Can Unstructured P2P Protocols Survive Flash Crowds?

Dan Rubenstein, *Member, IEEE,* and Sambit Sahu

***Abstract*—Today's Internet periodically suffers from hot spots, a.k.a., flash crowds. A hot spot is typically triggered by an unanticipated news event that triggers an unanticipated surge of users that request data objects from a particular site, temporarily overwhelming the site's delivery capabilities. During this time, the large majority of users that attempt to get these objects face the frustrating experience of not being able to retrieve the content they want while still being able to communicate effectively with all other parts of the network. In this paper, we examine whether simple, undirected peer-to-peer search protocols can be used as a backup to deliver content whose popularity suddenly spikes. We model a simple, representative, undirected peer-to-peer search protocol in which clients cache only those objects they have explicitly requested. Because the object that becomes hot initially has limited popularity, the number of cache points, were they to remain fixed, would be insufficient to handle the level of demand during the flash crowd. However, as searches complete, more copies of the object become available. We analyze this natural scaling phenomenon and show that during the flash crowd, copies are distributed to requesting clients at a fast enough rate such that these simple protocols can indeed be used to scalably retrieve content that suddenly becomes "hot."**

***Index Terms*—Average case analysis, flash crowds, peer-to-peer.**

I. Introduction

THE Internet has become a main source of access to timely content ranging from simple news to critical information. While it almost always provides an acceptable level of service, it makes no guarantees. Flash crowds (a.k.a., hot spots) are one such phenomena that limit the Internet's abilities to adequately deliver requested information. A well-known example of this phenomenon occurred on 9/11/2001 when, for most users, news websites were inaccessible. These phenomena are the result of an unpredictable, massive overload of requests to a site, seemingly bringing the site's delivery capabilities to a grind halt even though all or most of the remaining network capacity is available and operational at other locations throughout the network.

During these periods of overload, peer-to-peer (P2P) systems can be used to retrieve duplicates of recently generated objects (e.g., timely news web pages) that originate at the server that is currently exhibiting hot spot symptoms. The object is distributed to these users that seek it via end-to-end communication between peers (i.e., the other users that also seek these objects). The system's ability to deliver the content during the hot spot leverages off the fact that these pairwise end-to-end communications remain operational during the hot spot, and that a small number of users in the peer-to-peer network are still able to get through to the original server during the flash crowd. Prior to a hot spot event, users organize themselves into an *overlay network*: a directed graph that lies atop the underlying Internet infrastructure. The overlay enables its participants to communicate with one another by forwarding messages to and through other participants in the overlay. When a user (or its browser) finds a server unresponsive to its request, it can then search for the object by querying other participants within the overlay.

Most studies of search in P2P systems assume that the popularity of each content item remains fixed with time. In these scenarios, search costs are reduced by replicating objects throughout the distributed (but finite) memory of the P2P system at a frequency that is correlated with the object's popularity [1]. However, during a flash crowd, an item's popularity suddenly spikes such that most likely, an intelligent replication strategy would not have placed a sufficient number of copies of the item in the system. Search costs for the suddenly-popular item would therefore be higher. Does this doom simple, undirected P2P solutions to failure?

In this paper, we show that even simple P2P solutions naturally handle sudden spikes in demand. We develop a model of a simple yet representative, undirected search protocol similar to those used by commercial P2P systems such as Gnutella [2] that currently deployed in existing networks. As peers search for, find, and retrieve the "hot" object, they maintain a copy of this object and can then service subsequent requests that seek the object. As a result, the number of copies cached in the system grows over time as peers locate and retrieve the object. Initially, when the flash crowd first hits, the bandwidth consumed by the searches is high. However, as P2P members retrieve this "hot" object, the number of replicas in the system grows, decreasing expected search costs.

Our mathematical model of a simple, representative P2P search protocol allows us to examine P2P systems that contain millions of peers. Using the model, we compute bounds in large P2P systems on the time it takes for peers to recover objects, as well as the number of messages that are sent through the network to perform this recovery. We show that these simple protocols take orders of magnitude less time to deliver the object to all users in comparison to the hour-long delays times observed in practice using conventional methods. More

Manuscript received March 29, 2003; revised February 19, 2004; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor V. Padmanabhan. This work was supported in part by the National Science Foundation under Grants ANI-0117738, CAREER ANI-0133829, and ITR ANI-0325495. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

D. Rubenstein is with the Department of Electrical Engineering, Columbia University, New York, NY, 10027 USA (e-mail: danr@ee.columbia.edu).

S. Sahu is with the IBM T. J. Watson Research Center, Hawthorne, NY 10532 USA (e-mail: sambits@us.ibm.com).

Digital Object Identifier 10.1109/TNET.2005.845530

specifically, our analysis shows that all users that desire a copy of the object can expect on average to independently retrieve it within fewer than 30 communication rounds, where each round is on the order of a round trip time. In addition, we show that all users participating in the overlay that desire a copy of the object can receive this copy while sending and receiving on average fewer than 400 messages. We compute these results for two arrival patterns of user requests that lie on opposite ends of the spectrum in terms of the number of users whose searches are simultaneously active. First we consider the case where user requests arrive in sequence such that a user's search starts only after a previous search has completed. The other case is when all users simultaneously become interested in the content and begin their searches in unison.

The rest of this paper proceeds as follows. Section II discusses related work. In Section III, we describe the network model and the protocol within the context of that model. Section IV presents a simple, back-of-the-envelope analysis that provides intuition of the scaling phenomenon that we see in our detailed analysis that appears in Section V. Section VI extends the analysis when multiple nodes search for the object. Section VII evaluates the performance using these analytical results and simulations. Section VIII compares analytical results computed over fully connected overlays to simulation results computed upon overlays generated via our shuffling protocol. Section X concludes the paper.

II. Related Work

Much of the research community's recent attention has focused on structured P2P systems that efficiently locate content that is static (unchanging) and easily labeled [3]–[5]. These schemes were designed to efficiently point large sets of users looking for different "cold" objects to different parts of the overlay. A "hot" object would still be problematic in that all users' requests would be forwarded to the same point. This can be remedied by having each node that forward a request cache a copy of the object once it is located and returned. Caching a pointer is not sufficient since the location of the stored object will still be the bottlenecking point. Requiring explicit copying of this form of every object that passes through the system is costly. A preliminary proposal that explores how to design structured P2P networks to handle flash crowds is described in [6].

Much of the recent work in unstructured search P2P systems has involved measuring properties of existing architectures [7], [8]. In [1], an optimal caching strategy is identified for unstructured search networks where demand varies across the objects stored in the P2P system, but where each object's demand is fixed, i.e., the demand for the object does not suddenly spike. We have proposed the protocol analyzed in this paper in [9]. There, we used simulation and experimentation in a wide area prototype to evaluate its performance in more realistic, practical network settings. However, computational resources limited our investigation to P2P systems containing at most a few thousand members. Our analysis here allows us to investigate the protocol's scalability into the millions. Recently, alternative P2P approaches to alleviate flash crowds by means of an unstructured P2P system is discussed in [10] for web documents and in [11] for streaming media.

A wide body of work has considered the theoretical problem of resource location in networks. However, the models considered are often not directly amenable to the problem of delivering information that has suddenly become inaccessible as a result of flash crowds. For instance, in [12] it is assumed that the user seeks a resource that cannot be replicated, and that the network can store state of the search to prevent duplicate traversal along network paths. The massive distribution of popular information is the basis for work in gossip protocols [13]–[15]. A limitation of these gossip-style protocols within a flash crowd scenario is that efficient gossiping requires that those nodes that already have the object be in the position to determine whether this object is worth propagating further. In contrast, in a flash crowd, it is the set of nodes that are without the object that must make such a determination.

Last, we note that our results do not necessarily contradict previous points of view that blatantly claim that Gnutella cannot scale [16]. We are claiming that this type of protocol performs well for an application in which everybody wants a few highly popular objects. These works also suggest that often clients make use of, but do not assist in the searching and object delivery phase of the protocol. We also suspect that the similar content interest also makes it more likely that users of the system will be willing to be "good citizens" and participate in the process of delivering content.

III. Model

In this section, we introduce a simple probabilistic model of the overlay network and, in the context of this model, the P2P protocol that we subsequently analyze. Users in the P2P system can communicate directly with a subset of other users in the system. This subset is referred to as the user's set of *neighbors*. The neighbors are determined prior to the initiation of a search via a separate process. Example methods include the shuffling approach described in [9] and the neighbor selection process used within resilient overlay networks (RON) [17], [18].

