Quasirandom Rumor Spreading: An Experimental Analysis*

Benjamin Doerr† Tobias Friedrich‡ Marvin Künnemann§ Thomas Sauerwald‡

Abstract

We empirically analyze two versions of the well-known "randomized rumor spreading" protocol to disseminate a piece of information in networks. In the classical model, in each round each informed node informs a random neighbor. At SODA 2008, three of the authors proposed a quasirandom variant. Here, each node has a (cyclic) list of its neighbors. Once informed, it starts at a random position of the list, but from then on informs its neighbors in the order of the list.

While for sparse random graphs a better performance of the quasirandom model could be proven, all other results show that, independent of the structure of the lists, the same asymptotic performance guarantees hold as for the classical model.

In this work, we compare the two models experimentally. This not only shows that the quasirandom model generally is faster (which was expected, though maybe not to this extent), but also that the runtime is more concentrated around the mean value (which is surprising given that much fewer random bits are used in the quasirandom process).

These advantages are also observed in a lossy communication model, where each transmission does not reach its target with a certain probability, and in an asynchronous model, where nodes send at random times drawn from an exponential distribution. We also show that the particular structure of the lists has little influence on the efficiency. In particular, there is no problem if all nodes use an identical order to inform their neighbors.

1 Introduction

We conduct an experimental analysis of *randomized rumor spreading* protocols (also known as *random broadcast* [8] or the *push model* [3, 11]). Starting from one node of a graph having a piece of information ("rumor") unknown to the other nodes, rumor spreading protocols aim at efficiently making this information known to all nodes. They typically proceed in rounds. In the classical model, in each round, each node that already knows the piece of information transmits it to a neighbor chosen uniformly at random. If the rumor was yet unknown to the neighbor, it becomes informed.

Though not a very elaborate protocol, it works very efficiently on many graph classes. For complete graphs, hypercubes and sufficiently dense random graphs, only $\Theta(\log n)$ rounds are required to inform all nodes (where n is the number of nodes of the graph) [8, 10]. We refer to Section 2 for a precise statement of the model and the results.

Recently, three of the authors introduced a quasirandom variant of the randomized rumor spreading model [5]. Here, each node is equipped with a cyclic list of its neighbors. Once informed, each node chooses a single random neighbor. Its first transmission is directed to that neighbor, the next to its successor in the list, and so on. Hence all transmissions of this node are determined by the first, random one. It was then shown that, independent of the choice of the lists, this protocol also needs only $\Theta(\log n)$ rounds for complete graphs, hypercubes and sufficiently dense random graphs.

While it could be shown that the quasirandom model has a superior asymptotic runtime on a few graph classes like dense complete trees and sparse random graphs, unfortunately all other existing results only succeed in showing that the runtime of the quasirandom model for complete graphs, hypercubes and not too sparse random graphs is of the same order as the runtime of the fully random model. The obvious reason for this is that the current theoretical methods are mostly too coarse to make constant factors precise, let alone lower order terms. Since these are relevant in a practical application, we use experimental analyzes to obtain non-asymptotic results for concrete graphs. They yield the following results.

We shall see, as expected, that the quasirandom model is generally faster. For sparser graphs, which also represent the practically more relevant setting, savings typically are more than ten percent, which is more than what we expected.

What is more surprising is that the quasirandom model also is more robust in several respects. This was not expected due to the fact that a single random choice has a larger influence than in the fully random model. However, we see that the deviation of the actual

*Benjamin Doerr was partially supported by the German Research Foundation's (DFG) priority programme SPP 1307. Tobias Friedrich and Thomas Sauerwald were partially supported by postdoctoral fellowships from the German Academic Exchange Service (DAAD). This work was initiated while Tobias Friedrich was with the Max-Planck-Institut für Informatik and Thomas Sauerwald was with the Department of Computer Science at the University Paderborn. Marvin Künnemann was supported by an internship from the Max-Planck-Institut für Informatik.

†Max-Planck-Institut für Informatik, Campus E1 4, 66123 Saarbrücken, Germany

‡International Computer Science Institute, 1947 Center St., Suite 600, 94704 Berkeley, CA, USA

§Universität des Saarlandes, Fachrichtung Informatik, Postfach 151150, 66041 Saarbrücken, Germany

