# On the Design of Hybrid Peer-to-Peer Systems

Stratis Ioannidis
University of Toronto
10 King's College Rd.
Toronto, ON, Canada.
stratis@cs.toronto.edu

Peter Marbach
University of Toronto
10 King's College Rd.
Toronto, ON, Canada.
marbach@cs.toronto.edu

## ABSTRACT

In this paper, we consider *hybrid peer-to-peer systems* where users form an unstructured peer-to-peer network with the purpose of assisting a server in the distribution of data. We present a mathematical model that we use to analyze the scalability of hybrid peer-to-peer systems under two query propagation mechanisms: the random walk and the expanding ring. In particular, we characterize how the query load at the server, the load at peers as well as the query response time scale as the number of users in the peer-to-peer network increases. We show that, under a properly designed random walk propagation mechanism, hybrid peer-to-peer systems can support an unbounded number of users while requiring only bounded resources both at the server and at individual peers. This important result shows that hybrid peer-to-peer systems have excellent scalability properties. To the best of our knowledge, this is the first time that a theoretical study characterizing the scalability of such hybrid peer-to-peer systems has been presented. We illustrate our results through numerical studies.

## Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design; C.4 [**Performance of Systems**]: Modeling techniques

## General Terms

Design, Performance

## Keywords

peer-to-peer, scalability

## 1. INTRODUCTION

Peer-to-peer systems have been very successful at scaling without the need for high infrastructure investments, by utilizing the bandwidth available to end-hosts. In this paper, we investigate the behavior of particular peer-to-peer system architecture, called *hybrid peer-to-peer systems*, which allows to reduce the traffic at a central server by delegating a fraction of it to the server's clients. Applications such as Bit-Torrent and Skype are using this architecture to successfully to scale to a large number of users with little bandwidth requirements on the server side. BitTorrent, an application for peer-to-peer content distribution, allows simple Internet users with limited bandwidth capacity to upload bandwidth-intensive content to thousands of other users. It has also been used by traditional content distribution companies to relay part of their web traffic to their own clients. Skype, a company offering VoIP telephony, also utilizes end-host bandwidth to reduce its served traffic and therefore also its infrastructure costs. For example, Skype users form a peer-to-peer network that implements a decentralized directory mapping user IDs to their IP addresses. Whenever a user connects to the network, it propagates a query in the peer-to-peer system to locate the IP addresses of the users that appear in its "buddy list". Skype [13] claims that decentralizing this "resource-hungry infrastructure" and "leveraging all of the available resources in (the) network" has allowed the company to focus its own resources elsewhere.

The success of systems like BitTorrent and Skype suggests that hybrid peer-to-peer architectures can be very effective in reducing the server traffic, to an extent that a large user population can be supported while requiring only limited resources at the server side. However, there is currently no formal analysis that quantifies precisely how the server traffic behaves. In particular, there are currently no answers to very basic questions such as: (1) How fast does the bandwidth available at the server has to grow as the number of users increases, (2) how does the traffic load imposed on individual users grow as a function of the user population, and (3) to what extend can we establish a tradeoff between the traffic load at the server and the traffic load at individual users.

In this paper, we address these questions by considering a hybrid peer-to-peer system consisting of a central server and clients (users) that download data maintained by the server. There are many applications that are captured by this architecture, such as a search engine, an online encyclopedia, a general content distribution system or a decentralized BitTorrent tracker. The users of this system form an unstructured peer-to-peer network with the purpose of alleviating the traffic load at the server. This is achieved through the following mechanism: A user wishing to retrieve a particular data item first propagates a query over the peer-to-peer net-


*SIGMETRICS'08,* June 2–6, 2008, Annapolis, Maryland, USA.

work. If a user that has already downloaded this data item is reached then the data is retrieved (downloaded) without the intervention of the server. If, on the other hand, the query fails to locate such a user, the data is retrieved directly from the server.

The main contributions of this paper are as follows. First, we propose a mathematical model for the above hybrid peer-to-peer system. Second, we use this model to characterize the traffic load at the server, the traffic load at users and the tradeoff between them for two well-known query propagation mechanisms: the *random walk* [7, 10] and the *expanding ring* [10]. Our results formally show that hybrid peer-to-peer systems scale extremely well. For the random walk query propagation mechanism with a time-to-live that is proportional to the user population, we show that the traffic load at the server and individual peers can stay bounded as the user population grows. This result is surprising and has important implications: it shows that it is possible to construct hybrid peer-to-peer systems that can handle query traffic generated by a large (unbounded) number of users even when the bandwidth capacities of both the server and the users are limited. We also characterize the average response time for the two query propagation mechanisms. As expected, the expanding ring mechanism leads to a much shorter query response time compared with the random walk mechanism. We validate our modeling assumptions, and illustrate our analytical results, through numerical case studies. To the best of our knowledge, our work is the first to formally characterise the performance and scalability of such hybrid peer-to-peer systems.

While in this paper focuses on the design issues in hybrid peer-to-peer networks, the model and analysis also have the potential to address issues beyond what we discuss in this paper. In Section 6, we provide a summary of such possible extensions.

The remainder of this paper is organized as follows. In the next section we present the related work in this area. In Section 3 we give an overview of our model and in Section 4 we present our analysis. We validate our results through a numerical study on Section 5 and we present our conclusions in Section 6.

## 2. RELATED WORK

The random walk and the expanding ring query propagation mechanisms have been studied in the context of unstructured peer-to-peer networks. Gkantsidis *et al.* [7], compare flooding, random walks and uniform sampling search strategies for various overlay network topologies with respect to their hit-rate, *i.e.*, the number of copies of a file a search strategy can retrieve. In their work, Gkantsidis *et al.* also provide evidence that the overlay graphs of unstructured peer-to-peer networks tend to be expanders. This is an important result, as discussed in Section 3.5. Our analysis builds on this work, however our focus is on query traffic as opposed to hit-rates.

The random walk and the expanding ring have also been studied with respect to optimal replication strategies. Lv *et al.* [10] and Cohen and Shenker [4] investigated replication in unstructured file-sharing peer-to-peer systems. Assuming bounded storage capacity, the above papers show that to minimize the average search cost of the random walk search strategy a file should be replicated at a rate that is proportional to square root of the rate at which it is requested. Tewari and Kleinrock [16, 17], corroborate this result through simulations and extend it by showing that the optimal replication rate for the expanding ring is proportional to the request rate of a file. In our work, replication is determined by user demand and is therefore not treated as a design parameter of the system.

The scalability of peer-to-peer systems has been extensively studied in the context of structured systems (see for example [12,14]). For example, the search cost in Chord [14] is known to be $O(\log n)$ w.h.p., and maintenance costs are known to be $O\left(\log^2 n\right)$ w.h.p. Structured peer-to-peer networks were proposed as an alternative to unstructured peer-to-peer systems, as it was widely believed that the search cost in unstructured systems does not scale. Our analysis suggests that unstructured approach can be used to build a scalable hybrid system, if the query propagation mechanism is properly designed.

BitTorrent-like systems [11] as well as hybrid peer-to-peer systems for Video-on-Demand and live streaming [2,15] have been analysed in terms of the download times and playback rates achievable, respectively. The main challenge in such systems is in designing strategies for pushing or pulling partial content (or chunks) of the file or the stream to optimize the above metrics. Our approach is orthogonal, as our focus is on the traffic generated by query propagation; as such, our system can thus be used *e.g.* to decentralize a BitTorrent tracker or to index offered streams.

## 3. MODEL

### 3.1 Peer-to-Peer Network

Consider a hybrid peer-to-peer system where users issue queries for data stored at the server. Query packets are first propagated over the peer-to-peer network. If a user that stores the data requested is reached by the query then the query is considered successful and the data is retrieved (downloaded) from this user. If the query in the peer-to-peer network fails then the query is redirected to the server. After obtaining the data, the user stores it locally and shares it with other users.

The user population is dynamic and users may enter and exit the system. However, we assume that both the size of the user population and the overlay graph topology remain fixed as time progresses. In particular, users stay in the system for i.i.d. random times, exponentially distributed with parameter $\mu$. To keep the size of the population fixed, we assume that a departing user is immediately replaced by a new one. Furthermore, we assume that the new user occupies the vertex of the overlay graph that was vacated by the departing user, and thus the topology of the overlay graph remains unchanged. We denote with $n$ the *size* of the system, *i.e.*, the number of users in the system, and with $G_n$ the (undirected) overlay graph for a system of size $n$. Finally, we also assume that each overlay graph $G_n$ is connected, and that the degrees of vertices of the graphs $G_n$ are no more than constant $d > 2$. The latter assumption is motivated by the fact that unstructured peer-to-peer networks typically set a limit on the maximum number of connections per user, to keep maintenance traffic low.

The systems that we capture with the assumption of a fixed size $n$ are systems of relatively slow growth, in which the increase of the user population happens at a different time scale (*e.g.*, months) compared to the time scale in

which normal system operations take place (*e.g.*, minutes or hours). When seen over middle-ranged periods of time (*e.g.*, a day or a week), the size of such systems should oscillate around an operating point, which is reflected in our model by the size parameter $n$. Our model therefore characterizes the behavior around such an operating size, and the dependence of this behavior in $n$ describes the long-term scalability of the system. We note that we validate our results in Section 5 for a system with a size that is not fixed but varies around an operating point.