We begin with a description of the distributed, scoped search protocol. The reader familiar with this area of work will find this protocol similar to existing P2P protocols such as that used within Gnutella. However the abstraction that we consider here has some minor differences to make the analysis mathematically tractable.

When a user seeks an object, the user initiates a search. The search contains up to $\mathcal{I}$ *iterations*, where the $i+1$st iteration is performed only when the object has not been located during a previous iteration. During the ith iteration, a user searching for an object transmits a query to f of its neighbors in the overlay, selected uniformly at random. Each user that receives a query and has a copy of the object notifies the originator of the query (directly) that it has a copy available for download. If a user does not have a copy, if the query it receives has not been forwarded through a chain of i users, the query is forwarded to f of its own neighbors, selected uniformly at random. Otherwise, the query is dropped. This protocol is easily implemented in

practice by including in the query the identity of the object, the identity of the node that originated the search, and a TTL field that is initially set to i in the ith iteration. The TTL is decremented by each user that receives the query before sending it further—when a node receives a query with the TTL set to 1, that query should not be forwarded any further. If a user completes all iterations without finding the object, the user re-initiates the search, starting from the first iteration.

The above search process is similar but not identical to an expanding ring search protocol: during the ith iteration, neighbors that lie i hops away in the overlay may be contacted. This is slightly different from the traditional flooding approach in which all neighbors that are within i overlay hops are contacted during the ith iteration.

A. Overlay Network Model

Our model of an overlay network is a graph, $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ where $\mathcal{N}$ is a set of nodes and $\mathcal{E}$ is a set of directed edges between nodes. Each node $n \in \mathcal{G}$ represents a user that (has a browser which) is a willing participant within the P2P protocol. We focus on the retrieval of a single "hot" object that is initially stored at N_h of $N = |\mathcal{N}|$ nodes in the network where a total of N_d nodes desire the object (including the N_h that initially have the object), where $0 < N_h \leq N_d \leq N$. Note that we are assuming that $N_h > 0$, i.e., that some copy of the desired object has entered the P2P system. Such an assumption is not unreasonable, given that some clients are able to initially download content before its popularity spikes to excessive levels.

Our description and evaluation of the searching protocol discretizes time into *rounds*. Within a round, a node n_1 can send out queries to as many neighbors as it chooses. However, a node must wait until the subsequent round to transmit any information it receives within the previous round. In other words, for a query to travel along the path from n_0 to n_1, from n_1 to n_2, ..., from n_{k-1} to n_k in the overlay where $n_{i+1} \neq n_{i-1}$ requires k rounds.

B. Metrics of Interest

Within the context of the model, we measure the performance of this protocol using two metrics:

- **Node Bandwidth:** We measure bandwidth in terms of the number of queries that a node should expect to transmit to enable all N_d nodes that desire the object have successfully retrieved it.[1] The total number of queries transmitted by the protocol for a single object's retrieval can be obtained by summing over the individual node bandwidths.
- **Time (in rounds):** We measure time the number of rounds from the round in which a user initiates its first query until the round that the object is returned to that user.

Our analytical results are performed upon two arrival patterns of user search requests:

- **Isolated Attempts:** This is the case where only one user's query propagates through the network at any given time, i.e., the $i+1$st user searching for the object does not begin its search until the ith user has already received the object.
- **Simultaneous Attempts:** This is the case where all users actively seeking the object issue their queries simultaneously.

These patterns represent the two extremes in how an object's popularity can grow. The first is the slowest growth possible, the second is the fastest growth possible.

IV. Preliminary Simple Models

Before proceeding with our detailed analysis of the described protocol, let us first look at the scaling phenomenon using a significantly simplified model. We will focus in this section on the protocol for the case where $f = 1$, i.e., the nodes are searched in sequence. We show that in this case, the bandwidth cost per node scales logarithmically with the number of nodes searching the overlay.

A. Isolated Attempts

Let us first explore a simple model where N_h nodes initially have a copy of the object and the N_d nodes that desire a copy of the object perform their searches one after the other. When the ith searching node starts its search, $(N_h + i)/N$ nodes will have a copy of the object. Assuming a fully connected overlay where a query is forwarded to a neighbor uniformly at random, the expected number of queries made on the behalf of this node to find the object is $N/(N_h+i)$. The expected number of queries required for all N_d nodes desiring the object to receive a copy is $\sum_{i=N_h}^{N_d} N/i$. Since each of the N nodes in the overlay is as likely as any other to receive a query, the expected number of queries a node will receive is $\sum_{i=N_h}^{N_d} 1/i$. This value is clearly largest when $N_d = N$ and $N_h = 1$, and it is easy to show that $(\log_2 n)/2 \leq \sum_{i=2}^{n} 1/i \leq \log_2 n$.[2]

B. Simultaneous Attempts

Our simplified model for the case where users perform their searches at the same time will be a fluid model. Let us consider specifically the case where, in the end all nodes desire a copy of the object. Let $g(t)$ be the fraction of users at time t with a copy of the object, where $g_0 = g(0)$ is the initial fraction of nodes with the object. We assume that the search by each node that has not found the object by time t queries its next node at time $t+dt$. At time t, a fraction $1-g(t)$ of nodes are still searching for the object and after an additional time dt, each searcher chooses a node with a copy of the object with probability $g(t)$. Hence

$$\frac{dg}{dt} = (1 - g(t))g(t)$$

whose solution is (via substitution of variables, then $h = 1 - 1/g$)

$$g(t) = \frac{1}{1 + x_0 e^{-t}} \tag{1}$$

where $x_0 = (1/g_0) - 1$. Note that $g(t)$ is increasing, but never reaches 1, i.e., under this simple model, there is never a point

[1]Note that we are interested in the number of transmissions that arrive or depart from a node, and not in the number of transmissions resulting from a user's search request. The former's load is more evenly shared among overlay nodes, even in the case where users' searches are performed sequentially.

[2]The lower and upper bounds are obtained by noting that for $i > 1$, $1/2^{\lfloor \log i \rfloor + 1} < 1/i \leq 1/2^{\lfloor \log i \rfloor}$.

in time when all nodes have a copy of the object. This is an inaccuracy of the fluid model when $g(t)$ grows very close to 1. Note that in practice, if there are N nodes in the network, then once $g(t) > (N-1)/N$, then there is a single node without a copy of the object and clearly, in the next interval of time, dt, the copy will be found. Another way to apply our fluid model is to compute the time t_h where $g(t_h) = 1/2$. After this point of time, half the nodes are transmitting queries, and each query has probability of at least 0.5 of locating the object. If there are N nodes, then at time t_h, there are $N/2$ nodes, each of which will send a number of queries whose expected value is bounded above by 2. This bound of N on the expected number of queries transmitted after time t_h is distributed uniformly among the set of all nodes, such that the expected number of additional queries that each node must handle after time t_h is bounded above by one. We will therefore bound the expected number of queries per node during the lifetime of the search by determining the expected number of queries per node sent up to time t_h, and then adding one to the outcome.

Setting the left-hand side of (1) to 0.5, we get $t_h = \ln x_0$. We can now compute the expected number of queries in the system from time 0 to time an arbitrary end-time, T. Recall that at time t, there are $N(1-g(t))$ nodes whose searches generate a query each dt, hence

$$B(T) = \int_{t=0}^{T} N(1-g(t))dt.$$

Solving with $y = x_0 e^{-\lambda q}$ and setting $T = t_h$ gives

$$B(t_h) = N\ln x_0 + N\ln(x_0+1) - N\ln(2x_0).$$

Setting $g_0 = 1/N$ yields $B(t_h) = N(\ln N - \ln 2)$. Since this bandwidth is distributed evenly among all nodes in the network, the expected per node is $\ln N - \ln 2$ to time t_h, making the expected number of queries for all nodes to receive the object bounded above by $\ln N - \ln 2 + 1$.

V. Performance Analysis

In this section, we develop the more sophisticated mathematical tools that will allow us to evaluate the bandwidth and time overheads of the more general searching protocol. To make the model amenable to a mathematical analysis, as in [1], we assume the overlay graph is fully connected, i.e., a user's neighbor set is the set of all nodes. We will demonstrate in Section VIII that the results match almost identically to the results obtained through simulation overlays where each node's neighbor set is limited via the shuffling approach described in [9].

We begin by considering the case where a single node n wishes to retrieve an object that is stored at a fraction $1-\nu$ nodes in the network (i.e., a fraction ν do not have the object). With our assumption that the overlay is fully connected, when a node selects a neighbor uniformly at random, that neighbor has the object with probability $1-\nu$.