Graph class	Broadcast times
Bounded-degree graphs	$\mathcal{R}(G) = \Theta(\text{diam}(G))$ w. h. p. [8] $\mathcal{Q}(G) = \Theta(\text{diam}(G))$ w. p. 1 [5]
Hypercubes, complete graphs, and almost all random graphs $\mathcal{G}(n,p)$ with $p \geq (1+\varepsilon)\ln(n)/n$, $\varepsilon > 0$	$\mathcal{R}(G) = \Theta(\log n)$ w. h. p. [8] $\mathcal{Q}(G) = \Theta(\log n)$ w. h. p. [5]
Almost all random graphs $\mathcal{G}(n,p)$ with $p = (\ln n + f(n))/n$, where $f(n) \to \infty$ and $f(n) = \mathcal{O}(\log\log n)$	$\mathcal{R}(G) = \Theta(\log^2 n)$ w. h. p. [8] $\mathcal{Q}(G) = \Theta(\log n)$ w. h. p. [5]

Table 1: Broadcast times for different graphs G in the random ($\mathcal{R}(G)$) and quasirandom ($\mathcal{Q}(G)$) model.

broadcast time from the expected value is much smaller in the quasirandom model. For the sparse random graph $G(n,p)$ with $n = 2^{12}$ and $p = \ln(n)/n$, conditional on that it is connected, we observe that in 10% of all runs the fully random model needs more than 13 rounds more than the mean value. However, in 99% of all runs the quasirandom model terminates in less 6 rounds more than the mean (cf. Figure 2(d)).

We also observe robustness against transmission failures. This is again a well-known strength of the fully random model, cf. [6, 8, 11], and one where again the reduced amount of randomness could lead to inferior results for the quasirandom model. However, even in the setting where each transmission has a 50% chance of not reaching the addressee (without notice to the sender) the quasirandom model keeps its lead. For the hypercube with 2^{12} vertices, the average broadcast time is 40.41 in the quasirandom and 45.53 in the random model.

We also discuss the question if the order of the list has an influence on the quality of the protocol. The good news given by the theoretical results is that no such choice can lead to a real failure, that is, for the graph classes discussed so far the difference can at most be a large constant factor. Our simulations indicate that there are differences, but they are small. This is again good news from the view-point of application, since one advantage of the quasirandom model is that one can use an implicitly given list (which should be present at any node to have some means of addressing neighbors).

The remainder of this paper is organized as follows. We first describe the two broadcasting models in detail and graphs classes used in this investigation in Section 2. In Section 3 we compare the average runtimes and their concentrations around the means. In Section 4 we provide an explanation for the good performance of the quasirandom model for sparse graphs. The influence of the lists is discussed in Section 5. In Sections 6 and 7 we show that the advantages observed so far for the quasirandom model also hold in the presence of transmission failures and in an asynchronous model.

2 Preliminaries

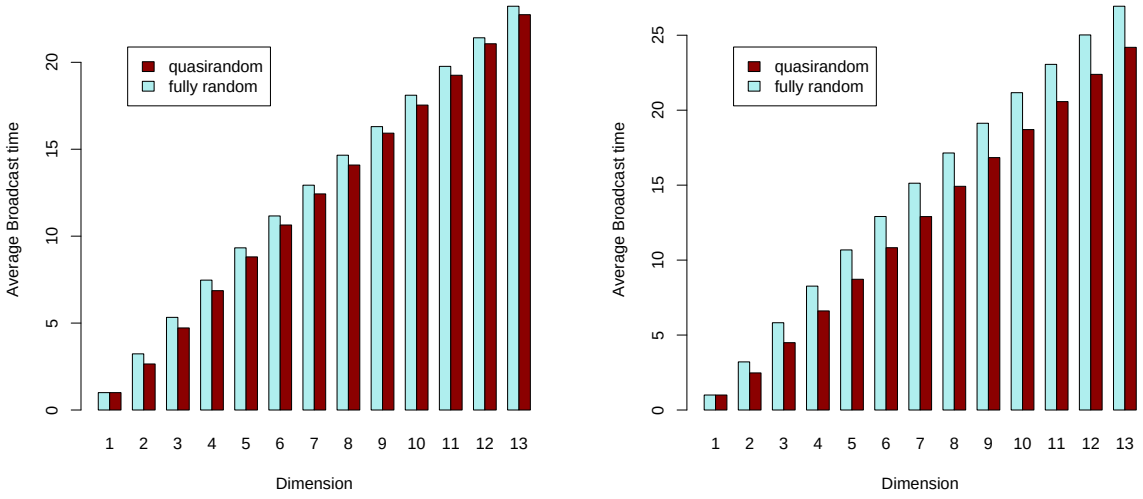
2.1 Broadcasting models. One of the simplest broadcasting protocol is the so-called push model, which we shall also call *fully random model* or simply *random model*). There, initially only one node of a graph $G = (V, E)$ owns a piece of information (or equivalently, knows a rumor) which is spread iteratively to all other nodes: in each time-step $t = 1, 2, \ldots$ every *informed* node chooses a neighbor uniformly at random, which the piece of information is sent to. We are interested in how many time-steps are required such that all nodes become informed.