The assumption that the topology of the overlay graph does not change is quite strong. In a real system, the overlay graph is dynamic and its topology is affected by user arrivals and departures. As we will see in Section 5 however, the results derived by our static model give a correct characterization, both qualitative and quantitative, of the behavior of a fully dynamic system. This is not a coincidence, as our analysis does not depend explicitly on the time-invariance of the overlay graph but rather the invariance of a single property, namely its relaxation time. As discussed in Section 3.5, this is bounded w.h.p. in unstructured peer-to-peer networks.

## 3.2 The Data Request Process

We assume that the server stores a finite number of $M$ data items. Furthermore, we assume that a new user will issue a query for a given item $j$, independently with a probability $p_j = p_j(n)$, $j = 1, \ldots, M$. We call $p_j$ the *request probability* of item $j$. Note that $p_j(n)$ serves as a characterization of the popularity of item $j$ and can be estimated empirically, *e.g.*, by observing the number of users that request a data item when the operating size of the system is $n$. To interpret the qualitative meaning of the dependence of this probability on $n$, observe that the expected number of users in the system that are interested in retrieving the item $j$ is equal to $np_j(n)$. Therefore, if *e.g.* $p_j(n) = 0.5$, item $j$ retains its relative popularity as the system size increases and the expected number of users that request the item $j$ is linear in $n$. If on the other hand $p_j(n) = 1/n$, while the user population increases only a constant number of users, in expectation, maintain an interest for the data item. This could be true if, for example, the item is popular only within a user minority or a special interest group that, at any point of the evolution of the system, is represented by a fixed number of active users. Similarly, a probability smaller than $1/n$ would indicate that the interest in the data item wanes to an extent that fewer users request it, in expectation, as the system grows.

Without loss of generality, we will focus in our analysis on a single item. In particular, we will be characterizing the query traffic loads (at the server and at peers) generated by queries for a single item $j$. The total traffic load generated by all $M$ types of queries can be obtained by summing the individual loads generated per data item. We therefore omit the index in our analysis and denote with $p(n)$ the probability that a given query takes place. Furthermore, we will assume for simplicity that all data items are requested by a user immediately when it enters the system. Our analysis and our results can be easily extended to the case where users issue a query for a data item at an epoch uniformly distributed over a user's lifetime. Such a system is also Markovian and the bounds we derive hold within a constant factor. However the derivations become more complicated; for this reason, we simply validate that our results also hold for this case in the numerical study of Section 5.

## 3.3 Query Propagation

We consider two query propagation mechanisms: the *random walk* and the *expanding ring*. In our analysis of the random walk mechanism, we will assume that the transmission of a query packet is exponentially distributed with mean $\delta$ time units. For the expanding ring mechanism, we will assume that the transmission time of a query packet is non-random and is exactly $\delta$ time units. This assumption is only made for technical reasons: the random walk is easier to analyse in the continuous realm, whereas the expanding ring is easier to analyse when hops are discretized. In practice, both transmission models yield exactly the same qualitative and quantitative behavior.

In the random walk mechanism, the user issuing a query chooses one of its neighbors in the overlay graph at random and forwards a query packet to it. A user that receives a query packet checks if it can resolve the query, *i.e.*, whether it stores the item requested locally. If it has the item, it notifies the user who initiated the query. If not, it forwards the packet to one of its neighbors, chosen again randomly. No information is maintained about the users that receive the query and a user may receive the same query packet more than once. Furthermore, each query packet contains an expiration time field that is initialized to a predefined value $T_{\max}(n)$ by the user issuing the query. A user does not forward a packet that has been in the system for more that $T_{\max}(n)$ time. Note that $T_{\max}(n)$ can be a function of the user population $n$. As we will see, a user does not need to know the precise value of $n$ - an estimate that is linear in $n$, no matter how far from the exact value (*e.g.*, $0.01n$ or $100n$), suffices. Such an estimate can be determined by crawling the overlay graph. As the systems that we consider are systems of slow growth and inaccuracy can be tolerated, such crawls can be scheduled infrequently (*e.g.*, once a day).

The user that issues the query waits for a response for a waiting period of $T_{\max}(n)$ time units. If this period expires and no response is received, the propagation is considered as failed and the user issues a new query directly to the server.

The larger the value of $T_{\max}(n)$ the more likely the query is to be resolved within the peer-to-peer network. As a result, a large value of $T_{\max}(n)$ tends to reduce the query traffic that reaches the server while also increasing the query traffic imposed on users. We will consider the function $T_{\max}(n)$ as a design parameter and study how $T_{\max}(n)$ should scale as the user population grows in order to achieve a desired tradeoff between the traffic load at the server and at individual users.

The second query propagation mechanism that we consider is an expanding ring mechanism [10]. Essentially, this is a sequence of simple flooding searches in which the time-to-live (TTL) is incrementally increased at each iteration. In simple flooding, a user sends a query packet to *all* its neighbors. A user that receives a query packet forwards it only if has not already received it in the past. A TTL field is used, that is decremented every time a user forwards a query packet. A user ceases to forward a packet when the TTL field becomes zero. In the expanding ring mechanism searching happens by flooding in several stages of increasing TTLs. In particular, the user that issues the query first searches by simple flooding with a TTL $= 1$. If the query is not successful within some time threshold $T = \text{TTL} \cdot \delta$,

the user increments the TTL by one and repeats the search. This process is repeated until either the query is resolved or the time threshold exceeds a value $T_{\max}(n)$. Like in the random walk mechanism, $T_{\max}(n)$ determines the tradeoff between the server load and the load per peer. We will again consider $T_{\max}(n)$ as a design parameter and study how $T_{\max}(n)$ should scale in order to achieve a desired tradeoff between the load at the server and at individual users.

Furthermore, we will make the following simplifying assumption throughout our analysis: the total query response time is negligible compared to the lifetime of a user. In fact, we decouple the propagation of a query from the rest of the dynamics of the system, assuming that queries are instantaneous when viewed in the timescale determined by user arrivals and departures. In effect, the system remains static during a query propagation. This assumption seems quite strong given that we allow the maximum waiting times $T_m(n)$ to be functions on the system size. In particular, for systems with long response times we expect this assumption will be violated and our model to be inapplicable. However, as we show in our numerical study, the results obtained under our model remain valid even if query response times are large, as long as, after obtaining the requested data, the user stays in the system and shares the data for a time that is exponentially distributed with mean $1/\mu$.

## 3.4 Load Metrics and Query Response Time

We assume that each query packet that a user receives has a cost of one unit, which accounts for the bandwidth required to receive and forward the query packet. For the random walk and the expanding ring mechanisms, we define the traffic load at a user as the expected cost (*i.e.*, the expected number of query packets) incurred at the user per unit time. In our analysis, we will focus on the *average load per user* $\rho(n)$, which we define as the average traffic load, in packets per second, over all users in the system. Similarly, each query that is sent to the server incurs a cost of one unit, and we define the *server load* $\rho_0(n)$ as the expected query packet cost per unit time on the server.

Ideally, we would like the above loads to be bounded in $n$, irrespective of the file popularity $p(n)$. This would suggest that the load delegated to users is such that a fixed amount of resources (bandwidth) is sufficient both on users and the server for the system to scale. Such a property indicates that the hybrid system indeed serves its purpose of alleviating the server load without overloading the users of the system. In practice, cases in which either of the loads grow slowly in $n$ are also of interest. On the server side, a slowly growing load indicates, *e.g.*, that the company maintaining it can invest in upgrading its infrastructure at a slow pace. Furthermore, the bandwidth available to users may also grow, albeit slowly, as faster connections become more affordable in the long-term period of time within which the system evolves. Even in the bounded resource scenario however, a slow load growth is of interest as it indicates that the system can operate for a large region of values of $n$.

We will also be interested in the *expected query response time* $D(n)$, *i.e.*, the expected time it takes to resolve a query. For the random walk, $D(n)$ is upper-bounded by $T_{\max}(n)$, whereas for the expanding ring $D(n)$ it is upper-bounded by $T_{\max}(n)(T_{\max}(n)+\delta)/2$: this is the sum of the time thresholds of each simple flooding stage.

## 3.5 Vertex Expansion, Relaxation Time and Expander Graphs

In our analysis, we will be making no additional assumption on the overlay graph $G_n$ other than that it is connected and of bounded degree. However, our results are of particular interest for a class of graphs called *expanders*. In this section, we give a brief overview of expander graphs and introduce two graph-theoretic properties that will play an important role in our analysis, the vertex expansion and the relaxation time of a graph.

Let $G$ be a connected graph of $n$ vertices and $V$, $E$ the set of vertices and edges of $G$ respectively. Without loss of generality, assume that $V$ is the set $\{1, 2, \ldots, n\}$. Let

$$N(S) = \{i \in V \mid \exists\ j \in S \text{ s.t. } (i,j) \in E\}$$

denote the neighborhood of a set $S \subset V$. Intuitively, if each user located at a vertex in $S$ has a query packet and forwards it according to the simple flooding mechanism, $N(S)$ is the set of vertices that these packets will reach. The *vertex expansion ratio* $g_G$ of the overlay graph is then defined as [3]

$$g_G = \min_{S:|S|\le n/2} |N(S) \setminus S|/|S|.$$

Hence, $g_G$ indicates the smallest possible "growth per stage" of an expanding ring propagation, and therefore is related to how quickly the propagation "expands" over the graph.