In the jth round of an iteration, a single query can lead to at most f^j transmissions. This number occurs only when no node contacted in the earlier rounds of the iteration had a copy of the object. We define a *schedule* to be a list of nodes $\{N_{i,j,k}\}$, $1 \le i \le \mathcal{I}$, $1 \le j \le i$, $0 \le k < f^i$ where a node can appear multiple times in the list. By connecting an directed edge pointing to $N_{i,j,k}$ from $N_{i,j-1,\lfloor k/f \rfloor}$ for all nodes $N_{i,j,k}$ where $j > 0$, a balanced f-ary tree is formed whose edges depict the transmission sequence: if node $N_{i,j,k}$ receives a query and does not have a copy of the requested object, then it forward the query onward to set of f nodes $N_{i,j+1,fk}$, $N_{i,j+1,fk+1}$, $N_{i,j+1,fk+f-1}$. When forwarding a query, we assume that each node selects a neighbor uniformly at random *with replacement*, including itself. This greatly simplifies the analysis in that each $N_{i,j,k}$ is selected uniformly at random from the set of nodes in the P2P system.

In any iteration, exactly $f + f^2 + \cdots + f^j = ((f^{j+1} - f)/f - 1)$ queries are forwarded when no node receiving the query during these rounds has a copy of the object. We define $p_j(\nu) = \nu^{(}f^{j+1} - f)/f - 1)$, which equals the probability that none of the nodes contacted within the first j rounds of an iteration contains a copy of the object. The probability that no nodes scheduled for a visit within iteration i contain a copy of the object is $p_i(\nu)$. We define the random variable $L_i(\nu)$ to equal 0 when no scheduled transmission within the first ith iterations arrives at a node, and 1 otherwise. We have $\Pr[L_i(\nu)] = \prod_{j=1}^{i} p_j(\nu)$. By convention, we set $\Pr[L_0(\nu) = 0] = 1$.

We first turn our attention to determining the expected number of rounds taken to either locate the object or proceed through all iterations without locating the object. Define $R_i(\nu)$ to be a random variable to equal k when the earliest visit in iteration i's schedule to a node with a copy of the object occurs during round k, and equals i when no such node exists. Then

$$E\big[R_i(\nu)\big] = \sum_{j=0}^{i-1} \Pr\big[R_i(\nu) > j\big] = \sum_{j=0}^{i-1} p_j(\nu).$$

Let $R(\nu)$ be the number of rounds used by the protocol to retrieve the object when ν is the probability that a node does not hold the object. Then the expected number of rounds for which the protocol runs, including the case where the object is never retrieved is

$$E\big[R(\nu)\big] = \sum_{i=1}^{\mathcal{I}} \Pr\big[L_{i-1}(\nu) = 0\big] E\big[R_i(\nu)\big].$$

We next turn our attention to determining the expected number of transmissions a single running of the search protocol makes (regardless of whether or not the object is located). We define the random variable $X_{i,j,k}$ to equal 1 if scheduled transmission $N_{i,j,k}$ occurs and equal 0 otherwise. Our assumption that the node for each entry $N_{i,j,k}$ is selected uniformly at random over all nodes gives us that that the probability that $N_{i,j,k}$ does not contain the object is ν. It follows that

$$\Pr[X_{i,j,k} = 0 | L_i(\nu) = 0] = \nu^{j-1}.$$

Since $X_{i,j,k}$ is an indicator r.v. which must equal 0 when $L_{i-1}(\nu) = 1$, we have

$$E[X_{i,j,k}] = \Pr\big[L_{i-1}(\nu) = 0\big]\nu^{j-1}.$$

We define $T_i(\nu)$ be the number of transmissions that take place during the ith iteration. Given that there are f^j sched-

uled transmissions scheduled in any iteration that contains a jth round, we have

$$\begin{aligned} E\big[T_i(\nu)\big] &= \sum_{j=1}^{i}\sum_{k=1}^{f^j} E[X_{i,j,k}] \\ &= \Pr\big[L_{i-1}(\nu)=0\big] \\ &\quad \cdot (f+\nu f^2+\cdots \nu^{i-1}f^i) \\ &= \Pr\big[L_{i-1}(\nu)=0\big] f\frac{1-(f\nu)^i}{1-f\nu}. \end{aligned}$$

Letting $T(\nu)$ be a random variable equaling the number of transmissions performed by the protocol, we have

$$E\big[T(\nu)\big] = \sum_{i=1}^{\mathcal{I}} E\big[T_i(\nu)\big]. \tag{2}$$

Having computed the expected number of rounds and transmissions of a single running of the protocol, we can extend our analysis to the case where the user reruns the search protocol until the object is found. Let $T^i(\nu)$ and $R^i(\nu)$ respectively be the number of transmissions and rounds needed during the ith running of the protocol

$$\begin{aligned} E\big[T^i(\nu)\big] &= \Pr\big[L_{\mathcal{I}}(\nu)=0\big]^{i-1} E\big[T(\nu)\big] \\ E\big[R^i(\nu)\big] &= \Pr\big[L_{\mathcal{I}}(\nu)=0\big]^{i-1} E\big[R(\nu)\big]. \end{aligned}$$

We define $T^+(\nu)$ and $R^+(\nu)$ to respectively be the number of transmissions and rounds that are required to retrieve the object, where the protocol is continuously rerun until the object is received

$$\begin{aligned} E\big[T^+(\nu)\big] &= E\left[\sum_{i=1}^{\infty}\Pr\big[L_{\mathcal{I}}(\nu)=0\big]^{i-1}\right] E\big[T(\nu)\big] \\ &= \frac{E\big[T(\nu)\big]}{1-\Pr\big[L_{\mathcal{I}}(\nu)=0\big]} \\ E\big[R^+(\nu)\big] &= E\left[\sum_{i=0}^{\infty}\Pr[L_{\mathcal{I}}(\nu)=0]^{i-1}\right] E\big[R(\nu)\big] \\ &= \frac{E\big[R(\nu)\big]}{1-\Pr\big[L_{\mathcal{I}}(\nu)=0\big]}. \end{aligned}$$

VI. Multiple Queriers

In the previous section, we considered the cost in terms of rounds and transmissions for a single querier to retrieve a copy of the desired object. We now use these results to compute the expected number of rounds for a node to receive a copy of the object from the time of its initial query, and the expected number of transmissions that a node receives (or, equivalently, transmits) when participating in the overlay for retrieval of one hot object by multiple users. We assume a network with N nodes where there are N_d nodes that desire the object with N_h of the N_d nodes starting out with copies of the object.

A. Isolated Attempts

We define $\theta_s(N_d,N,N_h)$ and $\rho_s(N_d,N,N_h)$ to respectively be the total number of transmissions and rounds used to deliver the object to the N_d nodes that desire it when each user transmits its queries at separate times (i.e., not in parallel). This gives us

$$E\big[\theta_s(N_d,N,N_h)\big] = \sum_{i=N_h}^{N_d} E\left[T^+\left(\frac{i}{N}\right)\right] \tag{3}$$

$$E\big[\rho_s(N_d,N,N_h)\big] = \sum_{i=N_h}^{N_d} E\left[R^+\left(\frac{i}{N}\right)\right]. \tag{4}$$

B. Joined Attempts

We define $\theta_s(N_d,N,N_h)$ and $\rho_s(N_d,N,N_h)$ to respectively be the total number of transmissions and rounds used to deliver the object to the N_d nodes that desire it. We assume the time intervals over which the nodes perform their searches overlap. However, we do not assume that all nodes first start their searches at the exact same time, $t=0$. To remove the effect of this unlikely synchrony, we assume that, at time $t=0$, searches have been proceeding for nonzero time, but no search has as of yet located the desired object. We wish to compute the number of rounds taken and transmissions received by a node who is actively searching at time $=0$ from time $t=0$ until it finds the object.

To perform this analysis we introduce a modified search process that will be used only to help us analyze the original search process. We say that the protocol is in renewal mode when a node, after locating a copy of the object, throws the copy away (i.e., does not store the object), and initiates a new search in the next round starting at iteration 1. Any searches in progress that were initiated by the node who just located the object are immediately terminated. We say these searches are "artificially suppressed" since implementing a mechanism to suppress these searches in practice would require instantaneously locating and contacting the nodes that are performing the search. When all nodes operate in renewal mode, the resulting search process will approach a steady state, where, at time t, the probability that a node's most recently initiated search has reached the jth round of the ith iteration is the same for all nodes and all t. Hence, each node's search can be viewed as a renewal process that renews itself whenever the object is located by the current search.

We assume that all N_d nodes that desire a copy of the object have been running in renewal mode for an arbitrarily long time up to time $t=0$, at time $t=0$, all nodes exit renewal mode and resume the normal search operation. Our measurements (time taken and transmissions received) are counted from the time $t=0$.