We compare this random broadcasting model with its quasirandom analogue introduced in [5]. In the quasirandom model, each node has a cylic list of its neighbors and informs the neighbors in the order of the list. The order of the lists can be arbitrary, but the starting point of each list is assumed to be random. For simplicity, we assume that a node does not stop sending the rumor, even if the list has been completed.

To describe the previous results in this section, we will use the following notation. Given a graph $G = (V, E)$, the number of rounds (or time-steps) of a broadcasting procedure until the rumor reaches all the vertices of G is a random variable that depends on the topology of G. Let $\mathcal{R}(G)$ be the number of rounds of the random broadcasting model until all vertices in G receive the rumor for all starting vertices. Analogously, let $\mathcal{Q}(G)$ be the maximal number of rounds of the quasirandom broadcasting model until all vertices in G receive the rumor for all starting vertices and all possible lists.

2.2 Graphs and related work. We shall investigate the broadcast times on the following topologies.

The *complete graph* K_n with n vertices is the graph where every pair of distinct vertices is adjacent. It was shown by Pittel [14] that $\mathcal{R}(K_n)$ tends to $\log_2 n + \ln n +$



(a) Complete graph

(b) Hypercube

Figure 1: Average broadcast times for complete graphs and hypercubes with $n = 2^1, \ldots, 2^{13}$ nodes.

$o(\log n)$ w. h. p.[1]. For the quasirandom model it has been shown that $\mathcal{Q}(K_n)$ is of order $\Theta(\log n)$ [5]. This result was recently improved to $\log_2 n + \ln n + o(\log n)$ w. h. p. [4].

The d-dimensional *hypercube* H_d, is the graph defined by $V = \{\{0,1\}^d\}$ and $E = \{\{u, u(i)\} \in \{0,1\} \mid u \in V\}$, where $u(i)$ is the bit-vector obtained by flipping the ith bit of u. Feige et al. [8] proved that $\mathcal{R}(H_d) = \Theta(\log n)$ w. h. p. Also this result was extended to the quasirandom model $\mathcal{Q}(H_d)$ by [5].

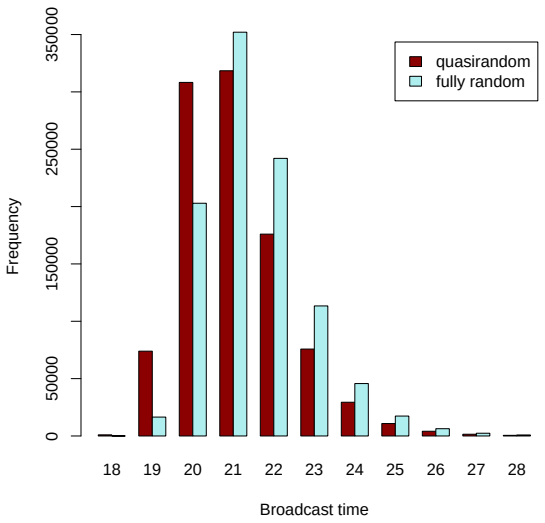
The classical random graph model introduced by Erdős and Rényi [7] is defined as follows. Given a value $p \geq \ln(n)/n$, pick an edge between each pair of vertices independently with probability p. By [7], such a random graph is connected with constant probability. The exact results for random graphs can be found in Table 1. Roughly speaking, for sparse random graphs, the quasirandom model outperforms its random counterpart by a logarithmic factor, while for more dense random graphs, both models have an optimal runtime of $\Theta(\log n)$.

We also examine a certain model of random regular graphs. For sampling from the space of all k-regular graphs randomly, we add edges between two random nodes of degree smaller than k until this is not possible anymore. Either we have reached a regular graph, or we redo the process from scratch.

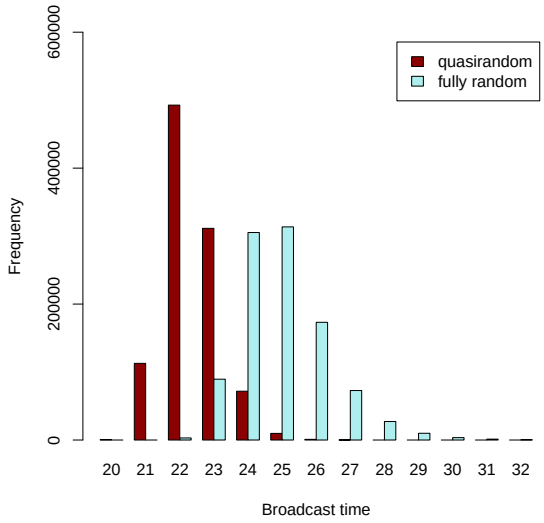
2.3 Experimental setup. We use simulations to obtain empirical estimates on the expected broadcast time and on the variance. Unless otherwise stated, we use the following lists also called canonical lists. For complete graphs, every vertex has the same list of neighbors $(1, 2, \ldots, n)$. That means after starting at a random integer, each vertex contacts its neighbors in increasingly order (modulo n), counting only steps where the vertex has been already informed. On hypercubes with dimension d, we set the list of each vertex u to $(u(1), u(2), \ldots, u(d))$. Since random graphs lack a particular structure, we only consider independent and randomly permuted lists for each vertex.