We will also be interested in a quantity that relates to the random walk on $G$. Let $d_i$ be the degree of vertex $i$ and $d_{\max} = \max_{i\in V} d_i$ the maximum degree over all vertices in $V$.We define the transition probability matrix $P = [P_{ij}]$ of a random walk on $G$ by

$$P_{ij} = \begin{cases} 1/d_i, & \text{if } (i,j) \in E, \\ 0, & \text{otherwise.} \end{cases}$$

Observe that $P_{ij}$ is the probability that a query at vertex $i$ is passed on to vertex $j$ under the random walk mechanism.

Recall that in random walk propagations we assume that transmission time of a query packet are exponentially distributed. For such random walks (formally called *continuized random walks* [1]) it is possible to show that if $G$ is connected, the steady state probabilities that the walk is at a vertex $i$ always exist and are equal to

$$\pi_i = \frac{d_i}{\sum_{i=1}^n d_i}. \tag{1}$$

In other words, if the query packet is forwarded for sufficiently long time, the probability it will be on vertex $i$ will be proportional to the degree $d_i$ of $i$.

We define the *relaxation time* [1] $\tau_G$ of graph $G$ as $\tau_G = 1/(1-\lambda_2)$ where $\lambda_2$ is the second largest eigenvalue of matrix of the following matrix $S = [S_{ij}]$:

$$S_{ij} = \pi_i^{1/2} P_{ij} \pi_j^{-1/2}.$$

The relaxation time relates to how quickly the random walk covers new ground as a query is propagated over the graph, just as the vertex expansion ratio did with respect to the expanding ring mechanism (see, *e.g.*, the proof of Proposition 1 for a more precise statement of this relationship). Moreover, for $d_{\max}$ the maximum degree of graph $G$, the relaxation time and the vertex expansion ratio can be bounded in terms of each other as follows [3]:

$$2g_G \ge \tau_G^{-1} \ge g_G^2/(4d_{\max} + 2d_{\max} g_G^2). \tag{2}$$

Consider now the sequence of overlay graphs $\{G_n\}_{n\geq 1}$ of our model, where $G_n$ is a connected graph of $n$ vertices. Recall that there exists a constant $d > 2$ such that for all $n$ the maximum degree $d_{\max}(n)$ of $G_n$ is bounded by $d$. Let $\tau(n) = \tau_{G_n}$ and $g(n) = g_{G_n}$, $n \geq 1$, be the relaxation times and the vertex expansion ratios associated with this sequence. Define $\overline{\tau}$ and $\overline{g}$ as

$$\overline{\tau} = \limsup_{n\to\infty} \tau(n), \qquad \overline{g} = \liminf_{n\to\infty} g(n). \tag{3}$$

Then $\{G_n\}_{n\geq 1}$ called an *expander family* [8] if $\overline{g} > 0$, *i.e.*, the sequence $g(n)$ is bounded away from zero. In fact, as $\overline{g}$ and $\overline{\tau}$ can be bounded in terms of each other through ineq. (2), an equivalent definition of expanders is as follows: $\{G_n\}_{n\geq 1}$ is an expander family if and only if $\overline{\tau} < \infty$, i.e. if $\tau(n) = O(1)$ (or, the sequence $\tau(n)$ is asymptotically bounded from above). We will occasionally be informal and say that "the overlay graph is an expander" meaning however that the sequence $\{G_n\}$ in our model is an expander family.

The expander property is of importance because it commonly believed that it arises in typical unstructured peer-to-peer systems, as a result of the way that nodes connect in such systems. For example, Gkantsidis et al. [7] argue analytically that the connection protocol employed by most common unstructured peer-to-peer systems today yields overlay graphs that are expanders with high probability. Furthermore, it is also known that for any degree $d > 2$ almost all $d$-regular graphs are expanders [5]. This implies that the expander property does not arise only on the graph formation processes considered by Gkantsidis et al.: any "rich enough" connection protocol (i.e. one that can generate a large enough set of potential overlay graphs) followed by a users in a peer-to-peer network would yield graphs that are expanders. For this reason, although we present general statements for general graphs of bounded degree, we will focus on expander graphs as a case of particular interest.

# 4. ANALYSIS

In this section, we present the theoretical analysis of the random walk and the expanding ring mechanisms. In particular, we characterize the expected query response time, the server load and the average load per user generated by each mechanism for a query that is issued with probability $p = p(n)$. We will pay special attention to the results in the case where the overlay graph is an expander.

## 4.1 Random Walk Mechanism

The main result of this section is that if (a) the overlay graph is an expander and (b) $T_{\max}(n)$ is proportional to the number of users $n$, both the average load per user and the server load generated by the random walk mechanism will be bounded. This suggests that a random walk on an unstructured peer-to-peer network can be used to significantly alleviate the traffic at the server -to the extent that having constant server bandwidth is sufficient- without imposing a significant burden on the users. We note that we also derive bounds on the above two loads and the expected response time $D(n)$ for the (general) case where the overlay graph is not an expander, as well as for non-linear waiting times $T_{\max}(n)$. The latter allows us to describe the tradeoff between the traffic load on the server and traffic load on users.

### 4.1.1 Expected Query Response Time

We begin by giving an upper bound on the expected query response time. Intuitively, queries for items that are not requested very often, and therefore are not carried by users in the system, should require long response times. On the other hand, items that are popular and are requested often should be widely available within the peer-to-peer system, and therefore queries for such items should have a small response time. We are able to quantify the above intuition with the following proposition.

PROPOSITION 1. *The expected response time $D(n)$ for a query that is issued with probability $p(n)$ under the random walk propagation mechanism is such that*

$$D(n) = O\big(\min\big[\tau(n)/p(n), n\tau(n), T_{\max}(n)\big] + T_{\max}(n)(1-p(n))^{n-1}\big).$$

*where $\tau(n)$ the relaxation times associated with the overlay graph sequence $\{G_n\}$.*

Before we present the proof of this statement, to illustrate the intuition behind it, we focus on the case where (a) the overlay graph is an expander, *i.e.*, $\tau(n) = O(1)$ and (b) $T_{\max}(n)$ is sub-linear in the system size, *i.e.*, it is $O(n)$.

Proposition 1 implies that, under the above two assumptions, queries can be categorized under two regimes: frequent queries, for popular items requested with probability $p(n) = \Omega(1/T_{\max}(n))$, and infrequent queries, for items that are requested with probability $p(n) = o(1/T_{\max}(n))$. Then, popular queries have an expected response time of $O(1/p(n))$ while infrequent queries have an expected response time of the same order as the worst case response time, namely $T_{\max}(n)$.

An important observation to make here is that not all queries have delay $T_{\max}(n)$. For example, queries that are issued with constant probability (*e.g.*, $p$ =0.1) will experience a constant expected delay, if the overlay graph is an expander.

PROOF OF PROPOSITION 1. We will first give a Markov process representation of our system. Let $A(t) \subseteq V$ be set of vertices of the overlay graph with users that have the data item at time $t$. Then $A(t)$ is the Markov process described by Figure 1. Because the transition rate between two states

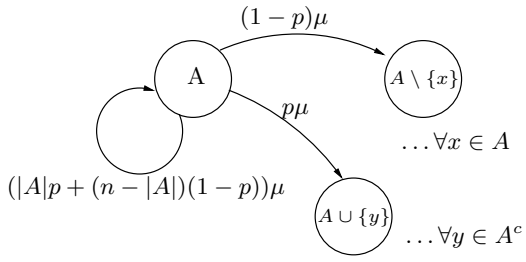


**Figure 1: The Markov process that describes $A(t)$, the set of vertices with users that have the data item.**

$A$ and $A'$ depends only on their cardinalities, the cardinality $|A(t)|$ of $A(t)$ (*i.e.*, the number of users in the system that have the item) is also a Markov process.

Their steady state distribution of $A(t)$ can be derived without the balance equations, merely by observing that the

steady state probability that a given user has the data item is equal to the probability it requests it, namely $p$. Hence, by independence, the steady state distribution of $A(t)$ is

$$\nu_A = p^{|A|}(1-p)^{n-|A|}, \quad A \subseteq V \tag{4}$$

whereas the steady state distribution of $|A(t)|$ is binomial:

$$\nu_k = \binom{n}{k} p^k (1-p)^{n-k}, \quad 0 \le k \le n. \tag{5}$$

Note that $A(t)$ is uniformized (see [6]), *i.e.*, the departure rate from all states is the same, namely $n\mu$. For this reason, the steady state probabilities of the embedded Markov chain (that characterize state transitions) are the same as the steady state probabilities of the Markov process $A(t)$.