To achieve a closed form solution, we are extraordinarily conservative in estimating the number of transmissions and rounds. We therefore suspect that these upper bounds are rather loose and that the actual upper bounds are significantly lower. This means that values we compute are likely to be much greater than what a node can expect (in terms of the number of rounds and the number of queries received/transmitted), and guaranteed not

to be lower (within the confines of the model considered here.) Our results demonstrate that even under a worst-case assumption that the upper bound is tight, it is feasible to use such a system in practice.

We compute an upper bound on the expected number of transmissions via the following set of steps:

1) We consider a search where $1-\nu$ nodes have a copy of the object and the other ν nodes are searching for the object in renewal mode (starting from steady state). We compute the expected number of rounds that pass from a round in which an object is located to the next round in which the object is located. We also compute the expected number of transmissions that occur during this period.
2) We conservatively count all transmissions of any schedule of an iteration that is initiated (i.e., even counting those iterations that were "artificially" suppressed).
3) We show that the expected number of transmissions and rounds counted above are a monotonically decreasing function of ν (i.e., for a graph containing a fixed number of nodes, as a larger fraction of nodes contain the object, the expected number of transmissions and rounds needed for a user to locate an object decreases).
4) We assume that at most one node locates the object in any given round.
5) We note that at any point in time, if m nodes currently have a copy of the object, then our assumption in step 4 implies that no fewer than $\max\left\{N_h, m-\sum_{i=1}^{\mathcal{I}} i\right\}$ nodes could have had a copy of the object when the current execution of the protocol was initiated. Using the monotonicity result in step 3, the steady state expected number of transmissions and rounds when the fraction $\max\left\{N_h, m-\sum_{i=1}^{\mathcal{I}} i\right\}/N$ of nodes has the object upper bounds the the number of transmissions and rounds that occur when m nodes have a copy of the object.

We proceed first with step 1 and analyze the expected number of rounds and transmissions that occur in the steady state system in which a node, upon locating an object, does not retrieve the object but, on the next round, re-initiates the search protocol. This modified system is a renewal process that renews each time the node locates its object.

To evaluate the expected number of rounds and transmissions within this renewal process, we divide rounds into m *ticks* where m is the number of users that are actively seeking an object. We assume that the transmissions that relate to a particular user's query within a given round are assigned their own tick, and are all transmitted concurrently during their tick. We assume that if a user is able to locate an object during its tick, then that user can return a copy of the object in response to subsequent transmissions to that user during the same round (on subsequent ticks).[3]

We define $\theta_j(\nu)$ to be the steady state number of transmissions that are performed (at the granularity of a tick) when a fraction $1-\nu$ of the nodes contain a copy of the object. Let $T_{i,\nu}$ be a random variable that equals the number of transmissions sent during the ith tick in system S_0 and let $G_{i,\nu}$ be a random variable that equals 1 when the object is not located on the ith tick of system S_0 where a fraction $1-\nu$ nodes have the object

$$E\big[\theta_j(\nu)\big] = \sum_{i=1}^{\infty} E\left[T_{i,\nu}\prod_{j=1}^{i-1} G_{j,\nu}\right] = \sum_{i=1}^{\infty} E[T_{i,\nu}]\prod_{j=1}^{i-1} E[G_{j,\nu}] = \frac{E[T_{1,\nu}]}{1-E[G_{1,\nu}]}. \tag{5}$$

The proof of why the second equality holds, included in the Appendix, involves partitioning the random variables in the equation into a series of random variables from which the independence results can be observed.

We now proceed on to step 2. To compute an upper bound on $E[\theta_j(\nu)]$, we consider the renewal process described above. To perform this computation, we label the states of the protocol as (i,j), which corresponds to the jth round within the ith iteration. Letting $\sigma(\nu)$ represent the sampled state when the fraction of nodes with the object is $1-\nu$, we use the fact that the system is a renewal process in which the renewal commences at iteration 1, round 1, giving

$$\Pr\big[\sigma=(i,j)\big] = \frac{\nu^{A(i,j)}/\big(1-L_{\mathcal{I}}(\nu)\big)}{\sum_{k=1}^{\mathcal{I}}\sum_{\ell=1}^{k}\nu^{A(k,l)}/\big(1-L_{\mathcal{I}}(\nu)\big)} = \frac{\nu^{A(i,j)}}{\sum_{k=1}^{\mathcal{I}}\sum_{\ell=1}^{k}\nu^{A(k,l)}} \tag{6}$$

where $A(j,k) = k + \sum_{s=1}^{j-1} j$, i.e., the number of rounds from the start of the protocol to the point at which the kth round of iteration j occurs.

We note that if the object is received during iteration/round (i,j), the protocol might continue to perform transmissions up until the last round of the ith iteration. We upper bound the number of queries transmitted in (i,j) by $a(j,k) = \sum_{s=k}^{j} f^s$, which is the number of transmissions in the schedule from the kth round of iteration j to the final round j of the iteration. Note that this not only overestimates the number of transmissions sent on and after (i,j) when the object is located during this round, but it even more grossly overestimates the f^s transmissions that must be counted when the object is not located during the round. Our upper bound on $E[T_{i,\nu}]$ and an exact computation of $E[G_{i,\nu}]$ are computed as

$$E[T_{i,\nu}] \le \sum_{j=1}^{\mathcal{I}}\sum_{k=1}^{j} a(j,k)\Pr\big[\sigma(\nu)=(j,k)\big] \tag{7}$$

$$E[G_{i,\nu}] = \sum_{j=1}^{\mathcal{I}}\sum_{k=1}^{j}\Pr\big[\sigma(\nu)=(j,k)\big]\nu^{A(j,k)}. \tag{8}$$

Combining (5)–(8) and applying algebraic manipulation, we get

$$E\big[\theta_j(\nu)\big] \le \frac{\sum_{j=1}^{\mathcal{I}}\sum_{k=1}^{j} a(j,k)\nu^{A(j,k)}}{1-\nu^{A(\mathcal{I},\mathcal{I})}}. \tag{9}$$

We now proceed onto step 3, which is completed via the following lemma, whose proof is a simple sample-path argument.

Lemma 1: $E[\theta_j(\nu)]$ is nondecreasing with increasing ν. Due to lack of space, the proof appears in the Appendix.

[3]This assumption greatly simplifies modeling, and we suspect does not have any significant impact on the metrics of interest.

Next, we proceed to step 4. This models a system in which if two nodes locate a copy of the object on the same round, then only one of them can retrieve the object. A simple sample path argument similar to the one used to prove Lemma 1 would show this assumption to be conservative. Last, step 5 follows trivially from the argument given in the step description. Hence we have successfully shown that

$$E\big[\theta_j(N_d, N, N_h)\big] \leq \sum_{i=N_h}^{N_d} E\left[\theta_j\left(1 - \frac{i}{N}\right)\right] \tag{10}$$

completing the computation for the upper bound on $E[\theta_s(N_d, N, N_h)]$.

1) Expected Rounds: We now compute an upper bound on the expected number of rounds when users' searches proceed in parallel. Let $\tau\,(a, b, c)$ be the time (in terms of rounds) that elapse until one node receives a copy of the object, where a of b nodes are performing the search in a network in which c nodes contain the object. Since each tick takes a fraction $1/a$ of a round (so that all a ticks complete in a single round), we have

$$\begin{aligned} E\big[\tau(a,b,c)\big] &= \sum_{i=0}^{\infty} \Pr\big[\tau(a,b,c) > i\big] \\ &= \frac{\sum_{i=0}^{\infty} E\big[G_{i,1-c/b}\big]}{a} \\ &= \frac{1}{a\Big(1 - E\big[G_{1,1-c/b}\big]\Big)}. \end{aligned} \tag{11}$$

We can then bound the expected number of rounds using our results from above that count the expected number of tick units that elapse between users' searches finding a copy of the object.

Lemma 2: $E[\rho_j(N_d, N, N_h)] \leq \sum_{i=N_h}^{N_d-1} E[\tau(N_d - i, N, i)]$.

