receives the message at the time of reception. That is, it is possible that the optimal target is different at the start of the routing and at the end of the same routing.

We can see that short link avoidance improves the hop count by a large margin. Overall, we observe almost twice the hop count as in the ideal case shown in Figure 4. However, in our hostile scenario with heavy churn this can be considered acceptable for this baseline approach. We also note that the actual routing performance observed in real botnets can be significantly worse; for example, the success rate of queries has been found to be extremely low in the Storm botnet [2].

## 5 Conclusions

In this paper we argued for the feasibility of P2P systems that operate in a stealth mode, where nodes communicate only with a very limited number of peers during their lifetime.

Our results have at least two implications. First, they are a strong indication that P2P botnets need to be taken seriously by the P2P community. In our previous work we showed that stealth mode P2P networks are practically invisible for state-of-the art methods for P2P network detection [6]. Current botnets do not exploit P2P technology to its full potential, and by the time they learn how to do that, they will be very difficult to detect and remove.

The second implication is not related to malware. There can be other applications where it is important to utilize very few connections because of a large associated cost. Detailed arguments for a constant degree design can be found in related works as well [8]. For this reason, the research issue that we raised, that is, the investigation of networks of a very small constant degree, is relevant to non-malicious applications as well.

## References

1. Holz, T., Steiner, M., Dahl, F., Biersack, E., Freiling, F.: Measurements and mitigation of peer-to-peer-based botnets: a case study on storm worm. In: Proceedings of the 1st USENIX

Workshop on Large-Scale Exploits and Emergent Threats (LEET'08), Berkeley, CA, USA, USENIX Association (2008)
2. Grizzard, J., Sharma, V., Nunnery, C., Kang, B., Dagon, D.: Peer-to-peer botnets: Overview and case study. In: Proceedings of the First USENIX Workshop on Hot Topics in Understanding Botnets (HotBots'07). (2007)
3. Porras, P., Saïdi, H., Yegneswaran, V.: A foray into Conficker's logic and rendezvous points. In: 2nd USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET'09), USENIX (2009)
4. Iliofotou, M., Pappu, P., Faloutsos, M., Mitzenmacher, M., Singh, S., Varghese, G.: Network monitoring using traffic dispersion graphs (TDGs). In: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement (IMC'07), New York, NY, USA, ACM (2007) 315–320
5. Stern, H.: Effective malware: The trade-off between size and stealth. In: 2nd USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET'09), USENIX (2009) invited talk.
6. Jelasity, M., Bilicki, V.: Towards automated detection of peer-to-peer botnets: On the limits of local approaches. In: 2nd USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET'09), USENIX (2009) `http://www.usenix.org/events/leet09/tech/`.
7. Manku, G.S., Bawa, M., Raghavan, P.: Symphony: Distributed hashing in a small world. In: Proceedings of the 4th USENIX Symposium on Internet Technologies and Systems (USITS'03). (2003)
8. Malkhi, D., Naor, M., Ratajczak, D.: Viceroy: A scalable and dynamic emulation of the butterfly. In: Proceedings of the 21st ACM Symposium on Principles of Distributed Computing (PODC'02), New York, NY, USA, ACM (2002) 183–192
9. Kong, J.S., Bridgewater, J.S.A., Roychowdhury, V.P.: A general framework for scalability and performance analysis of DHT routing systems. In: Proceedings of the International Conference on Dependable Systems and Networks (DSN'06), Washington, DC, USA, IEEE Computer Society (2006) 343–354
10. Kermarrec, A.M., Massoulié, L., Ganesh, A.J.: Probablistic reliable dissemination in large-scale systems. IEEE Transactions on Parallel and Distributed Systems **14**(3) (March 2003) 248–258
11. Kleinberg, J.: The small-world phenomenon: an algorithmic perspective. In: Proceedings of the 32nd ACM Symposium on Theory of Computing (STOC'00), New York, NY, USA, ACM (2000) 163–170
12. Manku, G.S., Naor, M., Wieder, U.: Know thy neighbor's neighbor: the power of lookahead in randomized p2p networks. In: Proceedings of the 36th ACM Symposium on Theory of Computing (STOC'04), New York, NY, USA, ACM (2004) 54–63
13. Naor, M., Wieder, U.: Know thy neighbor's neighbor: Better routing for skip-graphs and small worlds. In: Peer-to-Peer Systems III. Volume 3279 of Lecture Notes in Computer Science., Springer (2005) 269–277
14. Cooper, C., Frieze, A.: Hamilton cycles in random graphs and directed graphs. Random Structures and Algorithms **16**(4) (2000) 369–401
15. PeerSim: http://peersim.sourceforge.net/
16. Stutzbach, D., Rejaie, R.: Understanding churn in peer-to-peer networks. In: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement (IMC'06), New York, NY, USA, ACM (2006) 189–202
17. Jelasity, M., Montresor, A., Babaoglu, O.: T-Man: Gossip-based fast overlay topology construction. Computer Networks **53**(13) (2009) 2321–2339