Let $T_k$, $k = 1, 2, \ldots$ be delay of the $k$-th query propagation. The sequence $\{T_k\}_{k\ge1}$ can be viewed as a reward function over the embedded Markov chain of Figure 1. We are interested in computing the steady state expected delay per query $D(n) = \mathrm{E}[T]$, *i.e.*, $\mathrm{E}[T] = \lim_{k\to\infty} \mathrm{E}[T_k] = \lim_{k\to\infty} \frac{1}{k}\sum_{j=1}^{k} T_j$ *a.s.* where the last equality holds by ergodicity. By renewal theory,

$$\mathrm{E}[T] = \lim_{k\to\infty} \mathrm{E}[T_k] = \sum_{A\subseteq V} \mathrm{E}[T \mid A]\, \nu_A \tag{6}$$

where $\mathrm{E}[T \mid A]$, $A \subseteq V$, is the expected delay of a query occurring at a transition from state $A$ and $\nu_A$ the steady state probabilities of the embedded chain, given by (4).

Let $\mathrm{E}_i[T_A]$ is the expected time it takes an unconstrained random walk (*i.e.*, one that does not stop after time $T_{\max}$) that starts at vertex $i$ to hit set $A$. Conditioned on a search taking place, the next search may start at any $i \in V$ with equal probability $1/n$. Then,

$$\mathrm{E}[T \mid A] = \sum_{i\in A} \frac{\mathrm{E}_i[\min(T_{A\setminus i}, T_{\max})]}{n} + \sum_{i\in A^c} \frac{\mathrm{E}_i[\min(T_A, T_{\max})]}{n}$$

for any $A \subseteq V$ with $|A| > 1$, $\mathrm{E}[T \mid \{i\}] = \frac{1}{n}\big(T_{\max} + \sum_{j\in V\setminus\{i\}} \mathrm{E}_j[\min(T_{\{i\}}, T_{\max})]\big)$ for any $i \in V$ and $\mathrm{E}[T \mid \emptyset] = T_{\max}$. These give us

$$\begin{aligned}
\mathrm{E}[T] &= \frac{1}{n} \sum_{A\neq\emptyset,V} \sum_{i\in A^c} \mathrm{E}_i[\min(T_A, T_{\max})] \left[\nu_{A\cup\{i\}} + \nu_A\right] + \\
&\quad \frac{1}{n} \sum_{i\in V} \nu_{\{i\}} T_{\max} + \nu_\emptyset T_{\max} \\
&\overset{(4)}{=} \frac{1}{n} \sum_{A\neq\emptyset,V} \sum_{i\in A^c} \mathrm{E}_i[\min(T_A, T_{\max})] p^{|A|}(1-p)^{n-|A|-1} + \\
&\quad (1-p)^{n-1} T_{\max} \\
&= \sum_{A\neq\emptyset,V} \frac{n-|A|}{n} p^{|A|}(1-p)^{n-|A|-1} \mathrm{E}_{u_{A^c}}[\min(T_A, T_{\max})] + \\
&\quad (1-p)^{n-1} T_{\max}
\end{aligned} \tag{7}$$

where $\mathrm{E}_{u_{A^c}}[T_A] = \frac{1}{|A^c|}\sum_{i\in A^c} \mathrm{E}_x[T_A]$, is the expected time it takes a random walk starting from a point uniformly chosen from within $A^c$ to hit set $A$.

Let $\pi_i$, $i \in V$, be the steady state probabilities of a random walk on $G_n$ given by (1), and define

$$\mathrm{E}_{\pi_{A^c}}[T_A] = \sum_{i\in A^c} \pi_i \mathrm{E}_i[T_A] / \sum_{i\in A^c} \pi_i$$

It is easy to see that, for $d_{\max}/d_{\min}$ the ratio of the maximum to the minimun degree of $G_n$,

$$\mathrm{E}_{u_{A^c}}[T_A] \le (d_{\max}/d_{\min})^2 \mathrm{E}_{\pi_{A^c}}[T_A]. \tag{8}$$

On the other hand, it is a well known result (see Aldous and Fill [1], eq. 87 and Corollary 34 of Chapter 3) that

$$\mathrm{E}_{\pi_{A^c}}[T_A] \le \frac{d_{\max}\tau n\delta}{d_{\min}|A|} \tag{9}$$

where $\tau$ the relaxation time of $G_n$ and $\delta$ the expected transmission time per hop. As $d_{\max}/d_{\min} \le d$ where $d$ is a constant, and $\mathrm{E}_{u_{A^c}}[\min(T_A, T_{\max})] \le \min(\mathrm{E}_{u_{A^c}}[T_A], T_{\max})$ by the concavity of the min operator, the theorem follows by substituting the bounds (8) and (9) in (7) and carrying out the summation. □

### 4.1.2 Server Load and Average Load per Peer

The load at the server is characterized by the following proposition, the proof of which we present below.

PROPOSITION 2. *Under the random walk query propagation mechanism, the server traffic load $\rho_0(n)$ generated by queries issued with probability $p(n)$ is such that*

$$\begin{aligned}
&\rho_0(n) = O\big(np(n)\big[(1-p(n)+p(n)e^{-\frac{T_{\max}(n)}{n d\tau(n)\delta}}\big]^{n-1}\big) \quad \textit{and} \\
&\rho_0(n) = \Omega\big(np(n)\big[(1-p(n)+p(n)e^{-\frac{2dT_{\max}(n)}{n\delta}}\big]^{n-1} \cdot \\
&\qquad (1-2p(n)d\tau(n))\big),
\end{aligned}$$

*where $\tau(n)$ are the relaxation times associated with the overlay graph sequence $\{G_n\}$.*

Considering that if the overlay graph is an expander then $\tau(n)$ is bounded, Proposition 2 implies the following:

COROLLARY 1. *If the overlay graph sequence $\{G_n\}_{n\ge1}$ is an expander family and $T_{\max}(n) = \Omega(n)$ the traffic load at the server is bounded,* i.e., *$\rho_0(n) = O(1)$.*

We note that the above result holds irrespectively of what the request probability $p(n)$ is. Intuitively, if $p(n)$ is high users generate many queries for an item. For example, if $p(n)$ is constant, the number of queries generated by users per second is linear in $n$. However, because users store and share the items they request, an item with high request probability will be widely available within the peer-to-peer network. Hence, for such items, the probability that a query reaches the server is quite low. On the other hand, if queries for a data item are infrequent, users storing the item are less likely to be in the system when a query takes place. Such queries are more likely to reach the server. However, as such queries are not frequent to begin with, the overall traffic load they contribute to the server is small. Corollary 1 suggests that waiting times $T_{\max}(n)$ that grow no slower than linearly with the system size exhibit such a behavior on expander graphs to the extent that load on the server always remains bounded.

The above result is even more important when seen in contrast with the load on users.

PROPOSITION 3. *Under the random walk query propagation mechanism, the average traffic load per user $\rho$ generated by queries issued with probability $p(n)$ is such that*

$$\begin{aligned}
\rho(n) = O(&\min[\tau(n), \tau(n)p(n), T_{\max}(n)p(n)] + \\
&T_{\max}(n)p(n)(1-p(n))^{n-1}).
\end{aligned}$$

The proof also appears below. Again, the case where the overlay graph is an expander is most interesting.

COROLLARY 2. *If the overlay graph sequence* $\{G_n\}_{n\geq 1}$ *is an expander family and* $T_{\max}(n) = O(n)$ *the average traffic load per user is bounded,* i.e., $\rho(n) = O(1)$.

To understand the above result, recall from our discussion on response times that if the overlay graph is an expander and $T_{\max}(n) = O(n)$, queries for an item are either frequent, in which case they are served within approximately $1/p(n)$ hops, or infrequent, in which case they are served within $T_{\max}(n)$ hops. Proposition 3 implies that frequent queries generate no more than a constant amount of traffic per user: although such queries are issued often, they are served within a small number of hops ($O(1/p(n))$), and the overall traffic they generate is small. On the other hand, an infrequent query may require many ($O(T_{\max}(n))$) transmissions but, as it does not occur as often, if $T_{\max}(n)$ is sub-linear the overall traffic they generate is decreasing.

Combining Corollaries 1 and 2 we get the following result.

THEOREM 1. *If the overlay graph sequence* $\{G_n\}_{n\geq 1}$ *is an expander family and* $T_{\max}(n) = \Theta(n)$ *both the average traffic load per user and the server traffic load are bounded.*

An important observation based on Propositions 2 and 3 is that, if the overlay graph *is not* an expander, the server load and the average load per user cannot be simultaneously bounded: For $T_{\max}(n) = \Theta(n)$ both can grow as fast as $\tau(n)$, and other values of $T_{\max}(n)$ can only have the effect of increasing one of these two loads. This illustrates the impact that the relaxation time of the overlay graph of a peer-to-peer system has on its scalability.