Proof: The time taken by a tick, $\tau(a, b, c)$ increases as a decreases. Consider any sample path where A nodes initially seek the object, and let $S = \{t_0, \ldots, t_A\}$ be the ticks such that $t_0 = 0$ is the initial tick and $t_i, i > 0$ is the smallest tick for which i nodes have received the object since tick 0. Let $R = \{r_1, \ldots, r_n\}$ be the tick values for the starts of rounds up to tick t_{A-1}. To accurately compute the number of rounds that elapse, the amount of time that should elapse for tick i should be $1/(A - j)$ when the tick occurs in a round that commenced with $A - j$ nodes seeking the object, i.e., $t_j < r_k \leq i, i < r_{k+1}$, and $t_{j+1} \geq r_k$. However, the expression stated in the Lemma assumes that the time elapsed during tick i would be $1/(A - j')$ where $t_{j'} < i \leq t_{j'+1}$. Thus, we have $t_j < i \leq t_{j'+1}$, such that $t_j < t_{j'+1}$. Since the t_j are increasing, we have $j' + 1 > j$, or $j' \geq j$. Thus, our assumed tick time for time i, $1/(A - j') > 1/(A - j)$. ∎

Lemma 2 yields the upper bound on $E[\rho_j\,(N_d, N, N_h)]$.

VII. Evaluation

Using the analysis from the previous section and simulation, we evaluate the cost of the search protocol as a function of time (in rounds) for a user to find the object, and of the expected number of messages that a user can expect to receive and forward. Our simulations are performed on a home-grown, discrete-event simulator that allows us to experiment with variations in the connectivity of the underlying overlay. In particular, we use simulation to evaluate the performance metrics when bounding the permitted number of neighbor nodes by a constant, C. All simulation results presented include 95% confidence intervals, where each sample used to compute the confidence intervals is itself the average of 15 sample runs (such that the distribution was approximately normal). For each point plotted, a single overlay graph is generated (i.e., the overlay structure is fixed), but the nodes that initially have the object are chosen from a uniform distribution for each sample point used within the average.

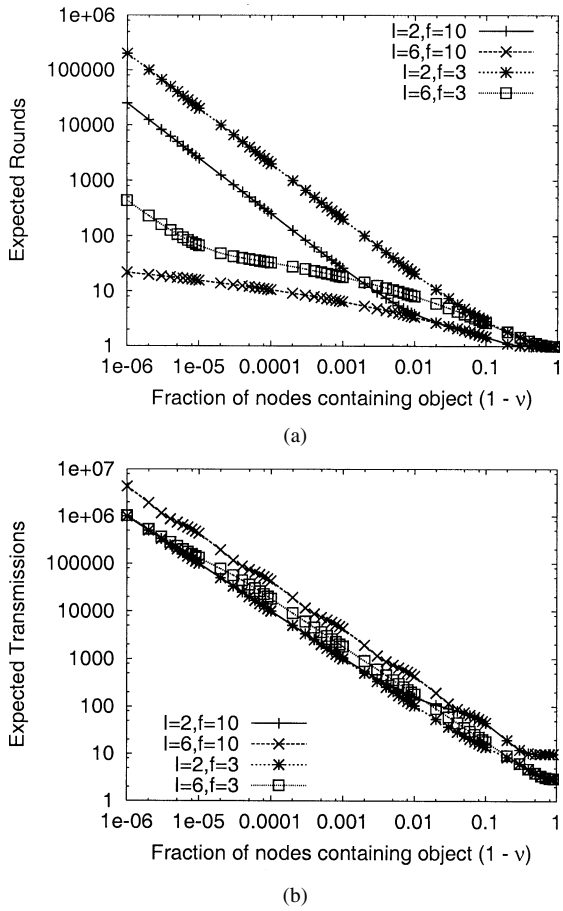

Fig. 1. Total rounds taken and transmissions generated by a single node's query for a fixed ν. (a) Rounds. (b) Transmissions.

Fig. 1 plots (using a log-log scale) statistics for a single user executing the protocol repeatedly until the object is retrieved as a function of the fraction, $1 - \nu$, of nodes that currently have the object. Fig. 1(a) plots the expected number of rounds that the search will take to locate the object (3), and Fig. 1(b) plots the expected number of transmissions that the user's search for the object generates [aggregated over all nodes in the network, (3)]. The different curves in the two figures represent different protocol configurations: curves labeled $I = x$, $f = y$ indicate that there are x iterations and the fanout is fixed at y.

We observe an obvious result that increasing the fanout or the number of iterations within the protocol decreases the expected number of rounds and increases the expected number of transmissions. We see, however, that reducing the fanout has little effect on the asymptotic performance of both the number of rounds and the number of transmissions as $1 - \nu \to 0$. The

number of iterations appears to increase the number of transmissions by at most an order of magnitude. We make some important observations here.

- Increasing the number of iterations can significantly reduce the number of rounds, but does not cause as significant an increase in the number of expected transmissions. By setting $\mathcal{I} = 6$, $f = 10$, all users can expect on average that their search will take no more than 25 rounds when more than 10^{-6} of the nodes in the overlay have a copy of the object. Most searches, however, are substantially shorter.
- The expected number of transmissions triggered by a query throughout the network does not grow much beyond $1/(1-\nu)$. In addition, we note that it is only possible for a fraction ν of nodes to not contain the object when there are at least $1/\nu$ nodes in the system. Since all nodes in the system are equally likely to receive a transmission, the number of transmissions that a node can expect to receive from a query does not appear to exceed 10 (i.e., it stays constant).
- The number of transmissions decreases at an exponential rate as $(1-\nu)$ is increased. This means that as queries are resolved and copies of the object propagate throughout the network, the expected number of transmissions to deliver the object to the subsequent querier is substantially lower.

We stress that the above observations are based on our use of a protocol that increases the number of rounds by one from the previous iteration. Our attempts to utilize other settings led to situations where the number of transmissions was significantly increased.

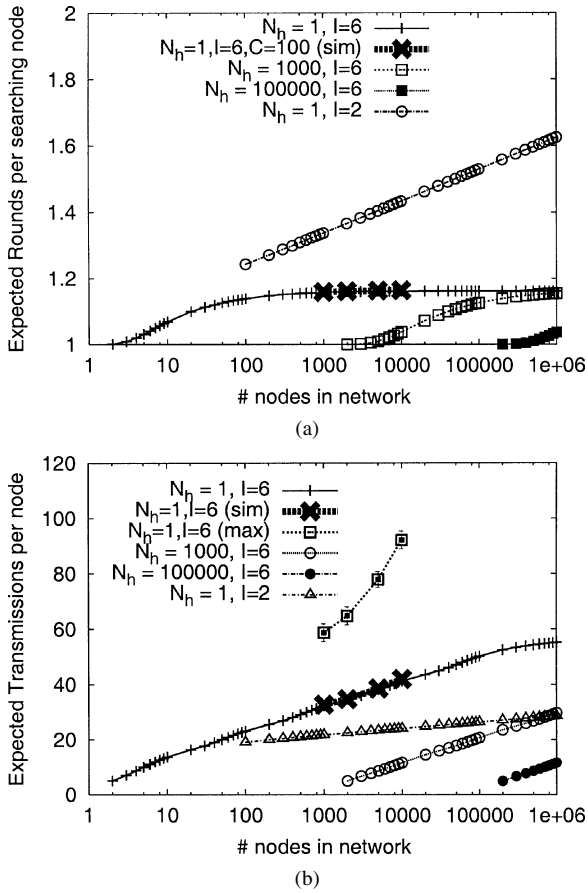

Fig. 2. Total rounds taken and transmissions generated by isolated joiners. (a) Rounds. (b) Transmissions.

A. Sequential Search Results

Next, we turn our attention to the analysis of the system in which multiple users wish to retrieve the hot object. We begin by noting that our equations of interest here, (3), (4), (11), and (10), are increasing functions of N_d. We limit our consideration to cases where $N_d = N$, since increasing the number of nodes that desire a copy of the object increases the expected number of transmissions, giving us a worst-case scenario for this expectation.

Fig. 2(a) plots the expected number of rounds a user can expect to wait until it retrieves the object. Fig. 2(b) plots the expected number of transmissions that a node *receives* over *all* queries transmitted in the network by all users' searches when they performed sequentially. Since each transmission sent by a node is also received by a node, this is also the expected number of transmissions sent by a node. In both figures, the number of nodes that participate in the overlay is given on the x-axis. The fanout, f is fixed at 10 in all curves. The number of iterations is set for the different curves at 2 and 6. In addition, we vary the number of nodes, N_h, that initially have the object. To summarize this figure plots the expected total number of transmissions that a node will receive and transmit during the entire search process.

In Fig. 2(a), we also plot the expected number of rounds computed by simulations for the case where $N_h = 1$, $\mathcal{I} = 6$, and $C = 100$ for $N = 1000$, 2000, 5000, and 10 000 [the curve's label contains "(sim)"]. In Fig. 2(b), the expected number of transmissions is plotted by simulation as well. Confidence intervals are plotted for these simulation results but are too tight to be visible in the graph. We note that in both of these figures, the simulation results almost exactly match the analytical results on a fully connected overlay where $N_h = 1$ and $\mathcal{I} = 6$. This demonstrates that our analytical results upon a fully connected overlay provide excellent approximations for these expected values upon a shuffled overlay in which nodes have a bounded number of neighbors (in this case, 100).