For each specific graph and list, we have performed at least 100,000 experiments to obtain accurate estimates of the expected broadcast time and the corresponding variance. Each time, the starting vertex s is chosen uniformly at random (unless the graph is vertex-symmetric). Additionally, for random graphs, the random and quasirandom model are tested on the same samples. A new sample of the random graph is generated every thousand run. For the case that the generated random graph is not connected, the graph is discarded and a new one is generated.

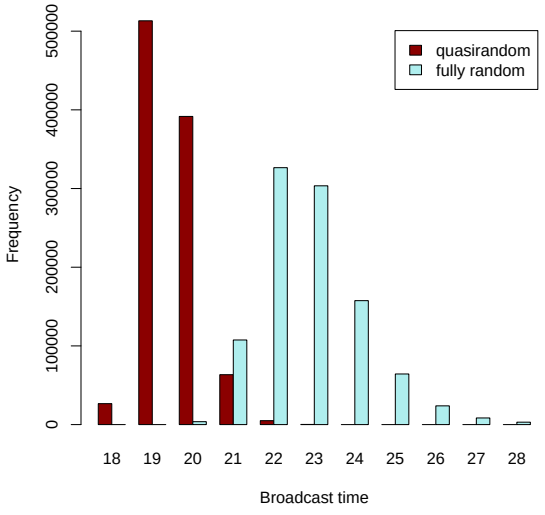
[1] w. h. p. stands for "with high probability", which refers to an event which holds with probability at least $1 - o(1)$.



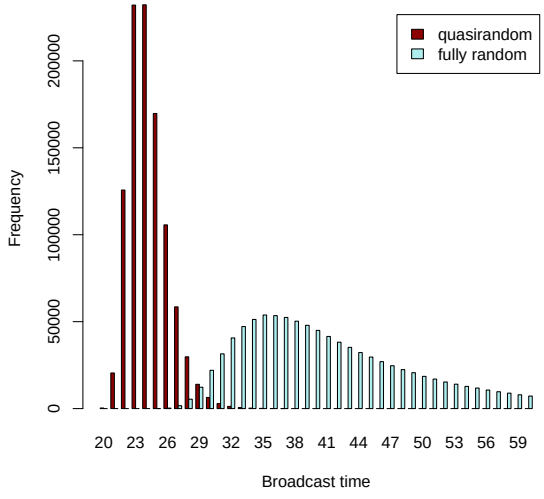
(a) Complete graph $K_{2^{12}}$



(b) Hypercube H_{12}



(c) Random 12-regular graphs



(d) Random graphs with $p = \ln(n)/n$

Figure 2: Empirical distribution of broadcast times on four different graphs with $n = 2^{12}$ nodes.

3 Broadcast times

A clear advantage of the quasirandom model is that no node will inform a neighbor a second time (unless it already has informed all its neighbors). This should make the quasirandom model faster, to a stronger extent for sparser graphs than for dense ones. Indeed, as Table 2 shows, we observe considerable gains for sparse graphs and still 2% for the complete graph with 2^{12} vertices. Taking into account that informing a neighbor twice does not happen very often in the relatively short runtimes, these gains are more than what we expected.

The averages and standard deviations are given in Table 2. For each graph, we have performed 1,000,000 runs and we used the canonical list as described in Section 2.3 (other lists are examined in Section 5).

We have already argued why it makes sense that the advantage of the quasirandom model is larger if the vertex degrees are small. This not only holds for the mean broadcast times, but also for median and other quantiles. E.g., for the hypercube H_{12}, also median, 75%, 90%, 95%, and 99% quantile are smaller by 2 for the quasirandom model.

To discuss the results on random graphs, recall that on complete k-ary trees the quasirandom protocol is faster by a factor of $\Theta(\log k)$ [5]. Since random regular graphs and sparse random graphs are locally tree-like (i.e., they have no small cycles [2]), this could explain the 15% advantage of the quasirandom model for random 12-regular graphs.