PROOF OF PROPOSITIONS 2 AND 3 . Let $N(t)$, $N_0(t)$ be the number of queries issued and the number of queries that reached the server up to time and including $t$, respectively. Note that $N(t)$ is Poisson with rate $n\mu p$. By ergodicity, the expected traffic load at the server in steady state is

$$\rho_0 = \lim_{t\to\infty} \frac{N_0(t)}{t} = \lim_{t\to\infty} \frac{N(t)}{t} \cdot \frac{N_0(t)}{N(t)} = n\mu p \cdot P_s \qquad (10)$$

where $P_s$ the steady state probability that, given that a query takes place, it reaches the server. Let $P_i(T_A > t)$ be the probability that a random walk starting from vertex $i \in V$ hits set $A$ in more than $t$ time units. Then as in the proof of Proposition 1, we can show that $P_s = \sum_{A\neq\emptyset,V} \frac{n-|A|}{n} p^{|A|}(1-p)^{n-|A|-1} \mathbf{P}_{u_{A^c}}(T_A > T_{\max}) + (1-p)^{n-1}$ where $\mathbf{P}_{u_{A^c}}(T_A > t) = \frac{1}{|A^c|}\sum_{i\in A^c} P_i(T_A > t)$. Let

$$\mathbf{P}_{\pi_{A^c}}(T_A > t) = \sum_{i\in A^c} \pi_i P_i(T_A > t) / \sum_{i\in A^c} \pi_i$$

where $\pi_i$ the steady state probabilities of the random walk. It is easy to show that

$$d^{-2}\mathbf{P}_{\pi_{A^c}}(T_A > t) \leq \mathbf{P}_{u_{A^c}}(T_A > t) \leq d^2 \mathbf{P}_{\pi_{A^c}}(T_A > t)$$

On the other hand, we have (see Aldous and Fill [1], Chapter 3, Proposition 21 part (iii) and Theorem 43) that

$$\left(1 - \frac{2d|A|\tau}{n}\right) e^{-\frac{d2|A|t}{n\delta}} \leq \mathbf{P}_{\pi_{A^c}}(T_A > t) \leq e^{-\frac{|A|t}{dn\delta\tau}}.$$

Using the above bounds, we can show that the probability that a query hits the server is

$$P_s \leq d^2(1 - p + pe^{-\frac{T_{\max}}{dn\tau\delta}})^{n-1}, \quad \text{and}$$

$$P_s \geq d^{-2}(1 - p + pe^{-\frac{2dT_{\max}}{n\delta}})^{n-1}(1 - 2pd\tau).$$

Proposition 2 therefore follows by substituting the above bounds in eq. (10).

To prove Proposition 3, we associate with each vertex $i = 1, \ldots, n$ of the overlay graph $G_n$ a counting process $\{M_i(t), t \geq 0\}$ that corresponds to the number of query messages that users residing on vertex $i$ have received up to and including time $t$, or, alternatively, the number of times a random walk has passed through or has terminated at vertex $i$. We formally define the load $\rho_i$ at vertex $i$ as the time average load at $v_i$, which is $\rho_i = \lim_{t\to\infty} \frac{M_i(t)}{t}$. We note that this limit exists almost surely in our model and, additionally, by ergodicity it is also equal to the expected load in steady state. The average query traffic load per peer is then $\rho = \sum_{i=1}^n \frac{\rho_i}{n}$. The aggregate search load is $\rho_{tot} = \sum_{i=1}^n \rho_i = \lim_{t\to\infty} \frac{1}{t}\sum_{i=1}^n M_i(t)$. Let again $N(t)$ the number of queries issued up to and including time $t$, and $\{C_j\}_{j\geq 1}$ be the sequence of message costs (i.e. hops of the random walk) associated with the $i$-th query. Observe then that $\sum_{i=1}^n M_i(t) = \sum_{j=1}^{N(t)} C_j$. Therefore $\rho_{tot}$ is:

$$\begin{aligned}\rho_{tot} &= \lim_{t\to\infty} \frac{\sum_{j=1}^{N(t)} T_j}{t} = \lim_{t\to\infty} \frac{N(t)}{t} \cdot \lim_{t\to\infty} \frac{\sum_{j=1}^{N(t)} T_j}{N(t)} \\ &= np\mu \cdot \mathrm{E}[C] \quad a.s.\end{aligned}$$

where $\mathrm{E}[C]$ the expected message cost of a query in steady state. From this we conclude that $\rho = \rho_{tot}/n = p\mu\mathrm{E}[C]$. Finally, one can show that $\mathrm{E}[C] = \Theta(D(n)/\delta)$, where $\delta$ the transmission delay of one message and $D(n)$ the expected query response time. Proposition 3 therefore follows from Proposition 1. □

### 4.1.3 Server Load vs. Load per User and Delay Tradeoffs

Propositions 1 to 3 describe the expected response time and the server and user loads for general maximum waiting time $T_{\max}(n)$ that may not be linear. Here, we investigate the tradeoffs established by Propositions 1 through 3. For simplicity, we focus on the case where the overlay graph is an expander, although the discussion below can easily be extended to the case where $\tau(n)$ is not bounded.

Proposition 3 suggests that the worst-case load at the server, over all request probabilities $p(n)$, is for $p(n) = \Theta(1/T_{\max}(n))$. In this case, the traffic load is $\rho_0(n) = \Theta(n/T_{\max}(n))$. On the other hand, as we discussed above, for $T_{\max}(n) = O(n)$ queries with $p(n) = \Omega(1/T_{\max}(n))$, *i.e.*, frequent queries, generate a bounded average load per user, while queries with $p(n) = o(1/T_{\max}(n))$, *i.e.*, infrequent queries, generate a decreasing load per user ($\rho(n) = o(1)$). We note that reducing $T_{\max}(n)$ and therefore the number of steps taken has the effect of increasing the queries for which the load will decrease as $n$ increases. In other words, when considering the aggregate load per user, generated by all data items, reducing $T_{\max}(n)$ has the effect of reducing the number of items that contribute traffic to the user. In addition, the worse case response time per query also decreases (as it is of the order of $T_{\max}(n)$). On the other hand, this comes at the expense of generating a a server load that scales as $n/T_{\max}(n)$ in the worst case over all probabilities $p(n)$.

To illustrate the above, we consider a numerical example. Suppose that the item popularity $p(n)$ is a decreasing function of $n$, *e.g.*, $p(n) = 1/n^c$ for some $0 < c < 1$, and

the expected number of users requesting the item (given by $np(n)$) grows slower than linear in $n$. For this case, we get that the load at individual users can decrease as the user population grows by choosing $T_{\max}(n)$ appropriately. For example, if $T_{\max}(n) = n^{c'}$, $0 < c' \leq c$, the load per user will be decreasing for all items such that $p(n) = o(1/n^{c'})$. Moreover, the worst-case response time will be no worse than $n^{c'}$. However, this comes at the cost of incurring a load at the server which scales as $n^{1-c'}$.

## 4.2 Expanding Ring Mechanism

As we saw in the previous section, the choice of $T_{\max}(n)$ determines the query response time $D(n)$, as $D(n)$ is of the order of $T_{\max}(n)$ in the worst case (over all probabilities $p(n)$). For example, if we choose $T_{\max}(n)$ to be proportional to $n$ in order to keep the server load and the load per user bounded as $n$ grows, then the query response time will grow linearly in $n$ for queries for unpopular items. Items that are requested with probability $\Omega(1/n)$ on the other hand will have sublinear expected response times. In the worst case however, while the random walk mechanism leads to good performance in terms of the load at the server and at individual users, it might have a poor performance in terms of the query response time.

This motivates our study of the expanding ring mechanism, for which we similarly characterize $\rho_0(n)$, $\rho(n)$ and $D(n)$ under different choices of $T_{\max}(n)$ (see Propositions 4 to 6 ). While we were able to obtain a bounded load both at the server and at individual users under the random walk, this was not the case for the expanding ring. However, if the overlay graph is an expander, we show that for $T_{\max}(n) = \delta \log_d n$ both the server load and the average load per user will be $O\left(n^{1-\log_d(1+\overline{g})}\right)$, where $d$, the bound on the degree of the overlay graph and $\overline{g}$ the limit of expansion ratios given in eq. (3). Note that these loads will grow very slowly if $1 + \overline{g} \leq d$ is large and, as we discussed in Section 3.1, the expanding ring covers the graph quickly. On the other hand, the query response time in this case is given by $D(n) = O\left(\log_d^2(n)\right)$ which is significantly better than the linear response time required under the random walk mechanism to bound both the server and the user load. This implies that the expanding ring mechanism leads to a much improved query response time compared to the random walk. As our bounds are not constant in $n$, they suggest a small increase in the query traffic when the expanding ring is used. However, our approximations involved in computing these bounds are much cruder than the ones used in the previous section for the random walk. As a result, the bounds are not as tight, which is also illustrated in our simulations results of Section 5, and can be improved.

### 4.2.1 Server Load and Average Load Per Peer

The server load is given by the following Proposition, the proof of which we omit for reasons of brevity.

PROPOSITION 4. *For the expanding ring mechanism with $T_{\max}(n) \leq \delta \log_d(n)$, the load at the server $\rho_0$ generated by a query that is issued with probability $p(n)$ is such that*

$$\rho_0(n) = O\big(np(n)(1-p(n))^{\xi(n)}\big)$$

*where $\xi(n) = (1+g(n))^{T_{\max}(n)/\delta}$ and $g(n)$ are the expansion ratios associated with the overlay graph sequence $\{G_n\}_{n\geq 1}$.*

The upper bound that we obtain on server load is not constant. In general, it depends on $p(n)$, but its worst-case order is for $p = \Theta(1/\xi(n))$, in which case it is $\rho(n) = O(n/\xi(n))$. If the overlay graph is an expander, for $T_{\max} = \delta \log_d(n)$, the load at the server is $O\left(n^{1-\log_d(1+\overline{g})}\right)$, where $\overline{g}$ is given by (3). Hence, the server load may grow in $n$, although sub-linearly (with an exponent smaller than one). The larger $1+\overline{g}$, the smaller this exponent is and the slower the server load grows. For smaller values of $T_{\max}(n)$, our bound on the load at the server increases accordingly and for a constant $T_{\max}$ the worst-case load becomes linear.