We also include a curve [containing the label "(max)"] generated from simulation results that plots the maximum number of transmissions received by any node while assisting in queries for *all* users' searches, averaged over all simulation runs. These preliminary results demonstrate that even the maximum number of transmissions received by any node is most often below 100 for up to 10 000 nodes. The parabolic shape of this curve suggests that for large overlays (of 1 000 000 nodes), it may be the case that a disproportionate load of requests may overwhelm a small set of nodes in the network. However, since it is likely that only a small number of nodes will be overloaded in this manner, we do not expect this overload to have a significant impact on overall expected protocol performance.

The total expected number of transmissions is obtained for a given curve by multiplying the value on the y axis by the value on the x axis. We find this quantity to be of lesser interest than

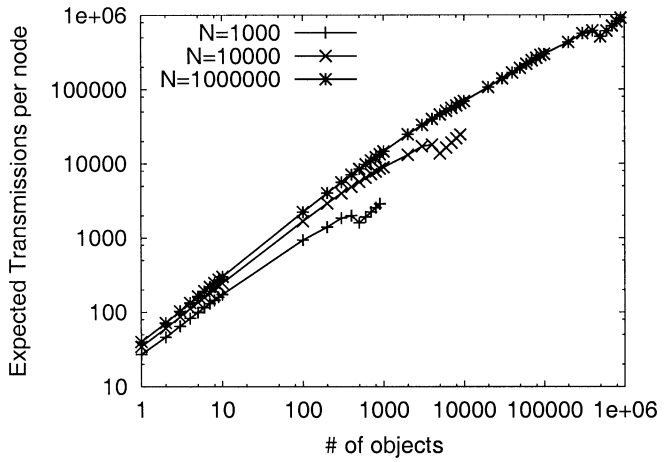

Fig. 3. Expected transmissions when searching for multiple objects.

the expected number of transmissions received per node since this number is distributed across a large set of nodes.[4]

These results show that for the case where users arrive in sequence, a user participating in the overlay can expect on average to receive on the order of at most 60 transmissions during the lifetime over all users' searches for the object, even in the case that there are 1 000 000 nodes searching for the object.

B. Searching for Multiple "Cold" Objects

We have demonstrated that there simple unstructured P2P search protocols exhibit a natural scaling phenomenon when a large body of users seek the same object. We now use our analytical model to demonstrate that this same phenomenon does not arise when there is a large body of searches, where there are multiple "cold" objects, each of which is being searched for by a small group of users.

In Fig. 3, we vary along the x axis the number of distinct objects that are searched for within the peer-to-peer overlay. We assume that one copy of each object resides initially within the peer-to-peer overlay, and that $\lceil N/x \rceil$ users seek each object (where x is the number of objects). This keeps the total number of searches fixed. We plot the expected number of transmissions that each client can expect to receive for overlays containing 1000, 10 000, and 1 000 000 clients.

We observe an almost linear growth in this number of transmissions as a function of the number of distinct objects searched for within the overlay. This demonstrates that our simple protocol has low transmission overhead when most peers search for the same object, as is the situation during a flash crowd. The protocol has high transmission overhead when used by many clients to search for many different objects, where each object is sought after by a relatively small group of clients. These results emphasize that this type of protocol does not scale well when users within the same overlay all seek different objects.

C. Simultaneous Search Results

Last, we examine the case where all users' queries are run simultaneously (searching for the same object). Fig. 4(a) plots

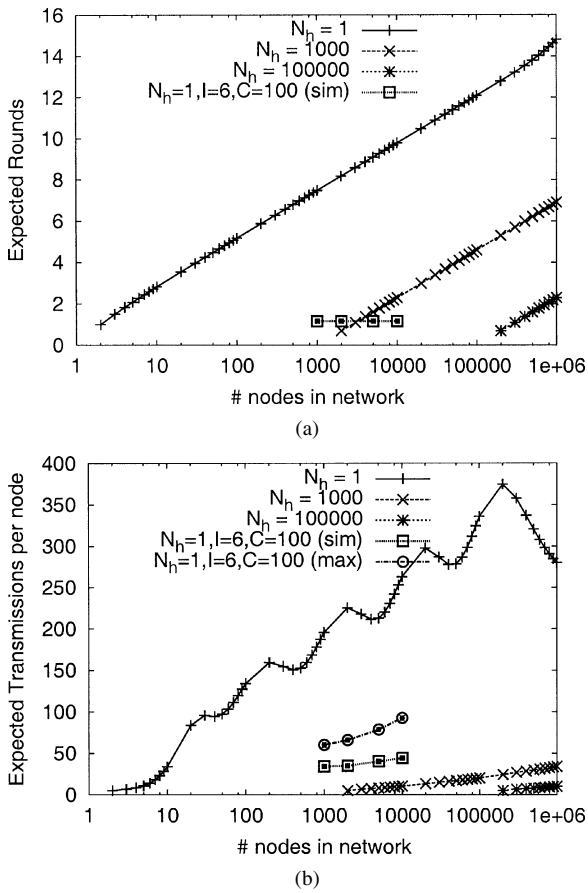

Fig. 4. Comparison of rounds and transmissions taken by isolated and simultaneous searches. (a) Rounds. (b) Transmissions.

upper bounds on the expected number of rounds of each user for this case (since all users start their searches simultaneously, the users are indistinguishable from the perspective of a mean-valued analysis) using the upper bound given in (11). Fig. 4(b) plots upper bounds on the expected number of transmissions received (or sent) by a node from all the queries of all the users that seek a copy of the object (11).

We note an interesting oscillatory behavior in the upper bound on the expected number of transmissions for the case where $N_h = 1$. We have no intuition for why the curve exhibits this behavior, but note that identifying a reason is not critical, given the curve is merely a loose upper bound. In fact, we have included simulation results in the plots [labeled "(sim)"] that demonstrate that the expected number of rounds and transmissions fall well below these bounds. These simulation results are generated from runs of 1000, 2000, 5000, and 10 000 node overlay topologies with $N_h = 1$, $\mathcal{I} = 6$, and $C = 100$.[5] For these simulations, we emulate a flash-crowd-like environment in which a user that has not initiated its first query by the ith round initiates the query on the $i+1$st round with probability $p = .001$. A user u will also initiate a query if another user u' "contacts" u during the round. Each round, each user u' that has already initiated a search contacts each user u with probability $q = .01$. This emulates an environment in which users who have initiated

[4]A possible exception is if one is concerned that numerous transmissions between overlay members traverse a common set of links, causing these links to bottleneck.

[5]99% Confidence intervals are in fact plotted here but are not visible, i.e., the confidence interval is too tight to be perceived along the current span of the y axis.

searches tell their friends bout the searches who then tell their friends, etc. The number of users that remain uninterested in the content diminishes stochastically at an exponential rate.

We see that the average number of rounds computed in simulation is significantly lower than our upper bounds predict. In Fig. 4(b), we also include a curve that, as in Fig. 2(b), plots the average number of transmissions received by the user that received the maximum number of transmissions within a given run. We see that, for the range of values we were able to simulate, this maximum value also falls way below our upper bound estimates.

Forming our conclusions based on our conservative upper bounds, we conclude that a node can expect on average to receive no more than 400 transmissions while participating in a protocol that delivers the object simultaneously to up to 1 000 000 users. In addition, the user should expect to receive the object in no more than 16 rounds.

We summarize our results in this section by considering their practical implications. First, we note that this loose upper bound of 400 transmissions applies to both the expected number of transmissions a receiver sends as well as the expected number of transmissions a receiver receives (there is a 1-1 correspondence between a packet sent and a packet received). Thus, is expected to handle on average no more than 800 transmissions, plus on average no more than one transfer of the hot object. With compression, it should be fairly easy to fit most queries into a 64-byte packet. Thus, 800 queries translates to 50 KB of data to be transmitted in total. A 28.8-baud modem can handle this amount of data within 15 s. Given that this quantity is based upon a loose upper bound, it seems clear that even today's low-end end-system technology can be a contributing component within this overlay protocol.