The highest difference between both models in Table 2 is attained for random graphs $G(n,p)$ with $p = \ln(n)/n$, which is precisely the connectivity threshold for random graphs [7]. We only use the sampled graph if it is connected. We observe empirically a large advantage of 44% for these graphs. In Section 4 we will explain how the large number of small vertices slows down the fully random broadcast on these graphs.

Figure 1 shows the difference between the quasirandom and the random model for different graph sizes, where for each size we have conducted 100,000 iterations. For complete graphs, one can see a small but stable additive gap between the sampled expected broadcast times. For hypercubes, this gap seems to be increasing with the dimension.

Finally, we briefly discuss the concentration of the runtime distribution. From a theoretical point of view, most papers usually prove tight bounds on the runtime that hold with high probability [4–6, 8, 10, 11, 14]. Though this implies bounds on the expected runtime, it does not give much further insight on the runtime distribution such as the variance or higher moments.

Additional to the average runtimes, Table 2 gives the empirical standard deviations. For all four examined graph classes, the deviations are higher for the random model. In view of the less randomness used by the quasirandom model, this is rather surprising.

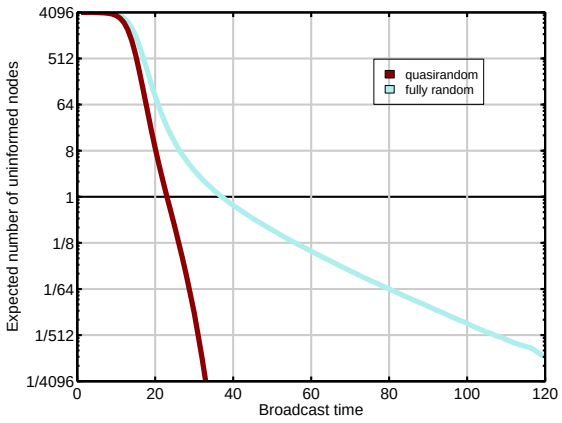


Figure 3: Decrease of the expected number of uninformed nodes during the two broadcasting processes on sparse random graphs with $n = 2^{12}$ nodes and $p = \ln(n)/n$. (Note that the y-axis is scaled logarithmically.)

As for the averages, the strongest difference occurs for sparse random graphs. There, the standard deviation drops by more than a factor of six. Accordingly, the histogram in Figure 2(d) shows an extremely heavy tail of the fully random broadcast time compared to the quasirandom broadcast time.

4 Influence of the degree distribution

In Section 3 we observed that a small minimum degree favored the quasirandom model in comparison to the random one. We now want to further support this explanation.

The largest difference between both models so far is attained for sparse random graphs according to Table 2. From a theoretical point of view, one can show that they have $\omega(1)$ small vertices, i.e., vertices of constant degree. As shown by [2], such small vertices have distance at least $\Theta(\log n/\log\log n)$ and each neighbor of a small vertex has a degree of $\Theta(\log n)$.

Let us now consider the quasirandom model. By its definition, every small vertex with at least one informed neighbor gets informed within $\text{maxdeg}(G) = \mathcal{O}(\log n)$ further steps w. h. p. On the other hand, consider the random model in a situation where all small vertices have only informed neighbors. Since all these neighbors have degree $\Theta(\log n)$, the expected time to inform a fixed small vertex is $\Theta(\log n)$ and the expected time to inform all $\omega(1)$ small vertices is $\omega(\log n)$. This hints for an asymptotic superiority of the quasirandom model.

On a more concrete level, let us reconsider the

	Random broadcast	Quasirandom broadcast
Complete graph $K_{2^{12}}$	21.50 ± 1.32	21.04 ± 1.32 (=2.1% faster)
Hypercube H_{12}	24.98 ± 1.32	22.37 ± 0.82 (=10.4% faster)
Random 12-regular graphs	22.87 ± 1.30	19.51 ± 0.68 (=14.7% faster)
Random graphs $G(n,p)$ with $p = \ln(n)/n$	43.20 ± 11.8	24.28 ± 1.83 (=43.8% faster)

Table 2: Averages and standard deviations of the broadcast times for different graphs with $n = 2^{12}$ vertices.