The average load per user is as follows:

PROPOSITION 5. *For the expanding ring mechanism with $T_{\max}(n) \leq \delta \log_d(n)$, the average load per user $\rho$ generated by a query that is issued with probability $p(n)$ is such that*

$$\rho(n) = \begin{cases} O\left(p(n)^{-(\log_{1+g(n)} d - 1)}\right), & \text{if } p(n) = \Omega(1/\xi(n)) \\ O\left(p(n)\xi(n)^{\log_{1+g(n)} d}\right), & \text{if } p(n) = o(1/\xi(n)) \end{cases}$$

*where $\xi(n) = (1+g(n))^{T_{\max}(n)/\delta}$ and $g(n)$ are the expansion ratios associated with the overlay graph sequence $\{G_n\}_{n\geq 1}$.*

We again omit the proof for reasons of brevity. The above bound on the average load at users also grows with $n$. For an expander graph with expansion ratio $\overline{g}$, the worst-case value (over all probabilities $p$) is for $p = \Theta(1/\xi(n))$, in which case it is $O\left(\xi(n)^{\log_{1+\overline{g}} d - 1}\right)$. For a $T_{\max} = \delta \log_d(n)$, the load is $O\left(n^{1-\log_d(1+\overline{g})}\right)$, *i.e.*, of the same order as the server load for this $T_{\max}$. For smaller values of $T_{\max}$, the average load per user decreases while the server load increases. For a constant $T_{\max}$, our bound on the average load per user becomes constant whereas the one on the load at the server becomes linear.

### 4.2.2 Expected Query Response Time

For the expanding ring mechanism, $D(n)$ is as follows:

PROPOSITION 6. *For the expanding ring mechanism, the expected response time $D(n)$ for a query that is issued with probability $p(n)$ is such that*

$$D(n) = O\left(T_{\max}^2(n)\right).$$

PROOF. The worst-case delay is $T_{\max}(T_{\max}+\delta)/(2\delta)$. □

The response time is therefore of the order of the square of $T_{\max}$. In particular, comparing the case of $T_{\max}(n) = \log_d(n)$ for the expanding ring and the and $T_{\max}(n) = n$ for the random walk, as for these values the server load and load per user become equal, we see that the response time of the expanding ring mechanism is of the order of $\log_d^2(n)$, considerably less than the linear response time of the random walk. The expanding ring mechanism therefore considerably reduces the response time compared to the random walk query propagation mechanism.

# 5. EXPERIMENTAL RESULTS

To illustrate the validity of our analytical results, we conducted a numerical study in which we relaxed several of the modeling assumptions of Section 3. First, instead of assuming that the overlay graph and the system size are fixed, in our simulations we let both vary as time evolves. Second, instead of assuming that the overlay graph is static during

query propagation, our simulated system can change while a query is being propagated. Finally, we allow queries to take place at times chosen uniformly at random within a user's lifetime, as opposed to when a user initially enters the system.

We restricted our numerical evaluation to overlay graphs that are expanders, motivated by the fact that unstructured peer-to-peer networks tend to have this property, as discussed in Section 3.5. To create an overlay graph that is an expander, we use an algorithm proposed by Law and Siu [9] that has been known to construct expander graphs w.h.p.

Even though we considerably relaxed several modeling assumptions of Section 3, for the random walk search mechanism our analytical results of Section 4.1 predict remarkably well the behavior of the simulated system. For the expanding ring mechanism, our analytical bounds on the traffic loads are not as tight. The main reason for this is that our approximations involved in computing these bounds are much cruder than the ones used for the random walk. Overall, the simulation results suggest that our analytical model indeed captures the important features of hybrid peer-to-peer systems, and that the analytical results that we obtain provide the correct insight for the scalability of such systems. In particular, for the case where the overlay graph is an expander, our the simulations confirm the analytical result that the system has very good scalability properties in terms of the traffic generated at both the server and the users.

## 5.1 Simulation Setup

As mentioned above, in our simulations we let the number of users be time-variant. In particular, we let users arrive according to a Poisson process with rate $\lambda$ and stay in the system for exponentially distributed times with mean $1/\mu$ equal to 20 minutes. To scale the system, we repeated our simulations with different arrival rates $\lambda$, ranging between $1,000 \times \mu$ and $500,000 \times \mu$. As a result, the expected number of users in the system in each of our experiments, given by $n = \lambda/\mu$, scales between a thousand and half a million nodes.

In addition, in our simulations we let the overlay graph topology change over time contrary to what we assumed in our model. In particular, we let users join and leave the peer-to-peer network according to the Law and Siu algorithm [9] which we briefly outline below; we refer the reader to [9] for a more detailed description. At any point in time, the overlay graph consists of $d$ Hamilton cycles [18]. Every user in the graph has precisely degree $2d$: for each Hamilton cycle $k$, $k = 1, \ldots, d$, user $i$ is connected to two other users, its predecessor $pred_k(i)$ and its successor $succ_k(i)$. When a new user $i'$ enters the system, it joins each of the $d$ cycles as follows: for each cycle $k$, $k = 1, \ldots, d$, user $i'$ picks a node $j$ at random and becomes its successor on the cycle while also becoming $succ_k(j)$'s predecessor. This way, each node maintains precisely $2d$ connections after an arrival, and each of the $k$ sequences of successors forms a Hamilton cycle. Departures are handled similarly: when a node $i$ leaves, its $d$ predecessors reconnect to the respective $d$ successors of $i$, maintaining thus both a constant degree and the Hamilton cycle property. We use a half-degree of $d = 8$ in the simulations presented here, although we repeated our experiments with values as low as $d = 2$ and obtained similar results.

Finally, in addition to the case where an arriving user requests the data item immediately upon its arrival, we also consider the case where requests are issued at a time uniformly chosen among a user's lifetime. In either case, requests occur with probability $p(n)$. We repeated our simulations for different request probabilities $p(n)$ given by $p(n) = 0.5$, $p(n) = 0.5\log(1000)/\log(n)$, $p(n) = 0.5\sqrt{1000/n}$, $p(n) = 0.5(1000/n)^{1/3}$, $p(n) = 0.5 \cdot 1000/n$, $p(n) = 0.5(1000)^2/n^2$, $p(n) = 0.5(1000)^{2.5}/n^{2.5}$ and $p(n) = (1000)^3/n^3$. We start each simulation in steady state: if the expected number of users is $n$ and the request probability is $p(n)$, we start with a system populated with $n$ users, each one storing a copy of the data item with probability $p(n)$. We observe the system for 20 million arrivals and measure the traffic load at the server, the traffic load at peers and the query response time.

## 5.2 Random Walk Mechanism

We first illustrate the results obtained in Section 4.1 for the random walk query propagation mechanism. Queries in our simulations are propagated according to a random walk, where the one-hop transmission delay of a query is exponentially distributed with mean $\delta = 20$ milliseconds. Queries are redirected to the server after a time period of $T_{\max}(n) = \delta n$, where $n$ the expected number of users in the system.

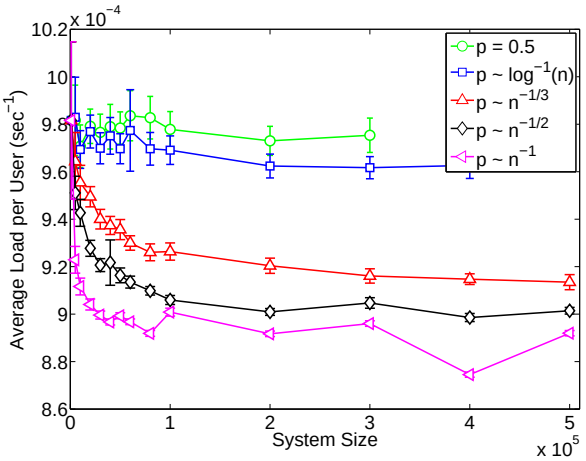


**Figure 2: The average traffic load per user for different request probabilities $p(n) = \Omega(1/n)$, with 98% confidence intervals.**

Our analysis in Section 4.1 suggests that, if $T_{\max}(n)$ is linear and the overlay graph is an expander, then queries can be grouped in to two categories: frequent queries, with request probabilities $p(n) = \Omega(1/n)$, and infrequent queries, with request probabilities $p(n) = o(1/n)$. We first present the results for frequent queries. Figure 2 shows the average traffic load $\rho$ per user, with 98% confidence intervals, for the request probabilities $p(n)$ that decrease no faster than $1/n$. The observed values of $\rho$ confirm Proposition 3 of Section 4.1: the traffic loads generated for all request probabilities $p(n) = \Omega(n)$ are constant, close to one query every thousand seconds. This is indeed the behavior predicted for frequent queries.