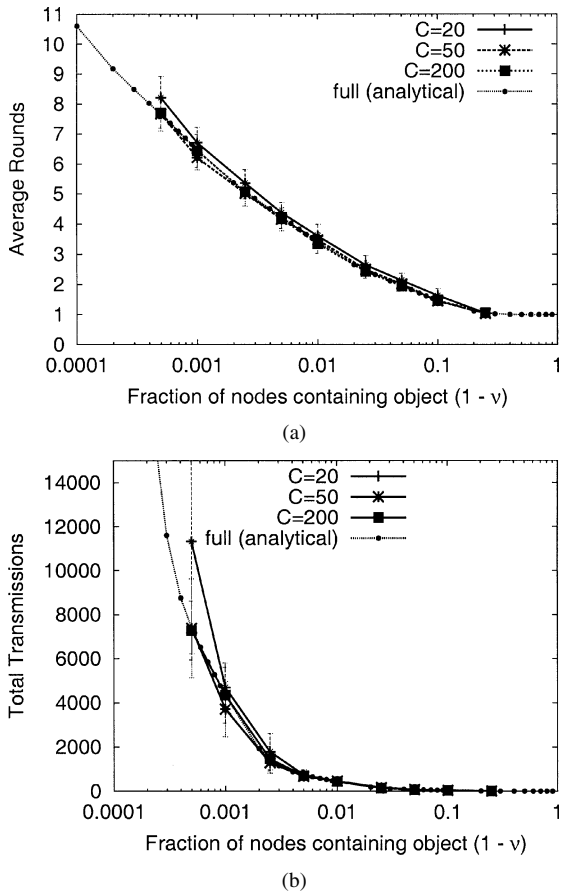

Fig. 5. Fully connected overlay versus values for C. (a) Rounds. (b) Transmissions.

VIII. How Valid are Analyses on a Fully Connected Graph?

In this section, we demonstrate that the assumptions made to make the model more analytically tractable do not significantly alter the performance results. We compare the measures from our analysis to measures performed in simulation using a protocol where a node selects f distinct neighbors (i.e., selected uniformly *without* replacement) from a set of C neighbors where $C \ll N$. The set of neighbors is chosen via the distributed shuffling algorithm described in [9], which is designed to maintain a random graph even as nodes join and leave the overlay.

Fig. 5 presents the expected number of rounds [Fig. 5(a)] experienced by each node and expected number of transmissions [Fig. 5(b)] experienced by the network as a whole when users queries are performed sequentially. In both figures, the x axis indicates the fraction of nodes, $1-\nu$ that do not contain the object. The curve labeled "full (analytical)" presents the expected number of rounds and transmissions derived from our analytical model on a fully connected overlay. The other curves plot these expected values of simulations for various values of C. The simulations are performed with $N = 2000$ (the expected values of are independent of N in the analytical model). The overlay is created by iterating over the set of nodes 200 times, and within each iteration, each node initiates a 10-shuffle.

We see that with 95% confidence, a fully connected overlay provides a good approximation for a shuffled overlay where each node is restricted to only $C = 20$ of 2000 neighbors. For $C = 50$ and $C = 200$, the differences in expected number of rounds and expected number of transmissions on the shuffled overlay and on the fully connected overlay are so slight they cannot be observed within the plots.

IX. Discussion

In the previous sections, we have demonstrated that in theory, protocols that use randomized, distributed, scoped searches on a P2P overlay is are an effective means for coping with server overloads due to flash crowds. Here, we briefly elaborate on some high level observations one can draw from our results, and also discuss limitations of our analysis and directions for future work.

Our analysis allows an exploration of the performance of a search as a function of several parameters. One can draw the following high level observations from our results and use these observations toward the design of P2P protocols that allow networks to survive flash crowds:

- The expected search time to find an object decreases exponentially with the increase in fanout, f, whereas the bandwidth increases by an amount that appears to be linear with f.
- Setting $\mathcal{I}$ to very small values decreases the bandwidth requirements slightly, but significantly increases search times. In addition, a small value for $\mathcal{I}$ can potentially hinder the spread of the object in networks with large diameters.

In summary, increasing f and $\mathcal{I}$ can reduce search times substantially, but seems to have a minimal effect on the bandwidth. This would suggest that, if bandwidth is not a highly constrained resource in the P2P network, that large values of f and $\mathcal{I}$ are preferred. We note that, since each round of iterations randomly selects f neighbors at each node from the set of total neighbors, it is important that one not choose a large f and a small $\mathcal{I}$. Otherwise, this leads to a continual repeat of (almost) the same set of neighbors selecting during each iteration.

Our analysis made several simplifying assumptions to produce a tractable analytical solution. One very important set of assumptions involves the topology and the manner in which users participate in the searching protocol. First, our analysis and simulations assumed that nodes did not join and leave the system during the course of all nodes' searches. There is a concern that users may join the P2P system, obtain their copy of the object, and leave the system immediately. However, we expect that, unless the process to leave immediately after download is automated, a normal leaving process will not significantly alter our results. In particular, in the flash crowd scenario we consider in this paper, numerous users' searches are taking place simultaneously or in rapid succession of one another. Since these searches are close to one another in time, and since our analytical results demonstrate that searches complete in only a few rounds after they were started, the rate at which nodes obtain copies of the object will greatly exceed the rate at which nodes with the object disconnect from the P2P network.

Second, our analysis is performed atop a fully connected graph and hence the graph has diameter one. Our simulation results showed that the bandwidth and temporal costs are similar when nodes are connected in a uniformly random manner to 20 or so neighbors. We suspect our results will continue to hold for graphs whose node degrees are described via a power law [19].

Last, our analysis uses a particular form of search protocol that is analytically tractable where neighbors to which queries are forwarded are selected at random. We suspect that in practice, a Gnutella-type protocol where each node contacts its entire set of neighbors would exhibit similar results. The one caveat is the case where nodes actually perform their searches in sequence. In this case, there is a possibility that a node that performs its search earlier on when copies of the object sparsely populate the network will be further from the object than the maximum distance probed by its search. In this case, that node would fail to receive a copy of the object, and because a repeat of the search would query the exact same set of nodes, the search would fail over and over again. Of course, the likelihood that, during a flash crowd, only a single node performs a search at a time and that there does not exist a copy of the object nearby during this search is an unlikely event.

X. Conclusion

We have provided a theoretical evaluation of the scalability of a distributed randomized P2P protocol that provides transmission of objects from servers currently suffering from hot spot conditions. The protocol itself is stateless and hierarchy-free, facilitating implementation and increasing its robustness. Our mathematical analysis and simulations of the protocol examine its performance in terms of the expected time to receive a copy of the desired object as well as the number of transmissions a node must send and receive as a participant in the overlay. We show that the times and bandwidth requirements scale even in scenarios where a minuscule set of users initially has copies of the object and a large majority of users seek to obtain a copy. Our results demonstrate that this simple protocol's performance is acceptable in overlays of up to 1 000 000 users, showing promise in a real network setting.

Appendix

Here, we prove the result that justifies (5).

Lemma 3: Consider a system in which the time, τ at which nodes initiate a query is uniformly distributed between 0 and $t-1 \pmod t$, where n is the total number of rounds within its protocol. Then $E[\theta_j(\nu)] = E[T_{1,\nu}]/(1 - E[G_{1,\nu}])$.

Proof: We begin by considering the expected transmissions generated by nodes at a given discrete point in time, t where a fraction $1-\nu$ nodes have the object and all users enter the system in their "steady state." We model the steady state by considering a renewal process in which each user who receives the object drops the object and restarts its protocol on the subsequent round. We define the random variable S_j^k, $0 \le j < kt$, $0 \le k \le N_d$ to be a shift variable that equals 1 when node k initiates its search at a time $\tau = j \pmod{kt}$, and 0 otherwise, where t is the number of rounds within the entire protocol (and therefore kt is the number of ticks within the entire protocol). In addition, we construct random variables $T_{i,\nu}^{j,k}$ and $G_{i,\nu}^{j,k}$ that are defined equivalently to $T_{i,\nu}$ and $G_{i,\nu}$ respectively for a node that started its transmission at a time $\tau = j \pmod t$. While these variables are not identically distributed (their distribution is a function of the number of transmissions that occur during the particular round, which is a deterministic quantity), they are mutually independent. Hence

$$
\begin{aligned}
E\big[\theta_j(\nu)\big] &= \sum_{k=0}^{N_d}\sum_{\ell=0}^{t-1}\sum_{i=1}^{\infty}\left(E\left[S_\ell^k T_{i,\nu}^{\ell,k}\left(\prod_{j=1}^{i-1}\prod_{m=1}^{N_d} G_{j,\nu}^{\ell,m}\right)\right.\right.\\
&\qquad\left.\left.\cdot\prod_{m=1}^{k-1} G_{i,\nu}^{\ell,m}\right]\right)\\
&= \sum_{k=0}^{N_d}\sum_{\ell=0}^{t-1}\sum_{i=1}^{\infty}\left(E\left[S_\ell^k\right]E\left[T_{i,\nu}^{\ell,k}\right]\right.\\
&\qquad\cdot\left(\prod_{j=1}^{i-1}\prod_{m=1}^{N_d} E\left[G_{j,\nu}^{\ell,m}\right]\right)\\
&\qquad\left.\cdot\prod_{m=1}^{k-1} E\left[G_{i,\nu}^{\ell,m}\right]\right).
\end{aligned}
$$