	Random broadcast	Quasirandom broadcast
Random graphs with $p = \ln(n)/n$	43.20 ± 11.8	24.28 ± 1.83 (=43.8% faster)
Random graphs with $p = 2\ln(n)/n$	26.78 ± 3.55	22.12 ± 1.42 (=17.4% faster)

Table 3: Averages and standard deviations of the broadcast times for different random graphs with $n = 2^{12}$ vertices.

sparse random graph with $n = 2^{12}$ at the connectivity threshold $p = \ln(n)/n$. There, the expected number of nodes with degree smaller than five is $\sum_{k=0}^{4} \binom{n}{k} p^k (1-p)^{n-k} n \approx 338$ and with good probability some of them will have only neighbors with large degree. To inform these, the random protocol has to spend significantly more than the quasirandom protocol. This is illustrated in Figure 3 where one can observe that at the beginning both models do not deviate much. In fact, on average more than 90% of the nodes get informed after 16 or 18 steps in the quasirandom resp. random model. On the other hand, within the 100,000 runs of Figure 3 it took never more than 37 rounds to inform the very last node (typically of very small degree) in the quasirandom model while it took up to 228 rounds for the random model.

This looks much different for denser random graphs. For $p = 2\ln(n)/n$, the expected number of nodes with degree smaller than five is smaller than one. Hence the degree distribution differs significantly from the sparser case and the aforementioned effect diminishes as can be seen in Table 3.

5 Influence of the lists

So far in our experiments we only regarded the canonical choice introduced in Section 2 for the cyclic lists describing the orders in which nodes contact their neighbors. Recall that existing theoretical work applies to all kinds of lists. Nevertheless, one might speculate that some orders are more efficient than others, and one might wonder if it is good or not if all nodes use the same lists (excluding of course the node itself as addressee). The results obtained in this section will show that there is some influence, but it is not very large.

A natural candidate different from the canonical choices and suitable for all graphs are random lists. They are interesting in that they demonstrate what would happen if in the fully random model we were picking the new addressee uniformly at random only from the nodes not contacted so far by the node under consideration. For a practical application, this model suffers slightly from its reduced simplicity.

An interesting idea for network topologies containing some geometric flavor are low-discrepancy approaches. See e.g. [12, 13] for introductions to some aspects of this broad concept. In our investigation, only the hypercube admits such ideas. Here, instead of informing the neighbors "along the dimension", that is, using the canonical choice $(u(1), u(2), \dots, u(d))$ as list of vertex u, one would serve the dimensions according to a low-discrepancy sequence, that is, take the list $(u(x_1), u(x_2), \dots, u(x_d))$, where $x_1, \dots, x_d$ is an injective low-discrepancy sequence in $\{1, \dots, d\}$ in the sense that each subinterval of the sequence has its entries evenly distributed in the (integer) interval $[1..d]$. There are many constructions of such sequences, cf. again [12, 13]. Taking a van-der-Corput sequence in base two of length 16, rescaling and shifting it to an integer sequence in [1..16] and deleting entries larger than 12 we obtain the low-discrepancy sequence $x = (1, 9, 5, 3, 11, 7, 2, 10, 6, 4, 12, 8)$.

Note that such ideas make little sense for random graphs and complete graphs. First, they both have no regular structure to exploit. In addition, the latter are that symmetric that all sequences (to be used by all nodes) are equivalent.

The results given in Table 4 show that the use of low-discrepancy sequences is minimally better that the canonical order, but by far does not reach the results obtained by random sequences. Here it seems that this difference is the (still negligible) price one has to pay for reducing the degree of randomness in quasirandom broadcasting.

	Random broadcast	Quasirandom broadcast		
		canonic	random	low-discr.
Complete graph $K_{2^{12}}$	21.50 ± 1.32	21.04 ± 1.32	21.48 ± 1.32	
Hypercube H_{12}	24.98 ± 1.32	22.37 ± 0.82	22.32 ± 0.80	22.36 ± 0.82
Random graphs with $p = \ln(n)/n$	43.20 ± 11.8	24.28 ± 1.83	24.28 ± 1.82	

Table 4: The influence of the chosen list on the broadcast times.

	Random broadcast	Quasirandom broadcast
H_{12}: 50% failure chance	45.53 ± 3.23	40.41 ± 2.68 (=11.2% faster)
no transmission failures	24.98 ± 1.32	22.37 ± 0.82 (=10.4% faster)
Increase (multiplicative)	1.822	1.806
$K_{2^{12}}$: 50% failure chance	39.38 ± 3.15	39.29 ± 3.18 (=0.23% faster)
no transmission failures	21.50 ± 1.32	21.04 ± 1.32 (=2.14% faster)
Increase (multiplicative)	1.832	1.867

Table 5: Average broadcast times for the hypercube H_{12} and the complete graph $K_{2^{12}}$ with a 50% chance of transmissions failing unnoticed by the sender and without transmission failures.

For complete graphs, we see that the canonical lists are superior to random ones. We currently have little explanation for this behavior. However, the number of 100,000 iterations ensures that this effect is statistically significant (even if practically not of utmost importance).

We conclude that we never observed a big influence of the list structures, even though a real understanding of their influence could not be gained.