In Figure 3, we plot the query response times for frequent queries. The 98% confidence intervals are too small to be displayed on the same graph as the observed values. We also plot with dashed lines the respective bounds on the response time obtained by Proposition 1. To compute these bounds, we use the results from Figure 2 to estimate the

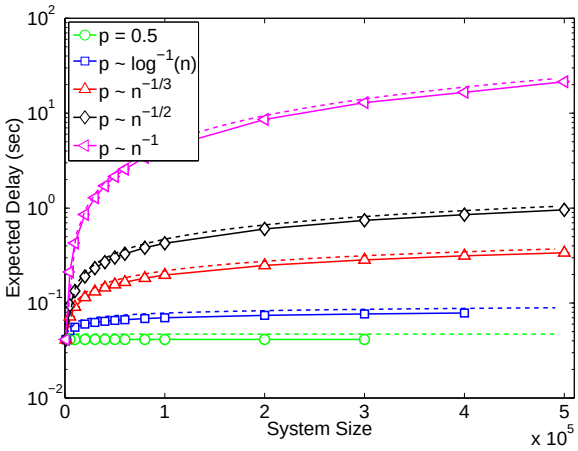


**Figure 3: The query response times for $p(n) = \Omega(1/n)$. The dashed lines indicate the theoretical bounds.**

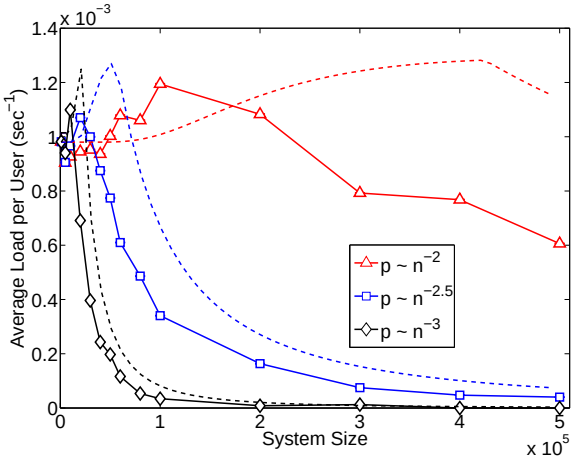


**Figure 4: The average traffic load per user for $p(n) = o(1/n)$. The dashed lines indicate the theoretical bounds.**

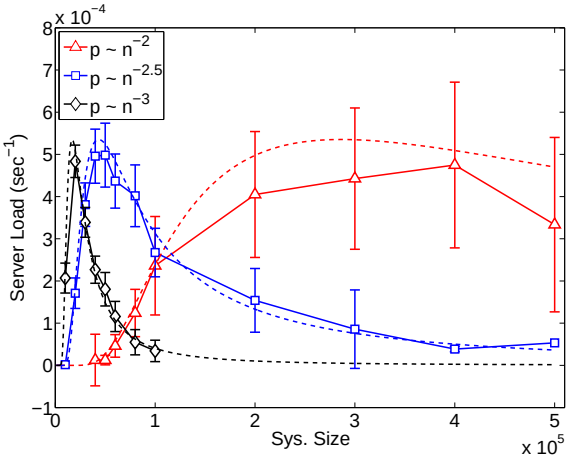


**Figure 5: The server load for $p(n) = o(1/n)$, with 98% confidence intervals. The dashed lines indicate the theoretical bounds.**

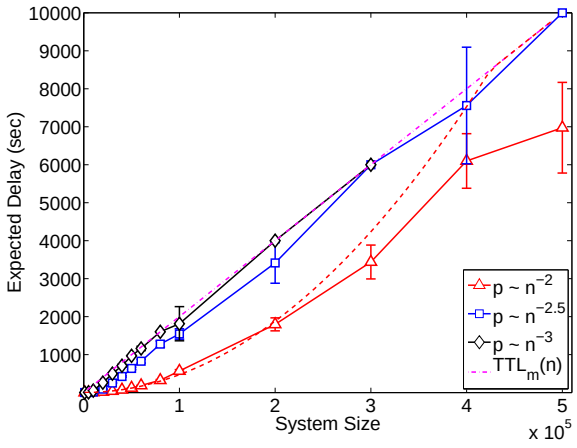


**Figure 6: The query response time for $p(n) = o(1/n)$, with 98% confidence intervals. The dashed lines indicate the theoretical bounds.**

(time-average) relaxation time of the Law and Siu network. In particular, from Proposition 3, we have that $\rho = \tau\mu$ for high request probabilities. Hence, can use the observed value of $\rho$ in Figure 2 to estimate the relaxation time as $\tau = \rho/\mu$. We estimated it to be $\tau = 1.176$ and then used this value to compute the bound on the delay as expressed by Proposition 1. The bounds we obtain give a correct prediction of the response times both qualitatively and quantitatively; in particular, the delays grow as $1/p(n)$. This indicates that our model predicts system behavior quite accurately, in spite of the simplifying assumptions it employs.

For all $p(n) = \Omega(1/n)$ and throughout the 20 million measurements of our simulations, we observed zero traffic at the server. This might seem surprising at first, however this is also consistent with our theoretical analysis. Using $\tau = 1.176$, Proposition 3 gives that, for queries with request probabilities $p(n) = \Omega(1/n)$, the server traffic load is less of $10^{-120}$ queries per second. Such a traffic load is too small to be observed from our measurements.

In Figures 4 to 6 show the average traffic load per user, the query response time and the server load for request probabilities $p(n)$ that decay faster than $1/n$. In each of the above figures, we also plot with dashed lines the theoretical bounds obtained in Section 4.1, using $\tau = 1.176$. For Figure 4, the 98% confidence intervals are too small to be plotted along with the observed values. The theoretical bounds for the age for $p(n) = \Theta\left(1/n^{2.5}\right)$ and $p(n) = \Theta\left(1/n^{3}\right)$ are too close to $T_{\max}(n)$ to be distinguishable, so only the latter appears in Figure 6. We note that, contrary to the frequent query regime, we observe a traffic load on the server.

In our model, we assumed that the network is effectively static during query propagation. As for infrequent queries the delays are quite large, this modelling assumption is clearly violated in our simulations. However, as we see from Figures 4 to 6, our analytical results still correctly predict all three measured quantities, both qualitatively and quantitatively. This suggests that the model used for our analysis indeed captures the important features of the system and provides correct insight on its behavior. As predicted by the analytical results, the above simulations show that (a) the system scales well in terms of both the server and peer traffic loads and (b) the query response times can be large.

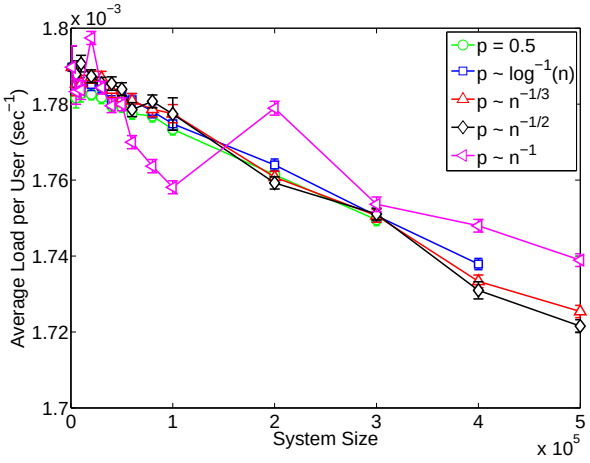


**Figure 7: The average traffic load per user for queries occurring at a time chosen uniformly within a user's lifetime, with 98% confidence intervals.**

Next, we investigated the case in which the queries where not issued upon the arrival of a new user in the system, but at a time uniformly chosen among a user's lifetime. Due to space limitations, we only present these results for the random walk mechanism and for frequent queries, although simulations on both infrequent queries and the expanding ring yielded the same behavior. Comparing Figures 2 and 7, we see that the traffic load is again constant (close to 1 query every 580 seconds), though it has increased roughly by a factor of two. Intuitively, obtaining the data item at a time uniformly chosen from within a user's lifetime reduces the expected time that a user shares the file to 10 minutes, *i.e.*, half the user lifetime (20 minutes), with a corresponding increase in the experienced delay and the traffic loads. In Figure 8 we plot the observed delays along with the theoretical bounds from Proposition 1 for a lifetime of 10min and $\tau = 1.176$. The results further attest to the fact that the system has the same behavior as the one represented in our model, where the mean lifetime has been reduced to 10min.

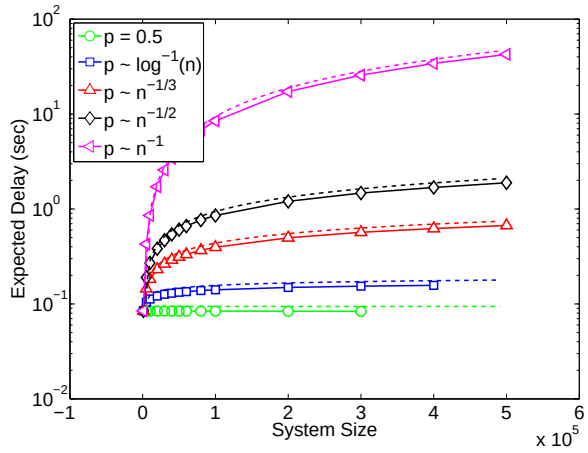


**Figure 8: The query response time for queries occurring at a time chosen uniformly within a user's lifetime. The dashed lines indicate the theoretical bounds with $1/\mu = 10$min.**

## 5.3 Expanding Ring Mechanism

Next, we investigate numerically the expanding ring query propagation mechanism. We repeat the above experiments for the expanding ring with $T_{\max}(n) = \delta \log_{16} n$.