By substituting $E\left[S_\ell^k\right] = 1/(N_d t)$, that $\sum_{k=0}^{N_d} \sum_{\ell=0}^{t-1} E\left[T_{i,\nu}^{\ell,k}\right] = tE[T_{i,\nu}]$, that $E\left[G_{j,\nu}^{\ell,m}\right] = E[G_{j,\nu}]$, and by re-indexing over j and m that $\sum_{i=1}^{\infty}\left(\prod_{j=1}^{i-1}\prod_{m=1}^{N_d} E[G_{j,\nu}]\right) \prod_{m=1}^{k-1} E[G_{i,\nu}] = \sum_{i=1}^{\infty}\prod_{j=1}^{i-1} E[G_{j,\nu}]$, we obtain equality with (5). ■

Proof of Lemma 1: The proof follows from a simple sample-path analysis. We consider two systems, S and S_0 that contain the same total number of nodes. In S, a fraction ν of nodes do not contain a copy of the object, and in S_0, fraction $\nu_0 < \nu$ nodes do not have a copy of the object. We utilize an arbitrary 1–1 mapping ϕ that maps a node in S to a node in S_0 with the property that if a node $n \in S$ has a copy of the object, then $\phi(n)$ has a copy of the object as well. Since $\nu_0 < \nu$, there are still some nodes in S that do not contain the object that are mapped to nodes in S_0 that do contain an object.

We write $\langle n, m\rangle$ to indicate that node n transmits a query to node m. Consider any legitimate sequence of transmissions in $S(\langle n_{1,1}, n_{1,2}\rangle, \langle n_{2,1}, n_{2,2}\rangle, \ldots, \langle n_{\ell,1}, n_{\ell,2}\rangle)$ in S in which $\langle n_{\ell,1}, n_{\ell,2}\rangle$ is the first transmission that successfully locates a copy of the object. Then in the sequence $(\langle\phi(n_{1,1}), \phi(n_{1,2})\rangle, \langle\phi(n_{2,1}), \phi(n_{2,2})\rangle, \ldots, \langle\phi(n_{\ell,1}), \phi(n_{\ell,2})\rangle)$ in S_0, in which $\langle\phi(n_{\ell,1}), \phi(n_{\ell,2})\rangle$ also contains the object, and there is perhaps a prior transmission $\langle\phi(n_{k,1}), \phi(n_{k,2})\rangle$, $k < \ell$ where $\phi(n_{k,2})$ contains a copy of the object as well. Finally, note that because nodes are selected from a uniform distribution, the probability measure of the two sequences is equal. We construct random variables $X()$ and $X_0()$ that, respectively, map a sequence in S and S_0 to the index of the first transmission that locates the object. Since $X()$ stochastically dominates $X_0()$, we have that $E[X] \geq E[X_0]$. Finally, we note that because transmissions are often performed in parallel, the sequence of transmissions performed by the protocol may extend beyond the receipt of the object. It is trivial to show that in the case of the 1–1 mapping provided above, the sequence in S will never be shorter than the sequence in S_0. ■

ACKNOWLEDGMENT

The authors would like to thank E. Coffman, P. Jelenkovic, J. Nieh, and H. Schulzrinne at Columbia University for discussions that motivated their pursuit of research on hot spots. They also thank A. Shaikh and A. Acharya at IBM Research for discussions on peer-to-peer communication protocols, J. Lui for his encouragement in pursuing this work, V. Padmanabhan for handling the submission to ToN, and the anonymous reviewers.

REFERENCES

[1] E. Cohen and S. Shenker, "Replication strategies in unstructured peer-to-peer networks," presented at the ACM SIGCOMM, Pittsburgh, PA, Aug. 2002.

[2] The Gnutella Protocol Specification v0.4, Revision 1.2. [Online]. Available: http://gnutella.wego.com

[3] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," presented at the ACM SIGCOMM, San Diego, CA, Aug. 2001.

[4] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for Internet applications," presented at the ACM SIGCOMM, San Diego, CA, Aug. 2001.

[5] A. Rowstron and P. Druschel, "Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility," in *ACM Symp. Operating Systems Principles (SOSP'01)*, Banff, Canada, Oct. 2001.

[6] T. Stading, P. Maniatis, and M. Baker, "Peer-to-Peer caching schemes to address flash crowds," presented at the 1st Int. Workshop on Peer-to-Peer Systems (IPTPS'02), Cambridge, MA, Mar. 2002.

[7] M. Ripeanu and I. Foster, "Peer-to-peer architecture case study: Gnutella network," Univ. of Chicago, Chicago, IL, TR-2001-26, 2001.

[8] S. Saroiu, P. Gummadi, and S. Gribble, "A measurement study of peer-to-peer file sharing systems," Univ. of Washington, Seattle, WA, UW-CSE-01-06-02, 2001.

[9] A. Stavrou, D. Rubenstein, and S. Sahu, "A lightweight, robust P2P system to handle flash crowds," presented at the IEEE Int. Conf. Network Protocols (ICNP'02), Paris, France, Nov. 2002.

[10] V. Padmanabhan and K. Sripanidkulchai, "The case for cooperative networking," presented at the 1st Int. Workshop on Peer-to-Peer Systems (IPTPS'02), Cambridge, MA, Mar. 2002.

[11] V. Padmanabhan, H. Wang, P. Chou, and K. Sripanidkulchai, "Distributing streaming media content using cooperative networking," presented at the Int. Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'02), Miami Beach, FL, May 2002.

[12] J. Bernabeu-Auban, M. Ammad, and A. Ammar, "Resource finding in store and forward networks," *Acta Informatica*, vol. 28, 1991.

[13] D. Kempe, J. Kleinberg, and A. Demers, "Spatial gossip and resource location protocols," presented at the 33rd Annu. ACM Symp. Theory of Computing (STOC), Crete, Greece, Jul. 2001.

[14] 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," presented at the 6th Annu. ACM Symp. Principles of Distributed Computing, Vancouver, BC, Canada, Aug. 1987.

[15] R. Karp, S. Shenker, C. Schindelhauer, and B. Vocking, "Randomized rumor spreading," presented at the 41st Symp. Foundation on Computer Science (FOCS'00), Redondo Beach, CA, Nov. 2000.

[16] J. Ritter. (2001) Why Gnutella can't scale. No, really. [Online]. Available: http://www.monkey.org/~dugsong/mirror/gnutella.html

[17] D. Andersen, H. Balakrishnan, M. Kaashoek, and R. Morris, "Resilient overlay networks," presented at the 8th Workshop on Hot Topics in Operating Systems (HotOS-VIII), Elmau/Oberbayern, Germany, May 2001.

[18] ——, "Resilient overlay networks," presented at the ACM Symp. Operating Systems Principles (SOSP'01), Banff, Canada, Oct. 2001.

[19] M. Ripeanu, A. Iamnitchi, and I. Foster, "Mapping the Gnutella network: Properties of large-scale peer-to-peer systems and implications for system design," *IEEE Internet Comput.*, vol. 6, no. 1, pp. 50–57, Jan.-Feb. 2002.

Dan Rubenstein (M'00) received the B.S. degree in mathematics from the Massachusetts Institute of Technology, Cambridge, the M.A. degree in mathematics from the University of California at Los Angeles, and the Ph.D. degree in computer science from the University of Massachusetts, Amherst.

He has been an Assistant Professor of Electrical Engineering and Computer Science at Columbia University, New York, since 2000. His research interests are in network technologies, applications, and performance analysis, with a recent emphasis on resilient and secure networking, distributed communication algorithms, and overlay technologies.

Dr. Rubenstein has received a National Science Foundation CAREER Award, the Best Student Paper Award from the ACM SIGMETRICS 2000 Conference, and a Best Paper Award from the IEEE ICNP 2003 Conference. He has been a member of the ACM since 2000.

Sambit Sahu received the Ph.D. degree in computer science from the University of Massachusetts, Amherst.

He has been a Research Staff Member in the Networking Software and Services group at IBM T. J. Watson Research Center, Hawthorne, NY, since 2000. Since joining IBM, his research has focused on overlay based communication, content distribution architecture, and design and analysis of high-performance network communication protocols. He has published a number of papers in the areas of differentiated services, multimedia and peer-to-peer communications.