6 Robustness against failures

A second important aspect of broadcast protocols, besides their broadcast time, is robustness against transmission errors. A good protocol should still work moderately efficiently even if some transmissions fail or some nodes for whatever reason do not participate as intended.

Naturally, randomized protocols are very robust. In Feige et al. [8], a model was considered where communications links break down forever. For complete graphs it was proven that even if up to $n/3$ links chosen by an adversary break down, randomized broadcast still requires only $\mathcal{O}(\log n)$ steps.

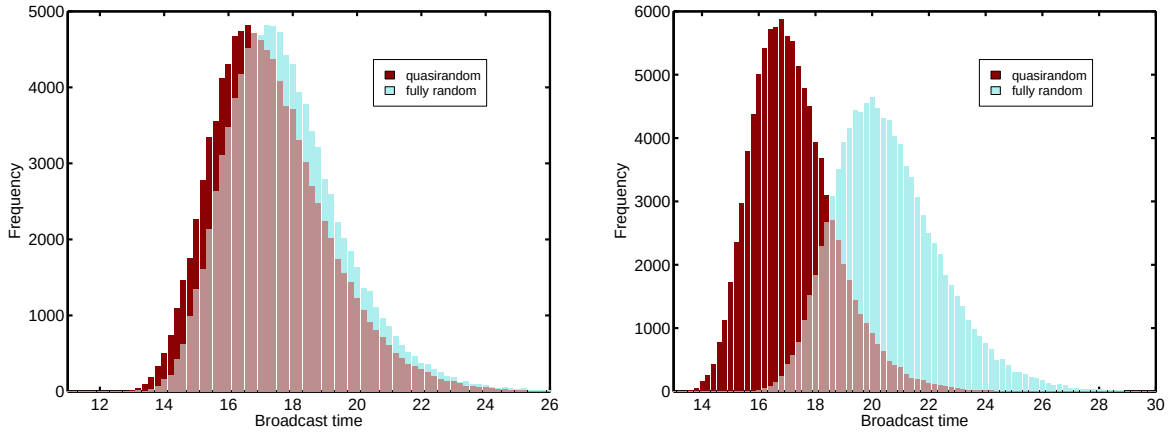
We consider a model where each transmission reaches its destination with some probability $f > 0$ independently. It was shown in [6] that for an arbitrary graph, the runtime of the random model with these transmission failure is at most $6/f$ times the runtime of the random model without failures.

With the fully random model being that robust against different kinds of failures, it is a natural question if the quasirandom broadcast model with its greatly reduced degree of randomness still has comparable robustness properties. To gain some understanding of this issue, we analyze the quite pessimistic model that each transmission fails independently at random with probability a half, and that this remains unnoticed by the sender. Hence the sending node continues his schedule with the next node on his list. Note that in this failure model the advantage that a low-degree node informs all its neighbors in time equal to its degree, clearly vanishes. We would thus expect the superiority of the quasirandom model to reduce significantly.

Surprisingly, this does not happen. We chose the hypercube of dimension $d = 12$ and a complete graph of corresponding size $n = 2^{12}$ as objects of investigation. Averaging over one million runs, we see that both the fully random and the quasirandom model with transmission failures slow down by factors between 1.81 and 1.87 while the standard deviation increases by factors around 3. The precise data is collected in Table 5. We do not have an explanation for the finding that the quasirandom model compared to the fully random one is sightly more affected by failures in the complete graph topology and slightly less in the hypercube topology. This effect, however, seems real and was observed also for smaller graph sizes with more repetitions of the experiment.

7 Asynchronous broadcasting

The broadcasting model discussed so far made the assumption that the agents (nodes) act in a synchronized manner. This assumption is not completely inline with the idea of a self-organized broadcasting pro-



(a) Complete graph $K_{2^{12}}$ **(b)** Hypercube H_{12}

Figure 4: Histograms of the observed runtime over 100,000 runs in the asynchronous setting. The width of each bar is 0.2.

tocol not guided by a central authority. However, there are works on a continuous model called "Richardson's growth model" in the mathematics community, e.g., [1, 9, 10]. In this model, nodes inform other nodes on a continuous time scale. The order in which a fixed node makes its transmission is the same as in the discrete-time models discussed so far. However, the times to elapse between two consecutive transmissions (and between a node first getting informed and its first transmission) are chosen according to an exponential distribution with expectation one[2]. For this model, a bound of $\Theta(\log n)$ was shown for complete graphs [10] and hypercubes [1, 9].

To see how (a) this model compares to the synchronized one and (b) how the random and quasirandom variants compare in this model, we also implemented it. For reasons of space, we only briefly discuss the findings for the 12-dimensional hypercube and the complete graph, both with 2^{12} nodes. Our empirical results averaged over 100,000 iterations are given in Table 6.