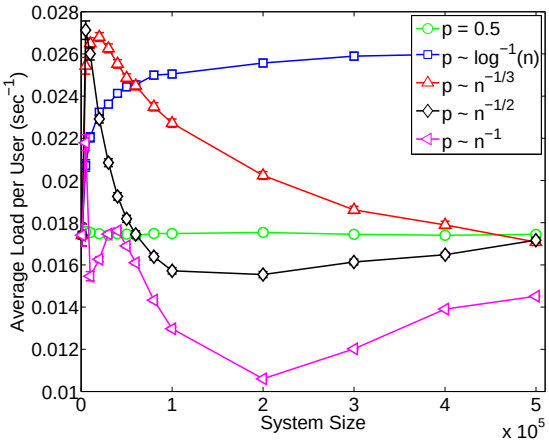


**Figure 9: The average traffic load per user for $p(n) = \Omega(1/n)$, with 98% confidence intervals.**

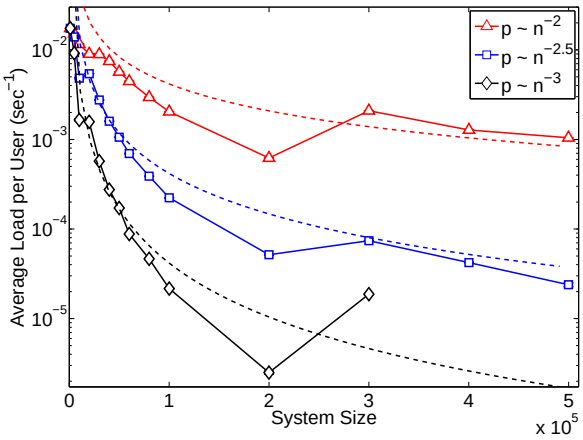


**Figure 10: The average traffic load per user for $p(n) = o(1/n)$.**

Figures 9 and 10 present the average traffic load per user for request probabilities that decay slower and faster than $1/n$, respectively. Comparing Figures 2 and 9, we see that the user loads are 16 to 20 times larger than the ones observed under the random walk. This is not surprising, as the expanding ring propagates messages more aggressively.

Propositions 5 and 4 give bounds for the average load at users $\rho$ and the server load $\rho_0$ for request probabilities $p(n) = \Omega(1/n)$ that grow as (fractional) polynomials of $n$. However, we do not observe such behavior. In fact, the observed loads $\rho$ of Fig. 9 are asymptotically constant, while no traffic was observed on the server. The above suggest that our bounds can be improved and that the expanding ring mechanism also yields a hybrid system with very good scalability properties in terms of traffic loads.

On the other hand, for request probabilities $p = o(1/n)$, our model correctly predicts the asymptotic behavior of the load on the users and the load on the server, as seen in

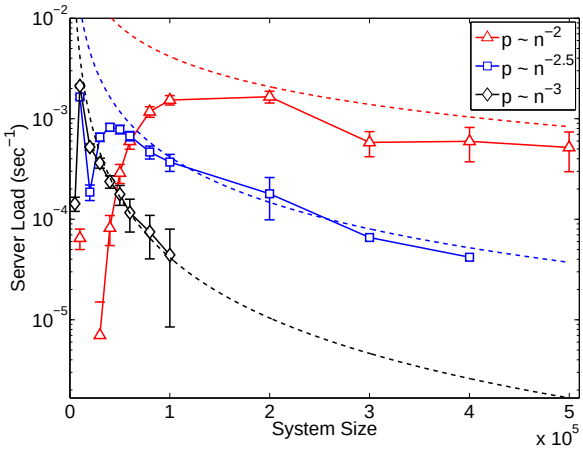


**Figure 11: The server load for $p(n) = o(1/n)$, with 98% confidence intervals.**

Figures 10 and 11. However, we note that we overestimate the behavior for low values of $n$ and that our bounds are not as close as the ones obtained for the same probabilities under the random walk (Figures 4 and 5, respectively).

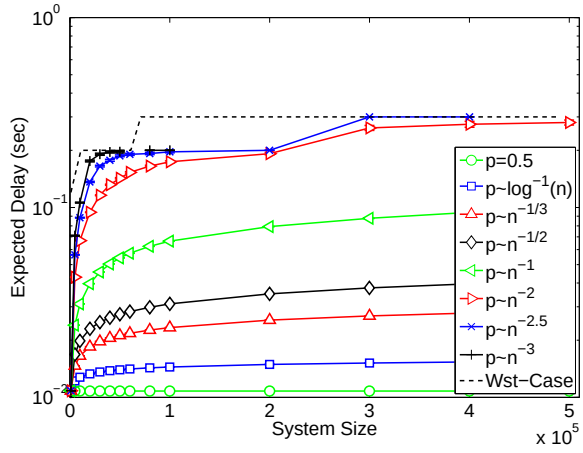


**Figure 12: The query response time for different request probabilities.**

As expected, the expanding ring considerably outperforms the random walk in terms of query response times. Fig. 12 shows the observed response times as well as the worst-case delay of Prop. 6. Response times are considerably smaller (by 2 to 5 orders of magnitude) than the respective ones under the random walk (Figures 3 and 6). In addition, they approach the worst-case bound of Prop. 6 only for $p = o(1/n)$.

## 6. CONCLUSIONS

Our theoretical analysis and our numerical experiments suggest that hybrid peer-to-peer systems with desirable scalability properties can be constructed based on the random walk and the expanding ring query propagation mechanisms. In particular, our analysis shows that a system in which both the traffic load at the server and the traffic load at peers can be bounded irrespective of the system size, a result corroborated by our numerical experiments.

Our proposed model, that captures the effect of overlay graph on the scalability of the system though the relaxation time, can be useful in addressing other questions as well. In particular, an interesting future direction of this work would be to investigate the scalability of different query propagation mechanisms, including variants of the above mechanisms that include path replication or caching of data items. Furthermore, our model could also be applied in a pure peer-to-peer setting. In that context, understanding the relationship between scalability, as expressed by the traffic on peers, and the likelihood that queries succeed, has many parallels with the problem we considered here.

## 7. REFERENCES


[1] D. Aldous and J. Fill. Reversible Markov Chains and Random Walks on Graphs. Monograph in preparation.

[2] S. Annapureddy and S. Guha and C. Gkantsidis and D. Gunawardena and P. R. Rodriguez. Is high-quality VoD feasible using P2P swarming? In *WWW*, 903–912, 2007.

[3] F. R. K. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.

[4] E. Cohen and S. Shenker. Replication strategies in unstructured peer-to-peer networks. *SIGCOMM Comput. Commun. Rev.*, 32(4):177–190, 2002.

[5] J. Friedman. A proof of Alon's second eigenvalue conjecture. In *STOC '03*, pages 720–724, New York, NY, USA, 2003. ACM Press.

[6] R. G. Gallager. *Discrete Stochastic Processes*. Kluwer, Boston, 1996.

[7] C. Gkantsidis, M. Mihail, and A. Saberi. Random walks in peer-to-peer networks. In *IEEE INFOCOM*, volume 1, page 130, 2004.

[8] S. Hoory, N. Linial, and A. Wigderson. Expander graphs and their applications. *Bulletin of the AMS*, 43(4):439–561, 2006.

[9] C. Law and K.-Y. Siu. Distributed construction of random expander networks. In *IEEE INFOCOM*, volume 3, pages 2133–2143, 2003.

[10] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. In *ICS '02*, pages 84–95, New York, NY, USA, 2002. ACM Press.

[11] D. Qiu and R. Srikant. Modeling and performance analysis of BitTorrent-like peer-to-peer networks. In *SIGCOMM*. ACM Press, 2004.

[12] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms*, 2001.

[13] Skype explained. `http://www.skype.com/products/explained.html`. Accessed on 30/7/2007.

[14] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM*. ACM Press, 2001.

[15] K. Suh, C. Diot, J. Kurose, L. Massoulié, C. Neumann, D. F. Towsley, and M. Varvello. Push-to-Peer Video-on-Demand system: design and evaluation. *IEEE Journal on Selected Areas in Communications*, 25(9): 1706–1716, 2007.

[16] S. Tewari and L. Kleinrock. Analysis of search and replication in unstructured peer-to-peer networks. In *SIGMETRICS*, pages 404–405. ACM Press, 2005.

[17] S. Tewari and L. Kleinrock. Optimal search performance in unstructured peer-to-peer networks with clustered demands. In *ICC*, June 2006.

[18] D. R. West. *Introduction to Graph Theory (2nd Edition)* Prentice Hall, 2000.