On the hypercube, the empirical expected broadcast times are 20.57 and 17.23 for the random and the quasirandom model, respectively. This means that the asynchronous quasirandom broadcast model is 16% faster than the corresponding random model. We measured a standard deviation of 1.88 and 1.50, demonstrating again a superiority of the quasirandom approach. Compared to the runtimes of the synchronous broadcast models, we observe a considerable decrease (cf. Table 2), e.g., the expected broadcast time of the quasirandom model goes down from 22.37 to 20.56.

[2] Actually, in the original "Richardson's growth model", the expectation is $1/\deg(G)$ on regular graphs. However, to make the results of this section comparable to the ones of the synchronous broadcasting, we use an appropriate scaling by setting the expectation to 1.

	Random	Quasirandom
H_{12}	20.57 ± 1.88	17.23 ± 1.50 (=16% faster)
$K_{2^{12}}$	17.79 ± 1.82	17.32 ± 1.82 (=2.6% faster)

Table 6: Empirical broadcast times for the hypercube H_{12} and the complete graph $K_{2^{12}}$ in the asynchronous model.

We also did simulations of the asynchronous broadcast model on a complete graph. Here, the empirical expected broadcast times are also slightly lower for the quasirandom model (17.32) compared to the fully random model (17.79).

In summary, our experiments demonstrate that the advantages of the quasirandom model extend to its asynchronous version.

8 Discussion

In this work, we experimentally analyzed the quasirandom version of the classical randomized rumor spreading model. Our investigation shows a number of interesting facts, which previous work via mathematical means was not able to show. In such, this work nicely complements existing theoretical work. The latter gives a guarantee that the quasirandom model does not fail, and in fact is not worse than the random model, no matter how the neighbor lists are chosen. Note that due to the sheer number of possible lists, there is no way to

gain such a worst-case analysis experimentally.

On the other hand, this work indicates that the typical behavior of the quasirandom model is better than what the theoretical bounds show, and also better than the random model. This statement holds in particular with respect to expected runtimes, deviations from the mean value and robustness against transmission failures.

Given the shown differences between the two models, this work also motivates a further development of the methods to analyze dependent randomized algorithms, so that results of this kind can also be obtained in a mathematically rigorous way.

References

[1] B. Bollobás and Y. Kohayakawa. On Richardson's model on the hypercube. In B. Bollobás and A. Thomason, editors, *Combinatorics, Geometry and Probability*, pp. 129–137. Cambridge University Press, 1997.

[2] C. Cooper and A. Frieze. The Cover Time of Random Regular Graphs. *SIAM Journal of Discrete Mathematics*, 18:728–740, 2005.

[3] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database maintenance. In 6*th ACM Symposium on Principles of Distributed Computing (PODC)*, pp. 1–12, 1987.

[4] B. Doerr and A. Huber. Tight bounds for quasirandom rumor spreading. Submitted.

[5] B. Doerr, T. Friedrich, and T. Sauerwald. Quasirandom Rumor Spreading. In 19*th ACM SIAM Symposium on Discrete Algorithms (SODA)*, pp. 773–781, 2008.

[6] R. Elsässer and T. Sauerwald. On the Runtime and Robustness of Randomized Broadcasting. In 17*th International Symposium on Algorithms and Computation (ISAAC)*, pp. 349–358, 2006.

[7] P. Erdős and A. Rényi. On random graphs. *Publ. Math. Debrecen*, 6:290–297, 1959.

[8] U. Feige, D. Peleg, P. Raghavan, and E. Upfal. Randomized broadcast in networks. *Random Structures and Algorithms*, 1:447–460, 1990.

[9] J. Fill and R. Pemantle. Percolation, first-passage percolation and covering times for Richardson's model on the n-cube. *Annals of Applied Probability*, 3:593–629, 1993.

[10] A. Frieze and G. Grimmett. The shortest-path problem for graphs with random arc-lengths. *Discrete Applied Mathematics*, 10:57–77, 1985.

[11] R. Karp, C. Schindelhauer, S. Shenker, and B. Vöcking. Randomized Rumor Spreading. In 41*st IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 565–574, 2000.

[12] J. Matoušek. *Geometric Discrepancy.* Springer-Verlag, Berlin, 1999.

[13] H. Niederreiter. *Random number generation and quasi-Monte Carlo methods*, Vol. 63 of *CBMS-NSF Regional Conference Series in Applied Mathematics.* Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1992.

[14] B. Pittel. On spreading a rumor. *SIAM Journal on Applied Mathematics*, 47:213–223, 1987.